Multicast in the Asynchronous Transfer Mode Environment

John **Matthew** Simon Doar

St. John's College

University of Cambridge

A dissertation submitted for the degree of Doctor of Philosophy

January 1993

Abstract

In future multimedia communication networks, the ability to multicast information will be useful for many new and existing services. This dissertation considers the design of multicast switches for Asynchronous Transfer Mode (ATM) networks and proposes one design based upon a slotted ring. Analysis and simulation studies of this design are presented and details of its implementation for an experimental ATM network (Project Fairisle) are described, together with the modifications to the existing multi-service protocol architecture necessary to provide multicast connections. Finally, a short study of the problem of multicast routing is presented, together with some simulations of the long-term effect upon the routing efficiency of modifying the number of destinations within a multicast group.

Preface

I would like to thank many people, most of whom have been associated with the Laboratory in some manner, for their encouragement, advice and help during the course of my research. My supervisor Ian Leslie, Noha Adly, Nigel Bean, Xiaoqiang Chen, Shaw Chuang, Simon Crosby, the inimitable David Greaves, Bhaskar Harita, Xiaofeng Jiang, Derek McAuley (Mac), Glenford Mapp, Roger Needham, Peter Newman and the denizens of the Trojan Room have all played their part, as have many others.

I am also indebted to the following people who proofread and commented upon all or parts of this thesis: Richard Black, Shaw Chuang, Simon Crosby, John Doar, Daniel Gordon, David Greaves, Narmhee Lee, Ian Leslie and Peter Newman.

My thanks are also due to Piete Brooks, Chris Hadley, Martyn Johnson and Graham Titmus for the day-to-day provision of a productive environment in which to study. Lewis Tiffany was ever helpful in my searches for relevant literature.

I would like to make a special note of thanks to my family, friends and the Backroom Boys for the necessary distractions from my work and to give thanks for God's grace and strength when I have had little enough of my own. Steve Shaw showed me other ways to look at all sorts of problems.

Throughout my research I have been supported by a grant from the Science and Engineering Research Council for which I am grateful. I am also indebted to the Master and Fellows of St. John's College, Cambridge for supplementary funding, both as a member of the Foundation and during the final months of writing.

"May the favour of the Lord our God rest upon us;
establish the work of our hands for us –
yes, establish the work of our hands."
Psalm 90, v.17.

Except where otherwise stated in the text, this dissertation is the result of my own work and is not the outcome of any work done in collaboration. Furthermore, this dissertation is not substantially the same as any that I have submitted for a degree, diploma or any other qualification at any other university. No part of this dissertation has already been or is being concurrently submitted for any such degree, diploma or any other qualification.

Publications

1. *A Slotted Ring Copy Fabric for a Multicast Fast Packet Switch*, P. Newman and M. Doar, Proceedings of International Switching Symposium, Stockholm, 1990.

2. *Multicast in ATM Networks*, M. Doar and N.G. Bean, Proceedings of the Thirtieth Annual Allerton Conference, University of Illinois at Urbana-Champaign, September 1992.

3. *Design and Performance of a Slotted Ring Multicast Fabric*, M. Doar and N.G. Bean, Proceedings of 8th U.K. Performance Engineering Workshop, Imperial College, London, September 1992.

4. *How Bad is Naïve Multicast Routing?*, M. Doar and I.M. Leslie, Proceedings of Infocom 1993, San Francisco.

5. *Analysis of a Cyclic Server with Irregular Server Movements: A Model of an ATM Multicast Fabric*, N.G. Bean and M. Doar, in preparation.

Trademarks Ethernet is a trademark of the Xerox Corporation.
Xilinx is a trademark of the Xilinx Corporation.

Contents

List of Figures

List of Tables

Glossary of Terms

AAL: Asynchronous transfer mode Adaption Layer — highest layer of the ATM Layered Model. (p.5)

ANSI: American National Standards Institute, a member of ISO. (p.1)

ASIC: Application Specific Integrated Circuit. (p.27)

ATM: Asynchronous Transfer Mode. (p.2)

B-ISDN: Broadband ISDN. (p.1)

CATV: Community Antenna TeleVision, i.e. cable TV. (p.9)

CAC: Call Acceptance Control. (p.18)

CAM: Computer Aided Manufacture, the flexible control of machines by computer. (p.11)

CMOS: Complementary Metal Oxide Semiconductor — An IC implementation technology. (p. 58)

CCITT: The International Telegraph and Telephone Consultative Committee (Comité Consultatif International de Télégraphique et Téléphonique). (p.1)

CIP: Cell Insertion Probability — probability that a cell will arrive at the wrong destination. (p.35)

CLB: Configurable Logic Block — basic element in Xilinx FPGAs. (p.89)

CLP: Cell Loss Priority. (p.36)

CLPb: Cell Loss Probability — probability that a cell will be dropped. (p.35)

CRC: Cyclic Redundancy Check — a set of error detection codes. (p.3)

ETSI: European Telecommunications Standards Institute, a member of ISO. (p.1)

Fanout: The degree of replication in a multicast tree or the number of copies of a cell at a switch. (p.11)

FDDI: Fiber Distributed Data Interface — A high speed local area network. (p.82)

FIFO: First In First Out — A queueing discipline and any component which implements it. (p.46)

FPGA: Field Programmable Gate Array — a device which can be configured many times to implement many different circuits. Also known as a logic cell array. (p.81)

GPS: Global Positioning System — U.S. satellite based system for locating any point on earth to within nanoseconds and metres in time and space. (p.17)

HDTV: High Definition TeleVision — 1920×1035 pixel resolution (American) compared to 768×576 for ordinary UK television. (p.1)

HEC: Header Error Check — error detection/correction on a cell header. (p.3)

HOL: Head Of Line — head of a queueing system. (p.53)

IC: Integrated Circuit. (p.12)

IOB: Input/Output Buffer — interface to external signals for Xilinx FPGAs. (p.100)

ISDN: Integrated Services Digital Network, Broadband (B-ISDN) and narrowband (N-ISDN). (p.1)

ISO: International Standards Organisation, also International Organisation for Standardisation. (p.2)

LAN: Local Area Network — a communications network spanning a few buildings. (p.2)

MAN: Metropolitan Area Network — a communications network covering the equivalent area of a town. (p.6)

MID: Multiplexing IDentifier — an identifier to say from which source information came. (p.20)

MIN: Multistage Interconnection Network — a network comprising smaller interconnection elements linked in a regular manner. (p.37)

MSNA: Multi-Service Network Architecture — a protocol suite designed for multi-service networks. (p.107)

MVCI: Multicast Virtual Channel Identifier — the VCI for a multicast connection. (p.108)

NNI: Network-Network Interface — internal interface within B-ISDN. (p.3)

OAM: Operations And Maintenance — gathering and reporting of network statistics. (p.3)

OSI: Open Systems Interconnection. (p.2)

PABX: Private Automatic Branch eXchange — a private telephone exchange. (p.24)

PCB: Printed Circuit Board — base for ICs and other circuit components. (p.91)

PDU: Protocol Data Unit — a unit of information in a communications protocol, together with the protocol overhead. (p.5)

QOS: Quality of Service — the standard demanded of a transmission channel in a network. (p.18)

RFC: Request For Comments — an Internet proposal or definition document. (p.25)

SAP: Service Access Point — uniquely identifiable points where a higher level can access a protocol level. (p.118)

SAR: Segmentation and Reassembly — the breaking up of packets into smaller data units and their reassembly again. (p.114)

SDH: Synchronous Digital Hierarchy — a CCITT defined set of data rates and switching system which allows payloads to be added and extracted without unpacking the entire data structure. (p.2)

SDU: Service Data Unit — a unit of information in a communications protocol, without the protocol overhead of that layer. (p.112)

SMDS: Switched Multi-megabit Data Service — a connectionless MAN internetworking scheme of the early 1990's. (p.6)

TAXI: Transparent Asynchronous Xmitter-receiver Interface — *de facto* interface for 100Mb/s ATM networks. Also ICs which implement the interface. (p.82)

TDM: Time Division Multiplexed. (p.24)

UNI: User-Network Interface — interface from within B-ISDN to an external device. (p.3)

VCI: Virtual Channel Identifier — a connection identifier in an ATM network. (p.3)

VPI: Virtual Path Identifier — an identifier for a group of connections in an ATM network. (p.3)

WAN: Wide Area Network – a communications network between cities and countries. (p.1)

1 Introduction

For many years the design of different telecommunications networks has reflected the specific characteristics of the classes of traffic which they were designed to transport. Thus the global telephone network is based around circuit switched connections, optimised for traffic which has real-time constraints, such as voice. Created for a different class of traffic, the packet switched networks which emerged in the late 1960's were designed to transport data reliably between computers, whilst efficiently sharing the available network bandwidth.

Recent increases of the bandwidth available on a single fibre towards Gbit/s have meant that many media such as video and high quality audio, which previously would have used most of a small network's bandwidth, can now be considered as realistic traffic. This had lead to a rapid growth in interest in multimedia systems, where audio, video and data streams are all used in a coordinated fashion to communicate with the user. Examples of such streams of traffic are voice, high quality audio, broadcast quality video, high definition television (HDTV), compressed video, images, text and facsimile of various grades of quality, as well as file transfer data and client/server data.

To use the enormous potential of multimedia systems efficiently, new communications networks have to be developed to satisfy the demands of all the classes of traffic. Such networks should also be relatively immune to future changes in traffic characteristics. For example, they should be independent of any particular data rate from a source.

Traditional circuit switched networks do not share the bandwidth efficiently and often provide the end user with only a rather limited rate of data transfer. Packet switched networks can introduce excessive delays or variance of the delay (jitter) into a traffic stream. The Wide Area Networks (WAN) which are being designed to solve these problems and support multimedia traffic are referred to as Broadband Integrated Services Digital Networks (B-ISDN). Standards are still evolving under the auspices of the CCITT, ETSI and ANSI and commercially available products and services are expected to appear in the mid 1990's and early twenty-first century.

To provide a separate network for each medium would be prohibitively expensive in the long-term due to multiplied access costs, cabling costs and the costs of modifications necessary by changes in the nature of a class of traffic. The decision has been made to use a single network to transport all of the media. The transfer mode which has been chosen as the underlying transfer mode for this network is called Asynchronous Transfer Mode (ATM).

1.1 Asynchronous Transfer Mode

ATM is defined by CCITT as

> *"A transfer mode in which information is organised into cells; it is asynchronous in the sense that the recurrence of cells containing information from a particular user is not necessarily periodic."*
> (from CCITT Recommendation I.113, Section 2.2 [CCITT I.113])

ATM is a transfer mode, just as packet or circuit switching are the general transfer modes for many different LANs and WANs respectively. The choice of ATM as the transfer mode to support the CCITT definition of B-ISDN means that a specific set of protocols using ATM are being developed, which are described in further detail below. These protocols should not be taken to define ATM in general, merely the CCITT version as it applies to B-ISDN.

1.1.1 The ATM Layered Model

The B-ISDN model for ATM is shown in Figure 1.1. Using the ISO OSI 7-layer model [OSI83] as a theoretical guide to the functions expected in each layer, the ATM layered model corresponds approximately to the bottom three layers of the OSI model.

ATM Adaption Layer	Convergence
	Segmentation and Reassembly
ATM Layer	Generic Flow Control
	Cell header generation/extraction
	VPI/VCI remapping
	Cell multiplex/demultiplex
Physical Layer	Transmission Convergence
	Physical Medium

Figure 1.1: B-ISDN ATM Protocol Reference Model

1.1.2 Physical Layer

The physical medium sublayer refers to the actual medium (copper, fibre, etc), line coding and bit timing for correct transmission and reception. Bit rates of 155.52Mbit/s and 622.08Mbit/s are recommended, using STM-1 and STM-3 from the CCITT Synchronous Digital Hierarchy (SDH) of transmission rates [CCITT G.707/8/9].

The transmission convergence sublayer is responsible for the generation and recovery of the transmission frames, recognising the boundaries between cells and generating and verifying the Header Error Check (HEC), which is a method of detecting corruption of the cell header bits. It also decouples the cell rate between transmitter and receiver by inserting unassigned cells.

1.1.3 ATM Layer

All traffic is sent between nodes (or switches) in a network in small, fixed length cells, containing 48 bytes of data and 5 bytes of header information per cell (see Figure 1.2). These cells are generated by sources as they are required, after connections have been set up in all the switches from the sources to the destinations.

The purpose of the ATM layer is to multiplex and demultiplex cells from different connections onto a single cell stream, remap the connection identifier (VCI/VPI) at each switch and possibly to implement a flow control mechanism at the User Network Interface (UNI) using the Generic Flow Control (GFC) bits in each cell header.

Figure 1.2 shows the structure for ATM cells at two points in their route from source to destination. The upper diagram shows a cell at the interface between the user and the network (UNI), the lower shows a cell at the interface between two switches within the network (NNI).

The fields within each ATM cell are defined as follows:

GFC 4 bits. Generic Flow Control for use at the user network interface.

VPI 8 or 12 bits for the Virtual Path Identifier , which can be used for less transient network switching, e.g. re-routing a large number of connections in the event of a router failure.

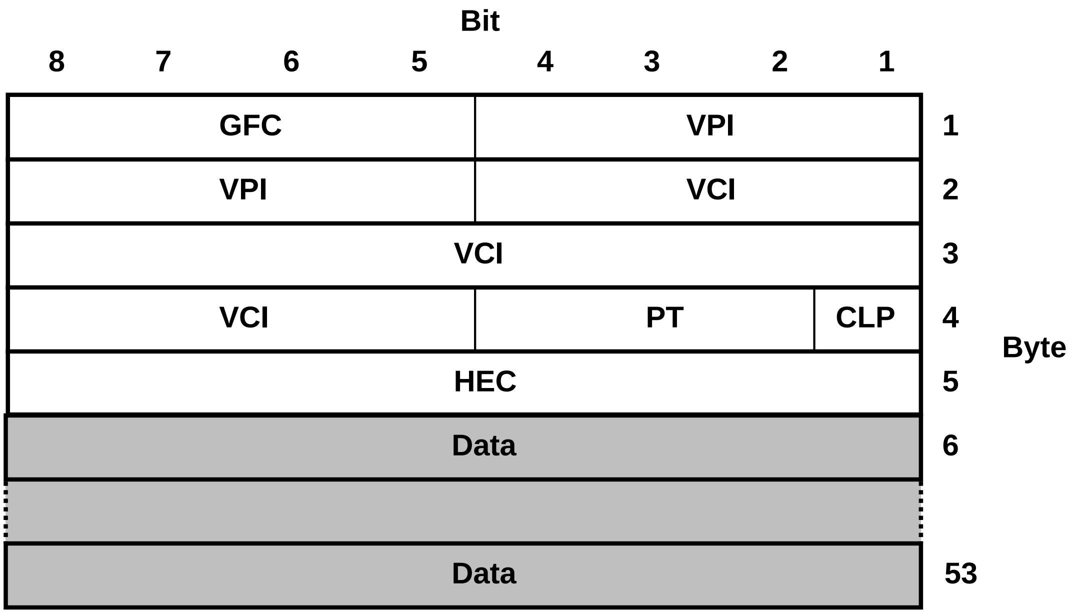
VCI 16 bits for the Virtual Channel Identifier[1], used for identifying individual channels within a Virtual Path.

PT 3 bits for the Payload Type. Indicates whether the cell contains user data or network operation and maintenance information (OAM), whether the cell is at the end of a higher level unit such as a frame and also the state of congestion of the network.

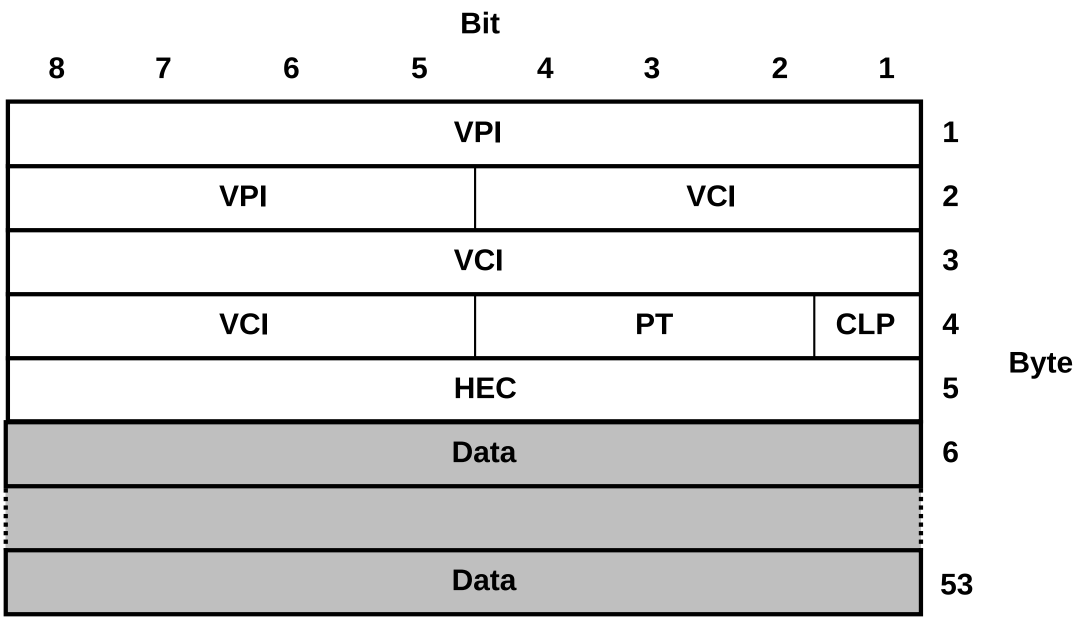
CLP Cell Loss Priority. Cells from sources which are relatively insensitive to loss could have this bit set to indicate that they should be discarded preferentially by congested switches. For instance, cells carrying video data might be discarded in preference to cells carrying the crucial timing data. Cells of a connection which has exceeded its permitted data rate might also be tagged in the same manner.

HEC 8 bits for a Header Error Check, usually some form of Cyclic Redundancy Check (CRC) for error detection.

[1] Also known informally as the Virtual *Circuit* Identifier.



ATM Cell Structure at User Network Interface



ATM Cell Structure at Network Network Interface

Figure 1.2: CCITT ATM Cell Structure

Bit Stream Oriented		**Packet Oriented**	
Constant Bit Rate	**Variable Bit Rate**		
Connection Oriented			**Connectionless**
Class A	**Class B**	**Class C**	**Class D**

Figure 1.3: AAL Service Classifications

The VCI/VPI can be considered as 24 or 28 bits in a single address space, which may be broken down in many different ways. It is likely to be used to provide a hierarchy of switching within different ATM networks. Throughout this work the term 'VCI' is used to refer to the VCI/VPI bits where no confusion would result. 'Byte' and 'octet' have also been used interchangeably throughout.

1.1.4 ATM Adaption Layer

The ATM Adaption Layer (AAL) deals with the segmentation and reassembly of the protocol data units (PDUs) of the different kinds of data from higher layers into ATM cells. Since this operation is dependent upon the characteristics of the individual services being carried over the ATM layer, four basic classes of traffic were originally proposed, A, B, C and D, with four respective AAL types (see Figure 1.3). Since then type 2 has come to be viewed as an extension of type 1, types 3 and 4 have been merged into a single type and a fifth type has been put forward as a simplification of type 3/4. A null adaption layer (AAL 0) can also be imagined, when applications may wish receive 48 byte blocks of data or even 'raw' cells. The original four types of ATM adaption layers are described below as examples of service types, together with the fifth type, AAL 5.

AAL 1 Time sensitive, constant bit rate, connection-oriented data, e.g. voice. One byte of the data is used to ensure correct reassembly of cells into higher level PDUs. Clock recovery can be explicit time stamps in the data or by implicit methods such as the occupancy of a receiving buffer.

AAL 2 Time sensitive, variable bit rate, connection oriented data, e.g. video which has been compressed by an adaptive algorithm. Sequence numbers and whether the cell is the beginning, middle or end of a message are necessary for later reassembly. The length of useful data in partially filled cells and a CRC to detect incorrect reassembly are also provided.

AAL 3 Insensitive to time, variable bit rate, connection-oriented data, e.g. connection-oriented data services, such as interconnecting WANs. Operates in two modes; 'message', for a single packet, and 'streaming', for a continual series of packets.

AAL 4 Insensitive to time, variable bit rate, connectionless data, e.g. SMDS (interconnection of MANs). As AAL 3, but has to support de-multiplexing of streams of data from one VCI, since just one VPI/VCI pair is used by higher level connectionless servers.

AAL 5 Variable bit rate, connection-oriented, designed for data services, such as file transfer over reliable transmission links. A simplified version of AAL 3/4 without such extensive error recovery and retransmission, enabling more of the data bytes to be used for data. Useful for its simpler processing requirements and smaller bandwidth of its overhead. Currently only defined for connection-oriented circuits.

A further AAL for signalling, the S-AAL, is currently being selected by CCITT. Initial releases will use predefined VCIs and a protocol derived from the one used in N-ISDN. Further details of the ATM layered model may be found in Section 2.10, the various CCITT Recommendations or in recent texts [de Prycker91, de Vries90, Händel91].

1.1.5 ATM Summary

The basic principles of ATM, taken from CCITT Recommendation I.150 [CCITT I.150] are as follows:

1. Fixed length cells with information and a header for identifying the virtual channel and routing the cell. Cells arrive at their destination in the same order that they were generated.

2. ATM is connection-oriented. VCIs are identifiers for lightweight connections which are meaningful only a per-switch basis. They are constant for the duration of a connection.

3. No error-control or similar processing is performed upon the cells within the network. That is done end-to-end.

4. All classes of traffic can be transported by ATM networks. Different adaption layers are provided for different classes of traffic.

The essential operations performed by a switch upon each cell passing through it are as follows:

- The incoming VCI/VPI bits are replaced by the ones stored in the remapping table in the switch for that channel.

- The cell is routed to the correct output port of the switch.

- Functions such as congestion control notification, cell-level scheduling and possibly replication of cells for multicast are performed.

ATM may be seen as a compromise between the synchronous transfer mode of circuit switched networks and the packet transfer mode of the packet switched networks. The bandwidth of the network is shared more efficiently since when a source is not transmitting, the only resource which it uses is the space reserved in the VCI/VPI remapping table in each switch, i.e. a few bytes per channel per switch. This assumes that the different classes of traffic can be statistically multiplexed together in a meaningful and controllable fashion. With ATM, the delay and jitter through each switch can be reduced due the minimal functionality required in each switch. The greatly increased line speeds mean that it now makes more sense to do the end-to-end checking necessary for reliable communication at higher levels in the protocol stack, further reducing the required complexity of each switch.

One hope is that future services will be able to be extended with little or no change to switch hardware or the control protocols. This hope is based upon the support of a choice of different protocols for differing services over the same switching hardware and the expectation that the performance characteristics of the hardware will be good enough to support most traffic classes' requirements for latency, jitter and bandwidth. In this sense B-ISDN should be safe against future changes in the nature of the traffic which it will carry.

ATM for LANs is being explored to provide multimedia services just as in the wide area, but with a different set of criteria. For instance, there is a possible trade-off between cost and reliability which is inappropriate in the national networks. To provide easier interconnection between WANs and LANs, ATM protocols are being considered for the local area as well as the wide area. This should hopefully enable users to take a more unified view of communication across networks of different physical scales [Greaves92]. This is felt to be important in order to encourage the growth of interconnected networks and the global use of B-ISDN.

1.2 Multicast

Multicast can be defined loosely as the ability to logically connect a subset of the hosts in a network. The number of sources and destinations can be one each, in which case the connection is referred to as 'point-to-point' or 'unicast', or all the hosts in a network, in which case the connection is referred to as 'broadcast', or they may be any numbers in between.

The meaning of multicast used in this work often assumes a one-to-many sense, generally lower down a protocol stack, due to the nature of existing multicast implementations. Of course multicast is logically a many-to-many connection. This sort of connection can be constructed from multiple 'one to many' multicast connections. The differences are revisited in Chapter 7. Some workers, notably Turner and others [Cox91] at Washington University in St. Louis, use the term 'multipoint' to refer to 'many to many' multicast, but this is felt to be unnecessarily restrictive, since the meaning of multicast is usually clear from its context. Where it is not, it has been explicitly defined as one or the other of its meanings.

'One to many' multicast as used here also assumes a sense of major direction, with most of the traffic flowing from the source to the destinations. A limited amount of traffic may be permitted to flow in the opposite direction to form a convergence multicast or 'concast'. This too is considered further in Chapter 7.

1.2.1 Uses of Multicast

File systems or databases which depend upon distributed resources or data often use multicast to replicate information to a specific subset of sites. Synchronous replication protocols use multicast to request the current status of each copy in order to find the most recent one [Gifford79]. Asynchronous replication protocols use multicast to propagate changes in the data [Birrell82]. In multiprocessor systems, parallel computations often need to exchange intermediate results, involving sending the results to multiple destinations [Strebenz90].

Receiving status reports or any form of telemetry from multiple sensors into a set of hosts is a form of convergence multicast. The destination address embedded in the large number of sensors can be used a redirection address to the current (small) set of destinations. One example of this is Computer Aided Machining (CAM), the remote control of multiple machines, typically on a production line in a factory. Another proposed use is the remote gathering of information from domestic meters for gas, water and electricity.

In addition to the above uses of multicast, another use is as a general level of indirection between a source and its destinations, especially when the destinations may change their names or locations with time. This is particularly attractive for locating such services as name servers on a network or files from which a machine should boot [Deering91].

1.2.2 Recent and Future Uses of Multicast

Any form of conferencing (audio, video, shared data) can benefit from using multicast when the number of destinations is non-trivial, but small compared to the total number of destinations in the network.

Whilst conventional voice conversations alone are not particularly useful with multiple participants[2], mainly due to the lack of coordinating signals, multiple party video telephony can be made to work well and promises great reductions in the costs of travelling for meetings. The bandwidth of video is such that the construction of a network without the provision for multicast, or even worse adding it later as a special case, may severely limit its capacity for use with multimedia. One instance of data conferencing which would benefit from the provision of multicast in a network is the 'virtual whiteboard' application, where all participants can see what is being

[2]Note the small growth in 'chat-lines' compared with the rapid growth of information and entertainment services.

drawn on a screen and may be able to modify the screen themselves.

For all forms of conferencing, an initial aim must be to provide primitives to support the different methods of interaction which occur when physical meetings occur. It must be possible to add members to the multicast group, delete them, establish subgroups within the group, possibly splitting the group into subgroups, and to merge groups together. The last three functions are part of how people work in groups and will be necessary to raise the state of interactive conferencing to level where useful work can be achieved. It is more difficult to see how they might be used regularly in strictly non-interactive applications.

Wide scale distribution of high quality audio (compact disc) and commercial television by digital networks as the next generation of CATV will certainly require efficient multicast, especially since the total number of modifications to a multicast group will be relatively high as subscribers change from channel to channel.

In the immediate future, the provision of multicast in ATM networks will be necessary to provide backward compatability with existing protocol structures. In the local area, ATM networks should be able to fit approximately into the bottom three layers of the ISO protocol stack with no disturbance to higher layers. This means that ATM networks of the near future will have to be able to emulate the broadcast nature of many of today's popular broadcast-based LANs.

An unusual use of multicast would be for the purpose of disguising the intended recipient of a message in an insecure system. An eavesdropper would be unable to determine the active recipient of a sensitive message since the same message would be sent to multiple destinations.

As video and other higher bandwidth services become an increasingly common class of traffic, it seems certain that the percentage of a network's bandwidth devoted to them will become highly significant and that carefully considered instances of multicast will be the only sensible way to support the network's traffic.

One final argument for the provision of multicast in future ATM networks is that it represents a different view of networks from the ones we are used to. As such it will permit applications as yet unthought of to be developed. Many such applications will be in the fields of education and leisure. For instance, 'learning at a distance' with a lecture being transmitted directly to the small group who are interested, rather than broadcast to the whole nation. Another example is multiuser games, which have been popular in academic computer systems for decades. If multicast is not supported, or is added in an *ad hoc* fashion, then we may miss the opportunity to develop such new applications.

1.2.3 Categories of Multicast

Multicast has been described as a logical set of connections, but the use of multicast connections by real applications can be further separated into several very different categories. Multicast connections may be categorised by the number of entities within the communicating group, i.e. the size of the network, the average degree of replication required for each piece of information (since each source may well only transmit to a small fraction of the destinations), the required bandwidth, the burstiness (e.g. peak/mean bandwidth ratio) and the time-dependent nature of the traffic.

In addition, some applications have multicast groups whose membership is relatively static over a period of time, whilst the membership of others is more dynamic. The membership of a multicast group can be chosen by the source(s) of the traffic or by requests from destinations to an entity which is aware of the existence of the different groups. This is often the source of a multicast connection, though not always[3]. Some groups can be modified by either source or destination, but tend to be modified by one or the other. For instance, setting up a video conference may be initiated by one source, but later modified by other destinations.

Other major attributes of multicast connections include the length of time for which a group remains in existence, since some groups may be deemed to exist even after all the current members have been removed. Any use of multicast as an indirection service using a 'well-known' address is an example of this. Another attribute is the timescale of a connection's duration. This affects the amount of space which must be allocated to storage of information about multicast groups, since short-lived groups will permit the rapid re-use of the storage space. Given that control and membership information must be propagated faster than the duration of a connection, this becomes a concern with the economies of memory and connection set-up times. A further attribute is the degree of reliability required from the multicast connection. Guaranteeing fully reliable multicast to all the members of a group is far more difficult than a 'best effort' multicast, since the requests for retransmission are hard to synchronise and there may be many of them. The requirements necessary for reliable multicast are considered further in the appropriate sections of the next chapter and are discussed more fully elsewhere [Birman87].

Table 1.1 shows some of the different categories applied to the multicast applications described in Sections 1.2.1 and 1.2.2. The values for the bandwidth data rates are approximate and derived from a variety of sources [Newman88, DuBose92]. The bandwidth quoted is the bandwidth of a single stream of data, which would be multiplied by the multicast size in a naïve system. The total size is the potential number of hosts in a network, and the multicast size is the mean number of hosts in a multicast group. 'Real-time' is whether the application is particularly sensitive to the delay across the network. The terms 'small', 'medium' and 'large' refer to numbers in the approximate ranges of 1 to 10, 10 to 100 and greater than 100 respectively.

Further categorisation of the methods of multicasting according to the criteria of signalling bandwidth, delay, amount of signalling state and degree of computation required can be found in the paper "Multicast Communication on Network Computers" [Frank85].

[3] A source may be unaware that a boot request, for example, is sent to a group of boot servers for reliability.

Type of Application	Total Size	Multicast Size	Bandwidth (bit/s)	Real-time	Static/ Dynamic	Modification Requested By
File system	Small	Small	10K–1M		Static	Source
Distributed Database	Medium	Small	1–10M		Static	Source
Multiprocessor	Medium	Small	1–100K	$\surd$	Static	Source
Telemetry	Large	Medium	$< 10K$		Static	Destination
CAM Control	Medium	Small	$< 10K$	$\surd$	Static	Destination
Service Location	Large	Large	$< 100K$		Static	Destination
Audio Conference	Medium	Small	4–64K	$\surd$	Dynamic	Either
Video Conference	Medium	Small	2–34M	$\surd$	Dynamic	Either
Whiteboard	Medium	Small	1K–1M	$\surd$	Dynamic	Either
Audio Distribution	Large	Large	1–2M		Dynamic	Destination
Video Distribution	Large	Large	2–32M		Dynamic	Destination
Broadcast Emulation	Large	Large	1–10M		Static	Either

Table 1.1: Characteristics of different applications which use multicast

1.3 Assumptions

Opinions vary concerning the amount of traffic which will be destined for multiple hosts in future networks. Estimates range from 1% to 50% as a percentage of the existing connections in the network, also depending upon whether the traffic is in the wide or local area. As a percentage of the bandwidth of the network, the proportion of multicast traffic is expected to be lower if the replication can be performed in hardware and the bandwidth used by the multicast connections can be consequently reduced.

Informal monitoring of local Ethernets suggests that less than 10% of the current traffic is multicast packets, including broadcast packets. Allowing for increases in the use of multicast for all the reasons given in Sections 1.2.1 and 1.2.2, an upper bound of 20% of a LAN's connections being multicast would seem reasonable. In the wider area, it would be expected that a smaller percentage of the connections would be multicast, since most traffic is inherently local in its nature. An estimate of 5% might be more appropriate in this case. In the work which follows, higher proportions of a network's connections being multicast are considered as potential worst cases, and also since these figures are only estimates.

The number of destinations in a multicast connection, or the *fanout,* is likely to have a bi-modal distribution in reality, as shown in Figure 1.4. This is the fanout of the entire connection, which is made up from the multiplicative effects of smaller fanouts at individual switches. As mentioned above, the great majority of connections will be unicast, so the distribution is heavily biased towards lower fanouts. Most multicast connections with a fanout greater than one will also have

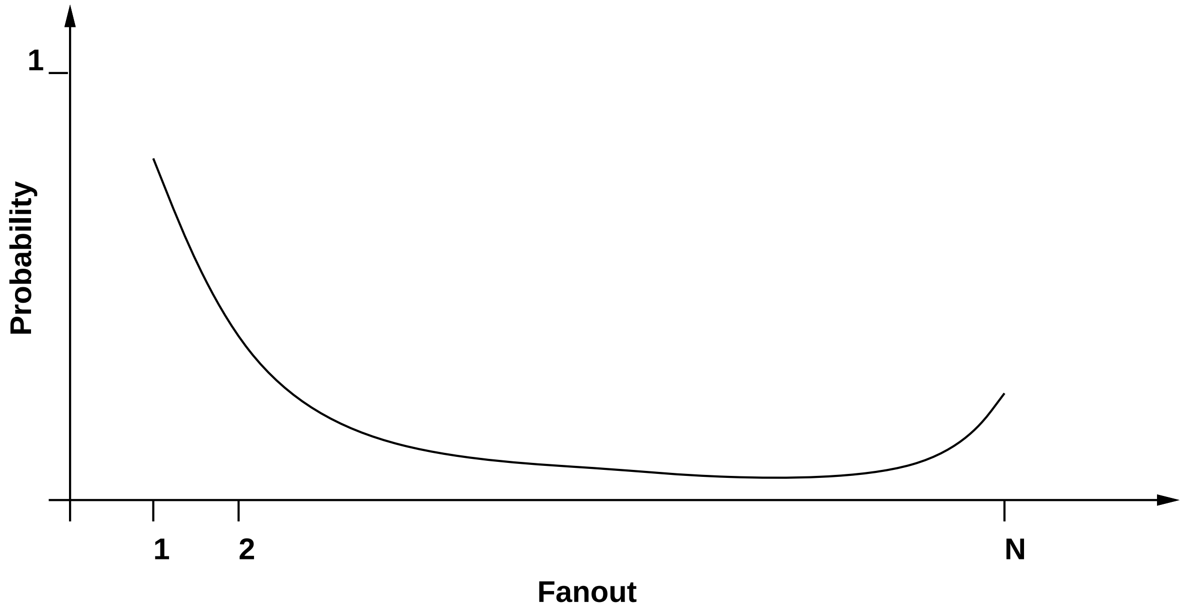


Figure 1.4: Approximate Fanout Probability Distribution

a small fanout since conferences, for example, with large numbers of participants are relatively rare events. In addition, if all the switches in a network provide multicast, then a small initial fanout at one switch can grow exponentially to connect larger multicast groups as required. There will also be a significant peak when broadcasts are performed, with the total fanout equal to N, the number of hosts in the network's domain. In networks based around a single switch the fanout at the switch is equal to the total fanout, whereas in ones which use multiple switches the multiplicative effects of a smaller fanout at each switch can be used to effectively perform a broadcast.

The size of switches, i.e. the number of inputs and outputs, which is needed is dependent upon the intended scale of the area of application. In the local area, switches as small as 4×4 inputs and outputs might be used, each one concentrating the information from a small number of locations onto a larger backbone network. On the other hand, a single star configuration for an entire building might have to cope with over a hundred separate inputs. In the wider area, major exchanges have to handle thousands of inputs each. The single chip switch size which is favoured for current technology is 32×32 [Henrion90, Yum92], with larger switches being constructed from these switch ICs. Other assumptions and any deviations from them are discussed as they arise in each subsequent chapter.

1.4 Outline of Thesis

The subsequent work in this thesis concentrates upon the use of multicast in a general communication network, as opposed to a tightly coupled multiprocessor system connected by a dedicated interconnection network [Byrd87].

Chapter 2 describes the difficulties which the provision of multicast in a network brings to each layer of a combination of protocol stacks and gives details of existing work which provides multicast capability. Chapter 3 describes how multicast ATM switches may be constructed, giving examples from four separate classes for two different categories of switches. The major design issues are outlined and the multicast fabric design, originally proposed by Newman [Newman89], chosen for further investigation is described in detail. Chapter 4 contains the results of simulation studies of the multicast fabric under symmetrical and asymmetrical traffic patterns. Chapter 5 describes some possible enhancements for the basic slotted ring multicast fabric design, together with further simulation results of two alternative methods of controlling access to the ring.

Chapter 6 is an introduction to Project Fairisle and describes the implementation details of the slotted ring multicast fabric and also includes experimental performance results. Chapter 7 gives an overview of the changes to the protocols used in the Fairisle network which would be necessary to support the full provision of low-level multicast. Chapter 8 is a separate discussion of the problems of routing multicast connections across networks and includes some simulation studies to model how the efficiency of a multicast tree changes as destinations are added or removed from the multicast group. Finally, Chapter 9 summarises the insights gained into the problems of providing multicast in future ATM networks, examines some of the reasons why multicast is not currently used more frequently and suggests some possible areas for future study.

Appendix A presents a summary of the analysis of the slotted ring multicast fabric carried out in conjunction with Nigel Bean of the University of Cambridge Statistical Laboratory. Appendix B contains the tabulated results of the simulations in Chapters 4 and 5.

2 Multicasting

The broad difficulties of providing multicast in a network are essentially two-fold. There is the increase in the bandwidth used, simply because the information has to be replicated at some point, whether inside or outside the network. There is also the increase in logical complexity due to the change from having just one destination to multiple destinations. The combination of these two factors makes the provision of multicast a formidable task, requiring modifications at some or all levels of the protocol stack. A combination of the ISO OSI 7 layer protocol reference model [OSI83] and the B-ISDN ATM protocol reference model described in Section 1.1.1 has been used as a basis for describing some of these changes in the sections which follow. This combination cannot be used to precisely characterise all existing networks and the exact relationship between the OSI model and the ATM layered model is still to be defined. The existence of user, control and management planes in the ATM model is described in Section 2.10, but they are not used at higher levels than those for which they are defined. The combination is sufficient, however, to indicate some of the difficulties of multicast. For these reasons, some functions which appear in a single layer may be considered important for other layers as well. This is indicated in each section as necessary.

2.1 Application Layer

Since the X.400 series of CCITT recommendations [CCITT X.400] for Message Handling Systems, i.e. electronic mail, permit messages to be sent to multiple recipients, multicast can sometimes be used to reduce the bandwidth used, especially in the local area. Deciding when a lower level implementation of multicast may reduce the total work to transmit a message is a hard problem to solve optimally. Heuristically, local addresses would be delivered by the efficient low-level multicast service in the well-known local networks and the distant (possibly globally) ones would be sent individually, unless a substantial proportion of them were destined for a site with known multicast capabilities.

The management of multiple connections to provide a reliable service can be performed by the Commitment, Concurrency and Recovery (CCR) service element in the application layer. This provides atomic transactions using a two-phase commit, guaranteeing reliable multicast for substantial numbers of applications. Fortunately, most applications which require reliable multicast are small bandwidth ones, such as financial transaction processing.

2.2 Presentation Layer

Encryption of a multicast connection may require that different destinations should receive the data encrypted by different methods or with different keys. For instance, if some form of public key cryptography were used then each destination might well require a different key in order to exchange signatures. Of course, encryption is not just a presentation layer issue and must be considered at all layers of a protocol.

Similar situations might arise where different 'endian', compression and data encoding schemes might be needed for each destination within a multicast group. For instance, multicasting video streams to different countries may require some form of transcoding to convert the signals to the standard in use in that particular domain. In this case it may be simpler and more efficient to construct the multicast connection from multiple instances of unicast connections, or as a number of multicast connections of a small degree.

2.3 Session Layer

The issue of synchronisation between different sources is the prime difference in the session layer when multicast is considered. Whether the synchronisation is between one member of a multicast group and the others, or whether the group has multiple dependencies, the problem is still an important one. Briefly, the problem of 'causal ordering', as defined for the fault-tolerant distributed computing system ISIS [Birman87], can be described by Figure 2.1. If source A is transmitting to host B, and host B replies to A after receiving A's message, then for some applications, such as conferencing, it may be vital that an observer C should receive the messages in that order. Failure to enforce this ordering could mean people appearing to reply before the other person has stopped speaking, or even replying before the other person begins to speak. Another problematic application is that of electronic mail which contains many such exchanges with little real-time constraint, but which must obey causal ordering in order to be useful.

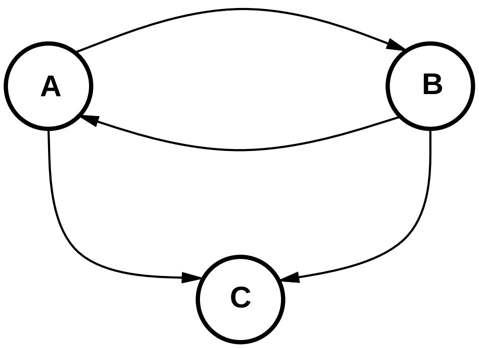


Figure 2.1: Causal Ordering with Multicast

This problem can be avoided by ensuring that the multicast group is connected by a pure tree, that is one with no cycles. This ensures that the ordering of messages is preserved when there is a causal relationship between the messages (such as query/response). Synchronisation of message streams is also an issue for connection management in the transport layer and for routing in the network layer.

For many applications, a sufficient degree of synchronisation can be provided by local clocks derived from time signals broadcast in each country, e.g. Rugby in England, or from the Global Positioning System (GPS) [Chaffee92]. In addition, the Network Time Protocol (NTP) [Mills91b] can be used. NTP can be used to keep "most portions of the Internet ... maintained to within a few tens of milliseconds". Present LAN experience in Cambridge suggests that on a day-to-day basis, the variance of local clocks between machines is less than 100ms and the absolute accuracy is of a similar order, even during loss of the synchronising signal. These figures are estimates derived from intermittent network monitoring over a period of months. These network independent solutions permit timestamps to be used for reordering messages into their original chronological order.

To provide a more general form of synchronisation between sources it is necessary that each destination should be aware of the relative delays between nodes in the network. The particular multicast tree chosen by the routing algorithm becomes less important in this case. If absolute synchronisation is required each time a message is received and global synchronisation is considered to be too variable for the degree of control required, a period of time equal to the longest possible delay within the multicast group must be allowed to elapse before presenting the message to the next higher level. This ensures that all possible messages which could have arisen at the time that the first message was generated have arrived. The order of their presentation to the next level can be adjusted according to what the destination knows about the likely delays through the network. Such strict ordering of messages might be necessary in financial applications, for example, to ensure fairness in bidding.

The problem of insertion of checkpoints into the data stream to permit reliable operation in the face of network failures is also complicated by the difficulties of absolute synchronisation between sources. These checkpoints can be used by the application layer CCR to coordinate multiparty interactions. It has also been suggested that floor control for conference applications should occur in this layer [Waters92]. This is the (possibly automatic) control of which source is broadcast to all participants and which conversations 'on the side' are permitted. This could be under the control of a separate conference manager application or the proposed 'Automated Digital TV Director' [Mullender92].

2.4 Transport Layer

In general, a multicast connection is defined for this layer for a number of sources and a number of destinations. The number of sources and destinations associated with a connection may change over time as members are added to and deleted from the multicast group. In addition, multicast groups can be partitioned or merged, and related, but separate, subgroups of the original multicast groups can be set up.

The difficulties of multicast group management are certainly eased if the connection set-up time is relatively small and multiple connections can be initiated and acknowledged in parallel. For ease of modification and reliability, a multicast connection is often established as a series of individual connections in parallel, rather than attempting to set up a monolithic connection in one attempt [Cox91, Leslie91].

Since the congestion of the whole network is considered at this level, the call acceptance control (CAC) algorithms must take into account the increased load in the network due to multicast groups. CAC is complicated still further by the fact that the members of a multicast group may change with time, so not only must the peak and average bandwidth of the traffic source be taken into account but also the current number of destinations and possibly a maximum number of future destinations. Support must be provided for at least two different kinds of connection set-up at this and higher levels. In one case, if all the destinations cannot be connected, then the whole multicast connection fails. In the other case, the connection is established but to fewer destinations than were originally requested. Multicast connections can be permanently set-up with well-known addresses or can be transient, being set-up and torn down as necessary. The number of destinations in a transient multicast group has to be monitored to detect when it falls to zero and the connection can be closed, after a suitable interval to permit other destinations to rejoin the group.

The Quality of Service (QOS) provided for a connection by an ATM network refers to the service parameters requested for a connection at set-up time and which the network attempts to guarantee. Values for maximum delay, jitter, throughput and cell loss probability are typical parameters, with peak and mean bit rates and some measure of the burstiness of the traffic being possible parameters supplied by the source [Leopold92, Campbell92]. Exactly what QOS means for multicast connections is rather unsettled as yet. One major issue is how to avoid the provided QOS being dominated by the least capable member of the multicast group. This problem is really connected with how the multimedia are transported through the network and is considered next, together with the problem of synchronising different media.

Synchronised Multimedia Streams

The substantial bandwidth required by many classes of traffic, such as video, means that some form of low-level multicast must be used to transport the data. This implies that one way of providing multicast QOS, by monitoring QOS on multiple totally separate unicast connections, is unrealistic due to the bandwidth involved for anything but small multicast groups.

Another approach is to multiplex all the media onto one VCI. This means that the QOS of the entire connection has to be determined by the least capable member of the group. For example, if a low resolution monitor was added to a video conference, the resolution at all the other monitors might have to be decreased to match it, which is clearly undesirable. In addition to this 'least capable member' problem, it is much harder for the destinations to restrict which media they wish to receive, particularly if the receiving restrictions are due to limited buffer space or slow line speed at the destination.

The third approach is the correct way to carry multimedia over a virtual connection-oriented network. This is to assign the different media to separate connections. This means that QOS considerations can be applied more sensibly on a per-media basis and choices of which media to receive (and pay for) can be made more easily.

At the cost of using more VCI/VPI space, yet more levels to the hierarchy of media can be permitted, perhaps dividing video into high and low quality grades. This approach can be taken even further and the components of each media can be transmitted on different connections, e.g. colour and brightness of video, or high and low spatial frequency components. However, this approach must ensure that both source and destinations have an agreed set of grades and that both can signal the network as to which grade they wish to send or receive. Ensuring that such as scheme can be easily upgraded as the expected bandwidths increase means that a fairly broad range of QOS levels should be defined prior to wide-scale use of B-ISDN.

The separation of media according to their nature (and possibly their quality) raises the problem of synchronisation between different media. The multicast connection is carried across the network in a virtual path, containing multiple virtual channels. Some channels are multicast at most switches, others at only a few, according to what each destination can receive. To ensure a reasonable degree of synchronisation between the media it would be helpful if the virtual path could be constrained at connection set-up and during modifications to make sure that the virtual channels are routed along the same physical series of links and switches *(call bundling)*. Individual channels within the virtual path can have totally different QOSs as is appropriate to the different media being carried over them.

QOS and synchronisation issues are not limited to the transport layer. The QOS is defined for a connection, which affects CAC at lower levels. Synchronisation is also a higher level issue, as mentioned earlier. Recent work on synchronisation at this and higher levels has been performed locally by Sreenan [Sreenan92], using a synchronisation server on a local network with minimal charging in order to reduce the overall costs of resynchronisation.

The transport layer is also partly responsible for the provision of reliable multicast, if such multicast is required. An example of this is described later in Section 2.11.4.

2.5 Network Layer 1

In many situations, the individual connection management functions of the network layer may operate above those of the ATM adaption layer. An example of this is where two networks are interconnected by another network, but know nothing about the format of each other and information is exchanged on a per-packet basis. In this case, connections are set up and routing occurs prior to the SAR and multiplexing of the AAL.

The call acceptance issues which were raised in the previous section were considering the requirements for end-to-end multicast connections and their effects upon the network as a whole. At the network layer, call acceptance will occur on a hop by hop basis, according to whether routes with suitable QOS can be found to accept the request for a connection. The connection set-up and tear-down in an ATM network is part of the control plane functionality summarised in Section 2.10.

2.6 ATM Adaption Layer (AAL)

The presence of one-to-many multicast complicates reassembly of cells into higher level PDUs in the Segmentation and Reassembly (SAR) sublayer. When PDUs for multiple destinations are being reassembled at the same time, a means of distinguishing each destination must be provided. This can be done by embedding a *Multiplexing IDentifier (MID)* in the cell payload or using the VCI field to indicate the source's identity and the VPI as the connection identifier. Since an ATM network can route multiple VCIs together on just one VPI, the latter approach does not use any of the small number of data bytes available in each cell. As an example of the former approach, the presence of an MID of 10 bits in the data is proposed for AAL 3/4.

SAR for many-to-many multicast is more complex since when multiple sources are being multiplexed onto a single VCI, a means for identifying the individual sources is also necessary. This may be done by embedding an MID in the data as before, but now requires greater control over the choice of MID by each source to guarantee that each MID is unique.

2.7 Network Layer 2

Connection set-up occurs under the guidance of the group management in the transport layer and involves using the control plane to set up the connections on a link-by-link basis. Call acceptance occurs on a link-by-link and connection basis at this layer and management plane information is used to determine the current state of each link as necessary.

The optimal routing of a multicast connection is equivalent to the connection of a subset of nodes in an undirected graph, so as to minimise a set of costs associated with each edge in the subset. This problem is known as the Minimum Steiner Tree problem and is known to be NP-complete [Karp72]. The problem is actually one of multiple Steiner Trees, since each virtual channel within a multicast connection will require a different, though related, multicast tree. The problem is complicated still further by the desire to construct multicast trees which are robust in performance with respect to addition and deletion of nodes from the multicast group.

One desire of routing based upon using minimum Steiner trees (or approximations thereof) is that the information should be replicated as late as possible in the transmission path, in order to help to reduce the total volume of data in the network. Unfortunately, this conflicts with the desire to replicate the data as early as possible for reliability, since widely disjoint paths are less likely to suffer from the failure of the same single node. The provision of fault-tolerant multicast and multiple paths for the purposes of redundancy is not addressed further here, but it may be assumed that if such failure is rare, then the connection can be re-routed and that the overall risk of cell reordering will be minimal.

In public networks, the accounting procedures must be modified to charge customers accordingly. The exact nature of billing for ATM resources has yet to be determined but might possibly be on a per-packet basis. Whether incoming packets are charged for as well is another matter for further discussion. Multicast connections might be billed according to both the number of packets generated at the source and the current number of destinations in the connection, or some fraction of it. A preliminary discussion of ATM billing may be found in [Mackay91] and a more general introduction to network accounting elsewhere [Mills91a]. The subject is also referred to again in Section 9.3.3.

2.8 ATM Layer

Since this is the lowest layer at which the information in the network can be sensibly replicated, the following substantial modifications to provide multicast are necessary.

Firstly, the number of hosts to which a connection is multicast, that is the degree of replication or fanout, has to be stored either somewhere in the network entities (switches) or within each cell. The cell structures shown in Figure 1.2 do not have a field in the header for this information, since the degree of replication would be limited by the space available in the header and the per-copy routing information would still have to be set up in the switches anyway. The most efficient place to store the fanout is at each switch, associated with the VCI/VPI information. Thus cells can be replicated on a per-connection basis and under the control of the call set-up procedures. The addition of a field in the VCI remapping table to indicate the fanout is relatively inexpensive in terms of storage space. For instance, if a maximum fanout of 256 is permitted at each switch and replication is done only on 2^8 different virtual connections and 2^8 different virtual paths, then an extra $(\lg 256\times\ 2^8\times\ 2^8)/8 = 64$Kbytes of storage is required at each switch[1].

Secondly, some means of identifying individual copies of cells has to be provided to enable subsequent switches and their final destinations to demultiplex them correctly from the incoming stream of cells. If the replication function is separate from the routing function, as described in Section 3.5, one approach to identification of copies is to leave the VCI unaltered in each copy, but to ensure that the remapping tables at each port of a switch map the same incoming VCI to differing outgoing VCIs and then route them to their respective output ports. This approach increases the risk of misdelivery of a cell if there is an error in the replication process, since the port at which a cell incorrectly arrives would quite possibly have an entry defined for that VCI in its tables.

[1]Here and in later chapters, lg is used to refer to logarithms with a base of two.

A different approach is to ensure at the time of replication that the VCI is different for each copy (perhaps chosen from a contiguous series derived from the original VCI). Whilst this does reduce the risk of misdelivery slightly, the differing VCIs still have to be remapped to preserve the need for VCIs to be meaningful on a per-switch basis.

Thirdly, the use of flow control to ensure proper matching between source and destinations must be extended to permit different overloaded destinations to request the source to reduce its output rate. The simplest approach is to permit any of the destinations to initiate flow control restrictions. This has the disadvantage that in a multicast connection with homogeneous destinations, all the destinations may attempt to restrict the source at approximately the same time, potentially leading to disastrous performance. One solution is to use some form of backwards explicit error notification (BECN), where congestion information is passed backwards through the network to the offending source. One way to do this is to have the source count the number of 'choke' packets or BECN cells from each destination in a fixed period and to use the peak count over all destinations as a guide to the degree of restriction required. Another possibility is to discard a certain fraction of the BECN cells and to ensure that the source reduces its output by a greater fraction. This approach risks long response times and widely oscillating traffic rates. Another minor disadvantage of BECN is the need for a reverse connection to be set up, though this will probably be necessary for OAM anyway. The overall observation here is that flow control must be dictated by the least capable member in the group of destinations, but that if the difference between destinations is small, then there are very real risks of a 'reverse flood' of control messages.

Fourthly, the presence of multicast in a switch increases the probability of a multiplicative effect upon bit errors, especially in the VCI/VPI field. This means that the multicast fabric in a switch should be made more reliable by using a CRC over the cell header as suggested by CCITT [CCITT I.432]. The switch architecture should also be designed to limit the effects of a misrouted cell by signalling higher levels about cells which arrive with unknown VCIs.

2.9 Physical Layer

The transmission convergence sublayer requires little change except possibly to ensure that the HEC CRC generator is effective for contiguous cells whose VCI/VPI fields may differ by as little as one bit. This situation could occur if the VCIs within a multicast group were assigned as a series of contiguous addresses and replication of the cells occurred in a sequential rather than parallel fashion.

The physical medium sublayer of the physical layer is concerned only with the medium itself and bit timing. Hence providing multicast requires no changes. Networks which use media such as radio, and networks based upon rings and buses, both have an intrinsic ability to broadcast information, can be used to provide multicast in the local area without the explicit replication of cells at the ATM layer. Whether a network has this intrinsic multicast capability must be known by the

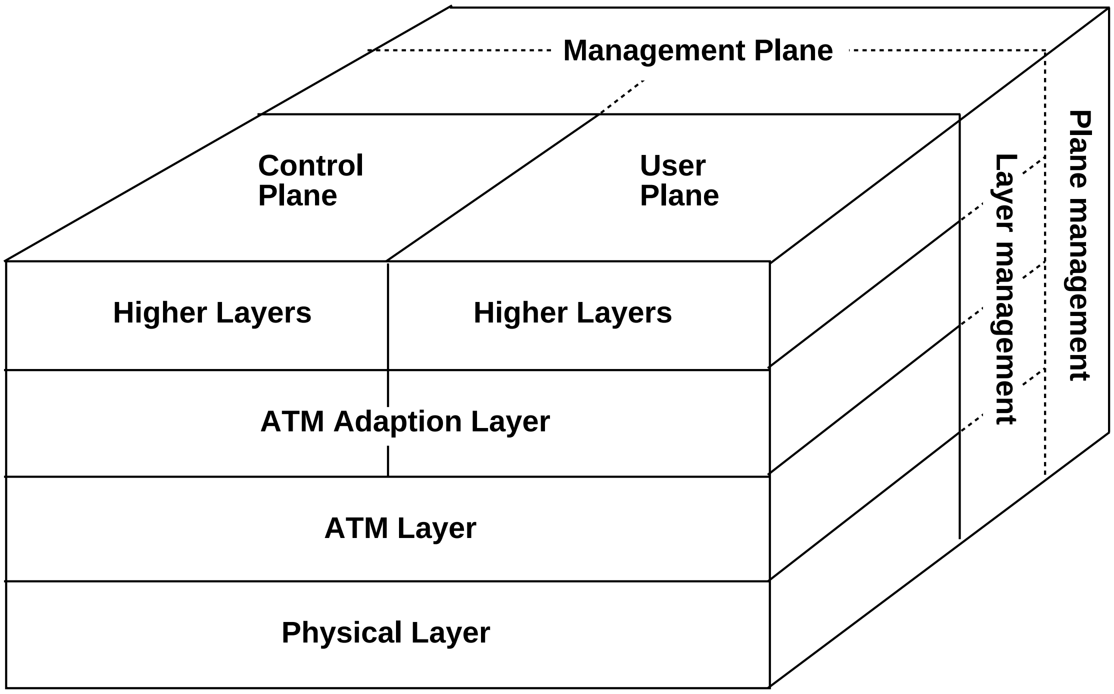


Figure 2.2: User, Control and Management Planes in the B-ISDN Protocol Reference Model

network bridge and the routing algorithms, but no other network entities should have to know.

2.10 User, Control and Management Planes

The B-ISDN protocol model for ATM is based upon the three dimensional model of a network developed for narrowband ISDN. In addition to the levels described in Sections 1.1.2 to 1.1.4 and again in Sections 2.6, 2.8 and 2.9, there is another dimension to the model, as shown in Figure 2.2. One plane in this dimension is the user plane, which transports the users' information. Another is the control plane, which is responsible for connection management, i.e. connection set-up, tear-down, QOS, call acceptance control and routing. Still another plane is the management plane, which maintains the network by performing usage monitoring and reporting, security management, fault detection and reporting, configuration and topology discovery and financial operations such as charging. Each of these planes is affected by the provision of multicast in the manner that these operations have been described in the preceeding sections.

2.11 Existing Work

This section considers some of the wide range of existing work in the literature concerning multicast in communication networks. The emphasis of the selection presented here is upon the provision of multicast by a network rather than, for example, the use of multicast by a distributed computing system.

Existing multicast within circuit switched networks such as the global telephone network relies upon replication in the local area, often by assigning multiple sources a single slot in a time division multiplexed (TDM) frame. This works reasonably well with voice traffic, since human voice interactions mean that when collisions occur the loudest speaker will probably continue to speak whilst the others become silent, rather like CSMA/CD. This approach is obviously not suitable for video or data traffic, nor does it work in the wide area or even scale particularly well in the local area. Due to these reasons, commercially available PABXs often limit the number of members in a multicast group to less than a dozen.

Some early work concerning multicast in ATM networks was performed locally in Project Unison [Tennenhouse87]. In this project, which was an experimental connection of cell relay LANs over an early primary rate ISDN, INMOS transputers were used to replicate and distribute Cambridge Fast Ring [Hopper88] packets (32 data bytes) across interconnections using the Alvey High Speed Network and then MultiLine IDA from British Telecom (2Mbit/s). Work by Shrimpton and Cooper [Shrimpton90] describes how a dedicated server at each source was used to make the required number of copies of every multicast cell and to route each copy individually to its destination. Maximum data rates of 1.1Mbits/sec were quoted for multicast of one source to a maximum of four destinations. The maximum data rate decreased in a linear fashion as the degree of multicast increased above four destinations, eventually reaching approximately 130Kbits/sec with fifteen destinations. The copying and routing were still not performed entirely in parallel and so a large delay could exist between the generation of the first copy and the last.

Much of the work in multicast over the last ten years has been concerned with the implementation of multicast for existing point-to-point protocols, for example IP for the Internet [Deering89, Waitzman88, Deering85] and JANET in the UK [Crowcroft92]. The use of multicast in distributed computing protocols has also been slow to become established, partly due to the wide range of requirements by applications which might make use of it [Paliwoda88]. It is only in more recent years that the problem of provision of multicast at a lower level in point-to-point switches for wide and local area networks has become an active topic of research.

Examples of significant work in the field of extending the capabilities of existing networks are given below, followed by examples of recent work for future networks based upon ATM. Details of existing work concerning the replication of cells at the cell level appear in Chapter 3.

2.11.1 Multicast in Ethernet

Each network interface attached to an Ethernet [Ethernet80] is initially configured to receive on both its own address and the local broadcast address (all ones). A many-to-many multicast group can be set up by requesting each member of the group to reconfigure their network interface hardware to receive a multicast packet on an address which has the bottom bit of the top byte set, e.g. (hex) `01.00.00.00.00.00`.

Any host can transmit to this address since it has to be known by the whole network.

Unfortunately, many implementations of network interfaces implement only a small number of multicast addresses and receive packets for each multicast address on a relatively large range of addresses. They depend upon discarding unwanted packets by software at a higher level, or else hope for a large separation of multicast addresses to avoid overlap. The actual number of separate multicast addresses which can be flagged for reception is very small, fourteen being common in current implementations.

These hardware restrictions often mean that when device drivers are subsequently written for network interfaces multicast is not supported, forcing applications to resort to broadcast. This is not too serious a problem in a shared medium network such as an Ethernet, unless each destination generates acknowledgements or some other such feedback occurs.

2.11.2 Multicast for IP

The provision of a multicast addressing scheme for the Internet TCP/IP protocols is a relatively recent addition to IP and is summarised well by Comer [Comer91]. It is described more fully by the relevant RFCs [Deering89, Deering85] and also in Deering's thesis [Deering91]. A brief description of the scheme follows, as an example of a working implementation of multicast in a heterogeneous networking environment.

Multicast groups are defined where a member of the group receives all the datagrams sent to the group. Any host can send datagrams to a multicast group, without being a member. Each multicast group has a unique multicast (class D) address. Some class D addresses are well-known and persist even when unused, others are transient. Networks are connected using multicast gateways, which communicate with each other using the Internet Group Management Protocol (IGMP) [Deering89]. Multicast gateways can be physically separate from existing gateways, or the capability can be added to existing gateways.

IP multicast addresses have the top four bits set to the pattern 1110 and the other 28 bits are viewed as a flat address space. The dotted decimal address 224.0.0.1 is reserved as the 'all hosts' group, which broadcasts to all hosts and gateways involved in multicast. Three levels of participation in IP multicast are defined for a host: none at all, 'send only' or 'send and receive'. Whilst the first two are not difficult to implement since the IP address is just another class of address, the third level requires more extensive modifications to identify multiple sources.

To provide IP support for local multicast, hosts join multicast groups which are specific to networks. This also helps to reduce the risk of network 'meltdowns', where a single host attempts to broadcast to every machine on the Internet. IGMP ensures that replies to multicast queries are spread more or less evenly over time and that only one reply is needed to ascertain the existence of a group, as opposed to risking a reply from every member of that group. No provision is made in IGMP for acknowledgement or retransmission of a lost message. It is presumed that this takes place at a higher level.

The mapping of the multicast datagram at a multicast gateway to addresses for the destinations is dependent upon the nature of the destination network. To provide IP over an ATM network supporting multicast, a mechanism similar to the way that IP multicast addresses are mapped onto Ethernet addresses should suffice if VCIs are plentiful. Details of this and future work on the interconnection of heterogeneous networks using ATM can be found elsewhere [Heinanen92]. In this situation, the low 23 bits of the IP address are placed in the low 23 bits of the Ethernet multicast address (hex) `01.00.5E.00.00.00`. Thus 2^{28} IP multicast addresses are mapped onto 2^{23} Ethernet multicast addresses. The probability of a two IP addresses chosen at random mapping onto the same Ethernet address is $2^{(28-23)}/2^{28}$, which is approximately 10^{-7}, but a higher level check of correct delivery becomes necessary. For strictly point-to-point networks such as an ATM network without multicast, multiple copies must be generated by the multicast gateway, which would further complicate its design.

The routing algorithm used for multicast groups in the Internet is the Distance Vector Multicast Routing Protocol (DVMRP) [Waitzman88], which is derived from work by Deering [Deering91], which in turn is a refinement of Dalal and Metcalfe's Extended Reverse Path Forwarding algorithm [Dalal78]. The essence of Reverse Path Forwarding is that if each node knows which link it should send packets on to reach a node by the shortest route, then a broadcast tree can be constructed with that node as source in a distributed fashion as follows:

1. The source broadcasts a packet to all its neighbouring nodes.
2. When a node receives a packet, it forwards it to all its neighbours if and only if it came from the link which the node would use as the shortest route to the source. Otherwise it discards the packet as a likely duplicate.

If the length of the route from the source to any node and the route from that node to the source are equal, then this algorithm generates a minimum spanning tree of the network. The extensions consist of different methods for reducing the number of packets which are forwarded during topology (re)discovery by detecting which links are actually used in shortest reverse-path trees. Approximations to optimal spanning trees can be obtained by generating 'pruning' packets at routers if no member of the multicast group is part of the local subnetwork. This prevents subsequent packets from reaching areas of the network which are not involved in the multicast group. A 'lifetime' is attached to the decision to prune a branch of the network to prevent routers becoming full of outdated information. A useful summary of distance vector routing and indeed multicasting routing in general is provided by Ballardie [Ballardie92].

DVMRP suffers from the well-known scaling problems of any distance-vector routing protocol. Whilst only a small increase is necessary in the size of each entry in the routing table, there is the increased cost of cacheing information about the currently active multicast senders and their destination groups. This extra cost is approximately proportional to the number of of active source and destination pairs in the network. The necessary cost of computing and storing a multicast spanning tree for each member of the multicast group also fails to scale well with increasingly large groups.

2.11.3 IETF Audiocast

The Internet Engineering Task Force (IETF) has held two global conferences to date, transmitting audio and video data over the Internet. The first of these [Casner92] involved 20 different sites in March 1992, whilst the second involved over 100 sites and took place in November 1992. The traffic anticipated during IETF multicasts was 100-300Kbit/s, with a design bandwidth of 500Kbit/s. This is the equivalent of five simultaneous 64Kbit/s voice conversations. Higher bandwidth sources such as video transmit with a smaller IP time-to-live value, so that lower bandwidth sources such as compressed audio can be allowed through during times of high congestion. Most data degradation and packet loss was found to be caused by the variance in the transit delay, especially on the longer paths.

A virtual network called MBONE has been created on top of the Internet to support the IETF multicast experiments. Since most IP routers do not yet support multicast, MBONE consists of multicast LANs (such as Ethernets) linked by 'tunnels', which are interconnected machines with operating support for multicast. MBONE is configured as a star-shaped backbone with supplementary links for robustness. Each regional network is another star connected to the backbone by a small number of links (usually one), to which the local area networks are then attached. The hierarchy of networks ensures that new users are attached to their nearest node, physically and logically, in MBONE.

2.11.4 XTP: Xpress Transfer Protocol

XTP [Chesson92] is a protocol designed for high performance networks, and refers to a transfer layer which combines the functions of the transport and network layers of the ISO OSI model. It has been developed, together with VLSI ASICs, for networks which require reliable datagram mechanisms, fast connection set-up and multicast capabilities. It has also been used as the basis for the ANSI proposal to ISO for a High Speed Transport Protocol for the next generation of OSI.

There are two basic packet formats in XTP; data packets for the users' data and control packets to carry information about the state of each system. Multicast is supported by the presence of a bit in the header of each XTP packet to indicate whether that packet is to be multicasted on a group address. Each source is uniquely identified in the XTP subsystem by a field in the first packet on each association (connection). This identifier permits destinations to distinguish between multiple sources. When sources are located in different subsystems and the identifier may not be unique, the route followed by a packet is also used to distinguish between sources.

In the simplest reliable multicast scenario, each multicast packet is replied to by each destination and all members of the multicast group receive these replies. These replies contain control information to permit the use of a 32-bit end-to-end windowing flow control mechanism and rate control. The replies can be staggered in time to avoid instantaneous increases in the network load (slotting). At the cost of reading all the control messages, each destination can be instructed not to send its reply if the identical message has already been observed from another destination (damping). This reduces the total amount of network traffic. It is suggested that slotting and damping should be used in conjunction when the number of destinations is more than about a hundred.

In addition to the exchange of the flow and rate control information, updated values for maximum rates and burst rates are also exchanged. Time-out values for response and network failure timers are also calculated, using periodic control packet handshakes. A destination which wishes to join an existing multicast group can send a packet to the source (whose existence is derived from a higher level entity not defined in XTP) which returns the control information for the current state of that group.

A 'bucket' algorithm defined in an appendix of the XTP Definition document [Chesson92] provides the mechanism for reliable multicast for members of a multicast group. A bucket (buffer) is defined for each packet and data for retransmission is stored in this buffer until either all the destinations' reply packets have been received or all the buffers are full, in which case the oldest is reused. Flow control and round trip delay information is derived from the returning control cells and used to determine when to retransmit the data and when a destination can be considered to have failed and be forced to drop out of the multicast group. Other algorithms involving selective retransmission can also be supported.

Possible future modifications to the XTP protocol could include support for multicast algorithms for routers and formal methods for specification and verification.

2.11.5 MOSPF: Multicast Open Shortest Path First Protocol

MOSPF [Moy92] is an extension of OSPF[Moy91], which is an interior gateway protocol based upon Shortest Path First (SPF) routing, where each node is aware of the topology of its entire area and regularly broadcasts information obtained from 'pinging' its neighbours to other nodes connected to it. Each node builds routes using the well-known single-source shortest-path algorithm due to Dijkstra [Aho83]. SPF also scales better than distance vector routing (Section 2.11.2) since the link status messages only carry status information about the immediate neighbours of a node, rather than distance information for a much larger number of nodes.

OSPF was designed to be an open standard to encourage its use and to provide support for security considerations. It also introduced to TCP/IP protocols the concept of multipath routing to provide support for type of service (analogous to QOS) and load balancing for congestion control.

MOSPF is the extension of OPSF to provide multicast for IP, eliminating the use of tunnels (Section 2.11.3) and ensuring that within a domain no more than one copy of a packet goes over any link. The routing is again based upon pruning a shortest-path tree, just as in Section 2.11.2, but the tree is only built upon demand and is then cached. In addition, the routing tables in each node depend upon both source and destination to route a packet. OPSF maintains a topology database, which is extended in MOSPF to include the local-group database. This is a list of the networks which contain members of a particular multicast group. It is maintained by a single node on each subnetwork making regular 'link state advertisements' indicating for which groups it requires packets. As before, if a network has no members of a particular group, it is pruned from that tree.

Inter-area, or inter-domain, multicasting is performed by defining some nodes in a network as inter-area multicast forwarders (bridges), which forward a summary of the group membership information and the costs from the forwarders to the source, into the backbone nodes. This information is not propagated beyond the backbone to avoid excessive traffic. The partial distance information can be used as an approximation when deciding upon which forwarder is nearest in the reverse direction. If the links are symmetric, then this is exact.

The problem of heterogeneous multicast capability in the network, in that some routers may have support for multicast whilst others may not, is dealt with by only using multicast routers for multicast connections. This means that although there may be unicast connectivity between two hosts, there may not be multicast connectivity. In addition, the link state advertisements do not carry any information about whether a node is capable of multicast, so forwarding multicast packets between different areas is vulnerable to loss of multicast.

No mention is made in the draft document [Moy92] of how modifications to a tree might affect the exchange of information between routers. Security is also not addressed as a separate issue. Interoperation between DVMRP and MOSPF is currently being developed to try to reduce the overall number of tunnels which have to be configured to support DVMRP.

2.11.6 CMAP: Connection Management Access Protocol

CMAP [deHart92] is a protocol for managing multipoint connections in high speed packet switched networks, especially ones which use ATM. It has been developed at Washington University in St. Louis and has been put forward to the ANSI T1S1 technical sub-committee as the basis for a connection management model. A proposal has also been put forward to the ATM Forum for a signalling protocol based upon Q.93B with extensions inspired by CMAP. CMAP controls the creation, modification and deletion of calls between a user and the network. Data is exchanged using protocols specified separately from CMAP. CMAP is intended for use in both the wide and local area, as is appropriate for networks which use ATM end-to-end.

CMAP uses the notion of a 'multipoint call' as a many-to-many connection, with one-to-many, many-to-one and point-to-point calls being restricted versions of multipoint calls. The notion of ownership of a call exists and the ownership can be transferred during the existence of a call. Calls can be specified so that they are *open,* where any end-point may join, *closed,* where end-points can only be invited by the source to join and *verify* where the source must give permission to an end-point for it to join.

The bandwidth requested by each connection in a call is specified by three parameters: peak bandwidth, average bandwidth and peak burst length. The peak and average bandwidth are the maximum and average number of cells per second respectively and the peak burst length is the maximum number of cells which should appear in a burst at peak rate. The bandwidth parameters can be static, i.e. fixed at the time of creation of the connection, or dynamic. Zero values for all three parameters will cause the network to carry the connection on a 'best-effort' basis.

The requested QOS parameter is kept deliberately vague and can be 'high', 'low' or 'medium'. This is because it is assumed that each network will have different cell loss behaviours and therefore the parameter should be considered meaningful only inside a single network. A three level priority parameter is also provided, unrelated to the ATM priority bit, for the purposes of call acceptance control.

Calls, connections, clients and their end-points are all assigned fixed-length unique identifiers. The client address is a combination of the CMAP independent address such as an IP address or a CCITT E.164 address [CCITT E.164], as used in public telecommunication networks, together with a local address which is passed transparently through the network for use by CMAP operations.

Possible modifications to CMAP in the future include adding support for transactions comprising multiple CMAP operations and further degrees of control over the abilities granted to end-points for modifying calls. The functionality proposed by CMAP has been used as the basis of the modifications to an existing multi-service protocol in Chapter 7.

2.11.7 SuperJANET

The ATM Technical Advisory Group of the major UK wide area network JANET (Joint Academic NETwork) is currently in the process of defining SuperJANET [Wilbur92], which is to be an integrated multimedia high speed (155Mb/s) wide area network using ATM as its transfer mode. It will initially operate between a limited number of sites and one of its aims is to investigate the use of ATM end-to-end, that is, in the local area as well as the wide area.

The issue of how multicast groups are supported in SuperJANET is still under discussion but one proposed routing scheme is along the lines of Core Based Trees (CBT) [Crowcroft92]. CBTs make use of the hierarchical nature of real networks which are composed of multiple subnetworks linked by a backbone. For each multicast group, a core of routers (or switches) is based upon a shortest-path spanning tree. There may be one or many routers in the core and the core may grow or shrink as members are added to or removed from the multicast group. Other routers involved in the multicast tree are termed 'non-core routers'.

When a destination wishes to join a multicast group, it sends out a *send-join* *request* addressed to the original core router. A directory service is assumed here. Intermediate routers on the path to this core become non-core routers. If in fact they are part of the backbone themselves as intermediate nodes (minor cores), but are attached to a different major core router, then they acknowledge the *join request* and return a *join-ACK* to the previous intermediate routers and source, indicating that the second major core router should be used in preference to the original core. In this way, multicast trees can grow around a backbone instead of a single source. The assignment of routers as core routers is controlled by the directory service and takes place when new members are added who are geographically distant from present members or for reasons of robustness.

Failure of major and minor cores is managed by keeping parent, child and time of joining information for the core of each multicast group. In the event of failure the youngest contacts the next oldest, and so on, until the oldest active core member is reached. The member which was added after the failed node is responsible for joining the parent, if any, of the failed node. In this way connectivity is reestablished. No indication of how a core is modified once a failed node recovers is given in the draft description [Crowcroft92].

The issue of potential cell-reordering during partition and reconnection of the core has not yet been addressed, nor has the merging of multicast groups for applications such as conferencing. However the hierarchical approach and use of a backbone which have been adopted by CBT seem to tie in well with the topological characteristics of current networks.

2.12 Areas of Multicast for Further Study

The range and number of problems indicated in Sections 2.1 to 2.10 is rather large. Two of the areas mentioned above have been chosen for further examination. Both were seen to be areas of current interest in the literature and of interest within the Laboratory.

The replication of ATM cells at the switch level has been a topic of interest since the mid 1980's, originally in the context of fast packet switching [Turner88, Newman88]. The problem contains both elements of mathematical complexity and also technical difficulties in constructing a multicast switch.

The application of the Steiner Tree problem to multicast routing dates from the early 1980's, with several heuristic solutions [Wall80, Kou81, Rayward-Smith86] appearing since then. As ATM networks were implemented, it was generally realised that much of this work was of a more abstract nature and that real ATM networks might well be routed in very different ways from that implied by this literature. Chapter 8 contains the results of reconsidering this problem.

2.13 Summary

Providing multicast requires consideration of its requirements at all levels of the protocol stack. Some of the problems which are raised are non-trivial and require substantial effort to solve and implement properly. Some existing work directly involved in the provision of multicast has been reviewed.

Two areas have been chosen for further examination; the replication of cells at switches at the ATM layer and the routing of multicast connections at the network layer.

3 Multicast ATM Switches

Section 3.1 describes the basic functionality of an ATM switch and the different places where multicast can logically occur. Section 3.2 considers some of the issues which affect the design of an ATM switch. Section 3.3 gives a simple classification of the different methods of replicating and routing cells in a switch. Section 3.4 describes various designs for multicast switches which replicate the cells within the switch fabric. Section 3.5 describes switch designs where a multicast fabric is placed before conventional switch fabrics to convert single multicast cells into multiple unicast cells. Finally, Section 3.6 provides a series of comparisons between the various approaches and designs.

3.1 Asynchronous Transfer Mode Switches

ATM switches are the points in ATM networks where the ATM functions described in Chapter 1, such as VCI remapping, take place. Logically, an $N \times M$ switch is an interconnection between N input lines and M output lines such that any input can have incoming cells routed to any output, and multiple instances of such routings can exist at the same time. If multiple inputs attempt to route cells to the same output at the same time then contention occurs and only one input succeeds. If one input wishes to send a cell to multiple outputs at the same time, then the switch has to be able to provide multicast. Many such virtual connections can be set up at each input port, according to the size of the VCI/VPI remapping tables, which are used to decide which output ports a cell should be routed to. Figure 3.1 shows a 4×4 switch with two multicast virtual connections set up, each one to two outputs, with cells being routed at the same time.

The basic purpose of the input and output port controllers is to perform the necessary VCI remapping and to ensure that the cells are routed to their correct destinations across the switch fabric. To this end, buffers may be provided in the input port controllers, output controllers, within the switch fabric, or at a combination of the three locations. These buffers maintain the queues caused by contention for output ports and by short-term traffic rate overloads.

The operation of a multicast switch can be separated into two parts: the routing of the cells to their correct output port and the replication of the cells to provide multicast. In the switch shown in Figure 3.1 both of these functions occurred in the switch fabric itself (contention is occurring at the upper output port). Another class of multicast switches can be constructed by separating the two functions physically into a multicast fabric to replicate the cells and a (unicast) switch fabric to route

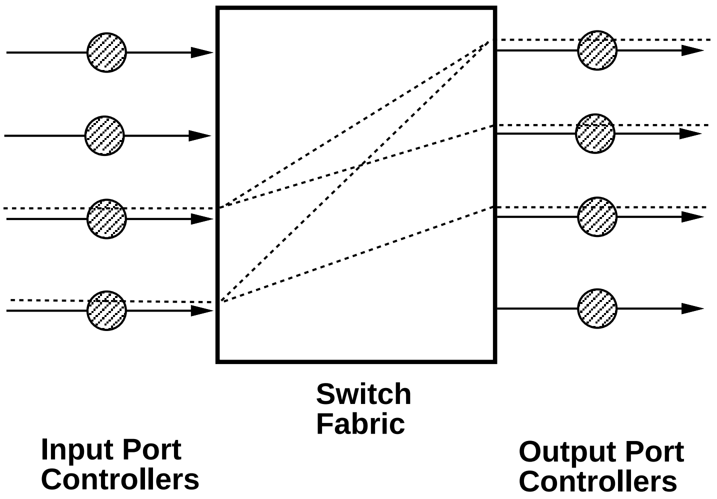


Figure 3.1: 4×4 Multicast Switch Fabric

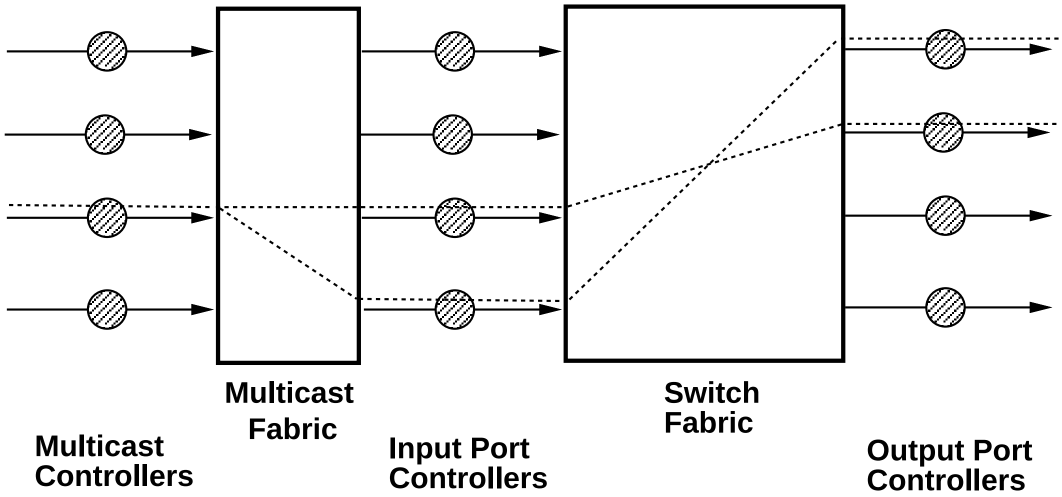


Figure 3.2: Cascaded 4×4 Multicast Switch

them. This is shown in Figure 3.2 for a connection of fanout two in a 4×4 switch. The two fabrics are *cascaded* [Lea88, Turner86b] together to form a multicast switch.

The replication of the cells may be performed in serial or parallel in either design. If the cells are replicated serially then the control of the port controllers is greatly simplified, since there is no need for more than one controller to know about a particular multicast connection. The part of the hardware which performs the replication is also greatly simplified. The major problem with serial replication is that the delay between the first and last copies of the multicast cell grows linearly with the number of copies. For applications such as video conferencing, which are sensitive to the delay across the network, the difference in delay for different destinations could be a major barrier to successful operation. For instance, using a 155 Mbit/s link with ATM cells being replicated serially gives a minimum delay of

$(53 \times 8)/155 \times 10^6 = 2.7\mu s$ from cell header to cell header. If any modification of each copy is necessary to uniquely identify it, the delay is likely to be nearer $5\mu s$. If an application requires even just ten copies of a cell then the total delay from first to last cell is $50\mu s$. This is larger than average routing delays across a switch even at this small fanout. For this reason, all the designs considered in this chapter replicate the cells either strictly in parallel or in parallel with small delays between each copy.

3.2 Switch Design Issues

When a switch is being designed or compared to other switches, there are several important issues which have to be addressed. Apart from the central issue of this chapter, the provision of multicast, these include the following:

Delay

The absolute delay experienced by a cell traversing the switch. At line speeds of hundreds of Mbit/s, delays of tens of μs are common with most designs. The delay is important for interactive applications.

Variance of Delay (Jitter)

This is critical for applications such as video, which have a stream of data which must arrive at regular intervals at the destinations. Usually measured as a percentile, e.g. 100μs at 99%, meaning that 99% of the cells have a delay of less than 100μs.

Aggregate Capacity

The theoretical switch capacity is defined as the product of the maximum throughput theoretically available between any input port and any output and the number of input ports. Some designs of switch have a maximum throughput which is far less than might be expected, due to contention between input ports for the same output ports (blocking). For instance, with buffering only at input ports, Poisson arrivals and traffic uniformly distributed across a switch, the maximum throughput is $(2 - \sqrt{2}) = 0.586$ [Hui87].

Reliability

This is related to the potential number of different connections, theoretically available across the switch, but which become unavailable after the failure of one or many individual components of the switch. Several designs have been proposed [Newman88, Tobagi90] which replicate the whole switch fabric, partly for the purposes of reliability or else can reconfigure existing switch modules in the event of failures [Yang92].

Cell Loss/Insertion Probability (CLPb/CIP)

Many switch designs cope with contention by using finite buffers to store cells until they can be transmitted. When these buffers are full, cells are discarded and lost. For successful network operation the probability of this happening should be somewhere between 10^{-8} and 10^{-11}. These values are derived from the typical error-characteristics of the transmission media (copper, fibre) [Li87] and the economics of buffer space in different switch designs [de Prycker91]. The probability, CIP, that cells are misrouted internally and arrive at the wrong destination, i.e. they are inserted into the incorrect stream of cells, should be a factor of at least 10^3 smaller than the cell loss probability for reliable operation.

Complexity

This is a difficult to measure combination of the number of connections required in a switch, the number of pins on each IC, the number of ICs, their power requirements and their immediate and future costs and availability.

Scalability and Modularity

How the complexity changes as the number of inputs and outputs increases. The modularity of a design refers to how the switch can be increased in size, whether the increments are large or small, and how much reconfiguration of the switch hardware is necessary.

Inter-copy Delay

The delay between individual copies of a cell as they leave the output ports.

Concentration and Expansion

How many traffic sources can be statistically multiplexed onto a single input line and whether the design can be used as a distribution switch, where the number of outputs is much greater than the number of inputs.

Priority

Briefly, the Cell Loss Priority (CLP) bit in the header bytes can be used by a switch to decide which cells to preferentially discard on a per-VCI basis in cases when traffic is overloading the switch. It can also be used to decide which buffer the cells belonging to a particular connection should use in a switch which has multiple buffers for different priorities of traffic. A combination of these schemes might be used to provide a guaranteed service for some classes of traffic and to allow other classes to use the residual bandwidth in an unreliable fashion, perhaps at a discounted rate.

Miscellaneous

- Is the switch design sensitive to particular patterns of traffic arriving at its input ports?
- Does it cause the incoming patterns of traffic to be distorted in ways which might increase congestion in the network? For example, does it increase the clustering of cells which arrive closely spaced in time?

- Is the service presented to each input port the same, or do some get better service than others? If this is the case, what is the worst case if the switch is inherently unfair?

Some designs require that the switch fabric should operate at a multiple of the line rate. The factor by which the speed and often the technological complexity of the fabric is increased is known as the 'speed-up'. Complexity and cost are directly related to the speed-up.

3.3 Classification of Multicast Switch Designs

Routing and replication of cells are closely related functions. This may be seen from the fact that the methods used for combined routing and replication which are described in the next section can also be used for designing a multicast fabric for replication alone, as described later in Section 3.5. Switch fabrics can be classified according to the scheme shown in Figure 3.3. This classification is a simplification of the one originally presented by Newman [Newman88].

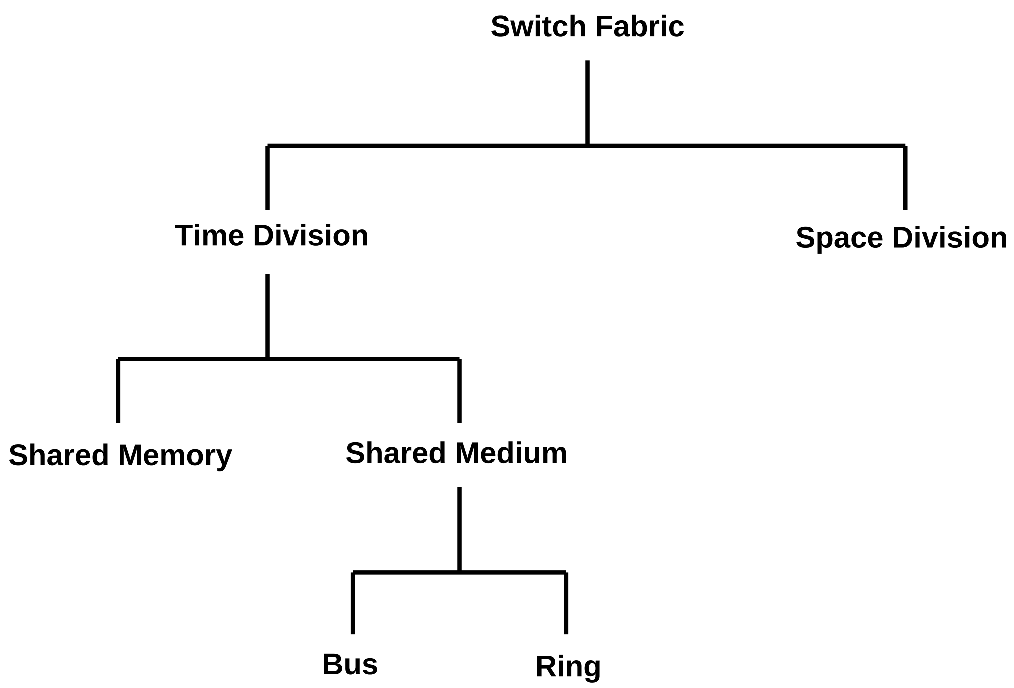


Figure 3.3: Classification of Switch Fabrics

Space division switch fabrics are constructed from a number of smaller switch elements in an interconnection network. Well known multi-stage interconnection networks (MINs) include the simple crossbar, where for N inputs and M outputs there exist $N \times M$ crosspoints, all of which have to be configured to make a set of connections. Other MINs with a smaller number of internal connections are the Clos, Beneš and Delta networks. Figure 3.4 shows two examples of delta networks, the omega network used in Section 3.5.4 and the Baseline network used in Section 3.5.1.

Interconnection networks can be further classified according to whether they are non-blocking, rearrangeably non-blocking or blocking. Non-blocking MINs can always form a connection if free links exist in the network, rearrangeably non-blocking MINs can make a connection by altering existing connections and blocking networks may not be able to connect some inputs and outputs though there are free links in the network, because of the nature of the network and existing connections. These networks are described in further detail elsewhere [Newman88, Hui90a, Chen91a].

Shared memory designs are the closest to traditional routers and bridges in packet-switched networks. A large memory is used to buffer the incoming cells and a controller decides which output to route the cell to when it is read from memory. For multicast, this has the advantage that the cell remains in the buffer and is readily accessible until the requisite number of copies has been produced. Shared medium designs make use of the broadcast nature of the underlying medium such as a bus or a ring. The fairness of both bus and ring-based systems are crucially dependent upon the method used to access the medium and unfairness can become a problem in some cases, especially if the access logic is distributed.

The number of switch designs which have been proposed in the literature is rather large, approaching at least fifty by late 1992. Most of the designs use space-switched interconnection networks and many are purely theoretical in nature. Further references to several of these switch designs may be found in the survey papers by Tobagi [Tobagi90], Ahmadi and Denzel [Ahmadi89] and Jacob [Jacob90] respectively. A recent overview of ATM switch designs with a wide-ranging bibliography may be found in [Newman92]. The book by Hui [Hui90a] contains an extensive treatment of the more theoretical aspects of space-switching and interconnection networks.

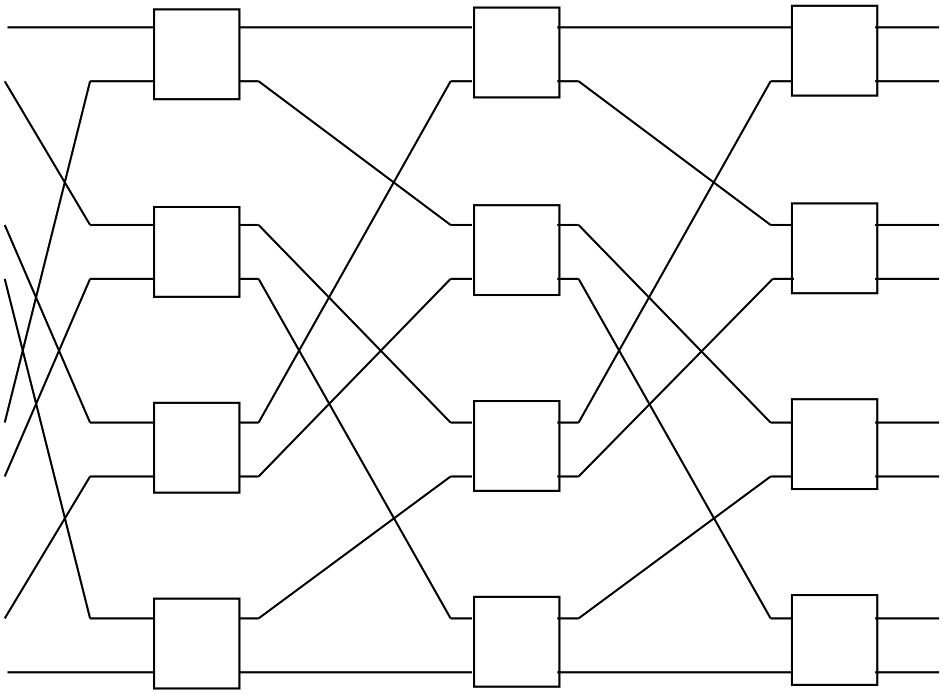
The following two sections contain examples from the three classes of space-switched, shared memory and shared medium multicast switch fabrics. Each design is described in turn and then a series of comparisons are made in Section 3.6. Note that the terms 'cell' and 'packet' have been used interchangeably except where indicated otherwise. 'Inlet' and 'input' have been used in the same fashion, according to the usage in the original reference.

3.4 Multicast Switch Fabric Designs

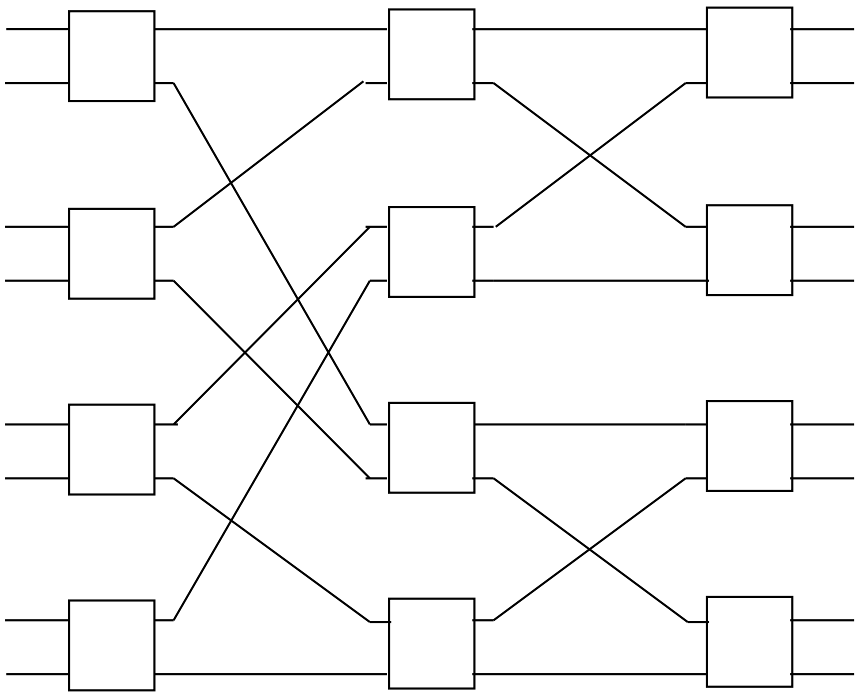
All of the examples described in this section implement the multicast functionality within the switch fabric itself, as shown in Figure 3.1, as opposed to using a cascaded multicast fabric as shown in Figure 3.2.

3.4.1 Space Switched – Richards and Hwang

Richards and Hwang [Richards85] propose a modification of the general Clos network to produce a rearrangeable non-blocking $N_1 \times N_2$ multicast switch fabric. Unlike any of the other designs presented in this chapter, this one was originally intended for circuit switching rather than fast packet switching and ATM. However, the approach



Omega Network



Baseline Network

Figure 3.4: Examples of Different Multi-stage Interconnection Networks

taken is still relevant to later discussions. Essentially, the first stage of switches are discarded, reducing the crosspoint count considerably and then each of the N_1 inlet channels is connected M times to the second stage of switches (now the first stage). The relationship between M and N_2 is not simple but upper and lower bounds have been established. Careful assignment of the inlet channels permits any final stage to access any inlet channel after rearrangement, regardless of the state of the rest of the switch. Each final stage switch is assumed to be a multicast switch itself, capable of broadcasting any input to all of its outputs. Thus any inlet channel can be connected to any number of outlets. Figure 3.5 shows the basic structure of such a switch fabric.

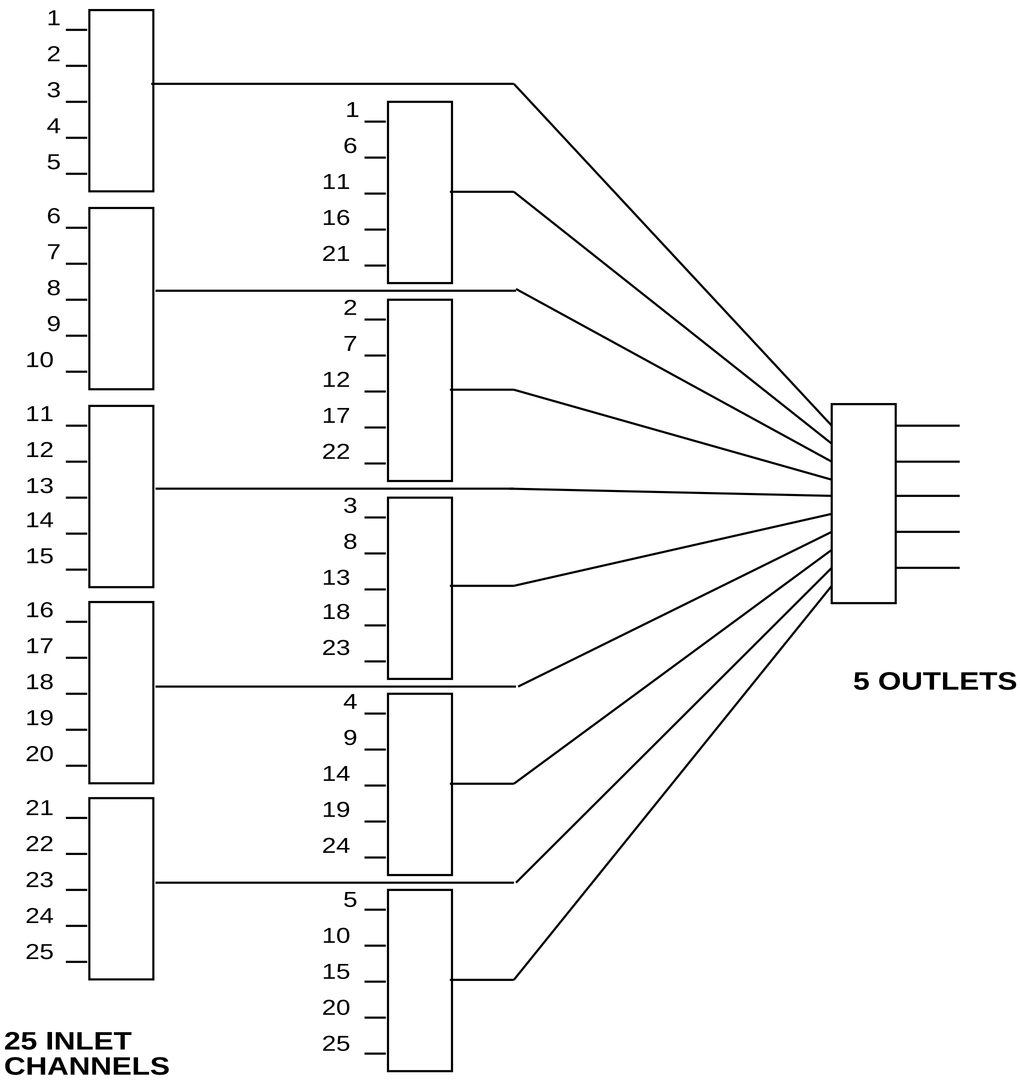


Figure 3.5: Richards and Hwang, first stage inlet assignment for $N_1 = 25$, $M = 2$, with one second-stage switch shown

One problem is that fully optimal assignment of the inlet channels to the first stage of switches requires as one condition that the number of inlet channels is the square of a prime. Thus further decomposition of each stages' switch introduces inefficiency, as some inlets are left unused. The optimal assignment patterns themselves, related to M and N_1, are as yet unknown. However, the growth of the number[1] of (electronic) crosspoints has been shown to be less than $O(N_1^{3/4}N_2)$ and is conjectured to be $O(N_1^{1/2}N_2)$.

3.4.2 Space Switched – Forcina, Di Stefano and Taormina

In [Forcina89], Forcina *et al* propose a multicast switch fabric composed of buffered Delta networks in parallel. Hybrid ATM (supporting both ATM cells and STM circuits) is provided by reserving part of the ATM frame (made up of ATM cells) for STM traffic. High priority traffic is served by considering it to be STM traffic and using the above mechanism, thus guaranteeing no conflict outside the switch between high and low priority traffic and hence helping to ensure low variance of delay for the high priority traffic.

Multicast connections are implemented in two phases using the same switch fabric. Firstly, incoming multicast cells (which become 'packets' within the switch fabric) have dummy maximum and minimum output port address fields attached to their headers. The entries for these fields are then calculated so that the range of output port addresses used is contiguous and their number is equal to the desired degree of multicast. The algorithms for this procedure require a centralised controller. The actual replication of the cells occurs in the switch elements, with each one being capable of broadcasting an incoming cell to all of its outputs. There is a monotonic relationship between the input port number of the multicast cell and the minimum output port address calculated. Secondly, the emerging packets are recirculated and re-enter the input ports in the same order in which they left the output ports. They then have their correct output port addresses attached, just as ordinary unicast cells do and re-enter the switch fabric to be routed to their destinations. This approach to emulating a multicast fabric and controllers is revisited in Section 7.2.2.

The entire switch is operated synchronously at the cell level using byte-wide data paths. The switch fabric has to be run at twice the rate of the incoming data lines to ensure that the re-entrant packets do not interfere with new cells on the input side. The two classes of priority are further enforced at the switch element level by the provision of high priority and low priority buffers within the switching nodes. The two phases described above are pipe-lined to ensure maximum throughput for the switch.

3.4.3 Shared Memory – Roxanne

ATM switch designs in which the incoming cells are buffered in shared memory include the Prelude switch [Thomas84] and the Hitachi switch [Kuwahara89, Sakurai90]. A packet is 'routed' through the switch when the output port receives a pointer to

[1]The notation $O(f(n))$ is used to indicate that the value under consideration has an upper bound of $f(n)$.

the packet header in memory. The design described in [Kuwahara89] permits smaller buffer sizes than would be needed for earlier designs and the performance for bursty traffic is also significantly improved, according to the simulation results. Multicast can be implemented in shared memory switch designs by permitting multiple output ports to share the same address pointer, thus receiving the same packet.

Roxanne

The chosen example of a shared memory switch fabric is the Roxanne switch fabric from Alcatel, Belgium [Henrion90, de Prycker91]. The basic design of the $N \times M$ switch element is the Integrated Switch Element (ISE), as shown in Figure 3.6.

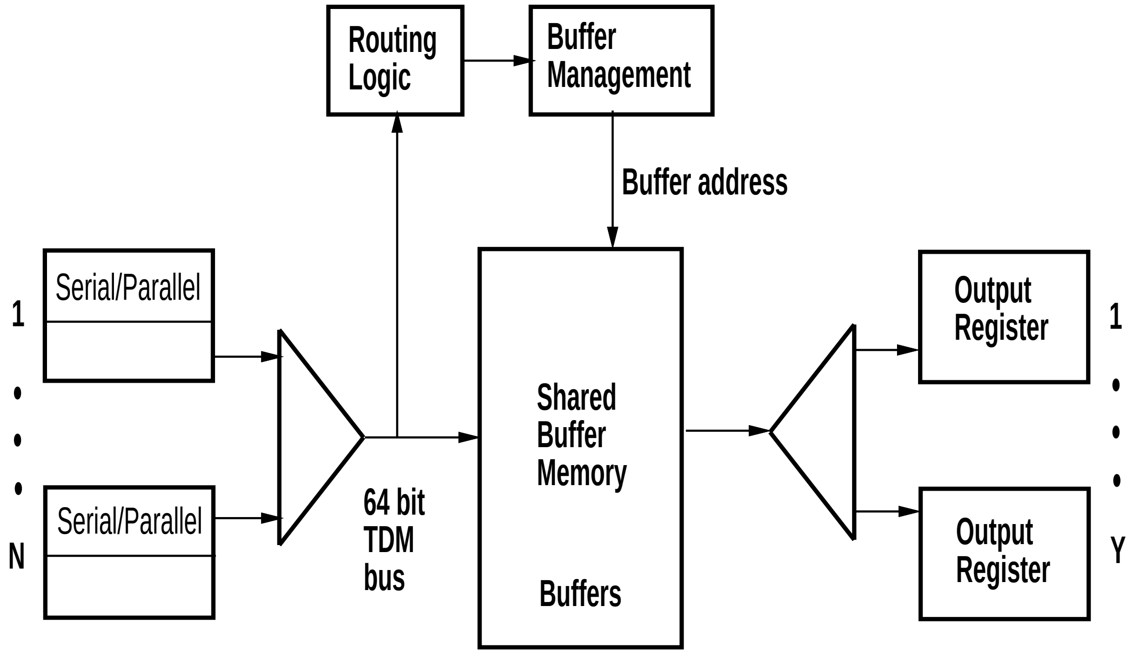


Figure 3.6: Roxanne Integrated Switch Element (ISE)

When cells arrive they are converted to parallel and then written into the Shared Buffer memory (SBM) via a TDM bus. For a 32×32 ISE at 150 Mbit/s, with a 64 bit TDM bus to the SBM, the memory cycle time is 13ns. It is suggested that for a CLPb of 10^{-11} at a total load of 0.8, storage for 145 64-bit wide slots will suffice, i.e. 1.16kB of dual ported memory per ISE.

The buffer management controls the linked lists of free and occupied cells. The routing logic determines for which logical group of 4, 8, 16 or 32 outputs the cells are destined. The outputs in a group are logically considered to be a single queue.

To provide multicast, a separate memory (not shown) stores the number of copies and the destinations of each copy. The internal reference number of the cell, generated upon entry to the switch fabric, is used to look up whether the cell should be placed in a group. Each queue (logical group) which is chosen indicates to the buffer management that it should decrease the copy count by one. When the copy count reaches zero, the buffer management authorises the release of the corresponding SBM cell location. The internal reference number is unique over the whole switch fabric for one multicast connection and all ISEs in each stage of the fabric must contain identical information about the connection.

When the switch element is used to construct switch fabrics, multiple switch planes are provided for reliability and for multiple paths to distribute the traffic evenly across the switch. The multiple paths may introduce reordering of cells, so an estimated increase of 20% in the SBM size is necessary to cope with resequencing cell streams to preserve cell order outside the switch. The individual ISEs also have an 'on board test routine' to continually monitor the state of the ISE and which propagates an 'available' signal to previous stages. If this signal is not received, then the previous stages ignore that ISE. This is especially important since if traffic is distributed evenly across the switch, a faulty switch element will affect all inputs.

3.4.4 Shared Medium (Bus) – Olivetti, Multibus

Some of the early designs of bus-based switch fabrics were IBM's PARIS switch [Cidon88], NEC's ATOM switch [Suzuki89] and the work by Nojima *et al* [Nojima87]. A recent implementation of a 16×16 LAN ATM switch as a commercial product has been based in part upon the use of a bus to both replicate and route cells [Adaptive92].

Olivetti Switch

Another bus-based switch fabric is that due to Milway [Milway92], currently being developed at Olivetti's Cambridge Research Laboratory. This design is for a small 4×4 switch as shown in Figure 3.7 and uses 32-bit wide FIFOs with a capacity for 64 cells. Cells can be read from any FIFO as they are visited in turn and written to any number of the output FIFOs using the time division multiplexed bus. The output FIFOs are much smaller than the input FIFOs and exist to cope with momentary delays in cells leaving the switch. A tag FIFO is used to maintain status information about each cell as it enters the FIFO (valid, short, HEC error, corrupt data) and is used as each cell comes to the head of the queue. The action of writing in the last word of a cell causes the cell to be read out of the FIFO if no other cells are waiting (i.e. the switch does not operate in cut-through mode). Basic flow control mechanisms are supported using the in-band signalling capacity of the transmission interfaces.

Multibus

A similar switch to that proposed by Nojima *et al* [Nojima87] is the multibus switch proposed by Yum and Leung [Yum92], which is described in further detail below. Its basic architecture depends upon multiple TDM buses connected in a crossbar fashion. In this sense Multibus is a space switch which happens to use buses to multiplex its inputs in time.

Figure 3.8 shows the architecture of the multibus switch. There are buffers at

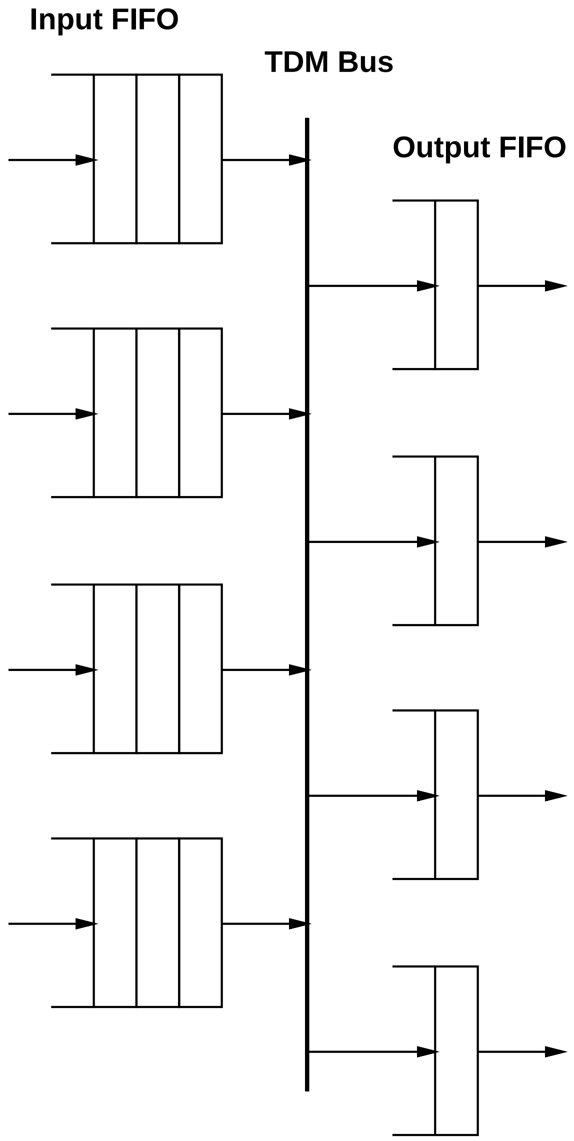


Figure 3.7: Architecture of the Olivetti switch

both the input and output adapters. The N input and output ports are grouped into M groups of L ports each. There are M^2 bus crosspoints, the switching elements. Each switching element can be constructed from $2b$ tristate buffers, b inverters and a shift register, where b is the bus width. The shift register stores a pattern of bits which at different times connects the incoming horizontal bus to either the downward vertical bus or the outgoing horizontal bus.

The control of the switch is divided into cycles and sub-cycles. It is intended that a one-to-one mapping of every input group to every output group can be arranged for each cycle, making the potential unicast throughput of the switch is equal to 100%, if no contention for output ports occurs within an input group. Each sub-cycle can be further subdivided to multiplex all the members of one input group onto

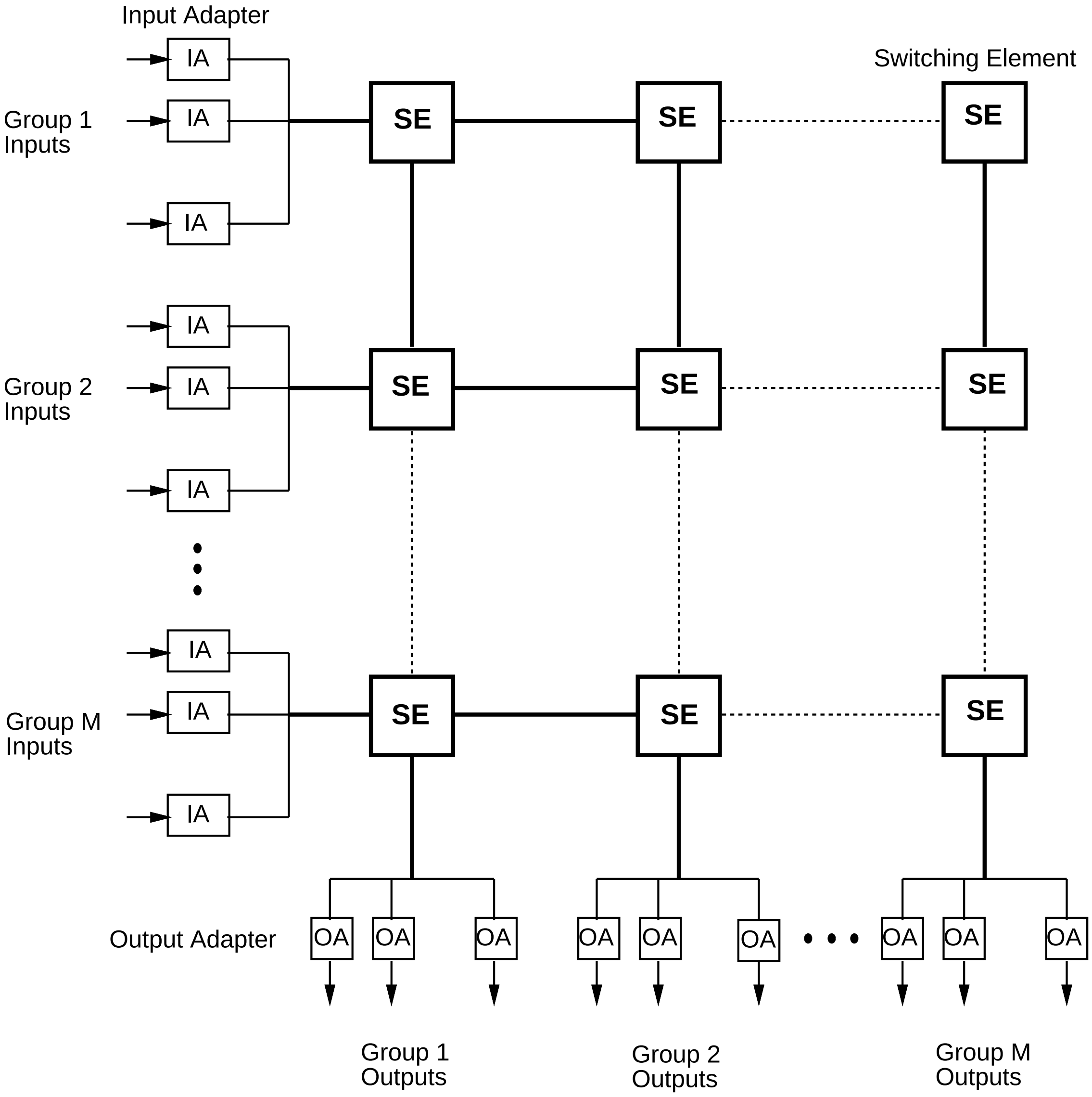


Figure 3.8: Architecture of the multibus switch

a single horizontal bus. A global clock is required across the entire switch fabric. An estimated speed-up factor of 1.6 (defined here as the ratio of the sum of the data rates in the vertical buses and the sum of the data rates of the input links) is necessary for a packet loss probability of approximately 10^{-9} with input and output buffer sizes of 30 packets and a total load of 0.8, assuming no output contention.

The modifications necessary to the design to support multicast are not described explicitly and so have been extrapolated from the description above. During each sub-cycle, each input group must be connected to multiple output groups. This requires modification to the switch element so that each horizontal bus is always connected throughout its length. The signal from the shift register becomes whether to connect the local vertical bus to each switch element.

The major advantage of bus-based switch fabrics is their simplicity and the gains in performance due to multiplexing multiple sources onto one bus, if the traffic has a statistical distribution which warrants it. Multicast is also theoretically simple to support. The major bottleneck in bus-based switch fabrics with centralised control is the configuration and set-up time for the switching elements. For a 32×32 switch with 4 ports per group, all at 150Mbit/s, the configuration of all the switching elements may have to be altered every 13ns, even with a speed-up factor of only one. This is similar to the set-up rate of Section 3.4.3. The problems of distributing the state information and a global clock over an entire IC at these rates are substantial.

3.4.5 Shared Medium (Ring) – Rahnema

Several designs have proposed the use of a ring-based switch element as part of a multi-stage interconnection network [Rahnema90, Gard90], as opposed to its use in Section 3.5.5 below, where it is used purely as a multicast fabric. The former design is described in more detail below.

Rahnema

Rahnema [Rahnema90] proposes the design of an ATM switch with an intrinsic multicast capability based upon constructing larger switches from smaller modules and using a slotted ring as the underlying network for each of the smaller modules. Assuming the ring to be any simple slotted ring system with regularly recurring time slots and output buffers, the design concentrates upon the input and output port controllers. These are designed to permit cut-through switching of variable length packets, that is, to start transmission of the packet before it has been fully received, reducing the absolute delay through each switching node for long packets. This is implemented by initiating transmission of the packet as soon as sufficient data has been received to specify the destination. Partial buffering provides a sufficient amount of elastic delay storage for the data portion of the packet. It is the job of the congestion control procedures to ensure that there is sufficient buffer space in the output buffers to receive the packet, which is otherwise lost. The possibly variable length packets are accommodated by ensuring that the FIFOs which make up the output buffers can accept the longest packet. 'Quasicircuit switching', defined as allocating bandwidth to a source according to its peak bit-rate requirements and expecting successive packets from a source not to enter the switch before previous packets have been transmitted is also implemented, using the proposed ability to selectively buffer packets at the output queues.

The multicast capability of the proposed design is provided by the use of a 'multipoint connection request switchword'. This contains the identification field (VCI) and the connection request information field (CRI), which carries one bit of information for each output port on the ring. The size of the CRI field is one of the limiting factors in this design, since the ring width must be wide enough to carry an entire CRI. One proposal is to break the information up into smaller units and to send a multicast packet to only some of its destinations each time. For each bit that is set, the packet is replicated at the relevant output port. To ensure that contention for output ports does not occur, either the ring width must be equal to the number of output ports, or the ring must be clocked at a much faster rate than the input lines, or blank fields must be inserted behind the CRI to compensate for the transmission time of the CRI on the ring. The use of the ring as a small module in a larger switch leads Rahnema to conclude that the first may be the appropriate choice, rather limiting the maximum possible size.

3.5 Cascaded Multicast Fabric Designs

Four designs which cascade, or prefix, a multicast fabric to a unicast ATM switch are summarised in this section. In all of these designs, the multicast cells are replicated in the multicast fabric (also known as the *copy network* [Bubenik89]) and are passed to the switch fabric as ordinary unicast packets, as in Figure 3.2.

One early example of this approach was the Starlite system [Huang84], in which the *destination* ports which were members of a multicast group caused multiple empty packets to be generated by the source, which were then filled from the source packet held in the multicast fabric before the switch fabric. For this approach to work, the source and destinations must be synchronised. This may be difficult to ensure across a broadband network, as packets suffer variable delays due to buffering in switches and multiplexing.

Note that the applied load as defined for multicast fabrics is the product of the arrival rate and the degree of multicast. It is normalised to ensure that the load on the whole multicast fabric is unity when the multicast fabric is producing a replicated cell at each output port per cell time.

3.5.1 Space Switched – St. Louis

Turner's original and subsequent papers [Turner86a, Turner88] described the construction of a broadcast packet switching network, which included a copy network based upon a self-routing Baseline network, with the ability to route and replicate the packets (cells) at each node in the network. The work was further extended by Bubenik and Turner [Bubenik89] to include simulations of both the multicast fabric and the composite switch fabric, observing some non-intuitive results showing that increasing the size of individual nodes in a switch fabric does not always improve the maximum throughput, mainly due to head-of-line blocking effects.

The multicast fabric described by Turner and implemented for Project Zeus [Cox91], is as illustrated in Figure 3.9, which shows the replication of a multicast packet with fanout seven. Since the multicast fabric is blocking, in that incident packets may contend for the same resources, each node is buffered with sufficient storage for one or two packets. A packet can pass through a node without being buffered at all if the desired link is available when the packet arrives (cut-through mode). The header of each packet contains the *fanout* field, as well as the broadcast channel number (VCI). The state of the buffer at each node is passed back to previous nodes ('busyback') and the entire network operates synchronously at a packet level. Byte-wide paths

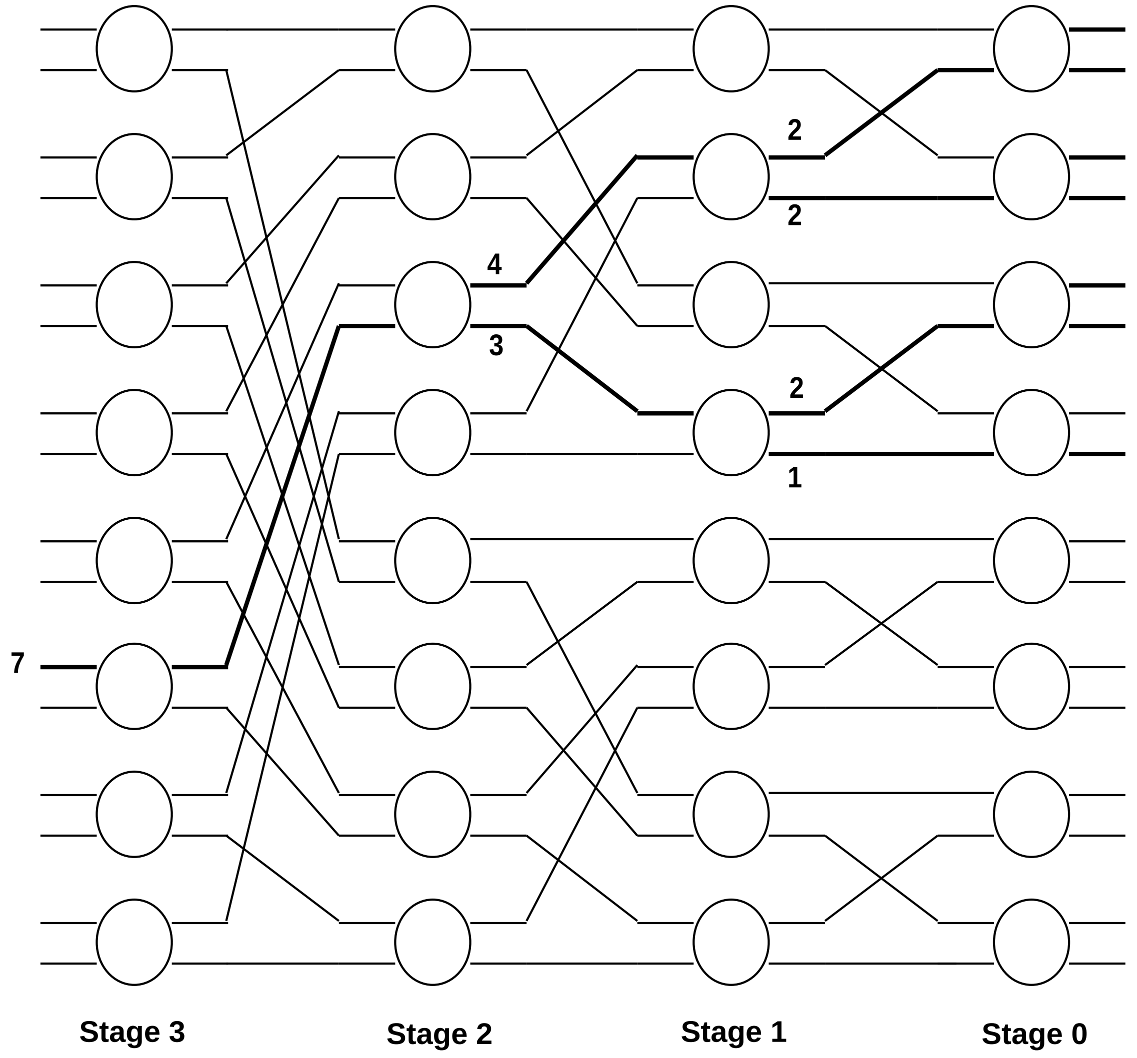


Figure 3.9: A 16×16 St. Louis Copy Network

are used throughout the switch and a clock rate of 25MHz is quoted. At each stage of transmission through the multicast fabric the following algorithm is performed:

Let f be the fanout and s be the stage number of the node in the switch, where stages are the columns of switching elements in a switch, labelled from right to left, starting at 0, as shown in Figure 3.9. At each node:

- Halve the current value of f, rounding up to an integer if f is odd.
- If $f > 2^s$, then request *both* output links and when they are available, simultaneously send the packet out on both.
- If $f < 2^s$, then route the packets alternately to each of the output links.

This algorithm delays the replication of packets until the last possible stage within the multicast fabric, thus limiting the potential for congestion.

There are two approaches to avoiding the effects of pathological traffic patterns, some of which could restrict throughput to $1/\sqrt{N}$, where N is the number of input ports. For lightly loaded switches in the local area, this is unlikely to prove much of a restriction. The first is to prefix a distribution network to the multicast fabric to ensure that communities of interest, that is contiguous groups of input ports contending for the same link in the central stage of switching, are distributed evenly across the multicast fabric. The second approach is to operate the multicast fabric at such a speed that none of the input ports ever have loads greater than 0.33 (a proof appears elsewhere [Bubenik89]). A simulated delay through the multicast fabric of approximately one packet (cell) time is quoted, with the fabric running at twice the rate of the input lines.

It should be noted that the outputs at which the replicated cells appear are chosen from a small set, since that the penultimate output port in the series of copies may be inactive, as in Figure 3.9. This means that all input port controllers in a switch must be able to map all potential incoming VCIs. There is also no guarantee that the cells will exit the multicast fabric at the same time or even at regularly spaced intervals, due to the internal buffering. This may in fact be an advantage in real networks, since it could avoid excessively large bulk departures from the switch to the rest of the network. However, jitter is likely to be increased by the breaking up of these potential clusters.

3.5.2 Space Switched – Lee

Lee [Lee88a] proposed a non-blocking, self-routing multicast fabric based upon a broadcast banyan network, producing packets with a constant latency. This was proposed as a possible extension to the experimental ATM switching system Sunshine developed at Bellcore [Lee88b, Giacopelli90]. Non-square switch fabrics, i.e. expansion switches for entertainment distribution, were also proposed in the same multicast fabric paper.

The *copy numbers* (fanout) are specified in the header of each packet and the replication process is achieved as follows:

- The copy numbers are synchronously presented to a running adder network which computes a range of contiguous output ports from which the required number of copies will emerge.
- This address range is then prefixed to the cell as a minimum and a maximum output address.
- All cells are synchronously applied to the multicast fabric. From the minimum and maximum address fields each 2×2 node may decide which output link to route the cell on, or whether to produce two copies. The decision to broadcast is based upon the result of a Boolean exclusive-or operation of the two bits in the minimum and maximum addresses which correspond to that stage of the copy fabric. If both bits are zero, then the cell is routed on the upper output port of the switch element, and if both are one it is routed on the

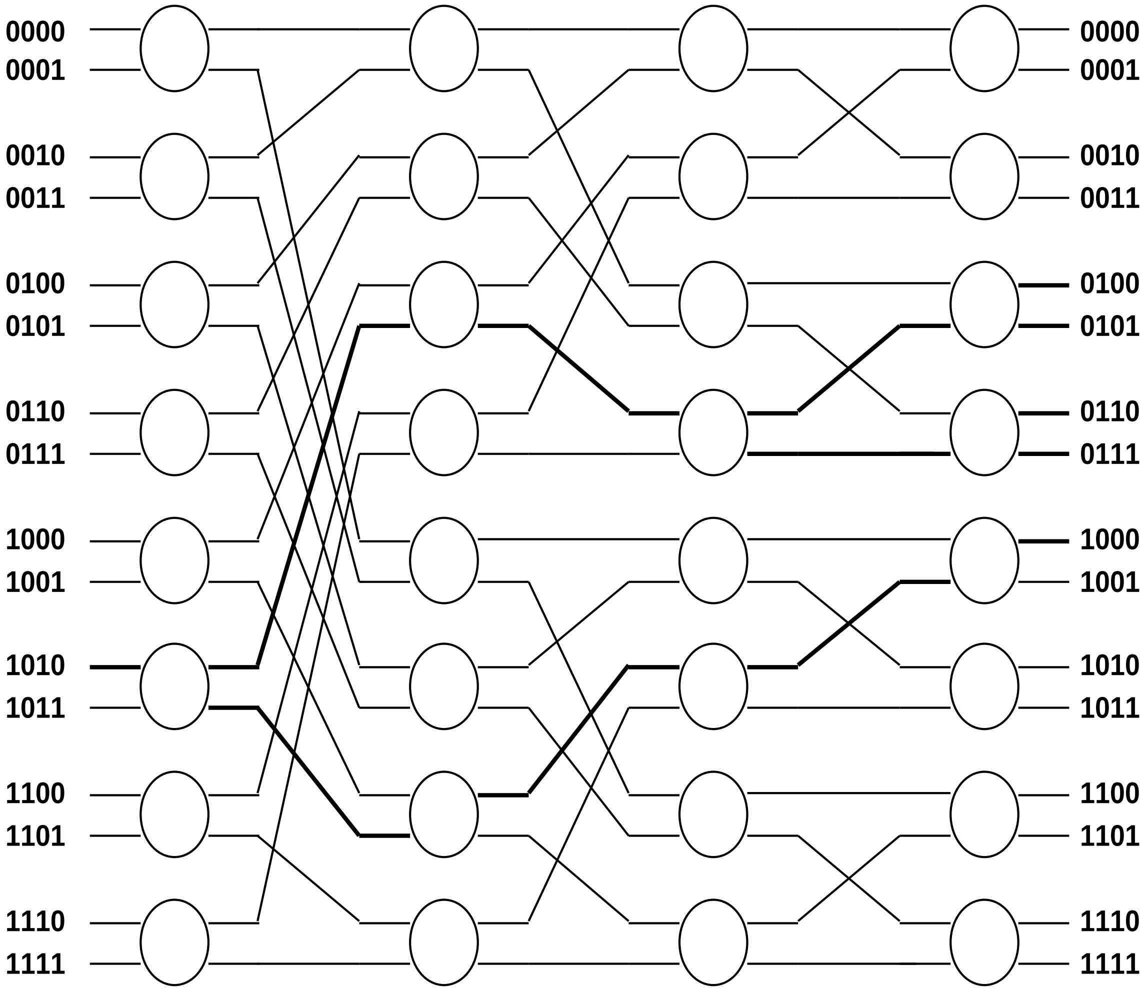


Figure 3.10: An Example of Lee's Boolean Interval Splitting Algorithm

> lower output port. The modified maximum and minimum addresses can also be simply computed, making the whole algorithm easy to implement in the hardware of each switching element.

Figure 3.10 shows an example of this boolean interval splitting algorithm. Suppose that the minimum and maximum output addresses are `0100` and `1000` respectively. The first comparison, working from the most significant bits, yields a one and so the cell is sent out on both outputs, the upper copy with the range of `0100` to `0111` and the lower one with `1000` as its maximum and minimum addresses. At the next stage, the next bit is compared for the range of addresses in the lower copy, and being all zero, the cell is forwarded on the upper output. The copy of the cell with just `1000` as its addresses is routed on the upper output. This process continues until the final stage is reached in the switch fabric.

In this multicast fabric all copies will exit the fabric at exactly the same time but the ports at which they leave are dependent upon the other traffic within the network. The running adder network mentioned above is in fact two anti-symmetrical running adder networks, used in alternate replication sessions in order to ensure a symmetrical distribution of the incident traffic across the switch fabric. Furthermore, for Lee's multicast fabric to work, active ports may not be interspersed with inactive ports, thus a concentration fabric (another dual running adder network and reverse banyan network, together with a sorting network if different cell priorities are required) is required prior to the multicast fabric. For the purposes of flow control and accounting, information such as the total number of copies requested at each replication session can be derived directly from the running adder.

As noted by Turner [Turner91], this design suffers from the fact that if two inputs have cells waiting for replication, one with a fanout of one (unicast) and one with a fanout of N, the unicast cell can prevent the other cell from being replicated if they enter during the same cycle. Thus the throughput of the copy fabric could be as little as one cell per cycle. Two solutions to this are proposed. Either restrict the fanouts to some value smaller than N or use an expansion copy network which can output more than N cells per cycle. The architecture proposed also requires large multicast remapping tables since all input port controllers must know about all possible multicast connections.

3.5.3 Space Switched – Lea

Lea [Lea88] considers the case where the number of outlets is assumed to be far greater than the number of inlets (e.g. broadband video distribution). He proposes a network which uses $O(6N \lg N)$ electronic crosspoints and $O(2N \lg N)$ optical ones. His approach is to prefix a *consecutive spreading* (CS) network to a routing network, which can be any rearrangeable non-blocking network such as a Beneš or Clos network. The purpose of the CS network is to spread the first k inlets across the first l outlets, where $l \geq k$, whilst preserving the order of the inlets. It is also assumed that all of the k inlets are active.

The topology of the CS network is as shown in Figure 3.11. Each node broadcasts the data that comes in from either its lower inlet or its upper inlet. The crucial observation is that if the cells are routed at each stage of the CS network according to the respective bits in the binary word which is the arithmetical difference between the outlet address and the inlet address, then the network is non-blocking. For example, the heavy line in Figure 3.11 traces the path followed by a cell from inlet 001 to outlet 110. At each stage, the relevant node is set to receive from its upper or lower inlet according to the respective bit in the difference word, i.e. $110 - 001 = 101$.

One of the necessary requirements for this algorithm to provide a non-blocking connection is that if inlet i requests a connection across the switch, then all the inlets j, $j < i$ must also be connected. Lea suggests that these unwanted inlets could be used as monitor stations in a real implementation. It should also be noted that the maximum number of copies which can be requested by inlet i is restricted to $N - i$, where N is the total number of inlets.

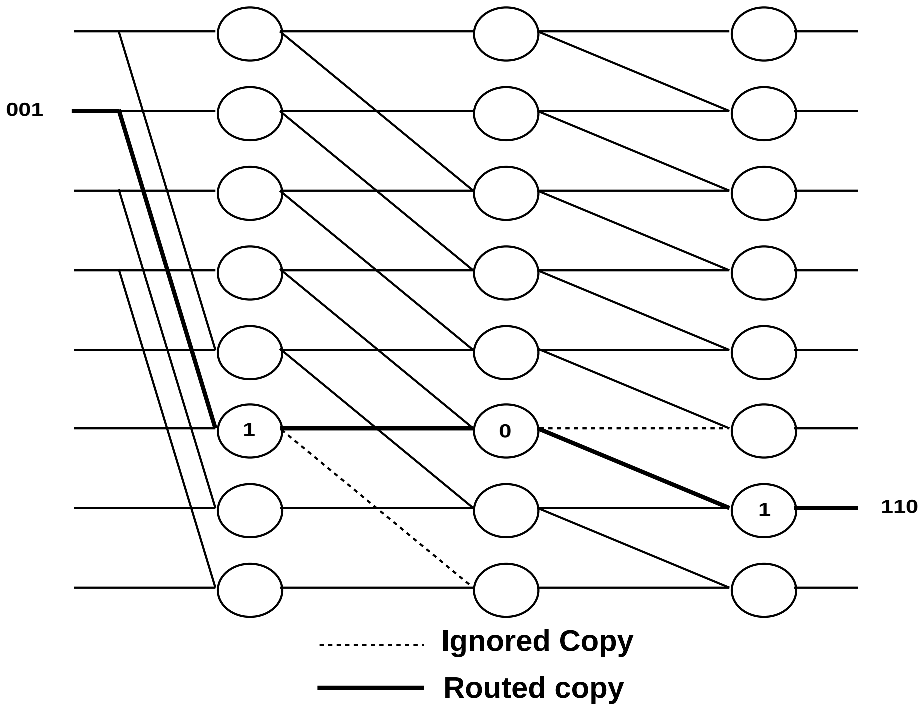


Figure 3.11: Topology of the Consecutive Spreading Network

3.5.4 Shared Memory – Kim

One example of multicast fabric design based upon the use of a shared memory is the design due to Kim *et al* [Kim92, Bianchini92]. The use of a buffer which is shared by all the inputs theoretically makes more efficient use of the space, compared to distributing the same amount of buffer space between each input. The design of an an N input copy network (multicast fabric) is shown in Figure 3.12

The operation of the network is as follows:

- Incoming cells are routed through the first omega network (an interconnection network with $O(N \lg N)$ switching elements which can be made non-blocking if there are no inactive input ports interspersed with active ones) to the bottom of the shared input packet queue.
- Cells waiting in the shared input packet queue are read out and replicated in the second omega network, N cells at a time. The cells arrive at the bottom of the multicast packet queue.
- Cells are read from the multicast packet queue and routed to their destination by the switching network. Packets which are blocked by this last stage are returned to the top of the multicast packet queue by two passes through the third omega network.

The packet queues are implemented as N banks of interleaved FIFOs so that up to N cells can be written and read simultaneously. The FIFOs are addressed in

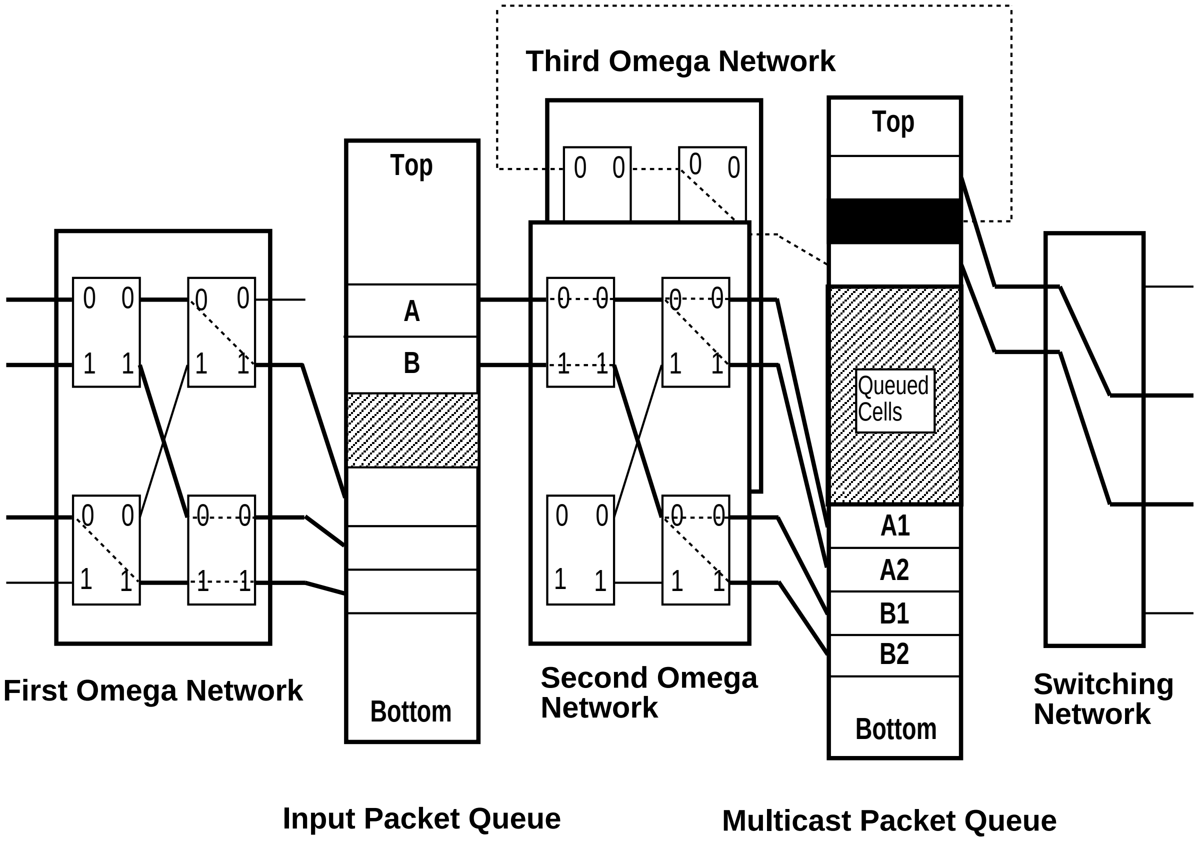


Figure 3.12: A Shared Memory Multicast Fabric

units of cells and partial cells are discarded and reported. Cell order is preserved by ensuring that consecutive addresses in the queues contain consecutive cells. The addresses for packet insertion are generated using multiple *fetch and add* operations which can be proven to preserve order. The problem of replicating cells within an omega network is solved in a non-blocking manner by ensuring that the active inputs are consecutive and permitting the desired outputs to be consecutive. The replication can then be performed in a similar fashion to the algorithm by Lee [Lee88a], by considering the maximum and minimum output addresses of the second omega network and routing or replicating according to the XOR of successive bits in the addresses (Section 3.5.2).

The maximum throughput for this design is known to be $(2 - \sqrt{2}) = 0.586$ due to Head of Line (HOL) blocking with Poisson arrivals and uniformly distributed traffic [Hui87]. Simulation results were performed using a geometric inter-arrival distribution with symmetric traffic across all the inputs and a truncated geometric fanout distribution for 16×16 omega networks. Such a distribution is typically modelled after the one proposed by Bubenik [Bubenik89] and this is described in Section 4.3. The results suggest that for a mean fanout of 8, a 64×64 shared memory multicast fabric can operate at near maximum throughput with space for less than ten cells in the packet queues for moderately low packet loss probabilities (10^{-7}).

As with the design due to Lee presented in Section 3.5.2, the output ports of the multicast fabric at which the copies of the multicast cells appear are dependent upon the other traffic in the network. This means again that all the input port controllers of the switching network must know about all possible multicast connections. The address generation, which can be a bottleneck in shared memory designs, was originally developed for the NYU Ultracomputer [Gottlieb83] and is performed in a distributed fashion.

3.5.5 Shared Medium – Slotted Ring

The use of a ring to perform the replication in a multicast fabric has been proposed twice in the literature. One proposal [Choi92] is unusual in that the multicast fabric is placed *after* the routing fabric. No indication is made of how access to the ring is controlled and the need to replicate and route the multicast cells with the ring seems to be a significant disadvantage of this design.

In the other, patented design [Newman89], which was first presented at the International Switching Symposium 1990 [Newman90], the intrinsic multicast capability of a slotted ring is used to construct a multicast fabric as shown in Figure 3.13. In the simplest configuration, there is one node on the ring for each output to the unicast switch fabric. The fanout is inserted into the header of the cell by the multicast controllers. Alternatively, an extra byte of multicast information (fanout, priority etc) can be prepended to the cell. The slots on the ring are the same length as the cell with the multicast information. Sufficient storage is located at each node to pipeline the necessary operations. The ring is closed by a FIFO to ensure that an integer number of slots exist on the ring.

When a cell arrives at a node, it is buffered until the next slot arrives on the ring. A slot is defined as empty if the fanout at the front of the slot is zero. If the slot is empty then the cell is written into that slot, otherwise the cell waits for the next slot. As a full slot passes each node, a copy of the cell is taken and transmitted to the unicast switch network, with no output buffering. Every time that this occurs the fanout at the front of the slot is decremented by one. The fanout present at the front of the cell can be used as the identifier for that copy of the cell. This copy identifier can be used by the input port controllers of the unicast switch fabric to help route the cell to its correct output port. This is discussed further in Section 7.2.

Each copy of the cell leaves the multicast fabric at successive output ports (independent of the distribution of traffic at other ports) and almost in parallel. There is a fixed delay of a few clock cycles between cells due to the delay in moving the slot head from node to node. This is represented in Figure 3.13 by the offset between each outgoing cell. Over the entire ring, copies appear at fixed points in time, all offset from each other by multiples of this inter-copy delay. The delay between the copy which emerges from the last node on the ring before the FIFO and the copy which emerges from the first node after the FIFO is a different delay. This delay depends upon how many bytes are buffered in the FIFO, which in turn depends upon the slot length and the number of nodes on the ring. For example, for a ring

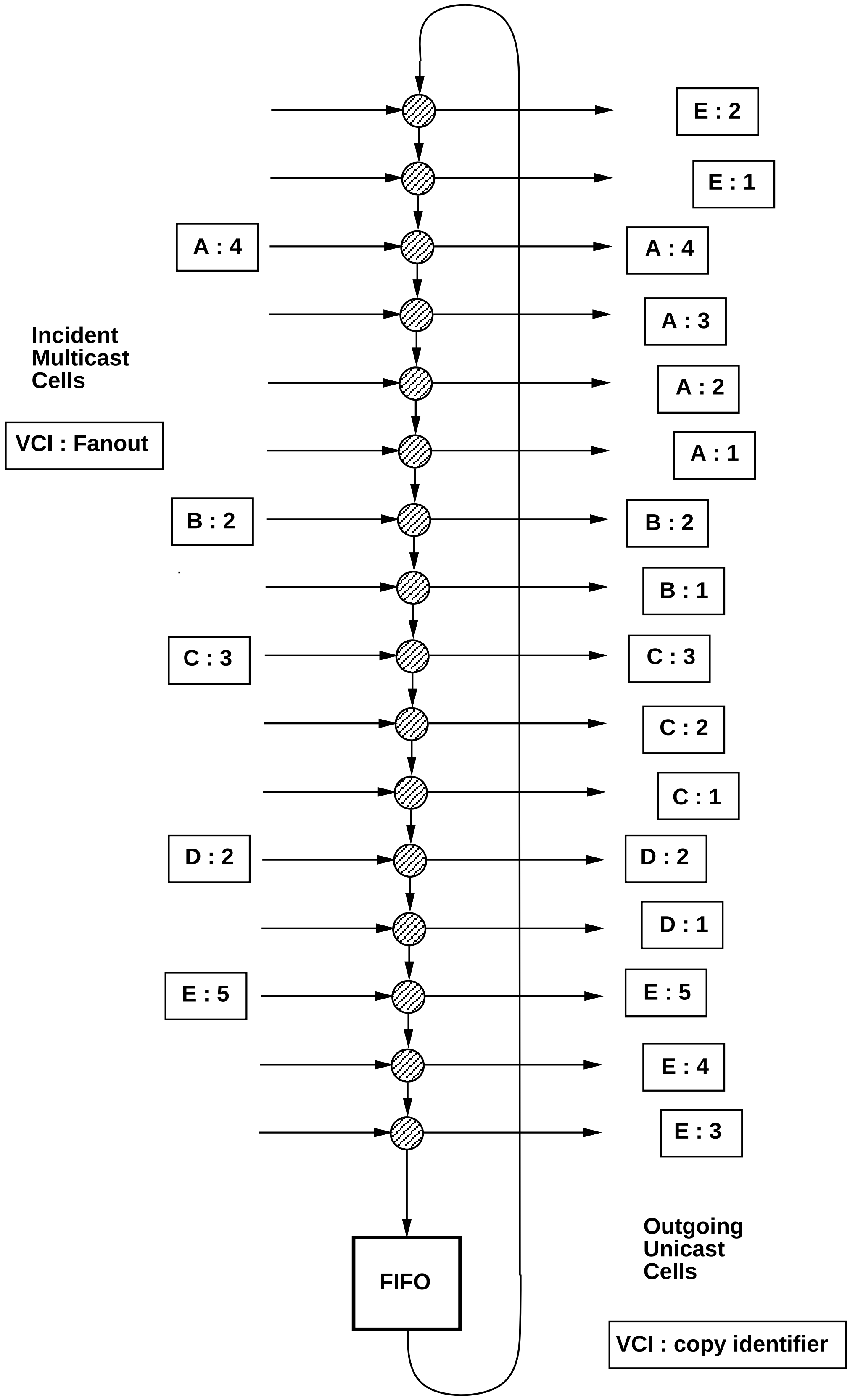


Figure 3.13: The Slotted Ring Multicast Fabric (at saturation)

with five bytes of storage per node and ten nodes and with the fanout in the header of a 53 byte cell, there would have to be three bytes of extra storage in the FIFO to make up one complete slot's worth of storage on the ring, which would give a comparable inter-copy delay. The maximum 'bottom-to-top' delay is just less than a cell-length. At the ring speeds discussed later, this delay is always less than 5μs.

The delay experienced by a cell at the head of its buffer before gaining access to the ring depends upon the distribution of the active inputs of the multicast fabric. A heavily loaded input port requesting large numbers of copies can deny nodes downstream of it fair access to the ring's bandwidth. This unfairness is one of the major issues addressed in Chapter 5.

3.6 Comparison of Designs

Cascaded *vs* Non-cascaded

In the wide area, it may be uneconomic to populate all switches, or indeed all inputs to a particular switch, with multicast capability. In the local area most switches will probably be small enough to be fully equipped and for ease of control and maintenance will be homogeneous. The use of cascaded multicast fabrics permits network designers to add multicast capability to switches and their inputs as necessary. The logical separation of replication from routing is also an advantage for control, since only the multicast controllers have to store information about the multicast groups. If multicast and unicast information is stored together, trade-offs may have to be made, some of which are difficult to estimate in advance. For example, it is rarely obvious how much multicast traffic a particular switch will experience, which affects the assignment of VCI space to multicast and unicast VCIs. For these reasons, most ATM multicast switches which are being implemented in research initiatives use cascaded multicast fabrics.

Three general aspects of the designs have been chosen to permit a surface comparison sufficient to justify the further investigation of the slotted ring multicast fabric.

Control Complexity

The complexity of controlling any of the above cascaded designs depends upon two aspects of where cells emerge from the multicast fabric. Firstly, if the set of output ports is dependent upon what other traffic is being replicated at the same time, then all of the input port controllers, before the switch fabric must contain the remapping information for all of the multicast groups that the switch is aware of. No possibility exists for restricting groups of inputs to a set of output ports. This is particularly wasteful if a multicast connection with a small fanout has its fanout increased by one and every other port controller on the switch has to be informed about the modification. It also makes connection set-up slower since all port controllers have to have their VCI remapping tables altered, instead of just those which will see the copies of the multicast cell. Switch designs whose set of outputs for a multicast connection is dependent on the input traffic pattern include those due to Turner, Lee, Lea and Kim *et al.* Shared medium designs such as the Olivetti switch and the slotted ring multicast fabric are independent of traffic patterns in this sense.

Growth Complexity

Cascaded switches such as the one discussed in Sections 3.5.1 to 3.5.3 are further examined by Hui [Hui90a, pp.85–111], where he shows that the crosspoint complexity of the designs due to Turner and Lee is $O(N \lg N)$, where N is the number of input/output ports in a square switch. Similarly, the switch fabric design due to Lea has an (electronic) crosspoint count which is $O(6N \lg N)$. The interconnection networks of the design due to Kim *et al* also scale in an $O(N \lg N)$ manner.

In contrast, the non-cascaded multicast switches of Sections 3.4.1 and 3.4.2, which copy and route in the same network, have higher crosspoint growth rates, typically $O(N_1^{3/4} N_2)$, where N_1 and N_2 are the number of inputs and outputs respectively. One of the earliest multicast interconnection networks, cited by Hui [Hui90a, pp.106–108], is that due to Pippenger [Pippenger74], which has a growth rate of $O(N\lg^2 N)$, which is better for square switches larger than 16×16, but still not as low as the growth rate for cascaded switch designs. Shared memory and medium non-cascaded switch designs tend to scale linearly in most aspects, except for limitations due to problems with memory access speeds and high speed TDM buses.

In all designs, modularity is a highly desirable feature, to permit switches to be altered in size as networks grow around them. Some of the designs based upon interconnection networks require the number of inputs and outputs to be powers of two in order to minimise the number of connections. This is not a problem for small switches, but for larger ones the incremental steps become rather large.

The cascaded ring-based approach of Section 3.5.5 scales in a linear fashion with the number of input ports, but suffers from technological limitations at relatively small switch sizes due to a number of factors. From the experience of implementing the design as an experimental multicast fabric, described in Chapter 6, synchronisation of nodes to the ring becomes more difficult and so a more complicated form of clock distribution becomes necessary. With a large fanout, the delay from the first copy of the cell to the last copy grows linearly with N. Eventually this will come to resemble the delay which would have existed if a small number of copies had been made serially. The embedding of the fanout in the header of the cell also limits the simple design to fanouts of less than 2^b, where b is the width of the ring (a byte). However, multiple bytes could be used in a slightly more complex design, so this is not really a major limitation. These issues are revisited for the slotted ring multicast fabric in Section 6.6

Implementation Complexity

Designs which use complex space-switched interconnection networks are prone to difficult implementations, due to the large number of connections necessary between switching elements. This is mainly a problem when connecting ICs together on boards and across backplanes. As chip manufacturing technology improves, larger switch fabrics of this nature can be fitted on one IC. One interesting possibility is the use of wafer-scale technology to build large switch fabrics, but the problems of connecting a large number of inputs and outputs still remain. Current single-chip limits seem to be approximately 32×32 inputs and outputs with $1\mu m$ CMOS technology.

Turner [Turner88] considers the scaling of his design in general terms and proposes an efficient 3-D packaging method with four inputs and outputs to a switch fabric chip and provides estimates for the complexity of a 64×64 switch constructed from these chips. His design has been used as the basis of the copy network in Project Zeus [Cox91], a metropolitan area ATM multimedia network in the locality of Washington University in St. Louis. A 16×16 copy network has been constructed on a single card using three different ASICs and is currently running at line speeds of 100Mbit/s. It should be noted that some of the very early switch designs based on interconnection networks did not consider implementation details or factors such as scaling and modularity. Later designs such as those due to Turner and Lee addressed these problems from the start. In practice it still appears to be IC interconnection complexity which limits the size of a practical switch on a chip or switch chips on a card.

Neither Richards and Hwang nor Lea give specific details of proposed implementations, though Lea does suggest that the importance of the number of (electronic) crosspoints is being overtaken by the need for a regular topology and modular construction. Rahnema suggests that with current CMOS technology and optical distribution of a 50MHz clock, a 1024×1024 T1 switch with a 32-bit wide data path should be possible, but does not describe this in detail.

In contrast, the Olivetti switch is under development at the moment and should fit onto an 11"×11" board, using 100Mb/s TAXI ICs [TAXI89] for interfaces. An ARM600 RISC microprocessor will be used for VCI remapping and asynchronous FIFOs will be used throughout, partly for their ease of configuration.

At the other end of the scale of implementation complexity from the space-switched interconnection networks lies the slotted ring multicast fabric described in Section 3.5.5. For small-scale multicast switches, this design is of interest for several reasons:

- The sets of output ports from which copies emerge are deterministic and independent of incoming traffic patterns.
- The design is simple and does not require large numbers of complex interconnection networks. Copies are generated with only a small delay between each copy. Implementation should be feasible with inexpensive technology.
- Both the design and its underlying analytical model of a cyclic server with limited service and irregular server movements have not been previously investigated and may be of interest to future designers of ATM switches.

The slotted ring multicast fabric has been implemented to provide multicast for the switches used in Project Fairisle [Leslie91], initially as a 4×4 fabric, using Xilinx programmable gate arrays and a four layer PCB, and has been confirmed as simple to construct. This is discussed further in Chapter 6.

3.7 Summary

The design issues for ATM switches which replicate and route cells in parallel have been considered. Examples from each class of multicast switch have been given and brief comparisons of the designs according to their control, growth and implementation complexities have been presented, together with a more general list of design issues for multicast ATM switches.

More specifically, the use of a slotted ring to construct a multicast fabric has been described, and it is this design which is investigated further by simulation in the next chapter.

4 Simulations of the Multicast Fabric

In this chapter the slotted ring multicast fabric design described in the previous chapter is investigated by simulation. The results of simulations of the basic design are presented below. Descriptions of some possible enhancements for the design are described in the next chapter, together with simulation results of the enhancements.

The performance of the ring based multicast fabric was investigated using a discrete time-based simulation model as opposed to an event-based model, since important events such as cells gaining access to the ring were usually close together in time anyway. The simulation programs were originally written in Modula2, which has the advantages of modularisation and formal interface checking [Page92]. They were later rewritten in C to improve the run-time of long simulations, which required more floating point operations for source modelling and extended results statistics. Multiple small simulations were developed which made debugging and testing much easier.

Note that the terms 'port' and 'node' are used interchangeably throughout this chapter to refer to the points on the ring where access occurs and cells are replicated.

4.1 Basic Traffic Models

The first traffic arrival model which was used was a Bernoulli process, modelling slotted traffic arriving with probabilities directly dependent upon the applied load. The applied load was normalised to the maximum throughput of the design and the mean fanout, so that a load of 1.0 indicated that N cells could be appear at the output ports per revolution of the ring in a symmetrically loaded N node multicast fabric. For an applied load, $0 \leq L < 1$, and a mean fanout $\overline{f}$, the probability of an inter-arrival time of x was

$$p(x) = \begin{cases} \frac{L}{\overline{f}}(1 - \frac{L}{\overline{f}})^x & \text{if } x \in \{0, 1, \ldots\} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

and hence the mean inter-arrival time was $\overline{f}/L$. All inter-arrival times were multiples of cell length times, i.e. each arrival slot was one cell long. The first set of simulations using this geometric traffic arrival distribution assumed that the traffic was distributed equally across all the input ports. Asymmetrical geometric distributions were simulated later, and the results are presented in Section 4.5.

The second traffic arrival model which was simulated was one with cells arriving in contiguous blocks. This was felt to more closely model the behaviour seen in networks with real traffic, where clusters of cells occur from the segmentation of packets or video frames and clusters of packets occur due to queueing disciplines which serve more than one request from a queue. This bulk arrival model had a similar geometric inter-arrival time distribution but the arrival rate, $L/\overline{f}$, was further scaled by k, the number of cells in a block, to maintain a constant overall traffic arrival rate. Thus the probability of an inter-arrival time of x was now

$$p(x) = \begin{cases} \frac{L}{k\overline{f}}(1 - \frac{L}{k\overline{f}})^x & \text{if } x \in \{0, 1, \ldots\} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

It was felt that the two traffic models described above were sufficient to gain insight into the likely behaviour of a multicast fabric based upon a slotted ring. One further model, a truncated geometric distribution, was used to investigate the effects of mixing multicast and unicast traffic. This is described further in Section 4.3. Other more accurate and complex models of the characteristics of different classes of multimedia traffic may be found elsewhere [Bae91].

4.2 Details of Simulations

The granularity of time in the simulations was the time the head of the slot took to move from one node to another. It was assumed that sufficient storage existed in the ring and FIFO for one cell-length slot on the ring. In the initial simulations the phase difference between the arrival slots and the head of the slot was constant and zero. This meant that the time spent waiting by a cell at the head of the queue was always an integer multiple of the cell length time. In later simulations a random phase difference between the slots arriving at a node and the head of the slot on the ring was chosen at initialisation and fixed for the simulation. The relationship between slots arriving at different nodes was also random and fixed. The effect of the random differences was to add an extra delay of half a cell length to the mean delay experienced by each cell. In the simulations presented here, this delay has been eliminated from the results by synchronising the slot arrivals to the ring as in the earlier simulations, with the aim of showing the performance characteristics more clearly.

In reality, there is a delay between the arrival of the header of a cell and the time the cell can be replicated in this design since a cell has to be completely buffered before it is available for transmission. Since this delay is independent of the multicast fabric, it was assumed to be fixed, meaning that the simulation model operated in cut-through mode.

The fanout (number of copies) was either fixed at integer values co-prime to the number of nodes, or chosen from a uniform distribution of random integers in the range $(1, N-1)$, with a mean of $N/2$. The reason why co-prime values were chosen was that if the number of nodes on the ring and the most common value of the fanout are multiples, one of the other, then patterns of access can build up around the ring. For instance, a ring with an even number of nodes and with most of the sources requesting even fanouts could deny access to the odd numbered nodes on the ring. This could occur if the even numbered nodes gained first access to the ring, and continued to do so on every subsequent revolution, blocking out the other nodes. This effect is only noticeable at high loads and is not a serious problem for practical versions of the slotted ring multicast fabric.

The fanouts in successive cells were assumed to be independent, which is only true in reality if a sufficiently large number of traffic sources are multiplexed onto an input port. All delay results are normalised to the time taken for a slot to completely pass a single node, i.e. the 'cell length' or 'cell time'. The speed of rotation of the ring was assumed to be constant and the same as the underlying slot arrival rate at the input ports. Infinite buffers were also assumed, so that no cells were lost due to buffer overflow.

Each simulation was allowed to reach stability and then run for a total of 100,000 transmitted source cells, whilst observing Student t-distribution confidence limits based upon multiple runs. Consideration of the action of the simulation at high loads (greater than about 0.99) suggests that the duration of a 'busy cycle', the time between two instances when a node has no cells queued, becomes excessive. The difficulties of completing sufficient busy cycles in reasonable real-time CPU time meant that the high load results should be considered to be only guides to eventual asymptotes. The theoretical performance of the multicast fabric at 'heavy' loads, i.e. near saturated, is derived elsewhere [Bean92].

Initial simulations confirmed that the delay performance of the fabric is independent of the number of slots (given that it is sufficient to reach all the nodes). This can be deduced by considering that increasing the number of slots on the ring does not decrease the time between slot-head arrivals if the ring speed remains constant. This was also confirmed later by the full analysis presented elsewhere [Bean93].

A 16×16 fabric was chosen as a suitable size for later comparison to reality (see comments on scalability in Section 6.6). It was also hoped to try to avoid any possible edge conditions which might be associated with very small sizes of switch fabric.

4.3 Simulation Results

The mean delay from the arrival of the head of a cell at a node until the cell gains access to the ring for transmission is shown in Figure 4.1 for fixed fanouts of 3 and 9, and in the graph on the left in Figure 4.2 for a fixed fanout of 15. The 99th percentile curve of the same delay and fanout are also shown in the figures, together with the results of the analysis summarised in Appendix A.

Results for the (more realistic) uniformly distributed fanout, with mean fanout 8, are shown in the graph on the right of Figure 4.2. The uniform fanout results are approximately comparable in value to those of a fixed fanout of 5, not 8 as might be expected. One explanation for this might be that the uniform fanout distribution produces lower mean delays per cell because a variation in the fanouts means that

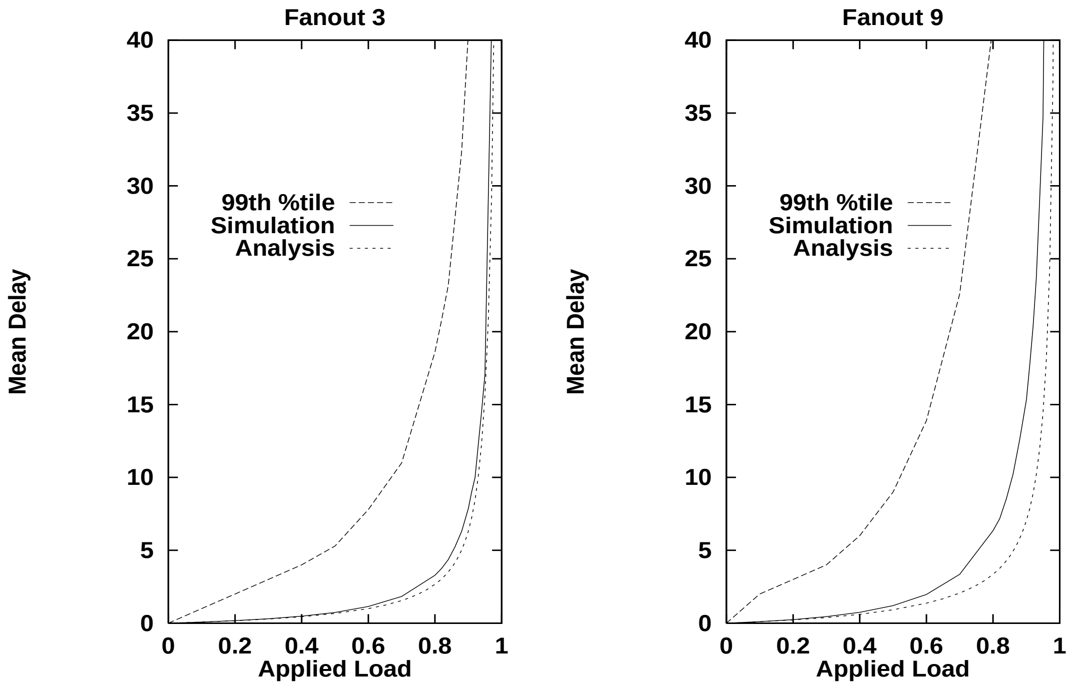


Figure 4.1: Mean and 99th Percentile Delay for Fixed Fanouts

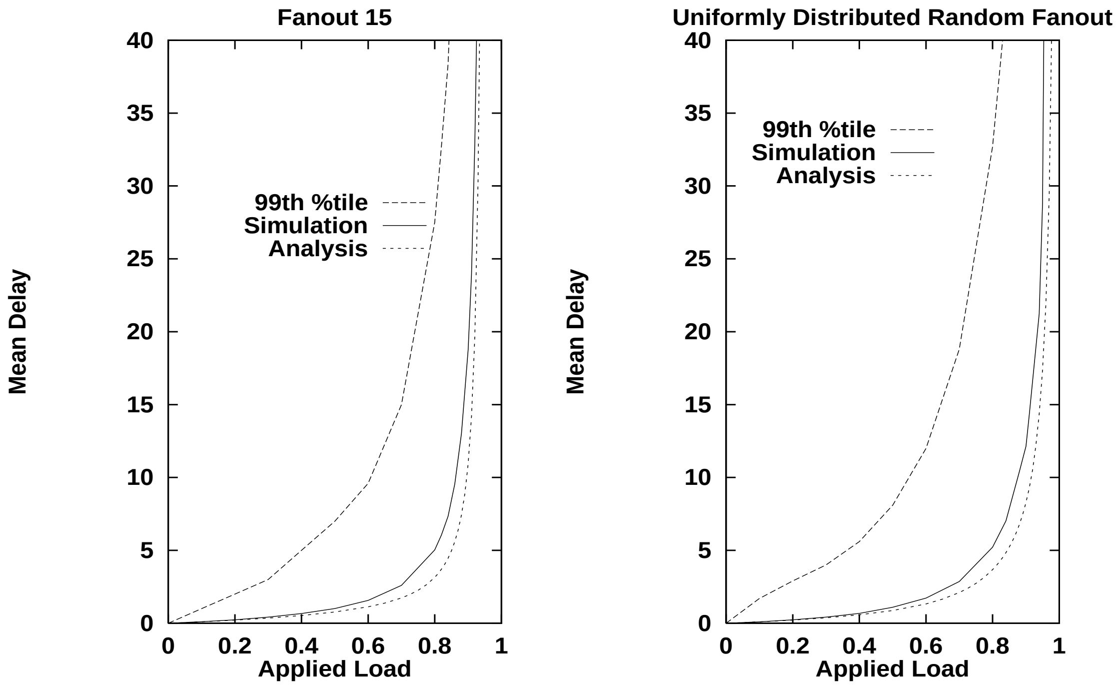


Figure 4.2: Mean Delay for Fixed and Uniformly Distributed Fanouts

smaller fanout cells can gain access to the ring where larger fixed fanout cells might not have been able to do so in a similar situation.

A preliminary series of multicast fabric simulations, which were reported at ISS'90 [Newman90], used a geometric distribution of fanout after Bubenik and Turner [Bubenik89], where fanouts were chosen from the range $(2, N)$ and specifically excluded unicast traffic. The results were approximately the same as those shown in Figures 4.1 and 4.2.

As an approximate indication of what might be expected from a real multicast fabric, it may be seen from the graph on the right of Figure 4.2 that even for a relatively high applied load of 0.7 the mean delay is 2.9 cell lengths and the 99th percentile delay is 18.8 cell lengths. These figures correspond to 7.9μs and 51.3μs at a line speed of 155.52Mbit/s. This is comparable to the delay across a unicast switch fabric operating at a similar load [Bubenik89].

Different Multicast Fabric Sizes

Figure 4.3 shows how the size of the multicast fabric, that is, the number of nodes on the ring, affects the delays experienced by cells traversing it. The solid lines indicate the mean delay and the dotted lines indicate the 99th percentile of the delay. The results shown are for a fixed fanout of 4, being the largest possible fanout for the smallest size of fabric which was simulated. As may be seen, as the size of the fabric increases, the delays remain approximately constant for the same load, except for a fabric of size 4, whose delays are a little lower. This may be due to the small size of the fabric exacerbating the effects of the fanout and size being the same. Table B.2 in Appendix B contains the same results as shown in the graph and confirms that the results for the other sizes are all close to each other.

Other results [Newman90] have also confirmed that the mean delay is almost independent of the fabric size, given that the mean fanout remains constant. This is further supported by the fact that the number of nodes does not appear in theorem A.3.2 of the approximate analysis summarised in Appendix A.

Mixed Multicast and Unicast Traffic

Figure 4.4 shows how the performance of the multicast fabric changes as the percentage of multicast traffic increases. These results are taken from an earlier set of simulations [Newman90] and are for a 64 node multicast fabric with a geometrically distributed mean fanout of 8. When all the traffic is unicast (0%), the delay is zero, since a cell never finds a slot full and hence always gains access to the ring. As the percentage of multicast traffic increases, the delay curve rises until 100% of the traffic is multicast. It can be seen that the slotted ring multicast fabric is sensitive to small amounts of multicast traffic and the maximum value is rapidly approached. However, for a low percentages of multicast traffic, e.g. 20%, the results are less than half of the maximum value.

The fanout distribution used for this graph is the geometric distribution mentioned in Section 4.1, where multicast cells have a fanout chosen from a truncated geometric distribution which permits the fanout to range from 2 to N with a parameter p. The probability that the fanout is k is given by

$$\Pr\ (\text{fanout} = k) = \begin{cases} p(1-p)^{k-2} & 2 \leq k < N \\ (1-p)^{N-2} & k = N \end{cases}$$

where $0 \leq p \leq 1$ and N is the size of the multicast fabric. The mean fanout obtained from this distribution is given by

$$E\ (\text{fanout}) = \begin{cases} (1/p)[1-(1-p)^{N-1}] + 1 & 0 < p \leq 1 \\ N & p = 0 \end{cases}$$

Bulk Arrivals

Figures 4.5 and 4.6 show the effect of altering the block size k, the number of cells in a contiguous block. The block size was fixed at k and the incoming traffic was distributed symmetrically across the multicast fabric. The results are for a 16×16 multicast fabric with a fixed fanout of 15 in the graphs on the left and a uniformly distributed fanout in the graphs on the right. The 99th percentile delay is included as an indication of the tail end of the delay distribution and hence a useful indicator of likely queue sizes.

As might be expected, the delay of the multicast fabric increases as the block size increases and the rate of increase is greater at higher loads than at lower. The uniform fanout results are closer to the fixed fanout ones than was the case in Figure 4.2, at least for loads less than about 0.85. This may be due to the number of cells which are waiting at the HOL of a node's queue increasing, which then means that proportionately more large fanout cells will be waiting (for uniform fanout traffic). This in turn means that fewer of the small fanout cells waiting at other nodes will be able to use the unused sections of the ring as they could do before. There may also be a factor due to the fixed fanout being one less than the number of nodes in the ring, which means that when large numbers of the nodes have a cell waiting, a pattern of nodes which gain access will appear, as the slot-head becomes free one node earlier on each revolution of the ring. This effect could reduce the delay compared to that for a uniform fanout.

Packet sizes in real networks are often widely variable. However, as an example, Ethernet packets have an absolute length of 1514 bytes, which is often limited by network interfaces to about 1400 bytes for local area networks and reduced still further to 576 bytes for internetworking. These values correspond to 32, 30 and 12 cells respectively, indicating that the range of block sizes simulated is realistic.

4.4 Asymmetric Traffic Arrival Patterns

As noted in Chapter 3, one of the problems of the slotted ring multicast fabric design is that a single heavily loaded input port may deny access to the ring for other more lightly loaded ports, i.e. access to the ring is unfair. In the long term this problem only exists if the arrival pattern of the traffic is asymmetrically distributed across the input ports and the traffic is arriving at high rates.

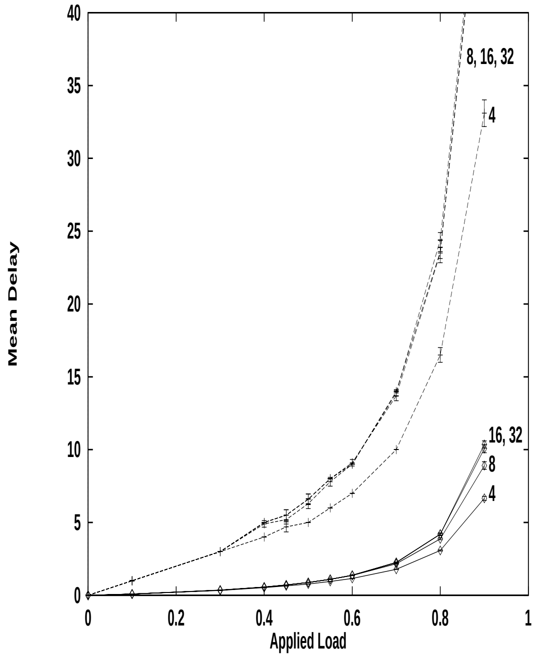


Figure 4.3: Mean Delay for Different Multicast Fabric Sizes with a Fixed Fanout of 4

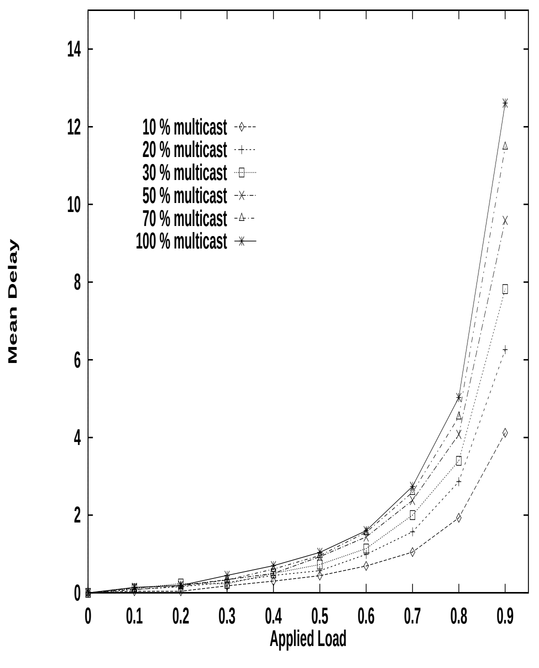


Figure 4.4: Mean Delay for Various Percentages of Multicast Traffic with a Geometrically Distributed Fanout

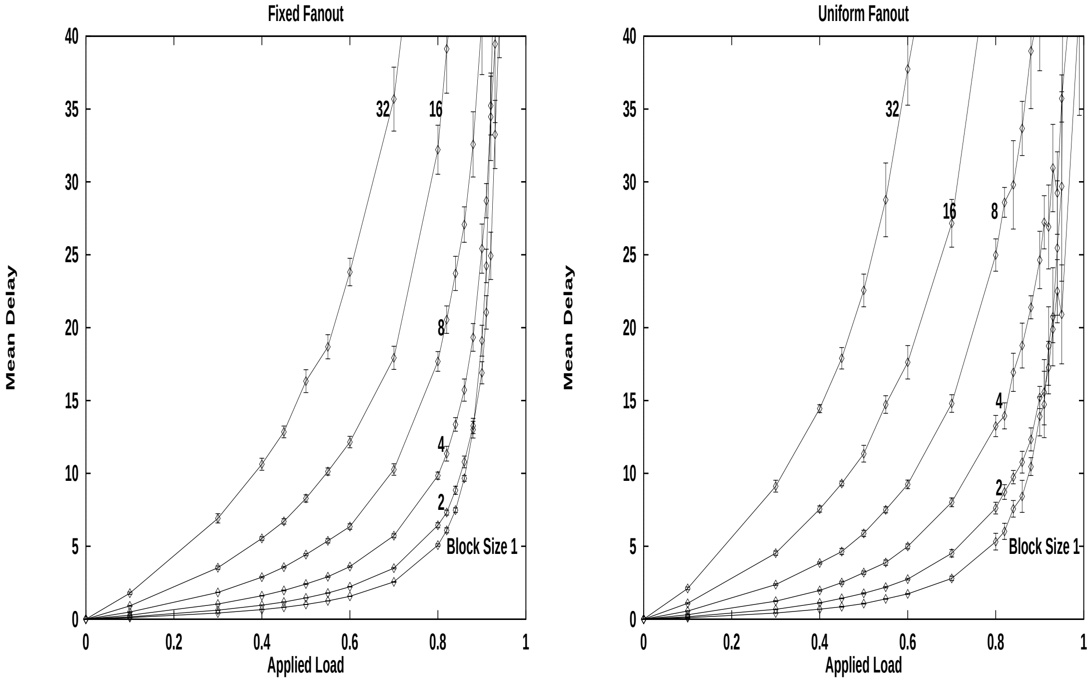


Figure 4.5: Mean Delay for Bulk Arrivals with Fixed and Uniformly Distributed Fanouts

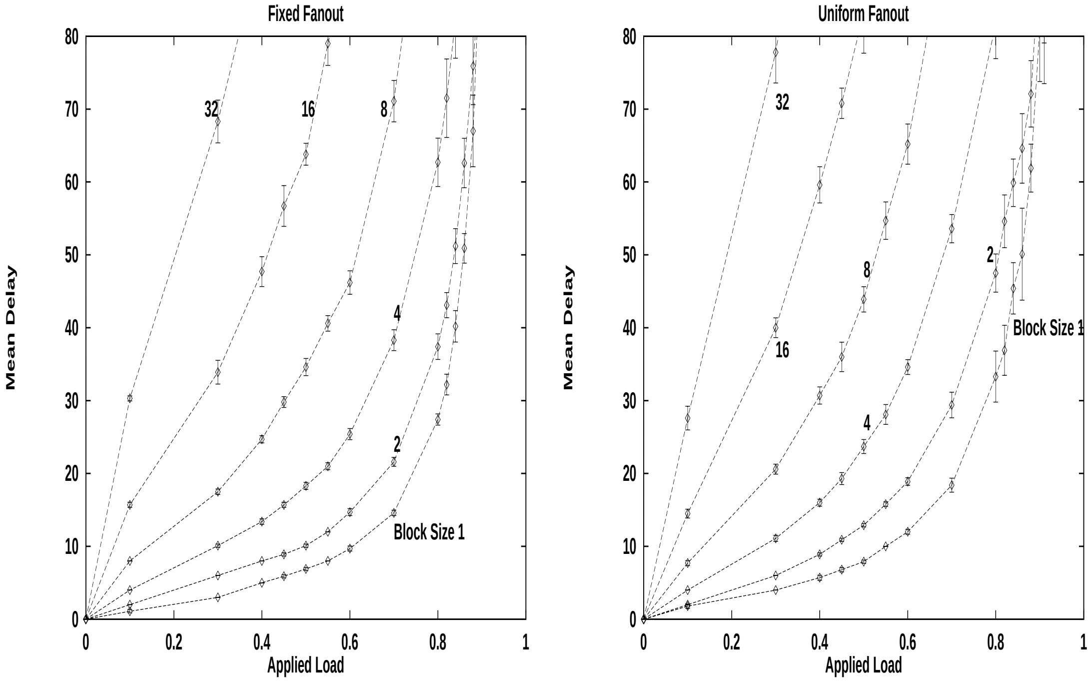


Figure 4.6: 99th Percentile Delay for Bulk Arrivals with Fixed and Uniformly Distributed Fanouts

Definition of Fairness

The definition of fairness used here is simple. Each port should suffer approximately equal mean delays when sending cells through the multicast fabric, independent of the relative position of the incoming traffic on the fabric. Other parameters to be considered in a definition of fairness include the input buffer queue size (and hence the Cell Loss Ratio at each node), the variance of the delay (jitter) and the 99th percentile of the delay as an approximate upper bound on the performance of the fabric. The measure of fairness used in the simulations described below is the difference between the mean delay at the port with the maximum mean delay and the mean delay at the port with the minimum mean delay. Since the purpose of the measure was to gain an insight into how asymmetric traffic affected the mean delay, a difference rather than a ratio was chosen for clarity.

The performance of the multicast fabric with asymmetrical traffic arrival patterns was investigated as a necessary basis for comparing the different methods of controlling access to the ring which are presented in the next chapter.

Asymmetric Traffic Model

The total incoming slotted traffic was assigned to each input port according to the bias parameter α, $(0 \leq \alpha \leq 1)$. The lightly laden ports all had their incoming traffic scaled by a factor $(1-\alpha)$ and a single heavily loaded port's arrival rate was scaled by a factor $(1+\alpha(N-1))$, where N was the number of input ports. This distribution is referred to as the *point source distribution*. When $\alpha = 0$, the traffic was uniformly distributed across the multicast fabric and when $\alpha = 1.0$, all the traffic entered the fabric through a single port. The uniform fanout distribution was used for all the traffic arriving on all of the input ports.

4.5 Asymmetric Simulation Results

Figure 4.7 shows the mean delay at each port for a 16 node fabric with the uniformly distributed random fanout (mean 8) mentioned in Section 4.1, at an applied load of 0.6. The value of 0.6 was chosen to emphasise the unfairness of the multicast fabric at high loads, since the effect is far less noticeable at lower loads.

It can be seen that the effect of a single heavily loaded port (port 0) is to increase the delay at certain downstream ports as might be expected. As the number of cells emerging from a node decreases, so does the delay at that node. The crossover point comes after the ports are further away from the heavily loaded port than the mean fanout.

The overall effect of varying α and the applied load can be seen in the graph on the left in Figure 4.8. As α increases, the maximum throughput reaches a plateau at decreasing values until at $\alpha = 1$ the maximum throughput is equal to $\overline{f}/N$, since only $\overline{f}$ copies are being produced on per slot passing N nodes. Analytical results elsewhere [Bean93] confirm the results shown here and show that for $L < 1/(1+\alpha)$ the throughput is equal to the applied load. This is a consequence of a necessary

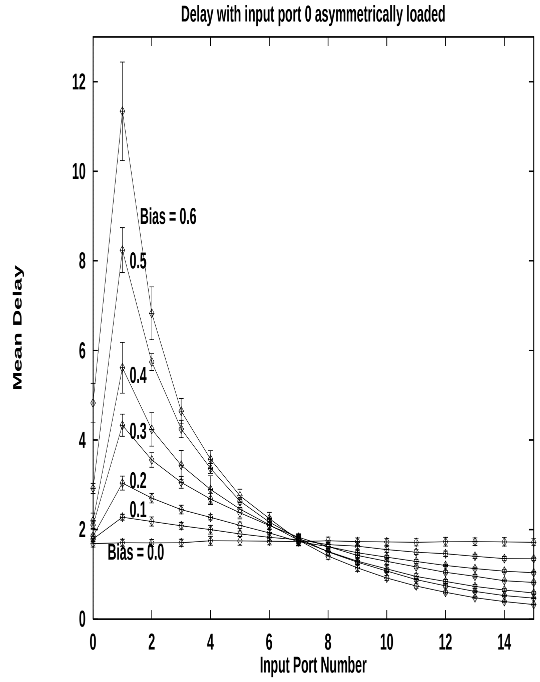


Figure 4.7: Mean Delay at Different Input Ports

condition for the stability of the fabric. Another consequence of the point source traffic arrival model is that, for applied loads less than $\overline{f}/N$, values of α up to $\alpha = 1.0$ will not cause instability in the ring.

The graph on the right in Figure 4.8 shows the difference between the mean delay (averaged over ten runs) experienced by cells entering the multicast fabric at the most delayed port and the least delayed port, for example ports 1 and 15 in Figure 4.7. This value is a useful comparative measure of the fairness of the multicast fabric. Whilst α is low, especially at low loads, the difference in the delays increases slowly. When $\alpha = 0$, the increase is due solely to fluctuations in the simulation results consequent upon finite simulation time and should be zero. For lightly loaded switches (≤ 0.4), the problems of unfairness, at least for the point source asymmetrical traffic model, do not appear to be excessively large.

4.6 Summary

The performance of the slotted ring multicast fabric has been investigated using several simulation models. Initial results for symmetrical traffic confirmed that the mean delay across the multicast fabric is comparable to that of other elements of an ATM switch, i.e. a few cell times , which is equivalent to tens of μs at the proposed speeds of operation.

Further results for asymmetrical traffic arrival patterns investigated the problems of unfairness with simple access to the ring. These results suggest that for low loads

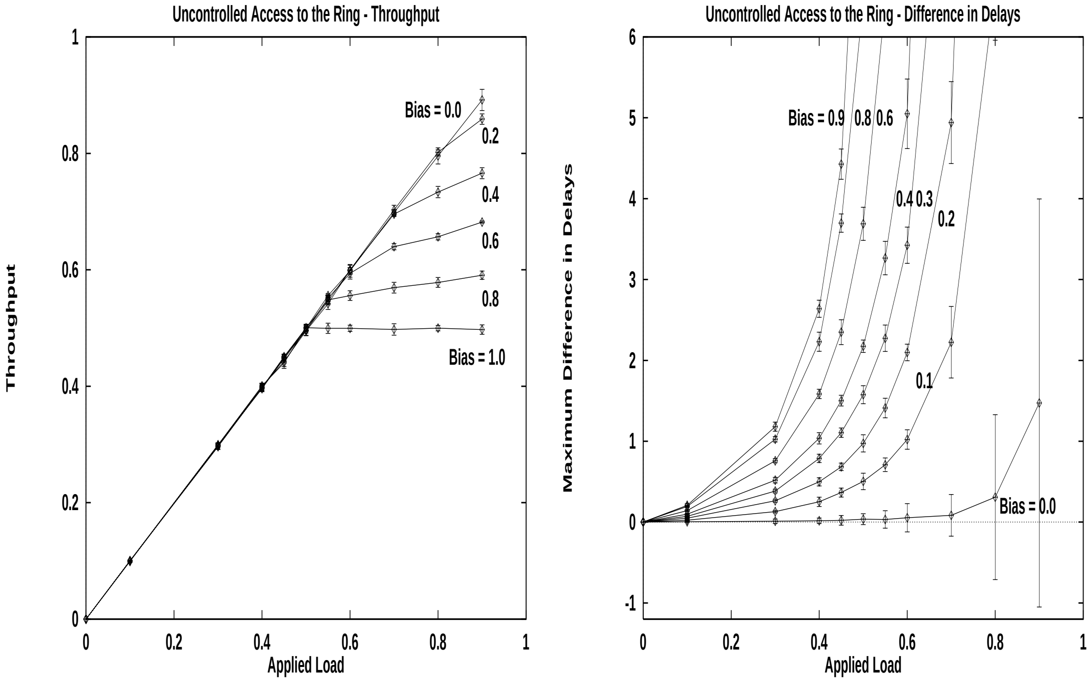


Figure 4.8: Unfairness in the Multicast Fabric

and small percentages of multicast, such as might be expected in an ATM LAN, the problems of unfairness in the slotted ring multicast fabric do not appear to be large, but at higher loads significant differences occur between the multicast fabric performance experienced by different nodes.

5 Enhancements for the Multicast Fabric

One problem with the slotted ring multicast fabric design presented in Chapter 3 is that a single highly loaded input port can deny downstream input ports their fair share of the ring's bandwidth. The effect of this asymmetrical loading on the performance of the multicast fabric was investigated in the previous chapter and it was found that the effect is significant when either the asymmetry is great or the applied load is high. Other factors which may cause unfairness are if the mean fanout is large and the multicast fabric is behaving as a (near) broadcast fabric, or if pathological values of fanout are used at high loads, such as values which are factors of the number of nodes on the ring.

5.1 Improving Fairness of Access

Even when one input port is heavily loaded, bandwidth on the ring is usually still available (unless the node is continually multicasting at the maximum fanout). The other nodes can make use of this bandwidth either by redirecting their traffic to enter the ring at another node, possibly by using a distribution network in front of the multicast fabric and recirculating blocked cells (see Section 5.5), or else the heavily loaded port can be denied access to the ring to permit other nodes to transmit their waiting cells. Denying a node access to the ring is likely to reduce the overall throughput of the switch, since more of the ring's total bandwidth may be left unused. The trade-off between maximum throughput and fairness may be seen in the results presented in Section 5.3.

5.2 Denial of Access to the Ring

With reference to a particular node, the decision to deny or grant access to it can be based upon one of several concepts.

1. An entire call can be denied access by the call acceptance control algorithms to ensure that cells with their CLP (priority) bits not set only suffer from the presence of other such cells.
2. A node may be denied access to the ring if a node downstream has been delayed for some time. The time may be either greater than a preset threshold or simply the longest of all the delays in the ring (again, possibly greater than

a preset threshold). The first idea is used in the threshold control algorithms described in Section 5.3.

3. A downstream node's input buffers have reached a certain level of occupancy. Access to the ring could then be assured for that node until its buffer occupancy had fallen below another lower level. This scheme can cause excessive delays at other nodes if large numbers of cells arrive at one node close together.

4. A cell may be denied access to the ring if a cell with a higher level of priority is waiting downstream. A drawback of this idea is that eventually all ports may have high priority cells waiting, negating the effect of this scheme. This can be disregarded if the call acceptance control algorithms have already reserved the bandwidth on the ring for the higher priority cells, in which case the lower priority ones can expect to be dropped. This is just like the use of the priority bit to decide which cells to drop preferentially in a network which is being operated at optimistically high loads.

5. A cell with a lower fanout is waiting downstream. Encouraging smaller fanout cells through the fabric before larger ones may increase the total number of copies made per ring revolution, but is likely to delay some cells an unreasonably long time. A more interesting control scheme, though impractical at the proposed speeds of operation, is the per-slot packing of the fanouts into the ring. An algorithm to do this would attempt to maximise the use of the ring and hence increase the maximum throughput under conditions of asymmetrical loading.

6. One idea, after Chen *et al* [Chen91b], for dealing with dense multicast traffic is to use another separate ring to assign a priority to each node, which changes the priority after each complete rotation of the multicast fabric ring. This would ensure a form of round-robin access, as used in some token rings.

All of the above ideas for denial of access to the ring assume that the fanout of the cell waiting at the node wishing to gain access to the ring is known, together with the distance downstream of the port potentially causing the denial. This permits a form of 'weak' access denial to be used, in which ports which are further downstream than the fanout of the cell waiting at the node being considered for denial of access, are not considered when making the decision. In fact, permitting the node upstream to access the ring may help the node downstream to gain access later this ring revolution, since other waiting nodes between the two will not be able to use the empty slot.

Preliminary investigations examined the consequences of using a stronger form of access denial, such that any node is permitted to deny access to the whole of the ring, even if the rest of the ring's traffic would not affect it. Not unexpectedly, the performance of the ring as a whole suffered greatly and large delays were observed at very low loads. Therefore, all of the simulations below use the weak form of access denial, where only nodes which can be directly affected by a node gaining access to the ring can cause denial of this access.

The ideas proposed above all also assume non-exhaustive service at each node, so that each rotation of the ring evaluates the state of the switch afresh. If exhaustive or gated service were to be used instead, where a node retains access to the ring until a number (possibly all) of its waiting cells have been transmitted, as in a token ring, the jitter would be expected to increase [Takagi90, p.12].

5.3 Simulation Results

Simulation results of two algorithms for denial of access by threshold, as mentioned above, are presented here. These were chosen for further investigation because they could potentially be implemented in hardware with moderate ease, as described in Section 6.5. The traffic models used are those described in the previous chapter. The effect of two control access algorithms upon the throughput and fairness, measured as the reduction in unfairness with no control algorithm against the unfairness with control, can be seen in Figures 5.1 and 5.2. Again, the results shown are for a 16 node fabric with uniformly distributed random fanout (mean 8).

STCA – Single Threshold Control Algorithm

Each time a node, A, wishes to access the ring, if there is a cell at another node, B, which has been waiting longer than the threshold, R, slots *and* longer than the one at A *and* would be blocked by A's transmission, then node A is denied access.

This procedure is termed the Simple Threshold Control Algorithm (STCA). High values of R mean that the threshold is rarely reached and throughput and fairness results tend towards the uncontrolled results shown in Figure 4.8. Lower values of R reduce the maximum throughput to the same limit but for lower values of α, since the threshold is more often reached at lower loads and more of the ring's bandwidth is left unused. An applied load of 0.6 was chosen for comparison with the results using no access control which were presented in Section 4.5. A threshold was chosen to yield the maximum reduction in unfairness below this load. This threshold was found to be $R = 2$.

The graph on the left in Figure 5.1 shows the maximum throughput with the STCA. For $\alpha = 0.6$, for example, the maximum throughput has been slightly reduced from nearly 0.67 in Figure 4.8 to just under 0.6. The graph on the right in Figure 5.1 shows the reduction in the unfairness measure when using the STCA. Confidence intervals have been omitted for the purposes of clarity. Points where the fabric becomes theoretically unstable have also been omitted. It can be seen that the STCA improves the fairness up to a load where the maximum throughput is reached. The negative values indicate that the ring has become unstable and that the queue lengths are increasing to infinity. The reduction in unfairness is especially marked for high values of α. For example, for $\alpha = 0.9$ and a load of 0.45, the maximum difference in delays is decreased from 4.42 to 1.43 cell times, a reduction of $(4.42-1.43)/4.42 = 0.68$.

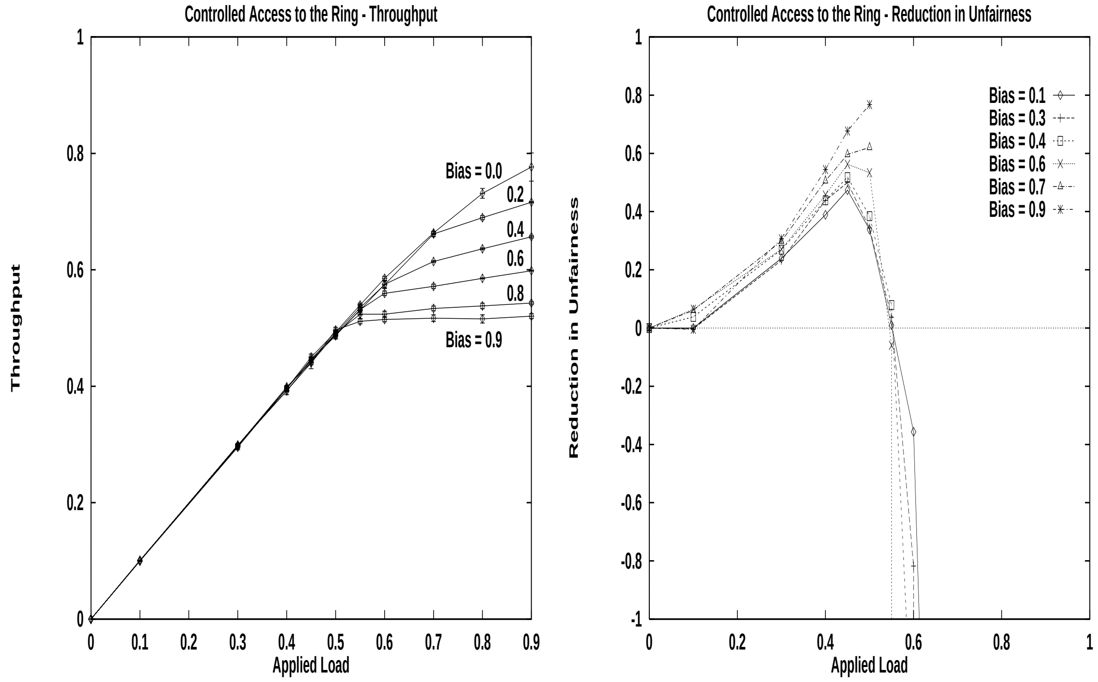


Figure 5.1: The Effect of the Simple Threshold Control Algorithm upon Unfairness

WTCA – Weighted Threshold Control Algorithm

The graphs in Figure 5.2 show the effect of the same basic control algorithm, but with independent thresholds calculated for each node, depending upon the sum of the traffic attempting to leave the fabric though each node. This procedure is termed the Weighted Threshold Control Algorithm (WTCA). The threshold R_i at each node i is given by

$$R_i = \lfloor \frac{R-1}{1-L}(L - \tau_i) + R \rfloor \tag{5.1}$$

where L is the applied load. R is the base threshold ($R \geq 1$) and τ_i is the output rate at node i, as implied by the point source bias distribution and described in Equation 5.2. This ensures that when $\alpha = 0$ (and $\tau_i = L$ with the uniform fanout distribution), the threshold is equal to R, but as the output rate at node i increases, the threshold is decreased, eventually reaching one. Similarly, decreasing the output rate increases the threshold.

The effect of this is that nodes which have a lot of traffic leaving through their output port on the ring will receive preferential access to the ring. A heavily loaded port may have only the traffic which it personally receives on its input port leaving on its output port, whereas ports downstream of the heavily loaded port will have all of the copies from the heavily loaded port together with the incoming traffic from

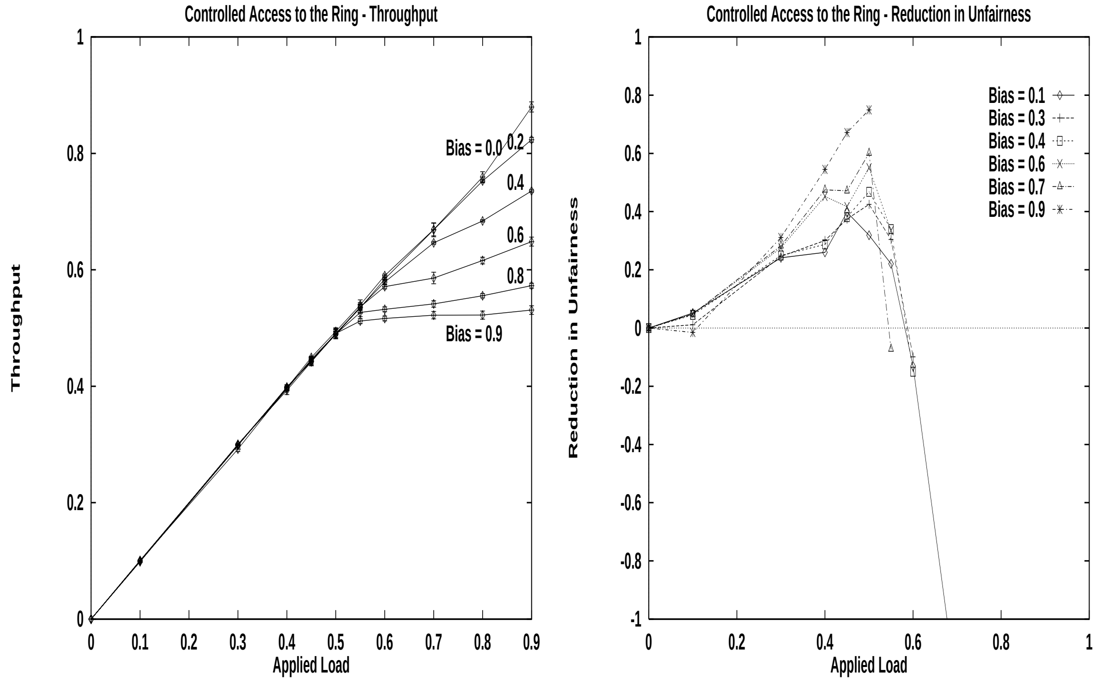


Figure 5.2: The Effect of the Weighted Threshold Control Algorithm upon Unfairness

their own input port. The output rate, τ_i, is given by

$$\tau_i = \begin{cases} \lambda_H + \lambda_L(\frac{N}{2} - 1) & \text{if } i = 0 \\ \lambda_H \frac{i-1}{N-1} + \lambda_L(\frac{N}{2} - \frac{i-1}{N-1}) & \text{if } i \neq 0 \end{cases} \tag{5.2}$$

where λ_H is the arrival rate at the heavily loaded input port (port 0) and λ_L is the arrival rate at the lightly loaded input ports. As defined in Section 4.4,

$$\lambda_H = (1 + \alpha(N-1))\frac{L}{\overline{f}} \tag{5.3}$$

$$\lambda_L = (1 - \alpha)\frac{L}{\overline{f}} \tag{5.4}$$

where $\overline{f}$ is the mean fanout, which in the case of the uniform fanout distribution is equal to $N/2$.

The graphs in Figure 5.2 shows the effect of the WTCA on the unfairness, again for a base threshold of $R = 2$. As may be seen in the graph on the left, the throughput function is changed much as with the STCA, with the maximum throughput being reduced by varying degrees relative to the uncontrolled version shown in Figure 4.8. The maximum throughput is also increased relative to the maximum throughput using the STCA, at least for low values of α.

The graph on the right shows how the fairness changes relative to the uncontrolled multicast fabric. Again, confidence intervals have been omitted for the purposes of clarity, as have points where the fabric becomes theoretically unstable.

Negative values indicate that the WTCA has caused the ring to become unstable and that the queue lengths are increasing asymptotically. For a small range of very high values of α there is an improvement over the results shown in Figure 5.1 For example, with $\alpha = 0.9$ and a load of 0.4, the reduction in unfairness is increased from 0.21 with the STCA to 0.55 with the WTCA.

However, at lower values of α, this improvement disappears. This is probably due to the small threshold of two which was used to maximise the improvement in fairness. If a larger threshold were used, the range of thresholds at different nodes would be wider and improvements over the STCA might be seen at lower values of α. However the WTCA appears to produce only a small improvement in fairness over the effect of STCA, and is for that reason probably unnecessarily complex for controlling access to the multicast fabric.

5.4 Separation of Unicast and Multicast Traffic

In the basic cascaded multicast fabric design, unicast traffic is passed through the multicast fabric together with the multicast traffic. If all the traffic is unicast, then the only delay experienced by the traffic is the delay whilst the entire cell is buffered and then the delay waiting for the head of the slot on the ring to arrive (on average half a cell time). As the proportion of multicast traffic increases the delay experienced by each cell converges rapidly towards the delays shown in Section 4.3.

One way to optimise the performance of the switch for unicast traffic is to bypass the multicast fabric and to pass unicast cells directly to the switch fabric input port controllers. This could be achieved by separating the two classes at the multicast controllers and using a larger switch fabric, where some inputs are dedicated to multicast traffic. Alternatively, a scheme such as that presented in Section 7.2.2 might be used, where all cells first arrive at the input port controllers and are then recirculated to the multicast fabric only if they are multicast cells.

5.5 Even Distribution of Traffic

As mentioned in Section 5.1, the performance of the multicast fabric at high loads will be impaired if the incident traffic is not evenly distributed across all inputs of the multicast fabric. For a high performance multicast fabric an even distribution of traffic across the multicast fabric may be ensured, either pragmatically by using prior knowledge of the likely traffic characteristics to order the placing of inputs to the multicast fabric, or by placing a distribution fabric before the multicast fabric.

A distribution fabric is a switch fabric whose routing of cells need not be determined by the contents of the cells' headers, i.e. their VCI fields, but can be performed according to patterns of traffic which have been observed in the recent past and what is known about the likely behaviour of certain connections in the future. Using a distribution fabric, cells arriving at any of the input ports of the switch may be switched to any of the input ports of the multicast fabric. To avoid cell reordering errors all cells belonging to the same connection have to follow the same path through the distribution and multicast fabrics. In this case, the distribution of the traffic has to be performed on a per call basis at call set-up and may be based upon an estimate of the current load on each section of the ring. Alternatively, different cells within a connection could be allowed to follow different paths through the multicast fabric, in which case cell-reordering with its attendant delays and complications (e.g. possible dropped cells) would have to be performed.

One possibility for the implementation of the distribution fabric is to use another slotted ring since the load on the distribution fabric will be lower than that on the multicast fabric. Multiple smaller rings might also be used as a distribution switch fabric to redistribute the traffic to subsets of the multicast fabric's input nodes.

5.6 Hierarchical Rings

The concept of a slotted ring is especially useful in practice if the area covered by the ring is small, so that the connections can be short and reliable. This is the case if multiple port controllers exist on one card. If this is so, then a single high-speed ring can exist on the card and can be connected to other rings by a hierarchy of rings. In this way, large rings can be built from smaller ones.

If the maximum degree of fanout is known before constructing the switch or can be limited by local applications, then this is an easy solution to the problem of providing multicast in a selective fashion for just some of the inputs of a large switch. One possible problem associated with this approach is the need for buffering at each level of rings, as cells are received from one ring and wait for access to another. This is still an area for further investigation.

Another advantage of the slotted ring multicast fabric is that the speed of the ring can be increased or decreased according to the amount of multicast traffic which is expected to be present. The simulations presented in the previous two chapters have all assumed that the ring carries data at the same rate that the data is accepted by the network interfaces. If the ring is clocked faster than this rate, then the delays experienced will be decreased. Conversely, the slotted ring implemented in the next chapter was deliberately slowed down to load the ring to its maximum level. This permits a relatively fine-scale approach to traffic generation for testing purposes.

5.7 Replication using Multiple Rotations

If the fanout field in the multicast cell header is set to a value greater than the number of active nodes of the ring, then multiple copies of the same cell will arrive at each input port controller. This can be a useful way to generate large numbers of copies in serial batches of almost parallel copies. If the mean fanout required is fairly low, this would be a very efficient way to provide multicast for a switch fabric using a small multicast fabric.

The decremented fanout field should provide just this capability for each input port controller receiving multiple copies of the same cell. One minor difficulty is that the connection set-up algorithms have to be aware of the number of active nodes on the ring to map the copy identifiers onto specific port controllers.

5.8 N Node, N Slot Rings

The number of slots on an N node ring can be increased either by increasing the 'padding' bytes in the FIFO which closes the ring or by increasing the number of bytes stored at each node in the ring. In the first case, the delay between copies emerging from the node above the FIFO and the node below the FIFO would become excessive and so it is not considered any further.

If enough storage is added in the second case, the number of slots can increase until there are N slots distributed around the ring in a symmetrical fashion. Though this may at first seem attractive, in fact it does not improve the delay experienced by cells traversing the multicast fabric. As mentioned in Section 4.2, the delay experienced by a cell is dependent upon the interval between slots arriving, not upon the number of slots which it sees.

The effect of N nodes would be to produce copies serially rather than in (near) parallel. Another effect is that the first copies of cells which are waiting at different nodes would tend to appear closer together in time, rather than separated by having to wait until all the copies of previous cells to gain access to the ring have been made. The short-term throughput may be increased by a marginal amount at high fanouts, since empty queues will not mean that the (single) server has to waste time moving to the next full queue. However, the serially produced copies appear at different output ports of the multicast fabric, which is unhelpful for connection control protocols and the problem of unfair access to the ring also still exists.

5.9 Summary

Several possible solutions to the problem of unfairness in the slotted ring multicast fabric have been examined. Two algorithms based upon denial of access to the ring have been investigated by simulation. The difference in mean delays experienced at each input port of the multicast fabric can be reduced, at the expense of the maximum throughput, so long as neither the load nor bias of the asymmetrical traffic are too great. Possible implementations of the enhancements and their complexity are discussed in the next chapter.

6 Implementation of the Slotted Ring Multicast Fabric

This chapter describes the implementation of the slotted ring multicast fabric which has been described and simulated in previous chapters. The modifications to the design as originally proposed which are necessary for a real implementation are explained and measurements of the performance of the multicast fabric are given. Some of the enhancements proposed in Chapter 5 are re-examined and the modifications which would be required to implement them are described.

6.1 Project Fairisle

Project Fairisle [Leslie91] was an investigation of the issues associated with ATM in the local area[1]. It was based upon an ATM LAN designed and constructed at the University of Cambridge Computer Laboratory. From 1989 to 1992 the Fairisle team built four 4×4 switches connected as a prototype ATM network. Future projects will build upon this base to further investigate control issues such as QOS and call acceptance in larger ATM networks. Such a project is currently being pursued in collaboration with British Telecom.

Since the aim of the project was to investigate the control issues of ATM networks, the design of the switch was for ease of development of the control functionality in the port controllers. Hence the switch fabric itself was not designed for maximum performance, but instead was designed for flexibility.

6.1.1 A Fairisle Switch

A Fairisle switch consists of the following elements:

Switch Fabric

A switch fabric is constructed from one or many single-chip 4×4 crossbar interconnection networks. Each 4×4 switch element was implemented in a 4200-gate equivalent Xilinx FPGA [Xilinx89]. Switch fabrics up to 16×16 have been constructed using the 4×4 switch elements in a delta network. The fabrics run at a speed of 144Mbit/s using byte-wide paths throughout. Two levels of priority exist within the switch elements and a NACK-based scheme is used to detect output contention. The fabrics are entirely input buffered for simplicity of design. A 16×16 fabric fits onto two

[1]Supported by SERC and initially by Hewlett-Packard Laboratories, Bristol.

standard full-height Eurocard boards (approximately 6"×9").

Port Controllers

Each port of a Fairisle switch has a port controller, which is physically a 0.8" wide, extended-depth, full-height Eurocard board, containing the input buffers and their queue management, VCI remapping, contention retries, priorities and connection management. The output port controller associated with the input port is also implemented on the same board but is minimal in this design.

The use of an ARM RISC microprocessor [Furber87] and Xilinx FPGAs to implement the functionality of a switch controller meant that as various functions became well understood they could be moved from software into hardware. The current split is that the VCI remapping and basic queue management, such as the management of free buffers, is performed by the Xilinx FPGA, whilst the connection management and more complex queue control is performed in software. 32-bit data paths have been used throughout the port controller hardware and throughputs of 50 Mbit/s with an ARM3 and 60 Mbit/s with an ARM600 have been observed in practice.

Transmission Interface

Whilst each port controller can be operated in loop-back mode and bootstrap itself at start-up over the switch fabric, other connections to further switches and other devices are made using a separate transmission interface daughter board. The daughter board also contains the reception interface, to complete the line interface. Each port controller has full duplex connections for connection to other switches or other port controllers or host devices within the same switch, using double ('shotgun') coaxial cable. The transmission rate used over this cable is 100Mbit/s using the AMD TAXI chip set [TAXI89], originally designed for early FDDI networks.

Multicast Fabric

The slotted multicast fabric as developed for the Fairisle network is placed logically within the transmission interface on the receive side (see Section 6.2.2). Each multicast node was a small PCB mounted on top of the transmission interface PCB, but the design could have been included on the port controller board. The actual size of the PCB used for the multicast node can be seen in Figure 6.4. As mentioned in Section 5.6, the slotted ring can be operated at a different speed to that of the transmission interfaces if necessary, either to improve the performance of the multicast fabric or to avoid overloading the port controllers.

Various

Apart from space for the power supply in the crate, each port controller in a 4×4 switch has enough space for a further two network interfaces to local networks such as the Laboratory Ethernets or a Cambridge Fast Ring. In a fully populated 16×16 switch, each port controller has one network interface. Each port controller has the capacity to interconnect these networks across the switch fabric, effectively acting as a bridge between a wide variety of networks.

6.1.2 The Fairisle Network

A fully populated 8×8 Fairisle switch would fit into a extended-depth 19" sub-rack and might be expected to cost in the order of £10 000 if it were to be made commercially available. It should be stressed that the first two versions of the port controllers have been designed for experimental flexibility and not for production. A complementary version of the port controller, known as the 'null port controller' has recently been developed and only implements the bare essentials of a switch controller (VCI remapping, simple FIFO queueing). Throughputs of much nearer the line speed of 100Mbit/s have been observed using this null port controller.

Due to the aims of the project, the complexity of Fairisle switches resides in the port controllers and not in the switch fabric. One of the advantages of the migration of the functionality into hardware is that the spare processor cycles can be used for other more complex functions, such as traffic generation for testing performance (Section 6.7.2) or more complex QOS schemes using rate-based control [Kalmanek90].

The traffic for the Fairisle network is drawn from a variety of sources. Conventional data from file transfer and network management can be combined with multimedia traffic from the workstations associated with Project Pandora [Hopper90]. (Project Pandora was a collaborative multimedia project between the University of Cambridge Computer Laboratory and Olivetti Research, Cambridge. Its main concern was the manipulation of digital video in a workstation environment, specifically the provision of video windows on a workstation display and the storage and retrieval of video and audio data.)

6.2 Implementation of a Multicast Fabric for Project Fairisle

The extension of Fairisle switches to allow multicast connections to be assisted by the replication of cells in hardware has been implemented using a slotted ring multicast fabric placed in front of the existing unicast switch fabric and input port controllers. The next three sections describe the design and implementation details of a 4×4 multicast fabric for one of the 4×4 Fairisle switches described above.

6.2.1 Multicast Cell Structure

The multicast cell structure used inside a Fairisle switch is as shown in Table 6.1 and is a modified version of the CCITT cell structure described in Section 1.1.1. A pattern, which is configurable for each switch but has a default of 101, in the least significant bits of byte 1 (VCI0) indicates that the cell has a multicast VCI. The top nibble of byte 1 is the fanout field indicating the number of copies required. Bit 3 is used to turn the telemetry information on and off (see Section 6.3.4). It could also be used as a priority bit for fairness control algorithms (see Section 6.5). Fanouts of `0000` or `0001` are both considered to be unicast in multicast cells.

This structure, and specifically the dedication of a range of 2^5 VCIs to be multicast VCIs, could be imposed upon the entire LAN for a simple implementation. More sensibly the multicast cell header structure could be defined as existing only within a switch after the logical multicast controllers.

Byte	Name	Description
0	Soc	Start of Cell byte
1	VCI0	First VCI byte Multicast cells: Fanout (4 bits) Telemetry (1 bit) Multicast pattern (101)
2	VCI1	Second VCI byte
3–4	SAR	SAR bytes
(5)	CRC	Optional CRC byte
5–52 (6–53)	Data	Data bytes (48 bytes)

Table 6.1: Fairisle Cell Structure (modified CCITT)

6.2.2 Device Interfaces

Figure 6.1 shows the signal interface between the each multicast board, or node, and the transmission board, the input port controller and other multicast boards. Ground and power are also connected though the transmission board, as is the reset signal, which is local to each port controller (though all port controllers can in fact be reset simultaneously from the switch fabric).

`DataNext` is a signal to say when the data in the transmission board latch is valid and is valid a half cycle before the data is latched. The FIFO write signal, `prxFifoWbar`, is generated by the transmission board for writing the data it receives directly into the receive FIFO on the transmission board. It is also passed to the multicast board for use in global unicast mode (see Section 6.3.2). The signals `NoNodes[0..3]` (the number of nodes on the ring), `StartUp` (whether the multicast board is the control board) and `GlobalUnicast` are all set by a dip-switch on each multicast board.

6.2.3 Overall Design of the Multicast Fabric

The multicast fabric is constructed from a series of multicast nodes, as shown in Figure 6.2, one for each input port of the switch fabric which requires multicast capability. Figure 6.3 shows the layout of a single node of the multicast fabric. Solid arrowheads indicate data paths and open ones indicate control signals. Solid boxes are latches, which are all nine bits wide, except for the ones on the ring data path which are ten bits. TxFifo, CloseFifo and CloseLatch are the only items in the diagram which are not within the Xilinx FPGA.

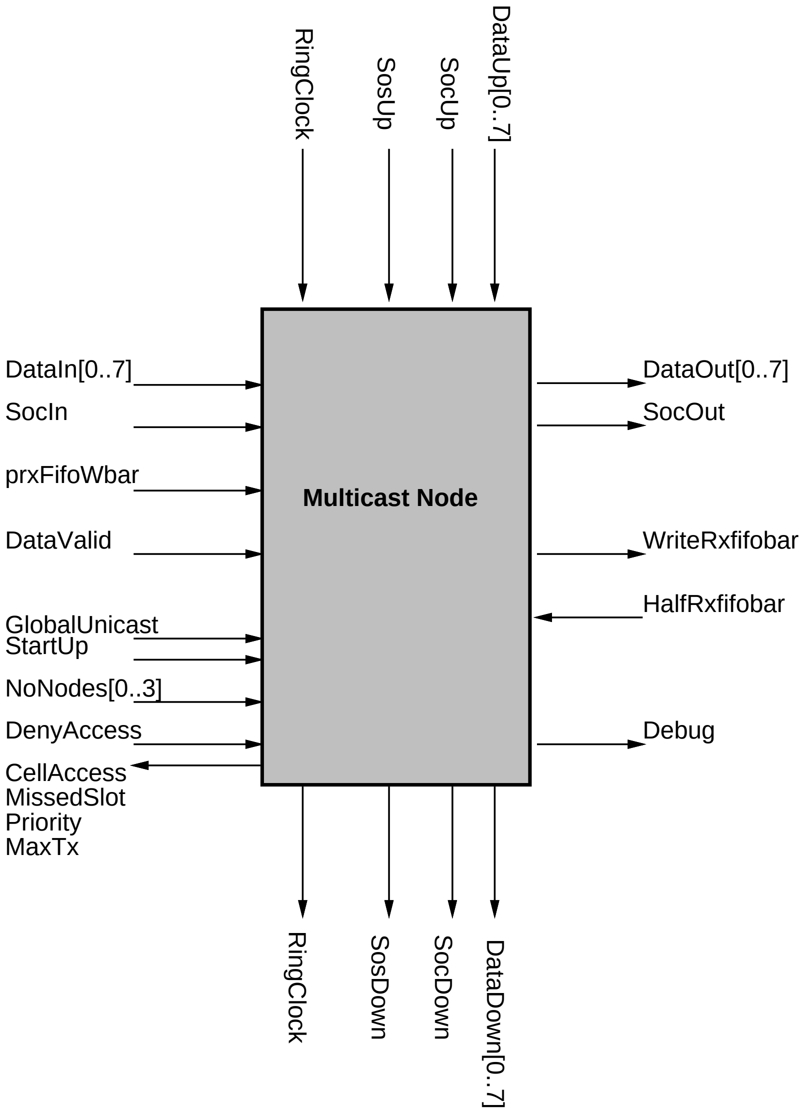


Figure 6.1: Interface to the Multicast Node

The node is placed after the incoming data latch on the transmission board and before the receive FIFO on the transmission board. Each node is synchronised to its local transmission board to receive data and to write it into the multicast board FIFO (TxFifo). Each node can read data from TxFifo and write it to the ring using the clock distributed with the ring. The data is read from the ring into the FIFO on the transmission board (RxFifo) at the required number of nodes, which is until the fanout field is zero. The ring is closed with a FIFO (CloseFifo), in which enough bytes are stored to make the total length of the ring an integer number of slot lengths. Each slot contains the same number of bytes as the cell shown above.

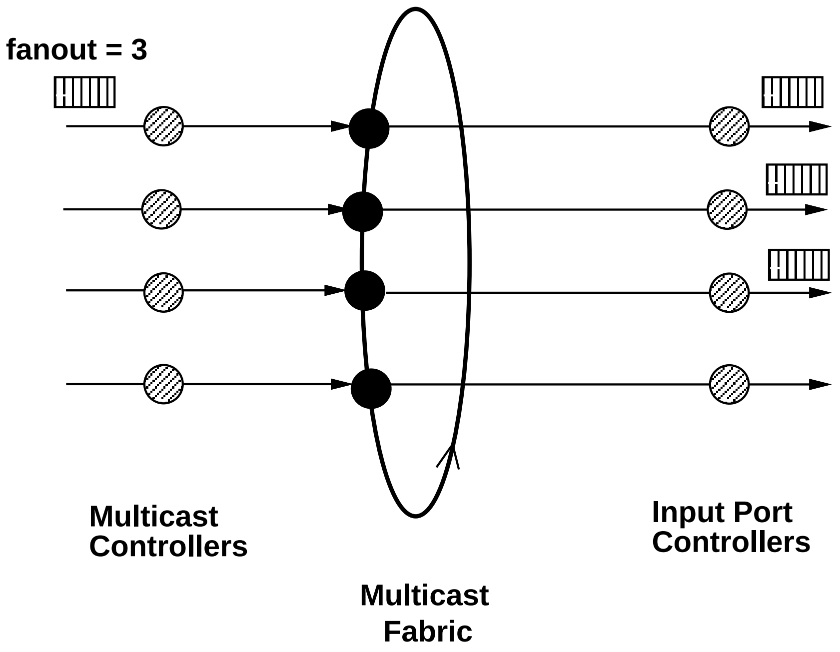


Figure 6.2: The Multicast Fabric Ring

6.2.4 Normal Operation

When data is read into the multicast fabric from the transmission board, the fanout byte is copied forward into the Start of Cell (Soc) byte (byte 0) and left intact in byte 1. If the cell is a multicast cell, then VCI0 in byte 1 is copied forward. If the cell is a unicast one, then zeroes are inserted into the Soc byte and the cell is not replicated by the multicast fabric.

The Soc is signalled on separate input lines (the control lines, here labelled `DataIn[9..10]`). The Soc is used to start the writing to the TxFifo. Spurious Soc bytes after the first one in a cell are transmitted transparently for the port controller to deal with. Thus each node in multicast mode provides cells of guaranteed length (their bytes are counted into TxFifo) with an Soc byte as the first byte, but possibly with other spurious Soc bytes within the data.

A slot on the ring is defined as being empty if the Soc byte, which is synchronised with the Start of Slot (`Sos`) signal at each node, contains a fanout of zero. When an empty slot-head arrives at a node where a cell is waiting, the cell is read out of the TxFifo and copied into the slot, as well as being passed onto the FIFO on the transmission board (RxFifo).

When a full slot-head arrives at a node, the contents of the slot are copied to the output port and also passed down the ring to the next node. The fanout in the Soc is decremented by one, both in the Soc byte for the multicast board and in the VCI0 byte. This ensures that the VCI of each copy is different by one from the preceding one. Fanouts of zero remain at zero after decrementing.

In one configuration of the transmission board, an 8-bit header error check based

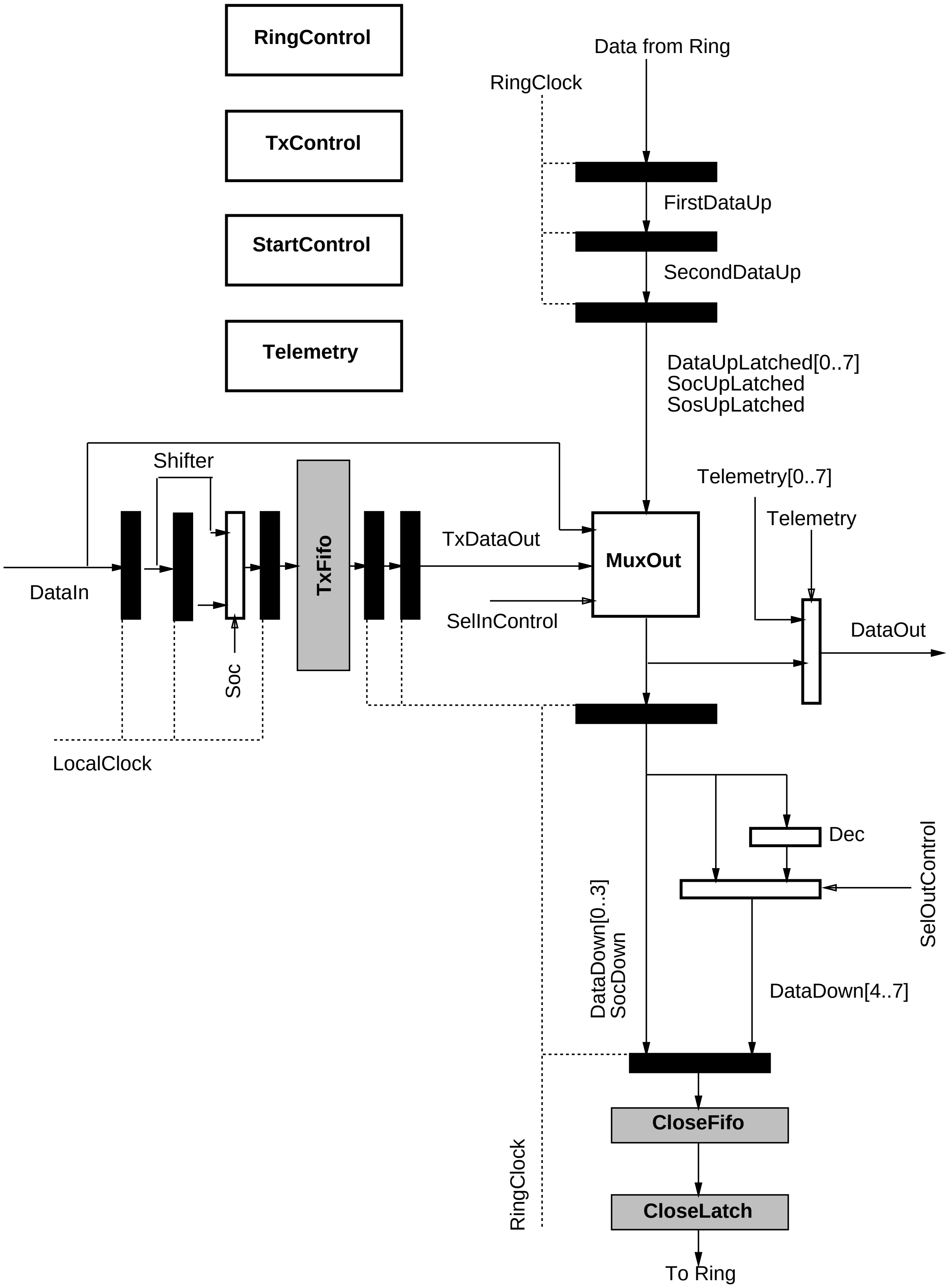


Figure 6.3: A Single Multicast Node

upon the CCITT CRC [CCITT I.432] can be inserted into the cells on the links between switches. When the cell is removed by the receiving transmission board the CRC byte is stripped out and `DataValid` goes low. The multicast board must ensure that the CRC data is not mistakenly written into TxFifo.

The number of complete cells waiting in TxFifo is counted using a 4-bit counter in the multicast node FPGA. The counter is incremented after the last byte of a complete cell has been written to TxFifo and is decremented when a cell in the FIFO gains access to the ring. If the count reaches a value, `MaxTx`, which is encoded in the Xilinx design, then subsequently arriving cells are not written into TxFifo, but are instead discarded. When cells have been read out of the FIFO, incoming cells are once more accepted, with no hysteresis. If the receive FIFO on the transmission board (RxFifo) is more than half-full, then cells are not copied from the ring to that transmission board and that port controller will lose its copy of that cell. This can be avoided by making sure that the port controller always reads the receive FIFO in preference to the loop-back FIFO and that it polls it sufficiently often that large queues do not build up in the receive FIFO.

6.3 Further Design Details

6.3.1 Closing the Ring

Each multicast board has the capability to act as the control node, which generates the first and subsequent Start of Slot (`Sos`) signals after reset. The control node also contains and controls the FIFO which is used to close the ring. All of the other nodes are slave nodes and just pass the `Sos` signal on to the next node downstream.

A counter on the control node begins counting after reset (which is assumed to be global across the multicast fabric) and upon reaching a value determined by the setting of `NoNodes` (the number of nodes in the ring) on the board and the (fixed) length of the cell, generates an `Sos` signal. When the ring FIFO (CloseFifo) has been filled with the correct number of null bytes to make up the remainder of a cell length, `Sos` signals are generated using the fixed length of a cell, which is embedded in the comparator circuits in the FPGA. The incoming `Sos` to the control node is used to permit cells waiting at the control node to gain access to the ring and is then discarded.

The control node also acts as a distribution point for the ring clock, which is derived from a separate crystal-based oscillator mounted on the control board.

6.3.2 Global Unicast

This mode of operation is supposed to imitate the performance of a transmission board when there is no multicast board attached. All incoming data is taken directly from the receiving latch on the transmission board and written back into the receiving FIFO on the transmission board (RxFifo). The operation is performed using the strobe signal generated by the transmission board FPGA (`prxFifoWbar`). Everything is handled as transparently as possible, with the only delay being that of passing the data and strobe signal through a 3:1 multiplexer (MuxOut in Figure 6.3). The fanout continues to be decremented at each node, ensuring that the VCIs of the copies which are destined for nodes below the one with global unicast enabled remain constant.

Global unicast is particularly useful for switching nodes in and out of the multicast fabric for debugging purposes. Further design modifications would be necessary to enable switch management software to detect which nodes were currently switched out of the ring if global unicast were to be used in any other fashion.

6.3.3 Reset State

When the multicast fabric is reset, all of the nodes enter a state where cells may be written to the ring as usual, but no data is read from other nodes on the ring. This is to ensure that multicasting can be safely enabled at a later point after the booting of kernels for the port controllers. The `SocDown` bits on the ring are also zeroed, but it is unnecessary to zero the fanout field since it will be automatically decremented to zero as it passes by other nodes.

Multicasting is enabled at a node by the presence of a multicast cell on the ring. This cell is usually one which has been transmitted by the control node. Since it is just the multicast pattern (101) which enables multicast, any fanout multicast cell should enable the whole ring. After multicast has been enabled, the operation of the node is as described in Section 6.2.4.

6.3.4 Telemetry

For experimental purposes, a means of extracting performance information from each node has been added to the original design. An extra 30 CLBs[2] were necessary for this and so a 6400 gate equivalent Xilinx 3064 FPGA with the same pinout was used. The number of cells in TxFifo and the number of full slots observed by the cell at the head of the queue in TxFifo before gaining access to the ring are measured and inserted into the user data of the copy which appears from the port at which the source cell arrived. Data on the ring is unaffected, so each cell is only monitored once.

A 4-bit counter is used to count the number of cells in TxFifo. If this value is zero, then `MaxTx` has been set and cells may have been lost. A 7-bit counter is used to count the number of missed slots. The eighth bit in the byte is used to indicate that the missed slot counter has counted more than 2^7 missed slots. It remains set until the HOL cell gains access to the ring. The queue length and missed slot telemetry data may be inserted into the first two bytes of the user data, respectively.

The multiplexing of this data can be turned on and off by bit 3 in VCI0. The control point is as the cell is written into TxFifo. So the telemetry information

[2]Configurable Logic Blocks. Xilinx FPGAs are made up of a grid of CLBs, with each CLB containing a configurable section of combinational logic and two flipflops. The CLBs are connected and configured according to the program loaded from a PROM at reset.

begins upon the transmission of the first cell in the FIFO when the activating cell's VCI0 byte arrived and ends upon the reception of a cell with bit 3 unset. Thus no more than the TxFifo limit, `MaxTx`, cells fail to receive telemetry information between the start and stopping of telemetry. Over a large period of observation, this number of cells is negligible.

6.3.5 Contiguous Cells

The arrival of contiguous cells at the multicast board from the transmission board requires that the data pipeline must be sufficiently deep to detect the arrival of another cell before the previous cell has had its final byte written into TxFifo. The current depth of the pipeline for writing data into TxFifo is four transmission clock cycles, which also gives sufficient depth to copy VCI0 (and the fanout) into the Soc byte.

Reading cells from the TxFifo is not a problem, since starting to read a cell depends upon when the latched `Sos` signal arrives. `Sos` is pipelined sufficiently deep (three ring clock cycles) to start reading the next cell from TxFifo and to begin writing it to the transmission board FIFO, RxFifo.

An observation here is that using counters within Xilinx FPGAs is made significantly easier if the counters have a reset which actively clocks a zero input into each latch. The use of asynchronous resets causes problems when a counter is required to return to zero and also count at the next clock edge. Asynchronous reset signals may be delayed enough within the FPGA to cause the next edge to be missed and the count to be one or more short.

6.4 Hardware Design Details

The hardware design environment used to construct the multicast fabric was a combination of commercial and in-house software tools. The design was formalised in a local hardware definition language (HDL), originally based upon Modula2. A local simulator was used to model the various elements of the switch, including such objects as FIFOs and the ARM processor. The hierarchical nature of the hardware definition meant that subsections of the designs could be tested separately.

Xilinx FPGAs were used extensively throughout Project Fairisle and also in the multicast fabric. Their increased design complexity was outweighed by the flexibility of being able to radically alter hardware designs on a daily basis. The turnaround time for minor modifications to the action of a Xilinx FPGA was in the order of 24 hours, after the long learning curve of familiarisation with the available tools was complete. The Xilinx FPGA tools which were commercially available were extended by several local enhancements, notably an X Windows placement tool to permit easier hand placement of the CLBs and a tool to extract critical paths from the predicted track delay values.

Certain difficulties with the Xilinx design methodology were encountered during the design process. Contorting designs to sit comfortably within blocks of five inputs and one output, or two four-bit inputs and two outputs, required some thought. Other difficulties arose from designs whose major critical path might be short enough to permit correct operation, but which had other sub-critical paths that prevented correct operation of the design. This was dealt with by ensuring that all signals were latched after passing through one, or at most two, CLBs, effectively deepening the pipeline, to ensure that the signals were in a known state before subsequent operations. In a similar way, the risks of static and essential hazards meant that the direct reset inputs to CLBs were not used. Instead the clock enable inputs were used to control latches, as mentioned in Section 6.3.5.

The revisions which arose during the design process were for a variety of reasons. Some early revisions were due to inevitable oversights and logical mistakes in the HDL and confusion over the specification of the interface to the transmission board. Later revisions were due in part to errors in the models (e.g. the FIFO model) and the introduction of extra functionality such as the telemetry option.

One cause of complexity in the implementation was the use of asynchronous FIFOs, which returned data read from them as valid for only 5ns after the $\overline{READ}$ signal went high again. The particular FIFOs used also caused problems in other areas of the Fairisle network because the signal indicating that the FIFO was empty came very close in time to the point where the decision to perform another read cycle had to be made. The solution to these problems which has been chosen for future developments is the use of synchronous FIFOs, whose data is returned synchronised to a reference clock.

6.4.1 PCB Design

The multicast fabric PCB was designed using the Racal CADSTAR system (which explains the many six and eight letter names on the floor-plan in Figure 6.4). The multicast board is joined to the transmission board by links in the areas marked 'TAXI' and can also be secured to it by a post, as marked. The holes VMEHL1 and VMEHL2 are for a front plate to attach the card directly to a VME sub-rack. EDGCN1 brings data from upstream on the ring.

The links OUTLK0 to OUTLK9 are for jumpers to connect the outputs DataDownLatched[0..7], SocDownLatched and SosDownLatched respectively from the Xilinx FPGA to either CloseFifo (ClseFo) if they are to the left, or directly to the ribbon cable attached to EDGCN2, if they are placed to the right. This permits a board to be configured as a control node by moving the jumpers, installing ClseFo and ClseLc (a 9-bit latch) and setting the dip-switch appropriately.

6.4.2 Testing and Debugging

Debugging of the multicast board was performed using a suite of test programs, logic analyser output and the output from the board interface. Four red and four green LEDs showed the state of various signals as listed below.

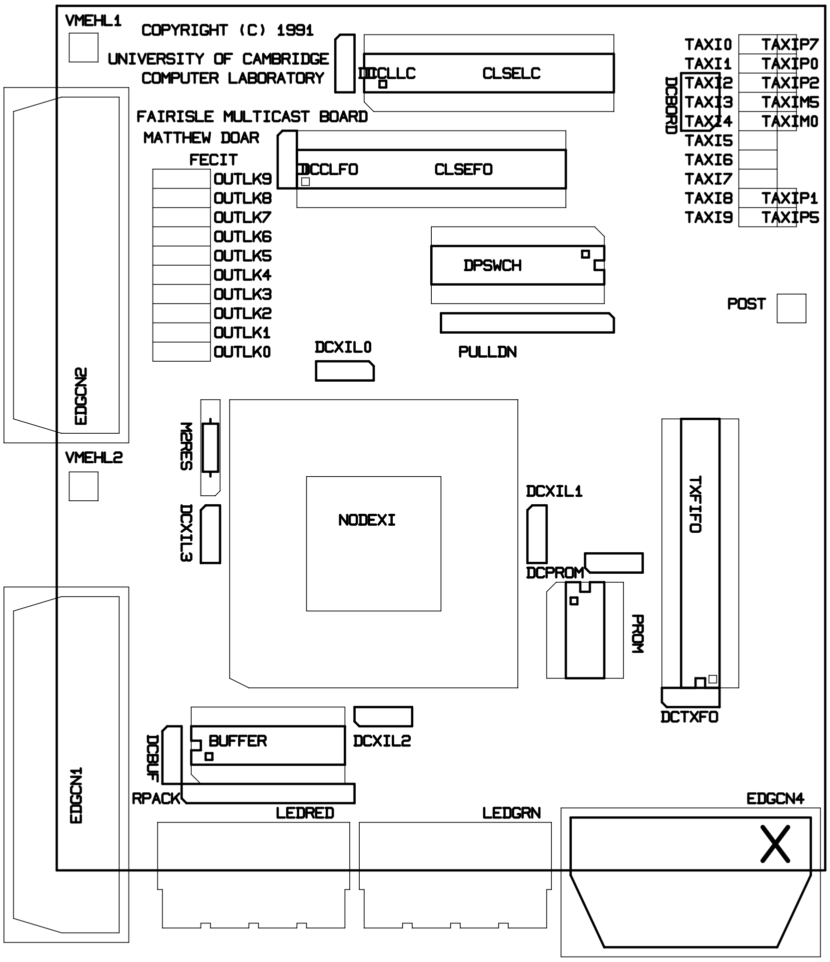


Figure 6.4: Floor-plan of the Multicast PCB (actual size)

ReadTxFifo Active indicates reading from the FIFO that receives the data from the transmission board.

WriteTxFifo Active indicates writing to the FIFO that receives the data from the transmission board.

WriteRxFifo Active indicates writing the FIFO that receives the data from the multicast ring.

MissedSlot Active indicates that a cell waiting in TxFifo has missed access to the ring.

CellAccess Active indicates that a cell waiting in TxFifo has gained access to the ring.

MaxTx Active indicates that TxFifo is full of cells and that any further cells arriving will be discarded.

Power Indicated power on/off.

One of the problems of debugging Xilinx FPGAs especially is that the packaging of the IC in PLCC or pin grid array packages (square with pins in concentric squares) often meant that pins were unaccessible for probes. One of the solutions to this was to ensure that the signal(s) `Debug` in Figure 6.1 were mapped to pins on the perimeter of the package, together with other useful debugging signals, such as whether a cell was waiting in TxFifo. Commercial products do exist to apply probes to all pins on a complex package, but they tend to be rather delicate and expensive.

6.5 Possible Implementations of the Enhancements

This section describes how the access control algorithms proposed and simulated in Chapter 5 might actually be implemented.

6.5.1 Necessary Signals for Access Control

Four outputs and one input to each node have been provided in the current design to permit a possible future implementation of a control algorithm to improve the access fairness to the ring at high loads with asymmetrical traffic.

- `MissedSlot`
  A cycle length pulse, active high, occurring when there is at least one cell waiting and it has just been denied access to the ring, either because of another cell from upstream or because `DenyAccess` is set.

- `CellAccess`
  A cycle length pulse, active low, when a waiting cell gains access to the ring.

- `Priority`
  Intended to give some idea of the current state of the cells in the FIFO. Currently indicates whether a cell is waiting in TxFifo.

- `MaxTx`
  The AM7200(35) asynchronous FIFO can theoretically hold 4 complete cells, but since `MaxTx` changes at the end of a cell's acceptance into TxFifo and a cell gaining access to the ring alters the number of cells counted at the start of a cell's transmission, the maximum number of cells is one less, i.e. 3. Since the decision to read the next cell may be taken before the count is incremented from the last cell read in, contiguous cells would mean that almost another complete cell could be read in. So as a safety margin, the number hardwired into the FPGA should be two cells less. Similarly, the AM7201 can hold a value of 8 cells and the AM7202 16 cells, so the values of `MaxTx` should be set at 6 and 14 cells respectively.

- `DenyAccess`
  When this signal is set during the cycle that `Sos` is latched at a node, any waiting cell is denied access to the ring and remains intact in TxFifo.

Other operations, such as removing the cell at the head of the queue in TxFifo, could be implemented by reconfiguring the Xilinx FPGA. The five input and output pins mentioned above could also be redefined.

6.5.2 Implementing Priority

Keeping track of two levels of priority within the multicast fabric is complicated by the fact that the priority bit is in the header of the cell, so that the priority of the cell at the head of the queue in TxFifo is not known until the cell is read out.

The simplest way to implement a priority detection scheme is simply to increment a counter in each multicast node when a high priority cell is received, and to decrement it when one leaves TxFifo. If the counter is non-zero then there must be least one high priority cell waiting at that node. This scheme is simple to implement, but still fails to give information about the priority of the cell at the head of the queue.

A better scheme is to use a FIFO with as many words as the maximum number of cells in the queue, i.e. `MaxTx`. This tag FIFO can also be used to keep track of the fanout of the cell at the head of the queue, which is necessary for the control algorithms described later, together with information about each cell, just as in the Olivetti design described in Section 3.4.4. Either a separate external tag FIFO can be used or one can be implemented on the FPGA. The design shown in Figure 6.5 is a simple N×1-bit synchronous FIFO which can be extended indefinitely in synchronous logic, since no signal is required to connect to more than one other block. A simple and reliable implementation for Xilinx FPGAs can therefore be obtained.

The operation of the FIFO depends upon two latches per stored bit, one to indicate the data (D) and another to indicate when the data is valid (V). To write a bit into the left end of the FIFO, the data is set up on DataIn and Write-Left is set. The data is latched into the node, becomes visible at the next node and V is set. Write-Right is then set and if Read-Right then becomes high, V is reset, since the data has been taken by the next node to the right. In this way, data passes down the FIFO until it comes to rest at the left hand end of the queue.

Reading from the FIFO proceeds in a similar fashion. If Read-Right at the right-hand end of the FIFO is set, then the current data (which has been externally observable) is changed to that latched from the node on the left of the head of the queue. If there is no more data, then V is unset.

The maximum delay from the presentation of data to be written into the FIFO until it can be read out at the right hand side is N clock cycles. Successive data bits emerge continually from the FIFO. The FIFO is empty when Write-Right at the extreme right goes low, and full when Read-Left remains low when Write-Left is set. In the context of using the simple FIFO to store the priority information about incoming ATM cells, the maximum useful length of the FIFO is approximately 50 bits, since the data has to be valid for reading by the arrival of the next (possibly contiguous) cell. A Xilinx implementation of the FIFO would require an estimated two CLBs per bit.

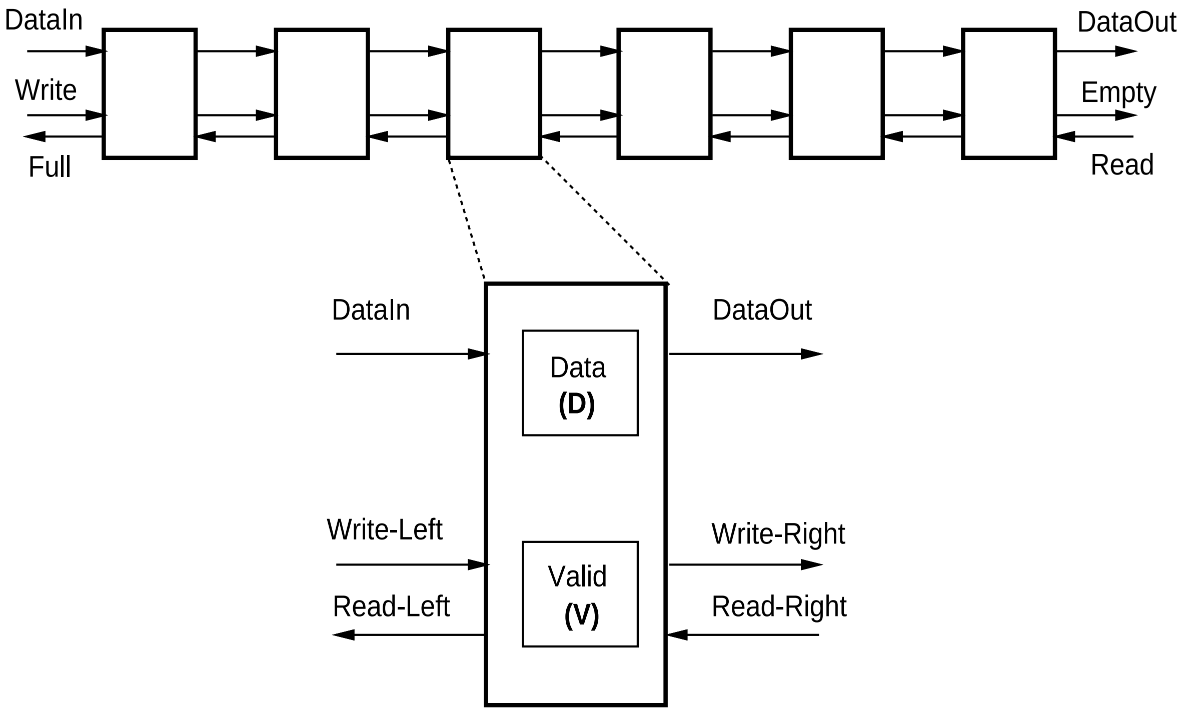


Figure 6.5: Simple FIFO for Priority

6.5.3 Access Control Algorithms

The two threshold control algorithms proposed in Section 5.3, STCA and WTCA, need to know the following information:

- Whether a cell is waiting at a node.
- The number of slots missed by a waiting cell.
- The fanout of the waiting cell (for 'weak' denial of access).
- The current threshold value.
- The traffic arrival rate at each node for the WTCA.

The simplest method of implementing the control algorithms in hardware would be to use a centralised control node, perhaps as an extension to the control board which generates the Start of Slot signals and has the FIFO to close the ring.

To decide which cell had waited the longest, a counter for each node is necessary, incremented by the signal `MissedSlot` and cleared by the cell at that node gaining access to the ring `CellAccess`. A cell is deemed to be waiting at a node if this counter is non-zero. The size of the counter, d, depends upon the maximum likely delay through the multicast fabric. A likely value of 255 slots would require an 8 bit counter, using approximately $8N$ latches and N inputs in total for an N node ring. An N node ring would also require $O(N \lg N)$ inputs to indicate the fanout of

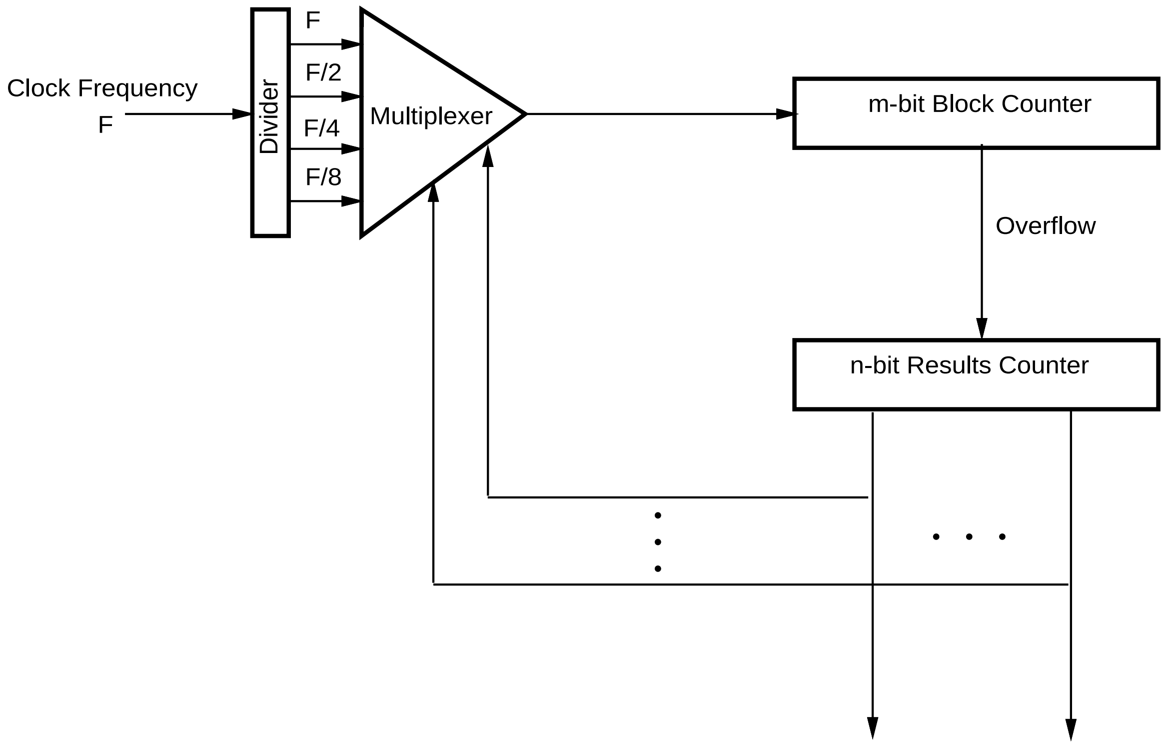


Figure 6.6: Implementation of Exponential Encoding of the Arrival Rate

the waiting head of line cell at each node. Storage and comparison to the current threshold value would require N sets of comparators, one for each node.

6.5.4 Encoding the Traffic Rate

The traffic arrival rate for the WTCA could be encoded in a non-linear series of ranges on an 8-bit counter. For instance, contiguous cells might be recorded as having an inter-cell arrival count of 0 steps, and this count could be arranged to increment in exponentially increasing steps. Figure 6.6 shows how this exponential encoding of the traffic rate might be implemented. The frequency of the incoming clock is repeatedly halved. Increasing values in the results counter are used to select a slower rate at which to clock the block counter, which in turn clocks the results counter more slowly. The block counter is an ordinary wrap-round counter with carry out (overflow), which can be used to adjust the granularity of the exponential result.

If the block counter comprises m bits, then zero as a final result in the results counter indicates that between 0 and $2^m - 1$ clock cycles were counted. Examining the bottom 4 bits of the results counter, 0001 indicates that up to $(2^m - 1) + (2^m - 1)$ were counted, 0010 indicates that up to $(2^m - 1) + (2^m - 1) + (2 \times 2^m - 1)$ were counted, 0011 indicates that up to $(2^m - 1) + (2^m - 1) + (2 \times 2^m - 1) + (2 \times 2^m - 1)$ were counted, and so on. If the results counter comprises n bits, then n stages of frequency division are required and the maximum number of clock cycles, N_{max}, which can be counted is given by

$$N_{max} = \frac{(2^m - 1)(2^{2n} - 1)}{3} \tag{6.1}$$

If 2^n frequency division stages can be reasonably constructed, i.e. n is small, then the bits in the results counter can be redefined so that 0001 indicates that up to $(2^m - 1) + (2 \times 2^m - 1)$ clock cycles were counted, 0010 indicates that up to $(2^m - 1) + (2 \times 2^m - 1) + (2 \times 2 \times 2^m - 1)$ were counted, and so on. N_{max} is then 'hyper-exponential' and is given by

$$N_{max} = (2^m - 1)(2^{2^n - 1} - 1) \tag{6.2}$$

Other designs of exponential counters which involve cascading small versions of the basic design can also be imagined. The major advantages of an exponential counter design over a linear counter are threefold.

- The exponential counter design scales well with size and reduces the size of the counter required or, equivalently, increases the maximum interval which can be counted.
- Frequency division is a basic function, easy to implement and one which scales well for small values of n in synchronous hardware.
- The full-scale design of large linear counters is not simple and ensuring synchronicity between each stage can be very difficult. The delay tends to increase $O(n \lg n)$, though this can be reduced to $O(\lg n)$ with a full look-ahead design.

As an example, to count up to an interval of 10s on a 150Mbit/s line, using byte-wide paths, a 28-bit counter is necessary, together with the logic to encode the count as a series of ranges for compactness. To perform the same task using an exponential counter requires a 15-bit results counter with the basic design and only 5 bits with the hyper-exponential design, even with a non-existent block counter ($m = 1$). Alternatively, a 16-bit result counter can represent just 3.5ms at 150Mbit/s using a simple linear counter, or 76s using the basic exponential design. The idea of exponential counters is of general interest for performing rate-based flow control in hardware.

6.5.5 A Possible Control Architecture

A possible architecture for implementing the threshold control algorithms for an N node ring is shown in Figure 6.7. Each node has a d-bit counter, which is reset by `CellAccess` (CA), to count the number of missed slots from the signal `MissedSlot` (MS). Each node also has $\lg N$ input lines to indicate the fanout of the cell at the head of the queue and a register to indicate the threshold for that node, which is based upon the base threshold R, but may differ from node to node according to the n-bits of traffic rate. An already existing counter on the control node which generates the Start of Slot signal, `Sos`, can be used to indicate at which node the `Sos` has just arrived. Each node is aware of its position in the ring. This might be found by using the first multicast cell, which enables multicast at all nodes (see Section 6.3.3), with a fanout of N, and extracting the current fanout from this cell for use as the node number. Alternatively, the position of each node could be hard-wired into the control node and the node number would be implied by where the signals from each node entered the control board.

The decisions required for either control algorithms to deny a node access to the ring are made when the head of the slot reaches that node. The procedure is as follows:

- The current count of missed slots is compared to the value in the threshold register. Let this result be A_1.
- The HOL cell's fanout is added to the node number to calculate the furthest node which would be affected by this node accessing the ring. This value is multiplexed onto a bus which reaches the control hardware for all the other nodes (the multiplexer is shown as separate boxes but is logically one unit). They compare their node number to this value, modulo N, and if it is within range, i.e. they would be affected by the node in question, then A_2 is set.
- If both A_1 and A_2 are high, then the current value in the missed slot counter is forwarded to the third stage, an N-input, d-bit comparator, which returns the maximum value presented. This could be a Batcher sorting network [Batcher68] as used by some of the switch designs in Chapter 3. If either A_1 or A_2 are low, then a zero value is presented to the comparator.
- The output of the comparator, which is the maximum number of missed slots by a cell within range of the current fanout of the HOL cell at the current node, is compared to the value of the missed slot counter at the current node. If it is greater, then the current node is denied access to the ring, since some other node has a cell which has been waiting longer.

The growth rate of the control architecture described here can be shown to be $O(N(\lg N)^2)$, if a Batcher sorting network is used for the comparator [Batcher68]. The necessary encoding and distribution of the fanout of the cell waiting at each node causes the architecture to grow only as $O(N \lg N)$. The fixed amount of complexity per node is not insignificant compared to the overall complexity in a multicast fabric node, but in real terms this would still be a simple architecture.

Perhaps the most limiting part of this design would be the need for two $(\lg N)$-bit buses, one to distribute the number of the most distant node which could be affected by the fanout of the currently waiting cell, and the other to distribute the number of the node where the current slot head is, to control which node writes to the first bus. In addition, if the comparator is implemented in a centralised fashion as shown for clarity in Figure 6.7, then two other $N \times d$-bit buses would be necessary. If the multicast fabric is distributed over a large area (possibly several sub-racks), implementation difficulties are often linked to the number and size of buses such as these.

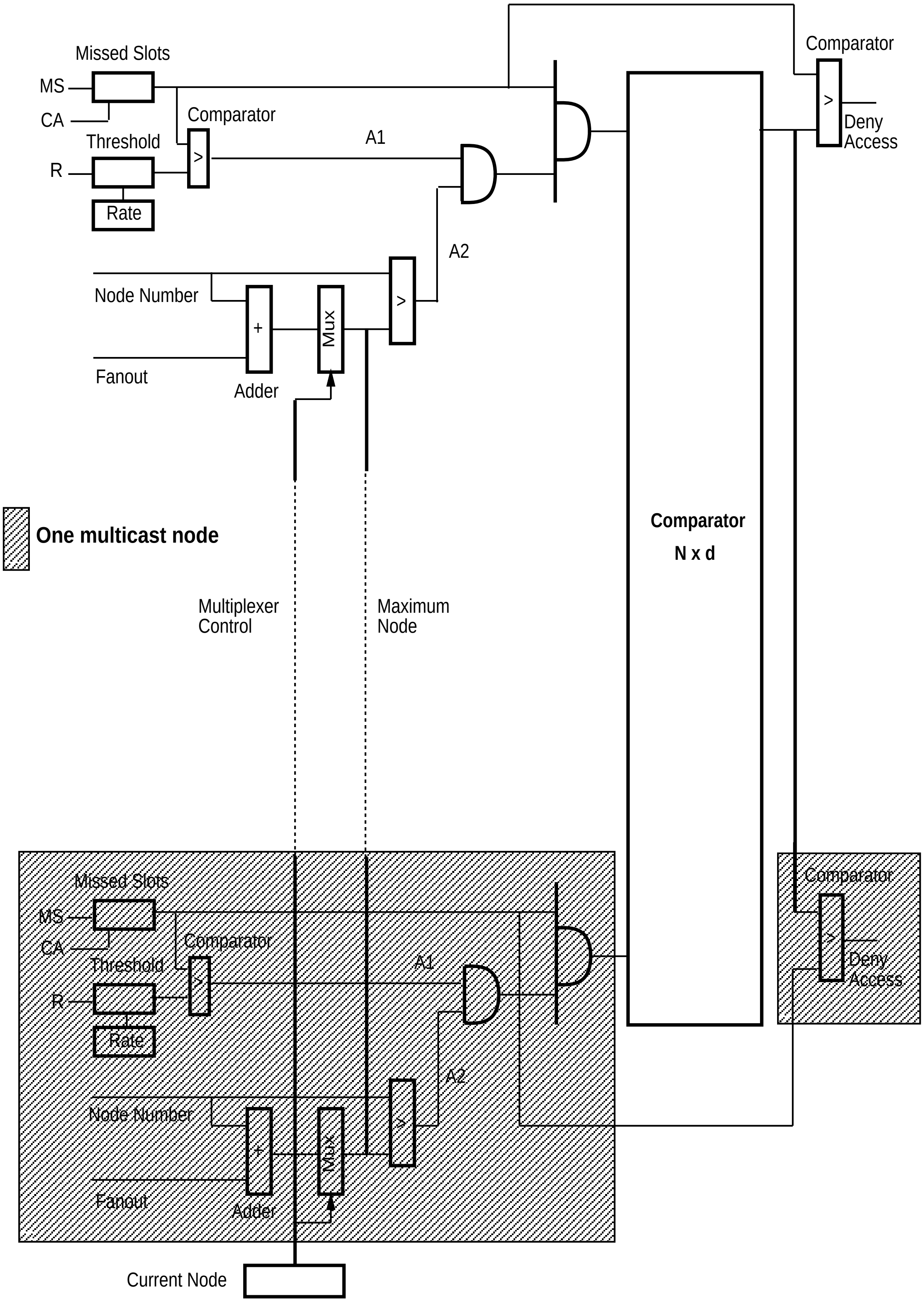


Figure 6.7: A Possible Control Node Architecture

6.6 Scalability of the Slotted Ring Multicast Fabric

The multicast fabric design presented here is simple to implement and has good characteristics for switch management. However, as the number of nodes in the ring grows some problems of scale become apparent.

As noted in Section 3.6, synchronisation of nodes to the ring becomes harder and a more complicated form of clock distribution would probably become necessary. The current method of clock distribution is a doubly-buffered distribution of the local clock from the control board to the others in a 'daisy chain', properly terminated at the end furthest from the control board.

With a large fanout, the delay from the first copy to the last copy grows linearly with N. With seven bytes of storage at each node, more than eight nodes on the ring means that the first copy leaves more than a cell length (56 clock cycles) before the last one, a time of approximately 5μs at 100Mbit/s. The design is most efficient when the number of bytes stored in the ring is just less than or equal to a multiple of 53, since then there is a minimal delay between copies from the nodes above and below the FIFO.

These issues, the convenient size of the fanout field in the multicast VCI and other economies of scale would suggest that a slotted ring multicast fabric with 32 nodes would be the largest appropriate size using current technology.

6.7 Measurements of the Multicast Fabric

Working multicast fabrics up to 4×4 have been constructed. Nodes were connected using ribbon cable with alternate grounds, which was successful for lengths of up to 40cm at 12.5MHz. The Xilinx FPGA designs were successively modified until a maximum throughput of 100Mbit/s was obtained. As noted earlier, the delay between copies of the same cell is determined by the number of bytes latched at each node. In the implementation described here there were seven bytes stored per node, giving a delay between copies of 560ns with a 12.5MHz ring clock. With a ring clock frequency of 8MHz, a working design using only three bytes of storage per node was obtained, producing an inter-copy delay of only 240ns for a data rate of 64Mbit/s.

The minimum delay across commercial grade Xilinx FPGAs (into an Input/Output Buffer (IOB) and out though an IOB) was measured at less than 10ns. This is what the global unicast mode of operation comes close to, since the data path goes through just one CLB. Typical delays for the design using a 3042PG132-70 Xilinx FPGA are around 40ns on most data lines. A ring clock speed of 12.5MHz means that the cycle time is 80ns, and by measurement the mark/space ratio was usually 3/5 after buffering. The read and write pulse width timings for –35 grade (35ns access time) FIFOs are theoretically violated by the reduced mark/space ratio, but appeared to work satisfactorily.

6.7.1 Parameters for Measurements

Averages of measurements could theoretically be taken per ring revolution, per input cell, per connection and per input port. Telemetry operations could have been performed using locally developed Xilinx Teaching Boards, which consist of a Xilinx FPGA with serial and parallel interfaces. However, the method described in Section 6.3.4 delivered the information directly into a logging file, which made post-processing of the data far easier.

Some measurements which might be monitored in a multicast fabric include:

- The average delay across the switch with and without the multicast nodes in operation.
- The average queue size in the buffers, i.e. cells counted into TxFifo.
- Signs of unfairness affecting the performance of the ring for an individual node, e.g. maximum difference between mean delays at each node.
- Minimum acceptable size of the buffers for some predefined standards of service.
- Maximum acceptable size of the buffers for some predefined standards of service.

The first two of these were implemented in hardware as described in Section 6.3.4.

6.7.2 Measured Performance

The telemetry operations described above yielded the delay as the number of missed slots and the queue size in TxFifo each time a cell gained access to the ring from TxFifo. Traffic generating kernels were designed and developed[3] which were observed to be capable of generating blocks of 8 cells with just one clock cycle between each cell in the block. The minimum time between blocks was one cell time for a small number of blocks ($\leq 1K$) and 32 cell times for larger numbers of blocks. This was due to the need to update more complex queue structures and to extract the telemetry information. These figures and the calibration constants required to ensure that the arrival rates could be accurately set were estimated by observing the timestamps placed in the arrived cells by the port controllers, which had a precision of one cell time. A four node ring was used, with all four nodes transmitting and receiving asynchronously, though they began transmitting the test sequence and then later logging the data together. To ensure that the port controllers were never overloaded, the ring clock frequency used was 5MHz, producing a maximum data rate from the multicast fabric of 40Mbit/s on each output port. The cells were received into the multicast fabric at a clock speed of 12.5MHz to ensure that the slotted ring could be adequately loaded by the traffic generating kernels.

[3]Implemented by Shaw Chuang (Computer Laboratory).

The mean and 99th percentile of the delay and queue length distributions were calculated for 108,196 cells received from the multicast fabric. Before data logging began 10,240 cells were received to allow the ring to reach a stable state of operation. The traffic generator was limited to fixed inter-arrival times between blocks, so values were used which corresponded to the means of the geometric distribution used in Chapter 4. The minimum inter-arrival time of 32 cell times meant that the applied loads which could be generated were limited to approximately $0.5\times\overline{f}$. There were two major problems encountered with the generation of traffic at high rates by low-level software. The first was minimising the impact of measurements upon the generation of traffic and calibrating the effects of increasing the fanout, which required more time to be spent receiving cells. The second problem was ensuring that measurements were taken when traffic was arriving at all four input ports at the same time, whilst each stream of traffic was in fact interfering with all the others. The four port controllers were rebooted together and then each one delayed by a carefully chosen time so that the traffic streams overlapped in the required manner.

Figure 6.8 shows the mean delays for various fixed fanouts and for a uniformly distributed fanout, both with the 4×4 multicast fabric. Delays are measured directly as integer numbers of slot lengths. As expected, as the load was increased, the mean delay also increased, though in a less regular manner than the simulations had suggested. This may be due to the difficulties of ensuring a complete overlap between all four sources, as mentioned above. The major anomaly in the graphs shown is that the delay decreases as the fanout increases, contrary to the simulation and analysis results. This is due to the method of generating the traffic, which meant that as the fanout increased and more cells were received, the real-time interval between blocks increased in a non-linear fashion with respect to the calibration measurements (taken for a fanout of one), reducing the effective traffic rate. This could be adjusted for in future experiments by improving the calibration, possibly by using a logic analyser to monitor the actual arrival rate for various fanouts and loads, though this would be a lengthy process. As shown in Table 6.2, the values of the delays are higher than those suggested by the simulations of Chapter 4, for a 4 node ring with a fixed fanout of 4. Multiple experimental runs for selected loads and fanouts suggest that the experimental results have a 95% confidence interval of at least 1.3 cell times.

The graph on the left in Figure 6.9 shows the distribution of the delay as the probability of a number of slots missed before gaining access to the ring. For a typical result, with an applied load of 0.4 with a fixed fanout of 4, most cells only miss a small number of slots. Interestingly, from the experimental results, there appeared to be a very small number of cells which were delayed for much longer times, grouped in regular clusters spaced approximately 16 cell times apart. The graph on the right in Figure 6.9 shows these clusters on a larger scale. As may be seen, the probability associated with each of these clusters is very small, but they do appear to have a significant impact on the overall mean delay, which may explain some of the discrepancies in Table 6.2. They also meant that the 99th percentile of the mean delay reached the ceiling (imposed by the size of the delay counter) of

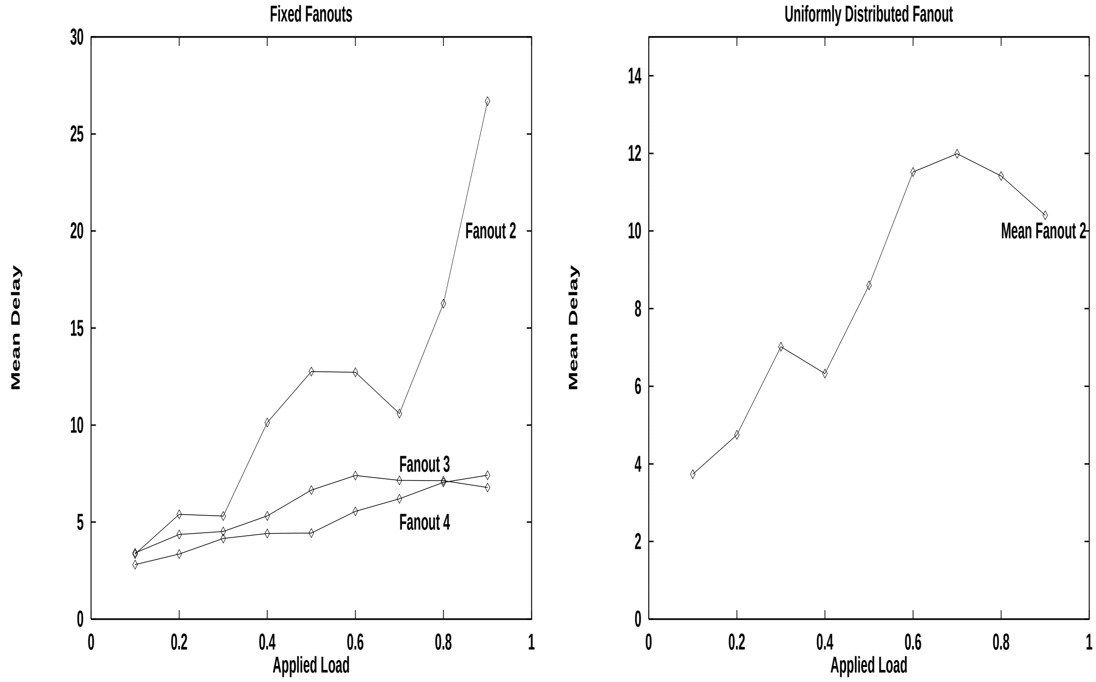


Figure 6.8: Mean Delay for Fixed and Uniform Fanout Distributions

$2^7 - 1 = 127.0$ at very low loads.

The frequency of these clusters was not related to the fanout in a ring of any given size, though their amplitude was related approximately linearly to the applied load. Therefore it seems unlikely that the clusters are the result of the fanout being a multiple of the ring size (Section 4.2). Decreasing the number of cells in a block from 8 to 4 and 2 whilst maintaining the same overall arrival rate did not change the spacing of the clusters. However, it did move the delay at which the clusters peaked closer to zero by a consistent number of cell times, confirming that their existence was not an artefact of the hardware. The exact cause of these clusters is as yet uncertain.

Measurements of the queue lengths confirmed that with a block size of 8 cells and applied loads such that the overall load on a switch did not exceed 1.0, each block of cells was read out of the buffer before the next one arrived. If the receiving clock and ring clock were run at the same speed, then the number of cells in the queue would be one (for a fanout of one), as each cell would gain access to the ring before the next one in the block fully arrived. With a clock ratio of $12.5/5 = 2.5$, the mean queue length was measured to be between 2.7 and 3.1 cells for all applied loads, with 99th percentile queue lengths of up to 12 cells. These rather low results may have been due in part to the use of fixed inter-block arrival times. A geometric distribution of inter-block arrival times would be expected to increase the mean queue legnth.

One observation derived from several months spent designing the experiments

Applied Load	Mean Delay	
	Simulated	Measured
0.1	0.3583	2.809
0.3	1.2387	4.155
0.4	1.7932	4.413
0.5	2.5124	4.435
0.6	3.4033	5.553
0.7	4.5459	6.199
0.8	6.2162	7.050
0.9	8.4239	7.417

Table 6.2: Measured and Simulated mean delays for a fixed fanout of 4 with a 4 node slotted ring

to measure the performance of the multicast fabric is that the generation of traffic in software is fraught with difficulties. As mentioned above, ensuring that events occur within an executing piece of code at set intervals is hard to guarantee without system support for an extremely fine-grained clock (WANDA does support a concept of microseconds, but with no guarantee of accuracy or precision) and a detailed understanding of the likely latencies of many operations. System support for global synchronisation to within milliseconds in processes spread across a small area such as a switch is also desirable. Whilst the interest in such real-time operating systems is growing, the difficulties associated with them suggest that traffic generators should currently be constructed in hardware. Such a traffic generator could be constructed using a large circular buffer which is preloaded with typical data and then transmitted at the required rate by a dedicated controller, running from an external clock of known accuracy. Other signals to and from the controller could ensure synchronisation between several such devices over a small area such as an ATM switch. Cells could have timestamps on an extremely fine scale inserted into their data and returning cells could be sampled intermittently to obtain accurate delay measurements. Loading a switch with asymmetrical traffic arrival patterns would also be possible, since the balancing of incoming and outgoing cell buffers would be easier than in the current kernels.

Future uses of the multicast fabric include the distribution of cells containing video data from a camera generating ATM cells at source [Pratt92]. The design of these cameras permits a wide range of data rates to be produced with relative ease for real data. As such, they would be excellent sources for obtaining delay and queue length results for real traffic, without as many of the problems associated with the generation of traffic by software.

6.8 Summary

A multicast fabric for Project Fairisle has been designed using the slotted ring design described earlier. The design has been implemented, tested and its performance has

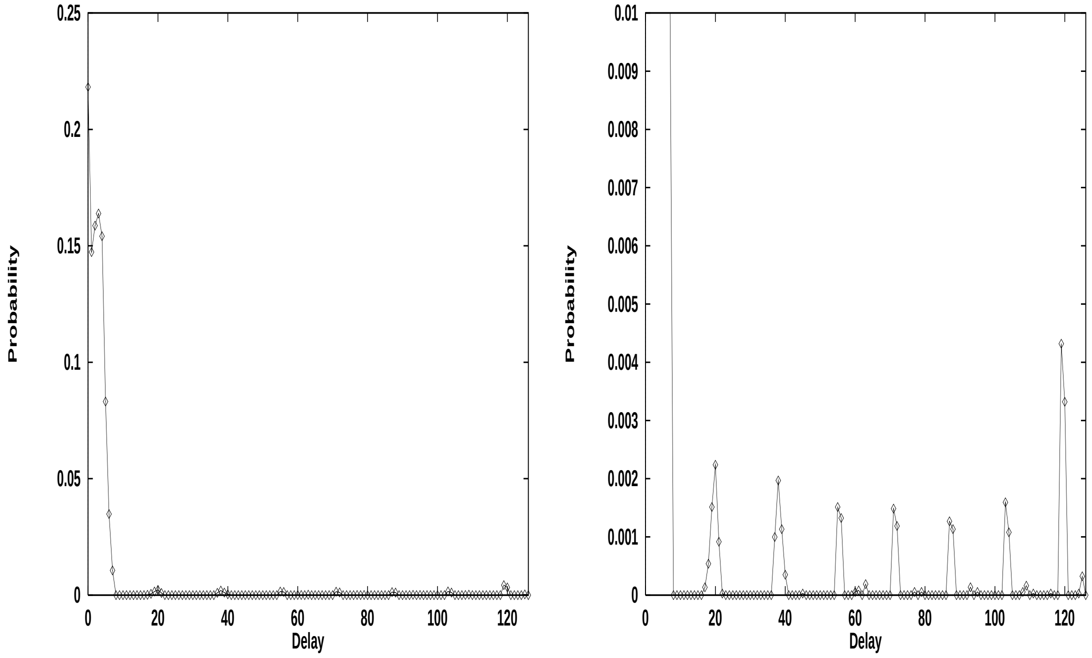


Figure 6.9: Delay Distribution for a Fixed Fanout of 4 and an Applied Load of 0.4

been measured. The results obtained show broad agreement with the performance characteristics obtained in earlier chapters and highlight some of the difficulties of generating real-time traffic in a controlled fashion using software.

An architecture for the proposed control algorithms of the previous chapter has been presented. In practice, the absence of unfairness at low loads and moderately symmetrical traffic patterns mean that the extra complexity of any control structure is likely to be unnecessary, certainly in the local area. However, a useful idea for counting large numbers of events has also been described, which could have applications in future rate-based flow control hardware.

7 Multicast in the Multi-Service Network Architecture

The general requirements for managing multicast connections and the problems of implementing multicast for an existing protocol suite are addressed in this chapter. Multicast connectivity is first formally defined and then connection management issues related to how multicast is actually implemented for a switch are considered. A description of how the particular implementation described in the previous chapter might be managed is given. A brief description of the Multi-Service Network Architecture (MSNA) is followed by the procedures which are currently used for unicast connection management. A connection management approach for MSNA is chosen and a series of modifications for multicast connection management are proposed, which could form the basis of more wide-ranging alterations of the MSNA architecture.

7.1 Definition of a Multicast Group

The term *multicast group*, which has been used rather loosely in previous chapters to imply that a set of hosts were in some way connected, usually one-to-many, can be defined more rigorously as follows:

> A multicast group is a relation $\mathcal{R} \subset \{\mathcal{S}\times\mathcal{D}\}$, where $\mathcal{S}$ is the set of possible sources and $\mathcal{D}$ is the set of possible destinations. The must be at least one member in each set, i.e. $|\mathcal{S}| \geq 1, |\mathcal{D}| \geq 1$. For every member, d_i, of $\mathcal{D}$ there is at least one member, s_i, of $\mathcal{S}$, such that $(s_i, d_i) \in \mathcal{R}$. Likewise, for every member, s_j, of $\mathcal{S}$ there must be a member, d_j, of $\mathcal{D}$ such that $(s_j, d_j) \in \mathcal{R}$. The graph created by $\mathcal{S}, \mathcal{D}$ and $\mathcal{R}$ must be directed, acyclic and have only one connected component. Some multicast group members' hosts may be in both $\mathcal{S}$ and $\mathcal{D}$, but their instances will be different.

When $|\mathcal{S}| = |\mathcal{D}| = |\mathcal{R}| = 1$ the multicast group is an ordinary unicast pair. The primitive actions which can be performed upon a multicast group are the addition and removal of a destination, addition and removal of a source and the addition and removal of members of $\mathcal{R}$ (connections). Primitive multiple group actions include splitting a multicast group in two and joining two groups together.

A multicast group can be constructed from $|\mathcal{S}|$ separate multicast groups which have just one member in their set of sources. The same logic also applies using groups with just one member in their set of destinations. Each of these groups can in turn be constructed from multiple unicast pairs. Exactly when a multicast group should be considered as a single group, a collection of single source/destination groups or a collection of unicast connections depends upon the level of abstraction away from the physical layer and partially upon the implementation of multicast in the hardware, as described below.

7.2 Multicast Switch Connection Management Issues

The design of a multicast protocol which makes use of an underlying multicast mechanism in the network should ensure that the details of a particular implementation are clearly separated from the more general actions of multicast connection management described above. To enable this separation to be made, we first examine some of the issues which are switch dependent.

At any moment in time, the state of the sources, destinations and connections of a multicast switch can be considered as a multicast group. Since the replication of an incoming cell in all the multicast switch designs considered in Chapter 3 is the basic action, the multicast group can be most easily considered as comprising multiple single-source multicast groups. Thus it seems reasonable to first consider the effects of different design choices upon a single-source multicast group.

One important difference between cascaded multicast fabric designs is whether the ports at which the replicated cells leave the multicast fabric are fixed or are dependent upon the rest of the traffic in the fabric. An example of the former is the slotted ring design of this thesis and examples of the latter are the designs by Lee [Lee88a] and Turner [Turner88]. If the cells may appear at any port, then all the input port controllers at the switch fabric must be part of the multicast group, whether they actually receive cells or not. This is not the case with the fixed exit ports. Of course, if the cells are broadcast, then there is no difference. Similar considerations apply to non-cascaded multicast switch designs.

Another issue is whether copies of cells have identical VCIs or whether they are tagged in some way, as is the case currently in the slotted ring design. If each of the f copies is unique and the set of exit ports is fixed, then f different VCI mappings have to be set up, one in each input port controller. If each copy is identical, and again the set of exit ports is fixed, then only one VCI mapping has to be set up in f port controllers but the same number of VCI mappings have to be set up in both cases. The extra effort involved in deriving the outgoing VCIs of the unique copies is offset by the ability to add and delete members of the multicast group without changing all of the VCI mappings. This is described in more detail below. If each copy can leave the multicast fabric at any port, then using a different VCI for each copy would mean that f VCI mappings would have to be set up at each input port controller, which is clearly undesirable.

One approach to the allocation of multicast VCIs (MVCIs) is that part of the VCI address space should be dedicated to them. This address space is assumed to be the same for each port controller in a switch. This multicast address space will probably be a relatively scarce resource since all the port controllers must use it for their multicast groups and so careful consideration should be given to ways of conserving it.

If each copy is unique, then the number of MVCIs should be a multiple of the switch size, N, for maximum efficiency. This may be deduced by considering a labelling scheme such as the one used in the slotted ring multicast fabric. This design ensures that the VCI is always decremented at each node on the ring to provide a unique identifier for each copy. In addition, copies of cells will leave the multicast fabric at the output port opposite the input port that the cell came in on, and will also appear on $(f-1)$ contiguous output ports, creating a fixed set of exit ports. Since the largest size of a multicast group is unknown when the group is created, the worst case of broadcast has to be accounted for. In this case, N different VCIs are reserved for expansion across the port controllers. Multiple blocks of N VCIs are thus required in the MVCI address space. Note that using the 'identification by decrement' scheme, N VCIs at the edge of the unicast address space can only be used for incoming copies of multicast VCIs, not unicast VCIs, since they may be in fact be decremented MVCIs.

7.2.1 Modifications to Existing Connections

One measure of the cost of modifying a multicast group is the number of different port controllers whose VCI remapping tables have to be altered, and how many of these entries must be changed in each port controller. Reducing the cost of modifying a multicast group for different multicast fabrics is an important factor in minimising call set-up times as the switch size increases.

The advantage of a fixed set of output ports which are independent of the traffic, either at the time when the connection was set-up, or subsequently during its use, is that the set of port controllers which have to be updated at connection set-up and connection modification is limited and known before-hand. This means that a far smaller number of updates are needed and hence the connection set-up and modification times can be significantly reduced.

Using the slotted ring design where the fixed set of exit ports is contiguous, whether the copies are unique or identical, adding another member to the multicast group involves just one VCI table update prior to increasing the fanout of the connection. Deletion of a member is more difficult due to the possibilities of cell-reordering, which is considered undesirable in ATM networks. The most obvious way to avoid the problem is to construct a 'shadow' set of VCI mappings and then bring them into use when the fanout is decremented. Carefully done, this can ensure that cells which have been buffered prior to reception by the switch fabric are dropped, rather than being reordered or misdelivered.

The shadow scheme proposed for the slotted ring makes use of the fact that the MVCI is decremented and not incremented. Table 7.1 shows an example of a VCI remapping table for MVCI x, which initially has a fanout of four. $x.n$ indicates the *nth* copy of MVCI x. If destination 'b' is to be removed from the multicast group, the shadow set is constructed as shown in Table 7.2 in preparation for the fanout being decremented, after which copies of MVCI x will only appear at input ports P1, P2 and P3. Table 7.3 shows the remapping table after the fanout has been decremented. The entry for port P4 could in fact be left intact until the next addition, since the slotted ring design guarantees that no $x.1$ copies would ever reach it. Misdelivery is avoided since all the incoming VCIs have been altered, so any cells from before the modification arriving from buffers will not find an outgoing VCI and can be dropped.

Port	VCI In	VCI Out
P1	$x.4$	a
P2	$x.3$	b
P3	$x.2$	c
P4	$x.1$	d

Table 7.1: VCI Remapping Table before deletion

Port	VCI In	VCI Out
P1	$x.4$	a
P1	$x.3$	a
P2	$x.3$	b
P2	$x.2$	c
P3	$x.2$	c
P3	$x.1$	d
P4	$x.1$	d

Table 7.2: VCI Remapping Table before decrementing fanout

Port	VCI In	VCI Out
P1	$x.3$	a
P2	$x.2$	c
P3	$x.1$	d
P4	$x.1$	–

Table 7.3: VCI Remapping Table after decrementing fanout

An alternative is to use the port controller which is being deleted to route the copy which it sees to the destination currently served by the port controller on the end of the contiguous set of ports. In Table 7.3 this would mean that P3 remained unchanged from Table 7.1 and P2 would route $x.2$ to 'd', requiring only one modification of a VCI remapping table.

7.2.2 Emulating Multicast Controllers

To provide a source with a multicast VCI which is constant throughout a call, though the fanout of the connection may alter, it is necessary to have MVCI remapping

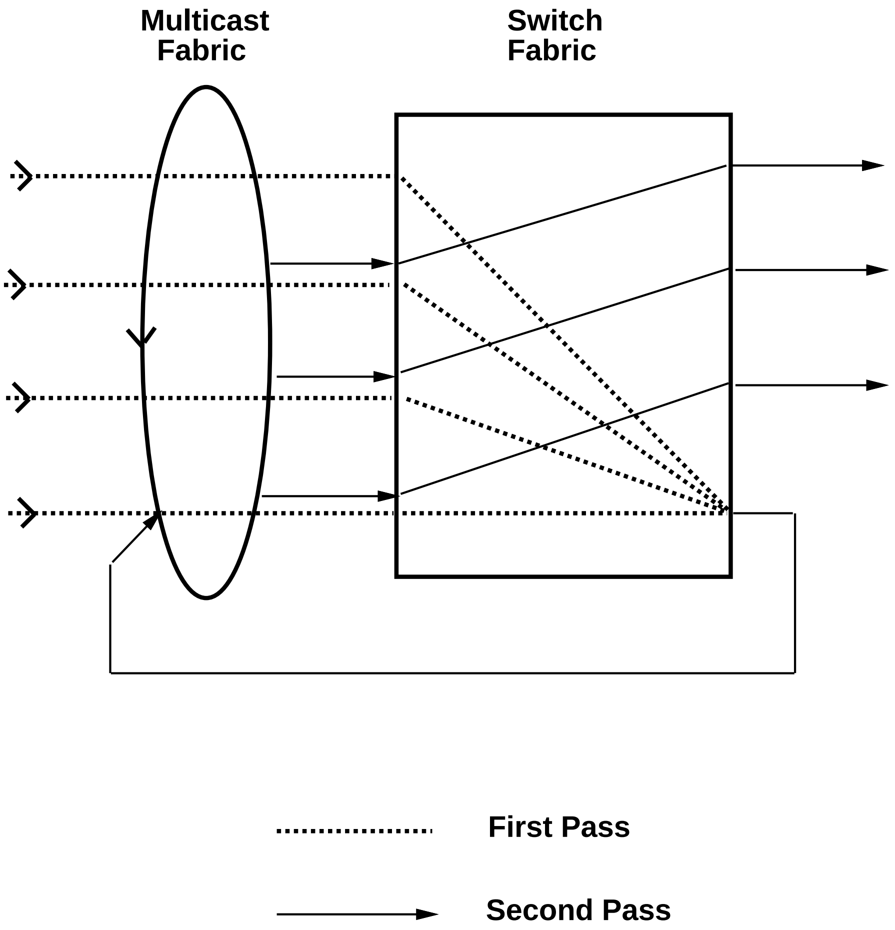


Figure 7.1: Recirculating Cells in a Switch

occur before the cells enter the multicast fabric. Parameters which may alter during a call, such as the fanout, can be stored in the MVCI tables and hidden from the source. For simplicity, this was not built into the original hardware (the fanout was embedded in VCI0) but the function can be provided by recirculating multicast cells as shown in Figure 7.1.

The sequence of events is as follows:

1. All cells pass through or around the multicast fabric upon their arrival at the switch.

2. Their VCIs are remapped to MVCIs if they are multicast cells or to their outgoing VCIs if not.

3. Multicast cells are all mapped to a well-known output of the switch, which is connected back to the multicast fabric. If an loop-back path to the multicast fabric outside the switch fabric existed in each port controller, then this could be used instead of a single well-known output in order to reduce traffic across the switch fabric.

4. The cells are replicated and distributed by the multicast fabric.

5. Upon entering the port controllers for the second time, the multicast VCIs are mapped to their respective output VCIs.

Recirculating multicast cells ensures that the updating of multicast VCIs is an operation entirely internal to the switch. This means that switches and sources can route and generate multicast traffic in exactly the same fashion as unicast traffic and only multicast switches need be aware that a connection is destined for multiple hosts. If the percentage of traffic which is multicast is kept relatively low, less than 20% say, then no speed-up of the fabric clock should be necessary. If a larger amount of multicast traffic is likely, then the fabric can be run at a faster rate than the input lines, to cope with the increased proportion of recirculated cells.

7.3 The MSN Architecture

The aim of the Multi-Service Network Architecture [McAuley90] is to support multimedia applications in an ATM environment. The architecture is based upon a connection-oriented model which has many similarities with the B-ISDN model. It has been used extensively in the University of Cambridge Computer Laboratory to support a wide range of applications over the past three years including those within Project Fairisle.

One major idea of the MSN architecture is that fine-grained multiplexing should be supported at only a few key points in the protocol stack, unlike the OSI 7-layer model, where multiplexing can occur at each layer. This permits efficient use of transmission links but ensures that minimal overhead is needed to process the data at each layer. Four layers are defined which contribute to the inter-networking service.

- MS-Access – Datalink/MAC.
- MSDL – Datalink.
- MSNL – Network.
- MSSAR – Segmentation and Reassembly.

MS-Access

MS-Access provides a consistent interface for the support of MSDL over heterogeneous networks. Variably sized protocol data units (PDU) from the different networks have to be mapped into the fixed length MSDL service data units (SDUs). Details of this mapping for Fairisle, Ethernet and several other networks may be found elsewhere [Dixon92]. QOS is defined on a per-instance basis for each MS-Access point, of which there may be several per MAC layer.

MSDL

MSDL defines lightweight virtual circuits, also known as *associations*, of the form (VPI_A, VPI_B, QOS), where the two VPI/VCIs are the connection identifiers between two MSDL entities, A and B, and QOS is the QOS instance, which identifies the MS-Access instances used between them. Joined MSDL entities must exist in the same network, but may be joined to other networks' MSDL entities by cell forwarding at the MSDL layer (in some cases). Thus the internetwork connection is still logically part of the MSNL layer, but is made more efficient. There is no error-recovery defined between MSDL instances, since the underlying networks are assumed to be highly reliable. One of the reasons for using a fixed size (32 bytes) MSDL SDU was so that processing in internetwork routers could be minimised and fully implemented in hardware. The major requirement of a router is simply to be able to remap the VPI at each router, just as in an ATM switch.

MSNL

MSNL defines *liaisons* as concatenations of MSDL associations. Liaisons are not multiplexed over associations and the MSNL PDU and MSDL SDU are the same size, i.e. MSNL adds no protocol overhead to the data stream. This enables the destination MSNL entity to find the identity of the MSNL user without demultiplexing the incoming data.

Connection Management

Connection set-up is done on a hop by hop basis, using a *promiscuous* association defined for this purpose. A promiscuous association is a well-known association which is willing to receive MSDL PDUs from any source. This is analogous to the permanent virtual circuits of B-ISDN, except that only single cell-sized PDUs and only one hop may be used. The connection management promiscuous association can be found indirectly by using an address management service on another well-known VPI and then used directly afterwards. Figure 7.2 shows the operations used for setting up and tearing down associations. A *request* for connection is sent on a promiscuous association to any entity on the same network, with the MSNL address of the source, the VCI to reply to the source on and an identifier for the association. The address of the source and the VPI which should be used to reply to it are contained within the single PDU. The next entity may reply with a *wait* packet, indicating that it is already in the process of setting up the connection, and then with a *reply* packet, containing the VPI with which to contact the destination node. If a connection set-up request fails, this can be detected either by a timeout or by a *kill* packet being returned. The same *kill* packet mechanism is used to unilaterally tear down an association or, with restraint, to indicate that data is arriving for a defunct association.

Liaisons are set up as a series of associations, with each entity in the chain using the MSNL address of the source in its request to the next entity. In some cases it may be acceptable to initiate data transfer after the first association in a liaison has

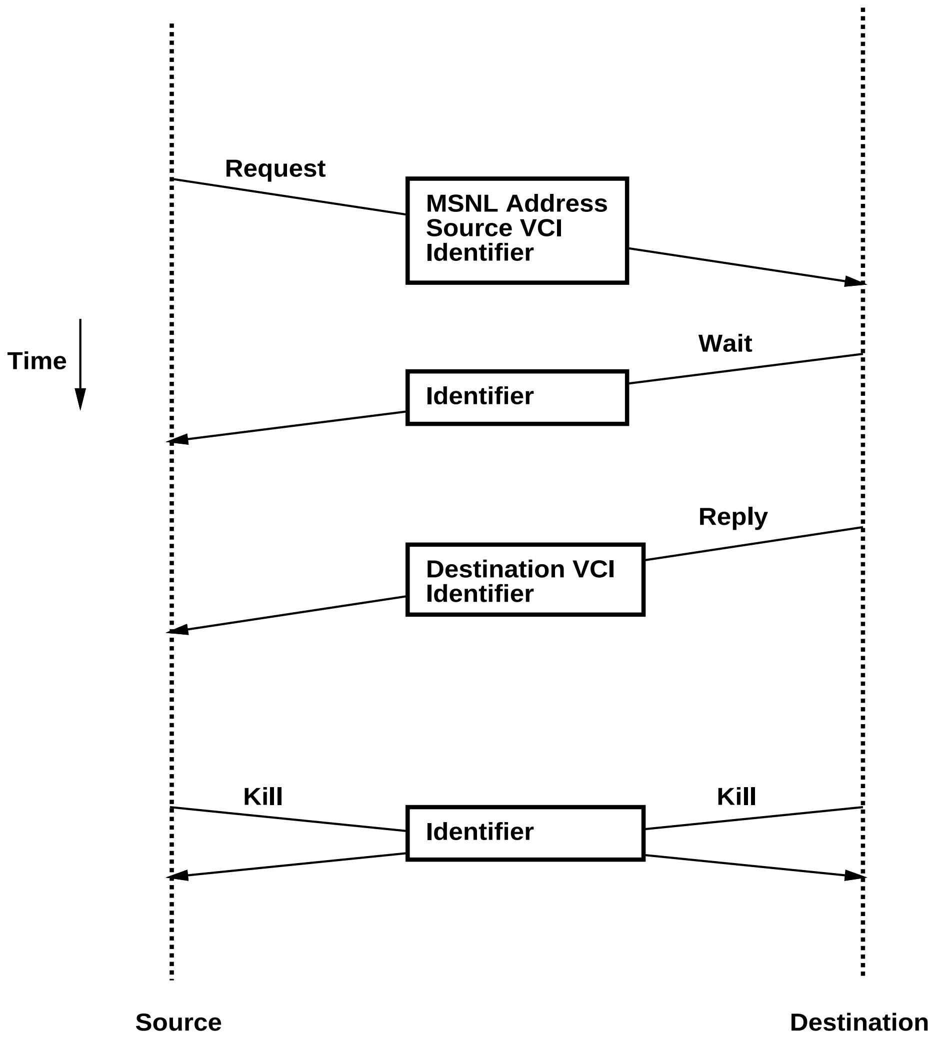


Figure 7.2: MSNA Connection Management Operations

been successfully set-up, optimistically assuming that the rest of the liaison will be successfully established. This approach risks the data catching up with the *request* packet, but reduces set-up times dramatically. The usual approach is to wait until all the associations have been set up before using the liaison for data transfer. Once a liaison has been set up in the MSNL, the actual forwarding occurs in the MSDL layer using the association remapping tables.

MSSAR

MSSAR is directly analagous to the ATM adaption layers described in Section 1.1.4. It handles the segmentation and reassembly (SAR) of variable length blocks from higher levels into fixed length MSNL SDUs. Different applications may require different types of service and which type is used has to be agreed upon when the connection is established. Currently just one SAR scheme is defined. The MSSAR header contains two bytes which are used for a block identifier (major), a sequence identifier (minor) and two bits to indicate whether a PDU is the start of a new block or whether it is a partly full 'end of block' PDU. The MSSAR level is a relatively heavyweight reassembly mechanism and more lightweight ones have been developed for use in ATM networks where cell reordering is unlikely. An example of such a mechanism is AAL 5, which uses an 'end of frame' bit to indicate the last SDU in a block.

7.3.1 Existing Connection Management Procedures

The operating system chosen to support MSNA for Project Fairisle was WANDA, a locally developed lightweight multi-threaded micro-kernel. More detailed accounts of the memory management for WANDA [Mapp92], MSN specific implementation details [Dixon92] and background reading may be found elsewhere [Wilson92].

The major procedures which deal with MSNL connection management in WANDA include:

`msnlSocket(so)`
Sets up a socket `so`. Sockets are the WANDA abstraction for a connection's end-point.

`msnlBind(so, adr)`
Binds a socket `so` to the MSNL service access point (SAP) `adr`.

`msnlOffer(so, timeout)`
Indicate an interest in receiving connection requests on socket `so` and wait `timeout` before withdrawing the offer.

`msnlAccept(so)`
Accept a call of `msnlOffer` on socket `so`.

The sequence of operations to establish a connection is that a SAP is bound to a socket and the socket is offered. If the offer is accepted, a request can be made to connect two sockets to form a connection.

`send_request(pod, associd, src, dst, qos)`
Send a request for connection to destination address `dst` on network interface `pod` from source `src` with requested QOS `qos`. This request may be broadcast if necessary on a promiscuous association.

`send_wait(pod, associd, src, dst)`
Used to acknowledge receipt of a `send_request` to prevent multiple connections being set up due to repeated requests. Not currently implemented as a timeout mechanism is used instead.

`send_reply(pod, associd, src, dst, qos)`
Reply to a connection request from `src` with QOS `qos`. This is always sent only to the original sender.

`send_kill(pod, associd, src, dst)`
Tear down the association `associd` between `src` and `dst`. If this is in response to receiving cells on an invalid VPI/VCI, it may be broadcast on a promiscuous

association.

7.4 A Model of Multicast Functionality

Prior to considering how MSNA might be modified to include multicast, a general model of what multicast functionality requires is necessary. The model and its objects and operations which are suggested in this section are derived from the existing MSNA work described above and those proposed for CMAP [deHart92] (described earlier in Section 2.11.6).

The primary object is the multicast group or *call*. Calls consist of *members*, who can be sources or destinations or both in separate instances, and *connections*, the elements of $\mathcal{R}$ in Section 7.1. The parameters of a call are shown in Table 7.4. There is the concept of call *owners* who can be given control over such factors as who joins the call and who is charged. Note that whilst unicast is logically just a special case of multicast, its dominance in real networks demands special consideration. Hence the inclusion of the Type call parameter.

Parameter	Description
Type	Multicast or unicast.
Accessibility	Calls may be *open*, so that anyone can join, *closed* so that only an owner may add new members or *verify* which asks permission.
Monitoring	Who is informed when changes occur.
Modifiability	Whether non-owners can add new members.
Priority	Call precedence for CAC.
Connection List	List of current connections in the call.
Member List	List of current members of the call.

Table 7.4: Parameters of a Call

Each connection object has a VPI/VCI, QOS, bandwidth specification and other parameters to handle dynamic data rates. Members indicate whether they wish to transmit or receive data. The operations upon these calls and connections can be defined as *commands, prompts* and *notifications* (Tables 7.5, 7.6 and 7.7. Commands are operations used by a client requesting a service from the network, which replies with either a positive or negative acknowledgement. A prompt is a request from the network for an action by a client. The client responds with an acknowledgement and receives a confirmation upon completion of the requested action. Notifications are used by the network to inform clients of changes in state, with no acknowledgement being expected.

An extension of CMAP functionality is multiple group (call) actions, which should include the ability to split a multicast group into two groups for the purposes of conference management. This would probably be most useful for large conferences where subgroups often form for local discussion. Using the notation of Section 7.1, if a group is to be split permanently into two groups, $\mathcal{S}_1$ and $\mathcal{D}_1$, and $\mathcal{S}_2$ and $\mathcal{D}_2$,

Operation	Description
open_call	Open a call ready for members.
close_call	Close a call.
add_member	Add a member to a call.
del_member	Delete a member from a call
add_conn	Add a connection to a call.
del_conn	Delete a connection from a call.
mod_call	Change a call's parameter values.
mod_conn	Change a connection's parameter values.
mod_member	Change a member's access privileges.
trace_call	Return information about a call.
trace_member	Return information about a member.
change_owner	Pass ownership to another member.
client_reset	Tell the network that a client has been reset.

Table 7.5: Multicast Commands – Client to Network

Operation	Description
invite	Invite a member to join a call
open_con_invite	Invite a member to add a connection.
mod_conn_invite	Invite a member to modify its values.
invite_owner	Invite a member to become an owner.
verify	Check whether a member will accept a change.

Table 7.6: Multicast Prompts – Network to Client

with the existing connections being used within each group, but with the interconnecting connections being torn-down, then at most $max(|\mathcal{S}_1||\mathcal{D}_2|, |\mathcal{S}_2||\mathcal{D}_1|)$ tests for inclusion of each unicast connection have to be made. Merging two multicast groups (calls) is useful for the same sort of applications for which group splitting is useful. Again, a potentially quadratic number of connections may have to be established to perform this operation.

7.5 Proposed Modifications to the MSN Architecture

The simplest way to implement the functionality described above is to make use of the existing unicast connection management procedures of the MSNA. This would mean inserting a logical multicast level into MSNL code to construct multicast connections from multiple unicast connections. If future developers wished to implement multicast connection procedures as single procedures for the purposes of efficiency, they should be able to reuse the same interfaces as defined in this section. As noted in Section 7.2, there should be a separation of the parts of the multicast procedures which are dependent upon the design of multicast switch used. In addition, the concept of the maximum fanout of each switch must be introduced, with a value of

Operation	Description
announce_member	Announce a new member.
drop_from_call	Notify a member that it has been dropped from a call.
announce_drop	Notify an owner that a member has been dropped.
announce_owner	Notify a member that a new owner exists.
announce_change	Notify a member of a change in parameter values.
network_reset	Tell the client that the network node has been reset.

Table 7.7: Multicast Notifications

unity indicating a unicast switch.

Separate proposals to create explicit control and management planes for MSNA, after the ATM Layered Model (Section 2.10), are currently being developed [Crosby92]. In these, a *management domain* (MD) is defined with an *MSNL master*, with which the MD entities exchange connection management signals. The master is a logical unit within the MSNL layer, aware of the configuration of the MD, which could be a switch with multiple entities such as port controllers, for example. Entities within the MD can only signal to the master, not directly into the network. The master would be a suitable place to hold switch specific information such as the maximum fanout. The master could also use separate services such as a router and a topology discovery service to make local optimisations. The discussion which follows assumes the existence of this control model.

The modifications for MSNA would implement the functionality of Section 7.4, together with necessary structures to maintain and manipulate the call and connection objects. At the upper MSNL level, all multicast would appear to be many-to-many, as defined for the procedures above. These calls would be constructed from unicast connections in the first instance. The use of separate connections may sometimes be necessary in any case, for example, if very different QOSs or coding schemes are required at different destinations. However, as noted in Section 2.4, this approach does not scale well with size and so is limited to use in the local area.

The initial implementation of multicast connection management would set-up and modify connections one connection at a time. The sequence of operations would be as follows:

1. The requested connection is first set up using unicast connections and VCIs along its entire route, using a single service access point (SAP) at its source. This SAP does not differ from ordinary MSNL SAPs. The difference is the type of connection requested, which must be able to have multiple destination SAPs and be unidirectional to avoid flooding the source with reverse traffic. Unicast connections for user data in the reverse direction can be set up separately as necessary.

2. Using knowledge of the topology of the network and the maximum fanouts available at each switch, input ports in the switch which are shared by VCIs

in the same call can have this unicast VCI replaced by a multicast VCI. This is an optimisation performed by the MSNL master.

3. If this optimisation has been made, this unicast VCI can now be freed for reuse by other calls by issuing a `send_kill` signal. The fanout of the multicast call is incremented at the relevant switches as necessary. If this moving of a connection onto a new VCI is done before the connection is confirmed, then no cells need be lost or reordered.

If there is only one member of the call, there is obviously no need to use a MVCI, which may be a scarce resource, as noted above. However, to avoid having to change from a SAP to a multicast connection SAP as the group grows, a SAP with a multicast connection should be requested when the group is created. Of course, a SAP could be used with just a unicast VCI for the first member and then the VCI could be changed later, at the risk of cell loss and reordering.

This manner of post-processed optimisation of a multicast call according to the particular state of a network is likely to be non-optimal in networks where extensive information about the network's connections is widely known. In this case, routing algorithms such as those described in the next chapter provide multicast calls which are more optimally routed for the network as a whole. However, in situations where local optimisations can be effective such as at network bridges, this approach has the benefits of flexibility and the ability to easily add and delete members from the multicast group. To ensure that the VCIs could indeed be merged, it would be useful to have a parameter for the unicast routing procedures to indicate not only which call the destination wished to join, but which sources were being requested within the call. This would give the routing algorithm a set of known routes from the sources to use to add a new destination to the multicast routing tree.

One example of a factor in implementing multicast for MSNA which depends upon the design of multicast fabric used is which port controller has to be informed of the addition of a new destination to an existing multicast VCI. In the slotted ring design, the port at which the new copy appears is simply the next one downstream of the existing set of output ports. In other designs, this might be calculated in a different manner. Similarly, the exact choice of which MVCI to transfer a unicast VCI to in order to use hardware replication of cells is dependent upon whether copies are unique or identical (Section 7.2). Another example from the slotted ring design is the use of multiple rotations of the ring to produce more copies than the number of nodes on the ring (Section 5.7). In this case, the master would need to know the current number of nodes which were active on the ring to update the VCI remapping tables at the relevant port controllers.

Semantics of multicast connection set-up can be based upon particular instances of the 'at least (s, d)' mode of multicast group establishment. 'At least (s, d)' will only succeed if at least s of the S requested sources can be connected to at least d of the D requested destinations. Useful forms of this will include 'total', which will only succeed if all the requested connections can be set-up, i.e at least (S, D), 'at least $(1, 1)$' for testing connection establishment capabilities and 'at least $(1, D)$' and 'at least $(S, 1)$' for adding members to existing multicast groups.

7.5.1 Modifications to Existing Connection Management Procedures

The major modification necessary is the definition of how a SAP should appear to a multicast connection. Two SAPs currently provide the necessary end-points to set-up a bidirectional connection. To reuse the existing unicast connection model, the SAP should remain a single access point. The difference is that when a connection is requested from the master, it must now be requested as a unicast or multicast connection. A request for a multicast connection has to connect one SAP to multiple SAPs in a directed manner and according to the semantics of 'at least (s, d)' as described above.

The multicast group will be constructed from one or many of these one-to-many connections, each of which must be capable of having single SAPs added and removed from it. Much of the existing code for setting up connections between two SAPs will be able to be reused to implement a `join_multiple_SAP` procedure. Cell forwarding at the MSDL layer should be able to proceed as before.

The routing request must be modified to support the notion of the existing `call_id` and possibly of preferred routers for multicast connections. The lists of existing members of `call_id` and their capabilities should be accessible to the routing algorithm, possibly in a hierarchical fashion to encourage the use of a backbone for a multicast tree, which should make future additions of new members easier.

7.5.2 New Connection Management Procedures

The more important new connection management procedures required for the proposed modifications to the MSNA are listed below:

`call_id = open_call(group_size, duration, parameter_list)`
Obtain space (`group_size`) for a call `call_id`. The expected duration (datagram, short, indefinite) of the call can be used to chose whether the connection set-up is done optimistically and possibly for further CAC. The other attributes described in Table 7.4 are passed to the master in `parameter_list`.

`close_call(call_id)`
Remove each of the connections of `call_id` from the list of connections and free the space allocated for the call.

`add_member(call_id, src, dst)`
Add a member as a source or destination to the multicast group `call_id`. If `src` is equal to `dst`, then add connections to `dst` from all the other members of `call_id` and from `src` to all the other members of `call_id`. This is useful for maintaining fully connected multicast groups for conferencing applications.

`del_member(call_id, src, dst)`
Delete a member `src` or `dst` as a source or destination from the multicast group `call_id`.

`add_conn(call_id, src, dst, qos)`
Add a connection from `src` to `dst` in the multicast group `call_id` with QOS `qos`.

`del_conn(call_id, src, dst)`
Delete a connection from `src` to `dst` in the multicast group `call_id`.

`mod_conn(call_id, src, dst, qos)`
Change the details of the connection from `src` to `dst` in multicast group `call_id` according to QOS `qos`.

7.6 Summary

Modifications have been proposed for the use of an existing point-to-point architecture as the basis for the implementation of multicast capability for an ATM LAN. Some of the dependencies between an implementation and the designs of multicast switch used have been discussed. A series of procedures which would form the core of the implementation have also been presented.

Issues which are still under further consideration for multicast in MSNA are the problems of error recovery and retransmission requests flooding a source. Also under discussion is how to efficiently increase timeouts to cope with a wide range of response times from multiple hosts in a network.

8 Routing for Dynamic Multicast Groups

When the problem of routing multicast connections in networks has been previously considered, the aim has been to minimise overall costs, possibly subject to a delay constraint. The emphasis has been on examining the problem with a source transmitting to a fixed set of destinations (the multicast group) which are known when communication is established. As described in Chapter 1, there are clearly some applications for which the set of destinations will be dynamic, with destinations joining and leaving the multicast group during the existence of the connection.

Under these conditions the computation of an "optimal" spanning tree for each new multicast group may not be the best way to proceed. An alternative is to make modest alterations to an existing spanning tree to derive a new one. An extreme variation of this is to use minimal cost source to destination routing for each destination, effectively ignoring the existing multicast tree. This naïve approach leads to a simple implementation but clearly produces non-optimal spanning trees.

In this chapter we examine just how non-optimal these trees are in random general topology networks and conclude that they are worse by only a small factor. The factor is reduced still further if a hierarchy is imposed upon the random network to give a more realistic model.

8.1 Introduction

One of the more difficult aspects of providing multicast is deciding how the connections should be routed within a network to minimise or meet constraints for a series of parameters including delay, jitter and the link capacity used. Any scheme which is chosen must also be able to cope with heterogeneous networks [Chow91], where some nodes provide multicast capability and others do not. This situation is likely to arise in the wide area for economic reasons but should be of less concern in the local area, where homogeneous networks are easier for local system administrators to manage. Another problem is that a route may have only a limited degree of replication available at an intermediate node [Tode92].

In this chapter, multicast groups are considered to be a one-to-many connection of hosts and may be either dynamic or static in nature. The membership of a static group, once the group has been set up, remains unmodified until it is discarded. Unlike a static group, the membership of a dynamic group can vary with time.

The problem of connecting a subset of source and destination nodes, $\mathcal{S}$, in an undirected graph $\mathcal{G}$, using the minimal cost subgraph such that there is a path between each pair of nodes in $\mathcal{G}$, is known as the Minimum Steiner Tree (MST) problem. The MST problem is one of the classic NP-complete problems of graph theory [Gilbert68, Karp72]. The nodes which are not in $\mathcal{S}$ but which are used in the subgraph to connect the chosen nodes are known as *Steiner nodes*. The construction of a Steiner tree for multicast routing is one extreme form of multicast routing, as shown in Figure 8.1.

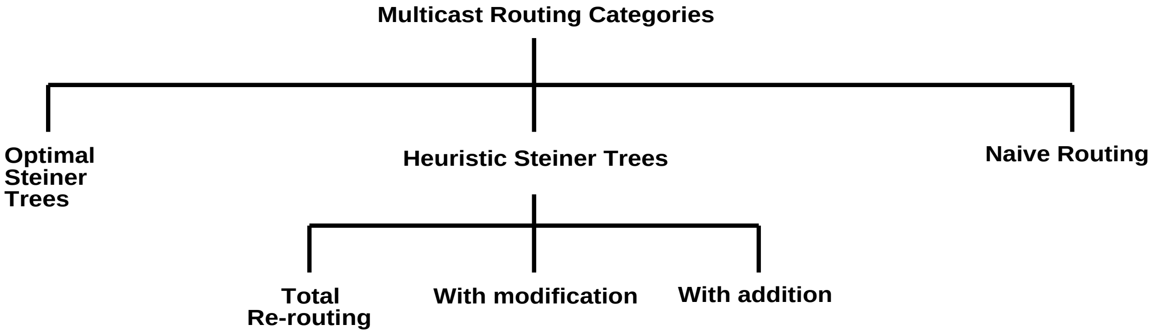


Figure 8.1: Multicast Routing Categories

Previous work on heuristic solutions (summarised by Winter [Winter87]) of the MST problem has generally taken one of the two following approaches. In the first approach, likely Steiner nodes are selected and then a minimum spanning tree of these nodes, together with the nodes in $\mathcal{S}$, is constructed [Rayward-Smith86]. In the second approach, a minimum spanning tree of the whole graph is pruned to remove superfluous nodes and then perturbed to try to include more Steiner nodes, which may reduce the overall cost of the spanning tree [Wall90, Waitzman88, Moy92]. It is one of the original versions of the latter approach by Kou, Markowsky and Berman [Kou81], referred to hereafter as KMB, which has been chosen as the basis for comparison with the simple algorithm described below.

The KMB algorithm is described in further detail elsewhere [Wall90] but briefly, consists of five steps. An all-pairs, shortest path graph of $\mathcal{S}$ is constructed and a minimum spanning tree of this graph is found. This tree is used as a template to add the relevant shortest paths in $\mathcal{G}$, possibly introducing new nodes. A minimum spanning tree of this new subgraph is then constructed. Finally, this new graph is pruned until all the leaves of the tree are destinations in the multicast group.

The performance of heuristic solutions can be described by two properties, their computational complexity and their *inefficiency*. By inefficiency we mean the ratio of the cost of the resulting spanning tree to the cost of the minimal Steiner tree.

The worst case computational complexity of the KMB algorithm is $O(|\mathcal{S}||\mathcal{G}|^2)$, where $|\mathcal{S}|$ and $|\mathcal{G}|$ are the number of nodes in $\mathcal{S}$ and $\mathcal{G}$ respectively. It has a worst case inefficiency which is bounded by 2(1-1/l), where l is the number of leaves in $\mathcal{S}$. The mean inefficiency of the heuristic algorithms is generally considerably better than this, often less than 1.05 [Waxman88]. The KMB MST heuristic algorithm, whilst not having a computational complexity as low as others, is relatively simple to implement and shows similar inefficiency characteristics when compared with the other more complex heuristics of both approaches [Jiang91, Waxman88]. The mean performance of the KMB heuristic is close enough to the optimal solution to permit its use for the purposes of comparison with the simple algorithm introduced in Section 8.3.1.

Whilst there are many parameters which are important in setting up a multicast connection, most of which involve some form of heuristic solution to the MST problem for various factors, the dynamic aspect of many multicast connections means that the time and effort expended in setting up a near-optimal solution is likely to be wasted soon after completion, when the configuration must be changed yet again. This is especially true as the size of $\mathcal{S}$ grows or the rate of change of group membership increases.

As shown in Figure 8.1, there are three ways in which heuristic Steiner trees may be used for routing multicast groups whose membership is dynamic. One is to re-route the whole tree whenever the group membership changes. This may have repercussions for members who remain in the group, since there may be a disturbance in the communication, and in an ATM network, such a change may allow cells to arrive out of order at the destination. A variation of this method is to permit partial or local rearrangement of the tree when modifications to the membership occur. This would appear to the same reordering drawback as total re-routing.

Yet another approach is to begin with an optimal (or near optimal) tree and make minimal changes to it as group membership changes — in the extreme causing no disruption to members who remain in the group. This is consequential to the approach described by Waxman [Waxman88]. As membership in the group changes, one would expect the inefficiency of this method to degrade and indeed simulation studies have shown occasional 'spikes' of very high inefficiency [Waxman92].

A third category of multicast routing is shown on the right in Figure 8.1. This is to choose a suboptimal tree which will be resilient to changes. In the extreme case, finding the minimum source to destination path for all destinations independently and taking the union of these paths produces trees which will be resilient to changes, but which may be suboptimal. By resilience we mean that the spikes of inefficiency should be much smaller, since the inefficiencies are independent of the order in which the modifications occur. The computational complexity of a single-source shortest paths algorithm, such as that due to Dijkstra, is $O(|\mathcal{G}|^2)$ but can be reduced to $O(e \lg |\mathcal{G}|)$ for sparsely connected graphs, where e is the number of edges in $\mathcal{G}$.

We consider that the scarce resource in future multicast networks will not be the bandwidth, but instead will be the number of signalling messages which must be generated and, in ATM networks, the need to ensure that no reordering of cells occurs. In the rest of this chapter we examine the performance of the third category of multicast routing on randomly constructed networks undergoing stochastic addition and removal of destinations from a multicast connection. The use of well-known routing algorithms such as minimum source-destination path routing also provides a simple way to involve the requisite constraints such as delay, bandwidth and reliability, since these are relatively well understood for point-to-point routing [Wang92a, Wang92b].

8.2 Network Models

To ensure that simulations of the effects of the different routing algorithms were independent of the idiosyncrasies of any specific network, random graphs were constructed. These were developed so that their connectivity characteristics approximated those observed in real networks.

8.2.1 Random Graphs

The first network model to be considered, following Waxman [Waxman88], was one where graphs were constructed by distributing n nodes across a Cartesian coordinate grid. The location of each node had integer coordinates and multiple nodes were permitted to exist at any location. The edges were added to the graph by considering all possible pairs (u, v) of nodes and using the probability function

$$P_e(u, v) = \beta \exp \frac{-d(u, v)}{\alpha L} \tag{8.1}$$

to create an edge, where $d(u, v)$ was the Euclidean distance between the nodes' locations, L was the maximum possible distance between two nodes and α and β were parameters in the range $0 < \alpha, \beta \leq 1$. A large value of α increases the number of connections to nodes further away, whilst a large value of β increases the number of edges from each node. The cost of an edge was defined simplistically as $d(u, v)$, the distance between its nodes, being a typical factor in routing problems. Another typical factor would be the call charging algorithm chosen by a network provider. Each graph was tested to ensure that only one connected component existed, i.e. that a spanning tree existed which covered all the nodes in $\mathcal{G}$. Graphs were generated until one with a single connected component had been created.

This method of graph construction produces graphs which model some aspects of real networks. For instance, values of 0.25 and 0.2 for α and β provide graphs which have the appearance roughly resembling that of geographical maps of the major nodes in the Internet. However, one problem with the model as originally proposed is that as the number of nodes increases, the number of edges from each node, or the *degree*, tends to increase as well. The graph on the left of Figure 8.2 shows a 150 node random graph with a relatively large number (approximately six) of edges from each node. The modification proposed and implemented is to scale P_e by a factor related to the number of nodes, $|\mathcal{G}| = n$, in the graph. This is successful since the probability of each edge existing is independently distributed for each node and hence forms a binomial distribution, which has a mean directly proportional to the number of nodes in the graph. To ensure that the mean degree of each node remains constant, another scale factor related to the mean distance between two

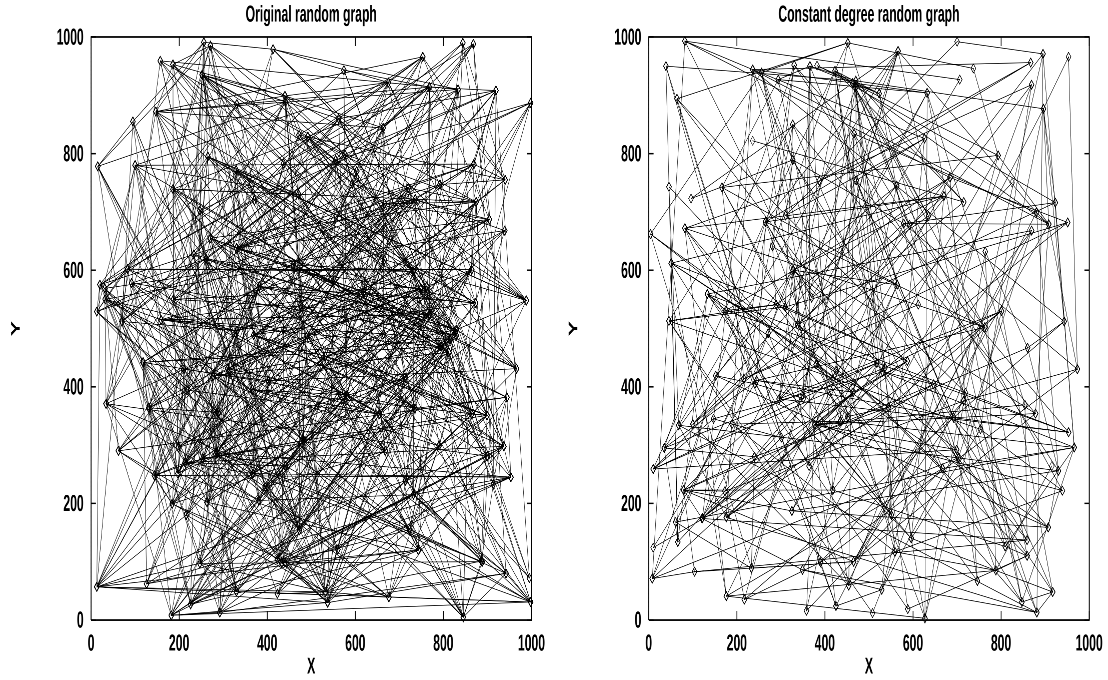


Figure 8.2: Original random graph and constant degree version

random points must be introduced. For $\alpha = 0.25$ and $\beta = 0.2$, this factor, k, is approximately 25, by measurement. The modified probability function is shown in equation 8.2, where $\overline{e}$ is the mean degree of a node and n is the number of nodes in th graph $\mathcal{G}$.

$$P_e(u,v) = \frac{k\overline{e}}{n}\beta \exp \frac{-d(u,v)}{\alpha L} \tag{8.2}$$

The graph on the right of Figure 8.2 shows a 150 node random graph with a mean degree of three edges from each node. As can be seen, the constant degree random graph network model provides networks which more closely resemble real ones in their connectivity.

8.2.2 Hierarchical Graphs

The second proposed model of networks attempts to model the hierarchy of networks that exist in reality, with multiple LANs connected to a WAN, or a LAN comprising many small switches. A proportion of the n nodes were used as a core, connected in the manner of the modified random graphs of Section 8.2.1 using the scaled version of the edge probability P_e and checking for the existence of a single connected component. The remaining nodes were divided into equal sized clusters, which were connected internally in the same fashion as the core. The clusters were then attached by a fixed number of edges, typically two, to randomly chosen nodes in the core, not necessarily unique. Both the size of the central core and the size of the clusters on the Cartesian grid were specified. The final parameter was whether the graph was

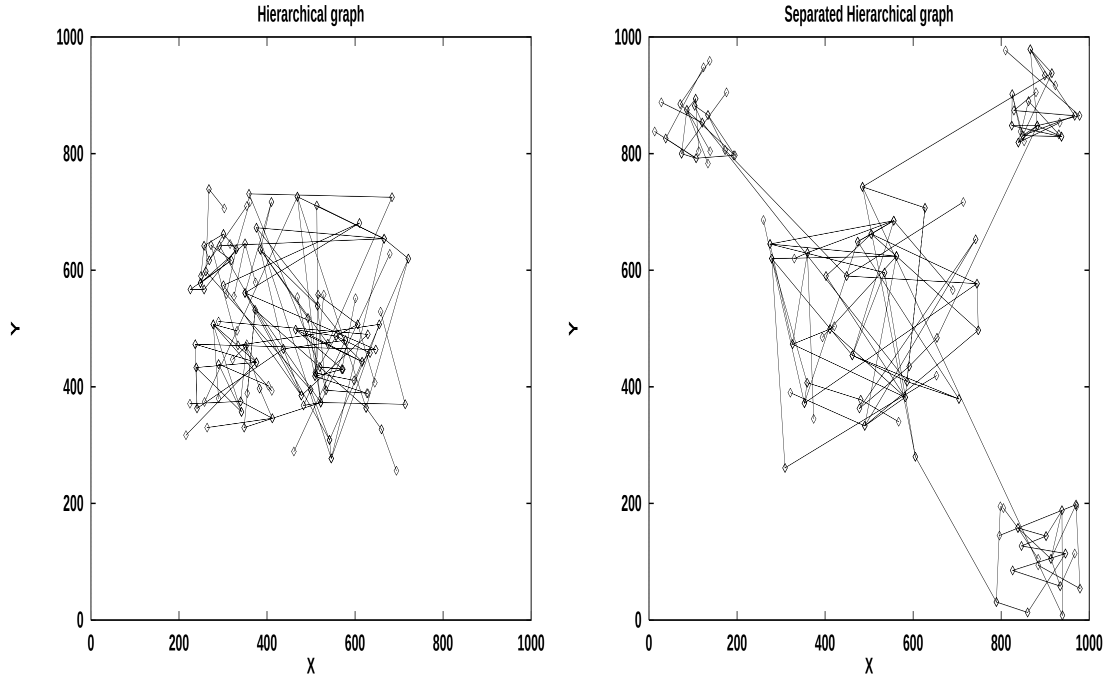


Figure 8.3: Hierarchical graphs without and with separation

separated or non-separated, i.e. whether the clusters were permitted to overlap the central core. In the simulations presented in Section 8.4, non-separated graphs were used and the clusters were permitted to overlap each other, modelling the existence of multiple LANs at a site.

Figure 8.3 shows typical network models constructed using these guidelines. Both graphs contain 100 nodes, 40 of which are in the core and the remaining 60 divided into three clusters of 20 each. The clusters were limited in area to squares of side 200 in a square grid of side 1000. The core was limited to a square of side 500 centred in the grid. The mean degree of each node was set at three and each cluster was connected to the core by two edges.

Further network models based upon random graphs are proposed for future work in Section 9.3.4.

8.3 Modification Algorithm

After the graph had been constructed, one node was chosen at random as the notional source of the multicast connection. Another different node was also chosen and the shortest path was found between them. Now that the multicast connection existed, further destinations were added or deleted from the multicast tree using the following probability function.

$$P_c(k) = \frac{\gamma(n-k)}{\gamma(n-k)+(1-\gamma)k} \tag{8.3}$$

where P_c is the probability that the modification is an addition, k is the number of destinations in the multicast connection, n is the number of nodes in the graph and γ is a parameter in the range $0 < \gamma \leq 1$. γ is the mean fraction of destinations/nodes in the connection. If $\gamma = k/n$, then $P_c(k) = 1/2$, if $k/n < \gamma$, then $P_C(k) > 1/2$ and if $k/n > \gamma$, then $P_C(k) < 1/2$. A modification was defined as either an addition or a deletion.

8.3.1 Algorithms for the Addition of Nodes

Waxman's original algorithms [Waxman88, Waxman92] for connecting new destinations to an existing multicast connection were:

Greedy Find the nearest node already in the connection to the node to be added and connect the two.

Weighted Greedy Take a proportion of the weight from the greedy algorithm and a proportion (e.g. 0.3) of the distance from a node known as the notional owner (the source) of the connection. Use the minimum of the sum of these two weights to decide which existing node in the multicast connection to route to the new node from.

Other proposed costs include the distance from the centre of the tree (see [Aho83], pp. 213–214 for a definition of centrality) or the number of connections attached to a node.

The above algorithms all fall into the category of heuristic Steiner trees with addition, as shown in Figure 8.1. The algorithm used in the simulations described below is one in which the tree which is constructed is independent of the order of the modifications. The particular algorithm chosen was to find the shortest path from the source to the node which had been chosen to be added and to add all the nodes along that path into the multicast connection as Steiner nodes.

An example of another simple algorithm would be to find the shortest path from the source using dynamic costs on the edges. If k paths to destinations used an edge in the graph, then its cost would be scaled by k, encouraging Steiner trees with minimal diversion until near the destinations.

8.3.2 Algorithm for the Deletion of Nodes

Multiple destinations were allowed to exist at the same node in the graph, modelling multiple destinations on the same LAN, for instance. If a node chosen for deletion was found to have multiple destinations, then the number of destinations was simply decremented by one. However, if there was only one destination at the node chosen for deletion and the node was a leaf, then, after deletion of the node, its immediate predecessor was considered recursively for deletion. The recursive deletion ceased when a node was encountered which was part of an internal path to another destination. In this case the node was left in the Steiner tree, but marked as deleted for later action if the other dependent destinations subsequently became deleted. The source node was never deleted in this scenario, and if the lower limit of two nodes in the connection was reached and the modification function indicated the deletion of a node, then no action was taken.

This series of decisions and recursions is realistic, since routing tables at each node have to contain enough information about the numbers of multicast destinations using the node to permit timely recovery of resources when a destination is removed from the multicast connection.

8.4 Simulation Results

Multiple simulations of the above models were performed and the results are shown in Figures 8.4 to 8.11. For each run a graph was generated according to one of the network models described in Section 8.2 and 2000 modifications were simulated. The ratio of the cost of the multicast tree obtained by the simple algorithm to that of the KMB Steiner heuristic tree constructed afresh for the current set of destinations was evaluated every 10 modifications. This ratio is close to the inefficiency mentioned in Section 8.1 and is hereafter treated as such. Ten series of the generation of a graph and a run of modifications were performed for each point to obtain the Student t-distribution confidence intervals shown in the graphs. Longer runs indicated little reduction in confidence intervals in most cases.

Figure 8.4 shows how the inefficiency changes with the number of modifications made. The graph shown on the left is for a 100 node constant degree random graph with a mean of 50 destinations in the multicast connection and a mean degree of three. The graph on the right shows how the number of nodes involved in the multicast connection changes during the simulation. The shape of these graphs is typical of the changes in the inefficiency and number of nodes as modifications are made for many different sizes of graphs, fractions of the graph nodes involved in the multicast connection and different network models. The spikes are not so sharp as those seen by Waxman [Waxman92] due to the nature of the naïve algorithm, since the multicast tree constructed by it is independent of the order in which the modifications occurred. The inefficiencies which do occur are due to occasions when a new node being added is close to one already in the multicast tree, but a (partially) disjoint path is used to reach it and the path is only slightly shorter than the path to the new node via the existing node.

Figure 8.5 shows how the mean and maximum values of the inefficiency change with an increasing number of destinations involved in the multicast connection for various sizes of the original random graphs network model. As can be seen, increasing the graph size results in an increase in the inefficiency whilst the changing proportion of destinations makes only a minor difference to the inefficiency. The inefficiency is unity when only two nodes are involved in the connection, since the shortest path between the two is also the (trivial) Steiner tree. If all the nodes are involved in a connection, the KMB tree is the minimum spanning tree of the network. This rarely occurs in the graphs shown since multiple destinations can exist at a single node. The increase in confidence intervals as the size of the graph

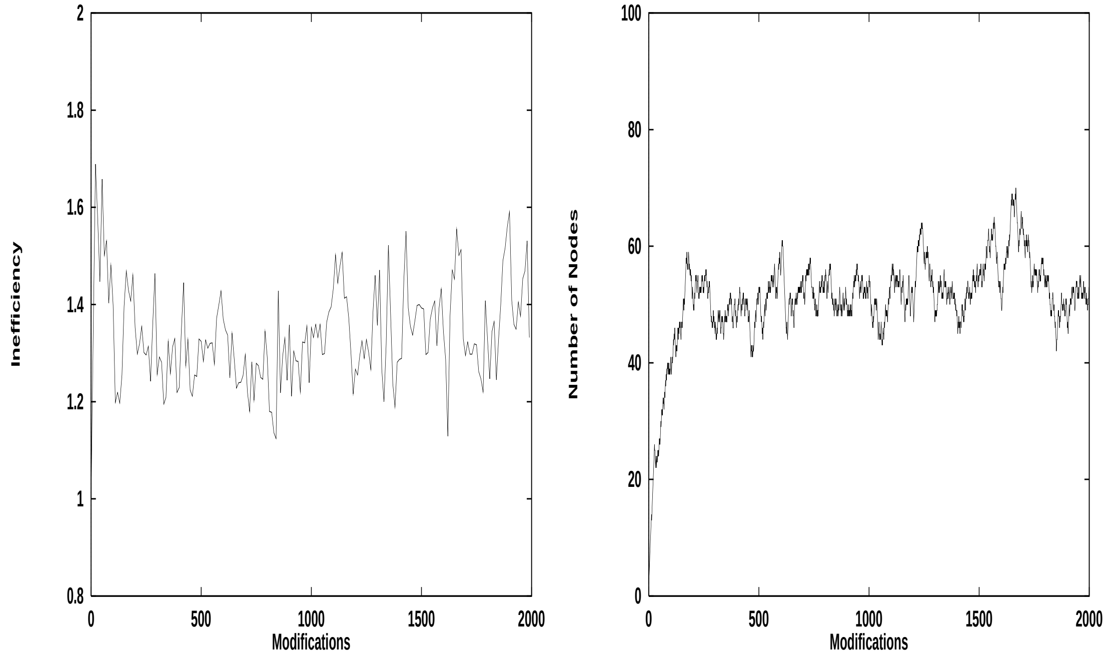


Figure 8.4: Inefficiency and number of nodes for a single run

increases is probably due to the increased possibility of the existence of relatively long paths mostly separated from the rest of the multicast tree, which affect the performance of the simple algorithm drastically. This effect can also be seen in the graph of the maximum values, though the general trend in both cases is that the inefficiency increases as the number of nodes, and in this case the degree, increases.

Figure 8.6 shows how the mean values of the inefficiency changes with an increasing number of destinations involved in the multicast connection for various sizes of random graphs with the constant degree network model. As can be seen in the graph on the left, increasing the graph size results in little change in the inefficiency if the degree is constant. It also shows that the inefficiency is relatively independent of the number of nodes in the network for a given proportion of destinations involved in a multicast connection. The graph on the right in Figure 8.6 shows the results of varying the degree of a 100 node constant degree random graph. It suggests that the performance of the simple addition algorithm is indeed directly related to the mean degree of the graph. Figure 8.8 confirms that this relationship is approximately linear.

Figure 8.7 shows how the mean value of the inefficiency changes with an increasing number of destinations involved in the multicast connection for various sizes of the hierarchical graphs network model. As can be seen in the graph on the left, increasing graph sizes again result in little change in the inefficiency. The values of the inefficiency obtained with the hierarchical model are less than those obtained with the constant degree model, probably due to a more limited set of inter-cluster

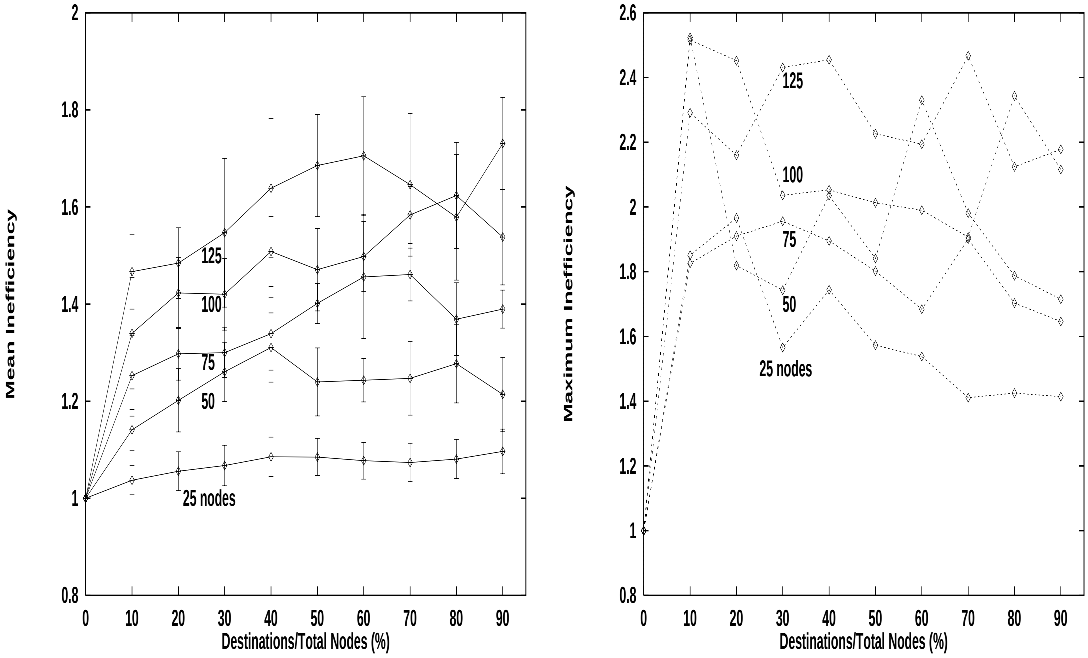


Figure 8.5: Mean and maximum inefficiency of original random graph

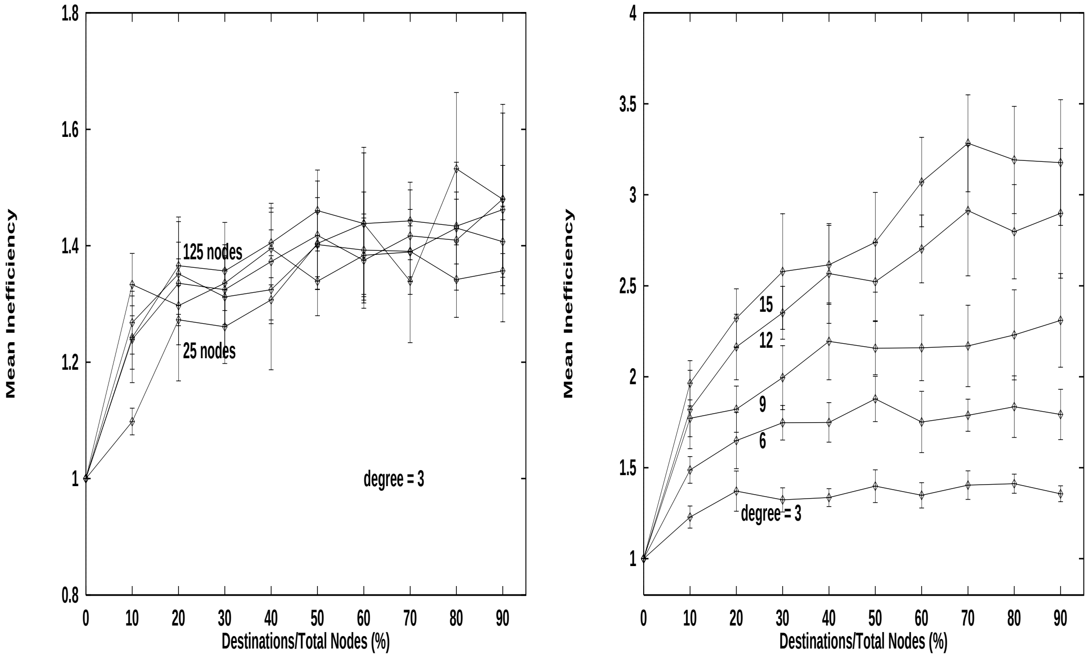


Figure 8.6: Inefficiency of constant degree random graph for various graph sizes and degrees

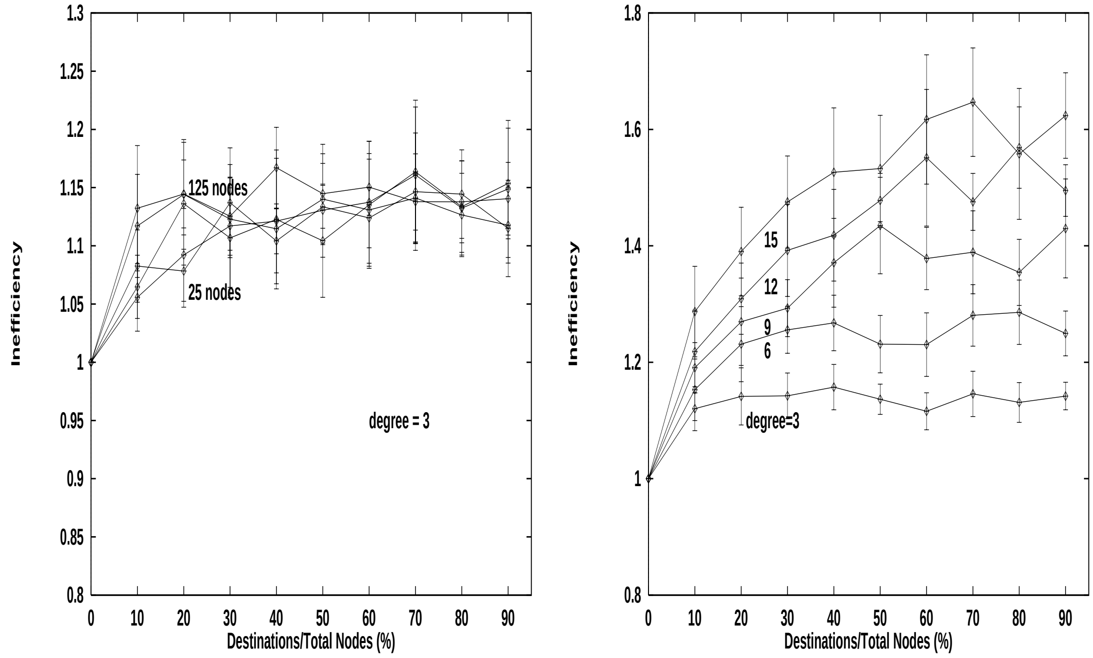


Figure 8.7: Mean Inefficiency of non-separated hierarchical graph for various sizes and degrees

paths from node to node, so the naïve algorithm is more likely to choose the same ones as an optimal MST algorithm. The graph on the right in Figure 8.7 shows the effects of varying the degree of a 100 node non-separated hierarchical graph. It suggests that, as with the constant degree model, the performance of the simple addition algorithm is related to the mean degree of the graph.

Figure 8.8 confirms the approximately linear relationship between the degree and the inefficiency for both network models. The values shown are for 20% and 60% destinations/nodes in the 100 nodes of the graph being involved in the multicast connection (on average). As the degree increases, the number of available paths between any two nodes increases and hence the probability of the shortest path tree coming close to the heuristic KMB tree decreases and the inefficiency increases.

Figures 8.9 to 8.11 all refer to random graphs with 100 nodes and give further insight into the difference in performance between heuristic Steiner tree algorithms such as KMB and the naïve but robust shortest path algorithm.

Figure 8.9 shows how the maximum path length varies with the percentage of destinations/total nodes and also with the mean degree. As may be seen, the paths chosen by the naïve algorithm are indeed shorter than those chosen by KMB and the length of the paths decreases as the degree of connectivity of the graph increases. The maximum path length tends to increase in KMB trees as the proportion of destinations increases. In trees constructed from shortest paths, the change is less marked. This is probably due to the extended nature of the trees constructed from

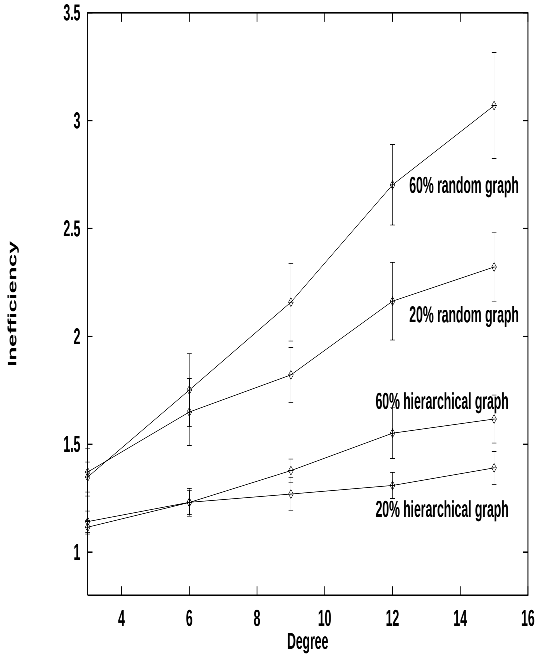


Figure 8.8: Mean Inefficiency for both network models and various degrees at 20% and 60% destination nodes/total nodes

pruned minimum spanning trees, compared to the more compact nature of shortest path trees.

Figure 8.10 shows the maximum hop count for each path in the two sets of trees, and how it varies with the percentage of destination nodes/total nodes and also with the mean degree. Again, very little difference is seen in the values from the shortest path graphs as the proportion of destinations changes. A small decrease with respect to the degree can be seen in the graph on the right. The changes for the KMB graphs are even more marked than the maximum path length seen in Figure 8.9, though the reasons for them are much the same. The maximum hop count for the KMB trees is observed to change very little with respect to the degree of the underlying tree, increasing slightly as the degree increases.

Figure 8.11 shows the mean bandwidth used by each tree, defined as mean of the reciprocal of n_e for each edge in the multicast tree subgraph, where n_e is the number of paths which use the edge e. The bandwidth is scaled to the bandwidth which would be required if no multicast were possible except at the source. Thus an edge which is only used by one path has a bandwidth of 100%, compared to one used by two paths, which uses 50%, and so on. As may be seen from the figure, the KMB algorithm gives a substantial reduction in the bandwidth used, compared to the shortest path algorithm. The reduction is again dependent upon the percentage of destination nodes/total nodes involved in the multicast tree, this time improving as the proportion increases. The naïve shortest path algorithm on the other hand gives

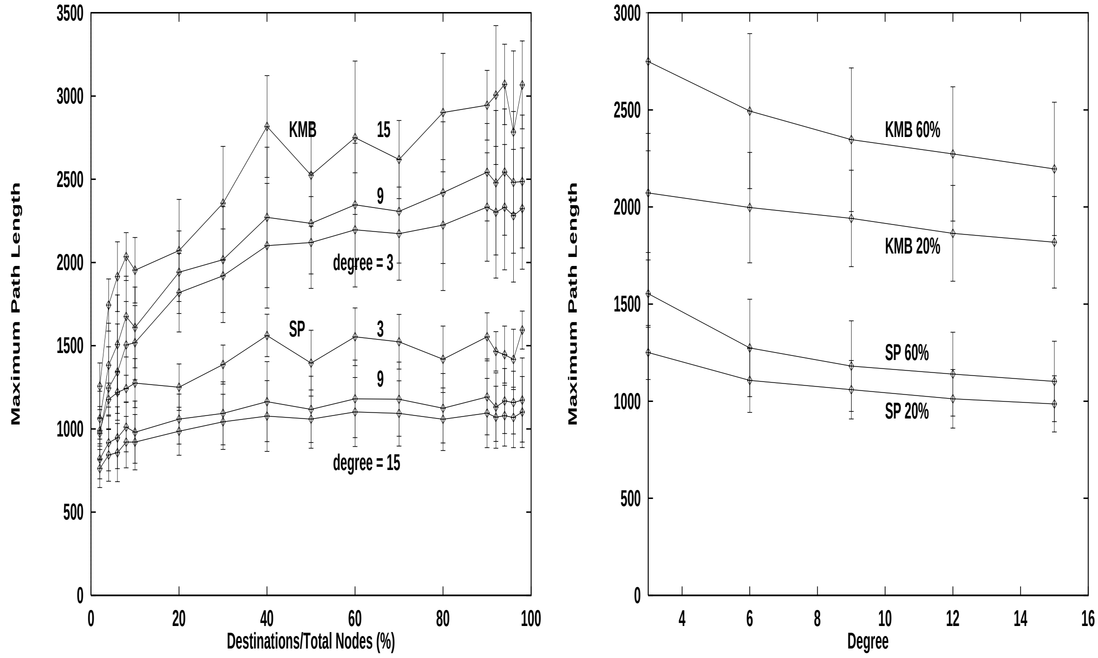


Figure 8.9: Maximum path length for Shortest Path (SP) and KMB trees with various percentages of destination nodes/total nodes and degrees

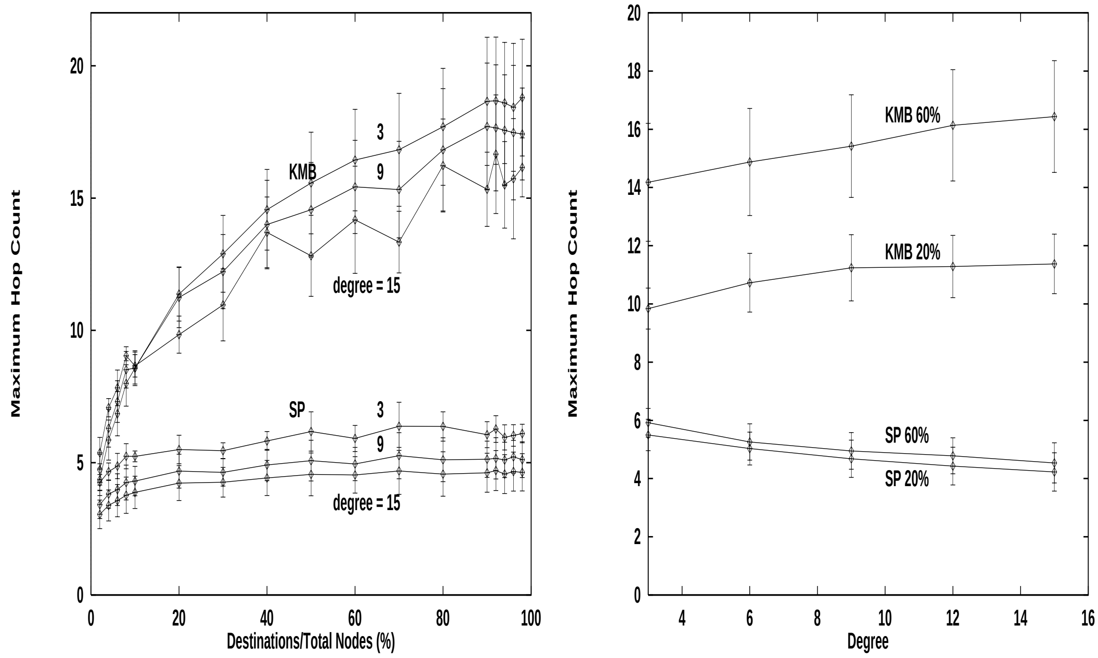


Figure 8.10: Maximum number of hops for Shortest Path (SP) and KMB trees with various percentages of destination nodes/total nodes and degrees

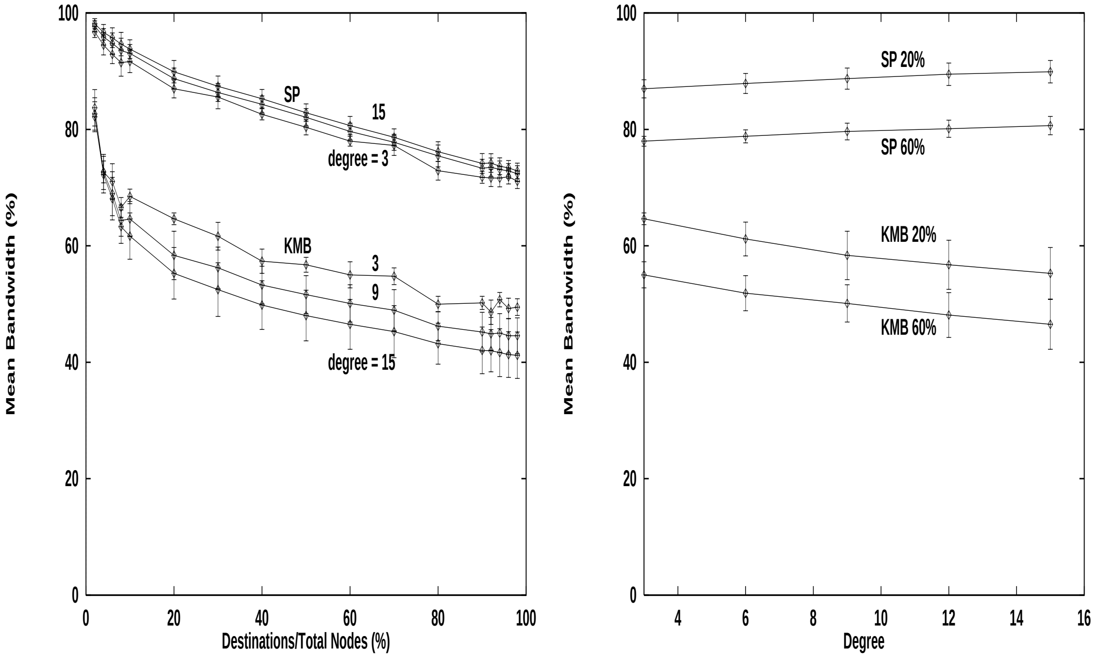


Figure 8.11: Mean bandwidth relative to unicast network, for Shortest Path (SP) and KMB trees with various percentages of destination nodes/total nodes and degrees

a relatively small sharing of bandwidth resources, though it is almost independent of the mean degree of the graph.

8.5 Routing Optimisations

Whilst the simple algorithm described above provides a set of multicast connections which appear to tend to a fixed multiple of the cost of an optimal solution in the long run, the erratic nature of call modification means that there may be opportunities for an intermittent process to exist, whose purpose would be to examine the existing tree and construct a set of modifications which would transform the tree into a near-optimal Steiner tree. The algorithm used for this process would be a modification of one of the previously mentioned heuristic algorithms, possibly limited by the number and type of routes that it would be permitted to alter. Alternatively, one of the heuristics could be used as they were presented and the whole multicast tree could be periodically torn down and set up again, probably closer to an optimum solution.

The timing of this optional process of improvement could be controlled by the source to avoid re-routing during transmission and the consequent risk of reordering. It might be encouraged by a possible reduction in the charge for the multicast connection in a WAN.

8.6 Summary

The proof [Karp72] that multicast routing was equivalent to constructing a Steiner tree, which is an NP-complete problem, lead to various heuristics being proposed, with worst case performance generally twice that of the optimum solution, but with average performance much closer to the optimum. These heuristics were modified to optimise the tree for various parameters, all of which are important, but none of which is crucial in itself.

In the long-term operation of any real network which provides multicast capability, the dynamic nature of multicast applications is such that the previously proposed heuristic algorithms are likely to provide solutions whose success is only ephemeral.

The simulations in Section 8.4 imply that even one of the simplest routing algorithms can provide a mean inefficiency (for a mean number of three edges per node) which is within a factor of 1.5 of typical heuristic solutions, which themselves have an upper-bound of being within a factor of 2 of the optimal solution (and typically within a factor of 1.2). More realistic network models such as the hierarchical model indicate even better performance.

9 Conclusion

The problems associated with the provision of multicast in networks, particularly ones which use ATM as their fundamental transfer mode, have been considered in this thesis. This final chapter summarises the major conclusions which can be drawn from each of the preceding chapters. It also includes a personal view of some likely environments and possibilities for the use of multicast in future networks. The final section is an outline of some of the problems which remain to be solved for networks with multicast capability.

9.1 Summary

The provision of multicast in networks is definitely not a trivial extension of unicast capabilities (which must exist in all networks by definition), since future traffic rates will overload any network which attempts to provide multicast in a naïve fashion. It is also rarely efficient to implement multicast as a restriction of some innate broadcast capability, which may be a by-product of the implementation of a particular network, except in a very limited geographical area.

To reduce the number of interactions with higher layers in the protocol stack and hence improve the performance characteristics, it is important to replicate the information to be multicast as close as possible to the physical layer. The bandwidth of the network is also used more efficiently if the replication of the information occurs as close to the destinations as possible. These desires are not always compatible with the needs of some applications and trade-offs will have to be made. Thus, a network will have to provide a low-level multicast service for the applications which can use it, but higher level multicast facilities will also have to be capable of operating gracefully without this. Future applications will make use of multicast from a combination of layers, depending upon the nature and distribution of the hosts involved in the multicast group.

The design of the switches and transmission interface hardware to support networks based upon asynchronous transfer mode has been an area of intense research interest for over five years and commercial products are now becoming readily available. Some of the theoretical designs which support replication in hardware are complex and would prove difficult to implement on a large scale. There are many design issues to be considered for any switch design, but the provision of multicast adds to these the issues of inter-copy delay, potentially harmful traffic patterns and non-determinism as to which output port of the multicast fabric replicated cells leave from.

Switches may be categorised as space division or time division or a series of stages of both. Replication can be categorised in a similar fashion, with time division designs further divided into shared memory or shared medium multicast switches. The design which was investigated in this thesis is a simple one based upon a slotted ring for just the replication of cells. The basic design complexity scales linearly with the size of the switch. Simulations, analysis and experimental results have shown that the fairness problems and delays associated with it are not major drawbacks, at least in a small-scale switch. The ability to add multicast capability to a switch at a later date or to selectively populate a switch's inputs for multicast capability are also attractive features of this design.

The construction of the multicast fabric for Project Fairisle proved the usefulness of a modular design, making debugging by comparison of hardware a feasible tool. The essential simplicity of the design was confirmed by the small size of the PCB which could be used to implement the fabric. The incorporation of telemetry facilities into the design at a late stage was eased by the use of FPGAs with their greater flexibility, analogous to the introduction of stored program computers in the early 1950's. The telemetry functionality made the measurement performance of the multicast fabric significantly more reliable than it would have otherwise been.

The multicast fabric design presented in this thesis is a design which in its simplest form requires very little additional hardware. Yet it presents a clean solution to the problem of replicating cells in (near) parallel at the switch level in ATM LANs. This solution is sufficient in the local area for the amount of multicast traffic which is expected to be present, even if up to 20% of the cells are multicast cells.

The separate examination of the problem of routing multicast connections in Chapter 8 suggests that the use of current unicast routing algorithms might be sufficient for many applications. This has the added advantage of being able to use the large amount of existing knowledge about the practical performance of such algorithms in current networks. One of the crucial issues in the choice of a routing algorithm for a multicast connection will be its ability to preserve cell order, both during route reconfiguration as members join and leave the multicast group and also in the more abstract sense of causal order. The problems associated with efficient multicast routing make it one of the most complex areas in the study of networks. Any truly successful algorithm is likely to have a substantially pragmatic approach to the matter, especially when considering inter-domain routing and even routing in the wide area using different carrier companies. Approaches such as the one taken here enable a wide variety of constraints and desired minimisations to be taken into account when designing multicast routing algorithms.

9.2 Discussion

It is instructive to consider some of the reasons why multicast is currently used relatively infrequently in existing networks, both in the wide area and the local area. The likely increase in the use of multicast in the future will be due to the cumulative effect of changes in some or all of the following areas.

Hardware

Many existing LANs use a shared medium to transfer information in a point-to-point manner. This is easy (though it scales badly with size) to extend to include multicast, since the information is reaching all possible destinations anyway.

Future networks based upon ATM are moving away from the use of a shared-medium. This is because technological improvements in the last decade have meant that it is now feasible to build space division switches which operate at full line speeds (which have themselves increased). This increases the proportion of the available bandwidth which can be used for point to point communication. Potential security problems are also greatly reduced since all traffic can no longer be intercepted by all receivers.

In space division switches the replication necessary to support multicast must be explicitly designed with the hardware. This is an extra level of complexity which has not been catered for in all switch designs, which will slow the use of multicast in the first generation of ATM networks.

Control

Apart from the difficulties of replication of the information in hardware, another significant barrier to the use of multicast in current networks has been the lack of clearly understood requirements concerning what a multicast service must provide. This is gradually being remedied, and adding and deleting destinations, merging and splitting groups and constructing connectivity graphs from smaller connections are all now known requirements.

Other areas of difficulty such as setting up routes for multicast connections have also received attention in the past decade, with several useful heuristic algorithms being described, together with some work based on the increasingly better understood characteristics of real networks.

The level of understanding of what is needed to control multicast connections and services seems to have now reached a critical state and should provide further improvements for multicast networks in the near future. This should dispell some of the sense of overwhelming complexity which has faced network architects when confronted with multicast in the past. This should in turn lead to an increase in the support for multicast communication.

Traffic Characteristics

Just as most traffic in an internetworking environment is local in its distribution due to nature of most applications, most traffic can be expected to remain point-to-point in the future, since most applications are by their nature point-to-point. The difficult question to answer is what fraction of it might reasonably be expected to be multicast. Estimates vary widely but imagining a desktop environment with access to video conferencing, multiple distributed databases and cable television, an estimate of between 5% and 20% would seem to be reasonable.

Part of the increase in multicast traffic will come from a change in the way we consider networks. As multicast becomes a standard form of communication, more applications will be constructed which make use of it.

Size

It seems likely that LAN ATM switches will be small, at most 32×32 inputs and outputs, to fit with the usage in a typical local area such as a single building. At this sort of size, hardware scaling problems are not a great difficulty and the provision of multicast is a relatively straightforward task. Since it is straightforward, the use of techniques such as multi-stage interconnection networks seems unnecessarily complex. Another factor in the local area is that most switches will probably be capable of providing multicast, if only for the increased ease of control and maintenance found in a homogeneous environment.

In the wide area, however, ATM switches may be as large as 1024×1024, for which it is far more difficult to provide multicast capability on every output port. Solutions to this will require the ability to provide multicast for a subset of the output ports and to use routing algorithms which can cope with heterogeneous networks, where not all switches provide hardware multicast.

This suggests that the first area where the amount of multicast traffic increases in newer networks will be in the local area, with local conferencing providing the impetus for ensuring full support of multicast in LANs. Wide area multicast will be much less frequent until the provision of services such as nationwide video distribution by the local carriers (or other customers) becomes a reality.

9.3 Future Work

Once the decision to provide multicast capability in hardware has been made, there are several difficult problems which remain to be solved. Some of these were touched upon in Chapter 2. The following four areas will be useful areas for research in the near future.

9.3.1 Quality of Service

The guarantee of a certain synchronisation QOS for the duration of a multicast connection will require that each media is carried on a separate VCI and possibly that different versions of the same media stream should exist (Section 2.4). These different versions may originate at the source or may come from some internal network transcoding service, for instance a video 'degrader' to reduce the bandwidth used by a video service or a service to translate between different formats of television signals.

The construction of a multicast connection from multiple different media and versions of the media, whilst guaranteeing a certain QOS for each destination requires a knowledge of the routing algorithm used and possibly some understanding of how the multicast connection is likely to change in the future. There is likely to be a desire to limit the ways in which such a multicast connection can change, for instance, a maximum size requested at set-up, in order that the problem of guaranteeing a set of QOS requirements will have some bounds. The different ways in which a multicast connection may be usefully restrained at set-up seem to be so far unexamined.

9.3.2 Call Acceptance

Associated with the issue of QOS is the problem of deciding which criteria to use when considering a multicast connection set-up request. Deciding which criteria to use for unicast connections is still an area of active research and the existence of multicast further complicates the matter.

When a unicast connection is requested, information from management entities is passed to the control plane to test the proposed route, link by link, to see whether the QOS requirements can be met. If a multicast connection is requested, much the same process occurs. Addition of a new destination to the connection can do the same thing, up to the point where the new route joins the existing multicast tree at a multicast point. Note that even if a suitable point for multicast exists, it might be overloaded (or limited to guarantee unicast traffic performance characteristics). In this case, the addition could only succeed if there were spare capacity at some other point. The construction of multicast routing algorithms from combinations of unicast algorithms, as seen in Chapter 8, also means that multicast CAC may considered as a series of unicast call decisions.

The proposed estimates for use by call acceptance algorithms, such as effective bit rate or burstiness, are usually discussed on a per-connection basis. The effective bandwidth of a multicast connection can only be evaluated as a function of the total load imposed upon the network. An effective definition of this load for practical networks is still under discussion.

9.3.3 Call Charging

One topic of interest to the providers of wide area networks in particular is how the traffic may be accounted and charged for. Whilst no definite plans for unicast connections have yet been made public, the issue of how to charge for multicast is even less clear. Charges can be made for call establishment, call holding and call usage. A holding charge may be necessary since some network resources are being consumed by the existence of a virtual connection. Whether usage in charged for on a per-cell or per-VCI basis and whether according to predicted use or actual use are still issues for further discussion.

One scheme is to charge whoever initiates a connection for the information sent. Thus if a destination wishes to join an existing multicast connection, it should be charged for the number of different connections which it receives information from (possibly all, since all can send to it in a many-to-many connection). The destination can also be charged according to the number and bandwidth of the media it wishes, or is capable, of receiving. If no multicast connection yet exists, the request by a destination to receive information should still be charged to the destination.

If a source initiates the connection addition, then it is charged for it, possibly according to the number of destinations in the multicast group or the effective bandwidth used in the network. The source(s) must be able to restrict the addition of new destinations, since they will be paying for the amount of network bandwidth they effectively use. It may be wise to do this by querying each source member of the multicast group in turn as to whether they will accept the charge for transmitting to a new member.

9.3.4 Network Models

Just as the two transfer modes traditionally used by telecommunications companies and computer communication networks have been widely different, so too has the growth of their respective networks. Telecommunication networks generally have been well-planned and consequently often possess a rigidly defined hierarchical structure and connectivity pattern. Computer networks on the other hand have developed in a more random fashion, with the hierarchies that exist seeming to have evolved more slowly. Further work developing better models of computer networks for the purposes of comparison of routing and other graph algorithms remains to be performed.

An interesting idea would be to model the growth of the Internet over the past decade in an attempt to predict the likely growth of B-ISDN in the next ten years. Unfortunately, the randomly generated network models described in Section 8.2 do not model real networks particularly well because real networks are randomly connected only in a very limited sense. Geographical, economic and administrative forces combine to impose a hierarchy on real networks. Connectivity costs tend to mean that many of the nodes in a large network are leaf nodes.

One way to reduce the number of edges from each node is to scale the probability function given in equation 8.2, so that the probability of adding another edge to a node decreases according to the number of edges which already exist at that node. Preliminary investigations have been made using a linear scale factor, S_L, where S_L is defined by

$$S_L = 1 - \frac{e_i}{E} \tag{9.1}$$

with e_i being the current number of edges connected to node i and E being the maximum number of edges permitted. A typical graph is shown on the left in Figure 9.1. The graph shown is a 50 node graph with $\alpha = 0.25$ as before and β increased to 0.8 to reduce the time taken to generate a fully connected graph. The maximum number of edges has been set to one going *away* from each edge during construction. Some nodes have more than two edges going away because a small number of edges have been added to connect separate components and multiple nodes can exist at a grid point.

A different scale factor, S_{EXP} can be defined exponentially as

$$S_{EXP} = \exp(\frac{-e_i}{K}) \tag{9.2}$$

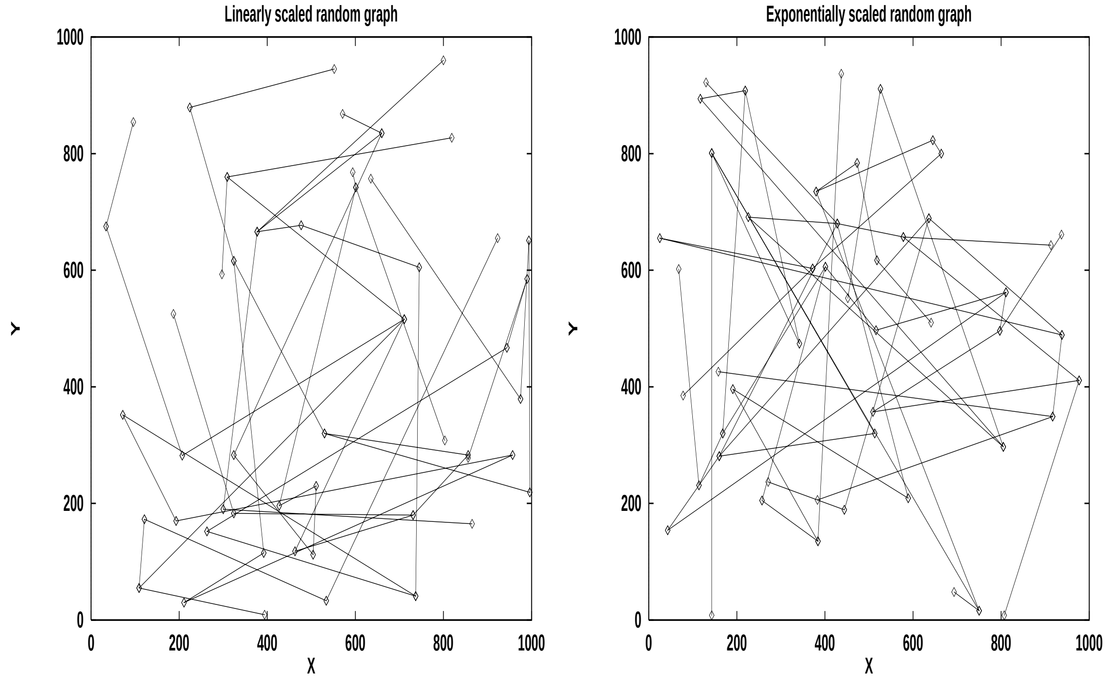


Figure 9.1: Random graphs with linear and exponential scale factors

where K is a fraction of the mean number of edges. A typical graph using S_{EXP} is shown on the right in Figure 9.1. Again, this is for a 50 node graph with $\alpha = 0.25$ as before and $\beta = 0.3$ as in the networks modelled in Chapter 8. The mean number of edges is three and the fraction of e_i taken was 0.1. It is interesting to compare the two graphs and note that although they appear similar, the exponential graph in fact has more nodes with a larger degree than exist in the linear graph.

This approach still has the drawback that geographical clustering and the hierarchical characteristics are not modelled. One possible solution to this is to abandon the probability distribution entirely and to create a hierarchy of networks using a different interconnection pattern. Three levels of network might be defined: the wide area backbone, metropolitan area clusters and heterogeneous local area networks.

Creation of a graph would parallel the growth of networks in reality. A backbone of a small number of widely distributed nodes would be created, first with each node being connected only to one other, then adding extra edges to model the robust mesh topology seen in real backbone networks. The merging of singly connected nodes to form a single connected component is rather like the operation of the well-known minimum spanning tree algorithm due to Prim [Aho83, pp.234–237]. After this, star-shaped clusters would be attached to nodes on the backbone. Lastly, different topologies of LANs (star, ring, fully-interconnected) would be added to the star at various points, possibly overlapping with each other. Different capacities could be assigned to each edge according to its level. In this way, more accurate models of real networks could be created for the purposes of simulation of network graph algorithms.

A Analysis

The work summarised in this appendix is part of a collaborative effort with Nigel Bean of the University of Cambridge Statistical Laboratory. It is described more fully elsewhere [Bean92, Bean93].

A.1 Related Work

Hui [Hui90a, Hui90b] has performed extensive simulations to confirm his and others analytical results concerning the performance of the cascaded multicast switches. In [Hui90b] various queueing disciplines with fixed length packets are considered and it is concluded that if multicast calls can be split, then a first-come, first-served approach to waiting packets provides the best saturation throughput and delay.

Lee [Lee88a] performs a rudimentary buffer overflow analysis of his network, assuming packets are discarded when overflow occurs. Assuming that every packet is a multicast one and that their fanouts are independent (which is valid if a large number of sources are multiplexed onto each link), the results obtained suggest that the overflow probabilities are very low for small fanouts, but rise rapidly as the fanout increases, both for fixed fanouts and for a geometric distribution fanout model.

Further work remains to be done in this area, especially concerning the likely effect of bursty traffic sources upon the performance of multicast fabrics.

A.2 Modelling the Multicast Fabric

The description of the multicast fabric in Chapters 3 and 6 is first translated into a mathematical model. Assume there are N input and N output ports for the slotted ring and label them $1, 2, \ldots, N$. In modelling the behaviour of the ring as a cyclic server with N queues and limited service, we can say that the server jumps from node n to node $n + f$, where f is the fanout. However, if node n is empty the server moves to node $n + 1$. Because of the timing requirements, it takes the same time for the server to move from node n to node $n + 1$ whether or not it is copying a cell.

The key approximation used to simplify the analysis is that the probability that the server arrives at a node and finds that node busy is independent of the time since the server last visited that node. This implies that the server jumps according to a discrete time Markov chain. This approximation is not exact, since if a node has been jumped over for a long time it is very likely to be full and so independence is lost and the jumps can become correlated. Nonetheless, it is a promising idea, as it can be shown in the heavy traffic limit that nearly every node will be busy and so all correlations must disappear. For simplicity alone, it will be assumed in this analysis, as in the simulations, that all cells arrive at a node just before the arrival of the head of a slot at that node. This implies that the expected waiting time for cells when the load is very low is zero slot cycles, rather than 0.5 slot cycles if cells arrived in an uncorrelated fashion.

A.3 The Probability that the Server finds a Node busy

Focus attention on node n. Every time the server visits node n it will offer to copy cells and so will jump away according to whether work was found or not. It will return at a later time dictated by the Markov chain. We need to determine the probability that the server finds work to be done at node n, S_n. However the length of time the server is away from node n depends on S_n and of course S_n depends on how long the server is away. This presents us with a fixed point problem that can be solved analytically.

Let us assume that the switch is symmetric and can thus drop the subscript n from S. To determine S, treat each node like an $M|G|1$ queue with server vacations. (This is again an approximation, since the service times can be correlated. However, it is an exact consequence of the independence assumption.) Imbed a discrete time Markov chain at the time points when the server arrives at node n, whether or not node n is full. For more details on this technique see pages 234–238 of Cohen [Cohen69] or pages 210–216 and 222–223 of Cooper [Cooper81]. Note, however that they imbed the chain at the epochs when customers depart the system, which is subtly different.

Let B represent the time taken for the server to return to node n, given that it copied a cell at node n, and let B^0 represent the time taken for the server to return to node n, given that it found no cells to copy at node n.

Theorem A.3.1 *If* $\beta = \mathbb{E}[B]$ *and* $\beta^0 = \mathbb{E}[B^0]$ *then,*

$$S = \frac{\lambda\beta^0}{1 + \lambda\beta^0 - \lambda\beta}.$$

Proof A.3.1 See Theorem 3.1 [Bean93]. ■

A technique for evaluating β and β^0 can be developed by considering the mean return times of the Markov chain. Recall this Markov chain is well defined if S is assumed to be known and the values of β and β^0 will depend on S.

Therefore, a formula defining S in terms of β and β^0 and an equation defining β and β^0 in terms of S are known. This is a classical fixed point problem. However, in this case the solution can be found analytically.

Theorem A.3.2 *If* $\phi = \mathbb{E}[fanout]$ *is the mean of the fanout distribution, then the* *probability that the server finds a node busy is given by,*

$$S = \frac{\lambda}{1 - \lambda(\phi - 1)}.$$

Proof A.3.2 See Theorem 3.3 [Bean93] ■

A.4 Mean Delay of each Cell

Now that S has been found, we wish to determine the mean delay experienced by each cell, from the moment the header starts to arrive to the moment the server starts to copy the cell. Again focus attention on node n. Imbed a Markov chain at the departure point of customers. This process is made simpler if we consider customer departures to occur when the server returns to the same node.

Theorem A.4.1 *The mean delay experienced by each customer is given by,*

$$\mathbb{E}[W] = \frac{\hat{r}''(1)}{2\hat{r}'(1)\lambda} + \frac{\lambda F''(1)}{2(1 - \lambda\beta)},$$

where $F(z)$ and $\hat{r}(z)$ represent the probability probability generating functions of the number of customers arriving in a service time and a vacation, respectively.

Proof A.4.1 This is precisely the discrete time analogue of Exercise 12, pages 222–223 of Cooper [Cooper81] with $\hat{r}(z)$ replacing $f(z)$. ■

It is clear that $F''(1)$, $\hat{r}'(1)$ and $\hat{r}''(1)$ are easily determined by considering the return times of the earlier Markov chain and some simple probability generating function results [Kleinrock75, p.385].

A.5 Simulation and Analysis Differences

As can be seen from Figures 4.1 and 4.2, the approximate analysis corresponds closely to the simulation results at low loads, since the correlations involving times between successive visits to a state become insignificant in this region. As the load increases, these correlations increasingly begin to affect the action of the multicast fabric, introducing larger differences between the analysis and simulation results at higher loads. Heavy traffic simulation results have been shown to tend to the limits predicted by the analysis for loads greater than 0.99 [Bean93].

B Tabulated Results

A selection of the simulation results for the various traffic models described in Chapters 4 and 5 are included here to enable more accurate comparison than is permitted by the graphs presented in those chapters. Confidence intervals have been omitted to save space.

B.1 Results for Fixed Fanout

Applied Load	Fanout						
	3	5	7	9	11	13	15
0.1	0.0764	0.0924	0.0993	0.1041	0.1063	0.1059	0.1053
0.2	0.1749	0.2169	0.2357	0.2484	0.2500	0.2444	0.2395
0.3	0.3035	0.3832	0.4207	0.4484	0.4558	0.4385	0.4199
0.4	0.4804	0.6177	0.6827	0.7444	0.7643	0.7201	0.6632
0.5	0.7395	0.9716	1.0794	1.2101	1.2600	1.1663	1.0147
0.6	1.1432	1.5445	1.7145	1.9668	2.1434	1.9563	1.5735
0.7	1.8345	2.5470	2.9359	3.3532	3.8895	3.7502	2.5961
0.8	3.2835	4.7269	5.7122	6.3475	8.1068	9.0246	5.0302
0.9	7.8798	12.2902	17.8521	15.3402	20.5014	30.7748	18.7735

Table B.1: Mean delay for fixed fanout at various applied loads for a multicast fabric of size 16

Applied Load	Size			
	4	8	16	32
0.1	0.0852	0.0866	0.0863	0.0866
0.3	0.3377	0.3598	0.3536	0.3489
0.4	0.5268	0.5727	0.5696	0.5638
0.5	0.7786	0.8827	0.8846	0.8794
0.6	1.1597	1.3669	1.3781	1.3824
0.7	1.7884	2.1700	2.2424	2.2658
0.8	3.0819	3.8749	4.1949	4.1951
0.9	6.6574	8.9111	10.0693	10.3566

Table B.2: Mean delay for fixed fanout 4 for various sizes of multicast fabric

B.2 Results for Uniformly Distributed Fanout

Applied Load	Size			
	4	8	16	32
0.1	1.0000	1.0000	1.0000	1.0000
0.3	3.0000	3.0000	3.0000	3.0000
0.4	4.0000	5.0000	5.0000	4.9000
0.5	5.0000	6.6000	6.6000	6.3000
0.6	7.0000	9.1000	9.0000	9.0000
0.7	10.0000	13.7000	13.9000	14.0000
0.8	16.5000	23.5000	24.4000	23.6000
0.9	33.1000	52.5000	53.6000	52.9000

Table B.3: 99th percentile delay for fixed fanout 4 for various sizes of multicast fabric

Applied Load	Delay	
	Mean	99th %tile
0.1	0.1002	2.0000
0.3	0.4246	4.0000
0.4	0.6924	6.0000
0.45	0.8721	7.0000
0.5	1.1005	8.6000
0.55	1.3723	10.2000
0.6	1.7440	13.0000
0.7	2.8355	20.4000
0.8	5.2971	35.2000
0.9	12.6722	83.5000

Table B.4: Mean and 99th percentile delays for uniformly distributed random fanout for 16 node multicast fabric

B.3 Results for Geometrically Distributed Fanout

The results shown in Table B.5 are for a 64×64 multicast fabric with a geometrically distributed fanout of mean 8.

Applied	Percentage of Multicast Traffic									
Load	100	90	80	70	60	50	40	30	20	10
0.1	0.14	0.08	0.04	0.08	0.08	0.13	0.10	0.11	0.09	0.04
0.2	0.19	0.25	0.26	0.17	0.17	0.19	0.14	0.24	0.16	0.04
0.3	0.45	0.39	0.43	0.33	0.40	0.34	0.39	0.23	0.27	0.18
0.4	0.70	0.69	0.66	0.61	0.59	0.50	0.61	0.50	0.45	0.30
0.5	1.04	1.03	0.98	0.96	0.91	0.93	0.80	0.73	0.57	0.44
0.6	1.60	1.60	1.62	1.56	1.50	1.44	1.31	1.14	0.99	0.69
0.7	2.74	2.72	2.66	2.59	2.43	2.38	2.16	2.00	1.57	1.05
0.8	5.03	5.04	4.87	4.53	4.64	4.08	3.80	3.40	2.86	1.93
0.9	12.61	12.26	10.89	11.48	11.23	9.59	9.53	7.82	6.26	4.12

Table B.5: Mean delay for various percentages of multicast traffic with a geometrically distributed fanout

B.4 Results for Bulk Arrivals

These results are for a multicast fabric with 16 nodes using a uniformly distributed fanout.

Applied Load	Bulk Size (Cells)					
	1	2	4	8	16	32
0.1	0.10	0.17	0.30	0.56	1.09	2.11
0.3	0.42	0.68	1.24	2.37	4.53	9.13
0.4	0.69	1.12	1.97	3.85	7.56	14.44
0.45	0.85	1.43	2.51	4.64	9.31	17.90
0.5	1.07	1.77	3.20	5.88	11.35	22.56
0.55	1.39	2.20	3.88	7.50	14.73	28.77
0.6	1.73	2.75	4.99	9.25	17.62	37.74
0.7	2.80	4.52	8.02	14.80	27.16	54.36
0.8	5.32	7.62	13.25	24.98	48.72	92.25
0.9	13.92	15.21	24.64	42.28	88.03	153.32

Table B.6: Mean delay for various bulk sizes with a uniformly distributed fanout (16×16 fabric)

Applied Load	Bulk Size (Cells)					
	1	2	4	8	16	32
0.1	1.8000	2.0000	4.0000	7.7000	14.5000	27.6000
0.3	4.0000	6.0000	11.1000	20.6000	40.0000	77.8000
0.4	5.7000	8.9000	16.0000	30.7000	59.6000	114.1000
0.45	6.8000	10.9000	19.3000	36.0000	70.8000	137.3000
0.5	7.9000	12.9000	23.7000	43.9000	83.3000	166.0000
0.55	10.0000	15.8000	28.1000	54.7000	106.4000	211.0000
0.6	12.0000	18.9000	34.6000	65.2000	122.0000	270.4000
0.7	18.4000	29.4000	53.6000	98.6000	177.7000	366.9000
0.8	33.3000	47.5000	81.7000	155.6000	304.0000	572.3000
0.9	81.3000	90.7000	147.9000	249.6000	532.6000	879.4000

Table B.7: 99th percentile delay for various bulk sizes with uniformly distributed fanout (16×16 fabric)

B.5 Results for Asymmetric Loading

For a uniformly distributed random fanout and using the biased traffic distribution described in Chapter 5, the bias parameter α must be less than $(1-L)/L$ for stability, where L is the applied load.

Applied	Bias parameter α									
Load	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.1013	0.0994	0.0977	0.0917	0.0848	0.0767	0.0662	0.0531	0.0373	0.0193
0.3	0.4184	0.4206	0.4130	0.3980	0.3722	0.3474	0.3023	0.2472	0.1828	0.1010
0.4	0.6981	0.6726	0.6730	0.6631	0.6364	0.6066	0.5362	0.4780	0.3644	0.2121
0.45	0.8732	0.8781	0.8536	0.8650	0.8378	0.8028	0.7340	0.6739	0.5654	0.3625
0.5	1.1040	1.0851	1.0740	1.0929	1.1074	1.1016	1.0826	1.0610	1.0346	0.9588
0.55	1.3553	1.3311	1.3727	1.4572	1.5235	1.5907	1.8031	2.2168	7.0318	–
0.6	1.7203	1.7560	1.8215	1.9850	2.1522	2.7613	4.3904	–	–	–
0.7	2.8791	2.9720	3.3333	4.3927	11.1241	–	–	–	–	–
0.8	4.9385	6.1695	10.5859	–	–	–	–	–	–	–
0.9	13.9745	–	–	–	–	–	–	–	–	–

Table B.8: Total mean delay for all the ports in a uniformly distributed random fanout, fabric size 16 and asymmetric loading

Applied	Bias parameter α									
Load	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.1025	0.0922	0.0830	0.0721	0.0621	0.0528	0.0434	0.0326	0.0215	0.0107
0.3	0.4218	0.3932	0.3638	0.3328	0.2916	0.2566	0.2137	0.1674	0.1186	0.0624
0.4	0.6990	0.6450	0.6130	0.5791	0.5321	0.4784	0.4084	0.3502	0.2598	0.1456
0.45	0.8707	0.8556	0.7976	0.7696	0.7195	0.6631	0.5914	0.5277	0.4318	0.2753
0.5	1.1038	1.0739	1.0228	1.0094	1.0019	0.9685	0.9319	0.9119	0.8931	0.8627
0.55	1.3358	1.3336	1.3598	1.4084	1.4752	1.5341	1.7481	2.2466	8.0477	–
0.6	1.7000	1.7972	1.8805	2.0759	2.2937	3.0767	5.3920	–	–	–
0.7	2.8575	3.3845	4.0929	6.0814	19.7934	–	–	–	–	–
0.8	4.9190	8.6585	21.9159	–	–	–	–	–	–	–
0.9	13.6784	–	–	–	–	–	–	–	–	–

Table B.9: Mean delay through the heavily loaded port for uniformly distributed random fanout, fabric size 16 and asymmetric loading

Applied Load	Bias parameter α									
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.1013	0.1006	0.1027	0.1017	0.1027	0.1037	0.1042	0.1055	0.1062	0.1033
0.3	0.4179	0.4254	0.4292	0.4315	0.4339	0.4488	0.4487	0.4501	0.4580	0.4742
0.4	0.6980	0.6778	0.6925	0.7061	0.7164	0.7503	0.7466	0.8023	0.8139	0.8502
0.45	0.8735	0.8822	0.8718	0.9138	0.9287	0.9593	0.9690	1.0444	1.1393	1.2011
0.5	1.1043	1.0871	1.0906	1.1360	1.1877	1.2504	1.3311	1.4379	1.6402	1.8812
0.55	1.3567	1.3307	1.3769	1.4820	1.5605	1.6539	1.8945	2.1410	2.6676	2.8570
0.6	1.7217	1.7487	1.8023	1.9382	2.0440	2.4089	2.7399	3.1905	2.8964	2.8624
0.7	2.8804	2.8977	3.0840	3.5181	4.4678	4.4613	3.7778	3.2976	2.9855	2.7784
0.8	4.9396	5.7187	6.8594	7.1044	5.7843	4.8556	3.9677	3.6006	3.1885	2.7487
0.9	13.9944	16.2121	12.1944	8.8090	7.1036	5.6083	4.2474	3.6583	3.2187	2.9188

Table B.10: Mean delay of the lightly loaded ports for uniformly distributed random fanout, fabric size 16 and asymmetric loading

Applied Load	Bias parameter α					
	0.0	0.2	0.4	0.6	0.8	1.0
0.1	0.0999	0.1001	0.0990	0.0998	0.1006	0.0996
0.3	0.2972	0.3000	0.2975	0.2994	0.2984	0.2993
0.4	0.4012	0.3970	0.3959	0.3947	0.3957	0.3995
0.45	0.4504	0.4472	0.4481	0.4410	0.4501	0.4495
0.5	0.5029	0.4943	0.4981	0.4959	0.5003	0.5020
0.55	0.5467	0.5437	0.5497	0.5497	0.5490	0.4986
0.6	0.5966	0.5981	0.5959	0.5958	0.5595	0.4991
0.7	0.7001	0.6968	0.6998	0.6391	0.5732	0.5035
0.8	0.7880	0.8028	0.7382	0.6573	0.5862	0.4985
0.9	0.9047	0.8568	0.7690	0.6749	0.5914	0.5027

Table B.11: Throughput for uniformly distributed random fanout, fabric size 16 and asymmetric loading

Applied Load	Bias parameter α									
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.0049	0.0233	0.0485	0.0713	0.0972	0.1200	0.1419	0.1658	0.1925	0.2072
0.3	0.0104	0.1294	0.2642	0.3834	0.5184	0.6586	0.7587	0.8944	1.0250	1.1801
0.4	0.0161	0.2509	0.4979	0.7881	1.0365	1.3451	1.5851	1.9186	2.2303	2.6383
0.45	0.0219	0.3654	0.6836	1.1053	1.5021	1.9603	2.3485	2.8975	3.6978	4.4272
0.5	0.0363	0.5031	0.9737	1.5750	2.1745	2.8530	3.6887	4.7926	6.4516	9.5123
0.55	0.0327	0.7093	1.4108	2.2732	3.2653	4.4287	6.4226	9.1542	14.4605	–
0.6	0.0538	1.0211	2.0988	3.4241	5.0498	8.1918	12.0004	–	–	–
0.7	0.0835	2.2235	4.9398	9.4056	18.9634	–	–	–	–	–
0.8	0.3084	6.6194	19.6226	–	–	–	–	–	–	–
0.9	1.4729	–	–	–	–	–	–	–	–	–

Table B.12: Maximum difference in mean delays of all ports for uniformly distributed random fanout, fabric size 16 and asymmetric loading

C Bibliography

[Adaptive92] *ATMX: Asynchronous Transfer Mode Switch for LAN Environments.* Adaptive Corporation, Redwood City, California, USA, July 1992. (p 43)

[Ahmadi89] H Ahmadi and W E Denzel. *A Survey of Modern High-Performance Switching Techniques.* IEEE J. Selected Areas in Communications, 7(7):1091–1103, September 1989. (p 38)

[Aho83] A V Aho, J E Hopcroft and J D Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983. (pp 28, 129, 145)

[Bae91] J J Bae and T Suda. *Survey of Traffic Control Schemes and Protocols in ATM Networks.* Proceedings of IEEE, 79(2):170–189, February 1991. (p 62)

[Ballardie92] A Ballardie. *Multicasting: An Overview.* Research Note RN/92/85, University College, London, Computer Science Department, August 1992. (p 26)

[Batcher68] K E Batcher. *Sorting Networks and their Applications.* In Proceedings of Spring Joint Computer Conference, pages 307–314, 1968. (p 98)

[Bean92] N G Bean. *Mathematical Modelling of Packet Switched Communication Networks.* J T Knight Prize Essay, University of Cambridge Statistical Laboratory, January 1992. pages 87–104. (pp 63, 147)

[Bean93] N G Bean and M Doar. *Analysis of a Cyclic Server with Irregular Server Movements: A Model of an ATM Multicast Fabric.* In preparation, 1993. (pp 63, 69, 147, 148, 149)

[Bianchini92] R P Bianchini and H S Kim. *Design of a non-blocking Shared-Memory Copy Network for ATM.* In Proceedings of IEEE Infocom, Florence, pages 876–885, 1992. (p 52)

[Birman87] K P Birman and T A Joseph. *Reliable Communication in the Presence of Failures.* ACM Transactions on Computer Systems, 5(1):47–76, February 1987. (pp 10, 16)

[Birrell82] A Birrell, R Levin, R Needham and M Schroeder. *Grapevine: an Exercise in Distributed Computing.* Communication of the ACM, 25(4), April 1982. (p 8)

[Bubenik89] R G Bubenik and J S Turner. *Performance of a Broadcast Packet Switch.* IEEE Trans. Commun., 37(1):60–69, January 1989. (pp 47, 49, 53, 65)

[Byrd87] G T Byrd, R Nakano and B A Delagi. *A Point-to-Point Multicast Communications Protocol.* Technical Report STAN-CS-87-1146, Department of Computer Science, Stanford University, California, USA, January 1987. Also numbered KSL-87-02. (p 12)

[Campbell92] A Campbell, G Coulson, F Garcia, D Hutchison and H Leopold. *Integrated Quality of Service for Multimedia Communications.* In Proceedings of Multi-Service Networks, Cosener's House, Abingdon, Oxfordshire, England, July 1992. Also submitted to IEEE Infocom 1993, San Francisco, USA. (p 18)

[Casner92] S Casner and S Deering. *First IETF Internet Audiocast.* ACM Computer Communication Review, pages 92–97, July 1992. (p 27)

[CCITT E.164] *Telephone Network and ISDN – Operation, numbering, routing and mobile service.* Geneva, 1989. Blue Book Vol. II, Fascicle II.2, Recommendations E.100 – E.300. (p 30)

[CCITT G.707/8/9] *Revised Draft Recommendation G.707, G.708, G.709.* Geneva, June 1990. (p 2)

[CCITT I.113] *CCITT Draft Recommendation I.113.* Geneva, May 1990. (p 2)

[CCITT I.150] *CCITT Draft Recommendation I.150, B-ISDN ATM Functional Characteristics.* Geneva, May 1990. (p 6)

[CCITT I.432] *CCITT Draft Recommendation I.432.* Geneva, May 1990. Section 4.3 contains details of the CRC. (pp 22, 88)

[CCITT X.400] *CCITT Recommendations X.400 – X.430, Data Communications Networks Message Handling Systems.* Geneva, October 1984. VIIIth Plenary Assembly, Torremolinos, Spain. Superseded in 1988 and 1992. (p 15)

[Chaffee92] J W Chaffee. *Observability, Ensemble Averaging and GPS Time.* IEEE Transactions on Aerospace and Electronic Systems, 28(1):224–240, 1992. (p 17)

[Chen91a] X Chen. *A Survey of Multistage Interconnection Networks in Fast Packet Switches.* International Journal of Digital and Analog Communication Systems, 4:33–59, 1991. (p 38)

[Chen91b] X Chen, J F Hayes and M K Mehmet Ali. *Performance Analysis of Cyclic Priority Input Access Method for a Multicast Switch.* In Proceedings of IEEE Infocom, Miami, pages 1189–1195, 1991. (p 74)

[Chesson92] G Chesson. *XTP Protocol Definition.* Technical Report PEI 92-10, Protocol Engines Incorporated, Mountain View, California, January 1992. Revision 3.6. (pp 27, 28)

[Choi92] J S Choi, C K Un and B C Shin. *Design of a Cascade Ring Multicast Switch.* Electronics Letters, 28(14):1330–1332, July 1992. (p 54)

[Chow91] C H Chow. *On Multicast Path Finding Algorithms.* In Proceedings of IEEE Infocom, Miami, pages 1274–1283, 1991. (p 123)

[Cidon88] I Cidon and I Gopal. *PARIS: An approach to integrated high speed private networks.* Int. J. Digital and Analog Cabled Systems, 1(1):77–85, 1988. Also in *IEEE International Conference in Communications (ICC'87)*, Seattle, WA, USA, June 1987, pages 764–773. (p 43)

[Cohen69] J W Cohen. *The Single Server Queue.* North-Holland, Amsterdam, 1969. (p 148)

[Comer91] D Comer. *Internetworking with TCP/IP.* Prentice-Hall, Englewood Cliffs, NJ, second edition, 1991. (p 25)

[Cooper81] R B Cooper. *Introduction to Queueing Theory.* Elsevier North-Holland Inc., London, second edition, 1981. (pp 148, 149)

[Cox91] J R Cox and J S Turner. *Project Zeus : Design of a Broadband Network and its Application on a University Campus.* Technical Report WUCS91-45, Department of Computer Science, Washington University in St. Louis, USA, July 1991. (pp 7, 18, 47, 58)

[Crosby92] S A Crosby. *MSNL Connection Management.* Internal report, Computer Laboratory, University of Cambridge, December 1992. (p 118)

[Crowcroft92] J Crowcroft, A Ballardie and P Tsuchiya. *Core Based Trees (CBT).* Internet Draft, University College, London, Computer Science Department, October 1992. Working draft, expires March 22nd 1993. (pp 24, 30, 31)

[Dalal78] Y K Dalal and R M Metcalfe. *Reverse Path Forwarding of Broadcast Packets.* Communications of the ACM, 21(2):1040–1047, December 1978. (p 26)

[de Prycker91] M de Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN.* Ellis Horwood, Chichester, England, 1991. (pp 6, 36, 42)

[de Vries90] R J F de Vries. *ATM Multicast Connections using the Gauss Switch.* In Proceedings of GLOBECOM, 1990. (p 6)

[Deering85] S Deering and D R Cheriton. *Host Groups: A Multicast Extension to the Internet Protocol.* Internet Network Working Group, RFC 966, December 1985. (pp 24, 25)

[Deering89] S Deering. *Host Extensions for IP Multicasting.* Internet Network Working Group, RFC 1112, August 1989. Obsoletes RFC 988 and RFC 1054. (pp 24, 25)

[Deering91] S E Deering. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Department of Computer Science, Stanford University, California, USA, December 1991. (pp 8, 25, 26)

[deHart92] J deHart, M Gaddis and R Bubenik. *Connection Management Access Protocol (CMAP) Specification.* Technical Report WUCS-92-01, Applied Research Laboratory, Department of Computer Science, Washington University in St. Louis, May 1992. Version 2.1.1. (pp 29, 116)

[Dixon92] M J Dixon. *System Support for Multi-Service Traffic.* PhD thesis, University of Cambridge Computer Laboratory, January 1992. Also published as Technical Report No. 245. (pp 112, 115)

[DuBose92] K DuBose and H S Kim. *An Effective Bit Rate/Table Lookup Based Admission Control Strategy for the ATM B-ISDN.* In Proceedings of 17th Conference on Local Computer Networks, Minneapolis, Minnesota, USA, September 1992. (p 10)

[Ethernet80] *The Ethernet Local Network: Three Reports.* Technical Report CSL-80-2, Xerox Palo Alto Research Center, Palo Alto, CA, 1980. (p 24)

[Forcina89] A Forcina, T Di Stefano and E Taormina. *A Multicast Broadband Switching Module in a Hybrid ATM Environment.* In Proceedings of International Conference on Communications, Boston, ICC '89, June 1989. (p 41)

[Frank85] A J Frank, L D Wittie and A J Bernstein. *Multicast Communication on Network Computers.* IEEE Software, 2(3):49–61, 1985. Also included in *Multicast Communication in Distributed Systems*, M Ahamad (editor), IEEE Computer Society Press, 1990. (p 10)

[Furber87] S Furber and R W Wilson. *The Acorn RISC Machine – An Architectural View.* IEE Journal of Electronics and Power, 33(6):402–405, June 1987. (p 82)

[Gard90] I Gard and J Rooth. *An ATM switch implementation – technique and technology.* In Proceedings of International Switching Symposium, Stockholm, Sweden, May 1990. (p 46)

[Giacopelli90] J N Giacopelli, W D Sincoskie and M Littlewood. *Sunshine: a High Performance Self Routing Broadband Packet Switch Architecture.* In Proceedings of International Switching Symposium, May 1990. (p 49)

[Gifford79] D K Gifford. *Weighted Voting for Replicated Data.* In Proceedings of the 7th ACM Symposium on Operating System Principles, December 1979. (p 8)

[Gilbert68] E N Gilbert and H O Pollak. *Steiner Minimal Trees.* SIAM Journal of Applied Mathematics, 16(1):1–29, 1968. (p 124)

[Gottlieb83] A Gottlieb, R Grishman, C P Kruskal, K P McAuliffe, L Rudolph and M Snir. *The NYU Ultracomputer – Designing and MIMD Shared Memory Parallel Computer.* IEEE Transactions on Computers, 32(2):175–189, February 1983. (p 54)

[Greaves92] D J Greaves and D McAuley. *Private ATM Networks.* In Proceedings of 3rd International IFIP WG6.1/6.4 Workshop on Protocols for High-Speed Networks, Stockholm, May 1992. Also available as Technical Report No. 258, University of Cambridge Computer Laboratory. (p 7)

[Händel91] R Händel and M N Huber. *Integrated Broadband Networks: An Introduction to ATM-Based Networks.* Electronic Systems Engineering Series. Addison Wesley, 1991. (p 6)

[Heinanen92] J Heinanen. *Multiprotocol Interconnect over ATM Adaptation Layer 5.* Internet Draft, Telecom Finland, December 1992. Working draft, expires June 17th 1993. Note that all Internet working drafts are available by anonymous `ftp` from nic.nordu.net, amongst other sites. (p 26)

[Henrion90] M Henrion, K Schrodi, D Boettle and M De Somer. *Switching Network Architecture for ATM based Broadband Communication.* In Proceedings of International Switching Symposium, Stockholm, April 1990. (pp 12, 42)

[Hopper88] A Hopper and R M Needham. *The Cambridge Fast Ring Networking System.* IEEE Transactions on Communications, 37(10), October 1988. (p 24)

[Hopper90] A Hopper. *Pandora – An Experimental Distributed System for Multimedia Applications.* ACM Operating Systems Review, April 1990. (p 83)

[Huang84] A Huang and S Knauer. *Starlite: A wideband digital switch.* In Proceedings of IEEE GLOBECOM, pages 121–125, 1984. (p 47)

[Hui87] J Y Hui and E Arthurs. *A Broadband Packet Switch for Integrated Transport.* IEEE J. Selected Areas in Communication, 5(8):1264–1273, October 1987. (pp 35, 53)

[Hui90a] J Y Hui. *Switching and Traffic Theory for Integrated Broadband Networks.* Kluwer Academic Publishers, Norwell MA, 1990. (pp 38, 57, 147)

[Hui90b] J Y Hui and T Renner. *Queueing Analysis for Multicast Packet Switches*, 1990. Unpublished document. (p 147)

[Jacob90] A R Jacob. *A Survey of Fast Packet Switches.* Computer Communication Review, pages 54–65, January 1990. (p 38)

[Jiang91] X F Jiang. *Multipoint Digital Video Communications.* PhD thesis, University of Cambridge Computer Laboratory, 1991. Also published as Technical Report No. 251. (p 125)

[Kalmanek90] C R Kalmanek, H Kanakia and S Keshav. *Rate Controlled Servers for Very High Speed Networks.* In Proceedings of IEEE GLOBECOM, pages 12–20, 1990. (p 83)

[Karp72] R M Karp. *Reducibility among combinatorial problems.* In R E Miller and J W Thatcher, (Eds), *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972. (pp 20, 124, 137)

[Kim92] H S Kim and A Leon-Garcia. *Non-blocking Property of Reverse Banyan Network.* IEEE Transactions on Communications, March 1992. (p 52)

[Kleinrock75] L Kleinrock. *Queueing Systems Volume I: Theory.* Wiley, New York, 1975. (p 149)

[Kou81] L Kou, G Markowsky and L Berman. *A Fast Algorithm for Steiner Trees.* Acta Informatica, 15:141–145, 1981. (pp 31, 124)

[Kuwahara89] H Kuwahara, N Endo, M Ogino and T Kozaki. *A shared buffer memory switch for an ATM exchange.* In Proceedings of IEEE Int. Conf. Commun., Boston, June 1989. ICC '89. (pp 41, 42)

[Lea88] C T Lea. *A New Broadcast Switching Network.* IEEE Trans. Commun., 36(10):1128–1137, October 1988. (pp 34, 51)

[Lee88a] T T Lee. *Non-blocking Copy Networks for Multicast Packet Switching.* IEEE J. Selected Areas in Communication, 6(9):1455–1467, December 1988. (pp 49, 53, 108, 147)

[Lee88b] T T Lee, R Boorstyn and E Arthurs. *The Architecture of a Multicast Broadband Packet Switch.* In Proceedings of IEEE Infocom, 1988. New Orleans, LA. (p 49)

[Leopold92] H Leopold, A Campbell, D Hutchison and N Singer. *Towards an Integrated Quality of Service Architecture (QOS-A) for Distributed Multimedia Communications.* In Proceedings of Multi-Service Networks, Cosener's House, Abingdon, Oxfordshire, England, July 1992. Also submitted to IFIP High Performance Networks 1992, Liege, Belgium. (p 18)

[Leslie91] I M Leslie and D McAuley. *Fairisle: An ATM Network for the Local Area.* In Proceedings of SIGCOMM, Zurich, September 1991. (pp 18, 59, 81)

[Li87] T Li. *Advances in Lightwave Systems Research.* AT & T Technical Journal, 66(1):5–18, 1987. This issue was devoted to lightwave technology. (p 36)

[Mackay91] W R Mackay. *A Charging Strategy for ATM Networks.* In Proceedings of ATM Network Planning and Evolution, Queen Mary Westfield, University of London, April 1991. (p 21)

[Mapp92] G Mapp. *An Object-Oriented Approach to Virtual Memory Management.* PhD thesis, University of Cambridge Computer Laboratory, January 1992. Also published as Technical Report No. 242. (p 115)

[McAuley90] D R McAuley. *Protocol Design for High Speed Networks.* PhD thesis, University of Cambridge Computer Laboratory, January 1990. Also published as Technical Report No. 186. (p 112)

[Mills91a] C Mills, D Hirsh and G Ruth. *Internet Accounting: Background.* Internet Network Working Group, RFC 1272, November 1991. (p 21)

[Mills91b] D L Mills. *Internet Time Synchronisation: the Network Time Protocol.* Internet Network Working Group, RFC 1129, October 1991. (p 17)

[Milway92] D R Milway. Private communication, December 1992. (p 43)

[Moy91] J Moy. *OPSF Version 2.* Internet Network Working Group, RFC 1247, July 1991. Obsoletes RFC 1131. (p 28)

[Moy92] J Moy. *Multicast Extension to OSPF.* Internet Draft, Proteon, Inc., September 1992. Working draft, expires February 1993. (pp 28, 29, 124)

[Mullender92] S J Mullender, I M Leslie and D McAuley. *Operating Support for Distributed Multimedia Systems.* Project Pegasus 92-1, University of Cambridge Computer Laboratory, September 1992. (p 17)

[Newman88] P Newman. *Fast Packet Switching for Integrated Services*. PhD thesis, University of Cambridge Computer Laboratory, 1988. (pp 10, 31, 35, 37, 38)

[Newman89] P Newman. *Copy Fabric for a Multicast Fast Packet Switch*. UK Patent Application 8927625.7, December 1989. US Application: 07/622,757, 5th December 1990. Now held by Adaptive Corporation, Redwood City, California, USA. (pp 13, 54)

[Newman90] P Newman and M Doar. *A Slotted Ring Copy Fabric for a Multicast Fast Packet Switch*. In Proceedings of International Switching Symposium, ISS '90, May 1990. (pp 54, 65)

[Newman92] P Newman. *ATM Technology for Corporate Networks*. IEEE Communications Magazine, 30(4):90–101, April 1992. (p 38)

[Nojima87] S Nojima, E Tsutsui, H Fukada and M Hashimoto. *Integrated Services Packet Network using Bus Matrix Switch*. IEEE J. Selected Areas in Communications, 5(8):1284–1292, October 1987. (p 43)

[OSI83] ISO OSI. *ISO International Standard 7498, Information Processing Systems – Open Systems Interconnection – Basic Reference Model*. International Organisation for Standardisation, Geneva, October 1983. (pp 2, 15)

[Page92] B Page. *Discrete Event Simulation in Modula 2*. In A Sydow, (Ed), *Computational Systems Analysis – Topics and Trends*. Elsevier, 1992. Also in Sys. An. Model. Sim, 1990 (5), page 339. (p 61)

[Paliwoda88] K Paliwoda. *Transactions involving Multicast*. Research Note RN/88/15, University College London, September 1988. (p 24)

[Pippenger74] N Pippenger. *On the complexity of strictly non-blocking concentration networks*. IEEE Trans. Commun., November 1974. (p 57)

[Pratt92] I A Pratt. *An ATM Camera for the Desk Area Network*. Part II dissertation, University of Cambridge Computer Laboratory, May 1992. (p 104)

[Rahnema90] M Rahnema. *The Fast Packet Switch: A High-Performance Efficient Architecture with Multicast Capability.* IEEE Trans. Comm., 38(4):539–545, April 1990. (p 46)

[Rayward-Smith86] V J Rayward-Smith and A Clare. *On finding Steiner Vertices.* Networks, 16:283–294, 1986. (pp 31, 124)

[Richards85] G W Richards and F K Hwang. *A Two-stage Rearrangeable Broadcast Packet Switching Network.* IEEE Trans. Commun., 33(10):1025–1035, October 1985. (p 38)

[Sakurai90] Y Sakurai, N Ido and S Gohara. *Large scale ATM multi-stage switching network with shared buffer memory switches.* In Proceedings of International Switching Symposium, Stockholm, Sweden, May 1990. (p 41)

[Shrimpton90] D H Shrimpton and C S Cooper. *Multicast communication on the UNISON network.* Computer Communications, 30(8):460–468, 1990. (p 24)

[Sreenan92] C Sreenan. *Synchronisation Services for Digital Continuous Media.* PhD thesis, University of Cambridge Computer Laboratory, October 1992. (p 19)

[Strebenz90] J Strebenz. *Axon: Host-Network Interface Design.* Technical Report WUSC-90-7, Washington University in St. Louis, March 1990. (p 8)

[Suzuki89] H Suzuki et al. *Output-buffer switch architecture for Asynchronous Transfer Mode.* In Proceedings of IEEE International Conference on Communication, ICC '89, Boston, pages 99–103, June 1989. (p 43)

[Takagi90] H Takagi. *Queueing Analysis of Polling Models: An Update.* Technical Report RT 0021, IBM Research, Tokyo Research Laboratory, Japan, 1990. Also in *Stochastic Analysis of Computer and Communication Systems*, H Takagi (editor), Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1990. (p 75)

[TAXI89] *Data Sheets for Am7968/Am7969-175 TAXIchip Inegrated Circuits*, November 1989. Publication 12834, revision A. (pp 58, 82)

[Tennenhouse87] D L Tennenhouse, I M Leslie, R M Needham, J W Burren, C J Adams and C S Cooper. *Exploiting Wideband ISDN: The Unison Exchange.* In Proceedings of IEEE Infocom, pages 1018–1026, 1987. (p 24)

[Thomas84] A Thomas, J P Coudreuse and M Servel. *Asynchronous time-division techniques: An experimental packet network integrating video communications.* In Proceedings of International Switching Symposium, Florence, Italy, May 1984. (p 41)

[Tobagi90] F A Tobagi. *Fast Packet Switch Architectures for Broadband Integrated Data Service Networks.* Proceedings of IEEE, 78(1):113–167, January 1990. (pp 35, 38)

[Tode92] H Tode, Y Sakai, M Yakamoto, H Okada and Y Tezuka. *Multicast Routing Algorithm for Nodal Load Balancing.* In Proceedings of IEEE Infocom, Florence, pages 2086–2095, 1992. (p 123)

[Turner86a] J S Turner. *Design of a Broadcast Packet Network.* In Proceedings of IEEE Infocom, Miami, pages 667–675, 1986. (p 47)

[Turner86b] J S Turner. *Design of an Integrated Services Packet Network.* IEEE J. Selected Areas in Communication, 4(8):1373–1380, November 1986. (p 34)

[Turner88] J S Turner. *Design of a Broadcast Packet Switching Network.* IEEE Trans. Commun., 36(6):734–743, June 1988. (pp 31, 47, 58, 108)

[Turner91] J S Turner. *A Practical Version of Lee's Multicast Switch Architecture.* Technical Report WUCS-91-46, Washington University in St. Louis, August 1991. (p 51)

[Waitzman88] D Waitzman, C Partridge and S Deering. *Distance Vector Multicast Routing Protocol.* Internet Network Working Group, RFC 1075, November 1988. (pp 24, 26, 124)

[Wall80] D W Wall. *Mechanisms for Broadcast and Selective Broadcast.* PhD thesis, Stanford University, California, USA, June 1980. Also published as Computer Systems Laboratory Technical Report 190. (p 31)

[Wall90] D W Wall. *Selected Broadcast in Packet-Switched Networks.* In M Ahamad, (Ed), *Multicast Communication in Distributed Systems.* IEEE Computer Society Press, 1990. (p 124)

[Wang92a] Z Wang. *Routing and Congestion Control in Datagram Networks.* PhD thesis, University College, London, January 1992. (p 126)

[Wang92b] Z Wang and J Crowcroft. *Analysis of Shortest-Path Routing Algorithms in a Dynamic Network Environment.* ACM Computer Communication Review, 22(2):63–71, April 1992. (p 126)

[Waters92] A G Waters. *Multicast Provision for High Speed Networks.* In Proceedings of 4th IFIP High Performance Networking, Liege, Belgium, December 1992. (p 17)

[Waxman88] B M Waxman. *Routing of Multipoint Connections.* IEEE J. Selected Areas in Communications, 6(9), December 1988. (pp 124, 125, 126, 129)

[Waxman92] B M Waxman. Private communication, May 1992. (pp 125, 129, 130)

[Wilbur92] S Wilbur. *The UK National Research Network: From JANET to SuperJanet.* In Proceedings of IEEE Eurocom Conference on Computer and Telecommunication Networks, May 1992. (p 30)

[Wilson92] T D Wilson. *Increasing the Performance of Storage Services for Local Area Networks.* PhD thesis, University of Cambridge Computer Laboratory, 1992. (p 115)

[Winter87] P Winter. *Steiner Problem in Networks.* Networks, 17:129–167, 1987. (p 124)

[Xilinx89] *Xilinx: The Programmable Gate Array Data Book.* San Jose, California, 1989. (Includes the 3000 family). (p 81)

[Yang92] S-C Y Yang and J A Silvester. *A Reconfigurable ATM Switch Fabric for Fault Tolerance and Traffic Balancing.* IEEE J. Selected Areas of Communication, 9(8):1205–1217, October 1992. (p 35)

[Yum92] T S Yum and Y W Leung. *A TDM-based Multibus Packet Switch.* In Proceedings of Infocom, pages 2509–2515, Florence, Italy, April 1992. (pp 12, 43)

"I have only made this letter longer because I have not had time to make it shorter." — Blaise Pascal, *Lettres Provinciales*