A Quality of Service Architecture

Andrew T. Campbell

Computing Department
Lancaster University

A thesis submitted for the degree of
Doctor of Philosophy

January 1996

Contents

Abstract

The notion of *quality of service (QoS)* has evolved rapidly over the past few years. Until recently, the term QoS referred to certain characteristics of network performance outside the control or influence of the end user. Recent years have seen great advances in field of QoS research, due mainly to the emergence of multimedia networking and computing. These technological developments are complemented by new user perspectives and the emergence of QoS demanding, multimedia applications.

This thesis is motivated by these recent technological changes and the need to provide *QoS assurances* to the end user. The first part of this thesis reports on the evolving notion of QoS in research and standards and identifies a number of limitations in the existing work. This thesis argues that for applications relying on the transfer of multimedia information, in particular *continuous media*, it is important that QoS is configurable, predictable and maintainable on an end-to-end basis.

The aim of this thesis is to contribute toward the development a generalised *quality of service architecture*. To address this aim an *integrated quality of service architecture (QoS-A)* is proposed, which offers a framework to specify and implement the required performance properties of continuous media applications over *asynchronous transfer mode (ATM)* networks. A major contribution of this thesis is the design, implementation and evaluation of the QoS-A *multimedia enhanced transport system (METS)*, and the required ATM network and operating system support. The second part of the thesis evaluates the METS transport system in a UNIX/ATM environment and attempts to place the work in the context of QoS architecture research reported in the literature.

The notion of QoS has moved on from where the end user had no influence on the delivered quality. Today, ATM networks not only have the capability of transmitting information at high speed, but they have the *potential* to offer end-to-end QoS configurable communications - and significantly this time under the management of the end user.

The research reported in this thesis has been influenced by the debates within the ATM Forum, IETF and ISO communities on QoS research. While the QoS-A evaluation is restricted to the operating system, transport and ATM networking areas it is hoped that this work can contribute toward the assessment of a generalised QoS architecture that ultimately will help harmonise the activities of these various communities.

Acknowledgement

The author would like to thank the students, research assistants and academic staff at Lancaster who contributed to the local ATM infrastructure on which the QoS-A transport system reported in this thesis was implemented. Special thanks are due to David Pegler, Nick Yeadon, Andrew Lunn, Andrew Scott and Doug Shepherd. The author would also like to acknowledge the many fruitful discussions which took place within the group. Special thanks are due to Frank Ball, Gordon Blair, Phillip Lougher and Andreas Mauthe. I would also like to express my appreciation to Philippe Robin and Francisco García for their contributions towards the formulation of a QoS-A model. Many thanks also to Aurel Lazar and the COMET Group for teaching me the importance of timescales in communication systems during my stay at Columbia University.

This thesis would not have been possible without the support of a number key people. First, I would like to express my thanks to my supervisor David Hutchison for his encouragement and sound judgement, and for giving me so many great opportunities to grow as a researcher under his thoughtful guidance. On a technical note, this thesis benefited enormously from contributions made by Geoff Coulson over the past four years. Geoff made key contributions to the development of the QoS model and operating systems support reported in this thesis. Next, a special thanks is due to my family - my sister Mary, my brother Edward and my mother - for their support.

Finally, my deepest thanks to my wife Susan Zak for her continued support and encouragement - and her amazing patience, thank you Susan!

The QoS-A project was funded as part of the UK SERC Specially Promoted Programme in Integrated Multiservice Communication Networks (GR/H77194) in co-operation with GDC (formally Netcomm Ltd).

To William and Miles McGinty Campbell

Chapter 1

Introduction

Recent years have seen great advances in computing and networking technology. At the network level, new high-speed technologies such as *Asynchronous Transfer Mode (ATM)* networks are actively being deployed in research institutions and industry. These networks not only have the capability of transmitting information at high speed, but also have the potential to offer a wide range of *Quality of Service (QoS)* properties including bounds on delay, guarantees on throughput and isochronous communications.

Multimedia workstation technology for generating, processing and displaying streams of digital video and audio is also available in the marketplace in a range of price/performance categories along with very high capacity storage systems and real-time video and audio codecs.

These technological developments are complemented by new user perspectives. New classes of distributed applications have been developed such as distance learning, desktop video-conferencing, virtual workshop and video on demand. These applications are characterised by their highly interactive nature and their significant use of multimedia information transfer. In these applications, communication requirements are extremely diverse [Leopold,92] and demand varying levels of service in terms of parameters such as latency, bandwidth, jitter and loss free delivery. Furthermore, it is often a requirement that levels of service are *guaranteed* for digital video and audio communications.

Other time-critical distributed applications such as distributed real-time control systems are also increasing in prominence. These may or may not involve multimedia information transfer, but have stringent requirements for both reliability and guaranteed bounds on message latency.

These various applications share the need for *quality of service control and management* to ensure that the requirements of the users are honoured.

1.1. Motivation

This thesis is motivated by the above mentioned recent technological changes and the need to provide quality of service assurance to the end user. This thesis argues that for applications relying on the transfer of multimedia information, in particular the *continuous media* of digital audio and video, it is essential that quality of service is *configurable*, *predictable* and *maintainable* on an *end-to-end* basis [NATO,88] [Anderson,90]; that is, system-wide, including the distributed system platform, operating system, transport system and the underlying network.

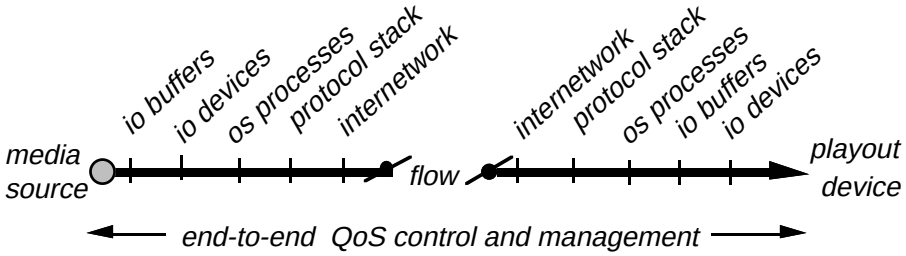

Figure 1.1: End-to-End QoS

Meeting quality of service guarantees in distributed multimedia systems is fundamentally an end-to-end issue: from application-to-application. For example, consider the remote playout of a sequence of audio and video: in the distributed system platform, quality of service assurances should apply to the complete flow of media: from the remote server, across the network to the points of delivery. As illustrated in figure 1.1, this generally requires the provision of a number of end-to-end QoS control and management mechanisms. These include 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 scheduling* and *flow control* in the network and, finally, active *monitoring* and *maintenance* of the delivered quality of service. Although previous research has addressed many isolated areas of QoS provision, little attention has so far been paid to the definition of an integrated and coherent framework that incorporates QoS interfaces, management and mechanisms across all architectural layers.

1.2 Media Quality of Service Demands

With the emergence of multimedia information exchange, stronger requirements are being placed upon communications support. Multimedia is characterised particularly by continuous media (e.g., voice, video, high quality audio, and graphical animation) which place greater demands on communications than still media such as text, images and graphics. Different types of continuous media require different levels of latency, bandwidth and delay jitter and they also require guarantees that levels of service can be maintained. For example, video communication requires high throughput guarantees but telephone audio requires only modest bandwidth. Error control should also be configurable. For example, uncompressed video is highly tolerant of communications errors whereas compressed voice cannot tolerate high errors rates and file transfers should be 100% error free. Delay jitter (i.e., variation in end-to-end delay) is an additional factor that must be taken into account for continuous media transfers and must be kept within particularly rigorous bounds to preserve the intelligibility of audio and voice information at playout devices.

1.2.1 Packet Video QoS

Many factors influence QoS demands made by video applications on underlying communications infrastructure. The problem of specifying and modelling quality of service for compressed video is rather challenging since bandwidth requirements and tolerance to loss are intimately related to both the coding scheme (e.g., MPEG, JPEG or H.261) and the source material (e.g., "talking head" or very high action). Some general observations concerning compressed video communications are:

i) burstiness is heavily dependent on the content of the image being coded and the coding algorithm used;

ii) source generation rates are highly sensitive to scene and background changes; and

iii) highly correlated source traffic is potentially persistent over very long periods. All this makes specifying and modelling QoS for compressed video applications rather difficult.

To illustrate continuous media quality of service demands on a transport system a real video stream trace is considered. Figure 1.2 shows a bandwidth trace for over 1700

compressed video frames from an MPEG video clip. The video sequence represents a mixture of material from low motion action to rapidly changing scenes. The playout video window dimension of 144 lines x 122 pixels (i.e., the spatial QoS) at the receiver translates to modest bandwidth demands on the communications system. The variable bit rate nature of MPEG video streams is apparent from the trace, which shows MPEG frame picture sizes as a time series. The peak frame size is in the order of 70 ATM cells with the minimum frame size less than 10 cells. This results in bursty traffic when injected into the network. The variable bit rate nature of the clip is a feature of MPEG coding algorithms that result in three different types of pictures (called I, P and B). On inspection, the I, P and B frames are visible in the trace. The I frames represent the major peaks at the top of the trace, the P pictures the dark peaks in the middle of the trace, and the B pictures the white peaks at the bottom of the trace. An important feature of MPEG (which will be discussed in Chapter 6) is that it can tolerate the occasional loss of certain pictures (e.g., B pictures) during transmission with little reduction of the perceived quality at a playout device.

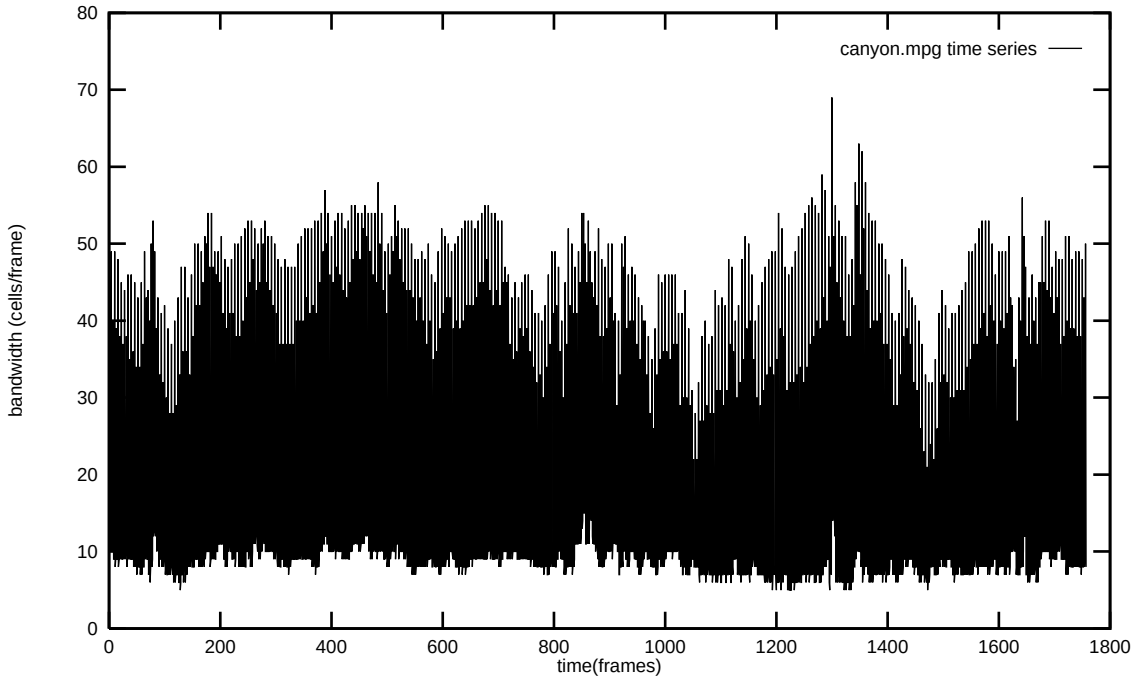

Figure 1.2: Bandwidth Trace (Time Series) of an MPEG Video Sequence

Delay requirements are dependent on the application itself. For example, if the video applications illustrated above represented an interactive session such as video conferencing the total end-to-end delay would be critical and should not in general exceed two to three

hundred milliseconds [Leopold,92]. In contrast, one-way video communications are not as sensitive to delay. Bounding jitter is, however, important in both cases and should be continuously monitored and maintained within fairly tight bounds.

In summary, QoS demands for multimedia communications are media, source and application dependent. In the case of compressed packet video communications, throughput is generally quite bursty, loss tolerance is picture dependent, delay is application dependent, and jitter is always important. These QoS characteristics are derived from analysing compressed video communications requirements [Leopold,92]. In general, however, each class of media (voice, video, animation, images) has different traffic characteristics and idiosyncrasies and each application of that medium may have quite different application level quality of service demands (e.g., colour or black and white playout).

1.2.2 End-to-End Demands

For distributed multimedia applications the concept of quality of service becomes applicable on a full end-to-end basis. In addition to the communications sub-system, this has implications for operating system scheduling for threads which are producing or consuming information for quality controlled communications. End-to-end QoS also involves *distributed application platforms* which are layered on top of the operating system to provide distribution transparencies and object based computational models.

The requirements of multimedia synchronisation such as ‘lip-sync’ impose quality of service constraints over multiple related transport level flows. The latter requirements is referred to as *orchestration* [Garcia,93]. QoS properties in this context are concerned with the ‘tightness’ of the orchestration required and the strategies required when QoS provision degrades.

Finally, the concept of QoS is also applicable to areas other than the traditional arena of point-to-point communications. For example, the prevalence of multicast and group communications in distributed multimedia systems leads to considerations such as the ordering semantics of group message delivery, meeting individual quality of service needs for each member of a multicast group, and resolving heterogeneity issues where individual senders and receivers may have differing capabilities to produce and consume audio-visual flows.

1.3 The Evolving Notion of Quality of Service

1.3.1 Retrospective

In the early 1990s, when this work commenced [Campbell, 92a], the notion of quality of service in communications architectures was a narrow one. Traditionally, the term 'quality of service' referred to certain characteristics of network services as observed by transport users. These characteristics were not generally controllable by users and described only those aspects of services attributable to the network provider. At that time, quality of service parameters in communications standards issued by the *International Standards Organisation (ISO) Open Systems Interconnection (OSI)* and *International Telecommunication Union-Telecommunication Standardisation (ITU-TS)* (formally Consultative Committee on International Telegraphy and Telephony (CCITT)) permitted the specification of some user requirements but these were rarely supported by the underlying network. For example, the OSI standards treated quality of service in a layer-specific fashion. This was a product of the QoS definition being addressed by separate ISO committees (i.e., the presentation, session, transport, network and data link committees), each working in isolation from one another. Thus, the relationship between the QoS layers was not clearly defined and there was no consistent, integrated notion of quality of service which related user requirements to the network provider services.

1.3.1.1 Open System Interconnection

The OSI service definitions do not provide for the specification of a comprehensive QoS parameterisation. In addition, there is no support for QoS negotiation and monitoring and the precise semantics of responsibilities and guarantees are not clear. Even more limiting is the fact that at the protocol level there is no notion of QoS management in terms of QoS re-negotiation, mapping, resource reservation and QoS maintenance. It is simply assumed that the underlying network provider will support the requested QoS levels. Another important observation is that the OSI upper layers are not QoS-aware. QoS parameters are simply mapped unchanged through to the transport layer. If users want to specify QoS they are forced to drop below the level of abstraction provided by the upper

layer architecture and interact with layers that are intended to be hidden from applications.

1.3.1.2 Internet 'Base Service'

During the same time the Internet only permitted the specification of *qualitative* QoS hints to the IP base service (using the type of service field in the IP header) such as ‘low’ delay, ‘high’ throughput and ‘high’ reliability. Even these limited QoS specifications are rarely honoured by the underlying network. Furthermore, the Internet architecture was based on a best effort performance model which was never designed to support *quantitative* QoS needed for a full range of multimedia communications. In the Internet, the support of reliable data transfer was a primary design goal and until recently (cf. Integrated Services Internet [Braden,94]) performance QoS was a marginal consideration.

1.3.1.3 ATM Networking

The ITU-TS in their 'I-series' recommendations recognised the need for QoS configurability in the emerging ATM standards for the *Broadband Integrated Services Digital Network (B-ISDN)* and also defined a fairly comprehensive set of parameters and reference model. These recommendations did not consider how the traffic characterisation at the ATM layer was to be derived from user QoS needs at the transport layer and above. Below the service interface, the state of ATM standardisation suffered from a lack of QoS management support comparable to that found in the OSI field. At that time there was no consensus on how resources would be allocated and how requested QoS levels could be maintained, policed and re-negotiated.

1.3.1.4 Evaluation

A limitation of most of these architectures was the *static* nature of service provision. In OSI protocols, the value of a QoS parameter remained constant for the duration of a connection, i.e., once negotiated a QoS parameter was never re-negotiated. One implication of this limitation was that users cannot dynamically adjust the communication QoS without undergoing a disconnection/re-establishment phase or opt for QoS trade-offs in the face of limited resources. For example, users could not choose to scale back the quality of an existing video connection in terms of spatial or temporal performance to allow the possibility of opening a new audio connection. Another implication was that the service-

provider was committed to provide the QoS over the lifetime of the connection. If the provider was unable to maintain its commitment there was no mechanism to inform the user and allow him/her to adapt intelligently to the new level of service. The only option available to the provider in the case of OSI and ATM protocols was the unilateral closure of connections. Unilateral disconnection in the face of QoS fluctuations of this type is generally too severe an action to take and, more important unsuitable as a paradigm for continuous media communications. Deriving suitable methods and mechanisms for adapting to fluctuations in delivered quality of service is more appropriate than closing the connection for continuous media applications.

1.3.2 New QoS Initiatives

Recent years have seen the emergence of a number of important initiatives from the standards and research communities. These initiatives broadly address limitations in the current QoS provision in light of new multimedia communication requirements.

1.3.2.1 Integrated Services Internet

The work by the *Integrated Services (int-serv) Group* [Braden,94] of the *Internet Engineering Task Force (IETF)* is a significant contribution to providing QoS guarantees for multimedia applications over an integrated services Internet. The int-serv group have defined a comprehensive integrated service model [Shenker,93] and QoS framework [Shenker,95a] used to specify the functionality of internetwork system components (known as '*QoS-aware elements*') which make multiple, dynamically selectable qualities of service available to applications in an internetwork.

The behaviour of elements, which constitute routers, subnetworks and end-system 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. The concatenation of these service elements along an end-to-end data path used by an application provides an overall statement of end-to-end QoS. Each service definition also specifies QoS parameters used to invoke the service. The current set of defined services (viz. controlled delay, guaranteed delay and predicated delay) are appropriate for a wide range of multimedia flows. These new services will be offered in addition to the current best effort service.

The int-serv QoS framework document [Shenker,95a] defines a flow as:

> *"a set of packets traversing a network element all of which are covered by the same request for control of quality of service. At a given network element a flow may consist of packets from a single application session, or it may be an aggregation comprising the combined data traffic from a number of application sessions".*

While the definition emphasises the 'network' element the framework in its full generality it is equally applicable to both the end-system and network. The goal of the int-serv group is to focus on the network in the first instance.

1.3.2.2 ATM Forum Networking

ATM standards have progressed over the past several years. The *ATM Forum* is comprises a group of users, vendors and telecommunications providers that have been responsible for the fast track development of a set of interim ATM standards. These standards have helped to clarify some of the shortcomings evident in earlier ITU-TS draft standards regarding quality of service specification, control and management. Prominent among the approved standards are the service model definition, the user-network interface (UNI) [ATMF,95a] and the pending network-to-network [ATMF,95b] interface draft standard. With the approval of UNI 4.0 by the Forum, ratified ATM standards now support several classes of service suited to carry multimedia information with QoS configurable connections.

The *Traffic Management* and *QoS Adhoc Working Groups* [ATMF,95c] have been key in the development, rationalisation and progression of earlier Forum and ITU-TS contributions. New QoS initiatives proposed by the Forum and int-serv groups are likely to have a profound effect on multimedia networking over the next decade. At this time both groups are equally poised to shape the future of QoS driven communications. With this in mind it is interesting to note that while the services and QoS parameterisation proposed by both groups have some similarities, interworking between the two worlds (i.e., mapping one set of services and traffic characterisation to the other) will prove very challenging.

1.3.2.3 ISO QoS Initiatives

Two new ISO QoS initiatives have emerged recently in the ISO. These are the *ISO QoS*

Framework by the ISO SC21 QoS Working Group and the *Enhanced Communication Functions and Facilities (ECFF)* by the ISO SC6 ECFF Working Group.

ISO SC21 QoS Working Group

A significant contribution to the field of quality of service support for OSI communications is the OSI QoS Framework [IS0,95a] developed by the SC21 QoS Working Group. The QoS framework broadly defines terminology, concepts and mechanisms for QoS and provides a model for the identification of 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 QoS framework concepts include:

i) *QoS requirements*, which are realised through QoS management and maintenance entities;

ii) *QoS characteristics*, which are fundamental aspects of QoS that have to be managed;

iii) *QoS categories*, which represent a policy governing a group of QoS requirements specific to a particular environment such as time-critical communications; and

iv) *QoS management functions*, which can be combined in various ways and applied to various QoS characteristics in order to meet QoS requirements.

ISO SC6 ECFF Working Group

The subject of QoS emerged as an important activity in another ISO project addressing enhanced communication functions and facilities for the lower layers of the OSI reference model. The SC6 ECFF Working Group has been instrumental in introducing key multimedia communication requirements [Hutchison,92] into the ECFF guidelines document [ISO,92] which, as anticipated, will direct the development of current and future ISO transport and network standards.

The Esprit OSI 95 project played a leading role in initiating and contributing toward the new project. The OSI 95 project's major contribution to the new work item was the specification of a new multimedia-capable transport system called TPX [Danthine,92]. TPX

provides support for connection-oriented services with sequenced delivery, QoS configurability and error notification. The enhanced connection-oriented service takes QoS parameters relating to throughput, delay, delay jitter, error selection policy and relative priority. Three novel transport quality of service semantics (viz. compulsory, threshold and maximal QoS) have been proposed in the specification while maintaining the best effort service.

ECFF service semantics are now being considered by the OSI QoS group for inclusion in the QoS Framework Committee Draft (CD) standard resulting in continuity between the "architects" view (i.e., SC21 committee) and "protocol designers" thinking (i.e., SC6 committee). It is anticipated that by building strong liaisons between the various groups working on aspects of *layer specific QoS* the ISO QoS group can have a major impact on the treatment of QoS across all layers of the OSI reference model. It is interesting to observe that there is little consistency between the services being proposed by the ISO ECFF and QoS groups and those being offered by the ATM Forum and int-serv groups. Such discontinuity may result in barriers to future interworking.

1.3.3 Limitations

To date, most of the developments in the provision of quality of service support have occurred in the context of individual architectural layers. Much less progress has been made in addressing the issue of end-to-end QoS for multimedia communications. There has been, however, considerable progress in the separate areas of Open Distributed Processing (ODP), end system and network support for quality of service. In end-systems, most of the progress has been made in the specific areas of scheduling, flow synchronisation and transport support. In networks, research has focused on providing suitable traffic models and service disciplines, as well as appropriate admission control and resource reservation protocols. Many current network architectures, however, address quality of service from a providers point of view and analyse network performance, comprehensively failing to address the quality needs of applications. Until recently there was little work on quality of service support in distributed systems platforms. Such work has been done in the context of the Open Distributed Processing ISO standard in SC21.

The essential characteristics of the current state of QoS research can be summarised as follows:

i) *incompleteness in service:* current service interfaces are generally not QoS configurable and provide only a subset of the facilities needed for control and management of continuous media;

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;

iii) *lack of continuity*: current QoS initiatives are primarily network-centric and do not address the suitability of network QoS semantics extended in to the end-system, i.e., can ATM Forum, int-serv and ISO QoS models be extended successfully to the distributed platform and operating system environments or is there an intrinsic discontinuity? and

iv) *lack of overall framework:* it is necessary to develop an overall architectural framework to build on and reconcile the existing notion of quality of service at different systems levels and among different network architectures.

1.4 Quality of Service Architectures

In recognition of the issues discussed above, the development of an integrated QoS model which spans both end-systems and networks and takes the support of QoS for a wide range of multimedia applications as its primary goal is proposed here. These models are called *QoS architectures* [Campbell,92a]. These architectures promote a *systems* approach to quality of service management in distributed systems.

The primary intention of QoS architecture research is:

> *"to define a set of quality of service configurable interfaces, services and mechanisms that formalise quality of service in the end-system and network, providing a framework for the integration of control, maintenance and management mechanisms to meet application level end-to-end QoS requirements".*

Because of the increased range and complexity of QoS provision required by the emerging distributed multimedia applications, it becomes evident that the necessary extensions to QoS provision cannot be carried out in a piecemeal fashion. Instead, the notion of a comprehensive QoS architecture is advocated whereby application requirements

can be mapped through all the levels of the system. Thus,

> *"communications abstractions at the application platform level should provide QoS abstractions which can be mapped down through all intermediate layers to the network access point in a coherent and integrated way - this strategy is described as integrated QoS in this thesis".*

This mapping - called *QoS mapping* - should be simple and efficient, protecting application programmers from communication details.

1.5 Thesis Aims

The aim of this thesis is to contribute toward the development of a generalised quality of service architecture. To address this aim an integrated *Quality of Service Architecture (QoS-A)* is designed which offers a framework to specify and implement the required performance properties of continuous media applications operating over ATM networks.

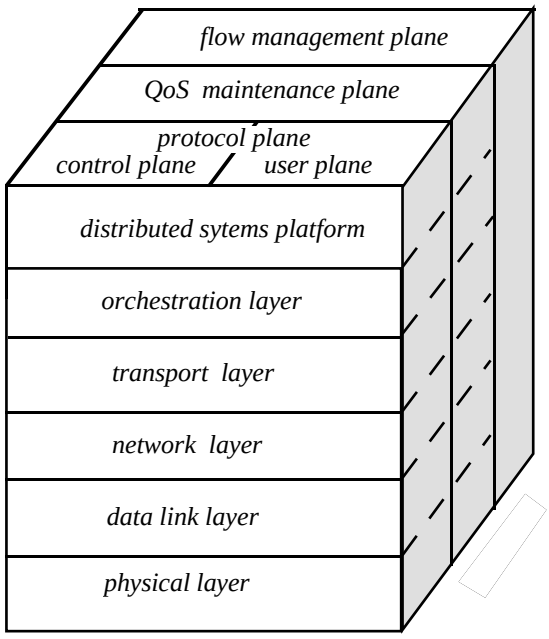

Figure 1.3: Quality of Service Architecture (QoS-A)

The QoS-A illustrated in figure 1.3 retains the best effort service model as a special case but augments it with new classes of service providing hard and soft end-to-end performance guarantees. These new service classes will be applicable in the end-system and

network and will be designed to fit into a highly dynamic application environment providing support for control, monitoring and maintenance of end-to-end QoS.

In addition to the need for a richer service model which allows the QoS requirements of the new applications to be fully specified, the QoS-A requires the integration of a range of QoS mechanisms in both the end-system and the network to meet end-to-end needs. In end-systems, this will include thread scheduling, traffic shaping, buffer management and jitter correction. In the communications subsystem, protocol support such as end-to-end QoS negotiation, re-negotiation and indication of QoS degradation are required. In networks, suitable resource reservation protocols, service disciplines in switch queues and multicast QoS support are needed. The QoS-A provides a framework for the maintenance and management of QoS over all system layers. This includes *flow management* functions such as QoS mapping between different layers of the architecture, admission testing for new flows and monitoring to ensure that QoS levels can be maintained.

A major contribution of this thesis is the realisation of the QoS-A at the transport layer and its assessment in the context of a generalised quality of service architecture. A new transport service and protocol collectively called the *Multimedia Enhanced Transport System (METS)* is proposed, designed and implemented. The METS transport system builds on previous work on continuous media transports [García,93] and includes QoS control, maintenance and management mechanisms to support *multicast flows*. It will be shown how QoS levels contracted at the transport level application programmers' interface (API) can be assured in the context of the Lancaster ATM Research Networking Environment.

1.6 Thesis Structure

The structure of the thesis is as follows. Chapter 2 sets out a framework for discussing QoS support in distributed multimedia systems. QoS terminology, principles and concepts are introduced which underpin quality of service architecture philosophy. Next, a set of modules used in building quality of service into distributed multimedia systems are described. These modules include QoS specification which captures application level quality of service requirements and a set of QoS mechanisms that realise the desired behaviour.

Following this, in Chapter 3, an initial review of QoS research by surveying layer-

specific work is developed; the distributed systems platform layer, operating systems, and transport and network research, respectively, are considered. A number of fledgling QoS architectures that have recently been reported in the literature are then evaluated.

In Chapter 4 the QoS-A model is described, primarily focusing on the role of transport layer in the architecture. The QoS-A incorporates the notions of *flows*, *service contracts* and *flow management*. Flows characterise the production, transmission and eventual consumption of single media streams, service contracts are binding agreements between users and providers and flow management provides for the monitoring and maintenance of the contracted QoS levels. A multimedia enhanced transport service, flow management and transport layer service contract is outlined. It is shown how QoS levels contracted at the transport API can be assured in the context of the Lancaster ATM Research Networking Environment.

Next, in Chapter 5 the QoS-A operating system support is described. The chapter begins by providing the necessary background material on the Chorus micro-kernel and the extensions made to the kernel to support QoS driven communications. The major components of the QoS-aware operating system are then detailed. These include a *Chorus QoS-based API*, *split-level scheduling framework*, QoS controlled communications and *QoS-driven memory management* and flow management system.

Chapter 6 addresses the challenge of meeting QoS demands in multicast communications. The salient features of scalable video flows are presented in this chapter, then a set of *scaling objects* used in the QoS-A architecture together with an extended application programming interface are described. Following this, the *Dynamic QoS Management (DQM)* of *adaptive multicast flows* is reported. The detailed operation of a number of specific types of scaling objects is described, followed by the introduction of an *adaptive network service*, a definition of the notion of *weighted fair share (WFS)* resource allocation, an explanation the rate control scheme and, finally, an outline of the use of several network oriented *QoS filters.*

The implementation details presented in Chapter 7 correspond to the end-system and network domains. In the end-system, METS communications support is embedded in the Linux operating system [Linux,94] and interfaces directly to the Lancaster ATM Research Networking Environment. In the network, METS signalling and QoS control mechanisms are embedded in the ATMos operating system [French,93] resident in the Olivetti Research Laboratory (ORL) 4x4 ATM switches.

To evaluate this implementation an *MPEG Networked Video System (NVS)* [Yeadon,95] has been developed. In Chapter 8, the performance impact of the METS transport system using NVS running on Pentium based PCs and RAID-3 based storage servers interconnected by ORL ATM switches is evaluated. Following this a discussion of QoS-A in relation to the other QoS architectures reported in the literature is presented. Chapter 9 presents a number of general conclusions and areas where further work is required.

Chapter 2

Quality of Service Terminology, Principles and Concepts

This chapter presents the terminology, principles and concepts used in later discussions of quality of service in QoS-based architectures. Where appropriate the terminology used is that adopted by the ISO QoS group in the QoS Framework committee draft standard. A set of modules utilised in building quality of service support into distributed multimedia systems is described. These modules include: *QoS principles* which govern the construction of integrated QoS architectures, *QoS specification* which captures application level quality of service requirements and *QoS mechanisms* which realise the desired end-to-end QoS behaviour.

2.1. Terminology

As a first step in the analysis of QoS provision in distributed systems, the term *activity* is introduced to refer to those aspects of a system to which it is useful to ascribe quality of service characteristics. Examples of activities are processes, communications, or complete computer systems. One particularly important type of activity in the context of multimedia and real-time distributed applications is the notion of a *flow* [Partridge,92] which is defined [Campbell,92] as:

> *"the production, transmission and eventual consumption of a single media source (viz. audio, video, data) as an integrated activity governed by a single statement of QoS; flows are always simplex but can be either unicast or multicast; flows generally require end-to-end admission control and resource reservation, and support heterogeneous QoS demands".*

In characterising the QoS of activities, it is necessary to identify *dimensions* along which QoS can be measured and quantified. To take a familiar example, it is common to measure the QoS of a timesharing computer system along the dimensions of system

throughput and user response time. It is also useful to group sets of QoS dimensions into QoS *categories* where each category contains dimensions pertaining to some logically identifiable aspect of QoS. As an example, a "system reliability" category which contains system-related reliability dimensions may be defined as the mean time between failure (MTBF) or mean time to repair (MTTR). Another reliability category relating more particularly to the field of multimedia and real-time distributed systems may contain dimensions relating to, for example, the permitted percentage of loss of media frames in a flow or the permitted bit error rate in ATM cells.

Other important QoS categories of relevance to the distributed multimedia application area are *timeliness* and *volume*. The timeliness category contains dimensions relating to the end-to-end delay of control and media packets in a flow. Examples of such dimensions are *delay*, measured in milliseconds and defined as the time taken from the generation of a media frame to its eventual display, and *jitter*, also measured in milliseconds and defined as the variation in overall nominal latency suffered by individual packets of the same flow. The volume category contains dimensions that refer to the throughput of data in a flow. At the level of end-to-end flows, an appropriate QoS dimension may be video *frames* delivered *per second (fps)*. Alternatively, at the ATM layer, a typical volume QoS dimension would quantify throughput in terms of *peak-rate* throughput and *statistical* throughput measured in cells per second. These examples illustrate that certain dimensions are often only applicable at certain system layers and imply that a complete category should contain dimensions for each system layer involved in the support of that category.

The above list of categories and dimensions is far from exhaustive. Other categories worthy of mention are *criticality* which relates to the assignment of relative priority levels between activities, *quality of perception* which is concerned with dimensions such as screen resolution or sound quality, and *logical time* which is concerned with the degree to which all nodes in a distributed system see the same events in an identical order. *Cost* is another important category. Cost considerations are also typically applied to the level of QoS provided in the various other QoS categories. A more complete selection of QoS categories can be found in [ISO,95a].

2.2 QoS Principles

Five principles govern the construction of QoS architectures. These are the principles of

integration, separation, transparency, asynchronous resource management and *performance.*

2.2.1 Integration Principle

The integration principle states that quality of service must be configurable, predictable and maintainable over all architectural layers to meet end-to-end quality of service [Campbell,93]. Flows traverse resource *QoS modules* (e.g., CPU, memory, devices, network) 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 QoS module traversed must provide QoS configurability (based on a QoS specification), resource guarantees (provided by QoS control mechanisms) and maintenance (based on monitoring mechanisms) of on-going flows.

2.2.2 Separation Principle

The separation principle states that media transfer, control and management are functionally distinct architectural activities [Lazar,92]. The separation principle states that these activities should be separated in architectural frameworks. One aspect of separation is the distinction between signalling 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 at the playout devices. On the other hand, signalling, which is full duplex and more asynchronous in nature, generally requires low bandwidth, assured-type services with no jitter constraint.

2.2.3 Transparency Principle

The transparency principle states that applications should be shielded from the complexity of underlying QoS specification and QoS management functions such as QoS monitoring and maintenance. An important aspect of transparency is the QoS-based API [Campbell,94] [Bansal,95] at which desired quality of service levels are stated (see QoS management policy in Chapter 4). The benefit of transparency is three-fold. It reduces the need to embed quality of service functionality in applications. It hides the detail of underlying service specification from the application and it delegates the complexity of handling QoS management activities to the underlying QoS framework.

2.2.4 Asynchronous Resource Management Principle

The asynchronous resource management principle [Lazar,92] 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 reflection of fundamental time constraints that operate in parallel between activities (e.g., scheduling, flow control, routing, QoS management) in distributed communications environments. The "state" of the distributed communication system is structured according to these different time scales. The 'operating point' of communication activities is arrived at via asynchronous algorithms that operate and exchange control data periodically among each other.

2.2.5 Performance Principle

Finally, the performance principle subsumes a number of widely agreed rules for QoS-driven communications design and implementation. These rules guide the division of functionality in structuring communication protocols for high performance in accordance with Saltzer's systems design principles [Saltzer,84], avoidance of multiplexing [Tennenhouse,90], recommendations for structuring communications protocols such as application layer framing and integrated layer processing [Clark,90], and the use of hardware assists for protocol processing [Chesson,88] [Zitterbart,92].

2.3 QoS Specification

QoS specification is concerned with capturing application level quality of service requirements and management policy and is generally different at each system layer and is used to configure and maintain QoS mechanisms resident at each layer. For example, at the distributed system platform level QoS specification is primarily user-oriented rather than system-oriented. Lower-level considerations such as tightness of synchronisation for multiple related transport flows, or the rate and burst size of flows, or the details of thread scheduling should all be hidden at this level.

QoS specification is therefore declarative in nature; users specify what is required rather than how this is to be achieved by underlying QoS mechanisms.

Quality of service specification encompasses the following areas:

2.3.1 Flow Synchronisation ("Orchestration") Specification

Flow synchronisation specification characterises the degree (i.e., tightness) of synchronisation between multiple related flows [Little,90]. For example, simultaneously recorded video perspectives (shown as video 1 and 2 in figure 2.1) must be played in precise frame by frame synchrony so that relevant features may be simultaneously observed. On the other hand, lip synchronisation in multimedia flows does not need to be absolutely precise when the main information channel is auditory and video is only used to enhance the sense of presence.

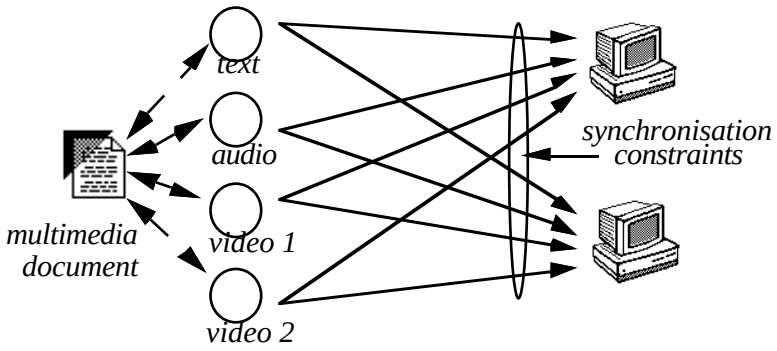

Figure 2.1: Flow Synchronisation / Orchestration

2.3.2 Flow Performance Specification

Flow performance specification characterises the user's flow performance requirements [Partridge,92]. 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 a QoS architecture one must have prior knowledge of the expected traffic characteristics associated with each flow before resources can be committed and resource guarantees made.

2.3.3 QoS Commitment

QoS commitment specifies the degree of end-to-end resource commitment required (e.g., deterministic [Ferrari,90], predictive [Clark,92], adaptive [Campbell,95] and best

effort). While the flow performance specification permits the user to express required performance metrics in a quantitative manner, QoS commitment allows these requirements to be refined in a qualitative way so as to allow a distinction to be made between hard, firm and soft performance guarantees. QoS commitment expresses a degree of certainty that the QoS levels requested at flow establishment or re-negotiation will actually be honoured.

2.3.4 QoS Management Policy

QoS management policy captures the degree of QoS adaptation [Campbell,95] (continuous or discrete) that a flow can tolerate and scaling actions to be taken in the event of violations in the contracted QoS [Campbell,95]. 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 user-level selection of QoS indications (called event *QoS signals*) in the case of violations in the requested quality of service, and periodic bandwidth, delay, jitter and loss notification (i.e., QoS signals) which are suitable for adaptive applications [Shenker,93a].

2.3.5 Cost of Service

Cost of service specifies the price the user is willing to incur for the level of service; cost of service is an important factor when considering QoS specification. If there is no notion of cost involved in QoS specification, there is no reason for the user to select anything other than maximum level of service (e.g., guaranteed QoS at peak rate). This philosophy would inevitably lead to resource inefficiencies. To counter this condition the cost function must incorporate pricing differentials [Chocchi,91] to encourage the user to select the optimum QoS commitment and performance QoS parameters such as a lower-commitment-costs-less pricing policy.

2.4 QoS Mechanisms

Quality of service mechanisms are selected according to user supplied QoS specification (described above), resource availability and resource management policy. In resource management, QoS mechanisms are categorised as either static or dynamic in nature. Static

resource management deals with flow establishment and end-to-end QoS re-negotiation phases (which is described as QoS provision) and dynamic resource management deals with the media-transfer phase (which is described as QoS control and management). The distinction between the former and latter is due to the different time scales on which they operate and is a direct consequence of the asynchronous resource management principle. QoS control distinguishes itself from management in that it operates on a faster more reactive timescale (e.g., cell scheduling operates on faster timescales than evolving the playout time for a particular flow at the playout device).

2.4.1 QoS Provision

QoS provision is comprised of three components: *QoS mapping, admission testing* and *resource reservation.*

2.4.1.1 QoS Mapping

QoS mapping performs the function of automatic translation between representations of QoS at different system levels (i.e., operating system, transport layer, network) and thus relieves the user from the necessity of thinking in terms of lower level specification. For example, the transport level QoS specification may express flow requirements in terms of QoS commitment, average and peak bandwidth, jitter, loss and delay constraints - all related to transport packets.

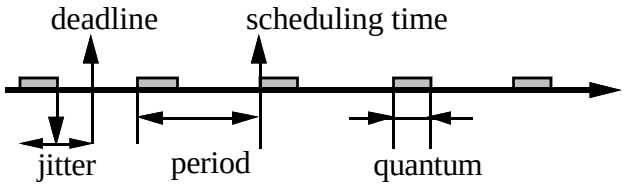

Figure 2.2: Scheduler QoS Parameters derived by QoS Mapping

For admission testing and resource allocation purposes this representation may need to be translated to something more meaningful to the end-system scheduler. For example (as illustrated in figure 2.2), one function of QoS mapping is to derive the period, quantum (i.e., unit of work) and deadline times of the threads associated with the transport level flow performance specification [Coulson,95].

2.4.1.2 Admission Testing

Admission testing is responsible for comparing the resource requirement arising from the requested QoS against the available resources in the system. The decision regarding whether a new request can be accommodated generally depends on system-wide resource management policies and simple resource availability. Once admission testing has been successfully completed on a particular QoS 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.

2.4.1.3 Resource Reservation

Resource reservation protocols arrange for the allocation of suitable end-system and network resources in accordance with 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 QoS module traversed (e.g. CPU, memory, I/O devices, switches, routers) end-to-end resources are allocated and committed. The net result is that QoS control and management mechanisms such as network-level cell scheduler and transport-level flow monitors are configured appropriately.

2.4.2 QoS Control Mechanisms

QoS control mechanisms operate on timescales close to media transfer speeds. They provide real-time traffic control of flows based on requested levels of QoS established during QoS setup. This is achieved by providing suitable QoS control mechanisms and arranging for time-constrained buffer management, scheduling and communication protocol operations. The principle QoS control building blocks include *flow shaping, flow scheduling, flow policing, flow control* and *flow synchronisation.*

2.4.2.1 Flow Shaping

Flow shaping regulates flows based on user supplied flow performance specifications as described in Section 2.3.2. Flow shaping can be based on a simple fixed rate throughput

(i.e., peak rate) or some form of statistical representation (i.e., peak rate, sustainable rate and burstiness factor) of the required bandwidth.

The benefit of shaping traffic is that it allows the QoS architecture to commit sufficient end-to-end resources and to configure flow schedulers and shapers to regulate traffic through the end-systems and network. It has been mathematically proven that the combination of traffic shaping at the edge of the network and scheduling in the network can provide hard performance guarantees. Parekh [Parekh,93] has shown that if a source flow is shaped by a token bucket with leaky bucket rate control and scheduled by the weighted fair queueing service discipline [Keshav,91], it is possible to achieve strong guarantees on delay. This is the theory behind int-serv's guaranteed delay service soon to be offered in an integrated services Internet [Braden,94].

2.4.2.2 Flow Scheduling

Flow scheduling manages the forwarding of flows (chunks of media based on application layer framing) in the end-system [Lui,73] and network (packets/cells) in an integrated manner [Zhang,91]. Flows are generally scheduled independently in the end-systems but may be aggregated or scheduled independently in the network. This is independent of the QoS commitment and the scheduling scheme adopted.

2.4.2.3 Flow Policing

Flow policing can be viewed as the dual of monitoring. The latter - usually associated with QoS management - observes whether the QoS contracted by a provider is being maintained whereas the former 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 [ATMF,95a]. A good flow shaping scheme at the source allows the policing mechanism to easily detect misbehaving flows. The action taken by the policing function can range from accepting violations and merely notifying the user, through to shaping traffic to an acceptable QoS level. It is considered that policing flows in the end-system or network should be a function of the end-system or network level flow scheduling and shaping QoS control mechanisms.

2.4.2.4 Flow Control

Flow control 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 committed in advance. In contrast, closed loop flow control requires the sender to adjust the rate based on feed-back from the receiver [Shenker,93] or network [ATMF,95a]. Applications using closed loop flow control based protocols must be able to rapidly adapt to fluctuations in the available QoS. Fortunately, many multimedia applications are adaptive [Jacobson,93] and can operate in such environments. Alternatively, multimedia 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 and managed in isolation from other competing flows (i.e., resources are fire-walled) for the duration of the session.

2.4.2.5 Flow Synchronisation

Flow synchronisation is required to control the event ordering and precise timing of multimedia interactions. Lip-sync is the most commonly cited form of multimedia synchronisation (synchronisation of video and audio flows at a playout device). Other synchronisation scenarios reported include: event synchronisation with and without user interaction, continuous synchronisation other than lip-sync, continuous synchronisation for disparate sources and sinks. All place fundamental QoS requirements on flow synchronisation protocols [Escobar,92]. Dynamic QoS management associated with flow synchronisation is mainly concerned with the ‘tightness’ of synchronisation between flows.

2.4.3 QoS Management

To maintain agreed levels of QoS it is often not sufficient 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. Over longer monitoring and control feed back intervals [Pacafici,95].

QoS management mechanisms include the following: *QoS monitoring, QoS maintenance, QoS degradation, QoS signalling* and *QoS scalability.*

2.4.3.1 QoS Monitoring

QoS monitoring allows each level of the system to track the ongoing QoS levels achieved by the lower layer. It often plays an integral part in a QoS maintenance (see 2.4.3.2) feedback loop which maintains the quality of service being achieved by the monitored resource modules. Monitoring algorithms operate over different timescales. For example, they can run as part of the scheduler (as a QoS control mechanism) to measure individual performance of on-going flows at the transport packet level or as part of the flow shaper which monitors cells injected into the network. In this case measured statistics can be used to control packet scheduling and flow shaping (which may include cell discard actions) and for admission control. Another use of flow monitoring is determining the playout time of media at a delivery device in the face jitter - this is particularly pertinent for adaptive applications. Alternatively, they can operate as part of a transport level feedback mechanism [Campbell,92b].

2.4.3.2 QoS Maintenance

QoS maintenance compares the monitored quality of service against the expected performance and then exerts a tuning operation (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 QoS modules (e.g., loss via the buffer manager, queueing delays via the flow scheduler and throughput via the flow regulator [Campbell,93a]) or remote resource to uphold the desired performance.

2.4.3.3 QoS Degradation

QoS degradation issues QoS indications (i.e., event QoS Signal) to the users when it determines that the lower layers have failed to maintain the QoS of the flows and nothing further can be done by the maintenance QoS mechanism. In response to such indications, the user can choose either to adapt to the available level of service or scale to a new reduced level of service (i.e., initiate an end-to-end renegotiation which is also known as *QoS scaling).*

2.4.3.4 QoS Signalling

QoS signalling allows the user to specify the interval over which one or more QoS parameters (e.g., delay, jitter, bandwidth, loss, synchronisation) can be monitored and the user is informed of the delivered performance (via a QoS signal) at the end of interval. Both single and multiple QoS signals can be selected depending on whether the user requested QoS management policy. Measured performance metrics are particularly useful in the case of adaptive applications, e.g., adaptive video applications operating over best effort networks as in the case of *vic* [McCanne,94] over today's best effort Internet.

2.4.3.5 QoS Scaling

QoS scaling is comprised of 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 quality of service. Based on the user supplied QoS management policy, QoS adaptation in the end-systems can take remedial actions to scale flows intelligently; to either adapt to the available resource or scale to lower levels of service.

Resolving heterogeneous quality of service issues is a particularly acute problem in the case of multicast flows. Here individual receivers may have differing capabilities to consume audio-visual flows; QoS filtering helps to bridge this heterogeneity gap (i.e., communication capability mismatch) while simultaneously meeting individual receivers' quality of service requirements.

2.4. Summary

In this chapter the importance of QoS control, maintenance and management in distributed systems has been indicated. It has been shown how these functions could be used as building blocks for future QoS frameworks appropriate for the *new environment* - that is, distributed multimedia applications operating over high-speed, QoS configurable networks.

The fundamental modules which underpin thinking on end-to-end QoS have been described and the terminology, principles and concepts which will used as a basis for developing a QoS-A for continuous media communications have been introduced.

Furthermore, the important notions of flow and QoS specification have been introduced as concepts key to capturing, requesting and negotiating end-to-end QoS.

Chapter 3

State of the Art in Quality of Service Research

This chapter assesses the state of the art in quality of service research in distributed multimedia systems. The approach taken is, first, to present work on *QoS in standards* focusing on the work of the IETF, ATM Forum and ISO, respectively. It is the intention of this thesis to build on the discussion of QoS presented in Chapter 1 by providing details on the type of QoS specification, QoS commitment and service models being advocated by the various standards bodies. Next, the current *layer specific QoS* research - considering the distributed systems platform, operating system, transport and network layers - is reviewed. Finally, the work on integrated approaches to QoS provision in system architectures is highlighted by describing several relevant architectures that have emerged recently from the computer communications and telecommunications communities.

3.1 QoS in Standards

Because the effects of the new technological and application environments discussed in Chapter 1 are just beginning to have an impact, it is not surprising that current network architectures fail to address comprehensively the need for QoS support for distributed multimedia applications operating over high-speed networks. To give an impression of the degree of QoS support in present-day systems, this section reviews the OSI reference model, the ATM Forum's networking recommendations and the int-serv working proposed draft standards. Some of this work is still in the early stages of development (e.g., the int-serv working documents are not at present ratified as accepted standards). The int-serv model, OSI reference model and ATM Forum's service model are highlighted as the most prominent of the currently standardised network architectures.

3.1.1 Open Systems Interconnection

The ISO has developed a set of standards for computer communications in the form of the seven-layer OSI reference model and these standards are now mature and widely implemented. However, the OSI reference model evolved in an environment of data-only applications running over low-speed networks and the QoS support provided by the OSI reference model reflects the limited QoS requirements of this class of applications.

Parameter	Description
Throughput	The maximum number of bytes, contained in Service Data Units (SDUs) that may be successfully transferred in unit time by the service provider over the connection on a sustained basis.
Transit delay	The time delay between the issuing of a *data.request* and the corresponding *data.indication*. The parameter is usually specified as a pair of values, a statistical average and a maximum. Those data transfers where a receiving service user exercises flow control are excluded. The computations are all based on SDUs of a fixed size.
Residual error rate	The probability that an SDU is transferred with error, or that it is lost, or that a duplicate copy is transferred.
Establishment delay	The delay between the issuing *connect.request* and the corresponding *connect.confirm*.
Establishment failure probability	The probability that a requested connection is not established within the specified maximum acceptable establishment delay as a consequence of actions that are solely attributable to the service provider.
Transfer failure probability	The probability that the observed performance with respect to transit delay, residual error rate or throughput will be worse than the specified level of performance. The failure probability is, as such, specified for each measure of performance of data transfer discussed above.
Resilience	The probability that a service provider will, on its own, release the connection, or reset it, within a specified interval of time.

Release delay	The maximum delay between the issuing of a *disconnect.request* primitive by the service user and a corresponding *disconnect.indication* primitive issued by the service provider.
Release Failure Probability	The probability that the service provider is unable to release the connection within a specified maximum release delay.

Figure 3.1: OSI Performance-oriented QoS Parameters (dimensions)

QoS support in the OSI reference model is limited to statically defined parameters intended to be supported at the session and transport layers. To enable applications to access QoS facilities, the OSI upper layers (application and presentation layers) simply map QoS dimensions (parameters) through to the lower layers unchanged. At the transport layer, QoS parameters relate to each of the phases of the session; that is, connection establishment, data transfer (aka *media transfer* in the new environment) and connection release.

Parameter	Description
Protection	The extent to which a service provider attempts to prevent unauthorised monitoring or manipulation of user data. The level of protection is specified qualitatively by selecting either *(i)* no protection; *(ii)* protection against passive monitoring; *(iii)* protection against modification, addition or deletion, or *(iv)* a combination of *(i)* and *(ii)*.
Priority	High-priority connections are serviced before lower ones. Lower- priority connection packets will be dropped before high-priority packets should the network become congested.
Cost determinants	A parameter to define the maximum acceptable cost for a network connection. It may be stated in relative or absolute terms. Final actions on this parameter are left to the specific network providers.

Figure 3.2: OSI Non-performance-oriented QoS Parameters

The parameters are also classified as either *performance-oriented* or *non-performance-oriented*. Non-performance-oriented parameters do not directly affect the performance of

the communications but are concerned with protection, priority and cost QoS categories. The complete set of parameters together with their interpretations is presented in figure 3.1 which lists the performance-oriented parameters and in figure 3.2 which lists the non-performance-oriented parameters.

3.1.2 ITU-TS and ATM Forum

The ITU-TS and later the ATM Forum recognised the need for QoS configurability in the emerging standards for B-ISDN which are to be based on ATM networking technology. As a result of this recognition, the ITU-TS issued a series of draft recommendations, known as the I-series recommendations, and ATM Forum a set of interim standards which define a fairly comprehensive set of QoS parameters (which the ATM Forum describes as *QoS attributes*) at the ATM and Adaptation layers. QoS characterisation in ATM networks is applicable at three different levels. The *call control* and *connection levels* are concerned with the establishment (i.e., QoS provision) and release of calls and the allocation of resources along a path of ATM switch nodes. The *cell control* level is concerned with the media transfer phase itself.

The list of QoS attributes in figure 3.1 is comprehensive. It is intended to allow traffic (flows) to be characterised in advance so that the network resource management function (i.e., signalling protocols) can allocate resources to support the desired traffic patterns. The QoS attributes are also used as a baseline to police the traffic inserted at the network by the user to ensure that the user does not attempt to inject media into the network at a higher QoS than previously agreed to. It is also intended to use these parameters to support QoS renegotiation (known as *in-call renegotiation*). Renegotiation of connection QoS has not as yet been approved by the ATM standards bodies.

Parameter	Description
peak cell rate (PCR)	The maximum instantaneous rate at which the user can transmit. For bursty traffic the inter-cell interval and the cell rate varies considerably. The PCR is the inverse of the minimum inter-cell interval.
sustained cell rate (SCR)	This is the average rate measured over a long time interval.

cell loss ratio	The percentage of cells that are lost in the network because of error or congestion and are not delivered to the receiver.
cell transfer delay (CTD)	The delay experienced by a cell between network entry and exit points is called the cell transfer delay. This includes the propagation delays, queueing delays at the various intermediate switches and service times at queueing points.
cell delay variation (CDV)	This is a measure of variance of CTD. High variation implies large buffering for delay sensitive traffic such as voice and video.
burst tolerance (BT)	This determines the maximum burst size that can be sent at the peak rate. This is a bucket size parameter for a leaky bucket algorithm. This is also a product of the maximum burst size (MBS).
minimum cell rate (MCR)	This is the minimal rate desired by the application.

Figure 3.3: ATM Forum UNI 4.0 and PNNI QoS Attributes

The ATM Forum's UNI 4.0 [ATMF,95a] includes a *traffic contract* that provides an indication for the *traffic class* selected by the user in addition to an associated list of QoS attributes. Figure 3.3 illustrates the complete list of QoS attributes. Not all QoS attributes apply to each of the service classes. Figure 3.4 indicates the QoS attributes that are used for each of the service classes. The set of service classes currently specified by the ATM Forum and ITU-TS includes:

• *Constant Bit Rate (CBR)* which emulates a leased line and is suitable for real-time applications which generate constant bit rate flows. The service is essentially circuit emulation and for applications which need a strong bound on delay and jitter;

• *Variable Bit Rate (VBR)* which allows applications to send at a variable rate is sub-divided into two categories: real-time VBR and non-real-time (NRT) VBR. The major difference between the two categories is that while CTD is applicable to both categories, CDV is only meaningful for real-time (RT) VBR;

• *Available Bit Rate (ABR)* primarily designed for data but is also considered by many to be appropriate for continuous media as well. Depending on the congestion

state of the network, the source is obliged to control its rate using a feedback mechanism; and

• *Unspecified Bit Rate (UBR)* this approximates the best effort service offered by today's Internet. UBR is designed for those applications that desire the use of any left over resources and are insensitive to cell loss and delay variation.

QoS Attribute	ATM Layer Service Categories				
	CBR	VBR (RT)	VBR (NRT)	ABR	UBR
CLR for CLP = 0	specified			specified	unspecified
CLR for CLP = 1	optional			specified	unspecified
CTD	specified		specified	unspecified	unspecified
CDV	specified		unspecified	unspecified	unspecified
PCR and CDVT	specified			specified	specified
SCR and BT	n/a	specified		not applicable	
MCR	not applicable			specified	n/a
Control Information	not applicable			yes	n/a

Figure 3.4: ATM Layer Service Categories and QoS Attributes

3.1.3 IETF int-serv and RSVP Groups

The IETF int-serv [Braden,94] and RSVP working groups [Zhang,95] are the primary generators of solutions to the end-to-end QoS problem stated in Chapter 1. While the int-serv charter broadly covers the definition of services and mechanisms to support end-to-end QoS, in an internetwork the work of RSVP is solely focused on the design, implementation and integration of a new reservation signalling protocol called RSVP. The two work items are crucial to the advancement of multimedia services in the Internet.

Work by the int-serv group (which is outlined in Chapter 1) has included the definition of a QoS framework used to specify the functionality of internetwork system components (called *'elements')* which support the multiple, dynamically selectable qualities of service to

applications in an internetwork. The behaviour of elements such as routers, subnetworks and end-system operating systems is captured as a set of services some or all of which are offered by each element. Each element is QoS-aware and supports interfaces required by the service definition [Shenker,95]. The concatenation of these services along an end-to-end media path used by an application provides an overall statement of end-to-end QoS.

The following int-serv services are offered in addition to the current best effort service:

- *controlled delay* [Shenker,95b], which tries to provide several levels of delay which the application can choose between;

- *predicted delay* [Shenker,95b], which provides a statistical delay bound similar to the Tenet Group's statistical service [Ferrari,92] and the Comet Groups guaranteed service [Lazar,90];

- *guaranteed delay* [Shenker,95b], which provides an absolute guaranteed delay bound.

Each service definition also specifies QoS parameters used to invoke the service. The current set of defined services are appropriate for a wide range of multimedia flows and data applications. While the definition emphasises the 'network' element the framework in its full generality is as applicable in the end-system as the network. The goal of the int-serv group is to focus on the network, in the first instance, then the end-system.

Flows are characterised by two specifications: (i) a *traffic specification (TSpec)*, which is a specification of the traffic pattern which a flow expects to exhibit. An example of this specification might take the form of the flow's bandwidth and burstiness specified by a token bucket [Partridge,92]; and (ii) a *service request specification (RSpec)*, which is a specification of the quality of service a flow desires from a network element and the form which is highly specific to a particular service. An example of this specification might take the form of representing the maximum tolerable delay. In the case of the guaranteed delay service outlined above, the service is invoked specifying the *traffic (TSpec)* and the desired *service (RSpec)*. The TSpec takes the form of a token bucket, with a *bucket depth (b)* and *bucket rate (r)*. The RSpec is a *rate (R)*, where R must be greater than or equal to *r*. Traffic specifications are most frequently created by the originator of the data flow. In contrast, service request specifications may originate from either the sender or receiver of a flow.

The integrated service model is restricted to the network in the first instance and comprises four components:

i) a *packet scheduler*, which forwards packet streams using a set of queues and timers;

ii) a *classifier*, which maps each incoming packet into a set of QoS classes;

iii) an *admission controller*, which implements the admission control algorithm to determine whether a new flow can be admitted or denied; and

iv) a *reservation setup protocol* (e.g., RSVP [Zhang,93]), which is necessary to create and maintain the flow-specific state in the routers along the path of the flow.

3.1.4 Assessment

The current OSI service definitions do not provide for the specification of a range of QoS dimensions including jitter, criticality and cost. In addition, there is no support for QoS monitoring or interface for renegotiation. The precise semantics of responsibilities and guarantees are not clear. More limiting is the fact that at the protocol level there is no notion of QoS management in terms of QoS negotiation, mapping, resource allocation, and QoS maintenance. It is assumed that the underlying network provider will support the requested QoS levels.

The ATM Forum recommendations are comprehensive and include a detailed traffic characterisation model in terms of ATM service categories and associated QoS attributes. The ATM service model characterises a flow by traffic type (based on the ATM service categories) and a set of QoS parameters (based on the ATM QoS attributes) which are subsumed in a traffic contract. This approach eliminates the need for QoS classes in UNI 3.0. At present each virtual connection has a traffic contract associated with it. The major service-related limitation here is the lack of consideration of how traffic characterisation at the ATM layer can be derived from user QoS needs at the transport layer and above. And how applications level QoS requirements are mapped to the current set of categories and QoS attributes. Below the service interface, the current state of ATM standardisation suffers from a comparable lack of QoS management support to that found in the OSI field.

There are a number of similarities in the approach of the ATM Forum and IETF in meeting QoS communications requirements. Both groups while approaching the QoS-problem from different perspectives (vis-a-vis hard state and soft state approaches of the Forum and IETF, respectively) have complementary approaches to the specification of

flows. As noted by Borden et al. [Borden,95], within the Internet Community it is assumed that traffic characterisation (TSpec) will, in general, be bursty and that bursty traffic can be modelled by a token bucket. While ATM does not assume all traffic to be bursty, it uses an equivalent formulation for QoS characterisations of traffic as the Internet Community. This is referred to as the *Generic Cell Rate Algorithm (GCRA)* in UNI 4.0 [ATMF,95a]. Interworking between the services classes of the ATM Forum and IETF affords less flexibility.

3.2 Layer-Specific QoS

In this section, layer-specific quality of service research (from non-standards organisations) is reviewed. The distributed systems platform, operating system and transport and network layers are considered.

3.2.1 Distributed Systems Platform

There has been considerable research in the area of distributed systems platforms over the past ten years [Mullender,93]. Until recently, however, there has been very little work on quality of service support in such platforms. With the emergence of distributed multimedia applications, however, quality of service has become a major issue in distributed systems research. In a distributed system, there are three areas where quality of service is applicable: i) message passing services, which allow the programmer to explicitly send a message between two or more processes in a distributed system; ii) remote invocation, which allows operations in a server process to be invoked by a client process [APM,91]; and iii) stream services, which are connections that support the transmission of continuous media flows [ODP,92]. A number of experimental QoS-driven distributed systems platforms are now beginning to emerge. Researchers at Lancaster University have developed an extended version of ANSAware [APM,91] featuring bounded invocations and QoS-controlled streams [Coulson,92]. Similar work has also been undertaken at Cambridge University [Nicolaou,90]. More recently, research on quality of service has centered on ODP standardisation. Ongoing research programs at CNET [Hazard,93], and BBN and Rome Labs [Zinky,95] are developing new languages to specify QoS for both operational and stream interface. The CNET work uses QoS logic statements in the

language to generate quality of service monitors. The BBN and Rome Lab research promotes object level QoS specification (i.e., methods per second) and not at the communication level (i.e., bits per second). Both approaches allow quality of service to be negotiated, measured and enforced. For full details on the state of the art in distributed systems support for quality of service see [Vogel,94].

3.2.2 Operating Systems

There has been considerable progress in operating systems support for quality of service with most progress having been made in the specific areas of communication protocols [Feldmeier,93] and scheduling [Govindan,91]. There has been considerably less work on the integration of the various components into an overall operating system [Coulson,95]. Communication protocol implementation involves predictability issues such as the need for correct scheduling of protocol activities and efficiency issues such as minimisation of data copying, system calls, interrupt handling and context scheduling, an avoidance of multiplexing, the use of hardware assists for protocol processing and the importance of executing protocol code in a schedulable process rather than as a interrupt service routine. Much of the work has looked to maintain a level of compatibility with the de facto UNIX interface. Two main approaches can be identified: i) modifying existing UNIX implementations, and ii) completely re-implementing UNIX. In the first approach, alterations are made to the existing UNIX kernel to provide more predictable behaviour. Hanko [Hanko,91] describes a proposal for time-driven resource management which allows applications to signal their likely forthcoming resource requirements in terms of QoS parameters such as quantity deadline and priority. The system will not attempt to guarantee performance, but instead will bias available resources in the requested directions and concentrate on degradation of service, optionally accompanied by notification of degradation to the effected process. The second approach is to re-implement UNIX in terms of the micro-kernel model. Examples of micro-kernels capable of supporting UNIX interfaces are Chorus, Mach and Amoeba. Work has been undertaken at CWI, Amsterdam to support continuous media in an Amoeba-based UNIX environment [Bulterman,91]. Other significant work is being carried out using Mach [Tokuda,93], Chorus [Coulson,95], Peagus [Leslie,93] and YARTOS [Jeffay,93] (and its rate-based extensions [Jeffay,93]) as the basis of a distributed system with end-to-end QoS support for continuous media. In the area of device management several research groups have suggested a new architecture for

multimedia systems in which multimedia devices that typically reside on the workstation I/O bus are placed on a high-speed ATM network. These architectures are typically referred to as *Desk Area Network (DANs)* [Hayter,91] [Tennenhouse,94]. It is still too early to determine what, if any, new requirements DAN architecture places on future operating system design [McAuley,94]. For full details on the state of the art in operating systems support for quality of service see [Hutchison,94].

3.2.3 Transport Layer

A number of research teams have investigated the provision of quality of service at the transport layer. Early work specifically addressed the provision of rate based protocols over high speed networks, e.g., XTP [Chesson,88] and NetBlt [Clark,87]. More recently protocols have emerged which are designed specifically to meet the needs of continuous media. The Esprit OSI 95 project proposed an enhanced transport service and protocol called TPX [Danthine,92]. TPX provides support for connection-oriented services with sequenced delivery, QoS configurable and re-negotiable QoS and error notification. The enhanced connection-oriented service takes QoS parameters relating to throughput, delay, delay jitter, error selection policy and relative priority. Three transport quality of service semantics in addition to “best effort” are proposed for this service: compulsory, threshold and maximal QoS. The Tenet Group at the University of California at Berkeley has developed CMTP[Wolfinger,91] which operates on top of RTIP [Ferrari,95] and provides sequenced and periodic delivery of continuous media samples with QoS control over throughput, delays and error bounds. Notification of all undelivered and/or corrupted data can be provided if the client selects this option. The HeiTS project [Hehmann,91] at IBM Heidelberg has developed a transport system which has concentrated on the integration of transport QoS and resource management (primarily CPU scheduling). HeiTS places emphasis on an optimised buffer pool which minimises copying and also allows efficient data transfer between local devices. Other significant work has come from Schulzrinne, Casener and Van Jacobson who have developed RTP [Schulzrinne,95] for the Internet suite of multimedia tools [Kanakia,93]. Other work [Campbell,92] reports on the development of a continuous media transport and orchestration service. For a full review of the state of the art in transport protocols and services see [Feldmeier,93] [Doeringer,90].

3.2.4 Network Layer

The subject of providing quality of service guarantees in integrated service networks has been widely covered in the literature [Keshav,93]. The multimedia networking community has developed sophisticated traffic models, control and management architectures for multimedia communications. Extensive work has considered flow specification, flow admission control, resource reservation, traffic shaping and queue management schemes. For researchers working on multimedia networking, the primary aim has been to provide performance bounds while exploiting statistical multiplexing of bursty sources to efficiently utilise bandwidth. Kurose [Kurose,93] provides a categorisation of the different approaches used in providing QoS guarantees found in the literature: (i) a tightly controlled approach, which is based on non-work conserving multiplexing service (queueing) disciplines (e.g., stop-and-go [Golestani,90] and Tenet's EDD [Ferrari,90]) which preserves the traffic shape guaranteeing the delivered flow characteristics are the same as the source; (ii) an approximate approach, which as its names suggests is based on a simple characterisation of the source model (e.g., equivalent capacity [Guerin,91]) and which can provide approximate guarantees using simple service disciplines such as FIFO taking advantage of statistical multiplexing gain; (iii) a bounding approach, which takes into account any distortion of the flow as it traverses work-conserving multiplexors (e.g., packetised generalised processor sharing [Parekh,93] and weighted fair queueing [Keshav,91]) resulting in mathematically provable performance bounds for statistical and deterministic service guarantees [Cruz,91]; and finally (iv) an observation-based approach, which uses measured behaviour (e.g., COMET's approach [Hyman,90] and Clark's predictive service [Clark,92]) of the aggregate traffic and the user supplied flow specification when making admission decisions.

The work on an integrated services Internet [Braden,94] is a significant contribution to providing QoS guarantees on a per-flow basis. The integrated service model is comprised of four components: (i) a packet scheduler, which is based on the CSZ scheduler [Clark,92] and Class Based Queueing (CBQ) [Floyd,93], which forwards packet streams using a set of queues and timers; (ii) a classifier, which maps each incoming packet into a set of QoS classes (iii) an admission controller, which implements the decision control algorithm to determine whether a new flow can be admitted or denied and; (iv) a reservation setup protocol, which is necessary to create and maintain flow-specific state in the end-systems and in routers along the path of the flow. There have been a number of significant

contributions to reservation protocols in communication networks which have emerged over the past few years: ST-II [Topolcic,90] and SRP [Anderson,91], and more recently RSVP [Zhang,93], RCAP [Banerjea,91] and HieRAT [Volg,95] and UNI 4.0 [ATMF,95]. For a full review of the state of the art in network support for QoS see [Keshav,93] [Kurose,93].

3.3 Emerging QoS Architectures

Until recently, research in providing QoS guarantees has mainly 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 to which end-systems are attached [Gopalakrishna,94]. Work on QoS-driven end-system architecture must be integrated with network configurable QoS services and protocols to meet application-to-application requirements [Campbell,92]. In recognition of this, researchers have recently proposed new communication architectures which are broader in scope and cover both network and end-system domains. This section reviews a number of distinct approaches which have recently emerged in the literature and which are born out of the Telecommunications, Computer Communications and Standards communities:

- *Extended Integrated Reference Model (XRM),* which is being developed at Columbia University;
- *OSI QoS Framework,* which is being developed by the ISO SC21 QoS Working Group;
- *OMEGA Architecture*, which is being developed at the University of Pennsylvania;
- *Heidelberg QoS Model*, which is being developed at IBM's European Networking Center;
- *Tenet Architecture,* which is being developed at the University of California at Berkeley;
- *End System QoS Framework,* which is being developed at Washington University;

• *IETF QoS Manager (QM)*, which is being developed by the int-serv group as part of its strategy for an integrated services Internet;

• *TINA QoS Framework*, which is being developed by the TINA Consortium; and

• *MASI End-to-End Architecture*, which is being developed at Université Pierre et Marie Curie.

In addition to examining the work above, section 3.3.10 outlines other QoS frameworks reported in the literature.

3.3.1 The Columbia University Extended Integrated Reference Model

The COMET group at Columbia University (New York) is currently developing an Extended Integrated Reference Model (XRM) [Lazar,94] 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; both classes of devices can be modeled as producers, consumers and processors of media. COMET organises the XRM into five distinct planes as illustrated in figure 3.5:

i) *management function*, which resides in the network management plane (N-plane), covers the OSI functional areas of network and system management;

ii) *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;

iii) *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

iv) *telebase*, which resides in the data abstraction and management plane (D-plane), collectively represents the information data abstractions existing in the network and end-systems. The telebase implements the principles of data sharing (via

asynchronous resource management) among all other XRM planes.

The subdivision of XRM into different planes is motivated by a number of QoS principles: the separation and layering principles [Lazar,92] and the asynchronous resource management principle [Lazar,92]. The subdivision between the management and traffic control functions, on one hand, and the information transport functions on the other, is based on the principle of separation between control and communication. The separation between management and traffic control is due to the different timescales on which these planes operate. This is, in turn, motivated by the asynchronous resource management principle.

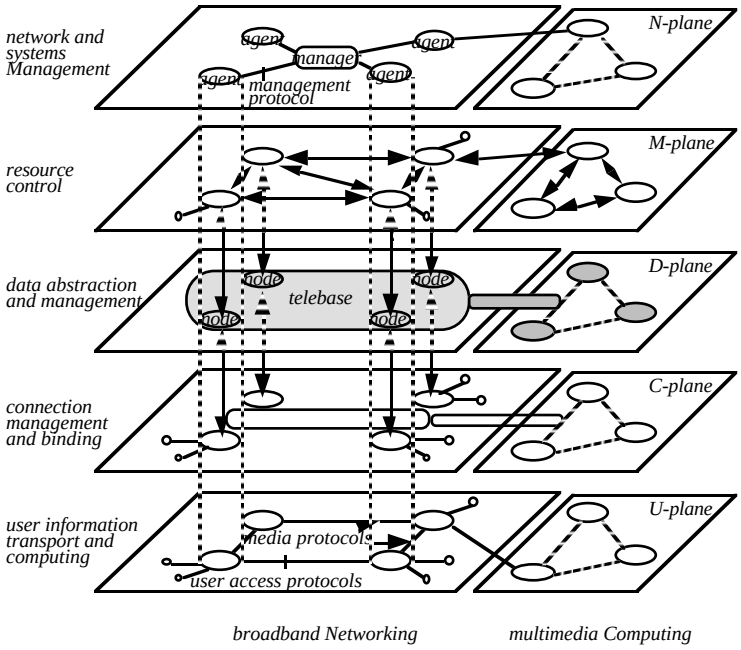

Figure 3.5: The Columbia University XRM

The XRM is built on theoretical work guaranteeing QoS requirements in ATM networks and end-systems populated with multimedia devices. General concepts for characterising the capacity of network [Hyman,92] and end-system [Lazar,94] devices (e.g., disks, switches) have been developed. At the network layer, XRM characterises the capacity region of an ATM multiplexer with QoS guarantees as a scheduleable 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 characterised 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.

The scheduleable region represents the multidimensional capacity of the multiplexer; its dimensionality depends on the number of traffic classes and represents the stability region. The scheduleable region is a resource abstraction that allows a separation of times scales: the time scales of cells and the time scale of call arrivals and departures. In [Hyman,92] it is shown how separation of time scales is an appropriate tool for resolving admission control decisions. Based on a calculus of scheduleable regions, the QoS in the network can be guaranteed. The three traffic classes in figure 3.6 correspond to video, voice and data flows. Class I traffic is characterised by a frame duration of 62.5 ms and a peak rate of 10 Mbps, Class II traffic is modelled as an on-off source with constant arrivals with an exponentially distributed active period and 64 Kbps peak rate, and Class III traffic is modeled as a Poisson source with 1 Mbps average rate. The surface depicted in figure 3.6 delimits the capacity region of the multiplexer. Any combination in the number of calls (i.e., active flows) below this surface has its QoS guaranteed.

XRM models the end-system architecture as a multiprocessor based multimedia workstation, comprised of the following multimedia devices: (i) an audio and video unit, which is responsible for multimedia processing, and supports media processing tasks in a deterministic manner, and runs on a dedicated processor(s); (ii) an input/output subsystem is similarly modeled, separately through a disk storage unit, and is also run on a separate processor(s); (iii) a main processor unit runs the system tasks, both to increase speed and to remove external interrupts, as well as the other operating system overhead associated with application tasks. 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. Quality of service for these classes is specified by a set of frame delay and loss constraints. The methodology of characterising 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. One significant difference between the scheduleable region and the multimedia capacity region is the number of classes supported. The number of service classes at the user level is expected to far exceed the number of traffic classes at the multiplexer. A number of service classes, however, can be mapped onto a single traffic class of the multiplexer, and therefore, the support of a large number of service classes will not require an increase in the number of traffic classes.

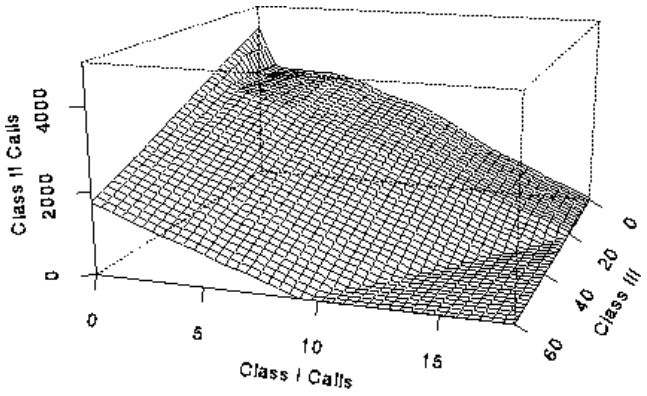

Figure 3.6 Scheduleable Region

The implementation of XRM including key resource abstractions such as the scheduleable and multimedia capacity region is currently being realised as part of a binding architecture [Lazar,94]. The binding architecture achieves seamless binding between networking and multimedia devices. The building blocks of the architecture consist of a set of interfaces, methods and primitives. The former abstracts the functionalities of multimedia networking devices and organises them into a binding interface base (BIB). The methods and primitives are invoked for implementing binding applications. Communication between the interfaces of the architecture is supported by OMG's CORBA [OMG,93]. Binding requirements arise in each of the planes of the XRM. Dynamic binding requirements, however, are particularly demanding in the C and M planes of the XRM. The binding architecture resides in the M, D and C-planes of the XRM. Specifically, the binding interface base resides in the D-plane and the binding algorithms execute from within the M and C-planes. The binding architecture represents a software environment on top of which binding applications execute. Examples of binding applications arise in connection set up for broadband networks, distributed systems implementing flow synchronisation protocols,

resource allocation protocol such as those intended for the Internet [Zhang,93], multimedia computing platforms, etc. New binding applications can be added without changing the underlying binding architecture.

3.3.2 The OSI QoS Framework

One early contribution to the field of QoS-driven architecture is the OSI QoS Framework [ISO,95a] which concentrates primarily on quality of service support for OSI communications [Sluman,91]. 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 QoS framework concepts include:

- *QoS requirements,* which are realised through QoS management and maintenance entities;
- *QoS characteristics,* which are a description of the fundamental measures of QoS that need to be managed in the communication system;
- *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 (illustrated in figure 3.7) is made up of two types of management entity that attempt to meet the QoS requirements by monitoring, maintaining and controlling end-to-end QoS.

Layer-specific entities: The task of the policy control function is to determine the policy which applies at a specific layer of the 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 generalised. 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.

System-wide entities: 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 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.

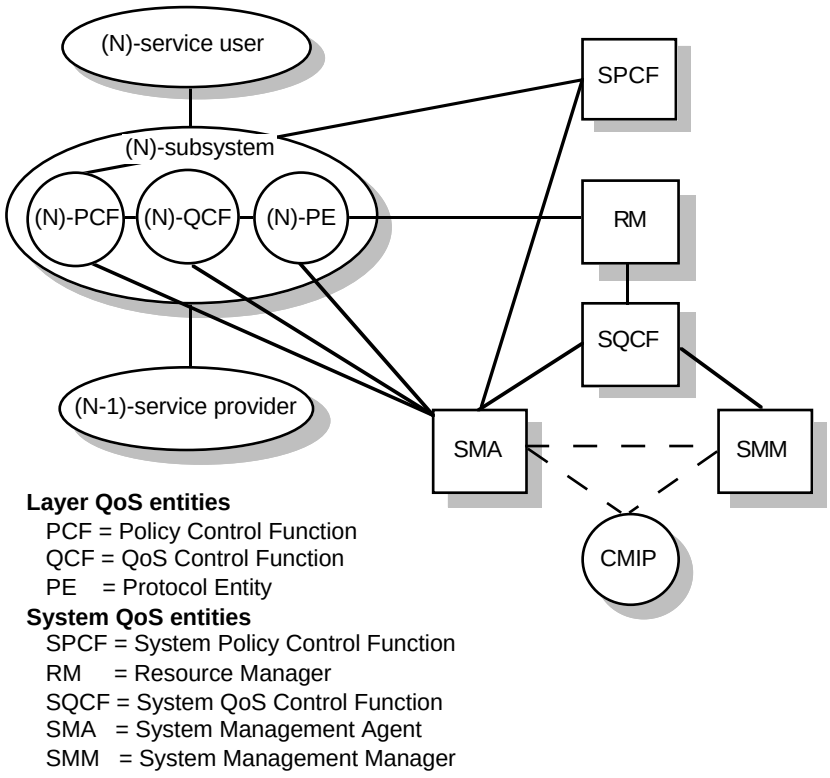

Figure 3.7 OSI QoS Framework

3.3.3 The Pennsylvania University OMEGA Architecture

During the last three years the University of Pennsylvania has been developing an end-point architecture called the OMEGA architecture [Nahrstedt,95c]. OMEGA is the result of

an interdisciplinary research effort 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 assumes a network subsystem that 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. Communications is preceded by a call setup phase where application requirements, expressed in terms of QoS parameters, are negotiated, and guarantees are made at several logical levels, such as between applications and the network subsystem, applications and the operating system, network subsystem and operating system. This establishes customised connections and results in the allocation of resources appropriate to meet application requirements and operating system/ network capabilities.

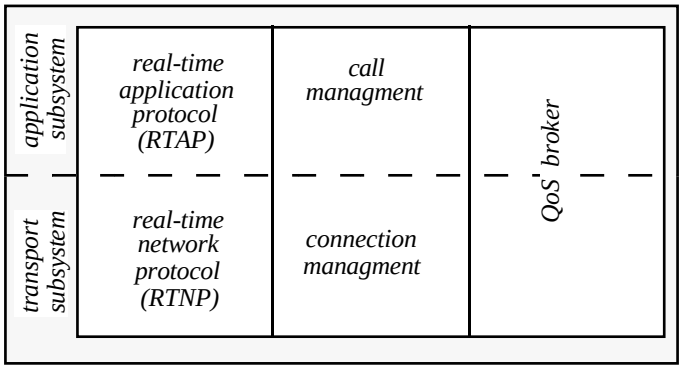

Figure 3.8: OMEGA Communication Model

To facilitate this resource management process the University of Pennsylvania has also developed a brokerage model [Nahrstedt,95a] which incorporates QoS translation and QoS negotiation and renegotiation (see [Vogel,94] for full details on similar work on QoS negotiation protocol at University of Montreal).The notion of eras is introduced in [Nahrstedt,93] to describe variations in QoS parameters for complex, long-lived applications. Negotiation and renegotiation provide a mechanism to signal variations in QoS performance parameters at the user–network interface. They are invoked at era boundaries and can aid resource allocation. In the model, application requirements and network resource allocation are expressed in fundamentally different terms and languages. A key part of the model, called a QoS Broker [Nahrstedt,95a] is responsible for the translation of QoS at the user–network interface. More recent work has addressed operating system issues such as admission control for guaranteed QoS [Nahrstedt,94] see also

[Nahrstedt,95b] for a comprehensive survey of resource management issues in networked multimedia

The OMEGA architecture is partitioned into distributed and local components: communications model and a resource model at the end-points. The communication system is modeled as a two layer system as illustrated in Figure 3.8. The transport subsystem layer is based on the performance principle (integrated layer processing). Functions such as connection management, forward error correction, timing failure detection and timely data movement form the core of the Real-Time Network Protocol. The application subsystem layer contains the functions of the application and session layers such as call management, rate control of multimedia devices, input/output functions (e.g., display of video), fragmentation and reassembly of application data units. These functions are the core of the Real-Time Application Protocol. (RTAP). Both subsystems must provide QoS guaranteed services over specified calls/connections for applications. Therefore, they require guarantees on the resources needed for communications. Resource guarantees are negotiated during call establishment by the QoS Broker protocol [Nahrstedt,95a] which is present in both the application and transport subsystems. For full details of the communication and resource models see [Nahrstedt,95d].

3.3.4 The Heidelberg QoS Model

The HeiProject at IBM's European Networking Center in Heidelberg have developed a comprehensive QoS model which provides guarantees in the end-systems and network [Volg,92]. The communications architecture includes a continuous media transport system (HeiTS/TP) [Hehmann,91] which provides QoS mapping and media scaling [Delgrossi,93] as illustrated in figure 3.9. Underlying the transport is an internetworking layer based on ST-II which supports both guaranteed and statistical levels of service. In addition, the network supports QoS-based routing (via a QoS finder algorithm) and QoS filtering. Key to providing end-to-end guarantees is HieRAT (resource administration technique), based on initial work in [Anderson,91]. HeiRAT comprises a comprehensive QoS management scheme which includes QoS negotiation, QoS calculation, admission control and QoS enforcement, and resource scheduling [Volg,95]. The HeiRAT scheduling policy used in the supporting operating system is a rate-monotonic scheme whereby the priority of an operating system thread performing protocol processing is proportional to the message rate accepted.

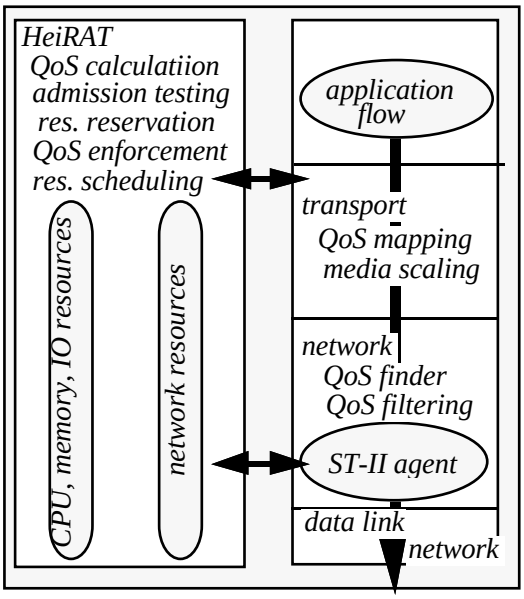

Figure 3.9: Heidelberg QoS Model

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 respectively. Media scaling and codec translation [Schulzinne,95] at the end-systems and flow filtering [Pasquale,92] [Yeadon,94] [Volg,95] and resource sharing [Zhang,93] [Ferrari,95] 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 they traverse bridges, switches and routers. Both schemes compensate for a variation in network load/performance by re-scaling or filtering the delivered QoS, respectively. Potentially this includes manipulating hierarchical flows, for example, delivering the I frames of an MPEG encoded flow and dropping the P and B frames to match the end system or network QoS constraints. Network level filtering looks very promising when used in conjunction with multicast protocols for dissemination of continuous media in support of heterogeneous receivers. Pasquale et. al [Pasquale,92] suggest that several receivers having disparate QoS communication requirements and needing to access the same video flow simultaneously can be supported by a propagating filter scheme which delivers the appropriate QoS to each receiver. This scheme promotes efficient use of network resources and, as the literature

suggests, reduces the likelihood of the onset of congestion.

3.3.5 The Tenet Architecture

The Tenet Group at the University of California at Berkeley has developed a family of protocols [Ferrari,95] which run over an experimental wide area ATM network. The protocol family (as illustrated in figure 3.10) includes a Real Time Channel Administration Protocol (RCAP) [Banerjea,91] as well as Real Time Internet Protocol (RTIP) and Continuous Media Transport Protocol (CMTP) [Wolfinger,91]. 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 client interface to CMTP includes facilities to specify traffic characteristics in terms of burstiness, which is useful for variable bit rate encoding techniques, and workahead, which allows the protocol to deliver faster than the nominal rate if data and resources are available. The Tenet Group [Ferrari,90] makes a distinction between deterministic and statistical guarantees for hard real-time and continuous media flows, 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 in a session might be lost.

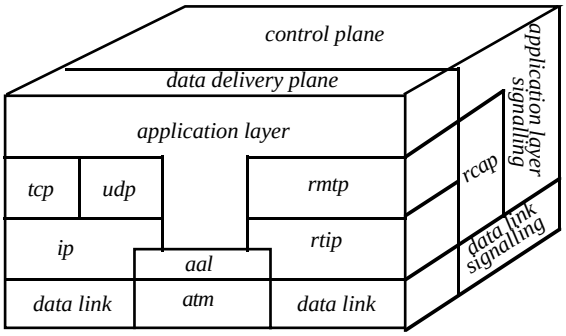

Figure 3.10: Tenet Architecture

The Tenet Architecture includes an application layer signalling protocol which spans the

end-system and the network and provides QoS mapping between the application, transport and network layers, translating QoS constraints at each layer into a form which is needed by resource reservation protocols RCAP. The architecture also includes a scheme for dynamically managing real-time channels called Dynamic Connection Management (DCM) which supports media scaling (i.e., QoS adaptation). The motivation that underpins dynamic connection management is to increase network availability and flexibility. The adaptation can be initiated by the application or by the network. For full details on modification contract and adaptation algorithms see [Ferrari,95]. Dynamic connection management guarantees either a transition from a primary to an alternative channel without any bound violations or a transition where a number of packets involved in a performance violation is bounded. Recently, the Tenet Group suite of protocols has been evolving to support multicast flows with heterogeneous QoS constraints [Ferrari,95].

3.3.6 The IETF QoS Manager

In [IETF,95] Clark introduces some early work on a Quality of Service Manager (QM) for an integrated services Internet suite of protocols. The QM (illustrated in figure 3.11) presents an abstract management layer designed to isolate applications from underlying details of specific services provided in QoS-driven Internet as described earlier in section 3.1.3. 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; in this case, the QM provides a degree of transparency whereby applications express desired levels of QoS in user-oriented language rather than using communication specifics. The QM is responsible for determining what QoS management capabilities are available on the application's communication path and choosing the path best suited to the application. There are a number of benefits from the migration of specific services knowledge from the application to the QM:

- *heterogeneity* is supported; the QM can match application needs with the underlying QoS capability;

- *transparency* is provided; applications will not need to be aware of the details of specific QoS management capability; and

- *extensibility* is supported; new QoS capabilities can be more easily deployed in the Internet because applications need not be modified as new services become

available.

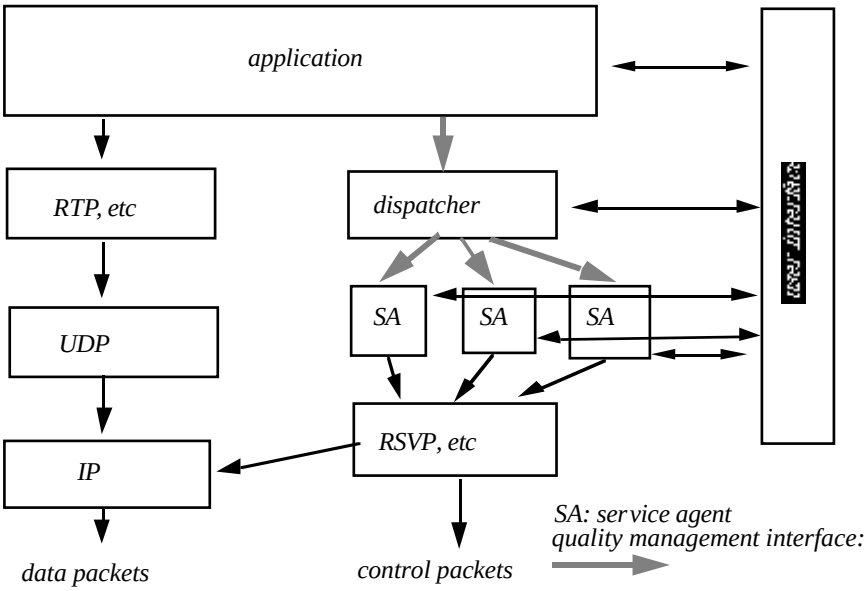

Figure 3.11: IETF int-serv QoS Manager

The initial thrust of the work has been to map application specific needs to one of the new set of integrated services (e.g., [Shenker,95]) and provide support for monitoring of performance. In the future, however, the interface between the application and QM may cover more general issues such as cost of service as well as more technical matters such as delay and bandwidth. In related work, Partridge [IETF,95] presents a multimedia-based Berkeley Sockets specification which includes support for flows in terms of a flow-spec and QoS management aspects of Clark's QM.

3.3.7 The Washington University End-System QoS Framework

Other work at Washington University by Gopal and Purulkar [Gopalakrishna,94] has resulted in the development of 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 figure 12: QoS specification, QoS mapping, QoS enforcement and protocol implementation. QoS specifications are at a high level and use 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 session of the application. Important resources considered are the CPU and network connection.

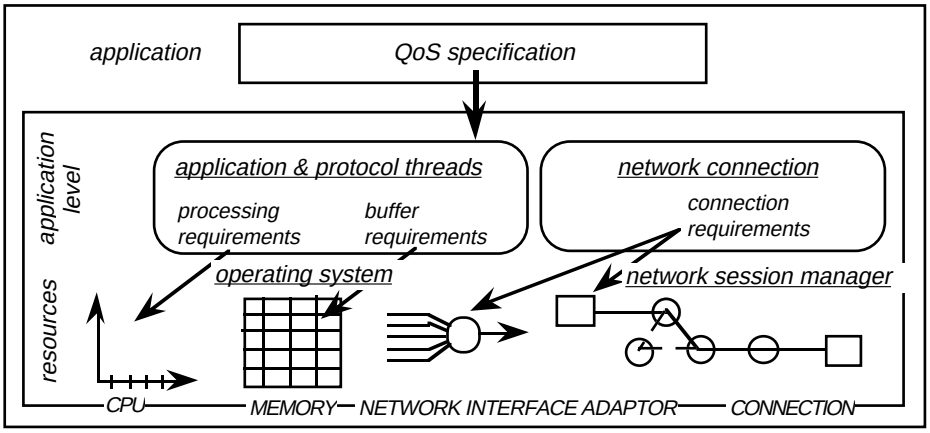

Figure 3.12 End-System QoS Framework: QoS Specification and Mapping

The third component of framework is QoS enforcement. QoS enforcement is primarily concerned with providing real-time processing guarantees for media transfer. A real-time upcall (RTU) facility [Gopalakrishna,95] has been developed for structuring protocols. RTUs are scheduled using a rate monotonic policy with delayed pre-emption 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. The research considers QoS specification, QoS mapping and QoS enforcement (i.e., rate shaping) as fundamental end-system QoS mechanisms integrated with the protocol implementation model. The notion of QoS within the end-system is extended from the network interface driver, through the protocol layers and up to the application threads that generate/consume media.

3.3.8 The TINA QoS Framework

TINA (Telecommunications Information Networking Architecture) architectural concepts are grouped into four functional domains: Computing Architecture, Service Architecture, Network Architecture and Management Architecture [Nilison,95]. The TINA

approach considers telecommunications software as a large, distributed software system and applies to it distributed computing and object oriented design techniques. The TINA Computing Architecture, which is largely based on the *ODP reference model* (RM-ODP) [ODP] and influenced by the work of the OMG [OMG,93], provides a basis for interoperability and reuse of distributed telecommunication software. The TINA QoS Framework [TINACa,95] describes a framework for specifying QoS aspects of distributed telecommunications within the context of the Computing Architecture. The QoS framework addresses the computational and engineering viewpoints of distributed telecommunications applications. Figure 3.13 illustrates the structure of the telecommunications software in the TINA Computing Architecture. It is governed by the separation between telecommunication applications and the *Distributed Processing Environment (DPE)* in the first instance; multimedia services offered by a provider utilise the DPE and the underlying computing and communications capabilities. These underlying capabilities correspond to operating system functions that are characterised by distinct native configurations. A TINA node is comprised of a DPE kernel, a *Native Computing and Communication Environment (NCCE)* and a hardware platform.

The TINA QoS framework is partly based on work in the literature (e.g., ANSA QoS Framework [Guangxing,94] and CNET Framework [Hazard,93]). In the computational viewpoint, QoS parameters required to provide guarantees to objects are stated declaratively as service attributes. In the engineering model, QoS mechanisms employed by resource managers are considered. By stating QoS requirements declarative, applications are relieved of the burden of coping with complex resource management mechanisms needed for ensuring QoS guarantees; this is motivated by the principle of QoS transparency.

Computational specification describes applications in terms of computational entities (i.e., objects) that interact with each other. Objects interact via operational interfaces (which correspond to client-server interactions) and stream interfaces (which represent a set of communication end-points producing or consuming continuous media). Computational objects can support multiple operational and stream interfaces. The TINA QoS Framework supports three types of QoS specification at the object and interface level:

i) the object QoS specification details any distinction between the offered and expected quality of service of an object. The quality of service categories currently considered at this level include availability, security, performance (in terms of response time) and reliability;

ii) the operational QoS interface specification focuses on timeliness and availability of quality of service categories: availability is concerned with maximising the likelihood that a service provided is available when requested and timeliness is concerned with timing constraints of operational interactions;

iii) the stream QoS interface specification includes stream flow signatures and synchronisation constraints on stream flows. The quality of service categories currently being considered at the stream level include throughput, delay, jitter and error rate.

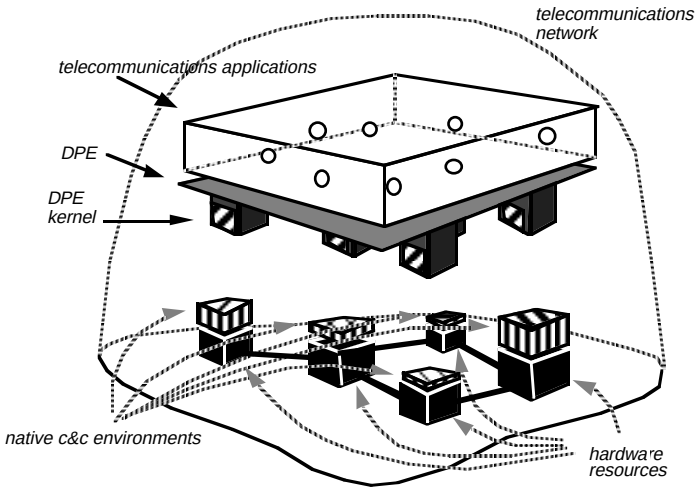

Figure 3.13 TINA-C Schematic

A computational specification language has been developed by the TINA consortium: TINA-ODL. It is an extension of OMG-IDL [OMG,93] for describing computational objects and their operational and stream interfaces. TINA-ODL provides a service attribute construct to capture the QoS specification of quality of service constraints. In related work [Leydekkers,95], an environmental contract is introduced. This allows the applications to specify the computational objects and related interfaces in a contract which is a binding agreement between user and service provider. Three levels of QoS are described: deterministic, statistically reliable and best effort. Each quality of service parameter specified in the environmental contract may have a different level of service attributed to it.

The concept of “binding” is used to address the QoS of an interactive session involving

multiple computational objects. The binding of computational interfaces is mapped down in the engineering viewpoint as a “channel”. A channel is comprised of three modules: stub, binder and protocol adapter. Figure 3.14 illustrates the computational and corresponding engineering view of a set of objects interacting. In the engineering viewpoint, the objects are distributed in different nodes. The TINA-DPE kernel running in each node offers applications QoS support. Application level QoS requirements are mapped down to services offered by the DPE kernel and the underlying NCCE. Mechanisms for reporting violations in the contracted quality of service guarantees are provided. Quality of service provision is considered to be either static (where the service contract is non-renegotiable) or dynamic (where the service contract is open to renegotiation by either the DPE kernel or the application).

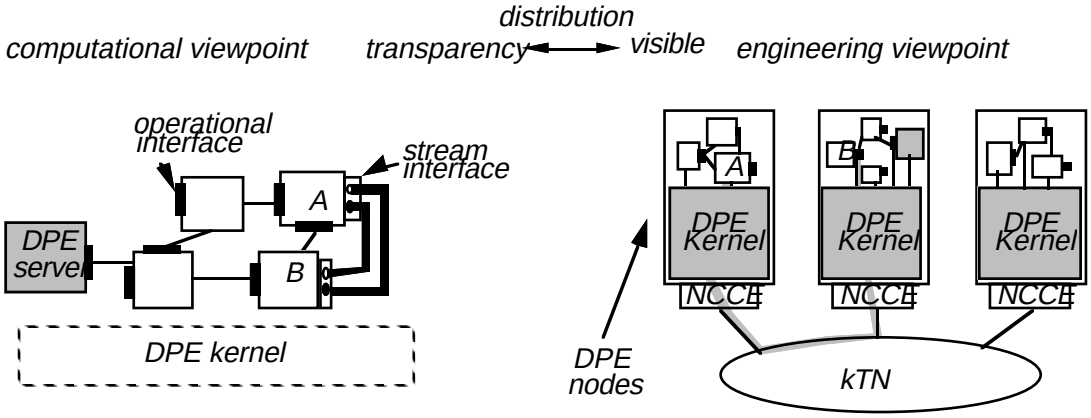

Figure 3.14: Computational and Engineering Viewpoints

The engineering viewpoint is concerned with the type of support required by the environment for realising QoS guarantees. Provision of QoS guarantees is intimately related to static and dynamic resource management (as described in section 2.3) of the different type of resource involved. Hence, the engineering viewpoint is interested in identifying the different resource managers involved in the provision of QoS, the resource domain under the control of each resource manager, and how the various resource managers interact and co-operate in the provision of end-to-end QoS.

While the TINA QoS Framework is still in the early stages of development, the approach taken is encouraging. It is important that the work on quality of service in the Computing Architecture is co-ordinated with on-going work in the rest of the TINA Architectures (for example, in collaboration with the Service Architecture initiative). Furthermore, the DPE nodes illustrated in figure 3.14 are interconnected to a *kernel*

Transport Network (kTN) [TINACb,95]. The quality of service provided by the TDPE infrastructure to the computational interactions implicitly relies on the service offered by the NCCE and the kTN combined. This requires strong coordination between TINA Computing and Network Architectures. In addition, quality of service management activities call for the co-ordination between the Management Architecture and all other TINA architectural components in turn.

3.3.9 The MASI End-to-End Architecture

The CESAME Project [Besse,94] at Laboratoire MASI, Université Pierre et Marie Curie, is developing an architecture for multimedia communications which has end-to-end QoS support as it primary objective. As with the Lancaster QoS-A, the MASI architecture 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 simple and efficient manner; ii) the need to resolve multimedia synchronisation needs of multiple related ODP streams [ODP,92]; and iii) the need to provide suitable communication protocol support for multimedia services.

The MASI architecture addresses the multi-layer, multi-service QoS problem in a comprehensive way. Concrete interfaces, mechanisms and services are defined [Fedaoui,94]. The architecture is comprised of a number of layers (which loosely follow the OSI reference model) and planes (which realise a number of QoS principles). As illustrated in figure 3.15, these layers include:

i) *application level,* which refers to an ODP platform which provides QoS conscious services to distributed multimedia applications; see [Besse,94] for full details of the QoS specification and support environments;

ii) *synchronisation layer,* which includes intra-flow synchronisation and inter-flow synchronisation between multiple related flows; and

iii) *communication level,* which subsumes the ATM, AAL and transport service and protocol; see [Fedaoui,94] for full details and related work at the University of Technology (UTS), Sydney [Fry,93].

MASI takes an object-oriented viewpoint based, in part, on the RM-ODP [ODP,92] approach for quality of service support of distributed multimedia applications. A number of functions (realised in planes) which span the multi-layered architecture are recognised as fundamental to resolving end-to-end resource management issues. These planes are comprised of:

- *QoS management*, which is the central arbitrator of end-to-end QoS, and is comprised of layer specific QoS managers that negotiate resources with peer QoS managers and maintain the internal state associated with application specific QoS;

- *connection management*, which manages multimedia session establishment based on a user supplied profile and which is made up of layer specific connection managers that bind multimedia processing units (MPUs) at each layer in order to meet end-to-end connectivity; and

- *resource management*, which is responsible for host operating systems and communication subsystem resource; this performs both admission testing and resource reservation at every level in the end-system.

The *Application QoS Manager (AQOSM)* translates application requests for multimedia flows to a set of QoS, services and protocol requirements. MASI QoS mapping is based on the concept of an application-level QoS profile. For each flow, the AQOSM derives suitable profiles. QoS profiles are used in the selection of protocol functionality and as a basis for determining flow specifications used by the communication subsystem. The AQOSM selects the desired values for performance parameters encapsulated in a flow specification. The AQOSM also selects appropriate communication and synchronisation protocol libraries based on the QoS profile template; see [Fedaoui,94] for full specification of the profile and the method of protocol selection. An important function of QoS management is to monitor layer specific QoS and report any QoS violations of the contracted profile directly to the applications. Other QoS violations fielded by QoS management include indications from the resource management plane during the negotiation phase. In this instance, the resource management function indicates the level of service provided to on-going flows, that is, either at the desired or minimum levels.

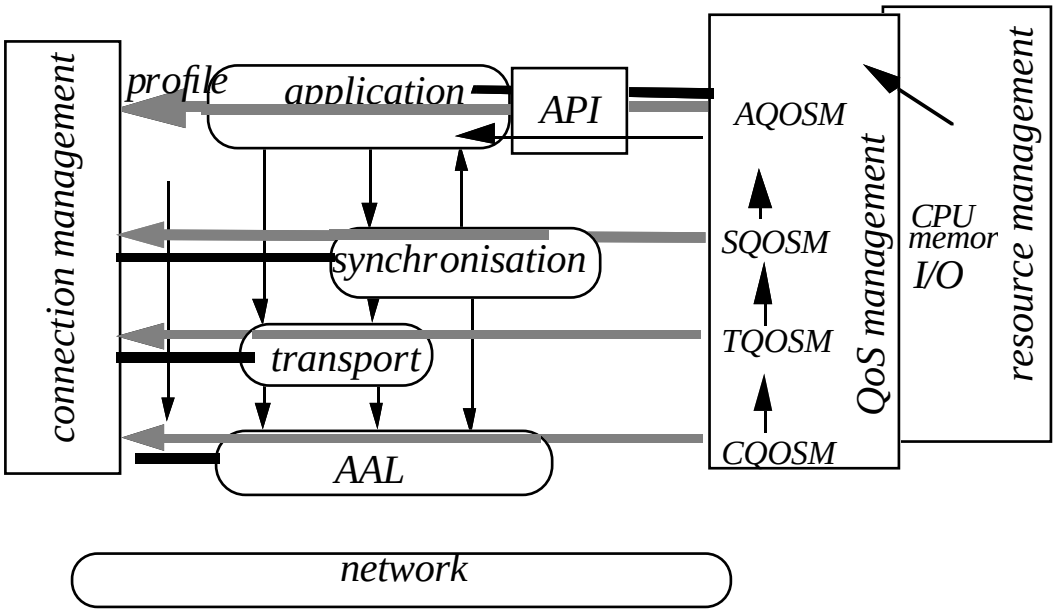

Figure 3.15: MASI Architecture

The MASI architecture focuses on end-system resource management: CPU scheduling, memory and I/O management. Network level admission testing and reservation will be addressed in future work. The CPU scheduling scheme adopted by the CESAME team is based on rate monotonic scheduling (RM) policy [Lehocky,89]. In this instance, the resource management plane actively measures the CPU usage and periodically informs the CPU scheduler of the utilisation. This is then used to accept or deny new flows in the end-system. A novel aspect of the MASI work is the use of application, system and communication libraries which are registered with known attributes in QoS *Management Information Base (MIB).* Users' QoS requirements captured in the QoS profile are used as a basis for the dynamic selection of the appropriate system functions. The selection of libraries is achieved at flow establishment time, QoS renegotiation time and is achieved dynamically by the resource manager to optimise or change the level of service available to a flow. The construction of profiles through re-usable software modules have been shown to be viable in the CESAME project [Besse,94].

3.3.10 Other QoS Frameworks

3.3.10.1 ATM Based QoS Models

Projects carried out at Indian Institute of Technology, Wollongong University, GMD FOKUS Berlin, Rutgers University and Columbia University have designed new QoS models which take application level QoS requirements and map them down to ATM based networks.

Native-Mode ATM Networking

The Indian Institute of Technology, Delhi and AT&T Laboratories, Murray Hill have developed a Native-Mode ATM Protocol Stack [Keshav,94] that allows the QoS provided at the ATM level visibility to applications programs. The service interface to the stack allows the establishment of simplex and duplex virtual circuits that feature error control, leaky bucket and feedback flow control. The framework is designed with four principles in mind: minimal functionality in the critical path, no logical layered multiplexing, separation of data and control flow and specific targeting to ATM. The native-mode ATM stack is a connection-oriented protocol stack which "bridges the gap between applications and the network". The protocol stack uses the Tenet Suite's RCAP protocol for network resource reservation and supports a set of orthogonal services which can be arbitrarily combined in order to match application requirements. Currently, the native-mode ATM protocols combine the services into three groups: i) guaranteed-performance service, which provides open-flow control, specification of performance guarantees and simplex and duplex multicast connections; ii) reliable service, which provides error control, feedback control, and simplex and duplex connections; and iii) best effort service, which provides only the choice of unicast or multicast service.

Simplified QoS Model

The Simplified QoS Model developed at GMD limits the number of performance parameters which the user can select. In doing so it simplifies the QoS mapping function and negotiation protocol. In [Damaskos,94] Damaskos and Gavras argue that applications can not be expected to configure large numbers of QoS parameters for flows.

Wollongong University QoS Model

Judge and Beadle [Judge,95] from Wollongong University describe a QoS model for an ATM capable end-system connected to a low speed ATM network (i.e., ranging 512 Kbps to 2.4 Kbps). The architecture supports two network traffic classes (guaranteed and best effort) for audio and data services. QoS mechanisms for monitoring are built on AAL protocols. Degradation in the requested QoS may be forwarded to the application level where compensatory actions may be taken.

GRAMS QoS Model

In [Hui,95] Hui, Zhang and Jun report on the progress of the development of the GRAMS architecture which is based on a client/server model for QoS control of flows over ATM networks. The major goal of GRAMS is to serve the heterogeneous QoS demands of clients exploiting the end-system and network utility. End-to-end resource management is based on a set of starvation counters used to measure system resource utilisation and individual flow QoS. These counters are integral to admission and rate control algorithms.

QuAL Based QoS Model

An important contribution by Florissi and Yemini [Florissi,94] from Columbia University describes the development of a *Quality Assurance Language (QuAL)* for the specification of QoS constraints on underlying computing and communication platforms. The specifications are compiled into run-time components that monitor the delivered QoS. Any QoS violations are fielded via user-level exception handlers. QuAL creates and manages a QoS-based MIB on a per-application basis for the management of flow statistics. The implementation work is based on Concert-C and ST-II networking. The network level QoS specifications are detailed for both senders and receivers. A distributed application is viewed by QuAL as a set of autonomous processes that communicate by message exchange (Concert-C). At the application level, QuAL uses a contract identifier to present a set of constraints to which a communication port must comply. Only ports with compatible QoS attributes are connected.

QoSMIC-TOMQAT QoS Models

Several projects in the European funded RACE program are concerned with QoS for integrated broadband networks. A contribution has been made by the *QOSMIC* (R.1082)

project which studied QoS concepts in broadband networks focusing on the user–network interface in particular. The major goal of the project was the specification of a QoS model for service life-cycle management which maps the user communication requirements to network performance parameters in a methodological manner. In related work Jung and Seret [100] propose a framework for the translation of the performance parameters between the ATM Adaptation Layer (AAL) and ATM layers. They extend the QOSMIC model to include QoS verification. In this case, the user can verify whether the achieved bearer QoS provided by the ATM network meets the contracted requirements expressed in terms of performance parameters. In related work, the *TOMQAT* project [TOMQAT,95] is developing the concept of total quality management in the context of broadband networks, analysing the end user quality of service requirements and designing QoS control and management mechanisms to meet end-to-end QoS guarantees.

3.3.10.2 Distributed Systems QoS Models

Recently a number of QoS models have emerged from the distributing systems community. These include the Integrated Multimedia Application Communication (IMAC) architecture [Nicolaou,93] and ANSA QoS Framework [Guangxing,94] which provide a model for real-time QoS.

IMAC-ANSA QoS Models

The IMAC architecture is based on the ANSA [APM,91] architecture and has been implemented as an extension to the ANSAware. IMAC provides a mechanism for specification of communication oriented QoS on per-invocation basis; interface operations may specify a set of QoS options. These QoS options are mapped to the underlying communications protocols as a set of QoS constraints on streams or bounded RPC communications. The work on the ANSA QoS Framework facilitates the enforcement of stringent time constraints found in distributed real-time applications. The model provides QoS specification and QoS-based binding for real-time programming in ANSA. The model, moreover, incorporates task and communication channels as its basic programming abstractions. It synthesises aspects of resource requirements, resource allocation and resource scheduling into an object-based programming paradigm.

Networld-Vnet QoS Model

The University of Pittsburgh is developing a distributed multimedia platform called *Networld* [Chrysanthis,95]. In the Networld, they are developing an architecture for integrating heterogeneous data repositories, service stations and viewing units. Within this distributed architecture they are investigating models for real-time workflow (i.e., data management and process scheduling), intermediate caching and resource allocation. Furthermore, all these aspects are studied within and across individual sites to establish multimedia delivery channels in WAN environments that support timing and reliability requirements of multimedia applications. The Networld architecture assumes the V-net as the network abstraction. The V-net is a versatile network abstraction that establishes flows with deterministic, statistical and best-effort characteristics.

3.3.10.3 Open Systems QoS Models

A number of projects are looking toward providing enhanced services and mechanisms for open systems; in particular the EuroBridge QoS-driven Architecture [Pronios,95] and QoS-based Adaptive Architecture for Packet Scheduling (Q-ADAPTS) [Tran,95]. Like the OSI QoS Framework, Q-ADAPTS and EuroBridge concentrate on quality of service for OSI communications. The EuroBridge research focuses on the upper layer architecture and the transport protocols and provides unifying concepts for the management of QoS. The ADAPTS model developed at California State University and the Aerospace Corporation supports dynamic management of flows through scheduling algorithms and resource reservation. The work is based on the integration of existing OSI-based protocols and real-time scheduling mechanisms in the end-system and network. The model addresses many limitations regarding QoS management which exist in the current OSI standards.

3.3.10.4 Transport System QoS Models

The Technical University of Berlin [Miloucheva,95b] is developing a QoS-based architecture with particular focus on support for the transport subsystem. The XTPX transport protocol, developed as part of the CIO (Coordination, Implementation and Operation of Multimedia Teleservices) Project, is at the heart of the architecture. The work considers QoS contracts, which are binding agreements between the application and transport provider; application classes, which flows are mapped into; and QoS management for the maintenance of flows. A multi-layer architecture is described which includes XTPX

operating over IP and ATM. Cross layer functions are used to map QoS between layers and for resource management; multimedia synchronisation is implied in the literature.

3.4 Summary

In this chapter key research in QoS support has been summarised and evaluated. Recent work directed at integrating and extending the layer-specific research into broader QoS architectures has also been presented.

The proposed QoS architectures promote the idea of integrated QoS, spanning the end-systems and the network and identifies the support for end-to-end QoS as an important goal. The review of the layer-specific QoS research indicates that much research has concentrated on applying QoS concepts described in Chapter 2 to either the network or the end-system in isolation. Emerging QoS architectures, in contrast, coherently apply QoS concepts across all architectural layers, resulting in a framework for the specification and implementation of end-to-end QoS.

In summary, while the area of QoS research in multimedia networking is mature, work on QoS architectures represented by the state of the art remain in the very early stages of development. Chapter 4 presents an outline of the author's contribution to end-to-end QoS research, an integrated QoS architecture for continuous media communications (QoS-A). In Chapter 8 the QoS-A will be compared to the QoS architectures presented above.

Chapter 4

Quality of Service Architecture (QoS-A)

This chapter proposes the development of an integrated quality of service architecture which spans both end-systems and networks to address the QoS deficiencies of communication architectures outlined in Chapters 1 and 2. The proposed quality of service architecture takes as its primary goal the support of performance QoS for a wide range of continuous media applications.

The QoS-A retains the best effort service model as a special case but is augmented by new classes of service that provide hard and soft end-to-end performance guarantees. These service classes are designed to fit into a highly dynamic application environment providing support for control, monitoring and maintenance of end-to-end QoS. The QoS-A offers a framework with which to specify and implement the required performance properties of continuous media applications operating over ATM networks.

In addition to the need for a richer service model which allows the QoS requirements of the new applications to be fully specified, the QoS-A requires the integration of a range of QoS mechanisms in both the end-system and network to meet end-to-end demands. In end-systems, this includes thread scheduling, flow shaping, buffer management and jitter correction. In the communications subsystem, protocol support such as end-to-end QoS negotiation, adaptation (i.e., QoS renegotiation) and indication of QoS degradation are required. In networks, suitable resource reservation protocols, service disciplines in switch queues and multicast QoS support are needed. The QoS-A provides a framework for management of QoS over all of these system layers.

This chapter describes the QoS-A primarily focusing on the transport layer. The chapter commences with an overview of the QoS-A model in section 4.1 and resource management issues in sections 4.2 and 4.3. Next, in section 4.4, a multimedia enhanced transport system based on the notion of a *service contract* agreed between the transport service user and the network provider is described. Section 4.5 describes the means by which quality of

service is contracted at the transport layer is realised in terms of suitable control, maintenance and management mechanisms. These QoS mechanisms are encapsulated in the *METS Protocol (METSP)* itself, a *Transport QoS Manager (TQM)* and a flow management module. It is shown how the QoS levels contracted at the transport level application programmers' interface can be assured in the context of the Lancaster ATM Research Networking Environment.

4.1 The QoS-A Model

The QoS-A is based on a subset of the QoS principles presented in Chapter 2. These principles govern the realisation of end-to-end QoS in a distributed systems environment:

i) the *integration principle* states that QoS must be configurable, predictable and maintainable over *all* architectural layers to meet end-to-end QoS [Campbell,93];

ii) the *separation principle* states that information transfer, control and management are functionally distinct activities in the architecture and operate on different time scales [Lazar,90];

iii) the *transparency principle* states that applications should be shielded from the complexity of handling QoS management [Campbell,92]; and

iv) the *performance principle* guides the division of functionality between architectural modules (viz. end-to-end argument [Saltzer,84], application layer framing and integrated layer processing [Clark,90], and the reduction of layered multiplexing [Tennenhouse,90]).

In functional terms, the QoS-A (see figure 4.1) 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 [Coulson,93]. Below the platform level is an *orchestration layer* which provides multimedia synchronisation services across multiple related application flows [Campbell,92]. Supporting this is a *transport layer* which contains a range of QoS configurable services and mechanisms. Below this, an internetworking layer and lower layers form the basis for end-to-end QoS support.

QoS management is realised in three vertical planes in the QoS-A. The *protocol plane*, which consists of distinct *user* and *control* sub-planes, is motivated by the principle of

separation. Separate protocol profiles are used for the control and data components of flows because of the different QoS requirements of control and data: control generally requires a low latency full duplex assured service whereas media flows generally require a range of non-assured, high throughput and low latency simplex services.

The *QoS maintenance plane* contains a number of layer specific QoS managers. These are each responsible for the fine grained monitoring and maintenance of their associated protocol entities. For example at the orchestration layer the orchestration QoS manager is interested in the tightness of synchronisation between multiple related flows. In contrast, the transport QoS manager is concerned with intra-flow QoS such as bandwidth, loss, jitter and delay. Based on flow monitoring information and a user supplied service contract, QoS managers maintain the level of QoS in the managed flow by means of fine grained resource tuning strategies.

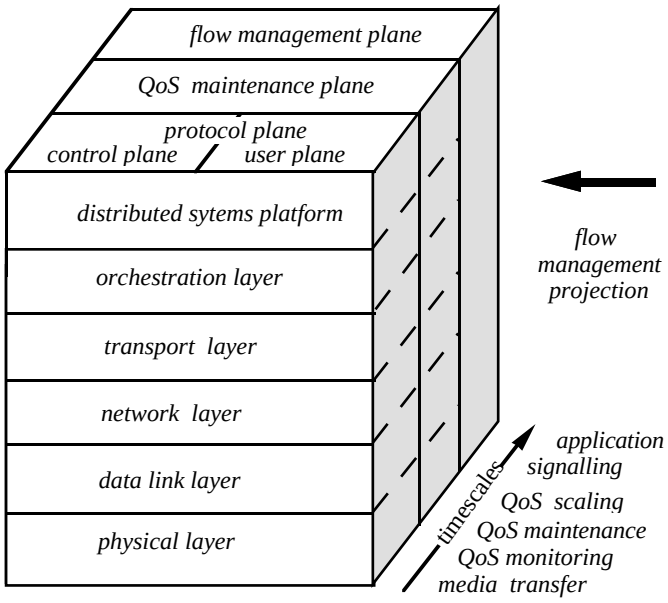

Figure 4.1: QoS-A

The final QoS-A plane pertains to *flow management*, which is responsible for *flow establishment* (including QoS based routing, end-to-end admission control and resource reservation), *QoS mapping* (which translates QoS representations between layers) and *QoS Scaling* (which collectively describes *QoS adaptation* and *QoS filtering* for coarse grained QoS management).

Figure 4.2 extracts a canonical set of QoS parameters to characterise continuous media communications and illustrate the relevant QoS mechanisms that realise each parameter. The chosen set of parameters are: jitter, delay, throughput and error QoS parameters. In addition to these fundamental parameters, two additional parameters are included: synchronisation between media streams and aspects of multicast quality of service. The essence of these latter two parameters is that they are applicable to multiple senders and receivers. The protocol plane QoS mechanisms are illustrated in figure 4.2 and include scheduling, jitter correction, rate regulation in the end-system and scheduling, flow control and QoS filtering in the network. Flow and QoS maintenance plane QoS mechanisms while not represented in figure 4.1 include mechanisms for resource reservation and QoS adaptation.

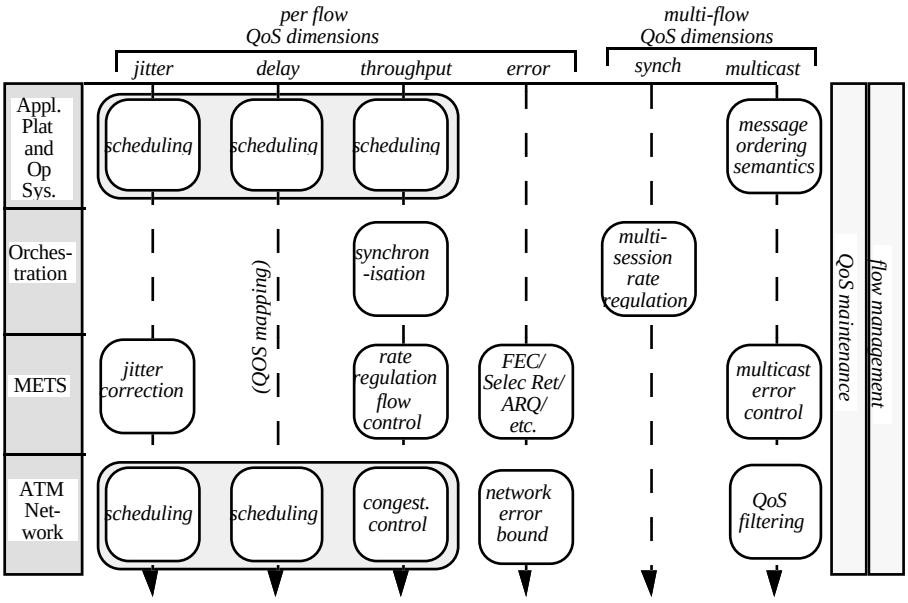

Figure 4.2: QoS Parameters and Supporting Mechanisms [Campbell,93a]

4.2 Resource Management Tree

The *resource management trees* presented in figure 4.3 illustrates the relationship between, on the one hand, layers, planes and QoS mechanisms and, on the other hand, resource management functions (e.g., static and dynamic functions). The resource management tree is subdivided into *Static QoS Management (SQM)* which includes mechanisms for signalling and flow establishment, and *Dynamic QoS Management (DQM)* which includes mechanisms for QoS management, maintenance and control.

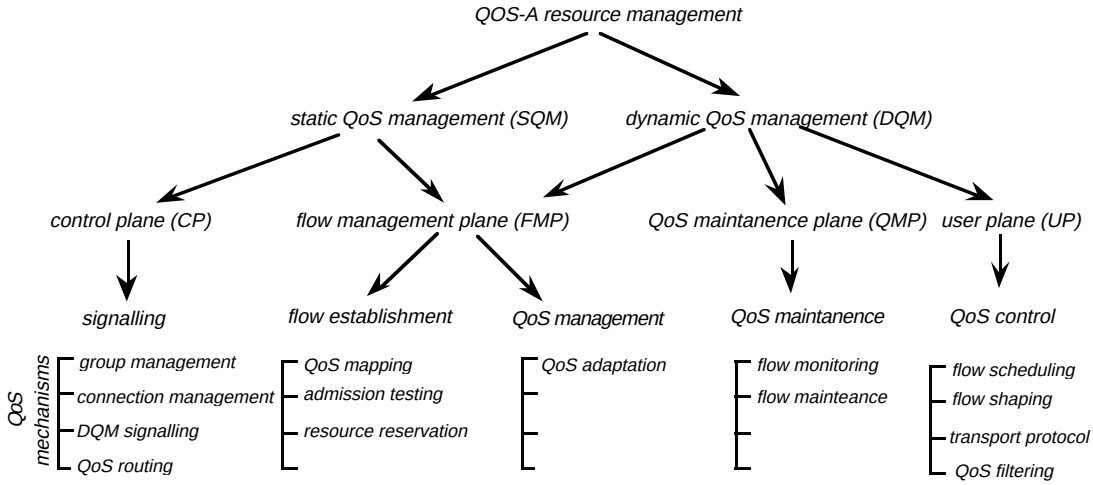

Figure 4.3: QoS-A Resource Management Tree

4.3 Timescales and Separation

QoS control mechanisms for media transfer operate on the fastest timescale in a QoS-A and include modules for flow scheduling, shaping, QoS filtering and METS [Campbell,93b] protocol control - all modules operate on the media directly and in real-time. Figure 4.1 broadly illustrates the relationship between timescales that are evident in a QoS-A model. The QoS maintenance plane includes flow monitoring and maintenance mechanisms which operate on a timescale close to the protocol timescale (i.e., at media transfer speeds). Flow management operates on a slower timescale than all other resource management entities in this model. It is responsible for the initial establishment and coarse gain QoS management of flows. The signalling QoS mechanism resides in the control plane and includes group management, connection management, and DQM signalling protocols. The division of QoS mechanisms into planes and layers is based on the principles of separation between control and data, and that these multiple timescales operating on communication resources.

The time constraints illustrated in figure 4.4 guide the division of functionality between the signalling, control and management modules. The control time domain is the fastest and operates close to the link speed of the network, signalling is at best one end-to-end or round trip away in time, flow management response time is coarser than the former in that it ranges from one round trip in time to seconds. It should be noted that flow management is

distinguished from the broader timescales experienced by traditional network management (e.g., SNMP or CMIP). QoS-A flow management lies in the time domain between signalling and network management and is considered to operate closer to the protocol and signalling speeds than traditional network management.

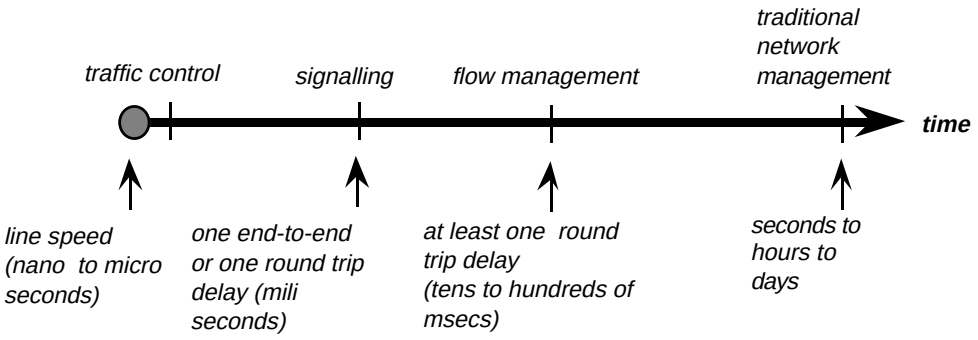

Figure 4.4: Timescales

4.4 QoS Specification

The success of ATM-based networks is dependent upon the availability of QoS configurable multimedia services with their strong emphasis on full end-to-end QoS guarantees. Currently these services and the supporting protocols are yet to be fully realised. A multimedia service is potentially composed of multiple flows which, in addition to QoS support, may require orchestration to meet multimedia synchronisation constraints and QoS scaling to meet multicast QoS demands.

An important aspect of a QoS enhanced communications service is the interface at which desired levels of QoS can be requested, negotiated and contracted. In the QoS-A, the QoS requirements of the user and the potential degree of service commitment of the provider are unified and formalised in a *service contract* agreed by both parties. Applications request the establishment of a continuous media flow with an agreed service contract via the following primitive:-

```
flow_id join( tsap_t *source, *sink; service_contract_t *QoS);
```

Flows can be source or receiver initiated. The call above shows the establishment of a

flow from a source transport service access point (*tsap*) to a destination tsap with a QoS as specified in the third argument, the *service contract*. This is described as peer-to-peer QoS. Note, however, that the multicast capable API described in Chapter 6 and implemented in Chapter 7 extends to multiple receivers joining a source flow having different QoS demands (i.e., all members having potentially different service contracts). Connectivity in this instance is achieved via a multicast addressing and signalling scheme based on Ballardie and Crowcroft's concepts for Core Base Trees multicasting [Ballardie,94].

The service contract subsumes the well accepted QoS parameters of jitter, loss, delay and throughput, but also allows the specification of a wider range of options. These are characterised in terms of the following clauses:-

- `flow_spec_t` characterises the user's traffic performance requirements [Partridge,92];
- `commitment_t` specifies the degree of resource commitment required from the lower layers;
- `adaptation_t` identifies actions to be taken in the event of violations to the contracted service;
- `maintenance_t` selects the degree of monitoring and active QoS maintenance required;
- `reservation_t` either negotiated, fast reservation or forward reservation connection services;
- `cost_t` specifies the costs the user is willing to incur for the services requested.

In implementation, the clauses are collected into a C structure as follows:-

```
typedef struct {
   flow_spec_t       flow_spec;
   commitment_t      commitment;
  adaptation_t       adaptation;
  maintenance_t      maintenance;
  reservation_t      service;
  cost_t       cost;
} service_contract_t;
```

The following sections motivate and describe the QoS options specified in this structure.

4.4.1 Flow Specification

The ability to guarantee traffic throughput rates, delay, jitter and loss rates is particularly important in networks supporting distributed multimedia applications. These performance-based metrics are likely to vary from one application to another. Moreover, the relative importance of these parameters for a particular flow is also application dependent. For

example, a digital voice connection requires a moderate throughput (e.g., 32 Kbps), a low degree of reliability (10^{-1}), a stringent upper bound on end-to-end delay (e.g., 250 ms) and a maximum permissible jitter of 10 ms.

To be able to commit transport and network resources, the QoS-A must have prior knowledge of the expected traffic characteristics associated with each flow. The flow_spec structure below permits the user to specify such metrics. In the flow spec, throughput is described in terms of frame size, frame rate, burst size and peak rate. The frame size and frame rate represent the average throughput requirement, whilst the maximal throughput is captured by the peak rate performance parameter. In addition, the flow spec accommodates the potential burstiness of the offered traffic using the burst parameter.

```
typedef struct {
   int   flow_id;    /* flow specification identification */
   int   media_type;       /* common flows for video, voice, data */
   int   frame_size;       /* frame/tsdu size */
   int   frame_rate;       /* token generation rate */
   int   burst;      /* size of the burst */
   int   peak_rate;  /* max transmission rate */
   int   delay;      /* end-to-end delay */
   int   loss;       /* loss rate */
   int   interval;   /* interval */
   int   jitter;     /* end-to-end delay variation */
} flow_spec_t;
```

The precise interpretation of the performance parameters (i.e., throughput, delay, jitter and loss) is determined by the commitment specification as described below. The flow id field, which is allocated by the QoS-A and returned to the user for subsequent use, uniquely represents the flow in the system. The media type field is used by the upper layer architecture to specify commonly used flows with pre-specified flow specifications such as StandardVideo, HifiAudio and LowQoSVoice.

4.4.2 QoS Commitment

While the flow spec permits the user to express the required performance parameters in a quantitative manner, the commitment clause allows these requirements to be refined in a qualitative way so as to allow a distinction to be made between hard and soft network performance guarantees. There are broadly *three* classes of service commitment the network

can support [Ferrari,92]:-

i) *guaranteed*, which is typically used for hard real-time performance applications;
ii) *statistical*, which allows for a certain percentage of violations in the requested flow spec, and is particularly suitable for continuous media applications; and
iii) *best effort*, the lowest priority commitment and synonymous with a datagram service.

The format of the commitment_t structure is illustrated below.

```
typedef enum { guaranteed, statistical, best_effort } commit_t;

typedef struct {
   commit_t    class;      /* commitment class */
   int         percentage; /* used for statistical service */
} class_t
```

Many continuous media applications require soft real-time guarantees as selected by the statistical service commitment. A statistical commitment allows for a certain percentage of violations of each QoS performance parameter. Taking loss as an example: an uncompressed digital video flow may suffer 50% loss (commitment.loss.class = statistical, commitment.loss.percentage = 50) and still reconstruct enough of the video signal to maintain acceptable playout picture quality. In the case of statistical commitment, the performance parameter values in the flow spec are interpreted as a target for the QoS-A which, however, may be violated if resources become scarce.

An important distinction between the guaranteed and statistical commitments is that the guaranteed commitment is based on fixed resource allocation where no resource gain is feasible; in contrast, the statistical commitment is based on shared resource allocation which encourages a high degree of resource utilisation [Campbell,93a]. It is for this reason that future pricing policy should encourage users to select statistical commitment over the guaranteed commitment when at all possible.

Together the flow spec and the commitment class are used by the flow reservation and admission control functions of the flow management plane to establish end-to-end QoS. In a QoS-A, resource reservation and admission control are conducted at each layer of the architecture. This means there is a set of admission tests for the end systems (for cpu, memory and network access) and the network (for bandwidth, delay, memory); for full details of these admission control see Chapter 5.

4.4.3 QoS Adaptation

Many continuous media applications can tolerate small variations in the QoS delivered by the network without any major disruption to the user's perceived service. In some cases quite severe service fluctuations can be accommodated. In such cases, however, it is often appropriate to inform the application of the service degradation so that it can intelligently scale to the new QoS baseline. If the delivered performance violates the contracted QoS then the user may choose to take some remedial action (i.e. adjust its internal state to accommodate the current load conditions, re-negotiate the flow's QoS, drop components of a multi-layer coded flow - e.g., drop MPEG enhancements, disconnect from the service or take no action).

To meet these requirements, the adaptation_t structure is used :-

```
typedef enum { loss, jitter, throughput, delay, disconnect } event_t;

typedef enum { adapt, signal, disconnect, no_action } action_t;
typedef struct {
  event_t   event;    /* QoS degradation */
  action_t  action1;   /* action  */
  action_t  action2;   /* auxiliary action */
  flow_spec_t     *new_flow;   /* new FlowSpec for QoS scaling */
} adapt_t;
```

The user can select up to two actions to be taken in response to each event. As an example of the use of the adaptation facility, consider the following:-

```
adapt_t action_list = {{delay, signal, no_action, 0},
            {throughput, signal, adapt, &newFlowSpec},
            {loss, no_action, no_action, 0},
            {jitter, signal, no_action, 0}};
```

The user is informed of QoS degradations in one of the following ways: *(i)* via a *QoS signal* (i.e., a QoS degradation indication): this is an upcall from the lower layers which notifies that one or more performance parameters in the flow_spec_t or commitment_t has been violated; *(ii)* via a disconnect indication, the QoS-A unilaterally initiates a disconnect; and *(iii)* via a QoS renegotiation indication: this is issued when the user has delegated the responsibility for QoS adapt (i.e., re-negotiation) to the QoS-A and the QoS-A has just

initiated a re-negotiation. Note that any combination of actions *i), ii)* and iii*)* can occur for any one violation.

To complete the example above, the following are the actions taken in response to the various possible events. If the maximum end-to-end delay is exceeded then the QoS-A will inform the user of the QoS event signal(event, measured_value, desired_value) upcall. The second action/event pair deals with degradation of measured throughput. If the throughput falls below the predefined minimum specified in the flow spec the QoS-A will initially inform the user of the event via a QoS signal and an adaptation indication (i.e., QoS renegotiation indication), then will initiate a full end-to-end re-negotiation based on the newFlowSpec, and finally issue a QoS renegotiation confirm to the user to inform him of the outcome of the adaptation phase.

4.4.4 QoS Maintenance

The QoS maintenance function operates directly on on-going flows to maintain the performance QoS requested in the flow specification and QoS commitment clauses, respectively. Three options are available within the service contract for control over QoS maintenance :-

```
typedef enum {signal, monitor, maintain } maintenance_t;
```

The *signal* and *monitor* option instructs the QoS-A to periodically deliver measured performance assessments (called QoS signals) relating to the specified flow to the application at the flow management interface which can be local (sender or receiver) or remote. The flow management interface is specified as being:

```
typedef enum {source, receiver, remote} monitor_t;
```

The *source* and *receiver* options default to the local flow management interfaces indicated as the "F" interface in 4.5. If the remote monitoring option is selected then QoS assessment messages are forwarded to a remote flow management entity for processing. This remote monitoring is particularly important for multicast QoS management and filter management [Yeadon,93]. Periodic performance notifications include measured bandwidth, delay, jitter and losses for on-going flows as measured at the source or receiver. The signal

fields of the QoS policy (described in Chapter 6) allow the user to specify the interval over which one or more of these QoS parameters can be monitored and the user informed. Periodic assessment messages are not forwarded to the application if the *maintain* mode is selected by the user.

In both monitor and maintain modes the QoS maintenance attempts to *transparently* exert fine grained corrective action (e.g., thread scheduling [Coulson,93], communication buffering, flow regulation and scheduling, queueing delays, jitter correction, playout time calculation, etc.) to maintain QoS levels according to the service contract. In both cases, coarse corrective action (i.e., QoS adaptation), should the contracted QoS drop below the prescribed levels, may be taken depending on the selected adaptation_t option.

Finally, the signal mode explicitly disables the active maintenance of flows. In this role the maintenance plane only forwards periodic signals to the application at the 'F' interface. It is the responsibility of the application to act on this information as it sees appropriate. Essentially the application is responsible for the maintenance role in this mode and can only exert coarse QoS management by issuing new service contracts should it detect violation in the requested QoS.

4.4.5 Reservation Styles

Reservations protocols can support full end-to-end *negotiated* service and in some cases, a *fast* reservation service where the reservation and data transfer phases coincide. In addition to these reservation styles, the QoS-A also offers a *forward reservation* mode where network and end-system resources are booked ahead of time; here the user specifies the expected starting time and duration of a flow. This service is useful to multimedia applications that require a high degree of QoS availability such as collaborative sessions. Three connection_t classes are defined in a fully generalised service contract to accommodate the connection styles described above:-

```
typedef enum {fast, negotiated, forward} service_t;
typedef struct {
   service_t   service;     /* fast, negotiated, forward service */
  time_t       start;    /* start of service hrs:min:sec */
  time_t   end;     /* termination time hrs:min:sec */
} connection_t;
```

The connection_t includes a start and end time for the forward reservation service. However, it is clearly difficult to determine the duration of interactive communication sessions, and therefore one remains somewhat sceptical about enforcing such a regime upon the user. The duration time is included in the service as a marker for further study. In [Ferrari,92] an advance reservation mechanism is described whereby the network may provide the user with an extension after the specified duration has expired. This is achieved without disruption to other users who have pre-reserved resources.

4.4.6 Cost

The thesis will not address the issues of cost and tariffing in detail, it is mainly concerned with a realisation of the architecture in a local ATM environment were no tarrifing apparatus exist. However, even in a local environment, cost is still likely to be an important factor. If the notion of cost is not involved, there is no reason for the user to select anything other than maximum levels of service commitment! This philosophy would inevitably lead to resource inefficiencies in a QoS-A. To counter this condition, the cost function must incorporate pricing differentials [Chocchi,91] to encourage the user selects the optimum QoS commitment such as a lower-commitment-costs-less pricing policy.

In contrast to a cost based policy, Kelly [Kelly,93] describes a simple tariff structure based on the user predicting a connection's effective bandwidth requirement. The tariff structure encourages the user to accurately declare this quantity over the duration of the connection. The incentive to accurately predict the effective bandwidth requirement is realised by admitting more flows in addition to optimising the network resources. The penalty for over-estimation of the connection's bandwidth is a possible reduction in QoS provided.

4.5 Multimedia Enhanced Transport System (METS)

In this section suitable QoS mechanisms needed to realise the transport service interface described above are presented. The mechanisms are embedded in the QoS-A flow management projection as shown in figure 4.1. The flow management projection is shown in greater detail in figure 4.5. Note that figure 4.5 only shows the transport layer and below; the upper layers have been omitted for clarity. This section details the composition and interaction between the various planes at the transport layer: flow management, QoS

maintenance, user and control planes.

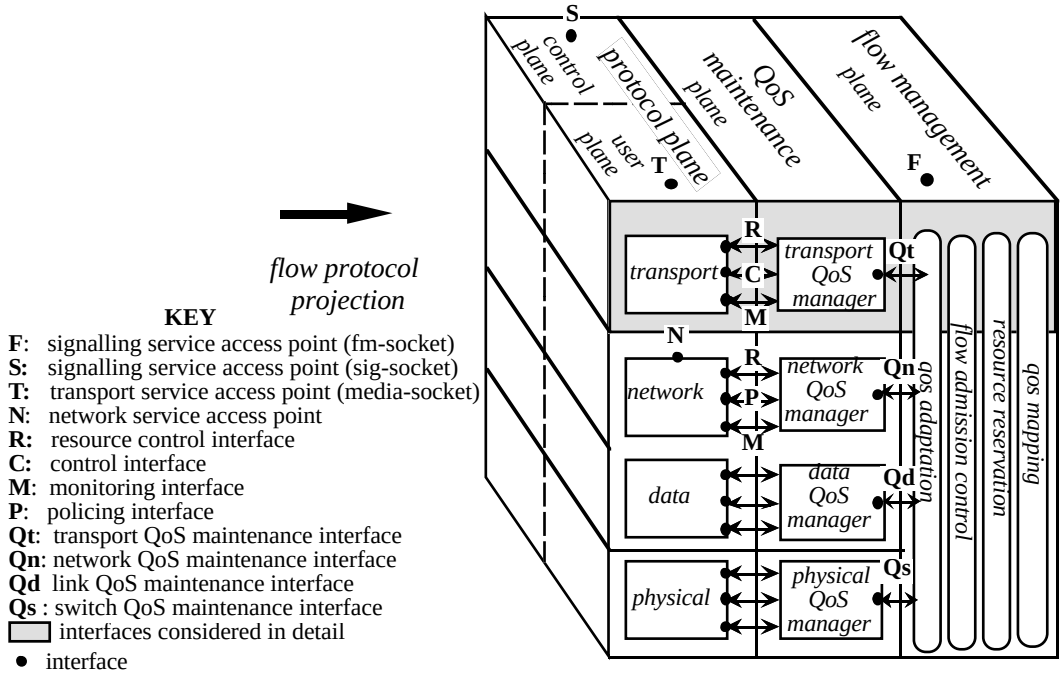

Figure 4.5: QoS-A Flow Management Projection

4.5.1 QoS Interfaces

At each layer, the various mechanisms in each plane present well defined interfaces to their peers. At the transport layer the support of QoS is dependent on interactions between the transport protocol, transport QoS manager, the flow management plane (viz. QoS adaptation, resource reservation and admission control, QoS mapping) and the network layer. In contrast to traditional communications architectures, the QoS-A carries all flow management and control messages on distinct out-of-band signalling channels. To reflect this logical division, the transport service access point is internally divided into flow management *(fm-socket)* /signalling interface *(sig-socket)* and media transfer interface *(media-socket)* components (there are also equivalent primitives at the network layer) as illustrated in figure 4.5:-

- the fm-socket ('F') interface contains primitives: openGroup, getInfo, closeGroup, joinFlow, leaveFlow, changeQoS, signalQoS;
- the media-socket ('T') interface contains primitives: send, receive.

Later sections describe the use of the various primitives on these interfaces in more

detail. In addition to the transport user's interface, the QoS-A defines the following internal interfaces between the three planes at the transport layer and below as illustrated in Figure 4.1 and Figure 4.5:-

- a *resource* control interface used to allocate, tune and release transport protocol resources, and alert the QoS management plane if protocol resources are short. It contains the following primitives: alloc_resource, tune_resource, resource_alert, and free_resource;
- a *monitor* interface used by the transport QoS manager to configure and control monitoring of flows in the transport protocol and to receive reports of actually achieved QoS performance over a preceding interval. It contains the following primitives: start_monitor, set_rate, qos_assessment, and stop_monitor;
- a *control* interface used by the QoS manager to set, modify and read internal transport protocol states during flow connection, data transfer and re-negotiation. The interface contains the following primitives: set_state and report_state;
- a *maintenance* interface that is supported by the transport QoS manager and used by the flow management plane. It contains the following primitives: start_maintenance, set_attributes, free_attributes, assessment, qos_alert and stop_maintenance;
- a *signalling* interface ('S') that is supported by the control plane signalling protocol. It contains the following primitives: openGroup, getInfo, closeGroup, joinFlow, leaveFlow, changeQoS.

4.5.2 User Plane

The METS transport protocol is based on a *Continuous Media Transport and Orchestration Service (CMTOS)* developed at Lancaster [García,93] which evolved from the work by Chesson on the XTP protocol [Chesson,92]. The CMTOS protocol provides an ordered but non-assured, connection oriented transport communication service. It allows the user to select upcalls for the notification of corrupt and lost data at the receiver and also allows the user to negotiate QoS parameters such as bandwidth, jitter, delay and the tightness of synchronisation between multiple related connections. The CMTOS protocol does not, however, provide any form of QoS commitment other than best effort QoS, and as such it is unsuitable to meet end-to-end QoS assurances.

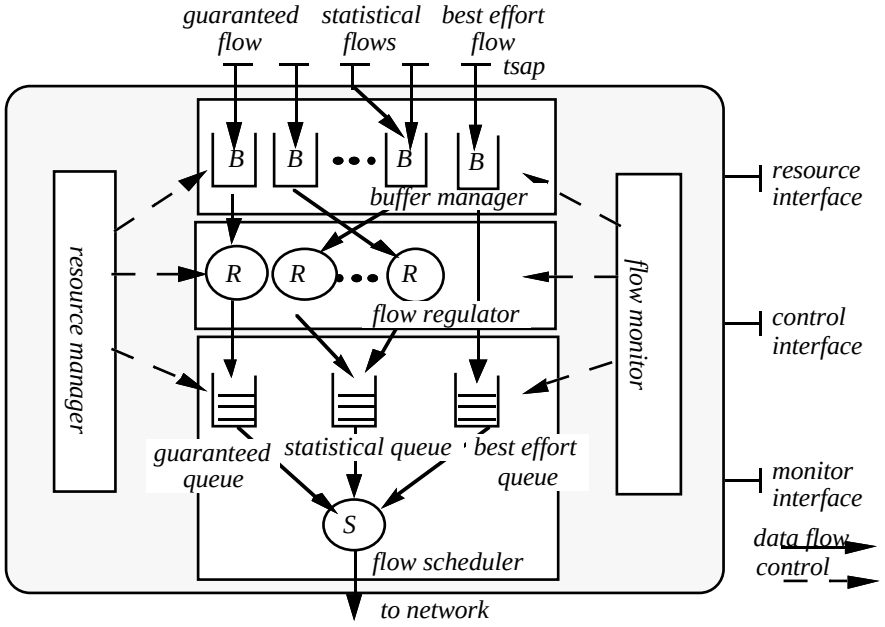

Figure 4.6: METS Protocol Mechanisms and Interfaces

It is the responsibility of the protocol to share communications resources in end-systems among flows with widely diverse QoS requirements. To meet this need, the METS protocol incorporates buffer sharing, rate regulation, scheduling, and basic flow monitoring QoS modules as illustrated in figure 4.6. Also included is a resource management component responsible for overseeing the allocation and adaptation of the various protocol resources. The buffer management scheme is structured to avoid duplication across layers [Hehmann,91] and uses separate pools for each commitment type [Robin,94]. Guaranteed flows each receive a fixed buffer allocation based on the flows peak rate whereas statistical flows share a common pool. For guaranteed and statistical flows the loss field in the Flow Spec is also taken into consideration when determining the buffering needs at the end-system and in the network. Best effort flows also use the common pool but are given a lower priority than any statistical flow. The remaining transport protocol modules, rate regulation, scheduling, flow monitoring and resource management are described in more detail below (refer to figure 4.6).

4.5.2.1 Flow Regulator

The transmission of frames to the network must be regulated to prevent buffer overflow at receivers and rate violations (via flow shaping) at the user-network interface and

intermediate nodes. In the QoS-A, the throughput rate of a continuous media source is characterised by the performance parameters and the service commitment clause. The regulator is configured, at flow establishment or QoS adaptation (i.e., re-negotiation) time, to shape the flow in accordance with this characterisation by assigning an *eligibility time* to each frame segment to be transmitted. Only guaranteed and statistical flows are given eligibility times; best effort flows are sent at a rate determined by the workload of the scheduler (see below). Figure 4.6 shows that a flow can be viewed as a stream of media taking a specific path through the sequence of buffers, regulators and scheduling queues.

4.5.2.2 Flow Scheduler

Once a flow has been shaped, the scheduler arranges for the transmission of frames in accordance with a pre-determined end system delay allocation. End system delays are allocated at flow establishment time when each intermediate switch commits to meeting a particular fraction of the permitted end-to-end delay [Anderson,91] [Robin,94]. By limiting the number of guaranteed and statistical flows (as part of flow admission control schedulability test) one ensures that each deadline will be met and each frame delay bounded. No such test is required for the best effort service queue.

Scheduling at the transport layer is based on an hierarchical deadline scheduling. Three scheduler communication service queues are used [Ferrari,92]: one each for guaranteed, statistical and best effort flows. The guaranteed and statistical queues are sorted by deadline where deadline is calculated as eligibility time plus the delay component. The scheduler queues are serviced in strict order of priority which is given first to the guaranteed queue, second to the statistical queue and lastly to the best effort queue. The scheduler services frames from guaranteed and statistical queues using a non pre-emptible discipline.

4.5.2.3 Flow Monitor

In addition to the above described QoS mechanisms, the protocol includes a flow monitoring mechanism which has been designed for point-to-point and multicast flow environments. For multicast communications different receivers are likely to have heterogeneous service contracts. It is also likely that the source and receivers will all have disparate flow specifications. This is illustrated in figure 4.7 which shows a source (i.e., atc) and a number receiving end-systems (viz. mr-little, dwp, njy) interconnected via three

ATM switches (viz. chuff, sparrow, rook). A multicast switched virtual circuit is established as indicated which carries a flow from atc to njy, dwp and mr-little, respectively. Each receiver has differing QoS demands on the flow as represented by service contracts (viz. QoS_0, QoS_1, QoS_2, QoS_3). In Chapter 6 describes how heterogeneous QoS demands are resolved in a QoS-A based network. At present, though, the need for multicast based QoS monitoring will be highlighted.

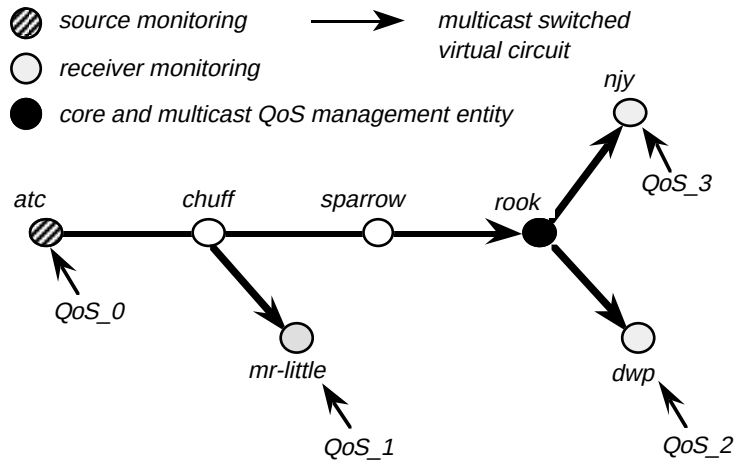

Figure 4.7: QoS Monitoring

Flow monitoring is a core mechanism in the QoS-A and provides the basis for a number QoS feedback loops that are fundamental to the operation of dynamic QoS management. The transport protocol flow monitoring component gathers statistical information regarding the ongoing flow of media, both at the source and the sink of the transport flow. This information is used by the transport QoS manager during the on-going QoS maintenance of flows. Based on a flow's measured performance the transport QoS manager and transport protocol interact over the resource, control and monitor interfaces (described in section 4.5.1 and illustrated in figure 4.1 and figure 4.5) to maintain the flow's QoS.

The flow monitor is configured to operate in one of two possible flow monitoring modes on a per-flow basis. These are:

i) source monitoring, which periodically monitors the flow, records its transmission statistics as media is injected into the network; and

ii) receiver monitoring, which periodically monitors the flow and records the reception details as media is received from the network.

In both cases, the resulting QoS assessment messages are forwarded to the transport

QoS manager for processing. Flow monitor measurements are made over a predefined per-flow measurement interval called an *era* [Nahrstedt,93] which is, by default, the reciprocal of the frame rate specified in the flow spec (i.e., for a frame_rate of 25 frames per second the interval is 40ms). In essence, the transport protocol *monitors* a flow's on-going performance and the transport QoS manager *maintains* it. In this sense, flow monitoring is a passive mechanism and maintenance an active one.

Flow monitoring is initiated when a start_monitor command (indicating the maint mode) is received by the transport protocol from the transport QoS manager at the protocol's monitor interface for a particular flow-id. When monitoring is enabled, the monitor operates in one of the two defined modes; the mode is implicitly based on the direction of the flow.

4.5.2.4 Resource Manager

The resource interface provides access to the protocol's buffer management, regulation and scheduling modules which are used both during the flow establishment time and QoS adaptation time. During flow establishment the various mechanisms are configured in accordance with the flow spec and commitment clauses in the service contract. Figure 4.8 presents the performance parameters together with the mechanisms required for their support and the resource configuration required for the three types of service commitment.

Guaranteed flows achieve guaranteed QoS by reserving dedicated communications buffers (based on peak rate allocation) and using non pre-emptible deadline scheduling and admission control. Statistical flows achieve a higher degree of resource utilisation by using a flexible resource allocation policy whereby pools of communication buffers are shared based on average rate allocation. Statistical flows also use deadline scheduling and admission control, and can be pre-empted by guaranteed flows. In this case statistical flows can potentially miss a percentage of their deadlines and still achieve the desired QoS (missed deadlines are bounded by an admission test for statistical flows [Robin,94]). Best effort traffic receives no resource or service commitment in the QoS-A; however, if resources (buffers, scheduler, etc.) are available and not currently in use by statistical flows they can be borrowed by best effort traffic. These borrowed resources, however, can be reclaimed at any point, making the resources available to a best effort service pre-emptible.

QoS commitment	

QoS parameter	QoS mechanism	guaranteed	statistical	best effort
loss	buffer management	fixed buffer allocation based on the peak rate	shared buffers based on average rate	no guaranteed buffer allocation
throughput	regulation (flow shaping)	eligibility time based on peak rate	eligibility time based on average rate	no regulation resources committed
delay and jitter	scheduling	flow always scheduled at eligibility time	flow scheduled at eligibility time resources permitting	flow scheduled if scheduler idle

Figure 4.8: Commitment Class Based Resource Reservation

4.5.3 Control Plane

The *METS signalling (METSig)* protocol module illustrated in figure 4.1 is responsible for the management of groups, the establishment of point-to-point and multicast connections, and signaling support for dynamic QoS management required by the flow management plane. Applications interact with the flow management plane over a flow management interfaced (the 'fm-socket' as illustrated in figure 4.5). Furthermore, applications issue group management (viz. openGroup, getInfo, closeGroup), connection management (viz. joinFlow, leaveFlow) and dynamic QoS management (viz. changeQoS) primitives on this interface. The flow management and QoS maintenance planes interact with METS signalling protocol modules over the control planes' control interface (i.e., 'S' interface) for the establishment, QoS management and tear down of multicast flows.

4.5.3.1 Meta-Signalling Protocol

METS signalling messages are conveyed on a dedicated connection called the *METS meta-signaling* channel as illustrated in figure 4.9. Network level QoS-A signalling modules resident in switches and end-systems interpret meta-signalling messages and forward them to the appropriate signalling module for processing. The METS meta-

signalling base service provides a low latency full duplex assured service for the transmission of group, connection and dynamic QoS management primitives and is implemented by the meta-signalling protocol.

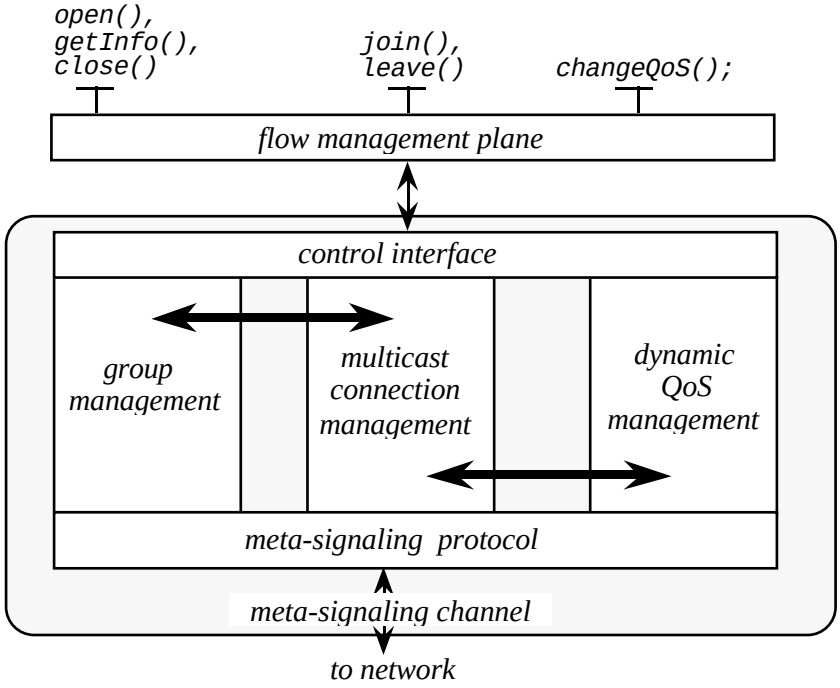

Figure 4.9: METS Signaling (METSig)

4.5.3.2 Group Management

METS group management allows the user to create a multicast group. Each multicast group is allocated a multicast group address. Group management advertises the service contract of the source QoS. Receivers interrogate *QoS groups* to determine the traffic characteristics of the offered media and select the components of the flow which meet their QoS capabilities. Group management primitives include open group, get group information and close group. For full details on group management and QoS groups see Chapters 6 and 7. When a multicast group has been created, via METS group management signalling, users are free to establish a connection to the flow using METS connection management signalling.

4.5.3.3 Connection Management

METS connection management is based on Core Based Tree [Ballardie,94] multicasting signalling where senders and receivers join and leave multicast flows in an independent manner. Members of a multicast group rendezvous at special network nodes called *cores* as illustrated in figure 4.10. The issue of de-coupling members of groups in this manner is an important consideration when dealing with scaling in networks. When a group is created a core is designated. When a core has been instantiated in this manner members of the group can then issue join commands using an allocated multicast address. This address is based on the address of the core in QoS-A connection management. Join messages are transmitted along the meta-signalling channel and include the source or receiver's QoS requirements for the flow specification and QoS commitment.

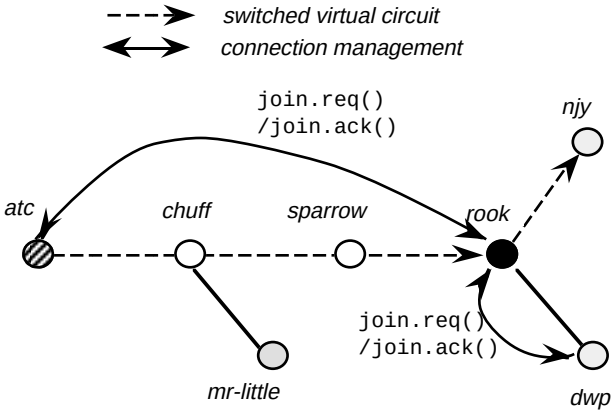

***Figure 4.10** : METS Connection Management Signalling*

4.5.3.4 Dynamic QoS Management Signalling

When a source and at least one receiver (this is derivative of multicast, i.e., a point-to-point connection) have successfully established connections media transfer can begin. In this mode source nodes send media toward the core on the designated multicast address and the core base tree disseminates the media to all receivers at the QoS level designated by each receiver oriented join command. During the media transfer phase any member of the group can renegotiate their QoS. In this case flow management signalling supports the QoS adaptation protocol for renegotiation of QoS.

4.6 QoS Maintenance Plane

The transport protocol and transport QoS manager are tightly coupled to operate in the same time domain. This is crucial for fine QoS adjustment as the QoS maintenance plane must be able to detect and react to real-time performance fluctuations which may occur in the protocol time domain.

According to the three maintenance policies available (see section 4.4.4), the transport QoS manager operates as follows. For all options the transport QoS manager receives periodic QoS assessment messages generated by the transport protocol.

Based on the mode of operation QoS maintenance takes the following actions:

i) the signal policy dispatches these messages on to the flow management plane and no other action is taken by QoS maintenance;

ii) the monitor mode dispatches the messages and attempts to actively uphold the end-to-end performance parameters via a *monitor-measure-adjust* loop based on whether the requesting node is a source or receiver;

iii) the maintain mode does not forward the QoS assessment messages but attempts to actively uphold the end-to-end performance parameters via a monitor-measure-adjust loop based on whether the requesting node is a source or receiver.

If the requesting node is a source node the transport QoS manager attempts to actively uphold the end-to-end QoS by making fine adjustments via the tune_resource primitive on the transport protocol's *resource* interface. Fine resource adjustment counters QoS degradation by adjusting loss via the buffer manager, queueing delays via the flow scheduler and throughput via the flow regulator. Within contracts, if the requesting node is a receiver node the transport QoS manager attempts to compensate for: i) jitter correction by adjusting playout delays via the flow scheduler and throughput via the flow regulator and ii) loss by signalling any lost packet to the higher layer error control modules.

If the transport QoS manager detects that any of the QoS parameters have degraded below their desired levels then a qos_alert message is sent to the flow management plane which takes the appropriate action based on the adaptation clause.

In addition to its role in supporting the various maintenance policies described above, the QoS manager is also responsible for implicitly maintaining the commitment clause in statistical flows by means of the same *monitor-measure-adjust* loop. Note that it is not necessary to actively maintain guaranteed flows or best effort flow. The former have

resources exclusively dedicated to them and the latter are not maintained by definition. However, service fluctuations are anticipated from time to time in statistical flows because they share transport and network communication resources.

The information passed periodically for QoS assessment to the receiver, sender or remote adaptation handler is as follows:-

```
typedef struct {
   int   flow_id;          /* flow-id */
  int    mode;             /* source or receiver */
   int   originator;       /* originator address */
   int   mcast;            /* multicast group address */
   int   frame_seqno;      /* Seqno of frame which generated msg */
   int   time_stamp;       /* timestamp */
   int   interval;         /* measurement interval */
   int   measured_delay;   /* QoS assessment begins */
   int   measured_jitter;
   int   measured_loss;
   int   measured_rate;
   int   desired_delay;
   int   desired_jitter;
   int   deired_loss;
   int   desired_rate;
} qos_assess_msg;
```

Multicast QoS managers are able to develop a statistical representation of the end-to-end QoS for each member of the group utilising the performance data supplied within QoS assessment messages. The qos_assessment message is partitioned into measured and desired QoS statistics; these include delay, jitter, throughput and loss rate measured during the era.

4.7 Flow Management Plane

The flow management plane is responsible for a number of static and dynamic QoS management functions as depicted in the resource management tree in figure 4.3. The major functions, described below, consist of the provision of network signalling infrastructure, resource reservation and QoS adaptation for the realisation of coarse grained dynamic QoS management as specified in the flow spec's adaptation clause. The flow management plane also performs other management functions such as the mapping of QoS representation between layers.

4.7.1 Flow Reservation

A major part in flow establishment algorithm is reservation in the end-systems and in the network - according to the user specified service contract. Resource reservation allocates resources in accordance with the QoS commitment specified in the service contract. For a guaranteed service all resources are allocated based on the peak rate. For the statistical service resources are allocated based on the sustained rate. No resources are allocated for best effort commitment. Flow reservation interacts with QoS mapping to determine the QoS parameterisation for a particular layer. Then it interacts with admission control to determine whether sufficient resources are available to accommodate a new flow. The details of QoS mapping and admission testing will be discussed in Chapter 5.

For the purposes of efficient flow management and QoS control the network is partitioned into *domains* [Crosby,93] which constitutes arbitrary collections of network devices (e.g., switches, routers, multimedia workstations, continuous media storage servers). In each domain, the network aspects of the flow management resource reservation function are realised as a single domains server called a *domain flow management (DFM)* as illustrated in figure 4.11.

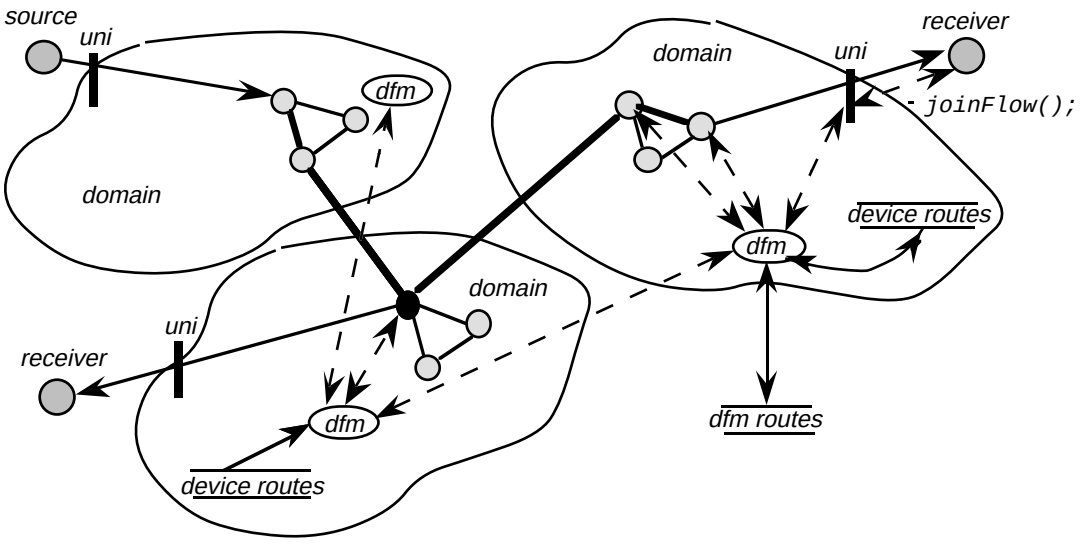

***Figure 4.11**: QoS-A Flow Management Domains*

A number of architectural choices are possible for domain flow management realisation. One proposal in the literature [Cidon,92] advocates a fully distributed architecture capable

of making resource management decisions for the complete flow such as flow admission control at any node. This has the advantage of reducing connection setup time since global state is available locally. However, disadvantages include the latency associated with maintaining consistency between multiple nodes and the additional network load incurred. Our approach is *partially* distributed in nature (one server per domain) and thus reduces the overhead incurred when maintaining database integrity while simultaneously avoiding bottlenecks introduced by an excessively centralised solution.

When a joinFlow is issued by a transport user, the control plane consults its local DFM. If the request can be satisfied, the DFM provisionally marks the requested resources as allocated. The DFM then multicasts a set_attribute_request primitive to the network QoS managers at all nodes in the data path toward the core to request that they allocate the resources requested. It then sends a confirmation to its requesting transport service user when all the QoS managers involved have acknowledged (via set_attribute_confirm) allocation of the requested resources. When a connection traverses a domain boundary DFMs interact directly to establish the connection.

4.7.2 QoS Adaptation

In its QoS adaptation role, the flow management plane is responsible for initiating the coarse grained QoS adjustments specified in the service contract's adaptation clause. The behaviour of the flow management plane is also determined by the maintenance clause.

If the maintenance mode is in signal or monitor mode the flow management plane simply receives periodic QoS assessments from the transport QoS manager and passes them on to the application. In maintenance mode the flow management plane does not receive any QoS assessments as the responsibility for flow management has been delegated to the layer transport QoS manager.

The flow management plane may receive QoS alerts [Campbell,93a] from the transport QoS manager if the latter is unable to maintain the flow within the prescribed bounds. In this case, the flow management plane takes appropriate action based on the adaptation clause. These actions consist of the issuing of a QoS degradation indication via a QoS event signal to the user, the initiation of a QoS adaptation (i.e., QoS renegotiation) on one or more specific performance parameters or QoS commitments.

QoS signals are forwarded to the application based on the monitor mode:

```
typedef enum {source, receiver, remote } monitor_t;
```

Options source and receivers default to the flow management interfaces as indicated at the "s" interface in figure 4.5. If the *remote* monitoring option is selected then QoS signal messages are forwarded to a remote flow management entity for processing. This remote monitoring is particularly important for multicast QoS management; see later for full details of this service. Periodic performance notifications include measured bandwidth, delay, jitter and losses for on-going flows as measured at the source or receiver. The signal fields of the QoS policy (described in Chapter 6) allow the user to specify the interval over which one or more QoS parameters are monitored and the user informed.

Remote QoS adaptation handlers are generally used for multicast QoS management. Actions taken by the multicast QoS management entity can include QoS adaptation or instantiation of new QoS filters to address fluctuation in the delivered QoS experienced by members of the multicast group. These and other multicast QoS management issues will be discussed in Chapter 6.

4.8 Baseline QoS-A

This thesis only considers the implementation of the transport and network layers of QoS-A model. The transport comprises the METS protocol (METSP) in the user plane and the corresponding METSig in the control plane. The flow management plane is realised as a per end-system *flow management protocol (FMP)* which subsumes the flow management functions of QoS mapping, admission control, resource reservation and QoS adaptation. The physical network layer is based on ATM switching.

The METSig protocol provides a framework for the establishment of dynamic QoS management of network resources using the meta-signalling protocol described in section 4.3.1. The FMP in the local ATM area has two primary resource reservation functions:

i) to request the allocation of CPU and memory resources on remote end-systems; and

ii) to allocate and manage resources in the network via interactions with METSig.

Below the METSP layer in the user plane, an ATM Adaptation Layer service is used to perform segmentation and reassembly of IP packets into/from 53 byte ATM cells. A minimal AAL5 service is utilised for this purpose.

The lowest layer of the architecture is based on the Lancaster Research ATM

Networking Environment. This delivers ATM to a combination of workstations, PCs, and multimedia devices designed at Lancaster [Blair,93] [Lunn,94] and also interconnects a number of Ethernets and interfaces to the rest of the UK via the SuperJANET 34 Mbps Joint Academic Network. The PCs are directly connected to ATM switches manufactured by Olivetti Research Limited (ORL) via 4x4 ORL ATM interface cards. The ATM switches are implemented using 'soft' switching and run a small micro-kernel called ATMos as identified in figures 4.12a and 4.12b. The PCs run an extended Chorus or Linux in order to provide the necessary operating system support for QoS-A. Next, in Chapter 5 the detailed design of the QoS-A operating system support will be presented.

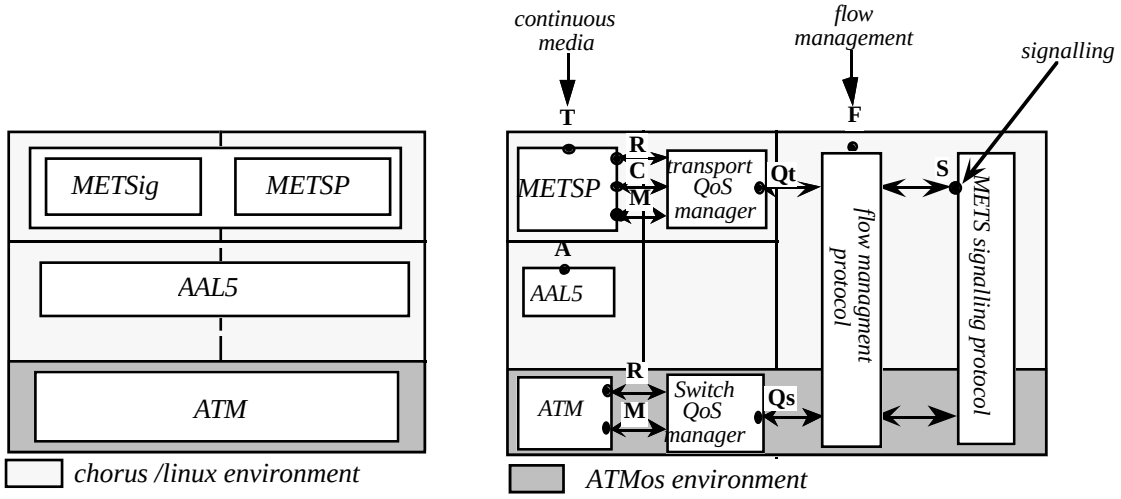

Figure 4.12a: Flow Protocol Projection *Figure 4.12b: Flow Management Projection*

4.9 Summary

This chapter outlined the QoS-A and focused, in particular, on the transport system. The METS transport system is comprised of signaling (METSig) and protocol (METSP) modules which map to the control and user plane of the QoS-A, respectively. To meet transport level QoS requirements, METSP incorporates buffer sharing, rate regulation, scheduling and basic flow monitoring QoS modules. Each module is configured based on the flow specification and QoS commitment described in the service contract QoS specification. Group management, multicast connection management and a signalling component are components of METSig. The QoS maintenance plane is comprised of a

transport QoS manager for the fine grained QoS management of on-going flows. Applications interact with a flow management protocol over the fm-socket interface for the establishment and dynamic QoS management of multicast flows.

This chapter concluded with a description of the baseline QoS-A which will be addressed in the implementation and evaluation chapters (Chapters 7 and 8, respectively).

Chapter 5

Operating System Support for Quality of Service

Operating systems support for QoS configurable communications is an important aspect of a generalised QoS-A. This chapter focuses on resource management components of the flow management, QoS maintenance and protocol planes in the end-systems with particular emphasis on CPU scheduling, network resource management and memory management issues. With a trend toward the use micro kernels in switches for the purposes of signalling and server creation (e.g., the ORL uses the ATMos microkernel [French,93]) the content of this chapter is also applicable (with certain restrictions) to network switch operating system design. The focus of this chapter is, however, the idealised end-system operating system support for a QoS-A which guarantees QoS levels of both communications and processing with varying degrees of QoS commitment as specified by a user level service contract. The flow management plane of the QoS-A uses admission tests to determine whether or not new activities can be accepted and modules for QoS mapping of flow specification into representations usable by the scheduling, network and memory management subsystems.

This chapter aims at providing system software support for continuous media applications in an environment of *standard* workstations utilising ATM based networking. Our specific aims are as follows:-

- to support a heterogeneous system consisting of PC and workstation end-systems connected by ATM,
- to enable continuous media applications to enjoy predictable performance in both communications and processing according to user provided QoS parameters,
- to retain the ability to run standard UNIX applications alongside continuous media applications.

The approach taken is to use the Chorus [Bricker,91] and Linux operating system [Linux,93] to underpin both UNIX and multimedia applications. This chapter primarily

focuses on extensions made to the Chorus micro-kernel to support QoS-A requirements. Many of the design decision made for Chorus are, however, applicable with certain relaxation to Linux; this is discussed this further in Chapter 7.

This Chapter begins by providing, in section 5.1, some necessary background material on Chorus. Following this section 5.2, presents an overview of the proposed QoS extensions to Chorus. This consists of:-

- a *CPU scheduling framework* which minimises kernel context switches in both application and protocol processing,
- an *ATM based communication stack* a in the QoS-A baseline architecture;
- a framework for *QoS driven memory management*, and
- a framework for *flow management* which integrates the management of resources in both end-systems and the network.

Section 5.3 investigates the management of CPU, communications and memory resources in this architecture. The various resource management functions are categorised as either *static* or *dynamic*. In essence, static QoS management deals with connect time aspects of the control and flow management planes including issues such as *QoS mapping* (i.e. deriving resource quantities from QoS parameters), and *admission testing* (i.e. determining whether new sessions can be created given their specific resource requirement and current resource availability) and *resource reservation.*

Dynamic QoS management (DQM) aspects of the flow management, QoS maintenance and user planes, on the other hand, deals with media-transfer time issues. In its full generality dynamic QoS management subsumes *maintenance*, *monitoring*, *flow scheduling*, *flow shaping* and *QoS adaptation* of QoS levels. The role of DQM *QoS control*, which is the only dynamic aspect treated in this chapter, is to actually achieve the requested levels of QoS given the resources statically dedicated at resource allocation time - e.g. by providing suitable scheduling mechanisms, and arranging for time constrained memory access and protocol operation.

5.1 Background on Chorus

Chorus is a commercial micro-kernel technology which supports the implementation of conventional operating system environments through the provision of ‘personalities’ (for

example a personality is available for UNIX SVR4 as mentioned above). The micro-kernel is implemented using modern techniques such as multi-threaded address spaces and integrated message based communications. The basic Chorus abstractions are *actors*, *threads* and *ports*, all of which are named by globally unique identifiers. Actors are address spaces and containers of resources which may exist in either user or supervisor space. Threads are units of execution which run code in the context of an actor. They are scheduled according to either a pre-emptive priority based or round robin timeslicing scheme. Ports are message queues used to hold incoming and outgoing messages. The inter-process communication sub-system supports both request/reply messages and asynchronous messages.

Chorus has several desirable real-time features and has been fairly widely used for embedded real-time applications. Its real-time features include pre-emptive scheduling, page locking, timeouts on system calls, and efficient interrupt handling. Unfortunately, Chorus' real-time support is not fully adequate for the requirements of distributed real-time and multimedia applications, principally because there is no support for QoS specification and resource reservation:-

- although it is possible to specify thread scheduling constraints relative to other threads, *absolute* statements of requirement for individual threads cannot be made,
- in the communications sub-system, the exclusive use of connectionless datagrams makes it impossible to pre-specify communications resource allocation,
- due to the use of a paged virtual memory system it is not possible to place bounds on memory access latency except by the extreme measure of wiring pages.

Note, however, that such limitations are not unique to Chorus: they are shared by most of the other micro-kernels in current use (e.g. [Accetta,86], [Tanenbaum,88]).

5.2 Operating System Support for Quality of Service

5.2.1 Chorus API with QoS Extensions

To remedy its current deficiencies for QoS specification and real-time application support, the Chorus system call API has been extended with new low level calls and abstractions as required by the QoS-A. The new abstractions, provided in both the kernel and a user level library, are illustrated in figure 5.1 and described below.

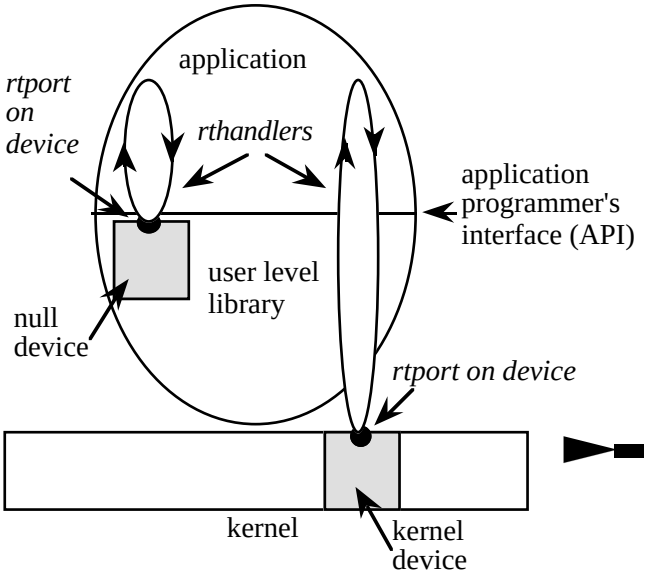

Figure 5.1: Devices, Rtports and Handlers

• *rtports*: these are extensions of standard Chorus ports and serve as access points for real-time communications. Rtports have an associated QoS which defines timeliness constraints on communication. They also provide direct application access to buffers thus minimising copy operations.

• *devices*: these are producers, consumers and filters of real-time data which support the creation of rtports and provide the memory for their buffers. One special type of device is the *null* device which is implemented in a user level library and permits user code to produce/ consume real-time data through the use of *rthandlers*.

• *rthandlers*: these are user supplied C routines which provide the facility to embed application code in the real-time infrastructure. They are attached to rtports at run time and upcalled on real-time threads by the infrastructure when data is available/ required. They encourage an event-driven style of programming which is appropriate for real-time applications and also avoid the context switch overhead associated with a traditional *send()*/ *recv()* based interface.

• *QoS controlled connections*: these are communication channels with a specific QoS[1]. A connection is established between a source and a sink rtport according to a

[1] QoS controlled connections are *abstractions* and are uniformly used for both remote and local communications. In the remote case, they are implemented in terms of the communications architecture described in section 3.3. In the local case, they are implemented in terms of optimised memory mapping mechanisms described in section 4.5.3.

given QoS specification. There are two types of connection: *stream connections* for periodic and continuous media data, and *message connections* for time-constrained messages. Stream connections are *active* in the sense that they initiate the transfer of data by upcalling a source rthandler (if attached). Message connections differ in that they *passively* wait for a source thread to pass them data via an *ipcSend()* call.

• *QoS adaptation handlers*: these are upcalled by the infrastructure in a similar way to rthandlers but are used to notify the application layer when QoS commitments provided by connections have been violated.

In addition to these features, the Chorus API includes calls for dynamically re-negotiating the QoS of open connections and for building pipelines of 'software signal processing' modules for local continuous media processing such as QoS filtering and scaling. It also has synchronisation primitives based on event counters and sequencers which incorporate the notion of *deadline inheritance* [Coulson,94b] whereby a 'worker' thread carrying out a task on behalf of a calling thread inherits the deadline of the caller. Full details of the continuous media API are specified in [Coulson,94a] and [Coulson,94b].

5.2.2 End-System Scheduling

The scheduling architecture exploits the concept of *lightweight threads* which are supported in a user level library and multiplexed on top a single Chorus kernel thread per actor. In this context, Chorus kernel threads are referred to as *virtual processors* (VPs). The scheduling architecture is a *split level* configuration [Govindan,91] consisting of a single kernel scheduler (KLS) to schedule VPs, and per-actor user level schedulers (ULSs) to schedule lightweight threads on those VPs (see figure 5.2).

The advantage of lightweight threads and user level scheduling is that context switch overhead is minimal. On the other hand, the drawback of user level scheduling is that, by definition, it cannot ensure that CPU resources are fairly shared across multiple actors. This is the role of kernel level scheduling. The split level architecture combines the benefits of both user level and kernel level scheduling by maintaining the following invariants:-

i) each ULS always runs its most urgent[1] *lightweight thread, and*
ii) the KLS always runs the VP supporting the globally *most urgent lightweight thread.*

[1] The notion of 'urgency' is dependent on the scheduling policy used (e.g. it would be *deadline* for EDF scheduling and *priority* for rate monotonic scheduling). The issue of scheduling policies is deferred until section 4.3.

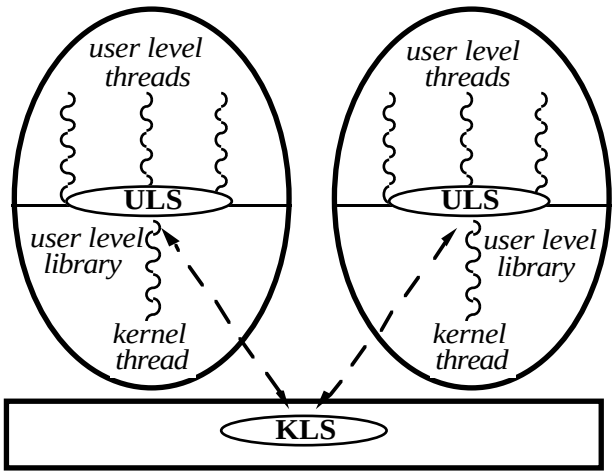

Figure 5.2: Split level scheduling architecture

The scheduling invariants are maintained via a KLS/ ULS information exchange realised in terms of shared KLS/ ULS memory areas and software interrupts [Govindan,91]. The shared memory area is divided into per-VP areas, each of which contains the urgency of the most urgent runnable lightweight thread known to its associated VP (along with some other information as described below). These urgency values are read by the KLS on each kernel level rescheduling operation to determine the next VP to schedule. Software interrupts are used by the KLS to inform VPs of the occurrence of real-time events in a timely fashion. Such events include timer expirations (used to implement pre-emption in user level scheduling), and data arrivals from local kernel devices or from the network. Software interrupts are always targeted at VPs but can be initiated either by kernel components (e.g. the KLS) or by library code in other application actors (see section 5.3.5.3).

The scheduling scheme also embodies the notion of *conditional urgency. This* allows not only the urgency of the most urgent runnable lightweight thread to be taken into account by the KLS (as above), but also the urgency of currently *blocked* threads. The implementation, which again exploits the shared memory area, uses per-VP *conditional urgency sets* which contain {*thread_id, event, urgency*} triples. In each triple, *urgency* represents the urgency that thread *thread_id* would have if only *event* was available to unblock it. The *urgency* values must all be greater than the urgency of that VP's most urgent runnable lightweight thread and the *event* values must all refer to events expected

from an external source. Thus, when the KLS has an event to deliver, it will run the VP to which *event* is addressed iff there is a matching triple in that VP's conditional urgency set *and* the indicated *urgency* value is globally more urgent than that of any other lightweight thread.

To avoid potential violations of the scheduling invariants, the system call interface seen by lightweight threads in terms of non-blocking system calls is implemented[Marsh,91]. If lightweight threads performing system calls were permitted to block their underlying VP, they would also necessarily block all other lightweight threads multiplexed on that VP. Then the scheduling invariants would be violated if one of these other threads happened to be the globally most urgent. Non-blocking system calls avoid this problem by returning *immediately* from system calls, thus allowing ULSs to block the calling lightweight thread at the library level while continuing to run other lightweight threads on the actor's VP. The results of calls are eventually notified to the ULS via software interrupts. On receipt of such an interrupt the ULS stores the result in the data structures of the original lightweight thread and then lets it 'return' from its system call. Thus application code sees only blocking system calls (as per standard Chorus), and the complexities of non-blocking calls are masked by library code.

The implementations of software interrupts and non-blocking system calls also exploit the shared kernel/ user memory area. To deliver a software interrupt, the kernel places an event identifier and parameters in the shared memory area and then alters the program counter field of the user VP's context structure (also in shared memory) to point to a well-known entry point in the ULS. Thus, when the VP is next scheduled by the KLS, the VP immediately enters the ULS which picks up the event identifier and parameters, and schedules a lightweight thread to deal with the event. The implementation of the proposed variant of non-blocking system calls which, because of the analogy with software interrupts is referred to as *asynchronous system calls* [Coulson,94b], is similar. The user level library places an operation identifier and parameters in shared memory and then sets an 'operation request' bit. The KLS, when it runs at the next system clock tick, notices that the operation request bit is set and copies the user's parameters to the appropriate VP or kernel server thread as determined by the operation identifier. Note that both software interrupts and asynchronous system calls avoid a special domain crossing; the call is actually effected the next time the recipient context (i.e. the ULS or the kernel) gets control by other means.

5.2.3 Communications

The standard Chorus communications stack was designed for the support of connectionless datagram services and uses retransmission strategies to enhance reliability. In contrast, the QoS-A baseline architecture (see chapter 4) is intended to support QoS controlled connection oriented communications. Because of these disparate design goals, the stack has been initially designed to operate entirely separately from the existing Chorus facilities.

In implementation, the baseline architecture is mapped partly onto per-actor user level libraries and partly onto a single, per machine, supervisor actor called the *network actor*. The transport protocol signalling is implemented as part of a flow management actor. The remaining transport functions and transport QoS manager are implemented in the same user level library that supports the service contract API discussed below so that its service interface can be provided using the rtport and rthandler abstractions. The media transfer aspects of transport protocol communicates with the network actor via system calls for send side communications, and software interrupts for receive side communications. All signalling associated with a flow communications with the *flow management actor* via system calls for the send side, and software interrupts for the receive side.

Below the transport protocol, the rest of the communications architecture, including the ATM card and AAL5 device driver, is also implemented in the network actor. A software AAL5 implementation is required because the ATM interface cards used only support data transfer at the granularity of ATM cells which is required for the cell based scheduling experimentation. The AAL5 implementation uses per-flow threads to perform segmentation and reassembly on both the send and receive side, with optional checksumming. This implementation choice reduces multiplexing in the stack to an absolute minimum as dictated by the performance principle.

Currently, the maximum service data unit sizes for the AAL5 and transport layers alike is restricted to 64 KBytes. This means that no further segmentation/ reassembly is required above the AAL5 layer[1]. The ATM cards are interrupt driven and communication between the interrupt service routines and the per-flow AAL5 threads is via Chorus ‘mini-ports’. See section 5.3.4.3 for more details of the low level cell handling functions and AAL5 implementation.

[1] It would be a relatively straightforward extension to support arbitrarily sized buffers at the API level by supporting segmentation and reassembly in the transport protocol if this proved necessary.

5.2.4 Memory Management

The standard abstractions used by the Chorus virtual memory system are *segments*, *regions* and *mappers* :-

•segments are the unit of information exchange between the outside world (e.g. files or swap areas) and the virtual memory (VM) layer in the kernel. In main memory segments are represented by so-called *local caches* of physical pages.

•regions are the unit of structuring of actor address spaces. A region contains a portion of a segment mapped to a given virtual address. Regions have associated access rights which are policed by the VM layer.

•mappers are supervisor actors which implement the link between external segments and their main memory representation and maintain the protection and consistency of segments. Mappers are accessed from the kernel via an upcall RPC interface when the kernel needs to bring in or swap out a page of a segment.

The purpose of the extended memory management architecture, which is built on top of the above abstractions, is to ensure that applications and QoS controlled connections can access memory regions with *bounded latency*. It is of little use to offer guaranteed CPU resources to threads if they are continually subject to non predictable memory access latency due to arbitrary page faulting[1]. Our design encapsulates most of the QoS driven memory management functionality inside a QoS enhanced agent called the *QoS mapper*. The roles of the QoS mapper are:-

• supplying application actors with memory regions offering latency bounded access,

• determining whether or not requests for QoS controlled memory resources should succeed or fail,

• pre-empting QoS controlled memory from ‘low urgency’ threads on behalf of ‘high urgency’ threads when necessary, and,

• efficiently re-mapping QoS controlled memory regions from one actor to another.

In addition to servicing requests from the kernel VM layer, the QoS mapper is used to

[1] Note that, in addition to buffers, it is also necessary to provide bounded latency access to code and stack regions of QoS controlled threads if QoS guarantees are to be maintained.

implement the connection abstraction in the intra-machine connection case (see section 5.3.5.3). User level code can also invoke the QoS mapper via extended versions of the *rgnAllocate()* and *rgnFree()* Chorus system calls. These respectively allocate and free a QoS controlled region of memory at connection establishment time.

5.2.5 Flow Management

Flow management includes aspects of both static and dynamic QoS management. In the case of static QoS management, the flow management actor must arrange, at flow establishment time, for the allocation of resources according to the user specified service contract. As illustrated in figure 5.3 the *flow management protocol (FMP)* co-operates with the CPU, memory, and network resource management modules and partitions the responsibility for QoS support among the individual resource managers. For example, for remote communications, the FMP partitions the API level delay QoS parameter between the network and the CPU resource management module in the end-system.

The FMP is also responsible for the dynamic QoS management described in chapter 4. In this role, it can adapt to degradations in resources based on the QoS adaptation clause. The maintenance function embedded in the flow management actor will compensate for momentary degradation in one resource in terms of another. In this mode it will do this without either involving the application or violating the service contract. For example, and increase in jitter caused by the network can be transparently compensated by increasing the buffer allocation and adjusting the playout time of the media - as long as the delay QoS in the service contract is not thereby compromised.

I realise the functionality of the flow management and control planes under a flow management scheme which adopts a split level structure to the end-systems and communications resources. First, when a new joinFlow requests for QoS controlled connection is requested, the QoS mapping function in a user level library determines the resource requirements of new connection requests. The output of the QoS mapper is then directed to each resource management module in turn to performs admission testing and resource allocation for the end-system and network resources. Flow management must communicate with each of the CPU, network and memory resource managers independently, and only if all are able to provide the required resource commitment can the connection request be granted.

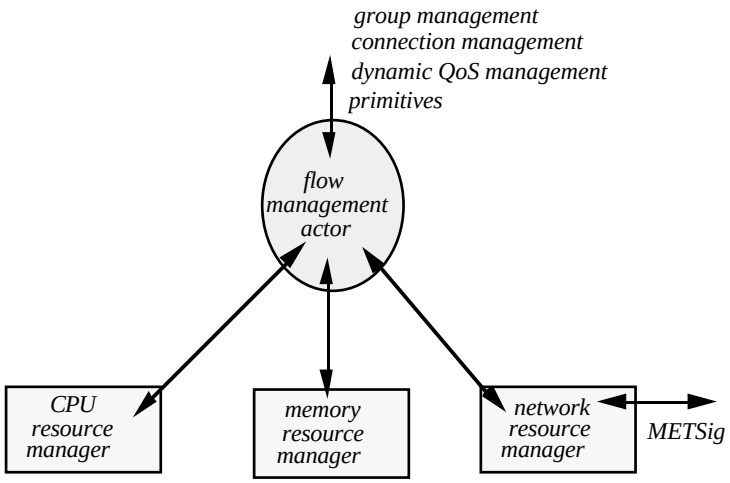

Figure 5.3: Flow Management

Connection management is necessary for remote communications. In this role flow management invokes METSig connection management to do this. In the end-system METSig is embedded in the network actor along with the transport and communication device driver functions.

5.3 Resource Management

Prior reservation of resources to connections is necessary to obtain guaranteed real-time performance. This section describes the resource reservation framework and shows how user level QoS parameters are used to derive the resource requirements of connections and make appropriate reservations. It also examines some dynamic QoS management issues. This chapter concentrates on the reservation of *specific* resources (i.e. CPU, memory and network resources) rather than treating resource reservation as an integrated activity driven by the FMP.

In outline, there are two stages in the resource reservation process. *QoS mapping* is the process of transforming service contract QoS parameters into resource requirements and *admission testing* determines whether sufficient uncommitted resources are available to fulfil those requirements.

5.3.1 Chorus Service Contract

Prior reservation of resources to connections is necessary to obtain real-time

performance. This section describes the resource reservation framework in the implementation and show how the user level service contract is used to derive the resource requirements of the connections and to make appropriate reservations. The are two stages to resource reservation: QoS mapping is the process of transforming the service contract's flow spec and commitment clauses into resources and flow admission control determines whether sufficient uncommitted resources are available to fulfil those requirements.

Our experimental ATM platform only uses a subset of the general service contract API described in section 5.2. The general service contract visible at the API is as follows:

```
typedef struct {
   flow_spec_t       flow_spec;
  commitment_t       commitment;
  delivery_t   delivery;
  maintenance_t      maintenance;
  adaptation_t       adaptation;
} service_contract_t;
```

The QoS parameters (which are further discussed in the text below) are visible at the API level are as follows:-

```
typedef enum {best_effort, guaranteed} com;

typedef enum {isochronous, workahead} del;


typedef struct {
   int   flow_id;
   int   frame_size;
   int   frame_rate;
   int   priority;
   int   burst;
   int   delay;
   int   loss;
   int   interval;
   int   jitter;
 } flow_spec_t;
```

Only two levels of QoS commitment are supported. If commitment is *guaranteed*, resources are permanently dedicated to support the requested QoS levels. Otherwise, if

commitment is *best effort*, resources are not permanently dedicated and may be preempted for use by other activities. The frame size is used to determine the required size of the internal buffer associated with the connection's rtports. *Priority* is a new QoS parameter f that was not described in chapter 4. It is used for fine grained control over resource pre-emption for connections; all things being equal, a connection with a low priority will have its resources pre-empted before one with a higher priority.

Delay refers to the maximum tolerable end-to-end delay, where the interpretation of 'end-to-end' is dependent on whether or not rthandlers are attached to the rtport. If rthandlers are attached, latency subsumes the execution of the rthandlers; otherwise it refers to rtport-to-rtport latency. When rthandlers are attached a further, implicit, QoS parameter called *quantum* becomes applicable. The value of this parameter is dynamically derived by the infrastructure whenever an rthandler is attached to an rtport. It is defined as the sum of the rthandler execution time and the execution time of the protocol code executed by the same thread directly before/ after the rthandler is called[1]. To determine the quantum value, the infrastructure performs a 'dummy' upcall of the handler and measures the time taken for it to return (a boolean flag is used to let the application code in the rthandler know whether a given call is 'real' or dummy). It is the responsibility of the application programmer providing the rthandler to ensure that the dummy execution path is similar to the general case. Although the value of quantum is dynamically refined as the connection runs, an inaccurate initial value will inevitably cause QoS violations.

Jitter, measured in milliseconds, refers to the permissible tolerance in buffer delivery time from the periodic delivery time implied by buffrate. For example, a jitter of 10ms implies that buffers may be delivered up to 5ms either side of the nominal buffer delivery time. *Delivery* is also an extension to the flow specification laid out in chapter 4 . It refines the meaning of buffrate. If *isochronous* delivery is specified, connections attempt to deliver *precisely* at the rate specified by buffrate; otherwise, if delivery is *workahead*, it is permitted to 'work ahead' (ignoring the jitter parameter) at rates temporarily faster than buffrate. One use of the workahead delivery mode is to more efficiently support applications such as real-time file transfer. Its primary use, however, is for pipelines of processing stages (e.g., QoS filters) where isochronous delivery is not required until the last stage [Coulson,94a].

---

[1] Actually there is a third component to the quantum value which is the per-buffer time taken by per-connection transmit threads at the ATM level. See section 4.4.3 for details.

5.3.2 Resource Classes

The following sections distinguish *four* major classes of QoS controlled connection for resource management purposes. These resource classes, named G_I, G_W, B_I, and B_W are selected on the basis of the *commitment* and *delivery* QoS parameters described above. They are defined and illustrated below:

Best effort { - isochronous (B_I) - workahead (B_W) } Guaranteed { - isochronous (G_I) - workahead (G_W) }

In addition to the two best effort classes a third best effort class, B_C, is distinguished which refers to non real-time Chorus and UNIX threads out of the scope of the real-time extensions. Additionally, all three best effort classes are often grouped together and referred to by the shorthand name B.

5.3.3 The CPU Resource

5.3.3.1 QoS Mapping

For admission testing and resource allocation purposes for stream connections, it is necessary to know the *period* and *quantum* of the threads associated with the connection. The period is simply the reciprocal of the *buffrate* QoS parameter and the quantum is implicitly derived at connect time as explained in section 5.3.1. Figure 5.4 illustrates the notions of period and quantum together with the related scheduling concepts of *scheduling time*, *deadline* and *jitter*.

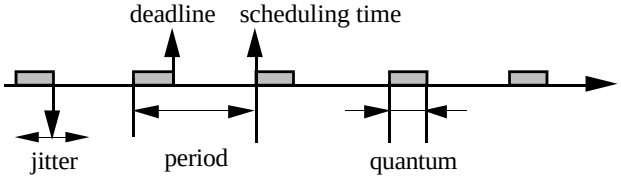

Figure 5.4: Periodic thread scheduling terminology

For message connections, *sporadic server* threads are used at the receive side[1]. One

[1] There is no thread implicitly associated with the source side of message connections. Dedicated threads are only applicable when rthandlers are used and, as pointed out in section 3.1, it is not useful to attach rthandlers to the source of message connections as message connections are not *active* in the sense of stream connections.

sporadic server per application actor is provided for each of the two applicable commitment classes (viz. G_W and B; isochronous delivery is not applicable to message connections), and each sporadic server handles all the message threads in its class. The quantum of each server is set to the *maximum* of the quanta of all the message threads in its class to ensure that adequate processing time is available for any of the server's associated threads. The period of each server is heuristically derived as follows:-

$$period = \min(recv\_latency_1, \ldots, recv\_latency_n)$$

$Recv\_latency_i$ is the proportion of the total end-to-end latency allocated by the FMP to the receive end-system for message connection *i.* This method of calculating *period* is a compromise which requires less resource than an optimal period (i.e. the optimal period, 1 / *quantum*, would ensure that the server was always ready to service a message but would take all the CPU resource allocated to the class!) while offering a reasonable probability that the server will be ready when a message arrives.

5.3.3.2 Admission Testing

The semantics of thread scheduling for each of the three resource classes are as follows:-

- G_I: threads for these connections are scheduled to run such that the completion of a quantum is guaranteed to be completed by the logical arrival time + *quantum* + *j* (where *j* is the jitter QoS parameter and logical arrival time is the start of the requisite period). An extended earliest deadline first [Liu,73] (EDF) algorithm and admission test is used to ensure this behaviour.

- G_W: these are scheduled according to the preemptible EDF policy. The jitter QoS parameter is ignored and quanta may be scheduled ahead of their logical arrival time to permit workahead. Again, an admission test is performed.

- B: these are scheduled according to the preemptible earliest deadline first policy but no admission test is used.

Each of the G and B resource classes is allocated a fixed portion of the CPU resource. Note, however, that the 'firewall' that this separation implies is used only to limit the number of threads in each class - not to restrict the use of CPU cycles at run time. If there are unused resources in one class, these resources are automatically exploited by the other

class at run time (see section 5.3.3.3).

The firewalls can be dynamically altered at run time by the programmer, but a typical configuration will allow a relatively small allocation for G threads. This is to encourage users to choose best effort threads wherever possible. Best effort threads should be perfectly adequate for many 'soft' real-time needs so long as the system loading is relatively low. The guaranteed classes should only be used when absolutely necessary - in particular, guaranteed isochronous threads should only be used for connections which are delivering data to an end device intended for human perception such as a video frame buffer.

The admission tests for G_I threads are:-

$$\sum_{i=1}^{N_G} \frac{quantum_i}{period_i} \leq R_G, \quad 0 \leq R_G \leq 1$$

$$\sum_{i=1}^{N_G} \frac{quantum_i}{quantum_i + jitter_i} \leq 1$$

The admission test for this class is a two stage process, and each of the two tests are modifications of the well known Liu/Layland test [Liu,73] (this guarantees that each quantum in the given set of tasks can be completed *at least* by the end of its period as long as it is runnable at the start of its period). The first test ensures that the overall resource used by all G threads is not greater than the allocated portion. N_G refers to the total number of G threads in the system and R_G refers to the portion of CPU resources dedicated to this class of threads (such that $R_G + R_B = 1$ where R_B represents the portion of the CPU resource dedicated to B threads).

The second test imposes the additional constraint that each quantum must complete by the end of its user stated jitter bound rather than simply by the end of the requisite period. Note that this second test is rather conservative (e.g. if a thread with zero jitter is requested the test will pass only this one thread!). However, this over-conservative property is relaxed by also taking into account the notion of *harmonic sets* (i.e. sets of threads all of whose periods are divisible by the period of the member with the smallest period). It can be shown [Mauthe,94] that harmonic sets can be scheduled without clashes as long as, in each period of the thread with the smallest period, it is possible to fit the quanta of all the threads in the set that fall within this period. This remains true even where the threads involved have a requirement for zero jitter. Future work optimally exploits the degrees of freedom allowed by the threads with relaxed jitter constraints for use by those with tight constraints [Mauthe,94].

For G_W threads the admission test is simply:-

$$\sum_{i=1}^{N_G} \frac{quantum_i}{period_i} \leq R_G$$

For B threads there is no admission test and the test for G_W sporadic servers is identical to that for G_W periodic threads. Each time a new message connection is created which alters the period or quantum of its server, a new admission test must be performed to ensure that the modified sporadic server can still be accommodated in the appropriate resource class.

5.3.3.3 QoS Control

At run time, the dynamic operation of the scheduling scheme uses a combination of *priorities*[1], *deadlines* and *scheduling times* to capture the abstract notion of 'urgency'. The scheduler uses three distinct priority bands into which the four classes of thread are mapped. The semantics of priority are that at any given time there is no runnable thread in the system that has a priority greater than the currently running thread. Within each priority band, all threads are made runnable when their scheduling time is reached and actually run when their deadline is earlier than the deadline of all other runnable threads in the band.

The G_I class is given a single high priority band (only critical Chorus server threads such as the pager daemon are allocated a higher band). B threads are given the next highest band and G_W threads are initially assigned to the lowest priority band. G_I threads are made runnable whenever their logical arrival time is reached (i.e. the start of the period pertaining to their current quantum). As mentioned above, G_W threads are initially assigned to the lowest priority band but they are 'promoted' to the G_I band when their logical arrival time is reached. This means that they can enjoy workahead when resources allow, but not at the expense of G_I and B threads. B_W threads are also runnable before their logical arrival time but are not similarly promoted. Finally, B_I threads only become scheduleable at a time indicated by the deadline minus the quantum time. This approximates isochronicity to the extent that it removes the possibility of jitter causing threads to complete *before* time although it still leaves the possibility of them completing *after* time. This overall scheme, in conjunction with the admission tests, ensures that G_I threads always meet their jitter

[1] Note that the 'priority' in this discussion is different from the priority API level QoS parameter. In this section priority is an internal thread scheduling attribute which is not visible or directly manipulable from the API level.

constraints, G_W threads always *at least* meet their rate requirement, and B threads optimally share the resources left to them.

Non real-time threads in the B_C class (e.g. those from conventional UNIX applications) are assigned appropriate priorities so that they receive reasonable service according to their role. Their deadline and scheduling time are always set to *now* so that they are effective scheduled solely on the basis of their priority. As an example B_C threads fulfilling an interactive role would have relatively high priority which may be greater than that of B threads. Other B_C threads, such as compute bound applications and non time critical daemons, will have accordingly lower priorities.

5.3.4 The Network Resource

5.3.4.1 QoS mapping

The network sub-system offers guarantees on *bandwidth*, *delay bounds* and *packet loss*. To enable it to do this, the QoS mapping function maps the API level QoS parameters onto a *flow spec* which is a representation of QoS appropriate to the AAL5 and ATM levels:-

```
typedef struct {
   int         flow_id;
   cmt         commitment;
   int         mtu_size;
   int         rate;
   int         delay;
   int         loss;
} flow_spec_t;
```

Flow_id uniquely identifies the network level flow. It is the virtual circuit identifier for flow specs used at the AAL5/ATM level. *Mtu_size*[1] refers to the maximum transmission unit size and *rate* refers to the rate at which these units are transmitted. These are directly derived from the buffsize and buffrate API level QoS parameters. *Delay* comprises that portion of the API level latency parameter which has been allocated, by the FMP, to the network. It subsumes both propagation and queuing delays in the network. Finally, *loss* is an upper bound probability of *mtu* loss due to buffer overflow at switches and routers.

[1] Although the discussion and admission tests in this section apply generically to both the IP++ and ATM layers, the admission tests are described here, for clarity, in an ATM context only. Mtu_size in the case of ATM cells is 53 bytes and in the case of IP++ packets is 64 KBytes. One restriction of the admission tests is that they are only applicable to switches/ routers with a *single* CPU. As we use single CPU ATM switches, this assumption is justified in our implementation environment.

Loss is trivially derived from the *error* and *error_interval* API level QoS parameters.

5.3.4.2 Admission Testing

In the network, only two traffic classes are recognised: *guaranteed* and *best effort* as denoted by the *commitment* API level QoS parameter. Admission testing and resource allocation are only performed for the former; best effort flows use whatever resource is left over.

For guaranteed flows, three admission tests are performed at each switch along the chosen path: a bandwidth test, a delay bound test and a buffer availability test. If, at the current switch, the admission control tests are successful, the necessary resources are allocated. Then the switch appends details of the cumulative delay incurred so far, and forwards the flow spec to the next switch. Eventually, the remote end-system performs the final tests and determines whether or not the QoS specified in the flow spec can be realised.

If the required QoS is realisable, the remote end-system returns a confirmation message to the initiating end-system. As it traverses the same route in reverse, the admission test protocol *relaxes* any over-allocated resources at intermediate switches [Anderson,91].

Bandwidth Test

The bandwidth test consists in verifying that enough processing (switching) power is available at each traversed switch to accommodate an additional flow without impairing the guarantees given to other flows. The admission test must satisfy worst case throughput conditions; this happens when all flows send packets back to back at the peak rate. As in section 5.3.3.2 the admission control test is based on [Liu,73]:-

$$\sum_{i=1}^{N} t_i \ . \ rate_i \leq R$$

Here, t_i refers to the service time (cf. mtu quantum) of flow i in the current switch, where there are N flows and $rate_i$ is the rate of the i'th flow (i.e. 1/ mtu period). R, $0 \leq R \leq 1$, represents the portion of resource dedicated to guaranteed flows.

Delay Bound Test

The delay bound test determines the minimum acceptable delay bound which does not cause scheduler saturation. There are two phases in the delay bound test. First, each switch

on the data path computes a local delay bound. Second, it is checked that the sum of all the local delay bounds do not exceed the flow spec's *delay* parameter.

The first phase calculation is taken from [Ferrari,90]:-

$$d = \sum_{i=1}^{N_U} t_i + T$$

Here, *d* is the local delay bound incurred at the current switch by the current flow. As before, *it* refers to the service time of flow *i* in the current switch, but here the index variable *i* ranges over the members of a set *U*. The set *U* contains those flows supported by the current switch whose local delay bound is lower than the sum of the service times of *all* flows supported by the current switch. N_U represents the cardinality of *U*. *T* represents the largest service time of all flows in a set *V* where *V* is the complement of set *U*. A full proof of the theorem underlying this formula can be found in [Ferrari,90]

The second phase calculation is:-

$$\sum_{n=1}^{N_S} d_n \leq delay$$

This merely requires that sum of the delays at each switch is less than the delay parameter in the flow spec. N_S refers to the number of switches on the path and d_n refers to the *n*'th value of *d* obtained from the first phase calculations.

Buffer Availability Test

The amount of per-switch memory allocated to a new flow must be sufficient to buffer the flow for a period which is greater than the combined queuing delay and service time of its packets. The calculation for buffer space is:-

$$buffersize = mtu\_size \lceil d \ . \ rate \ . \ loss \rceil$$

Here, *buffersize* represents the amount of memory that must be allocated at the current switch for the current flow. The combination of the queuing delay and service time is bounded by *d* as derived from the first phase delay formula above.

5.3.4.3 QoS Control

Much of the dynamic QoS maintenance for the execution of communications protocols in the end-system is encapsulated either in the protocols themselves (e.g. error control), in the scheduling subsystem (e.g. rate control, maintaining latency and jitter bounds) or in the

memory management subsystem (i.e. buffer management). However, one interesting QoS maintenance issue not in this category is the interleaving of ATM cells at the network interface from different connections on the basis of their QoS demands [Campbell,94]. On many ATM interface cards with on-board AALs this is taken care of in hardware but this testbed has been able to investigate this issue in software due to the fact that the experimental interface cards only deal with the ATM level.

The receive side cell processing is simple. The receiver interrupt service routine[1] reads the VCI of the current cell while it is still on the ATM interface card. The interrupt service routine then dispatches a receiver thread to copy the cell payload into the appropriate partially assembled AAL5 packet, and when the receiver thread sees the last cell of an AAL5 packet, it raises a software interrupt to the appropriate VP. Unfortunately, no QoS driven scheduling could be performed on the receive side as it proved imperative to get each cell off the board as quickly as possible to avoid excessive cell loss due to FIFO overrun. Thus the receive thread is given a scheduling priority higher than even G_I threads.

On the transmit side, cells were scheduled more intelligently by an EDF based *cell level scheduler*. Application actors running send side user level transport protocol code deliver buffers to the network actor via a system call. This informs the network actor of i) the location in its address space into which the buffer has been mapped and ii) the deadline of the buffer (which is the end of the quantum of the transport protocol thread).

The cell level scheduler runs in the context of the transmit interrupt service routine which is periodically activated by the ATM card to signal that cells can be copied to the card for transmission. The scheduler chooses to run one of a number of per-connection *transmit threads* by sending a message to a mini-port on which the transmit thread is waiting (see figure 5.5). The choice of thread to activate is made on the basis of priority, deadline and scheduling time as described in section 5.3.3.3. Each transmit thread is given the same priority band as its associated user level lightweight thread, and the deadline of each thread is derived from the deadline of the next cell in the thread's associated buffer. Cell deadlines themselves are derived by giving each cell in the buffer a specific temporal offset from the deadline of the entire buffer. The scheduling time of each thread becomes *now* whenever the thread has a buffer to send.

[1] Although the card interrupts on each cell arrival, the receive thread 'greedily' consumes any cells waiting in the card's FIFO each time it runs, thereby avoiding an interrupt for each cell.

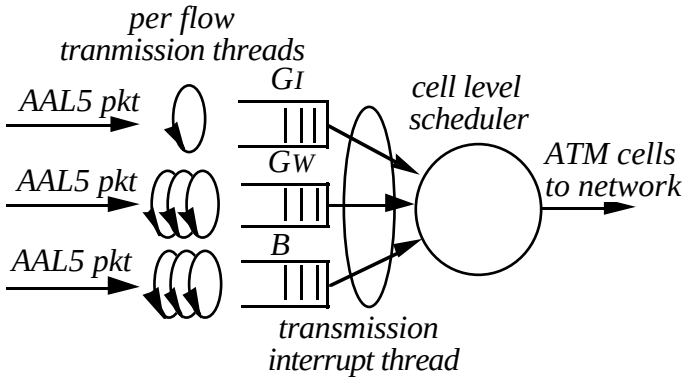

Figure 5.5: Cell level scheduler

The transmit threads are allocated at connection establishment time and are taken into account in the scheduling admission tests. This is done by adding a time t_{tx} to the *quantum* parameter of the connection's transmit side lightweight thread (see section 5.3.1); t_{tx} is calculated as *cells* x t_{cell} where *cells* is the number of ATM cells in a buffer of size *buffsize* and t_{cell} is the average time taken to transfer an ATM cell to the interface card.

5.3.5 The Memory Resource

5.3.5.1 QoS mapping

Two memory related quantities can be deduced from the user supplied QoS parameters at connection establishment time: i) the number of buffers required per connection, and ii) the required access latency associated with those buffers. Buffers are implemented as Chorus memory regions.

Number of buffers

To calculate the end-system buffer requirement, the *buffsize*, *buffrate* and *jitter* QoS parameters are used. It is also necessary to take into account the network delay bound, *delay*, offered by the FMP. The network delay bound will typically permit a larger degree of jitter than the API level jitter bound and any discrepancy must be made good through the use of additional jitter smoothing buffers. Given these input parameters, the expression for the number of buffers required at the receiver is:-

$$buffers = buffrate\ \left(delay + quantum + \frac{jitter}{2}\right)$$

In this formula, the expression in the brackets represents the maximum time for which

any single buffer must be held. *Delay* is the delay bound specified in the network level flow spec while *quantum*, *jitter* and *buffrate* are API level QoS parameters. *Jitter* is divided by two because the jitter parameter expresses both lateness and earlyness and it is only the lateness component that need be taken into consideration.

Only one buffer is required at the sender due to the structure of the send-side communications architecture: each buffer is assumed to be 'on the wire' before the start of the next period.

Region access latency

There are basically two qualities of memory access available in the standard Chorus system. These relate to the access latency of swappable pages and the access latency of locked pages. The latency bound of the former is a function of i) the delay due to the RPC communication between the VM layer and the mapper, and ii) the delay associated with the external swap device[1]. The latency bound of the latter is much smaller and is a function of the system bus and clock speed.

Swappable or locked regions are assigned to connections on the basis of their resource class as follows:-

- G_I: buffer regions allocated to these connections are locked and non-preemptible.
- G_W: buffer regions for these connections are locked but are potentially preemptible by memory requests from G_I connections if memory resources run low.
- B: buffer regions for these connections are assigned from standard swappable virtual memory. These regions may be explicitly locked by the API library code but are subject to pre-emption from by both G_I and G_W connections. The decision as to whether the library code should lock buffers or not is determined by the *priority* API level QoS parameter.

The QoS mapper can deduce the class of each memory request on the basis of the *commitment*, *delivery* and *priority* QoS parameters which are initially passed to the extended *rgnAllocate()* system call and retained to validate future operations on regions.

[1] We intend in the future to look at the possibility of bounding the access latency to swappable pages (e.g. through specialised page replacement policies and disc layout strategies), but our present design simply considers the access latency of swappable pages to be unbounded.

5.3.5.2 Admission testing

In its admission testing role, the QoS mapper maintains tables of all the physical memory resources in the system. In a similar way to the kernel level scheduler (KLS), it also maintains firewalls and high and low water marks between resource quantities dedicated to the different connection classes. The B section is used by all standard and non real-time applications as well as best effort connections.

If no physical memory is available to fulfil a request from a G_I connection, the QoS mapper can *preempt* a locked memory region from an existing B or G_W connection. Similarly, G_W connections can preempt locked regions from B connections. The QoS mapper chooses for pre-emption the buffer associated with the connection with the lowest *priority* in the lowest class available. The effect of pre-emption is simply to transform locked memory into standard swappable memory. This, of course, may result in a failure of the preempted connection's QoS commitment. However, a software interrupt is delivered to the ULS of a thread whose memory has been preempted so that if QoS commitments are violated, the connection concerned can deduce the likely reason.

5.3.5.3 QoS Control

The only dynamic QoS mapper function considered in detail is the region re-mapping function. This is used when buffers are mapped from one actor to another in a QoS controlled connection between local rtports. Region re-mapping is particularly important in the context of *pipelines* which arise when applications are structured as chains of modules which sequentially process a stream of real-time data. Pipelines can be implemented either within single actors or across multiple actors (or, indeed, across multiple machines although it is only the intra-machine cases that concern us here).

Pipelines across multiple actors are implemented using software interrupts as the control transfer mechanism. When a pipeline stage wants to send a buffer of data to a subsequent stage in another actor, the user level library implementing the QoS controlled connection performs a software interrupt:-

```
int raiseEvent(VP *dest; int event; bool unmap; VmAddr addr; VmSize size);
```

The *raiseEvent()* system call specifies a destination actor and details of the memory region to be remapped. When the kernel receives this call, it invokes the QoS mapper which maps the specified region into the destination address space (the boolean argument is used to control whether or not the region is also unmapped from the caller's address space). The QoS mapper then forwards a software interrupt to the target VP, passing as an argument the virtual address at which the region is mapped (see figure 5.6). Note that, in many cases, it is only necessary to perform this mapping the first time data is passed along the pipeline. Subsequent transfers can be accomplished using the existing shared region that *raiseEvent()* has already established and simply passing control with a call of raiseEvent with null *addr* and *size* parameters.

The extra cost due to the QoS mapper invocation is minimal. It incurs no protection boundary crossing and no virtual memory context switch as the QoS mapper is executed as a supervisor actor and thus shares the kernel's address space.

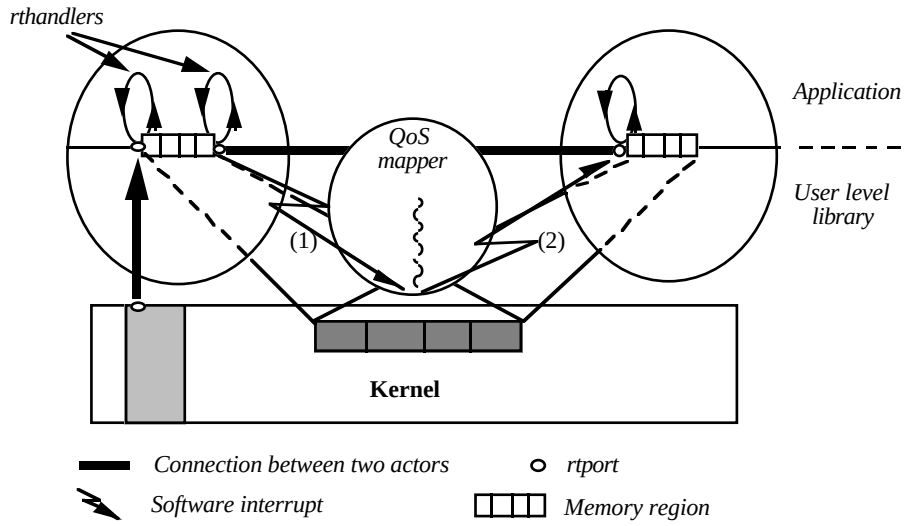

Figure 5.6 Pipeline example

The QoS mapper is also currently used as a repository of QoS related statistics of relevance to user level library code when it detects QoS degradations. The primary statistic is the number of page faults incurred by a region associated with a B connection. This information is used to better inform the choice of which B regions to lock and which to leave unlocked.

5.4 Summary

This Chapter has presented an idealised design of a QoS driven communications stack in a micro-kernel operating system environment. The discussion has focused on resource management aspects of the end-system operating system design and in particular has dealt with CPU scheduling, network resource management and memory management issues. The architecture minimises kernel level context switches and exploits early demultiplexing so that incoming media can always be treated according to the QoS of its associated API level connection. It also eliminates data copying on both send and receive (except for unavoidable copies to/from the ATM interface card). On send, the user's buffer is mapped to the lower layers which process it *in situ*, and, on receive, the lower layers allocate a buffer and map it to the transport layer which subsequently passes it to the application by passing the address of the buffer as an argument to an rthandler.

The Chorus micro-kernel QoS extensions provided an ideal operating system environment for QoS-A driven communications in the end-system. The idealised environment was not, however, available during the experimental phase of this work. Instead a Linux environment with ATM interconnect was considered to be flexible enough to realise a number of the important aspects of QoS support presented in this Chapter. Following this Chapter 7 describes a Linux-ATM implementation for the end-system and then, Chapter 8 presents an evaluation of the implementation.

Chapter 6

Dynamic QoS Management (DQM) of Scalable Multicast Flows

Distributed continuous media applications need to adapt to fluctuations in delivered quality of service. 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 made to adapt to fluctuating QoS with minimal perceptual distortion. This Chapter extends the QoS-A model described in Chapter 4 by populating the QoS management planes of the architecture with a framework for the control and management of *multi-layer coded* flows operating in *heterogeneous* multimedia networking environments.

Two key techniques are proposed:

i) an *end-to-end rate shaping scheme* which adapts the rate of MPEG-coded flows to the available network resources while minimising the distortion observed at the receiver; and

ii) an Adaptive Network Service, which offers “hard” guarantees to the base layer of multi-layer coded flows, and “fairness” guarantees to the enhancement layers based on a bandwidth allocation technique called *Weighted Fair Sharing*.

I also discuss a number of types of *scaling object* which are used to manage and control QoS. These include *QoS filters* which manipulate multi-layered coded flows as they progress through the communications system, *QoS adaptors* which scale flows at end-systems based on the flow's measured performance and user supplied QoS policy, and *QoS groups* which provide baseline QoS for multicast flows. All these components are inter-related by the QoS-A model and function as part of the *dynamic QoS management (DQM)* branch of the resource management framework described in chapter 4.

The structure of the chapter is as follows. First the salient features of scalable video flows is presented in section 6.1 before describing, in section 6.2, the set of scaling objects

used in the QoS-A architecture together with the application programming interface to DQM. Section 6.3 then presents a scheme for end-to-end DQM of adaptive multicast flows. In this section the detailed operation of a number of specific types of scaling object is described. Following this, section 6.4 introduces an adaptive network service, defines the notion of weighted fair share resource allocation, explains the rate control scheme used in the network, and describes the use of some network oriented QoS filters.

6.1 Characteristics and Composition of Scalable Video Flows

The fundamental design goal of digital audio-visual information representation schemes in the past several decades has been that of *compactness*: describe the signal's content with as few bits as possible. The algorithmic foundation of this work has been based on the assumption that information transport occurs over constant bandwidth and constant delay channels: an assumption that does not necessarily hold valid in an environment of packet switched networks working on the premise of *multiplexing gain*.

Our primary focus in this chapter is multi-layered coded video such as MPEG. Within this framework, there are two alternative techniques which underpin QoS adaptation: *intrinsic* and *extrinsic* techniques. The former are provided by the encoder, and are embedded in the coded bitstream. The latter are provided by QoS filters that operate directly on the compressed bitstream, performing the desired manipulations. Their difference lies in their complexity and performance. Intrinsic techniques can have a very simple implementation, but offer only a discrete range of possibilities. Extrinsic techniques are computationally more complex but can operate on a continuum of possibilities. The following section briefly describes the architecture of MPEG-2, with particular emphasis on the intrinsic adaptation capabilities it provides in the form of *scalability profiles*. Extrinsic adaptation can be provided through the use of dynamic rate shaping QoS filters [Eleftheriadis,95], which are discussed in section 6.3.2.1.

6.1.1 MPEG

The algorithmic foundation of MPEG is motion-compensated, block-based transform coding MPEG-1 and MPEG-2 [H.262,94]. Each block is transformed using the Discrete Cosine Transform (DCT), and is subsequently quantised. Quantisation is the sole source of quality loss in MPEG and, of course, a major source of compression efficiency. The

quantised coefficients are converted to a one-dimensional string using a zig-zag pattern and then run-length encoded. There are three types of pictures in a sequence as illustrated in figure 6.1: I, P, and B. I or *intra* pictures are individually coded, and are fully self-contained. P pictures are *predicted* from the previous I or P picture, while B (or *bi-directional*) pictures are interpolated from the closest past and future I or P pictures. Prediction is motion-compensated: the encoder finds the best match of each macroblock in the past or future picture, within a prespecified range. The displacement(s), or motion vector(s), is sent as side information to the decoder.

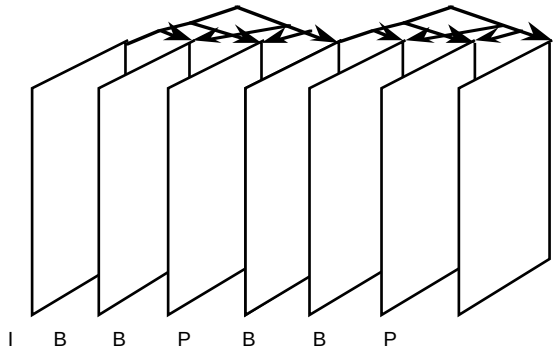

Figure 6.1: MPEG Picture Sequence

In order to increase the coding efficiency, MPEG relies heavily on entropy coding. Huffman codes (variable length codewords) are used to represent the various bitstream quantities (run-length codes, motion vectors, etc.). As a result, the output of an MPEG encoder is inherently a *variable rate* bitstream: the ratio of bits per pixel varies from one block to the next. Figure 6.2, which shows a bandwidth trace over time for an MPEG-1 video clip at 24 frame per second, illustrates both the variable bit rate nature and multi-layer composition of MPEG-1 video. The I frames which represents the base layer (BL) in the trace average approximately 200 ATM cells per second. By adding an enhancement layer E1, which represents the P pictures in MPEG, the bandwidth demand peaks above 400 cells per second. The final enhancement layer E2 (i.e., B pictures) increase the peak bandwidth demand to close to 600 cells per second.

In order to construct a constant rate bitstream, *rate control* is used. This is achieved by connecting a buffer to the output of the encoder. The buffer is emptied at a constant rate, and its occupancy is fed back to the encoder. This information is used to control the selection of the quantiser for the current macroblock. High buffer occupancy leads to more coarsely quantised coefficients, and hence less bits per block, and vice versa. Through this

self-regulation technique one can achieve a constant output rate.

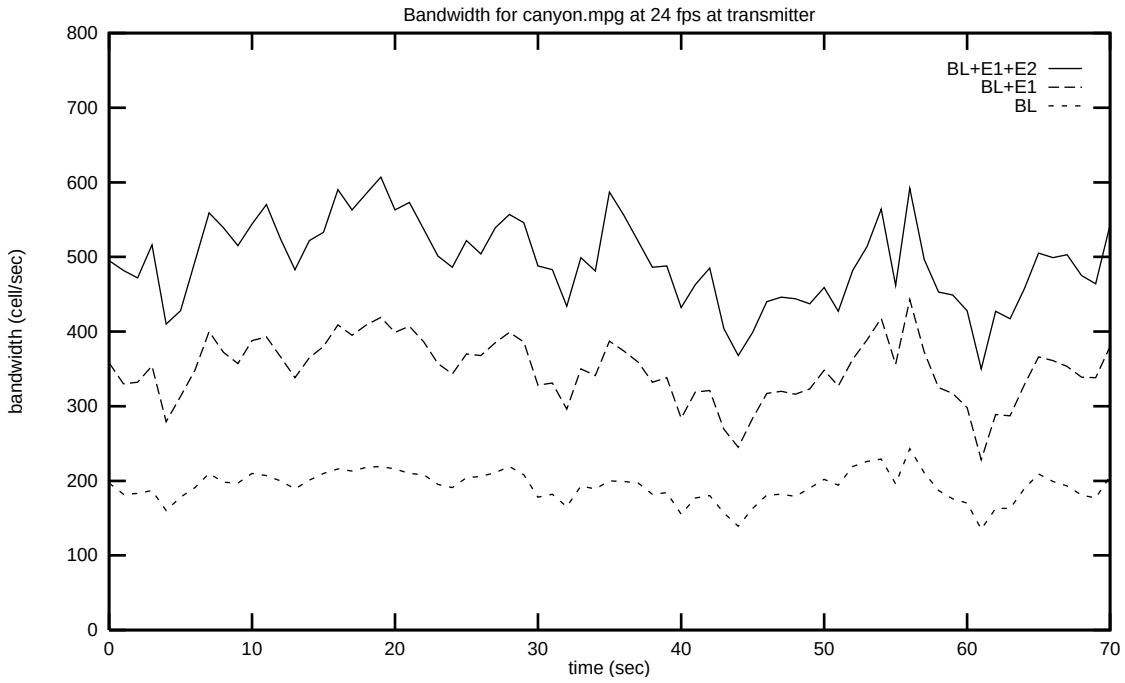

Figure 6.2: MPEG-1 Bandwidth Trace at 24 Frames Per Second

For transport purposes, video and audio are multiplexed according to the *system layer* of MPEG-2, which defines the packetisation structure and synchronisation algorithms between the audio and video signals. Two different packetisation structures are defined, namely *program streams* and *transport streams*. Both are logically constructed from “packetised elementary stream” (PES) packets; these are the basic units in which individual audio, video, and control information is carried. Program streams are designed for use in relatively error-free environments, use variable length packets, and combine PES packets that have a common time base. Transport streams are designed for noisy channels, utilise fixed-length packets (188 bytes), and can carry programs with independent time bases. The system layer's timing model is based on the assumption that a constant delay transport mechanism is used. Although deviations from this are allowed, the way to address them is not specified. All timing information is based on a common system clock, and timestamps (i.e. an absolute timing method) ensure proper inter-media synchronisation between the audio and video signals upon presentation.

6.1.2 Scalability Modes

From the above discussion it is clear that MPEG is especially tuned for transmission over traditional constant bandwidth and delay channels. Some support for flexible and/or robust transmission is provided through the use of *scalability modes* for channels exhibiting dynamic behaviour. The situation is even more challenging for scalable flows, i.e. in cases where the available bandwidth may vary over time. However, here benefits of multiplexing gain are possible.

MPEG-2 provides for the simultaneous representation of a video signal at various different levels of quality, through the use of multiple independent bitstreams or sub-signals. This is achieved through the use of pyramidal, or hierarchical coding: one first constructs a coarse or base representation of the signal, and then produces successive enhancements. The latter assume that the base representation is available, and only encode the incremental changes that have to be performed to improve the quality. There are four different scalability modes: *spatial*, *SNR*, *temporal*, and *data partitioning*. MPEG-2 allows the simultaneous use of up to two different scalability modes (except for data partitioning) in any combination, hence resulting in a 3-level representation of the signal. In spatial scalability, the base and enhancement layers operate at different spatial resolutions (e.g. standard TV and HDTV). In SNR scalability, both layers have the same resolution and the enhancement refines the quantisation process performed in the base layer. In temporal scalability the enhancement layer increases the number of frames per second of the base layer (e.g., from 30 frames per second to 60 fps). Data partitioning is slightly different from the other three scalability modes in the sense that the encoder does not maintain two different prediction loops and hence the base layer is not entirely self-contained. Its benefit is that it can be applied even in single-layer encoders.

As an example, the spatial scalability mode can be used to transmit digital TV in both standard and HDTV formats. Although two sub-signals are actually generated, this is not identical to simulcast: the total bandwidth required is much smaller, since the HDTV layer uses the base, standard TV layer as a reference point. Scalability can be very useful in transmission of video over adaptive channels.

6.1.3 Discrete and Continuous QoS Adaptation

Although MPEG-2's scalability features are useful in resolving heterogeneity problems described above [Delgrossi,93], and are useful in numerous applications, their use in continually QoS-varying channels is problematic. This is because they only allow the representation of the signal at a fixed number of discrete quality points (temporal or spatial resolution, or spatial quality). These points are typically significantly apart, and transitions between the two are perceptually significant. Table 6.1 [Paek,95] shows an example of hybrid scalability with spatial (E1) and SNR (E2) enhancement layers.

layer name	*profile*	*symbol*	*frame size*	*bit rate*	*subjective QoS*
base layer	main	BL	304x112	0.32 Mbps	VHS
enhancement 1 layer	spatial	E1	608x224	0.83 Mbps	super VHS
enhancement 2 layer	SNR	E2	608x224	1.85 Mbps	laser disc

Table 6.1: MPEG-2 hybrid scalable bitstream using spatial and SNR scalability (24 fps)

Consider, for example, a channel that temporarily undergoes rate variability for a period of a few seconds. Switching to a lower quality point (by discarding the enhancement layer(s)) for such a brief interval can create a perceptual “flash”. An additional issue is that, as soon as compression parameters are established, it is impossible to modify them later on (after compression is completed). Hence scalability modes can only really be used for well-defined, simple channels that vary slowly. Since the wide range of different access mechanisms to multimedia information makes it very difficult to select *a priori* a set of universally interoperable coding parameters, it is necessary to provide extrinsic mechanisms that allow the representation of the “signal at a continuum of qualities and rates” [Delgrossi,93], so that scalable flows can be accommodated.

This is possible through the use of a class of dynamic rate shaping QoS filters [Eleftheriadis,95b] and the provision of adaptive network services - providing a QoS continuum for fully scalable flows. The adaptive service introduced in this chapter uses explicit feedback from network resource management to dynamically shape the video source based on available network resources. Some benefits of an adaptive scheme are non-reliance on video modelling techniques and statistical QoS specification and specific support

for the semantics of scalable video flows e.g. MPEG-2 scalable profiles. Dynamic rate shaping filters manipulate the rate of MPEG-coded video, matching it to the available bandwidth (indicated by the adaptive service) while minimising the distortion observed by the receiver.

6.2 Scaling Objects and API Extensions

This section introduces a set of *scaling objects* used to manipulate hierarchically coded flows as they progress through the communications system. These comprise QoS adaptors, QoS filters and QoS groups as defined above. This section also introduces an extension to the QoS-A application programmer's interface which gives access to the scaling object types.

6.2.1 Scaling Objects

6.2.1.1 QoS Adaptors

QoS adaptors are used in conjunction with the QoS-A flow monitoring function to ensure that the user and provider QoS specified in the service contract are actually maintained. In this role QoS adaptors are seen as quality of service arbiters between the user and network. QoS adaptors scale flows at the end-systems based on a user supplied QoS scaling policy (see section 6.2.2) and the measured performance of on-going flows. QoS adaptation is discussed in detail in section 6.3.

6.2.1.2 QoS Filters

QoS filters manipulate multi-layer coded flows [Shacham,92], [Hoffman,93] [Zhang,95] at the end-systems and as they progress through the network. Three distinct styles of QoS filters are described:

i) *shaping filters*, which manipulate coded video and audio by exploiting the structural composition of flows to match network, end-system or application QoS capability; shaping filters are generally situated at the edge of the network at the source; they require non-trivial computational power; examples are the dynamic rate shaping (DRS) filter and the source bit rate (SBR) filter - see section 6.3.2;

ii) *selection filters*, which are used for sub-signal selection and media dropping (e.g. video frame dropping) are of low complexity and low computational intensity - selection filters are designed to operate in the network and are located at switches;

they require only minimal computational power; examples are sub-signal filter, hierarch filter, hybrid filter - see section 4.6.3;

iii)*temporal filters*, which manipulate the timing characteristics of media to meet delay bound QoS are also low in complexity and trivial computationally - temporal filters are generally placed at receivers or sinks of continuous media where jitter compensation or orchestration of multiple related media is required; examples are sync filter, orch filter - see section 6.3.3).

6.2.1.3 QoS Groups

Before potential senders and receivers can communicate they must first join a *QoS group* [Aurrecoechea,95]. The concept of a QoS group is used to associate a baseline QoS capability to a particular flow. All sub-signals of a multi-layer stream can be mapped into a single flow and multicast to multiple receivers [Shacham,92]. Then, each receiver can select to take either the complete signal advertised by the QoS group or a partial signal based on resource availability. Alternatively each sub-signal can be associated with a distinct QoS group. In this case, receivers "tune" into different QoS groups (using signal selection) to build up the overall signal. Both methods are supported in DQM. Receivers and senders interact with QoS groups to determine what the baseline service is, and tailor their capability to consume the signal by selecting filter styles and specifying the degree of adaptability sustainable (viz. discrete, continuous; see section 6.1.3).

6.2.2 QoS Specification Extensions for Scalable Flows

The original QoS-A service contract based API which formalised the end-to-end QoS requirements of the user and the potential degree of service commitment of the provider. This section proposes a number of extensions to the *flow specification*, *QoS commitment* and *QoS scaling policy* (which subsumes the QoS adaptation and QoS maintenance) clauses of the service contract required to accommodate adaptive multi-layer flows. The API presented here is not complete in that there are no primitives given for establishing and renegotiating connections or for manipulating QoS groups. Full details of these aspects are given in chapter 4 and chapter 7.

Multi-layered flows are characterised by three sub-signals in the *flowSpec* flow specification: a base layer (BL) and up to two enhancement layers (E1 and E2). Each layer is represented by a frame size and subjective or perceptive QoS as illustrated in Table 1.

Based on these characteristics, the MPEG-2 coder [Paek,95], [Eleftheriadis,95] determines approximate bit rate for each sub-layer. In the case of MPEG-2's hybrid scalability, BL would represent the main profile bit rate requirement (e.g. 0.32 Mbps) for basic quality, E1 would represent the spatial scalability mode bit rate requirement (e.g. 0.83 Mbps) for enhancement, and E2 would represent the SNR scalability mode bit rate requirement (e.g. 1.85 Mbps) for further enhancement. The remaining flow specification performance parameters for *jitter*, *delay* and *loss* are assumed to be common across the all sub-signals (i.e. a single layer of a multi-layer video flow). The QoS commitment field which is subsumed by the flow specification has been extended to offer an *adaptive* network service that specifically caters for the needs of scalable audio and video flows in heterogeneous networking environments (see section 6.4).

```
typedef enum {MPEG1, MPEG2, H261, JPEG} mediaType;
typedef enum {continuous, discrete}    adaptMode;
typedef enum {besteffort, adaptive, statistical, guaranteed} commit;

typedef enum {
   DRS, SBR, sub_signal, hierarch, hybrid, sync, orch
} filterStyle;

typedef struct {
   adaptMode         adaptation;
   filterStyle       filtering;
   events            adaptEvents;
   actions           newQoS;
   signal            bandwidth;
   signal            loss;
   signal            delay;
   signal            jitter
} qosPolicy;

typedef struct {
   gid               flow_id;
   mediaType         media;
   commit            commitment;
   subFlow           BL;
   subFlow           E1;
   subFlow           E2;
   int               delay;
   int               loss;
   int               jitter;
   qosPolicy         policy
} flowSpec;
```

The *qosPolicy* field of the *flowSpec* characterises the degree of adaptation that a flow can tolerate and still achieve meaningful QoS. The QoS policy consists of clauses that cover *adaptation modes*, *QoS filter styles*, and *event/ action* pairings for QoS management purposes. Two types of adaptation mode are supported: *continuous mode*, for applications that can exploit any availability of bandwidth above the base layer; and *discrete mode* for applications which can only accept discrete improvement in bandwidth based on a full enhancement (viz. E1, E2).

The QoS scaling policy provides user-selectable QoS adaptation and QoS filtering. While receivers select filter styles to match their capability to consume media at the receiver (from the set of temporal filters), senders select filter styles to shape flows in response to the availability of network resources such as bandwidth and delay (from the set of shaping filters). Network oriented filters (i.e. selection filters) can be chosen by either senders or receivers.

In addition, senders and receivers can both select periodic performance notifications including available bandwidth, measured delay, jitter and losses for on-going flows. The *signal* fields in the scaling policy allow the user to specify the interval over which a QoS parameter is to be monitored and the user informed. Multiple signals can be selected depending on application needs.

6.3. Dynamic QoS Management of Scalable Video Flow

6.3.1 Architectural Components

Dynamic QoS management spans the QoS-A (see figure 6.3) QoS maintenance and flow management planes. In the QoS maintenance plane, the most important aspect of DQM is QoS adaptation in the end-systems. Based on the receiver supplied QoS scaling policy, QoS adaptors take remedial action to scale flows, inform the user of a QoS indication and degradation, fine tune resources and initiate complete end-to-end QoS renegotiation based on a new *flowSpec* [Campbell,94].

In the flow management plane, DQM consists of two sub-components: *QoS group management* maintains and advertises QoS groups created by senders for the benefit of potential receivers; *filter management* [Yeadon,94] instantiates and reconfigures filters in a flow at optimal points in the media path at flow establishment time, when new receivers join QoS groups or when a new *flowSpec* is given on a QoS renegotiation.

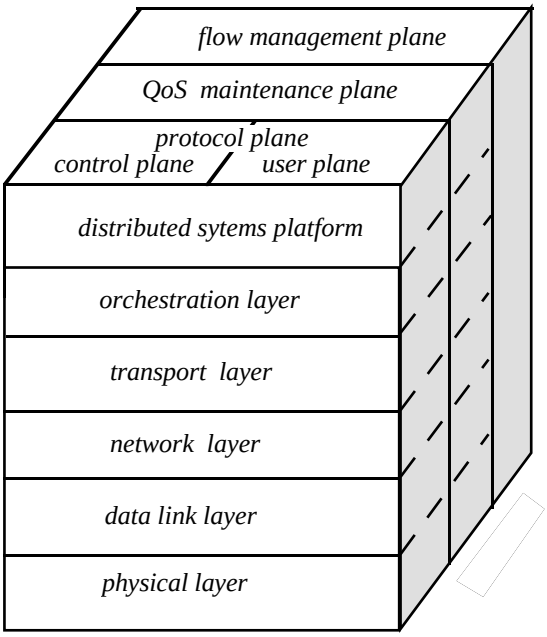

Figure 6.3: Quality of Service Architecture (QoS-A)

In implementation, each of these architectural modules has well defined interfaces and methods defined in CORBA IDL. CORBA [OMG,93] runs on the end-systems and in the ATM switches, providing a seamless object oriented environment throughout the communication system base (see [Aurrecoechea,94] for full details).

6.3.1.1 Illustrative Scenario

DQM can be viewed as operating in three distinct domains:

i) *sender-oriented DQM*, where senders select source filters and adaptation modes, and establish flow specifications. The sender-side transport protocol provides periodic bandwidth and delay assessments to the source filters (i.e. DRS or SBR filters) which regulate the source flow. Senders create QoS groups which announce the QoS of the flow to receivers via QoS group management;

ii) *receiver-oriented DQM*, where receivers join QoS groups and select the portion of the signal which matches their QoS capability. Receiver selected network based filters propagate through the network and perform source and signal selection. In addition, receiver-based QoS filters (i.e. sync-filter and orch-filter) are instantiated by default unless otherwise directed. These filters are used to smooth and synchronise multiple media. The receiver-side transport protocol provides

bandwidth management and produces adaptation signals according to the QoS scaling policy;

iii) *network-oriented DQM*, which provides an adaptive network service (see section 6.5) to receivers and senders. Network level QoS filters (i.e. sub-signal, hierarch and hybrid-filters) are instantiated based on user selection, and propagated in the network under the control of filter management.

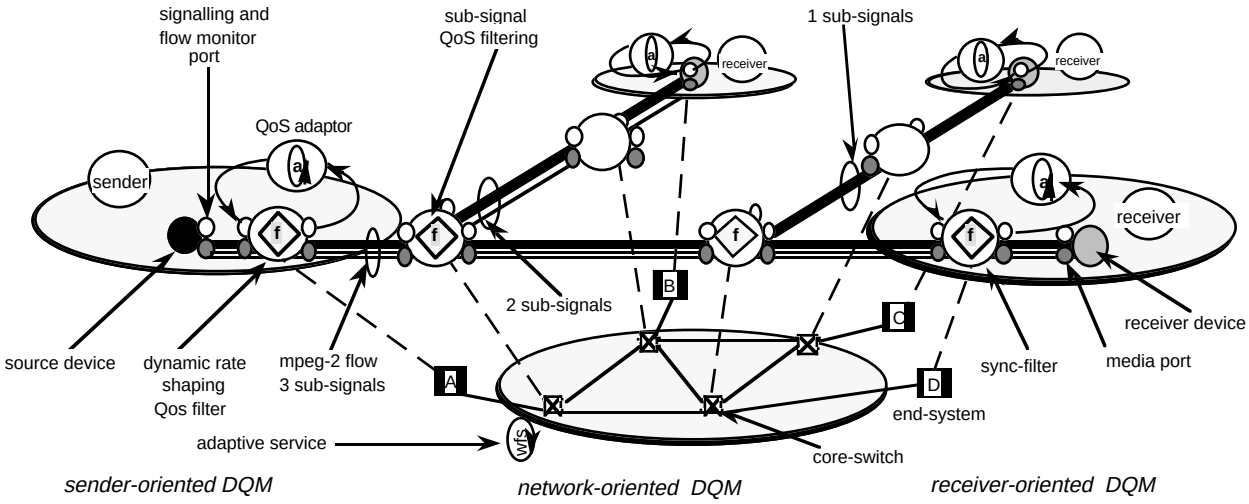

Figure 6.4: Dynamic QoS management (DQM) of Scalable Multicast Flows

In figure 6.4 a sender at end-system A creates a flow by instantiating a QoS group which announces the characteristics of the flow (viz. layer, frame size, subjective quality) and its adaptation mode. Receivers at end-systems B, C and D join the QoS group. In the example scenario shown the receivers each “tune” into different parts of the multi-layer signal: C takes BL, the main profile (which constitutes a bandwidth of 0.32 Mbps for VHS perceptual QoS), B takes BL and E1 (which constitutes an aggregate bandwidth of 1.15 Mbps for super VHS perceptual QoS), and D takes the complete signal BL+E1+E2 (which constitutes an aggregate bandwidth of 3 Mbps for laser disc equivalent QoS). In this example the complete signal is multiplexed onto a single flow, therefore, sub-signal selection filters are propagated by filter management. Receivers, senders, or any third party or filter management can select, instantiate and modify source, network and receiver-based QoS filters.

6.3.2 Sender-Oriented DQM

Figure 6.5 shows the functions of the sender-side transport protocol supporting dynamic QoS management, and the interface to a dynamic rate shaping filter. Currently, senders can select from two types of shaping filter at the source: dynamic rate shaping (DRS) and source bite rate (SBR) QoS filters. Both of these QoS filters manipulate the signal to meet the available bandwidth by keeping the signal meaningful at the receiver. The sender-side transport mechanisms includes a QoS adaptor, flow monitor and media scheduler. Bandwidth updates are synchronously received by the flow monitor mechanism from the network as part of the adaptive service (described in section 6.5). The QoS adaptor is responsible for synchronously informing the source filter of the current bandwidth availability (B_{flow}) and measured delay (D_{flow}), and calculating new schedules and deadlines for transport service data units [Coulson,95]. Media progresses from the source filter to the TSAP, and is scheduled by the media scheduler to the network at the NSAP based on the calculated deadlines.

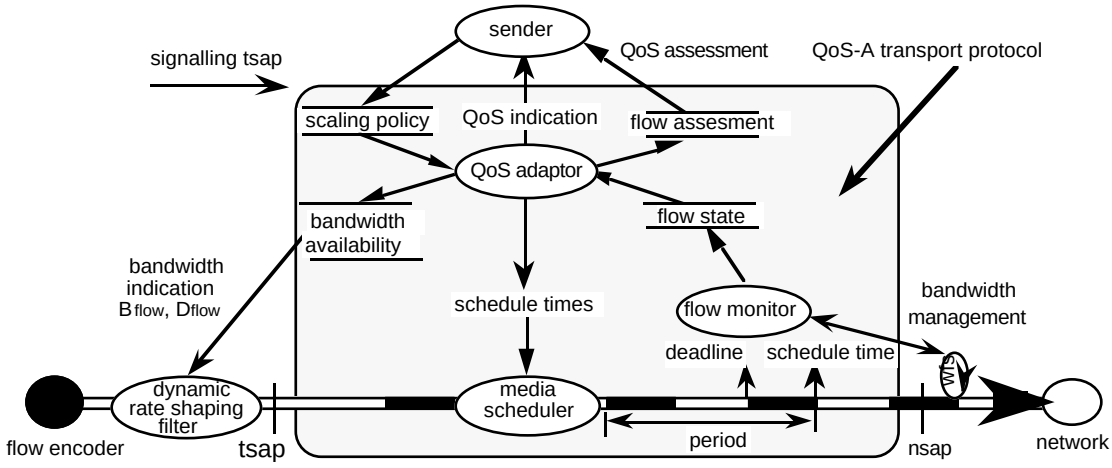

Figure 6.5: Sender-side transport QoS mechanisms

The QoS adaptor is also responsible for informing the sending application of the on-going QoS based on options selected in the QoS scaling policy. Informing the application of the current state of the resources associated with a specific flow is key in implementing adaptive applications in end-systems. In this case the application manages the flow by receiving updates and interacting with the QoS adaptor to adjust the flow, e.g. change

adaptation mode from continuous to discrete, request more bandwidth for BL, E1 and E2, or change the characteristics of the source filter, etc.

6.3.2.1 The Dynamic Rate Shaping (DRS) Filter

Rate shaping is defined as an operation which, given an input video bitstream and a set of rate constraints, produces a video bitstream that complies with these constraints. For this purpose, both bitstreams are assumed to meet the same syntax specification. In addition it is assumed that a motion compensated block-based transform coding scheme is used. This includes both MPEG-1 and MPEG-2, as well as H.261 and so-called “motion” JPEG.

Although a number of techniques have been developed for the rate shaping of *live* sources [Kanakia,93] these cannot be used for the transmission of pre-compressed material (e.g. in VoD systems). The dynamic rate shaping filter is interposed between the encoder and the network and ensures that the encoder's output can be perfectly matched to the network's quality of service characteristics. The filter does not require interaction with the encoder and hence is fully applicable to both live and stored video applications.

Because the encoder and the network are decoupled, universal interoperability can be achieved both between codecs and networks, and also among codecs with different specifications. An attractive aspect is the existence of low-complexity algorithms which allow software-based implementation in high-end computers. In order for rate shaping to be viable it has to be implementable with reasonable ease while yielding acceptable visual quality. With respect to complexity, the straightforward approach of decoding the video bitstream and recoding it at the target rate would be obviously unacceptable; the delay incurred would also be an important deterrent. Hence only algorithms of complexity less than that of a cascaded decoder and encoder are of practical interest. These algorithms operate directly in the compressed domain of the video signal, manipulating the bitstream so that rate reduction can be effected. In terms of quality, it should be noted that recoding does not necessarily yield optimal conversion; in fact, since an optimal encoder (in an operational rate-distortion sense) is impractical due to its complexity, recoding can only serve as an indicator of an acceptable quality range. In fact, regular recoding can be quite lacking in terms of quality, with dynamic rate shaping providing significantly superior results.

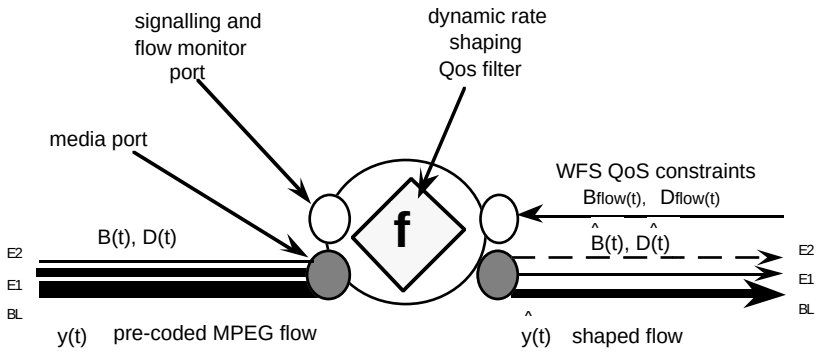

Figure 6.6: Dynamic rate shaping scheme

The rate shaping operation is depicted in figure 6.6. Of particular interest is the source of the rate constraints $B_{flow}(t)$. In the simplest of cases, $B_{flow}(t)$ may be just a constant and known a priori (e.g. the bandwidth of a circuit-switched connection). It is also possible that $B_{flow}(t)$ has a well known statistical characterisation (e.g. a policing function). In this approach $B_{flow}(t)$ is generated by the adaptive network service.

The objective of a rate shaping algorithm is to minimise the conversion distortion, i.e.:

$$\min_{B(t)<B_T(t)} \left\| y(t) - \hat{y}(t) \right\|$$

The attainable rate variation $(\hat{B}/B)$ is in practice limited, and depends primarily on the number of B pictures of the bitstream; no assumption is made on the rate properties of the input bitstream, which can indeed be arbitrary. There are two fundamental ways to reduce the rate:

i) by modifying the quantised transform coefficients by employing coarser quantisation, and

ii) by eliminating transform coefficients.

In general, both schemes could be used to perform rate shaping; requantisation, however, leads to recoding-like algorithms which are not amenable to fast implementation and do not perform as well as selective-transmission ones. A selective transmission approach gives rise to a family of different algorithms, that perform optimally under different constraints; for full details see [Eleftheriadis,95b].

6.3.3 Receiver-Oriented DQM

QoS adaptors, which are resident in the transport protocol at both senders and receivers, arbitrate between the receiver specified QoS and the monitored QoS of the on-going flow. In essence the transport protocol "controls" the progress of the media while the receiver "monitors and adapts" to the flow based on the flow specification and the scaling policy. When the transport protocol is in monitoring mode [Campbell,94] the flow monitor uses an absolute timing method to determine frame receptions times based on timestamps/sample-stamps [Jeffay,92], [Jacobson,93], [Shenker,93]. The flow monitor, as shown in figure 6.7, updates the flow state to include these measured reception times statistics. Based on these flow statistics, the sync-filter (see section 6.3.3.3) derives new playout times used by the media scheduler to adjust the playout point of the flows to the decoding delivery device.

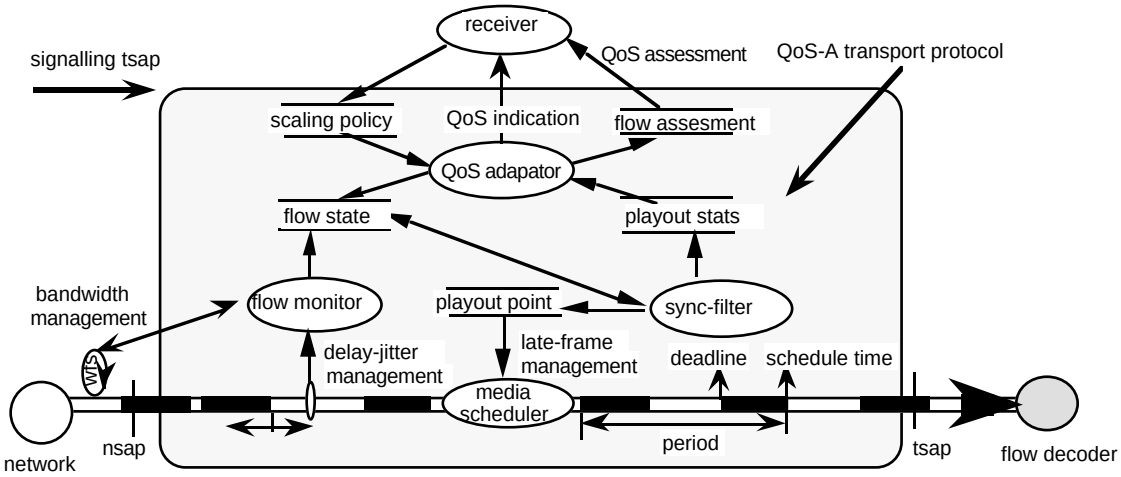

Figure 6.7: Receiver-side transport QoS mechanisms

QoS mechanisms that intrinsically support such adaptive approaches were first recognised in the late 70's by Cohen [Cohen,77] as part of research in carrying voice over packet-switched networks. More recently, adaptive QoS mechanisms have been introduced at part of the Internet suite of application-level multimedia tools (e.g., *vat* [Jacobson,93], *ivs* [Turletti,93], and *vic* [McCanne,94]). Vat which is used for voice conferencing, recreates the timing characteristics of voice flows by having the sender timestamp on-going voice samples. The receiver then uses these timestamps as a basis to reconstruct initial flow, removing any network induced jitter prior to playout. These multimedia tools are widely used in the Internet today and have proved moderately successful - given the nature

of best effort delivery systems, i.e., no resource reservation is made. In the near future, however, an integrated services Internet [Braden,94] will offer support for flow reservation (e.g., RSVP [Zhang,95]) and new QoS commitments (e.g., predictive QoS [Shenker,95]) which are more suitable for continuous media delivery.

Receiver-oriented adaptation can be broken down into a number of receiver-side transport functions, i.e. *bandwidth management, late frame management* and *delay jitter management,* which are described in the following sub-sections (please refer to figure 6.7). This chapter argues that these adaptive QoS mechanisms are inherently part of the transport protocol and not, as in the case of vat, ivs and vic, part of the application domain itself.

6.3.3.1 Bandwidth Management

Bandwidth management receives bandwidth indications in the control message portion of the TSDU (or in separate control messages) and adapts the receiver appropriately. The adaptive service, built on the notion of weighted fair share resource allocation, (see section 6.4.1) periodically informs the receiver that more bandwidth is available or announces that the flow is being throttled back. Bandwidth management only covers the enhancement signals of multi-resolution flows. The base layer is not included since resources are guaranteed to the base layer. The announcement of available bandwidth on a flow allows the receiver to take either a full or partial enhancement layer. The choice depends on whether the flow is in continuous or discrete adaptation mode.

6.3.3.2 Late Frame Management

Late frame management monitors late arrivals in relation to the loss metric and the current playout times and takes appropriate action to trade off timeliness and loss. Packets that arrive after their expected playout points are discarded by the media scheduler and the late-packet metrics in the playout statistics are updated. The media scheduler is based on a split-level scheduler architecture [Coulson,95] which provides hard deadline guarantees to base layer flows via admission control, and best effort deadlines to enhancements layers. Some remedial action may be taken by the QoS adaptor should the loss metric exceed the loss parameter in the flow specification. If the QoS adaptor determines that too many packet losses have occurred over an era, it pushes out the playout time to counteract the late state of packets from the network. Similarly, if loss remains well within the prescribed ranges then the QoS adaptor will automatically and incrementally “pull in” the playout time until

loss is detected.

6.3.3.3 Delay Jitter Management

Our transport protocol utilises *sync-filters* for delay-jitter management by calculating the playout times of flows based on the user supplied jitter parameter in the flow specification[1]. Sync filters calculate the mean and variation in the end-to-end delay based on reception times measured by the flow monitor. Sync filters take the absolute, mean and variation in delay into account when calculating the playout estimate. A smoothing factor based on a linear recursive filtering mechanism characterised by a smoothing constant is used to dampen the movement of the playout adjustment. Intuitively, the playout time needs to be set "far enough" beyond the delay estimate so that only a small fraction of the arriving packets are lost due to late packets. The QoS adaptor trades off late packets versus timeliness based on the delay and loss parameters in the flow specification. The objective of delay jitter management is to pull in the playout offset, while the objective of late-packet management is not to exceed the loss characterised in the service contract. The QoS adaptation manager moderates between timeliness and loss. Based on these metrics the adaptation policy can adjust the damping factor, and acceptable ranges over which the playout point can operate.

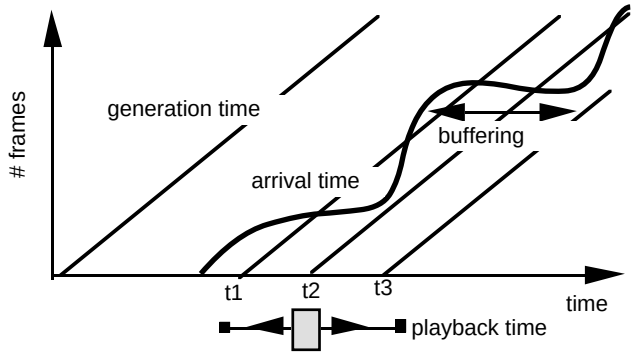

Figure 6.8: Sync-filter: timeliness and packet loss regulation

Figure 6.8 [Zhang,94] shows packets arriving at the receiving end-system. Each packet includes a timestamp used in calculating the flow statistics for delay-jitter management. Selection of the playout point is important: an aggressive playout time which favours

[1] Temporal filters can also operate on multiple related audio and video flows to provide low-level orchestration management (in conjunction with the orch-filter).

timeliness (such as t_1) will results in a large number of late-packets. In contrast, a conservative playout point (such as t_3) will be less responsive and timely but will result in no identifiable packet loss. In the DQM scheme late packets are the same as lost packets, and therefore the loss parameter in the flow specification moderates. An optimum playout schedule is represented by t_2 in the diagram; here, continuous media delivery benefits from timely delivery with the exception of some packet loss - which may be deemed acceptable to the receiver in media perception terms.

6.4 Adaptive Network Service

The adaptive network service provides "hard" guarantees to the base layer (BL) of a multi-layer flow and "weighted fair share" (WFS) guarantees to each of the enhancement layers (E1 and E2). To achieve this, the base layer undergoes a full end-to-end admission control test [Coulson,95]. On the other hand, enhancement layers are admitted without any such test but must compete for residual bandwidth among all other adaptive flows. Enhancement layers are rate controlled based on explicit feed back about the current state of the on-going flow and the availability of residual bandwidth.

6.4.1 Weighted Fair Share Resource Partitioning

Both end-system and network communication resources are partitioned between the deterministic and adaptive service commitment classes. This is achieved by creating and maintaining "firewall" capacity regions for each class. Resources reserved for each class, but not currently in use can be "borrowed" by the best effort service class on condition of pre-emption [Coulson,95]. The adaptive service capacity region (called the available capacity region and denoted by B_{avail}) is further sub-divided into two regions: i) guaranteed capacity region (B_{guar}), which is used to guarantee all base rate layer flow requirements; and ii) residual capacity region (B_{resid}), which is used to accommodate all enhancement rates where competing flows share the residual bandwidth.

Three goals motivate the proposed adaptive service design. The first goal is to admit as many base layer (BL) sub-signals as possible. As more base layers are admitted the guaranteed capacity region B_{guar} grows to meet the hard guarantees for all base signals. In contrast, the residual capacity region B_{resid} shrinks as enhancement layers compete for diminishing residual bandwidth resources. The following invariants must be maintained at

each end system and switch:

$$B_{avail} = B_{guar} + B_{resid}\text{ , and } \sum_{i=1}^{N} BL_{(i)} \leq B_{avail}$$

Our second goal is to share [Steenstrup,94], [Tokuda,92] the residual capacity B_{resid} among competing enhancement sub-signals based on a flow specific *weighting factor, W*, which allocates residual bandwidth in proportion to the range of bandwidth requested that in turn is related to the range of perceptual QoS acceptable to the user. In DQM, residual resources are allocated based on the range of bandwidth requirements specified by the users (i.e. BL.. BL+E1+E2 is the range of bandwidth required, e.g. from 0.32 Mbps to 3 Mbps for the hybrid scalable MPEG-2 flow in Table 1). As a result, as resources become available each flow experiences the same "percentage increase" in the perceptible QoS; this is described as*weighted fair share* (WFS). *W* is calculated for each flow as the ratio of a flow's perceptual QoS range to the sum of all perceptual QoS ranges.

$$W_{(i)} = (BL_i + E1_i + E2_i) / \sum_{j=1}^{N} (BL_j + E1_j + E2_j)$$

All residual resources B_{resid} are allocated in proportion to the *W* metric. The proportion of residual bandwidth is calculated using this factor and allocated to a flow to be $B_{wfs}(i)=W(i).B_{resid}$ and the proportion of the available bandwidth allocated to be $B_{flow}(i)= B_{wfs}(i)+ BL(i)$.

Our third and final goal is to adapt flows both discretely and continuously, based on the adaptation mode. In the discrete mode no residual bandwidth is allocated by the WFS mechanism unless a complete enhancement can be accommodated (i.e., $B_{wfs}(i) = E1(i) \mid E1(i)+E2(i)$, e.g. 0.83 Mbps or 2.68 Mbps from Table 1). In continuous mode any increment of residual bandwidth $B_{wfs}(i)$ can be utilised (i.e. $0 < B_{wfs}(i) \leq E1(i) + E2(i)$, e.g. from 0 to 2.68 Mbps from Table 1).

6.4.2 Rate Control Scheme

I build on the rate-based scheme described in [Campbell,94] where the QoS-A transport protocol at the receiver measures the bandwidth, delay, jitter and loss over an interval called an "era". An era is simply defined as the reciprocal of the frame rate in the flow specification (e.g. for a frame rate of 24 frames per second as shown in Table 1 the interval era is approximately 42 ms). The receiver-side transport protocol periodically informs the sender-side about the currently available bandwidth, and the measured delay, loss and jitter.

This information is used by the source or *virtual source*[1] to calculate the rate to use over the next interval. The reported rate is temporally correlated with the on-going flow. An important result in [Kanakia,93] shows that variable rate encoders can track QoS variations as long as feedback is available within four frame times or less. This feedback is used by the dynamic rate shaping filter and network based filters to control the data generation rate of the video or the selection of the signal respectively. In the case of dynamic rate shaping, the rate is adjusted while keeping the perceptual quality of the video flow meaningful to the user.

Based on the concept of eras, control messages are forwarded from the receiver-side transport protocol to either virtual source or the source-side transport protocol using reverse path forwarding. A core-switch [Ballardie,93] where flows are filtered is always considered to be a virtual source for one or more receivers; for full details see [Aurrecoechea,94]. The WFS mechanism updates the advertised rate as the control messages traverse the switches on the reserve path to the source or virtual source. Therefore any switch can adjust the flow's advertised rate before the source or virtual source receives the rate based control message. The source-side transport protocol hands the measured delay and aggregate bandwidth off (B_{flow}) to the dynamic rate shaping filter.

DQM maintains flow state at each end-system and switch that a flow traverses. Flow state is updated by the WFS algorithm and the rate-based flow control mechanism and comprises:

i) *capacity* (viz. B_{avail}, B_{guar}, B_{resid});

ii) *policy* (viz. filterStyle, adaptMode);

iii) *flowSpec* (viz. BL, E1, E2) ;

iv)*WFS share* (viz., B_{flow}, B_{wfs}, *W*).

The end-systems hold an expanded share tuple for measured delay, loss and jitter metrics. An admission control test is conducted at each end-system and switch on route to the core for the base layer signal. This test simply determines whether there is sufficient bandwidth available to guarantee the base layer BL given the current network load:

$$\sum_{j=1}^{N} BL_{(j)} \leq B_{avail}$$

[1] We use the term *virtual source* to represent a network switch that modifies the source flow via filtering.

If the admission control test is successful, WFS determines the additional percentage of the residual bandwidth made available (B_{wfs}) to meet any enhancement requirements in the flowSpec:

$$B_{wfs(i)} = W_{fact(i)}.(B_{avail} - \sum_{j=1}^{N} BL_{(j)})$$

The WFS rate computation mechanism causes new B_{wfs} rates to be computed for all adaptive enhancement signals that traverse the output link of a switch; switches are typically non-blocking which means the critical resources are the output links, however, the scheme can be generalised to other switch architectures [Coulson,95].

4.6.3 Network Filtering

Currently the scheme supports two types of selection filters in the network. These are low complexity and computationally simple filters for selecting sub-signals. Selection filters do not transform the structure of the internal stream, i.e. they have no knowledge of the format of the encoded flow above differentiating between BL, E1 and E2 sub-signals. The two basic types of section filter used are:

i) *sub-signal filters*: these manipulate base and enhancement layers of multi-layer video multiplexed on a single flow. The definition of sub-signals is kept general here; a flow may be comprised of an anchor and scalable extensions or the I and P pictures of MPEG-2's simple profile, or the individual hybrid scalable profile. Sub-signal filters are installed in switches when a receiver joins an on-going flow;

ii) *hierarchical filters*: these manipulate base and enhancement layers which are transmitted and received on independent flows in a non multiplexed fashion. In functional terms sub-signal and hierarchical filters can be considered to be equivalent in some cases. In sub-signal filtering one flow characterises the complete signal and in hierarchical-filtering a set of flows characterise the complete signal.

In addition, *hybrid filters* combine the characteristics of sub-signal and hierarchical filtering techniques to meet the needs of complex sub-signal selection. For example hierarchical filters allow the BL, E1 and E2 to be carried over distinct flows, and the user can accordingly tune into each sub-signal as required. As an example, the base and enhancement layers of the hybrid scalable MPEG-2 flow are each in turn made up of I, P and B pictures at each layer i.e. BL (I,P,B), E1 (I,P,B) and E2 (I,P,B). Using hybrid

filters, the receiver can join the BL QoS group for the main profile and the E1 QoS group for the spatial enhancement and then select sub-signals within each profile as required (e.g. the I and P pictures of the BL).

6.5 Summary

Resolving heterogeneous QoS demands in networked multimedia systems is a particularly acute problem that has been addressed within the framework of the QoS-A by extending the dynamic QoS management provision to meet the needs of scalable continuous media. As part of that work a scheme for the dynamic management of multi-layer flows in heterogeneous multimedia and multicast networking environments has been described. Dynamic QoS management manipulates and adapts multi-layer coded flows at the end-systems and in the network using a set of scaling objects. The approach is based on three basic concepts: the scalable composition of MPEG standards that can provide discrete adaptation, dynamic rate shaping algorithms for compressed digital video that provide continuous adaptation, and the weighted fair share service for adaptive flows.

Chapter 7

Implementation Details

The focus of the implementation work presented in this Chapter is communication support for adaptive MPEG-1 video flows operating over ATM networks. The implementation details given refer both to the end-system and network domains. In the end-system METS communication support is embedded in the Linux operating system [Linux,93] and interfaces with the Lancaster Research ATM Networking Environment.

The idealised environment (i.e., Chorus with an ATM interconnect) was not available during the experimental phase of this work. The Linux environment with ATM interconnect is considered to be flexible enough to realise a number of the important aspects of resource management presented in this Chapter 5 - on the idealised operating system support for QoS-A. Linux is distributed free of charge and supports Xwindows system (X11R6) and standard networking support including TCP/IP.

In the network, communication support is embedded in the ATMos [French,93] operating system resident in Lancaster Research ATM switches. To evaluate the implementation a networked video system (NVS) [Yeadon,95], based on the Berkeley software MPEG video player [Rowe,91], has been developed. In the next Chapter the performance impact of the architecture is experimentally is demonstrated using the NVS video system running on PCs and RAID-3 based storage servers interconnected by ATM switches.

This Chapter is structured as follows. Section 7.1 presents the QoS-A experimental environment. The key end-system and network implementation modules are introduced, and the author's contribution to the implementation work is given. Following this, section 7.2, presents a QoS configurable based socket API. Then, section 7.3 describes the implementation of METS transport system. Finally, this Chapter concludes with a summary of the implementation status, evaluation of the implementation follows on in Chapter 8 and conclusion drawn in Chapter 9.

7.1 Experimental Environment

The objective of the METS transport system, as illustrated in figure 7.1, is to make QoS visible at the transport API. This is accomplished by preserving application level guarantees throughout the end- systems and the network on an end-to-end basis. Enabling applications directly to access ATM level QoS is fundamental in meeting end-to-end QoS guarantees. ATM offers high bandwidth per connection QoS guarantees at the physical layer; making it suitable technology for multimedia networking. In many cases, however, the QoS benefits of ATM are hidden from the application and transport layers by an interworking layer (e.g., IPv4) rendering per connection QoS configurability and control impossible [Keshav,94]. Furthermore, many ATM networks do not, as yet, provide QoS configurability on a per connection basis. Rather, they allow the establishment of switched virtual circuits with best effort QoS making per connection control and management impossible. The objective of the implementation work presented here is the provision of QoS configurability, control and management on an end-to-end basis.

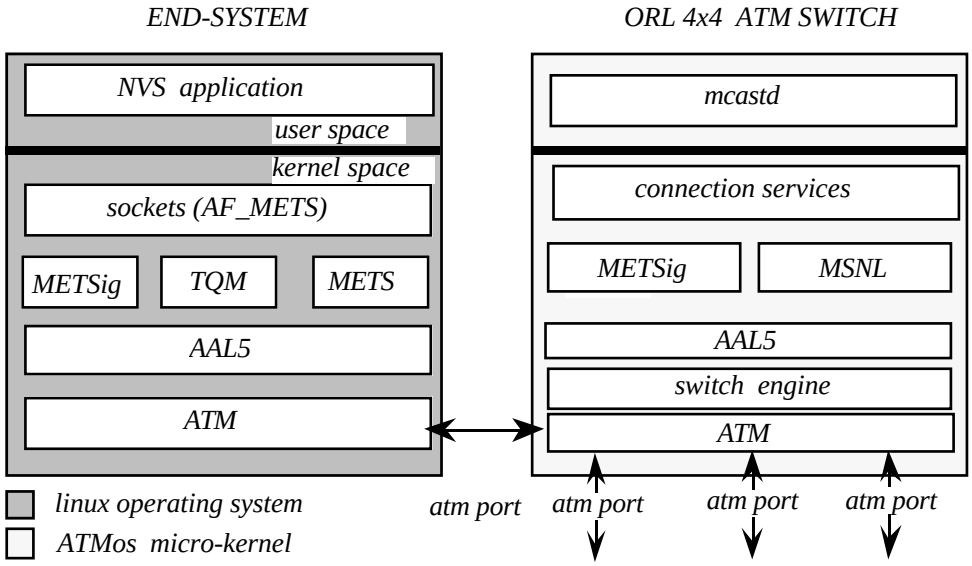

Figure 7.1: QoS-A Testbed Environment

7.1.1 End-System

The end-system implementation modules include:

i) a *QoS configurable Berkeley socket based API*,

ii) a *METS transport system*,

iii) a *ATM networking interface controller*. and

iv) a *networked video system (NVS)*.

The author's contribution to end-system implementation includes:

i) the *modification* to the kernel socket code to support the QoS configurable API,

ii) the *implementation* of METS transport systems,

iii) the *enhancement* of the ATM networking interface controller software [Lunn,95], and

iv) the *modifications* of the NVS client server networking interface [Yeadon,95] to take full advantage of the METS QoS configurable socket interface.

The QoS configurable API is based on a new protocol family called AF_METS. By preserving compatibility with the current Berkeley socket API such as AF_INET existing applications can run unchanged. In addition, the METS protocol stack operates aside an IP environment [Lunn,95] based on standard IP over ATM encapsulation [RFC 1483,93]. In this role, however, per flow QoS is not visible above the IP layer. The API is discussed in full detail in section 7.2

The METS transport system implementation is comprised of a set of communication, control and management modules to support application level QoS requirements. Based on the principles of separation between control, maintenance and management the METS transport system includes of the following four modules:

i) a *flow manager (FM) module*, which is responsible for the flow management, functions of flow establishment, QoS signalling, QoS adaptation and tear down;

ii) a *signalling (METSig) module*, which includes group management, connection management and signalling support for flow establishment and dynamic QoS management;

iii) a *QoS manager (TQM) module*, which is responsible for the QoS maintenance of on-going flows; and

iv) a *METS protocol (METSP) module*, which provides an MPEG encapsulation for

audio and video flows and a set of transport level resource management QoS mechanisms for flow scheduling, flow monitoring, flow control, flow shaping and jitter correction (i.e., sync filtering).

The transport system is detailed in section 7.3

7.1.1.1 Interfacing to ATM

The networking infrastructure used to implementation the METS transport system is based on the Lancaster Research ATM Networking Environment. This delivers ATM flows to a mix of PCs, storage servers, and multimedia devices designed at Lancaster [Lunn,94]. The Lancaster Research ATM Networking Environment is interconnected to a Campus-wide ATM network based on Fore Systems (i.e., ASX-100 and ASX-200 ATM switches) and ATM Ltd (i.e., Virata ATM switches) technology and to the rest of the UK via the SuperJANET 34 Mbps Joint Academic Network (SuperJANET). QoS-A implementation and evaluation (see Chapter 8) are based solely on the local ATM networking.

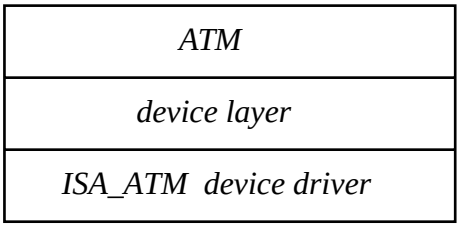

Figure 7.2: ATM Layer

The lower layers of the QoS-A communication architecture are based on an implementation of AAL5 adaptation layer over ATM. The purpose of the AAL5 layer is to provide a simple and efficient encapsulation for communications between end-points. An end-point is a transport service access point in this model. The AAL5 provides a link level service between end-points and acts as an interface between the service requirements of METS and the ATM layers. The AAL5 layer is divided into two sub-layers: the segmentation and reassembly sub-layer (SAR) and the convergence sub-layer (CS). At the transmitting end-point, METS packets are simply encapsulated into AAL5 Service Data Units (SDUs) and then the SAR performs segmentation of AAL5 SDUs into ATM cells. At the receiving end-point the inverse operation is performed.

As illustrated in figure 7.2, the ATM layer is comprised of three sub-layers in the Linux

implementation:

i) the *ISA ATM device driver layer*, which is responsible for the initialisation and control of the Olivetti Research Laboratory (ORL) ATM *Network Interface Controller (NIC)* and the transmission and reception of ATM cells;

ii) the *device layer*, which is provided by the Linux operating system and decouples protocols from devices keeping the layering of protocols generic and flexible; and

iii) the *ATM protocol layer;* which performs a minimum set of ATM functions.

Network Interface Controller (NIC)

The ORL NIC, which is designed to interface to 100 Mbps ATM, shares its port design with the 4x4 ORL ATM switch. However, instead of interfacing with a DMA engine and ARM processor, the NIC interfaces with the standard PC ISA bus architecture.

As illustrated in figure 7.3, the ISA bus interfaces to a Xilinx controller and the transmit and receive FIFO buffers. The FPGA provides the primary control for interfacing with the PC and transferring data between the TAXI chips and the FIFOs. The NIC has two 16 bit registers for control and provision of status information. The control register enables and disables interrupts for transmission, reception and connection status. The status register indicates various device states such as cells received, space available in the transmission queue and interrupt flags.

The NIC also contains hardware support for ATM header checksum (HEC) processing. When ATM cells are transferred to the NIC device 4 bytes of the ATM header are passed followed by the 48 data byte payload. The NIC calculates the HEC and places it into the header. On reception of ATM cells the HEC is checked and flags in the control register indicate the status of the ATM header information.

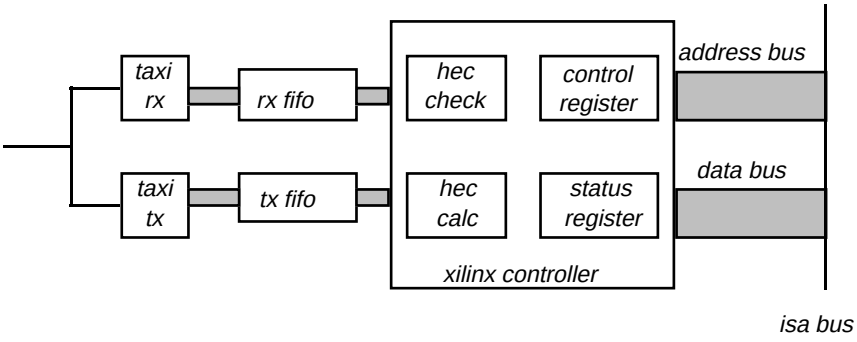

Figure 7.3: ISA ORL ATM NIC

7.1.1. MPEG Application Level Demonstrator

The QoS-A application level demonstrator is part of the Lancaster Filtering System (LFS) [Yeadon,95] developed to experiment with filtering audio and video flows in the Internet. The focus of the LFS work is the investigation of appropriate filtering mechanisms to meet specific QoS needs of individual clients. To date, the LFS work has focused on the manipulation of MPEG-1 and motion-JPEG video flows both in the compressed and semi-compressed domains. The LFS video MPEG decoder is based on the Berkeley software Continuous Media Player [Rowe,92]. The suite of LFS filtering mechanisms comprises codec, frame-dropping and re-quantization filters.

For the purpose of this research the LFS has been ported to the native ATM networking environment and is used in the capacity of an application level demonstrator. In this role the LFS systems is primarily used as a source and player of MPEG video clips as illustrated in figure 7.4.

The NVS *file daemon* listens on a well known port for a new MPEG video-on-demand request from clients. When a request is received the LFS system invokes an *MPEG-1 video agent* (i.e., video-on-demand server) which packetises and transmits MPEG-1 video using the METS transport system. Each MPEG-1 picture (viz. I, P, B pictures) is packetised into individual METS packets. These assist with the adaptive service described in sections 7.3.14 (cf. QoS adaptor) and 7.3.3.1 (DQM). As illustrated in figure 7.4, the video server and clients have been modified to utilise control sockets (conSoc) for application level signalling, media sockets (medSoc) for the transfer of continuous media streams and a flow management sockets (manSoc) for the management of on-going flows.

The NVS system employs the AF_METS socket family (to be described in detail in

section 7.2) rather than the AF_INET Internet family. MPEG-1 sequence header data and *group of pictures (GOP)* data are also packetised independently and uniquely identified. METS packets are carried over the ATM network as encapsulated AAL5 packet. The mapping of METS to AAL5 packets is one-to-one; that is, there is no segmentation between METS and AAL5 layers. Depending on the configuration, the LFS may send sequence headers and GOP header packets in-band on a medSoc or out-of-band a conSoc.

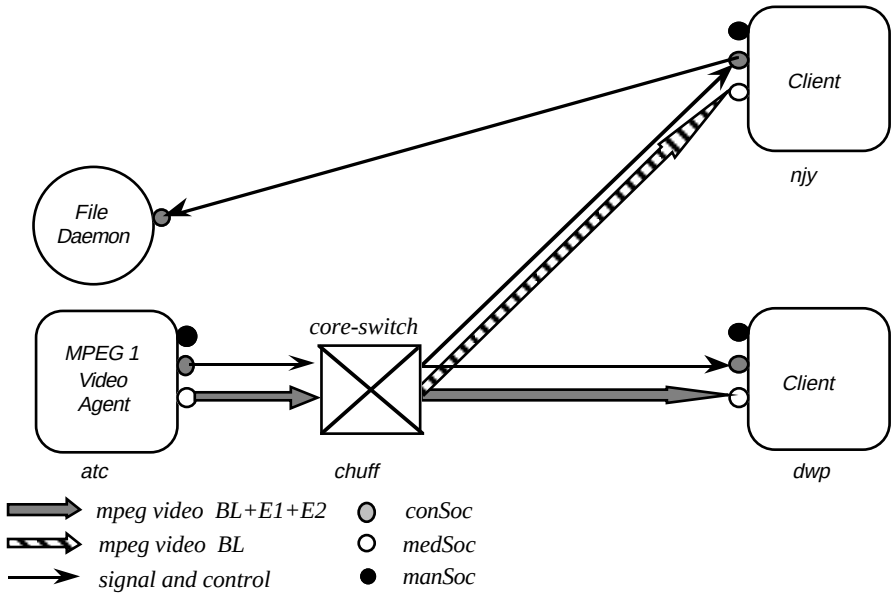

Figure 7.4: NVS Application Level Demonstrator.

Figure 7.4 shows two clients (located at the *njy* and *dwp* end-systems) and a server (located at *atc* end-system) interact. In the first instance, the client at atc interacts with the file daemon to request dissemination of client selected MPEG-1 video. Following this, the server interacts with the METS system to create a *QoS group* in the first instance, and then advertises the traffic characteristics and QoS policy associated with the on-demand video. The server subsequently joins the created *multicast session* and finally disseminates the requested video using the designated multicast address. Clients also interact with group management to determine the nature of the source video (e.g., the structure, coding and QoS attributed to each component of a video signal). From this interaction they choose which video components to consume. Once clients have chosen their preferences they also issue *joinFlow* commands on the designated multicast address (which is made up of a port and a MSNL address). At this point the clients are ready to consume their selected video

components.

7.1.2 Network

The software structure of each ATM switch is comprised of the following:

i) an *ATM device driver* module, which is responsible for the reception and transmission of cells;

ii) a *switch engine* module, which forwards cells directly or pass them to the higher layers for further processing;

iii) a *AAL5* module, which provides AAL5 layer functions and which forwards AAL 5 protocol data units (PDU) to the signalling layer or directly to the connection service layer;

iv) a *signalling layer*, which provides MSNL and METS signalling supports for connection management of switch resident applications processes;

v) a *connection service layer*, which implements ATMos connection management handling connection setup, data delivery and tear down; these functions are equivalent to the Berkeley socket layer API; and

vi) an *"application layer"*, which represents the ATMos application level process environment (e.g., it includes the METS *multicast daemon* for CBT signalling).

The author's contribution to this network implementation includes:

i) the *modification* to the switch engine module to support the adaptive network service described in Chapter 6;

ii) the *implementation* of METS signalling functions described in Chapter 4 and 6, and

iii) the *implementation* of a application layer multicast daemon for CBT signalling as described in Chapter 4 and 6.

The network implementation utilises ORL NICs and 4x4 ATM switches as the basic interconnect for PC, RAID-3 storage servers and multimedia peripheral devices. Each ORL device (e.g., NICs, switches, etc.) is comprised of standard component parts: the hardware consists of an ARM processor, up to 32 MBytes of memory and a 100 Mbps ATM TAXI

interface. These modules are used to implement both ORL ATM switches (i.e., 4x4 and 8x8 port switches) and end-point modules such as the RAID-3 storage servers. All ORL modules are controlled by the ATMos micro-kernel specifically designed to support ATM switch and multimedia peripheral operations. ATMos provides a mechanism for scheduling processes and controlling low level hardware and was designed by Leslie French and Ian Wilson of ORL Ltd [French,94] . The ATMos environment includes provision for inter-operation with other systems using protocols such as TCP/IP, XTP and a distributed platform based on CORBA [OMG,93].

The ATMos development environment has been ported to Lancaster's Linux machines allowing the development of switch software on Linux PCs [Lunn,95]. New switch software is built on the Linux PCs and booted onto the ATM switches using the ORL boot protocol [French,94].

7.1.2.1 Switch Software Structure

The MSNL module implements the ORL network layer - implementing level 3 of the ISO 7 layer model - derived from the Multi-Service Network Layer (MSNL) of the Multi-Service Network Architecture developed at Cambridge Computer Laboratory by McAuley. The protocol is connection-oriented and is built on a set of meta-signalling commands and is primarily used within the ATMos ATM network at ORL.

The METS system builds on this work by using a number of the meta-signalling commands and hooks provided by ATMos for the implementation of signalling. METSig is fundamentally a connection-oriented protocol supporting multicast communications with QoS guarantees as described in Chapter 6.

As described above clients and servers obtain multicast addresses which identify *sessions* and then issue *joinFlow* commands to join a particular flow. This results in the creation of a switched virtual circuit between the clients and server and *multicast daemon (mcastd)* located at one of the switches in the network. Clients 'rendezvous' with a server at the designated switch which is administered by a multicast daemon at the designated switch; this is called the *core-switch* for the multicast session.

The communication layering illustrated in figure 7.1 is implemented by 4 ATMos processes:

ii) the *ATM, switch engine* and a *AAL[1] layers* are implemented by a single ATMos process;

iii) the *METSig* and *MSNL* signalling functions are implemented by two independent processes;

iii) the *connection services* which is equivalent to end-system socket library functionality; and

iv) the *mcast daemon* listens on a well known port (which is part of a multicast group address) for METSig meta-signalling and dynamic QoS management messages.

Implementation of the METS transport system required a minor modification to the switch engine code, and the addition of METSig and mcast processes to the switch. All other items above are standard ORL release (e.g., ATMos, connection services, MSNL) which required no modification.

7.1.2.2 4x4 ATM Switch Hardware Structure

The ORL 4x4 port ATM switch contains 4 identical port controls, a DMA engine and an ARM processor (Figure 7.5). The port controls implement a 100 Mbps TAXI interface with FIFOs to queue 13 ATM cells on transmit and receive. These FIFOs interface with the DMA engine which transfers cells from the incoming queue to the outgoing queue under control of the ARM processor. The ARM examines the header of each ATM cell, performs the header remapping and routing to a port or higher layer (e.g., METS signalling).

The philosophy behind the design of such a small switch has been influenced the work of Hayter, McAuley and Leslie on Desk Area Networking [Hayter,91]. Rather than exploding the workstation, as is the DAN approach, ORL chose to treat each peripheral module (e.g., PC, ATM Camera, storage server, ATM video, ATM microphone, etc.) as an ATM device connected to a desk area switch which is external to the workstation; essentially using a network as the peripheral interconnect. The ORL multimedia model considers that most flows emanating from a module typically flow to multimedia devices other than the PC or workstation

[1]The switch implements a standard AAL5 for signalling and data transfer with application level processes resident at a switch. Therefore, connections can be terminated at the switch - and as is traditional, passed through the switch.

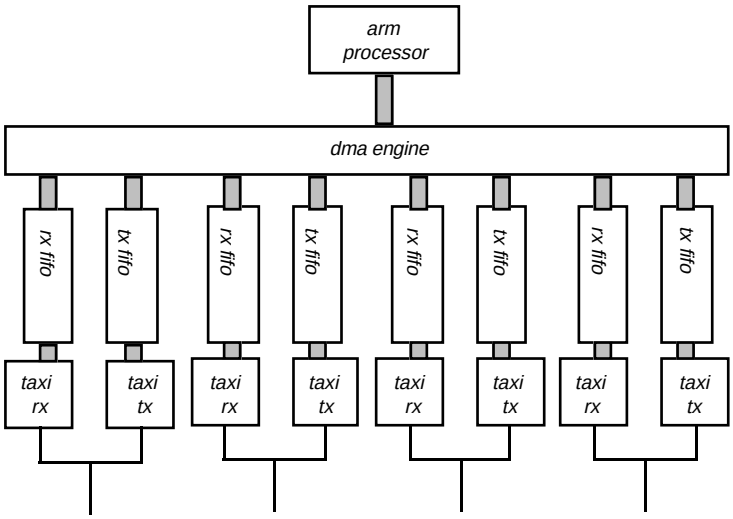

Figure 7.5: 4 x4 Port ORL ATM Switch

7.1.3 Experimental Configuration

The METS testbed consists of 4 Linux based PC and two ATMos based RAID-3 storage servers. Each PC and storage server is equipped with an ORL ATM NIC for interconnection to the ATM network. As illustrated in figure 7.6 the testbed included the following nodes: atc and dwp end-systems, which are both 90 MHz Pentium machines, and mr-little and njy which are both 66 MHz 486 machines. The RAID-3 storage servers (magpie and scaup end-systems as shown in figure 7.6) utilise the ARM 610 RISC processor and are comprised of 5xSCSI interface controllers and disk drives .

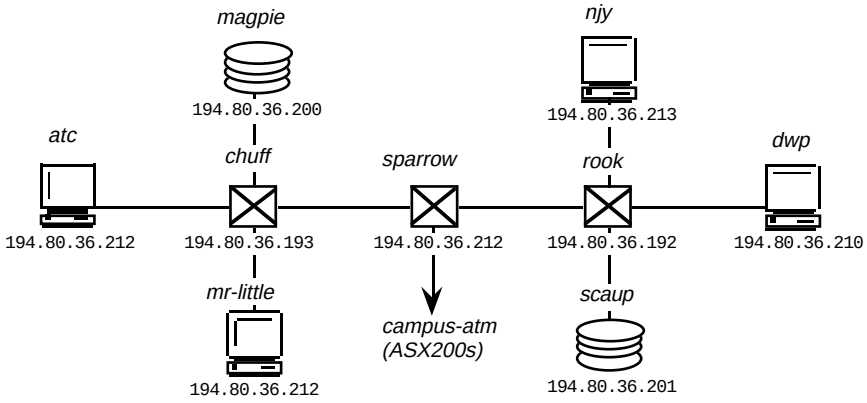

Figure 7.6: METS Testbed

Each PC and storage server is connected to one of three ORL 4x4 ATM switches:

chuff, sparrow and rook in figure 7.6.

7.2 Application Programmers Interface

The METS API consists of a set of Berkeley socket calls in addition to some new extension. The original socket code was tailored to meet the needs of the METS API first described in Chapter 4 and subsequently extended in Chapter 6. Extensions to the existing socket API benefited the needs of the QoS-A for the control, maintenance and management of QoS.

7.2.1 Overview

The METS API consists of a set of Berkeley socket calls in addition to some new primitives for group management. Note that where possible the author has used the standard socket library calls.

The METS API is comprised of the following set of primitives:

i) *group management* primitives, which allow the user to open a group, get group information and gracefully close a group;

ii) *socket* primitives, which use standard Berkeley socket interfaces to allow applications to create flow management, control and media sockets;

iii) *connection management* primitives, which allow the application to join and leave flows;

iv) *media transfer* primitives, which consists of the standard Berkeley socket interfaces for sending and receiving data; and

v) *flow management* primitives, which allow applications to change QoS policy and flow specification of flow;

The METS system provides three styles of sockets (illustrated in figure 7.8) as part of the AF_METS protocol family to the application layer.

These include:

i) a *media socket (MedSoc),* which is used for the transfer of continuous media, is QoS configurable and maintainable based on a user supplied flow and QoS policy specifications, respectively. Media sockets are characterised as being simplex and

non-assured in nature;

ii) a *control socket (ConSoc)*, which is used for the transfer of application level control information. Control sockets are full duplex and assured (i.e., provide a reliable delivery service); and

iii) a *flow management socket (ManSoc)*, which is used for the management of media flows. Flow management sockets are unique in that they provide applications an interface to QoS maintenance.

All devices on the ATM network use MSNL addressing rather than E.164 or NSAP as prescribed in UNI 4.0 [ATMF,92a]. The MSNL addressing information is carried in 8-octet a 4-octet MSNL address and a 4-octet port. Both MSNL addresses are usually represented 4 decimal numbers, separated by dots, like IP address (e.g., as illustrated in figure 7.6 the switch named *'chuff'* has an MSNL address 194.80.36.193). Certain numbers represent well known ports to METS signalling. For example, all the mcast daemons listen on port 0.0.0.60 on all end-systems and switch devices. In the METS system multicast addresses are based on MSNL unicast addresses of the core-switch and a port address. The designated core-switch is part of the core based tree (CBT) routing protocol used during flow establishment. The benefit of such an approach is that unicast and multicast addresses are the essentially the same [Ballardie,93] in CBT addressing.

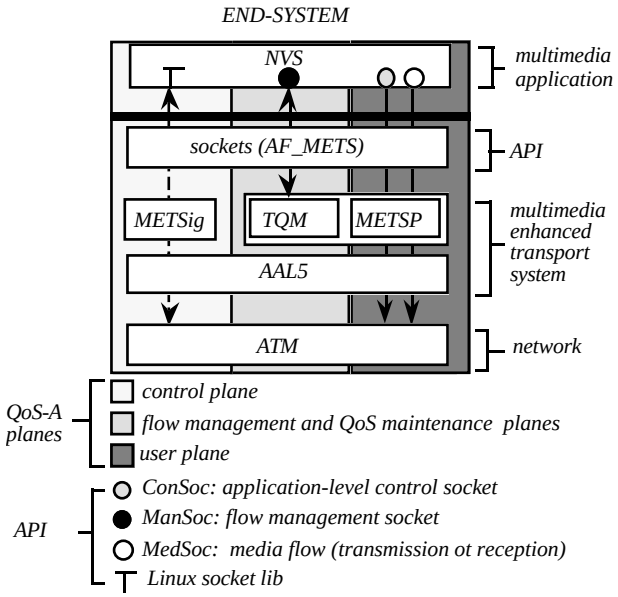

Figure 7.8: Mapping to METS

In the following section each set of API primitives (described above) are presented

indicating any differences that may exist between the client and server APIs.

7.2.2 Group Management Primitives

Items i) through v) outlined above in section 7.2.1 broadly characterise the order in which multicast switched virtual circuits are established. In the first instance, a server creates a QoS group by issuing an *openGroup* command on the manSoc specifying its flow specification and QoS policy - this is described as the server's *QoS profile*. Multicast group addresses must be known to all clients a *priori*. Furthermore, for the purpose of experimentation a multicast daemon at the core recorded the QoS profile and await *getInfo* requests from clients. This inevitably creates a bottle neck in the signalling system as the number of group members scale up. To resolve this problem the server's profile details and the 'state' of the group would need to be distributed and some form of a hierarchical management of state would be required. These scaling issues are for future work.

During the After a client has determined the server's QoS profile and selected the component part of the flow to consume they are then free to establish a connection to a multicast switched virtual circuit. In addition to requesting information about the server's profile, clients and servers may also retrieve detailed statistics about membership to a particular group.

Only servers which instantiate QoS groups have the privilege to close QoS groups. Before the session can be finally closed all currently joined members are issued a leave indication (see section 7.2.4). The group management primitives are described in figure 7.7.

Group Management Primitive	Parameters
Open Group Req	Multicast Address, QoS Profile (QoS Policy, Flow Spec)
Open Group Ack	None
Open Group Nak	None
GetGroupInfo Req	Multicast Address, Type (members | profile | all)
GetGroupInfo Ack	Type Data (members | profile | all)
GetGroupInfo Nak	None

Open Group Req	Multicast Address
Open Group Ack	Numbers
Open Group Nak	None

Figure 7.7: Group Management Primitives

7.2.3 Socket Primitives

The METS API uses the standard Berkeley sockets calls to:

i) *create* flow management (SOCK_RAW) sockets (manSoc), media (SOCK_DGRAM) sockets (medSoc) and control (SOCK_STREAM) sockets (conSoc) ;

ii) *bind* a port and address to a socket;

iii) *send* and *receive* control, flow management and *media* to and from a socket;

iv) *close* a socket;

v) *get host by name* details; and

v) *select* to wait on a socket or timer event.

Figure 7.8 illustrates the three styles of socket, controlling mechanisms and mapping to the METS transport system. Once a server has successfully established a switched connection it moves into the media transfer state and is free to commence the media transfer phase (i.e., free to produce or consume media). The current implementation allows the server to send media to the designated core switch in the absence of clients. This approach fully decouples the server and clients which is an important principle of scalability. Clients join the multicast session and receive media.

7.2.4 Connection Management Primitives

The connection management primitives allow servers and clients to join and leave multicast groups once a multicast group has been established using the group management primitives. Both the servers and clients state their flow specifications and QoS policy requirements in the *joinFlow* command. The joinFlow command distinguishes between clients and servers in order for the correct polarity of a connection to be established. Clients and servers can leave the session at any point by issuing a leaveFlow command on their manSoc. The current number of members joined to the connection is returned in the join

and leave acknowledgement. The connection management primitives are described in figure 7.9.

Connection Management Primitives	Parameters
joinFlow Req *joinFlow Ack* *joinFlow Nak*	Type (Server | Client), Multicast Address, QoS Policy, Flow Spec Members Reason
leaveFlow Req *leaveFlow Ack* *leaveFlow Nak*	Multicast Address Member Reason

Figure 7.9: Connection Management Primitives

7.2.5 Flow Management Primitives

The flow management primitives allow servers and clients to interact to register a flow, *change* the QoS of a flow and receive *QoS signals* associated with a particular flow. Whenever clients and servers create media sockets they register them (i.e., medSocs) using the *register* primitive. This allows flow management to interact with the application over the associated manSoc providing monitoring and maintenance information about the on-going flow.

At any point during a session, group members may change the QoS negotiated during the connection establishment phase. The *changeQoS* options include modification of the QoS policy and flow specification clauses. Similar in nature to the Berkeley setsockopt, *changeQoS* allows the application to affect the performance, control and management requirements of the medSoc. The flow management primitives are described in figure 7.10.

Flow Management Primitives	Parameters
registerSoc	ManSoc, MedSoc

changeQoS Req *changeQoS Ack* *changeQoS Nak*	ManSoc, Options (QoSPolicy | FlowSpec | Maint | Monitor | Signal | Adapt | Filter | Event), structure (QoSPolicy | FlowSpec | Maint | Monitor | Signal | Adapt | Filter | Event) sizeof (QoSPolicy | FlowSpec | Maint | Monitor | Signal | Adapt | Filter | Event) Status
signalQoS	type (Signal | Event) option (qosMetric | qosEvent)

Figure 7.10: Flow Management Primitives

The *changeQoS* options comprise:

i) *QoSPolicy* option, for specification of a new QoS policy;

ii) *flowSpec* option, for renegotiation of a current flow specification; and

iii) *maint*, *monitor* and *signal* refer to the QoS maintenace options:

- in *maint* mode the transport QoS manager actively maintains the flow,

- in *monitor* mode it maintains the flow but also forwards periodic QoS signal ; messages to the application over the manSoc;

- in *signal* mode it does not maintain the flow but forwards periodic QoS signal messages;

iv) *adapt* option, to change the adaptation mode (viz. continuous or discrete); and

v) *filter* option, to select new filters (viz. sync at the receiver or picture dropping filter in the network).

While a change in QoS initiated by a client only affects the local clients QoS a change by the server may impact all active clients in the current session. If the *monitor, signal* or *event* options are selected then the application expects to receive signalQoS. The monitor and signal options allow the application to specify an interval over which the transport flow monitor measures specific QoS metrics. At the end of the user specified interval flow

management issues a qosSignal to application (via the ManSoc) providing the performance details of the QoS metric monitored over the interval. QoS signals initiated by the selection of the monitor or signal options are synchronous in nature. In contrast QoS signals which are the result of event selection are asynchronous. The event option allows applications to attach alarms (or *QoS alerts* [OSI,95a]) to the occurrence of particular event thresholds. In this role flow management continuously monitors the selected QoS metrics against the specified threshold values. Should the threshold be exceeded then flow management informs the application of this event over the manSoc.

7.3 METS Transport System

This section describes the detailed the implementation of the METS transport system. This includes support for flow scheduling, flow shaping and jitter correction. The impact of QoS control, QoS maintenance and flow management on the end-system and ATM switches are addressed in detail. An important contribution of this thesis is the *realisation* of a set of QoS mechanisms designed to meet the transport needs of adaptive video operating over ATM networks. Implicit in the design of these mechanisms is their ability to distinguish between different QoS needs of flows in the end-system and as they traverse the network.

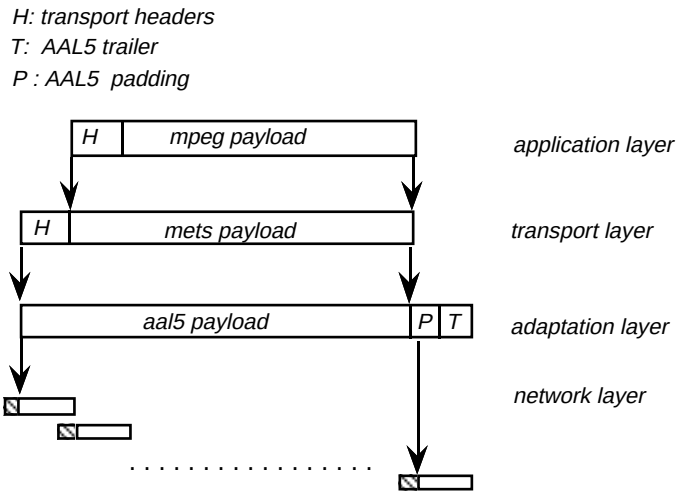

Figure 7.11: Encapsulation and Layering

Figure 7.11 illustrates the encapsulation used by the transport system. The NVS system packetises MPEG-1 video and then the transport system encapsulates each packet into a

single METS service data unit. Note that the MPEG picture boundaries are maintained by each METS and AAL5 service data unit. Each METS packet is directly encapsulated into a single AAL5 service data unit by appending the trailer and adding padding bytes for fragmentation into ATM cells. Encapsulation is kept to a minimum and as such is efficient. The maximum size AAL5 service data unit adopted for experimention is 32 KBytes which is suitable to hold the largest I frame used during the experimental phase discussed in Chapter 8.

The transport system as illustrated in figure 7.12 is comprised of four implementation modules which map closely to the QoS-A planes. The flow management plane is implemented as a *flow manager (FM)* and a *resource reservation* module. The resource reservation module subsumes the QoS mapping and admission control testing functions of the QoS-A flow management plane. The resource reservation module interacts with the CPU, memory and network resource management modules for the allocation of communication resources. The *connection manager* handles aspects of network resource management. The *flow manager* interacts with clients and servers over the manSoc interface using a set of kernel level socket functions as illustrated in figure 7.12. The control plane is realised as a set of meta-signalling, group management, connection management and dynamic QoS management (DQM) signalling functions.

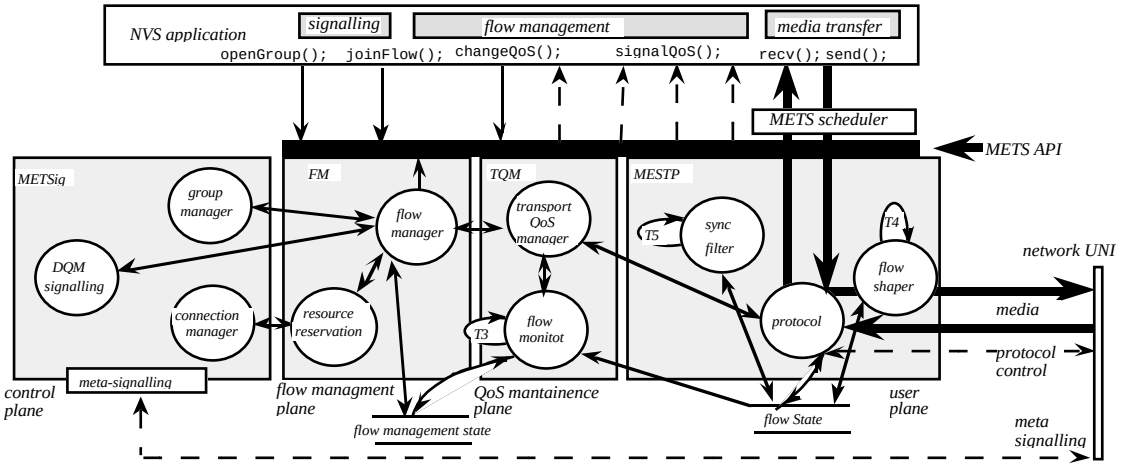

Figure 7.12: METS Transport System Components

The transport system provides a number of protocol and QoS mechanisms for end-to-end traffic control of flows. These include METS flow scheduling for transmission and reception of media, flow shaping on transmission of media and flow monitoring and jitter

correction (sync-filtering) on the reception of media. METS also provides support for dynamic QoS management of multi-layer code flows.

METS Modules	Signalling	QoS Control	QoS Maintenance	Flow Management
Management Entity	*connection manager* *group manager*	*protocol*	*transport QoS manager*	*flow manager*
QoS Mechanisms	*admission control* *resource reservation* *DQM signalling*	*flow shaper* *flow scheduler* *sync filter* *QoS monitor* *QoS adaptor* *protocol engine*	*maintenance* *flow monitor* *event monitor*	*adaptation* *QoS mapping* *admission control* *resource reserva*

Figure 7.13: End-System METS Modules

The end-system QoS control, QoS maintenance and flow management METS modules are now described in turn. The management entity and associated QoS mechanisms for each of these modules in discussed in detail. Following this, section 7.3.4, examines the implementation of the adaptive network service interworking with end-system QoS adpaters and network media schedulers to support dynamic QoS management of multi layered coded flows.

7.3.1 QoS Control Module

7.3.1.1 Flow Scheduling and Shaping

The approach taken to delivering per flow bandwidth and jitter guarantees at the application layer is based on a two stage scheduling framework as illustrated in figure 7.14. The first stage of the framework employs a transport level *flow scheduler* that schedules application level frames [Clark,90] on a *frame per second (fps)* basis. The second stage of

the scheduling framework is responsible for pacing or regulation of ATM cells to the user-network-interface. Flow scheduling and shaping work in unison to provide scheduling and rate control in the METS system. Because QoS requirements are visible to both the scheduler and shaper quality of service guarantees can be provided on an individual flow basis. The flow scheduler and flow shaper are examined in the following sections.

Flow Scheduler

The flow scheduler is implemented as part of the *flowLib* user space library and applies equally to the transmission and reception of media to and from the network. The flow scheduler uses the standard Linux clock timer to dispatch application level frames to the METS transport protocol. Application level frames are processed by the protocol and then paced onto the next flow shaper stage according to the flow specification. As shown in figure 7.14, the scheduler provides variable bit rate service by isochronous scheduling variable sized packets to the transport system. The flow scheduler uses the standard Linux clock timer to dispatch application level frames to the METS transport protocol. Application level frames are processed by the protocol and then paced onto the network according to the flow specification. As shown in figure 7.14, the scheduler provides variable bit rate service by isochronously scheduling variable sized packets to the transport system.

The flow scheduler polices the rate at which applications deliver and receive frames to and from the METS layer, respectively. The flow scheduler does not, however, police the size of packets exchanged at the transport service access point (i.e., medSoc). This is a function of the lower level flow shaping mechanism on transmit. The flow scheduler enables the application to transmit and receive one application level frame every $1/fps_i$ interval, where i represents the flow identifications. Any deviation from this isochronous rate (i.e., faster or slower) is managed by the scheduling mechanism. While the *workahead mode* (described in Chapter 5) is inhibited, the scheduler includes mechanisms which attempt to overcome any slippage in the agreed rate that may occur at the transmitter. At the receiver, the flow scheduler relies on the sync filter (described in section 7.3.1.2) to provide the schedule deadlines. In this role the sync filter adjusts the deadline of delivered frames but not the rate.

The Linux scheduler can not offer the same support for QoS commitment as the *idealised* operating system (i.e., delivered by the enhanced Chorus scheduler described in Chapter 5). Linux provides a best effort scheduling mechanism which cannot consistently deliver hard real-time guarantees to applications if resource utilisation remains unbounded.

It can, however, offer soft guarantees on a more consistent basis; this is again dependent on load on the system. By introducing admission testing for communication flow, resource utilisation can be bounded in the case of the transmission and reception of media. However, the flow scheduler still relies on the Unix scheduler to dispatch media from the application to the transport system. One product of this environment is *slippage* or *drift* can occur at the flow scheduling level. This occurs when a deadline is missed by some margin. To counter this the flow scheduler is designed to take slippage into consideration during the calculation of frame deadlines [Simpson,95].

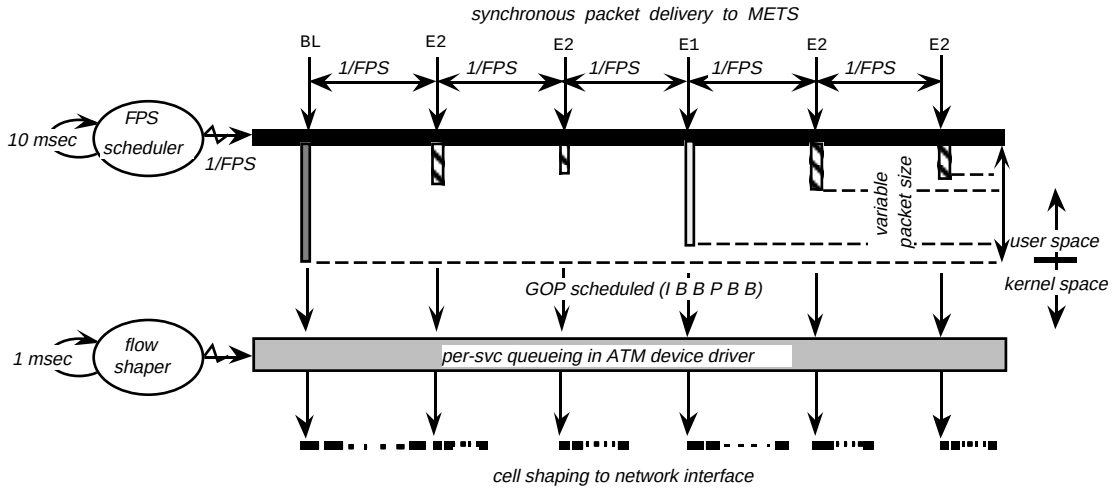

Figure 7.14: Two Stage Scheduling Framework

A drift compensation function is built into the flow schedules which takes into account any missed deadlines. If a deadline has been missed then the flow scheduler immediately allows the application to transmit or receive media. The duration of the next scheduling opportunity is then calculated and takes any drift in the isochronous rate of the transmitter into account. Taking drift into account implies that the interval between the last point of scheduling and the next deadline is less than $1/fps_i$. In some server cases complete scheduling opportunities can be missed by an interval greater than $1/fps_i$. In this case the flow scheduler keeps a tally of lost opportunities and credits the application during the following scheduling opportunity where the backlog is dealt with.

The flow scheduler keeps track of any missed deadlines and informs flow management should the number of missed deadlines exceed a pre-defined missed deadline threshold. As described later flow management upcalls applications (via a *loaded* QoS signal on the manSoc) to inform them of the loaded state.

QoS-A applications are designed only to forward base layer frames in a loaded state. when congestion has cleared flow management informs the application via a QoS event signal to resume its original rate. The flow scheduler does not discriminate between BL, E1 and E2 frames (i.e., it does not understand the semantics of the adaptive service or layered flows). It relies on the application to *source quench flows* in the loaded state. Additionally, the protocol perform any policing required should the application be deemed not to co-operate with the flow control directive.

Flow Shaper

The flow shaper is implemented as part of the kernel level ISA ATM device driver. Flow shaping only applies to the regulation of media as it is injected into the network. The flow shaper code is invoked every 1 ms using a dedicated hardware timer. The flow shaper provides open loop flow control based on a token bucket scheme that paces cells to the network interface. It constitutes the low level flow regulation component of the scheduling framework. The token bucket scheme is a variant of the leaky bucket algorithm [Turner,86]. As illustrated in figure 7.15, flows accumulate *credits* which represent the number of ATM cells that can be transmitted to the network over the next interval.

The flow shaper maintains per flow state:

i) the *token budget (b),* which represents the capacity or depth of the token bucket measure;

ii) the *token credit (r),* which represents the remaining credits (where 0 <r <b) left in the token bucket at any point over the token interval;

iii) the *token refresh rate (p),* which represents the interval at which the token bucket's is refilled; and

iv) the *token timeout (t),* which represents the ticks remaining until the token refresh rate expires and token credits are refreshed.

The token bucket scheme operates in a rather simple fashion. When the token timeout expires the token credit is set equal to the token budget. One credit represents one cell transmission opportunity. When the flow shaper executes it visits all the per flow queues that are in transfer mode (see the state machine description in section 7.3.1.5 for details). If the flow has cells queued ready for transmission and credits available then the flow shaper transmits one cell and decrements the token credit state variable. The flow shaper then visits

each per flow queue in a round robin manner. At the end of each *round* (i.e., a round is complete when all queues have been visited) it repeats the cycle. This achieves the desire effect of interleaving of cells from different switched virtual circuits onto the network.

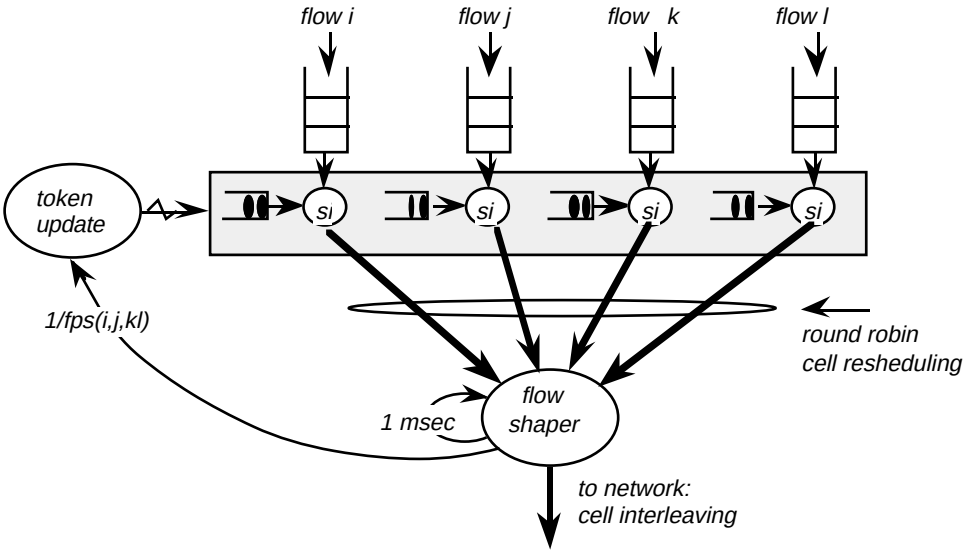

Figure 7.15: Flow Shaping in the ATM Device Driver

There are two possible outcomes at the end of a token refresh interval: either all the cells have been drained or the cells remain in the queue. If cells remain in the queue then all the credits have been used during the interval. If no cells remained queued then all queued cells have been dispatched to the network during the interval with credits *(0<=r < b)* potentially remaining.

At the end of a refresh cycle the remaining credit state variables are updated to the value configured by (b) the token bucket budget. Any credits remaining at the end of the refresh interval are lost, (i.e., the token bucket scheme does not accumulate credits over consecutive intervals). The admission control test at flow establishment ensures that there is sufficient resource to schedule application level frames and regulate cells to the network. Furthermore, all cells associated with one METS packet are given the same transmission deadline.

7.3.1.2 Sync Filter

The METS protocol does not assume that the network will provide a strict bound on delay performance guarantees. Rather, the protocol continuously monitors the end-to-end delay and calculates the playout time of media. As METS packets, which are fragmented into cell, traverse ATM switches distortion of the original *signal* (i.e., traffic shape) occurs. This is due to the variable delays experienced by each packet traversing the network. Furthermore, the end-systems potentially distort the *signal's* timing through interrupt, reassembly and protocol processing. The objective of the sync-filter is to restore the timing of the original *signal* before the video is delivered to the playout device.

Variable delays in the network consist of the queueing delay at the ATM switches. In the 4x4 switch each port controller has a 13 cell FIFO buffer on input and output; a *round robin service discipline* schedules cells between input and output ports. The media access delay in the end-system and the propagation in the network are fixed delays. Additional delays in the end-systems include packetisation and segmentation and reassembly.

End-to-End Delay Estimation

It is a straightforward process to recover the original video timing (at the receiver) for communication systems that provide a hard bound on delay. The process is as follows. Each packet carries an *emission time (T)* which represents the time at which each packet was generated and transmitted to the network. After traversing the network the packet arrives at the destination transport after a delay (d); that is, at time T+d as illustrated in figure 7.15. If the receiver knows the *maximum delay (D)* bound a priori then it is sufficient to buffer the packet between $t+d$ and $T+D$ and then deliver it via the flow scheduler to the playout application at the precise *playout time (T+D)*. This simple algorithm compensates for delay distortion induced in the signal as it transverses the network. This form of jitter correction results in the smooth playout of the original signal.

Many networks do not, however, know what the maximum delay is a priori as required by the previously mentioned playout algorithm. In such cases the receiver can estimate the this delay. According to the algorithm described in [Huitema,95] [Ramjee,94]. This estimate can then be used to determine the playout time of packets as they are received by the transport protocol. Statistical analysis of the per packet delay (i.e., the difference between the emission time carried in the packet and the *reception time* recorded by the flow monitor) is used to estimate the maximum delay. The *standard deviation (s)* and *average*

delay (d) can be estimated based on the per packet delays. An estimate of the maximum end-to-end delay takes the average delay and standard deviation into account: $D = d + r.s$, where *r* is a *filter coefficient* depending on the form of the curve and the number of failures that one is ready to accept [Huitema,95].

Each failure mentioned above corresponds to a transmission delay larger than the estimated maximum. Packets that arrive after the D are considered to be too late to help reconstruct the signal and, in this case, flow management is informed of a late packet event and the packet is dropped. A popular value for r is 2, which corresponds to an accepted loss of about 1% if one assumes a Gaussian distribution of the delays [Huitema,95]. From an intuitive point of view the 2.s term is used to set the playout time to be "far enough" beyond the delay estimate so that only a fraction of the arriving packets should be lost due to late arrival [Ramjee,94]. For a full discussion of these estimates and filter coefficient see [Jacobson,88].

The METS testbed utilises the NTP protocol to achieve global timing. The Network Time Protocol development spearheaded by Mills [Mills,92] provides clock synchronisation to within a few milliseconds or less, even across large Internets. NTP is straightforward to install on networks with IP and Linux running. Achieving ms synchronisation is important because the difference in clock times between the network systems will be insignificant.

The sync filter continuously estimates the average delay and standard deviation. It is based on a 'low pass filtering algorithm' used in TCP for the estimation of the acknowledgement delay time [Jacobson,88]. The sync-filter playout algorithm operates as follows. When a METS packet arrives the *transmission delay (t)* is determined as the difference between the received time and the emission time stamp in the METS packet header. A*new average delay (d')* and a *new standard deviation (s')* are computed. A new estimation of the maximum delay is calculated:

$$d' = d + a.(t-d)$$
$$s' = s + b.(|t-d| - s)$$
$$D = d' + r.s'$$

The constants *a* and *b* are *smoothing coefficients*, with values always less than one. Typical vales are 1/8 and 1/16 respectively which make the calculation of d' and s'

particularly efficient. The value |*t-d*| is the "absolute value" of the difference between the estimated delay and the prediction. As in the case of TCP [Jacobson,88], what is computed is an estimate of the standard deviation.

The playout algorithm results in a continuously evolving estimate of the maximum delay. It is not desirable, though, to continuously alter the actual playout time on a frame by frame basis. Rather, the playout time is continuously calculated but only evolves actual playout at the beginning of each GOP. The objective is to keep these adjustments imperceptible to the human visual system. For example, the *vat audio tool* [Jacobson,93] takes advantage of the natural break in speech (which is made up of a continuous series of talkspurts) to evolve the playout estimate. In this case silent periods between talkspurts are used to advance the playout point. The start of a GOP also is a natural point in a video stream to reassess the playout estimate and advance the playout point.

If the flow management determines that too many losses occur it calculates a new playout time; and while no losses are detected the same playout point is adhered to. If no losses occur over a number of monitoring periods the playout time is recalculated. In way role the sync-filter "pulls in" the playout estimate making the communications more timely. Using this algorithm, the playout point is only evolved when necessary.

Playout Algorithm

From the previous discussion it is clear that the choice of the playout time is important. A playout algorithm that calculates delivery times which are too early would be undesirable with potentially many packets arriving too late to be of any use. In contrast, if a conservative playout time is adopted then many of the packets will arrive well in advance of their playout points. This also has undesirable effects. First, more communications buffering is required to hold the packets until their time of delivery. Second, by adding a delay to the received packet latency in the flow is artificially created which is unsuitable for interactive communications. As discussed above the playout time must continuously trade off loss through late arrivals with timeliness of delivery. The playout time distribution is independent from measured delay distributions experienced during the session.

Figure 7.16 describes a common set of synchronisation scenarios experienced during experimentation on the Lancaster Research ATM Networking Environment. METS packets are generated and transmitted across the network and played out at the receiver. The emission time stamp in each packet represents the deadline the packet was transmitted to the

network at. The flow scheduler operates in an environment where slippage occurs in the transmission time. This is not reflected in the emission time stamp which is the deadline at which the packet should have been scheduled. Many times, however, slippage occurs resulting in delayed transmission of packets to the network. Figure 7.16 represents the *ideal transmission time* as T_i and the *actual transmission* as T_{slip_i} ; the *emission time* is always T_i. The *arrival time* of packet i at the receiver is defined as A_i the *laxity time (D-ai)* (the duration for which a packet is buffered before its playout deadline is current) is defined as *L*, and the *playout point* is defined as P_i.

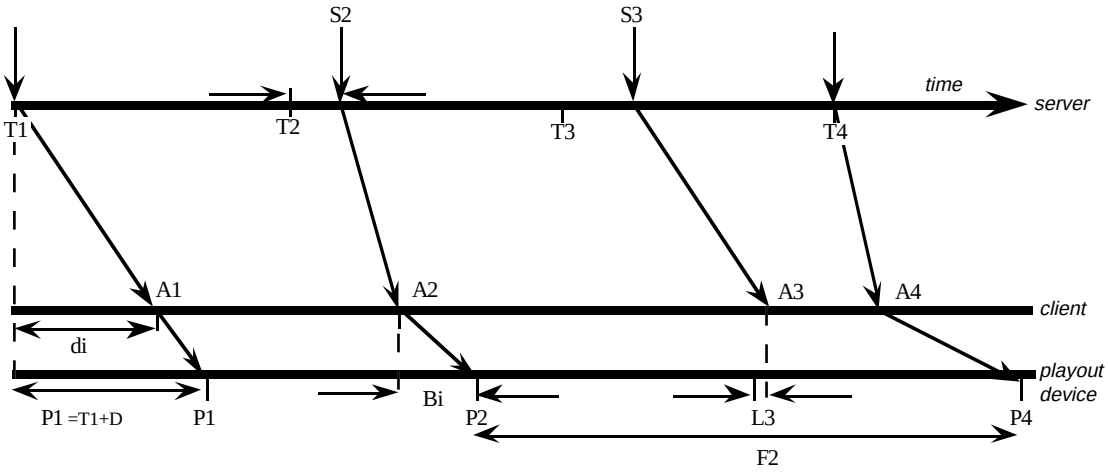

Figure 7.16: Playout Time Model

The initial playout calculation is *P1 = T1 + D*, subsequent playout packets of a GOP are recovered as *Pj = Pi + (Tj-Ti)*. The new estimate of D includes a confidence factor of 2s (where s is the standard deviation) which, as mentioned previously, compensates for error in the estimate. The subsequent calculation of the playout time is determined at the beginning of the next GOP and remains constant for the duration of the GOP. The playout algorithm variables are represented in figure 7.17.

The idea is to gradually evolve the playout interval over successive GOP intervals. By only updating the playout time in this manner the playout times evolve in a conservative manner. While the sync filter continuously calculates the maximum delay over the GOP interval it can only have an impact on the subsequent GOP interval. While the playout is continuously calculated it is updated only at the beginning of each GOP.

7.3.1.3 QoS Monitor

The QoS control module provides a "raw" measure for delay, jitter, loss and bandwidth statistics as input to the transport QoS manager's flow monitor mechanism. This is called QoS measurement metrics. These QoS measurement metrics are maintained on a per flow basis as part of the flow-state table illustrated in figure 7.18. This flow state is periodically polled by the flow monitoring algorithm to develop a detailed picture of a flow's measured performance. QoS measurement metrics consist of absolute and estimated measurements of QoS parameters.

Playout Variable	Description
T_i	transmission deadline at which the packet is generated based and the duration between samples/ METS packets (T2-T1) which is 1/fps
S_i	slippage time which is the actual time when the sample is transmitted to the network; the slip duration is |S2-T1|;
d_i	measured end-to-end delay of packet i;
$d_{i'}$	continuously estimated average delay;
$s_{i'}$	continuously estimated average standard deviation;
P_i	calculated playout time for packet i;
B_i	amount of time which packet i needs to be buffered at the receiver (P_i-A_i);
F_i	frame is frozen on the screen because of a late frames;
L_i	lateness, is the measure by which the frame is late;
A_i	monitored arrival time of the packet, inserted in the control block for ease of implementation

Figure 7.17: Playout Variables

All QoS metrics are maintained on a per METS packet and a per flow basis. Different METSP QoS mechanisms are responsible for calculating each QoS metric as illustrated in figure 7.18. The protocol determines the loss over an interval and records its occurrence as part of the loss flow state. The sync-filter keeps a record of the per packet delay variation (jitter), current end-to-end delay and average delay. The QoS adaptor simply calculates the

bandwidth as METS packets are received over each interval. These measured metrics are polled by the flow monitor every T3 interval as described in section 7.3.2. T3 is set to 10 ms in the current implementation.

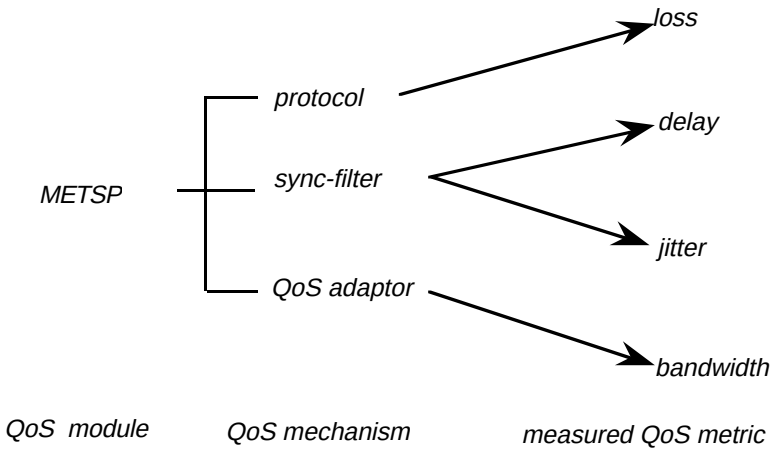

Figure 7.18: Per flow Measured QoS Metrics

7.3.1.4 QoS Adaptor

The QoS adaptor described in Chapter 6 has flow management and QoS control as its components. The flow management component is responsible for informing the application when higher resolutions (i.e., enhancement layers) are made available to clients. The QoS control component is responsible for maintaining sub-flows at the protocol level. This is achieved as follows. When a flow is established the base layer undergoes end-to-end admission testing and resource reservation. Requests for further enhancement of sub-flows depend on transitory resource availability. The QoS adaptor probes the network continuously for resources to provide higher level resolutions (i.e., better QoS). Periodically *resource reservation (RES)* messages are sent to the *core switch* to request additional bandwidth for the enhancement of sub-flows; that is, the RES messages include bandwidth allocation for E1 and E2 if requested in the flow specifications.

As discussed in Chapter 6, reservation messages are updated by each switch en route to the core, *virtual source* or *virtual receiver.* A *virtual source* is a node in the network which modifies the flow characteristics in some manner. For example, if a picture dropping filter [Yeadon,95] is located at the *rook* ATM switch (as illustrated in 7.6) then a client consuming media at *dwp* from a source at *atc* would consider rook a virtual source. Similarly, *atc* would consider rook as a *virtual receiver*. Whenever a core, virtual source or virtual receiver receives a *RES* message it responds by sending an *ADAPT* message to the

originating QoS adaptor. The resulting ADAPT messages include available bandwidth over the next interval (i.e., *era*). The *QoS adaptor* is responsible for the generation and handling of *RES* and *ADAPT* messages, respectively. It probes the network periodically with a request and receives a response. The dissemination tree is composed of clients, servers and virtual sources probing the link for bandwidth in this manner.

As a general rule, servers always request guaranteed resources between the source and the core switch for the base and enhancement layers. This is due to the fact that any fluctuation in the server QoS can potentially affect all clients. The server achieves this guarantee by equating the bandwidth requirement for the base and enhancement layer equal to BL in the flow specification (e.g., *BL=BL+E1+E2*).

7.3.1.5 Protocol Engine

The METSP protocol engine is a *finite state machine (FSM)* driven transport system as shown in figure 7.19. In this section an overview of the operations of the protocol engine is provided. However, a rigorous description of the protocol in terms of transition tables or algorithmic specification is not presented. The METS protocol FSM is comprised of four sub-FSMs: group management, connection management, adaptation control and flow control. The adaptation and flow control modes support the QoS semantics of adaptive flows (i.e., base and enhancement layers). The adaptation and flow control FSM reacts to fluctuations in end-system and network resource saturation. In this way the end-systems adapt and networks exert flow control based on the semantics of layered flows (e.g., BL+E1).

i) the *group control FSM*, which models server initiated group set up and close down and is comprised of the following states:

- *closed state*, which represents a null group state; that is, a server has yet to issue an open group command on a multicast group;
- group setup state, which reflects that an openGroup command has been issued by a server but no response has been received from the mcastd operating at the designated core-switch;
- *group open state*, which represents the state entered when a server has successfully opened a multicast group, resulting in the core switch multicast daemon responding to the openGroup command;

• *wait close state,* this state is entered when a closeGroup has been issued on a multicast group address. In this case the server must wait until leave indications have been issued to all clients before returning to the closed state.

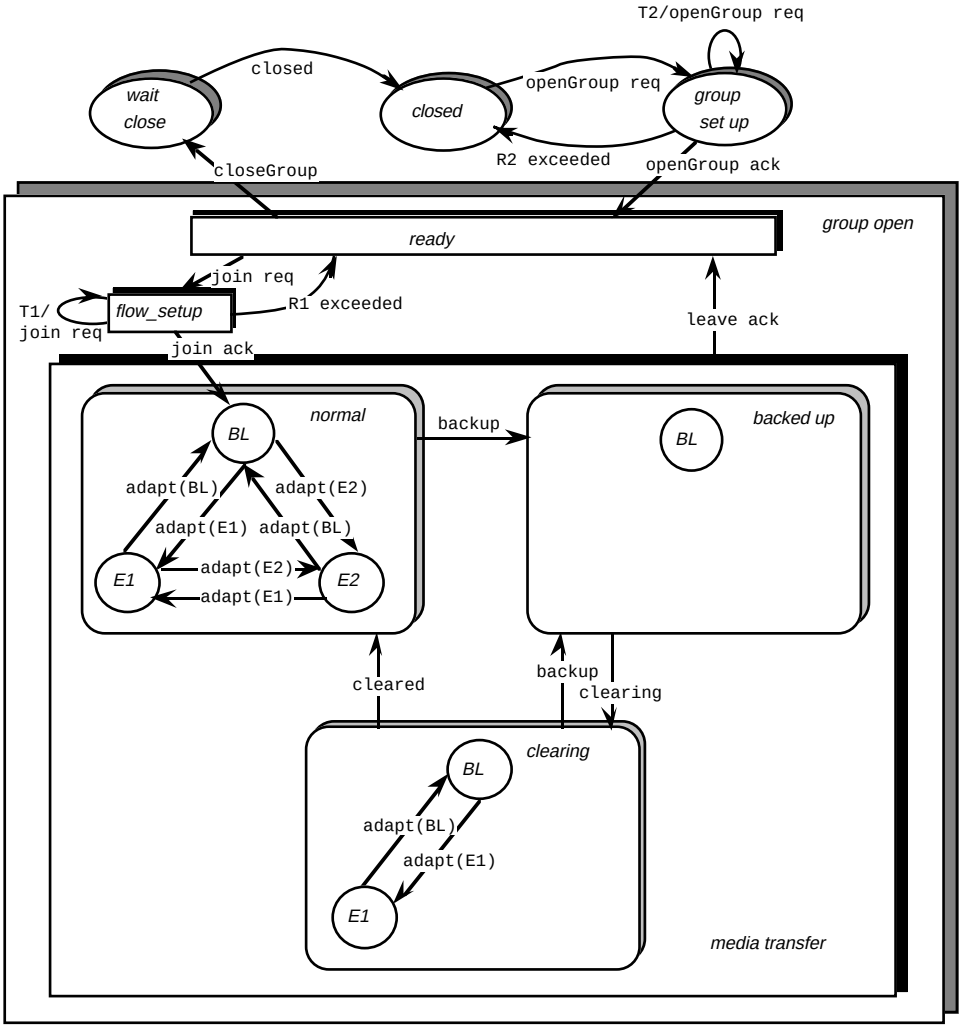

Figure 7.19: METS Protocol Engine

The protocol engine is comprised of four independent finite state machines:

ii) the *media transfer FSM*, which is only entered after the group has been successfully established. At this stage the server has advertised its QoS profile and clients have requested group information to guide their flow selection QoS requirements. In this FMS members can issue joins and leaves on the group and the server can also trigger a leave indication being sent in any state. The media transfer FSM is made up of the following

states:

• the *ready state,* which represents the state in which members can issue join requests on an existing multicast group;

• the f*low setup state*, which indicates that a join has been issued on the group but join acknowledgement is outstanding. The set up algorithm operates as follows: If after T1 timeout the acknowledgement is outstanding another joinFlow request is issued to the group. Furthermore, if after R1 retries no acknowledgement has been received the FSM returns to ready state and the user is informed of the failure;

• the *media transfer state*, which indicates that an acknowledgement has been received and the channel is ready to transmit or receive media. Media transfer FSM encapsulates a flow control sub-FSM which is executed within this state.

iii) the *flow FSM,* which supports the QoS semantics of the adaptive service, reflects the current state of each flow in terms of the sub-flow applications that have been elected (BL, E1 and E2) and when sub-flow communication sub-systems can support given the current available network resources; the adapt messages illustrated in figure 7.19 indicate network resource availability on a per flow basis. The flow control FSM is comprised of the following states:

• the *BL state*, while in this state clients and servers are capable of receiving or transmitting base layer sub-flows, respectively. If flows have been configured to transmit or receive enhancement layers the flow control state may transition to E1 or E2 states based on the user selection and network resource availability;

• the *E1 state,* while in this state clients and servers are capable of receiving or transmitting BL and E1 sub-flows, respectively;

• the *E2 state*, while in this state clients and servers are capable of receiving or transmitting BL, E1 and E2 sub-flows, respectively

iv) the *adaptation control FSM* takes into account resource overload conditions experienced in the end-system. Overload conditions arise when the flow scheduler detects that deadlines are consistently missed. Two overload conditions are identified: congested

and clearing. The semantics of these states are as follows:

• the *normal state*, which implies that no resource saturation has been detected and operates in steady state supporting all sub-flows (BL, E1 and E2);

• the *overload state*, which indicates that flow management has detected that frames are missing their deadlines. In this state only BL components are processed by the METS protocol; all other sub-flows are dropped;

• the *clearing state*, which is entered when flow management determines that deadlines are provisionally back of track. Only BL and E1 sub-flows are processed in this state. This represents a relaxation of the overload condition. After a further period when no deadline thresholds are exceeded the adaptation state reverts back to normal state.

7.3.2 QoS Maintenance Module

The flow monitor mechanism operates on three different levels. First, it periodically monitors the measured metrics (which are described as *synchronous QoS monitoring*) and sends QoS assessment messages to the flow manager for further processing. Second, it continuously monitors measured metrics for the occurrence of a particular user specified event (which is termed *asynchronous event monitoring*) and sends QoS alerts to the flow manager if the event occurs. Finally, it operates as a combination of the former options.

7.3.2.1 Synchronous QoS Monitoring

By default all flows and all QoS parameters are monitored. Flow monitoring updates a per flow QoS record every T3 seconds. At the end of the user specified interval T4 the flow monitor passes a pointer to the flow's current QoS record to the flow manager.

The flow monitor relies on the sync-filter, QoS adaptor and protocol to provide the raw measured data that is used to create a QoS record. The protocol maintains a per *flow loss metric* over the current interval and over for the complete duration of the session. This loss information is in terms of absolute number of METS frames lost. The flow monitor polls this data and provides a max burst loss and percentage loss to the flow manager. The max burst loss represents the maximum loss free of a video sequence. In the case of the flow

monitor, loss is measured as the percentage METS packet lost and max consecutive packets lost. The QoS adapter simply calculates the bandwidth as the number of METS packets received over last interval. The flow monitor uses this metric to determine the maximum bandwidth, average bandwidth and minimum bandwidth.

7.3.2.2 Asynchronous Event Monitoring

The QoS adaptation clause in the service contract allows the user to request notification if one or more of the QoS parameters degrades below a specified value. An asynchronous event QoS signal is delivered to the application's manSoc should this event occur. The QoS signal allows the applications to take remedial action - for example, scaling to the new baseline QoS. Event monitoring is conducted from within the synchronous monitoring mechanism. If the event state is active then the flow monitor compares the designated QoS parameter value against the current value. Event monitoring uses the maximum delay, minimum bandwidth, maximum jitter and percentage loss as comparison metrics. The flow monitor checks every T3 seconds to determine whether an event has occurred.

7.3.3 Flow Management Module

The flow management plane is realised as two modules that co-operate with the signalling and QoS maintenance planes for the establishment and management of flows. These modules include a resource management module which subsumes QoS mapping and admission control, and which is responsible for the negotiation and adaptation of end-to-end resources; and a flow manager module which interacts with the application over the manSoc and which is the central management entity in the end-system. In a generalised QoS-A model flow management spans all layers in the end-systems. In the current implementation, however, it is limited to the operating system, transport and network layers. The resource reservation module interacts with the connection manager in the control (signalling) plane for the allocation, admission testing of network resources, during QoS adaptation and selection of filtering mode and adaptation modes.

The flow manager plays a role in the synchronous and asynchronous monitoring of QoS parameters as illustrated in figure 7.20. In the synchronous mode the flow manager informs the application of the progress of one or more QoS parameters at the end of a user configurable interval. At the end of the T3 interval the flow manager is informed of the measured details via a QoS signal message from the transport QoS manager. The flow manager forwards the measured values of the selected QoS parameters at the end of the user

specified interval.

If the flow manager receives an event QoS signal indicating a QoS violation in the contracted QoS then its action is based on the QoS adaptation event action list. This includes the adaptation of the service, signal the application of the degradation or no actions. In the maintenance mode the application delegates the responsibility of the actions to the flow manager.

The flow manager processes changeQoS messages received on flow management sockets and from the network. Furthermore, it processes any signalQoS messages related to higher resolutions being added or removed. In both cases the flow manager informs the application of the event by issuing the appropriate QoS event signal on the manSoc. This informs the application that the appropriate event has occurred. The resource availability indicates which sub-flows are currently being supported - this is BL for the base layer, E1 for the base layer and first enhancement and E2 indicating that all sub-flows are supported.

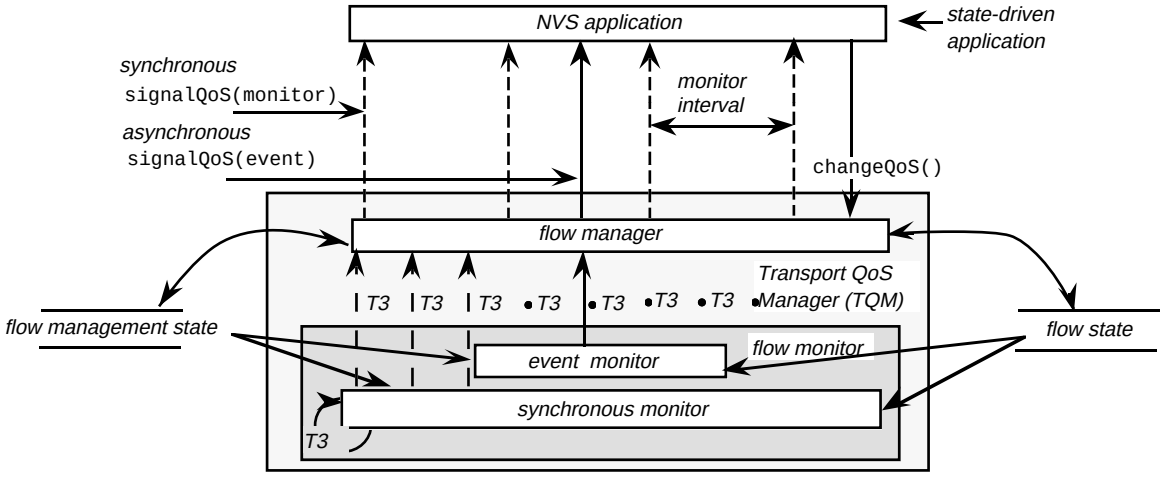

Figure 7.20: Flow Manager and Transport QoS manager

As part of overload processing in the end-system, the flow scheduler informs the flow manager of continually missed deadlines on a dedicated scheduler-to-FM manSoc. In this role, the flow manager updates the flow control FSM state and informs the application of any state changes. The flow scheduler interacts with the flow manager when it has determined that deadlines are again being met. This is a two stage clearing processes as described in section 7.3.1.5. The clearing mode is entered, then again after another uncongested interval normal transfer mode is entered.

7.3.4 Dynamic QoS Management

The adaptive network service is realised by per switch QoS adaptors and media schedulers. Currently these components only address a sub set of network level DQM scheme described in Chapter 6. End-system and network QoS adaptors dynamically determine the residual capacity available to support enhancement layers (i.e., higher resolutions) and the portion of the bandwidth made available to specific applications (i.e., clients and servers). The network level QoS adaptors achieve this via the RES/ADPAT signalling protocol as previously discussed. Each network QoS adaptor maintains per flow bandwidth state; that is, the base layer and higher resolution bandwidth requirements.

The QoS adaptor mechanism updates the advertised rate as the RES messages traverse the switch on the reserve path to the core, virtual source or virtual receiver. Therefore any switch can adjust the flow's advertised rate (E1 or E2) before the virtual source receives the RES message. The destination switch responds to the RES message with the ADAPT message which includes the available bit rate in cell/s: this represents E1 or E2 in the current implementation of discrete QoS adaptation (discussed in Chapter 6). If the originator is a server then it uses the advertised bandwidth for rate control of media over the next interval. In contrast, if originator is a client it uses the advertised rate as an indication of the delivered resolution over the next interval. When a virtual source receives a RES message it responds with an ADAPT message to the originator and then updates the flow state appropriately (viz. BL, E1, E2 state). The QoS adaptor and media scheduler implements the flow finite state machine (FSM) described in section 7.3.1.5.

Media Scheduler

Based on the flow state the media scheduler schedules either no resolutions (BL only), the lower resolution (E1) or the higher resolution (E2). The term E1 and E2 in this capacity subsume lower resolutions; that is, E1 represents BL+E1 and E2 represents BL+E1+E2.

The media adaptor distinguishes each resolution at the input port of a 4x4 switch based on a commitment field found in the AAL5 packet payload. The first word of each AAL5 packet is reserved for the commitment type. This is either BL, E1 or E2. In this capacity E1 and E2 do not subsume the lower resolutions. For example an E1 AAL5 payload only includes lower resolution media.

The commitment based scheduling algorithm is as follows: the QoS adaptor delineate an

AAL5 packet, inspects the commitment field in the AAL5 payload and schedules cells based on this and the current flow state. The QoS adaptor used the ATM-layer-user-to-user (AUU) bit in the ATM header of each cell to delineate AAL5 packet. An ATM-layer-user-to-user bit of 1 followed by a 0 represents the first cell of an AAL5 packet[1]. The first 16 bits of the AAL5 payload represents the commitment type.

While this algorithm is a violation of layering it provides the only way to identify different packets in the same AAL5 stream. This is because there is no logical multiplexing with AAL5. In this example layering is violated but it was found that strict layering was not be the most effective modularity for implementation in the switch. It is important to note that the QoS adaptor mechanism does not have to buffer AAL5 packets to determine the commitment of the packet before forwarding. The overhead for the scheme is minimal. All that is required is to continuously monitor the AUU bit of cells traversing a switch. In addition a single check is required for the first cell of a new AAL5 packet. The result is that cells are streamed through the switch with no additional delay than the unmodified switch code; see Chapter 7 for details.

Once the media scheduler determines the commitment type it interacts with the QoS adaptor to evaluate the current flow state of one or more outgoing port. METS flows generally consist of a multicast relay; that is, cells received on one of the four input ports will likely be multicast on more that one output port. The media scheduler and QoS adaptor are embedded in the switch engine shown in figure 7.1. The media scheduler is only invoke when a cell arrives on an input port and the virtual connection is in 'media transfer modes' (see section 7.3.1.5). The media scheduler interacts with the QoS adaptor to first determine the commitment of the current AAL5 packet. This may already be known (i.e., a the cell with the commitment field may have already been processed). In this case the flow state holds the current AAL5 packet commitment state. This state remains current until the last cell of the AAL5 packet arrives at the input port. The media scheduler then relays all cells based on the flow state and the current configuration of the switch table. The index to the switch table is the virtual connection identifier (VCI) carried in the received cell. The entry in the switch table holds the output ports and associated VCI to relay the cell on. There is a maximum of three entries for a 4x4 switch (i.e., a cell is relayed out all ports with the exception of the one it arrived on) in the switch table. In addition to the port and VCI

[1]The first cell transmitted on a virtual connection with the AUU bit set to 0 represents the first cell of the first AAL5 packet.

information the current flow state of each output VCI is maintained in the flow table. Based on the output VCI's flow state the media scheduler either relays the cell or drops it.

For example, a multicast virtual connection may exist between a server located at the atc end-system and two clients at mr-little and dwp (see figure 7.6). The source node atc transmits full Canyon.mpg resolution, dwp selects the E1 resolution and mr-little the E2 resolution. If a E1 cell arrives at the chuff switch and the adaptive network service can meet the clients QoS demands[1] then the media scheduler forwards the cell to both dwp and mr-little. On the other hand, should an E2 cell arrives on the same connection the media scheduler forwards it onto mr-little only. To further advance the scenario: some time later, both RES messages fail to secure sufficient bandwidth resources for higher resolutions other than BL. In addition the server requested that all sub-flows be guaranteed as described in Chapter 6. This means that the flow state of both clients at the chuff switch is in BL state. In the example whenever E1 or E2 cells arrive at the switch they are dropped; only BL cells are relayed by the media scheduler in this instance.

7.4 Summary

This Chapter has described the implementation of the QoS-A transport system previously introduced in Chapter 4. The objective of the METS transport system is to make QoS visible at the transport API. This is accomplished by preserving application level guarantees throughout the end-systems and the network on an end-to-end basis. The focus of the implementation work presented in this Chapter was the dynamic QoS management (DQM) of adaptive MPEG-1 video flows over ATM networks.

Before concentrating on the design of the API, METS transport systems and ATM network support, the Chapter presented details concerning the experimental infrastructure. Following this the Chapter reported on the development of an QoS configurable API which consists of a set standard socket calls based on a new protocol family called AF_METS. The METS API is comprised group management, multicast connection management, media transfer and flow management primitives. Three styles of AF_METS sockets (viz. media socket, control socket, flow management socket) were identified allowing the application to

1 The QoS adaptors at dwp and mr-little send RES messages indicating E1 and E2, respectively. Subsequently the QoS adaptors receive ADAPT messages verifying that E1 and E2 the bandwidth requirements can be supported.

request, control and manage end-to-end QoS.

The METS implementation corresponded to the end-system and network domains. In the end-system the communication support was embedded in the Linux operating system. As discussed in the Chapter standard Linux does not suitable QoS control and management for continuous media communications. The Linux scheduler can only offers best effort scheduling mechanisms which can not consistently deliver real-time guarantees to applications when resource utilisation grows unbounded. By introducing admission testing for video communications, resource utilisation can be bounded in the case of the transmission and reception of media.

This Chapter has described how METS transport system was added to Linux to remedy its deficiencies in delivering QoS guarantees. This included the implementation of flow scheduling in user space, and flow shaping and jitter correction QoS mechanisms in the kernel. An important contribution of this thesis was the realisation of a set of QoS mechanisms designed to meet the transport needs of adaptive video operating over ATM networks. Implicit in the design of these mechanisms was their ability to distinguish between different QoS needs of flows in the end-system and network.

Chapter 8

Evaluation

This chapter offers an evaluation of the research described in the preceding chapters. The evaluation is divided into two sections. Section 8.1 presents a qualitative architectural comparison of the QoS-A in relation to the QoS architectures presented in the literature. Common themes and open issues which emerged during the comparison section are discussed. In section 8.2, a quantitative performance evaluation of the METS transport systems implementation, as detailed in Chapter 7, is presented. This includes an analysis of the delivered end-to-end QoS (viz. bandwidth, delay, jitter and loss) and METS adaptive network service.

8.1 Architectural Comparison

In this section a qualitative comparison of the QoS-A and QoS architectures reviewed in Chapter 3 presented. The QoS modules outlined in Chapter 2 are used as the basis for the comparison shown in Figure 8.1.

The legend for the comparison 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"
Sig	"QoS signalling in detail"

The term "E2E coordination" referred to in figure 8.1 refers to the coordination of end-

system and network resources for flows. This may be provided by a resource reservation protocol (e.g., RSVP [Zhang,93], ST2 [Topolcic,90] or UNI 4.0 [ATMF,95a]).

QoS Architecture	QoS Provision			QoS Control					QoS Management	
QoS Mechanisms	QoS mapping	Adm. control/ resource res.	E2E coordin -ation	Flow schedul -ing	Flow shaping	Flow control	QoS filtering	flow synchron -isation	QoS monitor -ing	QoS mainten -ance
QoS-A	E N	E N	(E) N	E (N)	E (N)	(E) N	E N	E	E Sig D	E N R S
XRM [Lazar,95]	E N	E N	(E) N	(E) N	-	N	-	-	N	-
ISO [ISO,94]	(E) (N)	E N	E N	-	-	N	E N	-	E N	E N
HieTS [Volg,95]	(E) (N)	E N	E N	E (N)	(E)	(N)	N	-	E N	E N
TINA [TINA,95]	(E)	(N)	N	-	-	-	-	(N)	(N)	-
IETF [Shenker,95]	E N	N	E N	N	E	E	-	-	E N	E N R
Tenet [Ferrari,95]	E N	N	N	N	N	(E)	N	-	E D	E R S
MASI [Bess,94]	E (N)	E (N)	E	E	-	-	-	E	E	E
OMEGA [Nahrstedt,94]	E (N)	E (N)	E (N)	E (N)	E	E	-	-	E	E R
WashU [Gopal,93]	E	E	E	E	E	-	-	-	-	E R
Native [Keshav,94]	E	E (N)	E N	E	E	-	-	-	-	E R

Simple [Damaskos,94]	E N	E N	E N	-	E N	(N)	-	-	E	E R
WollU [Judge,95]	(E)	-	(E) (N)	-	-	-	-	-	E	(E) (R)
Grams [Hui,95]	-	E	-	E	E	-	-	-	E	E R
QuAL [Florissi,94]	E N	-	E (N)	-	-	-	-	-	E	E N R
UMont [Vogel,94]	E N	E N	E N	-	-	-	-	-	-	E R
IMAC [Nicolaou,94]	E N		E N	-	-	-	-	E	E	E R
ANSA [Guangxing,94]	E N	E N	E N	-	-	-	-	E	E	E R
Qadapt [Tran,95]	(E) (N)	(E) (N)	E N	N	-	-	-	-	E	E N R
EuroB [Pronios,93]	(E) (N)	-	(E) (N)	-	-	-	-	-	E N	E N
NEC [Bansal,95]	(E) (N)	(E) (N)	(E) (N)	-	-	(E)	-	-	E N	E N
XTPX [Miloucheva,93]	E N	-	E (N)	-	-	E	-	E	E	-
vnet [Chrysanthis95]	E	(E) N	E N	N	(E)	-	-	-	-	-

Figure 8.1: Comparison of QoS Architectures

All of the QoS architectures identified in figure 8.1 extend the end-to-end argument from the network to include the end-system. These QoS architectures, however, differ in several ways which may be the result of the different communities from which they developed. For example, the XRM and TINA QoS Framework have emerged from the Telecommunication community, the Heidelberg QoS Model and QoS-A from the Computer Communications community, and the ISO QoS Framework and IETF Integrated Services Model from the Standards community. Therefore, it would be inappropriate to declare one

approach ‘better’ than another. QoS architectures emanating from the same community also have been seen to differ. For example, XRM is ‘network centric’ and focuses teletraffic theory while, in contrast, the TINA QoS Framework lies in the application of distributed systems technology to resolve the end-to-end QoS problem but does not quantitatively address end-to-end resource management issues.

While commonalties exit between the QoS architectures described in Chapter 3, a comprehensive comparison of all architectures is beyond the scope of this chapter. Rather, a summary of the common themes, main differences and open issues which emerged during the comparison is provided.

8.1.1 QoS Specification

All QoS architectures reviewed consider QoS specification (e.g., contracts, flow specifications, and service and traffic classes, etc.) as fundamental to capturing user level QoS requirements. Some architectures consider QoS specification at different logical layers or planes in the end-system and network - as is the case of the QoS-A. In this case QoS mapping is used to translate QoS specifications between logical layers/planes.

Although there is a consensus for the need for a flow specification to capture quantitative performance requirements, there exist two schools of thought on what it should be. The XRM and ATM [ATMF,95a] solutions are based on a flow specification that is comprised of one or two QoS parameters that identify traffic class and average bandwidth. On the other hand, the QoS-A, Tenet and OMEGA architectures adopt a multi-valued flow spec (cf. RFC1633, ST-II, RSVP, HieTS). While both of these proposals appear to be similar philosophically they are rather different technically. The COMET group [Lazar,94] argues that by limiting flows to a set of well defined services (in the end-system) and traffic classes (in the network) complexity within the end-system and network will be manageable. In contrast, QoS-A, Tenet 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 its desired QoS requirements.

In summary, the first school of thought believes that all flows fall into a small set of general service and traffic classes with well defined delay, burstiness and loss characteristics. The other school suggests that flows are application specific and that traffic classes will change continually when new applications are identified. Ferarri [Ferrari,95]

argues that the latter approach has the ability to emulate the former, for example, it is effortless to provide a menu of traffic classes above the ST-II or the Tenet suite of protocols, present it to the user and extend it where need be. It remains unclear, however, whether networks can manage the complexity introduced when a continuum of choice is made available to applications (as advocated by the second school of thought).

8.1.2 QoS Commitment

QoS commitment expresses the degree of certainty that the QoS levels specified in a flow specification will be honoured. Each architecture offers a different set of services to applications. Terminology used to describe level of service [ISO,95a] found in the literature includes: service class, traffic class, QoS commitment, application class, QoS class, etc. For example, the Washington University QoS Framework supports three application classes to which it maps all applications level flows.

These include:

i) an *isochronous class*, which is suitable for continuous media flows;

ii) a *burst class*, which is suitable for bulk data transfer; and

ii) 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 classes. In contrast, the QoS-A supports three levels of service (viz. best effort, adaptive, guaranteed) called collectively QoS commitment.

The architectures provide services based on both hard (i.e., guaranteed service) and soft (i.e., best effort) QoS guarantees. Additional services presented in the literature include the predicted service (IETF), statistical service (Tenet, XRM and Heidelberg) and the available bit rate service (ATM Forum). It is too early to determine the extent to which these services will sufficiently address present and future application base. It is encouraging, however, provisional multimedia services may be provided using soft bounds provided by a best effort delivery system. This is illustrated best by the MBONE suite of multimedia tools (e.g., *vic* [MaCanne,94] and *vat* [Jacobson,93]) which are adaptive in nature (i.e., *network conscious applications* [Diot,95]) that have proved successful over the past several years.

8.1.3 Soft versus Hard State

Most QoS architectures consider both static QoS management (in terms of QoS mapping, admission control and resource reservation) and dynamic QoS management (in terms of monitoring, scaling and maintenance). With the exception of the IETF work, which uses RSVP maintained state, all architectures consider connection oriented or 'hard state' solutions to network level QoS provision; that is, they couple path establishment and resource reservation phases. Work in the IETF on an Integrated Services Architecture, using RSVP and IPv6 flows, has shown experimentally that network level QoS guarantees may be obtained using a 'soft state' approach; that is, no explicit connection is established but flows traverse intermediate routers on paths that are temporarily established (i.e., network state is timed out and periodically refreshed). In this instance path establishment and resource reservation are decoupled. It is argued that a soft state approach provides better scalability and robustness, and eradicates the round-trip call setup time found in connection oriented approaches [ATMF,95a] [Banerjea,91].

In [Turner,95], Turner suggests a hybrid approach called ATM-soft which benefits from the use of soft state in a native ATM environment. It is too early to determine which approach is 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.

8.1.4 End-System and Network Commonalties

Commonalties exist between QoS control and management mechanisms 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, is undetermined.

The COMET group argues that end-system and network devices may be similarly modelled and that the sole difference is the overall goal that end-system or network devices are set to achieve. The XRM models the end-system as a virtual switch [Lazar,94] and a set of configurable multimedia devices based on a desk area network (DAN) architecture. The XRM approach strongly endorses the notion of commonality between the network and end-system components.

Furthermore, it is evident that commonalties exist between scheduling strategies found in switches/routers and end-system operating systems (e.g., fair share techniques may be

found in the end-system and network switches/routers). While this seems encouraging, a counter argument exists stating that end-systems have fundamentally different scheduling goals compared with routers and switches. End-systems schedule a wide variety of both isochronous (e.g., continuous media flows) and asynchronous (e.g., RPCs) work while switches and routers are primarily involved with switching/routing of cells/packets, respectively. The end-system application execution time (i.e., quantum [Couslon,95] of work in figure 5.4) can vary widely (e.g., decompressing a video flow can be more computationally intensive than displaying it to a screen). In contrast, switch and router schedulers are generally moving packets/cells from queues to ports or vice versa and are optimised for such a task. Therefore techniques resident in switches (such as HRR [Keshav,93]) may be inappropriate in host operating systems.

8.1.5 QoS Mapping

QoS mapping development is still in the early stages. The work to date has focuses primarily on deriving appropriate QoS parameters [Gopalakrishna,94] for memory, CPU processing (e.g., threads requirements) and network connections in a rather static, architecture-specific manner. It is not clear to what extent QoS mapping must cater for higher layer (e.g., distributed systems platform) requirements where services other than flows are apparent. Currently there is no comprehensive study of QoS mapping presented in the literature.

8.1.6 Heterogeneous QoS Demands

The majority of QoS architectures reviewed in Chapter 3 are either sender or receiver oriented; the exception to this is the OMEGA architecture which supports both options. The Tenet, Heidelberg and QoS-A architectures support heterogeneous QoS demands from individual receivers in multicast groups. Supporting such flexibility is important because heterogeneity exists in applications, communications systems and media formats.

Heterogeneity places fundamental limits on the capability of end-systems to generate and consume continuous media. Resolving heterogeneous QoS demands requires the use of advanced techniques such as QoS filtering in the network and end-systems and QoS adaptation at the network edges. The success of such approaches is still unclear. Most experimental work on QoS filtering and scaling to date has been carried out in the local area only. A disadvantage of this approach is the additional state that would be required at

switches/routers to establish and maintain filters. Whether such filtering techniques will adapt to the wide area and gain general support remains to be seen.

8.1.7 Comparison

Many similarities exist in the QoS architectures reviewed in Chapter 3. There are, for example, several functional similarities between the XRM and QoS-A architectures. In general terms the QoS-A signalling, control and management planes may be mapped to the XRM framework. The QoS-A control and user planes are equivalent to the C-plane and U-plane of the XRM, respectively. During implementation the QoS-A user plane is populated with a multimedia enhanced transport system and AAL5 stack. The XRM user plane consists only of AAL5 in the first instance. This leads to the description of QoS-A as being 'user-plane centric' compared to the XRM. This view is reinforced when the XRM's M-Plane is mapped to the QoS-A. The M-Plane includes cell scheduling, flow control and call admission control. Scheduling and flow control are functions of the QoS-A user plane and admission control is a function of the QoS-A flow management plane.

The XRM N-Plane, which represents network and system management, does not easily map to the QoS-A . The reason is that the QoS-A does not consider network management within the scope of its research. QoS-A management is instead primarily focused on monitoring and maintenance of flows in real-time. The QoS mechanisms that realise these functions reside in the QoS maintenance plane of the QoS-A. Another architectural difference is that XRM explicitly models end-system and network "state" as a telebase (D-Plane). The telebase collectively represents all information, data and abstractions in the systems. There is no such functional equivalent in the QoS-A. Rather, system state is distributed to the various QoS-A mechanisms which maintain it. For example, flow managed by the flow state are managed by the flow manager and protocol respectively.

In summary, QoS-A is considered to be end-system centric whereas XRM network-centric. Both architectures include QoS mapping, admission control and resource reservation, are connection-oriented and provide end-to-end QoS support.

8.2 Performance Evaluation

The performance evaluation is comprises of five test suites for:

i) *bandwidth analysis*, which evaluates the flow scheduling, flow shaping and ATM infrastructure to respond to varying bandwidth

demands;

ii) *delay analysis*, which evaluates the ability of QoS mechanism is response to fluctuating load.

iii) *loss analysis*, which evaluates the protocol loss detection functions;

iv) s*ync-filtering analysis*, which evaluates the delay estimation and playout algorithms at the receiver;

v) *adaptation analysis*, which evaluates the QoS adaptor mechanisms at the end-systems and network and the network adaptive service.

All performance measurements are taken from the QoS-A testbed (illustrated in figure 8.2) with video flows sourced at the *atc* end-system and played out at *mr-little* and *dwp* end-systems. The designated core ATM switch used during the multicast sessions is the *chuff* ATM switch. All measurements presented in this section are captured and logged at atc and dwp end-systems. The distance between the server and client is 3 hops (i.e., flows emanating from atc are played out at dwp traversing the chuff, *sparrow* and *rook* ATM switches, respectively).

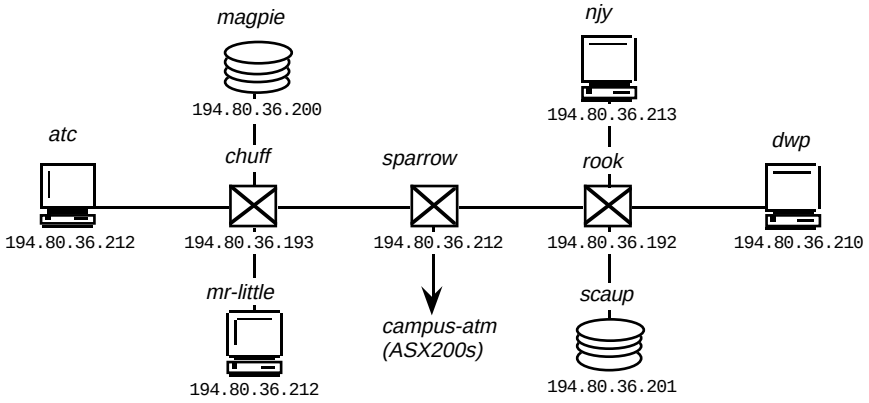

Figure 8.2 METS Experimental Setup

All measurements are taken on a experimental testbed illustrated in figure 8.2. In all cases the server and clients maintain timing logs of METS packet departure and arrivals times, respectively. The Network Time Protocol [Mills,95] provide global timing between all end-systems involved in the experimentation and logging process. The client log includes arrival times, absolute delay and jitter of received packets and loss of METS packets at the transport level. Since the objective of this study is also to examine the performance implications on the application layer, the performance of the NVS decode and display processes at dwp end-system is recorded.

Without the proper admission control and resource reservation functions the incremental addition of flows at the receiver produces overload conditions and rapid degradation and distortion of the played out video picture quality.

8.2.1 Test Video Clips

The video clips used during experimentation are pre-coded off-line and decoded in real time using the NVS software system. Three types of MPEG-1 test video are used in each test configuration. The selected video represents different degrees of action, scene changes, pans and zooms that manifest a variety of different traffic conditions in the network and end-system.

The test clips include the following public domain MPEG-1 video:

i) the *Phil.mpg* video, which represents highly bursty traffic with a frame resolution of 176x168 pixels, 220 frames of IBBPBBPBBPBB GOP configuration with the average I, P, B frames size of 77, 60 and 15 cells, respectively;

ii) the *Canyon.mpg* video, which represents bursty traffic with a frame resolution of 144x112 pixels, 1758 frames of IBBPBB GOP configuration with the average I, P, B frame size of 49, 39 and 9 cells, respectively; and

iii) the *Flight.mpg* video, which represents moderately bursty traffic that tails off with a frame resolution of 160x120 pixels, 114 frames of I GOP configuration with the average I frame size of 57 cells.

The lower resolution (e.g., moderate bandwidth with maximum frame resolution of 176x168 pixels) was chosen over high quality resolution (e.g., broadcast quality of 512x480 pixels) video clips because of the limitation of the end-systems to handle multiple high resolution video. The intention of the test scenarios is incrementally to increase the frame rates and number of active flows to be able to determine the performance of the system as new flows are added or removed.

The three MPEG video clips selected contain both active motion and lower motion video. The Phil.mpg video clip represents rapid scene changes combined with lower motion 'talking head' shots. Canyon.mpg contains a mixture of highly active and lower active motion as illustrated in figure 8.5. The final video Flight.mpg selected for experimentation is only moderately bursty and represents low motion video as shown in figure 8.13.

8.2.2 Bandwidth Analysis

The object of the first test of this suite is to determine the maximum throughput from application space to the network. Within the framework of this test the maximum transmission rates of the METS communication systems are measured. In this test the Canyon video clip is transmitted as rapidly as possible without consideration for achieving constant frame rate or packetising of one MPEG-1 picture per METS packet. Additionally, the traffic shaper and ATM device driver receive interrupt are disabled. Media traverses the METS transport system, AAL5 and ATM network interface controller (NIC) without considering whether the receiver can accommodate the reception of cells transmitted at the maximum rate ('goodput') - this is the objective of the second test of the suite in section 8.2.3.

Maximum Transmission Test

The results of the bandwidth test suite are presented in figure 8.3. The upper curve shows the maximum throughput achieved when the METS packet size is varied incrementally between 1 byte and 32 KBytes. The average rate for each configured transport packet size is measured by the ISA-ATM device driver before the cells are transmitted to the network. The throughput clearly increases as the packet size increment increases. The maximum transmission rate of 32 K cells/sec (13.6 Mbps) is disappointing considering that the line rate of the NIC is over 235 K cells/sec (i.e., 100 Mbps TAXI). The resulting reception rate achieved is under 14% of the line capacity.

The bottleneck during the transmission is a combination of the limitations imposed by an ISA architecture throughput across the system bus and the lack of hardware assist for some of the more computationally intensive communication functions.

Currently the segmentation and reassembly of AAL5 SDUs are functions of the ISA ATM device driver in software. One obvious enhancement would be to pass the responsibility of AAL SDU processing directly to the NIC, off-loading the CPU and enhancing the communication performance. One of the goals of the QoS-A project, however, is to investigate the issues associated with traffic shaping of cells into the network. As a result it is necessary to have the segmentation and transmission functions controlled by the CPU.

It is interesting to note that many of the ATM NICs presently available provide shaping,

AAL5 and ATM processing on the NIC with the aid of hardware assist. The Fore Systems 200E ATM NICs feature a dedicated segmentation and reassembly processor and a platform specific ASIC optimised for the I/O of the host computer. It is fundamental, however, to have the flow shaping mechanism under the control of the METS protocol. While all AAL and ATM processing functions are implemented in software on the CPU the AAL5 checksum is not calculated and the ATM HEC is calculated 'on the fly' and inserted into the ATM header by the NIC card.

'Goodput' Analysis

The second test in the suite measures the 'goodput' achieved between a server and a client using open loop flow control. Goodput is the maximum transmission rate at which cells can be injected into the network and consumed by the receiver with no detectable loss resulting. To achieve the optimum goodput cells must be 'paced' into the network at a rate agreed to between the transmitter and the receiver. It is evident that this rate can not exceed the capacity of the ATM device driver or the capability of the METS communications system to deliver packets to the API. The objective of this test is to determine this threshold bandwidth. For this second test the traffic shaper is enabled for open loop flow control. One of the objectives of the shaper is to limit any back-to-back transmission of cells to the network. This is important in this implementation since the ATM device can only buffer 13 cells before dropping additional cells. By applying open loop rate control using traffic shaping a maximum bandwidth of 17.5 K cells/sec (7.4 Mbps) is made possible.

While these results are disappointing with regard to the line card rate and the capacity of the network to operate near 100 Mbps, this is a direct result of the NIC design limitations to handle larger bandwidths. These limitations include an ATM device with limited buffering and a CPU with additional overhead for handling ATM interrupts and buffer management. In the worst case, there is an interrupt at the receiver for each ATM cell delivered by the NIC. The ISA ATM device driver, however, reduces this overhead by checking whether any cells have arrived at the end of each receive interrupt cycle. One obvious proposal for improvement of the receiver side bandwidth capability performance is to buffer complete AAL5 packets on the NIC and interrupt the CPU only when a full packet is available. This implies that the ORL NIC would require a more sophisticated buffer management capability. It should also be noted, however, that the design objective of the ORL NIC is to provide a simple ATM interface to the network. Another limitation [Lunn,95] is the

buffering scheme of the ATM device driver itself. When a new ATM cell arrives at the receiver the ATM device driver allocates a buffer and copies the payload to it. The performance results of the buffering scheme indicate the need for design re-evaluation. For example, doubling the amount of FIFO buffering would have beneficial effects on performance.

As presented in figure 8.3, the receiver can accommodate a maximum transmission rate of up to 8 K cells/sec after which it starts to drop cells. In the case of this current implementation the loss of one cell causes an entire AAL5 packet to be lost. This highlights the sensitivity of the METS system to single cell loss conditions. The loss of a single cell in this manner could have a major effect on the delivered bandwidth, especially if packets are large. It has been proposed [Pegler,95] that by adding dummy cells at the receiver whenever cell loss is detected then it would be possible to improve the results shown in figure 8.3.

8.2.3 Loss Analysis

The objective of the loss analysis experiment discussed in this section is to determine the impact on the client's ability to consume media as the number of video flows received increases. Figure 8.4 illustrates the percentage of lost frames (i.e., MPEG pictures) as a function of the number of full (i.e., BL+E1+E2 configuration) transmitted Canyon.mpg video clips operating at 12 fps.

Flow Shaping

The upper curve depicts the loss occurring from unregulated sources while the lower curve depicts the loss resulting from flows which are shaped by the METS protocol. The number of received video flows varies from 1 to 8 streams. Both curves show that as the number of MPEG streams increase the loss experienced by the receiver also increases. The maximum percentage loss measured at the receiver varies between 33% and 60% for regulated and unregulated traffic, respectively.

As illustrated in figure 8.4, in the case of unregulated traffic the performance of the system degrades rapidly when 3 flows are received simultaneously. As this figure shows, this represents a loss of greater than 10% - which is perceptible to the end-user. Regulated traffic on the other hand out-performs the former and only exhibits 10% loss when the number of flows approaches 6. For best effort systems which do not offer guarantees to the

base layer this level of loss may be significant to the user. For example, the video playout remains distorted if I pictures are continuously lost .

To address this situation the METS protocol informs the flow management whenever the loss reaches pre-configured thresholds. This in turn triggers a rate reduction request from the client that is included in the next RESV message sent to the server. If clients select the adaptive service then consistent loss causes a reduction in the quality of video to lower resolutions. A drawback of the current implementation is that while guarantees may be made to the base layer by the METS communications system no such guarantee may be given to specific flows as they are handled by the ATM NIC device. As a result the overflow of the receiver device buffers may cause base layer cells to be dropped. This results in a weakening of the QoS commitment to the adaptive network service. To address this the METS system employs admission control testing as part of the resource reservation process and traffic shaping at the server which constrains the operation of flows to be within the level of the agreed contract. These mechanisms help to limit the cell loss to within acceptable bounds (i.e., less than 10%). This limits the ability of the admission control algorithm to accept the equivalent of 3 Canyon flows at 12 fps without any loss detected at the receiver. This is an equivalent bandwidth of 10 K cells/sec for an average packet size of between 2048 and 512 bytes as illustrated in figure 8.3.

Loss and Timing Distortion

The next test in the suite examines the result of loss and timing distortion seen during transmission of video over the testbed. Figure 8.5 presents three flows transmitted from the source node (i.e., atc) to the receiver and figure 8.6 illustrates the distortion detected at the receiver. The three curves represent a base layer (BL) flow of I frames only, an enhancement flow (E1) of I and P frames and a further enhancement flow (E2) of I, P and B frames. The source clip is again the Canyon video. Video transmitted at 24 fps results in approximately 10% loss at the receiver. Close inspection of the three flows reveals that the signal by the transport protocol is very close to the original signal measured at the source. Because there is no additional load on the switches, little queueing delay is experienced by the flows as they traverse the three switches en route. The way in which the transport protocol corrects the timing distortion in the delivered signal through sync-filtering is presented in section 7.3.1.2 and evaluated in 8.2.5. A number of other factors affect the performance of delivered frames to the playout device in a Linux/Unix based system. For example, at the receiver, loss is compounded by the performance of the decode and display

functions of the NVS system.

8.2.4 Delay Analysis

This test suite presented in this section examines the delay distribution experienced by METS[1] packets as they traverse the network and end-system as a function of the number of flow consumed at the receiver. Figure 8.7 illustrates the end-to-end delay distribution of the Canyon video clip between atc and dwp end-systems.

Each METS packet is time-stamped at the server and client. The distribution measured delays for each of the 1758 frames is recorded and shown in figure 8.7. The test setup is a lightly loaded network with one flow running at 24 fps. The average delay measured by the transport protocol at the receiver utilising an absolute timing method is 4 ms. The minimum and maximum delay recorded is 2 ms and 19 ms respectively with a standard deviation of 2 ms.

The second test uses the same setup as the first experimental test with 6 flows being consumed at the receiver. The results measured at dwp measurably show an increase in end-to-end delay experienced by packets traversing the network. Figure 8.8 presents the distribution of delays when the sixth flow is added. The average delay measured by the transport protocol at the receiver using an absolute timing method is 7 ms. The maximum and minimum delays recorded are 40 ms and 2 ms, respectively, with a standard deviation of 2 ms. These results represent a two fold increase in measured delay over the case when one flow is consumed.

The third test measures the delay statistics experienced when the transmitted flows varied between one and eight. Figure 8.9 represents the end-to-end distribution recorded at the client. As can be seen in figure 8.9, there is little difference in the average delay measured as the number of flows increase. The average delay difference between the delay experienced by one flow and eight flows is 6 ms. Variation in the maximum delay experienced is, however, significantly large at 42 ms.

The final test of this delay suite investigates the effect of increasing the number of flows on the NVS decode and display functions. These functions are essentially part of the Berkeley Continuous Media Player [Rowe,92]. Figure 8.10 presents the distribution of delays measured for decode and display processing as the number of flows received varies

1Note that METS packet are packetised into AAL5 packets directly. Therefore all performance characteristics associated with METS packets are equivalent to AAL5 performance.

between 1 and 7. The upper curve depicts the average frame decode time for the Canyon video operating at 12 fps. As the number of flows increase the decode time increases rapidly from 18 ms for 1 flow to over 100 ms for 8 flows. The loss experienced by the receiver when 8 flows are simultaneously consumed nears 40%. This loss, while unacceptable, is not surprising since the video decode process is a software function executed on the host CPU. Like all other CPU-bound processes the decode and display processes must contend for limited CPU cycles as the number of flows and level of communication computation increase with the number of consumed flows.

The lower curve on figure 8.9 illustrates the video display processing for the same conditions as described above. The time taken to complete the display cycle is negligible for up to 3 flows at the receiver. Above 3 flows the time to get the frames onto the display increases to 30 ms per frame when eight flows are being consumed.

As anticipated, the decompress function is the most computationally intensive of the receive thread functions which includes communications, decompress and display. The figures above show that the response time of these functions varies considerably according to the load at the end-system. As the communications overhead increase the amount of CPU time remaining for decoding and display video diminishes. This inevitably leads to higher delays for the communications, decode and display functions - and eventual loss of packets. For example, at 12 fps a new frame is delivered to the receiver every 83 ms. Therefore, the communications decode and display processing must completed within this interval to avoid loss. As presented in figure 8.10, this point is reached when receiving 4 Canyon flows at 12 fps. After this point, picture loss increases rapidly.

8.2.5 Sync Filtering Analysis

This test suite investigates the ability of the sync filtering mechanism to adaptively adjust the playout delay experienced by flows at the receiver to meet end-to-end delay and jitter requirements. Chapter 7 described the implementation of a sync-filter mechanism for calculating the playout point of METS packets received from the network. Because of variable delays experienced by each METS packet during transmission (as shown in figure 8.9), it is important that the receiver attempts to recover the original video signal before playout. The sync-filter attempts to restore the original timing characteristics of the flows.

Figures 8.11 and 8.12 present the operation of the receiving end-system taking part in a networked video session. The source packetises a Canyon video stream and attempts to transmit it at an isochronous rate of 24 fps. Ideally this should result in the transmission

and reception of a METS packet every 42 msec. Because packets incur random delays at the source and while traversing the network they are rarely delivered precisely at 42 msec intervals. Rather, packets arrive with the type of distortion shown in figure 8.11. As described in Chapter 7 the sync filter attempts to restore the timing of the signal by smoothing out distortions introduced by estimating the maximum end-to-end delay as part of the playout algorithm. Figures 8.11 and 8.12 show the playout strategy for reception of frames as a function of arrival time. The upper curves on both figures illustrate the measured arrival times for consecutive packets. It should be noted that METS packets are always received in order of arrival. The lower curves represent the playout time curve as calculated by the algorithm described in Chapter 7. The values of the smoothing coefficients *a* and *b* as defined in the playout algorithm were chosen to be 1/8 and 1/16, respectively.

The results of this final test are quite promising. Figure 8.11 shows the transmission of the Canyon video over a short interval of the complete playout trace; this interval is consistent with the complete trace. The calculated playout time shadows the arrival rate very closely with no loss of packets due to underestimation of the maximum end-to-end delay. This is expected when one considers the relatively smooth delivery of packets (i.e., no sign of large jitter is manifest in the trace). The trace also shows the point at which the transport QoS manager adjusts the current playout estimate. This is accomplished at the beginning of each new GOP as discussed in Chapter 7.

In contrast to the relatively smooth behaviour of the scenario described above, figure 8.12 illustrates a situation where large deviations in the end-to-end delay are noticeable. In this case the measured statistics represent the Canyon video playout while 4 other background Canyon flows are simultaneously handled by the same receiver. All measurements are taken by the transport protocols flow monitor mechanism. Figure 8.12 shows that packet loss occurs between 43000 and 43500 ms due to underestimation of the maximum end-to-end delay. Unexpectedly large jitter such as this is difficult to predict. One solution, though, is to increase the confidence factor of 2s (where s is the standard deviation of the maximum delay described in section 7.3.1.2) which, as mentioned previously, compensates for error in the estimate by extending the playout time. Overestimation of this kind, however, leads to larger buffer requirements at the receiver and, significantly, loss of timeliness when considering the overall performance of the entire trace.

In figure 8.12, the playout curve tracks the arrival time curve to the first point of loss -

the region between 42500 and 43000 ms. The first point of inflection represents a sudden increase in the end-to-end delay and subsequent loss of a number of METS packets. The second point of inflection (between 43000 and 43500 ms) also represents a large increase in measured delay, subsequent loss of packets and then the simultaneous arrival of a group of packets at the receiver.

Note that the playout algorithm adjusts to fluctuations detected in the measured maximum delay but these adjustments are rather conservatively in nature. This is the optimal policy in the long term since, as is seen, shortly the estimate is back on track. The choice of filter coefficients dictates the rate of change of the estimated delay regarding the fluctuations in the arrival times of packets. Filtering coefficients influence the responsiveness of the playout algorithm to track changes in the arrival patterns. For example, larger coefficients would cause the playout to mirror fluctuations in the arrival time distribution. This is not, however, always the best policy. Optimally, the playout time should to evolve in response to *trends* in the arrival time patterns and not occasional *spikes* as illustrated in figure 8.12. During experimentation, the coefficient values used by the sync filter were determined to be the most appropriate for local area ATM networking. For a full discussion on filter coefficients see [Jacobson,88] and [Ramjee,94]

8.2.6 Adaptive Network Service Analysis

The final test suite of this thesis relates to the performance of the adaptive network service and QoS adaptation mechanisms. The adaptive network service described in Chapter 6 provides “hard” guarantees to the base layer (BL) of a multi-layer flow and soft guarantees to each of the enhancement layers (E1 and E2). To achieve full end-to-end admission testing is carried out on the base layer. On the other hand, enhancement layers are admitted without any such test but must compete for residual bandwidth among all other adaptive flows. Enhancement layers are rate controlled based on explicit feedback (i.e., RESV messages) regarding the current state of the ongoing flows and the availability of residual bandwidth.

Figure 8.13 illustrates the benefits of the adaptive network service. In this test the receiver (i.e., the dwp end-point) selects three flows for playout in the first instance. These comprise:

i) the *Canyon.mpg* video flow (selecting the BL, E1 and E2 components) at 24 fps;

ii) the *Phil.mpg* video flow (selecting the BL only) at 5 fps; and

iii) the *Flight.mpg* video flow (selecting BL) only at 5 fps.

These video flows, layered components and frame rates are chosen to best demonstrate the benefits of intelligently adapting to the available bit rate. The goal is to demonstrate the ease at which different resolutions of the Canyon video are automatically added and removed to take advantage of the available resources. In this test the admission control resource is restricted to supporting 600 K cells/sec in the end-systems.

The scenario shows the Canyon and Flights video clips being consumed starting at time zero. Both base layers are supported. The QoS adaptor in combination with the adaptive network service determines that none of the Canyon flow's higher resolutions can be accommodated, however, given the available resources. Twenty seconds into the scenario trace the Flights video terminates freeing up resources. At this point the QoS control judges that the highest resolution of the Canyon video (i.e., BL+E1+E2) may be displayed as illustrated in figure 8.13.

This situation remains until the user chooses to display the Phil.mpg video 50 seconds into the trace. Based on the semantics of the adaptive service resources are allocated to meet the base layer QoS requirements for the new video. The QoS adaptor protocol sends a RES message toward the core requesting resources to meet the highest resolution (E2) of the Canyon.mpg video. In this instance there are insufficient resources available to meet the QoS requirements of the highest resolution. However, the resulting RES message indicates these resources are adequate to support the lower resolution (E1) of the Canyon.mpg flow. As illustrated in figure 8.13, 50 seconds into the trace the Phil.mpg video comes on-line and the resolution of the Canyon video drops.

While discrete fluctuations between different video resolutions is noticeable to the end user the reported results are still very promising. However, it should be noted that continuous adaptation using the dynamic rate shaping filter as described in Chapter 6 would resolve such fluctuations. In this role dynamic QoS management continuously adapts the resolution of the delivered video to the available bit rate.

8.2.7 Discussion

The performance characteristics of the METS transport system were quantitatively analysed in part two of this section. The performance evaluation was comprised of five

experimental test suites: bandwidth analysis, delay analysis, loss analysis, sync-filtering and adaptive service. All performance measurements were taken using the QoS-A testbed and three types of experimental MPEG-1 video (which broadly represent different degrees of action, scene changes, pans and zooms). These video clips allowed the system to be tested under conditions that produced a wide variety of traffic conditions and loads in the network and end-systems.

While discrete adaptation was noticeable the resulting perceptible changes were not pronounced. The results of the adaptive service test appear to be very promising but it must be determined whether such a scheme translates to the wider area.

Many of the test results highlight the limitations of the network interface card and ISA bus architecture as opposed to deficiencies in the METS transport system or switching capability[1]. In particular the maximum transmission rate of 32 K cells/sec is disappointing considering that the line rate is over 235 K cells/sec. Similarly, the maximum reception rates with cell loss were found to be K cells/sec and without loss 17.5 K cell/sec. Again these results prove disappointing and only compare favourably to standard Ethernet performance.

METS frame loss was found to be very susceptible to single cell loss conditions. Admission testing, resource reservation and open loop traffic shaping out performed unregulated flows by a wide margin. The end-system could consume 4 Canyon flows at 12 fps with an overall loss of 10% of the METS frames when the flows were regulated. This rose to 33% loss for 8 flows consumed. Corresponding performance for the unregulated case was measured to be 15% and 60% for 4 and 8 flows consumed, respectively; this proving traffic shaping is a valuable means of constraining loss at the receiver.

The delay analysis highlighted that while there was minor variation in the average end-to-end delay as more flows were consumed there was considerable variation in the maximum end-to-end delay. The average delay difference experienced between one flow and 8 flows was found to be 6 ms. In contrast, the maximum delay difference measured was 42 ms. Since the synchronisation function works by estimating the maximum delay the latter results are significant. The delay evaluation test also investigated the performance of the decode and display processes as a function of increasing load in the end-system. As the number of flows displayed at the end-system increased so did the time to decode and

[1] The ORL 4x4 switch which is based on an ARM processor has a measured aggregate switching capability of 200 K cells/sec.

display frames. The decode duration for single frames ranged from 4 to 58 ms. This time duration represents the execution time of the decode function and includes operating system overhead of context switching, the contention by other decode and display processes for CPU cycles and ATM and METS communication processing. As the load increased each of these processes contended for more CPU. Similarly, the time taken to display a frame to the screen increased rapidly as the load increased: from 18 msec for 1 flow to over 100 ms for 7 flows. These results are coupled to the particular video clip chosen at 12 fps. However, the results indicate that in order to operate with acceptable QoS (i.e., loss less than 10% and acceptable decode and display times) the end-system can only consume 4 Canyon flows at full resolution.

Both the sync filtering and adaptive service testing demonstrated the benefit of these QoS mechanisms in providing jitter correction and maximising the utilisation of the available bandwidth. The sync-filter tracked the arrival time distribution by estimating the maximum delay and calculating the playout time of successive GOP sequences. This proved to be highly successful at both the low load and high load. While the choice of filtering coefficients produced dampened responsiveness to spikes in the arrival time distribution it was deemed suitable given the long term jitter trends trace. The adaptive network service test suite highlighted the benefit of adjusting the resolutions of flows to meet specific needs of different clients at the same receiver.

8.3 Summary

This Chapter has evaluated the work previously described the body of the thesis. First, a qualitative comparison of the QoS-A to other existing QoS architectures presented in the literature was provided. Each QoS architecture was reviewed for the QoS provision, control and management components presented in Chapter 3. While there was a broad consensus on a number of QoS issues many important questions remain unresolved (e.g., the choice of QoS specification, suitability of QoS commitments to cover the application base, the choice of hard state or soft state solutions). These issues will be resolved as more implementations become available in the future.

With the exception of ATM cell HEC computation, all switch, ATM layer, adaptation layer, METS transport system and NVS function processing is 'soft'; that is, flows are completely under software control. All networked devices (e.g., ATM switch) have the advantage of being are fully programmable. The METS approach is completely software

based. This software intensive approach provides a highly configurable and flexible systems environment. Such an approach is particularly appropriate for end-to-end research allowing complete software control of the end-to-end resource management process.

While a software intensive approach allows flexibility in the development of communications protocols it nevertheless burdens the host CPU with additional communication (e.g., segmentation and reassembly of AAL SDUs), codec and signalling overhead. In summary a software intensive approach results favourable flexibility but disappointing performance (cf. bandwidth, loss test results)

The wider implications of both this qualitative and quantitative work will be presented and discussed in the next Chapter.

Chapter 9

Conclusion

In this Chapter the conclusions of the thesis are presented. This Chapter begins with an overview of the argument of the thesis. Following this, the contribution of the thesis is presented. Finally, the thesis concludes by providing some indicators for future work in the area of end-to-end QoS research.

9.1 Summary of Thesis

Chapter 1 reported on the evolving notion of QoS in research and standards and concluded by identifying a number of limitations in the state of the art. This thesis argued that for applications relying on the transfer of multimedia information, in particular continuous media, it is important that quality of service is configurable, predictable and maintainable on an end-to-end basis - that is, system-wide, including the distributed system platform, operating system, transport system and the underlying network. Meeting quality of service guarantees in distributed multimedia systems generally requires the provision of a number of QoS mechanisms such as resource reservation, flow scheduling and flow shaping. In recognition of this, the thesis has argued for the need for an integrated QoS architecture which spans both end-systems and networks and takes QoS control, maintenance and management for continuous media flows as its primary goal.

Chapter 2 indicated the importance of QoS control, maintenance and management in distributed systems, and showed how these functions can be used as building blocks for a generalised QoS framework. Fundamental terminology, principles and concepts used for developing and discussing QoS architectures was introduced. The important notions of flows and QoS specification were presented as key concepts in capturing, requesting and negotiating end-to-end QoS for continuous media communications.

Chapter 3 presented a comprehensive survey of the current state of the art in QoS research reported in the literature. Recent work directed at integrating and extending layer-specific research into broader QoS architectures was addressed. These architectures promoted the idea of integrated QoS, spanning the end-systems and the network and

identified the support for end-to-end QoS as an important new goal. A review of layer-specific QoS research indicated that much of the work reported to date has concentrated on applying QoS concepts described in Chapter 2 to either the network or the end-system in isolation. On the other hand, emerging QoS architectures attempt to coherently apply QoS concepts across all architectural layers, resulting in a framework for the specification and implementation of end-to-end QoS. It was noted that while the area of QoS research in multimedia networking is now mature, work in QoS architectures research remains in very early stages of development.

Chapter 4 presented an outline of the author's contribution to end-to-end QoS research, an integrated QoS architecture for continuous media communications (QoS-A). A major contribution of this thesis was the realisation of the QoS-A at the transport layer. A new transport service and protocol collectively called the Multimedia Enhanced Transport System (METS) was proposed. The METS transport system comprised of signalling (METSig) and protocol (METSP) modules which mapped to the control and user plane of the QoS-A, respectively. To meet transport level QoS requirements, METSP incorporated buffer sharing, flow regulation, flow scheduling and basic flow monitoring QoS modules. Each module was configured based on the flow specification and QoS commitment described in a user specified service contract. METSig was made up of group management, multicast connection management and dynamic QoS management signalling components. The QoS maintenance plane comprised a transport QoS manager for the fine grained QoS management of on-going flows. Applications interacted with a flow management protocol over a dedicated interface for the establishment and dynamic QoS management of multicast flows.

Chapter 5 described operating system support for a QoS-A which guaranteed QoS levels of both communications and processing with varying degrees of QoS commitment as specified by user level service contract. The approach taken was based on an enhanced Chorus micro-kernel operating system environment. The discussion focused on resource management aspects of the design and in particular CPU scheduling, network resource management and memory management issues. The proposed operating system architecture minimised kernel level context switches and exploited early demultiplexing so that incoming media was always treated according to the QoS associated with API level connection. It also eliminated data copying on both send and receive (except for unavoidable copies to/from the ATM network interface controller card)

Chapter 6 addressed the problem of resolving heterogeneous QoS demands by extending the dynamic QoS management (DQM) provision of the QoS-A model to meet the needs of scalable continuous media. A scheme for the dynamic QoS management of multi-layer encoded flows in heterogeneous multimedia and multicast networking environments was detailed. Dynamic QoS management manipulated and adapted multi-layer coded flows at the end-systems and in the network using a set of scaling objects. The approach taken was based on three basic concepts: the scalable composition of MPEG standards that can provide discrete adaptation, dynamic rate shaping (DRS) algorithms for compressed digital video that provide continuous adaptation, and the weighted fair share (WFS) service for adaptive flows.

Chapter 7 described the implementation of the QoS-A transport system. The objective of the METS transport system was to make QoS visible at the transport API. This is accomplished by preserving application level guarantees throughout the end-systems and the network. The design of a QoS configurable API, METS transport protocol and ATM network signalling support was reported. The METS API included group management, multicast connection management, media transfer and flow management primitives. Three styles of AF_METS sockets (viz. media socket, control socket, flow management socket) were identified allowing the application to request, control and manage end-to-end QoS. A number of QoS mechanisms were added to the Linux environment to remedy its deficiency in support if continuous media; these included flow scheduling, flow shaping and jitter correction QoS mechanisms described above. An important contribution of the implementation work was the realisation of a set of QoS mechanisms designed to meet the transport needs of adaptive MPEG-1 video flows operating over ATM networks. Implicit in the design of these mechanisms was their ability to distinguish between different QoS needs of multi-layer coded flows in the end-system and network.

Finally, Chapter 8 evaluated the thesis and attempted to place the work in the context of QoS architecture research reported in the literature. A qualitative comparison of the QoS-A to other existing QoS architectures was presented. While there was broad consensus on a number of QoS issues many other important issues remain unresolved. For example, the choice of QoS specification and QoS commitments varied widely in the literature. It was noted that these issues may be resolved in the near future as the field matures. Chapter 8 also offered a quantitative analyse of the METS transport protocol performance. The performance evaluation comprised of five experimental test suites: bandwidth analysis,

delay analysis, loss analysis, sync-filtering and adaptive service. All performance measurements were taken using the QoS-A ATM networking testbed and three types of experimental MPEG-1 video (which broadly represent different degrees of action, scene changes, pans and zooms). Many of the test results highlighted the limitations of the network interface controller (NIC) card and ISA bus architecture as opposed to deficiencies in the METS transport system or ATM switching system. Both the sync filtering and adaptive service evaluation demonstrated the benefit of these QoS mechanisms in providing jitter correction and maximising the utilisation of the available bandwidth, respectively. The adaptive network service test suite highlighted the benefit of adjusting the resolutions of flows to meet the specific needs of different clients with heterogeneous QoS requirements.

9.2 Thesis Contribution

9.2.1 Integrated QoS Architecture (QoS-A)

The thesis has recommended that multimedia system designers take an integrated QoS approach to the development of new communication systems. Rather than considering QoS in the end-system and network in isolation the author has proposed a new integrated QoS model which incorporates QoS interfaces, control and management mechanisms across all architectural layers. The QoS-A is based on fundamental QoS concepts and principles for building QoS into multimedia communication system. It offers a framework to specify and implement the required performance properties of multimedia applications over high-performance networks. The QoS-A incorporates the notions of flow, service contract and flow management with particular emphasis on the enhanced transport service interface and dynamic QoS management. The notion of a flow and a service contract were introduced as key concepts in capturing, requesting and negotiating end-to-end QoS. The thesis also introduced the idea of flow management which provides for the monitoring and maintenance of the contracted QoS.

9.2.2 QoS Configurable Transport System

A major contribution of this thesis was the realisation of the QoS-A at the transport layer and its assessment in the context of a generalised quality of service architecture. A new transport service and protocol collectively called the Multimedia Enhanced Transport System (METS) was proposed, designed and implemented. The METS transport systems includes QoS control, maintenance and management mechanism to support multicast flows. It was shown how QoS levels contracted at the transport level application programmers'

interface (API) were assured in the context of the Lancaster ATM Research Networking Environment.

9.2.3 Design of QoS Controlled Operating System Support

This thesis has presented the design of QoS controlled operating system extensions to meet the needs of the QoS-A. The design was embedded in a micro-kernel/ PC environment and supported by QoS-A driven, ATM based communications. Resource management aspects of the design dealt with CPU scheduling, network resource management and memory management issues. The proposed extensions to the Chorus micro-kernel offered guaranteed QoS levels of both communications and processing with varying degrees of QoS commitment. The micro-kernel used admission testing to determine whether or not new flow activities could be accepted and included a QoS mapping module to translate user level QoS parameters into representations usable by the scheduling, network and memory management subsystems.

9.2.4 Dynamic QoS Management (DQM)

The thesis presented the design, and implementation aspects (see section 9.2.5) of dynamic QoS management (DQM) which controls and managed multi-layer coded flows operating in heterogeneous, multicast, multimedia networking environments. DQM extended the QoS-A model presented in Chapter 4 by populating the QoS management planes of the architecture with a framework for the control and management of multi-layer coded flows operating over ATM networks.

Two key novel DQM techniques were proposed:

i) an *end-to-end rate shaping* scheme which adapts the rate of MPEG-coded flows to the available network resources while minimising the distortion observed at the receiver; and

ii) an *adaptive network service*, which offers “hard” guarantees to the base layer of multi-layer coded flows, and “fairness” guarantees to the enhancement layers based on a bandwidth allocation technique called *weighted fair sharing (WFS).*

9.2.5 Operational QoS Platform

An implementation of the QoS-A transport layer has contributed towards a fully comprehensive realisation of QoS architecture. The platform realised the functionality of METS transport system described above and offered a validation of the suggested end-to-

end approach for QoS management. Flow scheduling, flow shaping and sync filtering have proven very effective in controlling individual flows in Linux/ native ATM based end-systems. It also has been shown that the QoS platform can supports DQM concepts in a very effective manner - tailoring applications level QoS to meet fluctuating resource availability. While the experimental infrastructure was limited to the local area it remains to be determined whether DQM can scale to the wide area and large numbers of receivers with widely varying heterogeneous QoS demands. The METS API was realised as a new protocol family presenting the application with three styles of service: control channel, media channel and flow management interface. The separation of this functionality at the API significantly enhanced the application's ability to specify QoS requirements, and control and manage end-to-end QoS.

9.2.6 Evaluation of Platform

A thorough evaluation of the METS transport systems operating over local ATM has been performed through the use of the NVS MPEG-1 application level demonstrator. This has served to improve the level of knowledge on QoS architecture based transport system. Providing flexibility at the transport API to state QoS policy, and QoS mechanism in end-system and network to interpret this policy, has confirmed the validity of the QoS-A approach. It has been demonstrated by the performance evaluation results that the transport QoS mechanisms (viz. adaptive network service, flow scheduling, flow shaping, sync filtering), which were designed to operate in an adaptive environment, responded well to fluctuating network and end-system resource availability while allowing applications to adapt to the delivered QoS in a control manner.

9.2.7 Contribution to QoS Standards

Standards have an important role to play in promoting a unified view of QoS. The QoS Project in the ISO has examined the requirements for QoS support in Open Systems standards addressing QoS in a consistent way. This activity covers QoS very broadly and has investigated user requirements for QoS, architectural issues and recently QoS mechanism [ISO,92a]. Lancaster University has played an active role in the ISO QoS project over the past few years, the author having provided early input on the QoS-A [Campbell,92b] into this activity. The subject of integrated QoS also emerged as an important activity in another ISO project on Enhanced Communication Functions and Facilities (ECFF) for the lower layers of the OSI reference model. As a member of the

ESPRIT-funded OSI 95 project the author participated in the ECFF activity. In addition, he was instrumental in introducing what was considered to be the key multimedia communication requirements [Campbell,92], into the ECFF guidelines document [ISO, 92b].

9.3 Future Work

There remain many specific areas of research in the field addressed by this thesis which should be addressed by future work. The major areas requiring investigation are enumerated in the following sub-sections.

9.3.1 Binding Architecture

The first topic to be addressed in the future research is a comprehensive specification and implementation of the higher layers of the QoS-A. This research has concentrated on the transport and ATM networking issues of the QoS-A implementation. The current implementation does not extend to the distributed system platform (e.g., CORBA). However a pre-requisite to provide QoS support in the distributed systems is suitable QoS conscious transport and networking services and protocols. The author is currently working towards this aim in the context of the work on a *Binding Architecture* [Lazar,94]

9.3.2 QoS Mapping

Apart from the static QoS translation of QoS parameters between the transport API and local and remote resource managers (viz. CPU, memory, network) no further QoS mapping was considered. The area of QoS mapping is still in its infancy with no comprehensive solutions has been reported in the literature. To date, most QoS mapping implementations are static in nature and application specific. This suggest that either the requirements of a comprehensive QoS mapping systems are not well understood or that QoS mapping has been considered to be a second order problem to date. Much research is required in the area to provide a fully comprehensive mapping scheme.

9.3.3 Internet QoS Architecture

The work by the Integrated Services (int-serv) Group is perhaps the most significant, challenging and ambitious realisation of a QoS architecture to date. The int-serv group have defined a comprehensive QoS model [Shenker,94] which make multiple, dynamically selectable qualities of service available to applications in an internetwork. Many issues need be resolved to before IPv6 flows offer QoS configurable and predictable communications to the end user. For example, the mapping of IPv6, int-serv QoS model and RSVP to the ATM environment will determine whether the QoS benefits of per-flow ATM performance are made visible to the end-systems. However, there are a number of similarities in the approach of the ATM Forum and IETF in meeting QoS needs. Both groups approach end-to-end research from different perspectives (vis-a-vis hard state and soft state approaches). However they have complementary approaches to the specification of flows but rather different approaches to the service classes they offer to the applications. Unifying these two approaches is essential in making ATM QoS visible to the application in any future integrated services internetwork.

9.4 Concluding Remarks

The notion of QoS has evolved rather rapidly over the past few years. In the early 1990s - when the work on this thesis commenced - the notion of quality of service in communications architectures was a narrow one. Traditionally, the term 'quality of service' referred to certain characteristics of network performance outside the influence of the user. Recent years have seen great advances in QoS research, due mainly to the emergence of multimedia networking and computing. Today, ATM networks not only have the capability of transmitting information at high speed, but with suitable control and management mechanisms they have the potential to offer end-to-end QoS configurable communications - and significantly under the management of the end user.

A number of important new initiatives from the standards and research communities have emerged recently. These initiatives broadly address limitations in the current QoS provision in the light of new multimedia communication requirements. Three significant pioneering contributions to the field of QoS research include the IETF's integrated service model (mentioned above), ATM Forum's QoS service model and SC21's OSI QoS framework. While each group is attempting to offer an 'end-to-end' solution, they do not

have an agreed, collaborative approach. This in itself is a point of major concern.

The research reported in this thesis has been influenced by the debates within the ATM Forum, IETF and ISO communities on QoS research. While the implementation of the QoS-A was restricted to the transport and ATM network layers it is hoped that this work can contribute toward the assessment of a generalised QoS architecture that ultimately will help harmonise the activities of these various communities.

References

[Abrossimov,89] Abrossimov, V., Rozier M. and Shapiro M., "Generic Virtual Memory Management for Operating System Kernels", SOSP'89, Litchfield Park, Arizona, December 1989.

[Accetta,86] Accetta, M., Baron, R., Golub, D., Rashid, R., Tevanian, A., and M. Young, “Mach: A New Kernel Foundation for UNIX Development”, *Technical Report* Department of Computer Science, Carnegie Mellon University, August 1986.

[Anderson, 90] Anderson, M., Tzou, S.Y., Wahbe, R., Govidan, R. and Andrews, M., "Support for Continuous Media in the DASH System", *Proc of the 10th International Conference on Distributed Computing Systems*, Paris, May 1990.

[Anderson,91] Anderson, D.P., Herrtwich, R.G. and C. Schaefer. “SRP: A Resource Reservation Protocol for Guaranteed Performance Communication in the Internet”, *Internal Report*, University of California at Berkeley, 1991.

[APM,91] APM Ltd , “ANSAware 3.0 Implementation Manual”, APM Ltd, Poseidon House, Castle Park, Cambridge CB3 0RD, UK, 1991

[ATM Forum,95] ATM Forum, “ATM User-Network Interface Specification Version 4.0”, 1995.

[Aurrecoechea,95] Aurrecoechea, C., Campbell, A., and L. Hauw “A Survey of Quality of Service Architecture”, Multimedia Systems Journal, November, 1995 (to be published).

[Aurrecoechea,95] Aurrecoechea, C., Campbell, A., Hauw, L. and Hisaya Hadama, “A Model for Multicast for the Binding Architecture”, Technical Report, Center for Telecommunications Research, Columbia University, USA.

[Baguette,92] Baguette, Y. et al, "TPX Specification", OSI 95 Report, ULg-5/R/V1, University of Leige, Belgium, October 92.

[Ball,94] Ball, F. and D. Hutchison, “Traffic control in an ATM LAN”, Proc. 2nd IFIP Workshop on Performance Modelling and Evaluation of ATM Networks, Bradford, UK, 4th-7th July 1994.

[Ballardie,93] Ballardie, T., Francis, P. and Jon Crowcroft, “Core Based Tree (CBT) An Architecture for Scalable Inter-Domain Multicast Routing”, Proc. ACM SIGCOMM ‘93, San Francisco, USA.

[Bansal,95] Bansal, V., Siracusa, R.J, Hearn, J. P., Ramamurthy and D. Raychaudhuri, “Adaptive QoS-based API for Networking”, Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, April, 1995.

[Benerjea,91] Benerjea, A. and B. Mah, “The Real-Time Channel Administration Protocol”, Second International Workshop on Network and Operating System Support for Digital Audio and Video”, Heidelberg, November 1991.

[Besse,94] Besse, L., Dairaine L., Fedaoui, L., Tawbi, W., and K. Thai, “Towards an Architecture for Distributed Multimedia Application Support”, Proc. International Conference on Multimedia Computing and Systems, Boston, May 1994.

[Black,93] Black, R., and S. Crosby, "Experience and Results from the Implementation of an ATM Socket Family", *ATM Document Collection 3 (The Blue Book,* Cambridge Computing Labs, 1993.

[Blair,93] Blair, G.S., Campbell, A., Coulson, G., Garcia, F., Hutchison, D., Scott, A., and W.D. Shepherd, “A Network Interface Unit to Support Continuous Media”, *IEEE Journal of Selected Areas in Communications (JSAC)* , February 1993.

[Boerjan,92] Boerjan, J., Campbell A., Coulson G., García F., Hutchison D., Leopold, H. and N. Singer, "The OSI 95 Transport Service and the New Environment", ISO/IEC JTC1/SC6/WG4 N824, International Standards Organisation, UK, December 1992, and *Internal Report* No. MPG-92-38 Department of Computing, Lancaster University, Lancaster LA1 4YR.

[Braden,94] Braden R., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", Request for Comments, RFC-1633.

[Bricker,91] Bricker, A., Gien, M., Guillemont, M., Lipkis, J., Orr, D., and M. Rozier, “Architectural Issues in Microkernel-based Operating Systems: the CHORUS Experience”, *Computer Communications*, Vol 14, No 6, pp 347-357, July 1991.

[Bulterman,91] Bulterman D. C. and van Liere R., “Multimedia synchronisation and UNIX”, Proc. Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Springer Verlag, 1991.

[Campbell,92a] Campbell, A., G. Coulson and D. Hutchison, "A Suggested QOS Architecture for Multimedia Communications", ISO/IEC JTC1/SC21/WG1 N1201, International Standards Organisation, UK, November, 1992.

[Campbell,92b] Campbell, A., Coulson G., García F., and D. Hutchison, "A Continuous Media Transport and Orchestration Service", *Proc. ACM SIGCOMM '92*, Baltimore, Maryland, USA.

[Campbell,93a] Campbell, A., Coulson, G., García, F., Hutchison, D., and H. Leopold, "Integrated Quality of Service for Multimedia Communications", *Proc. IEEE Infocom'93*, Hotel Nikko, San Francisco, CA, March 1993.

[Campbell,93b] Campbell, A., Coulson G., and D., Hutchison, "A Multimedia Enhanced Transport Service in a Quality of Service Architecture", *Proc. Fourth International Workshop on Network and Operating System Support for Digital and Audio and Video*, Lancaster, UK, October 1993, and ISO/IEC JTC1/SC6/WG4 N832, International Standards Organisation, UK, November, 1993.

[Campbell,94a] Campbell, A., Coulson, G. and Hutchison, D., "A Quality of Service Architecture", ACM Computer Communications Review, April 1994.

[Campbell,94b] Campbell, A., Coulson, G., and D. Hutchison, "Flow Management in a Quality of Service Architecture", *5th IFIP Conference on High Performance Networking*, Grenoble, France, June 1994.

[Campbell,95] Campbell, A., Coulson G. and D. Hutchison, "Supporting Adaptive Flows in a Quality of Service Architecture", Multimedia Systems Journal, November, 1995 (to be published).

[Chesson,88] Chesson, G., "XTP/PE Overview", Proc. 13th Conference on Local Computer Networks, Pladisson Plaza Hotel, Minneapolis, Minnesota, 1988.

[Chrysanthis,95] Chrysanthis, P., and D Mosse', "Management and Delivery of Multimedia Traffic", Proc. Second International Workshop on community Networking Integrated Multimedia Services to the Home, Princeton, NJ, June 1995.

[Cidon,92] Cidon, I., Gopal, I., Gopal P.M., Janniello and M. Kaplan, "The plaNET/ORBIT High Speed Network", *Internal Report* No. 18270 IBM T.J. Watson Research Center, August, 1992.

[Clark,84] Clark, D., and D.L. Tennenhouse, "Architectural Consideration for a New Generation of Protocols", Proc. ACM SIGCOMM '90, Philadelphia, 1984.

[Clark,87] Clark, D.D., Lambert, M.L., and L. Zhang, “NETBLT: A High Throughput Transport Protocol”, Computer Communications Review, Vol. 17, No. 5, 1987.

[Clark,90] Clark, D., and D.L. Tennenhouse, "Architectural Consideration for a New Generation of Protocols", *Proc. ACM SIGCOMM ‘90*, Philadelphia, USA

[Clark,92] Clark, D.D., Shenker S., and L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism" *Proc. ACM SIGCOMM’92*, pp. 14-26, Baltimore, USA, August, 1992.

[Cocchi,91] Cocchi, R., Estin, D, Shenker, S. and L. Zhang, "A Study of Priority Pricing in Multiple Service Class Networks", *Presented at ACM SIGCOMM ‘91*, pp. 123-130,1991.

[Cohen,77] Cohen, D.,"Issues in Transit Packetized Voice Communication", Proc. Fifth Data Communications Symposium, Snowbrid, USA.

[Coulson,92] Coulson G., Blair G. S., Davies N. and Williams N.” Extensions to ANSA for Multimedia Computing”, Computer Networks and ISDN Systems, 25(11), 305–23, 1992.

[Coulson,93a] Coulson, G., Blair, G.S., Robin, P. and Shepherd, D., "Extending the Chorus Micro-kernel to Support Continuous Media Applications", *Proc. Fourth International Workshop on Network and Operating System Support for Digital Audio and Video*, Lancaster University, Lancaster LA1 4YR, UK, October 93.

[Coulson,93b] Coulson, G., and G. Blair, “Micro-kernel Support for Continuous Media in Distributed Systems”,*Computer Networks and ISDN System.*

[Coulson,94a] Coulson, G., and G.S. Blair. "Micro-kernel Support for Continuous Media in Distributed Systems", Computer Networks and ISDN Systems 26 (1994), pp 1323-1341, Special Issue on Multimedia, 1994.

[Coulson,94b] Coulson, G., G.S. Blair, P. Robin, and D. Shepherd, "Supporting Continuous Media Applications in a Micro-Kernel Environment." in *Architecture and Protocols for High-Speed Networks*, Editor: Otto Spaniol, Kluwer Academic Publishers, 1994.

[Coulson,95] Coulson, G., Campbell, A and P. Robin, “Design of a QoS Controlled ATM Based Communication System in Chorus”, IEEE Journal of Selected Areas in Communications (JSAC), Special Issue on ATM LANs: Implementation and Experiences with Emerging Technology.

[Crosby,93] Crosby, S., "MSNL Connection Management " *ATM Document Collection 2*, Technical Note pp. 12-1, 12-11, Systems Research Group, Computer Laboratory, University of Cambridge, February 1993.

[Cruz,91] Cruz, R., "A Calculus for Network Delay: Part I: Network Elements in Isolation", IEEE Transactions on Info. Theory, Vol. 37. No. 1, Jan. 1991.

[Dabbous,95] Dabbous, W. and C. Diot, "High Performance Protocol Architectures", Technical Report, INRIA, Sophia Antipolis, France, 1995

[Damaskos,94] Damaskos, S. and A. Gavras, "A Simplified QoS Model for Multimedia Protocols over ATM", High Peformance Networking, S.Fdida ed., Elsevier Scince B. V. (North-Holland), 1994.

[Danthine,92] Danthine, A., Baguette Y., Leduc G., and L. Leonard, "The OSI 95 Connection-Mode Transport Service - Enhanced QoS", *Proc. 4th IFIP Conference on High Performance Networking*, University of Liege, Liege, Belgium, December 1992.

[Deering,94] Deering, S., "Simple Internet Protocol Plus (SIPP) Specification", Work in Progress, *Internet Draft,* <draft-ietf-sipp-spec-00.txt>, February 1994.

[Delgrossi,93] Delgrossi, L., Halstrinck, C., Henhmann, D.B, Herrtwich R.G, Krone, J., Sandvoss, C., and C. Vogt, "Media Scaling for Audio-visual Communication with the Heidelberg Transport System", Proc ACM Multimedia'93 Anaheim, USA.

[Doeringer,90] Doeringer, W., D. Dykeman, M. Kaiserswerth, B. Meister, H. Rudin, R. Williamson, "A Survey of Light-weight Transport Protocols for High-speed Networks", IEEE Transactions on Communications, November 1990.

[Eleftheriadis,95a] Eleftheriadis, A., and D. Anastassiou, "Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Code Digital Video", Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, USA.

[Eleftheriadis,95b] Eleftheriadis, A., "Dynamic Rate Shaping of Compressed Digital Video", Ph.D. Thesis, Columbia University, USA.

[Escobar,92] Escobar, J., Deutsch, D. and C. Partridge, "Flow Synchronisation Protoco", IEEE GLOBECOM'92, Orlando, Fl., December 1992.

[Fedaoui,94] Fedaoui, L., Seneviratne, A., and E. Horlait, "Implementation of a End-to-End Quality of Service Management Scheme", Cost 237 Workshop, Vienne, November 1994.

[Feldmeier,93] Feldmeier, D.,“A Framework of Architectural Concepts for High Speed Communication Systems”, Computer Communication Research Group, Bellcore, Morristown, May 1993.

[Ferrari,90] Ferrari, D. and D. Verma, “A Scheme for Real-Time Channel Establishment in Wide Area Networks”, *IEEE J. Selected Areas in Comm.*, Vol 8 No 3, April 1990.

[Ferrari,92] Ferrari, D., Ramaekers J. , and G. Ventre, "Client-Network Interactions in Quality of Service Communication Environments”, *Proc. 4th IFIP Conference on High Performance Networking*, University of Liege, Liege, Belgium, December 1992.

[Ferrari,95] Ferrari, D., “The Tenet Experience and he Design of Protocols for Integrated Services Internetworks”, Multimedia Systems Journal, November 1995.

[Florissi,94] Florissi, P. G. S., and Y. Yemini,”QuAL: Quality Assurance Language”, ITS’94.

[Floyd,93] Floyd, S., “Link-Sharing and Resource Management Models for Packet Networks”, Draft available via anonymous ftp from ftp.ee.lbl.gov: link.ps.Z, September 1993.

[French,94] French, L.J., Willson, I.D., and D.P. Gilmurry, "ATMOS II User Manual", Olivetti Research Limited, 24a Trumptington Street, Cambridge, 1994.

[Fry,93] Fry, M., Seneviratne, A., and A Richards, “Framework for the Implementation of the the Next Generation of Communication Protocols”, Forth International Workshop on Network and Operating Systems Support for Digital Audio and Video, University of Lancaster, November 1993.

[García,93] García, F., “A Continuous Media Transport and Orchestration Service”, PhD Thesis, Department of Computing, Lancaster University, Lancaster LA1 4YR, UK, June 1993.

[Golestani,90] Golestani, S.J., “A Stop and Go Queueing Framework for Congestion Management”, Proc. ACM SIGCOMM’90, San Francisco, June 1990.

[Gopalakrishna,94 Gopalakrishna, G., and G. Parulkar, “Efficient Quality of Service in Multimedia Computer Operating Systems”, Department of computer science, Washington University, Report WUCS-TM-94-04, August 1994.

[Gopalakrishna,95] Gopalakrishna, G., and G. Parulkar, “A Real-time Upcall Facility for Protocol Processing with QoS Guarantees”, (Poster) 15th ACM Symposium on Operating Systems Principles, December. 1995.

[Gopalakrishna,95] Gopalakrishna, G., and G. Parulkar, "Quality of Service Support for Protocol Processing within the Endsystem", Proc. High Speed Networks for Multimedia Applications Workshop, Dagstuhl, July 1995

[Govindan,91] Govindan, R., and D.P. Anderson, "Scheduling and IPC Mechanisms for Continuous Media", *Thirteenth ACM Symposium on Operating Systems Principles*, Asilomar Conference Center, Pacific Grove, California, USA, SIGOPS, Vol 25, pp 68-80, 1991.

[Govindan,91] Govindan, R., and D.P. Anderson, "Scheduling and IPC Mechanisms for Continuous Media", Thirteenth ACM Symposium on Operating Systems Principles, Asilomar Conference Center, Pacific Grove, California, USA, SIGOPS, Vol 25, pp 68-80, 1991.

[Guangxing,94] Guangxing, "An Model of Real-Time QoS for ANSA" , Technical Report APM.1151.00.04, APM Ltd, Cambrigde, UK, March 1994.

[Guerun,91] Guerun, R., Ahmadi, H., and M. Naghshineh,"Equivalent Capacity and its Application to Bandwidth Allocation in High Speed Networks" , IEEE Journal on Selected Areas in Communications, Vol. 9, No. 7, Sept. 1991.

[H.262,94] H.262, "Information Technology - Generic Coding of Moving Pictures and Associated Audio", Committee Draft, ISO/IEC 13818-2, International Standards Organisation, UK, March 1994.

[Hayter,91] Hayter, M and D. McAuley, "The Desk Area Network", ACM Operating Systems Review, Vol 25, No 4, pp14-21, October 1991.

[Hehmann,91] Hehmann, D.B., Herrtwich, R.G., Schulz, W., Schuett, T. and R. Steinmetz, "Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High Speed Transport System", *Proc. Second International Workshop on Network and Operating System Support for Digital Audio and Video*, IBM ENC, Heidelberg, Germany, 1991.

[Hoffman,93] Hoffman, D., Speer, M. and G. Fernando, "Network Support for Dynamically Scaled Multimedia Data Streams", Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster, UK.

[Hui,95] Hui, J., Zhang, J., and Jun Li, "Quality of Service in GRAMS for ATM Local Area Networks", IEEE Journal of Selected Areas in Communications (JSAC), Special Issue on ATM LANs: Implementation and Experiences with Emerging Technology, May 1995.

[Huitema,94] Huitema, C., "Routing in the Internet", Prentice Hall, ISBN 1-13-132192-7, 1994.

[Hutchison,92] Hutchison, D. and Campbell, A. “Key Issues in Multimedia Communications”, ISO/IEC JTC1/SC6/WG4 SD/14, International Standards Organisation, UK, November, 1992, and *Internal Report* No. MPG-92-39 Department of Computing, Lancaster University, Lancaster LA1 4YR.

[Hutchison,94] Hutchison, D., Coulson G., Campbell, A., and G. Blair , “Quality of Service Management in Distributed Systems”, to appear: M. Sloman ed., *Network and Distributed Systems Management*, Addison Wesley, chapter 11, 1994, and *Internal Report* No. MPG-94-02 Department of Computing, Lancaster University, Lancaster LA1 4YR.

[Hyman,90] Hyman, J., Lazar, A., and G. Pacifici, “Real-Time Scheduling with Quality of Service Constraints”, IEEE Journal on Selected Areas in Communications, Vol. 9. No. 7, April 1990.

[Hyman,92] Hyman, J., Lazar, A., and G. Pacifici, "Joint Scheduling and Admission Control for ATS-based Switching Nodes", *Proc. ACM SIGCOMM ‘92, Baltimore, Maryland, USA*, August 1992.

[IETF,95] Slides from IETF meeting 31, Integrated Service Working Group, ftp://mercury.lcs.mit.edu/pub/intserv, 1995.

[ISO,92] ISO, "Draft Guidelines for Enhanced Communication Function and Facilities for the Lower Layers", ISO/IEC JTC1/SC6/WG4 N7309 *International Standards Organisation*, UK, May 1992.

[ISO,95a] ISO, "Quality of Service Framework", ISO/IEC JTC1/SC21/WG1 N9680, *International Standards Organisation*, UK, 1995.

[ISO,95b] ISO, "QoS - Methods and Mechanism", ISO/IEC JTC1/SC21/WG1 N9310, *International Standards Organisation*, UK, 1995.

[Jacobson,93] Jacobson, V, " Congestion Avoidance and Control", *Proc ACM SIGCOMM'88*, Stanford, 1988.

[Jacobson,93] Jacobson, V., "VAT: Visual Audio Tool", *vat manual pages*,1993.

[Jain,95] Jain, R., “Congestion Control and Traffic Management in ATM Networks: Advances and a Survey”, Computer Networks and ISDN Systems, 1995.

[Jeffay,91] Jeffay, K., Stone, D. and F. Donelson Smith, "Kernel Support for Live Digital Audio and Video", *Proc. Second International Workshop on Network and Operating System Support for Digital Audio and Video,* IBM ENC, Heidelberg, Germany, Springer Verlag, 1991.

[Jeffay,92] Jeffay K., Stone, D.L., Talley, T. and F.D. Smith, "Adaptive, Best Effort Delivery of Digital Audio and Video Across Packet-Switched Networks", Proc. Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, USA.

[Jeffay,93] Jeffay, K., "The Real-Time Producer/Consumer Paradigm: A Paradigm for Construction of Efficient, Predictable Real-Time Systems," Proc. 1993 ACM/SIGAPP Symposium on Applied Computing, Indianapolis, IN, February 1993.

[Jeffay,95] Jeffay, K and D. Bennet, "A Rate-Based Execution Abstraction For Multimedia Computing," Proc. Fifth International Workshop on Network and Operating Systems Support for Digital Audio and Video,", Durhan, NH, April 1995.

[Judge,95] Judge, J., and P. Beadle, "Supporting Quality of Service on Multimedia Terminals Interconnected by a Low Speed ATM Network", SPIE Vol. 2417, 1995.

[Jung,93] Jung, J., and D. Seret , "Translation of QoS Parameters into ATM Performance Parameters in B-ISDN", *Proc. IEEE Infocom'93*, Vol. 3, San Francisco, USA, 1993.

[Kanakia,93] Kanakia, H., Mishra, P., and A. Reibman, "An Adaptive Congestion Control Scheme for Real Time Packet Video Transport", Proc. ACM SIGCOMM '93, San Francisco, USA, October 1993.

[Kelly,93] Kelly, F.P., "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, England.

[Keshav,91] Keshav, S., "On the Efficient Implementation of Fair Queueing", *Internetworking: Research and Experiences*, Vol. 2, pp 157-173, 1991

[Keshav,91] Keshav, S., "On the Efficient Implementation of Fair Queueing", Internetworking: Research and Experiences, Vol. 2, pp 157-173, 1991.

[Keshav,92] Keshav, S., "Report on the Workshop on Quality of Service Issues in High Speed Networks", *ACM Computer Communications Review,* Vol 22, No 1, pp 6-15, January, 1993.

[Keshav,93] Keshav, S., “Report on the Workshop on Quality of Service Issues in High Speed Networks”, ACM Computer Communications Review, Vol 22, No 1, pp 6-15, January, 1993.

[Keshav,94] Keshav and Saran, “Semantics and Implementation of a Native-Mode ATM Protocol Stack”, Bell Labs Technical Memorandum, http:// www.cs.att.com/ csrc/keshav/papers.html, 1994.

[Kurose,93] Kurose, J.F., “Open Issues and Challenges in Providing Quality of Service Guarantees in High Speed Networks”, *ACM Computer Communications Review*, Vol 23, No 1, pp 6-15, January 1993.

[Lazar,90] Lazar A.A., Temple, A.T., and R. Gidron , “MAGNET II: A Metropolitan Area Network based on Asynchronous Time Sharing”, *IEEE Journal on Selected Areas in Communications*, Vol 8, No 6, pp 1582–94, 1990.

[Lazar,90] Lazar, A. A., Temple, A and Gidron, “An Architecture for Integrated Networks that Guarantees Quality of Service”, International Journal of Digital and Analog Communications Systems, Vol. 3, No. 2.

[Lazar,92] Lazar, A.A., “A Real-time Control, Management, and Information Transport Architecture for Broadband Networks”, Proc. International Zurich Seminar on Digital Communications, pp. 281-295, 1992.

[Lazar,94] Lazar, A. A., “Challenges in Multimedia Networking”, Proc. International Hi-Tech Forum, Osaka, Japan, Februray 1994.

[Lazar,94] Lazar, A. A., Bhonsle S., Lim, K.S., “A Binding Architecture for Multimedia Networks”, Proceedings of COST-237 Conference on Multimedia Transport and Teleservices, Vienna, Austria.

[Leopold,92] Leopold, H., et al, "Distributed Multimedia Communications System Requirements", *OSI95/Deliverable ELIN-1/C/V3*, Alcatel ELIN Research, A-1210 Vienna, Ruthnergasse 1-7, Austria, April 1992.

[Leslie,93] Leslie, I.M., McAuely, D., and S.J. Mullender, “Pegasus - Operating Systems Support for Distributed Multimedia Systems,” Operating Systems Review, Vol. 27, No. 1, 1993.

[Leydekkers,95] Leydekkers, V. Gay and L. Franken, “A Computational and Engineering View on Open Distributed Real-time Multimedia exchange”, Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, 1995.

[Linux,94] Welsh, M., and L. Kaufman, "Running Linux", O'Reilly Associates, ISBN 1-56592-100-3, 1994.

[Little,90] Little, T.D.C, and A. Ghafoor, “Synchronisation Properties and Storage Models for Multimedia Objects”, IEEE Journal on Selected Areas on Communications, Vol. 8, No. 3, pp. 229-238, April 1990.

[Liu,73] Liu, C.L. and Layland, J.W., “Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment”, Journal of the Association for Computing Machinery, Vol. 20, No. 1, pp 46-61, February 1973.

[Lunn,93] Lunn, A. S., “A Mini cell Architecture for Multimedia Systems"”, PhD Thesis, Department of Computing, Lancaster University, Lancaster LA1 4YR, UK, Sept 1995.

[Lunn,94] Lunn, A.S., Scott, A.C., Shepherd, W.D and N.J. Yeadon, “A Mini-cell Architecture for Networked Multimedia Workstations”, to be presented: *1994 International Conference on Multimedia Computing*, Boston, USA, and *Internal Report* No. MPG-93-30 Department of Computing, Lancaster University, Lancaster LA1 4YR.

[McAuley,93] McAuley, D. R., “Operating System Support for the Desk Area Network,” Proc. Forth International Workshop on Network and Operating Systems Support for Digital Audio and Video,” Lancaster, England, November 1993.

[McCanne,94] McCanne, S., Jacobson, V., "VIC: Video Conference" U.C. Berkeley and Lawrence Berkeley Laboratory. Software available via ftp://ftp.ee.lbl.gov/conferencing/vic

[Miloucheva,95] Miloucheva, I. “Quality of Service Research for Distributed Multimedia Applications”, ACM Pacific Workshop on Distributed Multimedia Systems, 1995.

[Miloucheva,95] Miloucheva, I. and K. Rebensburg, “QoS-based Architecture using XTP”, Forth IEEE International Conference on Future Trends of Distributed Systems, Lisboa, Sept, 1993.

[Mullender,93] Mullender S., ed. (1993). Distributed Systems, 2nd end., Addison-Wesley.

[Nahrstedt,93] Nahrstedt, K. and J. Smith, “Revision of QoS Guarantees at the Application/Network Interface”, *Technical Report*, Distributed Systems Laboratory, University of Pennsylvania, 1993.

[Nahrstedt,94] Nahrstedt K. and J. Smith, “A Service Kernel for Multimedia Endstations”, Proc. IWACA’94: Multimedia: Adavnced Teleservices and High-Speed Communication Architectures, Heidelberg 1994.

[Nahrstedt,95a] Nahrstedt K. and J. Smith, “The QoS Broker”, IEEE Multimedia, Spring 1995.

[Nahrstedt,95b] Nahrstedt, K., and R. Steinmetz, “Resource Management in Networked Multimedia Systems”, K. Nahrstedt and R. Steinmetz, IEEE Computer Magazine, May 1995.

[Nahrstedt,95c] Nahrstedt K. and J. Smith, “Design, Implementation and Experiences of the OMEGA End-Point Architecture”, Technical Report (MS-CIS-95-22), University of Pennsylvania, May 1995, (submitted to JSAC).

[NATO, 88] Private communication with Chris Sluman, 95.

[Nicolaou,90] Nicolaou, C., “An Architecture for Real-Time Multimedia Communication Systems”, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, April 1990.

[Nicolaou,93] Nicolaou, C., “Integrating Multimedia into the ANSA Architecture”, Tecnical Report TR.028.93, APM Ltd, Cambridge, UK. 1993.

[ODP] ODP, “Draft recommendations X.903: basic reference model of open distributed processing”, ISO/IEC JTC1/SC21/WG7, International Standards Organisation, 1992.

[OMG,93] OMG, “The Common Object Request Broker: Architecture & Specification, Rev 1.3., December 1993.

[Pacifici,95] Pacifici, G., and R. Stadler, “An Architecture for Performance Management of Multimedia Networks”, Proc. IFIP/IEEE International Symposium on Integrated Network Management, Santa Barbara, May 1995.

[Paek,95] Paek, S., Bocheck, P., and Chang S.-F.,“Scalable MPEG-2 Video Servers with Heterogeneous QoS on Parallel Disk Arrays”, Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, USA.

[Parekh,92] Parekh, A. and R. G. Gallager, "A Generalised Processor Sharing Approach to Flow Control in Integrated Service Networks - The Multiple Node Case", *Proc. IEEE INFOCOM’93*, pp.521-530, San Francisco, USA, April 1993.

[Parekh,93] Parekh, A. and R. G. Gallager, "A Generalised Processor Sharing Approach to Flow Control in Integrated Service Networks - The Multiple Node Case", Proc. IEEE INFOCOM’93, pp.521-530, San Francisco, USA, April 1993.

[Partridge,92] Partridge, C., "A Proposed Flow Specification; RFC-1363" *Internet Request for Comments*, no. 1363, Network Information Center, SRI International, Menlo Park, CA, September 1990.

[Pasquale,92] Pasquale, G., Polyzos, E., Anderson, E. and V. Kompella, "The Multimedia Multicast Channel", *Proc. Third International Workshop on Network and Operating System Support for Digital Audio and Video,* San Diego, USA, 1992.

[Pasquale,93] Pasquale, G., Polyzos, E., Anderson, E., and V. Kompella, "Fitter Propagation in Dissemination Trees: Trading Off Bandwidth and Processing in Continuos Media Networks", Proc. Forth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster, UK.

[Pegler,95] Pegler, D., Hutchison, D., Lougher, P. and D Shepherd, "A Scalable Multimedia Storage Hierarchy", Technical Report, MPG-01-95, Lancaster University, England.

[Pronios,95] Pronios, N., "EuroBridge: A QoS-Driven Architecture", Technical Report, Intracom S.A, Greece, 1995.

[Ramjee,94] Ramjee, R., Kurose, J.,Towsley, D., and H. Schulzrinne, "Adaptive Playout Mechanisms for Packetised Audion Applications in the Wide-Area Network", *Proc. IEEE Infocom'93*, 1994.

[Rowe,92] Rowe, L., and Smith, "A Continuous Media Player", *Proc. Third International Workshop on Network and Operating System Support for Digital Audio and Video,* SanDiego, 1992

[Saltzer,84] Saltzer, J., Reed, D., and D. Clark,"End-to-end Arguments in Systems Design", ACM Trans. on Computer Systems, Vol. 2., No. 4.

[Schulzrinne,95] Schulzrinne, H. and S. Casner, "RTP: A Transport Protocol for Real-Time Applications", Work in Progress, Internet Draft, <draft-ietf-avt-rtp-05.ps>, 1995.

[Scott,92] Scott, A.C., Shepherd W.D. and A. Lunn, "The LANC - Bringing Local ATM to the Workstation", *4th IEE Telecommunications Conference*, Manchester, UK, 1993, August 1992.

[Shacham,92] Shacham, N, "Multipoint Communication by Hierarchically Encoded Data", Proc. IEEE INFOCOM'92, Florence, Italy, Vol.3, pp. 2107-2114.

[Shenker,93] Shenker, S., Clark, D., and L. Zhang, "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.

[Shenker,95a] Shenker, S., and C. Partridge, “Nework Element Service Specification Template”, Working Draft, draft-ietf-intserv-predictive-svc-00.txt, November 1995.

[Shenker,95b] Shenker, S., and C. Partridge, “Specification of Predictive Quality of Service”, Working Draft, draft-ietf-intserv--svc-template.02.txt, November 1995.

[Sluman,91] Sluman, C., "Quality of Service in Distributed Systems", BSI/IST21/-/1/5:33, British Standards Institution, UK, October 1991.

[Steenstrup,92] Steenstrup, M., “Fair Share for Resource Allocation”, pre-print.

[Tanenbaum,88] Tanenbaum, A.S., van Renesse, R., van Staveren, H. and S.J. Mullender, “A Retrospective and Evaluation of the Amoeba Distributed Operating System”, *Technical Report*, Vrije Universiteit, CWI, Amsterdam, 1988.

[Tennenhouse,90] Tennenhouse, D.L., "Layered Multiplexing Considered Harmful", *Protocols for High-Speed Networks*, Elsevier Science Publishers (North-Holland).

[Tennenhouse],94 Tennenhouse, D. L., Adam J.F., Carver, D., Houh, H.H., Ismert M., Linblad, C.J., Stasior, W., Wetherall, D., Bacher D., and T. Chang, “A Software OrinetedApproach to the Design of Media Processing Environmnents,” Proc. IEEE International Conference on Multimedia Computing and Systems, Boston, 1994.

[TINAC,95a] TINA-C, “The DPE Kernel”, Internal Technical Report,1995.

[TINAC,95b] TINA-C, “The QoS Framework”, Internal Technical Report,1995.

[Tokuda,92] Tokuda, H., Tobe, Y., Chou, S.T.C. and Moura, J.M.F., "Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network", *ACM Computer Communications Review*, 1992.

[Tokuda,92] Tokuda, H., Tobe, Y., Chou, S.T.C. and Moura, J.M.F., "Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network", *Proc. ACM SIGCOMM ‘92*, Baltimore, Maryland, USA, August 1992.

[Tokuda,93] Tokuda H. and T. Kitayama ,”Dynamic QOS Control Based on Real-Time Treads” *Proc. Fourth International Workshop on Network and Operating System Support for Digital Audio and Video*, Lancaster University, Lancaster LA1 4YR, UK, 1993.

[TOMQAT] TOMQAT, Deliverables, ftp://ftp.fokus.gmd.de/pub/race/tomqat

[Topolcic,90] Topolcic, C., “Experimental Internet Stream Protocol, Version 2 (ST-II)”, *Internet Request for Comments No. 1190* RFC-1190, October 1990.

[Tran]Tran, V., and T. Bradley Maples, “An Adaptive Model for Real-Time Management of Quality of Service in the OSI Reference Model”, ICC’95, Seattle, 1995.

[Turletti,93] Turletti, T, (1993), "A H.261 Software Codec for Video-conferencing over the Internet", INRIA Technical Report 1834, France.

[Turner,95] Turner, J. “ATM-Soft: A Mini-Proposal for ATM Network Control Using Soft State”, Technical Note, Washington University, 1995.

[Vogel,94] Vogel, A., G. v. Bochmann, R. Dssouli, J. Gecsei, A. Hafid and B. Kerherve, “On QoS Negotiation in Distributed Multimedia Application”, Proc. Protocol for High Speed Networks, April 1994.

[Vogel,94] Vogel, A.,Bochmann, G. v., Dssouli, R., Gecsei, J. and B. Kerherv, “Distributed Multimedia Applications and Quality of Service - A Survey”, IEEE Multimedia, 1994.

[Volg,95] Volg, C., Wolf, L., Herrtwich, R. and H. Wittig, “HeiRAT - Quality of Service Management for Distributed Multimedia Systems”, Multimedia Systems Journal, November 1995.

[Wolfinger,91] Wolfinger, B. and M. Moran, "A Continuous Media Data Transport Service and Protocol for Real-time Communication in High Speed Networks." *Second International Workshop on Network and Operating System Support for Digital Audio and Video*, IBM ENC, Heidelberg, Germany, 1991.

[Wolfinger,91] Wolfinger, B., and M. Moran. "A Continuous Media Data Transport Service and Protocol for Real-time Communication in High Speed Networks." *Second International Workshop on Network and Operating System Support for Digital Audio and Video*, IBM ENC, Heidelberg, Germany, 1991.

[Yeadon,93] Yeadon, N.J., “Supporting Quality of Service in Multimedia Communications via the Use of Filters”, *Internal Report* No. MPG-94-10 Department of Computing, Lancaster University, Lancaster LA1 4YR. March 1993.

[Yeadon,94] Yeadon, N., Garcia, F., Campbell, A and D. Hutchison, (1994), “QoS Adaptation and Flow Filtering in ATM Networks”, 2nd International Workshop on Advanced Teleservices and High Speed Communication Architectures, Heidelberg, Germany.

[Yeadon,95] Yeadon, N., “Continuous Media Filter Operations in Heterogenious Networks”, PhD Thesis, Department of Computing, Lancaster University, Lancaster LA1 4YR, UK, 1995 (in preparation).

[Yeadon,95] Yeadon, N., “X filters”, PhD Thesis, Department of Computing, Lancaster University, Lancaster LA1 4YR, UK, December 1995, (in preparation).

[Zhang,91] Zhang, H., and S. Keshav, "Comparison of Rate-Based Service Disciplines" *Proc. ACM SIGCOMM '91*, Zurich , August 1992.

[Zhang,93] Zhang, L., Deering, S., Estrin, D., Shenker, S and D. Zappala, "RSVP: A New Resource ReSerVation Protocol", *IEEE Network*, September 1993.

[Zhang,94] Zhang, L., Symposium on Multimedia Networking, Columbia University, USA.

[Zhang,95] Zhang, L., et. al., "RSVP Functional Specification", Working Draft, draft-ietf-rsvp-spec-07.ps.

[Zinky,95] Zinky, J., Bakken, D., R. Schantz, "Overview of Quality of Service for Distributed Objects" , Technical Report, BBN Systems and Technologies, Cambridge, 1995.

[Zitterbart,92] Zitterbart, M., Stiller, B., and A Tantawy,"A Model for Flexible High-Performance Communication Subsystems", IEEE JSAC, May 1992.

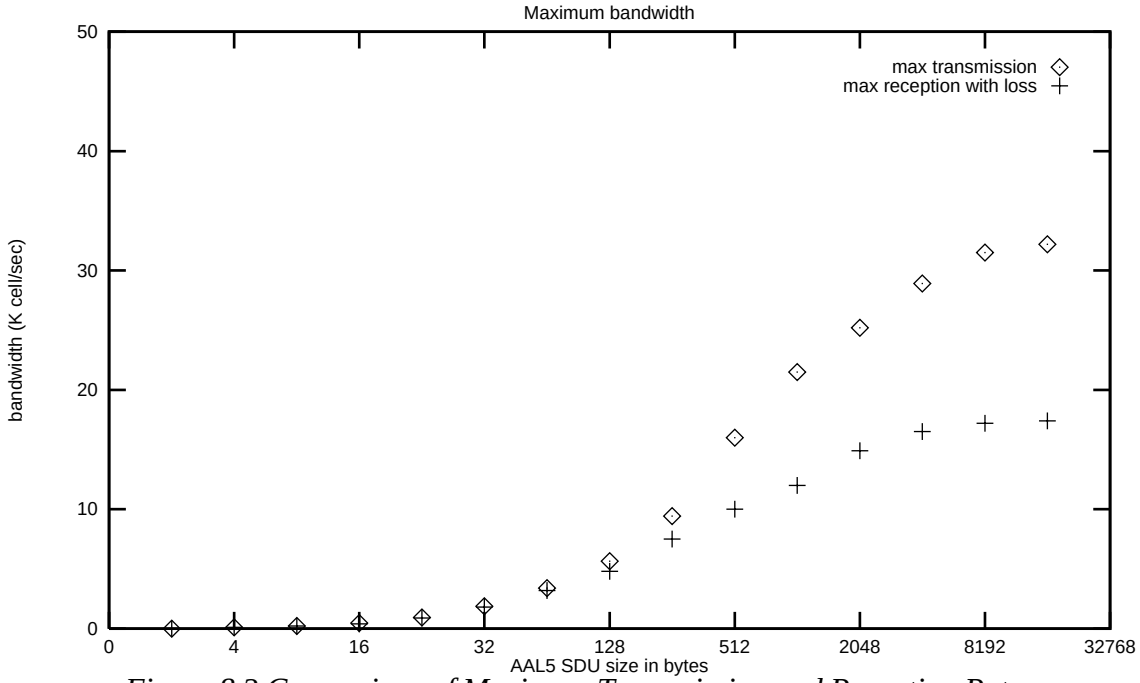

Figure 8.3 Comparison of Maximum Transmission and Reception Rates

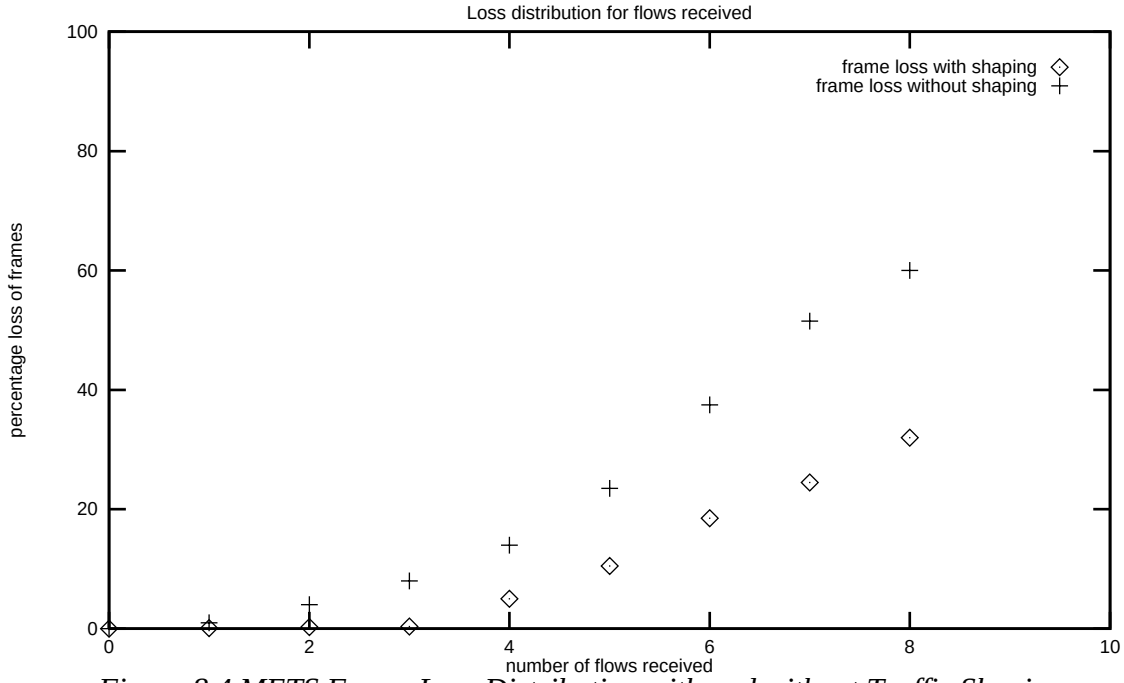

Figure 8.4 METS Frame Loss Distribution with and without Traffic Shaping

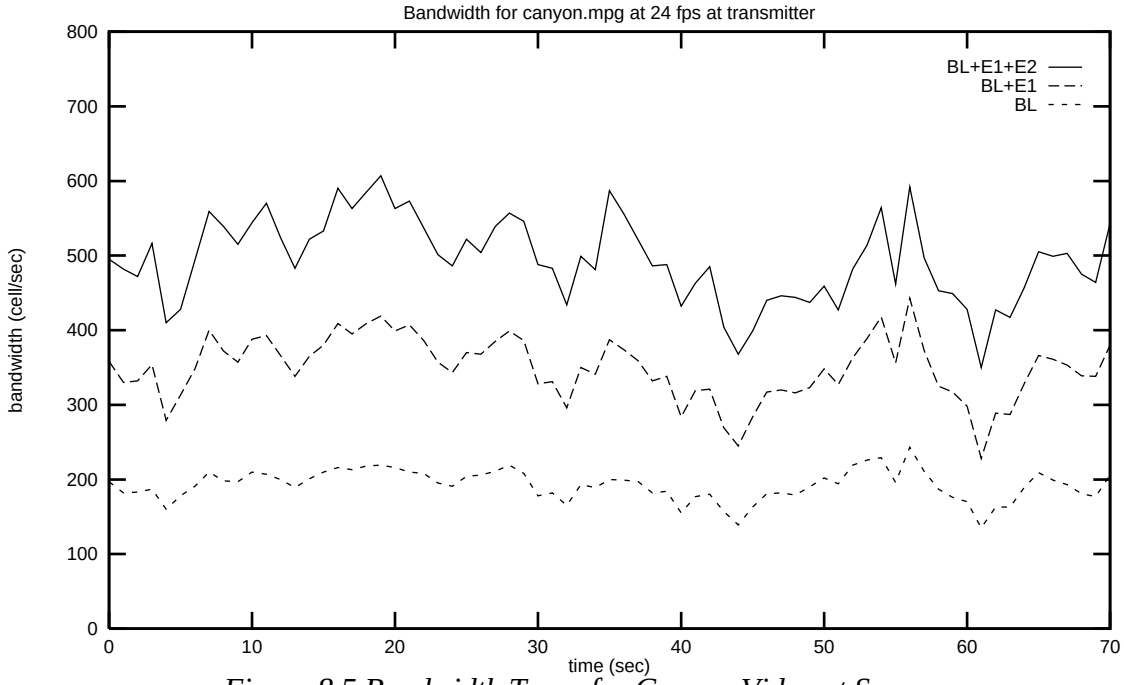

Figure 8.5 Bandwidth Trace for Canyon Video at Server

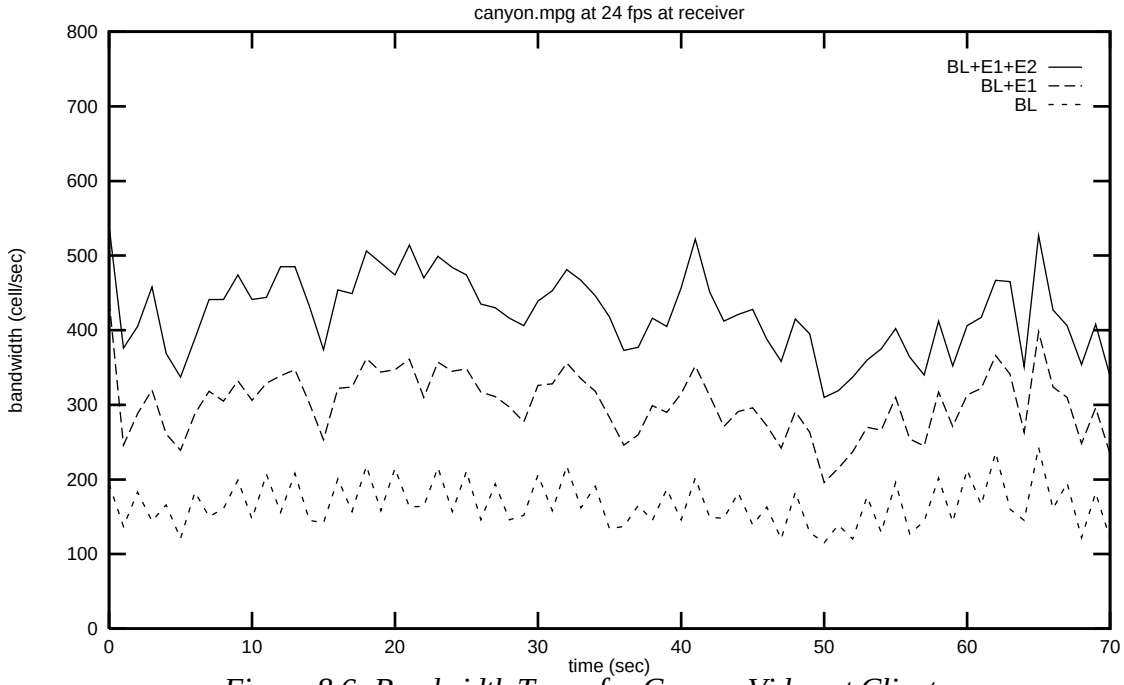

Figure 8.6: Bandwidth Trace for Canyon Video at Client

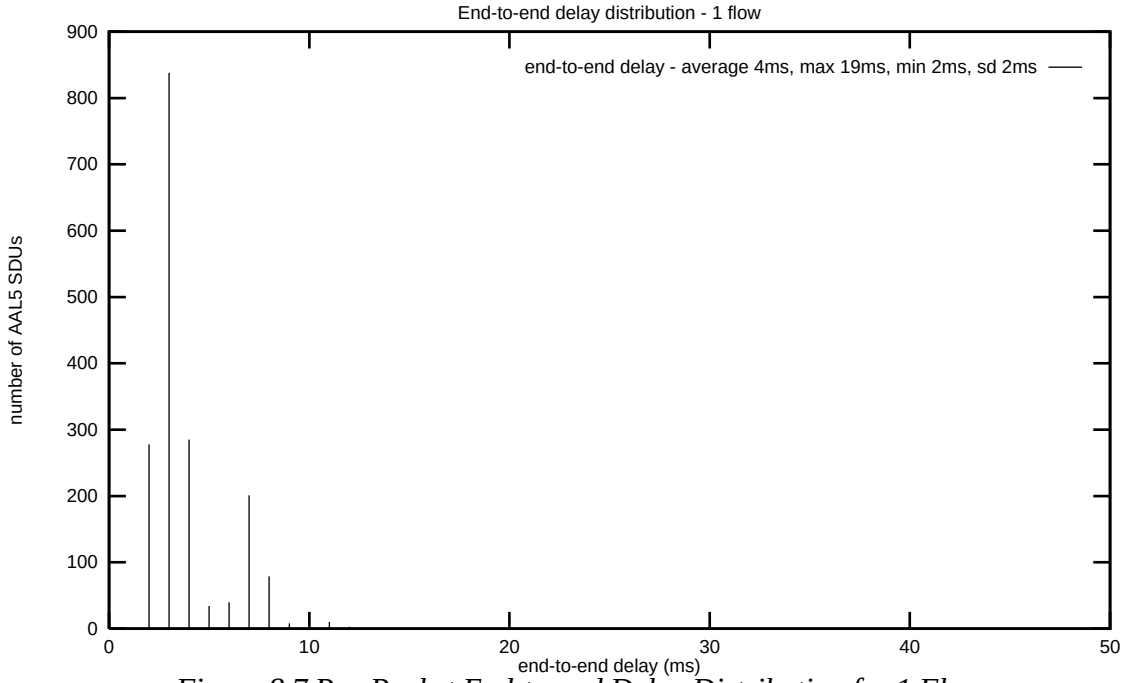

Figure 8.7 Per-Packet End-to-end Delay Distribution for 1 Flow

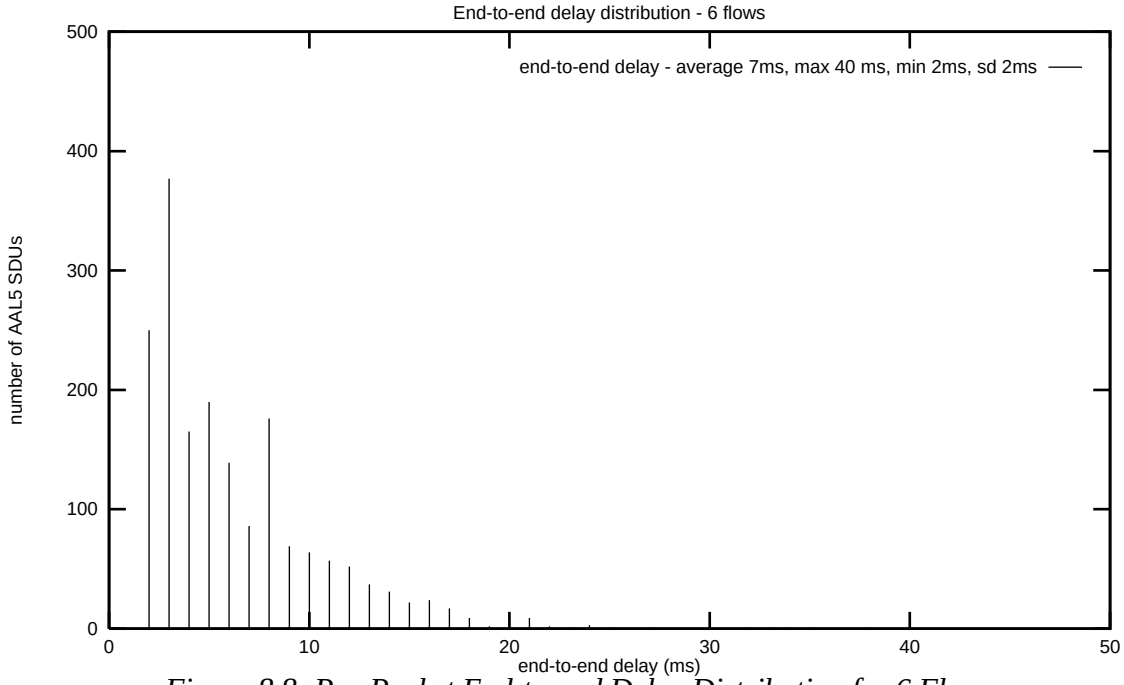

Figure 8.8: Per-Packet End-to-end Delay Distribution for 6 Flows

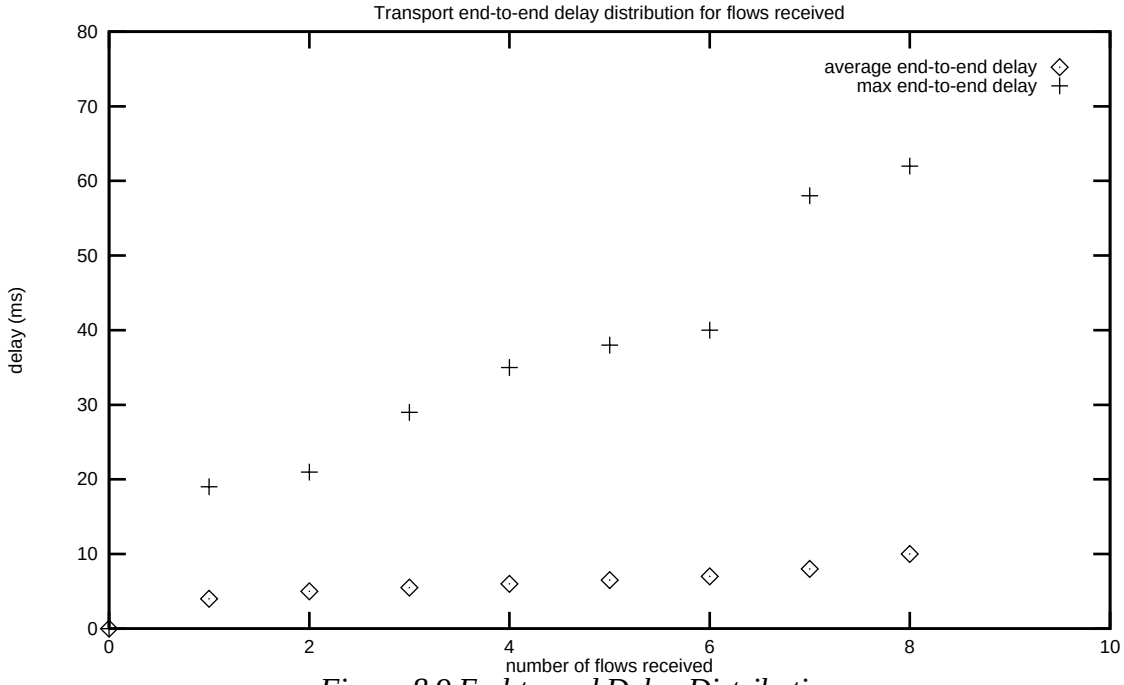

Figure 8.9 End-to-end Delay Distribution

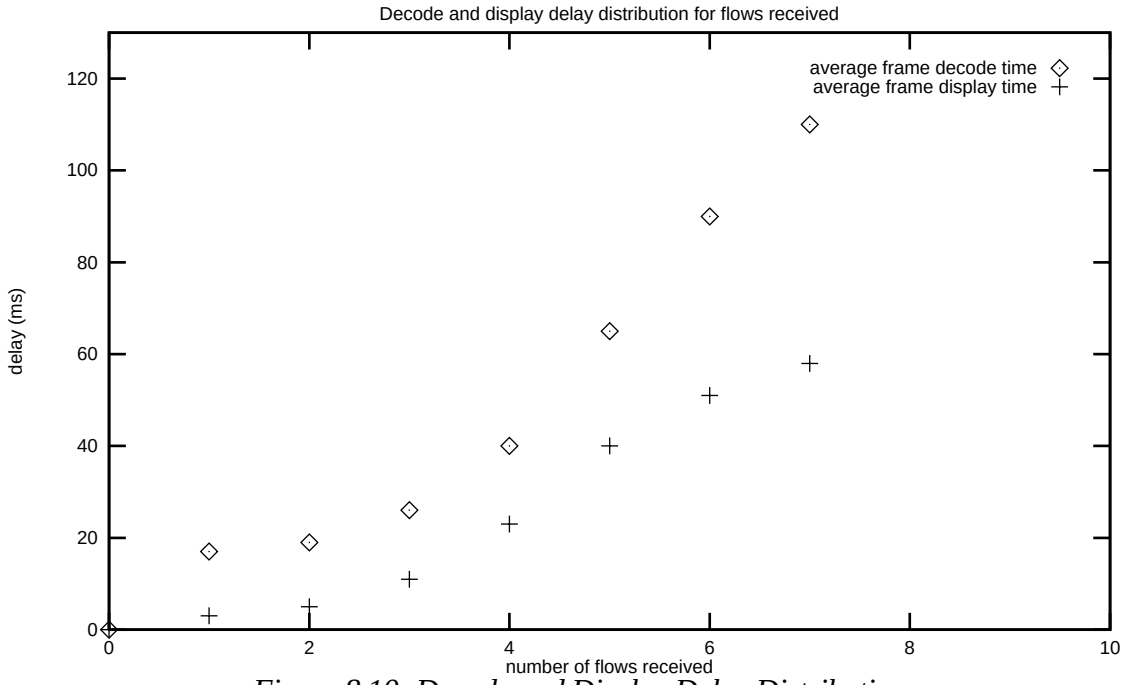

Figure 8.10: Decode and Display Delay Distribution

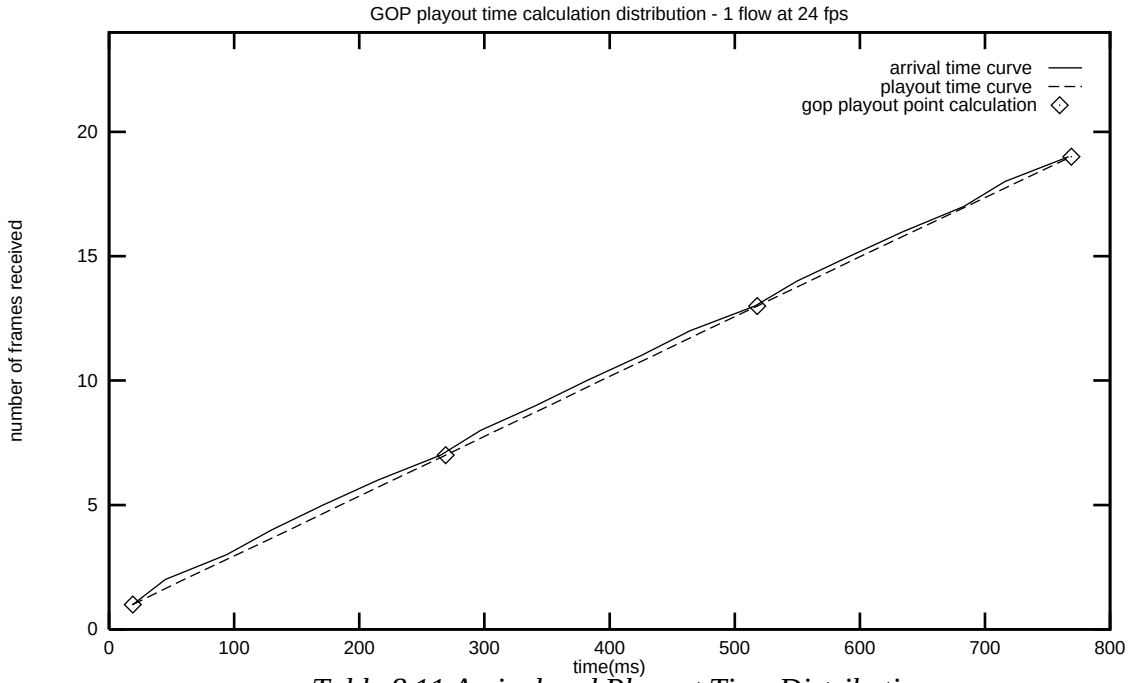

Table 8.11 Arrival and Playout Time Distribution

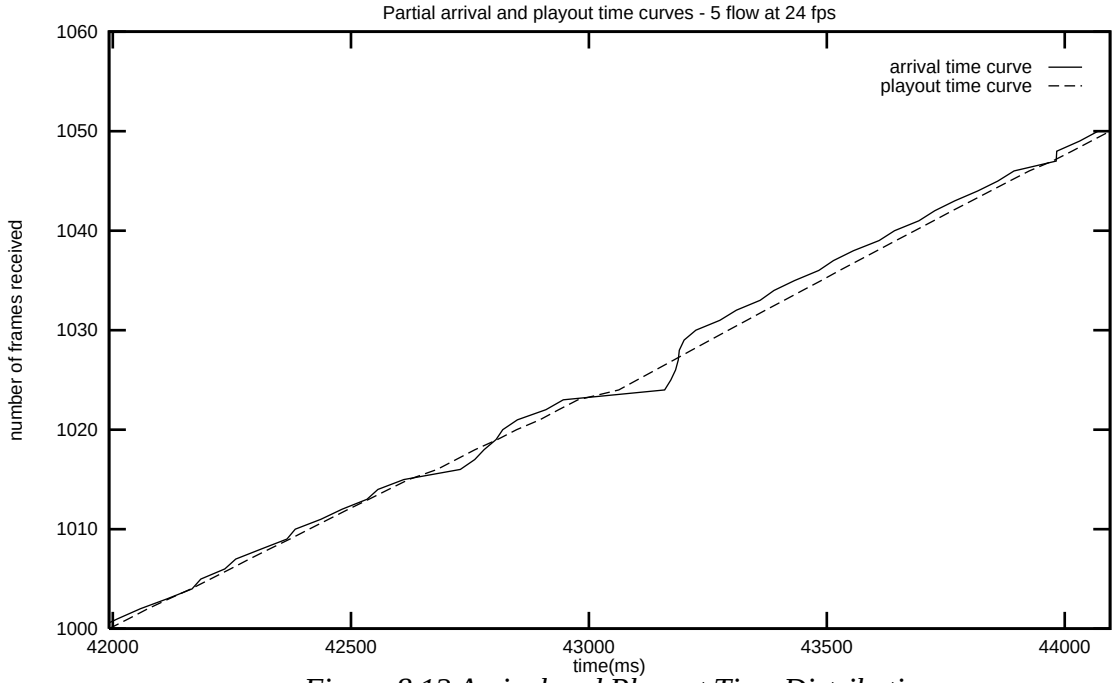

Figure 8.12 Arrival and Playout Time Distribution

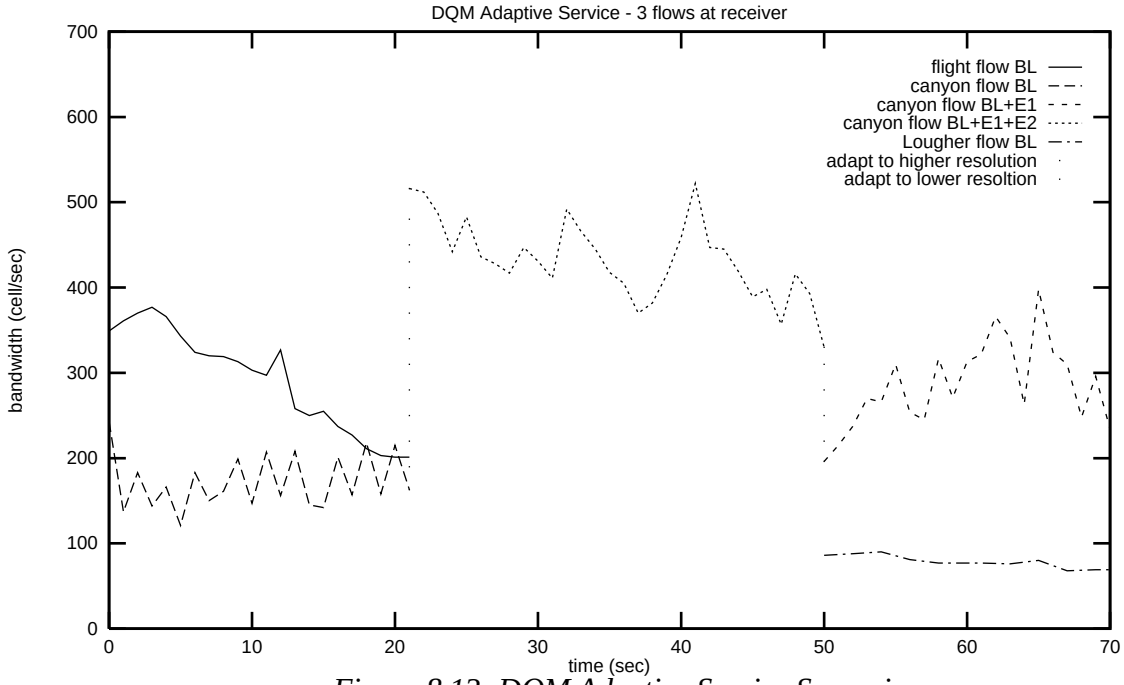

Figure 8.13: DQM Adaptive Service Scenario