Technical Report

UCAM-CL-TR-245
ISSN 1476-2986

Number 245

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

System support for multi-service traffic

Michael J. Dixon

January 1992

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted September 1991 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Fitzwilliam College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Preface

Except where otherwise stated in the text, this dissertation is the result of my own work and is not the outcome of work done in collaboration.

This dissertation is not substantially the same as any I have submitted for a degree or diploma or any other qualification at any other university.

Acknowledgements

I would like to thank my supervisor, Jean Bacon, for her encouragement and advice during my time as a research student. Special thanks are due to Mike Burrows, Mark Hayter, Derek McAuley, Cosmos Nicolaou and many other members of the Computer Laboratory for helpful discussions over the years. I would like to thank Roger Needham, Head of the Computer Laboratory, for his support, and for allowing me time to spend one summer as an intern at the DEC Systems Research Centre, Palo Alto. Thanks are also due to Sape Mullender and his colleagues at CWI for providing access to the Amoeba source code. The starting point they provided has been invaluable. The WANDA system as presented is the result of contributions by many people. However, Mark Hayter, Glenford Mapp and Tim Wilson deserve special mention. I am grateful to Andy Hopper for encouraging the close ties that exist between the Computer Laboratory and Olivetti Research Limited. In particular, David Greaves at Olivetti has been a source of help on all matters concerning the Cambridge Backbone Network. The dissertation has benefitted from inspection by the careful eyes of Jean Bacon, Richard Black, Andy Gordon, Andy Harter, Mark Hayter, Derek McAuley and Cosmos Nicolaou. The work was supported by a studentship from the Science and Engineering Research Council. Finally, I am grateful for support from the Fairisle project in my final year.

Summary

Digital network technology is now capable of supporting the bandwidth requirements of diverse applications such as voice, video and data (so called multi-service traffic). Some media, for example voice, have specific transmission requirements regarding the maximum packet delay and loss which they can tolerate. Problems arise when attempting to multiplex such traffic over a single channel. Traditional digital networks based on the Packet- (PTM) and Synchronous- (STM) Transfer Modes prove unsuitable due to their media access contention and inflexible bandwidth allocation properties respectively. The Asynchronous Transfer Mode (ATM) has been proposed as a compromise between the PTM and STM techniques. The current state of multimedia research suggests that a significant amount of multi-service traffic will be handled by computer operating systems. Unfortunately conventional operating systems are largely unsuited to such a task. This dissertation is concerned with the system organisation necessary in order to extend the benefits of ATM networking through the endpoint operating system and up to the application level. A locally developed micro-kernel, with ATM network protocol support, has been used as a testbed for the ideas presented. Practical results over prototype ATM networks, including the 512 MHz Cambridge Backbone Network, are presented.

Contents

Glossary

AAL	ATM Adaptation Layer
ANSA	Advanced Networks Systems Architecture
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CBN	Cambridge Backbone Network
CBR	Constant Bit Rate
CCITT	Comitée Consultatif International Télégraphique et Téléphonique
CDCS	Cambridge Distributed Computing System
CFR	Cambridge Fast Ring
COW	Copy On Write
DAN	Desk Area Network
DARPA	Defence Advanced Research Projects Agency
DMA	Direct Memory Access
EOF	End Of Frame
FCS	Frame Check Sequence
FCFS	First Come First Serve
FIFO	First In First Out
FIQ	Fast Interrupt reQuest
FPC	Fairisle Port Controller
FTP	File Transfer Protocol
FQ	Fair Queueing

HOL	Head Of Line
IMAC	Integrated Multimedia Applications Communications architecture
IOP	Input/Output Processor
IP	Internet Protocol
IPC	Inter-Process Communication
IPI	Inter-Processor Interrupt
IRQ	Interrupt ReQuest
ISDN	Integrated Services Digital Network
ISO	International Standards Organisation
IPL	Interrupt Priority Level
ISR	Interrupt Service Routine
LAN	Local Area Network
MAC	Media ACcess layer
MS-Access	Multi-Service media Access
MSDL	Multi-Service DataLink
MSN	Multi-Service Network
MSNA	Multi-Service Network Architecture
MSNL	Multi-Service Network Level
MSSAR	Multi-Service Segmentation And Reassembly
NAK	Negative AcKnowledgement
NCS	Network Computing System
ORL	Olivetti Research Limited
OSF	Open Software Foundation
OSI	Open System Interconnection
PDU	Protocol Data Unit
PTM	Packet Transfer Mode

QOS	Quality Of Service
RPC	Remote Procedure Call
SAP	Service Access Point
SAR	Segmentation And Reassembly
SAS	Single Address Space
SDU	Service Data Unit
SOF	Start Of Frame
STM	Synchronous Transfer Mode
TAS	Test And Set
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TLB	Translation Lookaside Buffer
VBR	Variable Bit Rate
VC	Virtual Circuit
VCI	Virtual Circuit Identifier
VPI	Virtual Path Identifier

Chapter 1

Introduction

The last decade has seen the emergence of the Integrated Services Digital Network (ISDN): a network based upon digitally switched, fixed capacity, narrowband circuits. Whilst still being only partially deployed, it is apparent that the original ISDN will be unable to support many future broadband customer requirements. The Broadband ISDN (B-ISDN) [**CCITT88**] is envisaged as an extension to the original network in order to meet such needs.

The Asynchronous Transfer Mode (ATM) is the transfer mode selected by the CCITT for implementing the B-ISDN. ATM represents a compromise between the Packet- (PTM) and Synchronous- (STM) Transfer Modes of operation. An ATM network is concerned with the transfer of fixed sized data units called *cells*. It is asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic. If ATM is to be a successful compromise then it must be able to meet the demands of both traditional STM and PTM clients. In particular STM networks offer guarantees regarding the *Quality of Service* (QOS) that a client can expect, typically in terms of bounded jitter, delay and bandwidth. For an ATM network to accept such a call requires that all network elements, particularly switching nodes, should be able to support the requested QOS for the call path and duration. The situation is complicated by the fact that an ATM network has to support calls with different QOS attributes simultaneously.

This dissertation is concerned with the design and construction of systems components, in particular endpoint and router software, for ATM networks which provide guarantees regarding QOS to users on a per call basis. The work has been carried out over a set of heterogeneous ATM network implementations, all initially developed at the University of Cambridge Computer Laboratory.

1.1 The Cambridge ATM Environment

Much of the work described in the dissertation is of a practical nature. This was made possible by the production of ATM hardware by related projects in the Computer Laboratory. In addition, Olivetti Research Limited (ORL), Cambridge, shared an active interest in ATM network and applications research. A summary of the major active projects is given below.

The Pandora project [**Hopper90**] is a joint undertaking between ORL and the Computer Laboratory. The project aim is to provide multimedia facilities integrated into the workstation environment. It uses an ATM network, the Cambridge Fast Ring (CFR) [**Hopper88**], for voice and video transport. By late 1990 Pandora workstations were deployed and in everyday use at three sites in Cambridge connected by a single CFR.

The Cambridge Backbone Network (CBN) [**Greaves90**], a slotted ring system like the CFR, was designed to interconnect CFR networks in the metropolitan area, and consequently shares the same cell format. By late 1990 a five station CBN with a 512 MHz line rate was operational, though inter-connectivity with other ATM networks had yet to be demonstrated.

The Fairisle project [**Leslie91**] is a joint undertaking between the Computer Laboratory and HP Laboratories, Bristol. The initial project aim is to build an experimental ATM network based on the Cambridge Fast Packet Switch [**Newman89**]. The network architecture is based on the Multi-Service Network Architecture (MSNA) [**McAuley90**]. The aim of MSNA is to support multi-service inter-network applications, particularly over ATM style networks. By late 1990 the first Fairisle hardware was built and being debugged.

The early Pandora infrastructure was incapable of operation outside a single CFR. Whilst this was enough for many useful applications and experiments to be performed, the benefits of inter-networking had yet to be realised. It was clear that the project would soon outgrow the original ring both in terms of aggregate bandwidth required and geographical area covered. As a result early 1990 saw the adoption of the MSN architecture by the project. Although the architecture defined an inter-networking facility there was no support for QOS. In order for Pandora applications to inter-network properly it was apparent that routers would have to be constructed that both defined and implemented QOS. The desire to have system servers handle media produced by Pandora leads to the examination, in the next section, of the systems environment then available.

1.2 The Cambridge Systems Environment

WANDA is an experimental kernel being developed at the Computer Laboratory in part by the author. It evolved from experience gained with the Amoeba operating system [**Tanenbaum87**] by the Computer Laboratory Systems Research Group. The kernel is intended primarily for use as a systems research vehicle. It is not envisaged that it should support a conventional operating system environment, e.g. UNIX, or that program development take place over WANDA.

Both kernel and user programs are built on UNIX systems using GNU C cross-compilers. Each WANDA machine contains a set of ROMs which allow it to boot a kernel image from the UNIX file system. Once the kernel is started mechanisms exist to run user applications dynamically, with the executable binary again being downloaded from UNIX.

The kernel was initially developed on the VAX architecture, in particular the DEC SRC Firefly [**Thacker87**] multiprocessor. By late 1990 it had been ported to 68000, 68020 and 68030 series VME systems and all generations of the Acorn ARM processor [**VLSI87**]. For each processor at least one of Ethernet, CFR, CBN or Fairisle type host interfaces exist.

The Advanced Network Systems Architecture Testbench [**ANSA89**], which is available for the UNIX, VMS and MS-DOS operating systems, provides support for distributed programming. In particular it is available on most Computer Laboratory UNIX machines. The Testbench has been ported by the author to run over WANDA. The SUN Remote Procedure Call (RPC) [**SUN86**] protocol is also supported over WANDA.

Recent systems research at Cambridge [**Nicolaou91**] suggested that many clients requiring QOS from the network would be software elements in a distributed computing environment. If such clients were to be satisfied then QOS would have to be extended from the network, through the endpoint operating system, to the application.

Given that the WANDA kernel already supported the MSN protocol suite (with drivers for the Ethernet, CFR and CBN already written) and that the Fairisle project had selected WANDA as the basis for their control software it was natural that it be chosen as the vehicle for research into the system requirements for handling multi-service traffic. The ability to support a sophisticated distributed computing environment on elements inside the network promised to be useful if the implementation of complex control and management algorithms proved necessary.

1.3 Outline

Chapter 2 reviews the MSN architecture and examines its relationship to the OSI reference model. The draft B-ISDN architecture is described. It is argued that results derived from experimentation with either architecture are applicable to the other. Subsequently the dissertation concerns itself primarily with the MSN architecture.

Chapter 3 begins with a description of the WANDA system architecture and philosophy. The WANDA networking architecture and the associated MSNA protocol suite implementation, prior to the incorporation of any QOS functionality, is then discussed in detail. Of particular interest is the MSNL router implementation.

Chapter 4 describes the construction of a prototype ATM network. The Cambridge ATM Backbone is used to switch voice, video and data traffic between ORL and the Computer Laboratory. The implementation is based on the work described in the previous two chapters.

Chapter 5 looks at the issues surrounding the incorporation of QOS into a network architecture. The support of multi-service traffic, with QOS guarantees, over ATM style networks is emphasised. Techniques for providing QOS in both the endpoints and routers are examined. It is shown that these techniques display a marked overlap.

Chapter 6 presents the extensions made to the WANDA interface that enable different QOS policies to be supported. The chapter concludes with a discussion of the interface implementation over the networks described in Chapter 2.

Chapter 7 examines the behaviour of the extended WANDA system when handling delay-sensitive traffic. A series of experiments is performed using the CBN and CFR networks. Proposals are made concerning the structure of future ATM network host interfaces.

Chapter 8 places the work described previously in the context of current research. Chapter 9 outlines some areas of further research necessary before building production ATM networks able to provide QOS guarantees. In addition future ATM research planned at the Computer Laboratory and ORL is outlined. Concluding remarks are presented in Chapter 10.

Chapter 2

The Multi-Service Network Architecture

The aim of the Multi-Service Network Architecture (MSNA) [**McAuley90**] is to provide a complete architecture which allows multimedia applications to use the full facilities of ATM networks in an inter-networking environment. The architecture is based upon the virtual circuit model, and so shares many characteristics with the proposed B-ISDN. This chapter describes MSNA and its realisation over a set of heterogeneous networks.

2.1 The Virtual Circuit Model

Both B-ISDN and MSNA define a virtual circuit (VC) model of operation. Each packet or cell comprises:

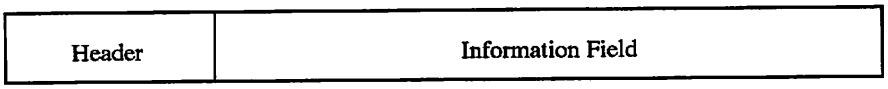

Figure 2.1: Generic virtual circuit cell format

At the very least the header contains a virtual circuit identifier (VCI) which enables the receiver to establish which circuit an incoming cell belongs to. Since a VCI is unidirectional a single hop circuit is described by a pair of VCIs. As the VCI is only interpreted locally it can be made small, an important consideration given the small sizes of ATM cells. An n-hop VC may be formed by the concatenation of n single hop VCs through $(n-1)$ VC gateways.

It is not necessarily the case that the same header encoding is used on heterogeneous networks. However, the contents of the information field must be preserved. The primary function of a VC gateway is header (i.e. VCI) re-mapping on a per cell basis, illustrated in Figure 2.2. A small VCI enables a gateway to use simple techniques, such as table lookup, when performing the translation. Likewise processing in the end system is simplified.

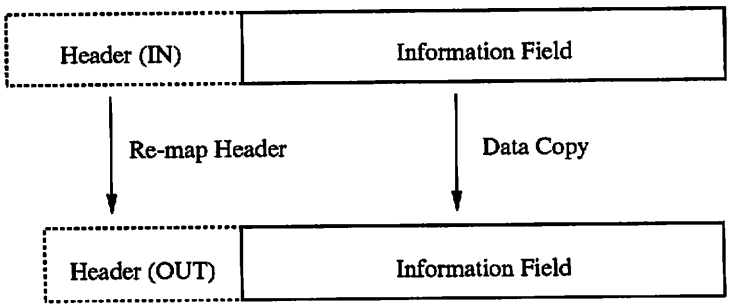

Figure 2.2: Virtual circuit gateway

The simplicity of cell forwarding makes implementation of a hardware router an attractive possibility.. The computational part (i.e. not including any queueing delays) of cell processing can be implemented in constant time and therefore a host (end system/router) can gauge its capacity effectively. This is important if it is to make guarantees concerning QOS.

2.2 The MSN Architecture

The MSN architecture defines three layers which contribute to the inter-networking service:

- **MSDL** and **MS-Access** (datalink level),
- **MSNL** (network level).

2.2.1 The MSDL Layer

The multi-service datalink layer (MSDL) defines lightweight virtual circuits, referred to as *associations*, over heterogeneous networks. MSDL addressing is in terms of VCIs. Each cell or packet transmitted is accompanied by a VCI which uniquely identifies an association in the context of the receiving MSDL entity. A VCI pair

defines an association over a particular datalink. The MS-Access layer deals with the encodings necessary to transfer MSDL service data units (SDUs) over heterogeneous networks (e.g. VCI format). The header format is guided by the characteristics of the underlying network. The MSDL SDU size bears historical dependencies on the Cambridge Fast Ring (CFR) [**Hopper88**], upon which it was first defined. Although the MSDL SDU is fixed sized, multiple SDUs (all for the same association) may be presented in a single MS-Access request. The maximum number of SDUs which can be presented is defined by the instance of MS-Access being used. Similarly an MS-Access indication may represent the arrival of several MSDL SDUs (all for the same association).

MSDL on the CFR/CBN

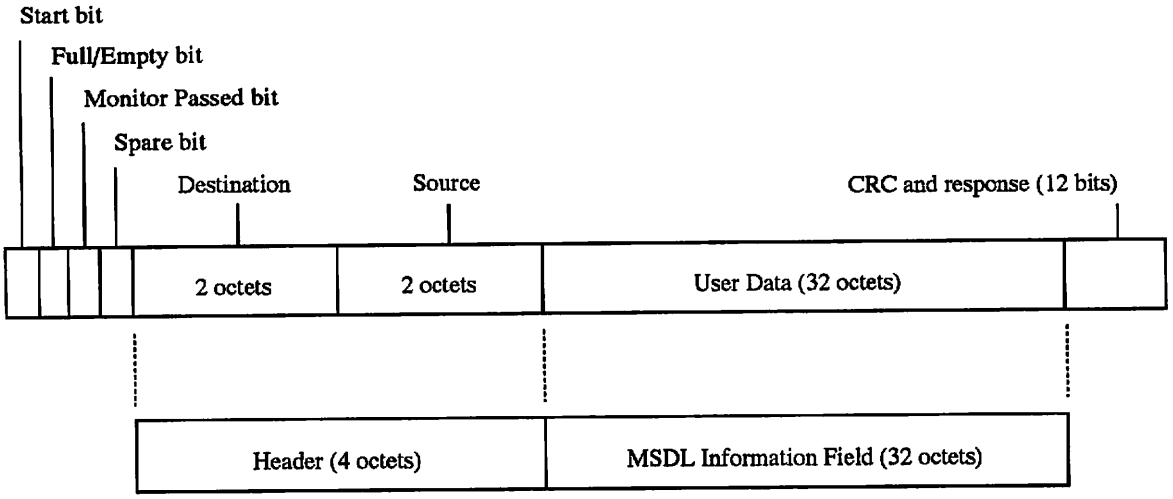

Figure 2.3: Encoding of MSDL in a CFR slot

Over the CFR the MSDL header is encoded in the Media Access (MAC) -layer source and destination addresses (Figure 2.3). The first 16 bits of the header comprise the destination VCI. At first sight this would seem to allow only one association per interface. However, when operating in *bridge mode* a CFR interface can accept cells on multiple dynamically assigned addresses. On initialisation each interface is allocated a *disjoint* portion of the MAC-layer address space from which it can assign VCIs. Intra- and inter-node multiplexing is thus achieved using the same 16 bits of address information. A maximum of 32 octets remains in the CFR slot; this becomes the MSDL SDU size. The Cambridge Backbone Network (CBN) [**Greaves90**] comprises a fibre optic ring whose bandwidth is partitioned into a number of Time Division Multiplexed (TDM) channels (currently four). Stations of varying cost and bandwidth may be constructed, parameterised by the number of channels they can use concurrently. Media access control is through the empty slot technique, with

transmitting stations able to fill *multiple* slots each ring revolution. Since the CBN has the same slot structure as the CFR, the format of Figure 2.3 was used to define MSDL on the CBN also.

CFR Interface

The CFR chip set includes an ECL repeater and CMOS station chip. The repeater chip serialises and de-serialises data from the ring so that the station chip may handle it at an appropriate speed. The station chip contains a single transmit and a single receive FIFO each with a one cell capacity. It is possible to program the station chip to generate an indication of both when the receive FIFO contains a cell and when the transmit FIFO is empty. No interface supporting Direct Memory Access (DMA) has yet been built so all data must be moved between the station chip and host memory using processor cycles.

CBN Interface

So far only a single type of CBN interface has been built, which is for the VME bus [**Greaves91**]. The interface consists of four transmit (one for each CBN channel) and one receive FIFO. Unlike the CFR each FIFO has a capacity of 256 cells. The FIFO RAM array is shared by both the receive and transmit sides of the station and so the interface is limited to a half-duplex mode of operation. The receive interrupt condition is programmable on a per VCI basis. It can be set to interrupt on every cell received or only when a cell with an end of frame bit set is received. Later versions of this interface include support for DMA.

MSDL on the Ethernet

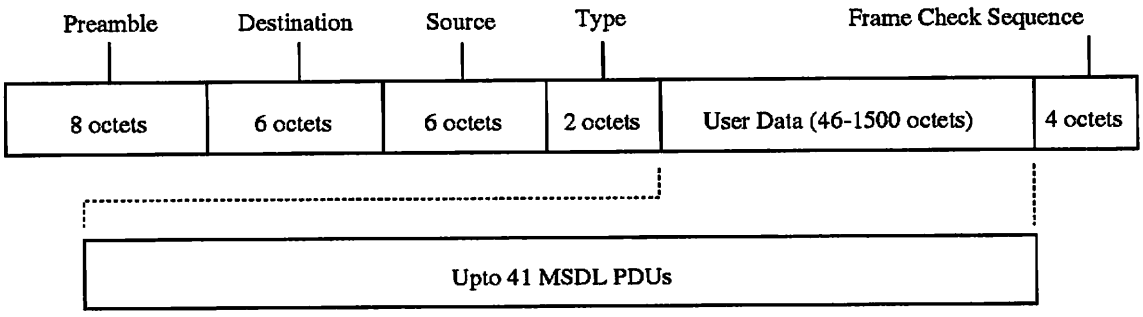

Figure 2.4: Encoding of MSDL in an Ethernet frame

The Ethernet [**Metcalfe76**] shares few characteristics of the CFR/CBN. In particular it allows variable size packet transmission. In order to use the network effectively multiple MSDL protocol data units (PDUs) must be sent in a single Ethernet packet (Figure 2.4). The peer MSDL host is identified by the 48 bit Ethernet MAC-layer address. The peer MSDL entity is identified by a 16 bit VCI (chosen for convenience to be the same size as the CFR/CBN VCI). The number of MSDL PDUs, all for the same association, can be ascertained from the length of the Ethernet packet.

Ethernet Interface

Due to the commercial success of the Ethernet much effort has been expended on producing high performance interfaces. Both the DEQNA [**DEC86**] and LANCE [**AMD85**] interfaces are used by WANDA systems in the Computer Laboratory. These interfaces use a ring of transmit and a ring of receive descriptors in main memory. Transmission and reception is done by DMA to and from the regions defined by the descriptor rings. An interrupt may be generated upon the emptying or filling of a buffer by the interface. A protocol is defined to allow concurrent access by the host and interface to the descriptor rings. By using large packets and a suitable memory system a host may achieve close to 100% utilisation of an Ethernet with one of these interfaces.

MSDL on the Fairisle network

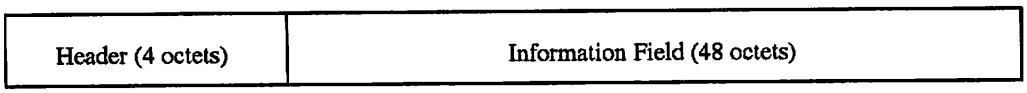

Figure 2.5: Initial Fairisle Cell Format

The Fairisle network [**Leslie91**] is an ATM network based on fast packet switching. Before a cell is injected into the switching fabric a routing tag must be prepended. Each stage in the fabric strips the first octet from the tag and forwards the cell according to the route specified in it. Upon arrival at the destination port the entire routing tag has been removed. The tag is analogous to the Ethernet MAC-layer address except that the same output port may have a different tag dependent upon which input port it is referenced from. The Fairisle design and hardware allow great flexibility in choosing the cell format. The initial layout is shown in Figure 2.5. As the information field is neither variable size nor an integral number of MSDL SDUs, the initial implementation of MS-Access over Fairisle assigns the first 32 octets of the information field to be the MSDL SDU (the remaining bytes being unassigned). The 33% loss in maximum throughput that this represents is addressed in Section 5.7.

Fairisle Interface

The Fairisle Port Controller (FPC) [**Hayter91a**] is attached to an input and output port on a Fairisle Switch Fabric. The port controller also has input and output interfaces to a transmission system. Cells received from the transmission system are buffered in the FPC cell buffer: thus a Fairisle switch is input buffered. The FPC control processor, an ARM3 [**VLSI90**], may arrange to be interrupted on receipt of a cell. The processor performs VCI re-mapping (implemented in hardware in later versions) determining which output port (if any) the cell is destined for. The cell is then linked into an appropriate transmit queue in the cell buffer. At no point does the processor need to manipulate the data portion of the cell.

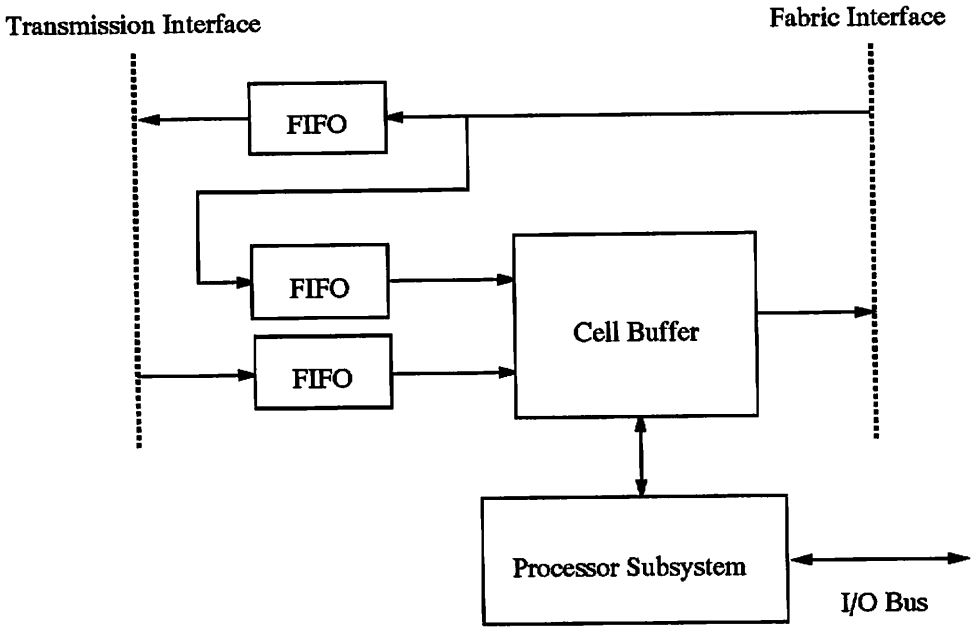

Figure 2.6: Fairisle Port Controller Schematic

Cells received from the fabric are either looped back into the cell buffer (typically for management) or passed through to the transmission system depending on a bit in the port controller portion of the routing tag.

A VME host interface has been designed and built [**Beeler91**] which attaches to the FPC transmission interface. The interface is half duplex and is only capable of VME slave operation. It supports 32 bit memory mapped accesses on a set of transmit and receive FIFOs. An interrupt is generated when data is received from the port controller and also when either of the FIFOs overrun. In an alternative mode of operation the VME interface is able to attach directly to the synchronous fabric interface.

2.2.2 The MSNL Layer

The previous sections have discussed MSNA at the datalink layer as defined by the MSDL and MS-Access layers. The multi-service network layer (MSNL) extends associations into an inter-networking environment. An MSNL *liaison* is a concatenation of MSDL associations. Liaisons are *not* multiplexed over associations. MSNL adds *no* protocol overhead to the data stream: the MSNL PDU and SDU are equivalent and defined to be the size of the MSDL SDU (32 octets). Once a liaison is established, an MSNL gateway is concerned with the forwarding of MSDL SDUs (although logically this is performed at the MSNL layer). To provide the inter-networking function MSNL defines both MSNL addresses and connection establishment procedures. An MSNL address, representing a service access point (SAP) comprises:

- a 4 octet MSNL identifier,
- a 4 octet MSNL port.

Typically there is a one-to-one mapping between an MSNL identifier and a single host. However, it is possible for several hosts to share the same MSNL identifier or, more commonly, for a single host to have multiple MSNL identifiers. An MSNL port identifier is allocated to an MSNL client either when a client registers an interest in listening for liaisons, or when it requests establishment of a liaison (Section 2.4).

2.3 B-ISDN

The ATM layer in the B-ISDN architecture provides functionality similar to that of MSNL. Figure 2.7 shows the format of the B-ISDN ATM cell. The Virtual *Circuit* Identifier above has been split in B-ISDN terminology into the Virtual Path Identifier (VPI) and Virtual *Channel* Identifier (VCI). The cell loss priority bit (CLP) if not set by the user identifies a cell which could be dropped (in preference to one which has the CLP bit set) during times of congestion. The payload type (PT) field is used to provide an indication of the information field content type. The hardware error control (HEC) field provides single- or multiple-bit error detection capabilities on the cell header.

The B-ISDN definition represents a set of interfaces; it does not specify the implementation of an ATM network. So long as a network conforms to that interface it can be part of a B-ISDN. The MSN architecture should therefore be regarded as complementing rather than competing with B-ISDN. An MSNA network should be able to carry B-ISDN traffic and vice-versa. Chapter 5, in particular Section 5.7, describes some of the issues associated with such an undertaking.

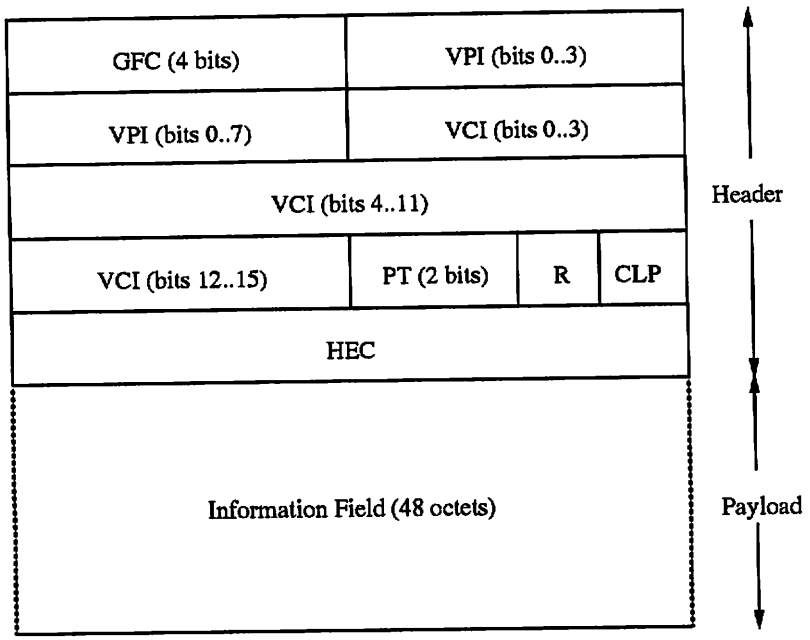

Figure 2.7: B-ISDN Cell Format

A router at the centre of an ATM network will be expected to handle the cells for a large number of connections. This does not imply that a router has to maintain state on each connection. Cells with the same source and destination, but for different circuits, may be multiplexed over a common channel. In B-ISDN this channel is termed a *virtual path*, and is the reason for the VPI in the cell header.

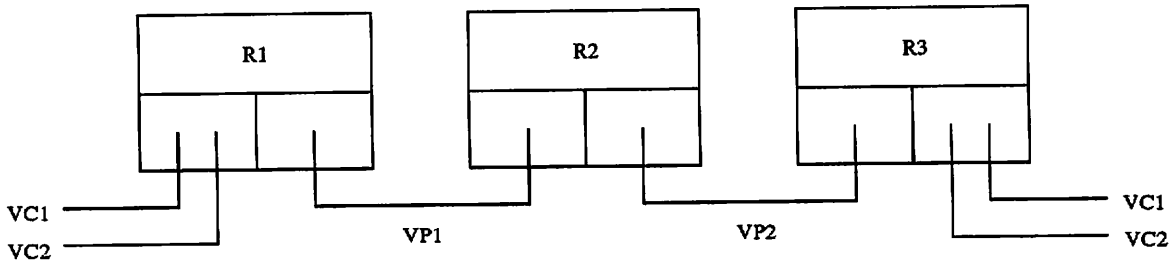

Figure 2.8: Virtual Path Call Multiplexing

The concept is illustrated in Figure 2.8. The virtual circuits VC1 and VC2 utilise a common virtual path (VP1-VP2) for part of their route (R1-R2-R3). R2 is a VPI router, it re-maps the VPI for an incoming cell and forwards it on that basis. The Virtual *Channel* Identifier is left untouched. The VP approach allows fast connection establishment as the routers internal to any VP used are effectively by-passed. It also has the benefit of simpler network management. For example, if a VP router fails then it may only be necessary to re-route each active VP rather than each

circuit. A virtual path has certain synchronisation properties since it maintains cell sequence integrity across all connections that are multiplexed over it.

2.4 Call Establishment

In a virtual circuit network dynamic call establishment requires a mechanism for VCI exchange on a hop by hop basis. The MSN architecture defines the *promiscuous* association mechanism for this purpose. A promiscuous association comprises a well known (public) VCI with which is associated a specific service. It is directly analogous to the pre-assigned virtual circuits of the B-ISDN. Any node on a network may send cells, using a promiscuous VCI, to any other node on the same network. One such promiscuous association is defined to be used for MSNL connection management.

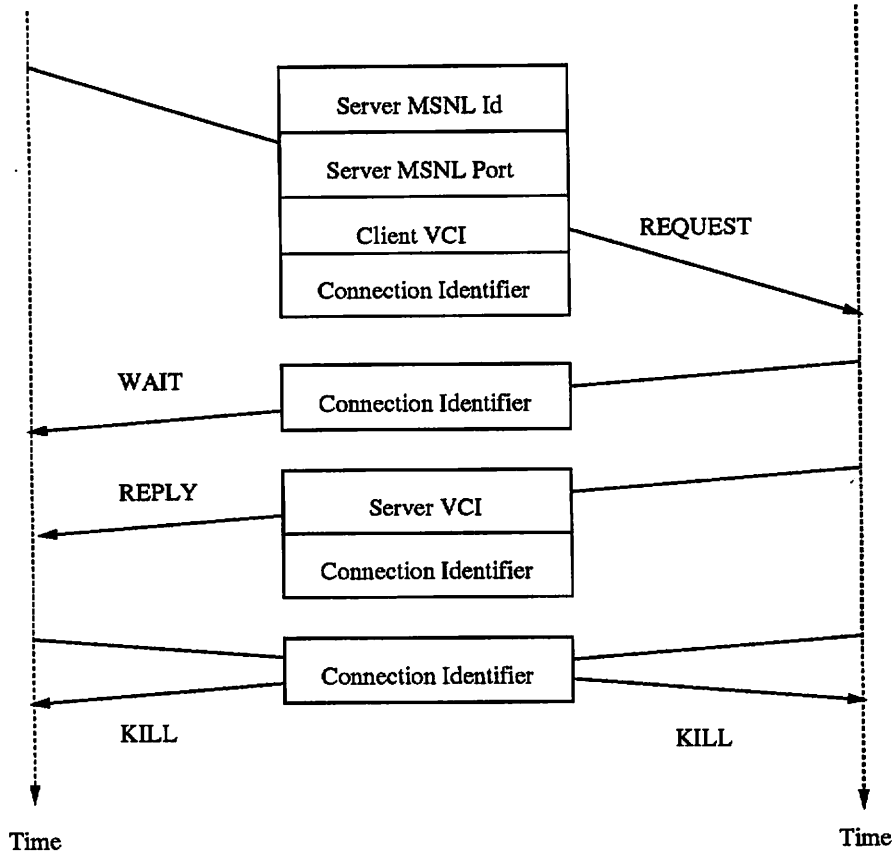

Figure 2.9: MSNL Liaison Establishment and Tear Down

The types of message used in MSNA connection management are illustrated in Figure 2.9. The terms client and server are used to refer to the connection initiator

and acceptor respectively. The protocol considerations are similar to those found in remote procedure call systems. A multi-hop connection will require the connection establishment procedure to be repeated on each intermediate network. The *wait* facility informs a retrying initiator that a connection is still in the process of being established. This is typically used by routers when setting up a multi-hop liaison.

2.5 Higher Levels

Additional functionality on top of the streamed cell interface that B-ISDN and MSNL provide is required to accommodate various services. The protocol to be used over a particular circuit is agreed upon during call establishment. Any further negotiation required is protocol dependent and in-band. Early implementations of the MSN architecture *assumed* only a single higher level protocol: the multi-service segmentation and reassembly (MSSAR) protocol. MSSAR is concerned with the transfer of variable sized blocks over MSNL liaisons and is a typical example of a higher level protocol.

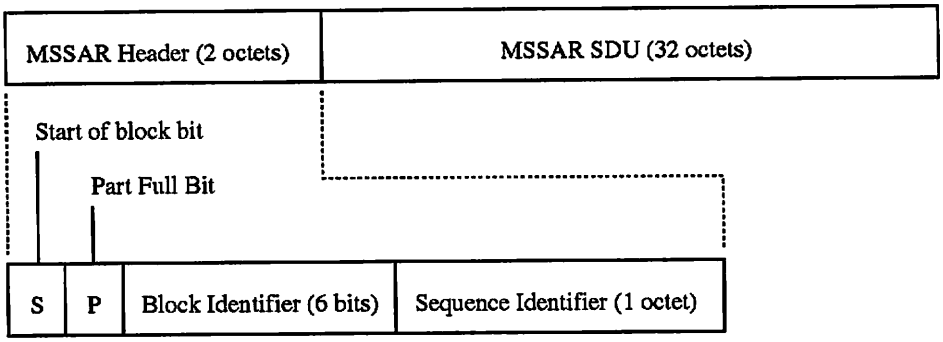

Figure 2.10: MSSAR PDU encoding

Figure 2.10 shows the encoding of the MSSAR header. The start of block bit defines the beginning of a new block. The sequence number, which counts down to 1, indicates the total number of SDUs remaining in the block. The block identifier requires at least all the cells for 63 consecutive blocks to be dropped before there is the possibility of a block being incorrectly assembled. If the block is not an integral number of MSSAR SDUs then the last header will have the part-full bit set, in which case the final octet of the MSSAR SDU indicates the number of valid SDU octets present. The B-ISDN architecture defines the ATM adaptation layer (AAL) above the ATM layer. A set of higher level protocols are defined one of which, AAL Type 3, offers similar functionality to MSSAR.

The MSSAR protocol is heavyweight considering that the MSNL interface defines that cell re-ordering or duplication does not occur on a connection. Of course an implementation cannot guarantee that such events do not occur and so any higher level protocol must be robust in their presence. Rather, their occurrence is viewed as extremely unlikely and so recovery from them should not dictate the design of any higher level protocol. A simpler block assembly protocol is illustrated in Figure 2.11.

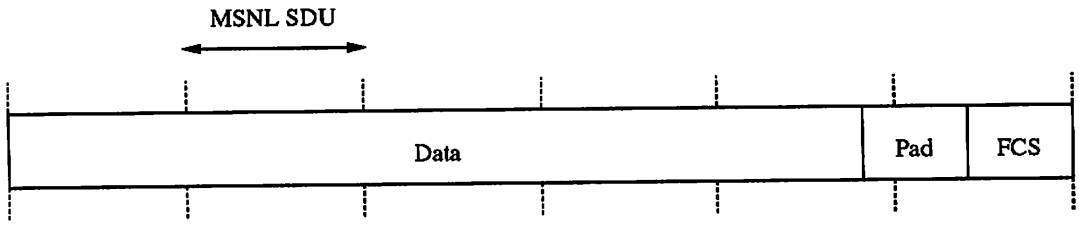

Figure 2.11: Simplified Block Assembly Protocol

A block is transmitted as a sequence of MSNL SDUs. The final cell, recognised by having an End-Of-Frame (EOF) indicator set, contains the Frame Check Sequence (FCS) field. The FCS is a check on the data octets. In most cases a lost cell will only cause the assembly of a single datagram to fail. If however the cell lost contains the EOF indicator then cells will continue to be assembled until the next EOF indicator, at which point the FCS check should fail. In this way a subsequent block although correctly received will be discarded. The protocol could be made more robust (and complex) by the addition of a Start-Of-Frame (SOF) indicator. No length field is necessary as it is believed that most higher level protocols already encode such information in the data field. If they did not then it could be added. In this way redundancy is avoided.

A protocol similar to the above has been proposed for B-ISDN AAL Type 5. The Simple and Efficient Adaptation Layer (SEAL) [**SUN91**] is a reaction to what is seen as the unnecessary complexity and overhead of AAL Type 3. SEAL requires the provision of an EOF bit, which does not currently exist, in the B-ISDN header. The proposal is to use PT 10 to signify EOF (values 00 being already assigned and 01 under consideration for other purposes). By eliminating the AAL Type 3 headers from the information field of the B-ISDN cell an additional four bytes of user data are available in each cell. This may reduce the total number of cells required per block by up to 9%. The SEAL FCS comprises a 32 bit CRC (the same computation as in IEEE 802.5). The proposal argues that the frame level checksum is more reliable than the per cell CRC present in AAL Type 3; the belief being that a single strong error check is better than a long series of weaker checks.

2.6 Relationship to OSI

The ISO Open System Interconnection (OSI) reference model [**Day83**] provides a useful *framework* in which to compare and contrast network architectures. The seven layer model is only intended as a guideline, a fact that becomes apparent when trying to describe most real architectures. An attempt is made to illustrate the relationship of the MSN architecture to OSI in Figure 2.12. The main observation to make is that no multiplexing above that at the datalink layer is necessary when using MSNA. If multiplexing at multiple layers, as allowed by the OSI model, is performed, then higher level guarantees may be compromised. A detailed discussion of the issues surrounding layered multiplexing may be found in [**Tennenhouse89**].

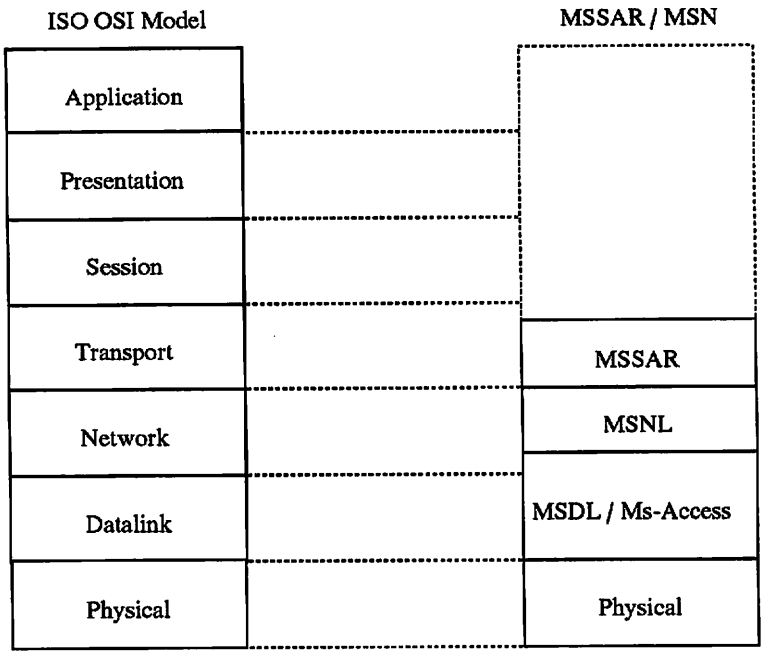

Figure 2.12: Relationship of MSNA to the OSI Reference Model

2.7 Summary

The Multi-Service Network Architecture (MSNA) has been described, and its relationship to OSI has been established. The draft B-ISDN definition has been outlined. Although differing in detail and complexity, both B-ISDN and MSNA display many similarities at and below the network level (where they are both most fully defined); experimental results derived using MSNA should be directly applicable to the B-ISDN, and vice-versa.

Chapter 3

WANDA MSNA Implementation

This chapter begins with a statement of the WANDA system philosophy and design. The features that make WANDA suitable for building high performance dedicated systems are discussed. The WANDA Inter-Process Communication (IPC) interface is then described in detail. A comparison is made with similar support available in other systems. The MSNA protocol suite has been implemented as a WANDA kernel protocol stack and support for building MSNL routers is available. An initial router supported MSSAR block forwarding in user space. The second router supports MSNL cell forwarding in kernel space. In both cases the connection management is controlled by a user space application.

3.1 WANDA

One traditional view of an operating system is as a monolithic piece of code running in a privileged mode, functions such as device drivers and the file system being part of the monolith. Inevitably, with logical separation of unrelated components un-enforced, interdependencies arise which make the system difficult to maintain and develop. Distributed systems research is concerned with the operation of physically separated, possibly heterogeneous, interconnected computers. As a result designers have been forced to decompose the single processor monolithic system into separate components which communicate via message passing. The WANDA philosophy represents a continuation of that found in the Cambridge Distributed Computing System (CDCS) [**Needham82**]. The CDCS has perhaps been the most highly decomposed distributed system attempted to date. The system comprised heterogeneous hardware and operating systems but made progress through a set of homogeneous interfaces. Note that for some applications, such as fault tolerant computing, it can be desirable that such heterogeneity exist in a single system.

Recent years have seen the widespread deployment of distributed systems technology [**SUN86**, **Scheifler86**, **Sandberg85**]. Unfortunately most such systems comprise nodes running operating system kernels whose design was dictated by the demands of time-sharing. A large amount of research [**Accetta86**, **Tanenbaum87**, **Herrmann88**] has been aimed at finding *lightweight* replacements for such kernels. Lightweight is used to refer to the smaller code size and the higher performance exhibited by such kernels. Two of the most attractive features of lightweight kernels are the relative ease with which they are ported and maintained.

There is a broad range and rapid evolution in the type and capability of processor hardware. This situation has been exacerbated by the emergence of Reduced Instruction Set Computing (RISC) [**Patterson80**]. It is important that a system be readily ported in order to take full advantage of new hardware. A smaller kernel (it is hoped that the kernel forms the basis of most of the work) helps to ensure a simpler port. The code that has to be altered per port must be well defined and kept to a minimum. Some pieces of code may be implemented in a machine independent, but less efficient, manner. In this way an initial port is made easier. Subsequently they may be re-worked in a machine dependent fashion for performance improvements. A successful example of this approach may be found in the device dependent interface of the X Window System sample server [**Angebranndt88**]. A further example is that the WANDA kernel is capable of functioning on machines without support for virtual memory (Section 3.1.4). If a quick port to a particular hardware configuration is required, then the code necessary to implement virtual memory may initially be left out.

The WANDA system was perceived as a high performance, lightweight kernel upon which system servers could be run. The WANDA programming interface is *not* a list of system calls (traps into the micro-kernel) even though a particular kernel implementation might choose to interpret it as such. For example, the user library code for a thread (Section 3.1.1) to determine its own identifier may, on a uniprocessor, simply read a well known location maintained by the micro-kernel thread switching implementation. On a shared memory multiprocessor the same code might involve a system call. Subsequent sections discuss the structure of the WANDA system as currently implemented at the University of Cambridge Computer Laboratory. Whilst not a primary aim, the WANDA interface has been designed so as not to preclude the implementation of a general purpose operating system environment. One possible route to achieving this would be the emulation of another micro-kernel based interface, such as MACH, for which an operating system environment already exists [**Golub90**].

Subsequent sections describe the support for concurrency and communication in the WANDA kernel. These are the areas most relevant to the dissertation. A detailed account of the virtual memory management system may be found in [**Mapp91**].

3.1.1 Concurrency

Distributed system servers exhibit natural concurrency due to the fact that a single server will typically handle the requests for multiple clients. Threads [**Birrell89**, **Swinehart85**] provide a mechanism for expressing this concurrency. On a uniprocessor they are a convenient structuring technique, but offer no gain in performance over a *suitable* non-threaded implementation. On a multiprocessor, where threads can be executed concurrently, they offer performance improvement compared to the non-threaded case.

A WANDA *domain*, or process, consists of a set of independently schedulable threads which all share a common address space. On a multiprocessor each thread, unless constrained, may be scheduled on any processor. Reasons for constraint might be the accessibility of a specific piece of hardware (possibly local memory) on a particular processor only. The basic synchronisation mechanism for WANDA threads is the counting semaphore [**Dijkstra68**]. The traditional *P(sem)* and *V(sem)* operations and a facility for waiting on a semaphore subject to a time out are provided. The kernel itself is multi-threaded; for example, in addition to kernel background threads, such as device driver watchdogs, the kernel must be prepared to handle concurrent system calls from multiple user level threads.

The efficiency of the thread switching and synchronisation facilities determines the grain of concurrency that an application will attempt. The WANDA thread switching implementation on a multiprocessor offers the possibility of fast synchronisation. When there are no runnable threads for a processor one approach would be to schedule a kernel "idle" thread which simply executes a dally loop with interrupts enabled. Instead, in WANDA, the last thread to block on a processor spins in the scheduler on its wake up condition. A thread that removes the blocking condition is able to wake the blocked thread by setting the memory location upon which it is spinning. There is no need to invoke a heavyweight inter-processor interrupt (IPI) operation. This technique works best if the threads for a domain are scheduled concurrently on separate processors (so called *co-scheduling* [**Ousterhout82**]). However, if there are multiple domains to be scheduled, such an approach may in fact reduce throughput due to the increased percentage of virtual memory context switches necessary upon a re-schedule.

Execution of a process on a multiprocessor is a stringent test of the concurrency correctness of the program. Although semantically the same as uniprocessor execution, subtle differences exist. In many cases the thread pre-emption interval is long enough such that a thread when scheduled will run until it blocks. Even if very fine grain time slicing is enabled most architectures will preserve instruction atomicity in the presence of interrupts (e.g. the timer). On a multiprocessor the execution stream of two threads may be interleaved at the sub-instruction level.

3.1.2 Communication

The WANDA IPC facility supports communication between threads in different domains on the same machine (Section 3.3.2) and on different machines connected by a network (Section 3.3.1). There is no reason why threads in the same domain cannot communicate with each other using the IPC facility. Typically, however, they would use a shared memory paradigm. The WANDA IPC interface (Section 3.2) is at a level unpalatable to most distributed applications programmers. A distributed computing environment, the ANSA Testbench [**ANSA89**], has been ported to run over WANDA. The Testbench can be viewed at the application level as having the following components:

- **Capsule**: the run-time system for a Testbench application, including support for threads and communications. A capsule is single instance of this run-time system.
- **IDL**: the Interface Definition Language which is used to define an ANSA interface. An associated compiler, STUBC, generates C language client and server stub code from an IDL definition. An example IDL specification is presented in Section 3.4.3.
- **PREPC**: a preprocessor which scans C language source programs for embedded DPL (Distributed Programming Language) statements. These are converted into calls on either the capsule library or the stub procedures generated by STUBC.
- **Trader**: a capsule which provides the mechanism for separate capsules in a distributed ANSA application to rendezvous. A server exports an interface reference to the trading service to make it accessible to other programs. An import operation is provided so that clients may extract interfaces from the Trader.

The provision of a distributed computing environment enables easy access to many traditional operating system components. For example, the demands placed on the file system by the WANDA applications discussed in this dissertation are relatively light (e.g. a process reading a file for configuration information). An ANSA interface to the UNIX file system provides sufficient functionality and performance.

3.1.3 Process Management

Figure 3.1 illustrates the steps involved in creating a process dynamically to run on a WANDA system. A typical WANDA machine contains a version of the kernel and a single user process in PROM. Upon reset this process will load the current boot file for the machine from a boot server using a simple transport protocol built over MSNL. The WANDA machine will initially have been booted with an image containing a kernel and single user level "loader" process. The loader is an ANSA application and, upon starting, exports an interface to the ANSA Trader. A party interested in creating a process on a WANDA machine first imports the appropriate loader service offer from the Trader.

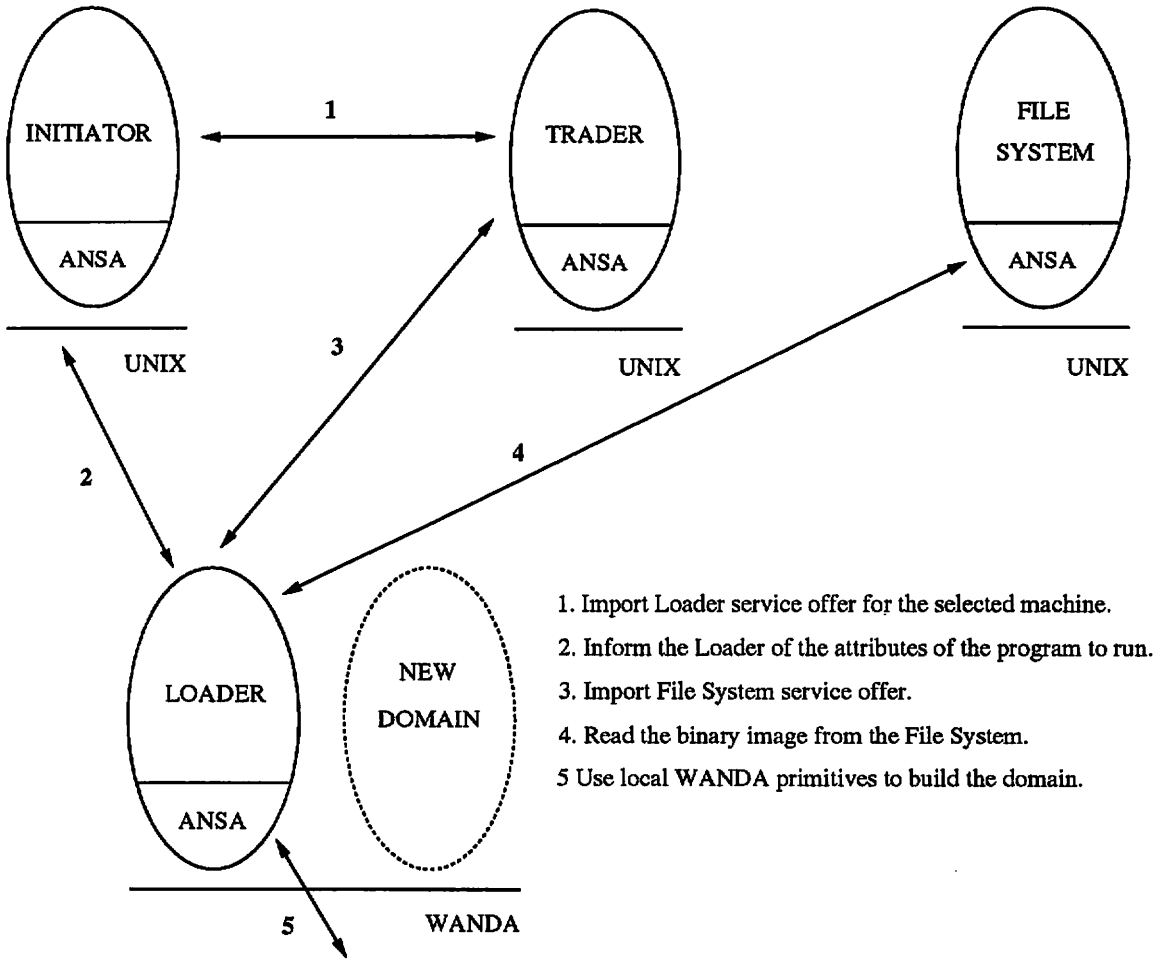

Figure 3.1: WANDA Domain Creation

The loader interface to create a process is then invoked, one of the parameters being the file name of the image to be executed. The loader imports the appropriate file system offer from the Trader and invokes it to read the contents of the executable

file. The domain is then built using privileged WANDA process creation primitives. There is nothing to prevent multiple loaders being run on the same machine. Typically this would be the case if different interfaces were being employed, e.g. one which was implemented to use SUN RPC.

The loader process assumes local responsibility for the management of the new domain. It is possible for the loader to stop the execution of a domain it has created. More typically the domain will stop due to an error, e.g. virtual memory protection violation, or natural program termination. In either case all threads for the domain are stopped in user space. This involves waking all threads for the domain in question which are blocked inside the kernel. The system call exit code contains a test to see if the current domain is stopping. Depending upon the system call that was attempted the thread stops itself with state such that if it were re-started it would either:

- return an error indication for the system call result, or
- re-attempt the system call with the same arguments.

When the domain is fully stopped the creator receives an *event* notification. Local primitives are available to query and alter the state (e.g. thread registers or memory locations) of the stopped domain. The ability to stop a process and query the state enables implementation of the following techniques to be considered:

- remote debugging [**Cooper88**, **Redell88**],
- process migration [**Powell83**, **Theimer85**, **Zayas87**].

It is not suggested that process migration be transparent at the system call level. Rather an application that is re-started on a different machine would experience errors upon trying to use kernel level resources which it had allocated on its original machine. If it is a properly coded application then it will handle such errors by re-trying at a higher level. However, the usefulness of a process migration facility for the types of application considered for implementation over WANDA is dubious. This is because many server instances display an affinity for a single machine, e.g. a file server for the disks on a particular machine.

3.1.4 Building Dedicated Systems

The WANDA system is designed primarily as a substrate for dedicated high performance servers. When writing an application it should be possible to develop much

of the code in a traditional program development environment, such as UNIX. The next phase is to complete the development and debugging on a WANDA machine that provides as much aid to the programmer as is feasible; examples include:

- virtual memory protection from others and yourself,
- comprehensive kernel checking of user arguments.

Once the developer is happy with the application it can be installed. Many of the above kernel features, desirable during development, now represent runtime overhead for the service. By removing them performance will be improved. This is analogous to turning on compiler optimisation and procedural inlining when building a program for installation. In a kernel which completely disables the memory management unit (and so is capable of running on a machine without one), termed a single address space (SAS) system, further improvements are possible. The overhead of using traps for users to link with the kernel can be avoided by using simple procedure calls. Upon initialisation the package containing assembler stub routines for trapping into the kernel would query the kernel for their procedural entry point replacements. Such an arrangement requires that user space programs run in kernel mode. Savings, depending upon the architecture involved, can include:

- no copying of arguments from user to kernel stack,
- system call entry dispatching eliminated,
- no switching between user and kernel mode.

In a similar fashion it is possible to translate same-machine cross domain remote procedure calls into simple procedure calls. This is similar to Lightweight RPC (LRPC) [**Bershad90**] except there is no need for passage through the kernel to change virtual memory context. Of course, with memory management disabled, another sink of performance when crossing domain boundaries, Translation Lookaside Buffer (TLB) faults, are eliminated. The penalty to be paid is the loss of virtual memory. This is not a major problem since most dedicated systems will have been written to avoid paging. However some system software techniques, such as asynchronous garbage collection [**Ellis88**], depend upon virtual memory being enabled. None of the applications described in this dissertation are so constrained.

3.2 The WANDA IPC Interface

The WANDA IPC interface is based upon the Berkeley UNIX socket abstraction [**Leffler89**]. In particular the need to support multiple protocol domains is recog-

nised. The important operations are listed below:

```
buffer:   WandaIOBuffer
socket:   WandaIPCDescriptor
address:  WandaIPCAddr

socket := WandaIPCSocket(PROTOCOL_TYPE)

WandaIPCBind(socket, address)

WandaIPCConnect(socket, address, timeout)

WandaIPCOffer(socket, timeout)

WandaIPCAccept(socket)

WandaIPCSend(socket, buffer)

buffer := WandaIPCRecv(socket, timeout)
```

All of the above operations are synchronous. As the interface is multi-threaded there is no requirement for the equivalent of the UNIX *select* operation. Incoming connections are established in a two phase process. Firstly, a thread registers an interest in receiving a connection request by invoking the *Offer* interface. A matching connection request will result in a successful return for the call. The connection is not yet established. The application has available information concerning the embryonic connection e.g. the peer endpoint address. This information is typically used to aid deciding whether or not to *Accept* the connection. Some call establishment protocols, such as the one for MSNL (Section 2.4), provide mechanisms which may be used to prevent the connection request timing out whilst this decision is being made.

The interesting part of the interface, that concerning buffer management, is elaborated in Section 3.3.3. Lightweight kernel philosophy [**Tanenbaum87**, **Cheriton88**] is to have a single IPC facility within the kernel. Additional protocols are implemented in user mode and accessed by interested applications via the kernel-provided protocol. Whilst acceptable for many applications, optimal performance for a particular protocol will always be provided by a kernel implementation. The main performance limitation in supporting multiple protocols in this way is the indirection involved in per socket control structures during the send and receive operations. This is minimal and could always be removed by the addition of a protocol specific interface alongside the generic one.

3.3 O/S Support For IPC

Distributed systems cannot exist without communication; its efficiency is one of the major factors determining the performance of the overall system. Many applications find difficulty in realising even a small fraction of the available network bandwidth. It has typically been thrown away by a combination of the host-network interface, system bus, communications protocol and user/kernel networking interface employed. The dominant factor affecting IPC performance is usually the cost of data copying. Approaches to minimising the cost are discussed below. Two classes of IPC can be distinguished, moving data between user space domains across a network and moving data between user space domains on the same machine.

3.3.1 Inter-Machine Communication

Many network interfaces contain DMA hardware which works on contiguous regions of physical memory. Application data however is presented as a linear region of virtual memory which in the general case will not be physically contiguous. If the DMA hardware supports chaining of data then it is possible to avoid copying into a contiguous buffer although at the cost of some software complexity. In a paging system the real pages which comprise the user's message must be marked unpageable whilst the operation is in progress, and unmarked when the transfer completes. The above complexity has forced many systems, in particular UNIX, to copy user data into intermediate kernel buffers before presenting it to the network interface for transmission.

Other systems, such as Amoeba [**Tanenbaum87**], rely on the backing of contiguous segments of virtual memory by contiguous equivalent arrays of real memory, and the absence of paging, in order for network drivers to access user virtual memory easily. Of course this requires that the user be informed when the transfer operation is completed so the application can re-use the virtual memory. Making the interface synchronous with respect to the transfer or providing some form of upcall into the application both represent additional overhead. Yet another technique, found in the V system [**Cheriton88**], is to treat small messages as special and arrange for them to be passed into the kernel using registers. This yields limited speedup over copying and introduces a test of message length into the main line code. Finally, a technique used in the Topaz environment [**Thacker87**, **Schroeder89**], is to map (with both read and write access) a set of physically contiguous buffers into the address space of each user domain. A lock, shared between the kernel and all users, controls buffer acquisition and release. The problem associated with this interface is that malicious or erroneous clients can overwrite data in buffers allocated to other users, including the kernel.

3.3.2 Intra-Machine Communication

Trends in structuring distributed systems are making the performance of intra-node IPC increasingly important. Consider the case of a domain dedicated to maintaining the client side cache of a remote file system. All file system requests from any domain on that node will go through the domain containing the cache. If the cache is functioning correctly most requests will be satisfied without the need for an inter-node operation.

When moving data between domains on the same machine several techniques are available. One method is to copy from the sending to the receiving domain. This is non-trivial as the two regions will be in different virtual memory contexts on most architectures. Many implementations copy into an intermediate kernel buffer. When the receiver is woken it copies data from the kernel buffer into its address space. An alternative is to arrange that all physical pages on a machine are always accessible using a fixed range of kernel virtual addresses. Whoever performs the copy, sender or receiver, accesses their own region with normal user space virtual addresses. To access the other region the relevant physical addresses are determined which may be accessed in the current context using the appropriate kernel virtual addresses. As the user page tables no longer protect the copier from page-outs a certain amount of care must be taken when performing this operation.

A second method is to use a virtual memory technique termed copy on write (COW) [**Trevanian87**, **Abrossimov89**]. The physical pages representing the message in the sender's address space are marked read only. The physical pages representing the destination region in the receiving domain are replaced by the sender's physical pages marked read only. If either side subsequently attempts to write one of these pages, generating an access control violation, it is replaced by a different physical page which is a copy of the original. Changing the virtual memory management tables is an expensive operation. In addition to the computation involved the TLB must be flushed. This impacts the performance of the processor due to increased subsequent TLB faults. On a shared memory multiprocessor the TLBs of all processors must be flushed. Unfortunately the same thought that has gone into cache coherence protocols for the memory system has not been directed at the TLBs. On most architectures flushing the TLBs requires an IPI to each processor.

3.3.3 The WANDA I/O Buffer System

The WANDA I/O buffer system is similar to the DEC SRC Topaz scheme (Section 3.3.1). A stylised version of the interface is given below:

```
buffer: WandaIOBuffer

buffer := WandaIOBufferAcquire(length, timeout)

WandaIOBufferRelease(buffer)
```

An I/O buffer is realised as a contiguous region of physical memory; it is uniquely identified by an easily verifiable handle. The attributes of each buffer, such as address and length, are described by a per-buffer structure. In a secure implementation of the interface, an allocated I/O buffer is mapped with read and write access into the acquiring domain's address space. If a domain attempts to access an I/O buffer it does not own then it will address fault.

In addition to the above interface an I/O buffer may be acquired upon receive and released upon send. Thus, the semantics of the IPC and I/O buffer interface dictate that, once sent, data in a buffer is no longer accessible. This means that a client wishing to maintain a copy of any transmitted data, e.g. for re-transmission, must perform a copying operation before transmission; a kernel space copy has been swapped for one in user space. The situation is not so bleak when it is realised that other stages are usually required in the preparation of data for transmission e.g marshalling or encryption. The marshalling code in a remote procedure call runtime system could arrange to marshall directly from client arguments into an I/O buffer. If re-transmission is necessary the arguments are re-marshalled; a small penalty considering this action is only necessary if a time-out occurs. Note that this re-marshalling could be initiated before the time-out if the application has no other work to perform. The important point is that intelligent clients have the opportunity to avoid copying. Not all implementations will adhere to the strict semantics of the interface. In particular, a secure machine supporting dedicated servers may choose to neglect all virtual memory operations. However, client code which depends on this is illegal. Other techniques for building such kernels were discussed in Section 3.1.4.

3.4 MSN Architecture Implementation

In an inter-network environment it is important to have rich connectivity [**Braden87**], this implies many networks having multiple gateways. In the initial WANDA MSNA implementation there was no code that enabled a WANDA machine to act as an MSNL gateway. The requirement for such functionality became apparent with the desire to place machines on the first Pandora CFR.

While the MSN architecture has been designed such that gateways can be built in

hardware it is not the case that all will be. New networks are more quickly supported in software. Not all routes are performance critical, e.g. one used for a mail feed only. Therefore solving the problem with a cheap general purpose processor can be more cost effective than using dedicated hardware. Such software gateways would be configured with downgraded quality of service attributes so that they are only used in response to failure or overload conditions on the primary routes. Even in a hardware gateway it is not envisaged that the hardware understand much more than header re-mapping and simple cell queueing. All protocol control packets would be handled in software and the hardware instructed when a liaison is established or terminated.

The use of a connection oriented protocol enables optimisations to be made at connection set up time. The code to implement intra-node MSNL quickly diverges from that implementing the inter-node case. The dispatch vectors for send and receive in each socket control block are accordingly adjusted when a connection is established.

3.4.1 MSSAR User Space Gateway

One approach would have been to put the gateway code into the kernel. This was rejected on the grounds of:

- implementation complexity,
- difficulty of debugging and maintenance,
- limited configurability.

A technique for enabling protocol implementation in user level processes, the *packet filter*, is described in [**Mogul87**]. However, moving functionality out of the kernel is not a panacea. The design described below attempts to balance the requirement for high performance with that of moving complex code into user space.

The MSNA user space interface already offered the ability to accept incoming and establish outgoing connections. A *wildcard* MSNL port was defined upon which connection requests for specific destination networks could be accepted. A connection so established has the liaison initiator as the peer address and the *intended* recipient as the local address of the socket. The gateway process then attempts to establish a connection to the intended recipient, making it look as if the connection request came from the initiator. Data received on the former socket should be forwarded on the latter, and vice-versa. As the receive interface is blocking, two threads will be

required to listen for incoming packets and data received on one socket is forwarded on the other. The code for this is extremely simple:

```
in, out: WandaIPCSocket

forever do
    WandaIPCSend(out, WandaIPCRecv(in))
end
except: error handling logic
```

Note that no lookup is required to determine the incoming or outgoing socket indices, it is all part of the thread state. The overhead is in the scheduler thread switching implementation but this is just one piece of code and a candidate for optimisation. No user space synchronisation is required *except* in handling exceptional error conditions, where the two threads have to agree whose responsibility it is to free user level resources associated with the connection, primarily the two sockets. Consider the code to achieve a similar function over an asynchronous, single threaded interface such as that found in UNIX. Upon reaching the idle condition the process would block inside the kernel, using the select system call, with a list of *all* I/O descriptors it was interested in. Upon return, for all descriptors that are indicated as having a receive pending, the process makes a receive system call, looks up the outgoing I/O descriptor and does a send system call.

Having a gateway in user space is not cheap in terms of system resources. Each through connection requires two threads and two sockets to be allocated. A typical gateway will consume hundreds of threads although, as can be seen, their user space stack demands are not great. If kernel data structures are not extensible the gateway will have to be configured with large defaults or be forced to refuse connections when not necessarily heavily loaded. The WANDA kernel implementation described here provides extensible thread and socket tables.

When receiving notification of an incoming connection the gateway can adopt one of two strategies. In the first, an incoming connection is not accepted *until* the outgoing one is established. If all gateways follow this strategy then there is a full round trip delay before the initiator can transmit on the liaison. In the second strategy, an incoming connection is *optimistically* accepted before the outgoing one is established. The initiator may begin transmitting after the round trip delay between it and the first hop gateway. The danger here is that the data will catch up with the connection establishment request, what happens in such a case is implementation dependent (data may be discarded or buffered). Data transmitted over a multi-hop optimistic liaison can arrive at the destination in the same order of time as if it had been sent over a datagram network. It is feasible to envisage a liaison being established, used and then deleted by the initiator before the destination received any indication.

Performance measurements for the user space gateway yielded an interesting result. It would be expected that the ping delay (time to send a cell and receive a return reply) across a gateway would be the sum of the ping times between the endpoints and the gateway. For the case involving no router a scheduler optimisation ensures that the source processor is idling in the context of the receiver thread when the reply returns. Therefore, no context switch is necessary. In the gateway case the same optimisation ensures that the gateway processor is typically in the context of the thread that received the cell from the source. A context switch is necessary resulting in the ping time through a gateway being longer than might be predicted. In the case of a blast protocol traversing a user space gateway the scheduler optimisation ensures that very little thread switching occurs. Such interactions are an example of the difficulties encountered when attempting to extend QOS from the network up through the endpoint operating system to a user level application.

3.4.2 MSNL Cell Forwarding

The user space gateway described above suffered from two problems. Firstly, as the WANDA MSNA user interface only dealt with the transmission and reception of MSSAR blocks the gateway was an MSSAR, and not MSNL, router. Secondly, although adequate for Ethernet type interfaces (large packet size and DMA) the user space implementation would have severe performance and/or jitter problems when forwarding cells between ATM networks. In addition, if QOS was to be provided in user space then real-time support would have to be added to the WANDA thread and IPC systems. As a result of these considerations it was decided to add support for MSNL cell forwarding in the kernel with forwarding taking place in network driver interrupt routines. Connection management was still maintained in user space. The IPC interface was extended so that two established liaisons could be joined together:

```
a, b: WandaIPCSocket

WandaIPCJoin(a, b) except: error handling logic
```

Incoming cells on liaison a are forwarded on liaison b, and vice-versa. Forwarding of MSNL cells in the kernel is relatively straightforward. The easiest situation is when forwarding between homogeneous networks. A single incoming CFR cell maps onto a single outgoing CFR cell. A single incoming Ethernet packet (possibly a collection of cells) maps onto a single outgoing Ethernet packet. The same ease applies to forwarding between similar networks such as the CFR and CBN.

Forwarding from an Ethernet to a CFR/CBN involves invoking the cell forwarding machinery once for each cell in the Ethernet packet. Forwarding cells in the oppo-

site direction is more complicated. Assigning each incoming ATM cell to a single Ethernet packet would exhibit miserable performance. Assigning multiple cells to an Ethernet packet raises the problem of when to forward a partially filled Ethernet packet. Such a decision is dependent upon the higher level protocol involved, of which an MSNL router should be unaware. However fixed bits in the MSDL header are allowed to be set by higher level protocols and may be interpreted by routers. One of these is the EOF bit (Section 2.5) which is used to trigger forwarding. It is interesting to note that the B-ISDN cell header (Section 2.3) includes no such facility. Two threads are still required in user space to wait for either side of the through connection to close (and take the appropriate action).

3.4.3 ANSA Interface

The implementation of gateway management in user space enabled a trivial ANSA control interface to be provided (there is no RPC implementation within the WANDA kernel).

```
Router : INTERFACE =

BEGIN

    MSNLAddr : TYPE = RECORD [host, port: CARDINAL];

    MSNLConn : TYPE = RECORD [client, server: MSNLAddr];

    ConnList : TYPE = SEQUENCE OF MSNLConn;

    ListConnections : OPERATION [ ] RETURNS [ConnList];

    DeleteConnections : OPERATION [ConnList] RETURNS [ ];

END.
```

Figure 3.2: ANSA Interface to MSNL Router

The initial interface to the router (Figure 3.2) simply supported a query on the state of all active connections. In addition, it allowed a client to close a connection(s) remotely. It is expected that the majority of network management software will be built using an RPC paradigm.

3.5 Summary

Some of the features which make WANDA a suitable vehicle for building high performance distributed systems components have been discussed. These are:

- easily portable (SAS systems),
- concurrency through threads,
- multiprocessor operation,
- high speed networking (I/O buffer system),
- ability to configure stream-lined versions.

Kernel techniques for the support of high speed networking have been discussed within the framework of the WANDA IPC interface. The WANDA MSNA implementation prior to incorporation of any QOS related functions has been described. The MSNL router connection management code has been implemented in user space without impacting data forwarding performance. In addition to ease of programming, a major benefit is the availability of a sophisticated distributed computing environment (ANSA) at this level.

Chapter 4

The Cambridge ATM Backbone

This chapter describes the Cambridge ATM Backbone network which has been built using infrastructure described earlier in the dissertation. At the time of writing this network is used to transport voice, video and data traffic between two sites in the Cambridge area.

4.1 Pandora Interworking

Pandora [**Hopper90**] is a joint project between Olivetti Research Limited (ORL), Cambridge and the University of Cambridge Computer Laboratory. The project is investigating the use of multimedia workstations in a *working* environment with particular emphasis on digital video. A Pandora workstation (see Figure 4.1) comprises a standard workstation to which has been attached a multimedia peripheral, called *Pandora's Box*. The control interface to the box enables most application and management software to run on the workstation. The X11 window system [**Scheifler86**] is used (with protocol extensions) to provide a programming interface to the video capabilities of the Pandora Box.

The Pandora Box acts as a switch between a set of multimedia input and output devices. The handling of video presents the greatest technical challenge. A video stream is typically sourced from the local camera device or network (CFR). Video stream sinks are typically the workstation monitor or the network. An analogue mixer, under workstation control, is used to display video on the workstation monitor. The network is used to provide a real-time communication path for voice and video streams between boxes. Pandora network protocols are built as an adaptation layer over MSNL virtual circuits. Both video and voice streams are constant bit rate. Popular applications include video-phone and video-mail. A video file server

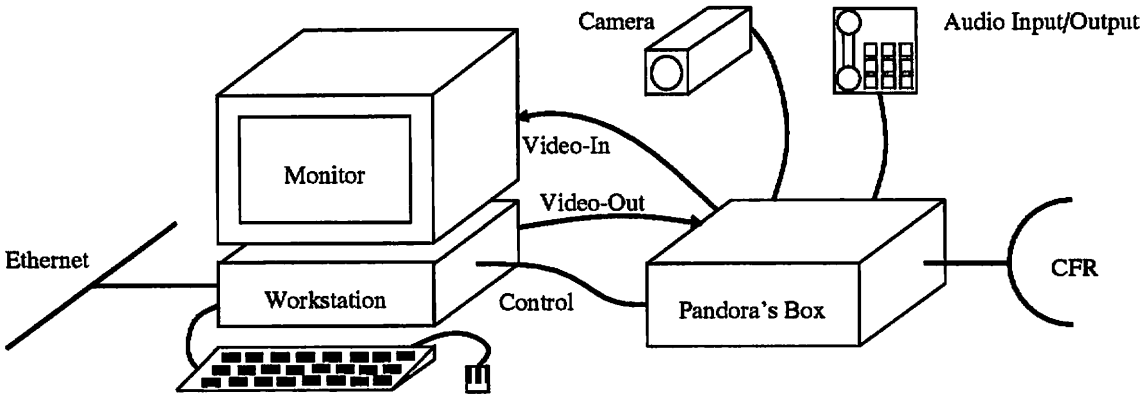

Figure 4.1: The Pandora Multimedia Workstation

is accessible over the CFR.

The initial network configuration comprised a single CFR, running between the two sites, to which all Pandora boxes were attached. Such a configuration has well known drawbacks:

- a single network failure can disable both sites,
- many management decisions are not local,
- ring bandwidth limits the number of active boxes,
- a single CFR has geographical limitations.

In response to these problems the CFR was split into two rings, one at each site. These were connected using a single CBN. The initial MSNA router hardware comprised one 68030 processor, with a single CFR and CBN interface, at each site. While solving the problems described above it has introduced new considerations concerning component placement. A video file server is required to handle a larger number of concurrent streams than any of the workstations it serves. The fairness properties of the CFR prevent it from acquiring the "unfair" share of the bandwidth it might justifiably desire. In practice the situation is worse due to the poor performance of most CFR network interfaces. Solutions can involve adding more than one interface to a particular file server or having multiple file server instances. The CBN exhibits properties more suitable for the support of a video file server. By moving the video file server to the CBN the threshold above which such solutions have to be considered is raised.

4.2 IP/MSNL Protocol Gateway

In addition to MSNA interworking between CFRs, the Pandora system also required Internet Protocol (IP) [**Postel81a**] connectivity between the Pandora host workstation Ethernets. Three options for connecting the Pandora Ethernets over the backbone were identified. Firstly, an IP gateway could be built that fragmented IP datagrams over ATM cells. This has the disadvantage that most of the information field of each cell would be consumed by the large IP header. In addition IP routers would be required within the ATM network. Secondly, all IP applications could have been rewritten to use MSNL. This is impractical in the amount of effort required and because for many applications the source code is not available.

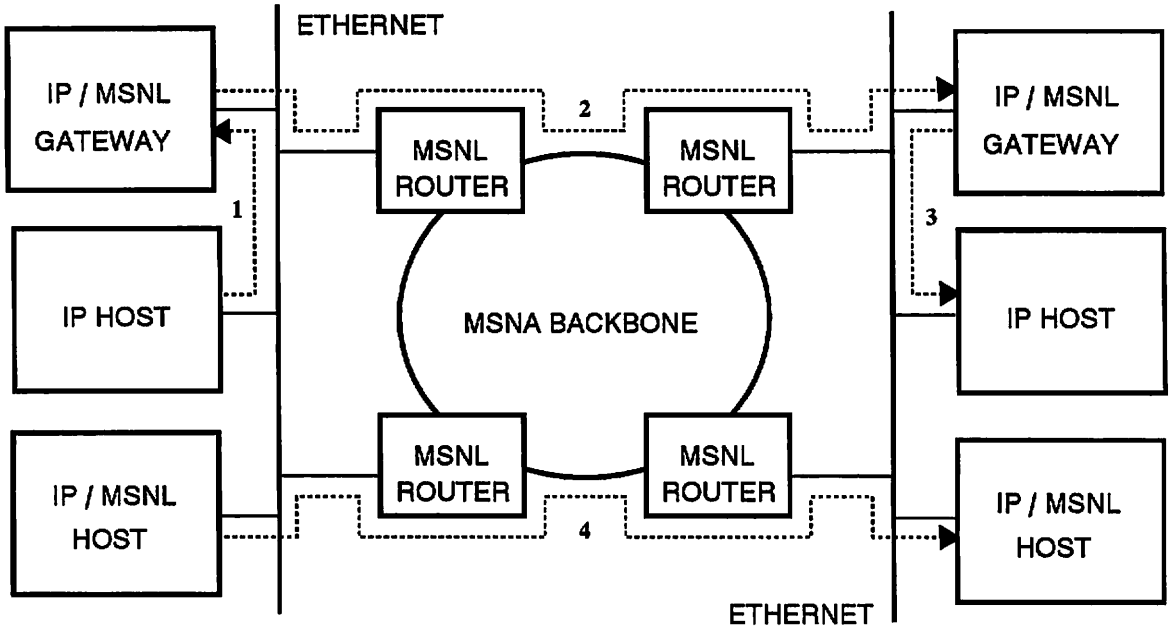

Figure 4.2: Sample IP over MSNL Configurations

Finally, a protocol gateway between IP and MSNL could be built. This was the option selected and is illustrated in Figure 4.2. An incoming IP datagram (1) is forwarded over an MSNL circuit (2) to a peer protocol gateway. If a circuit between the two gateways does not exist then it must be established. The peer gateway will then forward the datagram (3) as appropriate. It is possible to envisage several configurations of switching elements. If each Ethernet had a single IP/MSNL gateway then all IP traffic from one Ethernet to another could use a single virtual circuit. The gateway might attach to the Backbone directly, or use some other mechanism such as an Ethernet/Backbone MSNL router. Any Ethernet host with both IP and MSNL implementations is capable of routing its own IP packets onto outgoing MSNL circuits (4). This involves fewer switching elements between endpoints.

Upon opening an MSNL circuit for carrying IP traffic between sites a QOS should be supplied[1]. This is difficult as IP provides only *limited* information as to what sort of service is expected. A QOS equal to the most stringent required by Ethernet traffic could be allocated. This would be expensive in terms of Backbone resources and certainly undesirable if the Backbone was a charging network. Work on dynamic bandwidth management [**Harita91**] carried out within the Unison project [**Tennenhouse87**] would apply directly here. Control on the capacity allocated between two sites could be accomplished by either QOS re-negotiation, opening and closing of multiple circuits or a combination of both.

The IP/MSNL gateway was implemented[2] within the UNIX (specifically Ultrix) kernel. Use was made of the existing IP and MSNL implementations and support for IP gatewaying. In addition to presenting a socket interface the MSNL code was altered to register itself as another IP datalink interface. System administration was possible using existing UNIX network management tools. The MSNA router hardware comprised one 68030 processor, with a single CBN and Ethernet interface, at each site. A large number of Ethernets may be switched using a single CBN, particularly if it is true that most traffic stays local to a single Ethernet.

4.3 XNS/MSNL Router

Although a large percentage of traffic between ORL and the Computer Laboratory consisted of IP datagrams a significant fraction also comprised Xerox Network Systems (XNS) [**Xerox81**] datagrams. In order to achieve XNS inter-operability a MAC layer forwarding solution was adopted[3]. Each XNS router instance was implemented as a single WANDA application domain. A WANDA socket interface was available supporting the transmission and reception of "raw" Ethernet packets. This enabled the XNS router domain to promiscuously receive all XNS type packets on a specific Ethernet interface. By inspecting the source and destination Ethernet addresses of the packet the backbone XNS routers are able to construct a database mapping Ethernet addresses to an XNS router instance. Packets to be forwarded are exchanged by XNS routers over an MSNL connection. This type of bridging is limited in the number of nodes it can support due to the need to propagate broadcast packets to all co-operating networks. Each XNS router node presently comprises a 68030 processor with an Ethernet and CBN interface. This configuration is able to handle the promiscuous receptions from the Computer Laboratory main Ethernet of over 250 nodes.

---

[1] As yet not implemented.

[2] Implemented by Simon Kelley (Computer Laboratory).

[3] Implemented by Derek McAuley (Computer Laboratory).

4.4 Limitations

Whilst sufficient for most voice and data applications the software routers were of limited use in handling video streams. Only two concurrent Pandora video phone conversations (four voice and four data streams) could be supported with sufficient quality. The bottleneck was determined to be the cost of software data copy when forwarding from the CBN to the CFR (note the forwarding costs are asymmetric). However, a faster CPU would soon have run into the limitations of the CFR interface cf. video file server (Section 4.1). It is apparent that the bottleneck in modern networks is the switching nodes and not the transmission medium. The aggregate bandwidth in and out of a subnet may be increased by adding more routers. This adds no complication to the routing function which in any realistic implementation must cope with multiple routes to the same destination. For large network configurations the exchange of routing information becomes a serious sink of network and router resources. It would seem that lessons learnt from the telephone system [**Ash90**] would be applicable here.

4.5 Pandora Media through UNIX

The availability of a UNIX MSNL socket interface enabled the processing of Pandora media at the UNIX application level to be considered. Since none of the Computer Laboratory UNIX machines currently have CFR interfaces such media must be received on the Ethernet after passage through at least one router. A simple UNIX application[4] was written to display Pandora video using the bitmap painting primitives of the X protocol. This is contrasted with the hardware approach typically associated with the display of Pandora video (Section 4.1). Using the remote display capabilities of the X protocol it was possible to display video on workstations throughout the Internet with varying degrees of success. The problems experienced included:

- **network congestion**: The Computer Laboratory Ethernet is prone to periods of high activity when the probability of packet loss due to collisions becomes significant. Similarly, the bandwidth available to other sites depends on the capacity of often highly utilised links and routers.
- **X transport protocol**: An X client communicates with an X server using a protocol which requires to operate over a reliable byte stream. In many instances this is provided by the Transmission Control Protocol (TCP) [**Postel81b**]. Such a protocol is unsuitable for the transmission of real-time

[4]Implemented by Tim Glauert (ORL) and Timothy Roscoe (Computer Laboratory).

video paint requests. A lost packet (which requires re-transmission) will cause the entire stream to hang. One solution would be to define a separate unreliable channel for such requests. The User Datagram Protocol (UDP) [**Postel80**] would be a suitable substrate.

- **UNIX scheduling**: The UNIX scheduling algorithm meant that the application (or the X server) receiving Pandora video could, in the presence of competition for the CPU, be de-scheduled for appreciable intervals (in practice up to several seconds). This is enough to cause an observable discontinuity in the display of video on the workstation.

If the UNIX application was running on the same host to which the destination display was attached then it attempted to paint using the shared memory extension [**Corbet91**] to the X protocol. In this manner an otherwise unloaded DEC 3100 workstation running Ultrix 3.1 was able to display a Pandora video stream at 25 frames per second. The result of this simple experiment was to highlight the flexibility of handling multi-service traffic in software and the unsuitability of current operating systems and networks for such an undertaking (other than for demonstration purposes).

4.6 Summary

The feasibility of switching multi-service traffic over ATM networks has been demonstrated. The Ethernets and CFRs at the Computer Laboratory and ORL are linked using a single CBN. Traffic exchanged between the two sites includes:

- Pandora voice,
- Pandora video,
- XNS datagrams,
- IP datagrams.

Of particular interest was the ease with which IP was able to be implemented over MSNL circuits. For the length of the MSNL connection no IP switching is required. MSNL routers enjoy an inherent performance advantage over their IP counterparts [**McAuley90**]. There is a point where the cost of the IP/MSNL gatewaying function is outweighed by the switching time saved in the network.

Chapter 5

Quality of Service Issues

This chapter discusses some of the issues involved in providing performance guarantees to multi-service traffic. Some of the more important issues concern:

- architectural model i.e. virtual circuit or datagram based,
- granularity of guarantees,
- QOS specification parameters,
- implementation over ATM networks.

5.1 Introduction

Most communication systems are not designed to handle all the traffic that could possibly be offered. There is contention for resources between the endpoints. The telephone network has a switching capacity far below that necessary to support simultaneous trunk communication between all its customers. The network reacts to overload by refusing connections, probably after it has tried various congestion control schemes such as dynamic call routing [**Ash89**]. Note that congestion does not produce a degradation in the service experienced by existing connections. Unfortunately the STM technology upon which the current telephone network is based is ill suited to the demands of multi-service traffic.

A packet switched network, such as the DARPA Internet [**Postel81a**], has very different characteristics. There is no concept of a call at the network switching level. An endpoint may inject traffic, addressed to any other endpoint, into the network constrained only by the speed of its interface. Congestion can occur at

network routers, links and endpoints. No guarantee can be made regarding the service experienced by a particular packet. This fact permeates upwards to the application level so that successfully carrying traffic with performance guarantees is impossible in the general case. One particular example of an attempt to carry real-time traffic over a packet switched network is the Etherphone [**Swinehart83**] project. By using a dedicated Ethernet and having management software limit the number of concurrent calls each active endpoint receives satisfactory service.

A virtual circuit based network experiences both types of congestion found in STM and PTM networks. Firstly, a host on a virtual circuit based network is architecturally limited to accepting only as many connections as defined by the VCI field in the cell header. This is analogous to switches in the telephone network. By making the VCI large enough this form of congestion is typically not an issue, at least not for end systems. Secondly, each host has a limited forwarding and buffering capacity. If a source is unconstrained in the traffic it may send then congestion analogous to that experienced in a packet switched network can occur. By constraining the offered traffic, as in the Etherphone system, this form of congestion may be avoided.

The necessary constraint may be achieved by forcing each endpoint to describe the nature of its offered traffic using a QOS specification. The virtual circuit is the natural unit with which to associate a QOS. It is already the unit in which clients request network resources. It is the unit for which state is maintained in the endpoints and routers. Associating a QOS with each packet or cell sent on a virtual circuit is infeasible in the general case as the network has no *a priori* knowledge of the offered traffic. In addition to the size of the cell header, processing overhead would also be increased. A limited form of QOS may however be attached on a per cell basis. An example of this is the cell loss priority (CLP) bit in the B-ISDN cell header (Section 2.3).

As part of connection establishment, in addition to addressing information, a QOS specification must be supplied. The routing mechanism must reconcile these two parameters with current network load and topology. Each host examines the requested QOS. A connection request may be refused if its associated traffic would cause the QOS of any previously established connection to be potentially unsustainable. This process is termed *admission control* and is discussed in Section 5.3.2. Alternatively a currently established circuit with sufficiently "low" priority may be deleted to provide enough resources for the new request. The connection establishment process should return as a result the QOS of the connection established. This is not necessarily the same as that requested. A large number of applications will use the returned QOS to alter their subsequent behaviour e.g. initialise re-transmission timers, set transmission rate. Once a connection is established the network must ensure that the QOS contract is not broken by the user. This function is termed *traffic control*, or *policing*, and is discussed in Section 5.3.3.

5.2 Traffic Characterisation

The QOS requested by an application is dependent upon its traffic characteristics. A large amount of work has been done in establishing mathematical models to describe various types of traffic. A good survey may be found in [**Jungok91**]. The suitability of some of these models for describing real traffic has been contradicted by experimental data [**Caceres91**]. Therefore, only a general discussion of the features common to all such models is attempted here. At its source a stream of cells may be described in terms of:

- bandwidth,
- burstiness.

Bandwidth defines the maximum number of cells that may be sent in a particular time interval. Burstiness is a measure of how the cells sent are distributed in time. Table 5.1 shows the relationship between mean and peak service bit rates for some representative multi-service traffic examples.

Service Class	Coding	Transfer Rate (bps)		Example Service
		Mean	Peak	
Standard Audio	CBR	64K	64K	Telephony
	CBR	106K	106K	Pandora Audio [1]
	VBR	12K	24K	Telephony
High Quality Audio	CBR	2M	2M	HI-FI Distribution
Low Quality Video	CBR	2M	2M	Videophone
	CBR	1.56M	1.56M	Pandora Video [2]
Standard Quality Video	CBR	34M	34M	TV Distribution
	VBR	10M	34M	TV Distribution
High Quality Video	CBR	140M	140M	HDTV Distribution
	VBR	70M	140M	HDTV Distribution
Low Speed Data	VBR	-	64Kbps	Remote Login
	VBR	0.3K	64K	Teletex
High Speed Data	VBR	-	>10M	LAN Interconnection
	VBR	2M	10M	CAD/CAM

Table 5.1: Characteristics of Sample Multi-Service Traffic

[1] 8K frames/sec (40% framing overhead).

[2] 25 frames/sec at 256x240 resolution (2% framing overhead).

A distinction is made between constant- (CBR) and variable- (VBR) bit rate sources. A CBR source will submit a single cell for transmission a fixed time after the last cell was transmitted - any stream not sharing this attribute is termed VBR. An example of CBR traffic is an uncompressed video stream; a VBR example would be its compressed (depending upon the type of compression used) counterpart. There is a wide range in the burstiness displayed by VBR streams. For example, a voice stream with silence suppression shares all the characteristics of a CBR source except that it does not necessarily transmit a cell on *every* interval expiration. The traffic profile of a stream may be altered upon passage through the network. A QOS specification will include parameters which specify the limit of any such alteration. Three values of particular importance to multi-service traffic are:

- delay,
- jitter,
- cell loss.

The end-to-end delay is a combination of transmission, forwarding and queueing delays. The transmission and forwarding components may be fixed (as is the case for WANDA MSNA cell routing) and depend on the route between endpoints. The queueing component is affected by concurrent activity in the network. Jitter is a measure of the variation in cell delay time for a particular connection. The cell loss parameter specifies a probability that a particular cell will be dropped by the network. Different applications display a wide variation in their delay, jitter and cell loss requirements. Real-time streams, such as voice and video, often require tight jitter specifications and are tolerant of a limited amount of cell loss. If two way personal communication is involved then the round trip time is also significant. In general the greater the amount of compression the more sensitive a stream is to cell loss. Most protocols involving reliable data transfer will be affected by cell loss as any data lost must be re-transmitted. The cost involved in this case is dependent upon the round trip time for the connection. Some reliable data transfer applications, such as an interactive terminal session, may have a delay bound on successful transfer.

This dissertation is not concerned with the details of QOS specification but rather the mechanisms that are required for its support. At the application level there will be a rich specification language which allows clients to express their requirements independent of a particular network protocol or architecture [**Nicolaou91**]. The selection of the attributes that are used to parameterise QOS should be such that all higher level protocols are able to express their requirements adequately. This specification is transformed into an appropriate network level description for the protocol and architecture selected.

5.3 ATM Considerations

The MSN architecture (and indeed B-ISDN) is defined independently of any particular switching or transmission technique. Although oriented towards ATM style networks, implementation over different network types is not precluded. For instance it is likely that a sizeable fraction of multi-service connection requests will be for telephone grade service. There is no reason why such traffic should not be carried on STM based networks internally.

5.3.1 Statistical Multiplexing

The discussion concerning traffic characteristics has established the statistical nature of multi-service traffic. A fundamental result is that the peak-to-average bandwidth ratio is greater than one for many traffic instances. If link capacity is allocated such that the sum over all connections of the peak input rates does not exceed the output link rate then high bandwidth utilisation may not be achieved. A good example is the support of telephony type services. In one STM implementation each circuit is allocated a 64Kbps channel. This is not fully utilised due to the silence periods in normal conversation [**Bellamy82**].

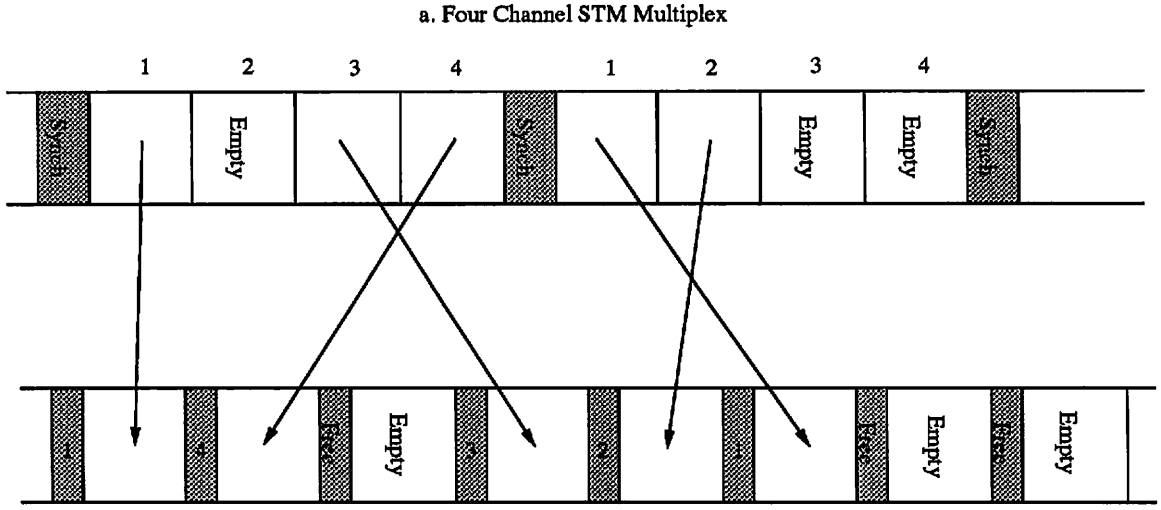

Figure 5.1: Comparison of STM and ATM multiplexing

An ATM implementation might multiplex 4 such channels on a link with a capacity of only 3 x 64Kbps. This is illustrated in Figure 5.1. By assuming that the traffic

flows in a network are independent of each other then statistical techniques can be applied to predict, within confidence limits, the behaviour of the network as a whole. Bounds for the delay, jitter and cell loss associated with a particular stream may be calculated. The larger the number of streams, the greater the significance that can be attached to such bounds. For the example, if the probability of any particular voice channel being in silence is 0.25 then the output link is capable of supporting the *average* bandwidth requirement. However, statistical variations will mean that the instantaneous load may sometimes exceed the outgoing link capacity. Depending on the duration of the overload and the amount of buffering, traffic may be lost. This technique of multiplexing streams, dependent on their traffic distributions, is referred to as *statistical multiplexing.* Examples of its application to ATM traffic streams may be found in [**Maglaris87**, **Dittmann88**, **Hirano89**]. The assumption of traffic flow independence is not wholly valid. One example being a multicast facility implemented using separate streams. In this case we must adjust our first assumption to state that the dependencies between flows are not significant. This is reasonable when considering the behaviour of a large number of streams.

5.3.2 Admission Control

Admission control is defined to be the process of accepting or rejecting a connection establishment request based upon the associated QOS and the current network utilisation. There is a tradeoff between the bandwidth utilisation gain and the increased complexity in the call admission and cell processing algorithms which must ensure that the QOS for a circuit is satisfied. Consider an endpoint or router which processes cells in strict FIFO order. The simplicity of the scheme precludes the acceptance of calls with strict delay requirements in the presence of bursty traffic. Consider a system which defines two classes of circuit, specified at connection establishment. The cells from the high priority class are processed in preference to low priority cells. Delay-sensitive traffic assigned to the high priority class may be isolated from traffic assigned to the low priority class. Within delay-sensitive traffic some streams may have different delay requirements. In such cases more than two priority classes may be required to achieve efficient statistical multiplexing. A survey of admission control schemes may be found in [**Jungok91**].

For many applications the QOS requested from the network is negotiable. Whilst the application may *prefer* to be given the QOS it requested it could adapt to operate with one of lower grade. The ability to specify a range of acceptable service characteristics should be a part of any QOS interface. The actual QOS achieved is returned as the result of successful connection establishment.

5.3.3 Traffic Control

So far the source has been viewed as a benign entity which promises not to exceed the QOS agreed at connection establishment. If the client breaks the agreement then the extra traffic may cause the QOS guarantees made to other clients to be broken. In addition to being caused by a malicious client the excess traffic may be generated by a software or hardware fault. This situation may be prevented by bandwidth enforcement, often termed policing, at the user-network boundary. The policing function is aware of the QOS which has been negotiated with the network. A violating cell is defined to be any which causes the agreed QOS envelope to be exceeded. Such cells may either be dropped at the network access point or specially marked. It is argued that if the network approaches congestion due to an influx of marked cells then by dropping such cells the situation may be retrieved.

5.4 Charging

The full benefits of a multi-service network may only be realised if users select a QOS appropriate for the application involved when requesting a circuit be established. In a private network, with no charging, the pressure to make a responsible selection comes out of a sense of peer co-operation. In a public network charging is the mechanism used to force clients to be responsible in making their selection. Any realistic charging strategy may be divided into call establishment, holding and usage components. There must be a holding charge as a circuit consumes resources in the sense that its existence may cause denial of service to other establishment requests. The holding charge will be dependent on the QOS of the allocated circuit. In addition to the holding charge there may be a usage charge. A study of charging in multi-service networks is made in [**Cocchi91**].

It is likely that in moving to a charging environment the communications demands of many applications will be reviewed. Consider an application such as file transfer implemented in a charging environment. In an initial phase the size of transfer and endpoint throughput limits are determined. If the user requires the transfer to complete in minimum time then a channel consisting of the minimum of the two endpoint throughput limits is requested. The size of the transfer and bandwidth of the channel allocated dictate whether the channel delay significantly affects the total transfer time. If the transfer is to take place over a charge network then performance may be traded for reduced cost. An application such as FTP, when asking the network for a channel with a particular QOS, is able to specify with reasonable accuracy a bound on the duration of the call if that QOS were granted. Such information could be useful to the network which may offer charge discounts for its provision.

File transfer is an application whose bandwidth requirements are readily predicted. Many other types of computer traffic do not share this characteristic, a popular example being remote interactive terminal access. In a charging environment it may be economical to close down the underlying communication channel during periods of inactivity and open it up again on demand. Modern workstations are breeding grounds for idle connections; a typical user will have many remote windows open of which only a fraction will be concurrently active. Shutting down inactive circuits is desirable even in a non-charging environment. Since the network can only support a finite number of circuits, holding open an inactive circuit may deny service to other network users. Of course when re-opening a closed circuit there exists the possibility that the request will be refused due to current network loading.

5.5 Simplex or Duplex

Traditional data communication virtual circuits implement a duplex channel, one popular example being the Transmission Control Protocol (TCP) [**Postel81b**] from the DARPA Internet suite. However, in practice, the application traffic demands made on a connection are not always the same in each direction [**Caceres91**]. In some cases they will be radically different, for example, a connection used to *retrieve* images from a video file server. The fundamental unit provided by the network should therefore be a simplex circuit with a single associated QOS. The abstraction of a duplex connection (comprising two simplex connections) may still be easily provided. In many situations this abstraction simplifies application programming. By decreasing the granularity of requests to the network, overall performance may improve because the management function has been given greater flexibility. For example, a duplex connection may require resources beyond the remnant capacity of any single router. However, when split into its simplex components the stream may be allocated across two routers.

With connections defined to be simplex it is possible to construct gateways which are also simplex. In some cases such gateways enjoy a throughput advantage over their duplex counterparts. Consider forwarding from one interrupt driven interface to another. It is possible for the processor to dally in the receiver interface waiting for incoming cells to forward. Due to less interrupt entry and exit overhead the throughput of the router may be increased. If cells are incoming in both directions then dallying can lead to increased jitter and cell loss, especially if the interface is not multiply buffered (as is the case with the CFR station chip). Of course, multiple receive interfaces could be polled but this leads to extra delay and complexity on the critical path of the cell forwarding code. In many instances a single duplex gateway will suffice between two networks. However, if more bandwidth is required it may prove desirable to install multiple simplex gateways.

5.6 Resource Allocation

For each connection multiple components will contribute to the overall QOS. When a connection is established each component agrees to place a limit on the amount it will alter the characteristics of the stream. This process is referred to as QOS negotiation and is discussed in [**Ferrari90, Nicolaou91**]. A zero hop connection is that between two hosts attached to the same network and thus involving no router. When an attempt is made to set up such a connection the state of both endpoints and the network may need to be examined to see if the requested QOS can be granted. Some networks, such as the CFR, guarantee a maximum access delay and hence minimum bandwidth available to any node. In such cases global state on network resource allocation need not be maintained, although this may lead to a request being rejected gratuitously. For a connection requiring multiple routers each will contribute a share to achieve the overall QOS. In many situations dividing the load equally may not be the optimum approach for the network. By giving one router freedom to introduce a larger delay component than others its capacity to accept subsequent connections is higher than it would have been otherwise. This may be important if it is already heavily loaded or if some higher level mechanism knows this will be the case in the near future. Time dependent routing based on predicted demand is already a component of the telephone system [**Ash90**].

5.7 Heterogeneity Issues

Independent ATM implementations may define different payload sizes. For example, both the CFR and CBN have a cell payload of 32 octets, whereas the B-ISDN and Fairisle define a payload of 48 octets. Several approaches can be taken in providing a virtual circuit which crosses such heterogeneous networks. The first option is to define the cell size for a connection to be the minimum of all the networks making up the connection path. This requires that the terminal equipment at either end of the connection, and routers, be capable of handling streams of different cell size i.e. only a certain fraction of the interface payload field is defined to be useful. A second option is to use the maximum cell size of the circuit's constituent networks. In this case network routers may have to implement some form of Segmentation-and-Reassembly (SAR) protocol when carrying cells over a network with a smaller cell size. This is undesirable as the ATM philosophy is to remove such protocol overhead from the network routers. The optimistic establishment technique (Section 3.4.1) breaks down in an internet with constituent networks of differing cell size. The cell size on the initiating network may not be the minimum for the entire connection. One solution is to declare a global minimum cell size (in the same style as the minimum IP fragment size) which is used on all optimistic connections.

5.8 Synchronisation

A multi-service network will be expected to carry a large amount of jitter-sensitive traffic. A typical ATM network implementation will introduce jitter into a submitted cell stream. The receiver endpoint is complicated by requiring extra buffering to remove the jitter introduced. By specifying tight QOS constraints on the jitter associated with a particular stream it is possible to reduce the amount of buffering required at the receiver. However, the tighter the constraints, the greater the cost in network resources and hence any associated charging function.

One approach being investigated at the Computer Laboratory [**Sreenan90**] is for such QOS constraints to be relaxed beyond the ability of the sink to cope with the resultant jitter. A dedicated *synchronisation server* is placed between the incoming stream and the sink in order to remove the jitter. A high grade QOS channel is used by the synchronisation server in forwarding cells to the sink. Since this channel is on the local network it is not subject to internet charging[3]. As the non-local network is not, in effect, being asked to perform full stream synchronisation, it is able to offer a higher aggregate service. There is additional traffic in and out of the synchronisation server on the sink network but this should not have inter-network ramifications. The total buffer requirements on the receiver network may also be reduced as the synchronisation server can be shared by multiple clients.

Some applications require the creation of multiple related streams. An example would be reading video mail where the video and voice are transmitted using separate circuits. In addition to each stream requiring jitter correction the streams must be synchronised with respect to each other in order to achieve "lip synch". Overall synchronisation is achieved by trading tight source synchronisation and QOS constraints for receiver complexity. An extended synchronisation server, which copes with multiple streams, may be used to smooth and synchronise the arriving streams on the sink network.

The Virtual Path (VP) concept discussed in Section 2.3 may, in some cases, aid the provision of synchronisation between streams. Since the cell sequence integrity is maintained across all circuits in a VP, circuits which are synchronised upon entry to a VP will be synchronised on output. A similar result may be achieved if the network provides a *call bundling* mechanism. If two connections are in the same bundle then they traverse the same set of routers. Each router maintains cell sequence integrity between all the circuits for each bundle. This can be simply achieved by assigning each of the circuits in a bundle to a common forwarding queue. Call bundling imposes additional constraints on the network, thus a higher blocking factor for such requests will be exhibited.

[3]It is assumed that local networks charge either little or not at all.

5.9 Summary

This chapter has examined some of the issues involved in providing guarantees when handling multi-service traffic, some of the main conclusions are that:

- the virtual circuit model with a QOS specified at connection request provides a suitable framework for the support of multi-service traffic,
- the network primitive should be a simplex virtual circuit; this is not necessarily the abstraction presented to an application programmer,
- statistical multiplexing is necessary in order to achieve high utilisation of ATM networks,
- admission control and policing are the mechanisms by which the network protects itself from congestion and malicious or errant clients respectively,
- the connection establishment procedure provides a convenient time to reconcile the difficulties of communication over heterogeneous networks.

Chapter 6

Quality of Service Extensions

This chapter is a description of the extensions, driven by the issues detailed in the previous chapter, that have been added to WANDA in order to support QOS in the MSNA protocol suite. Many algorithms for providing QOS have been proposed in the literature. For some [**Mankin90**], an implementation exists and practical results are available. For others [**Demers90**, **Zhang90**], only simulation or analytic results have been produced. The approach taken is not to choose or propose a *single* algorithm and implement it. Instead, a common set of mechanisms is identified which may be used to implement a variety of cell handling strategies. The extensions discussed are those which require *kernel* modifications and so particular emphasis is given to:

- queueing mechanisms,
- buffer management,
- policing mechanisms,
- statistics gathering.

By having a flexible interface it is possible to compare different strategies for providing QOS in a common environment. Issues of implementation complexity, not considered in many of the proposed algorithms, may now be considered. A host may easily be configured to reflect high level management decisions. For example, in the case of a router, the forwarding strategy to provide a traditional LAN-LAN service is different from that required in order to provide equal service across all clients.

6.1 Queueing Mechanisms

A cell which is to be transmitted, subject to buffer availability (Section 6.2), is added to an output queue (whilst preserving per virtual circuit cell sequence integrity). When a transmission slot becomes available a cell is selected from an appropriate queue (still preserving per virtual circuit cell sequence integrity). There is a tradeoff between queue insertion and removal complexity. All of the queueing strategies described below have analogies in operating system processor scheduling. An ATM cell processor experiences many of the conflicting goals of a processor scheduler. Different queueing strategies will fulfill, to some extent, a certain subset of these goals. Processor scheduling is a mature subject [**Coffman73**] and it is hoped that by drawing such analogies lessons learnt from that field may be applied. Different levels of scheduling can be identified. Low level processor scheduling may be compared to the per cell decisions made by an ATM router. High level processor scheduling (job initiation) is analogous to the call admission function. Intermediate level scheduling refers to the adjustment of a job's scheduling attributes in response to fluctuations in system load.

6.1.1 First Come First Served (FCFS)

The FCFS policy maintains a single transmit queue; cells are added to the end of the queue in order of request. When a transmission slot becomes available the cell from the head of the queue is sent. Often referred to as First In First Out (FIFO), the FCFS policy is easy to implement and is found in many PTM network routers. Due to its implementation simplicity FCFS exhibits the best performance in terms of cells processed per second when the bottleneck is the CPU. However, an outgoing cell may be subject to a large queueing delay variation and so a pure FIFO scheme is limited in providing QOS. A comparison can be made with the poor response times that can be experienced by interactive users with a FIFO CPU scheduler. Poor response times have been traded for increased system throughput due to the elimination of all unnecessary context switching.

In a network which provides the appropriate indication of transmission failure (such as Fairisle or the CFR), a failed cell may be re-transmitted. Reasons for transmission failure can include:

- full destination receive FIFO (CFR/Fairisle),
- switch fabric contention (Fairisle),
- fault in the destination interface hardware or software.

Cells which are queued behind the one that failed will be held up until it is successfully re-transmitted. This phenomenon is referred to as Head-Of-Line (HOL) blocking. Several techniques may be used to alleviate this problem. Firstly, a retry limit may be associated with each cell. If the limit expires then the cell is discarded. Secondly, instead of retrying the same cell straight away a different cell is selected. Care must be taken that it is not for the same circuit as the failed one, in order to preserve the cell ordering invariant. The usual approach is to search in preference for a cell which is for a different destination.

6.1.2 Fair Queueing (FQ)

A Fair Queueing (FQ) [**Nagle87, Demers90**] scheme allocates a separate queue for each virtual circuit. Depending upon buffer availability an outgoing cell is added to the end of its respective queue. The virtual circuit queues are themselves ordered by, for example, circuit creation time. When a transmission slot becomes available a cell is taken from the next non-empty queue succeeding the one from which the last cell was transmitted. A source sending cells too quickly only increases the length of its own queue. Note, however, that in the case of a router, other cells are delayed as the incoming errant cells consume router network interface bandwidth and processing cycles.

The FQ strategy is ideally suited to handling a number of streams each with *identical* QOS requirements, e.g. traditional STM traffic. It is not suitable for situations in which some streams require priority over others. A comparison can be made with Round Robin (RR) CPU scheduling. Interactive users will experience longer response times as the number of compute-bound processes in the system is increased. Note however that, depending on the number of jobs in the system, there is a bound on the response time. The FQ strategy easily lends itself to techniques which alleviate HOL blocking. The next cell to send after a failure is selected from the head of the next queue (checking for the same destination is again an implementation option). All multiple queue schemes share this property.

6.1.3 Weighted Queues

The FQ scheme is constrained in the sense that each queue receives an equal and fixed number of transmission opportunities per unit time when all queues are fully occupied. By arranging that selected queues receive multiple transmission opportunities relative to others some of the limitations of FQ when handling multi-service traffic are removed. A comparison can be made to a round robin scheduler which allocates disparate time slices. A high priority job is allocated a large time slice.

However, starvation cannot occur as a low priority process is guaranteed to receive some CPU time after a fixed interval.

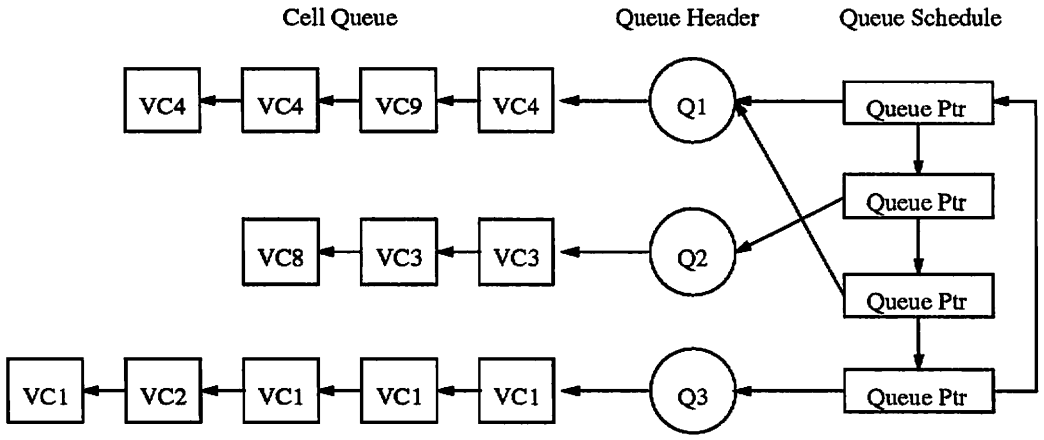

Figure 6.1: Weighted Queue Assignment Schedule

Figure 6.1 shows a weighted queue assignment schedule for a three queue system. A mark is kept of the current position in the queue schedule. Upon a transmission slot becoming available the cell to send is given by the head of the next non-empty queue in the schedule. Note that Q1 does not receive successive transmission slots but rather they are spaced evenly over the entire schedule. Consider a situation in which cells from VC4 (allocated to Q1), VC3 (allocated to Q2) and VC1 (allocated to Q3) are always available for transmission. By separating the two transmission opportunities that Q1 receives the output traffic profile for VC4 is made smoother. In addition, the probability of having to transmit to the same destination immediately after a transmit failure is reduced.

6.1.4 Multi-Level FCFS

Upon call establishment each virtual circuit is assigned to a particular queue, each such queue has a FIFO mode of operation and is allocated a priority. When a transmission slot becomes available the cell to be sent is selected from the head of the highest priority non-empty queue. This scheme can lead to the starvation of low priority queues in the face of unconstrained high priority transmission.

6.1.5 Priority Queueing

All of the above FIFO ordered schemes may be replaced by a priority ordered equivalent. Such a mechanism requires that a cell have an associated priority. This priority may be a per virtual circuit constant, in effect merging the separate queues of the Multi-Level FCFS scheme into one. Alternatively a cell's priority may be calculated dynamically. An example of this is the VirtualClock [**Zhang90**] traffic control algorithm (Section 6.3). The insertion of a cell into the middle of a queue is not a fixed-time operation. The time taken has a dependency on the current length of the queue. By setting bounds on the length of a queue, a limit can be obtained for the insertion time.

6.1.6 WANDA Queueing Extensions

The WANDA queueing extensions attempt to provide mechanisms which subsume the above and other queueing strategies. One design aim was that the cell processing machinery be capable of making fixed-time decisions regardless of the load on the host. This, and a desire that at least a partial hardware implementation be possible, dictated simplicity in the design. The discussion which follows concerns the semantics of the interfaces provided and hints at a particular implementation.

A queue is associated with a network interface instance. Queues may be created dynamically and are referenced by a per interface unique integer handle. Also maintained with each network interface is a program which controls the selection of a particular queue to use when a transmission slot becomes available. The program consists of a set of tables each of which selects a queue to use given the global queue state. The global queue state (i.e. whether each queue is full or empty) may be described by a bitmask. When a transmission slot becomes available the queue from which to take the next cell is selected by indexing the current program table with the bitmask describing the state of all the queues. The current program table pointer is then moved on to point to the next table in the program.

A two queue system is illustrated in Figure 6.2. If both queues never reach the empty state then twice as many cells will be forwarded from Q1 as Q2. In the example given the transmission order will be VC3,VC4,VC9,VC3,VC4 provided that no new cells arrive which may get inserted ahead of those already present. The host management code, running in user space (described in Chapter 3), is supplied with interfaces which enable it to:

- create a queue associated with a specific network interface,
- change the attributes associated with a queue,

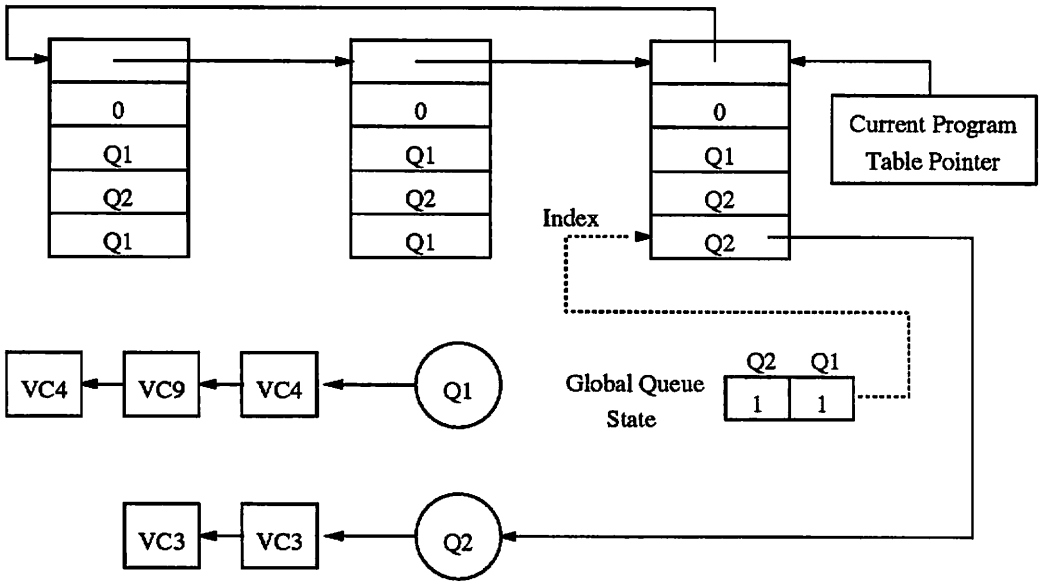

Figure 6.2: Example Two Queue Transmission Program

- assign a circuit to a particular queue,
- set the control program for an interface.

Associated with each queue is a method for inserting cells. For example, many types of traffic are susceptible to cell loss in the sense that losing one cell out of a frame makes any others received of no value. This is typically the case with data-type traffic. One type of insert procedure might be defined which upon cell loss would drop successive cells received on a circuit until the next end of frame marker. Of course this facility means that, at the very least, the transmission code is complicated by having to detect cell loss. On networks which provide a response mechanism, such as the CFR and Fairisle, detection may be done in the transmitter.

In addition, each queue has a network priority field. This is intended for use in networks which support priority. For example, each routing byte in a Fairisle cell header has a priority bit assigned. If there is contention at a switching element within the network then cells with the priority bit set take precedence over those in which it is unset.

A significant problem is that as the number of queues for an interface increases the size of the control program tables becomes prohibitive. For this reason most interface drivers in the current implementation limit the number of queues to eight. Support for more queues may be envisaged in a slightly modified interface. Only a subset of the queues are assigned to be tracked by the bitmask, say eight. The

assigned queues can be regarded as a cache of the active connections. Queues are moved between the assigned and un-assigned state according to their activity levels.

In practice the queue assignment of a virtual circuit, or queue attributes, may have to be adjusted dynamically in response to events, such as:

- connection QOS re-negotiation,
- re-balancing the host after circuit creation or deletion.

When dynamically changing the configuration care must be taken that per circuit cell sequence integrity is maintained. For example, if the insertion strategy of a queue is changed then all the cells in the queue are first discarded. If a circuit is moved from one queue to another then all the cells for that circuit are first discarded from the original queue.

6.2 Buffer Management Extensions

Buffer space is a fundamental resource of an ATM host. It is required in order to absorb peaks in the incoming and outgoing traffic streams. Buffer management is concerned with how the buffer pool is partitioned and the actions that are taken when a buffer allocation request cannot be satisfied. It is possible to use buffer allocation controls in order to achieve some form of QOS. Consider the case of a FQ scheme with unconstrained traffic sources. If one stream consistently sends at a rate above the available transmission capacity then that stream's buffer allocation requests will exhaust the pool. Cells generated on other streams will have to be discarded due to unavailable buffer space. By placing a limit on the length of a stream's associated queue such an unfair situation can be avoided.

Buffer controls in WANDA are associated with transmission queues rather than individual circuits. If required, per circuit buffer controls may be provided by mapping a circuit one-to-one with a queue. The attributes of a queue relating to buffer management are illustrated in Figure 6.3.

Each network interface driver has an associated buffer pool. This is further partitioned into a reserved and a common pool. The partition is usually affected under management control when the host is booted. Each queue has a limit (possibly infinite) to the number of reserved buffers it may contain. When this limit is reached it may request buffers from the common pool up to a total limit (again, possibly infinite).

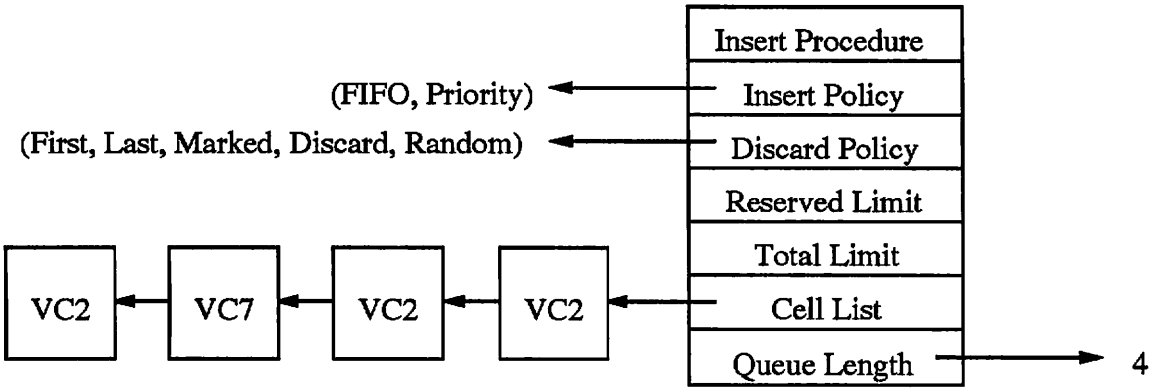

Figure 6.3: Transmit Queue Attributes

If a buffer allocation request cannot be satisfied from idle resources then the possibility exists of reclaiming a buffer that has already been allocated. This is controlled by the discard policy of a transmit queue. Note that reclamation is only attempted from the queue to which the allocation request was made. Options for specifying which buffer to reclaim are:

- *first* - the buffer from the head of the queue is reclaimed.
- *last* - the buffer from the tail of the queue is reclaimed.
- *marked* - any illegal cell in the queue may be reclaimed.
- *discard* - the incoming cell is dropped.
- *random* - a random buffer from the queue is reclaimed.

6.3 Policing Extensions

A policing mechanism (Section 5.3.3) identifies whether an incoming cell on a virtual circuit is outside its negotiated parameters. It is typically employed at the user/network interface and may be regarded as a router which applies an extra mapping to the cell stream. Indeed the same software that is used to implement the routers may run in the policing nodes.

The cell switching rate required of a policing node at the network periphery is not as great as that of a router internal to the network. In this case it may be feasible to implement the policing function as a user space forwarding gateway (Section 3.4.1)

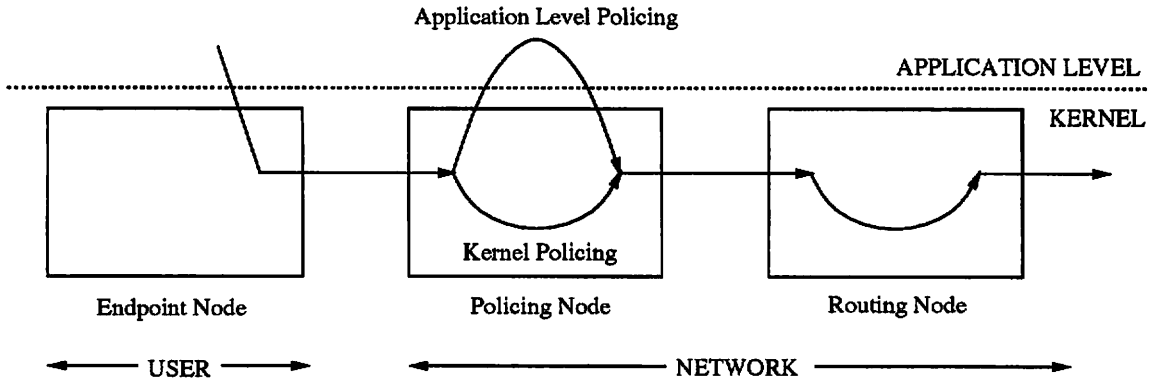

Figure 6.4: Policing Function Placement

in the policing node. A network configuration with a user space policing node is illustrated in Figure 6.4. User space implementation greatly increases the complexity of the policing function that may be considered and allows for ready experimentation with such techniques as traffic shaping. However it was still thought advisable to add support for policing within the kernel. An interface was added which enabled a virtual circuit to be assigned a policing mechanism. The action to be taken upon receipt of a illegal cell can include:

- forwarding the cell but marked as illegal.
- forwarding the cell as if it was legal.
- dropping the cell.

Given below is a summary of some common policing mechanisms. Nearly all of them depend on the manipulation of per cell arrival time stamps.

- **Null**: The simplest policing mechanism is none at all, every cell is by default legal. This is the operating mode of many endpoints and gateways on packet switched networks and may be useful when handling predominantly LAN type traffic. The null mechanism may also be employed by hosts relying upon buffer allocation and queueing controls to curtail flows. The null mechanism is also employed by internal network routers which are not concerned with policing.
- **Illegal**: At the other end of the spectrum from the null mechanism is the "mark every cell as illegal" scheme. This is useful as a mechanism for halting the cell flow for a circuit without breaking it.

- **Leaky Bucket**: The Leaky Bucket (LB) mechanism [**Turner86**] consists of a counter which is incremented by one each time a cell is presented for transmission and is decremented at a fixed rate so long as the counter remains positive. Associated with the counter is a threshold, a cell is deemed illegal if it causes the counter to exceed the threshold value.

- **Moving Window**: The Moving Window (MW) mechanism limits the number of cells accepted from a source within a fixed time. The time interval is applied continually. This mechanism requires the counter times for up-to-threshold cells to be stored.

- **Jumping Window**: A variation of the MW mechanism, with implementation complexity similar to the LB scheme, is the Jumping Window (JW) mechanism. The maximum number of cells accepted from a source within a fixed time interval is limited to a threshold. The time interval during which a particular cell can influence the counter value ranges from zero to the window width.

- **VirtualClock**: In the VirtualClock mechanism [**Zhang90**] each circuit, VC_i, has an associated virtual clock which is initially set to be equal to the real-time. Also maintained per circuit is its average interval $VTick_i$ between arriving cells, established at connection set up. When a cell arrives the appropriate VC_i is incremented by $VTick_i$ and the cell's priority is set to the result. A single transmit queue is maintained ordered by cell priority. The difference between VC_i and the real-time indicates how closely a circuit is adhering to its average transmission rate. Each virtual clock is periodically reset to the real-time to prevent a circuit from storing up credit. If the virtual clock is running ahead of real-time by a certain threshold then control action, such as cell discard, may be taken. In this way VirtualClock acts as both a policing and transmission ordering mechanism.

6.4 Statistics Gathering

There is a demand for traffic statistics from the higher level management software. Statistics are required to enable reaction to variations in traffic patterns. In addition, they are required as input to most proposed charging functions (see Section 5.4). The two main issues involved are *what* statistics should be kept and *when* should they be sampled. Statistics, in WANDA, are kept on a per circuit basis within the kernel. If desired, statistics at the transmission queue or per network interface level may easily be produced by summation of the statistics gathered at circuit granularity. An application level interface is available to read (and as a side effect reset) the current statistics for a circuit. The information returned includes the number of:

- cells successfully transmitted,
- cells rejected by the network or destination (NAKs),
- cells discarded because of successive NAKs,
- cells discarded because of buffer exhaustion,
- cells which had to be marked as illegal.

How often this information is sampled (if at all) is a management level decision. Two approaches to gathering the information may be identified. In the first, a single thread can block on a timer set to the shortest pending sample deadline across all circuits. The second approach is for the kernel, in response to events registered by the user, to "trigger" user level sampling. The trigger mechanism is simply an exceptional return from the receive interface. Trigger events can include:

- trigger after receiving n cells,
- trigger after receiving n marked cells.

The ability to monitor traffic on a stream provides sufficient mechanism for certain kinds of dynamic bandwidth management techniques to be considered. For example, if it was detected that an application was increasing its demands on a circuit (preferably before any policing function took effect), then management software could re-negotiate the QOS for that stream.

6.5 Implementation Concerns

The small cell size usually associated with ATM networks has ramifications for the host software. A host must deal with potentially very high cell arrival rates. If the per cell processing cost is too high then the host will suffer from poor performance. The heterogeneity of network interfaces (see Chapter 2) affects the structure of the low level networking software. The low level cell processing operation may be considered in three parts:

- interrupt dispatching,
- data transfer,
- protocol processing.

6.5.1 Interrupt Dispatching

The hardware interrupt dispatching mechanism will place the processor in an assembler level handler stub. This will save any volatile state before calling the appropriate "C" language interrupt service routine (ISR). When the ISR returns, the volatile state must be restored and exceptional operating system conditions, such as thread pre-emption, handled. The stub entry code, i.e. that which contributes to the interrupt latency, does not suffer any performance loss over that which would be found in a dedicated system. Software latency could be improved by the re-design of current hardware to interrupt at the start of a reception rather than after it completed. Some mechanism is required for the ISR to ascertain that the reception has completed (or that it is permissible to read the cell header); a polled interface would be suitable.

For a lightly loaded host, the interrupt entry cost contributes significantly to the average operation latency. For a host under load, multiple cells may be processed by a single interrupt. This is achieved by using interfaces that can fill a sequence of receive buffers, optimistic dallying in the receive ISR for incoming data, or a combination of both.

Latency may be further improved on vectored interrupt systems by using different vectors to encode cell header information. An example would be to employ one vector for cells with the EOF bit set and one vector for those in which it was clear. Another example would be to allocate a separate vector to each VCI. A more complex scheme would be a combination of the above; this would require two vectors for each VCI allocated.

Certain hardware architectures include special support for high speed interrupt processing. The ARM architecture, in addition to a conventional interrupt request (IRQ) system has a fast interrupt request (FIQ) line. An IRQ handler can be interrupted by a FIQ (if enabled). Significantly the FIQ handler has a set of private registers available. If a single device uses the FIQ line then its ISR may use these registers for storing state between interrupts or as a scratch pad.

6.5.2 Data Transfer

The network interfaces described in Section 2.2.1 differ both in the manner of transferring data to and from the host (DMA or memory mapped) and in the size of transfer supported: *cell* or *block*. The CFR, CBN and Fairisle are cell based networks. A block is defined to be a sequence of cells. For efficiency purposes a network such as the Ethernet must be considered to be block based. The operation of transferring data from the host to the network has an obvious implementation for all

types of interface. The data transfer operation for incoming cells is more complicated. The low level receive software attempts to leave the incoming data in a FIFO or DMA buffer until it has been decided where it should be moved (if at all). Higher level data copying software can be written in an interface independent manner (if FIFOs are contiguously re-mapped for the length of the maximum transfer unit). The only requirement is that the higher level code:

- read the data at most once for a cell,
- return an indication if the data was not accessed.

In this way the low level driver for a FIFO based interface is able to track the receive FIFO state. The receive FIFO on the CFR station chip is only one cell deep. Receptions from the network are impossible whilst the FIFO is full. Thus, even though the processor may be capable of handling multiple streams, their arrival distribution (e.g. back-to-back) can lead to dropped cells. The situation can be alleviated by immediately emptying the FIFO before computing what to do with the cell. This has the drawback of increasing operation latency and decreasing throughput.

The CFR FIFO on the ARM interface presents a special difficulty. Since the ARM I/O bus only supports 16 bit accesses the address of the FIFO cannot be passed to the processing routine (which expects to access a contiguous array of store). A callback interface to do the copy is possible but introduces a cost comparable to simply copying the data out of the receive FIFO in the driver. A similar problem arises with the CBN DMA interface (see below).

Later versions of the CBN interface have a DMA capability. A single cell may be copied to/from an address written into the interface at an operation dependent location by the VME host processor. Completion of the DMA operation is not signalled by the interface. Since the VME bus is locked while the copy is taking place any attempt to access the interface will hang until the operation completes. Therefore, no explicit synchronisation is required. DMA access to the FIFO has two direct benefits:

- data copy is faster due to implementation in hardware,
- overlap is possible between protocol processing and data copy.

As with the ARM CFR interface the higher level software is unaware of the correct (optimum) protocol for accessing the hardware. Again a callback interface could be used but with the same caveat as before.

Some networks provide a MAC-layer hardware indication of transmit success or failure. On the CFR, due to contention for the destination receive FIFO, it is quite common for the transmitter to see a Thrown-On-Ground (TOG) response in the returned cell. Depending upon the QOS of the cell's stream it may be necessary to re-transmit the cell. This requires that a copy of the cell has been maintained in the source host. In contrast, the CBN VME interface includes sufficient receive buffering that maintaining copies in the source host memory is not required.

6.5.3 Protocol Processing

When a cell/block is received the driver examines the header to determine the VCI. This indexes a table which, if the VCI is active, yields a record describing the virtual circuit. One of the fields in this record indicates what is to be done with the data:

- pass up to a thread on this host,
- forward on an interface of the current type,
- forward on a different cell interface,
- or forward on a different block interface.

If the data is to be forwarded then the record also contains a reference to a valid queue structure. This structure includes two generic insert methods: one for cells and the other for blocks. The queue and appropriate cell/block reference are passed as arguments when a method is invoked. The queue insert procedures are driver implementation dependent. For instance, the VME CBN driver insert routines do not implement any in memory queueing. Instead the data is copied directly into the transmit FIFO.

The cell insert interface uses a lightweight representation of a cell. The data portion is defined by a pointer, in this way the cell data may be left in a reception FIFO prior to final movement. The block insert interface is more heavyweight but will be called less often than the cell interface. The block interface in addition to consuming a buffer will also return one. Consider forwarding from one Ethernet network to another. The data buffer will be moved from the receive to the appropriate transmit DMA queue by adjusting pointers; no data copying is involved.

When forwarding between block and cell based networks care must be taken to achieve maximum throughput. Consider an Ethernet block being forwarded to the CBN. One approach would be for the Ethernet driver to invoke the CBN cell interface once for each cell in the block. However the invocation cost is large relative to

the small amount of work done by the procedure (copy/DMA into CBN FIFO). By calling the block interface the cells can be fragmented within the destination driver yielding a performance improvement. Since the contents are no longer needed the buffer can be returned immediately reducing requests to the buffer pool. Consider cells being forwarded from the CBN onto the Ethernet. To achieve sensible throughput multiple cells must be encapsulated in a single Ethernet frame. If they are assembled in the CBN driver prior to passing to the Ethernet block forwarding interface, then a set of per-cell procedure calls is saved. It can be seen that a block based network, such as the Ethernet, does not need to implement a cell transmission interface.

The network dependent interface includes a call to inform the driver when the state of a particular circuit has changed. In this way the driver can, if desired, shadow the generic data structures using a layout more suited to the particular network concerned.

6.6 Summary

Extensions have been made to the WANDA kernel in order that a broad range of QOS policies, under management control, may be supported. Such an organisation permits easy comparison of the implementation complexity involved in different strategies. The interface applies equally well to both endpoint systems and inter-network routers. It is likely that in the latter case much of the low level implementation will be in hardware. The policies supported by the kernel mechanisms are not exhaustive. However, the interfaces are defined in a manner that is easily extensible e.g. adding a new queue insert strategy.

Despite the heterogeneous nature of the network interfaces concerned a large amount of the networking code is implemented in an interface, network and architecture independent manner. For some interfaces, such as the CBN, quite large performance improvements are possible at the cost of moving (and hence duplicating) code down from the network independent to the driver level. At present, due to the the greater maintenance effort such improvements cause they have for the most part been avoided.

Chapter 7

Experimental Programme

This chapter examines the performance of the WANDA MSNA implementation over some of the ATM networks described in Chapter 2. It has long been the case that protocol performance figures are quoted in terms of throughput and delay. Little attempt has been made to break such figures down into individual component contributions, [**Schroeder89**] being a notable exception. [**Jacobson88**] is a good example of how detailed knowledge of a protocol's behaviour in a working environment can be used to improve performance. In the subsequent experiments absolute measures of delay are reported but with the emphasis on attributing any jitter so highlighted to a specific system component. The experiments are divided into three groups examining the delay response of:

- a single cell stream over an idle network and otherwise idle hosts,
- a router in both the presence and absence of contention traffic,
- a host handling Pandora audio and video simultaneously.

In each case any source of *intrinsic* (i.e. unavoidable) jitter is identified. Lessons for kernel and device driver structuring in order to minimise the jitter on time critical streams are detailed. The CBN and CFR network interfaces are contrasted in their suitability for the support of multi-service traffic. Proposals are made concerning the design of future network interfaces that aid the host in meeting application level communication requirements. The chapter concludes with a comparison of the startup latency, i.e. connection establishment followed by single byte transfer, for the TCP, UDP and MSNL protocols.

7.1 Experimental Apparatus

The experimental system, illustrated in Figure 7.1, comprised five processor nodes attached to two networks. Two of the processors were equipped with CBN interfaces whilst a further two had CFR only interfaces. The final processor had an interface on each network and was therefore capable of acting as a router. Some of the machines were also attached to the Computer Laboratory main Ethernet.

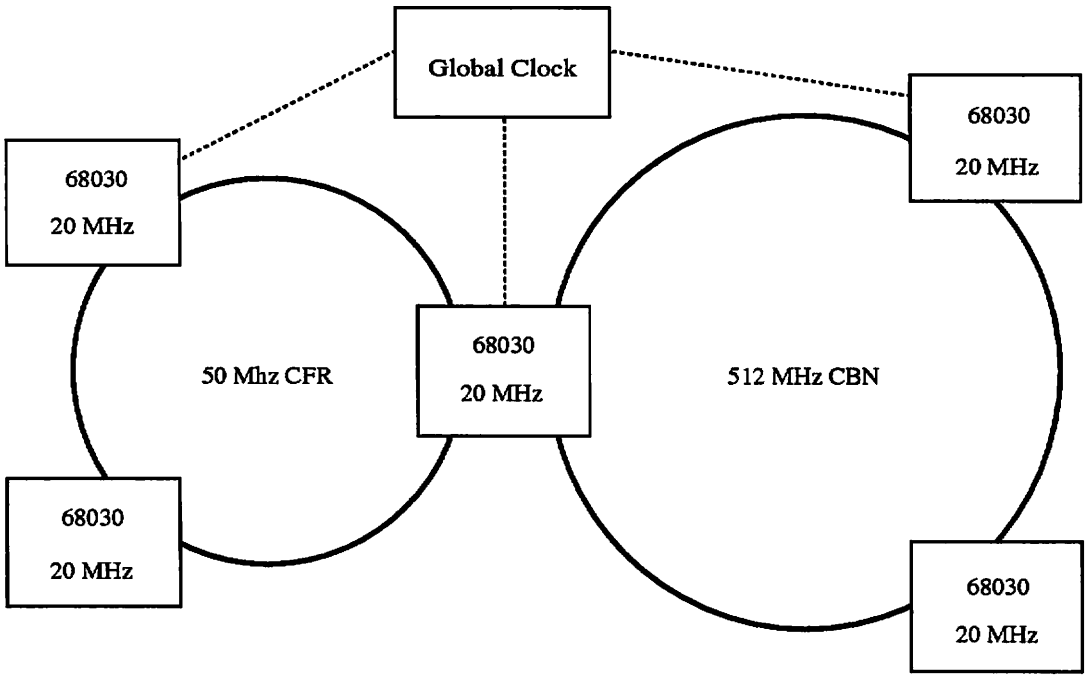

Figure 7.1: Experimental Configuration

Selected processors in the system were supplied with memory mapped counter cards[1]. A single oscillator, in this instance 5 MHz, was used to drive all of the counters at the same rate. A reset bus was provided between all cards so that the counters could be started together from zero to a tolerance which was dependent[2] upon the bus geometry. This hardware provided a cheap method of generating time stamps which was fairly unobtrusive. It would have been possible to use the 68030 local clock to generate time stamps. However, this would have required complex (and obtrusive) protocols [**Lamport78**, **Mills89**, **Ofek89**] to synchronise the clocks initially and correct for drift. In addition, the resolution of the local clocks on the 68030 boards available was only $6\mu s$. The procedure to read the global clock value takes approximately $15\mu s$ to execute on a testbed 68030.

[1]Designed and constructed by the author.

[2]Approximately 30ns for the configuration described.

The CBN geometry comprised a five station ring running at 512 MHz. The latency of the ring (time for a bit to complete a single revolution) was 10.4μs with approximately 79% of this comprising delay in the fibre. This provided the bit delay for a 3 frame ring with a gap of 1.44μs (the frame delay thus being 2.99μs). On an *idle* network the maximum theoretical delay in accessing the CBN is therefore 4.43μs (the CBN VME interfaces being capable of transmitting in only one of the four slots (Section 2.2.1) in *each* frame).

The CFR geometry comprised a four station (the testbed machines and a monitor station) ring running at 50 MHz. The delay in the ring comprised a single 6.08μs slot and a gap of 2.24μs. On an *idle* network the maximum theoretical delay in accessing the CFR was therefore 8.32μs[3].

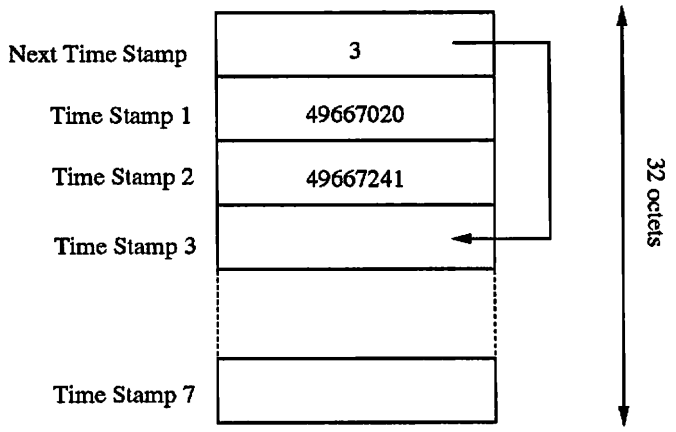

Figure 7.2: Time Stamp Cell Format

Part of the QOS specification "turned on" the generation of time stamps for a stream. On streams to be measured time stamps were placed in all cells at selected points in the network. The format of a time stamp cell is illustrated in Figure 7.2. A time stamp is produced by writing the current value of the global clock (32 bits) into the cell payload at a position given by an index at the start of the payload. The index is then incremented. The sink system would typically compress this information and write it, at the end of the run, to a remote file system for subsequent analysis. All code (application and kernel level) was written in "C", and compiled using GCC with optimisation enabled. Performance could have been improved by re-coding segments in hand crafted assembler but was rejected on the grounds of maintainability. All bootfiles included a WANDA SAS kernel (Section 3.1.4) with debugging checks disabled.

[3]The CFR was never totally idle as the monitor station (an ORL Metrobridge) would send a management cell once every second. For all practical purposes this cell may be ignored.

7.2 CBN Delay Response

This section details the performance of the CBN as perceived by WANDA user level applications running on the testbed hardware. The initial experiment was designed to determine the causes of the delay variation experienced by an application to application level stream of cells on a CBN with *no contention traffic*. The rate of transmission was set such that the results for each cell (32 octets of user data) could be regarded as an independent experiment.

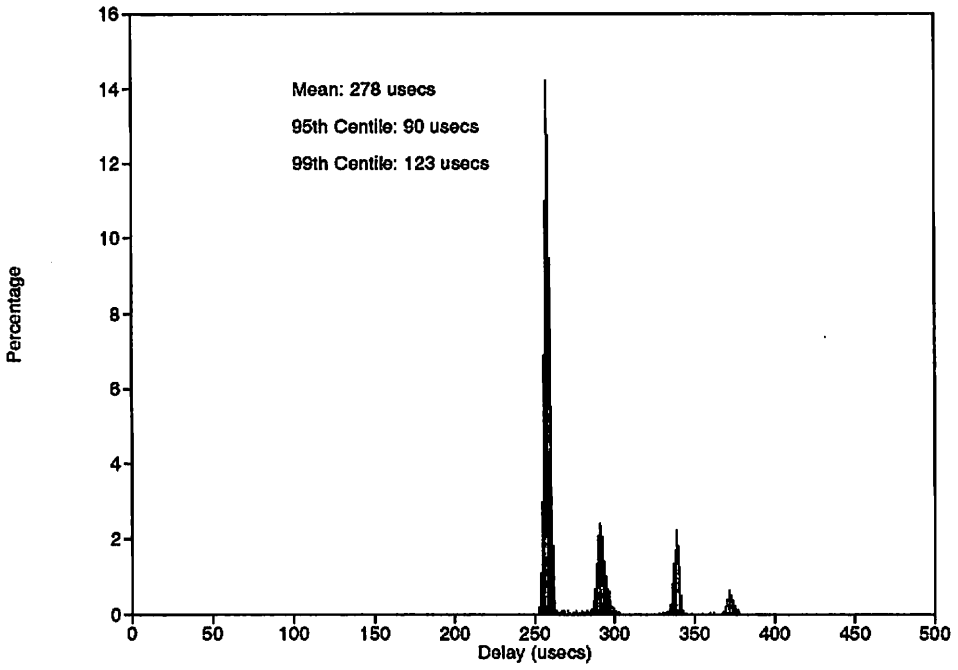

Figure 7.3: CBN Delay: Initial Experiment

Figure 7.3 shows the variation in delay for a sample run of 10000 cells. The nth centile is defined to be the minimum interval in which $n\%$ of the sample distribution may be found. Immediately apparent is the large difference between the CBN latency ($10.4\mu s$) and the minimum application to application level delay ($252\mu s$). The theoretical CBN jitter ($4.43\mu s$) is also insignificant when compared to that in the stream when it reaches the sink application (99th centile of $123\mu s$). The two obvious sources of jitter in the endpoints are due to contention from:

- threads (other than the source and sink) (Section 7.2.1),
- ISRs (other than the CBN) (Section 7.2.2).

7.2.1 Thread Contention

The WANDA scheduler used would only pre-empt a thread if one of higher priority became runnable. In the previous experiment both the source and sink threads ran at the default user priority. Higher priority threads existed at both the kernel and application level. In particular, each ANSA capsule included background threads running at priorities between the default user and kernel levels. The kernel included threads to handle domain termination and MSNL connection establishment. Device drivers typically utilise kernel threads. For instance, the CFR driver uses a thread to handle outages and check the sanity of the ring. In the next experiment the source and sink threads were created with priority greater than any other thread in each of their respective hosts. This ensured that in each of the endpoints neither thread could be pre-empted once running.

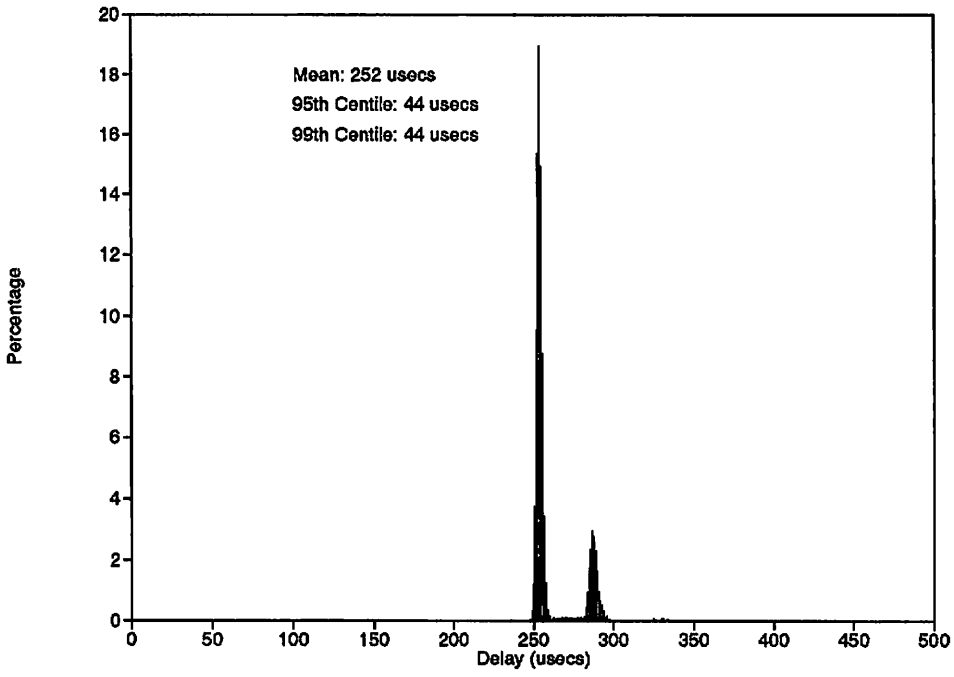

Figure 7.4: CBN Delay: High Priority Source and Sink

Since the source thread generated a CBR stream by dallying on the global clock counter this meant that it was the *only* thread scheduled on the source host for the duration of the experiment. The situation on the sink host is different. As the receiver thread blocks on a semaphore within the kernel the scheduler is given the opportunity to run other threads. When the receiver thread becomes runnable the thread switch can take one of two paths. If another thread is running then its state must be saved and that of the receiver thread re-loaded. The second case, which will take less time, is when the receiver thread is spinning in context on its wake

up condition (Section 3.1.1). Figure 7.4 illustrates the result of this experiment. It is argued that any contention present is a result of the processor having CBN level interrupts masked at the time a cell is received. This can be because of a higher level ISR executing (i.e. the clock) or because a thread has raised the interrupt level of the processor.

7.2.2 ISR Contention

By disabling clock interrupts, both of the above sources of jitter should be eliminated. The timer interrupt code has the task of maintaining the system clock and inspecting the timer event queue. It may also call into the scheduler if time slicing is enabled. The timer event queue is a list of blocked threads each with a specific time of when they wish to be made schedulable. The queue is ordered by time out value. Thus the interrupt code need only inspect the head of the queue to see if any calls into the scheduler are necessary (the common case is that no calls are required).

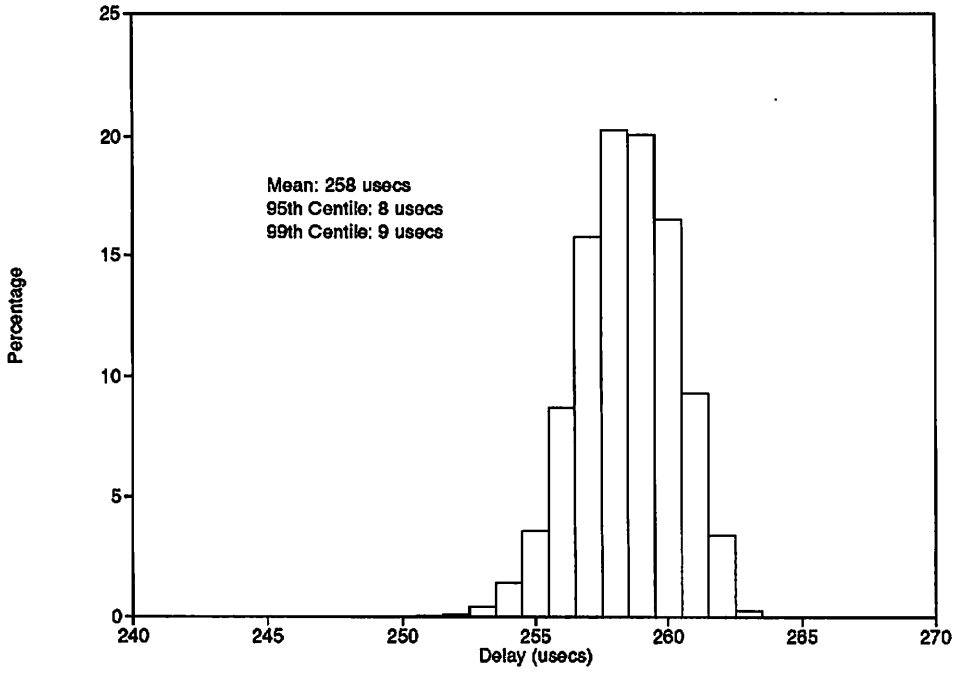

Figure 7.5: CBN Delay: Timer Interrupts Disabled

The timer on the testbed 68030s was programmed to interrupt once every 10ms. The timer hardware found on many processors is not ideal. For instance, both WANDA and UNIX maintain the system time as two 32 bit quantities (secs and μsecs). Implementation of this in hardware would be trivial. A second set of 32 bit registers could be used to indicate the time when software desired the next timer

interrupt. Thus there would be no need to have periodic clock interrupts. The WANDA "C" level timer interrupt code takes approximately 14μs to execute in the common case. For each thread that is required to be woken, the call into the scheduler may consume an additional 45μs. If, as a side effect of these calls, the scheduler decides that a re-schedule is required, this will occur on return from the (possibly nested) ISR. Figure 7.5 shows the result of disabling timer interrupts for the duration of the experiment. It can be seen that the jitter on the stream has been reduced to very nearly that expected from the CBN. Additional experiments were performed with an oscilloscope to determine the remaining sources of jitter. These were found to be:

- DRAM refresh cycles on the 68030 memory,
- synchronising global clock counter reads,
- synchronising CBN VME interface accesses.

The result of the same experiment using the CFR instead of the CBN yielded a mean delay of 314μs and a 99th centile of 12μs. The delay difference between the CFR and CBN is mainly due to the more complex interface to the CFR station chip and also the requirement to maintain a copy of the transmitted cell in the source host until notified of successful transmission (Section 6.5.2). The difference in delay *on the network* is, by comparison, small.

7.2.3 Kernel Structure Implications

From the initial experiments it is possible to draw some conclusions regarding the structuring of the kernel, and its associated device drivers, for handling multi-service traffic. For incoming network traffic to be handed to an application in a timely fashion requires, at the very least, that the processor be in a regularly pre-emptable state i.e.

- network level interrupts are enabled,
- thread pre-emption is allowed.

The reasons why the processor may not be in a pre-emptable state concern kernel synchronisation requirements. The kernel must protect itself from *activations* concurrently executing critical regions of code. An activation is defined to be the instruction sequence of a thread or ISR. Two mechanisms are provided for synchronisation within the WANDA kernel, they are:

- semaphores,

- locks.

A semaphore provides the mechanism to limit the number of threads concurrently executing within a critical code region. Most of the WANDA kernel critical regions, which are guarded by a semaphore, specify a maximum of a single activation within them at any one time. A thread executing within a semaphore region, unless it has arranged otherwise (this can only be done for kernel critical regions at present), may be pre-empted. In many instances this can lead to undesirable consequences. Consider a low priority thread pre-empted by a higher priority thread whilst executing within a semaphore region. If the new thread tries to access the same critical region then it must block until the lower priority thread is scheduled and releases the semaphore. This phenomenon is called *priority inversion.*

A lock protects a region of kernel code to be executed by *one* thread only. The most efficient implementation, on a uniprocessor, is to disable interrupts on entering and restore them on exiting the critical region. On a multiprocessor this protocol must be extended slightly. The lock is further defined by a bit in memory. After interrupts are disabled an atomic instruction, such as Test-And-Set (TAS), is used repeatedly until it succeeds. The critical code segment is executed and the bit cleared with another atomic instruction before the IPL is restored. Such locks are often described as *spin locks.* A lock system could be envisaged which allowed interrupts but disabled pre-emption. The lock entry code would set a boolean value to disable pre-emption. Scheduler code, if called by an ISR to wake a thread, would set a re-schedule pending flag if it decided to pre-empt the current thread. The spin lock exit code would check this flag and if necessary call the scheduler. Such an organisation also requires that interrupt code should not call into the critical region unless it can be guaranteed that no thread would ever execute the critical region on the *same* processor that fields the interrupt.

Of course, on a multiprocessor, there will be lock contention that leads to wasted cycles. One theoretical problem is that, certainly on the Firefly hardware [**Thacker87**], the spin locks as implemented in WANDA are unfair. In brief, if there are n processors and a critical region takes time t to execute then it is *not* the case that a thread may be spinning on a lock for maximum time $(n-1) \times t$. However, since the critical regions should be designed to be small (as pre-emption is disabled within them) and contention minimised, this problem is typically not an issue. Techniques such as calling the scheduler to run another thread on TAS failure should not need to be considered. The subsequent section considers the structure, in particular relating to synchronisation, of device driver code. This is important as one poorly written driver can cripple a systems ability to handle multi-service traffic.

Device Driver Structure

A typical device driver has a requirement to synchronise between thread and interrupt level code. Thread code that requires to protect itself against interrupts must raise the processor IPL above that of the device. This is not sufficient as a thread switch could occur (e.g. because of timer expiry) to a thread executing at an IPL below that of the device. In many cases the thread level code resorts to disabling both pre-emption and interrupts. The areas of the kernel which are affected by such concerns can be large due to overloading the amount of processing done in ISRs (in order to increase performance). Writing device drivers in such a manner requires much careful thought as the environment available within an ISR is different than that at the thread level. For example, it is not sensible to block on a semaphore within an ISR. This is not an argument against structuring systems using upcalls [**Clark85**], rather that their pervasive use can have detrimental effects in certain environments.

One technique is to only perform enough processing in the ISR to kick a device handler thread. Unfortunately for most architectures this cannot be done by generic code. Each ISR must contain device specific code to negate the interrupt condition. If all devices on receiving an interrupt acknowledge proceeded to de-assert the interrupt condition, until *programmed to do otherwise*, then a single simple ISR would suffice for all devices. Care must be taken on a multiprocessor that the thread to handle an interrupt is not scheduled on an unsuitable processor. For example the I/O architectures of some processors are asymmetric. On the Firefly hardware, devices may only be accessed from a single processor: the I/O processor (IOP). Either a device handler thread must be scheduled on the IOP or some form of mailbox protocol must be implemented between the device handler thread and the IOP. The WANDA thread creation interface includes a parameter constraining a thread to be scheduled on a single nominated processor if desired.

Application requests requiring I/O arrive at the driver interface via a downcall through the kernel (certainly for network drivers). The downcall thread may have acquired kernel lock(s). For many types of request (most importantly that for the transmission of data) the amount of work to be done on the call return path is null (save freeing of the lock(s) acquired). It is advantageous to make the release of any locks the responsibility of the driver. If a driver can arrange to release the lock(s) quickly, whilst still preserving correctness, then the lock contention jitter experienced by other threads may be reduced. Correctness will usually entail acquiring a driver level lock *before* releasing the higher level lock(s). Such techniques have the effect of *increasing the latency* of an individual operation in the absence of contention. A good discussion may be found in [**Birrell89**] on the trade-offs associated with the use of single vs. multiple locks.

7.3 Router Effects

The purpose of the next set of experiments is to analyse the behaviour of a stream when a router is interposed. In all cases the source and sink are on the CBN and CFR respectively. It is argued that the software router will show jitter effects of similar distribution to that of a hardware implementation. However, the absolute variations will be amplified. The router low level transmit and receive code is the same as that for an endpoint host. By examining router performance the response of this particular subsystem can be determined in isolation from higher level effects. The first experiment (Section 7.3.1) examines the basic router response with no contention traffic. The subsequent experiments show the effect of contention traffic combined with different router forwarding policies on the delay experienced by a single cell stream. In all cases the stream on which measurements are made consists of a single cell pulse. The minimum interval between each successive single cell pulse was set such that for an idle router no interference between successive cells is observable.

7.3.1 Basic Router Response

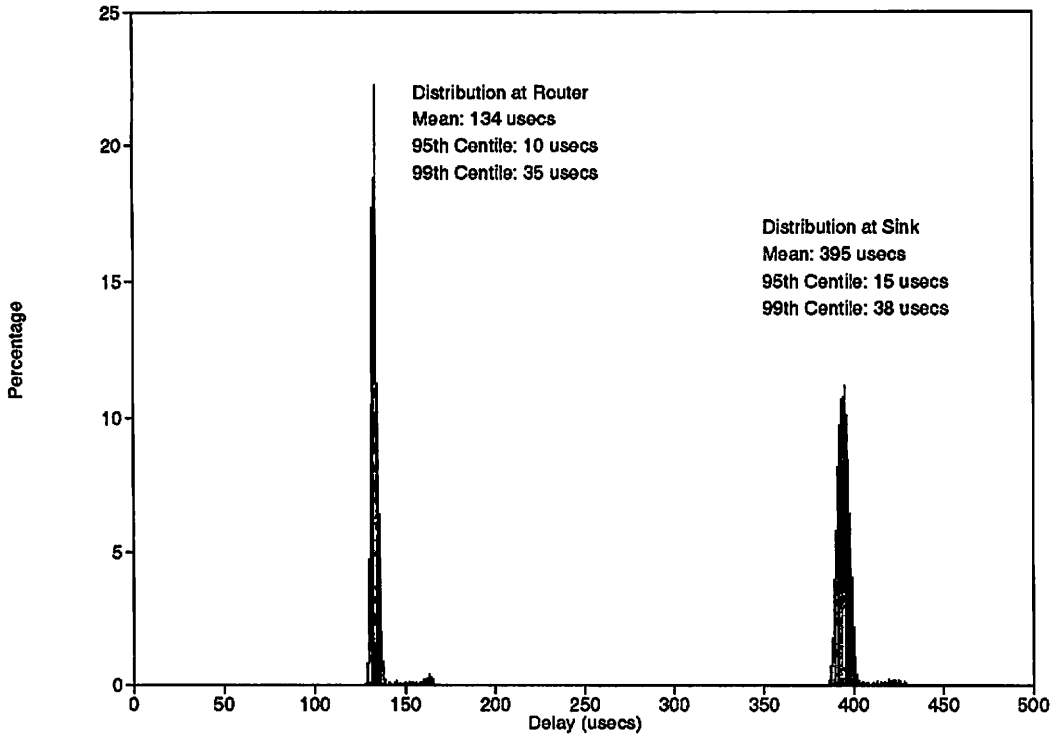

Figure 7.6: Single Cell Stream: CBN/CFR Router

The jitter of a single stream, through the CBN to CFR router, was measured. Figure 7.6 shows the result of the experiment. The timestamp at the router was taken just before the cell is added to the CFR output queue. As expected, the jitter experienced by a single stream is increased over the CBN only case (Section 7.2.2). This is due to the cumulative effect of jitter due to the CBN and CFR. The maximum theoretical jitter for the connection is that of the two networks combined i.e. 12.75μs. The experimental jitter (95th centile) is slightly greater than this due to the sources of contention detailed in Section 7.2. The 99th centile (38μs) is much larger due to timer interrupts being enabled in the router.

7.3.2 Contention Traffic

The previous experiment was repeated in the presence of contention traffic. Figure 7.7 shows the results for three types of contention traffic and transmission strategies:

- **1. Single Cell FIFO:** A contention stream of 1 cell per msec was generated. The QOS field of each stream was such that the router placed both streams on a single queue which used a FIFO strategy.
- **2. Two Cell FIFO**: A contention stream of 2 cells each msec was generated. The cells were sent in a back-to-back burst which gave a gap of 18μs. The router adopted a FIFO queueing strategy.
- **3. Two Cell Priority**: An identical contention stream of 2 cells each msec was generated. The QOS selected was such that timestamp cells would always be forwarded in preference to those from the contention stream.

In all experiments the contention stream was run for 100,000 impulses (single or double cell). The maximum impulse delay does not show a linear increase with the number of cells in the contention burst. This is due to periodic clock interrupt contention. When using priority forwarding it might be hoped that the impulse delay be reduced to something approaching that of the single cell contention experiment. Whilst the third experiment shows a reduction in the impulse delay, nothing approaching the ideal has been achieved. This may be explained by considering the nature of the CBN VME interface. The CBN receive interface is implemented as a FIFO. An incoming cell can experience jitter due to the FIFO not being empty. Even though the cell might have a higher priority than any before it in the FIFO, it will have to wait until all of the the former have been processed. The maximum impulse delay reduction of 26μs (189 - 163) represents the cost of placing the second cell of the burst stream into the CFR transmit FIFO and waiting for the FIFO to become available again.

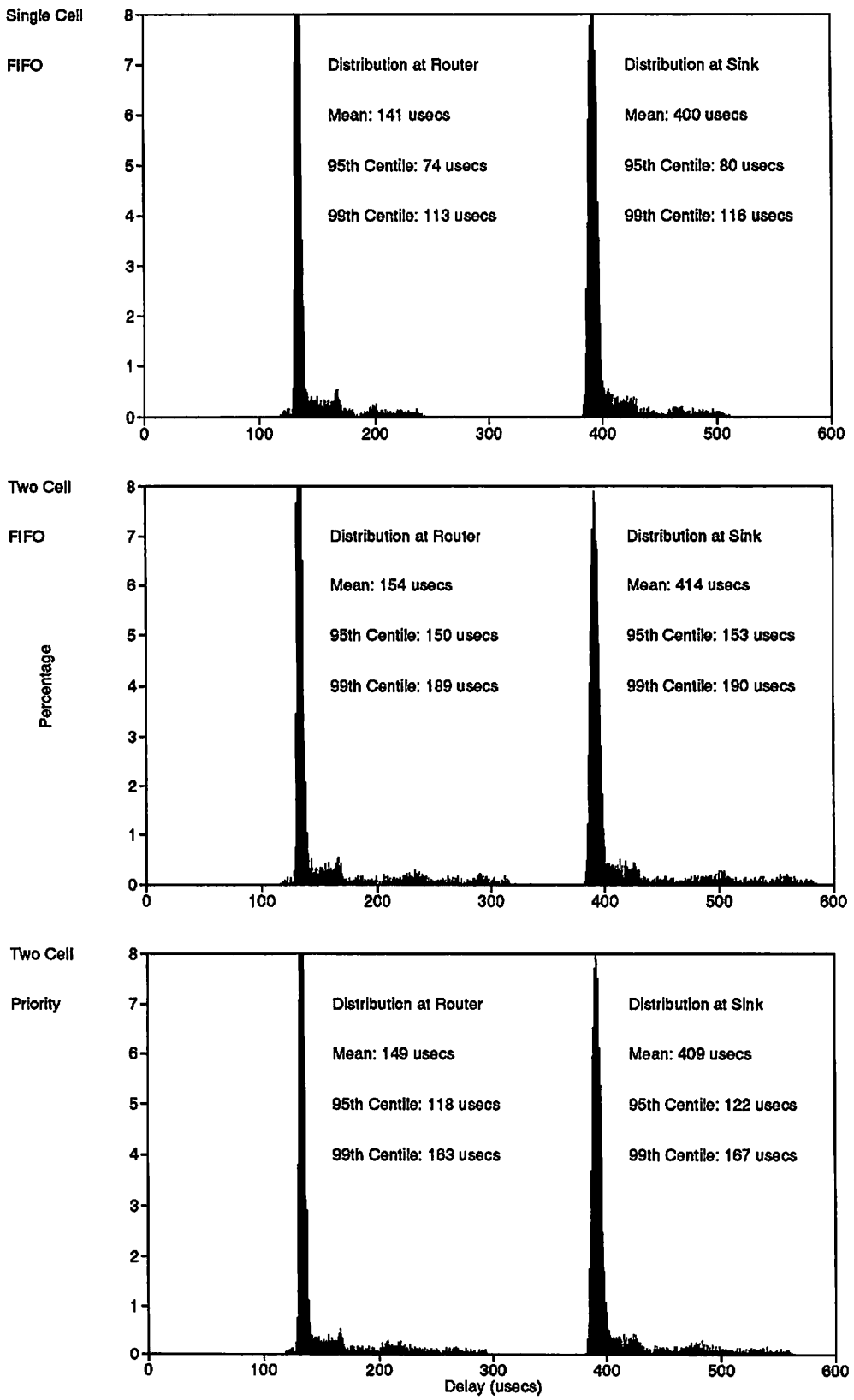

Figure 7.7: Router Contention

7.3.3 Network Interface Design

The contention experiments of the previous section suggest that some form of priority be incorporated into the interface hardware. This requires a mechanism for the hardware to determine which are priority cells. It is natural to associate this information with the VCI. One approach would be to statically partition the VCI space. A more sensible approach is to allow software to dynamically set the priority attributes associated with a VCI. The simplest improvement on the base CBN hardware would be to implement multiple receive FIFOs, the priority attribute for a VCI simply encoding the FIFO into which the cells for a stream are placed. By providing a status register which encodes the state of all the FIFOs, the host software is able to perform expedited cell processing. As with the basic interface only a single interrupt is generated for all conditions. Such an interface still presents an end system with difficulties in handling multi-service traffic.

Consider two end system threads; one attempting to receive a high priority stream the other a low priority one. The high priority cells will be read out of the interface in preference to the low priority ones. The thread receiving the high priority stream will be scheduled in preference to that for the low priority stream. However, incoming cells for the low priority stream will cause application level jitter due to interrupt processing costs. One solution is to provide multiple FIFOs each with an independent interrupt level which may be set by host software. By arranging for the high priority thread to execute at an IPL above that of the FIFO for the lower priority stream, jitter is reduced.

The cost of per cell processing exhibits a considerable burden on the host. In many cases the provision of an intelligent host interface is suitable [**Garnett83**]. Such an interface will typically only involve the host at the frame (block) level. This is achieved by using DMA techniques. Associated with each circuit will be a list of receive buffer(s) cf. Ethernet interfaces (Section 2.2.1). Incoming cells are placed in priority order into their associated buffers. An interrupt at the appropriate priority level is generated upon either:

- no buffer space being available for an incoming cell,
- an EOF cell being received.

It is possible for the DMA operation on a low priority stream to be overlapped with the application level processing of the frame from a higher priority stream. Depending upon the organisation of the memory system and the resolution policy between conflicting processor and I/O device memory requests, this might introduce jitter into the high priority stream. A memory interleaving scheme which placed the buffers for high and low priority streams in different banks would appear suitable.

7.4 Pandora Streams

The purpose of the next set of experiments is to examine some of the issues involved when multiple streams with performance constraints are handled at the application level. The experiments comprise of a WANDA machine processing a videophone connection from a Pandora box. An example of such processing might be removing noise on the audio stream and performing compression on the video stream. The experimental apparatus is illustrated in Figure 7.8.

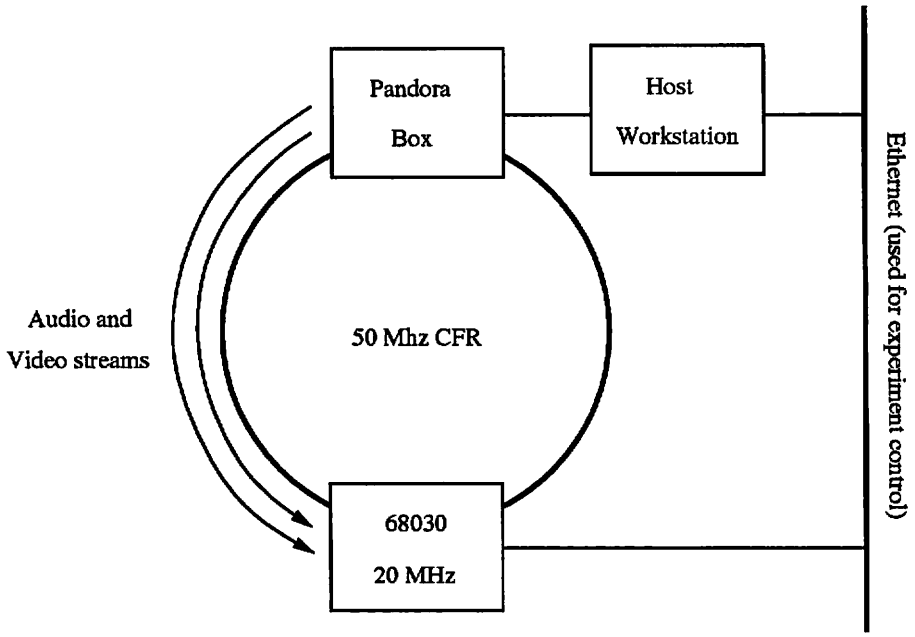

Figure 7.8: Video Phone Experiment

The sink WANDA machine simply emulates a Pandora box. This was easily achieved as Pandora applications use ANSA for their control channel. In fact the control software was run on a machine different than the sink WANDA host. Such flexibility is important as it means the endpoint terminals may be made simpler, cheaper and more reliable without the requirement to run complex management software. All Pandora multimedia streams currently use the MSSAR block protocol (Section 2.5). Each block includes a microsecond timestamp which is generated in the source Pandora box and represents the time since the stream was created. The video and audio arrives at the sink WANDA machine as two separate MSNL streams. The timestamps are used to aid intra-stream synchronisation. Unfortunately the timestamps for voice and video streams are generated by *separate* clocks within the Pandora box. This limits their application for inter-stream synchronisation.

Inter-arrival gaps are measured in the subsequent experiments as the Pandora box timer and global clock are not synchronised. It would have been preferable to extend the global clock system into the Pandora box for the experiment, however, given the limited time available this was not feasible. The Pandora *source* rates were set such that a single 96 byte audio packet was generated every 4ms and a 4192 byte video packet was generated every 40ms. Two threads were allocated on the sink WANDA machine for the reception of data from the two streams. The audio thread performed a simulated 8ms processing on each packet before discarding it, for the video thread this figure was 0.5ms. Each experiment was run for the duration of 100,000 audio samples.

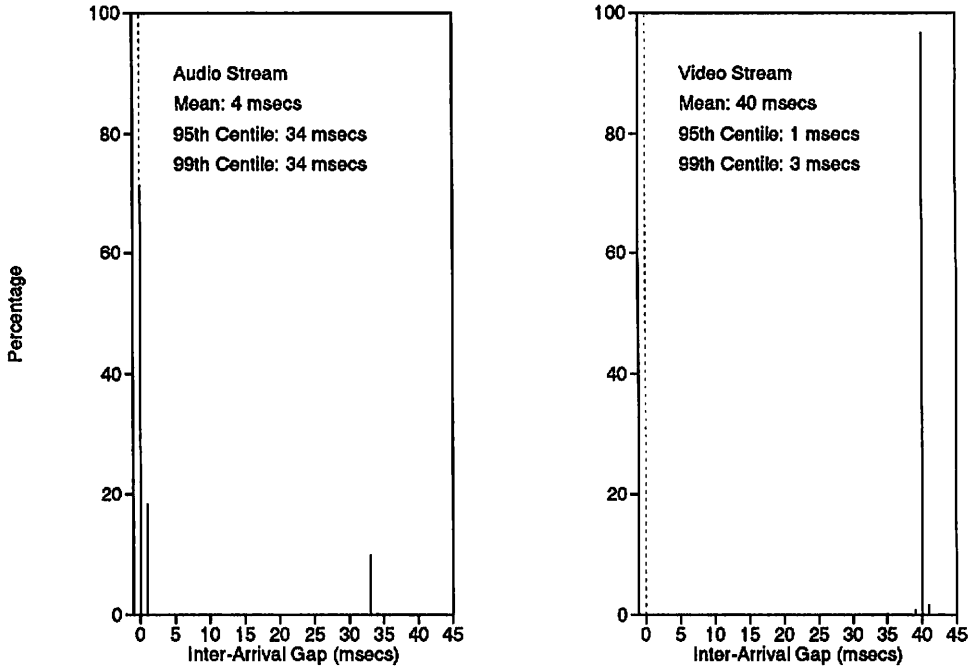

Figure 7.9: Equal Priority Threads / CFR Dally

Figure 7.9 shows the inter-arrival gaps of the audio and video streams *at the application level* on the sink WANDA machine. The two receiving threads executed at the same priority, which was above that of any other thread on the sink machine. Immediately apparent is the substantial jitter on the audio stream. This is beyond that expected, due to the processing time of the video thread (8ms). The audio distribution represents, on average, a single large inter-arrival gap followed by a sequence of *back-to-back* packets. Since the network was otherwise idle, the only source of contention which could cause such jitter must be in either the Pandora box or WANDA machine. The obvious source of contention at the sink which would cause this is the reception of the video at the network interface.

The contention was traced to an implementation detail in the CFR driver code. The CFR driver upon receiving a cell, if the EOF marker is not set, will dally on the expectation of receiving subsequent cell(s) for the same frame. This is due to the high cost of many interrupt entry and exits for a single frame. Associated with the dally is a time out which may be set on a per stream basis. Such a structure has possibly dire consequences. If the dally time is too small then latency may actually be *increased* due to the processor being in a non-interruptable state, returning from the dally loop, when the expected cell is received. If however dallying is successful then it offers optimal latency for both reception and transmission.

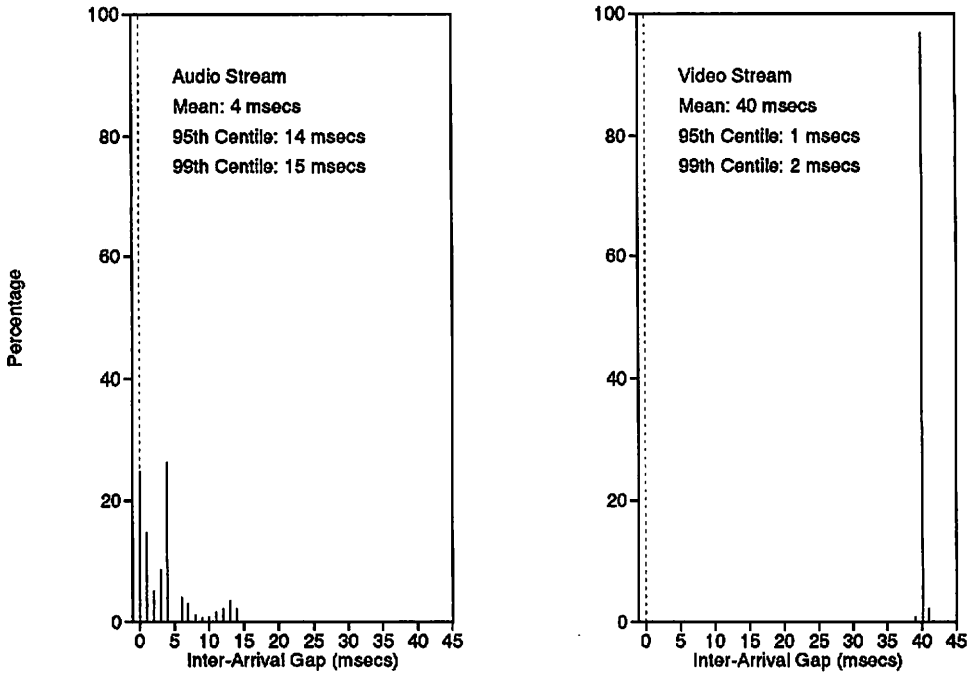

Figure 7.10: Equal Priority Threads / No CFR Dally

Figure 7.10 shows the result of repeating the previous experiment with no dallying in the CFR receiver ISR. There is a large reduction in the jitter of Figure 7.9. However, the sink audio stream still differs significantly from the distribution at source. The jitter attributable to video thread processing may be discovered by increasing the scheduling priority of the audio thread over that for the video thread. The result of such an experiment is shown in Figure 7.11. Whilst the jitter has been further reduced it is still substantial. Figure 7.12 shows the jitter on the audio stream when the video stream is sinked by a separate machine. The result indicates that a significant fraction of the audio stream jitter is produced at the source Pandora box. This is entirely due to contention with production of the video stream (a separate experiment showed a Pandora box able to produce a uniform CBR audio stream).

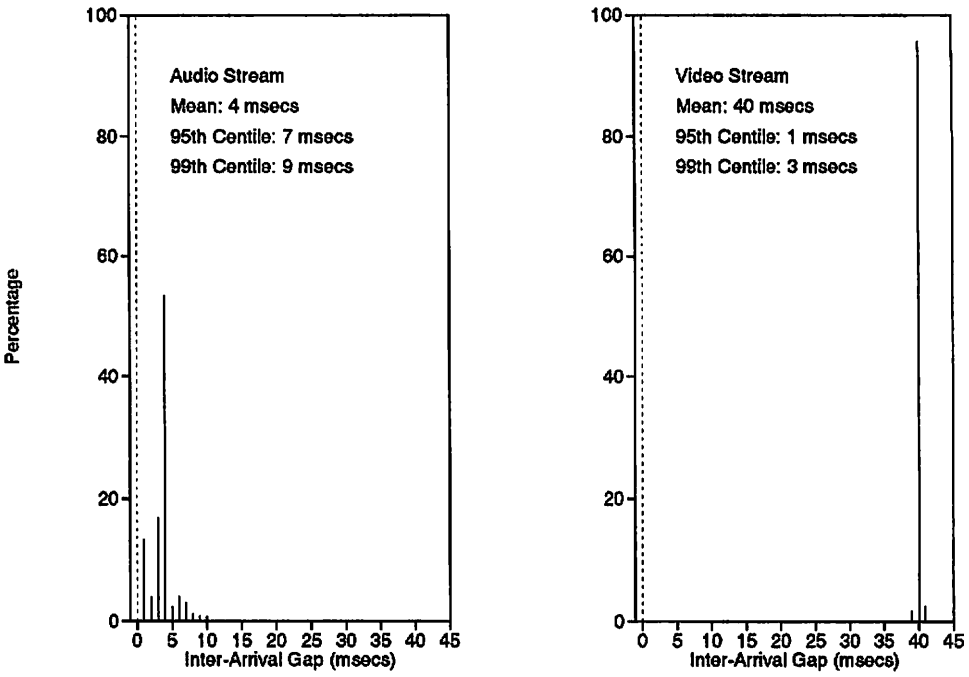

Figure 7.11: Priority Audio Thread / No CFR Dally

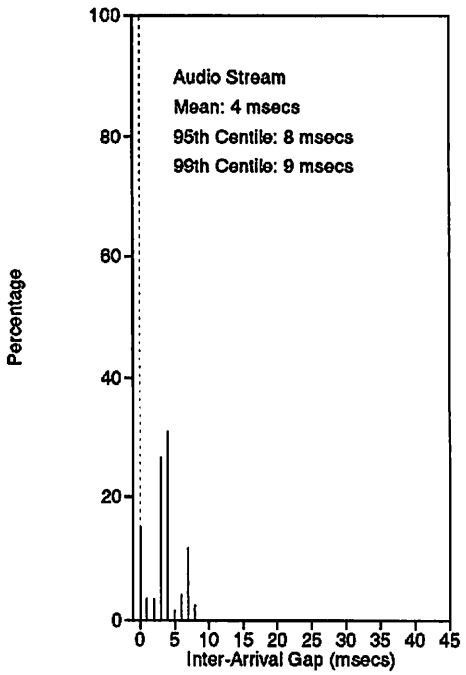

Figure 7.12: Video to Separate Sink

The Pandora box implementation comprises a set of INMOS transputers. Both the audio, video capture and video mixer devices are handled by their own transputer. A fourth processor handles the CFR interface. A final transputer, called *the server*, is responsible for communication between the above. The jitter on the audio stream (Figure 7.12) is due to contention in the server transputer and the fact that audio cells may not receive priority transmission in the CFR transputer. The production of a bursty audio stream increases the probability of cells being dropped by the network and also the jitter removal complexity necessary in the sink. The jitter on the audio stream could be reduced by giving it priority processing in the transputer controlling the CFR. Alternatively, each device could be allocated a separate network interface. The economics of such an approach may be acceptable as each interface need not be as complex as the current general purpose implementation.

In many respects the handling of audio presents a greater challenge due to the higher sample rate when compared to video. In an ATM environment an endpoint is forced to implement a degree of jitter removal (by provision of an *elastic buffer*). The presence of such functionality may be exploited in certain situations by *batched* processing of media sequences. For example, when replaying an audio recording from a repository ten samples may be sent together every 40ms rather than one sample every 4ms. Thread switching costs in the server for this component will be reduced by a factor of ten. In this way the repository can be engineered to handle significantly more traffic than if it were naively designed.

As can be seen it is a function of the program structure and scheduler policy as to how application threads handle network traffic. It is often heard that a *real-time* thread scheduler is necessary in order to support multimedia traffic. A real-time thread scheduler is best described as one which is able to make guarantees to its clients regarding the amount and distribution of CPU time that they will be allocated. The thread creation interface includes parameters to describe the scheduling demands of the thread. A management function will see if the thread's schedule, when combined with that of all the other threads in the system, is satisfiable. This process is analogous to the virtual circuit call admission procedure.

It is not necessary to employ a real-time scheduler to achieve real time performance. If an application programmer has knowledge of the underlying scheduling policy, for example that a fixed priority non-pre-emptive policy is employed, then an application may be structured accordingly. However, problems arise when there is other activity on the machine of which the programmer is unaware, if the underlying scheduling algorithm changes or if the code is ported to a different machine. A real-time interface enables an application to express it's requirements explicitly. In this way it is isolated from changes in the underlying implementation.

7.5 Startup Latency

An oft stated disadvantage of the virtual circuit model is the latency required to establish a connection before any data may be sent. The optimistic establishment mechanism (Section 3.4.1) enables the initial data latency for a long haul connection to approach that for datagrams over the same path. However, many connections are limited to the local area where such techniques yield smaller gain. There is a marked lack of performance figures concerning connection establishment in the literature. This is analogous to the situation found in the measurement of RPC system performance. The RPC bind time is ignored or at best amortised over a large number of invocations on the bound interface. It is not the case that all styles of communication will justify such a simplification. [**Caceres91**] gives results on the frequency and usage patterns of typical TCP connections.

Protocol/ Operating System	Establishment (μsecs)			Single Byte Ping (μsecs)		
	Min	Max	Mean	Min	Max	Mean
UDP/UNIX [4]	N/A	N/A	N/A	4636	7744	4856
TCP/UNIX [4]	5184	11780	5619	4580	12940	6377
MSNL/UNIX [4]	3356	4168	3505	4084	8776	4309
MSNL/WANDA [5]	2139	5603	3763	1604	4235	2423

Table 7.1: Call Establishment and Data Exchange Latency

Table 7.1 gives some results for call establishment and data latency (single byte exchange) for various protocol and operating system combinations. The timings were taken at periods of low network activity on the Computer Laboratory main Ethernet. Each machine, except for the normal complement of system applications, was otherwise unloaded. For each protocol/operating system combination the experiment consisted of 10,000 trials. Connection tear-down measurements were not attempted as they do not directly affect data latency. In addition, the system call to close a connection may return indicating successful termination before the kernel attempts to perform the close over the network. No QOS negotiation took place during connection establishment. Such negotiation would increase the latency of connection establishment (dependent upon the QOS algorithm considered).

- **MSNL/UNIX connection faster than TCP/UNIX**: The MSNL connection establishment protocol requires only two messages. In contrast TCP, since it is a duplex protocol, requires three messages to establish a connection.

[4] HP/UX 7.0 over HP Series 9000/375 with 4μs timer resolution (50 MHz 68030).

[5] WANDA 1.1 (20 MHz 68030 with global clock hardware).

- **TCP/UNIX connection faster than ping**: This is also the case for MSNL/UNIX. Since connection management is performed wholly by the kernel it is not necessary to schedule the receiving process before the client is notified of connection establishment.

- **WANDA connection/ping faster than UNIX**: The performance of MSNL/UNIX vs. MSNL/WANDA is indicative of the poor UNIX networking structure (particularly considering that the UNIX CPU was 2.5 times faster). The anomaly of seeing faster connection than ping timings is not repeated over WANDA. This is because the server process is given the opportunity of refusing a connection request after inspecting the QOS parameters.

If the time for the server application to receive the first byte of data is approximated to be half the ping time then it can be seen that the *minimum* startup data latency for MSNL/UNIX is 5398μs (3356 + 4084/2) vs. 2318μs for UDP/UNIX. Even for relatively short conversations such a small difference may be regarded as insignificant, especially when it is considered that MSNL data transfer outperforms that for UDP. In certain circumstances latency may be further reduced by maintaining a cache of frequently used connections to peer applications. This technique is used in both the UNIX and WANDA implementation of the ANSA runtime system.

7.5.1 Summary

The flexible nature of the interface to the WANDA QOS extensions was particularly useful when performing the experiments described. In most cases no re-compilation of the kernel or application programs was necessary. The global clock boards proved invaluable in instrumenting the experimental system. The use of such hardware for taking performance measurements on *local area* distributed system configurations is advised.

All sources of jitter in an application level connection over the CBN have been identified. One outcome of the experiment was that it was found to be easy for a user thread, on an otherwise idle machine and by use of a high resolution timer, to generate a well defined false traffic load. Such a facility is useful as a source of experimental network contention traffic.

A combination of both poor kernel/device driver structure and network interface design was found to contribute significantly to the application level jitter associated with a stream. For a host handling multiple streams the CPU scheduler is a key component in deciding the order of processing. Finally, a performance analysis was made on the startup latency of some representative protocol and operating system combinations.

Chapter 8

Related Work

This chapter surveys recent related work. A small number of the more important projects and results are detailed. In each case the relationship with this dissertation is emphasised.

8.1 Internet Stream Protocol

The Internet Stream Protocol (ST) [**Forgie79**] is an IP-layer protocol that attempts to provide an end-to-end guaranteed service across an internet. ST defines the concept of a stream which exhibits all the properties of a virtual circuit. ST represents a departure from original Internet switching philosophy in that each ST gateway is required to maintain connection state. This state includes data on the network resources allocated to each stream.

8.1.1 Terrestrial Wideband Network

The Terrestrial Wideband Network (TWBNet) is a DARPA testbed for research into high-speed protocols and applications. It was built by BBN STC as a part of the initial phase of the Defense Research Internet (DRI). The network consists of Wideband Packet Switching nodes (WPS) connected to form a serial backbone using T1 trunks. The switching nodes pass traffic using the TWBNet Dual Bus Protocol (DBP) [**Edmond90**]: a link level Distributed Queue Dual Bus (DQDB) type protocol which has been extended for wide area use and bandwidth reservation. Sites connect to the backbone via IP and ST gateways. An ST host specifies connection requirements as part of stream set up. The ST gateways translate these requirements into appropriate TWBNet bandwidth reservations.

8.1.2 The DARTNET Testbed

ST-II [**Casner90**] is a re-working of the ST protocol based on experience with the TWBNet testbed. ST-II will operate in the DARTNET testbed: a multi-site network comprising a backbone of T1 lines interconnected by Sun 4 SPARC systems acting as routers. Host nodes are attached to the SPARC routers across LANs. The ST-II kernel software that will run in both the host and router nodes will be structured for easy experimentation. This is indicative of the current state of protocol research concerning support for multi-service traffic. Some of the software approaches applied in this dissertation could usefully be employed in ST-II routers and endpoints.

8.2 IMAC

Developed at the Computer Laboratory, the Integrated Multi-media Applications Communication architecture (IMAC) [**Nicolaou91**], provides a framework which facilitates the construction of multimedia applications. IMAC recognises that most such applications will be of a distributed nature. Consequently IMAC is based on the existing Advanced Networked Systems Architecture (ANSA). IMAC has been implemented as an extension to the ANSA Testbench [**ANSA89**] (an implementation of the ANSA architecture). Of particular interest are the extensions necessary to support QOS. ANSA already provided a limited form of QOS. Associated with each *instance* of an ANSA service is a set of properties. When binding to a service a client supplies a constraint specification. This is applied to the property lists of all active service instances of the requested type. The result is a list from which a particular service instance may be selected. An example of its use may be found in Section 3.1.3. Each WANDA "loader" process exports the machine name on which it is running as part of the property list. In order to run a process on a specific machine an import request is made specifying the appropriate machine name as a constraint. However, no mechanism was available which enabled ANSA applications to influence the characteristics of the underlying communication channel.

IMAC provides a mechanism for the specification of communication oriented QOS on a per-operation basis. Interface operations may specify a set of QOS options with which they are prepared to be invoked. Although QOS appears in the specification of an interface it does not contribute to the type of that interface. The QOS options are expressed as constraints on the underlying communication system. A method of mapping from application (IMAC) level QOS to communications level QOS is provided. IMAC is complementary to the work detailed in the dissertation. The RPC paradigm has been found useful in the production of a large amount of distributed software. It is likely that a large amount of network management software (Section 9.1.1) will benefit from using RPC as a basis for communication. Network

management will make use of the same communication facilities, i.e. virtual circuits, as does application code. IMAC provides the mechanism for network management code to request the high QOS it requires.

8.3 The Flow Network

The Flow Network [**Zhang89**] aims to provide users with guaranteed performance by requiring explicit resource reservation and by employing rate based traffic control. The Flow Protocol (FP) is a network level protocol that provides a simplex communication channel, *flow*, between two end users. The Flow Control Protocol (FCP) is an auxiliary protocol that assists in connection management. FCP performs flow set up and tear-down, while FP carries out the transmission phase in between. A separate control protocol alleviates the need to overload the data packet format with control information hence reducing the per packet processing overhead. This is the approach taken by MSNA and B-ISDN, i.e. specific VCIs being allocated for management purposes. The VirtualClock mechanism (Section 6.3) is used to control packet transmission in Flow Network switches.

The Flow Network architecture is able to describe the provision of real-time circuits over an ATM network. The fixed size cells of an ATM style network being a degenerate case of the variable size packets the architecture defines. In an ATM environment the VirtualClock mechanism is able to act as a policing function. Cells which cause the VirtualClock for a stream to overtake real-time by a certain threshold may be marked as illegal. The VirtualClock mechanism for a stream is reset after the arrival of a stream specific number of cells. This is an appropriate time to inform higher level management if the VirtualClock for a stream is in advance of real-time. However, the use of VirtualClock at internal network switching nodes where policing is not required (but high switching rates are) is doubtful due to the per packet processing costs. At the time of writing no implementation of the Flow Network architecture exists.

8.4 The Aurora Host Interface

The Aurora project is one of the five DARPA gigabit testbeds. It will include ATM switches that are capable of handling traffic at speeds of 622 Mbps. [**Davie91**] describes an ATM host-network interface aimed at connecting workstation class hosts to the Aurora network efficiently. The interface is targeted initially for attachment to the *turbo*channel bus [**DEC90**] on a DEC 5000 workstation. Data transfers are by DMA in order to take full advantage of the bus architecture. The lessons described

in this dissertation concerning the system support for ATM interfaces, albeit at far lower data rates, should be applicable. The developers are aware of the requirement that the interface support expedited transfer of cells and will structure the host and interface software accordingly.

At the time of writing prototype host interfaces are in the final stages of construction. Some simple experiments have been made concerning the software which will run in the *interface* itself. If, as seems likely, the project selects Ultrix as the operating system to run in the host, a radical kernel re-structuring will be necessary for application programs to benefit from the interface design.

8.5 DASH

The DASH [**Anderson88**] project aims to produce an experimental distributed system with support for real-time communication. It addresses the lack of support found in general purpose operating systems for the handling of continuous media. The DASH resource model decomposes the system components that handle continuous media into resources. For example, a CPU and its scheduler can be considered a resource. Work is given to resources in units called messages. A workload is described by a so called *linear bounded arrival process* (LBAP). Resources provide a standard interface allowing clients to create sessions with the resource. In return for specifying a LBAP, upon session establishment, a client is returned a bound on the message delay it will experience within the resource. Sessions can be combined to form end-to-end sessions spanning multiple resources.

Much of the early project work involved the construction of a kernel that supported the DASH resource model. Current work is more oriented towards applications in the UNIX environment. A discussion of how an operating system can be made to conform to the DASH resource model may be found in [**Anderson89**]. The Mach operating system (Section 8.7) is used as an example. Included in the same paper is the Continuous Media Extensions to X (CMEX) design.

8.6 B-ISDN

The B-ISDN (Section 2.3), based upon ATM switching, is proposed as the basis for the future public carrier network. The draft B-ISDN specification [**CCITT88**] defines an interface (which supports the establishment of connections with performance guarantees) to an ATM network. Techniques for extending these guarantees through user systems are outside the scope of the specification. Given the time

consuming nature of the standards process, the fact that B-ISDN/ATM is still an area active of research and also the inertia of the currently installed STM system, the widespread deployment of ATM technology is at least several years distant.

It is doubtful whether the inherent complexity of B-ISDN is appropriate in the local area. However, the same reasons that make ATM suitable for wide area multi-service traffic also apply in the local area. It is possible to envisage a single site running simple protocols similar to those of the MSNA suite over its own ATM networks. Wide area access is achieved by the purchase of B-ISDN interface(s) of the appropriate capacity. Traffic between the public and private networks goes through what is essentially an ATM router. The QOS parameter of each connection provides the mechanism for controlled sharing of this resource.

8.7 Mach

The Mach system [**Accetta86**], developed at Carnegie Mellon University (CMU), is aimed at providing a lightweight micro-kernel foundation for the UNIX operating system. Mach has been selected to form the basis of the Open Software Foundations (OSF) operating systems offering: OSF/1. Mach has evolved from the earlier RIG and Accent projects [**Rashid86**]. Mach release 2.5 has many UNIX services still integrated at the "pure" Mach micro-kernel level. The current release (3.0) has all UNIX functionality moved up to the application level. Two approaches have been attempted in implementing the user level UNIX environment:

- as a single server process.
- as a set of (distributed) processes.

A Mach process is referred to as a *task*. A task may have multiple threads associated with it. The micro-kernel is capable of functioning on multiprocessor architectures, threads being independently schedulable on any processor. Mach provides interprocess communication between threads through constructs called *ports*. All services, resources and facilities within the Mach micro-kernel and between Mach tasks are represented as ports. The address space of a Mach task is represented as a collection of mappings from linear addresses to offsets within Mach *memory objects*. The primary role of the micro-kernel in virtual memory management is to manage physical memory as a cache of the contents of memory objects.

It is likely that Mach will be the micro-kernel of choice for many future systems. The level of support it provides for multi-service traffic is therefore of particular interest. The tight integration of the Mach IPC and VM systems gives high performance for large intra-node messages. Small messages are costly due to the large VM manipulation overhead. This has been identified as an area of further study by the Mach developers. Messages destined for ports on a different machine suffer a level of indirection through a *network server*. This penalty may be reduced by placing such servers within the micro-kernel.

The popularity of Mach has resulted in several attempts at providing real-time extensions to the micro-kernel. CMU's ART (Advanced Real-Time Technology) group is developing a real-time version of the Mach micro-kernel in addition to a real-time toolset for system design and analysis [**Tokuda90**]. The new micro-kernel includes an integrated time-driven scheduler, real-time synchronisation and memory resident objects. Nakajima [**Nakajima91**] describes real-time extensions to the micro-kernel specifically in order to support multimedia applications. A significant part of the work aims at providing support for the development of multimedia device drivers in user space.

8.8 Summary

Multimedia research is increasing the demands for the support of multi-service traffic on both the computer and communications industries. The same networking technology, ATM, is suitable for fulfilling the requirements of both. This represents a further strengthening of the marriage between the two industries. A good example may be found in the Aurora gigabit network testbed. In addition to using ATM in the backbone network its use has been extended all the way to the workstation. A more conservative example is found in the TWBNet. The backbone network supports service reservations but the LANs via which most hosts connect do not.

The amount of networking and protocol research activity is not matched by that concerning support for multi-service traffic in the endpoint operating systems. The DASH system at Berkeley is perhaps the most advanced attempt to date. Since most computer generated traffic has involved mainly data applications, of which few demand transmission guarantees, this is not surprising. The approach has been that *transient* peaks in the demand on network and host capacity, leading to poor response, can be tolerated by the system users. If such transient peaks persist then more capacity must be installed. This pragmatic approach is no longer tenable when multi-service traffic is concerned.

Chapter 9

Further Work

The realisation of an ATM service quality network offering QOS guarantees is still in the future. The work described in this dissertation is only in partial fulfillment of such a goal. However, the infra-structure which has been developed is stable enough that it is being used as the basis for future research. Some of the work outlined below will be continued within the MSNA framework at Cambridge. A division is made between future network and applications research.

9.1 Network Research

The physical hardware represents a fraction of the total effort necessary to build a service network. Whilst a great deal of work will continue in improving the performance of ATM hardware the principles upon which the controlling software is built should not change. The controlling software will be built using the products of distributed systems research. Properly designed, it will emphasise properties of:

- scaling to large configurations.
- fault tolerance.
- lack of centralised dependencies.
- flexible component placement.
- remote operation and upgrade.

9.1.1 Network Management

The comparatively small network described in Chapter 4 demands a significant amount of maintenance time from experienced personnel. A larger network would dictate the total or partial automation of most of these management functions. Some of the issues involved in network management are:

- fault management - fault detection and correction, trouble reports.
- configuration management - topology and routing changes.
- capacity management - identification of trends in the network.
- security management - intrusion identification, secured access.
- accounting management - billing reports.

Network management, akin to electronic mail, is a naturally distributed application. A significant amount of network management software will be developed within the framework of a distributed computing system e.g. ANSA/IMAC or Network Computing System (NCS) [**Kong90**]. A large amount of experience exists with simple Network management systems, an example protocol being the Simple Network Management Protocol (SNMP) [**Case90**] from the DARPA Internet family. It is an open question as to how much of such research can be applied to the management of ATM networks offering QOS.

9.1.2 Routing

The interaction of QOS and routing is an interesting area for future research. A knowledge of network topology allows a set of candidate paths to a given destination to be computed. A knowledge of the link and router capacity along each path in conjunction with the QOS requested allows the set of paths to be refined. So far the only information which has been used is of a slow changing nature and can be cached close to the connection originator. Of the candidate paths not all may be available due to lack of spare capacity or failed components.

One useful observation is that a large percentage of connection requests will be for the local area. In this case algorithms which make use of the *full* local configuration and state may be more applicable [**Snyder89**]. The Desk Area Network (DAN) (Section 9.2) is a good example where maintaining the full global state of the network is practical.

9.1.3 Fault Tolerance

A large amount of effort can be expended in making the probability of individual component (both software and hardware) failure in the network small. However, in a large network, failures will occur regularly. The effects of any such failure should be minimised. One basic mechanism for implementing network fault tolerance is call redirection. Call redirection is here defined as the ability to alter the route of a connection without causing a connection fail indication to either of the endpoints. Some scenarios in which it may be useful are:

- re-routing of calls established through a failed router.
- migrating all calls of a router prior to taking it down.
- network management may wish to alter the call distribution.

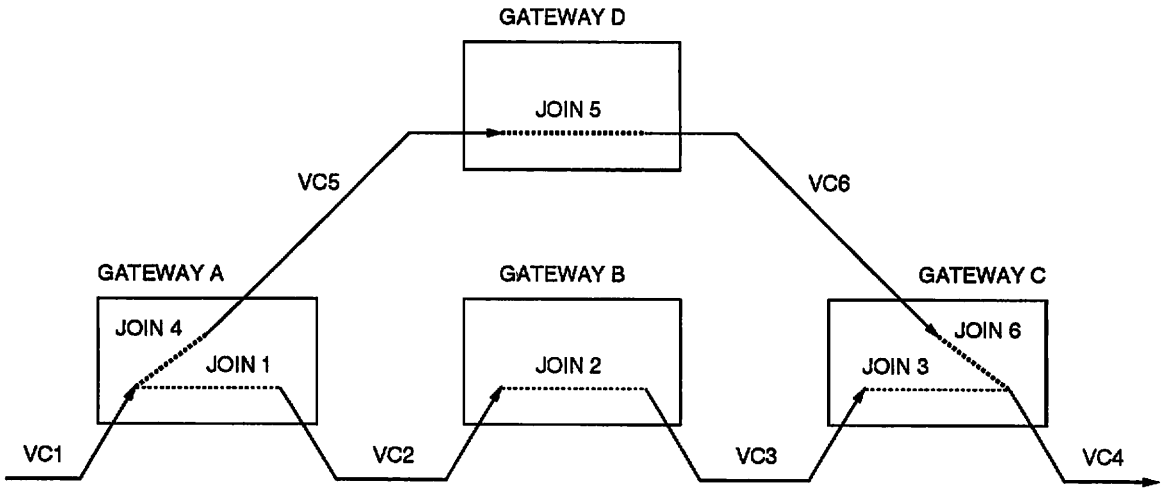

Figure 9.1: Single Hop Call Re-routing

In the general case the replaced part of a call will consist of one or more hops, the replacing path will also consist of one or more hops. The replacement path should have a QOS consistent with the original, unless QOS contributions are re-negotiated at other components along the call path. To perform a single hop redirect on the established call (VC1-4) through gateway D as illustrated in Figure 9.1 the following steps must be taken:

1. establish VC5, VC6 and JOIN 5
2. establish JOIN 6 (breaking JOIN 3)

3. establish JOIN 4 (breaking JOIN 1)

4. delete VC2 and VC3

Steps 2 and 3 may be performed concurrently. If step 2 is performed before step 3 then cells arriving on VC3 will be forwarded to sink. If step 3 is performed before step 2 then cells arriving on VC6 will be forwarded to sink. In either case, cells may be lost. If such cell loss is not consistent with the QOS for the call then the circuit may have to be closed and the terminal points notified.

In the case of a *scheduled* redirect the above process is made simpler by gateway B being alive. Gateway A is able to query gateway B for the location of the next hop gateway (C). In the case of a redirect being initiated because of gateway B's failure, gateway A is unable to easily ascertain the list of gateways downstream from B which support the call. One solution is for this information to be returned as part of connection establishment. Another is for the query process to begin once the call has been established.

9.1.4 Hardware Implementation

One of the advantages of the simplicity involved in ATM cell processing is the possibility of implementation in hardware. Research is needed to determine what the functional interface to such hardware should be. The Xilinx programmable gate array [**Xilinx89**] in the Fairisle Port Controller allows easy experimentation with new interface designs. Already experience with the prototype has dictated changes to the Xilinx configuration.

9.2 Application Research

No matter how much care has been taken in the design of an interface its utility can only be determined by active use. The development of an ATM network offering QOS is driven by applications development. If an application is unable to perform an operation it thinks natural then it is perhaps a candidate for integration into the network architecture e.g. the provision of a multicast facility. The migration of existing applications to use ATM in an optimal manner will be gradual. Section 4.2 showed how a large portion of existing data (IP) traffic can immediately be carried over an ATM network. The widespread installation of optical fibre in the Cambridge area, as part of Project Granta [**Cook91**], offers the promise of constructing more realistic testbeds. ATM application research at Cambridge will continue through a number of major projects.

Pandora 2

The Pandora 2 project at Olivetti Research Ltd. aims to build a second generation multimedia workstation based on the lessons learnt from Pandora. The multimedia devices and network interface will be inter-connected by a standard workstation bus: the *turbo*channel [**DEC90**]. The desire to support higher quality media has necessitated using the CBN, instead of CFR, as the local distribution network.

The Desk Area Network

The Desk Area Network (DAN) [**Hayter91b**] is a proposal to build a multimedia workstation using an ATM switch to inter-connect the various workstation components. The interface between the DAN and a high speed ATM network is simply a cell router. The ATM switch will be constructed from Fairisle port controllers (Section 2.2.1) running the WANDA kernel. One device of particular interest is a HDTV frame buffer that has been developed at the Computer Laboratory.

Video File Server

Applications such as video mail require the permanent storage of real-time media i.e voice and video. [**Jardetzky92**] is an attempt to build a file server capable of handling the requirements of these media, WANDA being used as the base kernel for the file server. It will be interesting to see what further system support, particularly concerning stream synchronisation, this project may require.

Chapter 10

Conclusion

This dissertation has examined the systems issues associated with supporting multi-service traffic in an ATM environment. Practical work has been carried out over a testbed comprising heterogeneous ATM network implementations including the 512 MHz Cambridge Backbone Network. The testbed has demonstrated experimentally the ability of an ATM based network to *concurrently* handle the different requirements of multi-service traffic: voice, video (Pandora) and data (IP/MSNA). In addition to dedicated multimedia source and sinks, the network included hosts running a locally developed micro-kernel (WANDA). The virtual circuit model was shown not to compromise the operation of a comprehensive distributed programming environment (ANSA). Attempts to handle multi-service traffic at the WANDA application layer demonstrated shortcomings in the design of the kernel and its associated ATM interfaces. Such shortcomings are also apparent in current general purpose operating systems, e.g. UNIX. The amount of delay and more importantly jitter introduced by the host can be orders of magnitude greater than that of the physical network. All major sources of jitter inside the host have been identified. In some cases these can be removed by a simple re-working of the kernel structure. Suggestions leading to further jitter reduction have been made concerning the design of current host network interfaces. As CPU performance improves, traffic previously handled by dedicated hardware will more often be processed by application software. This move will be driven both by the high cost of dedicated hardware and the flexibility offered by a software solution. A large percentage of multimedia processing is likely to take place in the workstation environment. Unless the operating system kernels for such machines are implemented with a view to handling multi-service traffic, then hybrid solutions such as that evident in the Pandora system will pre-dominate.

Bibliography

[Abrossimov89] V. Abrossimov, M. Rozier, and M. Gien. *Virtual Memory Management in Chorus.* In Proc. of the European Workshop, Lecture Notes In Computer Science, Berlin, FRG, April 1989. Springer-Verlag. (p 26)

[Accetta86] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young. *Mach: A New Kernel Foundation For UNIX Development.* In Proc. USENIX Summer Conference, August 1986. (pp 18, 91)

[AMD85] Advanced Micro Devices. *Local Area Network Controller Am7990 (Lance)*, 1985. (p 9)

[Anderson88] D. P. Anderson and D. Ferrari. *The DASH Project: An Overview.* Technical Report, Computer Science Division. University of California, Berkeley, February 1988. (p 90)

[Anderson89] D. P. Anderson, R. Govidan, G. Homsy, and R. Wahbe. *Integrated Digital Continuous Media: A Framework based on Mach, X11 and TCP/IP.* Technical Report, Computer Science Division. University of California, Berkeley, March 1989. (p 90)

[Angebranndt88] S. Angebranndt, R. Drewry, P. Karlton, and T. Newman. *Definition of the Porting Layer for the X V11 Sample Server*, March 1988. (p 18)

[ANSA89] Architecture Projects Management. *The ANSA Reference Manual*, March 1989. (pp 3, 20, 88)

[Ash89] G. R. Ash and E. Oberer. *Dynamic Routing in the AT&T Network - Improved Service Quality at Lower Cost.* In Proc. IEEE Global Telecomm. Conf., November 1989. (p 39)

[Ash90] G. R. Ash. *Design and Control of Networks with Dynamic Nonhierarchical Routing.* IEEE Comm. Magazine, October 1990. (pp 37, 47)

[Beeler91] R. Beeler. *Fairisle VME Interface.* Technical Report 219, Cambridge University Computer Laboratory, April 1991. (p 10)

[Bellamy82] J. Bellamy. *Digital Telephony.* Wiley, 1982. (p 43)

[Bershad90] B. N. Bershad, T. E. Anderson, E. D. Lazowska, and H. M. Levy. *Lightweight Remote Procedure Call.* ACM Trans. on Computer Systems, 8(1), February 1990. (p 23)

[Birrell89] A. Birrell. *An Introduction to Programming with Threads.* Technical Report 35, DEC Systems Research Centre, January 1989. (pp 19, 75)

[Braden87] R. Braden and J. B. Postel. *Requirements for Internet Gateways.* RFC-1009, June 1987. (p 27)

[Caceres91] R. Caceres, P. B. Danzig, S. Jamin, and D. J. Mitzel. *Characteristics of Wide-Area TCP/IP Conversations.* In Proc. ACM SIGCOMM, September 1991. (pp 41, 46, 85)

[Case90] J. D. Case, M. Fedor, M. L. Schoffstall, and C. Davin. *Simple Network Management Protocol (SNMP).* RFC-1157, May 1990. (p 94)

[Casner90] S. Casner, K. Seo, W. Edmond, and C. Topolcic. *Experimental Internet Stream Protocol Version 2 (ST-II).* RFC-1190, October 1990. (p 88)

[CCITT88] CCITT Study Group XVIII Draft Recommendation I.121. *Broadband aspects of ISDN*, February 1988. (pp 1, 90)

[Cheriton88] D. R. Cheriton. *The V Distributed System.* Comm. of the ACM, 31(3), March 1988. (pp 24, 25)

[Clark85] D. Clark. *The Structuring of Systems using Upcalls.* In Proc. ACM SIGOPS, December 1985. (p 75)

[Cocchi91] R. Cocchi, D. Estrin, S. Shenker, and L. Zhang. *A Study of Priority Pricing in Multiple Service Class Networks.* In Proc. ACM SIGCOMM, September 1991. (p 45)

[Coffman73] E. G. Coffman and P. J. Denning. *Operating Systems Theory.* Prentice-Hall, 1973. (p 52)

[Cook91] A. Cook et al. *Report of the Computer Syndicate.* Cambridge University Reporter (published by authority), 5481, August 1991. (p 97)

[Cooper88] R. C. B. Cooper. *Debugging Concurrent and Distributed Programs.* Technical Report 128, Cambridge University Computer Laboratory, February 1988. Ph.D. dissertation. (p 22)

[Corbet91] J. Corbet. *The MIT Shared Memory Extension.* X11 Release 5 Documentation, September 1991. (p 38)

[Davie91] B. S. Davie. *A Host-Network Interface Architecture for ATM.* In Proc. ACM SIGCOMM, September 1991. (p 89)

[Day83] J. Day and H. Zimmerman. *The OSI Reference Model.* In Proc. of the IEEE, December 1983. (p 16)

[DEC86] Digital Equipment Corporation. *DEQNA Ethernet: User's Guide*, 2nd edition, September 1986. (p 9)

[DEC90] Digital Equipment Corporation. *Turbochannel Developers Kit Version 2*, September 1990. (pp 89, 97)

[Demers90] A. Demers, S. Keshav, and S. Shenker. *Analysis and Simulation of a Fair Queueing Algorithm.* Internetworking: Research and Experience, 1(1), 1990. (pp 51, 53)

[Dijkstra68] E. W. Dijkstra. *The Structure of the THE Multiprogramming System.* Comm. of the ACM, 11(5), May 1968. (p 19)

[Dittmann88] L. Dittmann and S. B. Jacobsen. *Statistical Multiplexing of Identical Bursty Sources in an ATM Network.* In Proc. IEEE Globecomm 88, November 1988. (p 44)

[Edmond90] W. Edmond, K. Seo, M. Leib, and C. Topolcic. *The DARPA Wideband Network Dual Bus Protocol.* In Proc. ACM SIGCOMM, September 1990. (p 87)

[Ellis88] J. R. Ellis, K. Li, and A. W. Appel. *Real-time Concurrent Collection on Stock Multiprocessors.* Technical Report 25, DEC Systems Research Centre, February 1988. (p 23)

[Ferrari90] D. Ferrari. *A Scheme for Real-Time Channel Establishment in Wide-Area Networks.* IEEE Journal on Selected Areas in Communications, 8(3), April 1990. (p 47)

[Forgie79] J. W. Forgie. *ST - A Proposed Internet Stream Protocol.* IEN 119, September 1979. (p 87)

[Garnett83] N. H. Garnett. *Intelligent Network Interfaces.* Technical Report 46, Cambridge University Computer Laboratory, September 1983. Ph.D. dissertation. (p 79)

[Golub90] D. Golub, R. Dean, A. Forin, and R. Rashid. *Unix as an Application Program.* In Proc. USENIX Summer Conference, August 1990. (p 18)

[Greaves90] D. J. Greaves, D. Lioupis, and A. Hopper. *The Cambridge Backbone Ring.* Technical Report 2, Olivetti Research Ltd., Cambridge, February 1990. (pp 2, 7)

[Greaves91] D. J. Greaves. *Cambridge Backbone Ring Half-Duplex VME Station.* Olivetti Research Ltd., Internal Documentation, 1991. (p 8)

[Harita91] B. R. Harita. *Dynamic Bandwidth Management.* Technical Report 217, Cambridge University Computer Laboratory, April 1991. Ph.D. dissertation. (p 36)

[Hayter91a] M. D. Hayter and R. J. Black. *Fairisle Port Controller: Design and Ideas.* Technical Report 219, Cambridge University Computer Laboratory, April 1991. (p 10)

[Hayter91b] M. D. Hayter and D. R. McAuley. *The Desk Area Network.* Technical Report 228, Cambridge University Computer Laboratory, May 1991. (p 97)

[Herrmann88] F. Herrmann et al. *Chorus: A New Technology for Building UNIX Systems.* In Proc. EUUG Autumn 88 Conference, October 1988. (p 18)

[Hirano89] M. Hirano and N. Watanabe. *Characteristics of a Cell Multiplexer for Bursty ATM Traffic.* In Proc. IEEE ICC 89, June 1989. (p 44)

[Hopper88] A. Hopper and R. M. Needham. *The Cambridge Fast Ring Networking System.* IEEE Trans. Computers, 37(10), October 1988. (pp 2, 7)

[Hopper90] A. Hopper. *Pandora - An experimental system for multimedia applications.* ACM Operating Systems Review, 24(2), April 1990. (pp 2, 33)

[Jacobson88] V. Jacobson. *Congestion Avoidance and Control.* In Proc. ACM SIGCOMM, August 1988. (p 67)

[Jardetzky92] P. W. Jardetzky. *Network File Service Design for Continuous Media*, 1992. Cambridge University Computer Laboratory, Ph.D. thesis in preparation. (p 97)

[Jungok91] J. Jungok and T. Suda. *A Survey of Traffic Control Schemes and Protocols in ATM Networks.* In Proc. of the IEEE, February 1991. (pp 41, 44)

[Kong90] Kong et al. *Network Computing System Reference Manual.* Prentice Hall, 1990. (p 94)

[Lamport78] L. Lamport. *Time, Clocks, and the Ordering of Events in a Distributed System.* Comm. of the ACM, 21(7), July 1978. (p 68)

[Leffler89] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman. *The Design and Implementation of the 4.3BSD UNIX Operating System.* Addison-Wesley, Reading, Ma., 1989. (p 23)

[Leslie91] I. M. Leslie and D. R. McAuley. *Fairisle: An ATM Network for the Local Area.* In Proc. ACM SIGCOMM, September 1991. (pp 2, 9)

[Maglaris87] B. Maglaris. *Performance Analysis of Statistical Multiplexing for Packet Video Sources.* In Proc. IEEE Globecomm 87, November 1987. (p 44)

[Mankin90] A. Mankin. *Random Drop Congestion Control.* In Proc. ACM SIGCOMM, September 1990. (p 51)

[Mapp91] G. E. Mapp. *An Object Oriented Approach to Virtual Memory Management*, 1991. Cambridge University Computer Laboratory, Ph.D. thesis in preparation. (p 18)

[McAuley90] D. R. McAuley. *Protocol Design For High Speed Networks.* Technical Report 186, Cambridge University Computer Laboratory, January 1990. Ph.D. dissertation. (pp 2, 5, 38)

[Metcalfe76] R. M. Metcalfe and D. R. Boggs. *Ethernet: Distributed packet switching for local computer networks.* Comm. of the ACM, 19(6), July 1976. (p 9)

[Mills89] D. L. Mills. *Internet Time Synchronization: The Network Time Protocol.* Network Working Group RFC 1129, October 1989. (p 68)

[Mogul87] J. C. Mogul, R. F. Rashid, and M. J. Accetta. *The Packet Filter: An Efficient Mechanism for User-Level Network Code.* Technical Report 2, DEC Western Research Laboratory, November 1987. (p 28)

[Nagle87] J. Nagle. *On Packet Switches with Infinite Storage.* IEEE Trans. Comm., 1987. (p 53)

[Nakajima91] J. Nakajima, M. Yazaki, and H. Matsumoto. *Multimedia/Realtime Extensions for the Mach Operating System.* In Proc. USENIX Summer Conference, June 1991. (p 92)

[Needham82] R. M. Needham and A. J. Herbert. *The Cambridge Distributed Computer System.* Addison-Wesley, Reading, Ma., 1982. (p 17)

[Newman89] P. Newman. *Fast Packet Switching for Integrated Services.* Technical Report 165, Cambridge University Computer Laboratory, 1989. Ph.D. dissertation. (p 2)

[Nicolaou91] C. Nicolaou. *A Distributed Architecture for Multimedia Communication Systems.* Technical Report 220, Cambridge University Computer Laboratory, May 1991. Ph.D. dissertation. (pp 3, 42, 47, 88)

[Ofek89] Y. Ofek. *Generating a Fault Tolerant Global Clock in a High Speed Distributed System.* In Proc. 9th Int. Conf. on Distributed Computing Systems, June 1989. (p 68)

[Ousterhout82] J. K. Ousterhout. *Scheduling Techniques for Concurrent Systems.* In Proc. 3rd Int. Conf. on Distributed Computing Systems, October 1982. (p 19)

[Patterson80] D. A. Patterson and D. R. Ditzel. *The case for the reduced instruction set computer.* ACM Computer Architecture News, 8(6), October 1980. (p 18)

[Postel80] J. B. Postel. *User Datagram Protocol.* RFC-768, August 1980. (p 38)

[Postel81a] J. B. Postel. *The Internet Protocol.* RFC-791, September 1981. (pp 35, 39)

[Postel81b] J. B. Postel. *Transmission Control Protocol.* RFC-793, September 1981. (pp 37, 46)

[Powell83] M. Powell and B. Miller. *Process Migration in DEMOS/MP.* In Proc. 9th ACM SOSP, October 1983. (p 22)

[Rashid86] R. F. Rashid. *From RIG to Accent to Mach: The Evolution of a Network Operating System.* In Proc. of the ACM/IEEE Computer Society, Fall Joint Computer Conference, November 1986. (p 91)

[**Redell88**] D. D. Redell. *Experience with Topaz TeleDebugging.* In ACM SIGPLAN and SIGOPS: Workshop on Parallel and Distributed Debugging, May 1988. (p 22)

[**Sandberg85**] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. *Design and Implementation of the Sun Network Filesystem.* In Proc. of the USENIX Summer 1985 Conference. USENIX Association, July 1985. (p 18)

[**Scheifler86**] R. W. Scheifler and J. Gettys. *The X Window System.* ACM Transactions on Graphics, 5(2), April 1986. (pp 18, 33)

[**Schroeder89**] M. Schroeder and M. Burrows. *Performance of Firefly RPC.* Technical Report 43, DEC Systems Research Center, April 1989. (pp 25, 67)

[**Snyder89**] J. M. Snyder. *A Routing Architecture For Very Large Networks Undergoing Rapid Reconfiguration.* In Proc. ACM SIGCOMM, September 1989. (p 94)

[**Sreenan90**] C. J. Sreenan. *Synchronised Retrieval of Multi-Media Data.* In First Int. Workshop on Network and Operating System Support for Digital Audio and Video, November 1990. (p 48)

[**SUN86**] Sun Microsystems Inc. *Remote Procedure Call Protocol Specification*, February 1986. (pp 3, 18)

[**SUN91**] Sun Microsystems Inc. *ANSI T1S1.5 / Simple and Efficient Adaptation Layer (SEAL)*, August 1991. (p 15)

[**Swinehart83**] D. C. Swinehart, L. C. Stewart, and S. M. Ornstein. *Adding Voice to an Office Computer Network.* In Proc. IEEE Global Telecomm. Conf., November 1983. (p 40)

[**Swinehart85**] D. C. Swinehart, P. T. Zellweger, and R. B. Hagmann. *The Structure of Cedar.* In Proc. of the ACM SIGPLAN 85 Symposium on Language Issues in Programming Environments, June 1985. (p 19)

[**Tanenbaum87**] A. Tanenbaum and S. Mullender. *An Overview of the Amoeba Distributed Operating System.* ACM Operating Systems Review, 15(3), July 1987. (pp 3, 18, 24, 25)

[**Tennenhouse87**] D. L. Tennenhouse. *Exploiting Wideband ISDN: The Unison Exchange.* In Proc. IEEE INFOCOM 87, April 1987. (p 36)

[**Tennenhouse89**] D. L. Tennenhouse. *Layered Multiplexing Considered Harmful.* In Protocols for High Speed Networks, IBM Zurich Research Lab., May 1989. IFIP WG.1/6.4 Workshop. (p 16)

[**Thacker87**] C. Thacker, L. Stewart, and E. Satterthwaite Jr. *Firefly: A Multiprocessor Workstation.* Technical Report 23, DEC Systems Research Centre, December 1987. (pp 3, 25, 74)

[**Theimer85**] M. Theimer, K. Lantz, and D. R. Cheriton. *Preemptable Remote Execution Facilities for the V System.* In Proc. 10th ACM SOSP, December 1985. (p 22)

[**Tokuda90**] H. Tokuda et al. *Real-Time Mach: Towards a Predictable Real-Time System.* In Proceedings of USENIX Mach Workshop, Burlington, Vermont. USENIX Association, October 1990. (p 92)

[**Trevanian87**] A. Trevanian, R. Rashid, M. Young, D. Golub, R. Baron, D. Black, D. W. Bolosky, and J. Chew. *Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures.* In Proc. of the Second Symposium on Architectural Support for Programming and Operting Systems, October 1987. (p 26)

[**Turner86**] J. S. Turner. *New Directions in Communications (or Which Way to the Information Age ?).* IEEE Comm. Magazine, 24(10), October 1986. (p 60)

[**VLSI87**] VLSI Inc. *ARM Datasheet*, 1987. (p 3)

[**VLSI90**] VLSI Inc. *ARM3 Datasheet*, 1990. (p 10)

[**Xerox81**] Xerox Corporation. *Internet Transport Protocols*, December 1981. (p 36)

[**Xilinx89**] Xilinx Inc. *The Programmable Gate Array Data Book*, 1989. (p 96)

[**Zayas87**] E. Zayas. *The Use of Copy-On-Reference in a Process Migration System.* PhD thesis, CMU, January 1987. (p 22)

[**Zhang89**] L. Zhang. *A New Architecture for Packet Switching Network Protocols.* PhD thesis, MIT, July 1989. (p 89)

[**Zhang90**] L. Zhang. *VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks.* In Proc. ACM SIGCOMM, September 1990. (pp 51, 55, 60)