Protocol Design for High Speed Networks

Derek Robert McAuley

Fitzwilliam College
University of Cambridge

A dissertation submitted for the degree
of
Doctor of Philosophy

September 1989

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.

Trademarks

Acknowledgements

I would like to thank my supervisor, Ian Leslie, for encouragement and valuable advice during my research. I would also like to thank the various members of the Computer Laboratory who have provided assistance and useful discussions during the course of my research, notably Roger Needham, David Tennenhouse, David Greaves, Cosmos Nicolaou and Joe Dixon. Thanks are also due to Geoff Walsham who provided sound advice and guidance when most needed.

I am also indebted to Ian Leslie, Cosmos Nicolaou, Joe Sventek and Paul Jardetzky, who have read and commented on earlier drafts of this dissertation, and to Helen Moss who proof-read the final text.

This work was supported by a studentship from the Science and Engineering Research Council.

Abstract

Improvements in fibre optic communication and in VLSI for network switching components have led to the consideration of building digital switched networks capable of providing point to point communication in the gigabit per second range. Provision of bandwidths of this magnitude allows the consideration of a whole new range of telecommunications services, integrating video, voice, image and text. These multi-service networks have a range of requirements not met by traditional network architectures designed for digital telephony or computer applications. This dissertation describes the design, and an implementation, of the Multi-Service Network architecture and protocol family, which is aimed at supporting these services.

Asynchronous transfer mode networks provide the basic support required for these integrated services, and the Multi-Service Network architecture is designed primarily for these types of networks. The aim of the Multi-Service protocol family is to provide a complete architecture which allows use of the full facilities of asynchronous transfer mode networks by multi-media applications. To maintain comparable performance with the underlying media, certain elements of the MSN protocol stack are designed with implementation in hardware in mind. The interconnection of heterogeneous networks, and networks belonging to different security and administrative domains, is considered vital, so the MSN architecture takes an internetworking approach.

Contents

List of Tables

List of Figures

Glossary

CBN Cambridge Backbone Network: a 1 Gbit/s slotted ring network

CCITT Comiteé Consultatif International Télégraphique et Téléphonique

CFR Cambridge Fast Ring: a 75Mbit/s slotted ring network

CLNP ConnectionLess Network Protocol: a definition of the OSI network layer

CODEC COder-DECoder: a device for conversion between digital and analogue signals

DPCM Differential Pulse Code Modulation

DQDB Distributed Queue Dual Bus: a slotted dual bus network with support for synchronous channels

FDDI Fiber Distributed Data Interface: a 100 Mbit/s token ring network

FDM Frequencey Division Multiplexing

ICMP Internet Control Message Protocol

IP Internet Protocol: the internetworking protocol of the ARPA Internet

ISDN Integrated Services Digital Network

ISO International Standards Organization

MAC Media ACcess layer

MS-Access Multi-Service media Access

MSDL Multi-Service DataLink

MSNL Multi-Service Network Level

MSN Multi-Service Network

MSSAR Multi-Service Segmentation And Reassembly

PCM Pulse Code Modulation

PTM Packet Transfer Mode: CCITT nomenclature for asynchronous time division multiplexing

PTT Post, Telephone and Telegraph: the bodies within a country licensed to provide public data transmission services

QOS Quality Of Service

QPSX Queue Packet and Synchronous eXchange: precursor to DQBD network

RPC Remote Procedure Call

SDM Space Division Multiplexing

STM Synchronous Transfer Mode: CCITT nomenclature for synchronous time division multiplexing

TCP Transmission Control Protocol: the reliable transport protocol of the ARPA Internet

TDM Time Division Multiplexing

Unity A remote procedure call for Modula2 and other languages

VPI Virtual Path Identifier

Chapter 1

Introduction

This dissertation presents the design and an implementation of a communication architecture for a modern high speed internetwork, the Multi-Service Network (MSN) architecture. This deals with both local and wide area networks, and aims to provide support for the new classes of applications made possible by the advances in network technology which promise to supply significantly higher bandwidths than previously available.

In terms of the service requirements made on a network, a distributed application utilizing user interaction mechanisms such as, voice, video and computer generated visualization, place the most stringent demands on the network. An application of this form requires an *integrated service*, or *multi-service* network, which effectively supports real time traffic, such as digital voice and video, as well as distributed computing traffic. These requirements are radically different from those placed on current internetworks by standard applications, such as remote login, mail and file transfer, both in terms of required performance, and type and quality of service.

The ability to support heterogeneous network types within an internetwork is a vital requirement if the network is not to be obsolete as soon as it is built. An internetwork architecture capable of supporting heterogeneous sub networks provides the mechanism to allow gradual integration of new network technologies, without the requirement to change higher level protocols and applications, and the ability to utilize different mechanisms for local and wide area interconnection.

To be successful, a communication architecture must address *all* aspects of networking, from the formatting of bits and the multiplexing mechanism for physical transmission across a cable or fibre optic, all the way to the application itself. In terms of the International Standards Organization Open Systems Interconnection (ISO OSI) reference model, this requires attention to all seven protocol levels, from media access all the way up to the application level.

1.1 Issues

High bit rates at the physical network level are possible because of advances in integration technology and optical media. Bit rates in the range 100Mb/s to 1Gb/s are feasible with current technology, and even higher rates will be available in the future. The current design of many network architecture and protocol suites requires implementation in software, so the current rapid increase in network bandwidth means that a larger and larger percentage of host processor time is spent in network processing. Protocols which can be wholly, or partly, implemented in hardware could lead to a substantial reduction in the network load on a host processor. The use of similar integration technology to that currently used to implement the physical and media access levels, would allow a protocol engine implementing higher level protocols to track the underlying network in performance. The architecture presented here addresses issues required to enable protocol implementation in hardware.

Distributed computing systems and integrated services based on packet switched local area networks have been the subject of research at a number of establishments. These systems have not, in general, been extended into heterogeneous, or wide area networks, as they present network requirements which cannot be met by current interconnection architectures. For example, distributing computing mechanisms, such as remote procedure call (RPC) [Birrell 84], are sensitive to long round trip delays, while real time traffic requires constraints on the variation in delay, or *jitter*. In an internetwork, the largest source of delay and jitter is often the internetwork switching nodes, or routers. The architecture presented here addresses the issues raised in interconnecting heterogeneous networks.

The performance obtained from a network, as seen by an application, is often dominated by the performance of the implementation of functions associated with the upper levels of the OSI reference model. In turn, the implementation of these functions is often influenced by the particular host operating system environment, rather than by the underlying physical and datalink performance. The architecture presented here addresses issues of efficiency of implementation.

To obtain the full benefits of an integrated network approach, hosts and network interfaces need to support multiple simultaneous conversations, conveying a range of different traffic types, with a number of peer hosts. The interference between these parallel conversations must be minimized, in particular real time traffic should not be affected by other less time critical traffic. The architecture presented here addresses issues of association interference at all protocol levels.

1.2 Context

The implementation work in support of this thesis has been performed using a number of different networks, and in a range of different operating systems environments.

The thesis is concerned with an architecture for a multi-service network. In section 2.4.1 an asynchronous transfer mode (ATM) network, and in particular the Cambridge fast ring (CFR), is presented as providing a suitable base for such work. Certain aspects of the architecture, for example internetworking, are best related to currently deployed architectures and implementations, and in this case work has also been performed using Ethernet.

As others have found, experimental networking within currently deployed operating systems, such as the UNIX kernel, can be a time consuming and unrewarding process. Besides the unfriendliness of the runtime environment, operating system overheads can completely mask performance differences due to the network architecture [Nordmark 89]. For this reason most of the network and datalink level experimentation, has been performed within a simple message passing kernel, MICA, running over an Acorn Archimedes personal computer. This provides a *lightweight* kernel, tailored to networking, in which observations of the relative performance of different network architectures can be made.

Work concerned with demonstrating the utilization of the upper levels of the architecture (in terms of the OSI model, the transport, session and presentation levels) by a distributed computing mechanism, has been performed over UNIX, using the Unity RPC mechanism [McAuley 87].

1.3 Outline

Chapter 2 presents a characterization of digital networks in terms of their multiplexing mechanisms. The case for the use of asynchronous transfer mode is presented, along with the general mechanisms to support traffic normally conveyed on circuit and packet switched networks. A number of implementations and designs for ATM networks are also described, as it is at these types of networks that the architecture is aimed.

Details of, and experience with, previous internetworking mechanisms (X-25, DARPA Internet, and the Universe and Unison projects) are presented in chapter 3. The types of service supported by these various networks are described, and lessons for the MSN architecture drawn.

Chapter 4 deals with the general requirements of the MSN architecture. The design methodology used for MSN is presented, and its implications for hosts, network controllers and network switches. Finally an overview of the MSN architecture is presented in terms of a reference model, which is related to the ISO OSI reference model.

The detailed design and implementation work for MSN is split into two chapters, representing the different environments in which these two elements of the work were performed. Chapter 5 deals with the lower levels of the protocol stack, in terms of ISO OSI reference model, the media access to network levels. Chapter 6 deals with the higher level issues, and a particular distributed computing use, RPC. Both chapters present the design decisions taken, the implementation work performed, and experimental results obtained.

Related work in various areas of network research is presented in chapter 7. This relates aspects of the MSN architecture to a number of different pieces of ongoing research work, and compares the relative merits of the different approaches, and their applicability for multi-service use.

The MSN architecture presents the basis for a multi-service network, and chapter 8 presents further work concerning both the architecture, and its effective implementation and utilization.

Concluding remarks are presented in chapter 9.

Chapter 2

Networks

The types of service that must be supported in a multi-service network environment, include digital voice and video, as well as distributed computing and computer generated imaging traffic. Before embarking on the description of the design of the Multi-Service network (MSN) architecture, it is useful to look at the interaction between existing service types, and the characteristics of their network environment.

2.1 Network characteristics

The feasible service types which can be supported on any given network are related to two characteristics of the underlying network:

- inter-switch bandwidth,
- multiplexing mechanism.

The relation of services to inter-switch bandwidth is straight forward. Networks have grown from a few bits per second for telegraph systems to modern fibre optic transmission lines and high speed opto-electronic components allowing line rates in the gigabit per second range. The availability of these inter-switch bandwidths allows the consideration of new services, such as high definition digital video, which have high bandwidth requirements.

When considering building a network, the inter-switch bandwidth cannot be considered in isolation; the available bandwidth through network switching nodes must also be considered. As will be shown, this bandwidth is mostly dictated by the multiplexing mechanism being used, and so consideration is given to the effect on switch bandwidths when considering the various possible multiplexing mechanisms. The multiplexing factor can become even more significant when considering interconnecting heterogeneous networks into an internet.

The following section presents an overview of multiplexing techniques, and the services supported on networks using these different techniques. Finally, the case for ATM networks is presented, along with the description of a number of different implementations.

2.2 Multiplexing

As applied to the subject of networking, multiplex is defined in the Oxford English dictionary as:

> **multiplex** *a.* Manifold, of many elements; involving simultaneous transmission of several messages along a channel of communication.

To rephrase, multiplexing is concerned with *the simultaneous use of a common channel by a number of subchannels.* To be effective then, a multiplexing mechanism must provide a mechanism for the separation of the different symbols associated with the different subchannels. There are three main mechanisms widely used:

- separation by frequency - frequency division multiplexing (FDM),
- separation by distance - space division multiplexing (SDM),
- separation by time - time division multiplexing (TDM).

Networks exist based almost solely on FDM and SDM techniques (for example, broadcast TV and radio). These techniques allow bidirectional communication with only one correspondent *at a time*, whereas in an integrated digital network, it is desirable to be able to communicate with many correspondents concurrently. Hence, although both FDM and SDM are used within digital networks to increase total network bandwidth, each of the FDM or SDM channels is further multiplexed by the use of TDM. In fact within many architectures, TDM is performed in a layered fashion within many of the layers of the associated protocol suite. The OSI ISO protocol suite currently provides for multiplexing within six of the seven layers of the model[1].

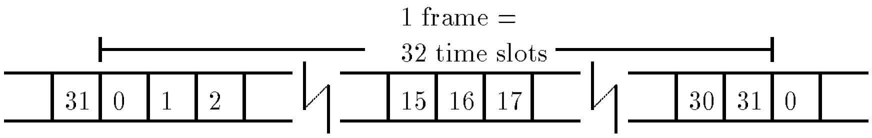
[1] The presentation layer is the exception, although for *architectural consistency*, the naming and addressing scheme incorporates presentation level address selectors [ISO 74983].

The characteristics of various networks, and hence the services they support, are almost entirely dictated by the exact design and implementation of this layered sequence of time division multiplexing mechanisms. There are two extremes, traditionally referred to as *circuit* switched, and *packet* switched networks. Perhaps a more useful way to characterize these various TDM strategies is by the *association* switch rate and *packet* switch rate.

The association set up time (i.e. the inverse of the switch rate) is the difference between the time at which an entity desires to communicate with an equivalent entity (with whom it has no current association), and the time at which it is first able to do so. The packet switch rate is composed of the time to decode, and act on, the in band information found in a received packet; this also represents the granularity of multiplexing within network components. The association set up mechanism can be either out of band, or in band. In the latter case, the control information for association set up must be scanned for in every packet, and hence has an effect on the packet switch rate in some network components. Circuit and packet switched networks exhibit the two extremes of these two rates.

2.2.1 Synchronous Transfer Mode

In digital circuit switched networks, the sub-channels, representing instances of concurrent communication, are multiplexed together by *synchronous* time division multiplexing. A *frame* (figure 2.1) is transmitted on the channel at regular time intervals, and symbols from each of the different sub-channels are inserted into fixed positions, called *time slots*, within each frame. The subchannel to which a certain symbol belongs is known from the time slot it occupies in the frame; this means that the subchannel is of a fixed and guaranteed bandwidth, and although no information need be conveyed in band to signify the subchannel to which the symbol belongs, there is a cost involved in allocating the time slots, and configuring the switches.



Slot	**Usage**
0	Alignment
16	D-channel / signalling
1-15, 17-31	B-channels / data

Figure 2.1: The STM framing for European primary rate ISDN

Such networks can be characterized by significant association set up time and fixed, but effectively zero, packet switch time. This mode of operation is referred to by the CCITT as synchronous transfer mode (STM).

The advantage of such systems is that the switching mechanisms are comparatively straight forward, and switches can be implemented in hardware to provide a very high aggregate bandwidth. The circuit nature is also favoured by the PTTs as it represents a simple and tangible resource which can be charged for.

Hardware for the user equipment for an STM network is also simple as it can be simply connected to the network by synchronizing it to the network *clock*. Equipment based on fixed rate CODECs, for example using pulse code modulation (PCM), is straight forward to build even for high bandwidth requirements such as video, as all data between the communicating entities (e.g. camera and display) is handled in synchronous hardware.

2.2.2 Packet Transfer Mode

In packet switched networks, the subchannels are multiplexed by *asynchronous* time division multiplexing. The term asynchronous applies to the channel access mode; each transmitting node unilaterally decides when it wishes to transmit some data, and then attempts to obtain the whole channel for a period of time, by the use of a *media access protocol*. An asynchronous media access protocol involves the use of some mechanism to resolve contention between multiple nodes simultaneously requesting access to the channel. A contention resolution mechanism always introduces variability in the delay, or *jitter*, associated with access to the channel. As the implicit timing information provided in STM networks is no longer available for discrimination of the symbols associated with particular subchannels, the symbols are grouped into packets, and some labelling, or *addressing*, information must be present in each packet. The presence of this control information on a per packet basis results in a reduction in the available bandwidth for data.

The addressing information present in each packet can be either global or relative to some shared context between nodes; the network is then said to be based on either *datagrams*, or *virtual circuits*, respectively. The benefit of relative addresses is that they are substantially smaller than global addresses, so use a smaller proportion of the available bandwidth, and can be used for direct hardware lookup for routing functions. The disadvantage of relative addressing is that it requires an association set up stage to pass the mapping between global and relative addresses to all involved switching nodes. Due to the presence of an association setup phase, as in circuit switched networks, these networks are referred to as *virtual circuit* networks. The advantage of global addressing networks, that is those based on *datagrams*, is that they do not require any association set up mechanism. However, this is at the at the cost of a more expensive switching mechanism per packet.

Packet switched networks based on datagrams represent the opposite extreme to circuit switched networks in digital networks. Such networks can be characterized by zero association set up time and, often, significant packet switch time. Virtual circuit networks fall between the two extremes. This asynchronous mode of operation, of both datagram and virtual circuit networks, is referred to by the CCITT as packet transfer mode (PTM).

The main advantage of PTM, is that two communicating entities have the possibility of utilizing all of the available bandwidth between the two nodes. The traffic generated by computers tends to be of a *bursty* nature, and so is well suited to PTM networks. The asynchronous nature is also well suited to the attachment of general purpose computer systems. Attempting to synchronize software closely with the network presents a number of problems which need to be addressed by the use of a real time operating system.

In current designs of datagram based networks, switches, whether media access layer (MAC) bridges or internetwork routers, need to be implemented in software, as the complexity of the routing algorithms mean that hardware implementation is often infeasible.

2.3 The ATM compromise

Asynchronous transfer mode (ATM) represents a compromise between the multiplexing mechanisms of STM and PTM, in an attempt to support traffic traditionally carried on both STM and PTM networks. ATM combines properties of STM, namely low delay, low jitter and ease of hardware implementation, with properties of PTM, which support applications with highly variable bandwidth requirements.

2.3.1 Asynchronous Transfer Mode

An ATM network deals in ATM cells; these cells are composed of a fixed size data part (typically 32 to 64 bytes) and an address part, or label (typically 2 to 4 bytes). Very high switching rates can be achieved in ATM networks as, although switching is performed on a per cell basis, the work involves simple manipulation of the label field, and can be easily implemented in hardware.

What would normally be synchronous traffic is conveyed on ATM networks by a stream of ATM cells. A number of samples are inserted into an ATM cell along with sufficient sequencing information to allow detection and recovery from lost cells. Packetization introduces some delay in the transmitter, and when considered together with the possibility of lost packets, requires that an ATM receiver becomes somewhat more complicated than the equivalent STM receiver. In particular, to absorb the jitter introduced by the network, an ATM receiver must provide some buffering on each incoming channel.

Packet based traffic can easily be conveyed over ATM networks by straightforward fragmentation of higher level protocol units into ATM cells. Such mechanisms have always been used in a number of networks, such as slotted rings (Cambridge ring [Wilkes 79]) or slotted bus systems (DQDB [Budrikis 86]). The implementation of the fragmentation and reassembly algorithm for ATM requires extra processing power at end points over that required for a PTM network. To support packet based traffic effectively, the fragmentation and reassembly algorithm must be efficient, and preferably implemented somewhere other than in the main host processor.

The design of the synchronization and packetization mechanisms for both synchronous and asynchronous traffic is dependent on the exact nature of the underlying ATM network. More particularly, the mechanisms depend on the failure semantics of the network (e.g. does the network drop, re-order, or duplicate cells), and the failure rate (e.g is selective cell retransmission needed). The information required to enable the synchronization and fragmentation functions must be conveyed in band, and hence represents an effective loss of bandwidth (to user data) in conveying traffic via ATM versus STM or PTM. Therefore the coding of the information required to achieve the synchronization and packetization functions must be kept to a minimum.

As previously discussed, synchronous access to a channel implies a fixed rate, and so to support variable rate traffic effectively, asynchronous access is required. Delay and jitter are introduced into the channel due to the use of the channel contention resolution process inherent in asynchronous access. As the network is required to support normally synchronous traffic, the contention mechanism must ensure that, at least for this type of traffic, the delay and jitter introduced is bounded. This is the main reason for the small size of the ATM cell. Besides representing the grain of multiplexing, it also represents the grain of contention resolution, and if kept small reduces the delay and jitter introduced. Selecting this small size restricts the possible media access algorithms which can be used, for example CSMA/CD is not a candidate considering the bandwidth and utilization required.

ATM represents a compromise between the needs of traffic traditionally carried on STM and PTM networks. Given this quite restricted view, it may still be the case that ATM performs well enough for both PTM and STM traffic types. The next section deals with the particular advantages, for both PTM and STM traffic types, which result from the choice of ATM networks.

2.3.2 The ATM advantage

One simple reason to move to an integrated network architecture supporting both synchronous and packet based traffic is the desire to install only a single set of cables or fibre optics in a building. Although ATM solves this problem, it is not the only solution. A number of attempts based on providing explicit isochronous channels to support voice and video have been investigated (e.g. FDDI-II [Ross 86], DQDB [Budrikis 86]). Unfortunately this approach does not represent an *integrated* solution, rather it provides time division multiplexing between an STM network and a PTM network, with the following effects:

- fixed bandwidth allocation,
- distinct channels,
- restrictions on real time equipment.

Although the sharing of the bandwidth between the two different traffic types (i.e. the ratio of the times spent in STM mode versus that in PTM mode) can be varied by including more STM slots, the mechanism is fairly restrictive, and cannot take into account the different loads being applied by different mixes of traffic on a short time scale (for example on every ring revolution or bus cycle). The separation of the channels for real time traffic and packet traffic, also leads to the complication of software system components and multi-media applications which need to deal with both traffic types simultaneously. Finally, real time equipment is restricted to deal in one, or perhaps multiple STM channels, but either way its bandwidth usage is strictly dictated by the network.

In this area, a particular advantage of ATM networks is their support for variable rate encodings of real time traffic. Voice and video traffic have traditionally been carried on STM networks, and so it was only natural that the CODECs produced for handling these traffic types provide a fixed rate sampling and coding, as is suited to STM networks. For example, toll quality voice pulse code modulation (PCM) CODECs designed for use over 64Kb/s channels, produce a 1 octet sample every 125μs. When used for a normal telephone conversation (between people), this represents inefficient use of a 64Kb/s bidirectional channel; such conversations take the form of alternating *talk spurts* interleaved with silence, so that the channel is only used half-duplex.

Differential pulse code modulation (DPCM) with silence suppression is the next logical coding technique to apply to voice; this naturally produces a variable rate coding. Rather than try to *smooth out* this variation, or allocate a fixed bandwidth channel able to cope with the instantaneous maximum bandwidth required by the coding, it is more natural to use a variable rate channel. This is representative of a more general observation about all types of traffic in a multi-service network from the field of information theory; if the information content of the data to be conveyed on a channel varies over time, any *efficient* coding will vary proportionally. A particular example is in video compression, where even simple differential schemes result in hugely varying bandwidth requirements.

The fine grained sharing of bandwidth and delay characteristics required to ensure that real time traffic is feasible on an ATM network also has advantages for packet based traffic in a number of circumstances, e.g.:

- performance during transient overload,
- rate based flow control.

Overload situations in PTM networks often result in significant packet loss due to buffer overruns within router and receivers. ATM local networks, by the nature of their media access, provide fairness, in that each node is guaranteed a certain minimum bandwidth. Overload situations then manifest themselves as larger packet queues in the *transmitting* hosts, and hence although packets may be delayed, they are not lost. The architecture presented here extends this mechanism to avoid the routing elements becoming the main source of packet loss, as they are in many other internetwork architectures.

Reliable transport mechanisms based on rate based flow control aim to vary the rate of introduction of traffic into the network to achieve end to end flow control. Such mechanisms work best when the transit time, or round trip time, between the two communicating entities can be estimated accurately. The much smaller variance in delay introduced by ATM networks, as compared to PTM networks, allows a much better rate estimate, and hence tighter control over the rate parameter. Similarly, if media access is used as a low level flow control mechanism, the rate can be varied depending on the achieved rate of traffic introduction into the network.

2.3.3 Disadvantages of ATM

As previously mentioned, ATM is a compromise between STM and PTM in an attempt to support traffic normally carried on these different types of network. While easing the implementation of the network, it presents problems which need to be solved in the internetwork routers and host interface equipment. As we shall see, to remove excessive load from the host system, the network interface needs to be considerably more complex than either an STM or PTM interface. In part this is due to the attempt to use PTM and STM derived protocols over ATM networks. However, some extra complexity will always be required, even with a well designed ATM protocol stack. For example, extra hardware is required in an ATM network interface to remove jitter from the stream of cells when used to carry voice traffic. Similarly, the increased cost of handling the increased number of small packets may require the implementation of hardware assistance for certain functions.

Finally, for some traffic types and uses, ATM will never be as efficient and cost effective as the alternatives. An important consideration in designing a multi-service network using ATM, is the the fraction of the total traffic that these uses represent.

2.4 ATM Implementations

A number of networks can be classed as ATM networks, and this section provides a brief overview of some of the network designs at which the MSN architecture is aimed.

2.4.1 The Cambridge Fast Ring

The Cambridge fast ring (CFR) [Temple 84] [Hopper 88] is a slotted ring network, designed as a local area network (LAN). Each slot is of a fixed size, and is composed of 4 control bits, 16 bits each of destination and source address, 256 bits of data and 12 bits of combined CRC and response information. The format is shown in figure 2.2.

The nature of the media access ensures a fine grained sharing of the bandwidth. A station wishing to transmit waits for an empty slot (i.e. one whose full / empty (FE) bit is clear), fills the data and addressing fields, and marks the slot full (FE set). When the slot returns, it is marked free, and passed on. Further transmission is possible in the next free slot. For a ring of N slots, the minimum time between accesses to the ring is $N + 1$ slot times[2], and, for $M(> 1)$ stations on a ring, the maximum time between accesses is $M + N$ slot times, when all M stations wish to utilize the ring[3].

A per slot response mechanism is incorporated, and is used in a number of ways:

- hardware retry,

[2]The current CFR station implementation is only able to access one in every $N + 2$ slots.

[3]The original CFR had provision for *channel mode*. This allowed some of the slots to be acquired by a station and reused every revolution, however, the maximum delay is still bounded.

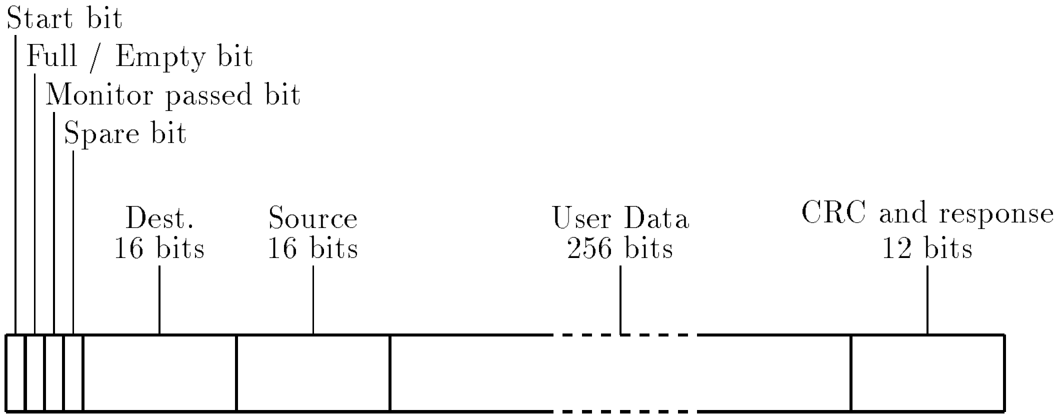


Figure 2.2: The CFR slot structure

- software retry,
- MAC layer flow control [Porter 89].

The CFR exhibits all the characteristics of an ATM network: fixed cell size, asynchronous and bounded media access time, with the media access decisions made in a distributed manner. The response mechanism enables significant simplifications to a data link protocol for such an ATM network, as it can be used to provide an arbitrarily reliable cell delivery mechanism, while ensuring no reordering of cells.

A node is currently implemented as a two chip set: an ECL line driver, and a CMOS station chip. The ECL chip performs clock recovery on the receive side, and encoding and decoding functions between the serial line code and an 8 bit parallel interface. The CMOS station chip interfaces between the host computer and the ECL chip. It contains the media access (MAC) logic, buffers for receive and transmit, and host interface logic. The current implementation is constrained by the clock rate of the CMOS chip; currently a clock rate of just under 10 MHz is possible, for a serial line rate of approximately 75 Mbit/s, and a slot time of 4 μs. Most of the experimental work for this dissertation, concerned with ATM aspects, has been performed using this network.

2.4.2 The Cambridge Backbone Network

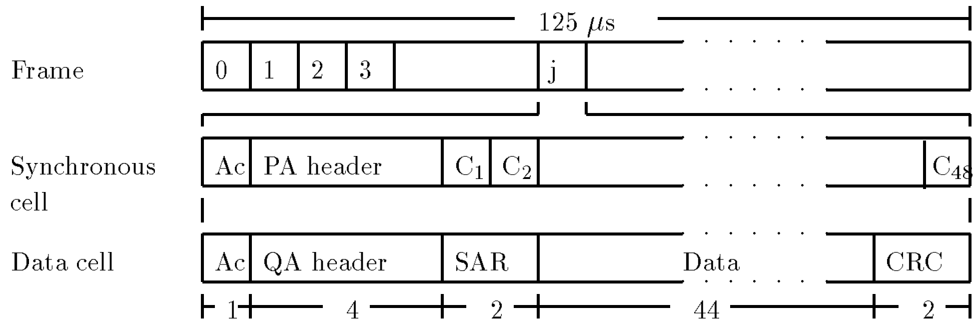
The Cambridge backbone network (CBN) [Greaves 88] is a metropolitan area backbone network, designed primarily to provide interconnection of CFR local area networks. It shares many of the features of the CFR, including the slot format, and represents another ATM network with subtly different properties. Although the slot structure is the same as the CFR, the CBN uses synchronous time division multiplexing (STM) to group four cells into a frame, which has the perceived effect of running four rings in parallel. The STM is utilized by having a station always receive in a particular time slot, and having the transmitter, which can utilize any one time slot per frame, select the correct one.

Having been designed for the metropolitan area (i.e. a network of tens of kilometers in diameter) and with a target line rate of 1 Gbit/s, the number of slots in the ring is considerably larger than in the CFR, as is the round trip time. For this reason, useful point to point bandwidth can not be obtained with the same media access protocol as the CFR. On the CBN, a node is allowed to transmit into multiple slots per revolution. Combined with the response mechanism, this again allows the provision of an arbitrarily reliable cell transfer but may not preserve cell order.

Full details of the design and implementation of the Cambridge backbone network are to be found in [Greaves 89b].

2.4.3 DQDB

The distributed queuing dual bus (DQDB) network [IEEE 8026 89] (based on the earlier QPSX [Budrikis 86]), provides a hybrid approach to multi-service networks. The underlying network is synchronous, and portions of the total bandwidth can be reserved for synchronous traffic while the remainder can be utilized to carry statistically multiplexed data orientated traffic.



Ac is the access control field which indicates whether the slot is free and its *type*. The PA and QA headers are for preallocated (synchronous) and queued (asynchronous) access slots repectively. The SAR field provides segmentation and reassembly support, and the CRC field protection of the contents.

Figure 2.3: DQDB frame and cell structure

The busses are unidirectional, and fixed length synchronous frames pass along each one every $125\mu s$. Each frame is in turn composed of some number of 53 octet cells (figure 2.3). The first 8 bits of each cell is the access control field and includes a type field which splits the cells into one of two categories:

- a synchronous cell,
- a data cell.

Synchronous traffic, e.g. voice, is carried in synchronous cells (so called PA cells) by splitting each synchronous cell into 48, 8 bit wide STM channels. These channels are centrally administered[4], and provide a range of synchronous circuits in steps of 2Kb/s.

Data cells (QA cells) come in a variety of subtypes, for example, control, error recovery and data cells. Access to all of these data cells is asynchronous, and media access is by a distributed queuing algorithm [Newman 86]. By continuously monitoring *all* data cells on both busses as they pass the network node, the state of a distributed access queue is maintained. The algorithm provides immediate access at low utilization while preventing wasted capacity at high load. Priority can be implemented simply by maintaining multiple queues, one for each priority level. The main problem with the media access algorithm arises from the asymmetric nature of the network, which leads to a slight bias in the access delay (and hence bandwidth allocation) in favour of *upstream* stations.

The data part of the DQDB network provides another example of an ATM network. The lack of a low level response mechanism, leads to problems of flow control and packet loss which are not present in the two ring designs discussed previously.

2.4.4 Fast Packet Switching

A wide range of network designs based on fast packet switching techniques have been suggested. The generic structure of a fast packet switch is shown in figure 2.4. A fast packet switch consists of a set of input lines and controllers, and output lines and controllers, all of which are interconnected by the switch fabric itself. Buffers can be placed in any of these three components, and the wide range of switch designs is in part due to these different selections. Many of these designs are based on short fixed length packets, with a slot structure applied to the input and output lines, and it is these networks that provide the relevant ATM properties.

[4] IEEE working group 802.6 has not yet addressed these management issues in detail.

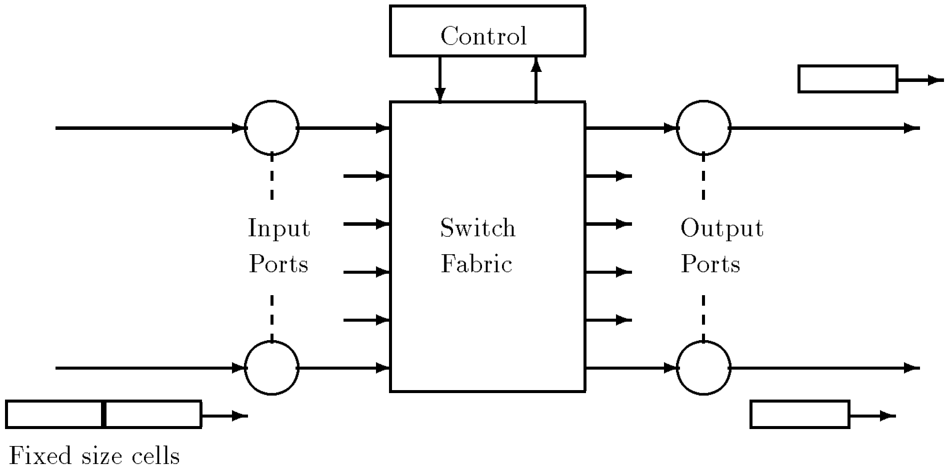


Figure 2.4: Generic fast packet switch for ATM

VLSI implementation of high capacity switches is possible due to the simplicity of the switching element itself, and the ability to use space division multiplexing (i.e provide multiple paths through a switch). Connecting multiple switches together into a network promises to provide a network of very high aggregate capacity. The Cambridge fast packet switch provides a switch capacity of up to 150 Gbit/s using conservative 50 MHz CMOS gate array technology [Newman 88].

Much of the work on fast packet switching has involved simulation and single switch design, only a few full size networks have been built. The Fairisle[5] network is a forthcoming project to build an ATM network based on the Cambridge fast packet switch.

2.5 Summary

Time division multiplexing is presented as the main mechanism of multiplexing within digital networks. Synchronous (STM) and asynchronous (PTM) schemes provide different characteristics in terms of delay, jitter, and response to *bursty* traffic. A multi-service network is required to handle both real time and bursty traffic sources, and asynchronous transfer mode (ATM) is proposed as a compromise between STM and PTM, which provides a base network with the necessary mix of characteristics.

A wide range of different techniques are available to implement ATM networks,

[5]Fairisle is a collaborative project between the Cambridge University Computer Laboratory and Hewlett-Packard.

and four different mechanisms are described. The implementation work for this dissertation, to be presented in later chapters, is based on the Cambridge Fast Ring.

Chapter 3

Previous Work

A wide range of different techniques for digital network interconnection have been designed and implemented, and this chapter surveys a number of these mechanisms with special attention to a number of areas: efficiency, reliability, heterogeneity, and scalability.

3.1 X-25

Many public packet switched networks are based on CCITT recommendation X-25, which defines a physical, datalink, and network level (in X-25 terms, packet level) interface between the user equipment (the data terminal equipment, or DTE), and the network node (the data circuit terminating equipment, or DCE). Examples of public networks based on X-25 are PSS in the UK, Telenet in the USA, and Datapac in Canada, while many private networks such as JANET (Joint Academic Network) in the UK also follow the recommendation.

The data transfer interface presented is connection-oriented. Virtual circuits are established by the exchange of in-band signaling information between the DTE and DCE, in which peers are addressed with up to 40 digit decimal numbers, according to CCITT recommendation X.121. Once a circuit is established, the only addressing information present in the data packets between the DTE and DCE is a 12 bit logical channel identifier. Each virtual circuit is individually flow controlled, in both directions, between the DTE and DCE using a windowing mechanism.

X-25 has been criticized for both over and under specification [Padlipsky 82]; on one hand it defines everything from bit level synchronization to the network layer, while it provides no conceptual model concerning the internals of the network. Recent work has aimed at separating the different layers present within the original X-25 specification with the aim of extending the service over different datalink layers (e.g. X-25 over logical link control (LLC) type 2 in IEEE 802 networks [ISO 88022]). This extended set of specifications has been adopted as the European *functional standard* to provide the OSI connection oriented network service (CONS)[1].

The OSI transport service [ISO 8073 84], and its precursor within JANET, the JNT Yellow Book Transport Service [YBTS 80], defines a transport service for use over CONS / X-25. By convention, X-25 provides a highly reliable service, and certain groups are pressing for an even higher reliability X-25 / CONS service so that the minimal OSI transport class (class 0) can be used [COSINE 88]. This high reliability is achieved by hop by hop flow control and retransmission mechanisms within the X-25 network. Such mechanisms not only complicate, and hence slow down, each switching node, but also mean that the end to end communication is only as reliable as the intervening switching nodes. Such a reliable service is inappropriate for real time and distributed computing traffic, and in fact prevents the effective implementation of some of the recently investigated transport protocols. The importance of the *end-to-end* argument [Saltzer 84] in these cases cannot be over stressed.

The X-25 virtual circuit interface is favoured by PTTs because, as in a fully circuit switched network, the *calls*, and their duration are a resource which can be charged for. Similarly, packets are easily associated with virtual circuits for charging purposes.

3.2 The DARPA Internet

The design philosophy of the DARPA Internet architecture has been summed up as [Clark 88]:

> "The top level goal for the DARPA Internet Architecture was to develop an effective technique for multiplexed utilization of existing interconnected networks."

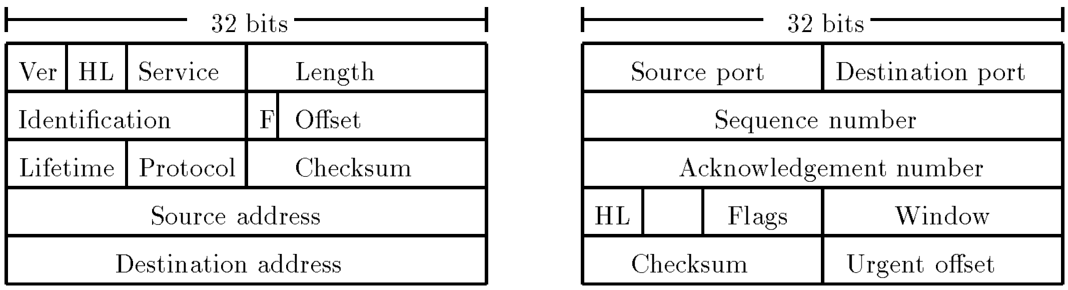
The *existing* networks to be interconnected were a collection of heterogeneous packet switched networks, and together with an interpretation of *effective technique*, led to the following major design goals:

[1] An OSI functional standard is a specification of a particular consistent subset of the OSI service primitives.

- maintenance of service in the face of failure of network components,
- support many varieties of networks,
- support for multiple transport protocols.

Unlike other networks, e.g. X-25, the desire to insulate against failure of internal network nodes led to the decision to place *all* the mechanisms concerned with reliability of communication at the end points of the communication. This allowed the network to reconfigure itself dynamically to cope with node and subnetwork failures. The hope was that, as long as the network did not partition, two hosts could always communicate no matter how the internal network configuration changed.

The reliability requirement, together with the fact that most of the networks to be connected together were datagram based, led to a datagram orientated network, with each packet containing global addressing information sufficient to convey it through the network. These datagrams formed the basis of the Internet protocol (IP), which in the ISO OSI model performs the functions associated with the network layer. The desire to support multiple transport protocols led to the provision of a protocol type in each IP datagram, so that at the receiving end, demultiplexing could be performed on a per transport protocol basis. The transmission control protocol (TCP) is one of these transport protocols, and provides the required end to end error recovery by use of windowed flow control, with positive acknowledgement and retransmission. The format of the protocol headers for IP and TCP is shown in figure 3.1.



a) IP header b) TCP header

Figure 3.1: TCP / IP headers

To support many varieties of underlying networks and media access layers, two areas need consideration: packet size and local network addressing. Many of the networks to be attached had restrictions on the maximum size of packet which could be conveyed, the range being quite large; from 64 to 64K bytes. Rather than implement ad hoc mechanisms for each different underlying network, a fragmentation and reassembly mechanism was built into the IP layer. The requirement that there be no hop by hop recovery mechanisms, led to each fragment being a self contained datagram. The addressing scheme chosen was also designed to allow for internetworking with an IP address being split into two parts, a network part and local host part, where the local host part may be interpreted in whatever way is most suitable for the underlying network.

The success of the Internet protocol family is seen by its widespread use, and that it has been basis for the ISO OSI connectionless network protocol (CLNP) [ISO 8473 86]. Many aspects of the protocols have been researched, and a number of them are worth closer scrutiny here:

- global addressing in each packet,
- packets routed individually,
- packet fragmentation possible at any node,
- per packet demultiplexing.

Each packet, or fragment, carries complete addressing information, and so the overhead for conveying small amounts of data can be considerable. A particular instance worth considering is the traffic between a terminal and host over a Telnet[2] connection. With minimal IP and TCP headers, each of 20 octets, some local network overhead, and a data part of 4 octets, this represents an overhead of over 90%. Even when working in *line mode* such small packets are common, the situation being even worse when interactive applications requiring single character input, such as full screen editors, are being used.

Besides the overhead lost to control information per packet, the overhead in routing each packet is also considerable. Although good implementation techniques, such as caching or hashing of the routing tables, can result in reasonably efficient gateways, the packet switch rate is still low when compared with the rates required to support even a modest number of voice channels.

Another aspect of this individual routing is that consecutive packets between two hosts may traverse radically different paths, and hence experience significantly different delays in the network. The large jitter in the network has a direct effect on the set of transport mechanisms it is sensible to try to implement over such a network; certainly such a system is inappropriate for voice traffic, and can lead to problems with rate based flow control.

[2]Telnet is the DARPA IP terminal access protocol.

Fragments are packets too, and hence are individually routed. This means that there is no guarantee about the order in which a host receives the fragments, and hence there is no sensible mechanism for deciding when to terminate reassembly. Experiments have shown [Kent 87] that performance can be radically improved by preventing IP fragmentation of packets altogether, and making use of the recovery mechanisms present in the transport level, in this case TCP. The only way to prevent this fragmentation within the network is to ensure that the packet is small enough to be carried intact across all of the subnetworks it will trasverse. This involves either setting a maximum size for the whole network, or selection of a size based on a route between the communicating pair of hosts. The first solution results in poor utilization of some parts of the network, while the latter would only be effective if the route which packets traverse between two hosts remains fixed, i.e. there is some form of IP *connection*.

Finally, although IP provides multiplexing for the support of multiple transport protocols, each of these protocols in turn implements further multiplexing[3]. This can have a significant effect on the efficiency of the implementation of the protocol, and on the jitter introduced into the traffic flow within hosts.

3.3 Universe

The Universe project (UNIVersities Extended Ring Satellite Experiment) was one of the first attempts to connect together a number of LANs in a transparent manner over a wide area. In the Universe project, the networks to be connected were Cambridge rings, and the wide area interconnect was provided by a 1 Mb/s satellite broadcast channel. The rings to be connected all supported the Cambridge model distributed computing environment [Needham 82], and the desire was to extend the distributed computing mechanisms present in this system to the wide area.

The Universe network used the concept of *lightweight virtual circuits*. These virtual circuits are referred to as *lightweight*, as although they provide many of the characteristics of virtual circuits (for example sequenced delivery of packets), they exhibit a number of vital differences; bridges perform no flow control or error correction, and most importantly, can make completely independent management decisions. These lightweight circuits are established as a side effect of name lookup. A chain of interactions starting with the name server on the local ring and involving any intermediate bridges and eventually the remote name server on the remote ring results in the establishment of a virtual circuit between the two relevant peers.

[3]The Internet control message protocol (ICMP) is one of the exceptions but this due to there being only one ICMP *client* per machine.

A number of features of these bridges can be directly related to the nature of the multiplexing mechanism, available to applications, in use on the Cambridge ring. Within the Cambridge ring, almost all traffic was conveyed by means of the *basic block* protocol. This used a sequence of cells (each conveying 16 bits of user data) to provide a *basic block* of up to 2K octets. In ISO OSI terms the basic block protocol provided the data link layer between hosts. As no protocol information could be realistically conveyed on a per cell basis, the mechanism chosen to implement this protocol was to have a node, on reception of the first cell of a basic block, *select* the relevant transmitting station, and lock out others. The use of the basic block protocol means that the fine grained sharing of the bandwidth provided by the ring media access layer is now nullified in a number of important instances. Although independent pairs of stations on the ring still experience sharing of the bandwidth, two stations, or two channels from one station, trying to transmit to a single receiving station experience contention, and have to resort to a time based back off algorithm for contention resolution.

The basic block protocol was defined because the small size of the Cambridge ring cell meant that these cells could not be treated as independent protocol entities, and for the same reason, the Universe bridges must forward at the level of basic blocks, and not at the level of cells. Bridges, by their nature, will need to receive from multiple sources, and the contention implicit in the Basic Block protocol for the bridge station means that there is significant interference between independent channels.

Although it is possible to implement a version of the basic block protocol to allow multiplexed reception from different stations by use of the source station as a demultiplexer, as in the Universal receiver protocol [Fraser 89], the problem of multiple channels between two stations, particularly two bridges, is not solved.

The Universe network was based on the interconnection of homogeneous networks (Cambridge rings), and provides none of the mechanisms that are associated with heterogeneous internetworking.

3.4 Unison

Building on the work of Universe, the Unison project aimed at a site interconnection network based on 2 Mb/s land lines. The aim was to provide a flexible interconnection architecture which could be used by a variety of client networks, rather than be restricted, as in Universe, to a single type of network. Besides this requirement, the design of the architecture was affected by two areas which were seen as important at the time: the Integrated Services Digital Network (ISDN) and ATM.

The use of ISDN, as opposed to leased lines, was considered desirable as it allowed the Unison network architecture to be implemented over the forthcoming widely available public ISDN networks, allowing simple connection to the Unison network. When the project was started, ISDN networks were not available, and a private prototype ISDN network, the Alvey High Speed Network, was used. The benefit of this approach is now clear, as the recent conversion of the network infrastructure developed in the Unison project to use the British Telecom supplied public primary rate ISDN service, Multi-line IDA, has been straight forward.

The requirement for a flexible architecture for the Unison network was based on the desire to support the interconnection of wide range of local networks, from PTM networks, such as Ethernet and IP, to STM networks, such as a PABX. To support the widely varying data transfer requirements of these networks, an ATM approach was taken. In the Unison architecture, a CFR at each site, the *exchange network*, acts as an ATM switch between the common carrier networks and the client distribution networks. The points of attachment of the exchange network to the ISDN, referred to as *ramps*, and the ISDN itself form a dynamically configurable set of bridges between the exchange networks at different sites. The client distribution networks are attached to the exchange ring at nodes referred to as *portals*.

Another aspect of this flexible architecture approach is the provision of out of band management services. These services are used by portals to establish communication with peer portals at remote sites. These functions include services such as ISDN I-series signaling and dynamic bandwidth management by use of *B-channel* aggregation [Harita 89] [Burren 89]. The use of out of band techniques for management, allows the in-band ATM channel to be utilized effectively in a completely portal specific manner.

A number of different portals have been implemented, both for different network architectures, and provide interconnection at different levels within the ISO OSI reference model. The Cambridge ring portal provides the functions of the Universe bridge between sites, i.e. transfer of basic blocks. This is an example of a portal acting at the datalink level, details of which are to be found in [Tennenhouse 88]. The IP portal treats the Unison network as a backbone IP network, and acts as a network level router. Currently this portal connects IP networks running over CFR and Ethernet.

3.5 Summary

Each of the presented interconnection techniques has its own advantages and disadvantages, and only a few brief points are highlighted.

The IP suite of protocols has been implemented on an unknown number of network types. Widespread encouragement to use IP based products for networking within the USA by DARPA is partially responsible, but it is the ability of the IP protocol to deal with a very wide range of network types that actually made such widespread implementation feasible. The Universe network provided none of the internetworking abilities of IP. However, the idea of using lightweight virtual circuits allows faster routing within the connecting nodes between networks. Unlike the mechanisms implemented within X-25 networks, both Universe and IP stress the importance of end to end reliability mechanisms, so that communication between these end points can continue even in the event of reconfiguration of the internal components of the networks. Finally, the Unison network, has demonstrated how ATM can be used to provide a wide area, site interconnection network which supports a wide range of different traffic types, as generated by the different local area networks attached.

Chapter 4

The Multi-Service Network architecture

A multi-service network is required to support traffic exhibiting a wide range of characteristics: current interest is in supporting data, voice, video, high bandwidth graphics, and distributed computing applications within the same network.

To support these applications, in particular video and graphical visualization, the first requirement is sufficient bandwidth. As has previously been noted, this has simply been achieved by continuing improvements in network integration and media technology.

It is then the task of the multi-service network architecture to provide this bandwidth to these applications in a usable form. This chapter deals with the requirements of such a network architecture: its implications for operating systems, hosts, network interfaces and internal networks components. The various requirements to be covered are:

- bandwidth sharing,
- bandwidth guarantees,
- internetworking,
- network interfaces,
- software elements.

In designing the architecture, one of the main guiding principles has been *out of band* management, or the separation of control and data. In STM networks, the complete separation of these functions implies virtual circuits and call set up overheads, but allows simple implementation of high bandwidth switches and interfaces in hardware. When choosing to utilize ATM, consideration of this principle leads to some of the same benefits.

4.1 Bandwidth sharing

The first requirement of a multi-service architecture is the need to provide a mechanism which enables sharing of the bandwidth to satisfy each application's requirements. The discussion of multiplexing mechanisms (section 2.2) was aimed at demonstrating how ATM networks can satisfy these requirements. The media access mechanism of ATM networks ensures a very fine grain of bandwidth sharing, while at the same time allowing statistical multiplexing to be used to provide dynamic bandwidth allocation. ATM is selected as the platform on which to support a range of traffic types with different quality of service (QOS) requirements.

The fine grained sharing provided by ATM networks can be completely wasted if it is not supported correctly within network nodes by the upper layer protocols, and by the internal network switching nodes and host interfaces. A design which caused loss of synchronization of a voice stream between two nodes because there was a simultaneous file transfer in progress would be unacceptable. The need to avoid this cross talk between simultaneous associations, perhaps with different quality of service requirements, implies that the protocols designed to run over an ATM network must provide multiplexing of these different traffic types at the same fine grain as the ATM MAC layer provides, at all points in the network.

Finally, the *implementation* of the multi-service protocol stack must continue to support this fine grained sharing. Within a network in which the majority of traffic is bursty in nature, such as seen in a lightly loaded network with distributed computing traffic, the great temptation is to optimize for packet based traffic from a single source at a time. These optimizations usually slow down the case of simultaneous receptions, and this is a trap into which elements of the Unison network have fallen. The reception of a single block at a time may be the case in a distributed computing system, but not for a multi-service network or for an ATM internetwork router; here the probability of these *collisions*, due to simultaneous reception of blocks, is much higher, and must be taken into consideration.

4.2 Bandwidth guarantees

To implement guaranteed bandwidth requires three basic functions:

- a reservation mechanism,
- an allocation mechanism,
- an enforcement mechanism.

Each different type of real time traffic generates requirements which can be represented as a set of statistics concerning the time varying behaviour of the demand and supply of bandwidth. Some of the simpler statistics are the peak, average and variance of the bandwidth demand, and the delay and jitter in the supply. The exact nature of the requirements may be extremely complicated, and, for real time traffic, depend closely on the coding algorithm used. For this reason, when designing a general purpose bandwidth guarantee mechanism, only the comparatively simple statistics can be considered, and some wastage of bandwidth is inevitable.

During association establishment between two applications which need bandwidth guarantees, these statistics are provided as quality of service parameters to all switching points in the network. If available, the relevant bandwidth is then reserved at all these points. Next a mechanism is required to allocate the available bandwidth through a switch to the packets belonging to the different associations, using the policy dictated by the reservation mechanism. In the case of an ATM network, this should be performed on a per cell basis, so that when real time traffic requires bandwidth, other less time critical traffic is held up *immediately*. The simplest mechanism is to provide a priority system which is supported throughout the network. Some ATM networks already provide such a system (e.g. the Cambridge backbone network), while others can implement a partial solution within the network controller by the provision of multiple queues; either way, the multi-service architecture is required to support priority.

Finally, an enforcement mechanism is required to prevent rogue applications reserving a high priority channel for some bandwidth and then transmitting at a much higher rate. Enforcement is best implemented as close to the transmitter as possible, so that the rogue traffic is stopped before it consumes too many network resources. The enforcement can be implemented in the network controller and host kernel, if they can be trusted, or in the router directly connected to the network with untrustworthy hosts. The latter would be the case when considering provision of a service by a common carrier.

Priority is one of the aspects of a multi-service network which leads to a high probability of *collision* between different associations in network interfaces during reception. For example, if during the transfer of a block of data, a scan line of video needs to be transferred, it will interrupt the lower priority transfer, and be transferred preferentially.

4.3 Internetworking

Although homogeneous ATM networks could be interconnected successfully over some limited area by MAC layer bridges, the need to connect them over the wide area, to connect between different administrative domains, and to support heterogeneous networks, leads to the requirement for an internetworking design[1]. An internet composed of ATM networks must try to maintain the low jitter and delay characteristics of ATM networks throughout the internet; this leads to a number of restrictions on the design:

- low jitter requires that all packets should traverse the same route,
- low delay requires ATM cells to be individually forwarded in routers,
- individual routing requires the cost of the routing decision per cell to be minimal,
- individual routing requires the cost of the *context switch* between different associations to be low.

The requirement that all cells traverse the same route leads naturally to the use of lightweight virtual circuits[2] at the datalink and network levels. By starting with this as the basis of the internetwork architecture, it is possible to remove much of the datalink and network protocol overhead usually transmitted in band in PTM channels (e.g addressing, lifetime, etc.).

Besides the need to interconnect ATM networks, it is also desirable to interconnect traditional packet based networks (e.g. Ethernet, Token ring) with multi-service ATM networks. Although these packet networks will be restricted, in that they are unable to carry any significant amount of real time traffic, they could utilize a high speed ATM network in a number of ways. For example:

[1] Section 7.1 covers these issues in some detail.

[2] I use the term *lightweight virtual circuit* in the sense defined in [Leslie 83] to indicate a virtual circuit where, unlike X-25, there is no hop by hop error recovery, and any node involved in the circuit is free to unilaterally terminate it at any time.

- as a backbone, for local and wide area,
- for access to equipment only available on an ATM network (and vice-versa).

4.4 Network interfaces

The network point of attachment is often a serious bottleneck in networked systems, and must be designed carefully because of the nature of an ATM network and the bandwidth requirements of real time devices. A number of inter-related points are considered in this section:

- preservation of IO and memory bandwidth,
- multiplexing,
- requirement for out of band management,
- hardware implementation of some protocol elements.

The internetworking requirement for individual cell *routing* extends itself into the network controller. The high bandwidth requirements of some of the real time traffic streams mean it would be unwise to attempt to transfer the incoming traffic over a host's IO or memory bus. A more reasonable approach is to attach real time devices directly to the network controller by private links. The network controller would be required to demultiplex the different associations, and direct individual incoming ATM cells for real time devices via these links while passing those cells associated with distributed computing traffic to some form of block reassembly engine. The treatment of these two different traffic types in a consistent way leads to a significant advantage when considering implementation of this demultiplexing function in hardware.

Real time devices may require a higher level of functionality than a stream of ATM cells, for example, a blocking (segmentation and reassembly) mechanism, and security features. In this case the use of a protocol which could be implemented within the network controller in hardware (or firmware) would be required to enable such a protocol engine to keep pace with the network.

An effect of this model of working, together with the requirement to keep the in band data path as simple as possible, is that data for many of the real time streams never passes through the main host, and so, information required for control purposes must by necessity be conveyed *out of band.* This is also desirable from the point of view of providing different programming models for the data and control parts of a particular traffic type; the data part of a video stream may be best implemented as a stream directly between source and sink, whereas a remote procedure call mechanism may be more appropriate for control purposes.

4.5 Software elements

Mention has been made of implementing certain protocol elements in the network controller to avoid excessive use of IO and memory bandwidth. Whether these are implemented in hardware or software within the controller, or software in the host, there is one aspect of the protocol stack which needs careful consideration: multiplexing.

Each point at which multiplexing is performed represents a sequencing of the associated protocol units, and hence a point of synchronization. Statistical delays are introduced as protocol units pass from the application through the synchronization points in each of the protocol layers implementing multiplexing, and similarly when these units are being demultiplexed on arrival. At high utilization there will be considerable contention, and the statistical delay at each point can be large and highly variable. Even when there is little or no contention at the synchronization point, for example due to low utilization, the statistical delay will be smaller, but will still be present.

Considering the minimal required multiplexing functions. Firstly, there is a requirement for *inter-system* multiplexing if it is desired to support sharing of the underlying transmission medium; this is contained in the MAC component of the datalink layer. Secondly, some form of *intra-system* multiplexing is required if a host is to be allowed multiple parallel associations with its peers.

In using an ATM network, inter-system multiplexing is performed at the grain of ATM cells at the MAC level. In the design and implementation of the associated synchronization mechanism (i.e. the media access), great care is taken to provide guarantees on the delay and jitter introduced. Synchronization for the intra-system multiplexing points is usually implemented in software, and a number of techniques can be considered when attempting to reduce the cost of this synchronization:

- the use of formal methods may provide some illumination, and perhaps even some liveness guarantees,
- the use of upcalls [Clark 85], as opposed to multiple layers of processes, increases performance by avoiding context switches at each synchronization point,

- the special case coding of the *most popular* path through the network code can allow significant optimizations (e.g. DEC System Research Centre RPC [Schroeder 89] and Amoeba RPC [Renesse 88]).

Within most protocol stacks, each level of protocol implements its own multiplexing over the service provided by the adjacent lower layer[3], and these techniques attempt to overcome the basic flaw in the architecture design by good implementation. A much simpler way to remove this repeated synchronization cost is to perform intra-system multiplexing *once only*.

The provision of a single multiplexing mechanism at the datalink level can be utilized by all higher levels, by enforcing a one to one mapping between the associations of the adjacent protocol layers. Although this is contrary to current ISO OSI thinking and the methods used in other protocol stacks, it is a requirement if the relevant properties of an ATM network are to be provided to applications. Each application level association now has bound to it a separate instance of a protocol stack, and these stacks intersect only at the one intra-system point of multiplexing, the datalink layer. This has advantages for the implementation of such systems, as within the popular multiprocessing thread based environment, a thread traverses the protocol stack without having to interact with other threads, until it arrives at the single point of synchronization within the datalink layer.

All of the techniques for minimizing synchronization cost described above can still be applied, and in fact others are now available. For example, in a thread based environment, the quality of service information can be used to affect the scheduling mechanism by using it to calculate a thread's priority. Attempting to use quality of service within a multiple layered multiplexing environment leads to a much more complicated implementation, which will almost certainly be slower.

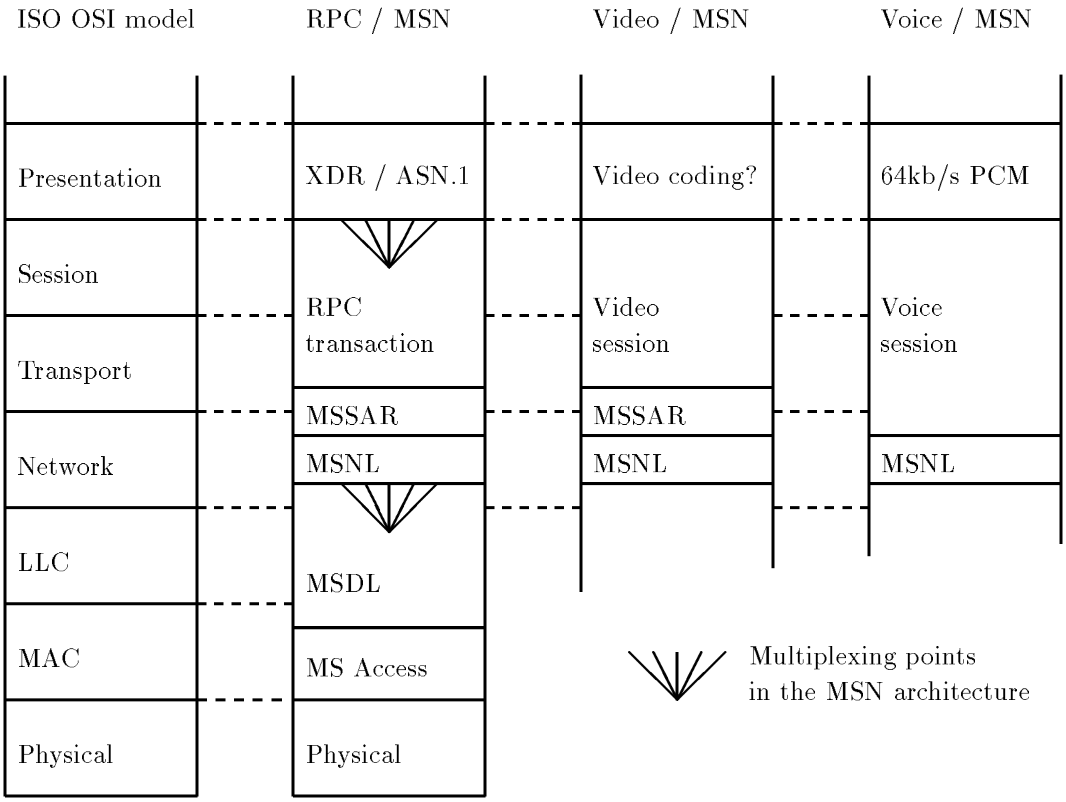
The decision to minimize the multiplexing in the architecture by providing only one intra-system multiplexing mechanism, leads naturally to out of band management for the associations between the higher level protocol entities, while having the benefit of being able to dispose of much in band protocol information. At some point in the protocol stack, it may become infeasible to continue to use the datalink multiplexing function, although this mechanism can be extended through most of the protocol layers, as shall be shown.

[3] ISO OSI supports multiplexing at 6 out of the 7 layers in the model.

4.6 Reference model

The layered reference model for the MSN architecture is presented here to provide an overview of the architecture. Details of the design and implementation of the distinct layers are presented in the following chapters.

Certain concepts of the reference model for the MSN architecture are similar to those of the ISO OSI reference model, but they have a very different implied interpretation. The use of different layers to provide services with greater functionality by building on lower level services is still present. However, above the datalink layer, the stacks should be thought of as being in parallel. This relationship of the multi-service protocol stack to the OSI reference model is shown in figure 4.1.



XDR [SUN 85] and ASN.1 [ISO 8824] are two possible presentation codings for RPC systems.

Figure 4.1: Relationship of MSN elements to OSI

The example of an application using remote procedure call shows multiplexing performed within the RPC transaction layer, at the OSI session level. This multiplexing represents the selection of a procedure within a remote interface, and the requirement for this is dealt with in some detail in section 6.2.2.

Multiplexing is often used at the presentation layer as a synchronization mechanism to coordinate related streams. For example, it is important to maintain the relationship between the voice, video and graphical components of a multi-media presentation system. This can be achieved by multiplexing these different streams onto a common session connection. Unfortunately this requires all the traffic to use a particular underlying network, and does not allow the different streams to be conveyed using networks with the most suitable quality of service for each type. Within the MSN architecture, the removal of the ability to multiplex at these levels, or at least severely discouraging its use, means that some other method of synchronization must be implemented. The problems and benefits of this approach, and the description of the presentation and session level extensions to the MSN architecture to realise it, are presented in [Nicolaou 90].

4.7 Summary

The requirements of a multi-service network architecture built over ATM networks have been presented. It requires attention to the design and implementation of routers, network interfaces and end host software support. It is envisaged that, for performance reasons, certain elements of the protocol stack may need to be at least partially implemented in hardware.

One particular design principle, which is related to all of the previously mentioned requirements, is seen as important: the separation of control and data. This principle leads to the desire to minimize layered multiplexing, which is seen as a performance bottleneck, and presents a serious impediment to protocol implementation in hardware.

Chapter 5

MSN low levels

This chapter presents the detailed design and implementation of the lower layers of a multi-service network architecture, the MSN architecture. The description of the work on the lower and higher protocol layers is split into two chapters representing the two different domains within which the implementation work was performed.

In terms of the ISO OSI reference model, the levels dealt with in this chapter perform the functions of media access, datalink, internetworking, segmentation and reassembly. Two different versions of these lower layers have been produced, the first derived from experience of, and compatible with, the Unison network, the second extending the MSN architecture into the MAC layer of the CFR.

The reference model for the lower layers of the MSN architecture has already been presented in figure 4.1. The MSN architecture defines four layers which contribute to the internetworking service:

- datalink - MSDL and MS-Access,
- network - MSNL,
- segmentation and reassembly - MSSAR.

The design and implementation of each of these is dealt with in turn. The results of experimentation with the prototype implementation are also presented.

5.1 Environment

In implementing and experimenting with these low level networking functions it is vital to have a lightweight and easily modifiable run time environment, so that the performance effects of the architecture can be readily observed. This was achieved by implementing a simple message passing kernel, MICA, on the Acorn Archimedes personal computer. Starting from scratch allows the consideration and implementation of the particular requirements of these low levels of protocols, without the need to implement, or be bound by, the requirements of a general purpose operating system. Such an approach is still useful from the point of view of attaching a general purpose host to a network, as it can be used in the implementation of a front end network controller.

Although the software environment could be tailored, there were still certain compatibility constraints which had to be considered. In particular, building, or rebuilding hardware has not been attempted as part of the implementation work for this dissertation, and instances where hardware implementation would have been effective are discussed as and when they occur.

The architecture has been designed mainly with heterogeneous ATM networks and the support of multi-service applications in mind. A secondary aim was to integrate traditional packet switched networks into the architecture, as besides the fact that certain aspects of the architecture provide valuable lessons for purely packet based networks, it was desired to provide access to the control mechanisms of multi-media applications from such networks.

5.2 Multi-Service Data Link

The multi-service datalink (MSDL) layer is designed to provide the basic datalink service, and is based on *lightweight* virtual circuits, referred to as associations. MSDL performs some functions similar to logical link control (LLC), but there is no error recovery between MSDL instances. The goals of MSDL are:

- to provide datalink *associations* over heterogeneous networks with different characteristics,
- to make the user interface to MSDL network independent (as far as possible),
- to support fine grained multiplexing,
- to provide support for fast internetworking between heterogeneous networks.

5.2.1 MS-Access

MS-Access is an attempt to provide a consistent interface, for the support of MSDL, over the different MAC layers presented by different underlying networks. This service provides sequenced and error-free transfer of packets between peer MSDL entities, its interface being characterized in terms of *request* and *indication* primitives modeling outgoing and incoming network packets. In designing this service, consideration was given to a number of possible underlying networks, including networks well established at Cambridge (e.g. Ethernet, CFR, ISDN), and newer networks (e.g. Fairisle, BISDN).

There are three aspects of the underlying MAC layers that require consideration when defining the form of the service data units (SDUs) supported by MS-Access:

- addresses,
- quality of service,
- packet size supported by MAC.

As it is desired to support the MSN architecture over existing networks, some method of dealing with the underlying MAC layer addressing of such networks must be found. The MSDL level actually deals in virtual path identifiers (VPI). A VPI defines half of a virtual circuit; a VPI is placed in a packet or cell by the sender and is understood by the receiver. A virtual circuit is then defined by a pair of these path identifiers. A mechanism for the completely dynamic allocation of these identifiers for all types of network would form a consistent and aesthetically pleasing architecture. However, the requirements of compatibility mean that some networks are used in a more *traditional* manner. In these cases, a VPI is formed by pairing the unique MAC layer address with a port. This is covered more fully in the discussion on MSDL in section 5.2.2.

Quality of service issues, perhaps implemented as priority within the MAC layer, are not directly dealt with in the MS-Access interface, as it is desired to deal with these issues *out of band*, hence there is no explicit reference to quality of service parameters in this interface. The solution adopted is to attach a quality of service implicitly to each MS-Access instance; this has the consequence that any underlying network which can support multiple qualities of service, such as DQDB [Budrikis 86], must present itself as two logical MS-Access instances. Quality of service considerations reappear later, in that they are directly responsible for lack of MS-Access independence at the MSDL management interface (section 5.2.3), and it as at this level that they are dealt with.

To avoid unacceptably poor performance on some networks, particularly traditional packet switched networks, it is obviously necessary that use be made of the packet sizes that the different networks support most effectively. This requires that MS-Access needs to support variable sized SDUs. Some of the pitfalls in the area of internetworking over variable sized transmissions units have been avoided in the approach taken by MSDL.

5.2.2 MSDL and MS-Access

Multiplexing

One of the requirements, when concerned with multi-service traffic, is that the multiplexing of data be performed at a fine grain. This requires that the multiplexing function be performed at the lowest possible level, and certainly below any segmentation mechanism. This function is performed by MSDL, and the approach taken, in the case of the ATM networks, leads to multiplexing at the level of single ATM cells.

The virtual path identifier (VPI) supports the multiplexing function, and uniquely identifies the association in the context of the receiving MSDL entity. This allows demultiplexing at the level of a single MSDL PDU, and hence facilitates interleaving of receptions on several associations simultaneously. The VPI format is dependent on the underlying network, in particular its MAC layer addressing.

For a network such as Ethernet, were dynamic assignment of MAC layer addresses is not feasible[1], a VPI identifier is the pair:

$$(MAC.Address, Port)$$

where the port numbers are allocated locally. In the first implementation of the MSN architecture for the CFR, this approach was taken for compatibility with deployed network components, in particular with the Unison network.

The latest design, of which only a preliminary implementation has been performed, utilises the ability of a CFR station to receive on multiple MAC level destination addresses simultaneously[2], and a dynamically assigned MAC layer address is used as a VPI. Figure 5.1 shows the two different VPI formats for the CFR.

[1] The possibility of using Ethernet multicast addresses was considered, but the work involved in rewriting UNIX kernel components to properly support this, was considered undesirable.

[2] The ability of the CFR station chip to receive on multiple addresses simultaneously, the so called *bridge* mode, was originally implemented for the purposes of MAC layer bridging.

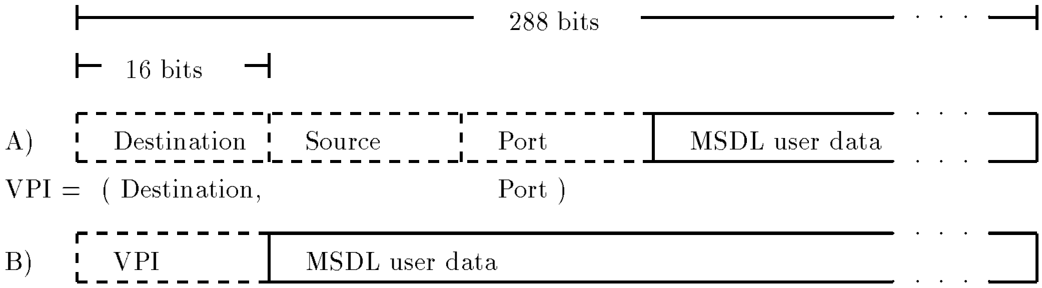


Figure 5.1: The different uses of the CFR slot

The VPI supports the function of both inter-system and intra-system multiplexing, and in the case of the MAC address / port pair, the split in these functions is obvious. The aim in using the dynamic MAC addressing scheme, is to utilize the underlying hardware efficiently to perform part of the intra-system demultiplexing in parallel with inter-system demultiplexing.

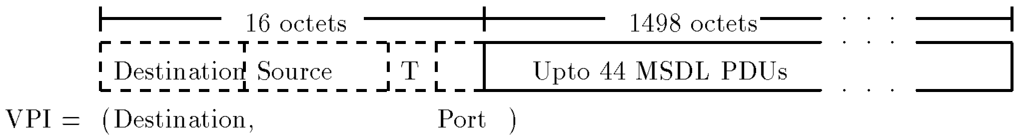
The choice of which VPI mechanism to use, MAC address and port, or dynamic MAC address is performed on a per network basis when defining how to layer MSDL over a particular MAC layer. The choice of the allocation strategy for the VPIs is another important consideration, as speed of lookup in the receiver must be of paramount importance. For example, in a software implementation, allocating in sequence to a particular node allows array lookup as opposed to searching.

PDU size

Although arbitrarily variable sized MSDL PDU data fields would make more efficient use of network bandwidth, such PDUs can require significant processing in hosts and internetwork routers. In particular, a router may have to perform fragmentation if it connects two MAC layers with different maximum transmission sizes.

Variable sized PDUs also require that the length be transmitted in the protocol header. In the case of MSDL this would require an additional field in the header, and hence an increase in the header length. As it is desired not to compromise the use of MSDL over ATM networks, one of the main aims of the design must be to ensure a minimum of protocol overhead in each packet, and so adding a length field is unacceptable.

As previously discussed, one of the main aims of the MSN protocol family is to have internetwork routers cause minimum delay to packets, hence they need to be simple and fast. This, combined with the undesirability of a length field, leads naturally to the solution adopted by MSDL; the data field of the MSDL protocol data unit (PDU) is of a fixed size, but multiple such PDUs (from the same association) may be presented in a single MS-Access request. Similarly an MS-Access indication may represent the arrival of several MSDL PDUs. The maximum number of PDUs which may be presented at once to a particular MS-Access instance is defined in its interface, and is obviously dependent on the maximum packet size of the underlying MAC layer. MS-Access layers are also free to perform optimizations, such as not transmitting multiple copies of the VPI in the same packet (see figure 5.2). This is another design decision taken on a per MAC layer basis.



T = Ethernet MSDL type field

Figure 5.2: The use of MSDL over Ethernet

As a candidate network must be capable of transferring at least one MSDL PDU, the size must be chosen with care. Consideration must be given to the MAC layers over which the MSN architecture is to be used, and the the size of smallest packet these can support.

In the case of the networks under consideration, the CFR slot size presented the most stringent constraint on packet size. In the first iteration of the MS architecture, 2 octets were allocated to the port field, leaving 30 bytes for MSDL data. In the original implementation on the CFR, the source address was dutifully checked on each cell reception but was found to be redundant, as the source address can be easily faked by any other station. The realization that the source field is irrelevant and the use of the MAC destination field as the VPI, meant that in the second iteration, the MSDL PDU data field was 34 octets. These two sizes represent the fixed sizes chosen for the MSDL data field in the first and second iterations of the MSN architecture; the different utilizations of the CFR slot are shown in figure 5.1.

In the second iteration of the MSN architecture, only 2 octets are used as a VPI on the CFR. This restricts the number of simultaneous associations that a single CFR is capable of supporting, to 2^{15} (there are two VPIs per association). This is not seen as a significant restriction as a CFR is designed to support only a limited number of stations *per ring*; perhaps a maximum of 30 stations, allowing up to 1000 associations per station. In fact, the technique of VPI allocation implemented was to allocate all VPIs with the same most significant 6 bits to one station, and allow each station to allocate the lower 10 bits, placing a restriction of 1024 associations per station.

5.2.3 MSDL management and MS-Access dependence

Many uses of MSDL, particularly data related ones, do not need to know anything about the underlying instance of MS-Access, and indeed the MSDL *user* interface (i.e. send and receive primitives) is independent of any particular MS-Access instance. All MS-Access dependencies are placed in the *management* interface.

To the MSDL layer, an association between itself (A) and a peer (B) over a particular MS-Access instance is defined by an association record of the form:

$$(VPI_A, VPI_B, QOS)$$

The management interface provides mechanisms for the selection of MS-Access instances based on quality of service parameters, and once this has been done, provides the ability to allocate VPIs and modify association records. The necessary exchange of the VPIs and quality of service information between MSDL entities is the job of the user of the MSDL management interface. Users of this interface must be able to cope with the many different formats of VPIs which can be generated by different underlying MS-Access instances.

5.2.4 MSDL and internetworking

Although internetworking is not a function of a data link layer, there are a number of aspects of the MSDL layer which are aimed at providing support for fast internetworking. Some of these points have already been covered:

- the provision of MS-Access to support MSDL over heterogeneous networks,
- the desire to cut down on the processing in an internetwork router was a major factor in choosing a fixed sized MSDL PDU.

The final mechanism provided in MSDL for fast internetworking is the ability to concatenate two associations to provide low-level forwarding of packets between incoming and outgoing associations. As with MSDL end points, the mechanism is split into two parts: one part responsible for the *in band* forwarding of packets, and the other an *out of band* management interface to establish these concatenations of associations.

The in band forwarding mechanism is implemented by taking the MSDL PDUs presented in an MS-Access indication from the incoming network and after routing them, issuing one or more MS-Access request primitives to the outgoing network. In the case of an ATM network with similar MS-Access interfaces this represents forwarding at the level of a single MSDL PDU. In the case of a network such as Ethernet, this currently means forwarding up to 44 MSDL PDUs at a time.

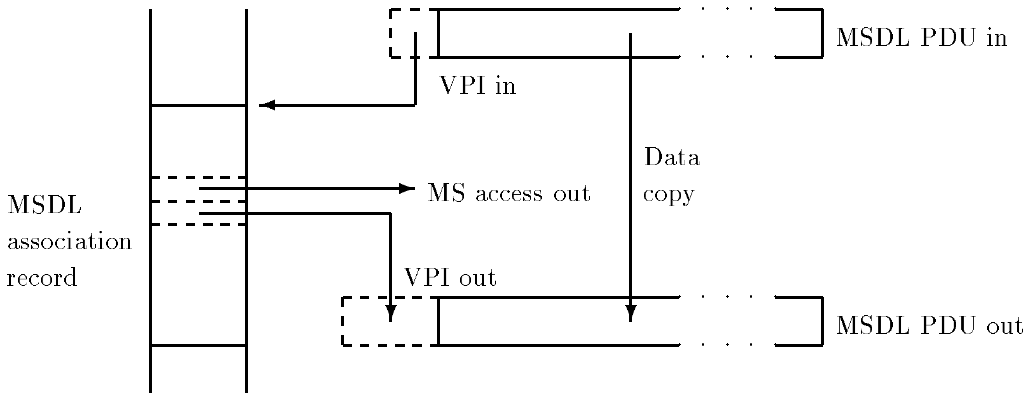


Figure 5.3: MSDL VPI mapping

The routing function is very simple (figure 5.3); it requires a lookup on the incoming VPI to find the association record containing the new association and MS-Access instance to forward on. This mechanism is simple enough that in this implementation, it is possible to place forwarded packets directly on the transmit queue of the relevant outgoing network from the receive interrupt routine of the incoming network. The aim in making internetwork routing so simple is to migrate this function into hardware.

When forwarding between heterogeneous networks, the desire to obtain a reasonable utilization of the outgoing network may require the MSDL code to perform some buffering. This is especially likely when forwarding from an ATM network to an Ethernet.

5.2.5 Association setup

A *promiscuous association* is one on which an MSDL entity is willing to receive MSDL PDUs from anyone. These PDUs must be self contained, that is the higher level information they carry must be contained within the single MSDL PDU as they cannot be related to any other PDUs arriving on the same association. The multi-service network layer (MSNL) performs the function of association set up, and uses these promiscuous association for this purpose. They are also used to implement various control functions, and it is not envisaged that these will be used for any other purpose, or be provided as a user service.

5.3 Multi-Service Network Level

The multi-service network layer (MSNL) is an internetworking service which is based on the idea of MSNL *liaisons*;. An MSNL liaison is a *lightweight* concatenation of MSDL associations. There are three important aspects to MSNL:

- it defines the MSNL address,
- it defines association and liaison set up procedures,
- it does not multiplex its liaisons over MSDL associations.

Defining liaisons as *lightweight* in nature, means that any node active in the liaison can unilaterally decide to tear it down. Similarly, liaison set up can be refused or could have the side effect of breaking another (underused) liaison. As all of these mechanisms result in the MSNL end point entities being informed of the break or refusal in a reasonably timely manner, they can be used to apply congestion control. Intelligent MSNL entities could use this information to try alternate routes, although care is needed to to avoid terminally catastrophic behaviour reminiscent of virtual memory thrashing.

5.3.1 MSNL Addressing

As presented in section 5.2, MSDL has no mechanism for naming peers before establishment of an association between them, and hence cannot itself provide a mechanism for setting up associations. An MSNL address provides this naming mechanism, and so *can* provide an association, and hence liaison, set up procedure.

An MSNL address is a pair (MSNL.id, MSNL.port). An MSNL identifier is 4 octets which uniquely identify an MSNL service access point (SAP) to all MSNL entities. An MSNL port is 4 octets which uniquely identify an MSNL client to the relevant MSNL entity.

In the simple case, an MSNL identifier may be one-to-one with a host. However, this is not necessarily the case; there may be one MSNL SAP for several hosts, or several SAPs per host. An MSNL port identifier is allocated to an MSNL client by its local MSNL entity; this happens either when a client indicates its interest in *listening* for liaisons, or when it requests *establishment* of a liaison.

5.3.2 MSNL liaison set up

Setting up an MSNL liaison involves establishing a concatenation of MSDL association hops. The liaison and association set up takes the form of a series of request / response interactions between the MSNL entities at the nodes concerned. The MSNL entities to be involved in a liaison are selected as part of the routing algorithm, with due consideration to the quality of service requested.

Promiscuous MSNL

A *promiscuous liaison* is built directly on top of a single promiscuous association, and is used as the basic mechanism for the establishment of real liaisons. These allow the transfer of a single MSNL PDU between MSNL entities, and so the *request*, *response*, *wait* and *kill* packets to be described (sections 5.3.2 and 5.3.2) must fit into a single PDU.

Before any attempt to establish a liaison, the VPI of the promiscuous association for the MSNL management entity must be found. The mechanism for finding this VPI is implemented by the address mapping service[3] (AMS) at one or more *well known* VPIs. The AMS performs a simple mapping function from an MSNL address to a well known VPI at which the relevant MSNL entity is listening for liaison setup requests. This AMS can also be implemented in a distributed manner by using a MAC level multicast or broadcast, if this exists, and can support the transfer of at least one MSDL PDU.

Once the VPI for a particular MSNL management entity is known, further liaison setup requests can be targeted directly at the relevant node, instead of using the AMS mechanism. In the case of repeated failure to contact a MSNL entity using a particular VPI, the AMS mechanism can be reused. Such loss of contact may well happen due to a reboot of the the relevant host or interface.

A demonstration AMS has been built. The functions supported by this server are even simpler than those performed by the original Cambridge Ring nameserver [Needham 82], and experience shows that such a simple object can be made extremely reliable. Current implementations of MSN in use on the Ethernet and CFR use the broadcast mechanism.

[3] Appendix A describes the functions of the address mapping service in more detail.

Routing and Quality of Service

When establishing the liaison, not only the desired MSNL address, but also the quality of service parameters are passed to the chain of nodes involved. This allows the algorithms within the MSNL, MSDL and MS-Access layers which implement the bandwidth reservation scheme to accept or reject the liaison, depending on their current state.

Each MSNL entity involved in the liaison set up process must perform routing based on the desired MSNL address and quality of service. This involves selection of an appropriate local MS-Access instance, and of the next node in the chain. Quality of service considerations are used to affect both of these decisions. Not only the type of traffic, but also the required rate must be considered when performing this routing. This allows access, albeit it at low rates, from current networks (e.g. Ethernet) to devices generating *real time traffic*, such as video and voice, which are only available on ATM networks. Apart from the quality of service considerations, the routing mechanism is very similar to that performed by IP, except that IP performs this routing *on every fragment*.

Whenever a liaison requires reserved bandwidth, all relevant entities involved in the liaison must record this information for the duration of the liaison, so that the reserved bandwidth is never greater than the entities' capacity. For different entities this capacity may depend on the network interface, network type, or switch / router capacity, and one of the requirements when implementing such entities is to determine their sustainable throughput under all possible types of traffic patterns. When such a liaison is successfully established, it is marked as a priority channel, and its bandwidth allocation is noted. MSDL PDUs being carried on these virtual circuits are then allocated their reserved bandwidth on demand by giving them priority over other traffic within the various switching components. Finally, by maintaining a rate count on each virtual circuit, the reservation policy may be enforced, firstly by discarding packets within switching nodes, and ultimately by breaking the liaison (section 5.3.2).

Whenever some entity is unable to fulfill the requested bandwidth reservation requirements, it refuses the liaison setup. The requesting MSNL entity is then free to try alternate routes, and this basic mechanism can form the basis of a network load sharing mechanism.

Distributed computing or data traffic will often ask for a quality of service which does not stipulate bandwidth requirements. However, certain switching nodes may wish to take account of such channels by applying a nominal bandwidth requirement. This can be particularly useful in avoiding flow control problems where a router connects networks with a large bandwidth mismatch, and the majority of traffic is data related.

The quality of service parameters mentioned so far have been related to bandwidth and priority. Higher level protocols also have requirements for end to end error detection, integrity (i.e. detection of data modification) and privacy. Such services are discussed later in the relevant sections.

Liaison Setup

Once an MS-Access instance and next node are selected, MSNL interacts with the peer MSNL entity to establish the MSDL association for this hop. Each association set up involves a request / response interaction, and although, on certain networks, this mechanism is superficially similar to the Address Resolution Protocol, it is not idempotent, and so has to deal with some issues of duplicate detection[4].

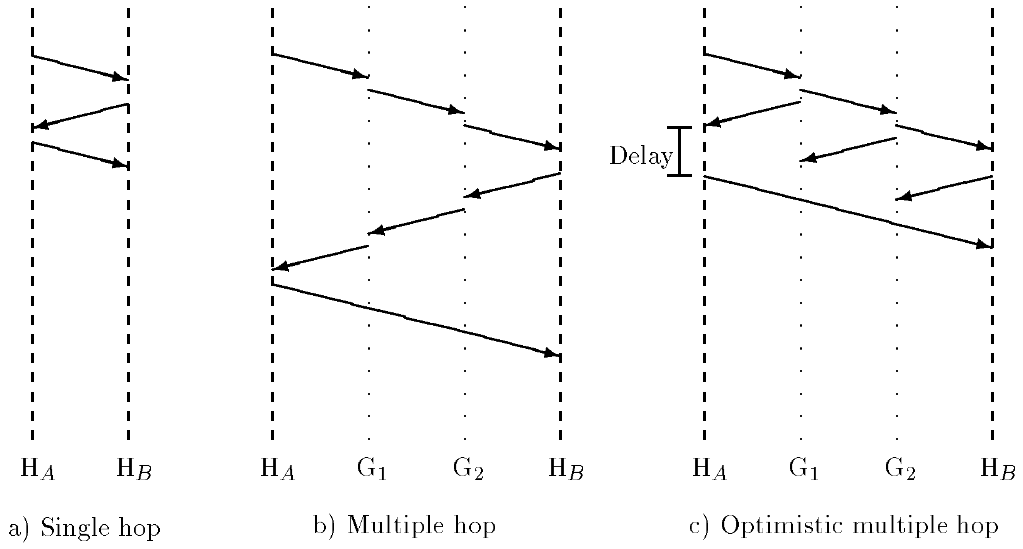


Figure 5.4: MSNL liaison setup

Considering the simplest case first, a liaison is to be established between two hosts on the same network, with no intervening MSNL router (figure 5.4(a)). The originating MSNL entity interacts with the MSDL management interface, selects a VPI for the backward half of the association, places this within a liaison *request* packet, and sends this to the desired correspondent by use of a promiscuous liaison. On arrival of this packet, the peer MSNL entity records this VPI in its active association table, and allocates a VPI for the forward half of the association. A *response* packet

[4]The mechanism is a simple special purpose RPC system, and the interface is more fully described in appendix A.

containing both VPIs is returned to the originator, which, once it has updated its tables, can then proceed to utilize the liaison for data transfer.

The mechanism which involves MSNL routers is the extension of this process, involving a whole series of request / response interactions (figure 5.4(b)). When a router receives a request it allocates and modifies the VPI field in the liaison request and forwards it to the next node in the chain of MSNL entities. This continues until the request finally arrives at the destination, where a response is generated, in the usual way. As this response passes back along the chain of MSNL entities, the sequence of returned VPIs completes the chain of associations to form the liaison. Once the originator receives the response, the liaison is considered established, and data transfer may occur.

To avoid multiple liaisons being incorrectly established simultaneously, duplicate requests are identified, and result in a *please wait* response to the originator from a router. The total time for a liaison set up then, is the round trip time from the originator to the destination, plus some management overhead at each switch node. Measurements show this management overhead to be about 1 ms per node, the round trip time depending on the network involved.

An optimistic version of this protocol is also possible in which a router returns its response *immediately*, on the optimistic assumption that the rest of the liaison will be successfully established (figure 5.4(c)). This leads to the originator being able to transmit data after only the round trip time between itself and the first router on its local network. Users of this mechanism need to be aware of the possibility that data packets transmitted immediately upon receipt of a reponse may *catch up* with the liaison set up request and response packets. In this case, the relevant MSNL node which has not yet completed the set up process for the next hop will drop the packet; this has the effect of adding a slight delay before the liaison can be used.

Liaison close down

Rather than leave a host transmitting into a void when a MSNL entity times out a liaison and deletes it (and the underlying associations), a mechanism is provided for the timely indication of such events to interested peer MSNL entities. Whenever a node decides to close a liaison, a *kill* packet is generated and transmitted over promiscuous liaisons to the one or two adjacent nodes in the chain of associations. On receipt of a kill packet, a node deletes its liaison and association entry and, unless it is one of the end points, forwards the packet. This mechanism alone does not cope with lost kill packets, so kill packets are also generated whenever any data arrives for defunct associations or liaisons. An MSNL entity also uses this mechanism when refusing a set up request.

Care must be taken to ensure that the bandwidth consumed by these kill packets does not swamp the network, particularly in response to a rogue host transmitting at a high rate on a dead liaison. For example, the introduction of a time delay between kill packets associated with the same liaison would be a possible mechanism to avoid excessive bandwidth concumption.

5.3.3 MSNL and multiplexing

The restriction that MSNL does not multiplex its liaisons over associations has the following direct benefits:

- no network level protocol information is transmitted in data packets,
- no demultiplexing is required in the MSNL destination entity to find the MSNL user,
- internetwork routing is performed once per liaison set up, not once per packet.

No network information is required on a per packet basis, as this information is transmitted to all concerned MSNL entities during liaison set up. As there is no multiplexing of liaisons over associations, this information can be implied from the MSDL association upon which the data is received. Cutting down on protocol header information is particularly important when the blocks of user data being transferred are small; e.g. a 2 ms toll quality voice sample is only 16 octets. The effect of this is that the MSNL PDU has exactly the same format as the MSDL PDU.

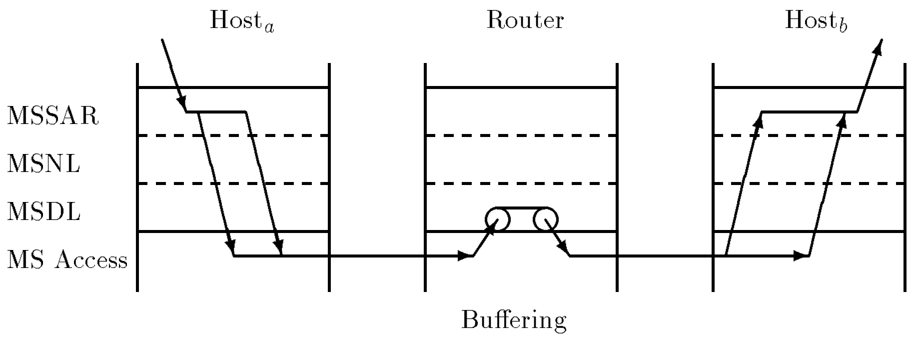


Figure 5.5: MSDL forwarding

As described in section 5.2.4, by implementing forwarding between MSDL associations at a low-level, packets could be forwarded with low delay. By stipulating no liaison multiplexing over associations, MSNL can make use of this facility to great effect. As part of the liaison set up procedure, an MSNL router informs the MSDL layer to connect the two associations together. While the associations last, MSDL will forward between these associations without reference to MSNL (figure 5.5). In terms of implementation, the MSNL layer acts only in a management function; although logically performing internetworking, this function is actually implemented in the MSDL level and managed by MSNL.

5.4 Multi-Service Segmentation and Reassembly

The service provided between MSNL users by the MSNL, MSDL and MS-Access layers is a stream of fixed size cells (perhaps presented in groups) each representing a MSNL SDU, with the possibility of this stream having certain bandwidth guarantees. Above this level there are differences in the services required by real time and data orientated applications; some real time applications may wish to use these cell streams directly while others, and most distributing computing traffic, will require a segmentation and reassembly function. Similarly, applications may require different levels of data security.

All these functions are supported within the multi-service segmentation and reassembly (MSSAR) level. As part of the liaison set up, the MSSAR service type is transferred between MSSAR entities, included in the quality of service parameters.

5.4.1 MSSAR Segmentation and Reassembly

The raw MSNL service is referred to as a *streamed liaison*, while the service provided by MSSAR is referred to as a *blocked liaison*. MSSAR inserts 2 octets of information per MSNL / MSDL SDU, concerned with both sequencing cells within a block, and discrimination between different blocks. The format of the header is shown in figure 5.6; an 8 bit sequence number provides blocks of up to 8K octets, while the 7 bit block idenitifier would require 127 consequtive blocks to be dropped before incorrectly reassembling a block.

IP fragmentation has been *considered harmful* by certain researchers [Kent 87], and although some of the ground rules for MSSAR are different, some lessons have been learnt. The sequence number starts at the maximum value and decreases to one. When an MSSAR PDU with the start bit set is received, the sequence number informs us of the amount of space required for reassembly. The MSNL / MSDL / MS-Access layers are defined (and implemented!) not to reorder PDUs, hence during reassembly of a block, any out of sequence packets received on a liaison imply lost packets, and the reassembly can be terminated. Similar action can be taken when

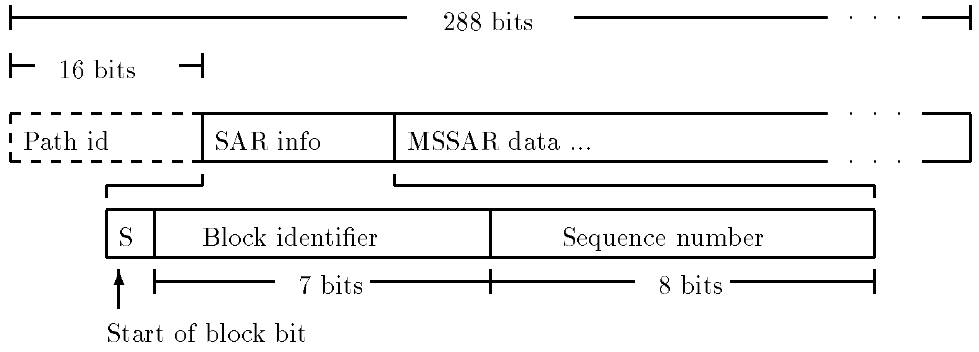


Figure 5.6: MSSAR and its encapsulation on CFR

a packet with a different reassembly identifier is received. Both these mechanisms mean that unlike IP, there is a well defined time at which to terminate reassembly.

Although the reassembly mechanism currently relies on the ordered delivery of MSSAR PDUs, it is possible in some of the ATM networks being considered that cell, and hence MSSAR PDU, reordering can occur[5]. If the reordering caused is fairly limited, the MSSAR mechanism can be simply extended to allow buffering for some number of out of sequence cells.

Within MSSAR, as in MSNL, no multiplexing is performed, and after reassembly of a block by MSSAR at a destination host, the data is fully demultiplexed by the underlying VPI. For example, in the case of a UNIX process using MSSAR directly (i.e. one requiring an unreliable block transport mechanism), this would be a socket in some user's address space.

Distributing computing applications may require reliable transport and bulk data transfer, and these issues are discussed later in section 6.2.

5.4.2 MSSAR and security

Applications built on top of the MSSAR service will sometimes require additional levels of service. Above the raw block service of MSSAR, there are three additional services, related to security:

- error detection,
- data integrity,

[5] Reordering is often found in ATM networks which have a low level response and retry mechanism *and* can have multiple outstanding transmitted cells; for example CBN.

- data privacy.

Although most networks provide a low level error detection mechanism, such as a cyclic redundancy check (CRC), end to end error detection is required to take account of errors in routers, and between IO interfaces and memory. In this case a simple checksum or CRC per block can be implemented. For data integrity, a cryptographic checksum must be used (e.g. encryption of the error detection code), while for data privacy the whole block needs to be encrypted. Both data integrity and privacy require a private key shared between the two communicating entities; the problem of key exchange is not dealt with by MSSAR.

Both error detection and data integrity require the insertion of in band information. To streamline implementation of these functions, this data is placed at the end of the block, as this allows calculation of the relevant quantity on the fly as part of the MSSAR function. For transmission this is straightforward, but on reception care must be taken with out of order cells. Two solutions are possible; one possibility is to use an algorithm which is immune to cell order, for example a simple additive checksum (if implemented in hardware, hopefully a more secure algorithm could be found). Another possible solution is to implement only the normal sequenced cell reception mechanism on the fly, and rely on a software implementation to cope when out of order cells are detected.

These functions are provided only for the blocked liaisons, as implementing them for streamed liaisons presents a number of problems. Error detection and data integrity work most efficiently on blocks, due to the extra in band information, and would use excessive bandwidth if implemented on a per MSNL PDU basis. For data privacy, a chained cipher (e.g. the data encryption standard cipher block chaining (DES CBC)) could be used on a streamed liaison. However, such systems often cannot recover from lost data[6]. Using a simpler block cipher (e.g the DES electronic code book) provides immunity from these problems, but results in the repetition of the cipher text when the same plain text is supplied; this may or may not be acceptable.

5.5 MSN lower levels and hardware

Many of the functions which have been described so far are amenable to efficient implementation in either software or hardware. This is due to placing the management functions out of band, hence simplifying the data path for both software and

[6]DES CBC works on 64 bit blocks, and can recover from a lost block in two cycles. In our case this would have the effect that a lost MSNL PDU would cause corruption of the first two 64 bit blocks in the following PDU.

hardware. In particular, consideration can be given to implementing the following in band functions associated with MSDL, MSNL and MSSAR in hardware:

- demultiplexing and multiplexing,
- internetwork routing,
- segmentation and reassembly
- security.

The relative merits of hardware implementation of these different components depends on the applications using them. Certainly using hardware to implement the MSDL demultiplexing function within the network controller allows the simple direction of real time streamed liaisons over private links from network controller to real time devices, which is required to prevent the host IO or memory bus becoming overloaded. Similarly, any real time devices wishing to utilize MSSAR or requiring an internetwork router to run at full line speed will dictate that the relevant functions be implemented in hardware.

Software implementations of encryption algorithms are several orders of magnitude slower than hardware, as are good error detection codes. Any implementation of the security features of MSSAR would probably provide these functions in hardware. A required feature of such hardware is that it must be possible to switch between different encryption streams rapidly, as on an ATM network, or even with large blocks on packet switched networks, cells for different secure streams will arrive interspersed. This is currently not the case for many hardware encryption implementations. Many DES implementations take a large number of cycles to load the key and initialization vector, and so would be unsuitable for an ATM environment.

Software is still required to implement the management functions, for example to establish and maintain associations and liaisons, and configure any hardware appropriately. This should obviously be as efficient as possible, but it is no longer required to operate at speeds comparable to the underlying network.

5.6 MSN Implementation

Implementation in hardware has not been attempted in the work for this dissertation. However, related work, presented in section 7.1.1, demonstrates the feasibility of certain aspects of a hardware implementation. On the other hand, implementation of the whole suite of protocols in software has enabled legitimate comparison of the protocol architecture's efficiency, with architectures providing similar functionality, which by necessity are implemented in software.

The presentation of the MSN protocol suite is in terms of layers of functional abstraction. Each level presents increased functionality over the lower. However any programmer who blindly followed this abstraction in terms of implementation would achieve extremely poor performance. A particular feature of the design of the MSN architecture is its refusal to multiplex at multiple levels, which allows an extremely compact and efficient software implementation of the data path. The provision of the MS-Access interface over the CFR and Ethernet device drivers requires only about 200 lines of C code each, while the data path for the 3 layers above are network independent, and both receive and transmit are implemented in about 300 lines.

The use of associations and liaisons, or more generally virtual circuits, allows many optimizations due to precomputation of, for example, MSDL and MSSAR headers, and appropriate fragmentation parameters for the relevant MS-Access instances.

5.6.1 Experimentation

A range of experiments, using both Ethernet and CFR, have been performed to demonstrate the lower levels of the MSN architecture. Three aspects have been investigated:

- end point, or host, performance,
- router throughput,
- router delay.

The performance in these areas has been measured, and compared to an implementation of the Internet protocols (IP, ICMP and UDP) written to the same coding standard, and in the same environment. The various experiments compared the performance of the use of MSSAR blocks versus UDP and ICMP datagrams.

The implementation for these protocols was performed in the MICA kernel on the Acorn Archimedes personal computer. The two networks used were a standard 10 Mbit/s Ethernet, and a 50 Mbit/s single slot CFR. A number of the performance limits were actually set by the nature of the IO bus of this computer, and so are worth mentioning. Both the Ethernet and CFR interface boards are restricted to a 16 bit interface, and (currently) cannot be accessed by instructions using burst mode memory cycles, such as performed when loading and storing multiple registers. This drastically reduces the rate at which data can be transferred between the host and the IO interface. In the case of the Ethernet board, on board RAM can be used to provide multiple buffers, and the operations of transmitting a packet and copying in the next can be performed simultaneously. Even so the best ever performance from this network controller was a disappointing 6 Mbit/s, and 4 Mbit/s was more common.

In the case of the CFR, multiple buffering is not provided, and the delay in the data transfer operation between host and interface becomes critical. Theoretically, for a ring of N slots, the CFR station chip can access 1 slot in $N+2$. As the ring on which this work has been performed is quite small (6 nodes), it is composed of only 1 slot, so, theoretically, the node should be able to get 1 in 3 slots. Although instances of this network now run at 75 Mbit/, even when using the slower 50 Mbit/s ring, only 1 in *16* slots was accessible, so that, of a theoretical point to point throughput of 12 Mbit/s, only 3 Mbit/s was achievable.

Even though the throughput of the network interfaces had a serious effect on the performance, the difference between the MSN and IP protocol families was still pronounced.

The experimental work has also used the first version of MSDL (section 5.2.2), i.e. the address / port pair form of VPI and a 32 byte MSDL PDU. The newer form with its 36 byte MSDL PDU would have resulted in better performance over the CFR (about 10%). However, time did not permit the conversion of all the components to the new format.

5.6.2 Results

In this section, all data rates are presented in terms of MSSAR or UDP user data rates. In the case of the Ethernet, this means that the bandwidth utilization of the transmission media and MSNL router is about 2% higher, while in the case of the CFR it is about 30% higher.

Host and interface performance was measured for the CFR and Ethernet as the maximum throughput obtainable on transmit and receive over a single, otherwise idle, network. Both single and multi-thread / multi-liaison tests were performed, as well as comparisons with UDP. The times shown in table 5.1 represent the best MSSAR throughput obtainable between user buffers, and were observed when transmitting maximum sized datagrams. The response mechanism in the CFR leads to back pressure on the transmitter if the receiver is unable to handle the data fast enough. Timings were also taken for a receiver which simply discarded the cells as soon as it read them from the interface, to avoid this interference.

Protocol	Network	Threads	Transmit	% Loss	Throughput
MSSAR	Ethernet	1	5172	0	5172
MSSAR	Ethernet	16	5957	20	4765
MSSAR	CFR	1	2690	0	2690
MSSAR	CFR	16	2789	0	2789
MSSAR	CFR + discard	1	2715	0	2715
MSSAR	CFR + discard	16	2830	0	2830
UDP	Ethernet	1	3064	17	2543
UDP	CFR	1	1648	0	1648

Maximum achievable transmission rate and throughput, in kbit/s, for MSSAR and UDP. This provides an indication of the upper bound on the host and network interface throughput.

Table 5.1: Host / network interface throughput

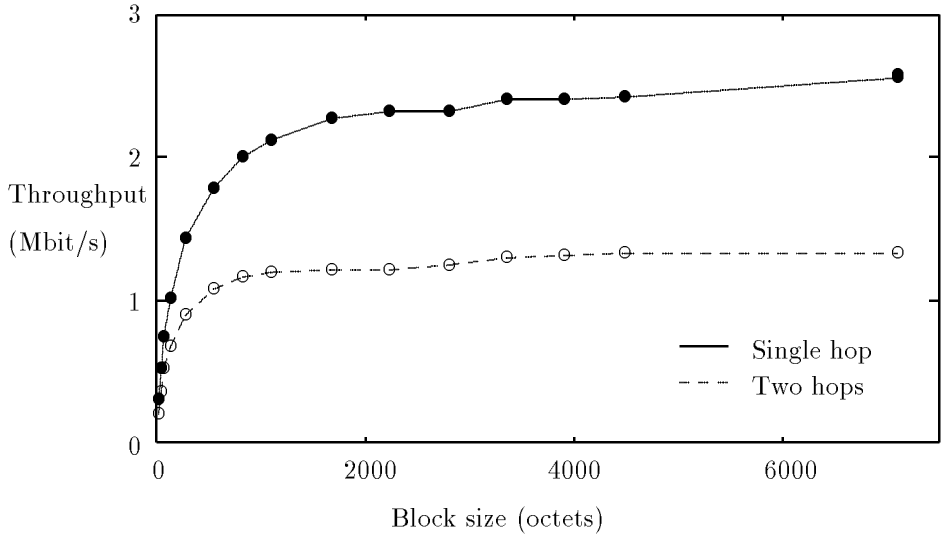
In the multi-threaded cases, all the bandwidth was shared evenly between the different threads. Higher rates could be obtained by modifying the driver code to transmit the same packet repeatedly, but the speed up was only slight, so the results presented represent a reasonable approximation to the maximum throughput for *user data* with these machines and interfaces. The very small increase in performance in the CFR when cells are simply discarded on reception shows that the receiver is not the major bottleneck. This is not the case in the Ethernet, where many parallel transmitters can swamp a receiver. Also worth note is the much lower maximum throughput for UDP, as in this instance the protocol code is the main bottle neck.

Protocol	Network	Throughput
MSSAR	Ethernet	2352
MSSAR	CFR	1219
UDP	Ethernet	800
UDP	CFR	848

Maximum MSNL and IP router throughput in kbit/s. In the case of the Ethernet, a variable delay was introduced into the transmit loop to achieve zero packet loss.

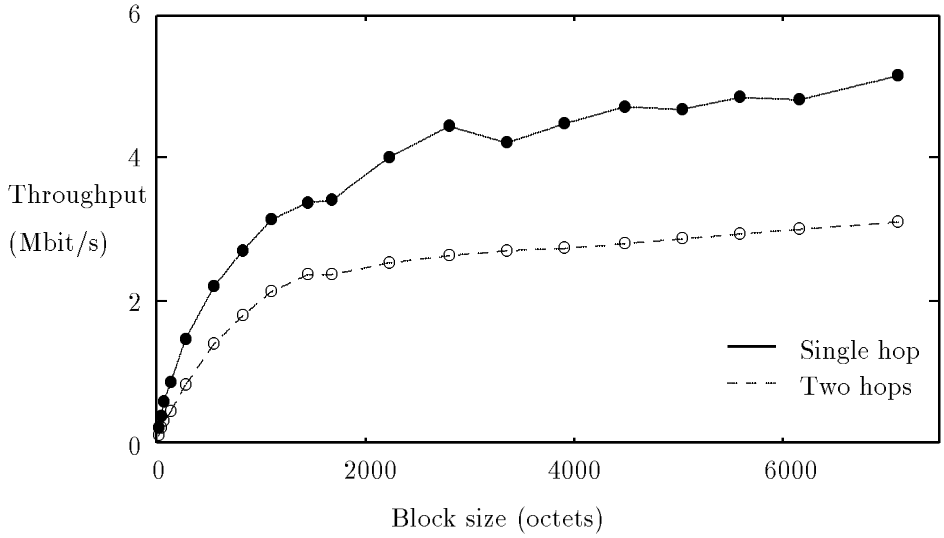
Table 5.2: MSNL v. IP router throughput

Throughput experiments with an Ethernet to Ethernet, and CFR to CFR, MSNL router confirmed this. In an MSNL router, the data must be copied across the IO bus twice, so if this were the bottleneck, approximately a halving in performance would be expected. The results presented in table 5.2 confirm this. Figures 5.8 and 5.7 present the router throughput graphically.



Maximum MSNL throughput on a CFR, and through a CFR to CFR router.

Figure 5.7: MSNL CFR router throughput



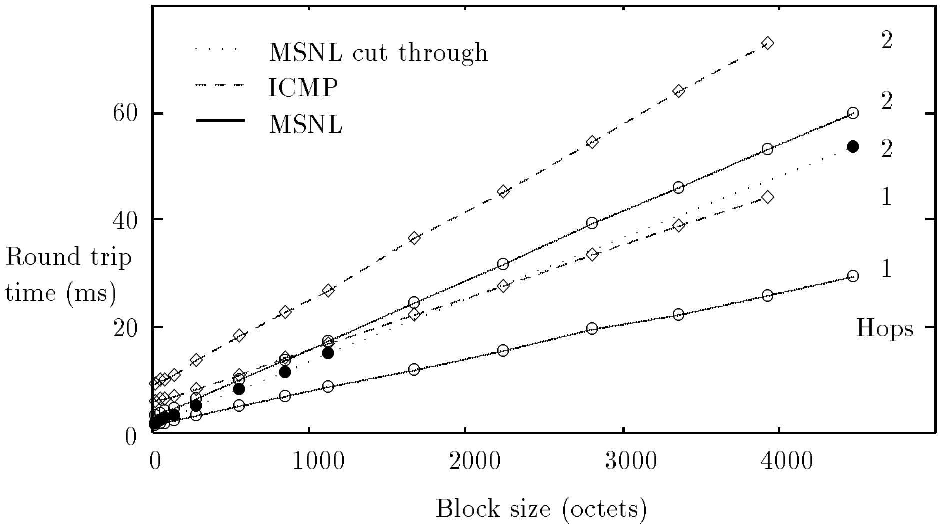
Maximum MSNL throughput for a single Ethernet, and through an Ethernet to Ethernet router. Timings through the router were obtained by slowing down the transmitter to achieve 100% block reception.

Figure 5.8: MSNL Ethernet router throughput

Kernel	Network	Block Size	MSSAR 1 Hop	MSSAR 2 Hop	MSSAR 2 Hop[1]	ICMP 1 Hop	ICMP 2 Hop
MICA	CFR	28	1.54	3.09	2.03	5.81	9.10
MICA	CFR	1456	10.80	21.40	18.55	20.00	32.64
MICA	CFR	3920	25.80	53.20	46.80	44.10	73.00
UNIX	CFR	28	-	-	-	4.00	7.00
UNIX	CFR	1456	-	-	-	29.00	53.00
MICA	Ethernet	28	3.80	7.30	-	6.60	10.00
MICA	Ethernet	1456	11.10	22.10	-	18.75	40.20
UNIX	Ethernet	28	-	-	-	6.00	9.00
UNIX	Ethernet	1456	-	-	-	51.00	65.00

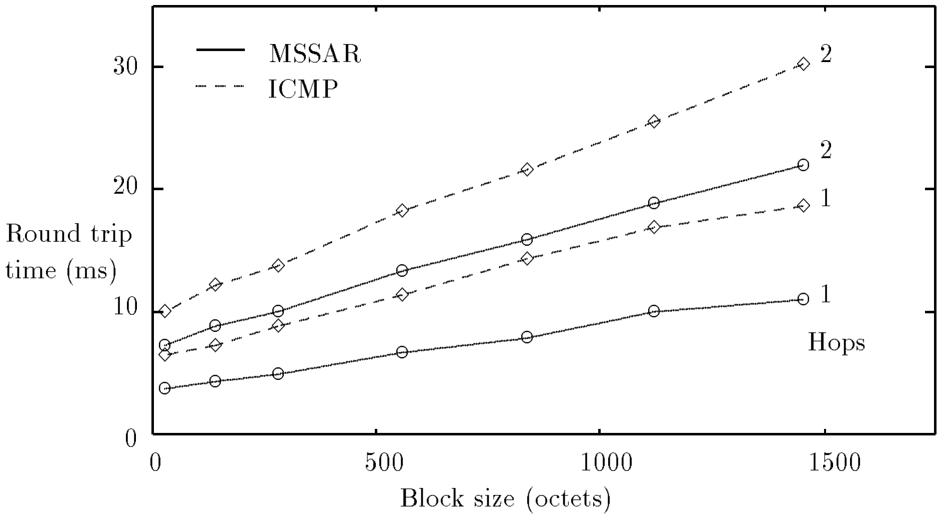
These timings represent the round trip time in milliseconds for MSSAR and ICMP *echo*.
Note [1]: in this instance, the MSNL gateway was performing single cell forwarding, or *cut through*.

Table 5.3: MSNL v. IP router delay



The round trip time for MSSAR and ICMP echo over CFR. ICMP shows a significantly larger delay even in the single hop case.

Figure 5.9: MSNL v. IP CFR router delay



The round trip time for MSSAR and ICMP echo over Ethernet.

Figure 5.10: MSNL v. IP Ethernet router delay

Router delay was measured by implementing an echo service over MSSAR[7], and comparing it to ICMP echo. The performance of IP fragmentation in the MICA kernel was significantly poorer than unfragmented IP, so measurements were only taken for unfragmented datagrams. Some of these results are shown in table 5.3, the full set are presented graphically in figures 5.9 and 5.10. Firstly it should be noted that the ICMP times for MICA are comparable, to those produced between UNIX kernels running on exactly the same hardware configuration, for both the CFR and Ethernet.

The MSNL router is capable of operating in a standard block store and forward manner, or in the single cell forwarding mode for the CFR (see section 5.2.4). For the single cell forwarding mode, the increase in end to end cell transit time in going from one hop to two hops is only 250μs per cell. This represents both the delay in routing (implemented in software), and the transmission time over the extra hop, including the two transfers between the network interfaces and memory. Calculations show that this splits into three roughly equal components:

1. set up of forwarding buffer - 90μs,
2. routing of cell - 70μs,
3. reception and transmission on next hop - 90μs.

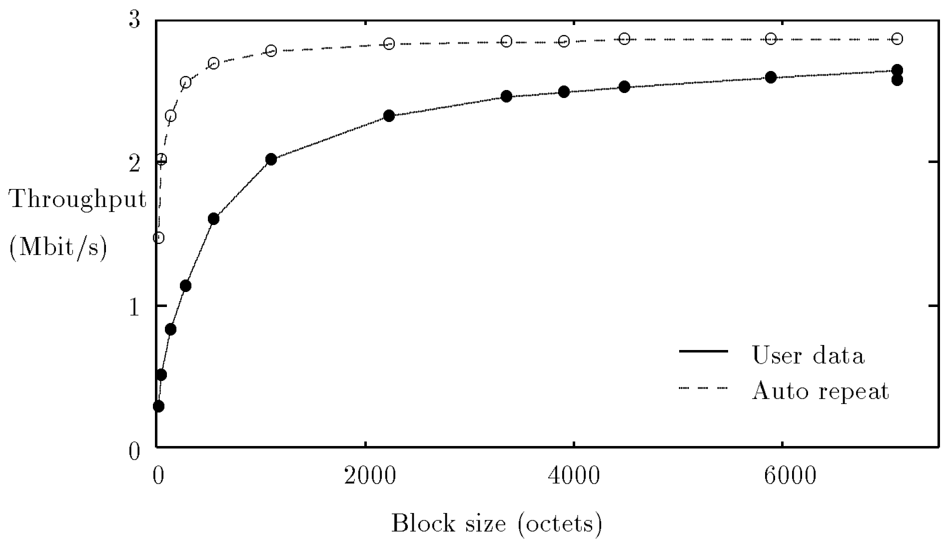
[7]This used the subset of the Unity RPC mechanism discussed in section 6.2.3

The setting up of a forwarding buffer is performed once per several hundred cells, and so only has a small impact on throughput for a continuous stream of cells. The time for the reception and transmission on the next hop is composed of the time to read and write the data across the IO bus, and the transit time around the ring. On the ring in use for these experiments, the transit time was about 6μs, to that most of the 90μs is involved in IO bus cycles.

Block size	Number of PDUs	User data Time	User data Rate	Auto repeat Time	Auto repeat Rate
7112	254	21529	2.64	19876	2.87
3920	140	12549	2.50	10999	2.85
1120	40	4432	2.02	3218	2.78
280	10	1984	1.13	872	2.57
56	2	877	0.51	222	2.02
28	1	787	0.28	152	1.47

This table presents the time interval between transmissions (in μs), and achieved rate (in Mbit/s), for different sizes of MSSAR block. To demonstrate the cost of interaction with user code, the performance for both user data and for a low level block repeat mechanism is shown.

Table 5.4: MSSAR performance



The performance of MSSAR for both user data and a low level block repeat is shown. This clearly shows the cost associated with interacting with the user, and that MSSAR is capable of near maximal performance even with small blocks.

Figure 5.11: MSSAR performance

Using a low level block retransmission scheme, timings were obtained which demonstrate the performance of the MSSAR segmentation and reassembly algorithm. The results are presented in table 5.4 and figure 5.11. Calculations show that the time between cell transmissions is approximately 80μs, with an overhead of about 90μs per *block* to perform segmentation and reassembly. Experiments with a receiver which simply discarded cells show that once running, segmentation and reassembly is not the bottleneck.

5.6.3 Conclusions

The first conclusion to draw concerns the design of network interfaces and their attachment to hosts. The poor performance of the CFR interface, is almost entirely due to its interrupt driven, lock / step mode of operation, such that very little pipelining is possible. This is in part due to the single buffered nature of the CFR CMOS station chip itself, and in part due to the absence of intermediate buffer RAM in the interface between the host and the station chip. The latter has been partly remedied in the design of the CFR MAC bridge (section 7.1.1).

This feature of the CFR is seen especially well in the MSNL router. Forcing all operations to be sequential means that the delay per cell through the router is cumulative, hence a time of 160μs, or a maximum (burst) throughput of about 1.4 Mbit/s. On the other hand, a pipelined system should allow a figure much closer to the slowest component in the path, or 90μs per cell giving throughput of 2.5 Mbit/s.

The reduction of throughput for the MSNL routers due to the reduction in block size can be misleading (figures 5.8 and 5.7), as it would seem to indicate that a real time device, which wished to use large numbers of small blocks or even streams of single cells, would experience poor performance. Obviously, for networks such as the Ethernet, better throughput is obtained when an MSSAR block fragments into some number of maximum sized packets. However, as we are interested in real time traffic, the problem encountered in ATM networks is of more relevance. There are two areas to be dealt with: the MSNL router and the MSSAR engine. The best times from the MSNL router were obtained when it was placed in single cell forwarding mode. In this mode, it knows nothing of blocks, and deals only in cells, and so routes the same number of cells per second independent of the block size. Similarly, although a small amount of extra work is required in MSSAR for the first and last cells of a block (totaling 90μs), it is significant only for the smallest of blocks.

5.7 Summary

The design and an implementation of the levels of the MSN architecture up to the network level have been presented. The low levels of the MSN architecture provide the basic mechanisms for higher level services to utilize ATM networks effectively. The facilities provided include:

- the use of a single addressing scheme and liaison set up mechanism for all traffic types,
- routing which takes into account quality of service and bandwidth allocation,
- fast internetwork routing,
- support for heterogeneous networks,
- block and security features,
- single cell forwarding and multiplexing.

The main data path of the architecture is kept simple to facilitate implementation in hardware, and hence provide the full bandwidth of the network to end systems. One important aspect of this is the removal of layered multiplexing.

Chapter 6

MSN high levels

The work presented in this chapter deals with the MSN architecture design and implementation work for levels of protocol equivalent to the ISO OSI reference model transport, session and presentation levels. The discussion of the protocol layers up to the network layer has been concerned with efficient end to end transfer of data, and expeditious transfer within a host system between the network controller and the user of this data. The applications using the data fall into two categories, real time and distributed computing.

6.1 Real time services

The discussion of the lower layers of the MSN architecture has concentrated on out of band management, or the separation of control and data. This has been presented as a mechanism to improve network performance. In the case of many real time devices, while the control functions can be implemented in software, implementation of the data path in hardware is a necessity, and so these devices fit naturally into the MSN architecture.

Often, the out of band control mechanisms are examples of distributed computing systems, and as such, the relevance of the MSN architecture for them is dealt with in section 6.2.

Details concerning the data path in real time devices raise the issue of particular coding techniques, for example coding for video and voice. These areas have not been worked on, as they are a topic of wide ranging research in themselves, and are beyond the scope of this dissertation. For this reason, although demonstrations of video over MSNL have been performed, they have involved low resolution pictures

(256 x 256 x 4 bits monochrome) without any compression.

6.1.1 Internetworking

The internetworking approach of the MSN architecture has a number of benefits, both direct and indirect, for applications generating real time traffic. The direct benefits come from the ability to internetwork with ATM networks, and hence allow extensibility and upgrading of networks components, and to extend the network over publicly available networks.

The indirect benefits come from the ability to interact with real time components from traditional packet switched networks. As an example, consider access to a video storage server. A video storage device will need to be attached to a high speed ATM network because of its bandwidth requirements during *record* and *playback*. The implementation of certain applications which cannot handle or do not require high bandwidth, may be better performed in a standard environment, such as UNIX over Ethernet, or Token Ring. There is a large number of such applications, including, for example:

- a video mail system which wished to use standard electronic mail transfer components, and hence is constrained to low bandwidths,
- a compression server which performed off-line compression of video files for storage optimization, which may only be able to run at a few frames per second,
- a *rogues*' gallery, which only wished to access single frames and display them as bitmaps on a window display, such as an X-server.

By providing an internetworking service, the MSN architecture provides the basic interconnection which allows such applications to interact with real time components from traditional packet based networks. This prevents real time components having to deal with multiple protocol stacks and presenting a number of different interfaces.

There is also a cost benefit as only a small number of MSNL routers (perhaps only 1) may be required between the ATM network and the packet switched network to provide such services. This is a cost effective option when compared to that of placing all machines on the relevant ATM network, especially if they cannot utilize the bandwidth supplied.

6.1.2 Real time transport

One aspect worthy of mention is that of a transport service for real time traffic. The MSSAR and MSNL services provide an unreliable service, and it might be envisaged that the MSN architecture would define a reliable transport service. In fact, no such service is defined for real time traffic, and this is related to the issue of coding.

The usual mechanisms of reliable transport, whether windowed or rate controlled with positive or negative acknowledgement, rely on retransmission of lost data, and are simply not responsive enough for real time traffic. For example, the retransmission of components of a video frame could well be useless, as the retransmitted data may arrive after its parent frame has been displayed. This is an area in which higher bandwidth networks and faster hardware will not solve the problem, as this situation will always arise when the round trip time, the lowest bound of which is dictated by the media transmission speed, is larger than the video inter-frame time. To overcome this problem, a number of mechanisms have been suggested which build directly on top of an unreliable transport service, such as MSNL or MSSAR, and are independent of round trip delay[1]. For example, the use of forward error correction or error compensation in the encoding [Verbiest 89], or splitting the traffic into a high and a low priority channel [Ghanbari 89], have been suggested. The selection of the exact parameters for these coding algorithms is highly dependent on the error properties of the underlying networks, and adaptive schemes may be a necessity.

6.2 Distributed Computing

One of the aims of the MSN architecture is to extend the use of distributed computing applications into wide area networks, where the traditional applications such as mail, file transfer and remote login, are mainly used.

In this area, the work has been aimed at investigating the effect of extending the MSN architecture into well understood distributed computing components, as opposed to inventing radical new mechanisms for distributed computing. The aim is to utilize the underlying MSN architecture capabilities effectively, as well as obtain advantage from applying similar design techniques to the mechanisms present in these distributing computing components.

The main distributed computing mechanism discussed is remote procedure call (RPC) [Birrell 84]. RPC is widely used in distributed computing, and when combined with tools for automatic stub generation and interface checking provides a

[1] Adaptive schemes to overcome the problem of a bounded round trip time have been suggested, but these work by modification of future frames to compensate for previous errors [Wada 89].

convenient application programming mechanism. From the ISO OSI reference model point of view, an RPC mechanism defines a session layer and a presentation level encoding. In particular it defines:

- session set up and maintenance, (binding of interface instances to addresses),
- session synchronization, or a transaction mechanism which deals with duplicate detection, acknowledgement and error recovery,
- a fixed set of presentation encodings in an implementation independent syntax.

Users of RPC systems sometimes have a requirement for very large amounts of data to be passed between the client and server. Most systems support this in one of two ways: either by defining a reliable bulk transport mechanism for the transport of large RPC messages, or by utilizing multiple small RPCs.

RPC has been used as an example of a distributed computing mechanism, but it is hoped that the general lessons concerning the use of the MSN architecture are also applicable to traditional networking, as well as different models of distributed computing (e.g. ISIS [Birman 87], Linda [Carriero 86], Mach [Accetta 86]).

6.2.1 Benefits of MSN

Certain benefits accrue directly from using the MSN architecture as the underlying internetworking layer. To satisfy the demands of real time traffic, the approach taken in the lower levels of the MSN architecture is to provide a service with guarantees on bandwidth and the delay and jitter in the channel. As a side effect, this approach also has advantages for distributed computing traffic (e.g. RPC) even when used naively. Although most distributed computing traffic will be low priority traffic[2] without a bandwidth reservation, at normal utilization this traffic still benefits from the low delay and jitter characteristics provide by the MSNL routers.

In many measurements of RPC systems, the performance of the system is critically dependent on the network round trip time. Many approachs have been taken to reduce this time, and most have involved removing in band information [Nelson 81]. Unfortunately, the mechanisms supported by this in band information are not usually replaced, and so we see many RPC mechanisms throwing out internetworking, and using raw datalink levels to achieve high performance. Some RPC mechanisms have not removed this internetwork level, thus accepting degraded LAN performance, for the ability to internetwork. The major issue then is the performance

[2]The exception here is distributed computing traffic associated with network management.

through internetwork routers. The lower layers of the MSN architecture support the extension of distributed computing to an internetworked environment, by providing the design of mechanisms for a fast internetwork router, while maintaining low (in band) costs in the LAN environment.

The move to distributed computing based on RPC does not replace the need for reliable bulk transport, and, as previously mentioned, underlying many RPC systems there is often a reliable transport mechanism; for example VMTP in the V-kernel [Cheriton 86] and REX in the ANSA testbench [ANSA 89]. Both these transport mechanisms, and others unrelated to RPC which have been investigated recently (e.g. NETBLT [Clark 87]), have been based on combinations of rate based flow control and selective retransmission. These mechanisms are in contrast to the windowed and positive acknowledgement mechanisms of more traditional transport protocols, such as the DARPA transmission control protocol (TCP [Postel 81] and the ISO OSI transport service [ISO 8073 84]). The suitability of ATM networks for rate based mechanisms has already been discussed (section 2.3.2), and as the levels of the MSN architecture presented so far preserve the characteristics of bandwidth sharing and graceful degradation under overload, these protocols continue to be well supported.

6.2.2 Effective MSN utilization

So far consideration has only been given to using the MSSAR service as a datagram service, and using an unmodified RPC package. This naive use leads to the inclusion of significant RPC data overhead, while the implicit information present in a MSSAR liaison is ignored. These RPC mechanisms usually deal with datagrams, so use of an unmodified RPC mechanism will cause demultiplexing information to be passed across a liaison with every RPC packet. In certain circumstances, this blind desire to find and use the lowest common denominator packet, the unreliable datagram, has even lead to the use of TCP as an unreliable datagram service.

Multiplexing

One of the design principles within the MSN architecture has been the reduction of multiplexing to the minimum required, and so far, the only multiplexing performed in the MSN architecture is at the MSDL level. This minimal multiplexing principle can be simply extended to RPC systems running over MSSAR. The technique is to pass over as much of the demultiplexing information as possible *once*, at the time of liaison set up. This has a number of effects:

- a saving in bandwidth,

- no client / server demultiplexing,
- no server interface demultiplexing,
- optimistic client demultiplexing.

The nominal bandwidth saving is significant for RPCs involving small request and reply packets, and although for networks with a minimum packet size, such as Ethernet, this may not result in an *actual* saving in network bandwidth, in the case of ATM networks a minimal RPC may then be able to fit in a single cell.

Many RPC systems allow a task to be both a client and a server at the same time, that is to invoke RPCs and to be invoked via RPCs. Within such systems datagrams must first be demultiplexed to decide whether the packet is concerned with an outgoing or an incoming RPC call. In these circumstances, the use of a liaison removes the need to demultiplex on a per packet basis, as the liaison establishment informs both client and server which *end* of the protocol they need to attach to that particular liaison.

Typically, the binding between a client and server is done at the level of an *interface*, each interface representing a module or set of procedures. A server may export multiple different interfaces, for example different user and management interfaces. A client binds a local interface, whose implementation is composed of simple stubs, to a particular remote interface implementation instance. When using the MSN protocol suite, a single liaison is used for each of these binding steps, so that when the liaison is established between client and server, it is possible for the server to precompute the relevant interface to invoke, and use this for all incoming packets, rather than demultiplex and select an interface implementation on a per packet basis.

There are two further demultiplexing steps required, which need careful consideration before extending the MSDL demultiplexing function further; namely selecting the required procedure within an interface, and demultiplexing replies for client threads. The use of separate liaisons per procedure is not considered reasonable, as this would result in several orders of magnitude more liaisons, and the associated management problems. In fact this demultiplexing step is not a bottle neck, as it does not require any synchronization between threads.

The simplest mechanism which is often used to perform the demultiplexing of replies, is to have a random receiver thread wake up, perform the demultiplexing, and then wake up the relevant client. In certain environments, for example if upcalls are possible, this action can be performed by the network interrupt handler, so saving a thread context switch. Using liaisons, most of the benefits of the latter mechanism can be attained without the requirement for upcalls or intelligent interrupt handlers, by using *optimistic* clients.

In the usual case, only one client thread has an outstanding call on a liaison, so when a reply arrives, this thread can be woken up immediately. A problem only arises when there are multiple client threads with outstanding RPC requests. Although this may be a rare occurrence, this style of programming must be supported, as it is sometimes used to increase throughput in a situation such as file transfer using RPC (for example in a file transfer mechanism). In this case clients with outstanding calls on a particular liaison wait for replies in a fifo order, that is, the order in which they made calls. When a reply arrives the first client thread is woken. Most of the time this reply will be the correct one for the woken thread; if it is not, the client thread must interrogate the queue of waiting threads, find the correct one and wake it up. In the few instances where the application has a mix of concurrent RPCs which cause the latter behaviour too often, the application can always resort to multiple liaisons.

All these multiplexing optimizations aim to reduce the multiplexing overhead in that part of the RPC run time systems which performs the functions of the session layer. As with the OSI ISO model, the presentation layer of the RPC mechanism performs no multiplexing.

Security

A number of mechanisms for secure and authenticated RPC, based on both public and private key encryption schemes, are to be found in the literature [Birrell 85] [Birrell 87]. All aim to distribute a key in a secure manner to both the client and server, which is then used as the key for some crypto system which is the basis of the security mechanism. The use of this key for more than one RPC, i.e. the use of some form of RPC session mechanism, is obviously required if secure RPC is to be efficient.

In implementing a secure RPC system over the MSN architecture, any of the published mechanisms can be used to distribute the key, while the security features of MSSAR can be used to implement the secure channel between the client and server directly. The use of a single liaison per RPC session makes use of the MSDL demultiplexing function to allow ciphering to be performed on the fly as packets enter and leave the host, as discussed in section 5.4.1.

6.2.3 Implementation

The particular system chosen for the implementation of these ideas was the Unity remote procedure call (RPC) system [McAuley 87]. Many of the application level entities within the Unison network, in particular those associated with exchange management, use this RPC mechanism, as it provides interoperability between a wide range of different application environments[3]. The intention was to investigate the benefits to be had from modifying the RPC code to use the concept of liaisons.

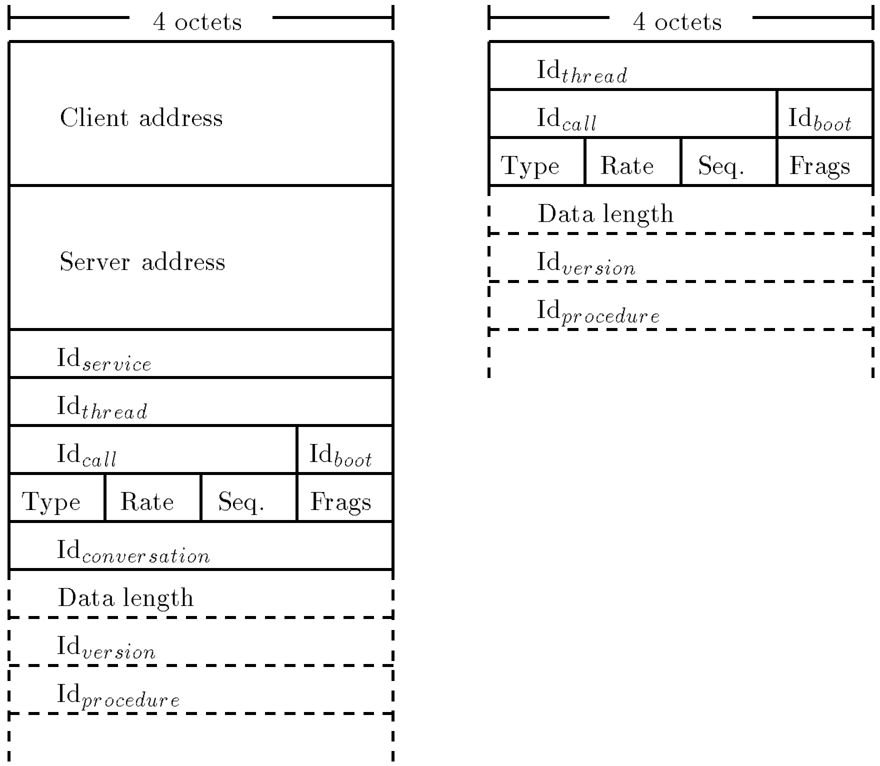
In a perfect world, all this work would have used the high performance version of the MSN architecture over MICA. However, a full implementation of the Unity RPC was not available for this system. Instead, a subset was implemented and used to obtain an estimate of the best RPC performance possible from this system, while the main work on the impact of liaisons, and multiplexing, was performed using the Modula2 implementation over SunOS. This was chosen over any other RPC system due to the author's understanding of the system, but the lessons learnt could equally be applied to the other Unity implementations and other RPC systems.

Within the MSN network level it was possible to dispose of all of the in band protocol information; in the case of Unity RPC it is only possible to cut this information down from 44 octets per packet to the 12 octets (figure 6.1) which are required to maintain the duplicate detection mechanism. The information which has been left out is still required to be passed to the server when the liaison is established between the client and server. This could be passed as *quality of service* data. However, this is an inappropriate use of this service, as the data passed is of no use to the underlying layers, as well as requiring the ability to convey large amounts of quality of service data. Instead, for the first call on a new liaison, the client always sends packets with *full* headers, and on reception, the server performs the demultiplexing functions, and caches the relevant information. Calculating and saving this data involves some overhead for the server on the first call, but subsequent RPCs are faster.

6.2.4 Results

The performance of the RPC system over the MICA kernel compares well with other RPC systems in terms of LAN performance (table 6.1). As in the case of the throughput experiments for the MSNL router, the performance achieved in these experiments was restricted by the relatively poor throughput of the network interfaces.

[3]Within Unison, Unity has been implemented in Modula2, C, BCPL and OCCAM, over a range of operating systems.



Datagram header **Liaison header**

The normal headers for the datagram and liaison based Unity RPC mechanism are shown. The protocol implements a rate based flow control mechanism for multi-packet messages. The data length, version and procedure identifiers are present only in the first packet in each message.

Figure 6.1: Unity RPC headers

Comparison of the performance of Unity and other RPC systems in an internetwork environment is not so simple. Some RPC systems do not support internetworking while others have not published the relevant measurements. For this reason comparison has been made, for both the LAN and internetwork measurements, with ICMP echo; this provides a useful lower bound on the performance of IP based RPC systems. These figures have already been presented in table 5.3, as these RPC based experiments were also used to measure the delay in the MSNL routers.

So far, the results have dealt with the straightforward use the MSN architecture for RPC. Work was also carried out within a single UNIX address space to investigate the effect of using liaisons in a more integrated fashion within a full implementation of the Unity RPC system. This involved implementing the mechanisms outlined in section 6.2.2.

Network	Kernel	Time (ms)	Throughput
Ethernet	MICA	3.80	3792
Ethernet	Amoeba[1]	1.40	5416
Ethernet	V[2]	2.54	3680
Ethernet	Taos[3]	2.66	1824
CFR	MICA	1.80	2568

Comparative timings for different RPC systems using Ethernet. The times are given for *null* RPCs and maximum throughput for a single thread. Timings for MICA over the CFR are included to demonstrate the overhead, about 1 ms, in dealing with the particular Ethernet interface used.
Note [1]: source [Renesse 88].
Note [2]: source [Cheriton 86].
Note [3]: source [Schroeder 89].

Table 6.1: Unity RPC performance

Configuration	Time (ms)	Overhead (ms)	% improvement
Normal loopback	3.70	2.24	0
Optimized client	3.27	1.81	19
Optimized server + client	2.85	1.39	38
Local call	1.46	0.00	-

Timings for null Unity RPCs within a single UNIX address space on a Sun-3/260, with and without the liaison multiplexing optimizations implemented.

Table 6.2: Unity multiplexing optimizations

Table 6.2 shows the results of implementing these mechanisms. The time for a *local call* represents the fixed overhead of the presentation layer within the RPC system: the client and server stubs, the marshalling code, and entry into the RPC run time system. The *normal loopback* mode is the original unmodified Unity system using datagrams, with the loopback performed at the point at which a UNIX *send* call would normally be used. The other figures give the performance for Unity with client, and with client and server, multiplexing optimizations implemented.

6.2.5 Conclusions

The figures for the MICA implementation of the subset of Unity, support the claim that the MSN architecture provides effective support for distributed computing mechanisms (in this case RPC), while also providing a general internetworking service. The low delay characteristics of the MSNL routers, even when operating in a store and forward mode, also lead to significant improvements in performance in an internetworked environment.

Applying the liaison related optimizations to the UNIX / Modula2 implementation lead to a reduction of nearly 40% in the time spent within the session layer of the RPC run time system. It should be noted that in these experiments, only one thread was involved. In a highly loaded system with many threads active, the removal of most of the competition for shared resources would result in an even greater speed up.

6.3 Summary

Real time traffic has transport and presentation (encoding) requirements which are highly interdependent, and specific to the particular characteristics of the traffic type. The MSNL and MSSAR services are presented as the foundation on which to place specific higher level protocol stacks for the different traffic types.

Distributed computing traffic benefits directly from the use of the lower levels of the MSN architecture. Low delay through gateways is critical in extending RPC mechanisms into the wide area, and low jitter benefits transport mechanisms using rate based flow control. The basic form of the Unity RPC mechanism implemented in the MICA environment achieves comparable performance with equivalent mechanisms. Unity achieves this performance by use of the general internetworking mechanisms provided by the MSN architecture, as opposed to providing any special support for RPC. Hence, similar performance benefits could be obtained by other distributed computing mechanisms.

Extending the minimal multiplexing arguments used in the design of the lower levels of the MSN architecture to the RPC session level also leads to significant performance improvements. Up to a 40% reduction in the RPC session level overhead has been achieved by modification of the existing Unity RPC system. Hopefully, even larger benefits could be obtained by complete reimplementation with the MSN architecture design principles in mind.

Chapter 7

Related work

This chapter presents currently active work in various related areas of network research. Some are presented to provide contrast to the MSN architecture, while others are presented to support certain aspects of the architecture.

7.1 MAC Bridging

MAC layer bridges aim to extend a local area network architecture in a transparent way. For a single site, MAC bridging allows simple network extension without requiring administration effort to configure routing mechanisms, and often provides better performance than internetworking, due mainly to the comparatively poor performance of many network level routers. Unfortunately this mechanism can not be extended indefinitely, and presents problems in a number of areas:

- administration,
- security,
- efficiency,
- functionally transparency.

Extension of a network by the use of MAC layer bridges, can, in general, be performed only within a single administration domain. The identification, isolation, and correction of network and host failures is an important part of system administration within a networked environment, and would be hopelessly complicated if multiple administrations were involved. The recognition that access to the underlying media presents a security risk means that separate security domains, even within the same administrative domain, must be interconnected by some means other than MAC layer bridges.

The desire to make the extension transparent to normal stations, means that the bridges must be passive, or at least, communicate only with other bridges. This severely restricts their ability to cope with changes in network traffic patterns, and means they can provide few guarantees about the quality of service they provide. This also leads to the problem of load balancing between different bridges, or even routes, connecting the same two networks.

Performance transparency is not possible, but is usually good enough that higher level protocols are unaffected. Functional transparency is more of a problem, one particular example being broadcast packets. Attempting to support broadcast packets can lead to broadcast *showers* when bridges are in transient states, and, as each packet must be processed by every receiving node, also leads to significant use of processing power. Conversely, removing the support for broadcast in a bridged network results in lack of transparency, and the breakage of some higher level protocols. An example of this problem can be seen within the Athena project [Treese 88]; Ethernet broadcast must be provided to support address resolution for IP, but applications using broadcast have been removed.

7.1.1 CFR MAC layer bridge

Of particular interest and relevance to the MSN architecture is the CFR MAC layer bridge [Porter 89]. This work has a number of specific aims related to MAC bridging:

- implement a low level flow control,
- transparently reconfigure bridges,
- maintain full network throughput.

The CFR provides a per cell acknowledgement and retry, which can be used as the basis of a low level flow control mechanism. A receiving station which is unable to keep up with the rate at which a transmitter is attempting to send cells, will cause negative acknowledgements to be returned to the transmitter. A sensible implementation of a transmitter would then back off, and select another station which has an outstanding transmit request, and return to sending to the original destination some time later. One of the aims of the CFR MAC layer bridge is to extend this to a multihop flow control mechanism. A bridge which notices that it is temporarily unable to forward packets between a pair of hosts, due to the reception of negative acknowledgements or the build up of queues in the bridge, simply starts returning negative acknowledgements to the source station. When the blockage clears, the bridge restarts the forwarding.

The bridges also implement dynamic routing in software to allow reconfiguration of the network, to allow for availability of bridges, and the changing locations of particular CFR station addresses on the network. Finally, to maintain full network throughput, the forwarding and flow control mechanism within the bridges is implemented in hardware. The main bottleneck is the available bandwidth from the current implementation of the CFR station chip.

The relevance of this work to the MSN architecture is that it demonstrates the possibility of performing MSNL cell forwarding for ATM networks in hardware. The CFR MAC bridge deals with the traditional, source / destination, addressing mechanism, and performs no address translation or cell modification as the cell traverses the bridge. The bridge provides fast access, by the use of hardware, to the two CFR stations chips which implement the media access on the two sides of the bridge, integrated buffering, and hardware support for the flow control.

In considering using this hardware for an MSNL gateway, there are two additional requirements from the hardware:

- to handle CFR destination addresses as VPIs (virtual path identifiers),
- to translate VPIs on cell forwarding.

The first requirement is to support the latest version of MSDL for the CFR and could be solved by rebuilding certain parts of the hardware. This would in fact simplify the design, as flow control need only be performed on a virtual circuit identifier (16 bits), as opposed to a source / destination address pair (32 bits).

A more direct solution to both requirements would be to allow manipulation of the cells as they pass through the bridge. The mechanism to do this is not available due only to an implementation feature of the current hardware, and not due to any design constraint.

7.1.2 MSNL router v MAC bridge

The discussion of the CFR MAC layer bridge is aimed at demonstrating the similarity in hardware implementation of an MSNL router and a MAC layer bridge. Using similar hardware technology there is no reason that MSNL routers should not provide the same throughput as MAC bridges. That such an implementation is possible for the MSN architecture, and not, for example, IP, is due to the nature of the MSNL and MSDL levels of the MSN architecture, in particular, the no multiplexing principle.

Of the arguments for the preference of MAC layer bridges over routers for extending a local network, transparency, performance, and administrative load, only the latter is still an issue when using the MSN architecture and MSNL routers to expand a network.

7.2 Autonet

The DEC System Research Center (SRC) Autonet uses fast packet switching technology, configurable in an arbitrary mesh to provide a local area network. Fast packet switching has previously been presented as one of the possible implementation mechanisms to provide an ATM network (section 2.4.4), and underlying the Autonet are the basic mechanisms for a ATM style network. As shall be shown, it is the design of the higher levels of protocol which drastically alter its characteristics.

In the Autonet, each switch is input buffered, and flow control is applied on each link, both between switches and end nodes. The maximum packet size supported by the network is 16K octets, and this large size implies that, while in transit, the packet can be *spread* over several switches simultaneously, locking down the intervening links. Interconnecting the switches in a general topology, together with the locking down off links due to *long* packets, leads to the possibility of deadlock. The Autonet employs deadlock avoidance so as not to restrict the physical topology of the network and to avoid highly unpredictable effects due to deadlock recovery. Avoidance is achieved by the imposition of a dynamically built spanning tree, such that packets may be routed only in a sequence of *up* hops followed by a sequence of *down* hops.

The network can be used in an Ethernet *service* compatible manner; that is 48 bit addressing, and a broadcast mode, although the broadcast packet is restricted to standard Ethernet size (1536 octets). The algorithm to support Ethernet broadcast also uses the spanning tree to perform a tree broadcast. The *up then down hops* deadlock avoidance mechanism previously used, does not work for the broadcast packets, and so the method used in this instance is to insist that each input controller to a switch must be capable of buffering a whole Ethernet sized packet.

Each packet contains a 16 bit address which is split into two 8 bit fields, indicating the switch node and the output port at that switch. The small size of these labels can then be used to perform routing of the packets by direct lookup in hardware.

Many aspects of Autonet are similar to the Fairisle network based on the Cambridge fast packet switch: packet routing in hardware, out of band software to manage routing tables, and a low level flow control mechanism. However, the complete architecture of the Autonet presents a number of problems when considering use as an multi-service internetwork:

- the fifo input buffering causes *head of the line block*, i.e. if the packet at the head of the line cannot currently be routed, all other packets after it are also held up,
- the large packet size means that very large delay and jitter can be introduced at the switches,
- the spanning tree mechanism reduces total system bandwidth, and can lead to bottlenecks,
- no support for priority, or different traffic types,
- no address mapping at switches, so an internetworking mechanism cannot make use of hardware switches.

Head of the line block is caused by use of a fifo as the input buffer, causing a packet at the front of the buffer, which cannot be transmitted due to a transiently blocked switch component or inter-switch link, to hold back other packets in the buffer even though they could be transmitted. Many of the other aspects are related to the fact that the Autonet has a traditional PTM network architecture, and as such, experiences problems when dealing with multi-service traffic.

7.3 DQDB

IEEE working group 802.6 has prepared a draft standard for the physical and media access levels of the DQDB network (section 2.4.3 briefly discusses the access mechanism and cell format in terms of an ATM network), and a segmentation and reassembly function to provide the LLC service over DQDB. Currently only the connectionless LLC service is addressed. However the design choices for the segmentation and reassembly function can be compared to those of the MSN architecture.

The segmentation and reassembly function uses a *message*, or block, identifier to relate cells belonging to the same packet, and a total length field to protect against cell loss. These mechanisms do not protect against one cell loss and another cell duplication within a block, which requires either a sequence number per cell or a block level CRC. Within a single network, this form of error may be an acceptably rare occurence.

A number of aspects of the mechanism become important only when considering interconnecting DQDB networks, or the use of DQDB as a metropolitan area backbone. The mechanism of single cell forwarding, or *cut-through*, implemented by MSNL level can no longer be used, and even MAC layer bridges between DQDB networks are forced to implement store and forward.

7.4 XTP

The Xpress transfer protocol (XTP) is an architecture designed with future high speed networks in mind [Chesson 89]. Even though the XTP architecture is designed with for packet transfer mode networks, and so does not deal with problems specifically related to multi-service traffic, the architecture still demonstrates a number of features similar to the MSN architecture:

- implementation of the protocol engine in hardware is considered mandatory,
- hosts interact with routers to obtain a *route* tag (virtual path identifiers) for all packets to a given destination, i.e. XTP uses virtual circuits when internetworking,
- dynamic bandwidth allocation by routers.

XTP does not extend these mechanisms to all network components in the same way that the MSN architecture does. The benefits obtained from route tags are not extended into the host, so that although routers benefit from the virtual circuit model, host interfaces must still multiplex and demultiplex each packet. Similarly, the bandwidth allocation scheme relates only to routers, while local area networks and network interfaces are perhaps assumed to have infinite bandwidth.

One feature of XTP is its novel transport protocol. This uses a technique in which the transmitter interrogates the receiver by setting a control field in one of the transmitted packets. This requests a *dump* of the receivers state, including error map, allocation window etc. A transmitter can then use this information to perform retransmissions or adjust parameters such as the flow rate.

7.5 The VMP network adaptor board

The VMP network adaptor board [Kanakia 88] is a design of a high performance network interface to support the versatile message transfer protocol (VMTP) for the VMP multiprocessor. This aims to offload the common case processing of packets from the main host processor. The adaptor is an intelligent network interface consisting of a controlling processor with pipelined hardware implementing the main internal data path between the network and the host receive and transmit buffers. For short messages, a programmed IO interface is provided, which minimizes delay, while large messages use a DMA interface to minimize cache and system bus traffic.

The adaptor itself fully implements a datalink level service. However, it also provides support for the VMTP transport level service. While receiving, the adaptor notices out of sequence and lost packets, and signals the main host processor. The main processor handles the organization of retransmissions, and this overhead is considered acceptable, as in the LAN environment for which the adaptor is designed, this occurrence is rare.

The adaptor uses similar overall hardware implementation techniques to those suggested for the MSN architecture. However, as it is designed with PTM networks in mind, it is of little use in a multi-service environment. One particular aspect of interest is the hardware signalling of out of order and dropped packets, which provides support for the transport level service. In the MSN architecture, this function could similarly be implemented in the MSSAR hardware.

7.6 Summary

Media access layer bridging offers a high performance mechanism for interconnecting a small number of networks within a single administrative and security domain. The requirement to be transparent, means that such bridges are restricted in the performance guarantees they can provide. The MSN architecture allows almost identical hardware to be used as MSNL routers and hence provides similar performance. As MSNL routers, these nodes are then active in association set up, and so can supply bandwidth guarantees to traffic. This can also form the basis of a network load sharing mechanism.

A brief discussion of the Autonet is presented. Although this network uses fast packet switching techniques, the higher levels of the network architecture introduce delay and jitter characteristics which mean that the Autonet is unsuitable for multi-service traffic. Similarly, the use of DQDB as an integrated multi-service network is limited by the currently defined link level protocols, even though at heart DQDB is an ATM network.

XTP uses some of the techniques used in the MSN architecture to provide faster and more reliable routing; this is achieved by the use of virtual circuits. Off-loading protocol implementation from the host system is considered important, and hardware implementation in the network controller is considered necessary.

VMP is the design of a network interface controller implementing many of its functions in specialized hardware to achieve low delay and high throughput. This also addresses the issues of host interfacing.

Chapter 8

Further work

Within DARPA, the Gigabit Working Group has been addressing a wide range of issues concerning high bandwidth networking [Leiner 88]. The MSN architecture addresses certain of these issues, while no mention has been made of others. Some of these issues are discussed here as they form the basis of future research. Similarly, certain aspects of the MSN architecture have not been investigated fully due to time and resource constraints, and are to be the subject of future work.

8.1 Unaddressed issues

8.1.1 Resource management

Most networks currently provide one level of service, and little attempt is made to allocate resources effectively to meet the range of application requirements. The MSN architecture provides support for a range of different service classes, and uses both the service class, and the current bandwidth allocation state, as input to the routing algorithm. Although the mechanism exists, only very simplistic control algorithms are currently in place. The same problem arises even in a local area network based on fast packet switching; to use such networks effectively at high utilization, load sharing between the switches becomes critical. Further work within the Fairisle project is aimed at devising effective algorithms for such resource management functions.

8.1.2 Adaptive protocols

Section 2.4 discusses a number of different implementation mechanisms for ATM networks. Individually, each of these networks exhibits slightly different error characteristics, particularly under saturation conditions, and when interconnected, their operation becomes even more complicated. For example, sudden changes in the amount of real time traffic with reserved bandwidth in the network, would radically effect the response of the network for data traffic.

Adaptive protocols are required, which can modify their internal algorithms (e.g. change timeouts, encodings, etc.) to adjust to the different characteristics that different instances of communication may exhibit, even when between the same end systems. Work on adaptive algorithms for TCP [Jacobson 88] has provided interesting performance improvements, and some work has been performed for multi-service traffic types [Ghanbari 89]. To execute intelligent adaptive algorithms requires a range of new services from a multi-service architecture. For example, accurate error and utilization statistics, and support for dynamic reconfiguration of circuits.

8.1.3 Multicast

Certain models of distributed computing (e.g. ISIS [Birman 87]), and video and voice conferencing, can benefit greatly from a multicast service. A sensibly implemented distributed multicast service can significantly reduce the total network bandwidth used by an application by ensuring that data is only transferred between two sites once, even when there are multiple recipients at the receiving site. Work in this area has been performed within the Unison project [Shrimpton 87], and providing a similar service within the MSN architecture should not be difficult.

To implement multicast, the MSNL addressing scheme needs to be extended to handle multicast addresses, and similarly the liaison set up mechanism needs modification. Multicast in a multi-service network must continue to multiplex at the level of the MSDL PDU, so an efficient MSDL PDU replication mechanism is required within the MAC level or within routers. The implementation of such a mechanism in software is straightforward. However, if the router is implemented in hardware, so must the replication mechanism. [Newman 89] suggests the insertion of a destination delete slotted ring between the input controllers and switch fabric; simulation studies are currently under way.

8.2 Implementation issues

8.2.1 Hardware router

The similarity between the hardware required to implement the CFR MAC layer bridge and an MSNL CFR to CFR router is discussed in section 7.1.1. Future versions of this bridge may provide the modifications required to support the MSNL routing functions, and if available, experiments could be performed with this configuration. A more promising candidate for future work in this area is in the connection of the CFR to the Cambridge backbone network (CBN).

8.2.2 Network interface

For the 50 Mb/s CFR used in the experiments, the network interface restricted the attainable point to point performance to about 25% of the theoretical limit. Even though the processor used was rated at 4 MIPS and has a very low interrupt latency. If similar network interface implementation techniques are used for the backbone network, a node would achieve less than 10% of the possible point to point performance and about 2.5% of total available bandwidth. The implementation of hardware assist for protocols for the CBN is considered mandatory [Greaves 89a] and will be the subject of future work. Similar techniques will need to be applied to the network interface for the Fairisle network where single node access rates of 140Mb/s are targeted.

8.2.3 WANDA and ANSA

The WANDA kernel is a multiprocessor kernel for distributed systems research. Within the kernel the MSN architecture has been implemented as the sole communications architecture, and work is under way to implement the ANSA testbench [ANSA 89] in this environment. Further experimentation on the integration of the MSN architecture with a distributed computing mechanism will use the ANSA RPC system over the WANDA kernel.

Chapter 9

Conclusion

This dissertation has presented the design of the MSN network architecture which provides the networking support for real time and distributed computing traffic for multi-media applications.

One of the key features of the design is the drastic simplification of multiplexing within the MSN architecture. This has been performed to prevent *cross-talk* between different traffic types, and to aid in the implementation of both hardware and software components. Only experience will show whether this simplification is a necessity for hardware implementation. The apparent simplicity of implementing layered multiplexing in software is a considerable problem, as its seems to provide additional functionality, while in fact it merely complicates the main data path. The aim in removing multiplexing within the levels implemented in software components is to aid in *efficient* implementation.

The use of fixed size cells and lightweight virtual circuits is presented as a requirement to implement high bandwidth internetwork routers, whether using the techniques of fast packet switching, or some other high speed switching mechanism. The extension of the architecture over traditional packet switched networks is considered important, as these networks represent a valuable installed resource from which users will wish access to the services only available on ATM networks.

In the realm of distributed computing, the results from the preliminary implementation of the MSN architecture have shown that performance of this general architecture can achieve comparable performance to specialist implementations of a particular *favoured* computing model, such as RPC.

Appendix A

Address Mapping Service

The address mapping service (AMS) is a simple service for mapping protocol specific addresses to virtual path identifiers (VPI). It is aimed mainly at at MSNL, although the interface is more general purpose, and could be used by the Internet protocol (IP) in place of the address resolution protocol for implementing IP over an ATM subnet which uses VPIs. For clarity, the discussion deals specifically with the MSNL use of this service.

```
Resolve(
  myhalf:VPI,
  type:  ProtocolType,
  target:ProtocolAddress,
  data:  ProtocolSpecificData
) RETURNS (
  done:  Boolean,
  next:  VPI
) RAISES badprotocol (
  type:  ProtocolType
) RAISES pleasewait (
) RAISES noresponse (
) RAISES nothere (
) RAISES killed (
) AT assoc: PromiscuousVPI;
```

Figure A.1: AMS and MSNL `Resolve` RPC

The AMS maintains tables containing many-to-one mappings of MSNL addresses to VPIs. These VPIs can represent either a single MSNL entity, or another AMS. In the case of an MSNL entity, the VPI is the promiscuous association, and hence liaison, on which the entity is listening for liaison setup requests.

As previously discussed (section 5.3.2), MSNL uses an RPC interaction to establish a liaison, and in fact this is integrated with the AMS. An MSNL entity and an AMS both support the `Resolve` interface (figure A.1), while the AMS servers support two other management functions (figure A.2).

The VPI of at least one local AMS server is either well known, or chosen from a well known set. This is used by an instigating MSNL entity for the first resolve RPC, which is called with the desired MSNL address, and the VPI allocated for one half of the association. The resolve RPC is implemented differently by the AMS and MSNL entities. An MSNL entity returns with done = TRUE, and the VPI returned represents the other half of a successfully established association, and hence liaison. An AMS returns with done = FALSE, and the VPI represents either another AMS or the desired MSNL entity. Returning the VPI for another AMS allows the implementation of an iterative process for interrogating a number of different AMS servers to find the desired MSNL entity, although care must be taken not to build infinite cycles.

```
EnsureThisEntryExists(       (* idempotent insert *)
  promis:PromiscuousVPI;
  type:  ProtocolType,
  test:  ProtocolAddress,
  mask:  ProtocolAddress
) RETURNS (
) RAISES badprotocol (
  type:  ProtocolType
) RAISES tablefull (
  count: Integer
) RAISES wouldclash (
) RAISES noresponse (
) RAISES nothere (
) AT assoc: PromiscuousVPI;

EnsureThisEntryDoesNotExist( (* idempotent remove *)
  promis:PromiscuousVPI;
  type:  ProtocolType,
  test:  ProtocolAddress,
  mask:  ProtocolAddress
) RETURNS (
) RAISES badprotocol (
  type:  ProtocolType
) RAISES noresponse (
) RAISES nothere (
) AT assoc: PromiscuousVPI;
```

Figure A.2: AMS and MSNL management RPCs

AMS management is performed at two levels. When an MSNL entity starts, it registers its MSNL address and related VPI with its local AMS(s) by use of the `EnsureThisEntryExists` RPC. This provides the basic mechanism for MSNL entities on a local network to interact with each other. At a higher level, management software is required to establish the entries for other AMS servers, or to supply the MSNL routing tables.

Bibliography

The pages on which each reference is cited are listed in parentheses after the reference.

[Accetta 86] 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. (66)

[ANSA 89] *The ANSA Reference Manual*. Architecture Projects Management, release 01.00 edition, March 1989. (67, 84)

[Birman 87] KP Birman and TA Joseph. Exploiting Virtual Synchrony in Distributed Systems. *Proc. ACM SIGOPS, Operating Systems Review*, 21(5), November 1987. (66, 83)

[Birrell 84] A Birrell and G Nelson. Implementing RPC. *ACM Transactions on Computer Systems*, 2(1), February 1984. (2, 65)

[Birrell 85] A Birrell. Secure communications using RPC. *ACM Transactions on Computer Systems*, 3(1), February 1985. (69)

[Birrell 87] A Birrell. July 1987. Private commmunication. (69)

[Budrikis 86] ZL Budrikis, JL Hullett, RM Newman, D Economou, FM Fozdar, and RD Jeffery. QPSX: A Queue Packet and Synchronous Circuit Exchange. In *Proc. 8th International Conference on Computer Communication*, 1986. (10, 11, 15, 38)

[Burren 89] JW Burren. Flexible aggregation of channel bandwidth in primary rate ISDN. In *Proc. ACM SIGCOMM*, September 1989. (25)

[Carriero 86] N Carriero and D Gelernter. The S/Net's Linda Kernel. *ACM Trans. on Computer Systems*, 4(2), May 1986. (66)

[Cheriton 86] DR Cheriton. VMTP: A transport protocol for the next generation of communication systems. In *Proc. ACM SIGCOMM*, August 1986. (67, 72)

[Chesson 89] G Chesson. XTP/PE Design Considerations. In *Proc. Protocols for High-Speed Networks, Zurich*, May 1989. (79)

[Clark 85] D Clark. The Structuring of Systems using Upcalls. In *Proc. ACM SIGOPS*, December 1985. (32)

[Clark 87] D Clark, M Lambert, and L Zhang. NETBLT: A High Throughput Transport Protocol. In *Proc. ACM SIGCOMM*, August 1987. (67)

[Clark 88] D Clark. The design philosophy of the DARPA Internet protocols. In *Proc. ACM SIGCOMM*, 1988. (20)

[COSINE 88] W Bauerfeld. *COSINE Specificiation Phase Report 7: Operational Aspects.* Technical Report, Réseaux Associés pour la Recherche Européenne, July 1988. (20)

[Fraser 89] AG Fraser. The Universal Receiver Protocol. In *Proc. Protocols for High-Speed Networks, Zurich*, May 1989. (24)

[Ghanbari 89] M Ghanbari. Two-Layer Coding of Video Signals for VBR Networks. *IEEE Journal on Selected Areas in Communication*, 7(5), June 1989. (65, 83)

[Greaves 88] DJ Greaves and A Hopper. The Cambridge Backbone Network. In *Proc. European Fibre Optic Conference (EFOC / LAN 88), Amsterdam*, June 1988. (14)

[Greaves 89a] DJ Greaves. September 1989. Private commmunication. (84)

[Greaves 89b] DJ Greaves. *Multi-Access Metropolitan Area Networks.* Technical Report, Cambridge University Computer Laboratory, 1989. Ph.D. dissertation. (15)

[Harita 89] BR Harita and IM Leslie. Dynamic Bandwidth Management of Primary Rate ISDN to Support ATM Access. In *Proc. ACM SIGCOMM*, September 1989. (25)

[Hopper 88] A Hopper and RM Needham. The Cambridge Fast Ring Networking System. *IEEE Trans. Computers*, 37(10), October 1988. (13)

[IEEE 8026 89] IEEE 802.6 Working Group. Proposed Standard: DQDB Metropolitan Area Network (draft). IEEE, August 89. (15)

[ISO 74983] International Standard Organization - Information Processing. OSI Reference Model - Part3: Naming and Addressing. International Standard 7498-3, ISO. (6)

[ISO 8073 84] International Standard Organization - Information Processing. Transport Protocol Specification. International Standard 8073, ISO, September 1984. (20, 67)

[ISO 8473 86] International Standard Organization - Information Processing. Protocol for Providing the Connectionaless-Mode Network Service (Internetwork Protocol). International Standard 8473, ISO, 1986. (22)

[ISO 88022] International Standard Organization - Information Processing. Logical Link Control. International Standard 8802-3, ISO. (20)

[ISO 8824] International Standard Organization - Information Processing. Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). International Standard 8825, ISO. (34)

[Jacobson 88] V Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM*, August 1988. (83)

[Kanakia 88] H Kanakia and D Cheriton. The VMP Network Adapter Board (NAB): High-Performance Network Communication for Multiprocessors. In *Proc. ACM SIGCOMM*, August 1988. (80)

[Kent 87] CA Kent and JC Mogul. *Fragmentation Considered Harmful.* Technical Report 87/3, Digital Western Research Laboratory, December 1987. (23, 50)

[Leiner 88] BM Leiner. Critical issues in high bandwidth networking. RFC-1077, November 1988. (82)

[Leslie 83] IM Leslie. *Extending the Local Area Network.* Technical Report 43, Cambridge University Computer Laboratory, 1983. Ph.D. dissertation. (30)

[McAuley 87] DR McAuley. *Unity: An RPC Mechanism.* Cambridge University Computer Laboratory, 1987. Project Unison working paper UC027. (3, 70)

[Needham 82] RM Needham and AJ Herbert. *The Cambridge Distributed Computer System.* Addison-Wesley, 1982. (23, 45)

[Nelson 81] B Nelson. *Remote Procedure Call.* Technical Report CMU-CS-81-119, Carnegie Mellon University, 1981. Ph.D. dissertation. (66)

[Newman 86] RM Newman and JL Hullett. Distributed Queueing: A Fast and Efficient Packet Access Protocol for QPSX. In *Proc. 8th International Conference on Computer Communication*, 1986. (16)

[Newman 88] P Newman. A Fast Packet Switch for the Integrated Services Backbone Network. *IEEE Journal on Selected Areas in Communication*, 6(9), Decmeber 1988. (17)

[Newman 89] P Newman. *Fast Packet Switching for Integrated Services*. Technical Report 165, Cambridge University Computer Laboratory, 1989. Ph.D. dissertation. (83)

[Nicolaou 90] C Nicolaou. An Architecture for Real-Time Multi-media Communication Systems. *Submitted to IEEE Journal on Selected Areas in Communication*, 1990. (35)

[Nordmark 89] E Nordmark and DR Chertion. Experiences from VMTP: How to achieve low response time. In *Proc. Protocols for High-Speed Networks, Zurich*, May 1989. (3)

[Padlipsky 82] MA Padlipsky. Critique of X.25. RFC-874, September 1982. (19)

[Porter 89] JD Porter. *MAC layer bridging of homogeneous Local Area Networks*. Technical Report, Cambridge University Computer Laboratory, 1989. Ph.D. dissertation in preparation. (14, 75)

[Postel 81] JB Postel. Transmission Control Protocol. RFC-793, September 1981. (67)

[Renesse 88] R van Renesse, H van Staveren, and AS Tanenbaum. Performance of the World's Fastest Distributed Operating System. *ACM SIGOPS Operating Systems Review*, 22(4), October 1988. (33, 72)

[Ross 86] FE Ross. FDDI - a tutorial. *IEEE Communications Magazine*, 24(5), May 1986. (11)

[Saltzer 84] JH Saltzer, DP Reed, and D Clark. End-to-End Arguments in System Design. *ACM Trans. on Computer Systems*, 2(4), November 1984. (20)

[Schroeder 89] M Schroeder and M Burrows. *Performance of the Firefly RPC*. Technical Report 43, DEC Systems Research Center, April 1989. (33, 72)

[Shrimpton 87] DH Shrimpton. *A Multicast service*. Rutherford Appleton Laboratory, 1987. Project Unison working paper UR043/48. (83)

[SUN 85] *External Data Representation Protocol Specification.* Sun Microsystems Inc., release 2.00 edition, May 1985. (34)

[Temple 84] S Temple. *The design of a ring communication network.* Technical Report 52, Cambridge University Computer Laboratory, 1984. Ph.D. dissertation. (13)

[Tennenhouse 88] DL Tennenhouse. *Site Interconnection and the Exchange Architecture.* Technical Report, Cambridge University Computer Laboratory, 1988. Ph.D. dissertation. (25)

[Treese 88] GW Treese. Berkeley UNIX on 1000 Workstations: Athena Changes to 4.3BSD. In *Proc. Usenix Winter Conference*, February 1988. (75)

[Verbiest 89] W Verbiest and L Pinnoo. A Variable Bit Rate Video Codec for Asynchronous Transfer Mode Networks. *IEEE Journal on Selected Areas in Communication*, 7(5), June 1989. (65)

[Wada 89] M Wada. Selective Recovery of Video Packet Loss Using Error Concealment. *IEEE Journal on Selected Areas in Communication*, 7(5), June 1989. (65)

[Wilkes 79] MV Wilkes and DJ Wheeler. The Cambridge digital communications ring. In *Proc. Local-Area Communications-Network Symposium*, May 1979. (10)

[YBTS 80] Study Group Three. *A network independent transport service.* Technical Report, The Post Office PSS User Forum, February 1980. SG3/CP(80)2. (20)