# A MULTIMEDIA ENHANCED TRANSPORT SERVICE IN 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)* and transport layer service contract and show how QoS levels contracted at the transport service interface can be assured in the context of a local ATM network.

## 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 reduce 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,93a] and service disciplines in switch queues are needed [Zhang,93b] [Parekh,92]. 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 aspects of the QoS-A, primarily focusing on the transport layer. Section 2 provides an overview the QoS-A and introduces the central notions of *flow* and *flow management*. Section 3 then describes a multimedia enhanced transport service 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 server. Finally, section 5 compares our design to related work in the field and section 6 presents our conclusion and future work.

## 2. Quality of Service Architecture

### 2.1 QoS-A Model

The QoS-A [Campbell,93] is a layered architecture of services and mechanisms for QoS management and control of continuous media flows in an ATM based network. In the rest of this paper we assume a local ATM environment as this is the platform on which we are currently evaluating the architecture though practical experimentation.

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* provided by an ODP compatible distributed systems platform embedded in a Chorus microkernel 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.

The communications infrastructure is provided by a local ATM network. On top of the ATM layer we have a signalling ATM adaptation layer (SAAL) for the exchange of control information and an ATM adaptation layer services for data transfer. The SAAL is a combination of two sub-layers: a common part and a service specific part. The common part convergence sub-layer (CPCS) provides a non-assured service over a segmentation and reassembly sub-layer (in our case, this is SAR-5 which is also used in the user plane for continuous media data). The service specific part provides an assured point-to-multipoint service and includes: *(i)* a service specific co-ordination function (SSCF) which provides a user-to-network (UNI) interface for point-to-point and point-to-multipoint signalling; and *(ii)* a service specific connection oriented protocol and service (SSCOP) which provides for the establishment and release of SAAL connections (FM-SSVC and CM-SSVC) for assured data transfer. Note, that SSCF UNI will initially support the QoS-A transport signalling protocol but in due course may provide the ATM Forum's UNI (Q93BF); this is for further study by the QoS-A project. For continuous media data transfer, the service specific region of the user plane is null as no assured mode of operation is required for the transfer of continuous media [Campbell,92a].

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

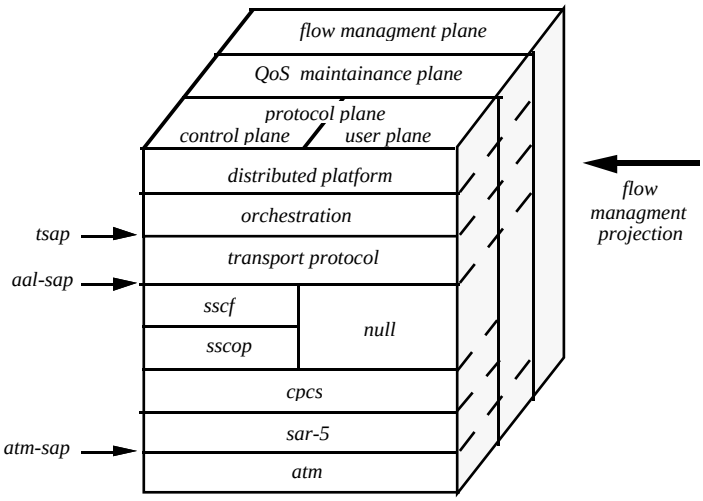


Figure 1: QoS-A

### 2.2 Flow Management Domains

For the purposes of efficient flow management and QoS control, we logically partition the ATM network into *flow management domains* [Crosby,93] which constitute arbitrary collections of network devices (network devices can be ATM switches, multimedia workstations and continuous media storage servers) can correspond to administrative domains whereby the network is partitioned according to ownership, organisational boundaries, etc. In each domain, the flow management plane together with supporting control plane functions are realised as a single network server which we call a *flow management server (FMS)*. Thus, network devices do not in general support the flow management plane: they only implement the data, control and QoS maintenance planes of the QoS-A.

A number of architectural choices were possible for FMS 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 FMS per domain) and thus reduces the overhead of maintaining database integrity while simultaneously avoiding bottlenecks introduced by an excessively centralised solution.

## 3. METS: 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     flow_connect_request( 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, error, 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 of the QoS-A;
- connection_t selects from negotiated, fast reservation, and 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 error 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;           /* error 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 [Campell,93].

### 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     error;
    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 ultilisation [Campbell,93]. 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 (how we intend to achieve this is for further study).

### 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 adapt 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, NULL_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    */
} 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, NULL_ACTION, 0},
                       {THROUGHPUT, INDICATION, RENEGOTIATE, &new_flow_spec},
                       {LOSS, NULL_ACTION, NULL_ACTION, 0},
                       {JITTER, INDICATION, NULL_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 new_flow_spec, 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 the service contract to accommodate the three connection styles described above:-

```
typedef enum {FAST, NEGOTIATED, FORWARD} service_t;
typedef struct {
    connection_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.

## 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 3. Note that Figure 3 only shows the transport layer and below; the upper layers have been omitted for clarity.

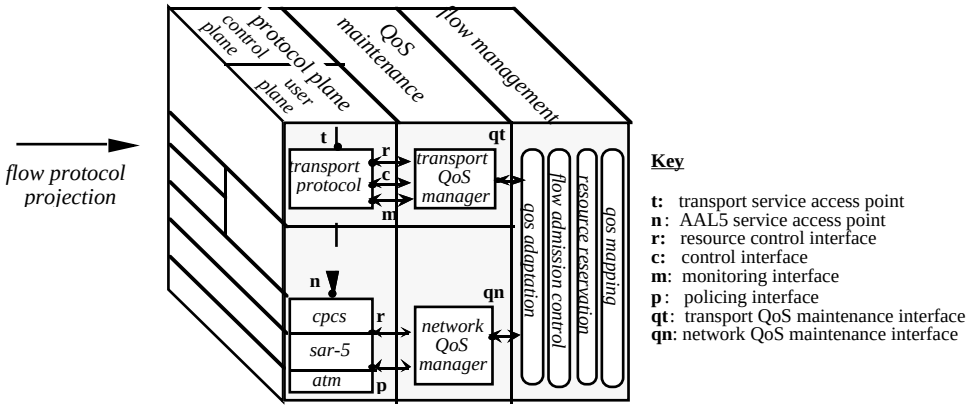


Figure 3: Flow Management Projection

### 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 FMS 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-TSAP) and data (DATA-TSAP) components (there are also equivalent primitives at the network layer):-

- the *FM-TSAP* interface contains primitives: get_tsap, free_tsap, flow_connect, qos_renegotiation, qos_degradation, qos_report, monitor_flow, flow_assessment, maintain_flow and flow_disconnect.
- the *DATA-TSAP* interface contains primitives: data_request, dataa_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 :-

- 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 manger 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 FMS. It contains the following primitives: start_maintenance, set_attributes, free_attributes, assessment, qos_alert and stop_maintenance;

### 4.2 User Plane: The Transport Protocol

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. Deterministic flows each receive a fixed buffer allocation based on the flows peak rate whereas statistical flows share a common pool. 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 2).

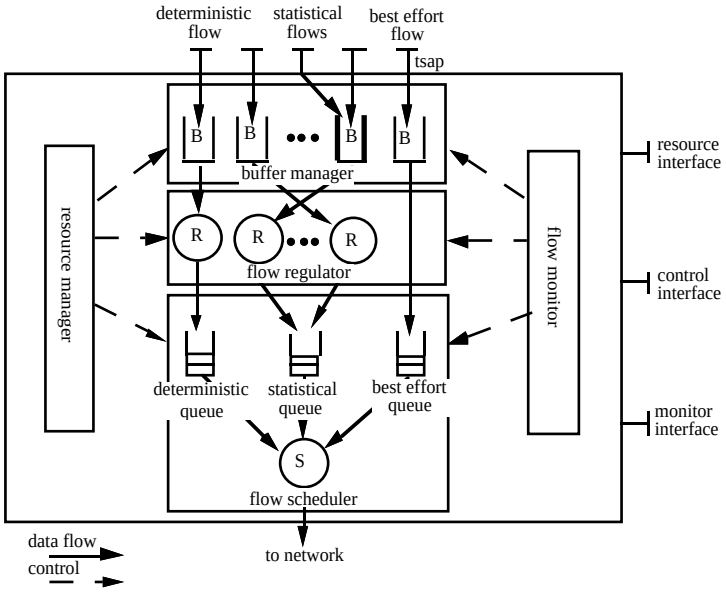


Figure 2 : Transport Protocol Mechanisms and Interfaces

**Flow Regulator**. The transmission of frames to the network must be regulated to prevent buffer overflow at receivers and rate violations at the UNI 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 2 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]. 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-preemptive 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.

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.

The information passed periodically from the receiver is carried in a control message which is already a part of the 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_error;   /* frame error 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 monitoring mechanism is able to build up a statistical representation of the end-to-end QoS using the performance data supplied in the control message. 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 error rate measured during the specified interval.

**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.

| | | *commitment type* | | |
|---|---|---|---|---|
| *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: Resource Reservation versus QoS Commitment

Deterministic flows achieve guaranteed QoS by reserving dedicated communications buffers (based on peak rate allocation) and using deadline scheduling. 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. 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

### 4.3 QoS Maintenance Plane: The Transport QoS Manager

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 FMS, 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 FMS 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 FMS).

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 Server

The flow management plane, realised as a per-domain flow management server as described in section 2.2, is responsible for a number of off-line management functions. The major functions, to be described below, consist of the provision of a network signalling infrastructure, the implementation of a resource reservation protocol, and the realisation of course grained dynamic QoS management as specified in the flow spec's adaptation clause. The FMS 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,93].

**Signalling Infrastructure**. To the transport service user, flows are accessed via their dedicated *data* transport service access point (DATA-TSAP) together with a single *flow management* TSAP (FM-TSAP) common to all flows on that node. The latter is used for flow initiation requests, re-negotiation requests, etc. At the AAL layer each DATA-TSAP maps to separate data and control AAL5-SAPs where the control SAP is used for purposes such as the flow control signalling and the transmission of control messages. The per-node flow management TSAP maps to a single, per node, SAAL service access point which serves as a user-to-network interface (effectively, a user-to-FMS interface).

To realise each flow, the QoS-A uses a single *data* switched virtual circuit (SVC) and a single *control* SVC. For the signalling functions it uses a per-node *meta-signalling* SVC and a *flow management* signalling switched virtual circuit (SSVC). All these circuit types are non-multiplexed as multiplexing above the ATM layer, as this is considered unsuitable for multimedia communications [Campbell,92a]. When a device is first attached to the local ATM network its flow management SSVC is established using the meta-signalling SVC. Subsequently, all user requests issued at the FM-TSAP (e.g. connect requests) are carried on this SSVC.

Before a flow can be established at the transport layer, a user must request the allocation of a valid TSAP address to be used in the flow_connect_request. This is done by issuing a get_tsap primitive on the flow management TSAP which maps to a command sent over the flow management SSVC channel to the FMS.

**Flow Reservation Protocol.** 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. The reservation protocol 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.

The FMS, as the central resource controller for its flow management domain, is the arbiter of all communication resource allocation requests. For the *negotiated* service, when a flow_connect_request is issued by a transport user, the FMS consults its local representation of domain wide resource availability. If the request can be satisfied, the FMS 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 source and sink nodes) to request that they allocate the resources requested. The FMS 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 FMS 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 FMS'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.

When a flow_connect_request spans multiple domains inter-FMS signalling is used. Each intermediate FMS is responsible for the allocation of local resources along the flow setup path through its domain.

**QoS Adaptation**. In its QoS adaptation role, the FMS is responsible for initiating the coarse grained QoS adjustments specified in the service contract's adaptation clause. The behaviour of the FMS is also determined by the maintenance clause. If the maintenance mode is *no maintenance* the FMS 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 FMS simply receives periodic flow_assessments from the QoS manager and passes them on to the application. If the commitment is *maintain*, the FMS does not receive any flow_assessments as the responsibility for flow management has been

delegated to the QoS manager. However, the FMS 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 FMS 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 FMS 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 flow_request 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. 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] 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, 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. Several different ways of categorising QoS guarantees have been identified. 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.

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,93a] 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 flow reservation 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, commitment and adaptation have not been addressed in any detail.

## 6. 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 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 which provides for the monitoring and maintenance of the contracted QoS. These QoS concepts emerged from work carried out on the OSI95 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, 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 have established an experimental infrastructure based on two 80486 machines running a multimedia enhanced Chorus micro-kernel. The communication support for the PCs consists of ATM interface cards connected to a Netcomm ATM switch. The PCs are also equipped with VideoLogic audio/ video/JPEG compression boards. We are currently implementing the resource reservation protocol, and the transport layer QoS manager, flow regulation and scheduling aspects of the QoS-A.

The area of network support for flows remains an important aspect of the QoS-A which we have not yet addressed. This future work will draw heavily from the recent literature on providing QoS guarantees in packet switched networks. In particular we plan to investigate suitable switch scheduling disciplines [Parekh,92], [Zhang,93b] and resource management strategies [Shenker,93], [Floyd,93] for the QoS-A, given the types of service commitment we are advocating at the transport service interface.

## 7. Acknowledgement

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. The authors would like to thank Francisco García, as this paper builds on his earlier work in the area of communication services for continuous media applications.

## 8. 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.

**[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, De, 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", *Presented at 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,93]** 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.

**[Cidon,93]** 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.

**[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.

**[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.

**[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., Campbell, A. and H. Leopold, "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.

**[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.

**[Parekh,92]** Parekh, A., "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.

**[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.

**[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.

**[Zhang,93a]** 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.

**[Zhang,93b]** Zhang, L. and D. Ferrari "Rate-Controlled Static-Priority Queueing"*Proc. IEEE INFOCOM'93*, pp. 227-237, San Francisco, USA, April 1993.