A QUALITY OF SERVICE ARCHITECTURE

Andrew Campbell, Geoff Coulson and David Hutchison

Department of Computing,
Lancaster University,
Lancaster LA1 4YR, U.K.
E.mail: mpg@comp.lancs.ac.uk

Abstract. *For applications relying on the transfer of multimedia, and in particular continuous media, it is essential that quality of service (QoS) is guaranteed system-wide, including end-systems, communications systems and networks. Although researchers have 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. To address this deficiency, we are developing a Quality of Service Architecture (QoS-A) which offers a framework to specify and implement the required performance properties of multimedia applications over high-performance ATM-based networks. The QoS-A incorporates the notions of flow, service contract and flow management. 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. This paper provides an overview of the QoS-A and focuses particularly on the role of the transport service and protocol in the architecture. We describe a multimedia enhanced transport service (METS), flow management and transport layer service contract. We show how QoS levels contracted at the transport service interface can be assured in the context of our local ATM environment.*

1. Introduction

Recent technological developments in high speed networks and multimedia workstations are making possible entirely new classes of distributed application such as distance learning, desktop video-conferencing and remote multimedia database access. In these applications, communication requirements are extremely diverse and demand varying levels of latency, bandwidth and jitter, etc. Furthermore, for *continuous media* such as video and audio it is often a requirement that levels of service are *guaranteed*. Other time critical distributed applications such as distributed real-time control systems are also growing in prominence: e.g. in the OSI Time Critical Communication Architecture (TCCA) forum. These applications have stringent quality of service (QoS) requirements for both reliability and guaranteed bounds on message latency.

In most existing communications architectures, however, the notion of QoS is extremely narrow. The Internet protocol IP, for example, only permits the specification of *qualitative* QoS hints (using the type of service field in the IP header) such as 'low' delay, 'high' throughput and 'high' reliability and even these limited QoS specifications are rarely honoured by the underlying network. Furthermore, existing architectures are based on a best effort performance model and were never designed to support *quantitative* QoS. In the Internet the support of reliable data transfer was a primary design goal and performance QoS was only a marginal consideration.

A further limitation of most current architectures is the *static* nature of service provision. In OSI protocols, the value of a QoS parameter remains the same through the lifetime of a connection: i.e. once negotiated a QoS parameter is never re-negotiated. One implication of this is that users cannot dynamically adjust the connection QoS without undergoing a disconnection/re-establishment phase or opt for trade-offs in the face of limited resources. For example, users cannot choose to scale back the quality of an existing video connection from colour to monochrome to allow the possibility of opening a new audio connection. Another implication is that the service-provider is committed to provide the QoS over the lifetime of the connection. If the provider is unable to maintain its commitment there is no mechanism to inform the user and allow her to request a suitably lower QoS. The only option is for the provider to unilaterally close the connection.

To address these deficiencies we are designing an integrated quality of service architecture (QoS-A) which spans both end-systems and networks and takes the support of performance QoS for a wide range of applications as its primary goal. The QoS-A 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 service classes are designed to fit into a highly dynamic application environment and thus provide facilities such as performance monitoring, notification of QoS degradation and in service QoS re-negotiation in addition to the traditional facilities.

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 configurable protocols and mechanisms in both the end-system and the network. In end-systems, these include thread scheduling, buffer allocation, jitter correction and co-ordination over multiple related connections [Campbell,92a]. In communications systems, protocol support such as end-to-end QoS negotiation, re-negotiation and indication of QoS degradation are required [Boerjan,92]. In networks, suitable resource reservation protocols [Zhang,93] and service disciplines in switch queues are needed [Keshav,91] [Parekh,92] [Hyman,93] . The QoS-A also provides a framework for the maintenance and management of QoS over all system layers. This includes management functions such as admission control for new connections and monitoring to ensure that QoS levels are being maintained by the service provider.

This paper describes a QoS-A, primarily focusing on the transport layer. Section 2 provides an overview of the QoS-A and introduces the central notions of *flow* and *flow management*. Section 3 then describes a multimedia enhanced transport service (METS) interface based on the notion of a *service contract* agreed between the transport service user and the network provider. Following this, section 4 describes the means by which services contracted at the transport layer are realised in terms of mechanisms. These include the transport protocol itself together with a low level transport QoS manager and an overseeing flow management entity. In section 5 we present our ATM based implementation of a QoS-A using a QoS enhanced Chorus micro-kernel for end-system support. An extended related work section for the reader's interest follows in section 6., and finally in section 7. we presents our conclusion and future work.

2. Quality of Service Architecture

2.1 QoS-A Model

The QoS-A [Campbell,93a] is a layered architecture of services and mechanisms for QoS management and control of continuous media flows in multiservice networks. The most fundamental architectural concept we use is the notion of a *flow*. A flow characterises the production, transmission and eventual consumption of a single media stream as an integrated activity governed by a single statement of QoS. Flows are always simplex but can be either unicast or multicast. They may carry a range of data types including both continuous media and control data such as messages or RPC packets. The realisation of the flow concept demands active QoS management and tight integration between the device management, thread scheduling, communications protocol and network components of the end-to-end data path.

In functional terms, the QoS-A illustrated in Figure 1 is broadly divided into a number of layers and planes. The upper layer consists of a *distributed applications platform* augmented with services to provide multimedia communications and QOS configuration 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 and jitter correction [Campbell,92a]. Supporting this is a *transport layer* which contains a range of QoS configurable protocols. For example, separate protocols are provided for continuous media and constrained latency message protocols [García,93].

The vertical planes in the QoS-A, of which there are three, are as follows:

i) the protocol plane

This consists of a *user plane* and a *control plane*. In our architecture we use separate protocol profiles for the control and data components of flows because of the essentially different QoS requirements of control and data. Control generally requires a low latency full duplex assured

service whereas multimedia data generally requires a range of non-assured, high throughput simplex services.

ii) the QoS maintenance plane

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. 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.

iii) the flow management plane

This is responsible for *flow establishment* (including flow admission control, resource reservation and QoS based routing), *QoS re-negotiation*, *QoS mapping* (which translates QoS representations between layers) and *QoS adaptation* (which implements coarse grained QoS maintenance control).

The flow management projection of the architecture (the shaded section of Figure 1) illustrates the relationship between the three planes which work together to monitor and maintain end-to-end QoS. This aspect of the QoS-A will be further described in section 4 with particular emphasis on the transport layer

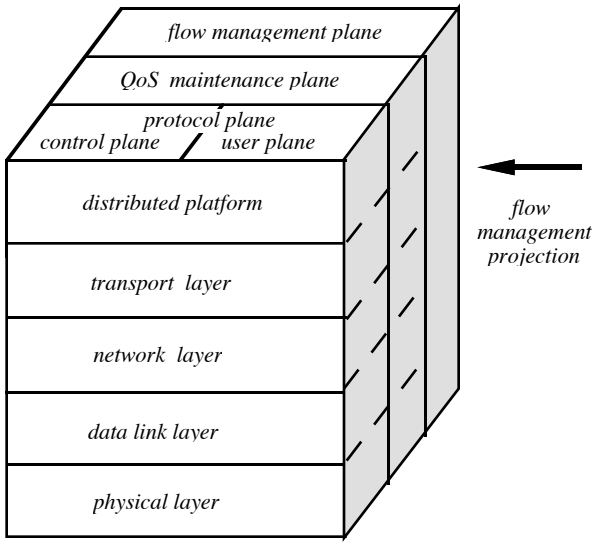

Figure 1: Quality of Service Architecture

3. A Multimedia Enhanced Transport Service

The success of ATM-based networks is dependent upon the availability of new 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 multicast for group distribution.

A crucial 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 flowConnectRequest( tsap_t *source, *sink; service_contract_t *QoS);
```

This call requests 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*. Note, that the sink tsap may also represent a group address to accommodate multicast flows [García,93].

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;
- connection_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 together into a C structure as follows:-

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

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

3.1 flow_spec_t

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 Kbits/s), a low degree of reliability (10^{-1}), a stringent upper bound on end-to-end delay (e.g. 100 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; see [Campbell,93a] for more details.

3.2 commitment_t

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) *deterministic*, 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 { deterministic, statistical, best_effort } commit_t;

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

typedef struct {
    class_t    throughput;
    class_t    loss;
    class_t    delay;
    class_t    jitter;
} commitment_t;
```

Note that the commitment structure allows separate specification of the commitment required of each of the performance parameters. The motivation behind this choice is that applications often need to treat commitment on different performance parameters as orthogonal. For example, file transfer may require a deterministic bound on loss (commitment.loss.class = deterministic) but only best effort on throughput (commitment.throughput.class = best_effort). A real-time control application, on the other hand, may require deterministic hard real-time guarantees on both loss and delay. A deterministic bound on delay (commitment.delay.class = deterministic) is a statement that no end-to-end delays will exceed the amount specified in the flow spec.

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 deterministic and statistical commitments is that the deterministic 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 the QoS-A pricing policy must encourage the user to select statistical commitment over the deterministic 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 tests see [Robin,94].

3.3 adaptation_t

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 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, disconnect from the service or take no action).
To meet this requirements, we use the adaptation_t structure:-

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

typedef enum { renegotiate, indication, disconnect_flow, 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, indication, no_action, 0},
                       {throughput, indication, renegotiate, &newFlowSpec},
                       {loss, no_action, no_action, 0},
                       {jitter, indication, no_action, 0}};
```

The user is informed of QoS degradations in one of the following ways: *(i)* via 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 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 event qos_degradation_indication(event, measured_value, required_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_degradation_indication and a qos_renegotiation_indication, then 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 re-negotiation.

3.4 maintenance_t

The options available in the service contract for control over QoS maintenance are as follows:-

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

The monitor option instructs the QoS-A to periodically deliver measured performance assessments relating to the specified flow. The maintain option, on the other hand, attempts to transparently exert fine grained corrective action (e.g. thread scheduling [Coulson,93], communication buffering, flow regulation and scheduling, queueing delays) to maintain QoS levels according to the service contract, but does not deliver periodic assessments. In both cases, coarse corrective action (i.e. re-negotiation),

should the contracted QoS drop below the prescribed levels, may be taken depending on the selected adaptation_t option.

The default case is to maintain deterministic and statistical flows, but to ignore the achieved service of best effort flows. Finally, the no_maintenance option explicitly disables maintenance. However, users can still choose to asynchronously solicit flow assessment updates on demand.

3.5 connection_t

Connection oriented transport and network protocols can support full end-to-end *negotiated* service [García,93] and in some cases, a *fast* connect service [Danthine,92] where the reservation and data transfer phases coincide. In addition to these connection 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.

We define three connection_t classes 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 or 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 we remain somewhat sceptical about enforcing such a regime upon the user. We include the duration time in the service as a marker for further study. In [Ferrari,92] an advance reservation mechanism is described whereby the network may allow the user an extension after the specified duration has expired. This is achieved without disruption to other users who have pre-reserved resources.

3.6 cost_t

The QoS-A project has not yet addressed the issues of cost and tariffing in any great detail, as we have been mainly concerned with a realisation of the architecture in a local ATM environment. However, even in a local environment, cost is still likely to be an important factor. If there is no notion of cost 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 to select 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 predicts the effective bandwidth requirement is realised by admitting of more flows, in addition to optimising the network resources. The penalty for over estimation of connection's bandwidth is potentially a reduction in QoS provided.

4. Flow Management at the Transport Layer

In this section we focus on mechanisms to realise the transport service interface described above. The mechanisms are embedded in the QoS-A flow management projection of Figure 1. The flow management projection is shown in more detail in Figure 2. Note that Figure 2 only shows the transport layer and below; the upper layers have been omitted for clarity.

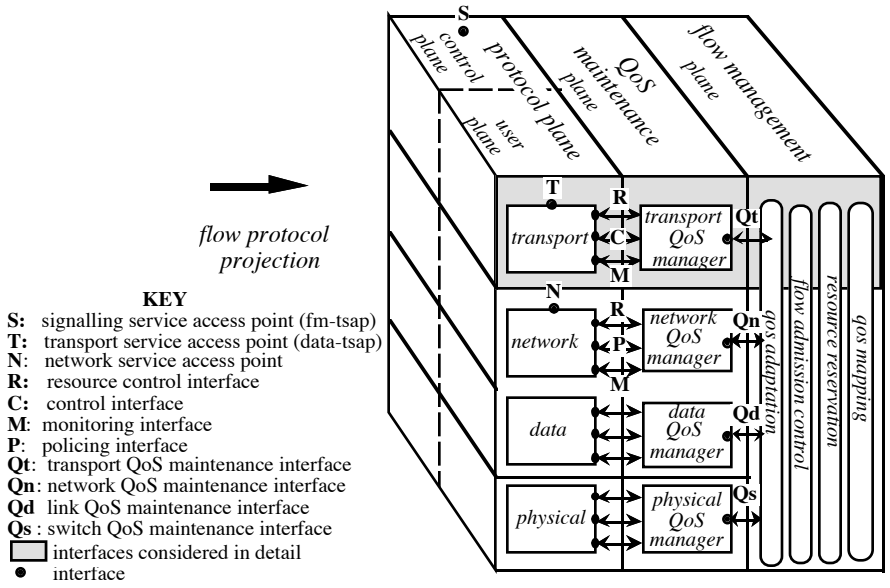

Figure 2: Flow Management Projection of QoS-A

4.1 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 /signalling (*fm-tsap*) and data (*data-tsap*) components (there are also equivalent primitives at the network layer) as illustrate in Figure 2:-

- the fm-tsap interface contains primitives: get_tsap, free_tsap, flowConnectRequest, qos_renegotiation, qos_degradation, qos_report, monitor_flow, flow_assessment, maintain_flow and flowDisconnetRequest.
- the data-tsap interface contains primitives: data_request, data_indication, data_response and data_confirm.

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 2 and Figure 3:-

- 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;

4.2 User Plane

Our transport protocol is based on a continuous media protocol developed by [García,93]. The protocol provides an ordered but non-assured, connection oriented communication service and features resource allocation based on the user's QoS specification. It allows the user to select upcalls for the notification of corrupt and lost data at the receiver, and also allows the user to re-negotiate QoS levels.

It is the responsibility of the protocol to share communications resources in end-systems among flows with widely different QoS requirements. To meet this need the protocol incorporates buffer sharing, rate regulation, scheduling, and basic flow monitoring modules. 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 copies across layers [Hehmann,91] and uses separate pools for each commitment type [Robin,94]. Deterministic flows each receive a fixed buffer allocation based on the flows peak rate whereas statistical flows share a common pool. For deterministic and statistical flow's 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 (see also Figure 3).

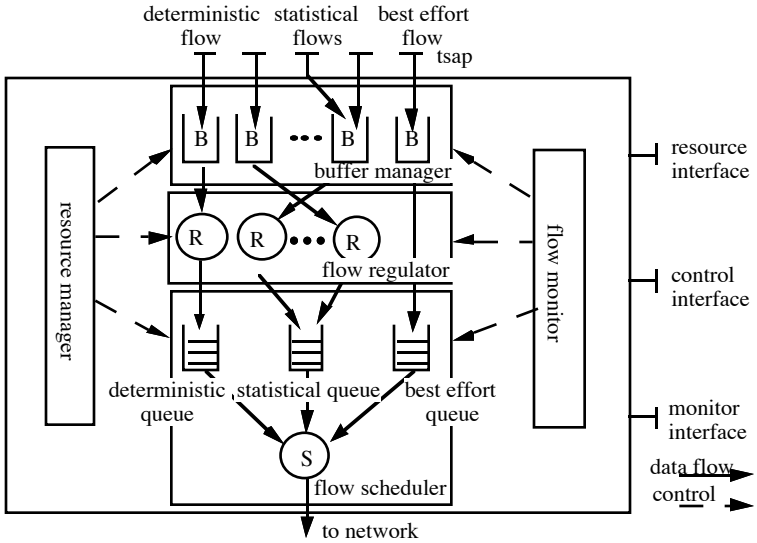

Figure 3 : Transport Protocol Mechanisms and Interfaces

Flow Regulator. The transmission of frames to the network must be regulated to prevent buffer overflow at receivers and rate violations 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 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 deterministic and statistical are given eligibility times; best effort flows are sent at a rate determined by the workload of the scheduler (see below). Figure 3 shows that a flow can be viewed as a stream of data taking a specific path through the sequence of buffers, regulators and scheduling queues.

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 deterministic and statistical flows (as part of flow admission control schedulability test) we ensure 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 hierarchical deadline scheduling. Three scheduler communication service queues are used [Ferrari,92]: one each for deterministic, statistical and best effort flows. The deterministic 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 deterministic queue, second to the statistical queue and lastly to the best effort queue. The scheduler services frames from deterministic and statistical queues using a non pre-emptible discipline.

Flow Monitor. In addition to the above described mechanisms, the protocol includes a component which gathers statistical information on the ongoing flow of data, both at the source and the sink of the transport connection. This information is used by the transport QoS manager in the QoS maintenance plane. The transport QoS manager and transport protocol interact over the resource, control and monitor interfaces described in section 4.1 and illustrated in Figure 2 and Figure 3.

In essence, the transport protocol *monitors* a flow's on-going performance and the transport QoS manager *maintains* it. Flow monitoring is initiated when a start_monitor command is received by the transport protocol from the transport QoS manager at the protocol's monitor interface. When monitoring is enabled, the source transport entity monitors the flow and records its transmission statistics as data is injected into the network. Measurements are made over a predefined flow measurement interval which is 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). The receiving protocol entity also monitors the achieved performance of the frame during the same interval. Based on its own measurements and the information from the receiver, the sending transport entity periodically compiles a qos_assessment message which is passed on to the transport QoS manager at the end of an assessment period called an *era*[1] [Nahrstedt,93]. The choice of the era period is critical. The QoS feedback in the qos_assessment report needs to be temporally correlated with the on-going flow to be useful. For example Kanakia et. al. [Kanakia,93] show that variable rate encoders can track QoS variations as long as the QoS feedback arrives within four frame times or less. Depending upon measured performance the transport QoS manager may exert some fine grained corrective action on the on-going flow. .

The information passed periodically from the receiver to the sender is carried in a control message which is already a part of the rate based transport protocol [García,93], the format of this message is as follows:-

```
typedef struct {
   int   frame_seqno;      /* Seqno of frame which generated msg */
   int   time_stamp_echo;  /* echo timestamp received in frame */
   int   measured_delay;   /* end-to-end delay */
   int   measured_jitter;  /* jitter */
   int   measured_loss;    /* frame loss rate */
   int   measured_rate;    /* bytes received */
   int   interval;         /* measurement interval */
   int   rtimer;           /* burst rate */
   int   burst;            /* max number of bytes per segment */
   int   credits;          /* available buffer segments at receiver*/
} control;
```

The transport protocol's monitoring mechanism is able to build up a statistical representation of the end-to-end QoS using the performance data supplied in control messages. The resulting flow statistics represents the actual end-to-end QoS experienced by the receiver. The qos_assesment message is partitioned into sender and receiver QoS statistics; these include delay, jitter, throughput and loss rate measured during the era.

Resource Manager. The resource interface gives access to the protocol's buffer management, regulation and scheduling functions which are used both during the flow establishment time and QoS re-negotiation time. During flow establishment the various mechanisms are configured in accordance with the flow spec and commitment clauses in the service contract. Table 1 shows the performance parameters together with the mechanisms required for their support and the resource configuration required for the three types of service commitment.

---

[1] The transport protocol receives control messages based on an interval. QoS assessments are generated by the protocol based on an era which must be greater or equal to the control message interval.

Deterministic 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 deterministic 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

		commitment class		
QoS parameter from flow_spec	**QoS mechanism**	**deterministic**	**statistical**	**best effort**
Loss	Buffer Management	fixed buffer allocation. based on peak_rate	shared buffers based on average rate	no guaranteed buffer allocation
Throughput	Regulation	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

Table 1: Commitment Class Based Resource Reservation

4.3 QoS Maintenance Plane

The protocol and 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 3.4), the transport QoS manager operates as follows. In both the *maintain* and the *monitor* policies the manager receives periodic QoS assessment messages generated by the transport protocol. The monitor policy merely passes these messages on to the flow management plane, whereas the maintain policy attempts to actively uphold the end-to-end performance parameters via a *monitor-measure-adjust* loop which measures the receiver QoS against the sender QoS in the assessment messages, and, if necessary, makes fine grained 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. From the sender/receiver measurement pair (e.g. s_measured_delay/ r_measured_delay, etc.) the manager determines which, if any, of the performance parameters have degraded and then tunes the appropriate resource(s). If the manager fails to recover from a QoS degradation, then a qos_alert message is sent to the flow management plane which may take appropriate action as discussed below in section 4.4. In the *no_maintenance* policy, no explicit action is taken by the QoS manager (although the manager will request QoS assessment messages from the protocol when explicitly asked to by the flow management plane).

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 deterministic flows or best effort flow. The former have resources exclusively dedicated to them and the latter are not maintained by definition. However, we anticipate service fluctuations from time to time in statistical flows because they share transport and network communication resources.

4.4 Flow Management Plane

The flow management plane is responsible for a number of static and dynamic QoS control functions. The major functions, to be described below, consist of the provision of a network signalling infrastructure, a resource reservation, and QoS adaptation for the realisation of course 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, the support of the forward connection mode (see section 3.5), and the provision of a history function to gather network loading statistics. More detail of these subsidiary functions can be found in [Campbell,93a].

Flow Reservation. A major part in flow establishment is the reservation of resources in the source and sink nodes and in the network, according to the requirements of the user supplied service contract. Resource reservation allocates resources in accordance with the QoS commitment specified in the service contract. For a deterministic 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.

For the purposes of efficient flow management and QoS control, we logically partition the network into *flow management domains* [Crosby,93] which constitute arbitrary collections of network devices (network devices can be switches, routers, multimedia workstations, continuous media storage servers, etc.). In each domain, the network aspects of the flow management resource reservation function are realised as a single network server which we call a *resource server* [Campbell,93b].

A number of architectural choices were possible for resource server realisation. One proposal in the literature [Cidon,92] advocates a fully distributed architecture capable of making resource management decisions such as flow admission control at any node. This has the advantage of reducing connection setup time because all the required state is available locally. However, corresponding disadvantages include the latency involved in 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 of maintaining database integrity while simultaneously avoiding bottlenecks introduced by an excessively centralised solution.

The resource server, as the central resource controller for its flow management domain, is the arbiter of communication resource allocation requests. For the *negotiated* service, when a flowConnectRequest is issued by a transport user, the resource server consults its local representation of domain wide resource availability. If the request can be satisfied, the flow management plane provisionally marks the requested resources as allocated and then multicasts a set_attribute_request primitive to the network QoS managers at all nodes in the data path (plus the transport QoS manager on the sink nodes) to request that they allocate the resources requested. The resource server 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.

The *fast* connect service largely follows the above procedure except that the resource immediately replies to the requester as soon as it determines that the requested resources are available. This connection mode eliminates the latency of the round trip communication with the QoS managers, but at the expense of being an unconfirmed service. The *forward* connection mode is implemented as an additional time dimension in the server's resource table. When a forward request is granted, resources are marked to be allocated at the time specified, and for the duration specified, in the request.

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 *no maintenance* the flow management plane takes no explicit action; it merely responds to individual user requests for a flow assessment by passing the request on to the QoS manager and returning the corresponding flow_assesment to the user. If the mode is *monitor*, the flow management plane simply receives periodic flow_assessments from the QoS manager and passes them on to the application. If the commitment is *maintain*, the flow management plane does not receive any flow_assessments as the responsibility for flow management has been delegated to the layer specific QoS managers. However, the flow management plane may still receive flow_alerts from the 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 to the user, the initiation of a QoS re-negotiation on one or more specific performance parameter (via set_attributes and free_attributes primitives on the QoS manager), or both. The flow management plane is also responsible for explicitly switching on and switching off maintenance via the start_maintenance and stop_maintenance primitives.

From the application's viewpoint flow maintenance can be initiated at flowConnectRequest time, or at any point during the lifetime of a flow. In latter case the application uses the maintain_flow primitive to dynamically request flow maintenance services.

5. ATM Based Implementation

At Lancaster we are currently evaluating an ATM based QoS-A implementation which uses an extended Chorus micro-kernel for end-system support. In this section we provide an overview of the implementation; a companion paper [Robin,94] describes our ATM implementation in full details.

5.1 Micro-kernel Extensions

Our experimental systems consists of a distributed applications platform and communication suite embedded in a Chorus microkernel augmented with services to provide multimedia communications and QOS configuration in an object-based environment.

The Chorus micro-kernel is implemented using modern techniques such as multi-threaded address spaces and inter-process communication with copy-on-write semantics. The basic Chorus abstractions are *actors*, *threads* and *ports*, all of which by globally accessible identifiers. Actors are address spaces and containers of resources which exits in either user or supervisor space. Threads are units of execution which run code in the context of and actor. Ports are message queues to hold incoming and outgoing messages. Unfortunately, Chorus' real-time support is not fully adequate for the QoS-A requirements, principally because there is no support for QoS specification, resource reservation and QoS commitment.

To remedy its current deficiencies for QoS specification and real-time support, we have extended the Chorus' system call application programmers interface (API) to support suitable QoS-A abstractions derived from section 3:

- *rtports*: these are extensions of standard Chorus ports which are used as tsaps for real-time communications. Rtports have an associated QoS which is realised as part of the flowConnectRequest procedure.
- *rthandlers*: these are user supplied C routines which provide the facility to embed application code in a real-time infrastructure. They are attached to rtports and upcalled on real-time threads by the infrastructure. They encourage an event driven style of programming which is appropriate for real-time applications and also avoid the additional context switches associated with send()/recv() based interfaces.
- *QoS adaptation handlers*: these are upcalled by the infrastructure in a similar way to rthandlers but are used to implement the QoS adaptation policy supplied by the user in the service_contract. This mean that applications which require notification of QoS degradation or re-negotiation (i.e. QoS scaling) must supply an address of a QoS adaptation handler.

Full details of the continuous media API and the scheduling architecture are specified in [Coulson,93]

5.2 Baseline Architecture.

In our implementation we realise the flow management plane as a per end-system *connection manager* supported by a network level *flow management protocol*, as illustrated in Figure 4b. The connection manager and the flow management protocol (FMP) together subsume the flow management and control plane functions (viz. signalling, QoS mapping, resource reservation and admission control, QoS adaptation).

The flow management protocol provides a framework for the management of network resources in both local and wide-area networks using out-of-band signalling. The FMP in the local ATM area has two main 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 itself. The FMP uses a specialised signalling protocol stack consisting of a subset of RSVP [Zhang,93] which we encapsulate in ICMP, and an ATM signalling protocol, called *ATMSig* [Campbell,93b], which is based on a subset of the ATM Forum's user-network interface 2.4 specification [ATM Forum,93].

The user plane is illustrated in Figure 4a. and consists of a connection oriented, rate based, transport protocol [Garcia,93] which provides support for METS [Campbell,93b]. The protocol facilities include

support for QoS configuration, performance monitoring, jitter correction, notification of QoS degradations, and in service QoS re-negotiation as described in sections 4. QoS maintenance is provided at the end-systems by peer transport QoS managers which are tightly coupled to the transport protocol itself. In fact the transport QoS manager is integrated into the existing protocol [García,93].

The IP layer, called IP++, allows us to interwork outside the ATM network in a heterogeneous environment. It offers QoS enhanced facilities along the lines of those proposed in Deering's Simple Internet Protocol Plus (SIPP) [Deering,93]. In particular, IP++ uses a new packet header field called a *flow-id* to identify IP packets as belonging to a particular *flow* or connection, and a *flow-spec* [Campbell,93b] to define the QoS associated with each flow. Flow-specs are held by IP++ routers and used to determine the resources dedicated to the router's handling of each IP++ packet on the basis of its flow-id. The state held by routers is initialised at resource reservation time by the flow management protocol. Below the IP layer we use an ATM Adaptation Layer service to perform segmentation and reassembly of IP packets into/from 53 byte ATM cells. We use a minimal AAL5 service for this purpose.

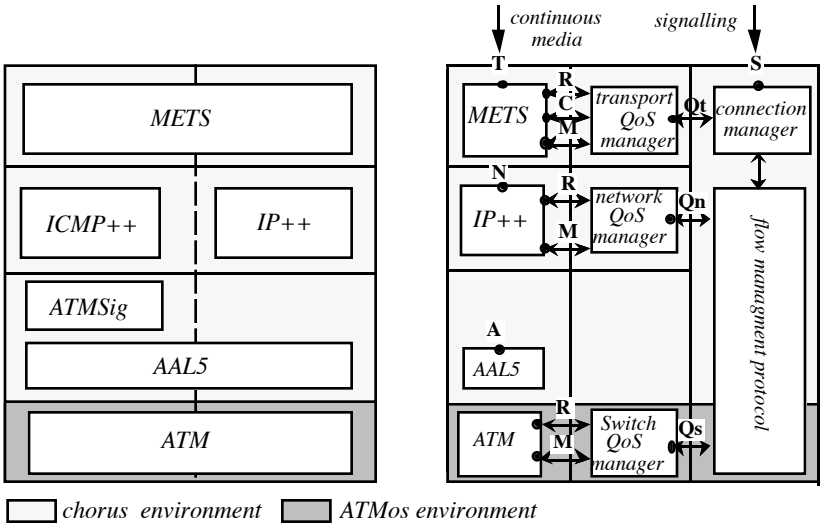

Figure 4a: Flow Protocol Projection **Figure 4b**: Flow Management Projection

The lowest layer of our architecture is based on the Lancaster Campus ATM network. This delivers ATM to a mix 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 100 Mbps Joint Academic Network. The PCs which run the system described in this paper 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 Figure 4a and 4b.

5.3 Realisation in Chorus

User and QoS Maintenance Planes. In implementation, we map the baseline architecture 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 in the connection manager actor (see below). The remain transport functions and facilities of METS and its related transport QoS manager are implemented in the same user level library that supports the flow API and delta service contract discussed below so that its service interface can be provided using the rtport and rthandler abstractions. The data transfer aspects of transport protocol (described as the data-tsap in section 4.1) 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 (described as the fm-tsap also described in section 4.1) communications with the *connection manager* 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 device driver, is implemented in the network actor. The major complexity involved in the IP++ implementation is in supporting the routing function. This is required when the current host is neither the source or the sink of a flow but is merely routing it from one network to another. In this case, CPU and memory resources are dedicated to flows on the basis of a flow-spec supplied by the FMP. Otherwise, the function of the IP++ layer is effectively null.

AAL5 is also implemented in the network actor. A software AAL5 implementation is required because our ATM interface cards only support data transfer at the granularity of ATM cells which is required for our cell based scheduling experimentation. The AAL5 implementation uses per-flow threads to perform segmentation and reassembly on both the send and receive side, and a per-flow selectable crc-32 option. This implementation choice reduces multiplexing in the stack to an absolute minimum as recommended in the literature [Tennenhouse,90]. Currently, the maximum service data unit sizes for the AAL5, IP++ and transport layers alike is restricted to 64 KBytes. This means that no further segmentation/ reassembly is required above the AAL5 layer[2]. The ATM cards generate an interrupt every time a cell is received, and every they are ready to transmit. Communication between the interrupt service routines and the per-flow AAL5 threads is via Chorus mini-ports which are specially optimised for this purpose.

Flow Management and Control Planes. Flow management in a QoS driven system is much more than the establishment of communication between remote machines. It includes all the dynamic aspects of the QoS semantic such as signalling, resource reservation, admission control, QoS mapping and QoS adaptation.

The communications resources in the end system are managed by three resource managers (viz. CPU, memory and network) which given a resource request (on the resource control interface in section 4.1) conduct admission tests and if successful allocate resources to the appropriate flow. It is the task of the connection management to arrange the allocation of suitable CPU, memory and network resources according to the specified QoS of the requested connection This includes partitioning the responsibility for QoS support among individual resource managers. For example, the connection manager partitions the flow spec delay QoS parameter between the network and the CPU resource managers on each end system.

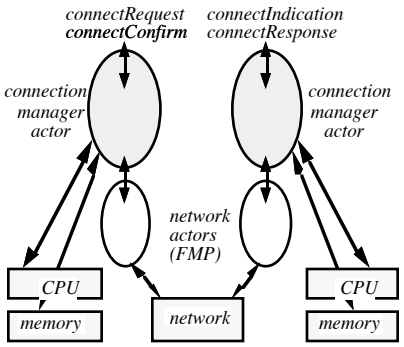

Figure 5: Flow Management

We realise the functionality of the flow managment and control planes under a connection management scheme which adopts a split level structure to the end-systems and communications resources. First, when a new flowConnectRequest for QoS controlled connection is requested, the QoS mapping function (see [Campbell,93a] for an example of QoS mapping, and [Robin,94] for the full details of the QoS translation algorithm) in a user level library determines the resource requirements of new connection requests. The output of the QoS mapper is then directed to a per-machine connection manager actor which performs admission testing and resource allocation for the end-system and network resources . The connection manager must communicate with each of the CPU, network and memory resource

[2] 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.

managers independently, and only if all are able to provide the required resource commitment can the connection request be granted.

The connection manager has to deal with both local and remote CPU and memory resources. The necessary signalling and communication functionality for the remote case is provided by the FMP described above. Figure 5 illustrates the relationship between the connection manager and the FMP (which is situated in the network actor). The figure shows two separate sites, each with its own connection manager.

5.4 Delta Service Contract

Prior reservation of resources to connections is necessary to obtain real-time performance. In this section we describe the resource reservation framework in our 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 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 3. This delta 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;
```

We omit the cost clause in the delta service contract as our ideas on tariffing have not matured enough to warrant its inclusion at the present time. Another omission is the connection_t and resource server: currently we only support the negotiated connection service; the area of fast connect, forward reservation and the realisation of the resource server in the local area are for further study. A new clause has been recently added to the delta service contract (the rationale for this addition is given later).

QoS commitment expresses a degree of certainty that the QoS levels requested will actually be honoured. Currently, our infrastructure only supports guaranteed and best effort commitment classes as shown below; realising a statistical class is also for further study by the QoS-A project. If commitment is guaranteed, resources are permanently dedicated and allocated at peak rate 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.

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

The flow spec described in section 3.1 is also revised by our experimental infrastructure - to take into account the currently supported commitment classes. Only peak rate flow spec parameters are required since guaranteed flows must allocate resource based on the worst case scenario. Therefore, frame rate which represents the statistical average rate is not included. Likewise, we drop the media type field which has not been implemented to date.

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

Delay refers to the maximum tolerable end-to-end delay, where the interpretation of 'end-to-end' is dependent on whether 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. Loss is used in conjunction with interval and refers to the maximum permissible number of buffer losses and corruptions over a given interval.

Frame size, peak rate and burst combine to describe the required rate at which buffers should be delivered at the sink of the connection. Jitter, measured in milliseconds, refers to the permissible tolerance in buffer delivery time from the periodic delivery time implied by peak rate. For example, a jitter of 10ms implies that buffers may be delivered up to 5 ms either side of the nominal buffer delivery time.

```
typedef enum {isochronous, workahead} delivery_t;
```

We have augmented the service contract with a *delivery* option described above. Delivery refines the meaning of throughput (frame rate, frame size and burst). If *isochronous* delivery is specified, the QoS-A delivers precisely at the rate specified by throughput otherwise, if delivery is *workahead*, it attempts to 'work ahead' (ignoring the jitter parameter) at rates temporarily faster than throughput. One use of the workahead delivery mode is to support applications such as real-time file transfer. Its primary use, however, is for pipelines [Coulson,93] of processing stages where isochronous delivery is not required until the last stage. At the moment the delivery parameter remains part of our delta service contract only since it is dependent upon the type of scheduling adopted. In our implementation we have selected the earliest deadline first (EDF) scheduling discipline for the end systems and are experimenting with EDF and weighted fair queueing [Keshav,91] in our ORL ATM swicthes. Promoting the delivery parameter to the generalised service contract is for further study in the QoS-A project.

6. Related Work

There is currently very little literature available on the integrated treatment of QoS across all architectural layers. One early contribution [Sluman,91] examined the requirements for QoS support in Open Systems standards and made preliminary proposals for QoS related enhancements to the existing OSI RM. Standards have an important role to play in promoting a unified view of QoS. In ISO, a new project on QoS has been initiated (ISO/IEC JTC/SC21, and in the UK IST21/-/1/5) which addresses QoS in a consistent way. This activity covers QoS very broadly and has investigated user requirements for QoS and architectural issues [ISO,92a]. The QoS-A project at Lancaster University has provided input on our QoS-A [Campbell,92b] [Hutchison,94] work into this activity.

The subject of integrated QoS has recently emerged as an important activity in another ISO project on Enhanced Communication Functions and Facilities (ECFF) for the lower layers of the OSI RM. As a member of the ESPRIT-funded OSI 95 project we participated in the initiation of the project on ECFF in the ISO. In addition, we were instrumental in introducing what we considered to be the key multimedia communication requirements [Hutchison,92], [Danthine,92], [Boerjan,92] into the ECFF guidelines document [ISO, 92b]. The context of Lancaster's work in standards is to feed the results of our research into SC6/WG4 and SC21/WG1 on the enhanced multimedia transport service and protocol [Campbell,93b] and QoS-A respectively. We feel strongly that the research community should play a more active role in influencing the shape of future communication standards.

In contrast to the integrated view of QoS, the subject of providing QoS guarantees in integrated service networks has been widely covered in the literature [Kurose,93] [Keshav,93] and is now progressing rapidly towards implementation in experimental gigabit network testbeds [Lazar,93]. In [Clark,92] a distinction is made between three different service commitments: *(i)* guaranteed service for real-time applications; *(ii)* predicted service, which utilises the measured performance of delays and is targeted towards continuous media applications; and *(iii)* best effort service, where no QoS guarantees are provided. A unified traffic scheduling mechanism is also discussed which is based on a combination of weighted fair queuing and static priority algorithms. In our QoS-A, commitment is supported both at the end-systems and in the network. The idea of QoS commitment introduced by Clark et al. is extended; that is, each performance parameter identified in the flow spec can be configured to meet a specific level of service commitment. More recently, and following on from their earlier work, Shenker, Clark and

Zhang have developed their ideas on service commitment and scheduling architecture. In [Shenker,93] a new scheduling service model is described in some detail. The service model is made up of two components: *(i)* a delay related component which supports two kinds of real-time service viz. guaranteed and predictive, and also multiple classes of ASAP elastic services which are synonymous with a best effort style of service; and *(ii)* a link-sharing component (based on similar work by Floyd [Floyd,93]) which addresses the need to allocate bandwidth between entities through sharing and regulating of the aggregate bandwidth of a link.

In [Lazar,90] an Asynchronous Time-Sharing network which provides a QoS capability is proposed. The switching provided by the network is novel in that the concept of QoS explicitly appears in the design specification at both the edge (at call level) and the core (at cell level) of the network. The network supports four well-defined traffic classes which support circuit emulation, voice and video, file transfer and network management flows. Following on from this work [Hyman,92] describes a joint scheduling and admission control mechanism used to guarantee the QoS of each traffic class.

The area of resource reservation is fundamental in providing end-to-end QoS guarantees. There have been a number of significant contributions to resource allocation in communication networks which have emerged over the past few years. In particular, ST-II [Topolcic,90] is a significant contribution, designed specifically for packetised audio and video communications across the Internet. In contrast to ST-II which provides source initiated point-to-multipoint flows, RSVP [Zhang,93] provides receiver initiated reservation and multipoint-to-multipoint support. SRP [Anderson,92] also designed for the Internet supports end-system and networks resource allocation. The QoS-A connection manager and flow management protocol is tailored for the local ATM environment and borrows heavily from ST-II and SRP. Our flow reservation service differs from the above ST-II, SRP and RSVP in that it supports a fast and forward reservation service.

In the area of QoS configurable transport systems, [Wolfinger,91] describes a protocol intended to run over a network layer offering comprehensive QoS guarantees. The protocol offers QoS configurability and includes an algorithm for bounding buffer allocation given throughput and jitter bounds. The design uses a shared memory interface between user and protocol threads. The HeiTS project [Hehmann,91] also investigated the integration of transport QoS and resource management (scheduling). HeiTS puts considerable emphasis on an optimised buffer pool which minimises copying and also allows efficient data transfer between local devices. The scheduling policy used is a rate monotonic scheme whereby the priority of the thread is proportional to the message rate accepted. The role of QoS monitoring, maintenance and commitment are not addressed by either of the above mentioned pieces of work.

[Danthine,93] reports on the development of an enhanced transport service in the OSI 95 project. Three transport level QoS semantics are proposed in addition to best effort service. Each performance parameter is specified by a structure of three types viz. compulsory, threshold and maximal QoS. When a compulsory value is selected the transport protocol commits to monitor the connection and will abort the service should the QoS drop below the requested value. The threshold QoS value, which is motivated by the needs of a multimedia service [Boerjan,93], commits the service provider to monitor the on-going performance of the connection. In this case however, a QoS indication informs the user should the QoS degrade below the requested value. The maximal QoS value deals with limiting the over utilisation of communications resources on a connection. It is possible to associate all three QoS values to the same performance parameter. In [Danthine,93] a number of negotiation rules are laid out for each of the QoS value. The threshold value is suitable for multimedia communications where applications may accommodate service fluctuations. The compulsory value however, is not a suitable semantic for the multimedia communications as many applications prefer degraded service to no service. The OSI 95 transport service provides a set of QoS features which are suitable for a wide range of transport service user's needs; however QoS maintenance and adaptation have not been addressed in any detail.

Several projects in the European funded RACE program are concerned with QoS for integrated broadband networks. A significant 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 methodical manner. The model takes into account the life cycle of service from both the user and network provider

viewpoints, and QOS performance mapping between the viewpoints based on the decomposition of end-to-end services into elements. The service life cycle covers conception, planning, provision and operation of multimedia services in an integrated broadband environment. In some related work Jung and Seret [Jung,93] 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.

Work carried out at the University of Pennsylvania [Nahrstedt,93] describes a *brokerage* model which incorporates QoS translation, and QoS negotiation and renegotiation. The notion of *eras* is introduced 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 *broker,* is responsible for the translation of QoS at the user–network interface.

In related end-system support, work on real-time extensions to the Mach micro-kernel, consisting of real-time threads, real-time synchronisation primitives and time driven scheduling, is described in [Tokuda,92]. The scheduling mechanism is derived from the ARTS kernel and permits hard real-time scheduling based on EDF. For end-to-end continuous media support, the main limitation of this work is the lack of API level QoS specification and integration with the communications sub-system. As an example of the latter, the API provides means to create periodically executable threads, but there is no way to associate this periodicity with the arrival of messages on a Mach port. More recent work by the same group has addressed QoS issues, including QoS monitoring through the concept of *deadline handlers* which are invoked when deadlines are missed [Tokuda,93].

Media scaling [Delgrossi,93] and codec translation [Schulzrinne,93] at the end systems, and filtering media traffic [Pasquale,92] [Zhang,93] in the network are very topical areas of QoS research at the moment. Media scaling[3] 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 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 constraint. Network level filtering looks very promising when used in conjunction with multicast protocols for dissemination of continuous media in support of heterogeneous receivers; for example, Pasquale et. el. [Pasquale,92] suggest that several receivers having disparate QoS communication requirements, and needing to access the same video flow simultaneous can be supported by a propagating filter scheme which deliver the appropriate QoS to each receiver. This scheme promotes efficient use of network resources, and as the literature suggests, reduces the likelihood of the on set of congestion.

7. Conclusion and Future Work

In this paper we have described in detail our QoS architecture 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. We also introduced the idea of flow management [Campbell,94] which provides for the monitoring and maintenance of the contracted QoS. These QoS concepts emerged from work carried out on the OSI95 [Danthine,92] [Boerjan,92] [Campbell,93] project and are motivated by the widely accepted communication needs of distributed multimedia applications.

The proposed QoS-A promotes the idea of *integrated* QoS, spanning the end-systems and the network, and takes the support of QoS for a wide range of applications as its primary goal. Many researchers to date have concentrated on either the network or the end-system in isolation. In contrast,

---

[3] The QoS-A supports media scaling as part of dynamic QoS adaptation management.

QoS concepts are coherently applied across all architectural layers, resulting in a complete framework for the specification and implementation of the multimedia flows in the local ATM environment.

At the present time we are experimenting with an infrastructure consisting of three 486 PC's running Chorus and connected to an ORL ATM switch via ISA bus ORL ATM interface cards. Our intention is to connect the ORL QoS testbed to our campus ATM network which comprises Netcomm and Fore switches. The PCs also contain VideoLogic audio/ video/ JPEG compression boards as real-time media sources/ sinks. The current state of the implementation is that the API, connection manager and flow management protocol, scheduling infrastructure, transport protocol and ATM card drivers are in place.

The next phase of our implementation will include network support. This work will draw heavily from the recent literature on providing QoS guarantees in fast packet switched networks. In particular we plan to experiment with suitable switch scheduling disciplines [Parekh,92] [Keshav,91] [Hyman,93], resource management [Shenker,93] [Floyd,93] and filtering strategies (see below) for the QoS-A, given the types of service commitment and flow management we are advocating at the transport service interface.

At Lancaster we are developing specialised hardware support called *filter servers* [Yeadon,93] which are based on MEND technology [Lunn,94] and are located close to end-systems. These servers can implement *intelligent filters* such as MpegToJpeg codec translators, and are the functional equivalent of Schulzrinne's filter bridges [Schulzrinne,93]. In addition to filter servers we are also implementing simple *cell level filters* for our ORL switches by augmenting the ATMSig[4]. The outcome of this phase of the research will be the subject of a forthcoming paper.

8. Acknowledgement

The authors are very grateful to the reviewers for their thoughtful comments and suggestions. In addition, we would like to thanks the members of the Multimedia Projects Group at Lancaster and the ISO QoS Working Group for the many enlightening discussions on end-to-end QoS. In particular, we are grateful to Francisco García, Frank Ball, Nick Yeadon and Philippe Robin for providing insight into network and end-system support for continuous media communications, and John Holmes and Chris Sluman for their support of our work in the ISO, and finally to Ian Wilson of Olivetti Research Limited.

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

9. References

[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.

[ATM Forum,93] ATM Forum, "ATM User-Network Interface Specification Version 2.4", August 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.

[Campbell,92a] Campbell, A., Coulson G., García F., and D. Hutchison, "A Continuous Media Transport and Orchestration Service", *Proc. ACM SIGCOMM '92,* Baltimore, Maryland, USA, August 1992.

[Campbell,92b] Campbell, A., Coulson G. and D. Hutchison, "A Suggested QOS Architecture for Multimedia Communications", ISO/IEC JTC1/SC21/WG1 N1201, International Standards Organisation, UK, November, 1992, and *Internal Report* No. MPG-92-37 Department of Computing, Lancaster University, Lancaster LA1 4YR.

[Campbell,93a] Campbell, A., Coulson, G., García, F., Hutchison, D., and H. Leopold, "Integrated Quality of Service for Multimedia Communications", *Proc. IEEE INFOCOM'93*, pp. 732-739, San Francisco, USA, April 1993.

---

[4] The QoS model presented in this paper does not reflect our work on filtering at the present time. Realising filtering services in a generalised QoS-A is for further study

[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,94] Campbell, A., Coulson, G., and D. Hutchison, "Flow Management", to be presented: *5th IFIP Conference on High Performance Networking*, Grenoble, France, June 1994, and *Internal Report* No. MPG-94-09, Department of Computing, Lancaster University, Lancaster LA1 4YR.

[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,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.

[Coulson,93] Coulson, G., and G. Blair, "Micro-kernel Support for Continuous Media in Distributed Systems", Internal Report No. MPG-93-04 Department of Computing, Lancaster University, Lancaster LA1 4YR and to appear in *Computer Networks and ISDN Systems*, 1993.

[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.

[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., Hehmann, D.B., Herrtwich R.G., Krone, J., Sandvoss, C., and C. Vogt, "Media Scaling for Audiovisual Communication with the Heidelberg Transport System", *Proc. ACM Multimedia '93*, Anaheim, August 1993.

[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.

[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.

[García,93] García, F., "A Continuous Media Transport and Orchestration Service", PhD Thesis, Department of Computing, Lancaster University, Lancaster LA1 4YR, UK, June 1993.

[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" *Second International Workshop on Network and Operating System Support for Digital Audio and Video,* IBM ENC, Heidelberg, Germany, 1991.

[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,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.

[ISO,92a] ISO, "Quality of Service Framework - Outline", ISO/IEC JTC1/SC21/WG1 N1145, International Standards Organisation, UK, March 1992.

[ISO,92b] 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.

[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,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.

[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,93] Lazar, A. A., "Control and Management of Enterprises COMET Research Group", Activity Report: 1991-1993, Center for Telecommunications Research, Columbia University, USA, 1993.

[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.

[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.

[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.

[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.

[Robin,94] Robin, P., Campbell, A., Coulson, G., Blair, G., and M. Papathomas, "Implementing a QoS Controlled ATM Based Communication System in Chorus", submitted to: *4th IFIP International Workshop on Protocols for High Speed Networks*, and Internal Report No. MPG-94-05 Department of Computing, Lancaster University, Lancaster LA1 4YR, March 1994.

[Schulzrinne,93] Schulzrinne, H. and S. Casner, "RTP: A Transport Protocol for Real-Time Applications", Work in Progress, Internet Draft, <draft-ietf-avt-rtp-04.ps>, October 1993.

[Shenker,93] Shenker, S., Clark, D., and L. Zhang, "A Scheduling Service Model and a Scheduling Architecture for an Integrated Service Packet Network" Draft available via anonymous ftp from parcftp.xerox.com:/transient/service-model.ps.Z, September 1993.

[Sluman,91] Sluman, C., "Quality of Service in Distributed Systems", BSI/IST21/-/1/5:33, British Standards Institution, UK, October 1991.

[Tennenhouse,90] Tennenhouse, D.L., "Layered Multiplexing Considered Harmful", *Protocols for High-Speed Networks*, Elsevier Science Publishers (North-Holland), 1990.

[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.

[Topolcic,90] Topolcic, C., "Experimental Internet Stream Protocol, Version 2 (ST-II)", *Internet Request for Comments No. 1190* RFC-1190, October 1990.

[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.

[Zhang,93] Zhang, L., Deering, S., Estin, D, Shenker S. and D. Zappala, "A New Resource ReSerVation Protocol" Draft available via anonymous ftp from parcftp.xerox.com:/transient/ rsvp.ps.Z, August 1993.