Technical Report

UCAM-CL-TR-101
ISSN 1476-2986

Number 101

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Binary routing networks

David Russel Milway

December 1986

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

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

This technical report is based on a dissertation submitted December 1986 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Darwin College.

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

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

ISSN 1476-2986

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In all fields of computer science there have been attempts to push the available technology to gain higher performance while retaining the previous architecture. Paralleled with this has been research into new architectures which attempt to give the desired performance increases. The development of computer communications has been prominent in both of these approaches.

The transfer of data between computer systems has been a topic of research for many years and has developed into a number of distinct areas. These can be categorised into *Wide Area Networks*, *Local Area Networks* and *Interconnection Networks*. The best distinguishing feature of these networks is the area covered. In the middle comes the Local Area Network which covers communications between computers separated by as little as a few metres and at most a few kilometres. Above this range come the Wide Area Networks and below, the Interconnection Networks. Table 1.1 gives a summary of the major features of each of the network categories. There will always be networks that fall between two of the categories and these divisions are meant only as a guide to the different types.

Wide Area Networks, networks which cover areas from the size of towns to continents and the whole earth, are normally constructed from medium to high speed links that pass from one computer centre to another and have a topology that is determined by the physical location of the computers, the ease of establishing a physical link and the amount of data traffic between the two computers. The last two points are directly associated with the economics of the network, as long distance connections using high data rates are very costly. These networks operate by passing packetised data between the nodes on the network. Where no direct connection exists between two stations on the network, the packets are passed through one or more intermediate nodes. The network will attempt to route the packets along the most direct path and ensure that, where different routes are

Type	Size	Data Rate	Approx. Delay	Protocol
Wide Area Network	>10km	64kb to 1Mb	100mS	Complex
Local Area Network	<10km	>1Mb	10μS	Simple
Interconnection Network	<4m	$\gg$1Mb	1μS	Very Simple

Table 1.1: Comparison of Generic Network Types

used, the packets arrive in the order in which they were sent. The controlling software in a sophisticated network would take into account the current load on the links around the network and route packets in such a way that the load is evenly distributed, avoiding heavily loaded areas and faulty links.

A Local Area Network is the term given to communications systems that connect together computer equipment which is typically located within the same room, building or site. Local Area Networks are usually constructed with high capacity channels with low delay properties which allow direct communications between the sending station and the receiving station. This topology provides an economical solution, as each station on the network needs only one network interface, and over the short distance of the network the technology required to drive the medium at high speed is not expensive. For most of the networks in this category the term *network* should not be applied as they are actually a media access mechanism which divides up the available bandwidth between all active stations. The service provided normally consists of broadcasting the packets to all stations which filter out packets not addressed to them. This removes the need to route packets and reduces the problem of out-of-sequence packets. Interconnection of these forms of network is made via gateways which introduce some of the properties found in the Wide Area Networks. Within the sub-networks, the Local Area mechanism still applies, but as soon as transmissions cross the boundary, protocol conversions may need to be applied.

Interconnection Networks are reserved for the communications between processors or processors and memory in a multiprocessor system. These systems are characterized by very high speeds and parallel data paths. They have come about because in multiprocessor systems, where there are large numbers of processors and memories, it is not possible to provide direct connections between all units. A single shared medium cannot supply the required bandwidth. The network is normally made up of a regular structure of interconnected switching nodes which direct packets from the input to the output via the internal connections of the

network. The topology of the network is designed so that it is possible for many packets to pass through the network without interference, while retaining a general structure that does not exclude any future application. The routing is normally implicit in the packet or setup by some controlling processor to a particular pattern allowing the nodes to remain simple and to process the packet without delay.

1.1 Local Area Networks

The development of the Local Area Network has been forced by the growing requirements for the sharing of resources between separate computer systems which are collected together in a small area. The cost of providing printing facilities and permanent file storage for many systems, which will only utilise them at low levels, far exceeds the cost of connecting a few services to a network that can be accessed by all. Other costs such as maintenance and provision of backup facilities as well as the additional cooling requirements for many duplicated systems also make the use of a network more attractive.

The main advantages of the Local Area Network are that the available bandwidth is large and that the delays are low. This has meant that much of the work that has gone into Wide Area Networks to make efficient use of the available bandwidth can be discarded to make way for lightweight protocols. These simpler protocols make access to the bandwidth easier, making the equipment simpler, resulting in a better overall performance of the system.

It may have been logistical pressures which caused the introduction of Local Area Networks, but the facilities that networks are able to provide have brought the introduction of new computing systems that rely on the network as a backbone, without which the systems would not be able to function.

The Xerox Network System[9] is a distributed office information system that provides a number of users of personal computer systems with tools for display, reproduction and shared access to documents. Services connected to the network, Ethernet[30], provide remote file storage, printing services and communications services. Each personal work-station is a self contained processor which does not require the network for general running but only for access to the enhanced services. This means that the network traffic is proportional to the usage made of these services.

The Cambridge Distributed System [32] operates in a different manner where the services connected to the network, the Cambridge Ring[47], provide every

function required by the user. A bank of processors is available to run a user's programs. These processors have no peripherals other than the network attachment which they use for all their communications with the rest of the system. Processors exist that have specialised peripherals, i.e. the file server for permanent file storage and the terminal concentrators that connect the user terminals to the system. For this system to run there must, by necessity, be a large amount of network traffic.

There have been studies carried out into the bandwidth usage of these two Local Area Networks. Shoch and Hupp[41] have measured the performance and utilisation of Ethernet while Temple[43] has measured the performance and utilisation of the Cambridge Ring. These studies have investigated the requirements that systems make of the bandwidth and the number and size of the packets being sent through the network. They have shown that the current utilisation of the networks is in general low but at certain times the bandwidth is heavily used. Temple found that the average daily utilisation of the ring was 1.3%, with a maximum hourly usage of 3.7% and a peak usage over 1ms at 20%. The Ethernet system operated similarly but with a burst utilisation of 37%. The overall utilisation of the network for a single day was approximately 2MBytes per station on the Ethernet and 10MBytes per station on the Cambridge Ring. The different philosophies of the two systems were borne out by the higher requirements per station on the Cambridge Ring.

1.2 New Applications for Local Area Networks

Still further changes in the structure of computing systems are being proposed where programs themselves are broken down into smaller self contained units that co-operate to solve some particular problem, for example the Cosmic Cube[42]. These systems are approaching the realms of the multiprocessor and the array processor where very large amounts of data need to be interchanged between large numbers of processing elements at high speeds.

As well as the servicing of computing needs there are also proposals for the interconnection of many other devices to be used in conjunction with computing services. One of the current areas of research is the integration of the telephone into digital networks along with video and multimedia documents, all of which will require large amounts of bandwidth to perform well.

The bandwidth requirements for the proposed services are varied. It is not

Service	Natural Rate	Burstiness
Voice	4Kb/s-64Kb/s	2-3
Interactive Data	1Kb/s-100Kb/s	>10
Bulk Data	1Mb/s+	1-10
Telemetry	<10Kb/s	>10
Image	10Kb/s-1Mb/s	1-10
Conference Video	1Mb/s	1-5
Video	>10Mb/s	1-5

Table 1.2: Data Rates for Various Telecommunications Services

in general possible to estimate the bandwidth requirements for any particular distributed computer system as this will always depend on how the particular programs run and how they are distributed amongst the many processors. For the telecommunications services some estimates of the bandwidth requirements have been made. Table 1.2 shows the characteristics of some of these services[20].

The traditional Local Area Network has relied on the use of a single communication medium for distributing information between stations. This form of network was suitable for the original applications of file transfers and printer service where peak loads could be spread out across time to make good use of the available bandwidth, but in newer systems where the network has become a fundamental part of the computer itself, such delays only result in the process as a whole running below capacity or not functioning at all.

There have, in recent years, been a number of networks proposed which are aimed at these new systems. These networks fall into two categories, higher data rate or multiple paths. Hubnet[25] and The Cambridge Fast Ring[43] are fast Local Area Networks which run at high bit rates and use new mechanisms which allow efficient allocation of the bandwidth. Floodnet[38] and the Flooding Sink[36] are networks which use parallel paths to improve the network throughput. They use a flooding technique to route packets through the network. These networks are capable of fault tolerant operation when a number of links or nodes have failed.

The application of packet switching techniques from Wide Area Networks to the operation of a Local Area Network is not in general desirable. In particular the delay when a packet needs to pass through a number of nodes and the need to manage complex routing information at each node, makes the operation slow and expensive. By removing all routing functions except for directing packets along a particular route, and by making the routing nodes simple, a practical packet

switching network can be constructed.

Binary Routing Networks apply ideas from Wide Area Networks and Interconnection Networks to produce a new form of Local Area Network that can provide the required performance.

1.3 Binary Routing Networks

To introduce the functions from Wide Area Networks and Interconnection Networks, a new form of network called *Binary Routing Networks*[16] has been proposed. It is research into this form of network which will be described in this thesis.

Binary Routing Networks can be described as *directed graph networks*. The network is constructed from a number of *Routing Nodes* linked together by unidirectional links. The nodes direct packets arriving at their inputs to one of two possible outputs, selected by a bit value in a route field in the packet. At the edges of the network the nodes are connected to host stations. Packets are injected into the network by the host stations and follow a predetermined route through the network until they again reach the edge of the network where they enter a host station's receiver. As there are many paths in the network, a number of packets can be moving through the network simultaneously, thus giving high total throughput.

There are many possible topologies for Binary Routing Networks. Some are useful but many are not. Typically the useful networks have regular structure which can be exploited in a number of ways. Irregular network structures are usable but they require more management and care in their use.

There are many features that make the Binary Routing Networks desirable. The high throughput, due to parallelism, that the networks are capable of, means that the technology used to implement the network does not need to be overextended to deliver the bandwidth required by some applications. The use of regular network topologies means that addressing throughout the network can be made uniform. The use of bridges is eliminated, as the need to separate groups of stations to reduce the impact of load on the network as a whole has been reduced. This also eliminates the use of special protocols when communications with stations on other networks is required. The packet's route delivers the packet directly to the destination and no intermediate station needs to handle the packet. No setup protocol is required to enable the requested packets to be passed from one network to the other. The expansion of the network into a much larger network is possible

without a significant degradation in the performance of the network. When the network is operated at high loads it does not reach a point where it is incapable of delivering packets, but continues to operate delivering packets at a fixed maximum rate dependent on the size of the network.

To make use of the features that have been described above restrictions need to be placed on the overall layout of the network. In particular if a tree structure was used there would be no parallelism and if irregular structures are used then addressing becomes complex. It is not intended that Binary Routing Networks replace existing networks but it will be shown that for certain applications they are preferable.

1.4 Earlier Work on Binary Routing Networks

In November 1980 a small group at Cambridge University, the Project H group, later called the Binary Routing Network Group, was set up to examine the possibility of constructing a network based on the basic principles described in Hopper and Wheeler[16]. This group consisted of David Wheeler, Andy Hopper, Steve Love, Steve Temple, Steve Crawley and Robin Williamson. In all the group held 14 meetings up until May 1981. The minutes of these meetings are available in the Cambridge University Computer Laboratory Library under the title Project H.

In their discussions the group considered a number of issues from the basic bit level to the strategy for caching and management of the routes in the individual stations, but most of the attention was placed on the protocols that could be used on the network and how they would influence the design of the stations.

In Chapter 4 of this thesis I cover a number of areas that this group discussed. In particular the use of randomization of the route field to avoid circulating packets, the routing nodes sending packets to some logger when an error is detected, and the use of special packets which are directed by the network to a route server. The mechanism described in Chapter 7 is my own idea.

1.5 The Aims of this Thesis

This thesis is concerned with the design of a Local Area Network based on the early ideas of Binary Routing Networks. The desired network should be capable of performing at high data rates and at low delay so that it can be used to supply the communications services that new applications require.

The main area of interest is centred around a new idea for the routing nodes. This idea dispenses with any buffering in the nodes. This modification to the network will be examined to see how it affects the operation and what difference in overall performance will be found.

Additional modes of operation that a non-buffering network provide will also be examined with a view to making the network more tolerant to faults and capable of delivering packets with less delay.

To compare the performance of the two forms of Binary Routing Network, the buffered and the non-buffered, as well as to the performance of other networks, a set of simulations will be developed. The simulations will model the operation of the networks with various sizes, topologies and transmission policies. The results will take into account the maximum throughput of the networks and the delay characteristics at fixed load values. These results will be contrasted with results from a numerical model of the network.

Finally the implementation of a non-buffered network will be described and the measured performance compared with that obtained by simulation.

1.6 Reading this Thesis

The rest of this thesis is broken up into a number of sections. Chapter 2 discusses a number of different techniques that have been used or proposed for Local Area Networks. Chapter 3 examines different systems used in interconnection networks of multi-processor systems. Chapter 4 introduces Binary Routing Networks and a number of options available to a designer. Chapter 5 introduces the Non-Buffered Binary Routing Network and covers its operation and new design options that it introduces. Chapter 6 examines the performance of buffer and non-buffered networks comparing them in a number of operating conditions. Chapter 7 discusses the design and construction of a simple Non-Buffered Binary Routing Network and compares its performance to that predicted by the simulation results. Chapter 8 ends the discussion with the concluding remarks, further work to be carried out and a proposal for using a Binary Routing Network as a component of an Integrated Services system.

Chapter 2

Architectures for Local Area Networks

This chapter examines some of the networks systems that have been developed to supply the communication requirements for Local Area Networks. I will examine each network in some detail, looking at the different methods by which stations transmit packets, and how the bandwidth is proportioned between these stations.

Along with each section there will be a small discussion of the merits of each system, in particular, areas of management such as the installation and expansion of the network, how the network can be divided up into smaller sub-networks, and how the network performs if the bit rate is increased.

2.1 Early Local Area Networks

Initial work into local area networks can be characterised by the single cable, multidrop, technology that provides for cheap interconnection of stations, while allowing high connectivity. There are two examples of this form of network which have provided the most interest since the mid 1970s. These are the Cambridge Ring and the Xerox Ethernet system. These networks have both proved themselves in the research environments and are now in commercial production. Many other systems exist but they mostly derive their operation from one or the other of these networks. Further examples of these networks can be found in general bibliographies on computer networks [44] [15] [1].

Ethernet

Ethernet [30] is a high speed packet switching system with distributed access control which operates with a basic data rate of 10Mb/s. Stations communicate by

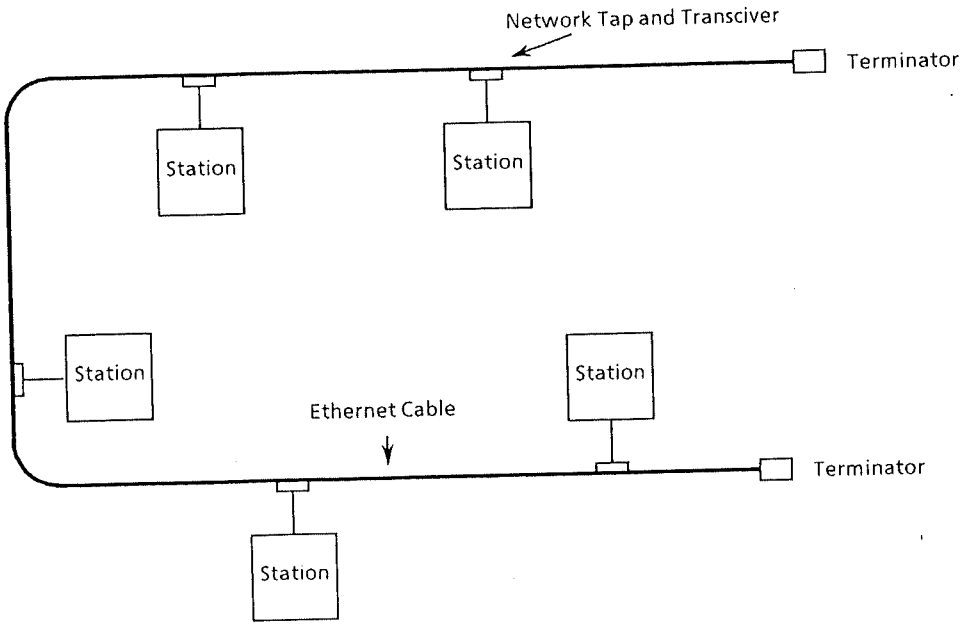


Figure 2.1: Structure of Ethernet

broadcasting packets on a single cable which connects to all of the stations on the network. This cable, often called the *Ether*, is passive and only provides a mechanism for distributing the signal. Figure 2.1 shows the basic structure of an Ethernet system.

To transmit packets, stations first listen to the network cable to determine if there is another transmission already in progress. If one is found then the station defers its own transmission and waits for the existing one to complete. If no signal is found then the station starts its transmission. This form of access control is called *Carrier Sense Multiple Access*. In addition to avoiding collisions with a packet on the network, Ethernet employs a *collision detection* mechanism which enables two or more stations which have commenced transmission (possibly at the end of the previously transmitted packet for which both transmitters have waited), to detect the multiple transmission and to cease transmitting. This allows the bandwidth lost to colliding packets to be kept to a minimum. The general name given to Ethernet and similar networks is CSMA-CD.

When a collision occurs Ethernet employs an *Exponential Random Backoff* to reschedule the retransmission of the blocked packet. This involves generating a random delay over a fixed interval. If a collision occurs again then the interval is

doubled and a new random delay is generated. The random delay is introduced to ensure that two stations do not retransmit packets which repeatedly collide with each other. The overall effect of this procedure is to throttle back the load that each station applies to the network. As the total load on the network increases, the amount of throttling increases in proportion to the number of collisions a packet experiences.

Packets on Ethernet are of variable length, which allows protocol implementors to select a size suitable for their particular implementation. An upper limit, of 1526 bytes, is imposed by the hardware to keep the size of buffers manageable and the average network latency small for stations using shorter packets. A lower limit on packet length is required to ensure that the network operates efficiently, although this minimum would normally be exceeded by the protocol overheads and preamble bits.

Ethernet can make efficient use of the bandwidth on the channel with 95% utilisation possible if the correct conditions occur. The main factors which limit the performance are the length of packets and the network round trip delay. The round trip delay is the time taken for a packet to travel from one end of the network to the other and back again. This time is important because it determines the maximum amount of bandwidth which can be lost due to a collision. When a station at one end of the network commences to transmit a packet, another station at the other end of the network will see an idle channel and may commence transmitting at any time up until the first station's packet arrives. The collision is only detected at the first station when a second packet has itself travelled the length of the network. If the packets being transmitted are very long with respect to the network turnaround time then the utilisation will be high. If the packets are small then the portion of the packet in which a collision can occur is large and under heavy load this means that the amount of wasted bandwidth will also be very large.

The performance is also affected by the use of a higher bit rate. If the packets remain the same size, then a greater proportion of the packet can be destroyed in a collision thus reducing the effective utilisation of the network. A fibre optic Ethernet variation, FIBERNET [39], which runs at 50Mb/s, requires packets of 50000 bits to achieve a 90 percent utilisation of the channel under heavy load conditions.

As Ethernet is based around a passive cable there is little that can go wrong with it which will stop the operation of the network. If a data error occurs then

the receiving station will, with very high probability, detect the error and reject the packet. Higher levels of protocol will detect the loss of a packet and a retransmission will occur. If, at a lower level, a transmitter fails, watch-dog circuitry can isolate the transmitter from the network leaving the other stations to operate. Not all errors can be detected and resolved by the hardware, but by reducing the places where such an error can occur the network will run with a high level of reliability.

Ethernet is one of the simplest networks to install. A single coaxial cable needs to be installed to pass near to each component to be attached to the network. When the component is ready to be attached a transceiver module is connected to the cable via a *tap*. This tap can be up to 50 metres away from the component. The connection of a new tap can be performed while the network is operating without interrupting the service. As the network operates in the broadcast mode there is no problem in adding new stations. Once the software has been informed of their existence then they can start operating.

As mentioned above, an increase in the bit rate of an Ethernet will not be reflected in a similar increase in the performance of the network. Under an equally increased load, the utilization of the network will be reduced. This is because the portion of a packet that is destroyed by a collision is increased wasting more of the network bandwidth.

Cambridge Ring

The Cambridge Ring[47][48] is a high speed synchronous Local Area Network based on *Slotted Ring* technology operating with a basic data rate of 10Mb/s. Stations transmit packets by inserting them into fixed *slots* as they pass by on the ring. Figure 2.2 shows the basic components of a Cambridge Ring. The ring is made up of a number of repeaters connected together in a loop. These repeaters are connected to station logic which is in turn connected to the host computers.

The links between the repeaters consist of dual twisted pair cables which carry data using a self-clocking modulation technique. It is possible that other forms of link can be used, the prototype ring at Cambridge had one link using optical fibre. The dual twisted pair has a second important function which is to carry the power to run the repeaters. This makes the repeaters independent of the station power supplies.

The repeater has a number of related functions. It first must recover the clock from the incoming data stream. This clock is used to regenerate the incoming

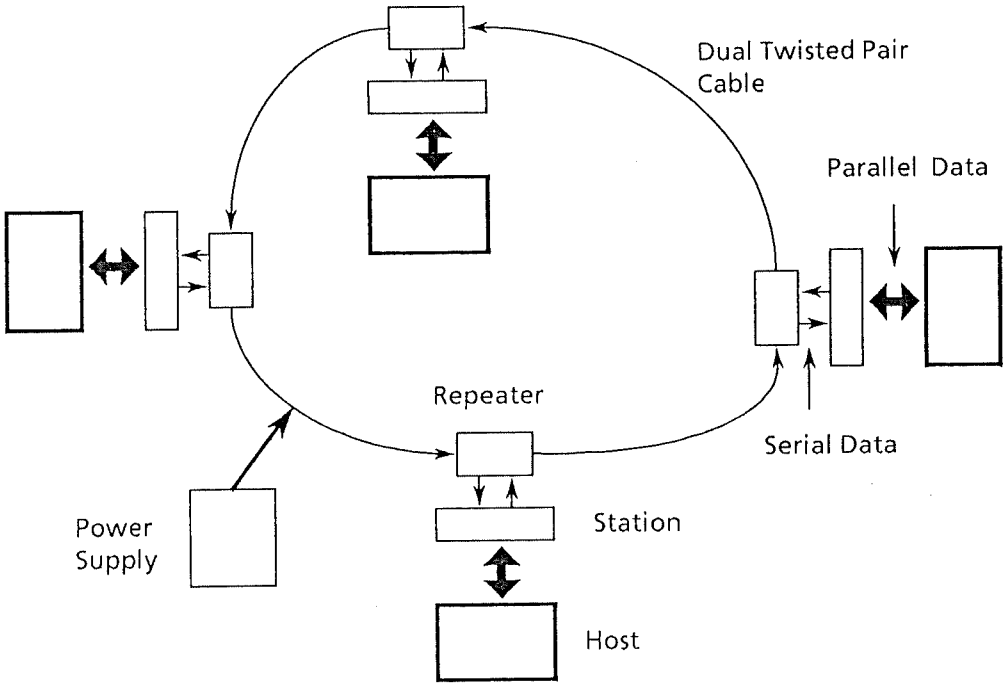


Figure 2.2: Cambridge Ring Structure

data and generate the outgoing data. If the host station is operating, then the clock and data are passed to the station logic and the returning data is sent on the outgoing link. If the station is not operating then the incoming data is passed directly to the output.

The station has the logic to examine the data bits searching for packet frames, and to insert and extract packets from the stream in accordance with commands that are sent from the host computer.

The repeater and station are separated so that they can be powered from separate power sources. The repeater is an essential part of the ring and thus must be able to continue its function even when the host computer is powered off. Provided that the host computers follow simple procedures when they power up and down no interruption to the operation of the ring is caused.

The Ring operates by having a sequence of bits continually circulate around the network. These bits, which form a train of packet outlines followed by a *Gap*, total up to be the exact electrical length of the ring. The bits are first generated by the *Monitor Station* at start time, using a local clock to generate timing information. The transmitted bit sequence consists of enough packets, marked *Full*, to reduce the Gap to less than a packet. When the returning bit stream settles down, and

is determined to be what is being transmitted, the monitor station connects the returning stream to the output, constructing a self oscillating network. Once this procedure has been performed the number of bits in the network remains fixed and changes in the electrical properties of the cable and repeaters, due to heat or ageing, will result in changes in the frequency of oscillation. Discussion of the problems with this form of clocking can be found in Leslie[27].

A station wishing to transmit a packet first fills up an internal shift register with a copy of the packet to be transmitted. When an empty packet comes past the station, the local packet is injected into the bit stream instead of the previous circulating data, with the packet marked as Full. The station must then wait for the packet to complete a circuit of the ring to mark it empty as it passes. The station must pass on this empty packet and must not re-use it. This procedure ensures that a single station cannot hog the entire bandwidth by injecting many packets into the ring and using the old packets when they have returned.

The bandwidth of the returning packet is not entirely wasted as the packet carries response bits indicating if the receiver had accepted the packet. There are 4 possible responses: 1, *Ignored*, the destination station does not exist or is not powered on; 2, *Accepted*, the packet was accepted; 3, *Busy*, the receiver was busy and could not accept the packet; and 4, *Unselected*, the station does not wish to receive the packet. These responses are used to control the transmission of larger packets which will be discussed later.

Each packet, called a *minipacket*, carries 4 bytes of data, one byte each of the source and destination addresses and 2 bytes of user data. The small size of the minipacket means that the amount of useful data that can be carried by a single packet is restricted and it is usual for systems to string many minipackets together to form a large packet called a *Basic Block*. The first minipacket, called the *Header*, contains a header pattern, a Basic Block type and a count of data minipackets to follow. The second minipacket is a port number which is used to direct the packet to the appropriate support software. This is followed by the body of the packet holding the data and a checksum, which covers the whole block.

To transmit a Basic Block the host first sends the Header packet. This packet will be continuously retried while the response indicates Busy. Once the receiving station has accepted a header packet it will set its selection register so that only packets from the one station will be accepted. This simplifies the reception routines which will not have to look at each packet individually. Once the Header packet is accepted the Port packet is sent. This will also be continuously retried until

accepted. At this point, if the receiving station determines that it is willing to accept the rest of the Basic Block, then it will leave the selection register set to the sending station and collect the remaining minipackets. If the receiving station does not wish to receive the basic block, either because the port is unknown or the size is too large for the allocated buffer, the receiver sets the selection register to reject all minipackets for a fixed period. It is then up to the transmitter to detect that the data packets have been rejected and to terminate transmission. If at any stage the response bits indicate Ignored or Unselected then the transmission is aborted and reported to a higher level in the protocol.

The above protocol for sending a Basic Block may seem to be complicated but it was the designer's opinion that the interface to the ring should be as simple as possible to reduce the cost of connecting stations. For small inexpensive stations where program time can be interchanged for hardware expense, the processor can poll the interface. More complex hardware can be constructed for machines, such as file servers or multi-user machines, that cannot afford to wait for transactions with the ring to complete.

Error control on the ring is performed by the Monitor Station, which monitors all packets as they pass. In particular it detects packets which have been marked as full and remain full as they circulate. These can be generated by either a data error or a station being powered off during a transmission. All full packets have the *Monitor Pass* bit set as they pass the monitor. If the packets have both the full bit and the pass bit set then an error is assumed and the packet is marked free. When a packet is transmitted the pass bit is always cleared so that the monitor station cannot assume an error on the first circuit. Another error control function that the monitor station can perform is to fill empty packets with random data as they pass and to check the data when the packets return. This can allow the early detection of errors caused by hardware which is on the verge of failing.

The structure of the Cambridge Ring requires the ring cables to connect directly to each of the stations on the network. To add a new station the ring cable must be broken to insert the new station's repeater. Once the new repeater is inserted the new station can be connected at some later time. As the repeaters are powered from the ring itself, the installation of a new station requires that there be enough capacity in these power supplies. In addition, care must be taken to avoid adding extra delay to the ring as this will reduce the maximum throughput of a single station. Adding enough extra bits could allow an additional packet to be introduced which would supply useful bandwidth to the stations. By clustering

a number of stations onto a repeater extender, the load on the ring power supplies can be reduced and the delay through the group of stations reduced. The repeater extender operates as a station connected to the ring by a single repeater. Instead of connecting to station logic, the repeater extender emulates the operation of a portion of the ring in TTL logic. The real stations connect to this pseudo ring section. The delay is reduced by eliminating the data recovery circuitry which would have been duplicated without the repeater extender.

The operation of the Cambridge Ring at higher data rates has been achieved in the design of the Cambridge Fast Ring reported later in this chapter. For networks where a large portion of the ring delay is found in the cabling, increasing the data rate will not increase the maximum point to point data rate for the network. This is because there is an increase in the number of packets circulating in the network. The time taken for a packet to return remains the same, although there will be many more packets in the network and the total throughput of the network will increase. For a network with just the fixed delays in the repeaters the number of packets will be unchanged when the bit rate is increased and they will circulate faster.

2.2 Newer Local Area Networks

The two networks described above caused a lot of interest in Local Area Networks and many different designs have been proposed. As mentioned before many of these have been slight modifications to the control mechanisms to make the allocation of bandwidth more equitable. Other changes have also included methods of making the network tolerant to failures including breaks in the communication media.

In the following section techniques such as token passing and flooding are described, as well as ways of using higher bandwidths and multiple paths.

IBM Token Ring

The alternatives to the synchronous slotted ring technology are the *token* passing rings. Token passing rings/networks are characterised by their method of allocating the bandwidth on a single shared medium. A single station is given the sole control of the channel for its own use and the authority to use the channel, or *Token*, must be handed over before any other station can use the channel.

The IBM Token Ring[6] is one of these networks. It is a synchronous ring, deriving the clock from the Monitor Station's local oscillator. The monitor station

provides an elastic buffer to ensure that the ring is an integral multiple of bits, varying the buffer length when the electrical length of the ring changes.

A packet frame consists of Start delimiters, an Access Control field and destination and source addresses which make up the frame header, a variable length information field, and a frame trailer carrying a Check Sum and End delimiter. To distinguish the frame header from data that may be carried in the information field, the Start and End Delimiters are represented by violations of the Manchester encoding used on the links. This means that a station that is just starting up and which has not yet synchronised does not misinterpret a random stream of data as a free token which it takes and uses as its authority to send packets. If such violations were to be generated by transmission errors then the ring must go into an error recovery state and resynchronise. In the Access Control field is the control information of the network, particularly the Token bit, which indicates whether the Token is busy or free. A packet with this bit clear is a free token and can be claimed by any station wishing to transmit a packet. If the bit is set, then the packet carrying another station's data is passing and must be left untouched.

Stations that are not transmitting data operate as repeaters, passing data from their input to their output. Stations that are not powered on are by-passed by a relay so that they do not interfere with the ring operation.

When a station wishes to transmit a packet it must first obtain the free token. Once the station has received the free token, which it marks as a busy token, it breaks out of repeater mode and inserts its packet into the data stream. Finally the station waits for the trailing end of its packet, passes on the free token and reverts to repeater mode.

For correct functioning of a token ring there must always be exactly one token. At start up time a Monitor Station must construct the first token. To aid in recovery from hardware failures all stations have the logic for becoming the monitor station. If it becomes necessary for the token to be reconstructed because of a catastrophic failure, or at power on, all stations enter the monitor mode and start sending out *monitor-recovery* messages which contain the address of the station. Each station then listens for a returning message comparing the address value in the packet with the station's own address. When the station's address is found to be lower than the address of the incoming packet, the station returns to the repeater mode. Finally one station remains in the monitor mode sending out the recovery message which it detects affirming its position as monitor. The new monitor then transmits an *end-of-monitor-replacement* message and generates the

new free token.

Loss of the token can be detected by the monitor station by observing the absence of either a busy or free token which must come within the interval defined by the maximum packet length and the ring latency. When this time has expired the monitor station generates a series of idles to ensure that the network has settled down before generating a new free token. If the monitor station has failed, a longer timeout will be detected by one of stations which will then enter and force all other stations into the monitor recovery phase, to determine the next station to become the monitor, which will then generate a new token.

Corruption of a free token, causing a busy token to circulate indefinitely, is detected by a monitor pass bit which is contained in the access control field. Every time the busy token passes the monitor station the pass bit is set. When the token returns this bit is examined, and if it is set and the token is still busy an error is assumed to have occurred, and the monitor station clears the ring and generates a new free token.

The third main error associated with token passing is the duplication of the token. This form of error is detected by the transmitting stations, who examine the returning packet to ensure the reception of their own transmission by examining the source address. When a different transmission is detected the stations revert to the repeater mode without generating a free token. This forces a lost token situation which is detected and corrected by the monitor station.

There is a priority mechanism which allows the normal sequence of token passing to be interrupted. This allows certain nodes which require a guaranteed bandwidth to transmit their packets by pre-empting the free token. A station which requires guaranteed bandwidth may set the Reservation Indicator in the access control field of the token as it passes the station. When the station transmitting the packet receives the access control field back, it sees the reservation bit set and will not issue a free token, but will transmit idles until the packet has completed its circuit when it reverts back into a repeater. The Requesting station, after receiving the end of the packet followed by the idles, generates a new priority token. Later the requesting station will generate a new free token, with the reservation bit set followed by the address of the station that was pre-empted. The protocol specifies that no other station is allowed to take this free token. When the packet returns to the pre-empted station the reservation bit is cleared and the free token is released to carry on its normal pattern of operation. This mechanism is desired as the pre-emption of the token and its subsequent release is completely fair to

those stations which are running in the asynchronous mode.

The installation of a Token Ring is straightforward. A cable must pass in a circuit around a series of distribution panels, each capable of connecting to a number of hosts. From these panels the hosts are connected via bidirectional cables. When the station is inserted into the ring the data passes down the cable to the host through the interface circuitry and back to the distribution box. Provided that there is an available connection to the distribution box new stations can be attached. When a station is inserted or removed from the network there is a high probability that a packet will be corrupted as there is a delay as the connection is made or broken via the relay in the distribution box. This loss of a packet must be detected by the high levels of software, and if it is the token that is lost recovery must be undertaken by the monitor station. As the system operates in broadcast mode there is no additional requirement other than to inform the software when a new station is inserted.

Increasing the bit rate of the Token Ring will not achieve a direct increase in the performance seen by each of the active stations on the network. There will still be the physical delay as the packet circulates around the network. The time for the head of the packet to pass completely around the ring will vary depending on the ratio of the delay that comes from the cables and the delay that comes from the stations. If the cable delay is dominant, the time for the header to circulate will be little affected by an increase in the bit rate, whereas the opposite effect will occur when the stations provide the most delay. This will result in a non-linear increase in the available bandwidth. The gain in performance will come from the ability of sending longer packets in the same time period. If the packets remain the same size only a smaller improvement will be achieved.

Hubnet

Hubnet [25] is a packet switching network with a rooted tree structure. The network is constructed of interconnected *Hubs* with attached hosts via *Network Access Controllers*. Each hub has two jobs; first it arbitrates in the selection of packets, and secondly it broadcasts selected packets to the connected hosts and hubs. The network access controller's function is to transmit packets on the network and to ensure, as far as possible, that the packets reach their destination by retrying until no transmission error is detected at the transmitting station's own receiver. Figure 2.3 shows the basic structure of Hubnet with a number of stations and sub-hubs. Figure 2.4 is an expanded version of the same network showing the selection tree

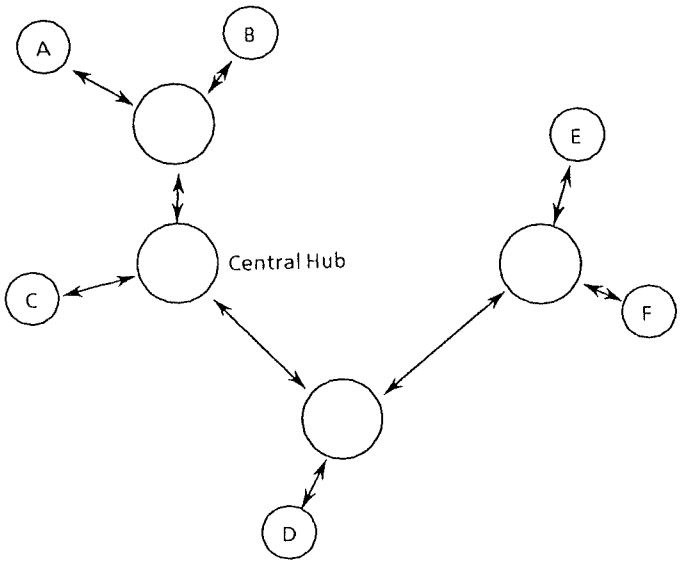


Figure 2.3: A Simple Hubnet Network

and the broadcast tree.

In the simplest form Hubnet consists of a single hub connected to host computers. The connections are made from a dual optical fibre cable operating at a speed of 50Mbits/s, with an outgoing cable and an incoming cable. The transmission cables from all stations are connected to the arbitration half of the hub and the reception cables are connected to the broadcast half of the hub. Finally the two halves of the hub are also connected so that the packets selected by the hub are broadcast to all stations.

The network access protocol is very simple. Stations which have packets to send place them in the transmitter buffer in the network access controller, and the packet is then transmitted. When the packet arrives at the hub there are two possible outcomes. If the hub is idle the packet will be selected and relayed out on the line to the broadcast half of the hub, where it will be sent to all stations in the network. If the hub was already passing another packet then it will ignore the new packet and continue to pass the original packet. If two or more packets arrive at the hub simultaneously, then one will be selected and the other ignored. The transmitting stations listen to the packets coming in from the network and compare them with the packet that they are transmitting. One of these stations

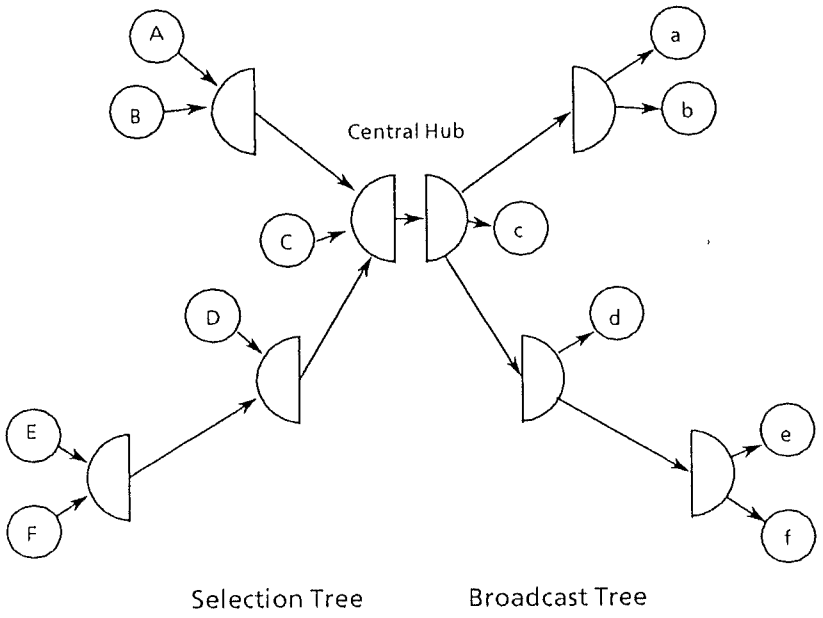


Figure 2.4: Hubnet:Expanded to show Selection and Broadcast Trees

will hear its own packet and will, if the packet is long, continue to send until the transmission is completed. The stations which find that their packet was not selected will discontinue sending and retry the packet later. It is possible for a transmitted packet to be so short that the station has already completed sending the packet before the detection of the incoming packet can determine if the transmission has succeeded. The packet length does not have any effect on the success or failure of a packet transmission as the selection is made on the leading edge of the packet, and once selected a packet will be allowed to pass unhindered.

The more complex form of Hubnet has a number of hubs, each with their own hosts, connected together. One of the hubs is chosen as the central hub and has the arbitration half connected to the broadcast half. The other hubs take the selected packet and pass them on to the more central hub for selection there and possible transmission. They also take the packet from the central hub and broadcast it to all their connected hosts. All hubs are similar in construction and the only difference is that the link between the two halves is only made directly in the central hub. In this mode of operation the selection of packets is distributed between a number of hubs and can thus limit the number of inputs a single hub needs to have, even for a large network.

The layout of Hubnet is not constrained by the access protocol of the network and it is possible to have very long links. Whenever a new node is to be added to the network, provided a connection to a hub can be established, the station can be connected without any other operations than to plug in the cable. If the local hub has no free connections then one of the existing nodes can be removed and a new hub connected in place of the node. This is then connected along with the new station to the new hub. If a station is added to the network via a long cable it will not cause the network to perform badly and the original nodes will not notice any difference as the network arbitration will have been unchanged for them. The remote station will notice the extra delay in the return of its transmitted packet and the time between retries may need to be adjusted to suit, which may result in a reduction in the number of packets which that station can generate.

Depending on the layout for the hub interconnections, different stations can be given different priorities. Stations that are connected to the network nearer to the central hub will have a higher probability of gaining access to the network than those which are connected further out.

The operation of Hubnet can be seen as an Ethernet with an active arbitration mechanism. This mechanism eliminates the possibility of a transmitted packet from being corrupted by a second packet, and thus reduces the wasted bandwidth to just the period between the end of the transmission of a packet and the beginning of the next transmission. This allows Hubnet to operate at throughputs verging on the bandwidth on the transmission medium used.

As mentioned before, the addition of new stations to the network is simple and can be performed without disrupting the operation of the network, provided that no rearrangements are required. The network can be operated at high data rates, though 50Mb/s is currently considered high for a Local Area Network, with all the bandwidth available for use. The only loss comes from the physical delay between the end of one packet and the start of the next.

The Cambridge Fast Ring

The Cambridge Fast Ring[43] has been designed to fulfil the requirements for higher throughput networks and to reduce the complexity and cost of construction of a ring station. It has also been designed to provide a mechanism for the interconnection of rings into a larger network of many rings by simplifying the construction and operation of bridges.

The Cambridge Fast Ring is a slotted ring with a basic clock rate of 100MHz,

10 times the rate on the original Cambridge Ring. If the same structure, as found in the Cambridge Ring, was to be used at this higher data rate the number of packets in the ring would be increased by a factor related to the ratio of delay in the repeaters and the cable, which would be less that 10. The bandwidth available for point to point operations between two stations would only increase by the reduced factor. This is because the stations would still be restricted to use 1 packet per ring revolution. To make the extra bandwidth available to the stations the size of the data field in the packet has been increased to 32 bytes. This means that the packet overhead is reduced to a smaller percentage of the overall packet.

The address fields in the packet are 16 bits. This allows absolute addressing over the whole network and avoids the use of sub-net address extensions. By having a global addressing scheme, bridges need only be concerned with the destination addresses of the packets passing their receivers. A 64K bit table is used to indicate which addresses will be accepted on the other ring, including those routed by a further bridge. The operation of the bridge consists of listening to packets and filtering out remote destinations and transmitting these on the other ring. The only management function required is to initialise the table, avoiding generating loops that could cause packets to circulate indefinitely.

The Fast Ring implementation is divided into two functional parts. The high speed logic to implement serial to parallel and parallel to serial conversions is implemented in ECL using a custom designed chip. This chip is known as the 'ECL' chip. The ECL chip also implements the synchronising and gap detection circuitry. The bytes from the ECL chip, which are clocked at 12.5MHz, one eighth of the serial clock rate, are processed by the second chip, which is constructed in CMOS using a semi-custom design. The 'CMOS' chip implements all the functions required for the receiver and transmitter, including the interface to the host, an 8 bit microprocessor bus, FIFOs for transmit and receive data and all the logic needed for the node to operate as a monitor station. The CMOS chip is also capable of being connected back to back with another CMOS chip to form a ring-ring bridge. The CMOS chip also has the logic to drive a 64K dynamic RAM chip which can be used to implement an address filter, only allowing desired packets to be received. Figure 2.5 shows the basic components of the fast ring.

The normal configuration of the network has one ECL chip and one CMOS chip operating in a similar manner to the Cambridge Ring. Alternative configurations allow a number of CMOS chips to be connected to a single ECL chip, but this places constraints on the minimum size of the Gap. A further configuration

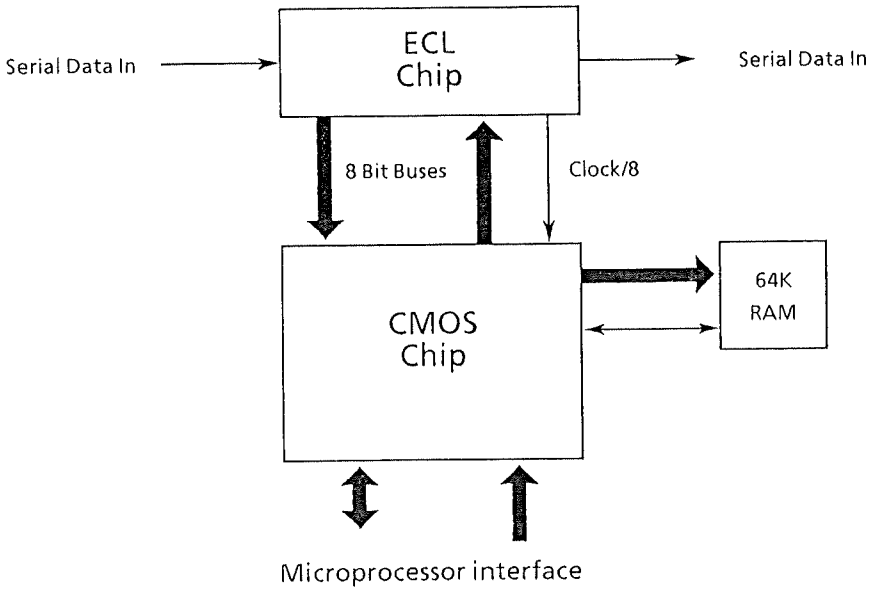


Figure 2.5: Cambridge Fast Ring: ECL and CMOS chip Interconnections

allows a number of CMOS chips to be connected without the ECL chip, thereby implementing an 8 bit parallel ring. This network will still operate at the same rate but is confined to a small area.

The Fast Ring packet does not carry any response bits as in the original ring design. The only bit of information that is returned indicates that a packet should, or should not, be retried. This information is not passed to the user as experience has shown that communications across a bridge cannot return such codes to the user if a failure occurs on the other ring(s) and that the higher levels of protocol need to cover lost packets any way. The only time a packet will be marked for retransmission is when the receiver's buffer is full. Retransmission will then take place some time later.

Another protocol change is the addition of *Channel* slots in conjunction with the normal packets. These slots, which can also be used as normal slots, can be reused by a sending station provided it can supply the data fast enough to keep up with ring speeds. If a transmitting station has a second packet ready when the first packet returns then, provided the original slot was a channel slot, the second packet can be transmitted in the same slot. If the slot was not a channel slot or there was no data to transmit, the slot is marked as being free. This allows

two stations to communicate at high speed along with other stations operating at lower speeds.

Most of the same comments for the Cambridge Ring apply to the Cambridge Fast Ring. However the fast ring uses new protocols to make more efficient use of the bandwidth.

2.3 Flooding Local Area Networks

In a packet switching store and forward network it is necessary for the switching nodes to have enough information about the topology of the network to determine where a packet is to go on its next hop. This is usually implemented as routing tables which indicate the most desirable link for a packet to be sent out on to complete its journey. A simple implementation would indicate the shortest route while a more complicated system would take into account the load on the links within the network and distribute the packets so as to spread the load. The routing tables are the main cause of trouble for these networks. When a link is broken it is necessary to inform the network to avoid the build up of undeliverable packets at the switching nodes around the broken link and to arrange the use of other suitable routes to carry the packets until the link failure is corrected. The reverse problem also exists when the link is repaired. Secondly the routing tables need to be updated when a new node is added and again when a node is moved.

A technique whereby the network is flooded with packets eliminates the need for routing tables and the updating mechanisms needed to maintain them. A packet arriving at a node is sent out on all the links other than the one on which it arrived. This way the packet is sent to all nodes in the network and thus to its destination. The effect is that the packet will arrive at the destination via the current shortest route.

The main problem is that there are now many more packets in the system which must be eliminated. The normal way to eliminate the extra packets is to give the packets a lifetime after which a switching node receiving the packet will discard it without retransmitting it. For efficient use the lifetime should be set as small as possible to remove the packets as quickly as possible, but long enough for the packet to follow the longest possible route to the destination. It is also desirable to avoid packets circulating around loops in the network as this will take up bandwidth usable by other packets. The nodes must also ensure that when a packet arrives all further duplicates that arrive along alternate paths are

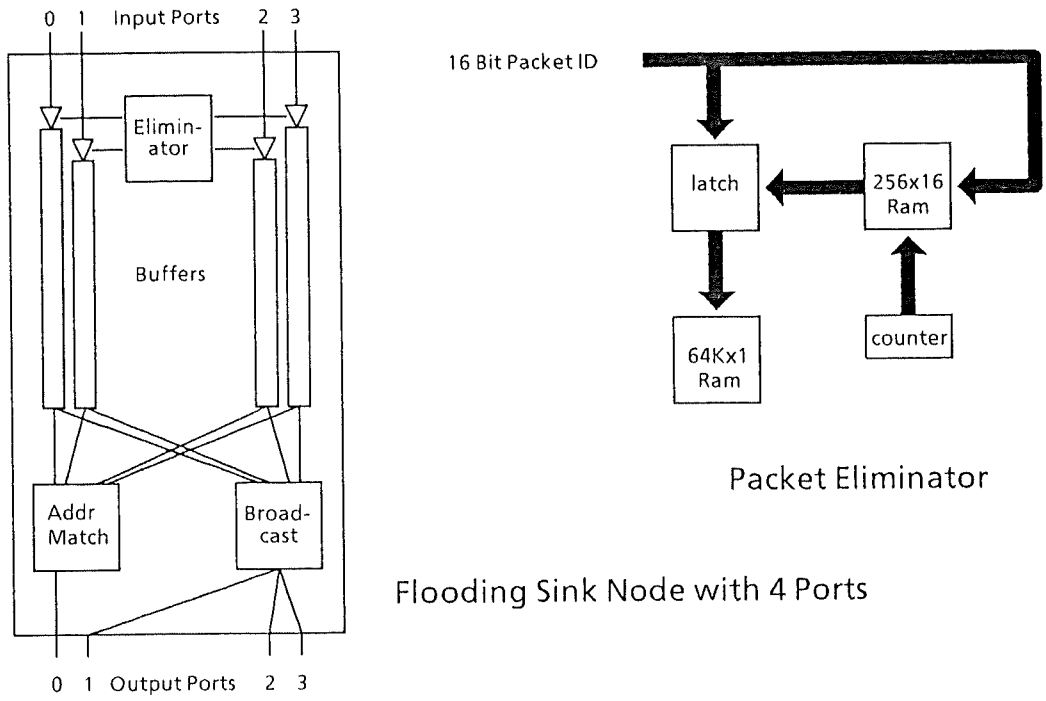


Figure 2.6: A Flooding Sink Switching Node

eliminated.

The following two networks describe different ways that the flooding technique can be used in the Local Area Network environment where the application of this technique has been previously considered impractical because of the complexity of the nodes and the delay in delivering the packets.

The Flooding Sink Network

The Flooding Sink[36] uses the flooding technique to route packets to their destinations and uses a table of the last N packets received to eliminate duplicate packets which arrive at a node, thus killing off the packets in the shortest possible time. The size N for the table is dependent on the size and structure of the network.

The network is constructed of interconnected switching nodes which are connected in an arbitrary topology. In the prototype implementation the switching node has 8 inputs and 8 outputs, the input/output pairs 1-7 are used for network connections while the pair 0 is used for a connection to a station (see figure 2.6).

Address Comparison logic is used to match the address of the station with the destination address of the incoming packet. If the packet is destined for the station

the address comparison logic will direct the packet to the 0^{th} output, otherwise the packet is broadcast on all the other outputs excluding the output paired with the input on which the packet arrived. The address comparison logic is capable of matching address patterns which contain "don't-care" fields. This allows the filtering of groups of addresses and can be used for bridging between distinct sub-networks, avoiding internal packets from one sub-network flooding into another sub-network.

The packet header contains a source address, a serial number and a destination address. Each of these fields in the prototype implementation are fixed at 8 bits. The source address and the serial number are used together to make a *message identifier*. The message identifier is used to uniquely identify the packet within a particular time frame. It is assumed that the serial number is large enough to ensure that by the time it has wrapped around, the previous messages will have been eliminated from the network and the new packet will not be mistaken for the duplicate of a previous received packet and discarded.

Each node has a packet eliminator whose job it is to detect replicated packets that arrive at a node within a time frame. It is divided into two parts. The first is a single bit wide map of all possible message identifiers (2^{16} entries in this system) and a table of the last 255 received message identifiers. Each map entry indicates whether the message identifier is one of the table of the last 255 received messages by presence of the value 1. If the entry is found to be 1 then the packet is discarded. If the entry is found to 0 then the packet is assumed to be new. To enter a new packet into the table, its message identifier is added into the table of last received packets in the next empty slot and its entry in the map is marked with a 1. The 256^{th} entry in the table, the least recently received message identifier, is used to clear the corresponding map entry to allow the reception of another packet in the future. This table slot is then released.

The Flooding Sink has a number of interesting properties. As packets will arrive at a destination from a number of different routes, each packet should be delivered to every node in the network. Packets that are destroyed by errors will still be delivered to their destinations via a possibly longer path. If there are many alternate paths in the network, then many copies of the packet can be destroyed and the packet will still be delivered. This mechanism means that the low level protocol can treat the network as a reliable datagram service, and the use of acknowledgement packets can be moved to a higher level. Limited size of buffers in the nodes can also cause a number of packets to be lost. When a packet arrives

at node where the buffers are full it can be discarded but there are a number of copies of the same packet still in the network and one should eventually reach the destination . By analysing the load on the network, the size of the buffers can be designed to use the least amount of memory while still enabling the network to operate at low error rates.

The Flooding Sink is effectively a broadcast network that distributes packets to all other stations along a number of different paths instead of along a single channel. The network is easily expandable in size although the total bandwidth of the network will not be any greater as every packet, unless completely lost, will eventually pass through every link. The bandwidth can be increased by splitting the network into sub-networks and using particular nodes as bridges for filtering internetwork packets. This splitting will improve the overall performance of the network and there will be no extra cost in the bridge function, as packets would have taken the same route and suffered the same delays in a combined system. Increasing the speed of the Flooding Sink will have a direct effect on the performance of the network, as no collisions exist to waste the bandwidth which should increase proportionally as the bit rate increases.

Floodnet

Floodnet[38] uses a meshed topology to connect together many stations in a partially connected network. Stations are connected to nodes, which are in turn connected to other stations and other nodes, by full-duplex links. Each node may be connected to many links. Figure 2.7 shows the basic structure of Floodnet.

The nodes work at the bit level passing packets from the input link to the output links with little or no delay. Packets pass between stations and nodes and between nodes and nodes along a path that is established by a mechanism based on flooding. Nodes that are not involved in transmitting a packet are free to be used in setting up and transmitting other packets. Thus there may be many packets in transit across the network, provided that they do not require a common node.

The mechanism for establishing a path is as follows: All nodes not presently involved in passing a packet are in the idle state. When a packet is injected into the network, it floods all of the reachable links on the network with the header information. When a packet enters a node, which is not already busy, the node changes to the busy state and transmits the packet on all of the other links that are connected to that node. This occurs at each node radiating out from the transmitting station. If the node is already active then the packet is blocked at

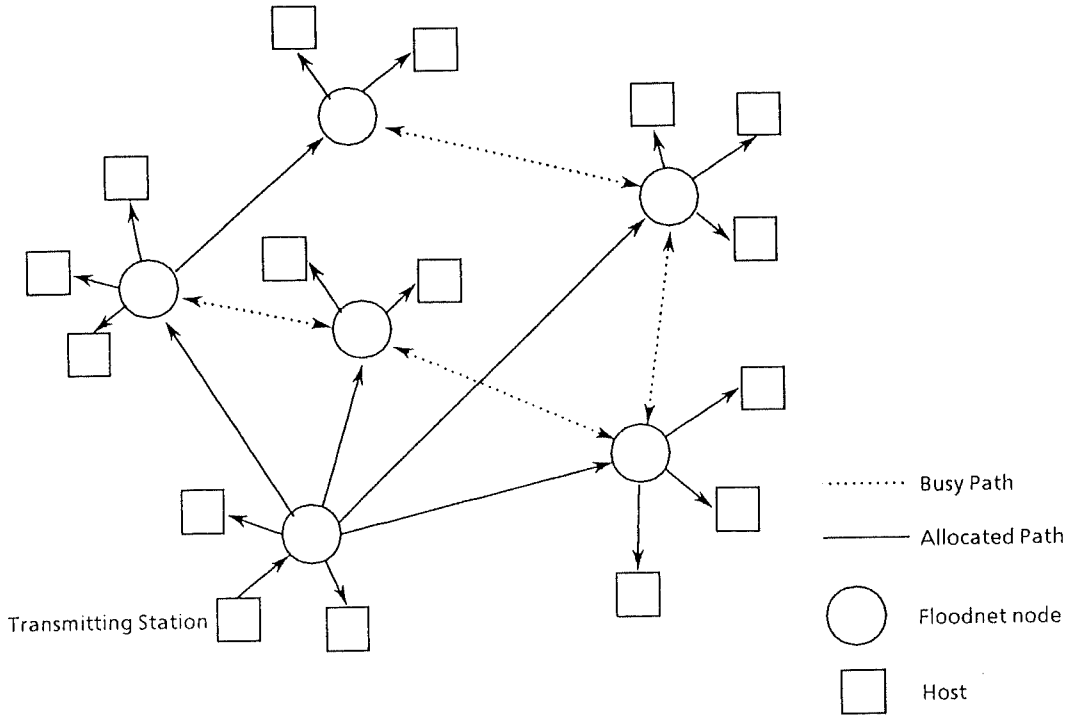


Figure 2.7: Floodnet: General Layout

that point. The blocking node may be carrying a packet for an existing connection or may be flooding the network in an attempt to establish a connection, possibly as part of the same flooding sequence. Eventually the packet will reach the edge of the network where it will be delivered to the stations. All stations examine all incoming packets. A station will reject those packets which do not carry the station's own address. Rejecting stations pass back a signal to their node. If a node receives a rejection from all its output links, it then sends a rejection on the link that originally brought the packet and sets itself back into the idle state. A blockage on a link is considered as rejection. The destination station does not send a rejection but continues to receive the packet. This results in a route that follows the shortest path available, and allows all nodes not directly involved in sending the packet to return to the idle state, enabling them to establish other connections in parallel. If there is no path to the destination then the rejection will be passed right back to the transmitter, which will cease to transmit. Packets thus blocked would be retried later after some waiting period.

Once the path has been established it is possible to send a large amount of data. It is not necessary for the network to impose any maximum length on the packet, apart from the limit in transmitter and receiver buffer, but a timeout on packets

passing through a node can be used to detect failures, isolating the offending node, and allowing the network to operate at a reduced level of service.

As with the Ethernet performance, Floodnet has a better throughput when the packets are longer. This is because proportionally less time is used in flooding the whole network with routing information. If short packets are used then there will be a larger proportion of the time when the network will be taken up with flooding by packet headers. If larger packets are used then once the path is set up by the flooding mechanism, which will take a smaller proportion of the packet time, there will be a greater probability that other packets can be routed to run concurrently.

Routing is distributed through the network. This enables the logic of the node to be kept very simple, as no routing information needs to be stored. Redundancy comes automatically as the packet will find the shortest available route, if one exists, which will exclude any faulty links or nodes. Expansion of the network is relatively simple as new links can be added to any node without any change to the rest of the network, or changes to software dealing with routing. High volume traffic can be handled by adding extra links where required. If the bit rate is increased then there will need to be a corresponding increase in packet size to ensure that the same network utilisation is achieved. This is because the physical delays will remain the same and thus the flooding phase will take up a larger portion of the original packets transmission time.

2.4 Summary

In this chapter we have seen a number of different ways that Local Area Networks have been implemented. In general these networks have used the broadcast method to deliver packets from one station to another. The use of restrictions, in size and number of stations, has achieved low cost networks using simple protocols, which are capable of operating at high bandwidths with low delays, and which combine to give good overall performance.

For many of these network a single cable is used to distribute the packets amongst the many stations, which gives a maximum load that the network is capable of handling. To gain a performance increase by the use of multiple channels, requires the intervention of special circuits, bridges, to select and distribute packets from one channel to the other This increases the performance for stations in the same sub-network but limits the performance between stations on different

sub-networks. Increased performance can be obtained by the increasing of the bit rate used on the network, but it does not apply to all networks.

The use of multiple paths is proposed in some of the newer architectures for Local Area Networks to achieve a higher network performance without increasing the bit rate. Multiple paths mean packet routing problems where the network is required to select the most desirable path for the packet, resulting in added complexity to the network nodes. Simplified flooding algorithms have been applied to these networks to bring the design of the nodes back to those associated with Local Area Networks, of high speed and low delays coupled with simple protocols.

Chapter 3

Interconnection Networks

At the other extreme of the communications spectrum from local area networks are the multi-processor systems where data and instructions are passed through a communications network. These networks are termed *Interconnection Networks*, or sometimes *Permutation Networks*, because they allow paths to be set up between the processing elements and the memory modules, and processors and processors. These systems have many arrangements and it is usual for the interconnection network to be designed for a specific processor which is being constructed to solve a particular class of problems, but other more general interconnection networks are available to allow general permutations.

These networks are divided into two categories *Static* and *Dynamic*, dependent on their implementation. The static networks are normally used to construct a processor array to solve a particular class of problem. The dynamic networks are suitable for a larger class of problems where the interconnection patterns are not known when the processor is constructed, though the required patterns should be carefully thought out to avoid bottle-necks that may arise from the particular implementation.

3.1 Static Interconnection Networks

Static interconnection structures are not of direct interest in this discussion but a few examples have been included for purposes of illustration. Further information on these structures can be found in Feng[14].

There are many different strategies for the static interconnection of processors (see Figure 3.1). These can be divided into groups depending on the degree of interconnection. The shape of the network tends to restrict the class of algorithms which can be run on the processor by incurring high costs where data needs to be

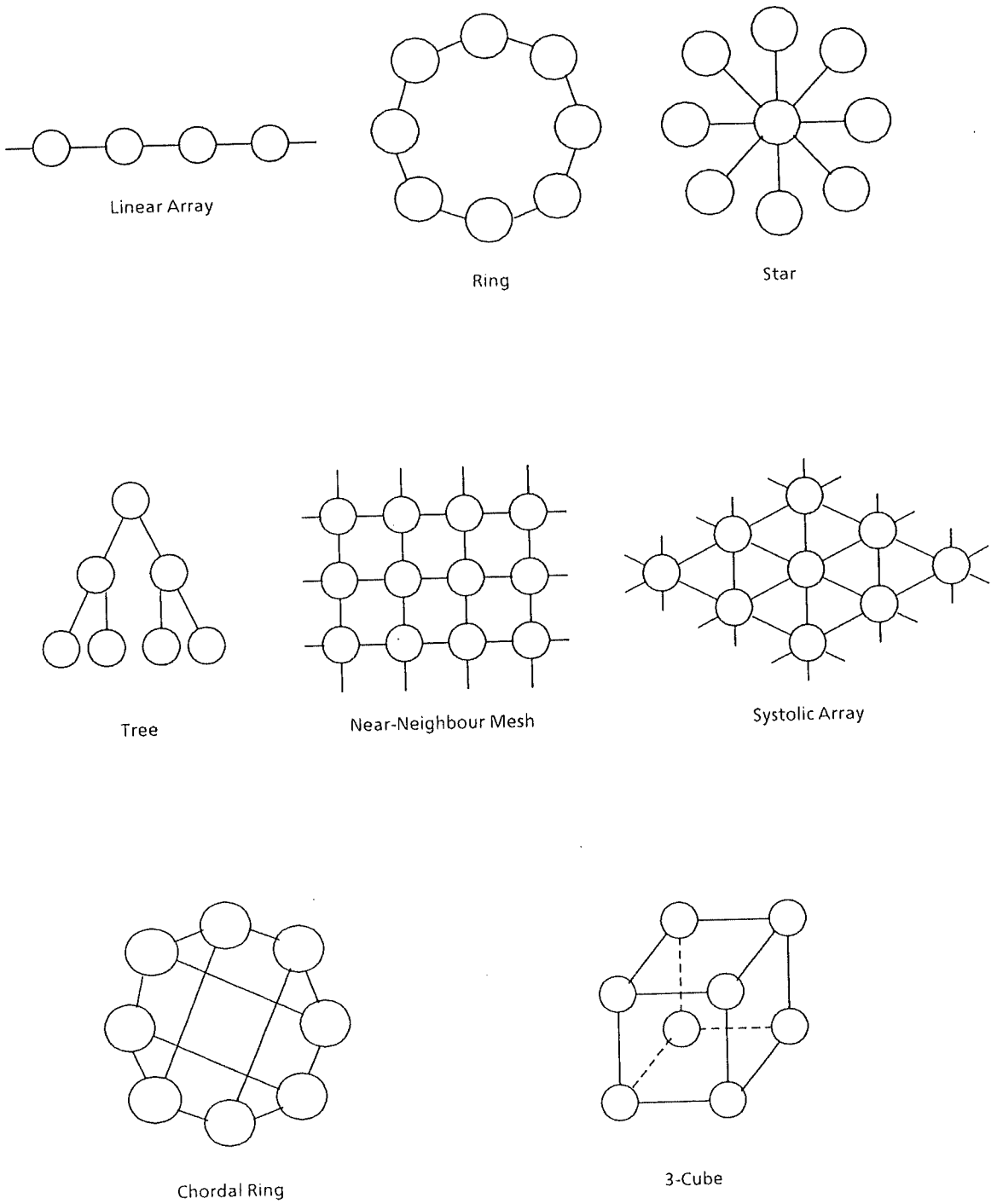


Figure 3.1: Some Static Interconnection Structures

transferred between two units of the processor which are not directly connected. The topologies can be loosely divided into the following groups, linear arrays, rings, stars, trees, near-neighbour mesh, systolic arrays, chordal rings, multidimensional cubes and fully connected systems.

In the following two sections are examples of processors constructed using static interconnection networks.

DAP

A well known example of a statically connected multiprocessor is the DAP [40]. This processor has a 64×64 array of processing elements connected in a square grid to produce a near-neighbour mesh. Algorithms that require operations on adjacent elements, such as linear differential equations in a matrix are simply implemented, but any other combinations require the data to be moved about, consuming many processor cycles, before the operations can be performed. The unique feature of the DAP is that each processing element only works on a single bit of data at a time, instead of the more normal structures which manipulate whole words. This reduces the amount of logic required to implement the network as other systems require many parallel connections.

The DAP is an example of the more general systolic array structures[23] which cover many different shapes in a 2-dimensional plane. Different structures can be chosen to suit the problem to be solved, but the networks are normally rigid and the processor is restricted to one class of problems.

Cosmic Cube Computer

By constructing an interconnection network in three dimensions a highly interconnected structure can be achieved, where the cost of the movement of data between two processors that are not directly connected only grows logarithmically with the size of the network. These network are generalised under Chordal Rings[2]. Classes of these networks modeled on a cube in the dimensions from 3 upwards have processors at the vertices and with communications channels running along the edges. These networks are called Hypercubes have been proposed as Indirect Binary n-Cubes[37].

The *Cosmic Cube Computer*[42] is a Hypercube multiprocessor based on the interconnection of 64 small self contained processors, with an interconnection structure based on a cube in 6 dimensions. As each processor has only six connections to other processors it is necessary for a packet to be routed through an average of

three nodes before it reaches its destination. A packet passing to the topologically furthest node, the node diagonally opposite in 6 dimensions, will need to pass through, at most, 5 nodes on the way.

Programs that run on the Cosmic Cube are broken up into small segments to be run on separate processors and interact by passing messages over the network. Programs need to be analysed to determine the flow of data between each of the portions of the program to avoid developing bottlenecks where many packets need to pass through a few nodes. Such an analysis would cluster such a group to ensure that the number of links involved would be kept to a minimum. Also, as there are many possible paths through the network which packets may follow, it may be more efficient to scatter packets amongst these routes to spread the communications load across a number of links.

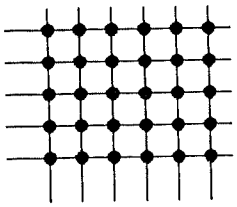
3.2 Dynamic Interconnection Networks

Dynamic Interconnection structures are very dependent on the requirements for the multi-processor, but in general these networks are constructed of stages of switching elements connected together with a permutation network. The paths through the network are established by the setting of switches to particular values, and the permutations are such that paths between all elements are possible though not necessarily all at once. Some examples can be found in Figure 3.2.

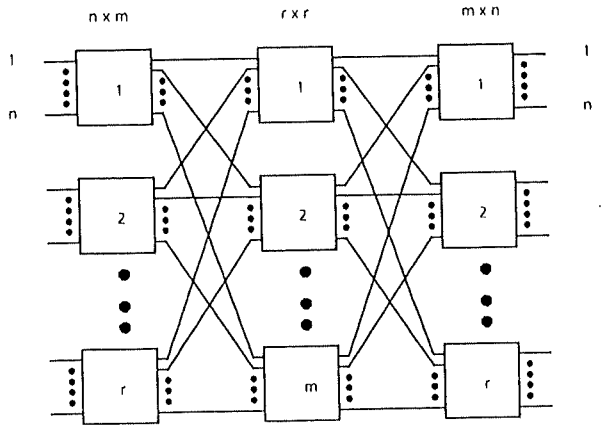
This arrangement has resulted from the work done on telephone switching by Closs[8] and Beneš[4] in the late 1950s and early 1960s. It was shown that by using a number of smaller cross-bar switching stages whose connections are permuted, switching systems could be constructed where all connections could be established when required, either immediately or by re-arranging the other connections. These systems were desirable as they required fewer switching elements than a cross-bar capable of making the same number of connections. Consequently the systems are called *Non-Blocking* and *Re-arrangable*. A third possible arrangement, called a *Blocking* system, is also possible, which requires even less components but at the expense of not being able to establish all required paths simultaneously.

Dynamic Interconnection Networks can be divided into two groups, the globally controlled networks and the locally controlled networks.

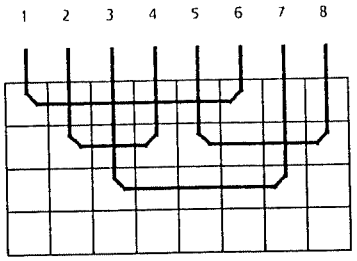
In the networks with global control a central control unit sets up the paths through the network to some predetermined pattern. Once set up, the processing elements can then pass data without any routing information. When a particular



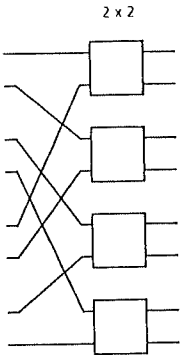
N X M Crossbar



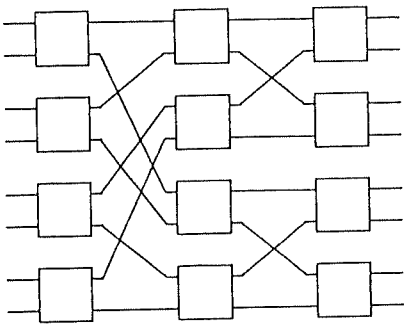
Closs Network



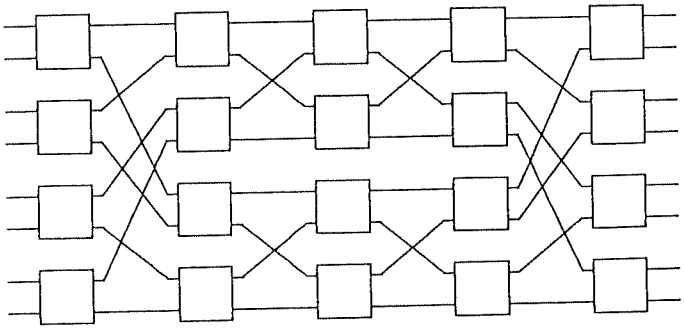
One-Sided Cellular



8 X 8 Shuffle Exchange



8 X 8 Baseline



8 X 8 Benes

Figure 3.2: Some Dynamic interconnection Structures

phase of the program has been completed the central control unit can then set up another connection pattern. An example of this form of interconnection network is the Data Manipulator as proposed by Feng[13]. This network takes data from memory units and rearranges it so that it is in the correct form to be passed to the arithmetic units. Results from the ALUs are then passed through a similar unit before being returned to the memory. The Data Manipulator can perform rotation, shifting and masking operations on the data, as determined by the settings in the central control unit. These operations make it possible to access the data in a number of different ways, for example a matrix may be accessed as a number of rows for one cycle of a program and then accessed as a number of columns for the next. Similarly sub-fields of words can be extracted and masked before being operated on.

In a local control network the switching structures are controlled by the packets that are passing through them, in a similar fashion to a packet switching network. Packets carry information about their destination and this information is used to control the setting of switches as the packet passes through a particular stage. The topologies used to construct these networks are normally very regular which allows their operation to be fair and simple. There are many topologies used for these networks and some detailed examples are given in the next sections.

3.3 Single Stage Shuffle Exchange Networks

Single Stage Shuffle Exchange networks use a single sorting stage and a permutation network to sort packets so that they are delivered to the correct station. The sorting stage consists of a small unit capable of sorting all input packets. Packets are routed through the sorting unit by examining a portion of the packet route field which selects the desired output. If the desired output is already in use then the packet is considered blocked. The permutation network distributes the packets back to the sorting nodes for further sorting.

Packets are delivered to the buffer stage for transmission to another station. They then pass through the sorting and shuffle stages until they have reached the correct destination. Each packet is marked with a destination address and a pass counter. One bit of the destination address is used for the sorting stage at each pass. This bit is selected by the value in the pass counter, which is incremented as the packet passes through the sorting stage. Figure 3.3 shows the layout of a Single Stage Shuffle Exchange Network. It is constructed from the simplest form

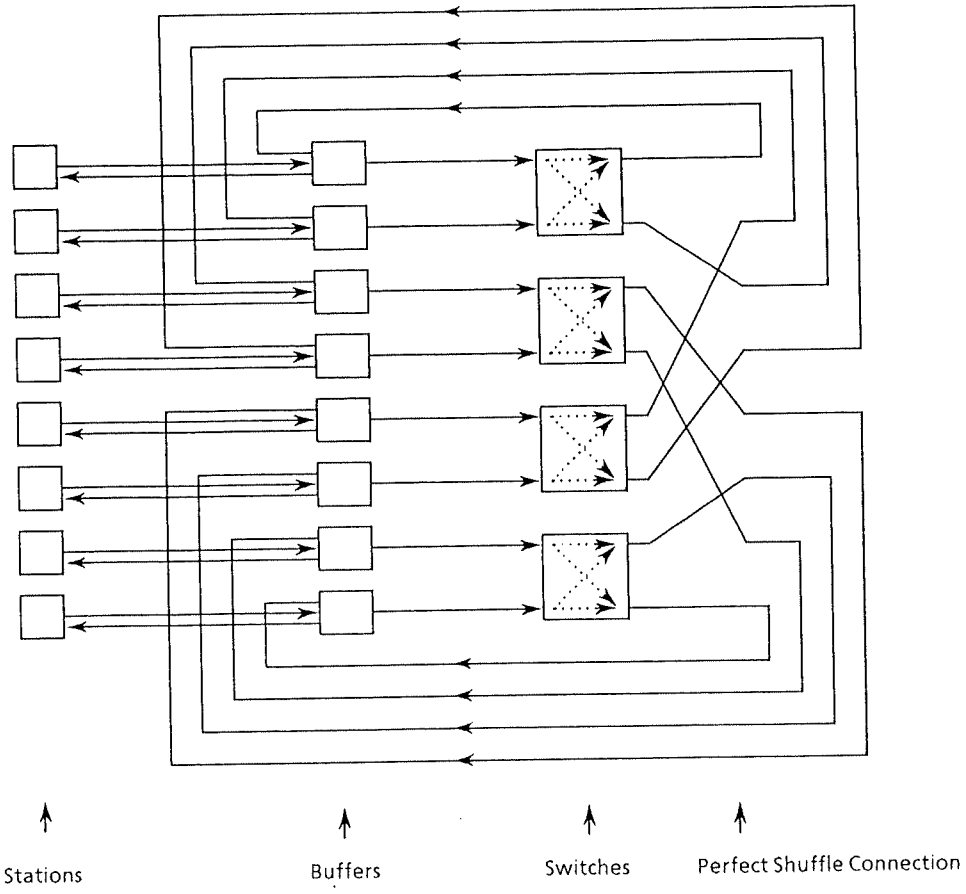


Figure 3.3: Single Stage Shuffle Exchange Network

of sorting node, the 2×2 node.

Packets for transmission are passed to the buffer stage connected to the network. The packets are then passed to the sorting stage where the required output is selected. If free, the packet is sent back around to the buffer stage connect via the shuffle stage with its pass count incremented. Blocking of the packets may occur in two ways; firstly the requested output may have been selected to the other packet connected to the sorting stage. In this case the packet remains in the buffer. Secondly, the buffer at the other end of the shuffle network may not be able to accept the packet and the packet must again be retained. Unless the nodes are carefully designed it is possible for a buffer dead-lock to occur.

A packet remains in the network for as many cycles as necessary for it to reach the buffer connected to the desired station. This may be between 1 and $\log N$ cycles. When the packet reaches its destination it is removed from the buffer and passed to the station's receiver.

It is also possible to design a node which contain buffers which are incapable of dead-locking [33]. They have a third form of collision management which they can apply when two packets require the same output for the next stage of the network. This scheme involves selecting one of the packets to pass out on the requested link,

while the other packet is sent out on the other link. The first packet continues on its journey, moving on to the next stage of its passage while the other packet has its stage counter (which indicates how many stages it has passed through and which is used to select the correct route bit), reset to zero. The resetting of the stage counter has the effect of re-injecting the packet into the network as though a new request was being made, and works because all inputs to the network are indistinguishable.

Techniques to improve the performance of Single Stage Shuffle Exchange network have been proposed by Kumar et al.[21] and Kumar[22]. In these networks the buffer stage is divided up into a number of separate buffers. Packets are routed into the buffers, which are selected by the number of times a packet has cycled through the network. When the stage is preparing to start the next phase of the sorting the packet that has been processed the most is selected, thereby ensuring that packets which have been in the network the longest are processed first.

3.4 Multi-Stage Shuffle Exchange Networks

By taking the Single Stage Shuffle exchange network and connecting a number of them in series a Multi-Stage Shuffle Exchange network is constructed. These structures are called Delta Networks[34]. These networks operate in the same way as the single stage networks described in the previous section but are capable of handling many more packets, as the sorting is carried out in parallel.

Delta networks can be constructed from the switching nodes in a number of different ways. In general there are $\log N$ sorting stages connected together by permutation networks, of which many exist. The general structure of a Delta network is given in Figure 3.4. The structures of two particular forms, the Omega network[24] and the network I call the Perfect Shuffle, are given in Figure 3.5.

These networks can be divided into buffered and unbuffered categories depending on whether the sorting nodes are capable of holding packets in internal buffers when a blockage occurs, or not.

In the unbuffered networks, packets to be transmitted through the network are presented at the inputs to the network at the beginning of a time period. The packets are presented to the switching units in groups of 2 and the switching units will attempt to route the packets to their desired output. If two packets require the same output then one will be selected and the other will be discarded. The selected packets will be passed on to the next stage of the network while a signal

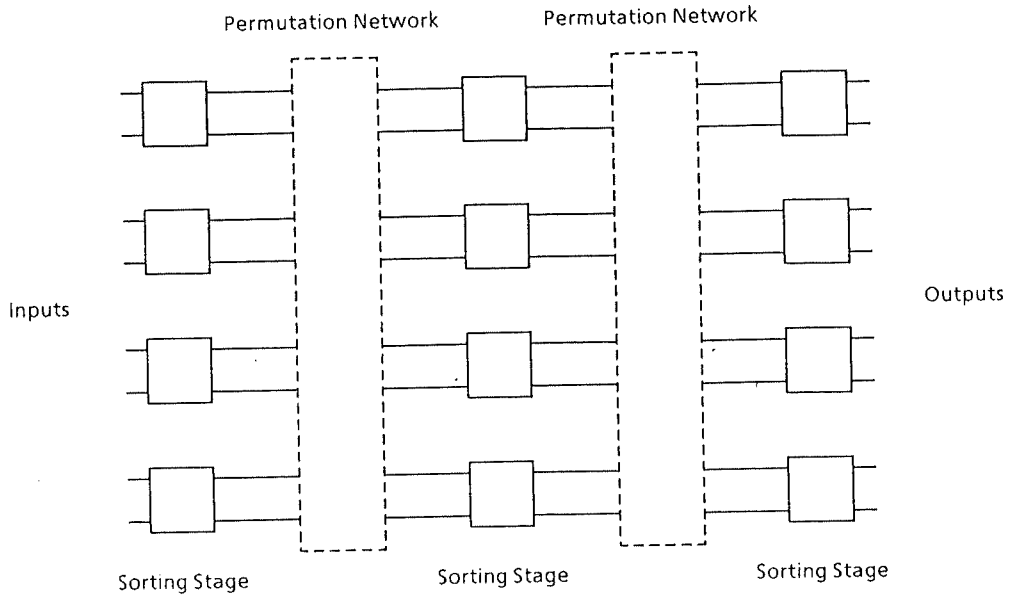


Figure 3.4: General Delta Network Structure

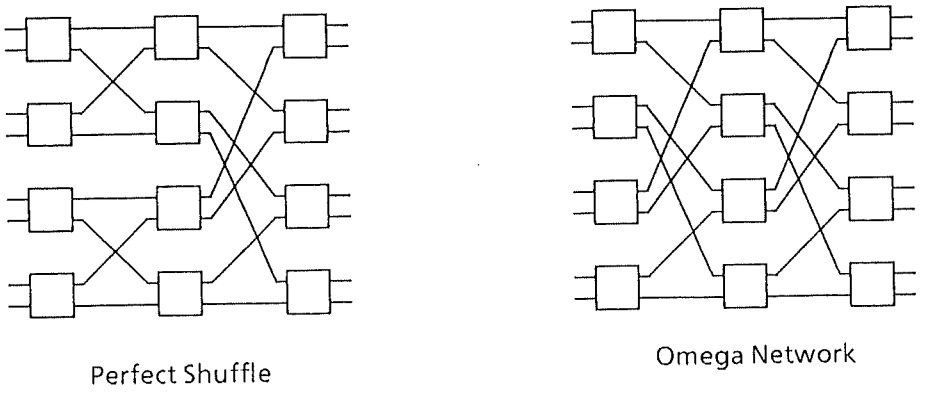


Figure 3.5: Alternative Network Structures

will be passed back for all those packets that have been discarded. The packets progress through the network passing through each stage in turn. When they have passed through all stages of the network they will have arrived at their desired destination. Packets that are blocked are retried at the next time period. An alternative solution is for the requesting station to know how long it takes for the reply to return. If a reply does not arrive within this time period then the station can assume that the packet was blocked and that it should be retried. Both these techniques have the effect that in general the service time is not predictable and that if the network is heavily loaded it may become infinite.

In the buffered networks, packets are clocked from the buffer of one node to the buffer of the next. If two packets require to leave by the same output one is selected and passed on and the other is held in the buffer until the next clock cycle. The packets can only move through the network if the buffer in the next stage is empty or will become empty. Thus a signal is required to be passed back through the network to indicate if the packets are to be held. This form of network will not dead-lock as packets will always be able to move from the network to the receivers, allowing other packets to move in the network.

3.5 Distributed Circuit Switching

A communication sub-network, Starnet[49] based on *Distributed Circuit Switching* has been proposed for use in multiprocessor systems. It is based on the Baseline network structure of the multistage interconnection networks to provide the facilities of full connectivity while keeping the network cost-effective. It differs from the other interconnection networks by establishing a path through the network before any data is transferred. Once the path is established the source can hold the connection for any period of time to transfer the required data.

The network consists of a set of switching nodes connected in a Baseline structure. Each link between the nodes consists of a number of control signals; request, acknowledge, release and direction, along with a bi-directional data bus. The data bus is as wide as required for the application. As well as these signals the node is provided with some general control signals; reset, priority and tag. Figure 3.6 shows the structure of the switching nodes.

Operation of the network proceeds in two phases, the path establishment phase and the data transfer phase, controlled by two clock phases. During phase $\emptyset_1$ stations wishing to establish a path place the required address on the data lines of

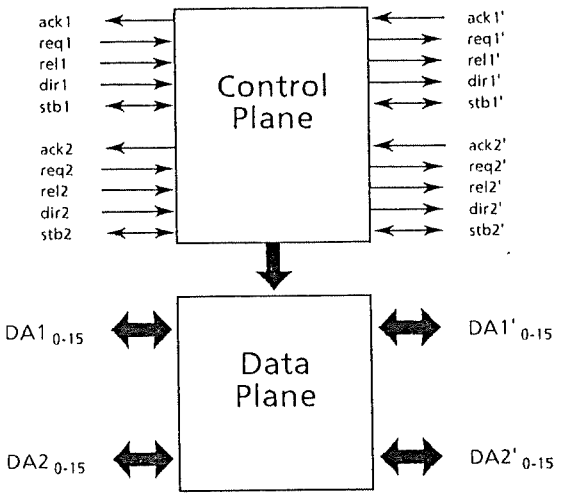


Figure 3.6: Starnet Switching Node

their link and request the path to be established. This request signal will ripple through nodes while there is no conflict, until it is either blocked or reaches the receiver. Those stations whose paths are set up will receive an acknowledgement and, during phase $\emptyset_2$, all the nodes in the established paths will set their control registers to set up the related connections in the data plane. At the end of phase $\emptyset_2$ a physical path is established between the two stations under the control of originating station.

Once the data path has been established, data can be transferred in either direction under the control of the active interface. Higher level protocols can thus be implemented using the data path as a reliable communications channel.

An implementation of this structure has been proposed[28]. The design is estimated to take 33ns for signals to propagate through one node. This would allow approximately 15 nodes to be passed within one microsecond with an allowance for acknowledgement time, which is a reasonable time period for a microprocessor system. The device would need to be constructed on a package with 88 pins to allow a half-duplex channel of 16 bits.

A similar structure is implemented in the Numerical Aerodynamic Simulator[3]. In the multiprocessor, 512 processors are connected to 521 memory units. The

prime number of memory units is used to reduce memory access conflicts by effectively randomizing allocation of data elements such as arrays.

3.6 Summary

In this chapter examples of the structures for interconnection networks used in multiprocessor systems have been examined.

These networks can be divided into 2 groups, static and dynamic. In the static networks, data follows fixed paths as it passes between processors and memory. These paths are fixed by the hardware configuration and cannot be changed and packets may require the intervention of intermediate processors to reach their destination. In dynamic structures, data follows a path through a switching network. Packets are routed directly from source to destination by the network. The path may either be set up by a global control processor or it may be set up dynamically as the packet passes through the network.

The operation of a class of interconnection networks, the Shuffle Exchange networks, is of interest as the structure and procedures that are contained in it can also be applied to the Local Area Network environment. Structures such as the Perfect Shuffle and the Delta network allow data to be routed through the network at high rates using multiple channels running in parallel. They also provide simple network structures that allow uniform addressing across the whole network.

Chapter 4

Binary Routing Networks

Binary Routing Networks[16] apply techniques developed for multiprocessor systems to the local area environment. They produce a Local Area Network which has features from interconnection, local and global networks. From interconnection networks the binary switch and the interconnection structures are used to give the network multiple paths for improved throughput. Direct local or global addressing (routing) eliminates the requirements for bridges when connecting self contained networks.

Binary Routing Network consists sets of two components. These components are the *Terminus*, which is the interface between the host machines and the network, and the *Routing Nodes*, which route the packets to the desired destination. To help in flow control the routing nodes contain buffers and are connected with *buffer full* signals to hold up the flow of packets. The routing nodes are connected in the form of a directed graph and packets traverse the network by selecting path elements according to the route specified in the header of the packet.

The packet consists of a number of fields, a start of packet indication, a route field and a data field. There may be other fields included in the packet, for example special packet indication, parity and checksum to help in avoiding errors.

Packets are injected into the network from the transmitters and pass through a number of nodes until they eventually come to a receiver. If a packet cannot proceed because the required output is in use, then the packet is considered blocked and is saved up in the internal buffer within the node to be transmitted later when the output becomes free. If the buffers are of limited capacity then when they become full a signal, called *Backward Busy*, is sent to the node or station connected to the input, to indicate that no more packets can be accepted.

For the purposes of normal operation, a load well below the maximum throughput of the network, a buffer could be considered infinite if it could accept a large

number of packets. Provided that the buffer never overflowed the network could be constructed without a busy signal. A buffer overflow could be considered a network error requiring software to recover from the loss.

4.1 The Routing Node

The most fundamental part of the whole network is the Routing Node. These nodes are connected together in large numbers to provide communications between all stations on the network. These nodes require to be simple and not to involve complex circuitry or state information which would make the network expensive and slow. They need to function with a small set of operations to make the design simple and their operation efficient.

The basic operations of a routing node are to:

- Route packets from inputs to outputs.
- Modify the route to remove to old routing information from the front of the packet.
- Arbitrate when more than one packet requires a single output.
- Provide flow control when the network is congested.

The simplest symmetric form of the node is the *Binary* node with two inputs and two outputs, shown in Figure 4.1. The use of binary nodes simplifies the routing procedure because only a single bit is necessary to determine which output is to be followed. Packets may enter by either of the two inputs and leave by either of the two outputs, with the possibility of two packets being simultaneously processed by the node provided they wish to leave by different outputs.

It is possible to build nodes based on other units, but unless it is a power of 2 then some of the routing information would be wasted. The 2×2 node is the simplest to construct but for an implementation on an integrated circuit it might be more practical to design a 4×4 node or an 8×8 node which selected the output by using 2 or 3 bits of the route respectively. These would make better use of the pins of the integrated circuit and the printed circuit board space required to connect them into a large system.

The node routes the packet by examining the first bit, or bits, of the route field and uses the value to select the output. If the selected output is free then the packet can start to pass on to the next node without any additional delay, otherwise the packet will be blocked. This form of routing is known as *cut-through*[18].

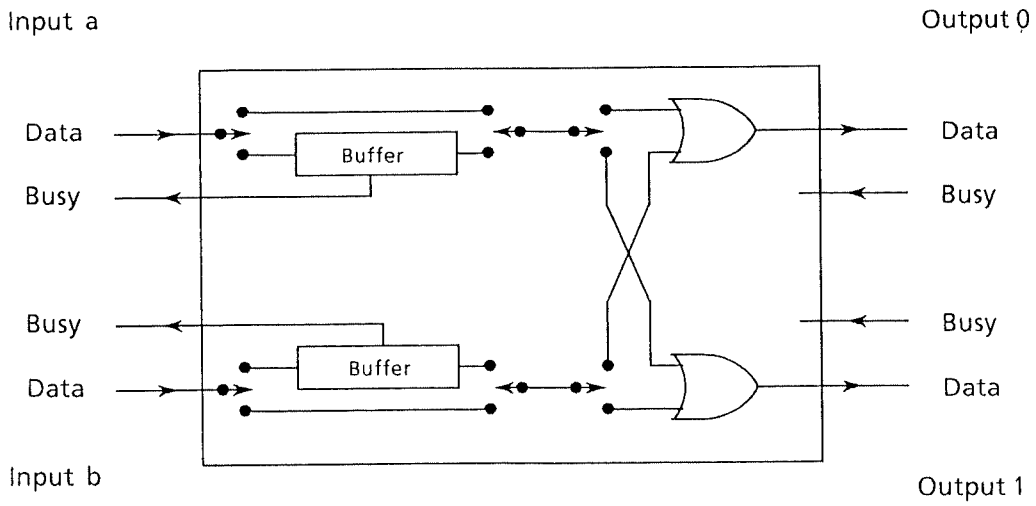


Figure 4.1: Binary Routing Node

As the packet moves on to the next node, the route field must be modified so that the first bit of the route field of the outgoing packet is the bit required for the next node to do its own selection. With each node applying this transformation to the packet all nodes can be constructed alike. If the route field was not modified then each routing node would be required to look at a different bit of the route field, requiring extra hardware to implement the node and additional management of the network when changes to the layout were to be made. The modification can come in a number of forms which either discard the used route bit or remember it in a different part of the packet.

It is possible to break down the functions of the routing node into two smaller components. These are the Fork and Join functions. The Fork function takes a packet from its input and routes it to one of two outputs, performing the route manipulation operations, while the Join function takes packets from its two inputs and passes them to its output, performing collision resolution. The 2×2 routing node can be constructed from 2 of each of the simpler functions, while alternate structures such as the 3-way node can also be constructed as depicted in Figure 4.2.

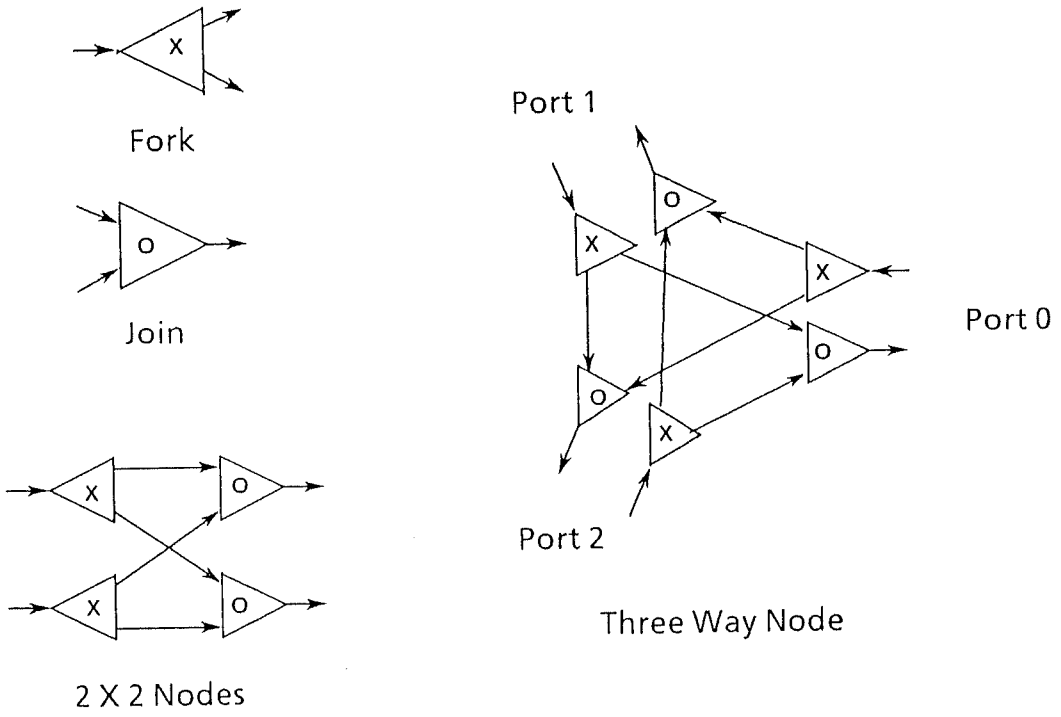


Figure 4.2: Routing Nodes constructed from Fork and Join Units

4.2 Network Topologies

Binary Routing networks are constructed as directed graphs. There are thus many different topologies that can be used to form the interconnection structure of the network. In general a regular structure would be preferable to an irregular structure for the simple reasons of management and of ensuring that modification of the network does not involve large changes to all components. Similarly the different structures will affect the way routes are generated. If the network is regular then the routes may be mapped directly from the addresses of the sending station and the receiving station. For an irregular network the route may need to be looked up in a table. This would require a Name Server or Route Server to maintain a map of the network and to calculate the relative route between two stations.

The simplest structure which can be used, and the one which uses the least number of routing nodes is the Ring. Figure 4.3 shows a Binary Routing Network connected as a ring. One pair of inputs and outputs is connected to the station while the other pair is connected into the ring. Packets injected into the ring would have their route set to follow the ring path until the packet had reached the

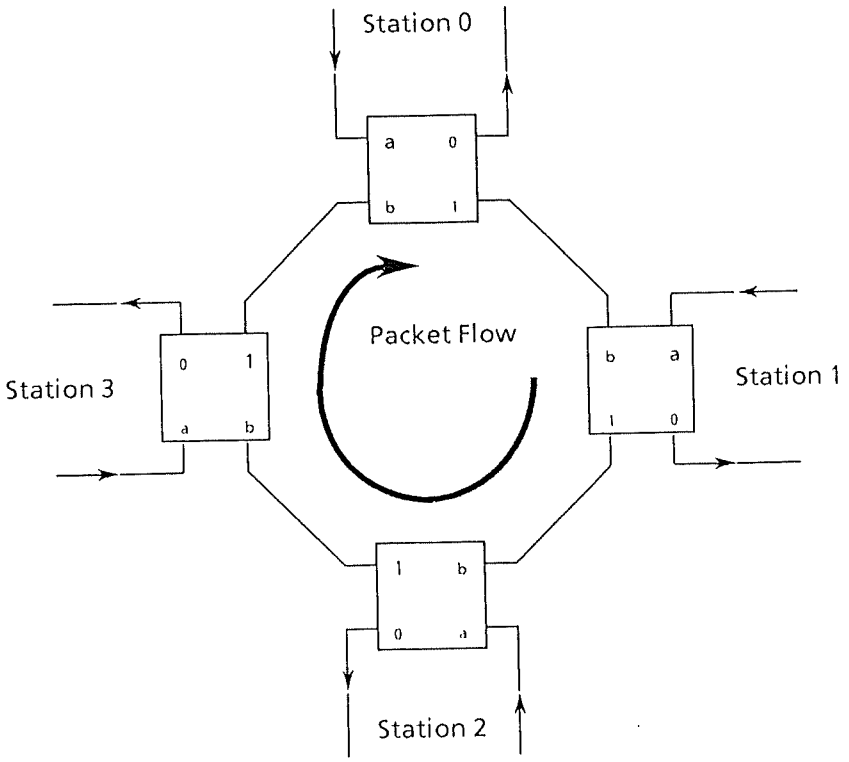


Figure 4.3: BRN Connected as Ring

desired node, where it would select the other path and enter the receiver of the station. It can be seen that this form of network would allow a number of packets to be in transit and, as the packet is removed from the network when it reaches the receiver, that the useful bandwidth would be higher than the Cambridge Ring and similar to that of the IBM Token Ring. One of the problems with this structure is that to send a packet to a particular station the position of both stations in relation to one another needs to be known. It can also be seen that as the size of the network grows the number of routing bits grows proportionally. A second problem of the ring structure is the possibility of deadlock. If the buffer in one node cannot be emptied until the buffer in the next node is emptied then the situation of all buffers being full can arise. A third problem is that a bit error in the route field of the packet could easily lead to a packet circulating forever.

A desirable feature of a network is to make the transmission of packets more efficient the nearer two stations are in the network. By using a structure that is capable of local routing (addressing), a group of processors can communicate without interfering with each other. This can be achieved using a tree or triangular structure [11], depicted in Figure 4.4. In this network small identical subnetworks are connected together to generate a larger network. Communications between

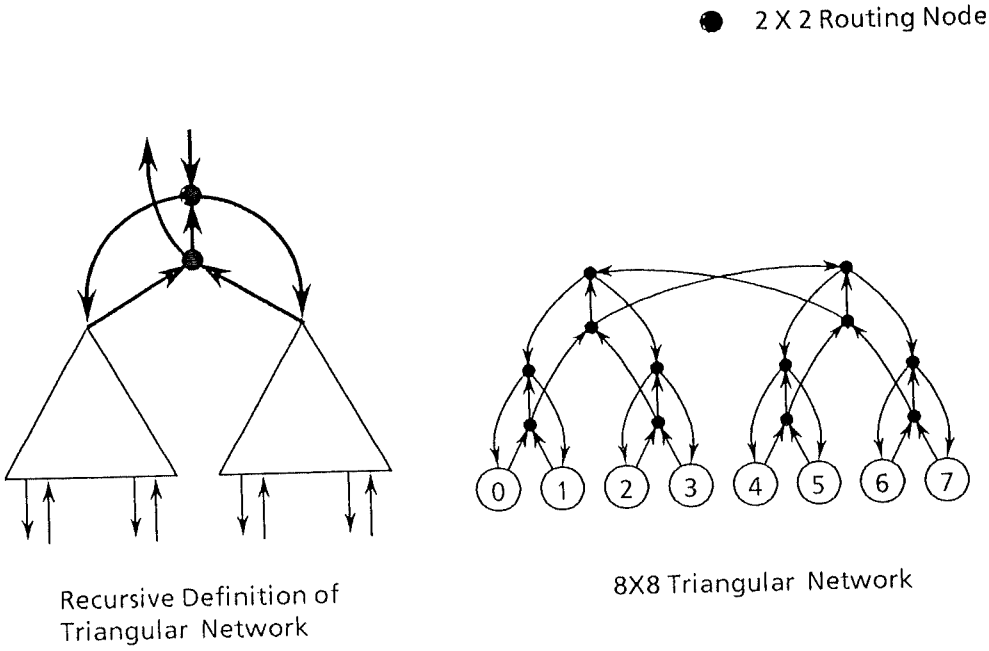


Figure 4.4: A Triangular Network Structure

stations within a subnetwork is wholly contained within that network and packets need only leave the subnetwork to pass into other parts of the greater network structure. The main disadvantage with this structure is that the number of address bits to cover the whole network grows linearly with the number of stations. It does have the advantage that routes can be determined from the relative source and destination addresses.

Structures that have received a lot of attention in research on multiprocessor systems are the Perfect Shuffle and other topologically similar interconnection networks. The major property of these structures is that the number of nodes a packet passes through is $\log_2 N$ where N is the number of stations in the network. The second property of these structures is that the route is now also the address of the destination station. That is, packets with the same route field set will arrive at the one receiver no matter which transmitter they were sent from. The Perfect Shuffle structure is shown in Figure 4.5. Topologically similar structures are the *Delta* and *Baseline* networks, whose similarity has been shown by Lawrie[24].

From Figure 4.5 it can be seen that to send a packet to a particular receiver the route field can be generated by taking the binary representation of the destination address and placing the most significant bit of the address in route bit 0, the

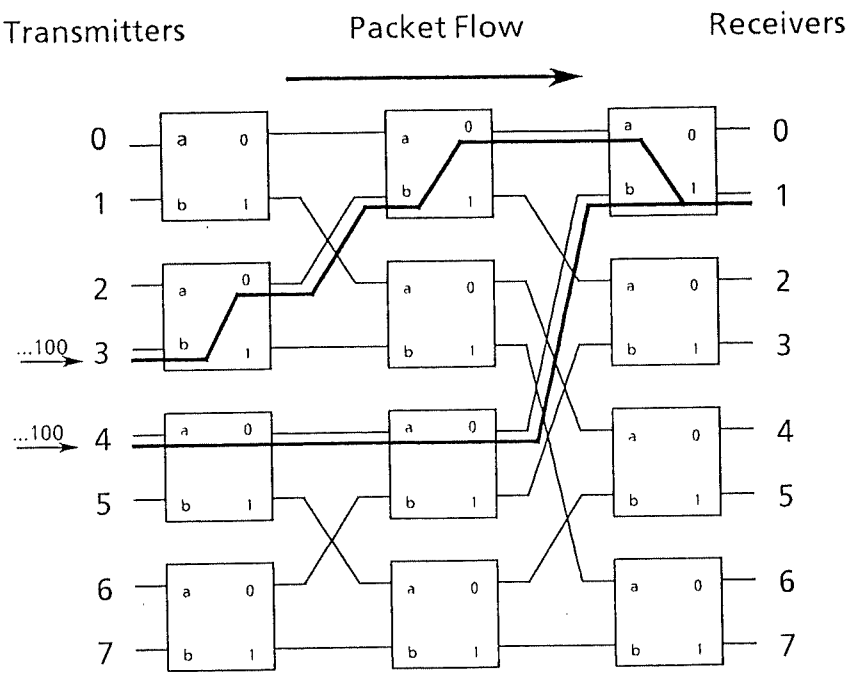


Figure 4.5: Perfect Shuffle Network

second most significant bit in route bit 1, and so on. This figure shows the route taken to station 1 from stations 3 and 4.

In all the networks mentioned above the flow of data has always been in one direction. By constructing a three way node it is possible to build a bi-directional network. A packet arriving at one of the three ports of a node will be directed to one of the remaining two ports. By specifying that a zero bit means go to the left and a one bit means go to the right, a reverse path can be constructed by reversing and complementing the original route field. Figure 4.6 shows the basic three way node and a possible network. This structure uses more routing bits than the Perfect Shuffle but as a reverse path can be easily established, it is useful for both regular and irregular structures.

4.3 Flow Control

In a network where the progress of a packet can be impeded by other packets, which are themselves traversing the network, some form of flow control is required unless packets are to be discarded and throughput severely limited.

This requires that there be some form of control signal passed backwards

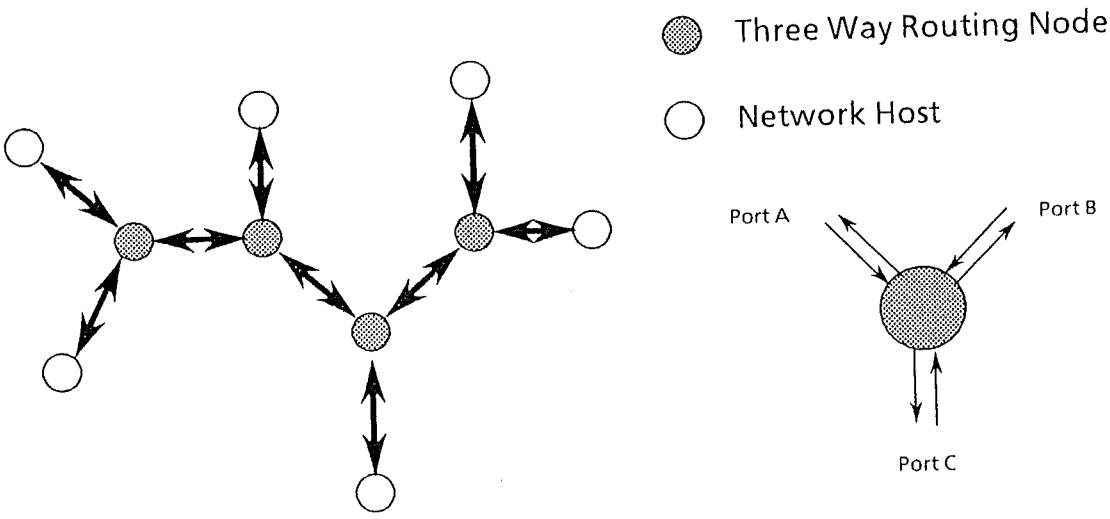


Figure 4.6: BRN Constructed from Three Way Nodes

through the network to inform earlier nodes and transmitters that their transmission must be stopped. In a structure where there is a one way flow of packets, such as the Perfect Shuffle network, there must be physical wire to carry this signal. It is possible in networks constructed of 3-way nodes for the busy signal to be carried on the reverse path, as a control bit sent every N data bits. The first method is the simplest to implement but at an extra cost of having two wires. The second method requires much more logic and slightly reduces the bandwidth of the link, as well as requiring the link to be continuously passing data bits to enable the control bits to reach the other end, even when there is no packet to send. On a network with all clocks synchronised this would not present a difficult problem, but where asynchronous clocking is used the link would require a local clock to deliver the control information when a packet was not passing.

There are a number of design issues that must be taken into account when designing the buffering system and flow control of the network. While a packet is passing through a node the output rate must be determined by the input rate or there will be problems with the difference in the clock rates. Either the packet will require some buffering in the node if the input is higher than the output, or there will be a need to stretch the packet if the output is faster than the input. In the

first case a dual ported memory running at the network speed would be required. For the second case a mechanism to add padding to the packet and remove it at the receiver would be required. Alternatively the clock rate of the output link is synchronized to that of the input link while a packet is passing. If the packet must be buffered, then it is completely buffered and later transmitted at a rate set by a local oscillator in the node. This method is by far the simplest but will have an effect on the maximum throughput and delay of the network. It is this third method which is adopted for the further study of the buffered Binary Routing Networks.

The question of packet size must also be dealt with. If the buffer is capable of accepting more than one packet, there must always be room to fit in a single packet of the maximum length. If the packet size is fixed there is no real problem, but for variable length packets the amount of buffer space that must remain could possibly take a number of smaller packets. By allowing the nodes to discard some packets when the buffers overflow and using higher level protocols to recover them the buffer's capacity can be fully exploited by smaller packets but with a side effect of less reliable larger packets.

In the next sections some of the possible solutions that have been considered for construction are outlined. These proposals are examined in the chapter on network performance and are only a few of the many possible designs.

Single Packet Buffer Routing Node

The simplest form of buffered routing node has a single buffer for each of the input lines. While a packet is being buffered the Busy signal is passed back to allow enough time to stop the transmission of the next packet. The signal is then held until the buffer is capable of accepting another packet.

This form of node is the simplest to construct. The buffer memory needs to be single ported only, as either a packet is being received or it is being transmitted but not both simultaneously.

Multiple Packet Buffer Routing Node

The addition of many buffers will improve the performance of the network as the flow of packets through the network will smooth out. By making the buffer capable of holding more than one packet the logic to implement the node is made more complex as now the buffer needs to be dual ported. The operation of the busy signal remains the same, but only for the last packet slot in the buffer.

Alternatively this buffer could be made large enough so that it can be considered infinite. For normal operation the node would pass packets as normal until the output is blocked when it would store the packet. While the packet is waiting to be delivered further packets entering the node on the same input would also be buffered. Given that the buffer is not infinite but very large, and that the normal operating conditions of the network would not cause packet queues to build up at a particular node, no *Backward Busy* signal would be required. It is only when some network error occurs that there is any likelihood of the buffers filling up. If this occurs the node is then entitled to discard packets, and only then is there any requirement for signalling backward through the network to indicate that some major error has occurred and, where possible, where the error may be located.

Split Buffer Routing Node

When multiple packet buffers are employed in the routing node, packets may be blocked that are destined for a free output of the node. This results in reducing the effective throughput for the node, as available bandwidth is being wasted. To overcome this it would be necessary for the node logic to look through the buffers for packets that can be sent, increasing the logic of the buffers as they would no longer be FIFOs. Alternatively the packets coming in from a particular node can be separated out and placed in one of two buffers depending on the output desired. Each input would have its own set of buffers to avoid collisions. The node outputs would then select packets from the respective buffers, selection either being sequential or random. This arrangement, shown in Figure 4.7, always allows packets to move if there is an output free. It is not possible to use a single packet buffer as each packet is capable of moving into one of two buffers. The backward busy signal would need to be asserted if either buffer was filled, thus negating any improvement gained from the modification. This system could be considered as a 2×2 node constructed from fork and join primitives with buffers at the inputs to the join units.

4.4 Buffer Control

The operation of a Binary Routing Network needs to take into account the possibility of stations not accepting packets because they are either powered off or they have failed. If this condition occurs then packets waiting in node buffers can block the passage of other packets. The management of buffers is required to

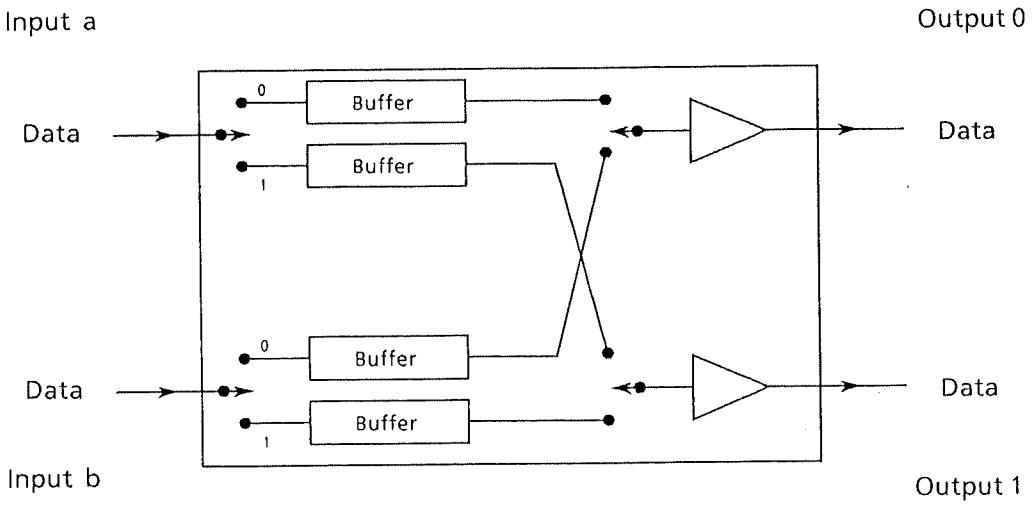


Figure 4.7: Routing Node with Split Buffer Scheme

complement the operation of the network flow control.

If a station is powered off, a signal from the receiver to the node can be used to indicate the condition, and the node can discard any packets destined for the station.

If the station fails in either the hardware or software, such a signal can only be generated by the use of some form of "aliveness" timer. This timer would indicate that the system has not performed some operation within a specified time, and generate the signal allowing the node to discard packets. By placing such a time limit on all buffers in the network, any internal node that is blocked by a failure will also be able to function, albeit with a reduced capacity.

The method discussed in the last section on flow control, where packets are discarded when the buffer is full, can also be used to solve this problem. By discarding the packet at the front of the buffer only newer packets will remain in the network, and these will be slowly removed as even newer packets are injected. To use this form of control the designer will need to determine the load on the stations to ensure that, where there can be periods of time when the station is busy, the buffer is capable of accepting enough packets to avoid an overrun.

The simulation results in Chapter 6 indicate that for loads ranging between

0% and 25% per station on the network there is very little difference between the buffering methods. In fact, for the network of 512 stations operating at 25% load the buffers were never observed to have more than 6 packets with the average queue length 0.117 packets.

4.5 Packet Format

The basic structure for a packet in a Binary Routing Network is fixed. After the packet start bit comes the route field which is followed by the data field. This structure is required so that the delay through the routing node is small by enabling the decision on where the packet is going to be made early, before any large portion of the packet has been buffered in the node.

The design of the packet is an important part of the design of the routing nodes and to a lesser extent of the transmitter and the receiver. Important issues that influence the design of components in the network are "Does the packet have fixed or variable field size" and "Where is the old route bit placed"?

If the packet is to be constructed with fixed fields, which makes both the node and the station simple, then the network as a whole will be limited. A fixed route will limit the number of nodes a packet can pass through before it is required to enter the destination receiver. Even if the route field is made large, for example upto 48 bits, the structure of the network may be such that the packets pass through numbers of clusters of switching nodes, and then if there is a short path for local transmissions requiring 8 bits of route, then 40 bits will be wasted. The advantage of a fixed field packet is that the nodes will know when a packet has completely passed, and the receiver will know where to extract the data from the packet.

If a variable length field is to be used then there needs to be some form of marker that is distinguishable from the normal route and data bits. It would be unsuitable to have the length of the fields in the header as this would cause delay while it was received. For example to have a route of 48 bits a 6 bit field count is required. A packet would be delayed an extra 6 bits per node. After passing through 8 nodes the extra delay would total 48 and is already over the extra delay for the equivalent fixed field network. If the packet passed through 48 nodes the extra delay would be 288 bits. A length field preceding the data portion of the packet would not significantly add to the delay in transmitting the packet. The only extra delay would be in transmitting the length.

To have a marker some form of encoding violation would be required, possibly a missing transition in a code requiring a least one transition per bit. Such a system would require complex detection circuitry to avoid small phase errors being detected as markers.

If there is no marker available then the field of the packet could be variable in groups of bits. For each group of bits there would be a tag bit which would indicate if the particular field is to be continued in the next group or not. For the route field this grouping would be small, say 8 bits, as larger lumps would again be wasteful in small systems where the size had just passed the limit of a group. For the data field of the packets the grouping could be in terms of many bytes, say 16 or 32 bytes. This system introduces extra bits of packet structure and thus reduces the real bandwidth of the network but, in terms of saving bits transmitted, could actually give an increase in performance.

If the above method is taken to the logical limit within the route field, where each group is reduced to a single bit followed by a tag bit to indicate if there is any more of the field to come, the route field could be considered to come in groups of digits in base 4. From this space of 4 possibilities, 2 are required to indicate the direction the packet is to be routed, one can be used to indicate the logical end of the route field and the last, the physical end of the route field. The separation of logical and physical ends of the route field is useful if the route field is to be saved by rotating the used bits to the end of the field. The logical end of the field can be used to indicate when the packet has run out of field bits in the case of an error, allowing the packet to be discarded, and the physical end of route field allows the receiver to separate the route field from the data field. Following the physical end of the route field the data field can start, and this can be framed in any of the previously mentioned ways.

To implement the above two methods extra bits of delay are required at each node. In the first method rotating a field consisting of more than 1 group requires the first bit of the next group to be known 2 bits in advance, so that it can be moved to the end of the earlier group. In the second method 2 bits must be received before any decisions can be made.

4.6 Packet Length

The maximum length of a packet is an important factor in the design of a Binary Routing Network node. This length determines the amount of buffering that must

be supplied and thus the cost of constructing a node.

For a packet with a maximum length of 2k bytes, which is not an uncommon size in Local Area Networks, each node would require a minimum of 4Kbytes of buffer space when just considering the single buffer node. If a node with a limit of 4 packets per buffer were to be implemented then this would require 16K bytes.

A network of 64 stations interconnected with a Delta network requires a total of $6 * 32$ nodes, a total of 192 nodes. If each of these nodes has 4K buffers, the simplest form, then the total memory requirements for the network is 768K bytes of memory. If larger amounts of buffer space are supplied then proportionally more memory is required. For a network of 1024 stations, 5120 routing nodes are required to implement a Delta network. At 4K bytes per node this is 20MBytes of buffer space.

Clearly there is a need to limit the packet size as the number of stations connected in a Delta network increases, in order to keep the network realistic in terms of size and cost.

Alternatively, if the packet size was reduced to some very small size, and the network investigations discussed in chapter 1 have revealed that most packets are small[43], the buffer space can be reduced to some proportionally smaller size, making its implementation in VLSI along with the control logic of the routing node feasible. Unfortunately reducing the packet size will have its effect on the structure of the higher layers of protocol and, where an acknowledgement is required for each packet transmitted, reduce the total throughput of the network.

4.7 The Route Server

It is typical for Local Area Networks to have a name server whose function is to map textual service names into machine addresses. Alternatively, broadcast packets carrying the requests for a service are detected by all stations who filter out packets intended for the service or services they implement. Unfortunately neither of these methods can be used in a Binary Routing Network.

Depending on the structure of the network, the route to the Route Server (the service whose job it is to generate the required route between two stations), will not necessarily be the same for all stations and thus cannot be built in to the software. Even if the route is the same for all stations, then it will always be necessary for the Route Server to be connected at the same point on the network. This means that if there is an error in the hardware, the service cannot be moved

to another machine.

To solve this problem it is necessary to get packets to the route server without the requesting station knowing where the server is. To enable an arbitrary network structure to be used the only solution is to build the Route Server path into the network so that when stations inject a Special packet into the network it follows this path and not the one contained in the packet. This can be achieved by placing a switch at each node which is set to indicate which output should be selected for one of these Special packets. Unfortunately this requires a large amount of management in a large network as many nodes will need to be updated if the Route Server is moved or the network structure changed.

Alternatively if the structure of the network is regular then routes can be synthesised, but unless the destination is known a similar problem exists.

4.8 Error Control

One of the major problems in computer networks is the control of errors whether just to detect incorrect data, to avoid packets arriving at the wrong station or to ensure that corrupt packets do not remain in the network passing from one node to another and never reaching a station receiver.

The detection of errors in the transmission of a packet is a well known process. Packets are transmitted with a *parity bit*, or bits, or a *Cyclic Redundancy Checksum*. These bits can be checked against values recomputed at the receiver and if found to be different, the packet, which is definitely in error, can be discarded.

Such a system can cope with most errors in Binary Routing networks and will ensure that corrupted data is not interpreted incorrectly. Either the packet arrives at the correct destination with a bad data field, or it arrives at a random station, possibly with a valid data field. In both cases the packet should be discarded.

Unfortunately it is possible for a packet to be corrupted in such a way that the route it takes around the network brings it back to a node it has already passed through, resulting in a packet that could possibly circulate indefinitely. If the networks are constructed without circular paths, the problem would not exist, but as we would like to build a general network it must be dealt with.

Binary Routing Networks which discard the route bit as the packet passes through the node will not suffer from an indefinitely circulating packet as a bit is used up by each passage through a node. Any error that caused the packet to follow an incorrect route and to pass through many more nodes than expected

would also use up all the route bits and then the data bits of the packet. If the packet manages to avoid arriving at a station before all the data bits are consumed, then the node would be able to discard the packet, which would now only consist of the start bit followed by the end of packet indicator.

In networks where the route field is not discarded but retained, the packet length remains the same and the bits already used would return to the front of the packet.

The addition of a count field to the header of the packet has been the solution used in packet switching networks. Such a field is set to the maximum number of switching nodes the packet expects to pass through, before reaching the receiver. As the packet passes through the node the count is decremented, and if it reaches zero the packet is discarded. This mechanism works well in systems where the packets are completely received at each node, or at least where the header is received, before transmission on the next link commences.

In a Binary Routing Network it is desirable to keep the delay of a packet passing through a node to a minimum. Any extension in the amount of the header that must be received before transmission on the next link would seriously affect the performance of the network. The operation of decrementing the count field could be done on the fly as the packet passed through the node, but the packet could not be discarded by the node as it would already be part way through to the next node. This could be overcome by forcing the packet to be buffered in a least one node in a network which has a circular path. At this node any packet that has a count, only decremented at this special node, which has reached zero would be discarded. Packets should only pass through such a node once and thus the count could be reduced to a single bit. Such a solution is practical as the construction of large networks would necessitate nodes with buffers at the interface between remote switching centres, to handle the changes in clock rates and line modulation where packets would need to be completely buffered.

An alternative approach is to ensure that the packet is forced to leave the network. This can be achieved by having an address which will guarantee that the packet will reach a particular station. The simplest pattern, which the nodes must be able to place in the packet as it passes, is all zeros. As the packet passes through the node and the route field rotates, the bit moved to the end of the field is zero. This unfortunately restricts the structure of the network again, requiring special attention when loops in the network are created, to avoid a loop with an all zero route.

Another simple modification to the route field would be to generate the new bit from a combination of all the other bits presently in the field. Such combinations in the form of pseudo-random sequences can be generated on the fly as the packet passes through the node. With a 16 bit route field the sequence of bits generated would last for 2^{16} bits before repeating. The logic, required is very simple, consisting of a number of *exclusive-or* gates and counting logic to pick out particular bits of the passing route field. This does not guarantee to be perfect as it would be possible to construct a large enough network which has a path as long as the pseudo random sequence.

The packet can also be discarded by a node if the 2 bit pairs, described earler, are used in the route field. One of the four values is used to indicate the end of the route field. When this value reaches the front of the packet, the packet can be discarded.

4.9 Diagnostics and Maintenance

In any network it is desirable to be able to use the network itself for diagnosis and maintenance operations. In a Binary Routing Network it is particularly desirable because of the number of routing nodes which make up the network. A failure in one of the links or nodes does not mean the total failure of the network.

It is not possible for a single station to carry out the diagnosis of the network as packets from that station will not necessarily be able to pass through all nodes or links in the network. Thus to completely test the network the diagnosis operations will require the participation of all network stations.

One of the simplest test procedures is to have all the stations run diagnostic programs, each testing the part of the network that they can access. This procedure will cause the network to be tested many times over and has a number of problems. Firstly the results must be collected and correlated by some managing station, secondly it requires the nodes to run diagnostic programs instead of the user application programs (not a large problem if the station is a multi-process environment), and thirdly the station must be able to run the diagnostic programs which may not be possible in a network where there are many different types of machines.

A second method would be to have a special packet type which is sent only by the diagnostic station. This packet would carry a route to be tested and would include a forward route and a number of supplementary routes. On receiving such

a packet a station would transfer the packet from the receiver to the transmitter where it would launch it back into the network with the first of the supplementary routes. Where the packet is capable of having long route fields these supplementary routes could be carried as extensions to the main route field otherwise they must be carried in the data portion of the packet. To diagnose the network's problems, the test station would construct the route and supplementary routes to cause the packet to follow a particular path through the network, thereby allowing it to test all paths and nodes in the network. This form of testing requires a special packet format which can be distinguished by all stations, and requires that the stations perform the reflection operation. It would be possible to have the hardware do the reflection of the test packets, a preferred option as it would take away all the requirements from the stations, but this would add to the complexity of the station.

As well as determining which paths are functioning in the network, it is also desirable to know which nodes in the network are experiencing transmission errors. This form of diagnosis can detect the early stages of a fault before it completely impairs the operation of the network. Packets passing through a node have their parity check regenerated so that an error does not propagate past the node-link-node position where it was generated. A special packet passing through the network, as described above, can have a secondary function which is to collect statistics from a node as to the number of errors that it had detected. As one of these packets passes through a node the statistics can be appended to it, provided that it is of variable length and that the length can be extended on the fly. Alternatively the special packet can carry the address of a node in the data portion of the packet and as the special packet passes the node the address is compared and the node places its statistics in the field set aside for it. With these methods a fixed packet can be sent and predetermined nodes can be covered. Unfortunately this means that the complexity of the node is increased.

A second solution to the early detection of transmission problems is for the node to inject packets into the network whenever the number of errors detected reaches some threshold. Such a packet cannot necessarily be directed at the diagnostic station, because of the node's position in the network, but any stations receiving such a packet would retransmit the packet with the correct address. If this Logging Service was incorporated in the same machine as the Route Server then the same mechanism as used to route special packets through the network can be used to forward logger packets.

4.10 Expanding to a Wider Network

When there are a number of clusters of computers connected by Binary Routing Networks, these can be bridged together to form a larger network. By simply connecting together these networks, each sub-network will view the other sub-networks as extensions to itself. The operation of the small clusters will not be affected by the operation of the other clusters as internal communications will be contained within the local portion of the network, although there may be requirements for inter-network packets to pass through a network.

To communicate with other processors outside the local grouping the only alteration to the packet needs to be an extension of the route field to first direct the packet out of the local network to the link, to either the desired network or to another network where it can pass through. Provided that the packet can contain the complete route there is no requirement for the packet to be processed in any of the networks it may pass through.

If there is a large separation of the network clusters where the high bit rate links cannot be used, then lower rate links can be inserted. For packets to pass through these links they must first be buffered on entry into the low speed link and again on exit from the link so that their speeds can be altered. Packets using these links will again not require any modification although the protocols controlling the packets will need to use longer timeout periods to match the longer delays.

By interconnecting clusters, the problem of circular paths is again introduced. Even with the simplest division of a network, into 2 clusters joined with one link each way, a circular path is constructed. At the points where the external connections are made to a cluster, bit rate changes will need to occur and this will mean the packets will need to be buffered. When this happens circulating packets can be removed.

4.11 Summary

Binary Routing Networks are a new form of communications network that can be used in the local and the global environment. They use a mixture of spatial and time division multiplexing to provide higher throughput and, with a simple buffering and routing strategy, can be easily constructed without large amounts of logic. They differ from the traditional Local Area networking approach, of broadcasting packets to all stations, by routing packets directly to the desired station along a predetermined path. This also differs from the Wide Area Packet

Switching networks approach which requires an intelligent controller at each node to determine the output link the packet must take on the next leg of its journey. These gains are obtained by placing restrictions on the layout of the network and by ensuring that the network is fully operational at all times.

There are a number of topologies that can be used to construct Binary Routing Networks. Interconnected rings and trees can be used for networks where local communications are heavier, and the requirements reduce the further away the destination station is situated. Delta networks can be used to give an even performance for any station communicating with any other station. It is even possible to construct random topologies but they are less likely to be as useful.

The management of these networks all have varying degrees of complexity. For the Delta network there is no management required, a packet can be sent to another station by the simple use of the destination address in the route field of the packet. Other networks require increasingly more effort in routing packets through the network, which can only be resolved by constructing a Route Server which has a built-in map. This then only leaves the problem of how to send packets to the route server. Expansion of the simpler networks requires less effort and logic whereas the Delta network can only be expanded by doubling the number of stations in the network.

The use of buffered networks has a number of problems. First, there is the case where packets cannot be delivered because the station has either failed or is powered off and some form of buffer control is used to ensure that the rest of the network is not held up. Secondly, as there is no intelligence in the routing nodes the packets must follow a predetermined path which cannot be modified by the network and cannot be redirected to avoid congested areas of link failures. If alternative path selection was available it would have to be performed by the routing nodes which do not have any state information about the network other than the node's local links. Also different paths would result in out of sequence packets, increasing the complexity of the protocol. Finally the provision of large amounts of buffer space would make the nodes expensive to construct.

Chapter 5

Non-Buffered Binary Routing Networks

This chapter introduces a new form of Binary Routing Network, the *Non-Buffered Binary Routing Network.* As its name implies, this form of network does not buffer the packets at the nodes when they are blocked, but rather discards the packet. The retransmission of the blocked packets is not a function of the network nodes but that of the transmitting station. The station is informed by the network that the packet was blocked, which causes it to stop transmitting and to retransmit the packet at some later time.

The amount of logic to implement such a routing node is considerably reduced from a node which requires buffers. The extra logic required in the station does not significantly increase as the buffers to hold the packets are necessary in all forms of the network. Little logic is required to retry a transmission, whether automatically or under the control of the host computer.

Other advantages are available to a non-buffered network. First, if a station is incapable of receiving a packet the transmission will be interrupted and the network resources freed. Secondly, an error in the network detected as a blockage will halt the transmission and again free network resources. Thirdly, any form of blocking in the network which causes retransmission can be used, in a network with alternative path selection, to redirect the packet through another portion of the network which may not be suffering from the same problem. Fourthly, if a packet is blocked for any reason the transmitter can select another packet from its output queue which, provided it has a different route value, may succeed in finding a path to its destination.

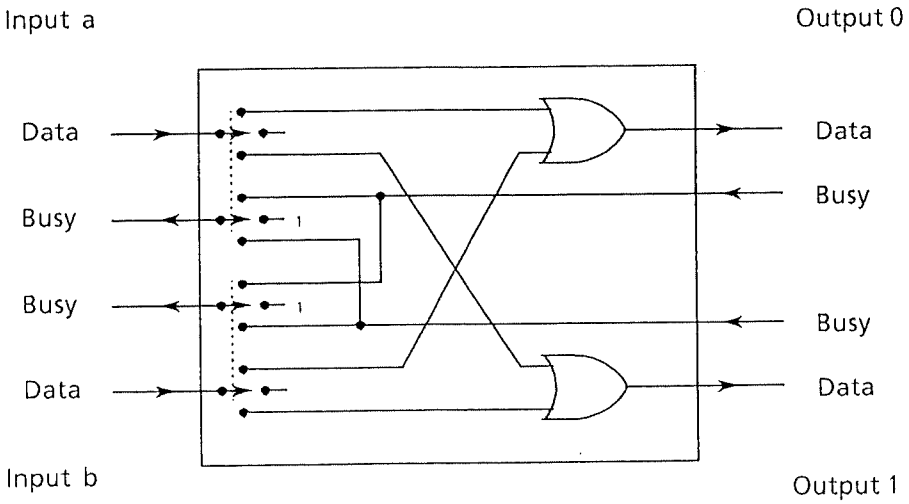


Figure 5.1: Basic Structure of Non-Buffered Routing Node

5.1 Non-Buffered Routing Node

The basic operation of the Non-Buffered routing node is the same as that for the node which handles collisions by buffering. The node directs incoming packets from the inputs to the outputs depending on the route field bit. It modifies the packets so that the second significant bit is moved to the front of the packet and it performs flow control. Figure 5.1 shown the basic control outline of a Non-Buffer Routing Node.

When a packet arrives at a node and the required output is already in use passing a packet from the other input, then the new packet is considered blocked. At this point the node starts discarding the packet and passes a *Busy* signal back to the transmitter via a reverse path which has been set up by the other nodes involved in the routing of the packet. This signal informs the transmitting station that the packet has not reached the receiving station and that it should be retried at some later time. If the transmitter was not informed of the blockage, it would be necessary for higher levels of the network software to time out on the non-reception of a reply. As the blocking of packets is a highly probable event then a timeout scheme would severely restrict the maximum throughput of the network.

As well as indicating blockage in the network, it is possible for the receiver to generate the busy signal when it is not in a position to receive a packet. This means that when the transmitting station does not receive a busy signal during the transmission of a packet, it can assume that the packet was received, though not necessarily correctly.

5.2 Packet Format

The structure of the packet does not need to be modified to use the non-buffering scheme. The only modification that could be required would be to enable the packet to be shortened as discussed in the next section.

By avoiding the buffering of packets at nodes it is possible for any restrictions on maximum packet length to be removed, except for those imposed by the transmitters and receivers.

5.3 Packet Termination

In a Non-Buffered Binary Routing Network it is desirable to quickly terminate the transmission of a packet when a blockage has been detected in the desired path. This will effect the system in a number of ways. Firstly it releases the nodes used to set up the path of the packet, allowing other packets to use the links which would otherwise have been used to transmit an unwanted packet. Secondly, a quick retransmission will find an open path earlier. Both of these mechanisms will increase the throughput and decrease the delay characteristics of the network.

The main problem area is terminating the transmission and informing the nodes that transmission is complete and that they can be released in some minimum time.

The simplest solution is to have a timeout. When a *busy* is received it is immediately passed back to the previous node. Transmission is chopped off with the output line returning to the idle state and the timeout set. When the timeout expires it is assumed that the previous node has terminated transmission and that this node can put itself back into the idle state and await the arrival of the next packet. The problem with this is how long to make the timeout. A short timeout is desired to clear the network quickly, but the timeout must be long enough for the signal to have passed back to the previous node, the transmission to be completed and the line to have settled. Where the line is long this time will also need to be long and thus careful tuning will be needed to give an optimal solution. In general, where the nodes are clustered together to build a large switching unit, the nodes

will be very close and the packets can be removed within a few bit times. It is only in the long links of the network where there may be many bits in the link that problems will arise.

A second solution is to have the *busy* passed back through all nodes to the transmitter, which on reception will terminate transmission at the next available point, ie. when the end of a data block is generated. The nodes will then see the end of the transmission and thus will be able to release themselves. This solution has the advantage that the time through the network and the interconnection cables need not be known to give an optimal solution. To implement this solution there needs to be some way of informing the nodes that the transmission has ceased. This can be achieved by either having a unique marker that is distinguishable from the normal data of the packet, or by having some fixed packet structure which has variable component lengths which can be reduced to some suitably small size automatically. The normal operation of the network requires some means of detecting the end of the packet structure, discussed in the previous chapter, and this could be combined with the truncation procedure.

5.4 Minimum Packet Length

To make a Non-Buffered Binary Routing Network perform efficiently it is necessary for the transmitter to detect the presence of a blockage in the network. To ensure this, the packets must be longer than the time for the header of the packet to be clocked through all stages of the network and for the busy signal to propagate asynchronously back through the return path. Packets that are smaller than the minimum length can pass through the network but they risk being blocked without the transmitter's knowledge.

If the delay through the node is M bits and assuming the delay in the links is negligible then, to ensure that the packet is still being transmitted when a busy signal is received at the transmitter, the packet must be at least $M \cdot N$ bits to cover the forward time period, where N is the number of nodes in the path to the receiver, and some extra bits to cover the return path delay. Given the delay in a node is 3 bit times and a network with a depth of 10 nodes, a packet would require a minimum of 30 bits plus a small extra amount. With 10 bits already existing for the route field this leaves 20 bits for data, or just under 3 bytes. Also given that higher level protocols will be implemented on top, a minimum of 10 bytes of data would not be an unreasonable figure. Temple[43] has measured the flow of

traffic on the Cambridge Ring and found that 50% of all packets contain between 8 and 12 bytes of data. At 10Mb/s a single bit time is approximately 100 feet of cable which means that within a cluster of nodes no additional packet length is required for the cable delay. In a network with long connections, say between remote clusters of routing nodes, buffering would normally be required for bit rate conversion. In this case the transmission would be to the output buffer and would not continue down the link.

5.5 Low Level Protocols

At the very lowest level it is possible to gain information that is useful to the higher levels of protocol. In the Cambridge Ring each minipacket transmitted on the network returns one of 4 possible codes and this information can be used by higher levels of protocol to recover from some forms of network errors. In particular it is possible for the transmitter to detect if the receiver is powered off, is in the process of receiving another packet, or does not wish to receive a packet from that particular transmitter.

By coding the Busy signal it is possible for the transmitting station to know the reason for blocking of a packet. If the packet is blocked in a node by another packet passing through the node then the transmitter should retransmit the packet with a short timeout. If the packet is blocked by a receiver which does not have any buffer space free, the transmitter could delay the retransmission. If the receiver is not powered on then the packet would be blocked in the final stage and the busy signal would indicate to the transmitter that the packet should not be retransmitted and an error passed to the high level protocols. Similarly the non-reception of a reply can, if the packet is long enough, indicate that there has been a failure in the network and that a retransmission should be attempted.

Unfortunately experience with the Cambridge Ring has indicated that most of the returned information is not useful when the communications pass through a bridge. The bridge, which has been optimised to be able to accept packets quickly, will not be able to inform the sending station of a powered off or rejecting receiver when the packet is transmitted on the second ring. In the Cambridge Fast Ring the return code information is not passed out of the control chip, which only uses the information to determine if a particular packet requires retransmission due to a receiver buffer being full. Any other return code results in the packet being discarded.

For a Binary Routing Network which is split into clusters and which have buffers on the connecting links, the return code information again becomes redundant. The user will see a successful transmission of a packet but this will only indicate that the packet has reached the first buffer and not that it has reached the receiver. So, as with the Cambridge Fast Ring, the information should only be used by the hardware to indicate that a retransmission should be attempted.

5.6 Improving Performance

In any network it is desirable to have more than one path between nodes, to provide for times when a link fails or just to decrease the probability for having a packet blocked.

In multi-drop networks such as Ethernet and the Cambridge Ring it is not possible to add redundancy without adding a lot of extra hardware and greatly increasing the cost of the network. For other networks, e.g. Flood Net and the Flood Sink, redundancy is an inherent part of the network structure which can both improve the performance and satisfy the desire for tolerance to link failure.

Alternative Path Selection

In a Binary Routing Network it is not, in general, possible to have multiple paths as the packets follow the one route that is set in the packet header. The network shapes that have been described previously do not allow for multiple paths, although such a network could be constructed. If multiple paths are available, it would be necessary for the route server to supply alternative routes when a route lookup operation is performed and for the user to select the route to try. Also it would be desirable to have the redundancy transparent to the users of the network and applied automatically.

When examining the structure of an Omega shaped network, see Figure 5.2, it can be seen that all of the inputs are identical, the same route presented at any input would direct a packet to the same output. It is possible to create some redundancy by adding an extra stage to the network. This stage would route the packet to one of two inputs of the original network by selecting one of the stage outputs at random. The packet would pass this node untouched but would pass through the rest of the network in the normal way. If the packet is then blocked the retry would also pick a route at random, or possibly the alternate output to the previous selection, and the packet would at some time take the second path which

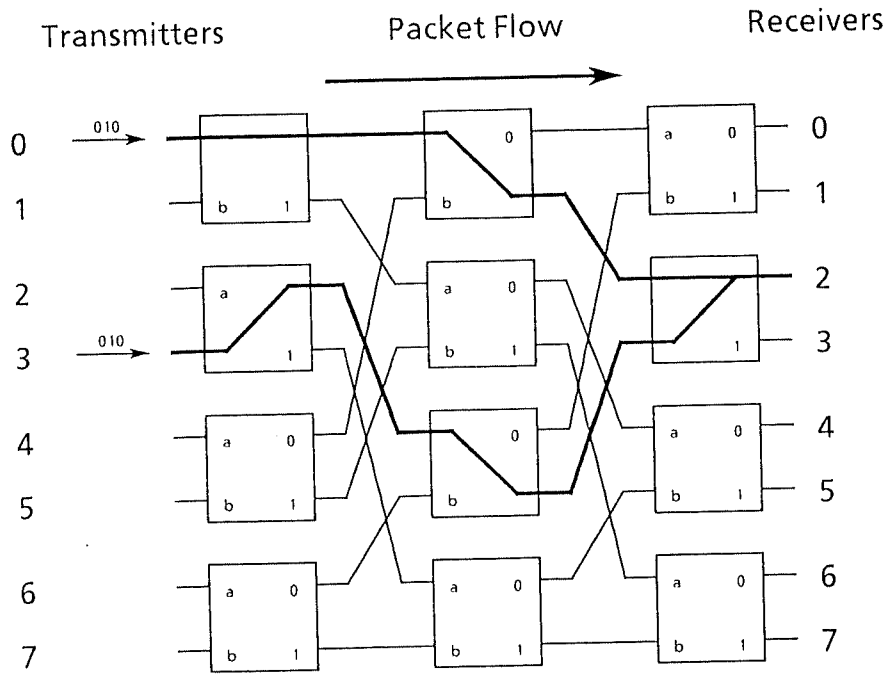


Figure 5.2: Path Equivalence in an Omega Network

may not be congested. The selection scheme would not be completely random and would only select at random if both outputs were free. If only one output was free then that one would always be selected.

Figure 5.3 shows a network with a layer of randomizing routing nodes. The Omega network structure has been chosen for this description because the addition of an extra stage at the input can be made by simply adding another unit to the front of the network. The resulting network will then be able to route a packet via two paths that do not contain a common link until the end of the network is reached. A similar addition can be made to a Perfect Shuffle shape or any other arrangement which does not distinguish between inputs.

This form of redundancy consists of just rearranging the inputs at random, thus allowing a packet to follow one of two separate routes. If a second redundant stage is added then each packet has the chance to use one of four routes that will all lead to the same output.

By using the randomizing stage, the alternative paths that a packet will follow can make the network fault tolerant. If the transmitting station decided that its packet did not get through, either by a busy signal or the non-arrival of a positive acknowledgement, then the station will retransmit the packet. The retry may

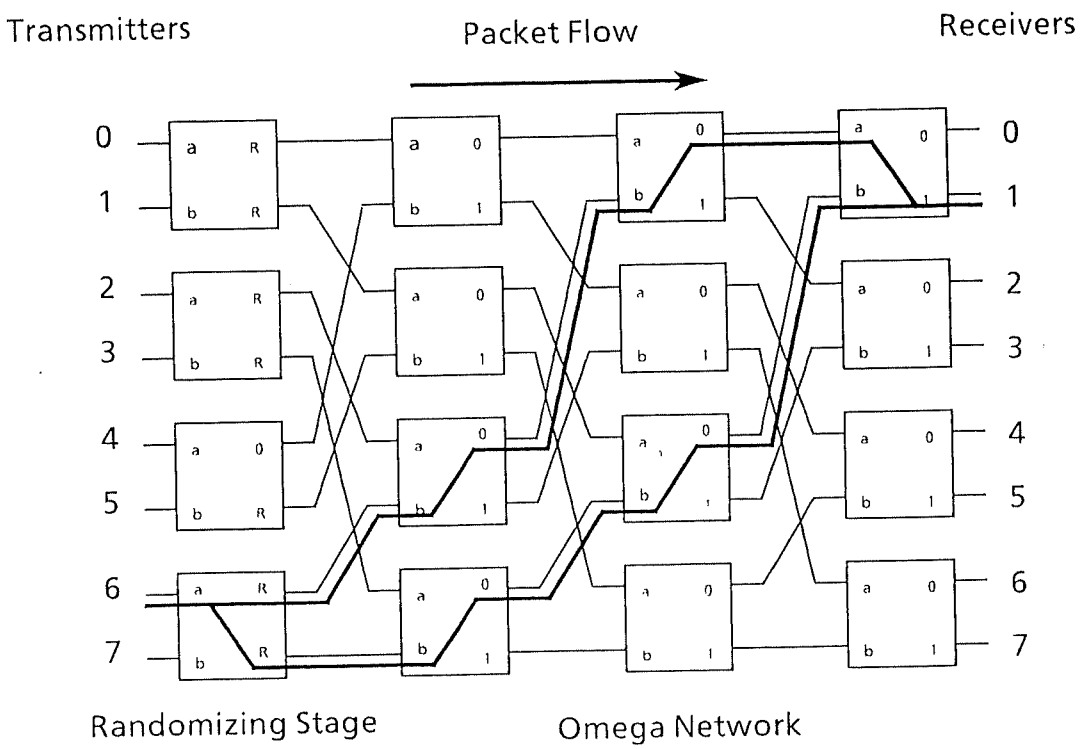


Figure 5.3: BRN with Redundant Paths

select an alternative path which may not contain the same fault. This will result in an average of twice the number of transmission attempts when a link has failed but it is a small price for a network kept running under these conditions. It is also possible for the failure to occur at the very beginning, in the randomizing stage, or in the final stage where it is not possible to avoid the problem. It is not possible for this sort of error to be overcome without complete duplication of the network, where failures in the receiver or the transmitter would have the same effect.

One of the main benefits of this scheme is that at light loads the delay in delivering packets is reduced. With a single stage of random routing, packets will have a choice of two routes to follow thus reducing the probability that they will find a link which is already occupied. At high loads the maximum throughput of the network will not be affected as much because a packet just starting will find that its passage through the randomizing stage network is forced as the other station will most likely be sending a packet itself.

It is not easy to define what is a worst case distribution for a network with a randomizing stage. One possible scenario is for all stations to transmit packets to themselves. For a Perfect Shuffle network, this would mean that a few links would carry most of the load and would result in heavy blocking of packets. By adding a

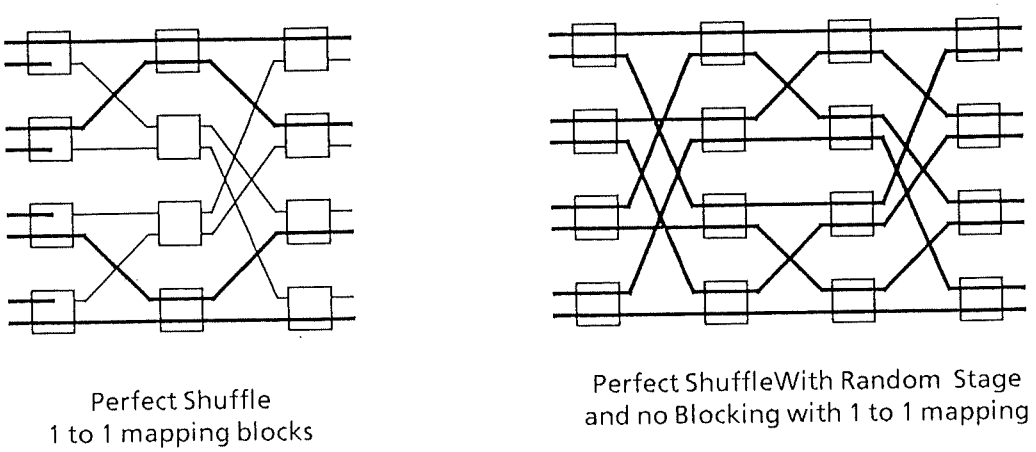


Figure 5.4: Best Case Pattern

randomizing stage, as in Figure 5.4, the same distribution pattern would become non-blocking. Unfortunately this is only one particular example and counter examples with different routing patterns could be constructed, which would operate best without the randomizing stage.

A second way of adding the extra redundancy is to duplicate the network and to have two identical networks running in parallel. At the transmitter, the packets can choose one of two paths which takes them to either of the two networks. At the receiver there is a selector which chooses a packet coming from the two layers of the network. The cost of such a network is of course more than twice the cost of a single network, but it does add redundancy as well as increasing the throughput. It is possible for the packets to be injected into both layers of the network. This would mean that at each transmission both paths would be attempted at the same time. The busy signal would only pass back to the transmitter when both paths were blocked and in the event of both paths being set up only one would be selected at the receiver, the other being blocked and thus breaking down to allow other packets through the network along the links it had occupied.

Adding extra capacity

Where a station, such as a file server, receives many packets, additional capacity can be added by having two receivers. Provided the receivers are connected to the network by a common final stage which does not use the routing bit but just passes the packets directly through, the same route can be used by all stations communicating with this special station. A simple modification to the receivers can allow the packet to be routed to either receiver, depending on which is ready to accept a packet at the time. These changes involve the receivers passing out the busy signal when they are not ready and the routing node selecting a ready receiver as the packet comes in.

Alternative Retry Strategies

So far the networks examined have used a retry policy of repeatedly retransmitting the same packet until it manages to connect to the destination's receiver. This policy would be the only one available where the stations has only one packet to transmit. Stations similar to the small servers in the Cambridge distributed system normally only deal with requests from stations one at a time resulting in only a single packet to transmit at any one time. For other systems designed to handle many requests simultaneously, or for a main frame computer's communications controller, a queue of packets to transmit could well be available.

Where more than one packet is available for transmission the transmitter can apply different retry strategies. Instead of repeatedly retransmitting the same packet until it is delivered, the station could, after a packet was blocked, place the packet back on the queue and attempt to transmit the next packet. Provided the packet had a different route it would have a better chance of getting through the network. The controlling software would need to ensure that packets destined to the same station were not transmitted out of order, as software to correct this would add to the protocol over heads. This scheme would be very dependent on the implementation of the stations interface logic. If the station can quickly set up a different transmission an increase in the network performance may be obtained. If a long time is required the simpler retry strategy, avoiding repeatedly reloading the transmitter buffer, may result in a better performance.

The implementation described in Chapter 7 has two transmit buffers. These buffers take time to load and thus the only possible retry scheme would be to alternate between the first two packet on the queue.

Combining this scheme with the alternative path selection scheme may not

produce an increase in the performance of the network. A patten of route values and retry rates could set up a condition where the two schemes interfered with each other producing only a marginal improvement over the basic system. This would be especially true if the route selection scheme just alternated the chosen route instead of choosing it at random. The simplest solution would be to try each packet a number of times before giving up and changing to the next packet.

One area where this scheme would be best applied is in a multiprocessor system where data is broadcast to a number of other stations. A result that needs to be sent to many stations would present a short bust load to the network and result in a queue of packets at the transmitter with a number of different routes. These packets would have a long average delay if just transmitted in sequence as all waiting packets would experience the delays that the packet being attempted experiences. By rotating around the queue the total delay can be reduced as the load on the network increases by avoiding the waiting period while a packet is blocked.

5.7 Error Control

In a Non-Buffered Binary Routing Network where circular paths exist, it is possible for a packet which has been corrupted to circulate forever. There is no restriction on a packet arriving at a receiver before the transmitter has completed sending and, provided the packet is small enough to be contained wholly within the nodes, it can circulate. This state of affairs can only continue while the network is at very low load levels, for as soon as the load increases the possibility of a collision increases, and once the packet has collided it will disappear from the network. Since there is no station to retransmit the packet it will not reappear. The amount of lost bandwidth due to such a circulating packet will be small, firstly because it can only exist when the network is operating at low loads when there is plenty of bandwidth and secondly, as soon as the load increases, the packet will disappear and packets it caused to be blocked will be retried.

If a large packet is corrupted, in a network with a circular path and loops, it will eventually return to a node where the desired output is held by itself. This will cause a collision in the normal way and the packet will cease to be transmitted. When the packet is retried the probability of a similar corruption occurring again will be small and the packet should succeed in reaching the receiver.

Where buffering is used between clusters of routing nodes, all the same prob-

lems as discussed in Chapter 4 apply again. Direct action would then be needed to remove circulating packets from the network.

5.8 Applications of a Non-Buffered BRN

Binary Routing Networks are suited to applications in switching centres where large number of packets need to be passed through a switch from many different sources to many different destinations. To achieve the high performance possible from these networks it is necessary to use large regular structures and to ensure that packets from each input are well distributed amongst the outputs. This becomes even more necessary when Non-Buffered Binary Routing Networks are considered.

If the network is constructed with routing nodes distributed over a large area, then it will be impossible to use a structure such as the Perfect Shuffle without a large number of cables running between the nodes. By using simpler structures, to avoid large amounts of cabling, the throughput of the network will be reduced and the gains from using this form of network will also be reduced.

The implementation of a routing node in a Non-Buffered Binary Routing Network is very simple and could be done using very small amounts of logic. Consequently when it comes to building a network with these nodes the addition of large amounts of extra logic to interconnect the nodes, in the form of line conditioning, will add to the cost of the network. Additionally, to allow the network to be clocked asynchronously may require the addition of extra delay at the front of the packet while the clock is recovered and more complex circuitry for the arbitration of two inputs that are not synchronised.

By clustering the routing nodes in a switching centre a number of savings can be made. It becomes possible to build the network using one of the complex structures which will allow high throughput to be achieved. It is also possible for the interconnection of the routing nodes to be done without the use of complex line conditioning, possibly by direct TTL connection. The clocking of the network can be made synchronous over the switching centre, avoiding clock recovery circuitry and the network delays needed to achieve this. At the edges of the switching centre the connections to the rest of the network can have the more complex circuitry where it may be required to synchronise with clocks or to buffer the packet to change clock rates.

5.9 Summary

In this chapter a new form of Binary Routing Network has been introduced. This form of network, a Non-Buffered Binary Routing Network, dispenses with buffers within the network switching fabric and moves them to the edges of the network and into the transmitters. Packets that are transmitted and cannot propagate through the network are blocked and discarded by the routing nodes. Transmitters whose packets are blocked retransmit the packets at a later time when they may or may not connect. By removing the buffers from the routing nodes, the nodes become simpler and thus less costly to construct, making their fabrication in VLSI practical where high order routing nodes could be constructed on a single device.

It is possible to gain extra performance by using a scheme of selecting a different packet to transmit instead of retransmitting each packet until it is connected. In addition to the transmitters selecting different packets, it is possible for the network to select a different path for the packet to follow at each retransmission, thereby avoiding either blockages caused by other packets or by failures in the network.

Chapter 6

Network Performance and Simulation Results

This chapter examines the performance of a number of Binary Routing Networks. This takes the form of results from simulation. Many people have analysed networks with structures similar to Binary Routing Networks, by queueing analysis, numerical analysis and simulation. Unfortunately the systems differ in many ways. These analyses can be found in Bhuyan et al.[5], Cheemalavagu and Miroslaw[7], Davis and Siegel[10], Dias[12], Jenq[17], Kruskal and Snir[19], Kumar[22], Mirza[31] and Wah[46]. I have used one of the numerical models to find a solution which I compare with the simulated performance for the Binary Routing Network. In Chapter 7 results of the measured performance of the prototype system will be compared to the simulation results.

6.1 Simulator Operation

The simulator is a discrete time event driven simulator. It is constructed in three parts, a simulator module which processes the events, a network description module which manages the movement of packets through the network and a station module which defines the actions of the stations connected to the network. The network module and the station module have a number of variations which can be interchanged for performance simulations on networks with varying parameters.

The Network Model

The network model is divided into two main groups; those dealing with unbuffered networks, and those dealing with buffered networks. The main function of the network model is to define the shape of the network interconnections and to define

the actions taken by the network to pass a packet.

Unbuffered Network Model

The unbuffered network models define three network shapes:

1. Perfect Shuffle network shape.
2. Omega network shape.
3. Omega network shape with Randomizing stage.

The first model, is the Perfect Shuffle. To construct a network with 2^N stations, two networks of size 2^{N-1} are joined with a shuffle exchange unit. Each sub-unit of the network is similarly constructed by the same recursive model until the subdivisions reach a single network switching node. This shape is important because it can be used to construct networks that can be expanded, admittedly in increasingly larger lumps, whereas other shapes require the whole network to be reconstructed to retain the same connection strategy.

The second model, the Omega, has been included in the simulation to verify the premise that there are many ways to construct the network and that they can all carry the same load. The shape was chosen from the many other shapes because it is easy to construct. The same connection pattern is used between each stage of the network.

The third model, Omega with Randomizing stage, relates to the mechanism which can be used to improve the performance of the network by increasing the available paths that a packet can take. The first node in the network does not examine the route information but chooses a random value instead. The Omega shape was chosen because the redundant stage can be added to the network simply by adding an extra stage using the same algorithm used to construct the rest of the network. Any shape network could have been chosen for this model but construction would not have been as simple.

The node model that has been used in the majority of the simulations does not conform to the real model in one important area. To speed up the operation, when a packet is to be transmitted through the network, the network is examined to see if the route for the packet can be set up. If the route is free then it is set up and all involved node outputs are marked as Busy. If the route cannot be set up then the packet is marked as blocked and a later retry time set. None of the node outputs, even the ones that the packet could have reached, are marked as Busy. All this testing happens within the same simulation time unit, and may occur with other

such route setup attempts, none of which will collide. This means that packets which are starting their transmissions will not interfere with other similar packets unless they manage to connect to a receiver. In the real network operation new packets will collide both with connected packets and also with packets attempting to set up their routes. The main effect of this simplification of the model is to increase the throughput of the network.

To examine the effect of the above simplification a special network model, based on the Perfect Shuffle, where packets move from node to node one per clock period, is also investigated. As packets move from node to node the required outputs are marked as busy, thus blocking other new packets entering the system. Also when a packet is blocked the outputs are not immediately released but held for a short period corresponding to the time for the end of the packet to reach the last node. This model unfortunately runs approximately 10 to 100 times slower than the simpler model and is used only in one model to compare the results with the other simulations.

Buffered Network Model

The Buffered Network Models are all based on the Perfect Shuffle. The nodes are subdivided into four types defined by the buffering strategy. The different buffering strategies are:

1. Single Packet Input Buffer.
2. Limited Packet Input Buffer.
3. Infinite Packet Input Buffer.
4. Infinite Packet Internal Buffer.

The Single Packet Input Buffer model has the buffer placed at the input of the 2×2 switching unit. Each buffer is capable of taking a single packet and while the buffer is full, the Busy signal is passed back to block further transmissions.

The Limited Packet Input Buffer model is similar to the previous model but instead of a single packet buffer there is space for a limited number of packets. Again the Busy signal is used to stop the flow of packets when the buffer becomes full.

The Infinite Packet Input Buffer model also has the buffers placed before the switching unit. This time the buffers are capable of holding an infinite number of packets. There is no Busy signal and packets can always move into a buffer in the next stage provided they are allocated the output.

The Infinite Packet Internal Buffer Model has the buffers placed inside the switching unit after the direction selection, but before the output arbitration. This requires that there are four buffers instead of the original two, because there must be two buffers for each input corresponding to each of the two outputs. A single packet buffer model is not possible with this arrangement because it is not possible to determine if a packet should be blocked before it arrives at the node.

The Network operates in the Cut-Through mode, which allows packets to pass through a number of nodes before they need to be saved in a buffer. The model restricts the packets to be completely buffered, so that the whole packet is held before transmission in the next stage of the network is commenced. When a packet has completely left the node buffer and the output becomes free, the simulator operation is such that if there is a packet in the other buffer requiring the released output, then it will have priority over a new packet entering the node. If there are two packets already in the node buffers requiring the released output, then one will be selected either by random or selecting the least recently selected buffer. The delay in a buffered packet restarting when an output is released is one simulation time unit. This short period would correspond to direct hardware signaling within the node.

The Station Model

Non-Buffered Networks

The station can provide a number of different options to do with the processing of packets, in particular what action to take when a packet is blocked. These actions are:

1. Immediate retry of packet.
2. Move the packet to the end of the queue and try the next.
3. Alternate retries between the first two packets on queue.

The first system is a very simple model. It keeps trying the same packet until it gets through. This model is important in that it is expected that most small servers connected to a Binary Routing Network would only be sending to one other station on the network and thus would only have one packet to send at any particular time.

The second model is based on the mechanisms normally employed in high performance host interfaces where there is possibly more than one packet ready to be transmitted. If a packet transmission is attempted and the path found to be

blocked, then the packet is placed back on the end of the transmit queue and the transmitter attempts to send the next packet. To gain an improved performance with this strategy there must be at least two packets on the transmit queue, and preferably more. These packets also require their routes to be independent and well distributed amongst all the destinations.

The third model is included to reflect the state of the implementation. The station interfaces have two transmit buffers. While one buffer is being transmitted the other can be filled up. If both buffers are full then it is a simple matter to switch between one or the other with little cost, increasing the probability that one of the packets will get through in a shorter time provided that they are following different routes.

Buffered Networks

The operation of the station in a Buffered network is very simple. If a station has a packet to transmit, then it will attempt to send it. If the packet is blocked because the buffer in the first stage is full, then the station will suspend its operation until the buffer is empty. No special retry strategy needs to be applied when a packet is blocked as the blocking is a function of the buffer in the first stage which cannot be affected by the route that the packet takes, and thus nothing can be gained by attempting to transmit a different packet.

All models

In all the models it is assumed that there is a small delay between the completion of one transmission and the start of the next, for both successful transmission and blocked transmissions. This delay is added to simulate the interrupt time that would be expected on a working system to switch from one packet to the next. This time is very hard to quantify as it is very dependent on the construction of the interface. For a high performance interface this would require the setting of registers which point at the packet to be transmitted and the time to enter the interrupt service routine. In the simpler interfaces it may be necessary to copy the packet from memory to the transmitter buffer which would take a much longer time.

Another function of the station model is to define the arrival of packets and their contents. Packet arrivals are defined to be *Exponential* around a given mean and the retry rate is also exponential about a second mean. The randomness of

the retry rate is to reflect the variable time that a processor may take to process an interrupt.

A final variation which can be made to the station model is in the range of the packet lengths. The packet length is also a parameter to the simulation and is in terms of the same time units as the arrival rate. This allows for a simple formula in determining the applied load by working out the percentage of time necessary to deliver all the packets into the network at the maximum bit rate.

Random Number Generation

In order to speed up all simulations the use of real numbers inside the simulator has been kept to a minimum. The only area in which they were used was in the generation of random numbers. To avoid using a logarithm function for every random number a simple table was generated with 1024 elements and use as a lookup table for the function $-\ln i/1024$.

The results from these generators were then rounded to integer values. This has meant in some cases that numbers needed to be generated with a mean close to zero. This was required to keep the statistical results within the limits of a 32 bit integer, for example 1000 stations delivering 1000 packets of length 1000. If a finer grain was used then the totals could possible exceed the integer size.

To ensure that no adverse effects were produced from this truncation some simulations were done which did use a finer grain. In all cases the results were the same to within 0.1-0.2%. As well the mean of the the generated random numbers was calculated and these agreed very closley to the mean value used by the generator. For example a generator with mean 2.0 produced a average of 2.0231 over 100000 samples, while a generator with mean 20.0 produced an average of 20.2987.

6.2 Simulation Results

Maximum Throughput

Figure 6.1 compares the maximum throughput of most of the network models. These results were obtained by running the station in a mode where it always has a new packet to transmit when the previous packet was completely sent. The operating conditions were such that the retry time was very much less that the packet transmission time. The long packet time also removes the significance of the path setup time which varies with the size of the network.

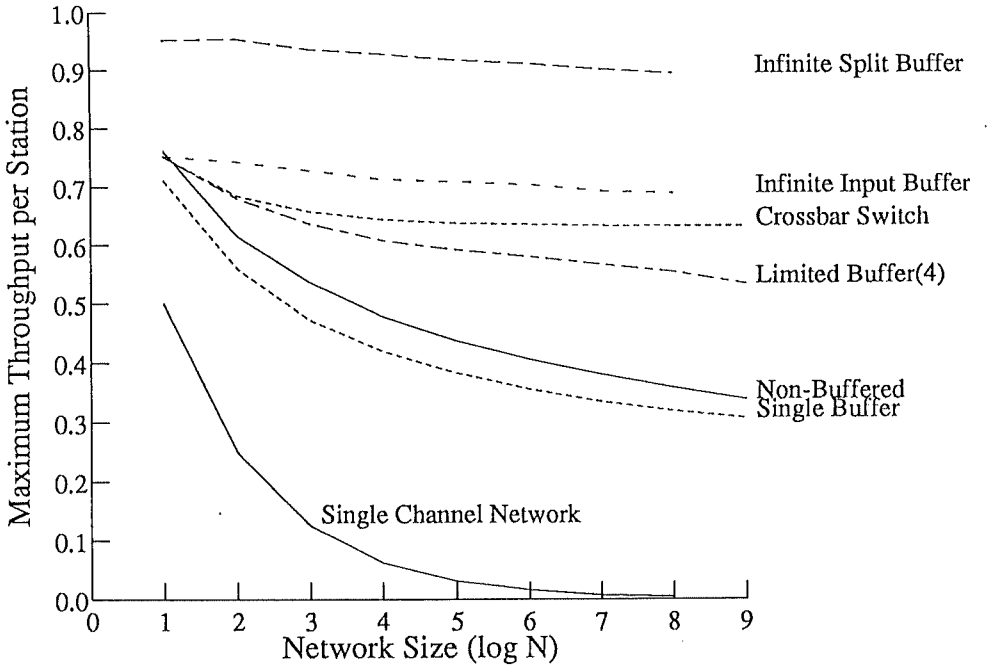


Figure 6.1: Maximum Throughput

The results are plotted as Throughput vs. Network Size. The throughput is the measure of the average number of packets that each receiver received during the simulation period as a percentage of the possible number of received packets. To help in the comparison for the different network the data points of each system have been connected. These graphs are in fact discrete and not continuous.

These results represent the maximum throughput achievable by each of the network structures and in some cases require the use of infinite length queues to enable the network to operate at these loads. These maximum throughputs must thus only be considered as the upper bound.

Two other lines on the graph indicate the maximum throughput expected, for each station, for networks connected with a single channel and a full cross-bar switch. For the single channel network, for example Ethernet or the Cambridge Ring, the line is in fact just the function $1/N$ and would represent ideal operation. It does not take into account any lost bandwidth from collisions or the particular allocation strategy used. For the cross-bar switch there can be no loss in the network, as there is full connectivity between all stations, and the line represents the probability of finding the destination station already connected to some other station. The function here is $1 - (1 - \frac{1}{N})^N$ which has as a limit as the network

size increases of $1 - e^{-1} = 0.632$[35].

Some of the results for the larger more complex networks are not complete. This is because of limitations in the machines used for the simulations. These networks have very large queues in the nodes and when these systems were simulated for long periods the memory limit of the machines used were exceeded well before the required period had been achieved.

Buffered Networks

Figure 6.2 compares the performance of the different buffer options. The interesting point to note is that as the networks reach capacity and enter overload the volume of packets being delivered does not decrease. This would be expected as new packets entering the system cannot have any effect on packets that have already passed into the network. Thus, this form of network would be capable of accepting and clearing a short overload.

Non-buffered Network with Immediate Retry

Figure 6.3 shows the performance of the Non-Buffered Network where the packet is retried indefinitely until it is transmitted successfully. As with the buffered networks the throughput of the network reaches a plateau and does not fall back as the applied load passes the maximum value.

The different lines on Figure 6.4 represent different retry rates. These rates are in terms of a fraction of the packet length and range from 2% to 20% of that length. This indicates that the maximum throughput is obtained when the retry rate is very short.

It could be argued that at the higher retry rates there would be more interference between the packets trying to set up, and thus the maximum throughput would be decreased. These results show that this is not the case, and the increased throughput amounts to the reduction in time in finding a path open when a connected packet completes.

Non-Buffered Network with Queue Manipulation

When using the non-buffered method it is possible to select which packet is to be transmitted next. In particular it is possible to select a different packet if the first selected packet is blocked, and by this method increase the throughput of

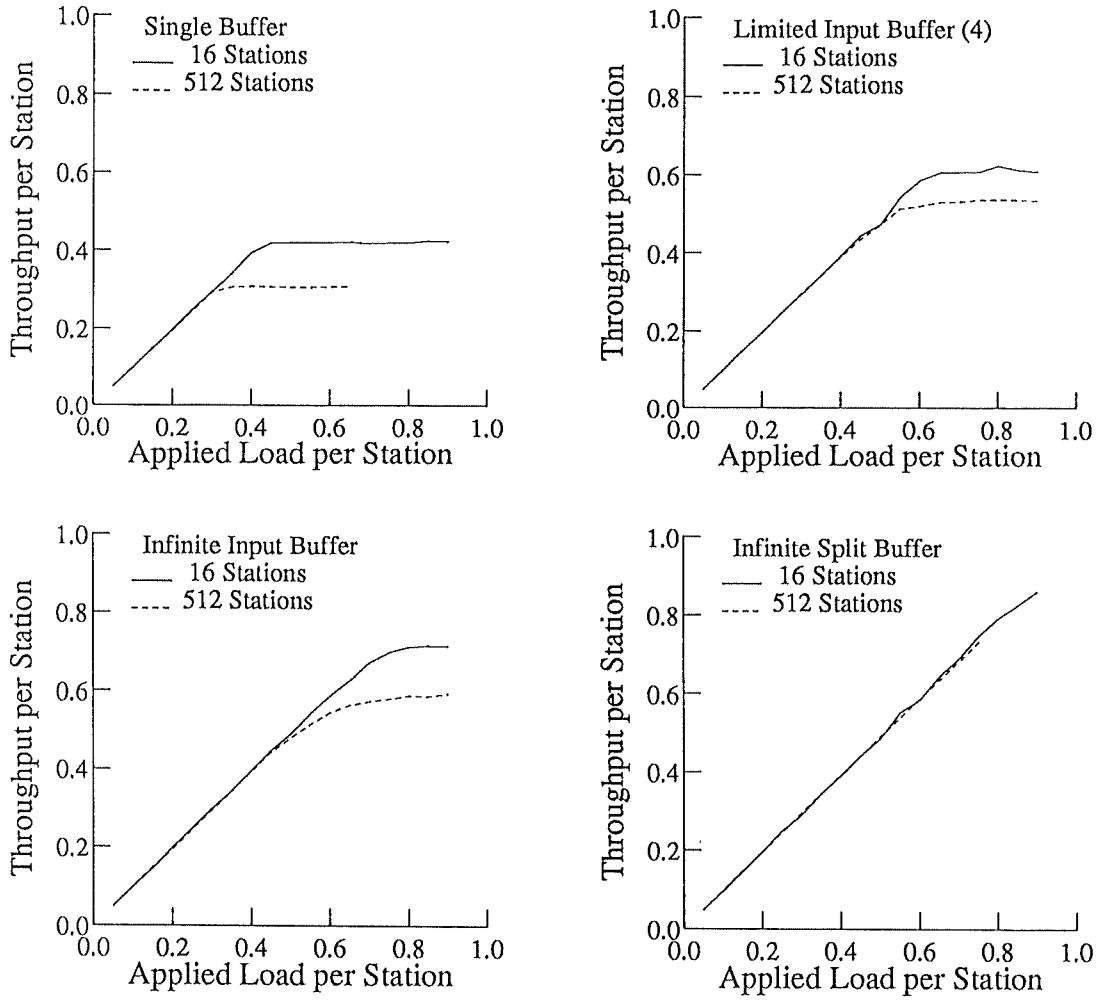


Figure 6.2: Throughput vs. Applied Load: Buffered Network

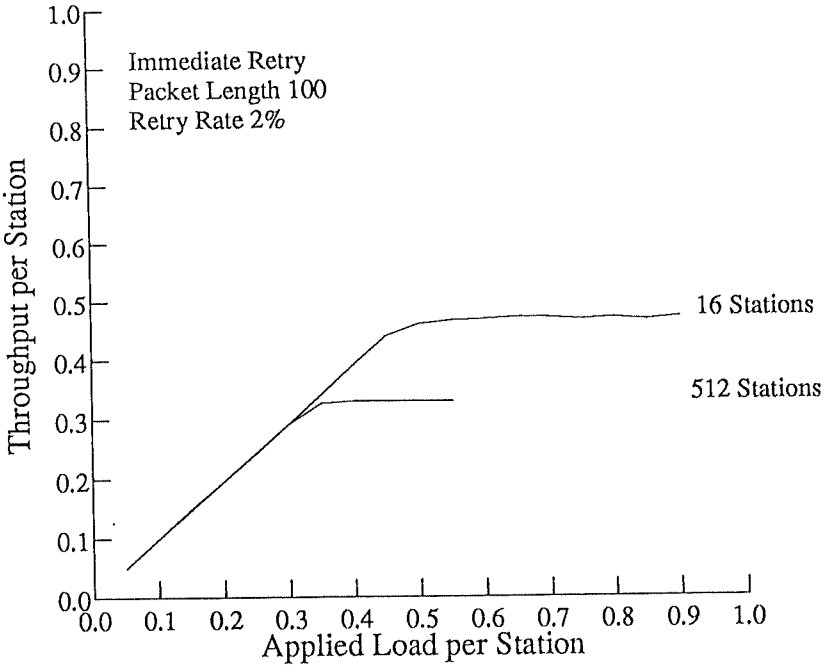


Figure 6.3: Throughput vs. Applied Load: Non-Buffered Network

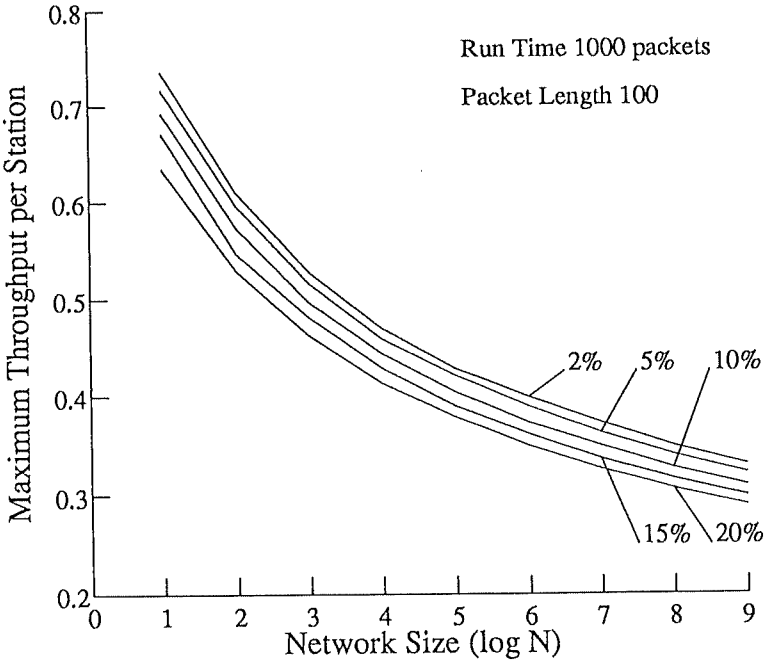


Figure 6.4: Maximum Throughput vs. Network Size for different retry rates: Non-Buffered Network with Immediate Retry

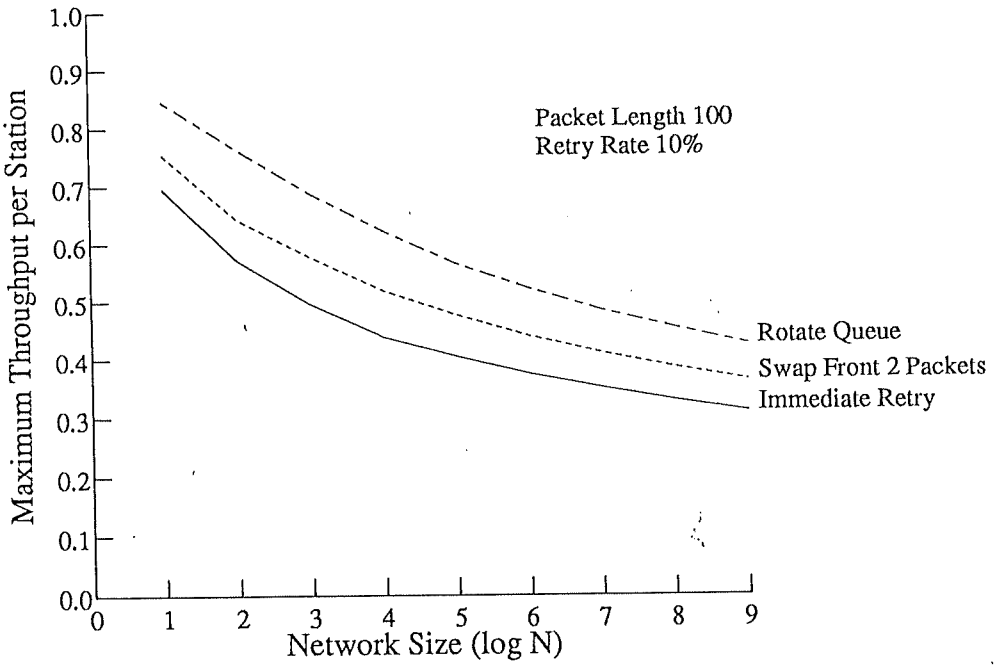


Figure 6.5: Maximum Throughput using Queue Manipulation

the network. This method will only work correctly if the packets in the queue are addressed to different stations.

Two methods of selecting a packet to retry were simulated. These were, to move the blocked packet to the end of the queue and try the second packet repeating this process until a transmission is successful, and, to alternately retry the first two packets.

Both methods give an increase in the maximum throughput of the network over the simpler scheme of continuously retrying of the first packet. Comparison of these three systems is given in Figure 6.5.

A faster retry rate for the queue rotating scheme would produce a higher maximum throughput, as there will always be at least one free path a packet can follow from an idle station. As the queue at each station will have many packets with a distribution of addresses, the station has just to find one that can be successfully transmitted and the faster it can retry the sooner it will find one. The other scheme will only have a slight improvement in line with the gain of the immediate retry scheme.

Figure 6.6 shows the delay relationship between these schemes at low loads. The vertical axis shows normalised delay. This delay is normalised with respect to

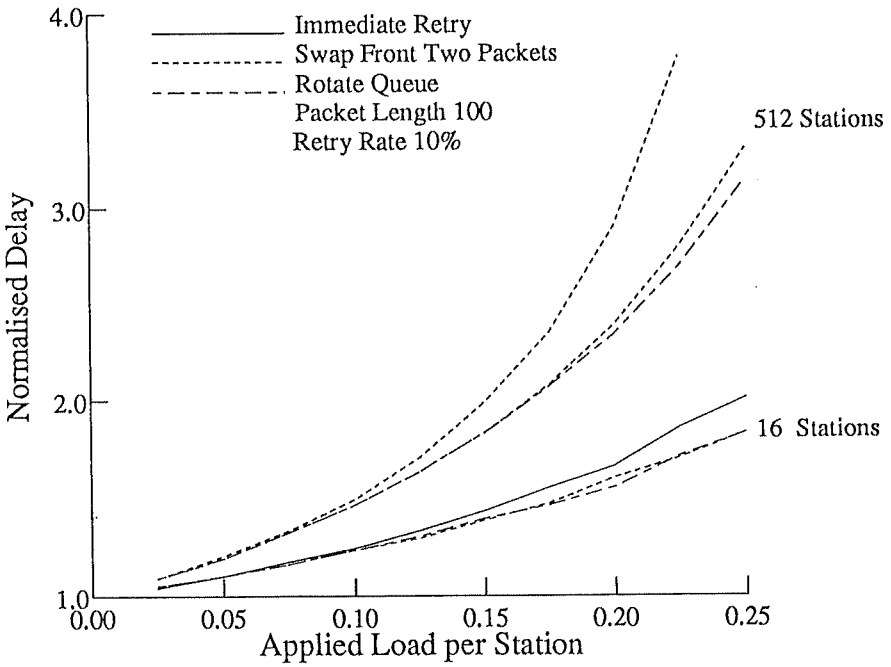


Figure 6.6: Delay at Low Loads using Queue Manipulation

the transmission time of the packet such that a packet that does not experience any blockage will have a delay of 1. This normalised delay does not take into account any delay in passing through the nodes which is assumed to be small compared to the packet length. This definition of normalised delay will be used for the rest of this chapter. In these results packets were generated randomly over a period, and, as the network was operating in the range of loads where it can deliver all packets in short time, there was rarely more than 1 packet on the queue. This forces these modes to operate in a similar fashion. It is only as the load increases and the queues have a number of packets to send that any improvement is shown.

If the stations were to operate in a burst mode where packets are generated in groups, for example in a multiprocessor system, the queue manipulation schemes can be used to great advantage. Figure 6.7 shows the delay characteristics for two systems operating in the burst mode. With 16 stations, each generating packets in groups of 4, both manipulation systems give a slightly better performance over the straight retry of the same packet. With 512 stations generating packets in groups of 9, the 25% load is actually approaching the maximum throughput of the network and there is now a considerable gain obtained by using the manipulation schemes. At the low end of the load, approaching 0%, the delay does not become

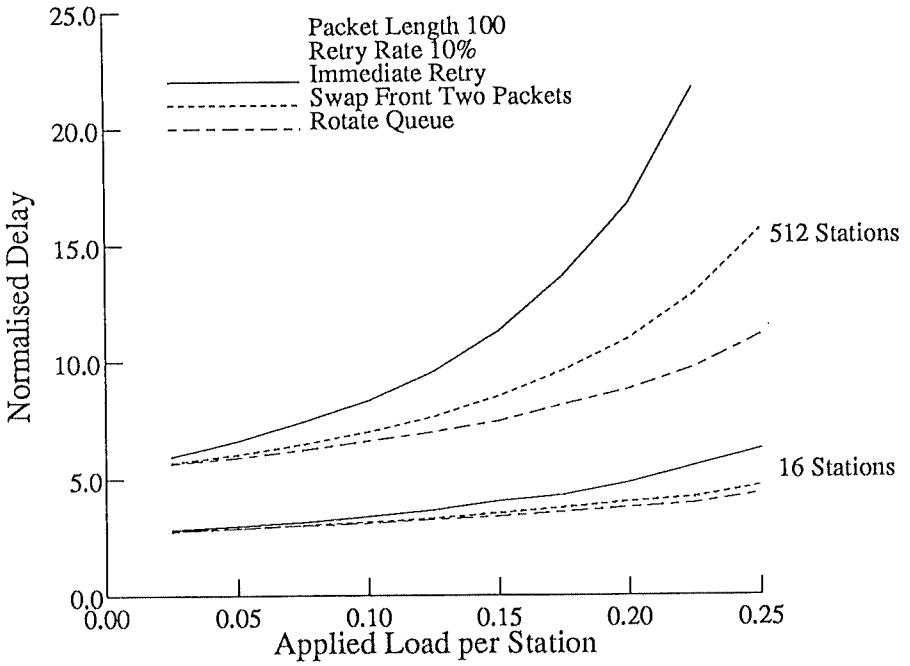


Figure 6.7: Delay for Burst Mode Operation and Queue Manipulation

1 as the packets still arrive at a station in groups and can only be transmitted one at a time.

Comparison of Non-Buffered Networks with and without Random Routing Stage

The addition of Random Routing nodes to the front of the network has the effect of allowing a packet to pass through the network on any of a number of routes. This does not increase the maximum throughput of the network, but at lower loads it means that a packet can possibly find a path that avoids another packet which would block its path in the simpler network.

Figure 6.8 shows the delay characteristics of the two networks at low utilisation. Even at a load of 25% from each station the delay on transmitting a packet is halved. As the load increases, packets will find that the other input to the random routing node is occupied, and they are restricted to a single path again resulting in the performance of the two networks converging.

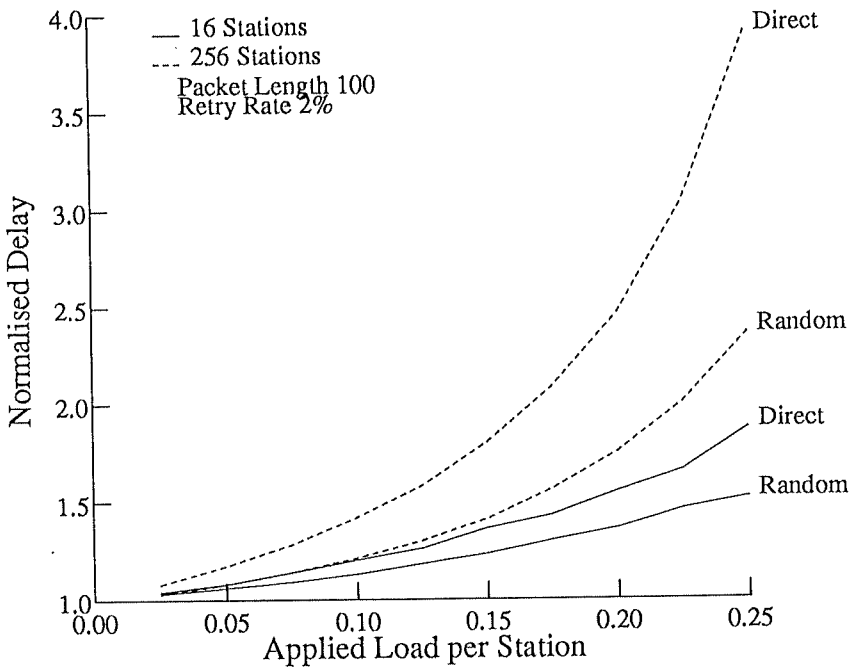


Figure 6.8: Delay of Non-Buffered Networks with and without Random Routing

Comparison of Buffered and Non-Buffered Networks at Low Loads

We have seen in the previous sections that the maximum throughput of a Delta network increases as the amount of buffering is increased, but with a side effect that the delay through the network also increases and becomes very large when the maximum is approached. At lower loads, up to 25% of the channel capacity per station, all the networks are capable of delivering the packets within a reasonable time.

Figures 6.9 and 6.10 compare the delay of the networks of 16 and 512 stations respectively at these low loads. Examination of the delays for the buffered network models shows that there is not much difference between the simple buffering schemes over this range of loads and that only the Infinite Split buffering scheme shows a significant improvement in performance.

The delay characteristic of the Non-Buffered scheme is found in general to be better than any of the buffering schemes in this throughput range. This is because when a packet is blocked in the buffered networks it must be completely buffered in the node before it can continue, meaning that if it is blocked only once the

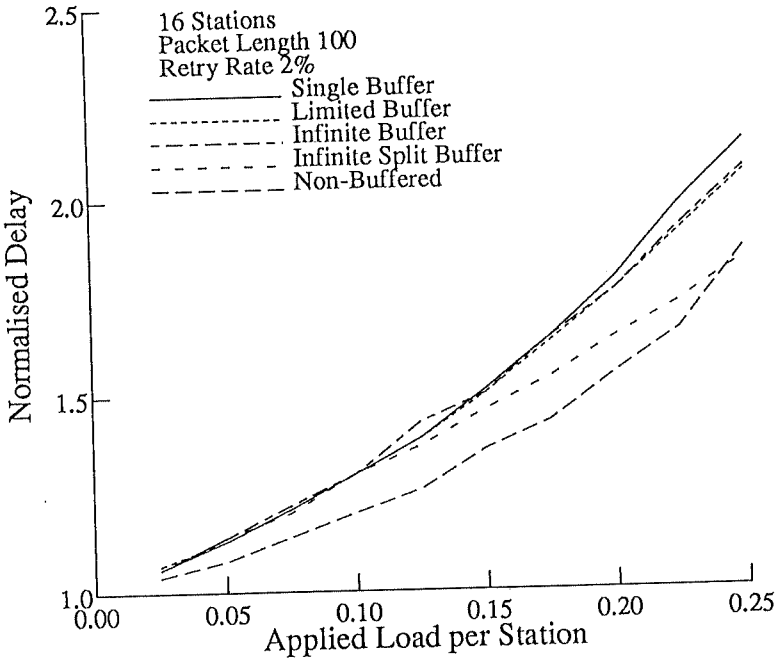


Figure 6.9: Normalised Delay at Low Loads: 16 Stations

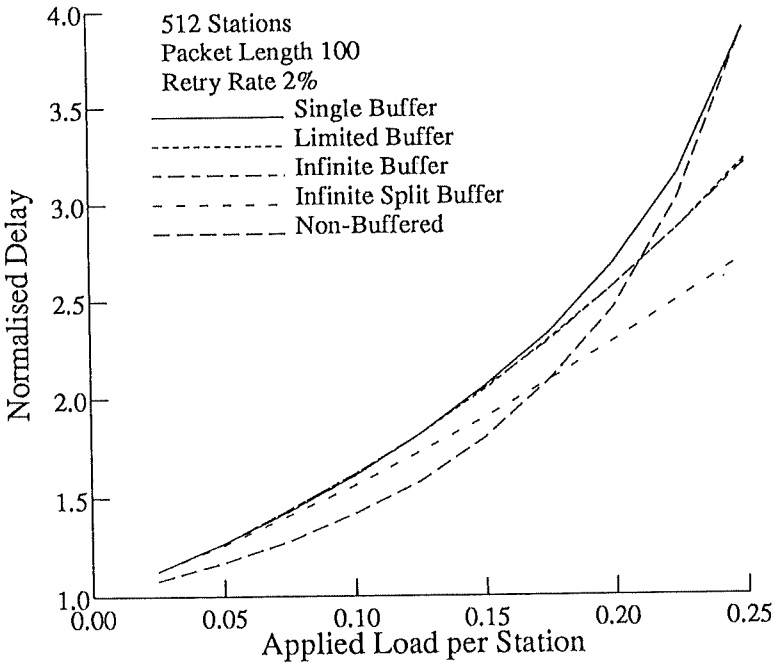


Figure 6.10: Normalised Delay at Low Loads: 512 Stations

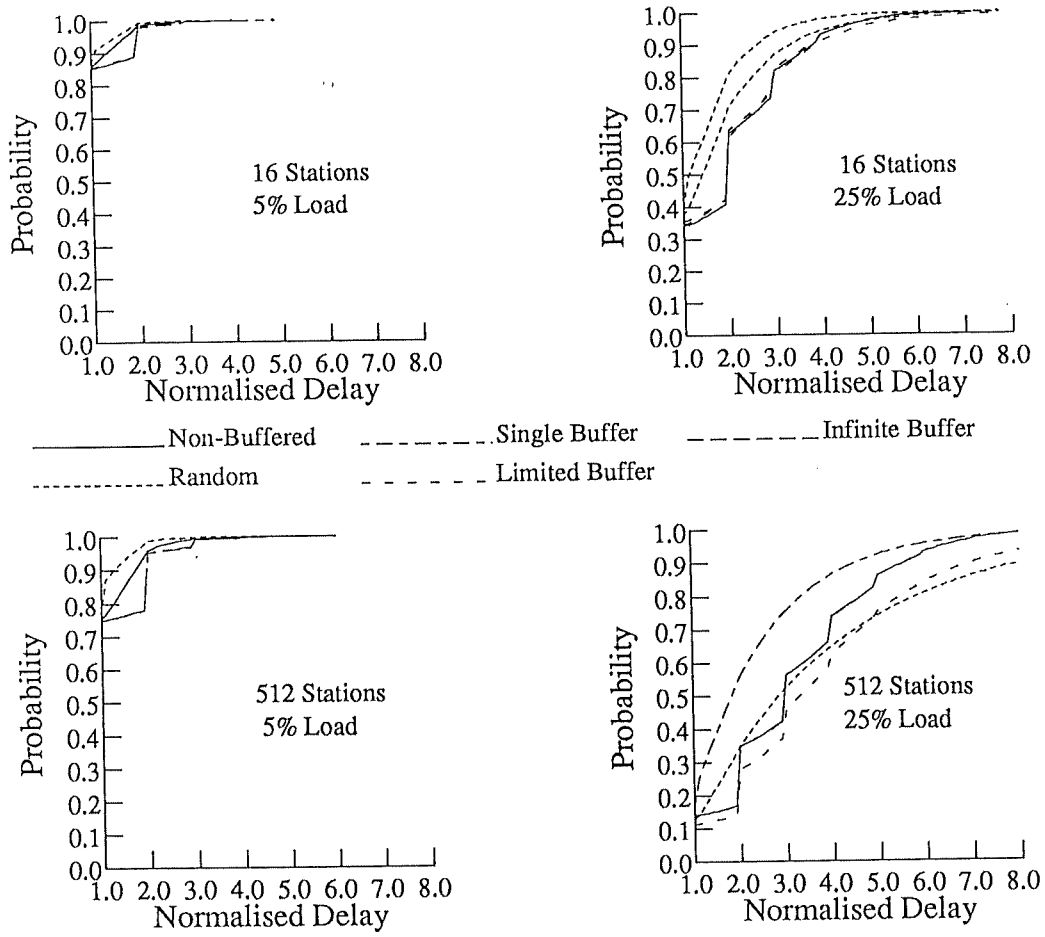


Figure 6.11: Cumulative Probability Functions for Packet Delays

additional delay will be 1 packet time. In the Non-Buffered scheme a blocked packet will need to wait an average of half a packet time for the blocking packet to complete before the blocked packet can continue.

The distribution of the packet delay is also an important quantity. The cumulative probability functions for the different systems are given in Figure 6.11.

Each of these graphs plot the probability of a packet being delivered within the specified period. For example a network of 16 stations running at 25% load would expect 90% of all packets to be delivered within 4 packet times.

The step shape of the buffered networks results from completely receiving the packet at a node where it is buffered before the next stage of the transmission can be started. Also at these levels of load the buffered networks are almost indistinguishable and the single buffered network is separated out only when operating at

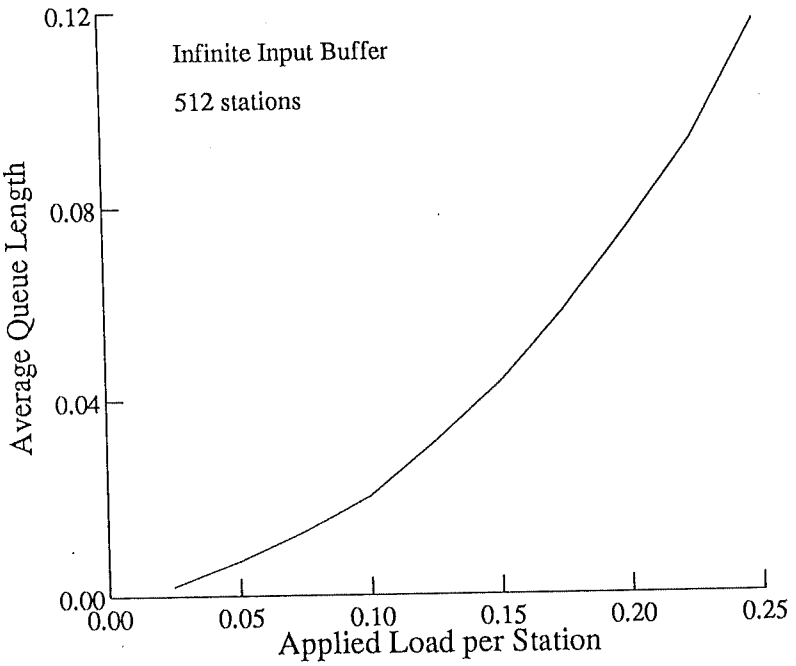


Figure 6.12: Average Node Queue Length for a Buffered Network

the higher loads with a large network. This would indicate that for this region of operation there is no point in having a buffer capable of holding more than one packet.

Another variable that is important to the buffered network is the number of packets that are held at any one time in a node buffer. A network of 512 stations operating in the range of 0% to 25% load has an average queue length as shown in Figure 6.12.

For the same system the maximum queue length was 6. A probability of 88.4% gave the queue of length 0, 11.4% of length 1 and the remaining 0.2% was divided amongst length 2, 3, 4, 5 and 6. This indicates that, for the load range considered here, there is very little difference between a buffer capable of holding 4 packets and a buffer capable of holding many more. This is also clear from the cumulative delay probabilities where the two sets of data completely overlap.

Comparison between Simple and Complete Simulations

The simulation model used for the majority of the simulations of non-buffered systems is simplified by removing the possibility of collisions between packets that are in the process of setting up a path. This is done by simply testing the desired

	Size	2	3	4	5	6	7	8
Complete	Thrput.	0.618	0.532	0.480	0.439	0.405	0.379	0.356
	Std.	0.030	0.011	0.015	0.013	0.012	0.013	0.011
Simple	Thrput.	0.616	0.536	0.479	0.437	0.406	0.380	0.358
	Std.	0.013	0.023	0.018	0.016	0.012	0.012	0.012

Table 6.1: Complete and Simple simulations with very high retry rate

	Size	2	3	4	5	6	7	8
Complete	Thrput.	0.570	0.484	0.431	0.392	0.357	0.330	0.307
	Std,	0.020	0.016	0.011	0.014	0.012	0.012	0.010
Simple	Thrput.	0.547	0.497	0.444	0.404	0.373	0.349	0.327
	Std.	0.015	0.016	0.013	0.013	0.013	0.012	0.011

Table 6.2: Complete and Simple simulations with 10% retry rate

path in an instantaneous manner and only allocating links when a complete path is available. As the packets are handled one at a time there can be no interference between the packets unless a connection is established. In the real system each packet moves one node at a time through the network and if two packets were attempting to setup, a path would allocate and hold intermediate links until they either completed the setup phase or were blocked by some other packet.

To compare the two simulation methods a second slower simulator was constructed to model in more detail the operation of the network. The new model allowed the early links to be held while the packet progressed through the network and was held for a period of 10 time units after a busy event, to simulate the tail end of the packet passing through the network, releasing the links. The results of a run of the different simulators are given in Tables 6.1 and 6.2. Each simulation was run at the maximum capacity of the network, the transmitter always having a packet ready to transmit as soon as the previous packet is completed. The model used is the immediate retry on busy model, and the simulations were run until one of the receivers had received 1000 packets.

In the first set of simulation results, the retry is very much smaller than the packet length, a packet length of 1000 time units with a mean retry rate of 2 time units. It can be seen from these results that the mean maximum throughput of the network for each network size is very close to equal and the deviation was always approximately 1%.

The second set of results relates to the network run with a retry rate of 10%. The packet length in this case was 100 units and the retry time 10 units. Now

the time taken for the packet to set up is a significant portion of the total packet length. For a network of size 8 the portion is 8%, which, when added on to the retry time, makes a larger difference in the throughput. The difference in the two methods for this simulation is 0.02 out of 0.30 which is about 6%, and is within the difference expected from the change in technique.

Both of these sets of simulation results indicate that the simplification to make the simulator run faster does not alter the final results significantly.

A network size of 9 has not been examined here as the simulation had taken over 10000 minutes of CPU time to execute without approaching the same conditions applied to the other networks. As the simulation would have run for many thousands more minutes it was decided that the time could be used more productively on other simulations.

Other System Parameters

A number of other system parameters have been investigated but not reported in detail in this chapter. These are the different network configurations, the use of different length packets and the use of random or alternating selection of paths and buffers.

The two network structures that were used for the simulations were the Perfect Shuffle and the Omega networks. In all cases the results for these networks were within 1% of each other and no trend inferring that one was better than the other was observed. Lawrie[24] has actually shown that these networks are topologically similar.

This simulation of a Non-Buffered Binary Routing Network made simplifying assumptions that result in a performance with very little dependence on the size of the packet. A system where random length packets are used does not show any significant difference to a network with fixed length packets. The obvious difference will come in the extra time spent in routing the larger number of smaller packets required to give the same applied load. It will be more pronounced as the systems grow in size due to the higher probability of collisions. This can be seen in the following section on Numerical Results which compares packet of 1000 units and 100 units with the same retry rate. The buffer networks do not have the same problem as there is no wasted bandwidth where a retry is required, so a network with variable length packet will have the same properties as one with fixed length packets.

Where two buffers in a node hold packets that require the same output two

methods of selection are possible. Either a buffer can be selected at random or a least recently used algorithm can be applied. In either case a system run for a long time shows no advantage in terms of throughput, but the maximum time a packet can spend waiting in the network is reduced for the nonrandom selection. For the random selection a packet could in theory remain in a buffer forever given the correct conditions but for the alternating selection method an upper bound on the delay could be determined. Similarly with the non-buffered network a random path selection or an alternating path selection within the first stage has no significant effect on the average delay of a packet.

Comparison with other Networks

As was mentioned earlier in this chapter it is not easy to compare a Binary Routing Network with other networks in the Local Area Network category. With the diverse topologies that a Binary Routing Network can use, many more packets can be handled compared to many networks that can only handle a single packet at a time. Even the networks such as Floodnet where a number of packets can be traversing the network at any given point in time would require a very regular structure to come near the throughput of a Binary Routing Network.

The cost of attaining a high level of throughput is the amount of logic that is required to implement the network. It is possible to implement simpler networks, mentioned in Chapter 4, but they have not been examined in this thesis. Similarly the complexity of other networks could be increased to improve their throughput to match that of a Binary Routing Network.

6.3 Numerical Solution

An approximate solution of the maximum throughput for a Non-Buffered Binary Routing Network can be achieved by using the technique described by Lee and Wu[26].

In their paper they describe three models of a network based on the baseline structure. The models assume that the packets are distributed uniformly across all addresses. The packets are generated at a rate determined by a fixed probability when the station is idle. The length of a packet data field is greater than the number of nodes the packet must pass through, and if two packets arrive at a node simultaneously then one will be selected and the other blocked.

In all they describe three models of operation. The *Regeneration* model assumes

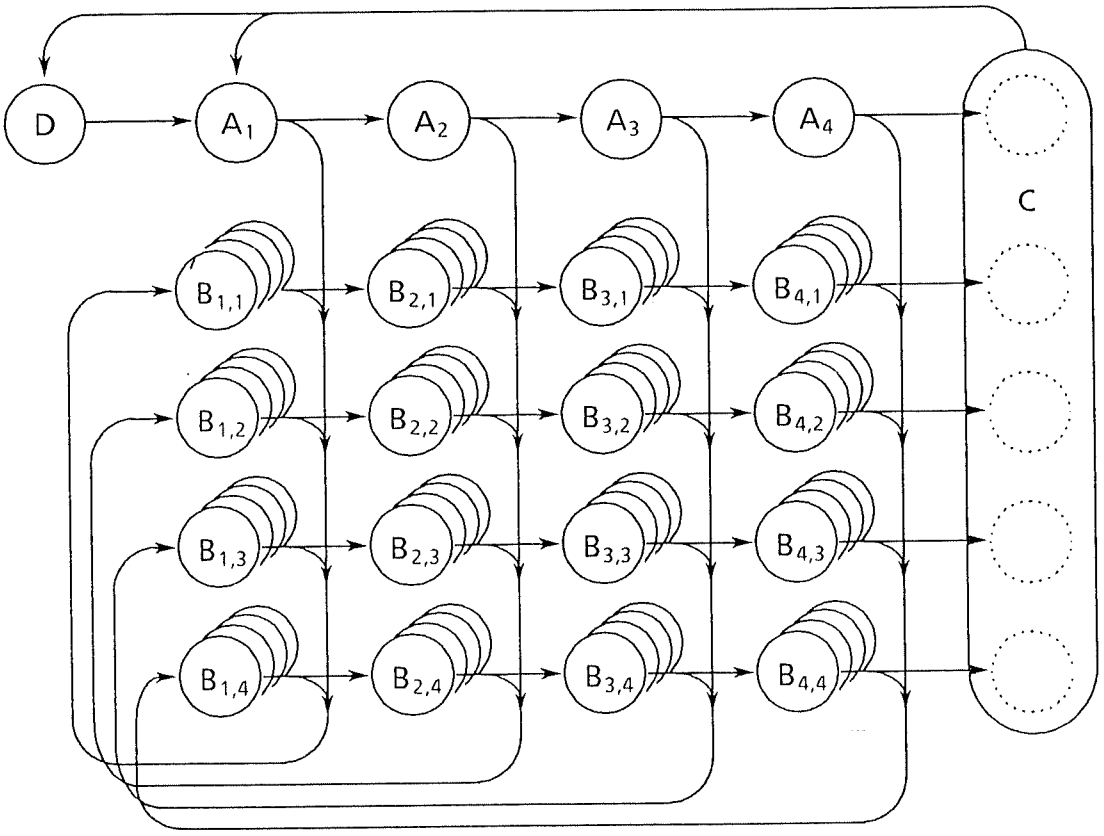


Figure 6.13: State Assignment in the Drop Model

that when a packet is blocked the transmitter replaces the packet with a different and independent packet which is immediately retried. The *Hold* model assumes that when a packet is blocked in a stage, then it waits at the input to that stage for the desired output to become free, holding all the links in the earlier stages on the network. In the third model, *Drop*, a blocked packet releases all the links it held and is immediately retried until it succeeds in connecting to the receiver. This model is the one that models closely the operation of the Non-Buffered Binary Routing Network.

The models all work by constructing a state transition diagram and calculating the probability that a packet is in one of the particular states. The final result,

the throughput of the network, can be calculated from the proportion of packets existing in the connected state. The state transition diagram for the Drop model is given in Figure 6.13. State D is the state for transmitters that are idle, i.e. there is no packet to transmit. States A_1 to A_n are the states that new packets to the network pass through before they are blocked for the first time. State C is the connected state. Once packets have succeeded in reaching this state they become connected and remain in this state for enough time for the data portion of the packet to be transmitted. When a packet is blocked it releases the links that it holds and, at the beginning of the next cycle, starts the process of setting up a path through the network. This time, as the packet has been blocked at least once, it enters the network through the B states. The B states are divided into sub-states to give the packet some history, in particular the stage in the network where the packet was blocked last. This is required because the retried packet is not independent of the rest of the network, i.e. the blocking packet may still be there. These sub-states are further subdivided by the blocking packet stage to facilitate the calculation of the probability of the blocking packet remaining in the network.

To determine the final distributions of probabilities amongst the different states, a set of rules is applied to the current set of state probabilities to determine the redistribution of probabilities for the next cycle. Packets in states A_i are assumed to be independent of all other packets in the network as are those in the B states that are not at the stage where they were last blocked. This independence means that the probability of a packet moving forward from the current stage to the next stage can be calculated from both the number of packets passing through the stage and the number of packets that are competing for an output in the stage. If the packet has returned to the stage where it was previously blocked, additional account must be made of the probability of the blocking packet still holding the output. Special cases exist for the first stage such as a packet blocked by another packet in stage 1. Similarly at the final stage of the network there are also some special cases.

The operation of a program to determine the values of the state probabilities is as follows: First, assign an initial probability to the states. Next, calculate the probability of the packets to move forward and backwards in the network. Then calculate the new values for all the probabilities in the different sub-states of stage 1. Finally, using the forward probabilities calculate new probabilities for the rest of the stages. The resulting set of probabilities may not add up to 1 and they

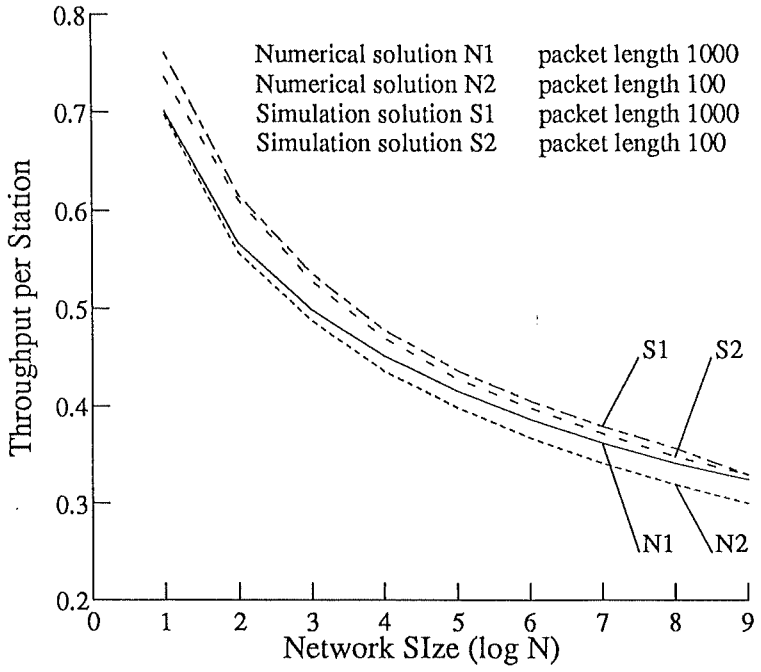


Figure 6.14: Numeric and Simulation Values for the maximum throughput of Non-Buffered BRNs

must all be normalised so that this condition holds. This operation is repeated many times until the states settle to fixed values.

If the solution is found for a system where there is always a packet to transmit, an arrival rate of 1.0, then the proportion of packets in the C stage is also the maximum throughput of the network under these conditions. This has been performed on a range of networks sizes with different packet lengths. A long packet length is used to reduce the effect on the size of the network and the shorter packet length to show the effect of packets attempting connection more often. These results are displayed in Figure 6.14. Along with these results are the results, for networks with similar parameters, found by the simulation method. This shows that the numerical solution gives a lower result for the maximum throughput than the simulation, which can be explained by the non-independence of packets in the B state causing the packet to collide with other packets which are themselves considered independent of the present packet. The simulation results themselves make a simplification that they do not collide with other packets during the routing phase. Removing this assumption reduces the maximum throughput found by the simulation bringing it closer to numerical solution.

6.4 Performance with Non-Uniform Loads

All the results reported in this chapter have been for systems where the average load from each station is equal and all the packets have been evenly distributed across the network. These conditions would not be found in a working system except perhaps a multiprocessor system.

Conditions that could be expected in the operation of a network are very hard to predict. These would include a few stations operating at high loads while the other stations operated at very low loads, for example a file server. Coupled with this there would be a non-uniform distribution of packets with many stations directing packet at a single station. Measurements of the distribution of an operating Ethernet system are reported by Shoch and Hupp[41].

A Binary Routing Network will continue to operate under abnormal conditions but the throughput may be reduced. If all stations are sending to a single station the throughput will be reduced to slightly below the bit rate of the network. A single heavy user cannot monopolise the network. There are many paths of which only one can be used by this user and where this path coincides with those required by other users the routing nodes will share the bandwidth fairly.

If the pattern of addresses is such that a few internal links of the network are used to carry the whole load, the throughput of the network will again be reduced. This condition can be alleviated by the addition of randomizing nodes to the front of the network. These nodes will have the effect of distributing the packets, making the applied load appear as though it was randomly distributed. As the number of randomizing nodes is increased, the effect on the network of fixed address pattern or heavy users will be reduced.

Only simulations on well behaved networks have been performed in this thesis. A recent study by Wu[50] has examined the effect of these conditions on a single buffered Banyan network without a cut-through mechanism. In this analysis the node model resolves contention by giving priority to one of the buffers. This differs from the simulation in this chapter where the buffers have equal probability of selection to resolve contention. The result of Wu's analysis shows that as the load applied to certain fixed paths increases the worst case throughput for stations generating uniform loads decreases linearly when the point-to-point load increase above 45%.

6.5 Summary

In this chapter the performance of various buffering schemes for Binary Routing Networks have been examined. These have included Single packet buffers, Limited packet buffers and Infinite packet buffers. The Non-Buffered Binary Routing network has also been examined and its performance compared with that of the buffered networks. These simulation results have been compared with results of similar networks from other sources to verify their accuracy.

It has been shown that the greatest throughput can be obtained by using a buffering scheme which splits the packets into one of two queues as they enter the node before they contend for their required output port. The buffers must be able to take all packets, resulting in nodes with a large amount of logic. By providing less buffer space the maximum throughput of the network also decreases until the state where there is a single buffer per input of each node. At this point changing to a network without buffering in the node can give a small increase in the maximum throughput with a large reduction in the amount of logic required to implement the node.

It has also been shown that for the network operating at low loads, away from the maximum throughput of the network, the non-buffered scheme has a lower delay characteristic which makes it desirable over the buffered schemes. It is also shown that the different buffering schemes are very much the same at low loads, and the addition of more buffer space does not have much effect on the delay characteristic. Similarly at low loads the addition of an extra routing stage, ostensibly to increase the fault-tolerance of the network, can dramatically reduce the delay of a packet by giving it the choice of two paths.

The range of load values that has been considered to be at the low end of the network performance is the range from 0% to 25% per station. Each station supplies data at up to 25% of the line bit rate. This can produce a total applied load far in excess of the network bit rate but at a level where the data can still be delivered within a reasonable time. For a 16 station network operating at 25% load the throughput is 4 times the network bit rate, and for a network of 512 stations it is 128 times the network bit rate. Such loads are not possible on a network with a single channel, e.g. Ethernet or Cambridge Ring, but to gain this capacity large amounts of routing logic is required.

In all the simulations carried out in this section the main assumption was that packets were evenly distributed over all the addresses and that they were all independent. This pattern of traffic is not what would be found in a general LAN

environment, where some stations would receive packets from a number of stations and some stations would send many packets to a number of other stations. In a less general system, in a multiprocessor environment such a distribution of packets could be obtained by the careful selection of stations for particular fragments of the program to be run.

For a general LAN similar to the Cambridge Distributed system where there are a lot of small systems providing simple services that are used by other services, it can be assumed that the station will have only one packet to transmit. The policy of immediate retry of the same packet in the non-buffered network is the mode that most stations will operate in. The multiprocessor environment can operate with stations generating packets for many stations and a queue scanning policy for the retransmission of blocked packets will be desirable.

Further work that should be carried out into the performance of these forms of networks includes investigating the performance of systems with limited buffering which discard packets when the buffer is full, to determine the minimum amount of buffers required to keep the number of lost packets below a given level at different network load factors.

Chapter 7

Implementation of a Non-Buffered Binary Routing Network

A prototype Binary Routing Network has been constructed to demonstrate the practical use of this form of routing network and to verify, as far as practical, that the network performance matches that predicted by the simulation results.

The non-buffered version of the network was chosen for construction for two main reasons. Firstly it was considered that the performance of the non-buffered system was effectively similar to that of the buffered systems when stations were offering light loads. Secondly it was decided that as the construction of the non-buffered routing node would be vastly simpler to that of a buffered node, and that any loss of performance could be made up by adding extra paths where high congestion was found to occur.

7.1 Packet Specification

The packet structure is fundamental to the design of the rest of the network. It must carry enough control information to allow the route to be followed in a simple manner, without the requirement for a large amount of the packet to arrive at a node before it can be directed to the next node in the path. It must also be flexible so that it does not constrain the size of the network to some fixed depth, or interfere with the higher level protocols by forcing the data to be rigidly structured.

To this end it was decided to use a packet format where the fields were arbitrarily expandable in multiples of a single unit of of 8 bits. As well, three extra bits are provided for use in maintenance of the system. Figure 7.1 shows the basic

1	SP	ROUTE FIELD	DATA FIELD	P	PED

Figure 7.1: BRN Field Layout

R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_C

Figure 7.2: Route Field

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_C

Figure 7.3: Data Field

structure of the packet.

The packet header contains two bits, the Start bit (S) which is always '1', and a Special Packet bit (SP) which if set indicates that the packet is to be directed to the Route Server. The Route Field is made up of a number of 9 bit units: 8 bits of route information and a Field Continue bit which, if set, indicates that the route field has not completed. The route field sub-unit is shown in Figure 7.2. The Data Field is made up of a number of 9 bit units, 8 bits of data and a Field Continue bit, which if set indicates that the data field is to continue. The data field sub-unit is shown in Figure 7.3. The Trailer consists of two bits, the Parity bit (P), which is set to generate an even parity, and the Parity Error Detected bit (PED), which is set by a node when a parity error is detected and passed on by the other nodes. All bits in the packet including the S and SP bits are used to generate the parity. The only exception is the PED which comes up after the parity bit.

The minimum packet is: the header, 1 route byte with '0' field continue, 1 data byte with '0' field continue and the trailer making a total of 22 bits.

There is no requirement for a preamble at the front of the packet as the network is synchronously clocked and the single start bit is ample for the circuit to detect the start of the packet.

Figure 7.4 is an example of a packet which has a 16 bit *route field* and a 32 bit *data field.*

There were a number of reasons for choosing an 8 bit unit, mostly to do with implementation. First the memory for use in the buffers of the terminus is 8 bits wide, as is the bus on the Z80 processor used to drive the terminus. Secondly the routing node requires logic to count the bits in each unit as they pass and choosing

1	SP							
R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	1
R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	0
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	1
D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	1
D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	D_{21}	D_{22}	D_{23}	1
D_{24}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}	D_{30}	D_{31}	0
P	PE							

Figure 7.4: Example packet with 16 route bits and 32 data bits

both the route field unit and the data field unit to be the same simplifies that logic. It would have been possible to use a larger unit size, for example 16 bits, but this would have just increased the amount of logic required for the implementation without any real gains in performance.

7.2 Routing Node

The following points are a minimum set of requirements to make up an operational routing node. As discussed in Chapter 4 it is possible to include many management functions into the node to help in the detection of errors. These functions were left out of this design because it was decided that they would only make the node more complex.

1. The node has two inputs and two outputs. The inputs are indistinguishable.
2. A packet can be passed from either input to either output. Two packets can pass through a node if they are going to different outputs.
3. The delay through the node is made as small as possible.
4. There is no buffering in the node.
5. The whole network is driven by a global clock.
6. Routing is based on the first bit of the Route Field. The Route Field of the packet rotates so that the next significant bit of the route is moved to the front.
7. In the event of the Special Packet bit in the packet header being set, direct the packet, via a prearranged route, to the Route Server. The direction is determined by a signal generated at the Route Server.
8. In the event of the desired output channel being already assigned, generate a Busy signal which is passed back up the appropriate link for the duration of the packet.

9. Pass back, to the appropriate input, a Busy signal arriving at an output channel.

10. Check and correct the packet parity. If the parity is found to be incorrect set the Parity Error Detected bit.

Control and Data Signals

The cable that connects the output of a node or transmitter to the input of a node or receiver carries 4 signals, 2 in the forward direction and 2 in the reverse direction.

The *Data* signal flows in the forward direction from the output of a node to the input of the next node along with a *Connected* signal which is used to enable the receiver when the cable is in place and the first node is powered on.

The *Busy* signal flows in the reverse direction to signal to the transmitter that the packet cannot reach its destination and its transmission should be stopped. Along with the Busy signal is the *Route Server* signal. This signal indicates that if a packet is to follow this link it will eventually arrive at a Route Server.

Clocking Restrictions

This implementation of the network will have a Global clock which will be distributed to all nodes and stations. This will mean that all signals arriving at the node will be constrained to a minimum phase delay made up of the gate and cable times and, provided that this delay is limited to half the bit time, reconstructing the signal at the input to the node is simple.

If the clocking was to be controlled by an independent clock in each transmitter then the signals arriving at the two inputs would have independent phase as well as independent frequency, requiring complex logic to resolve contention. In order to attain the correct frequency and phase either logic operating at a higher speed than the basic data rate is required or some form of Phase Locked Loop (PLL) is required. The PLL may require a sequence of *Preamble bits* to enable it to lock onto the incoming signal. This would increase the delay through the node as this preamble must also be sent ahead of the packet along the output link. The output link cannot be chosen until the first of the route bits has been received, which will now come much later after the first bits of the packet. Alternatively the clock signal could be passed with the data along a separate line. This also requires the use of a PLL to avoid the pulse shrinking problem cause by the asymmetrical propagation times when many gates are connected in series. As the packet is

delayed by a number of bits within the node a local oscillator is required to supply the clock as the packet leaves the node. If a PLL is used, then after N nodes there would be a number of bits, of O(N), or the packet will not be controlled by a fixed oscillator. Such a situation could result in the bit rate becoming unstable as the PLL attempted to lock onto non-existent signals. It would be feasible to have a fixed local oscillator which could takes over for the last 3 bits. Provided it was close to the frequency of the incoming data, the transmission would not be affected.

Restricting the network to run on a global clock does have some major disadvantages, in particular that the distance between stations and the nodes must be kept very small to avoid long propagation delays in distributing the clock and delivering the data. The significance of this problem will most likely diminish as, given that the network is not generally distributable, it will be built in clumps which are grouped in adjacent racks where it is feasible to have a global clock.

Line Conditioning

The Line Conditioning is performed by differential line drivers on all signals. In addition the Data line is only enabled after a time out period has expired which is initiated by the establishment of the Connected signal. This time out is to enable the signals to settle before the Input Processor attempts to search for the start of a packet.

The Input Processor

The function of the Input Processor is to examine the input line to determine the framing of an incoming packet, to extract from the packet the route information, and to produce a new modified packet to be sent out along the output line.

The Input Processor is constructed from a single PLA device, 82S105, a counter and two flip-flops. There are two Input Processors, A-IP and B-IP, which are identical in every respect except for processing the Route Server signal. The 82S105 functions as a finite state machine examining the bits as they come in from the input line. It is supplied with the current bit and the last two bits which are stored in the flip-flops. The counter is used to count the bits in each field, it is not possible to do this within the state machine as there are only 4 bits available for state information. The basic structure of the node is shown in Figure 7.5.

While the network is idle, the input line remains in the low state. The Input Processor waits for the first high level bit to arrive, the Start bit. As soon

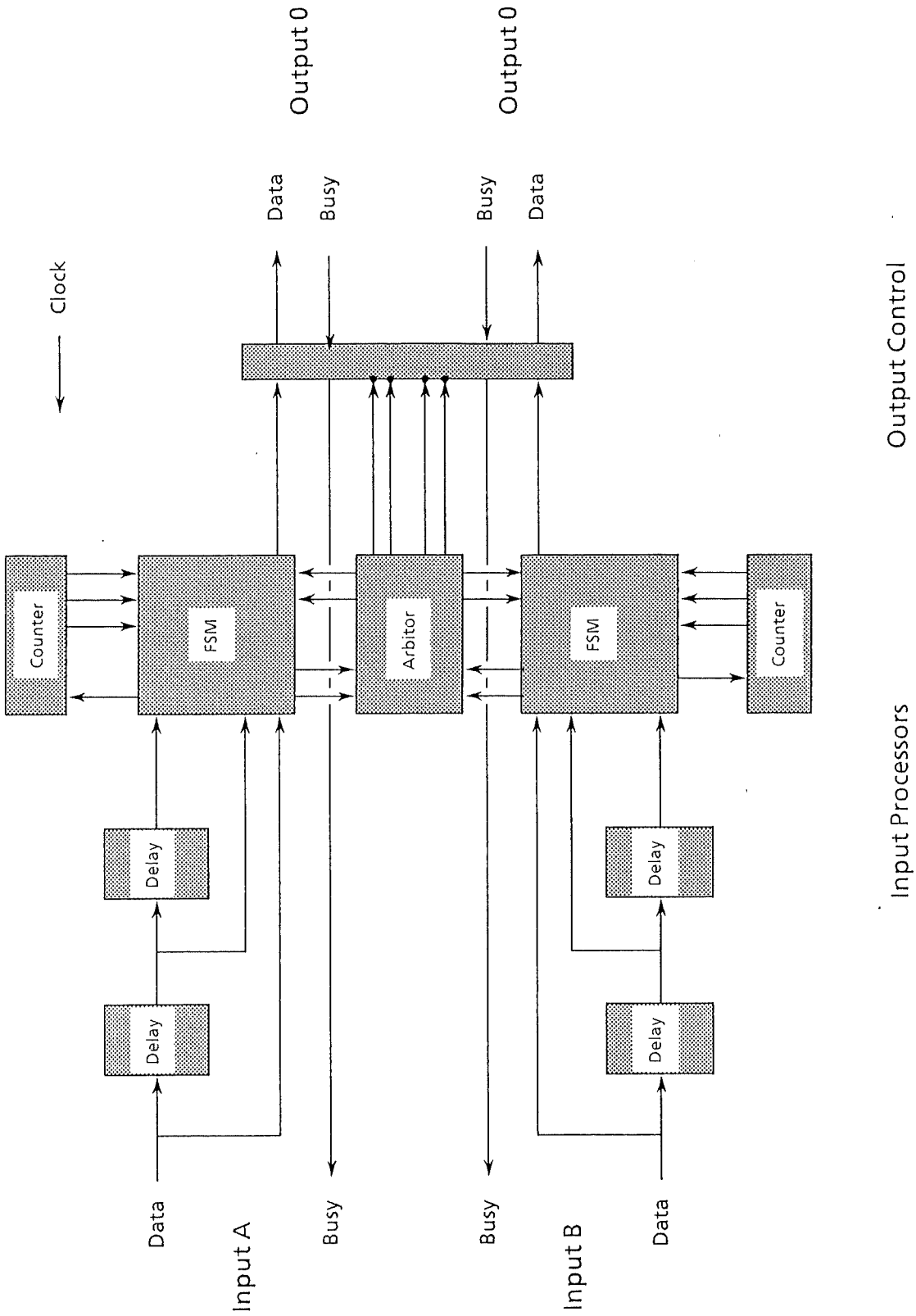


Figure 7.5: Structure of the Routing Node

as the Start bit of a packet is detected, the Input Processor sets 3 signals, the IN_PACKET signal, and the GOING_LEFT and GOING_RIGHT signals. The combination of these three signals being high indicates the reception of the start of a packet and requests the Output Arbitrator to start the arbitration operation.

The IN_PACKET signal is set to indicate that a packet is being processed. It remains set until the last bit of the packet has left the node. It is used by the Output Processor to gate the packet onto the select output link.

The two signals, GOING_LEFT and GOING_RIGHT, are set at this point to request both output links. This is required to increase the amount of time allowed between the start of arbitration and when the packet needs to start on the selected output link. If the Input Processor was to wait until the first route bit is received before arbitration starts, there would only be one half a bit time for arbitration and to select the output link and start the packet. Using the above method the Input Processor knows, when the route bit arrives, if the desired output is available and the only task is to select the link and start the packet.

The disadvantage of this method of arbitration is that if two packets arrive simultaneously or within a clock period then neither will be able to proceed. This limitation is unlikely to affect the performance on the network to any great extent as, in this implementation, the packets concerned will be retried with a delay dependent on the state of the station processor. In a system where the retry is automatically initiated by the hardware either a random backoff or the selection of one of the colliding packets would need to be considered.

When the first route bit arrives, the Input Processor drops the request signal for the unwanted output. If this output was not already allocated, as both outputs were idle, it is then freed for use by the other Input Processor. If the desired output is already allocated then the Input Processor clears both request signals, this causes a busy signal to be generated.

Once the initial processing of the packet is completed, the Input processor continues to monitor, and modify, the incoming packet. During this time the IN_PACKET and the request signals are unchanged.

When the packet completes its passage through the node, the IN_PACKET, and the request signal if there is one still active, is cleared and the Input Processor resumes waiting for the next packet.

Parity checking is performed by the Input Processor on all packets as they pass through the node. The resulting parity value is compared with the parity bit of the incoming packet and a corrected value appended to the outgoing packet. It is

possible for the outgoing packet to have a different parity to the incoming packet, as Special Packets have their route fields set to the direction taken to find the Route Server, thus two parities are generated.

If a parity error is detected then the parity error bit, PED, is set to indicate the error otherwise the incoming value is passed on. This allows errors detected in earlier stages to be flagged at the receiver. Also later stages will receive packets with correct parity which allows the station detecting the first error to flag the possibility of connections which may be failing. In this implementation there is no mechanism for signaling the error when detected at a node because there was a shortage of outputs on the PLAs necessitating the elimination of non-essential functions. During the testing of the system and the measurement of its performance there were no parity errors reported, which would be expected as the network was not run near its limit.

The Output Processor

The Output Processor is constructed on a programmable logic array, an 82S153. It has a number of operations to perform: 1) Arbitrate for use of the outputs. 2) Route packets to the selected output. 3) Generate the Busy signal when a packet is blocked. 4) Pass through a Busy signal from an output link to the input. These functions have mostly been described in the previous section so they will only be briefly described here.

Arbitration starts when both of the two request signals from an Input Processor are set high. If both Input Processors commence arbitration then neither will be granted an output. When the arbitration cycle completes, one of the two request signals is cleared, and at this point the Output Processor gates the data signal from the Input Processor to the selected output.

If both of the request signals are cleared, which only happens when the packet is blocked, the Output Processor generates a Busy signal which is sent back through the input link.

If, during the passage of a packet, a Busy signal is received from the output link then it is passed out to the input link.

All control signals generated by the Output Processor for an input are negated when the Input Processor negates the IN_PACKET.

Selecting the Route Server Direction

The 'Start of Day Problem', in a Binary Routing Network, is 'How do we find the Route Server when we don't know the route to the Route Server?'. As mentioned before, this is done by setting the Special Packet bit in the packet header and letting the nodes do the routing. The Special Packets follow a route built into the network. This route can be set by informing each node which way it is to direct the Special Packets. The simplest way is to have a switch in each node which is set when the network is constructed, but this would lead to more work when either the network was modified or the Route Server was moved.

Alternatively a dynamic path setup scheme can be employed which allows this path to be setup by the Route Server. This scheme requires a extra wire connecting each node input to its corresponding output, called the 'Route-Server-This-Way' (RS) line. The Route Server sets a bit in its receiver control register saying that it is willing to accept Special Packets. When this bit is set, the RS line from the receiver to the node is asserted.

When a node finds that one of its RS lines is set, it directs all Special Packets to that link. It also asserts the signal in its own two input links to indicate that packets arriving at this node can be directed to a Route Server. In this way the signal propagates through the network to the transmitters, which in the current implementation set a bit in a status register allowing the software to know if Special Packets it generates will arrive at the Route Server. There is also a LED which indicates this information.

It is possible to construct networks where not all transmitters have access to a Route Server, but in the Perfect Shuffle or Omega network structures this is not so. In a network where loops have been built in it is necessary to ensure that the RS signal does not feed back on itself as it will lock up and cause problems if the Route Server station is to be changed.

The Input Processors differ in their way of treating the RS signal but the overall result is the same. The A-IP uses the Right link RS signal. If the signal is set then the packet is directed to the right, otherwise it is directed to the left. The B-IP uses the Left link RS signal. If the signal is set then packet goes to the left, otherwise it goes to the right. If both RS signals are set then a packet from the A-IP input will go to the right while a packet from the B-IP input will go to the left. Either way the packets will find a Route Server. If neither RS signal is set then the reverse of the rule, for the both set case, is applied. Either way the packet will not find a route server and will be discarded when it reaches a receiver.

It is assumed that the Route Server will be able to generate a return route to the requesting station by examining its tables.

7.3 The Terminus

The interface between the host computer and the Binary Routing network is called the *Terminus*. Its function is to inject packets into the network with the correct packet framing and to receive packets from the network.

For the purposes of the experiment, the prototype station was constructed from existing Z80 processor cards with their connection to the Cambridge Ring. This facilitated the loading of the software into the processors for the debugging of the hardware and for the test programs to report on the performance of the network. Without this facility, development would have required many cycles of programming and erasing of EPROMS to get both the hardware and software to work together.

The Cambridge Z80 processor card is designed for operation as a small server on the Cambridge Ring and as such did not support any sophisticated hardware, all peripherals being operated in polled mode. In particular it was not possible to use DMA for reading or writing packets into the processor's memory.

The Terminus is broken up into 2 major parts, the transmitter and the receiver. This division is made at the obvious point of separation and makes each section independent, allowing a station to send packets to itself. Figure 7.6 shows the major parts of the Terminus.

Access to the station is via registers on the Z80's I/O bus. These registers contain control and status information as well as ports into the buffer memory. Each section has two buffers which operate as First In/First Out memories (FIFO). This allows one buffer to be used in transmitting or receiving while the other buffer is accessed by the host processor. The buffers are constructed with Static RAM memories and pointers, instead of standard FIFO devices. This allows larger buffers to be implemented but more importantly in the transmitter it gives the possibility of restarting the transmission without reloading the packet. Having to reload the packet would require large amounts of host time and, as it is expected that packets will be blocked some of the time, be a waste that needs to be avoided.

The control logic for each section is mostly implemented in a single programmable logic array each, an 82S105, with some extra logic to count bits and to drive the line signals.

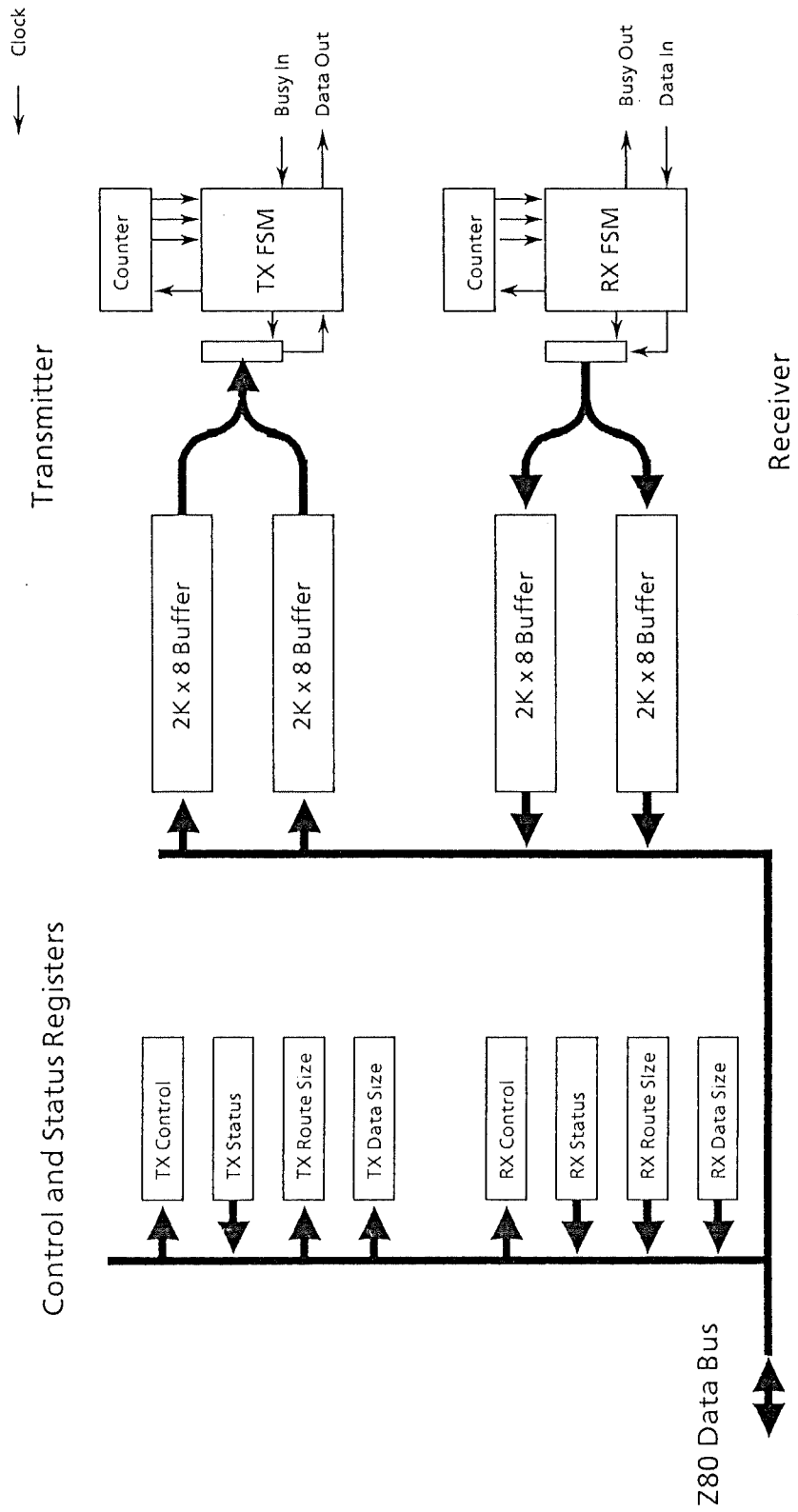


Figure 7.6: Terminus for Binary Routing Network

The operation of the station is simple. To transmit a packet the host first selects the buffer to use and writes the packet into the transmitter's data register which is a port into the selected buffer. The count registers, which define the size of the route and data portions of the packet are set, and the transmitter is told to start sending. If during the sending the packet is blocked, the transmitter truncates the packets, and interrupts the host indicating that the packet was blocked. The packet is truncated by transmitting data up to the end of the next subunit and marking it as the end of the packet and following with the parity check bits. If the route field was still being transmitted, then that route subunit is completed and one data subunit is sent. To retry the packet all the host needs to do is tell the transmitter to go again, since each of the sizes and the packet data are remembered. This makes the retransmission simple but does force a delay between the end of one transmission and the start of the next, limiting the maximum throughput by the time taken to enter the interrupt routine. While a packet is being sent the other buffer may be made ready for the next transmission and it may be possible if the blocking of one packet is persistent for the second packet to be sent first. This should only be performed if it is going to a different destination, as it could lead to out of sequence packets.

Reception of a packet is just as simple. When the station wishes to receive a packet it informs the receiver, indicating which buffer to place the packet in. When a packet arrives the host is interrupted, and copies of the size of the route and data fields are placed in the interface registers. The host may request the reception of a second packet by issuing a new command to the receiver indicating the second buffer. As the buffers are designed as FIFOs the host must first remove the route bytes from the front of the packet before the data can be removed. This data is accessed through a received data register in the hosts I/O bus.

While the host station is powered on, the receiver is always monitoring the incoming data line. If a packet starts arriving before the receiver has been enabled by the host the receiver generates a busy signal. If such a blocked packet is arriving when the receiver is enabled, the receiver waits for the completion of the packet before accepting any data. The receiver will also become disabled immediately after the reception of a packet so that subsequent packets will be signalled with a busy signal and will not overwrite the previously received packet.

The only time when a possible misinterpretation of incoming data can be made is when the station is powered on. This period is covered by the node connected to the receiver blocking packets for a time period after the station is powered on

which allows the station to perform a power up reset and the receiver to enter the wait state.

7.4 Performance of the Prototype

From this prototype system it has been possible to take some measurements of the performance of the network. In particular it is possible to measure the maximum throughput of the network under the Immediate Retry and Swap policies.

To make the hardware run at the maximum rate a single packet is repeatedly transmitted both when the packet is blocked and when a successful transmission has completed. In the second case the packet has a new random route inserted but the data remains the same. Using this method there is only a very small amount of wasted time between retry attempts.

The packet length was set to 2040 bytes, just short of the maximum, one byte for the route information and the rest for the data. The transmitter could send this packet in 3.7mS. These times were measured using an oscilloscope. Packets of 2040 bytes of 9 bits, to take into account the continuation bit, have 18364 bits in total which, when transmitted at 5MHz, should take 3.673mS, giving a maximum possible packet rate of 272.2 packets per second.

To measure the average number of packets transmitted, samples over a period of one minute were collected for networks of 2 stations and 4 stations. The time between packets was measured at 0.51ms, although a small number were much shorter. This time was mostly taken up by the software polling the interface and managing the ring byte streams to the terminal. This retry rate is approximately 13% of the packet time.

Two retry schemes were measured, the Immediate Retry and the Swap two packets shemes. Their average measured throughput is plotted in Figure 7.7 along with the simulation results for the same systems. As can be seen the measured figures are very similar to the simulation results. Unfortunately a larger network would have given more significant results but the resources of the experiment did not allow the construction of more stations and routing nodes.

7.5 VLSI Implementation

To build a Binary Routing Network with a large number of stations would require implementation in VLSI to keep down the amount of logic required to implement the many routing nodes.

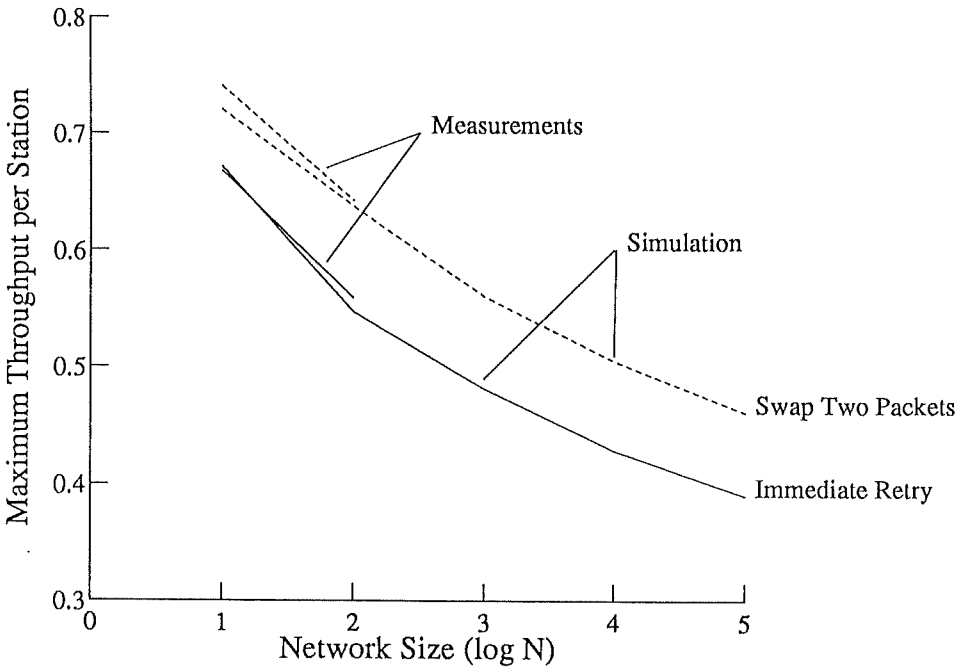


Figure 7.7: Comparison of Simulation and Measurement Results

Each routing node described in this required 2 82S105 PLA and a single 82S153 PLA. The largest amount of logic by far is in the finite state machine used in the input processor. An equivalent VLSI layout has been designed to implement the logic of the 82S105s. This structure was 491×414 lambda when constructed from NMOS using the Mead and Conway[29] design techniques. This gives the structure an area approximately $1mm^2$, using current implementation techniques which have lambda set to 2 microns, and allow a chip area of approximately $25mm^2$.

The design of the stations transmitter and receiver has also been made around the 82S105 PLA. There is a small amount of extra logic around the PLA, but this consists of only a counter and a shift register.

It is thus possible to construct on a single device enough logic to implement a 4×4 routing node as well as a transmitter and receiver.

The above discussion assumes that automatic tools are used to design the major portions of the logic. These tools use very conservative layout rules to ensure that technology violations do not occur. By further optimization of the layout and using up to date technology with smaller physical dimensions a reduction in size of approximately 10 times could be made, enabling even larger switching structures to be constructed. Alternatively additional functions could be implemented to

make the diagnosis of the system automatic.

7.6 Discussion

In this chapter the design of a simple Binary Routing Network has been outlined. A number of system restrictions have been applied to make the components of the network easier to construct and debug. The main restriction, global clocking, would in any event be applied to the construction of a switching centre, where many switching nodes would be located in a very small space.

Although there were many functions left out of the routing node design to simplify its design, an implementation in VLSI would not be similarly constrained. It would thus be practical to implement a 4×4 or even an 8×8 node in a single device.

The operation of this network has come up to expectations. It was simple to implement, not using any complex technology, and it is easy to operate. Further work in the network is required to measure the delay characteristics at low loads and to implement the random routing nodes to make a fault tolerant network.

A design for a large scale network has not been considered here. It would be possible to take the station board and connect two sets back to back to create a buffering module. A small amount of extra control logic would be required to automatically start the transmitter and receiver and to manage the buffer allocation. It would not be impractical to implement this logic on a VLSI device along with the rest of the logic described in this chapter.

Chapter 8

Conclusion

Local Area Networks have been used to connect computers together for many years. These networks have been used to provide services to a group of locally situated computers. Mechanisms for the shared use of a single channel have been developed to enable reliable, fair and cheap communications between any of the computers connected to the network.

Developments in computer technology in recent years have changed the accent on the Local Area Network from providing a simple communications service to forming an integral part of the computer system. These systems require large amounts of bandwidth with low delay to ensure that the system functions at efficient levels.

Developments in the field of telecommunications are applying the results from computer networks to provide new systems and services. Circuit Switching systems that have supported the telephone system for many years are being replaced with digital packet switching systems to make efficient use of the physical connections. New services, for example video and documents, require larger levels of bandwidth. For some services, e.g. voice and video, low delay is critical. Data must flow in and out of these services at a fixed rate and any loss caused by a buffer underrun or overrun will appear as a reduction in the quality of the service.

All these developments require networks which can provide higher throughput with low delay. Local Area Networks as they were originally developed cannot supply this demand and keep their original structures without large increases in cost. A number of issues in Local Area Networks limits the amount of data that the network can carry. The use of a single transmission medium immediately restricts the maximum rate at which data can be passed. The procedure of waiting for the return of the data to the transmitter in a ring system creates a delay which is dependent on the length of the network. For a long network where cable

delays are dominant increasing the bit rate has little effect on the delay and may only slightly improve the point-to-point throughput of the network. Splitting the network into a number of independent sub-networks which are connected by a bridge, will give an improved performance to stations communicating locally, but may hinder internetwork communications.

Increasing the bit rate of the network increases the cost of the attachments and of the interconnection of the network components. Current technology can increase the bit rate of current networks by a factor of 10, from 10Mb/s to 100Mb/s. This increase can only marginally cover the new requirements.

Topologies that allow many packets to be processed in parallel are required to give the networks the high performance capabilities. Proposals for Local Area Networks with multiple paths connecting switching units have been made. These proposals have been designed to implement networks with irregular topologies and have relied on flooding techniques to route packets from source to destination. The designs that have been studied have limited the number of simultaneous transmissions. Using the techniques described in these proposals a reliable network can be constructed. The flooding technique will find the shortest available path to the destination and this will bypass congested and faulty links in the network.

Interconnection networks designed for multiprocessor systems are capable of passing many packets simultaneously. These networks consist of a set of interconnected sorting nodes. The interconnection between nodes consists of a number of control lines and a wide bus. Each sorting (routing) node examines the destination address contained within the packet header and directs the packet to one of the node's outputs. Packets can be blocked in a stage if more than one packet requires the same output, which results in one of the packets remaining in its buffer until the next cycle. This process is applied many times, either by repeatedly passing the packet through the same network which feeds back on itself or by passing through a number of identical stages connected in series. The sorting nodes have no knowledge of the network topology and only use the information contained within the packet and the state of the nodes itself. Each node examines a different bit of the route field depending on the node's position in the network. Packets are clocked through the network one stage at a time. The interconnection networks are designed so that the destination address is used as the route information. This way the controlling procedures for the network can be kept simple. Other topologies that have different address/routing mappings are also possible.

Binary Routing Networks use structures and procedures similar to the multi-

processor Interconnection Networks to produce a network that is capable of delivering many packets in parallel and provide the high throughput required for the new network applications. They are constructed from interconnected routing nodes which direct packets through the network along a route carried in the header of the packet. Various topologies are available for constructing Binary Routing Networks. The interconnections can be any form, from a single data wire to a wide bus, though the lower the number of wires the smaller will be the cost of constructing the network. They can be connected to form rings, trees and busses that resemble the structures used in other Local Area Networks or they can be connected in structures that are derived from interconnection networks which use regular patterns. The irregular network structures will require more management as small changes to the interconnection pattern, for example inserting a new station, may change the route between many stations.

The simplest node structure is constructed with two inputs and two outputs, though other designs are possible. The routing nodes have no state information about the network and only deal with packets that are presented to their inputs and the information that they carry. A single bit in the route field of the packet is examined to select the desired output. As the packet passes through a node the route field is modified to remove the bit just used from the front of the packet. This allows the next node in the network to also examine the first bit of the route field. Consequently the routing nodes can be identically constructed. With simple operating procedures, nodes do not require the intelligent control circuits associated with Wide Area packet switching networks and can be constructed with little logic.

Provided that the packet's path is clear, there will be little delay in the nodes. A small delay is required to receive the first route bit of the packet before transmission on the selected output can start. This form of operation, known as cut-through, was first introduced in to the Wide Area Networks to reduce the delay as a packet passed through a switching node. If the desired route is not clear then the packet must be held up.

This form of network can be considered as a Fast Circuit Switch. Provided the path is clear, a circuit is set up by the front of the packet and is broken down when the tail end of the packet passes. Alternatively it can be considered as a Fast Packet Switch. In either case a Binary Routing Network is capable of passing many packets simultaneously to produce a high total through put, while implementing the network in standard technology which does not need to be run at its upper

limits of performance. For example a network of 512 stations, connected with a Perfect Shuffle, transmitting packets at random to other stations, can deliver data at a total rate of 150 times the basic bit rate of the network. For a network running at 10Mb/s this means packet switching at 1.5Gb/s, with other address pattern combinations allowing even more data to pass. To operate at this level though, each station on the network would only be able to deliver data at an average of 3Mb/s.

Binary Routing Networks can be used in the local area environment and can be expanded to cover larger areas that would be considered to constitute a Wide Area Network. As with Wide Area Networks switching centres would be established where a number of stations would be clustered on a local Binary Routing Network. Interconnecting these centres would be medium speed channels. Packets would then have their route set for local or global routing. Local routing would take the packet directly to the destination in the local cluster, while globally routed packet would be directed towards a link that would take them to the correct switching centre. Packets for either areas would be treated in the same way and stations need only know that replies may take longer from the more remote destinations.

Earlier designs for Binary Routing Networks, and similar switching systems, have all relied on the switching nodes to buffer packets when a transmission is blocked. Various buffering schemes can be chosen and by selecting a suitable buffer size the network can be made to perform almost as well as a cross-point switch. Buffered systems however, suffer from a number of problems. Firstly, for a large network a vast amount of total buffer space needs to be provided. Although modern integrated circuit technology has reduced the cost of large memories, the memory costs will still be significant. Secondly, stations that are not powered on, or are just not functioning, can cause the network to fill up with packets that cannot be delivered. Mechanisms need to be designed to allow this network resource to be reallocated, while still enabling a station to suspend reception while it processes data it already has.

The design of a Binary Routing Network which does not use buffers in the routing nodes has been proposed in this thesis. By removing the buffers, the cost of constructing the node is again reduced and this makes integration simpler and allows for a number of nodes to be constructed on a single integrated circuit. Secondly, the problems associated with networks of buffers are eliminated, as a packet which cannot get through the network is removed and retried at a later time when it may possibly be discarded altogether if blockage occurs again. Thirdly,

any restriction on the length of a packet made to ensure that it could fit into the nodes buffers can now be lifted and the only restriction is determined by the buffer space provided in the transmitters and receivers.

The cost of removing the buffers from the network is to reduce the maximum throughput of the network, but it results in an improvement in the delay characteristic at lower loads. The throughput of a simpler Non-Buffered Binary Routing Network is lower than for the majority of the proposed buffering schemes. At loads up to approximately 20% per station, the non-buffered network is capable of delivering the packets with less delay when compared to the buffered networks operating with cut-through.

Another problem of a non-buffered network is that there is no upper bound on the maximum delay that any particular packet will experience. As each packet is transmitted independently of every other packet in the network, a packet may collide continuously with many other packets, though for the loads considered in this thesis most packets are delivered within a small number of packet times. In a buffered system the delay can be bounded if the selection of packets from the buffers within the node occurs in a fixed pattern.

The reduction in maximum throughput can be offset by operating the network in different modes and by making modifications to the network structure. The simplest scheme that can be used on a non-buffered network is to continue to retry the same transmission until the packet is delivered. In the cases where more packets are available to transmit, an alternative scheme is to try different packets. In a network constructed from a perfect shuffle, there will always be an available path that some packet could follow and thus trying many different packets will improve the performance of the network. Alternatively, the addition of a random routing stage, which can transparently select different paths through the network can, in certain cases, also improve both the maximum throughput and the delay characteristic of the network. This change can also make the network more tolerent to network failures as the different paths will have few common components.

Binary Routing Networks will not replace the simpler Local Area Networks where a single cable can be run from room to room and stations can be attached by a simple tap into the cable. Such Binary Routing Networks can be constructed but they require much more management when a modification is made to the network as each addition or removal of a station from the network will upset the addressing for all other stations in the network. For complex structures the Binary Routing Network will require more cabling or result in a star network structure

with a central switching unit, with an associated increase cost for laying the cables.

Binary Routing Networks will find a place in larger switching units where data is already concentrated at a single point and is required to be distributed amongst many consumers. Apart from being able to process data at high rates, the concentration of switching nodes allows for further reduction in the logic to implement the network. Expensive logic to perform clock recovery and line conditioning can be removed from all connections within the switching unit. With large groups of nodes, redundant paths can be built into the network to provide for fault tolerant operation of the network which can be performed transparently to the user.

8.1 Further Work

Route Server

An area that still requires an amount of work is the operation of the Route Server. It has so far been assumed that the Route Server is capable of calculating a route through the network for any two stations. This requires the Route Server to have a complete map of the network and is the cause of the varying levels of management required to maintain the system.

The most desirable property of the Route Server would be for it to maintain the route table itself. This would allow any changes in the network configuration to be immediately, or as soon as practicable, reflected in the route table. There has been a lot of research into the management of route tables in wide area packet switching networks, but this is related more to directing packets via routes bypassing failed links or congested regions in the network.

For a Binary Routing Network the switching fabric of the network will hopefully be reliable and the Route Server would only be required to determine the structure of the network and to locate different services, when the system is powered on. This would necessitate the service sending packets through the network to locate stations or for stations that have just powered up sending packets to the Route Server. Some mechanisms and algorithms would need to be devised to ensure that such operations were reliable.

Buffering

One of the restrictions placed on the buffering methods examined in this thesis is that a packet can either pass through the node without being buffered or it

must be completely buffered before it can proceed. This restriction was imposed to remove the problems in clock generation and synchronisation.

For a network structure, as has been proposed, with global clocking, there is no longer a problem with the clocks as data will enter and leave the buffer at exactly the same rate. This arrangement will allow packets to start leaving the node before they have been completely received, without the need for complex buffering schemes. The new mode of operation should give an improvement in the performance of the network.

Flooding

An alternative to the use of a Route Server in establishing a route between two stations is the use of a flooding technique.

By using a flooding technique, it may be possible to do away with the Route Server. The special packet indicator would then become a flood indicator, and packets with this bit set would flood all the accessible network. Nodes that received a packet with this bit set would pass the packet out all the available outputs, with the appropriate bit inserted into the route field as it is rotated so that the receiving station may determine the route the packet took. A similar procedure to that used in Floodnet would allow the transmitter to know if the packet was not delivered to the addressed station and reschedule it for a later attempt. Once a path has been set up, further packets could use the normal routing techniques.

This form of routing would be most useful in irregular structures. For a network where the forward address was the same as the reverse address only one flooding phase would be required to set up a communications channel between two stations, but where this address structure was not available two flooding phases would be required. By caching the known routes locally, the route server function would be distributed to all stations.

Alternatively, the route server could use the flooding technique to locate services and stations that it did not know about, making its operation more dynamic.

8.2 An Application for a Binary Routing Network

As discussed in the previous chapters, a Binary Routing Network can operate as a switching centre. This has a number of advantages. Firstly the whole system can be globally clocked and secondly it makes the routing nodes simpler to implement.

Also, when the routing nodes are grouped together regular structures can be used to give high performance.

An application has arisen which requires a number of channels, carrying packetised data, to be interconnected with a switching unit. The Project Unison[45] is investigating using high speed Telecom channels to interconnect Local Area Networks in many locations across England. In the first instance these networks will be Cambridge Fast Rings and the high speed channels will run at 2Mb/s. The interconnection structure will be a star network for reasons of cost, but there is no reason why a partially or fully connected structure could not be used. The packets on the high speed channels will consist of Fast Ring packets with extra control information. This application is particularly well suited to a Binary Routing Network.

The main requirement is to get packets from an input to an output with as little delay as possible and without the intervention of any intelligent logic. The switch would need to be able to interconnect 16 sets of channels which are a mixture of the high speed links and stations connected to the ring acting as bridges. A possible outline of the system is given in Figure 8.1. A normal circuit switching system would require an amount of intelligence and slow down the passage of the packets. A Binary Routing Network, using a Perfect Shuffle structure, in this application could be considered a fast circuit switch, quickly setting up a path and then breaking it down as the end of the packet passes. It would be able to switch packets with very little delay and process many packets in parallel.

To determine how the system will operate the following assumptions will be made. First the network will consist of 16 input/output pairs. Each pair will operate generating packets at a maximum of 2Mb/s with distributed addresses. The network will be clocked at 40Mb/s which will mean that the equivalent offered load will be 5%. Packets will need to be buffered at both ends of the switch to enable bit rate conversion. Assuming each packet represents a Fast Ring packet with some extra control, the packet length will be approximately (2+2+32+2)= 38 bytes + 2 bytes of extra addressing gives 40 bytes or 360 bits if an extendible packet scheme as described in chapter 7 is used.

A load of 5% means that a packet will be arriving every 180μs. Each packet will take 9μs to cross the switch from the input buffer to the output buffer if it is not blocked. This means that if the packet is blocked it could be held for a total of 20 packet times before the input buffer overran. By examining the delay graphs in Chapter 6 the average delivery time of a packet within the switch would

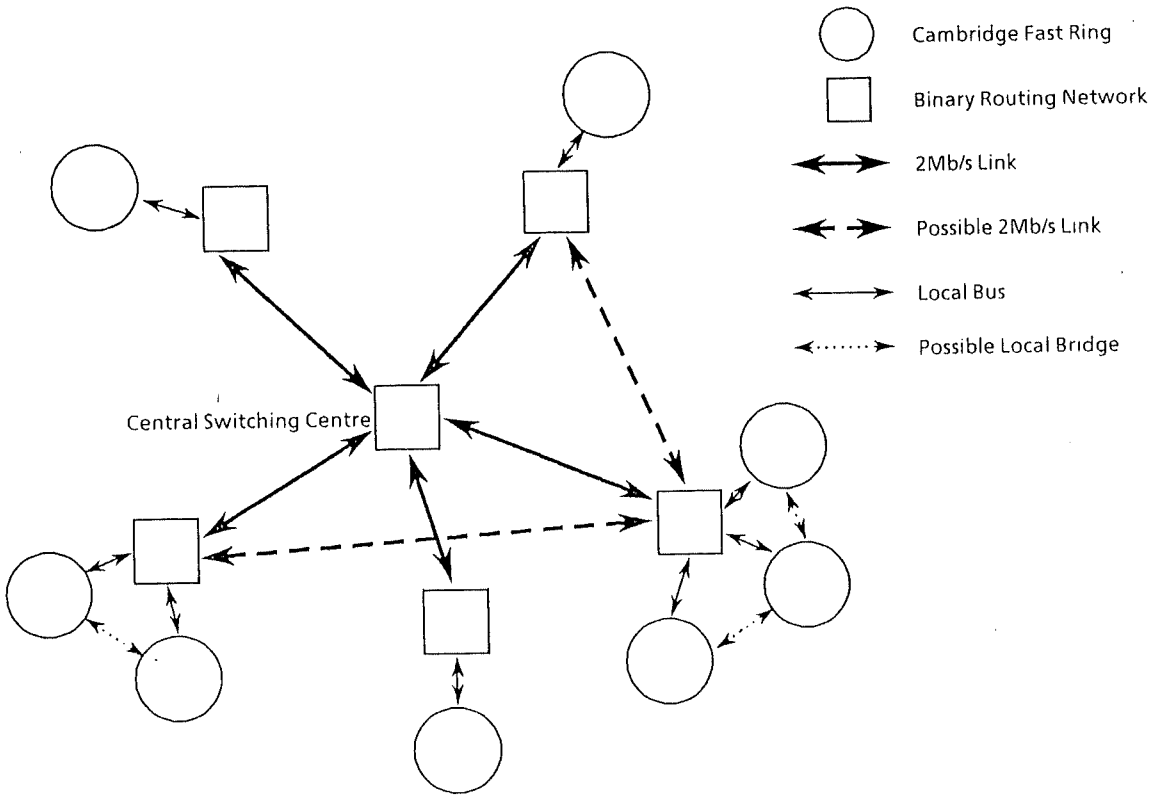


Figure 8.1: Possible Unison Structure using a Binary Routing Network

be 1.1 or 9.9μs. At this load almost all packets could be delivered in 2 packet times. This would mean that provided the output was able to accept a packet the delay through the switch would negligible and the provision for more that one buffer would be required only for exceptional conditions. The total delay through the switch would then be just a packet time while it is buffered to convert to and from the higher rate.

The above figures assume that there will be a continuous interchange of information between networks which will be well distributed. In general the operation of a network will not be like this and it is unlikely that the links will be completely occupied except for bursts of data which would involve many packets following the same route. Similarly if two stations were sending along the one high speed channel the delays could become large and additional buffering would be required.

The network could be operated at a lower rate, but 40Mb/s will be well above the rate that data can be supplied by the Fast Ring interface circuit. This would mean that the switching system could also function as a bridge between many Fast Rings at the same site.

References

[1] A. L. Ananda, B. Srinivasan, "An Extensive Bibliography on Computer Networks", *Computer Communications Review*, Vol. 13/14, pp. 78-98. ACM, Jan/April 1984.

[2] B. W. Arden, H. Lee, "Anaysis of a Chordal Ring Network", *IEEE Transactions on Computers*, Vol. C-30, No. **4**, pp. 291-295. April 1981.

[3] George H. Barnes, Stephen F. Ludstrom, "Design and Validation of a Connection Network for Many-Processor Multiprocessor Systems", "IEEE Computer", Vol. 14, No. **12**, pp. 32-41. IEEE, December 1981.

[4] V. E. Beneš, "On Rearrangeable Three-Stage Connecting Networks", *The Bell System Technical Journal*, Vol. XLI, No. **5**, September 1962. pp. 1481-1492.

[5] Laxmi N. Bhuyan, Dharma P. Agrawal, "Design and Performance of Generalised Interconnection Networks", *IEEE Transactions on Computers*, Vol. C-32, No. **12**, pp. 1081-1090. IEEE, December 1983.

[6] W. Bux, F. Closs, K. Kuemmerle, H. Keller, H. R. Mueller, "The Token Ring", *Local Area Networks: An Advanced Course*, pp. 32-62. D. Hutchison, J. Mariani and D. Shepherd (Ed.), Springer-Verlag, 1985.

[7] S. Cheemalavagu, M. Malek, "Analysis and Simulation of Banyan Interconnection Networks with 2x2, 4x4 an 8x8 Switching Elements", *Real-Time Systems Symposium*, pp. 83-89. IEEE, Los Angeles, California, December 1982.

[8] C. Closs, "A Study of Non-Blocking Switching Networks", *The Bell System Technical Journal*, Vol. XXXII, March 1953. pp. 406-424.

[9] Y. K. Dalal, "Use of Multiple Networks in the Xerox Network System", *Computer*, IEEE, Vol. 15, No. **10**, October 1982. pp. 82-100.

[10] N. J. Davis, H. J. Siegel, "The Performance Analysis of Partitioned Circuit Switched Multistage Interconnection Networks", *The 12th Annual International Symposium on Computer Architecture*, IEEE, June 1985. pp. 387-394.

[11] J. B. Dennis, G. A. Boughton, C. K. C. Leung, "Building Blocks for Data Flow Prototypes", *Proceedings of the 7th Annual Symposium on Computer Architecture*, ACM, May 1980. pp. 1-8.

[12] D. M. Dias, "Packet Communication in Delta and Related Networks", *PhD. Dissertation: Rice University*, April 1981.

[13] T. Feng, "Data manipulating functions in parallel processors and their implementation", *IEEE Transactions on Computers*, Vol. C-23, No. **3**, pp. 309-318.

[14] Tse-yun Feng, "A Survey of Interconnection Networks", *IEEE Computer*, Vol. 14, No. **12**, pp. 12-27. IEEE, December 1981.

[15] H. A. Freeman, K. J. Thurber, "UpDated Bibliography on Local Computer Networks", *Computer Communications Review*, Vol. 10, No. **3**, 1980. pp. 10-18.

[16] A. Hopper., D. Wheeler., "Binary Routing Networks", *IEEE Transactions on Computers*, Vol. C-28, No. **10**, October 1979. pp. 699-703.

[17] Yih-Chyun Jenq, "Performance Analysis of a Packet Switch Based on Single Buffered Banyan Network", IEEE, *Selected Areas In Communications*, Vol. SAC-1, No. **6**, Dec 1983. pp. 1014-1021.

[18] P. Kermani, L. Kleinrock, "Virtual cut-through: a new computer communication switching technique", *Computer Networks*, pp. 267-286. March 1979.

[19] Clyde P. Kruskal, M. Snir, "The Performance of Multistage Interconnection Networks for Multiprocessors", *IEEE Transactions on Computers*, Vol. C-32, No. **12**, pp. 1091-1098. IEEE, December 1983.

[20] J. J. Kulzer, W. A. Montgomery, "Statistical Switching Architectures For Future Services", *ISS 84*, May 1984.

[21] M. Kumar, D. M. Dias, J. R. Jump, "Switching Strategies in a class of packet switching networks", *Proceedings of the 10th Annual International Symposium on Computer Architecture*, Vol. 11, No. **3**, June 1983. pp. 284-300.

[22] M. Kumar, "Performance Improvement in Single-Stage and Multi-Stage Shuffle-Exchange Networks", *PhD. Dissertation, Rice University*, July 1984.

[23] H. T. Kung, "Why Systolic Arrays", *IEEE Computer*, Vol. 15, No. **1**, January 1982. pp. 37-46.

[24] D. H. Lawrie, "Access and Alignment of Data in an Array Processor", *IEEE Transactions on Computers*, Vol. C-24, No. **12**, December 1975. pp. 1145-1155.

[25] E. S. Lee, P. I. P. Boulton, "The Principles and Performance of Hubnet: A 50Mbit/s Glass Fiber Local Area Network", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-1, No. **5**, IEEE, November 1983. pp. 711-720.

[26] Manjai Lee, Chuan-lin Wu, "Performance Analysis of Circuit Switching Baseline Interconnection Networks", *11th Annual International Symposium on Computer Architecture*, pp. 82-90. IEEE, 1984.

[27] I. M. Leslie, "A Master Clock Repeater for the Cambridge Digital Communications Ring", *Proceedings of the IEE*, Vol. 128, Pt E, No. **2**, March 1981. pp. 64-66.

[28] W. Lin, C. Wu, "Design of a 2x2 fault-tolerant switching element", *Proceedings of the 9th Annual Symposium on Computer Architecture*, April 1982.

[29] C. Mead, L. Conway, "Introduction to VLSI systems", Addison-Wesley, 1980.

[30] R. M. Metcalfe, D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks", *Communications of the ACM*, Vol. 19, No. **7**, pp. 395-404. ACM, July 1976.

[31] J. H. Mirza, "Performance of self-routing shuffle-exchange interconnection networks in SIMD processors", *1982 International Conference on Parallel Processing*, Bellair, MI, pp. 13-15. August 24-27 1982.

[32] R. M. Needhan, A. J. Herbert, "The Cambridge Distributed System", Addison-Wesley, 1982.

[33] D. A. Padua, D. J. Kuck, D. H. Lawrie, "High-Speed Multiprocessors and Compilation Techniques", *IEEE Transactions on Computers*, Vol. C-29, No. **9**, pp. 763-776. IEEE, September 1980.

[34] J. H. Patel, "Processor-Memory Interconnections for Multiprocessors", *Proceedings of the 6th Annual Symposium on Computer Architecture*, April 1979. New York, NY, pp. 168-177.

[35] Janak H. Patel, "Performance of Processor-Memory Interconnections for Multiprocessors", *IEEE Transactions on Computers*, Vol. C-30, No. **10**, pp. 771-780. IEEE, October 1981.

[36] Terry Patten, Norm Hutchinson, Brian Unger, "The Flooding Sink: A New Approach to Local Area Networking", *Department of Computer Science University of Calgary Research Report No. 83/124/13*, May 1983.

[37] M. C. Pease, "The Indirect Binary n-Cube Microprocessor Array", *IEEE Transactions on Computers*, Vol. C-26, No. **5**, pp. 250-265. May 1977.

[38] C. Petitpierre, "Meshed Local Computer Networks", *IEEE Communications Magazine*, Vol. 22, No. **8**, pp. 36-40. IEEE, August 1984.

[39] E. G. Rawson, R. M. Metcalf, "Fibernet: Multimode Optical Fibers for Local Computer Networks", *IEEE Transactions on Communications*, Vol. com-26, No. **7**, July 1978. pp. 983-990.

[40] S. F. Reddaway, "DAP - a distributed array processor", *1st Annual Symposium on Computer Architecture*,

[41] J. F. Shoch, J. A. Hupp, "Measured Performance of an Ethernet Local Network", *Communications of the ACM*, Vol. 23, No. **12**, December 1980. pp. 711-721.

[42] C. L. Seitz, "The Cosmic Cube", *Communications of the ACM*, Vol. 28, No. **1**, January 1985. pp. 22-33.

[43] S. Temple, "The Design of a Ring Communication Network", *PhD. Cambridge University, Tr 52*, January 1984.

[44] K. J. Thurber, H. A. Freeman, "Local Computer Network Architectures", IEEE, 1979.

[45] D. L. Tennenhouse, "The UNISON Exchange Architecture", *Unpublished Note*, September 1985.

[46] Benjamin W. Wah, "A Comparative Study of Distributed Resource Sharing On Multiprocessors", *The 10th Annual Symposium on Computer Architecture*, pp. 301-308c. ACM, Stockholm, Sweden, June 1983.

[47] M. V. Wilkes, "Communication using a Digital Ring", *Proceedings PACNET*, 1975. Sendai Japan,

[48] M. V. Wilkes, D. J. Wheeler, "The Cambridge Digital Communication Ring", *Proceedings of the Local Area Communications Symposium*, May 1979. Boston, pp. 47-60.

[49] C. L. Wu, W. Lin, M. C. Lin, "Distributed Circuit Switching, STARNET", *1982 International Conference on Parallel Processing*, Bellair, MI, August 24-27 1982.

[50] L. T. Wu, "Mixing Trafic in a Buffered Banyan Network", *Proceedings of the 9th Communications Symposium; in Computer Communications Review*, Vol. 15, No. **4**, September 1985. pp. 134-139.