A DYNAMIC INTERACTIVE PROTOCOL FOR DISTRIBUTED MULTIMEDIA OVER ATM NETWORKS

Gheorghita Ghinea

A research report submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science

Degree awarded with distinction on 4 December 1996.

Johannesburg, 1996

DECLARATION

I declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

Gheorghita Ghinea

The 26th day of August 1996

ABSTRACT

This report describes a dynamic Quality of Service (QoS) - based call admission protocol for distributed multimedia over ATM networks. The protocol incorporates the innovative idea of an *extended QoS*. This is a composite term which takes account of not only classic QoS multimedia measures, but also of the human aspect of the interaction through *human receptivity*. For this scenario, different QoS negotiation strategies have been formulated and then simulated with a view towards the establishment of a *protocol knowledge base*. Separately, a session pricing policy has been elaborated and its effect on user behaviour and network resource allocation studied.

To my father,

Benone Ghinea

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Valentin Kisimov for the help, advice and guidance that he generously provided. Thank you for always being there when I needed you. Thanks also goes to the members of the Departmental Committee who reviewed my research as well as to Professor Hanoch Neishlos, all of whom provided valuable input and hindsight. I would not have been able to complete this undertaking without the support and understanding of my family - I am deeply grateful. Lastly, thanks goes to any person who has helped me in any way whatsoever towards the completion of this project and whose name I have omitted to mention.

CONTENTS Page

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The recent advent of fibre optic technology has provided a tremendous increase in the bandwidth available to applications as well as a drastic reduction in transmission error rates. This increased bandwidth - to the tune of several gigabits per second - has offered communication networks the potential to handle, in an integrated fashion, a heterogeneous mix of network traffic including voice, data, and conference video in one transmission and switching fabric technology. This would yield a single telecommunications infrastructure offering a multitude of advantages, including vast economies of scale, ubiquity of access, and improved statistical multiplexing. The *Asynchronous Transfer Mode* (ATM) is a high-speed networking technique being developed for both the public [Mce92] and local area networks (LANs) [Kun92], capable of supporting this vision.

ATM networks are characterised by their switch-based architecture. All computers are connected to a switch, and the communication between each pair of computers is established through the switch. This is in contrast to the situation where all computers share one communication medium, as in the case of traditional LANs such as Ethernet. A switched network is capable of supporting multiple connections and, within a predesigned limit, the available bandwidth of ATM switches increases as the number of ports increases.

Multimedia, on the other hand, is an interdisciplinary, application-oriented technology that capitalises on the multisensory nature of humans and the ability of computers to store, manipulate and convey non-numerical information such as video, graphics, and audio in addition to numerical and textual information. Multimedia has the intrinsic goal of improving the bandwidth and effectiveness of computer-to-human interaction and, ultimately, of human-to-human communication.

Distributed network computing offers great potential for increasing the amount of computing power and communication resources available to applications, and part of the success of multimedia will depend on the ability to support networked applications. Initial multimedia applications such as training, education, marketing and business presentations were desktop applications. However, such stand-alone systems were soon found to be wanting, especially as regards real-time access to and update of data. On the other hand, desktop video conferencing, access to remote libraries of multimedia material, access to archived multimedia records, messaging, and downloading server-based multimedia instruction [Cat92] are just a few example of desirable networked multimedia applications. The economic advantages in a transition to networked multimedia are clear, particularly in the context of mission-critical functions and productivity gains.

The connectivity associated with distributed multimedia requires readily available and affordable high-capacity networks. ATM is just such a technology that has the ability to satisfy the communication and bandwidth needs required by distributed multimedia applications. What further makes ATM an attractive prospect for distributed multimedia is the fact that local ATM networks are appearing in advance of long-haul ATM networks, which both makes them an appealing alternative to traditional LANs and enables major synergies between the local area and public networks because both rely on the same transport technology [MinKei94].

1.1. Problem Statement and Motivation

The focus of our research has been the development of a cost-effective, dynamic call admission protocol which provides QoS management - including *human aspects of QoS* - in ATM switched networks running distributed multimedia. The emphasis has been on the design of dynamic QoS management strategies for the protocol and the overall evaluation of the protocol's functionality through simulation. Separately, based on the simulation results, a *protocol knowledge base* has been built which provides QoS management recommendations during protocol execution.

There are several reasons which motivate this research : firstly, ATM is the envisioned transport mode of the near future in the telecommunications industry and, consequently, there is considerable research interest and scope in the field [Tur86], [AhmDen89], [Cat92], [New92], [Par94], [SiuJai95]; secondly, the majority of multimedia systems currently in operation are not stand-alone, but are distributed according to the client-server paradigm - this leads to a variety of traffic with massive bandwidth requirements which can, however, be efficiently supported by ATM-based networks; thirdly, provisioning of QoS guarantees in ATM networks, although extensively studied remains an active interest area, especially since very few of the strategies proposed thus far in the literature can lay claim to having provided simple solutions to this problem [Gol90], [DemKesShe90], [FerVer90], [LazPacWhi90] , [GueAhmNag91], [ZhaKes91], [ClaSheZha92], [ParGal92], [Kur93], [LowVar93]; linked to this point is the fact that the human element is an important part of the overall QoS picture which, is often neglected in QoS negotiation protocols - as such, attention given to the human factor might result in more economical resource allocation strategies being used; lastly, an investigation into the effect of pricing policies in multimedia resource allocation protocols is also warranted - this is because, although pricing policies exist for non real-time networks (datagram networks), as well as for conventional public utilities networks (such as the telephone network) offering a single QoS [Co*etal*91], ATM multimedia networks offering real-time capabilities with multiple qualities of service are expensive and complex enough to justify a new look at accounting in networks and the role that pricing plays in congestion control.

1.2. Hypothesis

Two main facets comprise the hypothesis of the research work undertaken. The first is that the trade-off between the multimedia quality of service, as perceived by the human subject, and the amount of network resources needed to deliver the respective quality of service, leads to the possibility of accommodating a greater number of multimedia sessions in the network at the expense of a much smaller decrease, in

proportional terms, of the human aspect of the quality of service. The second facet of the hypothesis is that usage of a multimedia session pricing policy in conjunction with the designed call admission protocol results in more efficient resource allocation and a consequent increase in the average quality of service experienced by those multimedia sessions which can afford to pay for the required network resources.

1.3. Assumptions

A few main assumptions lie at the basis of the design of the call admission protocol. The first, and most important, is that the main measure of a network's resources is given by the bandwidth available on its links. This is a logical assumption to make bearing in mind the bandwidth-intensive nature of current multimedia applications and the fact that the bandwidth available on communication links is the single most important component in the delivery of a requested quality of service.

Secondly, although the protocol accommodates both variable and constant bit rate applications, it has also been presumed that multimedia sessions will reserve bandwidth according to their peak requirements for a particular level of human receptivity. Stemming from this premise, it has further been assumed that the whole of a network link's bandwidth is available to multimedia sessions. This is in contrast to some traffic management schemes where a portion of the bandwidth is reserved for peaks in network traffic.

Lastly, it has been assumed that those multimedia sessions that cannot be accommodated by the available network resources are rejected outright and not, for instance, queued for re-admission. This assumption is wholly justifiable by observing that it reduces protocol overhead, as well as by the fact that, from the point of view of session call acceptance - the main focus of our research - a blocked session that is finally admitted when sufficient network resources are released can always be viewed as being an instance of a new session with the same characteristics as the blocked one being admitted.

1.4. Objectives of the Investigation

The objectives of the research work undertaken are closely linked to the hypothesis under scrutiny. Accordingly, one of the two main objectives of our research was to examine the impact on call admission and QoS delivery, that sacrificing the human aspect of the quality of service in exchange for bandwidth has Related to this goal, a further two objectives can be identified. The first is to investigate what the effect of a QoS negotiation strategy's complexity on its ability to deliver a consistently high average QoS is. The other is to explore the effect that a negotiated and dynamically varying QoS has on link utilisation in an ATM switched environment.

The second main objective of our endeavour was to examine the consequences of introducing a pricing policy in the overall framework of QoS negotiation. These consequences were analysed with respect to user behaviour and resource allocation - call admission, implicitly - as well as level of delivered QoS.

Lastly, a protocol knowledge base was built using results obtained exploring the above two main goals. This is a rule-based system and comprises recommendations regarding QoS negotiation and pricing strategies to be used according to the current network configuration (network topology and traffic load) as well as multimedia sessions' priority and desired response time.

1.5. Methodology

Multimedia sessions were divided into those requiring a best-effort type of service and those requiring a guaranteed QoS. Only best-effort multimedia sessions' QoS was subject to negotiation. A total of seven different dynamic QoS negotiation strategies were designed within the context of the protocol. All of these strategies were based on

discretely varying the QoS of multimedia sessions according to a predefined multi-stage negotiation scenario. Within each stage of the negotiation, best-effort sessions reduced their QoS according to their priority and time spent in the system.

A three-pron approach can be pursued in order to examine protocol performance: analytical, simulation, and actual hardware testing. Due to a lack of hardware facilities, however, the choice of protocol performance evaluation had to be made between the former two methods.

Analytical techniques are highly desirable to evaluate performance with reasonable computational requirements and accuracy. However, they do have limitations, such as the inability to represent general input traffic flows or heterogeneous inputs, and an analytical approach should be used only if it leads to approximations based on realistic assumptions. Moreover, the application of exact models is often too computationally complex to provide QoS controls in real-time, an essential requirement for many multimedia applications.

Simulation methods remain the most flexible for examining node operation and performance under a variety of conditions, and they will constitute our main approach. Through simulation, it will be possible to make general observations about the effectiveness of the designed protocol in delivering QoS guarantees under conditions of ATM multiplexing. Furthermore, the fact that simulation can be achieved within a reasonable time frame and without special computational requirements will facilitate a thorough examination of the effect that variations and adjustments in protocol parameters has on overall protocol performance. The results obtained by simulating the different QoS negotiation strategies that have been formulated will then be examined on a comparative basis, to see what is the best overall tactic that the protocol should use. The *COMNET III* network simulation package will be used to examine protocol performance in an ATM networking environment, while simulation coding of the different QoS negotiation strategies will be done in *C*.

The structure of this document is as follows. In Chapter 2 an overview is given on ATM networking issues. The actual protocol design is detailed in Chapter 3, while the protocol simulation is described in Chapter 4. Chapter 5 contains the presentation and analysis of the simulation results obtained. Finally, in Chapter 6, conclusions are drawn and possible extensions to our work identified.

2. ATM - Concepts, Switches, QoS Issues and Pricing

ATM is based on packet switching and uses small, fixed-length packets called cells, into which the original message to be transmitted is broken. Such fixed-length cells simplify the design of an ATM switch at the high switching speeds involved, ensuring that switching and multiplexing is carried out quickly and easily. Additionally, they provide the full advantage of the dynamic and fair allocation of bandwidth, even when messages vary considerably in length. Packet-switching itself reduces the delay and jitter (variance of the delay) of real-time applications such as voice and video. Furthermore, packet-switching using short, fixed-length cells tends to require smaller storage capacities at the intermediate switches, because short packets are less likely to be rejected at a node than full-sized messages. Because the ATM standards bodies chose to prohibit cell reordering in ATM networks, low-cost attachments using simple forms of buffering can be used. Lastly, cells of limited size are less prone to transmission errors than messages; however, if an error does occur in a packet, only that packet will have to be retransmitted [Hal92].

ATM is a connection-oriented technology, all cells in such a network belonging to a pre-established virtual connection (circuit). A virtual circuit allows the capacity of each link to be statistically multiplexed (i.e. shared on a demand basis) among connections, rather than give each connection a fixed allocation. The sequence integrity of all cells on a virtual connection is preserved across each ATM switch to simplify reconstruction of the original traffic at the destination. ATM cells are 53 bytes long, comprising a 5 byte header and a 48 byte data (payload) field. Each cell's header contains a *virtual channel identifier* (VCI) to identify the virtual connection to which the cell belongs. These VCIs have only

local significance, to avoid problems associated with a unique global allocation. As each cell traverses a switch, a table is consulted and the VCI used to identify the channel on the last hop is changed to the value identifying the channel on the next hop. The connections also enable the network to guarantee the QoS, by allowing a user to declare key service requirements and traffic parameters at connection set-up time, with the added possibility of dynamic control over these parameters.

2.1. ATM Protocol Reference Model

In modern communication systems, a layered approach is used for the organisation of all communication functions. The functions of the layers and the relations of the layers with respect to each other are described in a *protocol reference model* (PRM). The ATM protocol architecture is based on the Broadband Integrated Services Digital Network (B-ISDN) PRM, which consists of three planes : the *user plane*, the *control plane*, and the *management plane* [Itu91].

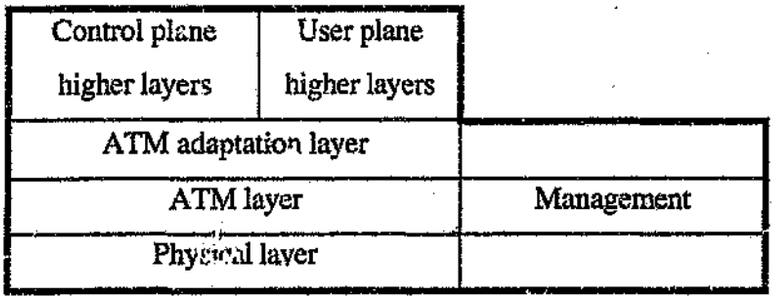

Figure 1 ATM protocol architecture

The management plane incorporates two types of functions namely the *layer management* functions and *plane management* functions. All the management functions that relate to the whole system are taken care of by plane management which is responsible for providing co-ordination between planes. Although no layered structure is employed within plane management, such a structure is, however, used within layer management. Layer management performs the management functions relating to resources and parameters residing in its protocol entities. It also handles the operation and management of information flows for each layer.

The user plane provides for the transfer of user information. All associated mechanisms like flow control and recovery from errors are included. A layered approach is used within this plane.

A layered structure is also used within the control plane. This plane is responsible for the call control and connection control functions. These are signalling functions which are necessary to set up, supervise and release a call or connection.

The functions of the *physical layer* (PL) and the *ATM layer* are the same for the control plane and the user plane. Different functions may however occur in the *ATM adaptation layer* (AAL) as well as in higher layers.

The physical layer is subdivided into the *physical medium* (PM) sublayer and the *transmission convergence* (TC) sublayer. The physical medium sublayer is the lowest sublayer and includes only the physical medium dependent functions. It provides the bit transmission capability, including bit alignment. Line coding and, if necessary, electrical/optical conversion are also performed by this sublayer. In many cases, the physical medium will be an optical fibre, although other media, such as coaxial and twisted pair cables, are possible. The transmission functions are medium specific. Over long distances, such as within the telephone providers' backbones, it is envisioned that the cells will be encapsulated inside SONET (*Synchronous Optical Network*) frames at 155 and 622 Megabits per second (Mbps).

The transmission convergence sublayer performs five functions. The lowest function is the *generation and recovery of the transmission frame*. *Transmission frame adaptation*, on the other hand, is responsible for all actions which are necessary to adapt the cell flow according to the used payload structure of the transmission system in the sending direction. These first two functions are specific to the transmission frame; the rest of the functions are common to all transmission frames. *Cell delineation* is the mechanism

that enables the receiver to recover the cell boundaries. *Header error control* (HEC) *sequence generation* is done in the transmit direction, with the HEC sequence being inserted in its appropriate field within the header. In the sending direction, *cell rate decoupling* inserts idle cells in order to adapt the rate of ATM cells to the payload capacity of the transmission system. Conversely, in the receiving direction, this mechanism suppresses all idle cells.

The ATM layer is the layer above the physical layer and operates on the ATM cell header. Its characteristic features are independent of the physical medium and it performs the following main functions : *cell multiplexing* and *demultiplexing*, *cell header construction* and *verification*, as well as *cell routing*. Cell routing is based on a two-level addressing structure given by the *virtual path* and virtual channel identifiers (VPI and VCI). The VPI identifies the physical path, which is then used by a set of VCIs. The *generic flow control* (GFC) field of a cell's header is used to implement crude congestion control at the *user network interface* (UNI). The header itself is protected by one byte of *cyclic redundancy checksum* (CRC) information.

The AAL lies between the ATM layer and higher layers. Its basic function is the enhanced adaptation of services provided by the ATM layer to the requirements of the higher layer. Higher layer *protocol data units* (PDUs) are mapped into the information field of an ATM cell. AAL entities exchange information with their peer AAL entities to support *AAL functions*. AAL functions are organised into the *segmentation and reassembly* (SAR) sublayer and the *convergence sublayer* (CS) functions. The main functions of the SAR sublayer are, at the transmitting side, segmentation of higher layer PDUs into a suitable size for the information field of the ATM cell and, at the receiving side, reassembly of the particular information fields into higher layer PDUs. The CS is further composed of a *common part* (CPCS) and a *service specific part* (SSCS). The CPCS provides services such as padding, CRC verification as well as flow management to and from the SAR sublayer. It takes a SSCS PDU, adds padding if needed, and then adds an eight byte trailer such that the total length of the resultant PDU is a multiple of 48. The trailer consists of two reserved bytes, two bytes of packet length, and four bytes of CRC

information. The SSCS is service dependent and is responsible for such services as confirmed data transmission.

To minimise the number of AAL protocols, the International Consultative Committee for Telephony and Telegraphy (CCITT) has proposed a service classification which is specific to the AAL [SiuJai95] :

- Class 1 : *constant bit-rate applications*, such as pulse code modulation telephony and some present video systems;
- Class 2 : *variable bit-rate applications*, such as compressed video - note that both Class 1 and 2 services require delay bounds;
- Class 3 : *connection-oriented data applications*, intended to support applications that had previously run over X.25;
- Class 4 : *connectionless data applications*, which support datagram networking protocols such as TCP-IP.

There are four AAL protocols. AAL1 and AAL2 support class 1 and class 2 applications, respectively. AAL1 employs packetisation/depacketisation at the UNI to convert *continuous bit-stream oriented* (CBO) traffic into cell-based traffic and vice versa. At the receiving side a fully synchronised (clocked) bit stream has to be delivered, which requires tight delay control within the network. AAL1 and support for synchronous voice and H.261 video are prerequisites for ATM/ISDN internetworking. Implementation of AAL2 is difficult because the variable bit rate makes it hard to reserve resources for this kind of traffic. Either the peak bandwidth is reserved - which is not very efficient - or guarantees cannot be held. The details of AAL2 are not yet defined, so products will not be available in the near future.

When it was recognised that, in practice, there is little or no distinction between framing for a connection-oriented protocol and framing for a connectionless protocol, AAL3/4 was specified to handle both class 3 and 4 services. Because ATM is inherently

connection-oriented, connectionless service needs to be provided by connectionless servers, which themselves are accessed through connection-oriented communication. The main function of AAL3/4 is the segmentation and reassembly of large messages. AAL3/4 uses a *message identifier* (MID) field to allow interleaved transport of different messages over the same virtual circuit.

Finally, because of concerns about AAL3/4 performance and complexity, the ATM Forum, a consortium of vendors, carriers, and users, formed to expedite industry agreements on ATM interfaces, proposed AAL5 (also known as the *simple and efficient adaptation layer* or SEAL) instead. It is a substantially lean AAL compared with AAL3/4 at the expense of error recovery and built in retransmission. This trade-off provides a smaller bandwidth overhead, simpler processing requirements, and reduced implementation complexity It does not support interleaving, thus there is no need for a MID field per cell. Consequently, AAL5 provides for better utilisation of the available bandwidth. AAL5 reduces AAL3/4's 4-byte-per-cell segmentation overhead to 8 bytes per message. As such, larger messages use only a fraction of a byte per cell as segmentation overhead, and the cell overhead drops from 17 percent to 11 percent for large packets.

The reassembly process itself is also slightly simplified, since each cell requires less checking. The drawback is that a corrupted cell will always lead to a discarded message in AAL5, whereas AAL3/4 provides means to localise bit errors to individual cells. This is interesting in the context of multimedia communication, since some media streams might be able to use partially correct messages [Par94].

While work has already been done on how to encapsulate existing protocols such as IP over AAL5 [Hein93], [Atk94], [Lau94], the ATM Forum is currently working on a sixth AAL for supporting packetised multimedia streams, in particular for MPEG and MPEG-II video. Issues under discussion include the use of *forward-error-correction* (FEC) techniques to raise link security to a level where no extra error recovery is needed, and the support of MPEG-II synchronisation requirements.

2.2. ATM Switches

Ever since ATM was selected as the multiplexing and switching technique for the B-ISDN, there have been numerous proposals for ATM switch architectures. The interest generated by them has resulted in several books and surveys being published on the topic [AhmDen89], [Jac90], [New92], [Pat93], [Par94], [Tob90], [Zeg93]. The first such switch worthy of mention was the Starlite Switch [HuaKna84]. This was followed in quick succession by a host of other architectures: the Knockout switch [YehHluAca87], Hui and Arthur's switch [HuiArt87], the switches proposed by Turner [Tur88], Lee [Lee88], Kim and Leon-Garcia [KimLeo90], Pattavina [Pat91], Itoh [Ito91], and Chao [Cha91], the Sunshine switch [Gia*etal*91], as well as more recent efforts [Karl92], [EngKarYeh92], [Kim94], [LeeByu94], [ChaCho95]. However, of the vast number of architectures put forward - and we have enumerated but a few above - only a scant minority has seen actual commercial realisation, most authors limiting themselves to a theoretical analysis of the switch followed by a presentation of simulation results. This is in main due to the switches' general poor scalability and reliability, inefficient utilisation of bandwidth, as well as the difficulties encountered in connecting with existing networking technologies [RooCheGar94]. To compound these problems even more, testing of already implemented

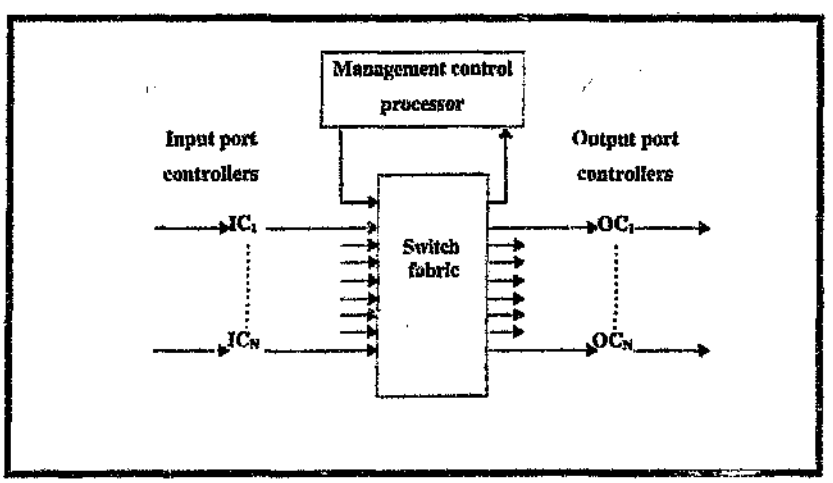

Figure 2 ATM Switch Model

switches is extremely difficult, this being caused by the very features that make ATM such an attractive prospect, namely its flexibility and high bandwidth [Rui94], [Mil94].

A general model of an ATM switch architecture is given in Figure 2. This consists of a set of *N* input and *N* output ports, the ATM *switch fabric* itself, and a *management control processor*. Each input and output port is managed by an *input* and *output port controller*, respectively. Typical functions of the input port controller include buffering, cell duplication for multicasting, cell processing, VCI translation, multiplexing low-speed traffic, as well as managing path connection requests and reservations through the switch fabric. The output port controller carries out similar functions such as buffering, VCI translation and demultiplexing [RooCheGar94].

The management control processor incorporates high-level functions such as connection establishment and release, bandwidth allocation, maintenance, and management. It could for example be the host of a Simple Network Management Protocol (SNMP) or an Open Systems Interconnection (OSI) management agent.

Input controllers are usually synchronised so that all cells arrive with aligned headers at the switch fabric. The effect of this is to simplify the design of the switch fabric and to allow cells to be accepted according to their priority. The switch fabric operates in a synchronous manner and, generally, one cell is taken from an input port and sent through the switch fabric. Cells have their VCI translated in the input controllers before being submitted to the switch fabric. This is done by consulting a lookup table on the basis of the incoming VCI's value. This table could, for a particular VCI, contain additional data such as routing and buffering information, priority and quality of service requirements, as well as specify a particular output port to which the cell is to be directed. Multicast operations are usually supported in ATM switches through the replication of arriving cells by the switch fabric followed by the routing of the multicast cells to the appropriate output port. In these cases, the output port controllers must be able to allow the multicast copies to exit the switch without any required value of a VCI [Sch88].

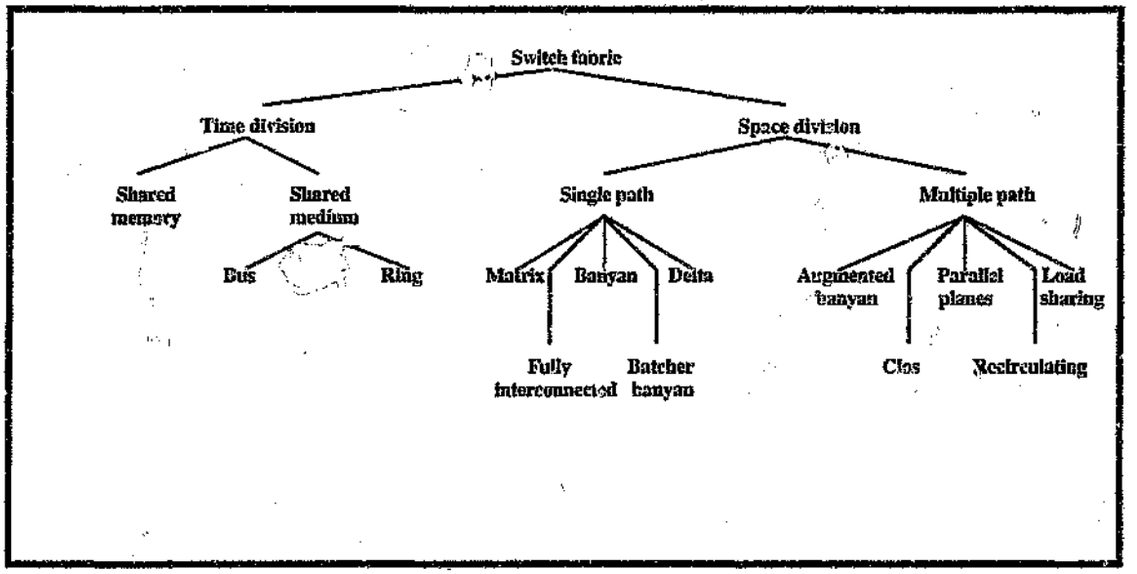

Figure 3 Classification of Switching Fabrics [New92]

The ATM switch fabric is the most important component of the hardware design of an ATM switch. It affects issues like cost, performance, scalability, capacity, as well as complexity of switch design [New92]. As already remarked, many switch architectures have been put forward in the literature, and a unique taxonomy is hard to find, for there are many criteria on which it could be bas For example, Ahmadi and Denzel identify six classes in their survey [AhmDen89], while Tobagi, by taking a more abstract view of a switch fabric, only singles out three such classes [Tob90]. In Figure 3 we have presented Newman's classification, based on the internal architecture of a switch towards the application of ATM as a corporate backbone [New92].

The switch fabric is the mechanism through which cells are routed from input to output ports. In an ATM switch, cell arrivals are not scheduled. As a consequence, cells arriving at different input ports may have the same output port as their destination, a situation referred to as *output port contention*. Since the output port can only accept one cell at a time, this conflict is usually resolved by either having the fabric discard or buffer the contentious cells. The location of buffers in an ATM switch therefore impacts upon its design complexity as well as overall performance.

An ATM switch fabric consists of hardware and software that deal with a host of functions, among which : path establishment between input and output ports, input ports' service discipline, contention resolution mechanisms and switch parallelism (the possibility of having multiple input-to-output port connections) [RooCheGar94]. Usually, switching fabrics are modular and are themselves constructed from basic switching components called *switching elements* (SEs). The hardware implementation of switching elements comes in the form of integrated circuits, while the switch module is realised on one or possibly more circuit cards [New92].

The physical connection between the input and output ports of an ATM switch fabric can be implemented using a *time-* or *space-division* technique. The former can itself be divided into two further classes, *shared memory* or *shared medium*. In a shared memory implementation, the switch consists of a single dual-ported memory shared by all input and output lines. Packets arriving on all input lines are multiplexed into a single stream which is fed to the common memory for storage. In the memory itself, packets are organised into separate output queues, one for each output line. The output stream is formed by retrieving packets from the output queues sequentially, one per queue; this stream is then demultiplexed, and packets are transmitted on the output lines. Two main design considerations arise in this type of architecture : the first is that the management control processor must be capable of selecting N incoming and outgoing packets in each time slot; the second, and more important, is that memory bandwidth should be large enough to simultaneously accommodate all input and output traffic. The Prelude switch, developed by the Centre National d'Etudes des Telecommunications (CNET), is an example of a shared memory switch [DevCocSer88].

In shared medium switches, all packets arriving on the input lines are synchronously multiplexed onto a common high-speed medium - typically a parallel bus or ring - of bandwidth equal to N times the rate of a single line. Each output line is connected to the bus via an interface consisting of an address filter and an output *first-in-first-out* (FIFO) buffer. The role of the interface is, essentially, a demultiplexing one, namely to

receive all packets transmitted on the medium, to filter them according to their VCI, and then write them into the FIFO buffer. Similarly to the shared memory type, this type of switch is based upon multiplexing all arriving packets into a single stream, which is then demultiplexed into several output streams, one for each output line. The distinction between the two types of time-division switches is that the shared medium type has complete partitioning of the memory among output queues. The ATOM switch from NEC represents an example of a shared medium switch [Ito*etal*91].

The main difference between time-division and space-division techniques is that, whereas in the former a single communication highway is shared by all input and output ports, in a space-division switch, multiple concurrent paths are established from the inputs to the outputs, each with the same data rate as an individual line. The result of this is that no memory component of the fabric has to run at a speed higher than the line speed, and that control need not be centralised, but could be potentially distributed throughout the switching fabric. The total capacity of the switch, being the product of each path's bandwidth and the number of paths that can, on average, transmit a cell concurrently, has a theoretical unlimited upper bound. This, however, is limited in practice by a set of physical constraints such as device pinout, connector restrictions, and synchronisation restrictions [New92]. Obviously, time- and space-division techniques can be combined in a switch fabric. One could envision the case of clusters of time-division switches which are interconnected in a hierarchical fashion via a space-division switch [RooCheGar94].

There are two main routing procedures used in space-division switches : *self-routing* and *label-routing*. In self routing, the input port controllers attach a routing tag to incoming cells, based on the same process as in VCI translation, and cells are then routed according to this tag. The self-routing property proves to be important as it reduces the complexity of controlling the switch fabric by distributing the control function over all crosspoints. This mechanism relies on the regular interconnection of switching elements in order to work and results in very fast switches, as the switching elements need only inspect a cell's tag to make the appropriate routing decision. In contrast, label-routing

relies on the VCI in a cell's header to index routing tables in the switching elements of the fabric. This technique is independent of the regular interconnection of switching elements and can be used when switching elements are connected arbitrarily. Although multicasting is efficiently implemented using label-routing, the major drawback of this scheme is the large number of translation and routing tables that must be maintained by the switch fabric. It is not surprising then, that the majority of space-division switches use a self-routing technique. However, hybrid implementations, where self-routing is used for point-to-point connections and label-routing for multicast traffic, have been suggested [New92].

Notwithstanding their advantages, space-division switches are beset by problems of their own. Depending on the particular fabric used and the available resources for path establishment, it may be possible for two or more cells to contend for the same internal link. This characteristic, referred to *as internal blocking*, limits the throughput of the switch and thus becomes a central issue in the architecture design of space-division switches. Consequently, switches can be classified as *blocking* or *non-blocking*. In the former case, paths internal to the switch fabric can share some interstage links; thus the control of packet loss requires the adoption of additional techniques, such as a packet storage capability in the switching elements, or deflection routing (i.e. routing by a path not necessarily the shortest). In the latter case, no storage capabilities are required as all paths are guaranteed to be disjoint and switching elements are much simpler to implement. Nevertheless, more interconnection stages of switching elements are required if the switch fabric is non-blocking. A closely related issue is *buffering*. As already remarked, blocking fabrics need some sort of internal buffering in order to limit potential cell loss. Other buffering strategies are also possible; however, we will defer our discussion of them and other buffering-related issues to the next section.

2.3. QoS issues

The ability of ATM-based networks to provide a guaranteed QoS, while still taking advantage of the statistically multiplexed transport mechanism provided by such networks, is a decisive criterion for the acceptance of this technology by both potential users and network providers. This is a far from easy problem, since not only do service guarantees have to be provided for the wide spectrum of traffic carried by ATM, ranging from bursty, variable rate sources to smooth, constant bit rate sources, but, for an appreciable part of the envisaged applications (such as teleconferencing), it must be done within the confines of real-time constraints. This would mean, for instance, that the corresponding procedures must be computationally simple enough so that their overall complexity is consistent with real-time requirements.

Kurose [Kur93] identifies three more reasons why providing QoS guarantees in high speed networks is a potentially difficult problem. The first is that studies [Le*letal*94] have shown that packetised traffic over ATM networks is highly correlated, self-similar (fractal-like) and thus exhibits behaviour which is much more complex than that associated with traditional data sources. The second is that QoS guarantees cannot be given any longer on an aggregated, network-wide, basis, but should be provided for at a much finer-grained, per-session level. Lastly, performance must be evaluated in a multi-hop network environment, taking into account the interactions between sessions as they pass through the nodes of the network.

2.3.1. Quality of Service Guarantees

QoS guarantees may be broadly categorised into *deterministic* guarantees and *statistical* guarantees. In the former case, a bound is provided for the performance of cells switched during a particular session, while the latter promises that cells that do not satisfy a certain performance criterion will not exceed a specified fraction of the total number of cells. Deterministic guarantees are necessary in hard real-time applications. Statistical guarantees, on the other hand, can further be divided into "steady state" guarantees - ones

that would hold with certainty over an infinite period of time - and guarantees that hold only over specific periods of time. It must be noted, however, that if a session having "steady state" guarantees lasts only a finite amount of time, then there is some chance that the guarantees would not hold. In contrast, if guarantees are only given over time intervals, then performance is assured never to go below a certain threshold for more than a certain fraction of those intervals [Kur93].

Four approaches have been identified for providing QoS guarantees in B-ISDN networks [Kur93] :

- the *tightly controlled* approach - this approach ensures that an individual session's output traffic characteristics are the same as that session's input characteristics;
- the *approximate* approach - here relatively simple models are used to characterise the traffic sources of the network;
- the *bounding* approach - this approach gives guaranteed bounds (be they deterministic or statistic) on a session's performance, based on a worst case analysis, and
- the *observation based* (or *predictive*) approach - here previously made measurements of certain traffic parameters are used to characterise an incoming call and to determine call acceptance.

Woodruff and Kositpaiboon propose a set of traffic management guidelines for providing guaranteed QoS in ATM networks [WooKos90]. These are :

- *simplicity*;
- *reasonable bandwidth utilisation* if statistical multiplexing is used - the increased complexity of statistical multiplexing should only be implemented if the bandwidth utilisation gain is important;

- *performance robustness to traffic uncertainties* - this is because it will be difficult to predict a connection's exact traffic flow characteristics;
- *component decoupling* - a network-wide (re-)analysis of performance should not be done each time the topology or connection routing of the network changes, and
- *node architecture independence* - traffic rules should be independent of the underlying architecture of switching nodes.

2.3.2. Quality of Service Attributes

At the user-level the service provided by the underlying network can be categorised into two broad classes : *best-effort* and *guaranteed.* With best-effort service, the network provides a service that strives to perform its functions to the best of its capability and tries to use the available resources as well as possible, but does not guarantee performance or resource availability. This is in contrast to the guaranteed type of service where system performance and resource usage are certified to meet agreed upon requirements. However, when dealing with the heterogeneity of distributed multimedia applications supported by an integrated services communication network, quality of service parameters must further be provided for on at least three abstraction layers beneath the human user. These are [NahSte95] :

- the *application level* - here QoS parameters are usually specified in terms of media quality (such as media data-unit rate and end-to-end delay) and media relations (such as media conversion and inter/intrastream synchronisation);
- the *system level*, including communication and operating system services - these can be further subdivided into quantitative parameters (these typically include bit rate, number of errors, task processing time and data unit size) and qualitative parameters (such as the types of interstream communication, scheduling and error-recovery methods employed), and

- the *device level*, including both network and multimedia devices - the parameters associated with multimedia devices usually specify timing and throughput demands, while network devices' parameters typically characterise network load and performance.

In the case of ATM, network load parameters that need to be specified when setting up a connection are [SiuJai95] :

- *Peak Cell Rate* (PCR) - the maximum cell rate at which the user will transmit; it is the inverse of the minimum cell inter-arrival time;
- *Sustained Cell Rate* (SCR) - this is the average cell rate measured over a long period of time;
- *Minimum Cell Rate* (MCR), and
- *Burst Tolerance* (BT) - this determines the maximum number of back-to-back cells (the *maximum burst size* or MBS) that can be sent at the peak cell rate. BT and MBS are related through the formula :

$$BT = (MBS - 1)\left(\frac{1}{SCR} - \frac{1}{PCR}\right) \qquad (1)$$

Performance parameters which can be specified in an ATM context are :

- *Cell Loss Ratio (CLR)* - the ratio of lost cells to the total number of transmitted cells;
- *Cell Transfer Delay* (CTD) - the end-to-end delay experienced by a cell in the network;
- *Cell Delay Variation* (CDV), also known as *jitter* - a high value implies large buffering requirements for delay-sensitive applications;

- *Cell Misinsertion Rate* (CMR) - the number of misinserted cells per connection second, and

- *Cell Error Ratio (CER)* - ratio of errored cells to the total number of delivered cells.

A cell is deemed to be lost if it never reaches its destination, or if it does so, but only after expiration of a time-out period. A cell is also considered to be lost if there are buffer overflows within the network. A cell is errored if it arrives in time but there are errors in the cell header that cannot be corrected. A cell is misinserted if it arrives at its destination on a wrong virtual connection. Undetected or erroneously corrected header errors can produce misinserted cells [HanHubSch94].

2.3.3. ATM Traffic Contract

Any user of distributed multimedia applications such as multimedia mail, conferencing, video on demand, screen sharing and virtual desktops, expects a certain level of quality associated with these services. To provide this quality, a *traffic contract* is established during ATM connection set-up. Here, user-defined requirements are communicated in the form of QoS parameters to the resource management entities of all system constituents involved. A *negotiation* stage is then entered into, in which the final values of the QoS parameters are agreed upon (or not, in which case the connection is refused). Depending upon the differing specifications between layers, the negotiation process can be preceded as well as followed by one of *QoS parameter translation*. Lastly, the required resources must be *admitted*, *reserved*, and *allocated* for on the path between sender(s) and receiver(s) [NahSte95].

The QoS parameters thus negotiated completely characterise the type of service. Guaranteed services are required for if QoS parameters are specified with either deterministic or statistical bounds [Fer90]. On the other hand, predictive service is called for if QoS parameter values are estimated from past behaviour of a service [ClaSheZha92]. Lastly, unspecified QoS parameters in a contract can be provided for on a "best-effort" basis.

The following types of traffic are usually specified in an ATM traffic contract [SiuJai95] :

- *Constant Bit Rate* (CBR) - used for circuit switching emulation; if the *cell loss priority* (CLP) bit of the ATM cell header is zero (i.e. high priority) then the CLR must be specified; if CLP = 1 (low priority cells) the CLR may or may not be specified;

- *Variable Bit Rate* (VBR) - this is further subdivided into real-time VBR (VBR-RT) and non-real-time VBR (VBR-NRT) depending on whether or not the application is sensitive to jitter; CTD is specified in both categories, while CDV is only specified for VBR-RT;

Attribute	CBR	VBR-RT	VBR-NRT	ABR	UBR
CLR for CLP = 0	Specified			Specified	Unspecified
CLR for CLP = 1	Optional			Specified	Unspecified
CTD	Specified	Specified	Specified	Unspecified	Unspecified
CDV	Specified	Specified	Unspecified	Unspecified	Unspecified
SCR and BT	Not applicable	Specified	Specified	Not applicable	
PCR and CDVT	Specified			Specified	Specified
MCR	Not applicable			Specified	Not applicable

Table 1 ATM Layer Service Categories [SiuJai95]

- *Available Bit Rate* (ABR) - this is usually provided for normal data traffic such as file transfer and e-mail; MCR can be specified in the contract, while the CTD and CLR are to be kept to a minimum possible, and

- *Unspecified Bit Rate* (UBR) - this class of traffic is designed for applications that are insensitive to both cell loss and delay, with file transfer and e-mail being typical examples; ABR and UBR are the types of traffic usually specified when a "best-effort" service is required of the ATM network.

In the context of QoS negotiation, two main approaches can be distinguished [NahSte 95] : *peer-to-peer* (e.g. application-to-application) or *layer-to-layer* (e.g. human

user-to-application or application-to-system). The number of parties participating in the negotiation process is commonly limited to two or three. Accordingly, *bilateral peer-to-peer negotiation* takes place between the caller and callee of a specific service, with the service provider not being allowed to change the parameter values suggested by the users. *Bilateral layer-to-layer negotiation* only takes place between the user and provider of a specific service. This type of negotiation involves two scenarios : (1) between service users and providers *locally*, e.g. between an application and the underlying operating system, and (2) between host and network, as happens in the case of multicast multimedia streams. Lastly, *triangular negotiation* takes place between both the caller and callee as well as the provider of the particular service. Here, the QoS parameter values are passed from the caller to the service provider which, unlike the situation associated with bilateral-peer-to-peer negotiation, can possibly modify them as long as they are within a specified *range*. The parameters are then passed on to the callee, in whose power the final decision of whether or not to accept the call rests.

2.3.4. Call Admission and Session Policing

Closely related to the problem of QoS is *congestion control*. The primary function of congestion control is to ensure good throughput and delay performance while maintaining a fair allocation of network resources. Because of the high flexibility of ATM based networks - needed for the varied traffic types which they handle - special congestion control mechanisms need to be introduced. Two complementary and related functions, the *call acceptance control* (CAC) function and the *policing* function will implement this control. The CAC function acts only during call set-up phase to decide whether a new connection should be admitted or not and to determine how much virtual bandwidth (resources) should be allocated to the call on acceptance. These decisions depend on the traffic already supported by the network and on the declared parameters of the source requesting access. The parameters themselves could at one extreme be a full characterisation of the source (unlikely), or, at the other extreme, be only its peak cell rate or a partial characterisation such as peak and mean cell rates together with the maximum

burst length. A call, once accepted, should respect its declared (or negotiated) parameters, while the network must guarantee the negotiated QoS [Fer90].

In the context of call admission procedures to guarantee a QoS, much attention has been devoted to the notion of *effective bandwidth* of sources. The notion of effective bandwidth describes a parameter which lies in between the mean and peak rates of sources, such that if the allocated bandwidth is lower than the effective bandwidth then the QoS will be violated, whereas allocating bandwidth greater than the effective bandwidth will imply that the QoS will be met.

Several network traffic control functions such as flow and congestion control, as well as routing and call admission policies depend on the characterisation of the effective bandwidth of individual connections and the resulting load on network links. These are essentially asymptotic results based on the transmission rate function associated with the traffic source and the queueing buffer mechanism which models the destination. The effective bandwidth method can, by virtue of the taxonomy already presented, be categorised as a bounding approach [CopDamMel93]. Guerin, Hamadi and Naghshineh have derived simple approximations for the effective bandwidth of both individual and multiplexed connections, based on their statistical characteristics and desired QoS. Denoting by R_{peak} and ρ the peak rate and utilisation, respectively, of a connection, with b the mean of the burst period, with x the available buffer space, and with $a = -ln(\varepsilon)$ the natural logarithm of the inverse of the buffer overflow probability ε, the equivalent capacity of an individual connection is found to be :

$$c = \frac{ab(1-\rho)R_{peak} - x + \sqrt{[ab(1-\rho)R_{peak} - x]^2 + 4abx\rho(1-\rho)R_{peak}}}{2ab(1-\rho)} \qquad (2)$$

When N multiple connections are multiplexed, the equivalent capacity is given by:

$$C = \min\left\{\sum_{i=1}^{N} m_i + \sqrt{[2a - \ln(2\pi)](\sum_{i=1}^{N} \sigma_i^2)}, \sum_{i=1}^{N} c_i\right\} \qquad (3)$$

Here m_i and σ_i are the mean bit rate and standard deviation of an individual connection, while the c_i for each connection are given by (2). The authors use both a fluid flow and a stationary model to arrive at their approximations, whose computational simplicity make them practical to be applied in real-time network traffic control applications [GueAhmNag91].

Related to this work is the one done by Low and Varaiya. They come up with an iterative, decentralised algorithm, where the network offers different services for rental to the users, who, taking into consideration the resource (bandwidth and buffer) prices at nodes along their envisaged route, together with their own traffic and quality parameters, decide, subject to budgetary constraints, how to best combine the available services to meet their needs [LowVar93].

The function of the policing mechanism is then to ensure that each accepted source conforms to its parameters. Thus, the policing function could intervene when a source input exceeds any of the negotiated parameters in order to prevent a degradation in the QoS of other connections. At all times, a connection should experience very low cell loss rates, typically 10^{-9}. Furthermore, the policing function must operate in real time, which means that it must detect any contract violation fast enough to prevent any QoS degradation. It is to be remarked that the above requirements do not in any way specify how the policing mechanism must intervene to attain the objectives. When a policing mechanism detects that emitted cells from a source are in contravention of the contract, three modes of intervention are possible : *rejection*, *tagging*, or *smoothing* [Rath91]. Rejection is the simplest strategy applicable and consists of outright discarding the contravening cell. With tagging, a cell is marked (most commonly by having a bit status flag set) as having contravened the session's traffic contract, but is allowed to continue

along the route to the destination. However, at any subsequent point along this route, the tagged cell cannot obtain network resources (such as buffers and bandwidth) ahead of untagged cells - if resource contention does happen, the tagged cell is usually *recirculated*, *deflected* or rejected. Using recirculation, contention is handled by passing violating cells through a *recirculation network* (a set of simple delay units) at the output of which they will contend for the required resource again in the next time slot(s) with newly arriving packets. The deflection strategy results in a cell being delivered to its desired destination, but using a path other than the shortest available one. The taken path might either be a recirculation path or an alternative, longer, forward path. Lastly, smoothing tries to control bursty network traffic through various congestion management algorithms.

A heuristic framework for the design and analysis of the source policing problem was presented by Rosenberg and Lague [RosLag94]. They introduce two independent functions to characterise the source and policing mechanisms, respectively, in ATM networks. It is shown that the two functions, used in conjunction, yield a simple tool providing good approximations of the response time (time to detect contract violations) of policing mechanisms.

In ATM networks, there are two main congestion control algorithms used by the policing function : *leaky bucket* and *token bucket*. In the leaky bucket scheme all arriving cells of a flow are put into a bucket of depth b and are drained out of the bucket and sent on the network at a sustained rate r. If more cells arrive than the bucket can store, the excess cells are discarded (or alternatively, are sent over the network with the CLP bit set to 1). The main effect of this scheme is that it converts a bursty stream into a more regular pattern, but, although it works well for constant bit rate data flows, variable bit rate data flows must request that r be set equal to the peak cell rate. This, however, is extremely poor statistical multiplexing and, in consequence, a waste of system resources. The token bucket alleviates this shortcoming and is a variation of the leaky bucket scheme : here r is the rate at which tokens are placed in the bucket. In the general case, each token represents a single bit, and sending a packet consumes as many tokens as there are bits in the packet - for cell networks such as ATM, this is extended so that each token represents

a cell; sending a cell consumes one token. A flow is said to conform to its token bucket filter if no cell departs when the token bucket is empty. When the flow is idle or transmitting at a lower rate, tokens are accumulated up to *b* tokens, newly arriving tokens being discarded. Thus, flows that have been idle for a sufficiently long period of time can transmit a whole bucket full of data back-to-back, permitting burstiness, but bounding it [Par94].

2.3.5. Resource Reservation and Allocation

Resource reservation and allocation are based on the results of the call admission process and have as main objective the efficient utilisation of scarce resource capacities by reserving only the real (or best approximation thereof) demand at any particular moment in time. The reservation can be done according to one of two possible approaches : *pessimistic*, in which resources are reserved assuming a worst-case situation, and *optimistic*, in which resources are reserved according to an average-case scenario [NahSte95].

In general, resource reservation/allocation protocols work according to the following mechanism with "accept" response [NahSte95] :

- the initiator sends QoS parameter specifications in a reservation request;
- at each network entity along the path the reservation protocol passes a request to the respective resource manager;
- the resource manager then reserves the resources according to one of the two modalities specified above and, if necessary, updates the QoS parameters;
- at the end of the path the last entity sends an allocation message (with an accept/modify/reject primitive) together with - if the connection is not rejected - the possibly new set of QoS parameters on the backward path towards the initiator of the connection;

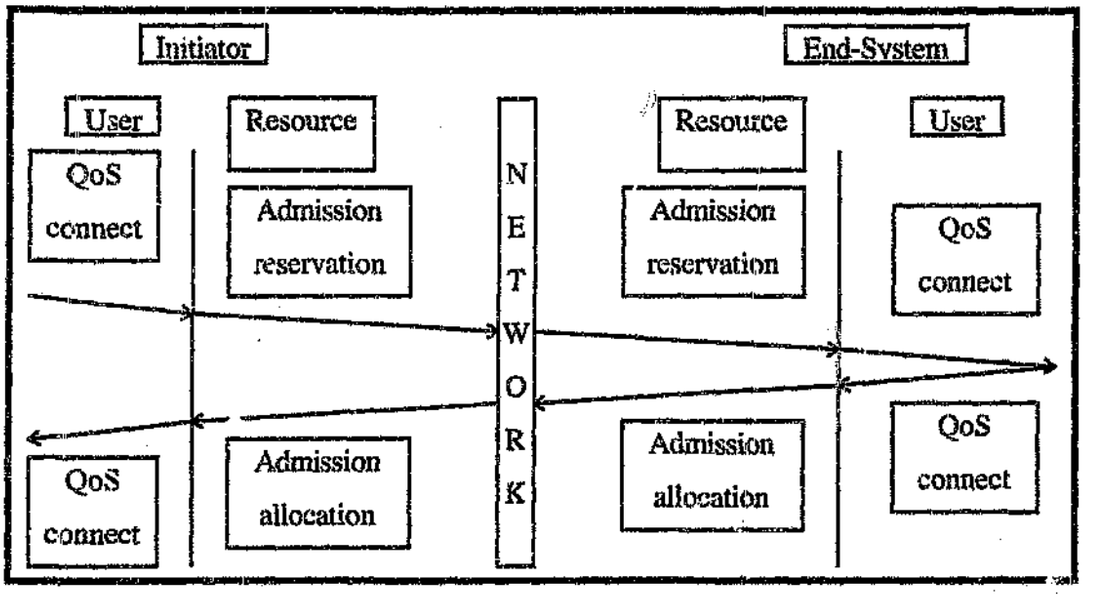

Figure 4 Resource reservation/allocation protocol with "accept" response [NahSte95]

- along this path, the network entities' resource manager allocates, relaxes or releases resources according to the response primitive contained in the allocation message.

Resource reservation protocols can be further categorised according to *direction* and *style*. Thus, as regards the first criterion, the protocol can either be *sender-* or *receiver-initiated* [NahSte95]. Widely used receiver-initiated protocols include RSVP [Zha*etal*93] and the Tenet protocol suite [Ban*etal*94], while Version 2 of the Stream Protocol, ST-II [Top90], is an example of a sender-initiated protocol. Style differentiates the reservation protocols according to the number of receivers (single or multiple) and senders (also single or multiple). For a sender-initiated reservation, the style might comprise the sender creating a single reservation along the route to the receiver, or a multicast reservation to several targets.

In a receiver-initiated reservation protocol, *filters* can specify which data sources the receiver is prepared to accept : in the *wildcard-filter* style the receiver makes a single resource reservation shared among all senders for the given session; fixed-filter reservation allows the receiver to receive data only from a list of senders specified in the original

request; lastly, a dynamic-filter reservation reserves resources for *N* senders but allows the membership (but not the cardinality) of this list to change dynamically [Zhaetal93].

2.3.6. Queue and Buffer Management

Queueing is a fundamental consequence of the statistical sharing that occurs in cell networks. Both queueing sizes and queueing delays determine the effective cell loss ratio and network traffic load. Accordingly, *queue management* is very closely linked to the problem of QoS [Par94].

Queue management policies can be distinguished according to whether or not they are *work conserving*. If a policy is work conserving, then no work is created or destroyed within the system. This would mean, for example, that a switch with a non-empty queue of cells waiting to be transmitted would not be idle, and that no cell leaves the switch before it is completely serviced. The simplest cases of work conserving queueing disciplines would be FIFO and LIFO (*last-in-first-out*). An example of a discipline that is not work conserving would be a switch that waits a random amount of time before serving the next packet in the queue [Kle75b].

From a theoretical point of view, it has been shown that, as long as the queueing discipline is work-conserving and selects cells to be serviced in a manner that is independent of their service time, all such queueing disciplines will boast the same average number of cells in the system and the same average cell waiting times [Kle75b]. Thus, once one proves that a queueing discipline satisfies the above requirements, one has a large body of theoretical results to draw upon.

In an ATM context, queue and buffer management are distinguished at two levels: one is the end-to-end (session) level, the other is the switch level. They are now presented according to this taxonomy.

A. Session-Level Queue and Buffer Management

FIFO+ is a queueing discipline which induces FIFO-style sharing across all the hops of path to reduce jitter. Cells are divided into classes, and each class computes its average delay for any particular hop. For each cell, the difference between the average delay of its class and its own particular delay is then added (or subtracted) from a field in the cell's header. This field then allows each switch to schedule the cell as if it had arrived at its real arrival time plus this offset. The net result is that cells that have been "jumping the queue" on one hop are pushed to the end of the queue at the next hop, while those that have arrived behind schedule are pushed to the front of the queue [ClaSheZha92].

Demers, Keshav and Shenker have also proposed a slight variation on the classic FIFO queueing discipline [DemKesShe90]. Their *fair queueing* algorithm tries to remedy one of the major shortcomings of a FIFO discipline, namely that, although it provides a fair allocation of cells, it does not provide a fair allocation of bandwidth because of variations in cell sizes. Thus, instead of basing their scheduling algorithm on a cell round-robin scheme, they rely on bit-by-bit round-robin service, a scheme which is clearly fairer. Although cells are not transmitted in a bit-wise manner, they are scheduled to be transmitted at the time when their last bit was supposed to have been sent if a bit-by-bit round-robin discipline would have been employed, an asymptotically equivalent scheme. A limitation of this scheme is that each source gets the same fraction of bandwidth, which would be discriminating against sources such as file servers which would need considerably more bandwidth. To alleviate this problem, a *weighted fair queueing* discipline has been suggested [DemKesShe90]. Here, the bandwidth is partitioned as if there were more queues in the system than there actually are, with the sources requiring extra bandwidth being allocated supplementary queues. Parekh and Gallager have reintroduced the weighted fair queueing algorithm under the name of *packetised generalised processor sharing* and have proven that, if token bucket flow control coupled with weighted fair queueing is enforced throughout the network, strict upper bounds can be given for the delay and buffering requirements needed by any class of traffic in the network [ParGal92].

The *Virtual Clock* algorithm involves an extremely similar underlying cell scheduling algorithm to weighted fair queueing [ZhaKes91]. Thus, it emulates time division multiplexing, allocating each cell a transmission time as if the service provider were really doing time division multiplexing. However, the algorithm was expressly designed for a context whereby resources were pre-apportioned and therefore had as a fundamental part of its architecture the assumption that flow rates were arbitrary.

In classic *Earliest Due Date* (EDD) scheduling, each packet is assigned a deadline, and the packets are sent in order of increasing deadlines. The *Delay EDD* [FerVer90] service discipline is an extension where the server negotiates a service contract with each source. The contract states that if a source obeys a peak and average sending rate, then the server will provide a delay bound. The key lies in the assignment of deadlines to packets. The server sets a packet's deadline to the time at which it should be sent had it been received according to the contract. By reserving bandwidth at the peak rate, Delay EDD can guarantee a channel delay bound.

Jitter Earliest Due Date (Jitter EDD) is one of the most widely used schemes for jitter control. It is an extension of Delay EDD and, similarly, is a non-work-conserving policy. At each hop, any departing cell is stamped with the difference between its deadline and transmission time. At the next hop, this difference, call it t, is added onto the cell's earliest and latest transmission times (if the jitter bound for the switch is j and the latest possible transmission time is d, then the earliest possible transmission time is d-j), giving the new time range between which the cell must be sent : $[t+d-j, t+d]$. Thus, a cell's transmission time at any hop along its route is determined by and guaranteed to compensate for the queueing jitter of the previous hop. If this scheme is employed at all network hops, then network-wide jitter guarantees can be provided for [VerZhaFer91].

The *Stop-and-Go* queueing discipline [Gol90] is a non-work-conserving discipline which aims to preserve the 'smoothness' property of traffic as it traverses the network. Time is divided into frames. In each frame, only packets that arrived at the server in the previous frame are sent. It can be shown that with this scheme, a packet receives both a

minimum and a maximum delay as it goes from a source to a destination. Since the delay and jitter bounds are linked to the frame time, Stop-and-Go proposes multiple frame sizes.

The *Hierarchical Round Robin* (HRR) server has several service levels, where each level provides round-robin service to a fixed number of slots [KalKanKes90]. Again this is a non-work-conserving discipline where a channel is allocated some number of service slots at a selected level, and the server cycles through the slots at each level. The time a server takes to service all the slots at a level is called the frame time at that level. The key to HRR lies in its ability to give each level a constant share of the bandwidth. 'Higher' levels get more bandwidth than 'lower' levels, so the frame time at a higher level is smaller than the frame time at a lower level. Since a server always completes one round through its slots once every frame time, it can provide a maximum delay bound to the channels allocated to that level.

Kroner describes different *buffer management* mechanisms for ATM networks, according to the value of the CLP bit in each cell's header [Kro90] :

- *Common buffer with pushout mechanisms* - here the buffer is shared among cells of both priority classes, but if a cell with a high priority (CLP=0) arrives and finds the buffer full, then a low priority cell (CLP=1) in the buffer (if there is one) will be discarded to make room for the cell with higher priority

- *Buffer separation* - a simple to implement strategy, with each priority class having its own separate buffer

- *Partial buffer sharing* - this is a threshold-based scheme, where although high priority cells have access to the whole buffer, low priority ones only gain access if the buffer's occupation does not exceed a certain threshold *t*; by varying *t*, different load situations can be catered for by the buffer, and this strategy was shown to be the best compromise between performance and implementation complexity.

In a distributed multimedia context, three main techniques for buffer management design can be identified : *data copying*, *offset management*, and *scatter/gather* systems [NahSte95].

A buffer management technique based on data copying across protocol boundaries allows for the perfect separation of different protocol layers, with each layer allocating exactly the amount of memory needed to hold the packet currently being processed. However, protocol processing normally needs little memory and CPU bandwidth compared to the overhead incurred when copying the data twice - once when receiving and once more when transmitting - and such a scheme is usually employed only for comparison purposes.

In an offset management approach, a buffer is placed in one memory segment and the buffer must be large enough to contain all data to be transmitted, as well as all the *Protocol Control Information* (PCI). In order to transmit, an application will place its data into the buffer, leaving enough space for the PCI. As the buffer is passed through the communication system, each protocol entity will update the offset in the buffer so as to reserve space in the same buffer for its PCI. The offset management technique is easy to implement, with little processing overhead for each buffer, and prevents data copying. One disadvantage is the fact that all buffers must be maximally-sized, so as to ensure enough space for all protocol information. However, there exists a large class of protocol requirements, such as segmentation and reassembly, that cannot easily be solved by offset management alone, unless the buffer management system resorts to data copying.

In a scatter/gather system, each request for space in a buffer is satisfied by linking in a new memory segment. A control structure is used to keep track of all the memory segments for a particular buffer. The scatter/gather system prevents data copying, and also allocates memory space only as it is needed, thereby not causing memory over-allocation. This technique enables the efficient implementation of protocol requirements such as the dynamic expansion of buffers, segmentation and reassembly, and concatenation and separation. Furthermore, the support of logical copies of a buffer can only be efficiently

implemented with a scatter/gather system. A scatter/gather buffer management system needs only to mark the memory segments as having multiple references to them, and to create more control structures referencing the same memory segments. Although the scatter/gather system has many positive properties, with some being indispensable for an efficient implementation of a buffer management system, it still has the disadvantage of a high overhead for control structures, especially for small memory segments.

B. Queue and Buffer Management - Switches

Buffer placement and congestion control are major issues in the design of ATM switches. Both issues lead to queueing and the standard reference on the subject still remains Kleinrock's comprehensive two volume treatment [Kle75a], [Kle75b]. The main buffering strategies used in ATM switches are to place buffers either at the input or output ports, or to use a combination of the two.

With *input buffering*, a separate buffer is placed on each input to the switch. A packet arriving at an input line first enters the buffer and, if it cannot proceed due to a conflict, it remains in the buffer waiting to be switched at a later time. Placing the buffers at the input of the switch separates the buffering and switching functions, which is very desirable from the point of view of layout and circuit compactness. The simplest queue discipline applicable from a control and implementation point of view is FIFO in which only the head of the line (HOL) of each input queue may contend for access to the switch's outputs. If a packet is successfully switched, then it is removed from the input buffer, and the next in line is considered in the following time slot; otherwise the packet remains at the head of the queue and contends again in the following time slot. Although very simple, the main problem with FIFO is *HOL blocking* : if there are k packets contending for the same path, only one packet will be served in a time slot, and the remaining k - 1 wait for the next time slot. This also implies the fact that k - 1 output lines will remain idle in that time slot, despite the fact that there may be packets, queued behind the unsuccessful HOL packets and destined to idle output lines, which are nevertheless being prevented from reaching their destinations. The starvation that thus results drives

the switch's throughput below its nominal value. It has been shown that a large (i.e. $N \rightarrow \infty$, where N is the number of inputs) input buffered switch with a FIFO queueing policy, saturated with uniform random traffic, has a throughput of $2 - \sqrt{2} = 0.586$, and that this is the maximum achievable for large N [KarHluMor87].

It is possible to increase the throughput, however, with a technique called *input queue bypass* (or *windowing*). This relaxes the FIFO queueing discipline and allows packets behind the HOL to be considered for routing. Accordingly, each input still sends at most one packet into the switch fabric per time slot, but not necessarily the first packet in its queue, such that no more than one packet is allowed to pass through the switch fabric to each output in a time slot. The HOL packets contend first for access to the switch outputs. Those inputs not selected to transmit their first packets then contend with their second packets for access to any remaining idle output ports, and so on, repeatedly, up to a maximum window size w. Simulation results have shown that the throughput increases with increasing values of w, but has diminishing improvements beyond $w = 4$ [HluKar88].

Two other techniques have been adopted to relieve the throughput degradation due to HOL blocking. They are *switch expansion* and *channel grouping*. In the former case the number of output ports M is greater than the number of input ports N. Having $M > N$ improves the throughput performance by reducing the number of output conflicts. An expansion factor of $M/N = 8$ is enough to increase the switch capacity above 0.9. In channel grouping, the switch's output ports are subdivided into groups of size R each and packets address these groups, and not individual members. The allocation network does become more complex with this scheme, but a capacity of over 0.9 is provided by a group size of $R = 32$ [Pat93].

With *output buffering*, all queueing is done at the outputs of the switch with a separate FIFO queue provided for each output. One can think of the switch fabric as operating N times as fast as the inputs and outputs so that if k ($k = 1, \ldots, N$) packets

arrive in the same time slot on different inputs all addressed to the same output, all k can be routed through the switching fabric and into the proper output queue within the same time slot. Only one packet, however, can be transmitted on the output line in any time slot; the remaining k - 1 packets must wait in the FIFO output queue for transmission during subsequent time slots. Note that with output buffering, unlike input buffering, arriving packets do not interfere with packets going to different outputs. It is only at each output port that one could find the unavoidable congestion caused by multiple packets simultaneously arriving on different inputs addressed to the same output. It can be shown that the delay versus throughput performance in output buffered switches is optimal since the packet delay is only determined by the congestion for access to the same output link [HluKar88].

A hybrid approach that can be taken is to provide the switch with both input and output buffers. The inherent complexity of such a queueing arrangement (called *input-output buffering*) generally requires implementations based on multistage self-routing structures. Two variants are possible : the first allows up to k packets to be switched to each output queue during any particular time slot. Here, the hardware required to guarantee the absence of external conflicts is substantially the same as in input queued architectures with channel grouping, and a speed-up of k can be achieved. The second approach assumes that only input (and not output) queues can overflow. Here suitable hardware has to be designed to signal eventual saturation of the output queues back to the input queues, and cells that would otherwise have been discarded at output queues will be retained in the input queues for transmission during the next time slot. Results obtained using this variant of input-output queueing have shown that switch capacity grows with increasing output buffer size, thus lending support to the idea that a prescribed switch capacity can be obtained acting on the output buffer size, given a speed-up compatible with the selected architecture and current technology [PatGia94a], [PatGia94b]. The switches suggested by Lee [Lee90] and Pattavina [Pat91] employ this kind of buffering.

2.4. Pricing in ATM Networks

While pricing policies exist for non real-time networks (datagram networks), as well as for conventional public utilities networks (such as the telephone network) offering a single QoS, ATM networks offering real-time capabilities with multiple qualities of service are complex enough to warrant a new look at accounting in networks.

The main objectives of a network pricing policy would be to [ParFer92] :

- allow the service provider to collect revenue commensurate with the QoS provided to the client;
- allow clients to predetermine the charges for the services they need;
- discourage client actions that will decrease the efficiency of the network;
- allow the policy mechanism to be implemented with minimal overhead, and
- allow a simple relationship between the performance characteristics of the network and the revenue derived from the network.

To meet these objectives, prices could be made sensitive to [ParKesFer92] :

- the amount of resources that are reserved;
- the number of cells used;
- the duration of the connection;
- the time of the day;
- the priority of the connection, and
- the quality of service.

Based on the above factors, different *pricing policies* can be elaborated. These policies have two main aims : to generate revenue to the provider commensurate with the services rendered and to have a fair nature (e.g. a user requiring a short session should not

end up subsidising already connected users with longer sessions). However, it must be mentioned that, in the final instance, the choice of pricing policy does depend on what can be observed, measured, and accounted for, and realisation of a pricing policy that would be fair to all users is extremely hard to achieve. Bearing this in mind, three different pricing schemes that have been suggested for cell-based networks are [ParKesFer92]:

- *per packet pricing*, which charges users only on the basis of the number of packets that they send, regardless of the rendered QoS;
- *set-up pricing*, which charges users for connection establishment, after which users are charged either according to the per packet pricing scheme or to a pricing scheme which takes into account delivered QoS, and
- *peak period pricing*, in which users are charged extra for requesting connections at peak times of network operation.

Invoicing the users based on a differentiated pricing policy has however been shown to be more Pareto-efficient than a conventional flat rate scheme and, furthermore, can encourage users to behave in a certain manner beneficial to the sharing of scarce resources in a network [Coc*etal*91]. Thus, the choice of pricing schemes in ATM networks lies between set-up and peak period pricing.

Parris and Ferrari have put forward a peak period pricing policy in which the pricing equation is given by [ParFer92]:

$$Total\_Charge_i = QoS_i * t_i * tod_i \qquad (4)$$

Here QoS_i is the type of service required by the client for channel i, t_i is the lifetime of channel i, while tod_i represents the period (time of day) factor associated with channel i.

It is suggested that QoS_i be made a weighted average of the client's required bandwidth, buffers, CPU time reserved, and session delay :

$$QoS_i = \alpha * reserved\_bandwidth_i + \beta * reserved\_buffers_i + \gamma * reserved\_cpu\_time_i + \delta * delay_i \quad (5)$$

The values of the coefficients α, β, γ and δ are chosen to reflect expected traffic loads and scarcity of resources, as well as general economic and political conditions. It is remarked that, if users are to be encouraged to chose an appropriate type of service suited to their needs, negotiating the time and price of a service can be incorporated as part of the existing quality of service negotiations.

It must be mentioned that (4) can easily be extended to a set-up pricing scheme simply by additively including the set-up cost of the connection. This is precisely the approach taken by Parris, Keshav and Ferrari who also suggest that a user be characterised by the triplet *(QoS_class, connection_duration, available_budget)* during the negotiation process [ParKesFer92]. As the time of the day factor can be multiplicatively incorporated in the *QoS_class*, the only new thing that this triplet introduces compared to (4) is the *available_budget* parameter. At the end of the negotiation, the network will respond with either an *(Accept, Price)* or *Reject* message. The authors report that usage of a peak-load pricing scheme spreads network utilisation more evenly and raises revenue, while simultaneously reducing the call blocking probability. However, it is also concluded that, for the same revenue generated, a set-up pricing policy is better than peak-load pricing since the call blocking probability of sessions is reduced.

3. PROTOCOL DESIGN

The advent of distributed multimedia systems such as client/server multimedia applications, conferencing, screen sharing, and virtual desktops has meant that new network and resource management techniques need be elaborated to deal with their specific bandwidth and QoS requirements in an integrated fashion [WooKos90], [Rath91], [NahSte95]. One aspect of this trend has been the perfection of protocols for QoS negotiation, reservation, and resource administration. The best known of these are Version 2 of the Internet Stream Protocol [Top90] and the Tenet Protocol Suite developed at the University of California, Berkeley [Ban*etal*94], while an Internet protocol stack based on the Resource Reservation protocol [Zha*etal*93] is currently being developed

We have developed a call-admission protocol that provides dynamic QoS management of distributed multimedia. The main objectives of the protocol design were to incorporate the influence of the user in the overall QoS framework and to formulate QoS management strategies providing for the negotiation and dynamic administration of multimedia sessions' QoS. Once this has been accomplished, the extent to which a multimedia session's QoS, as perceived by a human subject, can be sacrificed in order to accommodate a greater number of sessions, may be studied.

Separately, two other objectives were studied in our research. The first was the influence that the usage of a pricing policy in conjunction with the developed call-admission protocol has on congestion control and QoS negotiation. Accordingly, a pricing scheme, closely linked to the implemented QoS management strategies, has been elaborated. The second was the construction of a protocol knowledge base which the protocol can dynamically consult in order to obtain QoS management recommendations.

3.1. Human QoS and Extended QoS

Research has been done [Apt*etal*94], [Apt*etal*95a], [Apt*etal*95b] evaluating the QoS from a new perspective which takes into account the role of the user in the perception of multimedia data. The existing QoS parameters, as specified in ATM documents, have a purely technical aspect, and we shall refer to them as *technical QoS* (it could, for instance, be made up of bandwidth required, peak cell rate, sustained cell rate, cell loss ratio, cell error ratio, cell transfer delay, jitter, burst tolerance, and minimum cell rate). In the above-mentioned research it has been suggested to enhance existing technical QoS parameters with *human receptivity*, generating a new *human QoS* measure. Human receptivity is a term (expressed as a percentage measure, with 100% indicating complete user satisfaction with the multimedia data) that encompasses diverse aspects of user acceptance of a video message, such as clarity and acceptability of audio signals, continuity of visual messages, lip synchronisation during speech, and the general relationship between visual and auditory message components.

Research in our department has derived a set of human receptivity versus required bandwidth curves for a total of eight different categories of data resulting from a common *Video Classification Scheme* (VCS) [Apt*etal*95b]. These *VCS curves* were obtained on the basis of the temporal nature of the data (i.e. its dynamic nature) as well as the importance of the auditory and visual components of the video message. Various video clips were thus classified into eight main categories as detailed in Table 2 overleaf.

The result most relevant to our research which has been obtained has been that the dependency between human QoS and technical QoS is non-linear [Apt*etal*95b]. Consequently, for certain human QoS ranges, a small variation in the human QoS leads to

Category Number	Video Information	Temporal Nature of Data
1	Logo/Test Pattern	Low
2	Snooker	Low
3	Talk Show	Low
4	Stand-up Comedy	Low
5	Station Break	High
6	Sporting Highlights	High
7	Advertisements	High
8	Music Clip	High

Table 2 Video Classification Examples [Apteral95b]

a much larger relative variation of the technical QoS (required bandwidth being the main measure of technical QoS). This in turn leads to an interesting trade-off raising the possibility of sacrificing a small amount of human QoS in return for the release of a much larger relative amount of bandwidth (we shall refer to this feature as the *asymptotic property* of the VCS curves), which could then be potentially used by future multimedia sessions. Moreover, in [Cou*etal*95] additional parameters (such as commitment required - guaranteed or best-effort -, delivery type - isochronous or workahead -, and session priority) which could serve as session management parameters are proposed.

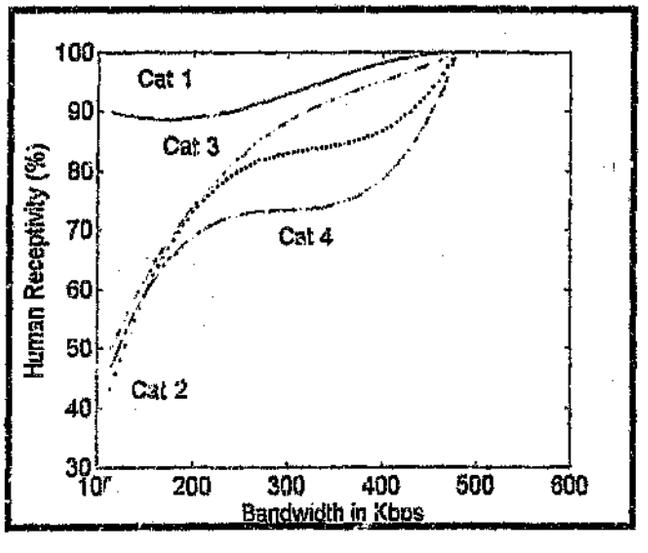

Figure 5 VCS curves : Categories 1 - 4

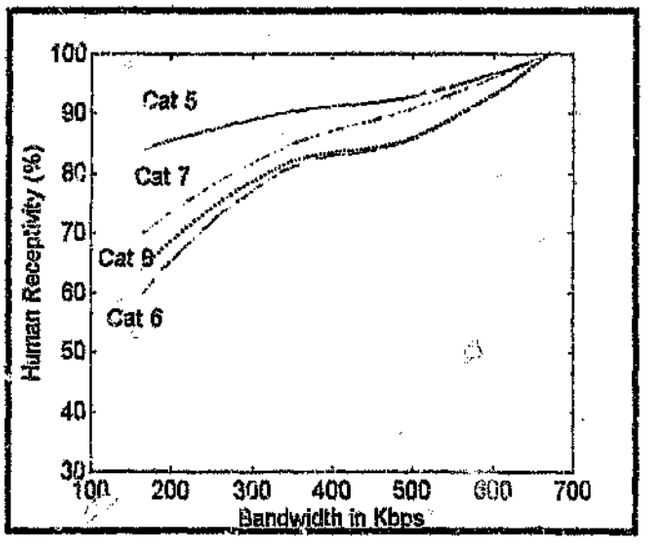

Figure 6 VCS curves : Categories 5 - 8

We use human receptivity as the main criteria of user acceptable QoS, and the afore-mentioned curves to obtain an indication of the bandwidth needed for the session. Furthermore, if so wished, the user can also modify other QoS parameters; if not, recommended values will be used.

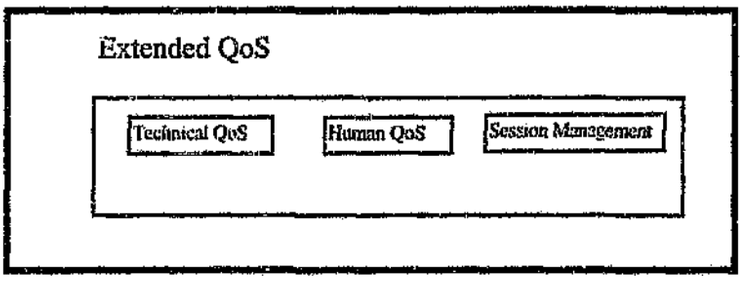

Figure 7 Extended QoS Components

In our research we have investigated the incorporation of an *extended QoS* coupled with a pricing policy in a call admission protocol for distributed multimedia over ATM networks. The extended QoS is made up of technical QoS, human QoS and our upgraded set of session management parameters (we have added the VCS scheme to the parameters proposed in [Cou*etal95*]). This represents our vision of QoS which we have implemented in the design of the protocol.

3.2. Session Characterisation

The developed protocol distinguishes two main classes of multimedia traffic : *guaranteed* and *best-effort*. Sessions having the former characteristic are accepted only if their desired QoS can be guaranteed throughout their duration; best-effort multimedia sessions, on the other hand, express their desired QoS through a range bounded by minimum and maximum desired values. When accepted, the QoS of such sessions will fluctuate between these two bounding values, although every effort will be made by the protocol to try and run these sessions at their maximum desired QoS. It is here that the dynamic nature of the protocol comes into play in that best-effort sessions' QoS varies in response to changes in network traffic and load.

We use human receptivity as the main criteria of user acceptable QoS, and the afore-mentioned curves to obtain an indication of the bandwidth needed for the session. Furthermore, if so wished, the user can also modify other QoS parameters; if not, recommended values will be used.

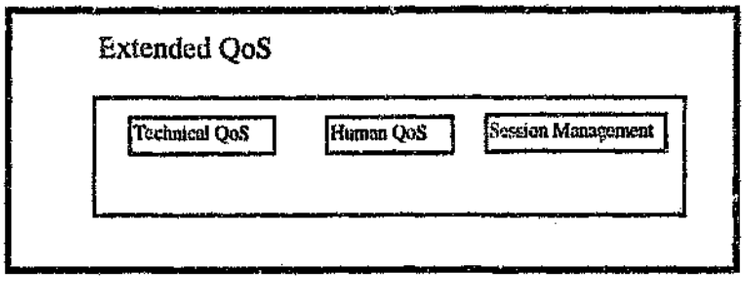

Figure 7 Extended QoS Components

In our research we have investigated the incorporation of an *extended QoS* coupled with a pricing policy in a call admission protocol for distributed multimedia over ATM networks. The extended QoS is made up of technical QoS, human QoS and our upgraded set of session management parameters (we have added the VCS scheme to the parameters proposed in [Cou*etal95*]). This represents our vision of QoS which we have implemented in the design of the protocol.

3.2. Session Characterisation

The developed protocol distinguishes two main classes of multimedia traffic : *guaranteed* and *best-effort*. Sessions having the former characteristic are accepted only if their desired QoS can be guaranteed throughout their duration; best-effort multimedia sessions, on the other hand, express their desired QoS through a range bounded by minimum and maximum desired values. When accepted, the QoS of such sessions will fluctuate between these two bounding values, although every effort will be made by the protocol to try and run these sessions at their maximum desired QoS. It is here that the dynamic nature of the protocol comes into play in that best-effort sessions' QoS varies in response to changes in network traffic and load.

Accordingly, a multimedia session is characterised by its :

- *identifier number*;
- *type* (guaranteed or best-effort);
- *desired QoS* (expressed either as a singular value or as a range, according to the session's type);
- *current QoS level* (expressed in terms of the corresponding bandwidth currently being utilised by the session);
- *priority* (a number between 1 and 5, with 1 being the highest and 5 the lowest priority level);
- *category* (a number between 1 and 8 corresponding to the respective VCS);
- *source* and *destination nodes*;
- *duration* (expressed in seconds), and
- *available budget* (when pricing constraints are enforced)

3.3. QoS Management Strategies

The QoS management strategies of the protocol consist of two components : an *interval division* strategy and a *QoS negotiation* strategy. Both strategies are closely linked to the requested *QoS interval* of best-effort sessions, as specified by the sessions' minimum and maximum acceptable QoS. Thus, before the actual negotiation begins, the protocol divides the requested QoS interval of an incoming best-effort session into five adjacent sub-intervals with six corresponding interval bounds. The interval division strategy gives a modality to obtain the sub-intervals, while the QoS negotiation strategy specifies the number of levels a best-effort session must drop its QoS at each stage of the negotiation process.

There are two interval division strategies, both based on the VCS curves. Accordingly, if the first strategy is used, the required QoS interval is split into five equal sub-intervals and the corresponding bandwidth values are obtained from the curve in question. This strategy attempts to exploit the asymptotic property of VCS curves : the maximum required QoS is successively decremented in equal amounts in the hope that the corresponding reduction in bandwidth is, in relative terms, much larger. The second strategy splits the QoS interval in three equal parts; it then splits the bandwidth interval (corresponding to the original QoS interval) also in three equal parts, with the five final sub-intervals being the intersection of these six along the bandwidth axis. This is a hybrid strategy which, while exploiting the asymptotic property of VCS curves (through the equal division along the QoS axis), also attempts to deal with any portions of the VCS curves that might not obey the asymptotic property (by dividing the bandwidth interval in equal parts and thus avoiding the narrower bandwidth subintervals that would have been obtained had the first bandwidth division strategy been used).

Let a QoS sub-interval be called "higher" than another if the first has a greater upper bound. Bearing in mind that QoS sub-intervals are disjoint in our case, we have assumed all along the design of QoS negotiation strategies that the number of QoS sub-intervals is equal to the number of different session priorities. An intuitive explanation behind this choice would be that we would desire that high-priority best-effort sessions have their average QoS situated somewhere on the highest subinterval, that second highest priority best-effort sessions have their average QoS situated on the second highest subinterval, and so on, till lowest priority sessions have their average QoS situated on the lowest QoS subinterval. Accordingly, the following principles guided us when we formulated the QoS negotiation strategies :

- only best-effort sessions are susceptible to having their maximum required QoS reduced;
- the QoS negotiation strategies will be *multi-stage* processes at the end of which all best-effort sessions are guaranteed to have reached their lowest required QoS;

- during each stage of a QoS negotiation strategy, best-effort sessions will lower their QoS according to their priority - thus, lower-order priority best-effort sessions will reduce their QoS before higher-order priority best-effort sessions, and by a number of QoS levels at least as great as the number by which higher-order priority sessions reduce their current QoS;

- as the protocol has to meet real-time requirements, it is desired for a QoS negotiation strategy to have as few stages as possible;

- consequently, if there is scope for a best-effort session to lower its desired QoS (i.e. the session has not reached its desired minimum level yet), this will always be done at an earlier rather than a later stage;

- a best-effort session will reduce its QoS in steps which are larger initially, but which in subsequent stages will grow progressively smaller, till the lowest requested QoS level is reached, and

- the number of session priority and QoS levels are equal.

Formally, a QoS negotiation strategy will be denoted by a sequence of *l-tuples*, where l is the number of session priority levels defined :

$$(n_{11},n_{21},\ldots,n_{l1}) - (n_{12},n_{22},\ldots,n_{l2}) - \ldots - (n_{1p},n_{2p},\ldots,n_{lp}) \qquad (6)$$

Here :

- each l-tuple represents a *stage* of the QoS negotiation strategy;
- $n_{ij} \in \mathbf{N}, 0 \le n_{ij} \le l, i, j \in \mathbf{N}, 1 \le i \le l, 1 \le j \le p$;
- n_{ij} represents the number of QoS levels that a session with priority l+1-i will drop at stage j of the negotiation process;
- p (the number of stages) is the least positive integer such that :

$$\sum_{j=1}^{p} n_{ij} = l, \qquad \forall i, 1 \le i \le l \qquad (7)$$

where l is the number of QoS levels - this ties in strongly with our strategy-design principles in that we have as few negotiations as possible but are guaranteed that, after the last negotiation stage, all best-effort sessions have lowered their QoS to the minimum level possible (a session cannot lower its QoS by more than the available number of levels, l; furthermore, if a session is currently on QoS level m and negotiations dictate that it should decrease its current level by n levels, the session will simply switch to its lowest level if $m - n \leq 0$);

- $n_{ji} \geq n_{ki}$ for $j \geq k$ and $n_{ji} > 0$, $\forall i$, $1 \leq i \leq l$ - this again conforms to our design strategy, in that, within the same negotiation stage, it is sessions with lower priority that reduce their QoS levels by a greater amount;
- $n_{ij} \geq n_{ik}$ for $j \geq k$, $\forall i$, $1 \leq i \leq l$ - which, for any particular category of best-effort sessions, amounts to a coarser decrease in QoS levels in the initial stages of negotiation, followed by refinements in the QoS-reduction level in subsequent stages - moreover, this condition, coupled with the relation in (7), ensures that if a best-effort session has, at any particular stage of the QoS negotiation, not reached its lowest level, then that particular session's QoS will be decreased at that stage;
- lastly, we mention the fact that, notwithstanding our last strategy design principle, it does not make sense for the total number of different priorities to be greater than the total number of QoS levels - if this is the case, then the relation $n_{ji} \geq n_{ki}$ (made two points previously) used with strict inequality, would ensure that, in any particular negotiation stage, high priority best-effort sessions with priority immediately greater than the number of QoS levels would only start decreasing their QoS levels after all other lower-priority sessions have reached their lowest QoS, which is contrary to our objective of having a best-effort session always decrease its QoS level if there is scope to do so.

Seven QoS negotiation strategies have been tested in the context of the protocol. If the first strategy is used, the lowest priority sessions drop their QoS by three levels; if this does not result in sufficient bandwidth being freed, then sessions with the next lowest priority drop their QoS by three levels, and so on, till sessions with highest priority drop

their current QoS by three levels in an effort to accommodate the new session. From the pool of sessions bearing equal priority, it is those sessions that have been active the longest that drop their QoS first. Furthermore, an incoming session will be accepted as soon as enough bandwidth becomes available, irrespective of whether or not all sessions of equal priority have, for the current negotiation stage, reduced their QoS. In the next stage of the negotiation process, all sessions drop their QoS by two levels, in ascending order of priority, as before, until either the new call is accepted, or rejected, in the case of which all sessions have lowered their QoS to the minimum level (there are only six available QoS levels so a reduction of five levels guarantees that the lowest possible level has been reached). Using formal notation, this approach will be referred to as *(3,3,3,3,3) - (2,2,2,2,2)*. Using the same notation the other six strategies are :

(i) *(4,4,4,4,4) - (1,1,1,1,1)* : this is a similar strategy to the one just described, the only difference being that the current one uses a much coarser QoS negotiation level initially;

(ii) *(2,2,2,2,2) - (2,2,2,2,2) - (1,1,1,1,1)* : this is a variant of the previous strategy, in which the initial negotiation stage has been broken down into two identical stages ;

(iii) *(1,1,1,1,1) - (1,1,1,1,1) - (1,1,1,1,1) - (1,1,1,1,1) - (1,1,1,1,1)* : this is the finest-grained negotiation approach, where, at each stage, each best-effort session reduces its QoS by a single level;

(iv) *(5,5,5,5,5)* - this is the coarsest-grained approach, where best-effort sessions try to free up as much bandwidth as possible by dropping their QoS to the minimum level in one go;

(v) *(5,4,3,2,1) - (0,1,1,1,1) - (0,0,1,1,1) - (0,0,0,1,1) - (0,0,0,0,1)* - this is a hybrid of the previous two strategies, and

(vi) *(3,3,3,2,2) - (2,2,2,1,1) - (0,0,0,1,1) - (0,0,0,1,1)* : this is our only four stage QoS negotiation strategy, in which lower-priority best effort sessions reduce their QoS

according to the first strategy defined, while higher-priority (the top two priority levels) best-effort sessions use the only negotiation sequence possible for a four stage strategy.

In the context of traffic management of our protocol, we have also experimented with a tuneable threshold for guaranteed traffic. This threshold essentially reserves a fraction of the total carrying capacity for guaranteed traffic, thus blocking access of best-effort sessions to a portion of the bandwidth. Its usefulness would come into play when traffic management requirements dictate that a certain fraction of overall sessions need to run at their maximum requested QoS.

3.4. Protocol Description

The protocol that has been developed by us sits atop the ATM Adaptation Layer 5 (AAL5), the highest such layer in the ATM protocol reference model. We do not use LAN protocol emulation because ATM QoS parameters are hidden at this level, and are firstly accessible only at the AAL5 level. It runs in an ATM switching environment and has been developed within the framework of the client-server paradigm.

An ATM switching environment has been chosen because it is ideally suited for the multimedia client/server scenario, as it increases performance by facilitating LAN segmentation. ATM switching also provides the scalable bandwidth (through the addition of more switched ports) and flexibility needed for future growth, favouring performance and technology migration, while at the same time preserving the considerable current desktop investment. Furthermore, such an environment is attractive from another viewpoint, namely that it facilitates the creation and management of *virtual LANs* (a logical group of users, independent of their physical location). Here, network performance, security concerns as well as workgroup modifications can all be addressed and managed by software rather than by actual hardwiring, resulting in much simplified processes.

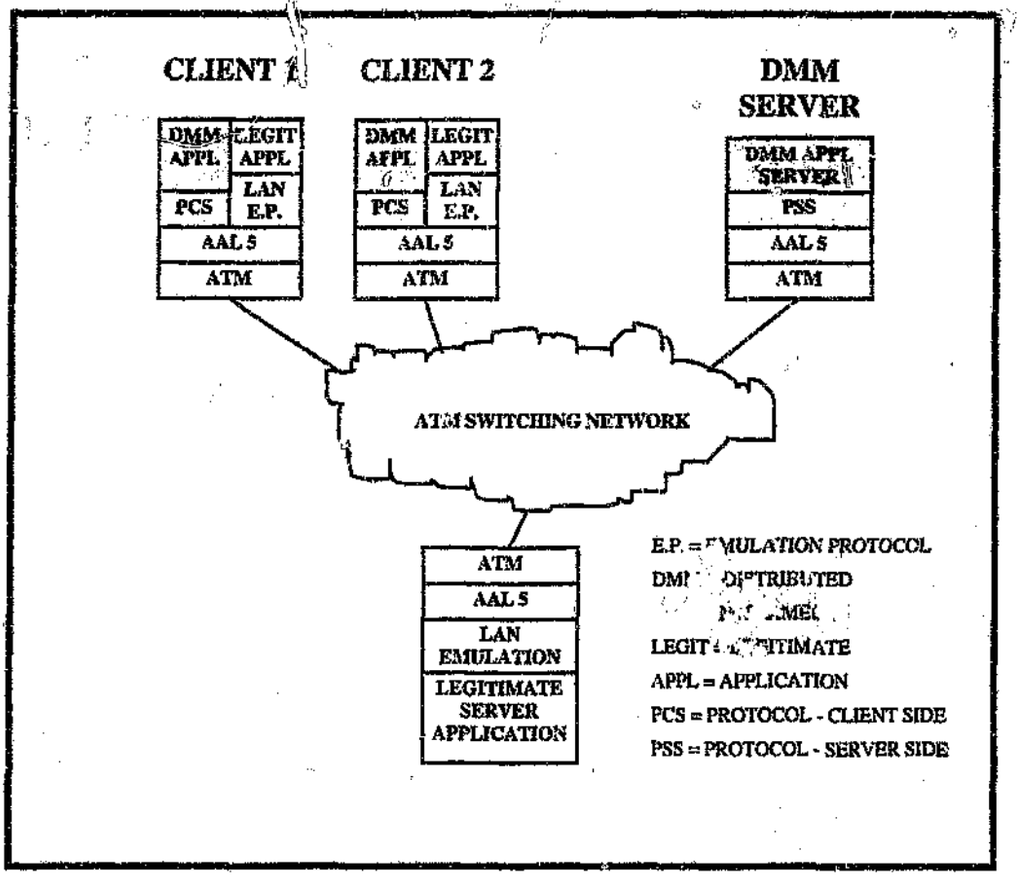

Figure 8 Location of our protocol in an ATM context

The protocol relies heavily on the centralisation of complexity and the distribution of simplicity. As such, the client side interfaces and interacts with the user and communicates with the server for the establishment of the final values of the multimedia session's QoS parameters. If the session is accepted, the client then receives the multimedia data and is responsible for its visualisation.

The server side of the protocol, on the other hand, is responsible for session management. To this end, it maintains a *Network Information Base*, a *Session QoS Base* and a *Session Knowledge Base*.

The Network Information Base holds information regarding the topology of the network. As such it contains details regarding the number and types of switches and links, as well as client and server workstations. Furthermore, it details the existing inter-

connections among the nodes of the network. At any particular moment in time, the Network Information Base also records information pertinent to the dynamic functioning of the protocol. This is comprised of current available bandwidth along each link, available switch ports, as well as the current state (up or down) of links and switches in the network.

The Session QoS Base of a server contains information concerning the characteristics of sessions (as given in Section 3.2) which are currently inter-acting with the respective server. The Session QoS Base also contains, in equivalent tabular form, the VCS curves corresponding to the eight categories of multimedia data considered in our research.

The Session Knowledge Base is built on previous experience and contains suggestions regarding protocol negotiation and pricing strategies, as well as recommended values for session QoS parameters, all on the basis of which QoS negotiation with the client is handled. If this negotiation is successful, the server then selects the appropriately characterised multimedia data from its library of copies to the client. Furthermore, during the session itself, the server can receive user interaction from the client side (requests such as repeat, next/previous, stop, activate screen area, pull menus down) and acts accordingly.

The protocol consists of three main parts :

- *route establishment*;
- *call admission*, and
- *renegotiation*.

Route establishment and call admission take place when deciding whether or not to admit a new multimedia session, while renegotiation is a process that takes place when a session currently being accommodated by the network finishes.

When a new multimedia session must be accommodated by the ATM network, the protocol firstly tries to establish a route between the client and server nodes based on Dijkstra's Shortest Path Algorithm [Baa92]. Accordingly, the server consults its Network Information Base and applies the algorithm in order to find out the shortest route between client and server. The distance metric used is the number of hops between nodes. This particular choice of metric ties in strongly with the pricing policy used (see Section 3.5. below), where the session would pay a smaller start-up cost if less fixed resources (ports and switches) are reserved for the session along its route. If the available capacity on a link is less than the session's required bandwidth, that particular link is assigned (dynamically) an infinite weight, otherwise it has, by convention, a weight equal to unity. The protocol then tries to accommodate the new session, applying the call admission process along the shortest route found between the client and the server. Obviously, if there is no route between the client and the server, the session is blocked from the start.

It must be mentioned that we considered finding all routes between the client and server workstations, ordering them according to increasing length, and then have the protocol apply the call admission process successively on these routes, till either the session was admitted, or the protocol exhausted all routes. However, the process of finding all routes between two nodes of a graph is NP-complete, as it contains the problem of finding the longest path between two vertices, which is known to be NP-complete [GarJoh79]. Since the protocol will, in many cases, have to meet stringent real-time requirements, finding all routes between the client and server workstations was thus not considered to be a feasible solution. In contrast, Dijkstra's shortest path algorithm is polynomial [Baa92], being $O(n^2)$ in the worst case (n is the number of nodes in the graph), and can consequently be used in a real-time environment.

The call admission process of the protocol determines if the session's maximum QoS can be provided for. If not, the protocol then consults the Session QoS Base, and all currently running best-effort QoS sessions (plus the incoming session, if it is also best-effort) will drop their current QoS requirements according to the multi-stage negotiation strategy in use. If, at any stage of the negotiation, the best-effort sessions have dropped

their QoS requirements sufficiently for the new session to be accepted then the negotiation stops and network parameters are correspondingly updated to reflect the new situation (an extra session). Conversely, the session is blocked if the end of the negotiation process is reached and network resources cannot accommodate the new session.

Lastly, when a session finishes, a process of bandwidth renegotiation takes place. During this stage, the protocol again consults the Session QoS Base and attempts to increase the current QoS levels of best-effort sessions, profiting from the relinquished resources made available by the just-finished session. At the end of this process, the Session QoS Base is correspondingly updated.

3.5. Pricing

There are several reasons why pricing is difficult in networks. The first is that the perishable nature of the service offered by a computer network makes it very difficult to manage. Bandwidth cannot be stored for later use. A cell in an ATM network occupies a real-time slot and, if not filled with data, cannot be used at a later stage. If the cells cannot be 'sold' in time, they are lost and the service does not run at full capacity. If it were possible to sell all cells, profits would be maximised and prices possibly dropped. Secondly, the unpredictability and burstiness of computer traffic makes the situation difficult to manage. Whereas the service provider has a predictable (ignoring down-time) and steady supply of transport cells available on a link, the demands of the users are highly unpredictable and bursty, which make it very difficult to run the network at full capacity. Lastly, an even greater problem is the service provider's difficulty to meet QoS requirements, while still trying to maximise the number of cells sold. This is especially so in routers and switches, where large numbers of traffic flows intersect [Smu94].

In our protocol, we follow the approach of Parris, Keshav and Ferrari and characterise a user by the triplet *(QoS_class, connection_duration, available_budget)* [ParKesFer92]. If there are enough network resources to accommodate the session, the network will respond with either an *(Accept, Price)* or *Reject* message. The *Price*

parameter represents an estimate of the cost of the connection as computed by the protocol; the call will be accepted if :

$$Price \leq (1+thr)* available\_budget \qquad (8)$$

Here *thr* is a tuneable threshold that can be modified according to traffic requirements. If the cost of the connection exceeds the available budget, but is accepted, then the duration of the connection will be proportionately reduced. If condition (8) is not satisfied, then the call will be rejected.

Strongly linked with the design of the protocol is the elaboration of a pricing strategy for estimating the cost of multimedia sessions set up with the protocol. The pricing strategy formulated by us is:

$$Cost = \left(\sum_{links} \frac{bw\_reqd}{total\_bw}\right) * \left(\sum_{links} \frac{bw\_reqd}{avail\_bw}\right) * duration + Fixed\_Cost \qquad (9)$$

In analogy with (4), the first term between brackets represents the QoS cost, expressed as the fraction of the total bandwidth along the route from start to destination node that the session's required bandwidth represents; the second term tries to model the impact the network load (time of day in (4)) has on a new session, by expressing the required bandwidth as a fraction of the available bandwidth along the session's route, while the duration term has the same significance in both (4) and (9). Lastly, the *Fixed_Cost* component reflects leasing and maintenance costs of switch resources such as ports and buffers.

It must be mentioned that the pricing policy of (9) incorporates aspects of both peak load pricing - through the second bracketed term in (9) - as well as set-up pricing - through the *Fixed_Cost* component. Thus, it attempts to combine the reported [ParKesFer92] benefits of both pricing schemes : an increase in revenue and better resource utilisation due to the effect of peak load pricing, coupled with a decrease in call blocking probability caused by the set-up pricing component.

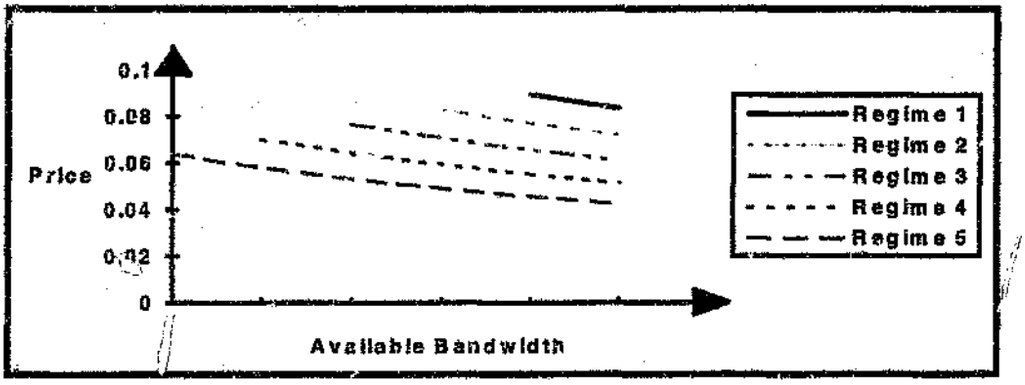

Figure 9 Pricing Policy Graph

Since the bandwidth required by a session, determined by the minimum bandwidth available on the links along the route from the session's source to destination, five *price regimes* (curves) are distinguished which correspond to the five bandwidth intervals which could possibly contain the minimum available bandwidth. The first four of these intervals correspond to the four bandwidth sub-intervals obtained through the particular bandwidth division strategy in use, while the last interval represents the complement of the first four relative to the bandwidth interval having as lower and upper bounds, respectively, the minimum requested bandwidth of the session and the total capacity of the link with the minimum available bandwidth. Remarking that, according to the implemented QoS negotiation strategies, the required bandwidth (*bw_reqd*) parameter of (9) does not vary continuously, but discretely, taking a total of six different values, each of the pricing policy regimes also correspond to the bandwidth currently being used by a session, with the first regime corresponding to the two highest bandwidth levels required, the second regime corresponding to the third highest requested bandwidth and so on, up to the last regime which corresponds to the lowest requested bandwidth of a session. Thus, each regime is only defined for available bandwidths greater than the currently requested one. Moreover, during the life-time of a session, the regime which gives the session cost will vary dynamically in tune with the session's required bandwidth.

There are two further remarks which must be made as regards the choice of pricing policy. The first is that the price paid for a particular requested bandwidth (regime)

decreases as more bandwidth becomes available on the links. This is evident from the pricing equation of (9), where the price paid is inversely proportional to the amount of free bandwidth on the links. The second is that, for any given available link bandwidth, the price paid along any particular regime is lower than that paid on the regime corresponding to the immediately higher requested bandwidth. This is again evident from (9), where the price paid is directly proportional to the bandwidth used by a session. Both remarks can also be observed graphically in Figure 9.

The pricing policy of (9) will now be studied in more detail. Let us denote by req_i (i = 1..6) the six requested bandwidth levels of the protocol. To do an analysis that is independent of a session's duration, it can be assumed without any loss of generality that the calculated price as well as the available budget of a session are given *per unit of time*. The pricing regime corresponding to each such requested bandwidth is then given by :

$$Cost(i,avail\_bw) = \left(\sum_{links} \frac{req_i}{total\_bw}\right) * \left(\sum_{links} \frac{req_i}{avail\_bw}\right) + Fixed\_Cost \qquad (10)$$

since the start-up cost is independent of the duration of the call.

We are interested in the highest and lowest prices that could possibly be paid on a regime. The highest cost paid on a particular regime occurs when the available bandwidth is the same across all links along a session's route and equal to the requested bandwidth, req_i, for that regime. Thus, the highest cost paid on a regime is given by :

$$Cost(i,avail\_bw)_{highest} = \left(\sum_{links} \frac{req_i}{total\_bw}\right) * \left(\sum_{links} \frac{req_i}{req_i}\right) + Fixed\_Cost \qquad (11)$$

decreases as more bandwidth becomes available on the links. This is evident from the pricing equation of (9), where the price paid is inversely proportional to the amount of free bandwidth on the links. The second is that, for any given available link bandwidth, the price paid along any particular regime is lower than that paid on the regime corresponding to the immediately higher requested bandwidth. This is again evident from (9), where the price paid is directly proportional to the bandwidth used by a session. Both remarks can also be observed graphically in Figure 9.

The pricing policy of (9) will now be studied in more detail. Let us denote by req_i (i = 1..6) the six requested bandwidth levels of the protocol. To do an analysis that is independent of a session's duration, it can be assumed without any loss of generality that the calculated price as well as the available budget of a session are given *per unit of time*. The pricing regime corresponding to each such requested bandwidth is then given by :

$$Cost(i,avail\_bw) = \left(\sum_{links} \frac{req_i}{total\_bw}\right) * \left(\sum_{links} \frac{req_i}{avail\_bw}\right) + Fixed\_Cost \qquad (10)$$

since the start-up cost is independent of the duration of the call.

We are interested in the highest and lowest prices that could possibly be paid on a regime. The highest cost paid on a particular regime occurs when the available bandwidth is the same across all links along a session's route and equal to the requested bandwidth, req_i, for that regime. Thus, the highest cost paid on a regime is given by :

$$Cost(i,avail\_bw)_{highest} = \left(\sum_{links} \frac{req_i}{total\_bw}\right) * \left(\sum_{links} \frac{req_i}{req_i}\right) + Fixed\_Cost \qquad (11)$$

or, equivalently, after some simple manipulation, by :

$$Cost(i,avail\_bw)_{highest} = (no\_of\_links)*(req_i)*\left(\sum_{links}\frac{1}{total\_bw}\right)+Fixed\_Cost \quad (12)$$

where *no_of_links* is the number of links (hops) along the session's route

Since we are especially interested in the usage of a pricing policy to implement congestion control, for the purposes of our analysis we shall assume that the maximum available bandwidth on links can never be higher than req_6 (i.e. the highest bandwidth level requested by a session). Thus, the lowest cost paid on a regime corresponding to req_i occurs when a single link has an available bandwidth of req_{i+1}, while all others have an available bandwidth of req_6. We must make the remark that this scenario is a limiting case one since the minimum available bandwidth on the regime corresponding to req_i is always strictly less than req_{i+1} and can only equal req_{i+1} if limit behaviour is taken into consideration (an actual minimum available bandwidth of req_{i+1} would imply that the session cost should be given by the regime corresponding to this bandwidth, and not by the one corresponding to req_i). The lowest cost paid on a regime corresponding to req_i is therefore given by :

$$Cost(i,avail\_bw)_{lowest} = (req_i)^2*\left(\sum_{links}\frac{1}{total\_bw}\right)*\left(\sum_{links}\frac{1}{avail\_bw}\right)+Fixed\_Cost \quad (13)$$

which, bearing in mind the considerations stated above, becomes :

$$Cost(i,avail\_bw)_{lowest} = (req_i)^2*\left(\sum_{links}\frac{1}{total\_bw}\right)*\left(\frac{1}{req_{i+1}}+\frac{no\_of\_links-1}{req_6}\right)+Fixed\_Cost \quad (14)$$

It is our intention to study the effect that the choice of bandwidth division scheme (i.e. the form that req_i might actually take) has on the pricing policy. Thus, we will examine comparative prices on two consecutive bandwidth regimes. To this end we are interested in the difference between the lowest price paid on a regime with a higher

bandwidth and the highest price paid on the regime corresponding to the immediately lower requested bandwidth. This difference is given by :

$$Cost(i+1,avail\_bw)_{lowest} - Cost(i,avail\_bw)_{highest} =$$

$$(req_{i+1})^2 * \left(\sum_{links} \frac{1}{total\_bw}\right) * \left(\frac{1}{req_{i+2}} + \frac{no\_of\_links - 1}{req_6}\right) -$$

$$(no\_of\_links) * (req_i) * \left(\sum_{links} \frac{1}{total\_bw}\right) \qquad (15)$$

After factoring out common terms (15) becomes :

$$\left(\sum_{links} \frac{1}{total\_bw}\right) * \left[(no\_of\_links - 1) * \left(\frac{req_{i+1}^2}{req_6} - req_i\right) + \left(\frac{req_{i+1}^2}{req_{i+2}} - req_i\right)\right] \qquad (16)$$

Since the first multiplicative term of (16) is a positive number, the sign of the difference in (15) is given by the term between square brackets in (16) :

$$(no\_of\_links - 1) * \left(\frac{req_{i+1}^2}{req_6} - req_i\right) + \left(\frac{req_{i+1}^2}{req_{i+2}} - req_i\right) \qquad (17)$$

We are now in a position to comment on the impact that the bandwidth division strategy has on the pricing policy. If the expression given by (17) is a positive number, then this would imply that all prices paid on the regime corresponding to a requested bandwidth of req_{i+1} are higher than the prices paid on the regime corresponding to a requested bandwidth of req_i, thus encouraging low-budget sessions to request less bandwidth.

Conversely, if the bandwidth division strategy is such that the expression given by (17) is a negative number, then the highest price paid on the regime corresponding to a requested bandwidth of req_i is higher than the lowest price paid on the regime corresponding to a requested bandwidth of req_{i+1}. This highlights the strong effect that the

loading factor has on the pricing policy in that a session, even if it has, due to lack of available bandwidth, dropped its bandwidth requirements, may still have to pay a higher price than if it were requesting a higher bandwidth (which cannot be granted because of the heavy network load). Consequently, the pricing policy is expected to smooth out peaks in network utilisation by discriminating against sessions with a low budget.

3.6. Protocol Knowledge Base

A knowledge base is *"a collection of simple facts and general rules representing some universe of discourse"* [Fro87, pp.3] and constitutes the central component of *knowledge-based systems* (or *agents*). The other component of such systems is an inference unit, on the basis of which the agent reaches its conclusions and provides recommendations. It is now a generally accepted fact in the artificial intelligence (AI) community that the real power of knowledge-based agents comes from the knowledge they possess rather than the particular inference schemes and other formalisms that they might employ. There has consequently been much work done on the application of knowledge-based systems in various domains such as vision, learning, medical diagnosis, general problem solving, and natural language understanding [RusNor95].

The designed protocol contains and can consult a Session Knowledge Base (SKB), the main purpose of which is to provide, if needed, recommendations regarding QoS negotiation and pricing strategies according to current network configuration (topology and traffic load). Although the emphasis of the protocol design lies not in the SKB, the decision to create the SKB has been motivated by pragmatic considerations, in order to provide a foundation for a practical implementation of the protocol.

Knowledge gathered through the protocol's simulation is encoded in the SKB by the way of production rules of the form :

if C *then* A (18)

where C is some condition which has to be satisfied by a multimedia session in order for the rule to be applied and the action A taken. Furthermore, the information in the SKB can be refined and added onto, as new knowledge becomes available and is fed back.

Finally, it must be mentioned that knowledge-based systems do have a big disadvantage in their relatively slow execution time. As the designed protocol has to meet strict real-time requirements, it was decided to overcome this drawback by insisting that the size of the created SKB be small, holding no more than a hundred rules. The full details regarding the construction and composition of the SKB are given in Chapter 5.

4. PROTOCOL SIMULATION

Generally speaking, *simulation* is a technique that reproduces actual events and processes under test conditions. In *computer simulation*, a computer is used to numerically evaluate a model and collect data in order to estimate the model's true characteristics [LawKel91].

4.1. Justification and Purposes

The advent of computational power has lead to computer simulation becoming an increasingly important tool in the analysis of communication systems. This development has further been spurred by an increase in both the number and complexity of existing communication networks [LawMcc94].

Protocol performance was examined through simulation (this term shall henceforth be used to actually denote computer simulation). The justification for this is three-fold : firstly, through simulation it was possible to make general observations about the effectiveness of our protocol in delivering QoS guarantees under conditions of statistical multiplexing, with these observations constituting the foundation on which the protocol knowledge base was built; secondly, simulation is highly useful as it allows the fine tuning and measurement of protocol parameters, as well as the estimation and validation of protocol performance, within a reasonable time frame, and without special computational requirements; lastly, simulation methods remain the most flexible (compared with

approaches involving, say, analytic methods or actual hardware implementation) for examining node operation and performance under a variety of conditions [LawKel91].

4.2. Development and Methodology

Since simulation is essentially software, for the purposes of simulation development we have chosen to use the waterfall life cycle paradigm [Som92]. Although there are numerous variations in this model, five stages were actually considered by us : *requirements' analysis and definition* (here, for instance, the actual data to be gathered by the simulation process was determined), *design* (the simulation development environments, the simulation blocks and their associated interfaces, as well as the type of simulation to be used were decided upon at this stage), *coding*, *testing*, and *integration* (here all simulation blocks were combined into one entity, the *simulator*). In the model used in the simulation, reviews and testing are performed after each stage and, when faults are found, they are referred back to the requirements' analysis and definition stage to determine the source of the fault. Thus, requirements were rechecked to ensure their correctness, then the design is checked, and so on, up to the very last stage, integration.

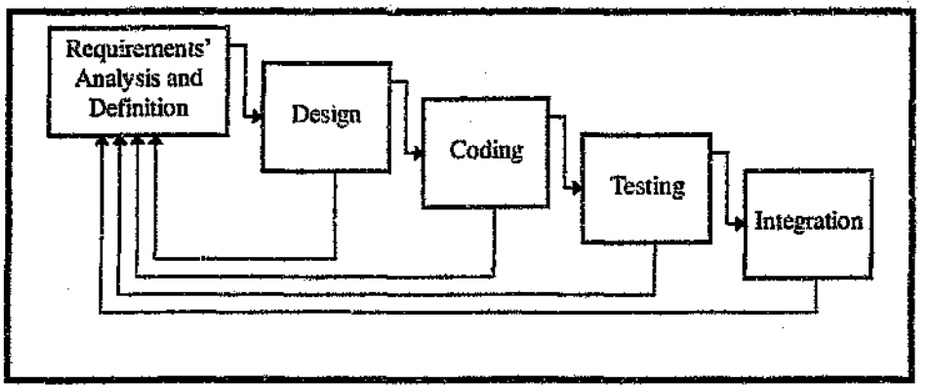

Figure 10 The waterfall model

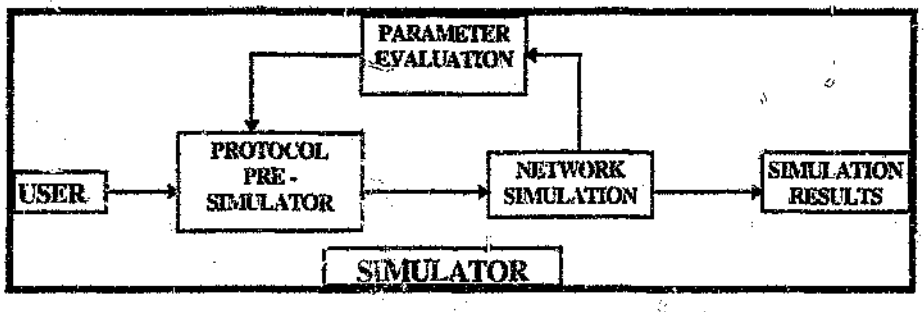

Figure 8 Project Block Diagram

The simulator developed for evaluating the designed protocol consists of:

- a *protocol pre-simulator block* which simulates the extended QoS negotiation strategies implemented in the protocol;
- a *network simulation block* which simulates the network environment (ATM layer protocol, network topology and ATM switches), and
- a *parameter evaluation block* which takes a subset of the network simulation results and builds the protocol SKB.

Protocol performance was examined on a total of eight different topologies. These topologies were obtained from two basic configurations by varying the number of clients (for our purposes, a client is a *non-switched network component* such as a mainframe or a group of networked workstations) and the switch types. Accordingly, Topologies 1-4 and 5-8 use ATM switches with 2.5 Gigabits per second (Gbps) and 1.2 Gbps switching

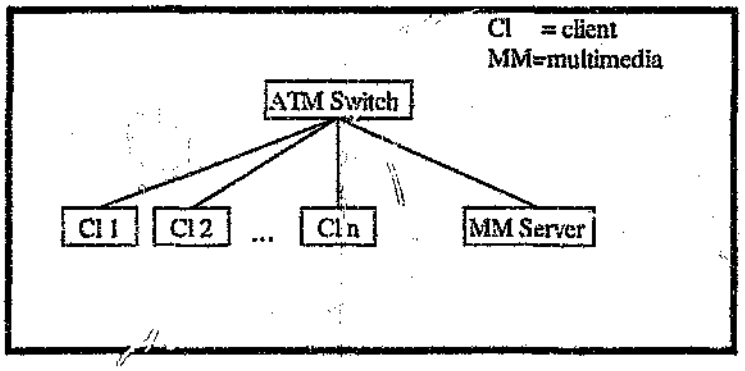

Figure 11 Topologies 1,4,5,8

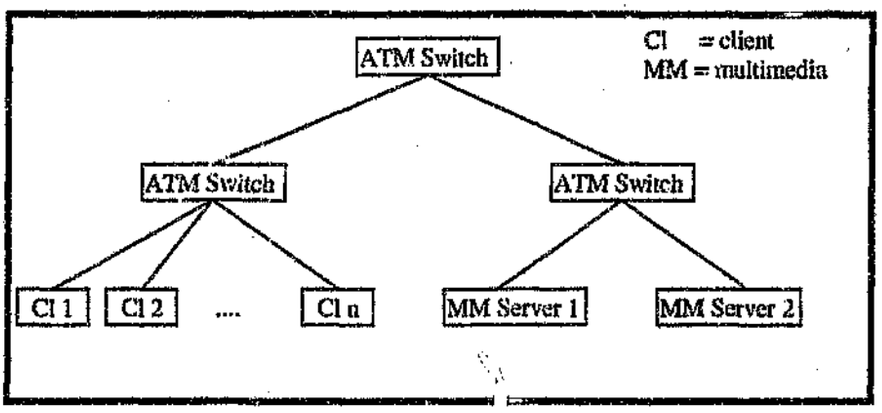

Figure 12 Topologies 2,3,5,6

capacities respectively; both switches have 256K output buffers. The number of clients varies according to the switch and line types used. For example, if a 2.5 Gbps switch is used and the lines have a 622 Mbps carrying capacity, there are four clients in the configuration. Thus, there are 4 clients in Topologies 1 and 2 (622 Mbps lines) and 25 in Topologies 3 and 4 (100 Mbps Fast Ethernet lines); in Topologies 5 and 6 there are 8 clients (155 Mbps lines), while in 7 and 8 there are 12 clients (100 Mbps Fast Ethernet lines).

An extremely congested scenario was simulated in order to test the limiting performance of the protocol. The video resolution of the multimedia sessions was assumed to be 640*480 pixels - as opposed to the 160*120 pixels used in [Apt*etal*95b] to obtain the original 8 VCSs - thus resulting in much higher bandwidth requirements. The duration of a multimedia session is taken from an exponential distribution with a mean of 1200 seconds, but is assumed to be at least 10 seconds and at most an hour long, while the starting time of new multimedia sessions was given by a Poisson distribution with rate $\lambda = 0.4$. A total of 1000 accepted sessions were simulated for each run of the simulation.

Finally, when the effect of pricing on user behaviour and network resource allocation was studied, it was decided to factor out the duration parameter in (4) and (9); thus the available budget parameter used in the simulation represented the price *per unit of time* that the user was prepared to pay. Since, for any accepted multimedia session, the negotiated bandwidth will be less than the available bandwidth, the summation terms of (9)

are all sub-unitary fractions. It was therefore decided to also assume a sub-unitary set-up cost for all the switch resources along the session's path.

4.3. Pre-Simulation

The entire pre-simulation process is event-driven. Each event represents either the start or completion of a multimedia session. When a new multimedia session is created, it is assigned an identifier number, a type, priority, category, duration, an originating client and destination server, as well as a minimum and maximum desired QoS. A minimum of 50% desired QoS has been assumed for all sessions, while for guaranteed sessions only the maximum desired QoS was taken into consideration. When pricing concerns where simulated, a session was further characterised by its available budget. All these characteristics (except the duration) of a session are obtained through the use of a normal random number generator.

After a multimedia session is created, the pre-simulation program first tries to accommodate the session with its maximum desired QoS. If this cannot be done, then a process of QoS negotiation is entered into. If this negotiation is unsuccessful, the blocked sessions' count is correspondingly increased. After the call admittance stage, the pre-simulation clock is increased to reflect the starting time of the next multimedia session. The pre-simulator then checks the *completion times queue* - the only event queue of the program, ordered according to increasing completion time values - to see if any existing multimedia sessions have finished in the meantime. If this is the case, a process of QoS renegotiation is entered into, during which network bandwidth is released. After all completed events have thus been processed, the program terminating condition is tested. If the condition does not hold a new session is created, after which the whole admittance process is re-started; otherwise, all remaining session completion events are processed, the required reports are generated and the pre-simulation ends.

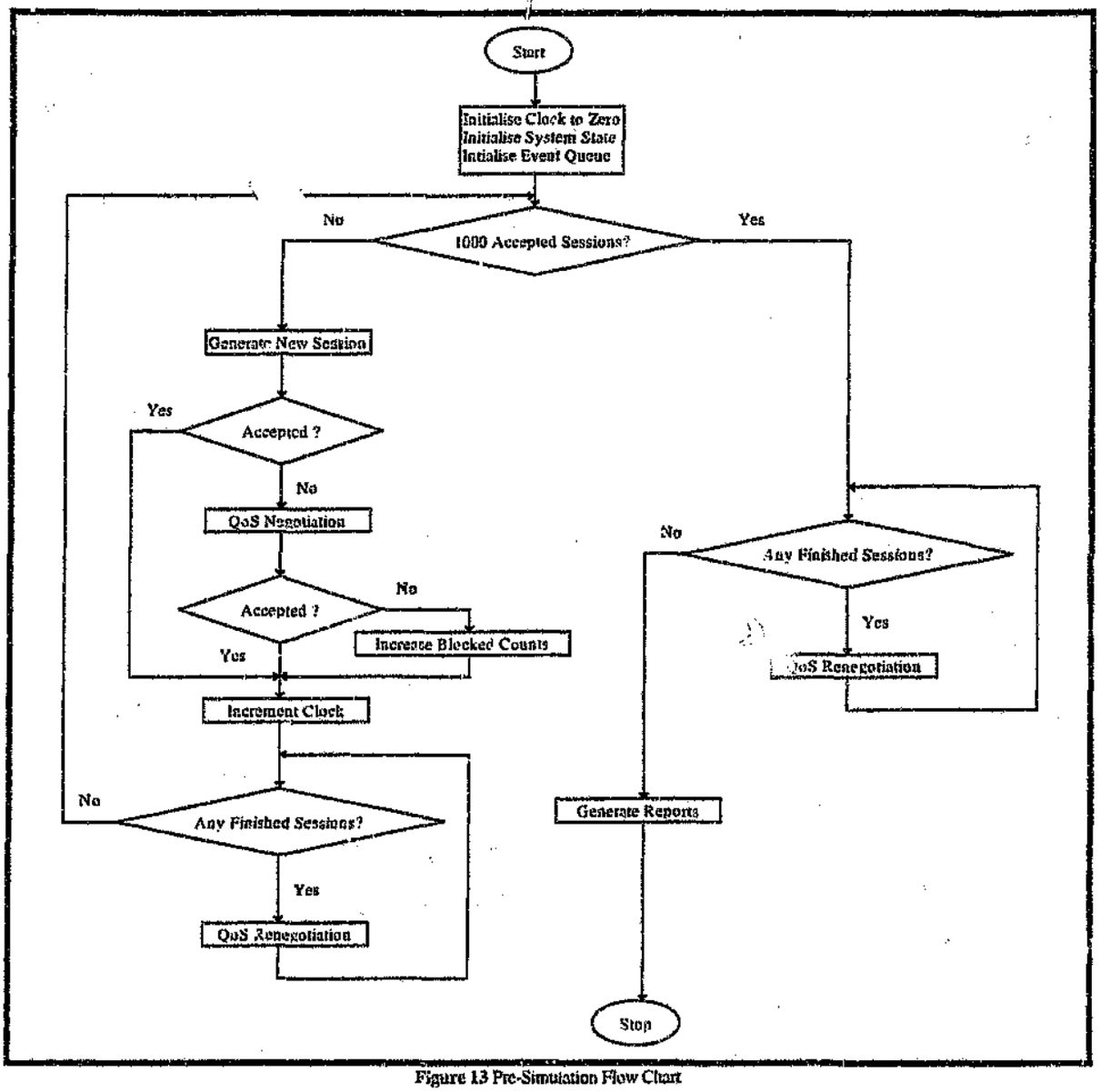

Figure 13 Pre-Simulation Flow Chart

The pre-simulation block gathered statistics on the number of best-effort sessions and of blocked calls. Furthermore, data on the average QoS of best-effort sessions was reported - both globally as well as on the basis of the sessions' priority. The average QoS was expressed in two ways, as a percentage of both the session's minimum and maximum desired QoS. Lastly, the interface between the pre-simulation and the network simulation blocks was accomplished through an ASCII text file generated by the former block. This

file was used as an external traffic input file to *COMNET III* and conformed to a pre-defined format accepted by *COMNET III*.

4.4. ATM Network Simulation

The actual ATM network environment was simulated with the help of the *COMNET III* simulation package. *COMNET III* is a graphical, event-driven, object-oriented simulator for networks ranging from simple LANs to enterprise-wide systems. *COMNET III* is integrated into a simple windowed package which performs all functions of model design, model execution and presentation of results. To build a network simulation model, the user inputs, via a graphical user interface, the topology of the network (by connecting together node and link icons in the display window - see Figure 14 below), the workload placed on the network (this is specified through dialogue windows obtained by double-clicking the workload icons attached to the appropriate network nodes) and the protocols for scheduling applications and routing traffic (again

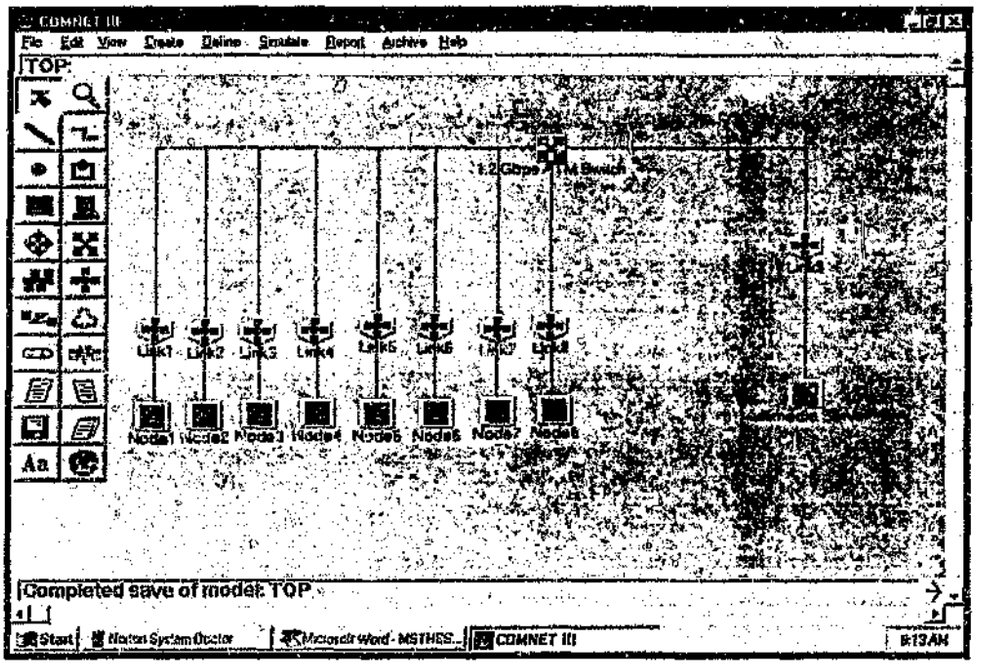

Figure 14 Topology 5 - COMNET III model

specified through dialogue windows obtained by double clicking the model's link icons). Lastly, *COMNET III* produces standard output reports from which the user can obtain an estimate of the expected performance of the real network.

COMNET III was used in order to gather specific, ATM-related data about the execution of the designed protocol. To this end, the various simulated topologies and their parameters (link capacity, ATM switch buffer size), as well as the associated ATM transport protocol were specified with the aid of *COMNET III*'s graphical user interface. Although *COMNET III* does not have ATM as an available library transport protocol, it does have the capability to create one. Accordingly, ATM was defined as a transport protocol in which the maximum packet size and packet overhead are 48 and 5 bytes, respectively, thus ensuring that the ATM cell and the *COMNET III* packet are synonymous.

The actual workload of the network was given by an *external traffic file* generated by the pre-simulator. This file contained, in column form, the identifier, starting time (specified in seconds, relative to the previous session's starting time), destination and source nodes, along with the size (in bytes) of all accepted sessions. The traffic file format - field starting positions, field lengths, what lines of the file to ignore, the time units used, as well as whether the time field of the file contained absolute or relative values - was specified using *COMNET III*'s Traffic File Configuration Form.

Finally, for each tested topology and QoS negotiation scenario, the *COMNET III* simulation gathered statistics on link and buffer utilisation, as well as end-to-end session delay.

5. SIMULATION RESULTS AND IMPLICATIONS

The pre-simulation stage of the protocol was coded in *C* and the program run on a SGI workstation with a MIPS R4000 processor. The *COMNET III* simulation package set up on a 486 DX2/66 machine with 16 MB of memory was used for the realisation of the ATM network simulation stage. For a particular negotiation strategy and topology, 10 iterations of both the pre-simulation and the corresponding simulation were run. All results were then averaged out with a view towards eliminating simulation inconsistencies. Where average QoS results are reported, they represent the average *relative* QoS (reported to the maximum desired QoS) of best-effort sessions. Where results are given on an unitary scale, it is to be assumed that the 0 - 100% interval has been scaled to the unit interval. Lastly, although all our results are reported, we usually illustrate them graphically with findings only from a particular topology and negotiation scenario.

5.1. The Case for QoS Negotiation

In our research we have tried to satisfy two opposing desiderata : to accommodate a session at an average QoS close to its maximal desired value and, at the same time, to accept as many sessions as possible. To see the effect that QoS negotiation has on these goals, three situations were firstly simulated : (i) the first was one where there was no QoS negotiation (all sessions were of guaranteed type); (ii) the second was one in which there was a traffic mix of both guaranteed and best-effort sessions with corresponding QoS

negotiation, and (iii) the last was one in which all sessions were best-effort and therefore subject to QoS negotiation.

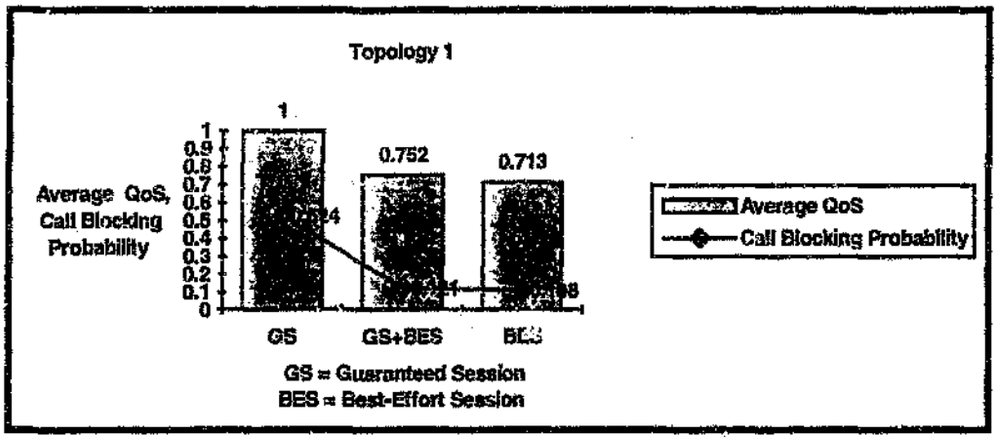

Figure 15 Effect of QoS Negotiation

The results obtained have shown that there is an interesting trade-off between the sessions' average QoS and their call blocking probability. Accordingly, when the traffic consisted of only guaranteed sessions, although, as expected, the required maximal QoS was always met, the call blocking probability was unacceptably high (a figure slightly above 50% in the majority of cases). When the traffic mix comprised both best-effort and guaranteed sessions, although the sessions' average QoS decreased (by roughly 25-30% in all cases), there was a dramatic drop in the call blocking probability (by between 25-40% in the majority of cases, which represents a relative decrease of roughly 50-80%). Finally, when the traffic consisted of only best-effort sessions, the trade-off was further evinced by a decline in both the sessions' average QoS and call blocking probability (a relative drop of approximately 5-10% in the average QoS lead to a relative drop of between 15-23% in the call blocking probability). These results are shown in Figure 15 for the particular case of Topology 1 (the first VCS interval division strategy was used; there was also no traffic threshold employed).

There are a few further interesting points to note here. The first is that, even in the case where all the sessions' QoS is negotiated, there still are sessions that are blocked due to insufficient bandwidth being available. The second is that, in this same case, the average

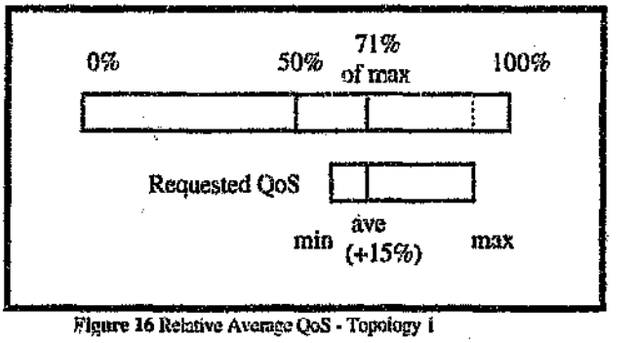

Figure 16 Relative Average QoS - Topology 1

delivered QoS is quite close to the requested minimum (this is illustrated in Figure 16 for the case of Topology 1, where the QoS is only 15% above the minimum requested). Consequently two further observations arise - the first is that the protocol functions as expected, getting close to the minimum requested QoS in an effort to accommodate as many sessions as possible, while the second, and more important, is that the relative difference between the delivered QoS and its requested minimum is greater than the call blocking probability. This shows the effectiveness of the negotiation strategies employed in that the gain in QoS due to insufficient bandwidth is greater than the probability of a session being blocked due to the same cause.

Finally, it must be mentioned that, all along our simulations, there was no big difference, *in absolute terms*, between the protocol's performance (average delivered QoS and call blocking probability) in the cases where only best-effort sessions were allowed and where both best-effort and guaranteed sessions constituted the traffic mix. Thus, if traffic management requirements dictate that some sessions must be run with maximum desired QoS, then this situation can be accommodated with negligible performance impact (a slight increase in both the QoS as well as the call blocking probability).

5.2. The Effect of A Traffic Threshold

Two scenarios were simulated for all topologies and associated negotiation strategies : with and without a traffic threshold t for guaranteed traffic. When such a threshold was in fact used, the actual chosen value was 30%.

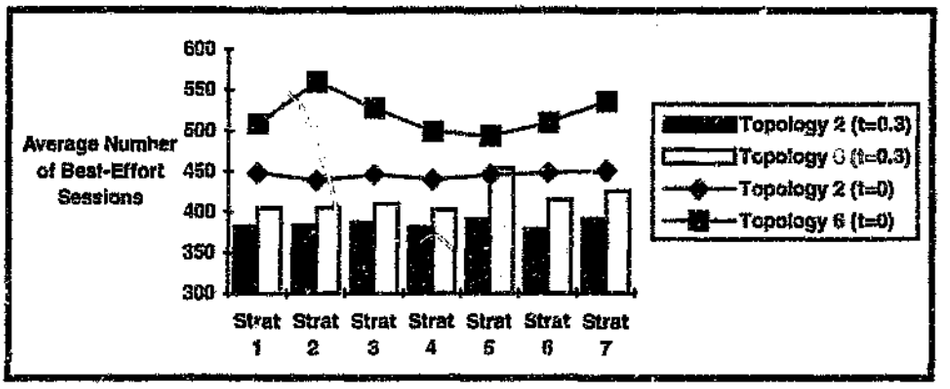

Figure 17 Topology 2 vs. 6 : Number of Best-Effort Sessions

Our simulation results show that, on average, the imposition of this threshold results in denying access to an average of 15% (or roughly half the threshold value) of best-effort sessions that would otherwise have been accepted had the threshold not been enforced. Thus, use of such a threshold would ensure that a minimum percentage of traffic (say $n\%$) would run at 100% desired QoS, provided that the corresponding decrease of roughly $\frac{n}{2}\%$ in the number of best-effort sessions is consistent with overall traffic requirements.

Conversely, if use of such a traffic threshold is in place and an increase of $m\%$ in the number of best-effort sessions is required, then the traffic threshold should be reduced by $2m\%$. Although the cost of implementing such a policy can be seen in the higher number of overall blocked calls (both best-effort and guaranteed), the difference in link utilisation between the two policies is negligible.

5.3. Bandwidth Division Strategies

Two bandwidth division strategies were simulated in order to see which one took better advantage of the asymptotic performance of VCS curves. The first, simpler, strategy just divided the requested QoS interval into five sub-intervals of equal size and took the corresponding sub-intervals on the bandwidth axis. The second strategy involved dividing both the requested QoS and the corresponding requested bandwidth intervals into

three sub-intervals of equal size, and then taking the intersection of these sub-intervals along the bandwidth axis to obtain the final five sub-intervals on the basis of which QoS negotiation is handled. This latter strategy is more complex and will be referred to as the *axis-adaptive* strategy.

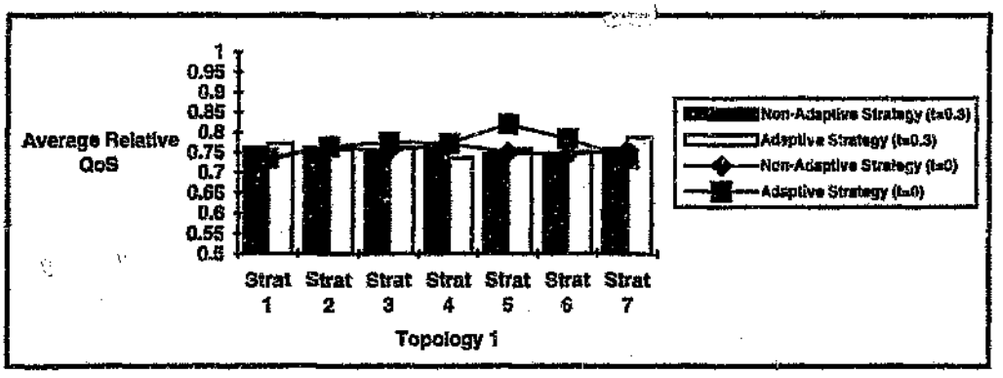

Figure 17: Bandwidth Division Choice

Although, on average, usage of an axis-adaptive bandwidth division does result in better relative (to the maximum desired) QoS being delivered to sessions, the improvement gained is not significant, being in the order of at most 5 percentage points. These results have been obtained irrespective of the simulated topologies and the usage of a traffic threshold for guaranteed traffic (see Figure 18 for an example) and indicate that the extra complexity involved in the adaptive scheme does not take significantly better advantage of the asymptotic property of the VCS curves. Thus, if the protocol management goal is to obtain good average QoS results fairly fast, then the simpler, non-adaptive bandwidth division scheme should be used instead.

5.4. Choice of QoS Negotiation Strategies

Our simulation results show that, even though we have an extremely congested scenario (in terms of flow, bandwidth available on links, as well as session-call frequency), the protocol ensures an average QoS for its best-effort sessions above 66% in all cases and above 70% in 92.8% of cases - irrespective of the bandwidth division and QoS

negotiation strategies employed and the topology used. Furthermore, for any particular (topology, bandwidth division, traffic threshold) triplet employed, the seven QoS negotiation strategies offer remarkably similar average QoS, the maximum difference in delivered QoS between best and worst strategies being at most 8.4% (for Topology 2 with an adaptive bandwidth division scheme and no traffic threshold) and in most cases being less than 4%.

Thus, there is a shift in emphasis from seeking a strategy which delivers best average QoS to looking for a strategy that delivers a better-than-average QoS *fast*. To this end, two approaches were pursued : firstly, in order to better appreciate the complexity associated with each QoS negotiation strategy, *complexity points* were awarded to each scheme according to how many stages it comprised (one point for each stage), while, within the negotiation stage itself, points were awarded on the basis of how many non-zero elements the respective 5-tuple contained. The results are shown in Figure 19.

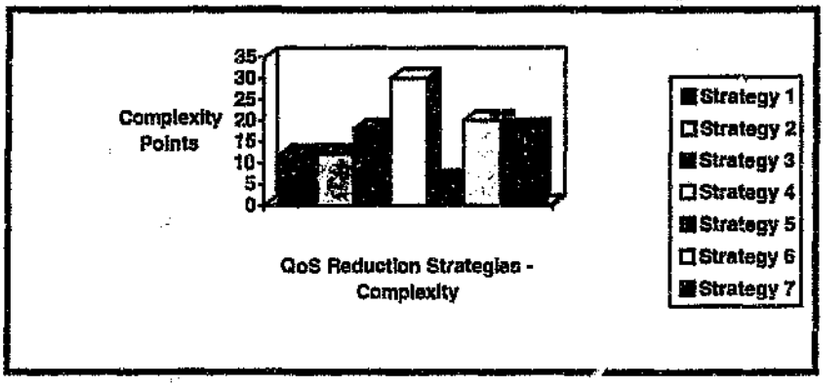

Figure 19 QoS Negotiation Strategies - Complexity Classification

Secondly, what was done was that, for each (topology, bandwidth division, traffic threshold) triplet, integer weights between 1 and 7 were correspondingly assigned to the QoS negotiation strategies in order of increasing delivered average QoS. The strategies' weights were then added across all possible scenarios and the results averaged out (Figure 20).

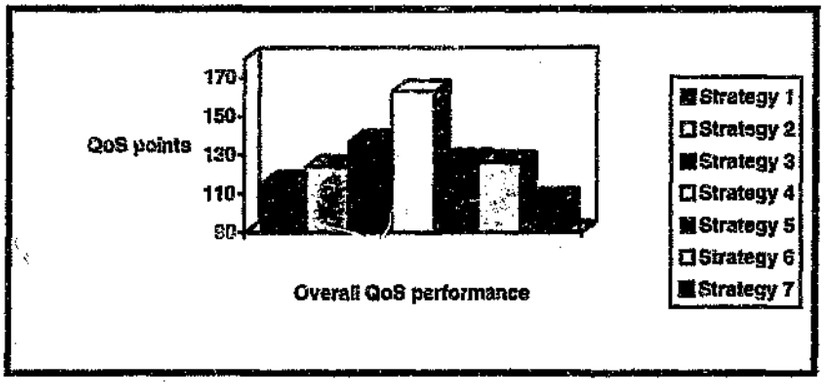

Figure 20 QoS Negotiation Strategies - QoS Classification

From the above figure, it can be seen that, although usage of the finest-grained negotiation policy (Strategy 4) does ensure the highest average QoS for best-effort sessions, this must be put in perspective considering the extra time overhead incurred by the negotiation process. On the other hand, usage of the coarsest-grained - and fastest - negotiation policy (Strategy 5) yields an average QoS which overall places it in the median position among QoS negotiation strategies. Therefore, if time is of the essence, this policy can be relied upon to quickly provide a median-valued QoS for its best-effort sessions, which, taking into account the usually small difference between maximal and minimal QoS delivered by the negotiation strategies, is a totally acceptable compromise.

Lastly, two more observations arise for strategies having equal complexity : from the fact that Strategy 2 delivers better average QoS than Strategy 1, with both having the same number of stages and complexity points, it would seem that better approach to follow would be to have a drastic reduction in QoS in the first stage followed by a finer reduction in later stages, rather than a more balanced reduction across the negotiation stages; also, from the fact that Strategy 3 delivers better (second-best overall, in fact) average QoS than Strategy 7, although they both have the same number of complexity points, the conclusion to be drawn would be that a better QoS negotiation strategy would have as few non-zero elements in any 5-tuple of its negotiation stages as possible - this conclusion is also borne out by the fact that the only simulated strategies having zero

elements in their 5-tuple - Strategies 5 and 7 - are located fifth and last in the overall classification of average delivered QoS.

5.5. Effect of Session Priority on QoS

The implemented QoS negotiation strategies attempt to favour higher priority best-effort sessions in two ways : firstly, by insisting that, within the same stage, lower priority sessions reduce their QoS before higher priority sessions, and secondly, by having lower priority sessions reduce their QoS by a greater number of QoS levels than higher priority sessions, again within the same stage of a QoS negotiation strategy. Of course, priority plays no part in guaranteed sessions' QoS, as their QoS is never negotiated and ensured of being the maximum desired.

Our expectations have been confirmed by our results. Thus, across all topologies and bandwidth division strategies, all the seven QoS negotiation strategies lead to a direct proportional dependency between a best-effort session's priority and its delivered QoS (see Figure 21), with sessions of highest priority boasting an average delivered QoS of above 86% (of the requested maximum) in all cases, while, at the opposite end, lowest-priority sessions have a delivered QoS which can sink as low as 58% of their requested maximum. Bearing in mind that a session's minimum desired QoS is, in absolute terms, always above 50%, this means that lowest-priority sessions are almost constantly on their lowest desired QoS levels.

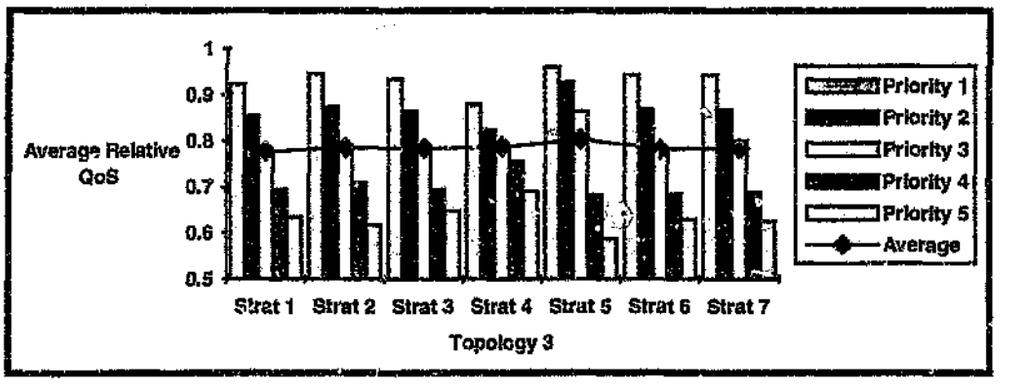

Figure 21 Effect of Session Priority on Average QoS

There is also a strong dependency between the standard deviation of the QoS delivered by a strategy and how fine-grained the strategy actually is. It is to be expected that the more coarser (i.e. having a greater number of stages and/or a greater reduction in the QoS levels within a stage) a negotiation strategy is, the higher the standard deviation of the delivered QoS. Thus, the coarsest-grained strategy, Strategy 5, always has the highest QoS standard deviation and invariably its lowest-priority sessions have a lowest average QoS, while the finest-grained strategy, Strategy 4, always has the lowest QoS standard deviation. These results are shown in Table 3 for the first simulated scenario of Topology 3 (a traffic threshold of 0.3 and the first bandwidth division scheme were used) which will now be analysed. The standard deviation calculated is the "sample" standard deviation, that is it is normalised by $n-1$, where n is the sequence length.

Strategy	QoS Standard Deviation
1	0.1163
2	0.1300
3	0.1184
4	0.0709
5	0.1618
6	0.1287
7	0.1296

Table 3 QoS Standard Deviation - Topology 3

The coarseness' effect of the QoS negotiation strategy can also be seen when studying strategies having an equal number of stages. Strategies 1 and 2 have the same number of stages, the only difference between them being that, in the first stage of Strategy 1, all best-effort sessions drop their QoS by 3 levels, whereas in Strategy 2, they drop their QoS by 4 levels. As expected, the more drastic (coarse) strategy, Strategy 2, delivers an average QoS with a higher standard deviation.

The effect that the number of stages of a QoS negotiation strategy has on the standard deviation of the delivered QoS can also be seen in a comparison between Strategies 2 and 3. Strategy 3 can be considered as being derived from Strategy 2, with the only difference being that the first stage of Strategy 2 is split in two identical stages in Strategy 3. Again, the finer-grained strategy, Strategy 3, delivers an average QoS with a lower standard deviation.

We now examine the influence the priority of a session has (via the QoS reduction size within the same stage of a strategy), on the delivered QoS of sessions. To this end, it is interesting to compare best-effort sessions having the topmost two priority levels in Strategies 6 and 7. In both strategies, lower priority sessions are always reduced by amounts strictly greater than those by which sessions of the topmost two priority levels are reduced. The only difference as far as the QoS negotiation goes is that Strategy 6 is finer grained, since highest priority sessions take the maximum number of stages (five) to get to their lowest QoS level. Both strategies deliver almost identical average QoS but Strategy 6 gives higher QoS for its two topmost priority levels than Strategy 7 does, showing the influence that even a slight bias in negotiation strategy towards sessions of a particular priority has on those sessions' QoS.

It is to be remarked that although best-effort sessions of second highest priority have the same negotiation sequence in both Strategies 6 and 7, their average delivered QoS is higher in Strategy 6. This is most probably due to first stage of Strategy 6 which is much more heavily biased against lower priority stages than the first stage of Strategy 7 is. Thus, although Strategy 6 has more stages than Strategy 7 and consequently a lower QoS standard deviation, its initial bias in favour of higher priority sessions results in a higher delivered QoS of those sessions. Hence, for the same QoS negotiation sequence of best-effort sessions of a particular category, an initial strong bias in their favour results in higher delivered average QoS.

5.6. Impact of Link Bandwidth on Protocol Performance

The effect of a reduction in link bandwidth on protocol performance was also analysed. As expected, there was a decrease in the average QoS coupled with an increase in the call blocking probability when link bandwidth is reduced.

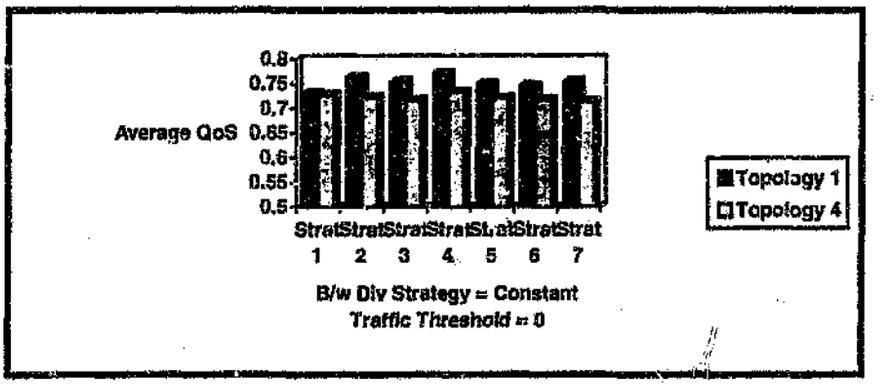

Figure 22 Topologies 1 vs. 4 : Average QoS

What is noteworthy, however, is the fact that the relative decrease in delivered QoS is far less than the relative increase in the call blocking probability. In all cases, the QoS never decreased by above 8%, while the increase in call blocking probability was, in many cases, roughly two-fold. These results are illustrated in Figures 22 and 23, for the case of a reduction in link bandwidth from 655 Mbps (Topology 1) to 100 Mbps

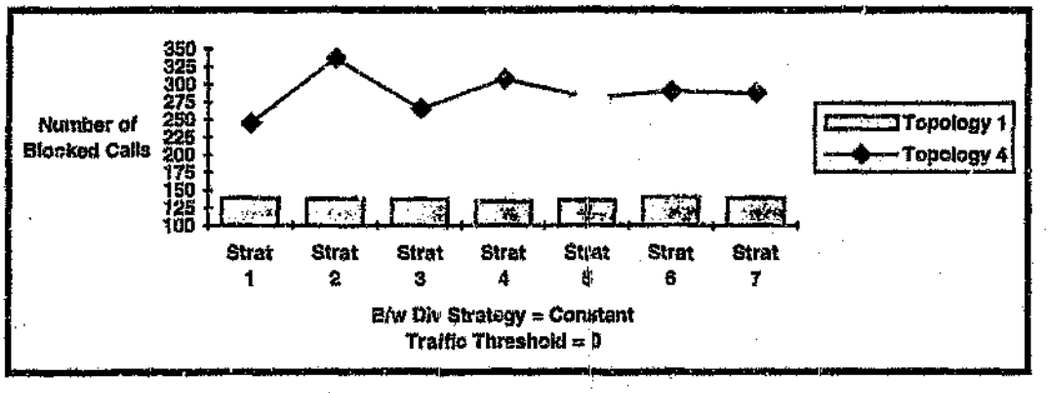

Figure 23 Topologies 1 vs. 4 : Blocked Calls

(Topology 4) and can be explained by the fact that, at any stage in time after the "warm-up" period of the simulation, the number of sessions currently active in the network is greater when the link bandwidth is higher (a fact also confirmed by the lower call blocking probability in this case). Thus, there is a better chance that more bandwidth will be freed in the negotiation process by best-effort sessions and that an incoming session will (a) be accommodated and (b) have a higher QoS.

5.7. Protocol Impact on Link Utilisation

For each type simulated topology, link utilisation was measured using *COMNET III*. This was done with the purpose of appreciating how well the protocol uses the available network resources - in this case link, bandwidth.

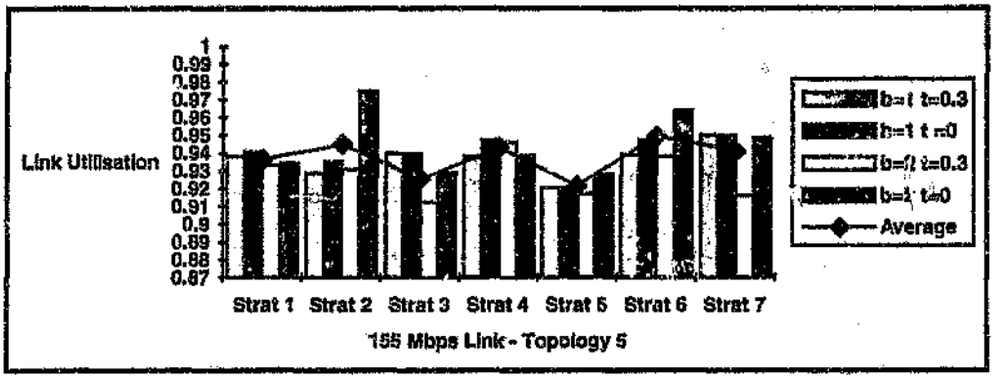

Figure 24 Topology 5 - Link Utilisation

The average link utilisation results show that links are never used to their full capacity; the maximum percentage of link bandwidth that is always completely used, although it varies according to the line type employed and the specific topology in which it is utilised (see Figures 24 and 25 for examples), is always above 87.4% (measured from the time that a first blocked call occurs) and above 90% in 95.9% of all simulated cases. Such consistently high utilisation of links indicate that the negotiation strategies of the protocol do an efficient job of bandwidth utilisation.

The fact that link utilisation never reaches 100 % can be attributed to sessions being blocked because the spare capacity is below their minimum required. Thus, if a best-effort session is blocked and the link utilisation is above the simulated average for the particular situation, what could be done in order to increase link utilisation would be to

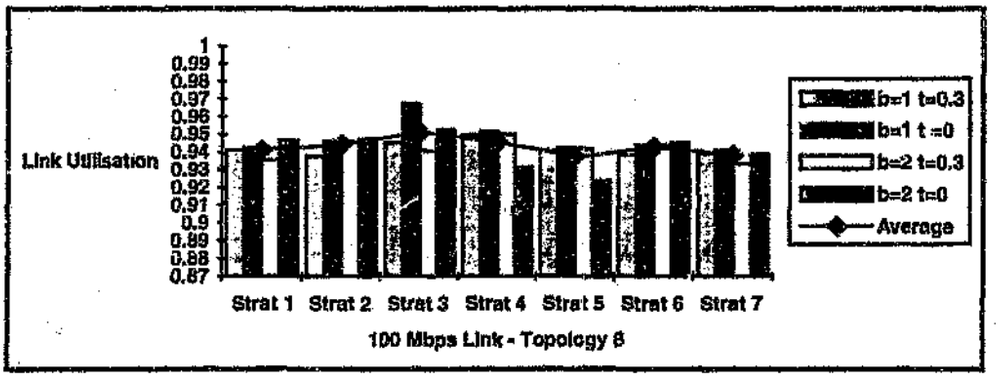

Figure 25 Topology 8 : Link Utilisation

decrease the sessions' bandwidth needs by having it, for instance, reduce a combination of its video resolution, colour depth and audio bit sampling rate, until enough bandwidth becomes availat le to accommodate the session's initial requirements.

Lastly, there is a good correlation between link utilisation and average delivered QoS (e.g. 0.7178 and 0.8193 in the first scenarios of Topologies 5 and 8, respectively) - this is to be expected since our measure of the technical component of QoS is required bandwidth.

5.8. Pricing

The effect that introducing the pricing policy of (9) into the overall QoS negotiation framework is now studied. While the network uses the designed protocol to provide QoS guarantees, pricing is used as a means of influencing session behaviour and resource allocation.

The parameters of multimedia sessions' arrival time and duration are identical to those used in all our previous simulations. From the sessions' point of view, the only difference is that they are now allocated an *available budget*. Accordingly, a session is now characterised by the triplet *(QoS_class, connection_duration, available_budget)*.

For simplicity's sake, in simulations we have assumed that the available budget of a session represents the price per unit of time that the session can afford to pay. To this end, a session's available budget is obtained from an exponential distribution with mean $\beta = 1$. This is in contrast to the analysis done in [ParFer92] where it is assumed that user wealth is bimodally distributed (i.e. takes one out of only two possible values).

The exponential distribution was selected to generate a session's budget because it closely emulates the inversely proportional relationship between a budget's size and the number of sessions that can afford that budget. The fixed start-up cost of a connection was set equal to:

$$no\_of\_switches*(port\_cost + buffer\_cost) \qquad (19)$$

where *no_of_switches* represents the number of switches along the session's route and the *port_cost* and *buffer_cost* parameters represent the cost of leasing a switch's port and part of its overall buffering capacity, respectively. Since all of the terms involved in the variable part of (9) involve sub-unitary values, the *port_cost* was set equal to 0.25, while the *buffer_cost* was set equal to $0.15/no\_of\_ports$, where *no_of_ports* is the switch's total number of ports. These considerations have also motivated our choice of mean for the exponential distribution which generated the sessions' budgets. The point must be made that our choice of simulation parameter values was made not in order to reflect any current or future market situation, but rather to illustrate the pricing policy with quantitative examples.

We aim to study the following questions ; How does the implemented pricing scheme affect the call blocking probability and network link utilisation? Of the total number of blocked calls, what percentage is given by calls blocked by budgetary constraints? How does introduction of the pricing scheme affect the QoS delivered by the designed protocol? How does the pricing policy discriminate against users?

The combined effect of the designed protocol and pricing policy was simulated with respect to Topology 7. In all QoS negotiation strategies the first bandwidth division scheme was used as it is simpler to implement (its complexity is not under scrutiny here) and no threshold for guaranteed traffic employed (we are only interested in calls blocked by budgetary constraints, not in those blocked by traffic-mix considerations). The *thr* parameter of (8) had a value of 0.3 in all simulated cases. Our results are given below.

No Pricing Policy Implemented				**Pricing Policy In Use**			
Strategy	Average Delivered QoS	Link Utilisation	Blocked Calls	Average Delivered QoS	Link Utilisation	Blocked Calls	Budget-Blocked Calls
1	0.7686368	0.9367	138.3	0.8899032	0.8364	432.6	371.9
2	0.7601892	0.9673	144.9	0.8991470	0.8402	467.3	389.9
3	0.7195094	0.9219	123.1	0.8614176	0.8732	429.4	356.5
4	0.7828772	0.9542	123.2	0.8998342	0.8452	433.5	372.3
5	0.7320973	0.9532	140.1	0.8773146	0.8356	433.2	382.7
6	0.7815366	0.9225	122.0	0.8960547	0.8371	438.4	379.5
7	0.7588326	0.9373	99.6	0.8836708	0.8382	430.2	383.3

Table 4 Pricing Policy Effect - Topology 7

The most noteworthy result obtained is the over three-fold increase in the number of blocked calls, independently of what QoS negotiation strategies have been employed. This increase is due to the pricing policy's discrimination against sessions with a low budget, a fact further reflected by the high proportion of sessions blocked due to budgetary constraints. Thus, in all simulated cases, this proportion represents at least 83%

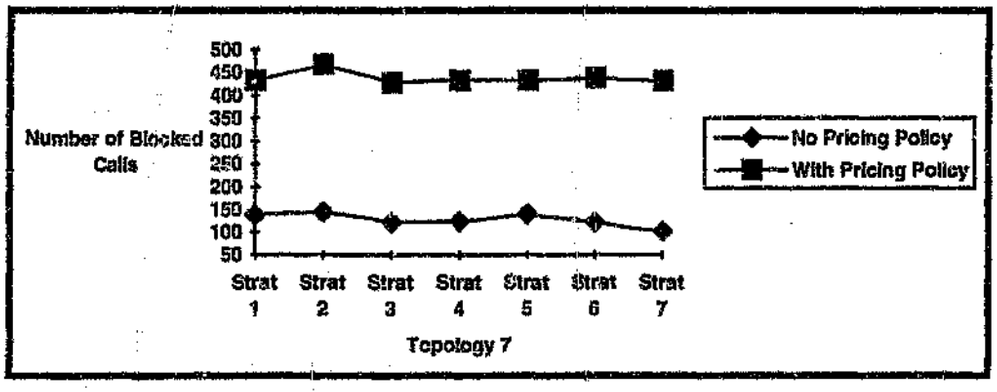

Figure 26 Effect of Pricing Policy on Blocked Calls

of the total number of blocked sessions, and the figure can go up to 89% as is the case with Strategy 7. On the other hand, sessions that are blocked purely because of network constraints (i.e. lack of bandwidth) experience a drastic reduction in numbers to roughly half the values encountered (the average decrease is 54.44%, but it can be as high as 63.95%) when no pricing policy was enforced. This fact is of course made possible by the greater availability of bandwidth that a pricing scheme introduces.

The greater bandwidth available to sessions when a pricing policy is in place means that the best effort sessions will be able to run at QoS levels closer to their maximum desired ones. This is illustrated by the increase in delivered average QoS experienced by best-effort sessions with budgets big enough for them to be accommodated. This increase is again experienced irrespective of the QoS negotiation strategy used and comes at the expense of poorer sessions who are blocked because they cannot afford to pay the price charged by the network pricing policy for their requested QoS. What must furthermore be mentioned is that, in the context of our protocol, where the variations in delivered average QoS are small across all simulated scenarios, the average *absolute* increase of 12.9% represents a very good result, especially if it is put into perspective by noting the fact that the margin for improvement of best-effort sessions' QoS was, on average, only 24.2%. Thus, most accepted best-effort sessions take advantage of a pricing policy to increase their QoS by an amount representing more than 50% of the maximum possible.

The pricing policy affects link utilisation in two ways. Firstly, the greater call blocking probability means, that at any instant in time, there are fewer multimedia sessions on the network, so link utilisation should go down. Secondly, the fact that there are fewer multimedia sessions means that there is more bandwidth available on the network, and consequently improved chances that a session should run closer to its desired maximum QoS. Thus, link utilisation should increase because of this factor. Since there is an across the board decrease in link utilisation when the pricing policy is introduced, this indicates that the dominating tendency is the first one enumerated. Thus, although best-effort sessions have a much better delivered average QoS and consequently use up correspondingly increased amounts of bandwidth, there is still bandwidth available on the system that could have been used by other sessions had they afforded to pay for it.

5.9. The Session Knowledge Base

All our major results have been incorporated in the protocol SKB. This is a rule-based system and comprises three main categories of rules : the first deals with traffic management principles, the second with bandwidth division strategies, while the last is concerned with QoS negotiation strategies.

In the construction of the SKB it has been presumed that the sessions can further be characterised by a measure of how fast their required response time is. Thus, sessions were divided into two groups : those needing a *fast* response time, and those requiring an *ordinary* response time. For our purposes, a response time is considered to be fast if the multimedia server's response time is between 200 and 300ms, and the overall end-to-end delay experienced by the session is under 3s. Sessions which are not bound by these constraints are considered to require an *ordinary* response time.

The following rules are thus far contained in the protocol SKB :

- **Traffic Management Rules**

1. If *traffic_threshold* > 0 and *increase(best_effort_sessions,m%)* then

*traffic_threshold = traffic_threshold - minimum(traffic_threshold,2*m)*

2. If *traffic_threshold* < 1.0 and *increase(guaranteed_sessions,n%)* then

*traffic_threshold = maximum(1.0, traffic_threshold+2*n)*

Here, we have denoted by *increase(session_type, x%)* the request to increase the number of sessions of the specified type which are currently running in the network by *x%*, while *minimum(a,b)* and *maximum(a,b)* return the minimum and maximum, respectively, of the two numbers *a* and *b*.

- **Bandwidth Division Rules**

1. If *Session_is_Fast* then

use first bandwidth division scheme

2. If Not(*Session_is_Fast*) then

use axis adaptive bandwidth division scheme

The *Session_is_Fast* predicate returns true if the session under consideration is deemed to require a fast response time, and false otherwise.

- **QoS Negotiation Rules**

Since one of the main objectives of the protocol is to provide multimedia sessions with a high average QoS, what was firstly decided upon when building this set of rules was, for each simulated topology, to only consider the strategies which provide the top five average QoS for their best effort sessions. This choice also provides a nice one-to-

one correspondence between the number of available strategies and sessions' priority, so that sessions with the highest priority use the best strategy, sessions with second highest priority use the second best strategy, and so on, up to lowest-priority sessions which make use of the fifth best QoS-provisioning strategy.

What was further decided was to differentiate between topologies that have been simulated and those that have not. Accordingly, for simulated topologies, if a session was deemed to be fast, then the strategies' QoS-delivery performance was moderated by their complexity. Thus, for any simulated scenario, the five best QoS strategies were ranked according to how high they scored on the following *performance index* :

$$\frac{delivered\_QoS}{best\_QoS * complexity} \qquad (20)$$

where *delivered_QoS* is the average QoS delivered by the strategy in question, *best_QoS* is the highest QoS delivered for the particular network scenario and *complexity* represents the strategy's complexity points, as given by Figure 19. For each simulated scenario, the strategy ranking has been compiled and the results stored into tables which can be consulted by the protocol. Sessions then use the strategy that corresponds to their respective priorities, as has been explained. If the session is not deemed to be fast, then the session uses the strategy ranking table which has been obtained without being moderated by strategies' complexity.

Finally, if the protocol is running on a topology other than the ones which have been simulated, the overall average QoS results given by Figure 20 are used. The same principles used for simulated topologies guide us in our choice of strategies. Thus, Strategies 1 and 7 are not considered because they deliver the lowest QoS. If a session is deemed to be fast, then the strategies' ranking is also given by (20), but *delivered_QoS* now represents the number of QoS points of the strategy (taken from Figure 20), *best_QoS* represents the maximum number of QoS points provided by any strategy (this is Strategy 4 with 163 points), while *complexity* has the same meaning as

Strategy	QoS Points	Complexity Points	Performance Index	Rank
2	124	12	0.06339	2
3	137	18	0.04669	3
4	163	30	0.03382	5
5	127	6	0.12986	1
6	126	20	0.03865	4

Table 5 Overall Strategy Ranking

before, being the respective strategy's number of complexity points. The strategy classification thus obtained is given in Table 5. The following rules can now be formulated for this case :

1. If *Session_Priority* = 1 and *Session_is_Fast* then

   use Strategy 5

2. If *Session_Priority* = 1 and *Session_is_Fast* then

   use Strategy 2

3. If *Session_Priority* = 1 and *Session_is_Fast* then

   use Strategy 3

4. If *Session_Priority* = 1 and *Session_is_Fast* then

   use Strategy 6

5. If *Session_Priority* = 1 and *Session_is_Fast* then

   use Strategy 4

Lastly, if a session is not deemed to be fast, then it uses the strategy ranking obtained without taking into account the strategy's complexity. The protocol then consults

this ranking and allocates the strategy which corresponds in rank to the session's priority, as before.

The protocol can, if required, consult the SKB during execution to improve performance. The traffic management rules can be used by a human agent, or by an intelligent program, both of which can dynamically advise on desirable traffic mixes. The bandwidth division and QoS negotiation strategies rules are used by the protocol on a per-session basis in order to deliver the best possible QoS, taking into account the session's time constraints.

6. CONCLUSIONS

We have presented the design of an ATM call admission protocol for multimedia which incorporates human QoS. The main idea behind our research has been to study the extent to which human QoS can be sacrificed in order to accommodate a greater number of sessions in a distributed multimedia environment. Eight different topologies were studied. For each topology, a combination of seven QoS negotiation strategies, coupled with two different bandwidth division schemes were simulated.

There are two main results that we have obtained. The first is that there is indeed the case for negotiation of human QoS in order to accept a greater number of sessions - the loss of average QoS due to negotiations is more than compensated for by a significant drop of the call blocking probability. The second is that, even when dealing with an excessively congested scenario, the protocol delivers an average QoS which is situated in a fairly narrow range (between 66% and 82% in all cases and between 70% and 82% in proportion of 92.8% of cases). This indicates that protocol performance scales reasonably well and is not significantly affected by the topology over which it is applied (the protocol delivers relatively constant QoS irrespective of the topologies tested out and their bandwidth constraints).

We have also formulated and implemented a pricing policy. The policy is based on the designed protocol and incorporates aspects of both peak-load and set-up pricing. Its effect on protocol performance has been studied through simulation. The results obtained have shown that usage of the protocol coupled with the pricing policy leads to a significant increase (over 50% of the available margin for improvement) in the average

QoS delivered to best effort sessions. In the context of the protocol where, if no pricing scheme is used, the delivered QoS is high and situated in quite a narrow range this increase represents a commendable result. The improvement in delivered QoS comes at the expense of sessions who cannot afford to pay for the required resources. Consequently, there is a drastic, three-fold, increase in the call blocking probability. However, the blocking probability of sessions that are blocked purely because of lack of network resources experiences a sharp decline of roughly 50% when the pricing policy is used.

A protocol Session Knowledge Base (SKB) has also been implemented. The SKB can be dynamically consulted by the protocol in order to better performance. It is a rule-based system whose productions have been obtained from our simulated results. The SKB comprises traffic management rules and two sets of rules dealing with QoS negotiation strategies. The latter contains recommendations for strategies to be used, based on a session's priority and time constraints. An interesting result which has been obtained is that if one takes into account the complexity of the QoS negotiation strategies, then the best overall strategy to be used is the simplest strategy (i.e. the coarsest grained) possible, while the most complex (fine-grained) strategy, although it delivers by far the highest average QoS, performs the worst if its complexity is reckoned with.

It must be mentioned that, although offering extremely interesting results, the development of the protocol SFB and the associated pricing policy was not the main focus of our research - the design of the protocol was. Bearing this in mind, it would be interesting to compare our choice of pricing policies with others (such as ones that use different estimates of sessions' required bandwidth). The SKB is itself a dynamic entity, where rules can be added, removed, or tuned, as more experience with running the protocol becomes available.

Within the context of the protocol itself, more bandwidth division and QoS negotiation strategies can be explored and experimented with. Although we are fairly

confident that our choice of complexity measure for QoS negotiation strategies does provide a good appreciation of the strategies' complexity, perhaps better measures of a strategy's complexity might be found, especially if the protocol is actually implemented on a hardware testbed.

Finally, it would be interesting to further investigate the human response to new compression algorithms (such as MPEG-II), different resolutions and window sizes, as well as colour schemes and audio bit sample rates, and apply these results in an ATM context. This might lead to more economical resource allocation strategies being used and would afford us a better understanding of both the role the human element plays in the reception and synthesis of multimedia data, as well as the place in the ATM protocol stack where user QoS parameters should be incorporated.

REFERENCES

[AhmDen89] AHMADI, H., and DENZEL, W.E., "A Survey of Modern High-Performance Switching Techniques", *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 7, Sept. 1989, pp. 1091-1103

[Apt*etal*94] APTEKER, R.T., FISHER, J.A., KISIMOV, V.S., and NEISHLOS, H., "User Perception and Dynamic QoS", SPIE conference on *"High Speed Networking and Multimedia Computing"*, USA, 1994

[Apt*etal*95a] APTEKER, R.T., FISHER, J.A., KISIMOV, V.S., and NEISHLOS, H., "End User Receptivity : An Important Factor in the Design of a Distributed Multimedia System", *International Symposium on Multimedia Communications and Video Coding*, IEEE Communication and Signal Processing Societies, New York, 1995

[Apt*etal*95b] APTEKER, R.T., FISHER, J.A., KISIMOV, V.S., and NEISHLOS, H., "Video Acceptability and Frame Rate", *IEEE Multimedia*, Vol. 2, No. 3, Fall 1995, pp. 32-40

[Baa92] BAASE, S., *"Computer Algorithms - Introduction to Design and Analysis"*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1992

[Ban*etal*94] BANARJEA, A., FERRARI, D., MAH, B., MORAN, M., VERMA, D., and ZHANG, H., *"The Tenet real-time protocol suite : design, implementation, and experiences"*, Technical Report TR-94-059, International Computer Science Institute, Nov. 1994

[Cat92] CATLETT, C.E., "In Search of Gigabit Applications", *IEEE Communications Magazine*, Vol. 30, No. 4, April 1992, pp. 42-51

[Cha91] CHAO, H.J., "A Recursive Modular Terabit/Second ATM Switch", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 8, Oct. 1991, pp. 1161-1172

[ChaCho95] CHAO, H.J., and CHOE, B.-S., "Design and Analysis of a Large-Scale Multicast Output Buffered ATM Switch", *IEEE/ACM Transactions on Networking*, Vol. 3, No. 2, April 1995, pp. 126-138

[ClaSheZha92] CLARK, D.D., SHENKER, S., and ZHANG, L., "Supporting Real-Time Applications in an Integrated Services Packet Network : Architecture and Mechanism", *Proceedings ACM SIGCOMM '92*, Baltimore, Aug. 1992, pp. 14-26

[Coc*etal*91] COCHI, R., ESTRIN, D., SHENKER, S. and ZHANG, L., "A Study of Priority Pricing in Multiple Service Class Networks, *Proceedings ACM SIGCOMM '91*, New York, Sept. 1991, pp. 123-129

[CopDamMel93] COPPO, P., D'AMBROSIO, M., and MELEN, R., "Optimal Cost/Performance Design of ATM Switches", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 5, Oct. 1993, pp. 566-575

[DemKesShe90] DEMERS, A., KESHAV, S., and SHENKER, S., "Analysis and Simulation of a Fair Queueing Algorithm", *Internetwork : Research and Experience*, Vol. 1, No. 1, John Wiley & Sons, Sept. 1990, pp. 3-26

[DevCocSer88] DEVAULT, M., COCHENNEC, J., and SERVEL, M., "The Prelude ATD Experiment : Assessments and Future prospects", *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, Dec. 1988, pp. 1528-1537

[EngKarYeh92] ENG, K.Y., KAROL, M.J. and YEH, Y.-S., "A Growable Packet (ATM) Switch Architecture : Design Principles and Applications", *IEEE Transactions on Communications*, Vol. 40, No. 2, Feb. 1992, pp. 423-430

[Fer90] FERRARI, D., *"Client Requirements for Real-Time Communication Services"*, UC Berkeley, RFC 1193, November 1990

[FerVer90] FERRARI, D., and VERMA, D.C., "A Scheme for Real-Time Channel Establishment in Wide Area Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990, pp. 368-379

[Fro87] FROST, R.A., "An Introduction to Knowledge Base Systems", 2nd ed., Collins, London, 1987

[GarJoh79] GAREY, M.R., and JOHNSON, D.S., *"Computers and Intractability : A Guide to the Theory of NP-Completeness"*, 1st ed., W.H. Freeman and Company, New York, 1979

[Gia*etal*91] GIACOPELLI, J.N., HICKLEY, J.J., MARCUS, W.S., SINCOSKIE, W.D., and LITTLEWOOD, M., "Sunshine : A High-Performance Self-Routing Broadband Packet Switch Architecture", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 8, Oct. 1991, pp. 1289-1298

[Gol90] GOLESTANI, S.J., "A Stop and Go Queueing Framework for Congestion Management", *Proceedings ACM SIGCOMM '90*, Philadelphia, Sept. 1990, pp. 8-18

[GueAhmNag91] GUERIN, R., AHMADI, H., and NAGHSHINEH, M., "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 7, Sept. 1991, pp. 968-981

[Hal92] HALSALL, F., *"Data Communications, Computer Networks and Open Systems"*, 1st ed., Addison-Wesley Publishing Company, Reading, Massachusetts, 1992

[HanHubSch94] HANDEL, R., HUßER, M.N., and SCHRODER, S, *"ATM Networks"*, 1st ed., Addison-Wesley Publishing Company, Reading, Massachusetts, 1994

[HluKar88] HLUCHYJ, M.G., and KAROL, M.J., " Queueing in High-Performance Packet Switching", *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, Dec. 1988, pp. 1587-1597

[HuaKna84] HUANG, A., and KNAUER, S., "Starlite : A Wideband Digital Switch", *Proceedings of GLOBECOM '84*, Atlanta, Georgia, Nov. 1984, pp. 121-125

[HuiArt87] HUI, J.Y., and ARTHURS, E., "A Broadband Packet Switch for Integrated Transport", *IEEE Journal on Selected Areas in Communications*, Vol. 5, No. 8, Oct. 1987, pp. 1264-1273

[Ito91] ITOH, A., "A Fault-Tolerant Switching Network for B-ISDN", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 8, Oct. 1991, pp. 1218-1226

[Ito*etal*91] ITOH, A., TAKAHASHI, H., NAGANO, H., KURISAKA, M., and IWASAKI, S., "Practical Implementation and Packaging Technologies for a Large-Scale ATM Switching System", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 8, Oct. 1991, pp. 1280-1288

[Itu91] INTERNATIONAL TELECOMMUNICATIONS UNION - T : Recommendation I.321, *"B-ISDN Protocol Reference Model and its Application"*, Geneva, 1991

[Jac90] JACOB, A.R., "A Survey of Fast Packet Switches", *ACM Computer Communication Review*, Vol. 20, No. 1, Jan. 1990, pp. 54-64

[KarHluMor87] KAROL, M.J., HLUCHYJ, M.G., and MORGAN, S.P., "Input Versus Output Queueing on a Space-Division Packet Switch", *IEEE Transactions on Communications*, Vol. 35, No. 12, Dec. 1987, pp. 1347-1356

[KarI92] KAROL, M.J., and I, C.-L., "Performance Analysis of a Growable Architecture for Broadband Packet (ATM) Switching", *IEEE Transactions on Communications*, Vol. 40, No. 2, Feb. 1992, pp. 431-439

[KalKanKes90] KALMANEC, C., KANAKIA, H., and KESHAV, S., "Rate Controlled Servers for Very High-Speed Networks", *Proceedings of GlobeCom '90*, pp. 300.3.1-300.3.9, 1990

[Kim94] KIM, H.S., "Design and Performance of Multinet Switch : A Multistage ATM Switch Architecture with Partially Shared Buffers", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 6, Dec. 1994, pp. 571-580

[KimLeo90] KIM, H.S., and LEON-GARCIA, A., "A Self-Routing Multistage Switching Network for Broadband ISDN", *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990, pp. 459-466

[Kle75a] KLEINROCK, L., *"Queueing Systems, Vol. 1 : Theory"*, 1st ed., Wiley, New York, 1975

[Kle75b] KLEINROCK, L., *"Queueing Systems, Vol. 2 : Computer Applications"*, 1st ed., Wiley, New York, 1975

[Kro90] KRONER, H., "Comparative Performance Study of Space Priority Mechanisms for ATM Networks", *Proceedings of the INFOCOM '90*, San Francisco, 1990, pp. 1136-1143

[Kun92] KUNG, H.T., "Gigabit Local Area Networks : A System Perspective", *IEEE Communications Magazine*, Vol. 30, No. 4, April 1992, pp. 79-89

[Kur93] KUROSE, J.F., "Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks", *ACM Computer Communication Review*, Vol. 23, No. 1, January 1993, pp. 6-15

[Lau94] LAUBACH, M., *"Classical IP and ARP over ATM"*, Hewlett-Packard Laboratories, RFC 1577, January 1994

[LawKel91] LAW, A.M., and KELTON, D.W., *"Simulation Modelling and Analysis"*, 2nd ed., Mc-Graw Hill, New York, 1991

[LawMcc94] LAW, A.M., and McCOMAS, M.G., "Simulation Software for Communications Networks : The State of the Art", *IEEE Communications Magazine*, Vol. 32, No. 3, March 1994, pp. 44-50

[LazPacWhi90] LAZAR, A.A., PACIFICI, G., and WHITE, J.S., "Real-Time Traffic Measurements on MAGNET II", *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990, pp. 467-483

[Lee88] LEE, T.T., "Nonblocking Copy Networks for Multicast Packet Switching", *IEEE Journal on Selected Areas in Communications"*, Vol. 6, No. 9, Dec. 1988, pp. 1455-1467

[Lee90] LEE, T.T., "A Modular Architecture for Very Large Packet Switches", *IEEE Transactions on Communications*, Vol. 38, No. 7, July 1990, pp. 1097-1106

[LeeByu94] LEE, T.T., and BYUN, J.W., "The Design and Analysis of an ATM Multicast Switch with Adaptive Traffic Controller", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 3, June 1994, pp. 288-298

[Lel*etal*94] LELAND, W.E., TAQQU, M.S., WILLINGER, W., and WILSON, D.V., "On the Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 1, Feb. 1994, pp. 1-14

[LowVar93] LOW, S.H., and VARAIYA, P.P., "A New Approach to Service Provisioning in ATM Networks", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 5, Oct. 1993, pp. 547-553

[Mce92] McEACHERN, J.A., "Gigabit Networking on the Public Transmission Network", *IEEE Communications Magazine*, Vol. 30, No. 4, April 1992, pp. 70-78

[Mil94] MILLER, A., "From here to ATM", *IEEE Spectrum Magazine*, June 1994, pp. 20-24

[MinKei94] MINOLI, D., and KEINATH, R., *"Distributed Multimedia through Broadband Communications Services"*, 1st ed., Artech House, Norwood, Massachusetts, 1994

[NahSte95] NAHRSTEDT, K., and STEINMETZ, R., "Resource Management in Networked Multimedia Systems", *IEEE Computer*, Vol. 28, No. 5, May 1995, pp. 52-63

[New92] NEWMAN, P., "ATM Technology for Corporate Networks", *IEEE Communications Magazine*, Vol. 30, No. 4, April 1992, pp. 90-101

[ParGal92] PAREKH, A. and GALLAGER, R., *"A Generalised Processor Sharing Approach to Flow Control in Integrated Services Networks"*, Technical Report LIDS-TR-

2089, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1992

[ParFer92] PARRIS, C. and FERRARI, D., *"A Resource-Based Pricing Policy for Real-Time Channels in a Packet-Switching Network"*, Technical Report TR-92-018, The Tenet Group, International Computer Science Institute, Berkeley, 1992

[ParKesFer92] PARRIS, C., KESHAV, S., and FERRARI, D., *"A Framework for the Study of Pricing in Integrated Networks"*, Technical Report TR-92-016, The Tenet Group, International Computer Science Institute, Berkeley, 1992

[Par94] PARTRIDGE, C., *"Gigabit Networking"*, Addison-Wesley Publishing Company, 1st ed., Reading, Massachusetts, 1994

[Pat91] PATTAVINA, A., "An ATM Switch Architecture for Provision of Integrated Broadband Services", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, Dec. 1991, pp. 1537-1548

[Pat93] PATTAVINA, A., "Nonblocking Architectures for ATM Switching", *IEEE Communications Magazine*, Vol. 31, No. 2, Feb. 1993, pp. 38-48

[PatGia94a] PATTAVINA, A., and GIANATTI, S., "Performance Analysis of ATM Banyan Networks with Shared Queueing - Part I : Random Offered Traffic", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 4, Aug. 1994, pp. 398-410

[PatGia94b] PATTAVINA, A., and GIANATTI, S., "Performance Analysis of ATM Banyan Networks with Shared Queueing - Part II : Correlated/Unbalanced Offered Traffic", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 4, Aug. 1994, pp. 411-424

[Rath91] RATHGEB, E.P., "Modelling and performance comparison of policing mechanisms for ATM networks", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 2, April. 1991, pp. 325-332

[RooCheGar94] ROOHOLAMINI, R., CHERKASSKY, V., and GARVER, M., "Finding the Right ATM Switch for the Market", *IEEE Computer Magazine*, Vol. 27, No. 4, April 1994, pp. 16-28

[RosLag94] ROSENBERG, C., and LAGUE, B., "A Heuristic Framework for Source Policing in ATM Networks", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 4, Aug. 1994, pp. 387-397

[Rui94] RUIU, D., "Testing ATM Systems", *IEEE Spectrum Magazine*, June 1994, pp. 25-27

[RusNor95] RUSSELL, S., and NORVIG, P., *"Artificial Intelligence - A Modern Approach"*, 1st ed., Prentice-Hall, Englewood Cliffs, New Jersey, 1995

[Smu94] SMUTS, W.B., "Accounting in High-Speed Networks", *Proceedings of the FRD and Telkom Conference on Teletraffic and Rural Communications"*, pp. 95-98, Pretoria, August 1994

[SiuJai95] SIU, K-Y., and JAIN. R., "A Brief Overview of ATM : Protocol Layers, LAN Emulation, and Traffic Management", *ACM Computer Communication Review*, Vol. 25, No. 1, January 1995, pp. 6-20

[Sch88] SCHWARTZ, M., *"Telecommunication Networks : Protocols, Modelling and Analysis"*, 2nd ed., Addison-Wesley Publishing Company, Reading, Massachusetts, 1988

[Som92] SOMMERVILLE, I., *"Software Engineering"*, 4th ed., Addison-Wesley Publishing Company, Reading, Massachusetts, 1992

[Tob90] TOBAGI, F.A., "Fast Packet Switch Architectures for Broadband Integrated Services Digital Network", *Proceedings of the IEEE*, Vol. 78, no. 1, Jan. 1990, pp. 133-167

[Top90] TOPOLCIC, C., *"Experimental Internet Stream Protocol, Version 2 (ST-II)"*, Network Information Centre, RFC 1190, Oct. 1990

[Tur86] TURNER, J.S., "New Directions in Communications (or Which Way to the Information Age)", *IEEE Communications Magazine*, Vol. 24, No. 10, Oct. 1986, pp. 8-15

[Tur88] TURNER, J.S., "Design of a Broadcast Packet Switch Network", *IEEE Transactions on Communications*, Vol. 36, no. 6, pp. 734-743, June 1988

[VerZhaFer91] VERMA, D., ZHANG, H., and FERRARI, D., "Delay Jitter Control for Real-Time Communication in a Packet Switching Network", *Proceedings IEEE TriCom'91*, pp. 35-41, 1991

[WooKos90] WOODRUFF, G.M., and KOSITPAIBOON, R., "Multimedia Traffic Management Principles for Guaranteed ATM Network Performance", *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, April 1990, pp. 437-446

[YehHluAca87] YEH, Y.-S., HLUCHYJ, M.G., and ACAMPORA, A.S., "The Knockout Switch : A Simple, Modular Architecture for High-Performance Packet Switching", *IEEE Journal on Selected Areas in Communications*, Vol. 5, No. 8, Oct. 1987, pp. 1274-1283

[Zeg93] ZEGURA, E.W., "Architectures for ATM Switching Systems", *IEEE Communications Magazine*, Vol. 31, No 2, Feb. 1993, pp. 28-37

[Zhaetal93] ZHANG, L., DEERING, S., ESTRIN, D., SHENKER, S., and ZAPPALA, D., "RSVP : A New Resource ReSerVation Protocol", *IEEE Network*, Vol. 7, No. 5, Sept. 1993, pp. 8- 19

[ZhaKes91] ZHANG, L., and KESHAV, S., "Comparison of Rate-Based Service Disciplines", *Proceeding ACM SIGCOMM '91*, New York, Sept. 1991, pp. 113-122