DISTRIBUTED ROUTING ALGORITHMS FOR INTERCONNECTED FDDI LANS

IAN JAMES WARFORD

Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

NOVEMBER 1992

THE UNIVERSITY OF ASTON IN BIRMINGHAM

DISTRIBUTED ROUTING ALGORITHMS FOR INTERCONNECTED FDDI LANS

IAN JAMES WARFORD

Submitted for the Degree of Doctor of Philosophy

1992

Summary

The Fibre Distributed Data Interface (FDDI) represents the new generation of local area networks (LAN). These high speed LANs are capable of supporting up to 500 users over a 100 km distance. User traffic is expected to be as diverse as file transfers, packet voice and video. As the proliferation of FDDI LANs continues, the need to interconnect these LANs arises.

FDDI LAN interconnection can be achieved in a variety of different ways. Some of the most commonly used today are public data networks, dial up lines and private circuits. For applications that can potentially generate large quantities of traffic, such as an FDDI LAN, it is cost effective to use a private circuit leased from the public carrier.

In order to send traffic from one LAN to another across the leased line, a routing algorithm is required. Much research has been done on the Bellman-Ford algorithm and many implementations of it exist in computer networks. However, due to its instability and problems with routing table loops it is an unsatisfactory algorithm for interconnected FDDI LANs. A new algorithm, termed ISIS which is being standardized by the ISO provides a far better solution.

ISIS will be implemented in many manufacturers routing devices. In order to make the work as practical as possible, this algorithm will be used as the basis for all the new algorithms presented.

The ISIS algorithm can be improved by exploiting information that is dropped by that algorithm during the calculation process. A new algorithm, called Down Stream Path Splits (DSPS), uses this information and requires only minor modification to some of the ISIS routing procedures. DSPS provides a higher network performance, with very little additional processing and storage requirements.

A second algorithm, also based on the ISIS algorithm, generates a massive increase in network performance. This is achieved by selecting alternative paths through the network in times of heavy congestion. This algorithm may select the alternative path at either the originating node, or any node along the path. It requires more processing and memory storage than DSPS, but generates a higher network power.

The final algorithm combines the DSPS algorithm with the alternative path algorithm. This is the most flexible and powerful of the algorithms developed. However, it is somewhat complex and requires a fairly large storage area at each node.

The performance of the new routing algorithms is tested in a comprehensive model of interconnected LANs. This model incorporates the transport through physical layers and generates random topologies for routing algorithm performance comparisons. Using this model it is possible to determine which algorithm provides the best performance without introducing significant complexity and storage requirements.

KEY WORDS: Local Area Networks, LAN Interconnection, Internetworking, Routers

To the memory of my mother, Beryl Warford.

Acknowledgements

I would like to thank my supervisor, Dr. R. L. Brewster for his assistance over these last three years, and the remarkable speed with which he proof-read the chapters of this work. I would also like to thank Geoff Harland of BNR (Europe) who sponsored this project along with the SERC. I am particularly grateful to Dr. Andrew Bartholomew of Layer 8 Consultancy for providing me with the latest copy of the ISIS standard, without which, much of the research could not have been accomplished. Finally, my girlfriend Ann Marie Lord for her continuous encouragement and for putting up with me when I worked every weekend and evening!

Contents

Figures and Tables

List of Figures.

List of Tables.

Chapter 1

Introduction

1.0 Introduction

The use of local area networks has grown rapidly over the last ten years. One of the main reasons for this tremendous growth is due to the increased processing power and reduced cost of personal computers. This has produced a movement away from large mainframe centres where users brought information to be processed to bringing the computer to the user via local area networks. Stallings defines a local area network as "a communication network that provides interconnection of a variety of data communicating devices within a small area" [2]. The variety of data devices includes any device that communicates over a transmission medium. This encompasses such devices as terminals, peripheral devices, sensors, telephones, facsimile, television transmitters and receivers to name but a few.

The local area network has become in a relatively short time since its conception the backbone of both the university and corporate networking environments. In some organizations the cost of having a local area network failing can cost thousands of pounds in lost earnings within a short time period.

Local area networks in common use today include the 10 Mbps CSMA/CD (carrier sense, multiple access with collision detection) and the 16 Mbps Token Ring. Both of these technologies have been standardized by the IEEE 802 committee (referred to as 802.3 and 802.5 respectively) and are designed for use with a copper medium.

As the cost of computer processing power has dropped so too has the cost of lasers and fibre optic cable. This has led to increased interest in using fibre optics in local area networks. The advantages of fibre optics over the copper media are many; improved security, increased bandwidth, less interference to name but a few. Fibre optic LANs are capable of providing speeds up to 2 Gbps and beyond.

At present the Fibre Distributed Data Interface (FDDI) is the only fibre optic LAN going through the standardization procedure. Expectations for this LAN are high. Already many companies have built the required hardware to be incorporated in the LAN. This LAN runs at 100 Mbps and can support up to 500 users over a 100 km distance. Initial usage of FDDI LANs is expected to be as a backbone connecting CSMA/CD and Token Ring LANs together around a single site.

As the use of FDDI LANs increases, so to will the desire to interconnect geographically remote FDDI installations. This is the problem of corporate networks, where each individual office around the country, or even world, has its own LAN. Interconnecting these LANs to enable the sharing of information and facilities presents a multitude of problems.

One of the problems of interconnecting remote LANs together is how to send traffic from one LAN to another. Routing is the set of algorithms which attempt to find the best path from the source LAN to the destination LAN. A special microcomputer on each LAN, called a gateway, is in charge of selecting the best path to be used to forward data from one LAN to another. This is achieved by exchanging information about the networks between gateway devices. Routing algorithms are essential for the smooth operation of an interconnected LAN environment.

The first implementation of a routing algorithm in a computer communications network was in the ARPANET sponsored by the U.S Department of Defence. This work inspired much research and many routing algorithms have been developed as a result. Many of these algorithms have been designed for networks where the amount of traffic generated is relatively low. These algorithms are unsatisfactory for an FDDI LAN internetwork which is capable of generating a vast quantity of traffic within a very short time interval.

The objective of this work, therefore, is to develop a practical routing algorithm for interconnected FDDI LANs. Although the routing algorithm will be developed for an FDDI internetwork they are suitable for use with any LAN or packet switching network.

Chapter 2 provides an overview of the FDDI LAN and the various methods that can be used to interconnect remote LANs. The chapter also discusses routers and bridges and suggests which technology is more suitable for interconnected LANs.

Chapter 3 begins by giving an overview of routing algorithms, their function and one common classification method. It goes on to discuss the four algorithms which operate independently but interact to perform the routing function. Since the choice of routing algorithm is highly dependent on the type of network under consideration, a section of this chapter is devoted to examining the type of network to be considered. This chapter provides a framework for the type of algorithm that may prove satisfactory for an interconnected FDDI LAN environment.

Chapter 4 concentrates on one particular type of algorithm - the so-called Bellman-Ford algorithm. The characteristics of this algorithm are discussed and the relative merits and demerits of the algorithm are presented.

Chapter 5 presents an algorithm that was originally produced by Dijkstra. A modified form of this algorithm has been adopted by the standards bodies for use in interconnected LANs. A new algorithm which exploits information discarded by the standard algorithm is presented.

Chapter 6 introduces alternative path algorithms. These algorithms allow the use of higher cost paths to be used during times of congestion on the main paths. Three new algorithms which are based on the standard algorithm are presented and analysed

Chapter 7 combines the algorithms of chapter 5 with those of chapter 6. These are referred to as combination algorithms.

Chapter 8 presents the simulation model used to derive the results for the previous routing algorithms. This simulation model is fairly comprehensive and the various protocols used in the simulation are discussed.

The thesis is then concluded in chapter 9

Chapter 2

LAN Interconnection

2.0 Introduction

In the 70's and early 80's many different computer manufacturers had different methods of communicating with their proprietary product. This meant that if a user purchased IBM equipment he was then tied to that manufacturer for all subsequent purchases of network equipment. Large companies such as General Motors and Boeing Aircraft Corporation, which utilized a vast number of computers throughout their organizations, felt it was not in their best interest to be tied to one manufacturer. As such, pressure from these companies was put on to the International Standards Organization to come up with some standards to which the computer companies would abide to allow machines from one company to communicate with those of another company. Needless to say, such an idea was met with considerable resistance from the computer manufacturers.

Nevertheless, the ISO came up with the Open Systems Interconnection (OSI) reference model. The theory being that if manufacturers were in agreement as to what each layer of the model did, then communications between different machines could easily take place. Systems were then said to be 'open'. Computer manufacturers are slowly moving toward the OSI model in their products. However, with literally thousands of IBM and Digital networks already in existence it is unlikely such a transition will occur overnight, if ever [1].

Due to the complexity of interconnecting computers to allow them to communicate, network functions are always divided into a series of layers, each with a specific task to perform [3]. The OSI reference model consists of seven layers as shown in figure 2.1. The application through session layers of the model are of no interest in this theses since they have no bearing on the interconnection of LANs or the subsequent simulation model.

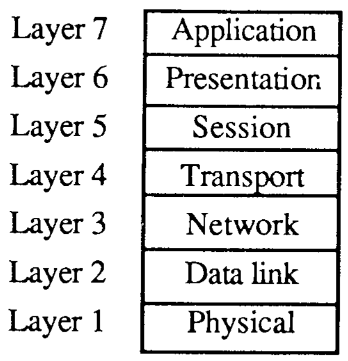


Figure 2.1 The seven layer OSI model.

2.1 LAN Standards

The OSI model has adopted the standards developed by the IEEE 802 committee for local area networks, they are the 8803, 8804 and the 8805 which correspond to the CSMA/CD (802.3), token bus (802.4) and token ring (802.5) respectively. The IEEE standards fit into the data link layer of the OSI model as shown in figure 2.2. The data link layer is further split into the medium access control sublayer (MAC) and the logical link control sublayer (LLC).

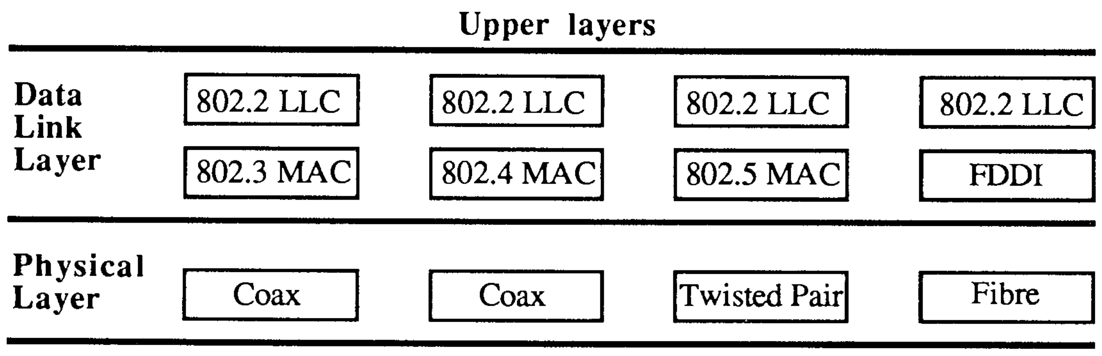


Figure 2.2 The data link and physical layers of the four standard networks.

2.2 The Fibre Distributed Data Interface

The Fibre Distributed Data Interface (FDDI) is the most recently standardized of the four local area network technologies. FDDI has been standardized by the ANSI X3T9.5 committee and uses the logical link control defined by the IEEE. The four LAN standards and how they relate to the OSI model are shown in figure 2.2. FDDI is a fibre optic token ring network which operates at a data rate of 100 Mbps and can support up to 500 stations over a 100 km distance. The FDDI protocol has used the IEEE 802.5 16 Mbps token-ring protocol as a starting point and modified it where necessary to to cope with the higher speed. The advantages of fibre optics over conventional copper are many and include greater bandwidth, lower attenuation, less noise, greater security and lower cost [4,7].

2.2.1 FDDI Operation

Access to the fibre optic medium of FDDI is by way of a token which circulates around the ring. Stations on the ring remove the token and are then permitted to transmit any data which they may have until their token timer expires. Upon expiration of this timer a station must release the token allowing other stations on the ring to transmit. The medium access technique of token passing works well under heavy traffic loading and often outperforms contention based techniques such as CSMA/CD [28].

The FDDI ring is a combination of two independent counter rotating rings, each running at 100 Mbps. Both rings have their own tokens which they are responsible for. If both rings operate simultaneously the effective throughput is 200 Mbps. The advantage of having two rings is that if one fails, the network can reconfigure using the other ring and still keep operating. This ring reconfiguration occurs as shown in figure 2.3. When a link failure occurs, the stations on either side of the fault make use of the opposite direction ring in order to isolate the failed link. Following ring reconfiguration, only one token remains in circulation.

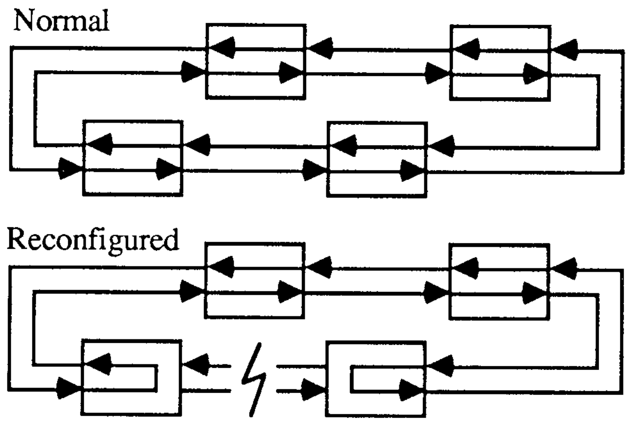


Figure 2.3 Reconfiguration of an FDDI LAN following link failure.

The nodes connecting to the rings are divided into two categories: class A and class B. Class A stations connect to both rings simultaneously, whereas class B stations only connect to one ring. Class B stations can therefore be implemented at a lower cost than the class A stations. The disadvantage of this is that a class B station may become isolated if its link fails. Class A stations on the other hand, require additional hardware to connect them to both rings but are protected against failure. Normally those stations that require an element of fault tolerance are configured as class A stations. Less critical stations can be configured as class B.

2.2.2 Usage of FDDI

Originally, FDDI was proposed by ANSI as a back-end network between mainframe computers and their peripherals. Later the committee expanded the scope of FDDI to emphasize application as a backbone network between lower speed LANs such as the IEEE 802 networks. Finally, use as a front-end, next generation LAN between powerful workstations is expected. Such connections to every desk will probably not take place until the cost per connection drops sufficiently. Figure 2.4 shows FDDI as both a back end network and a backbone to lower speed LANs [5,6].

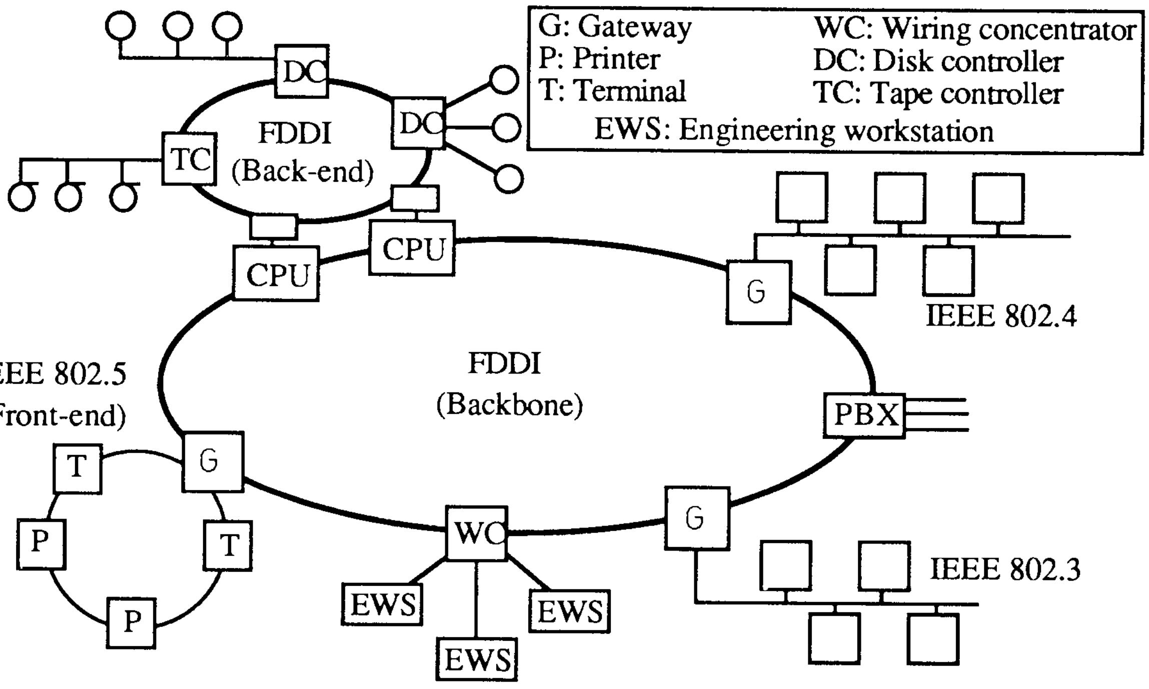


Figure 2.4 FDDI as both a backbone and back end LAN.

2.3 LAN Interconnection - Links

Many users demanding high performance networking will install FDDI at different sites and then want to connect these installations while sacrificing as little performance as possible between the sites [8,9].

FDDI at remote sites can be interconnected in a variety of ways. The choice of which method to use is based on many factors which include inter-site traffic volume, cost of the links and the distance between the sites. All three issues are intertwined and it is not possible to consider one without the other. This section will look at various ways of interconnecting FDDI LANs.

Methods of interconnecting FDDI LANs which are readily available today include single-mode fibre, dial-up lines, public data networks and leased lines. Other methods, such as metropolitan area networks, frame relay and SDH links are not yet widely available. Each technology has its supporters and detractors. In a real internetworking environment it is highly likely that a variety of the above methods would be used.

2.3.1 Interconnection by Single Mode Fibre

The FDDI standard defines two different physical layers that can be used with the technology. These are the use of multi-mode fibre and single mode fibre. Multi-mode fibre is a cheaper technology which uses low cost LED's for signalling whereas single-mode fibre requires lasers for signalling. However, single-mode fibre allows the signal to traverse distances of up to 50 km before repeaters are needed as compared to 2 km for multi-mode fibre [9].

The distance between two, or more, FDDI LANs can be increased by using single mode fibre between sites. This is achieved by breaking the FDDI multi-mode fibre LANs open and inserting the single mode fibre between the sites as shown in figure 2.5. Interconnecting LANs in this way can extend the distance between LANs to that of a metropolitan area. However, it is not without its problems.

This method is not bandwidth efficient. A station waiting for the token must wait for it to circulate around the entire extended ring before being allowed to transmit. This is referred to as token latency. If the ring is large this may lead to unacceptable delays for a station. Additionally if one of the rings fails and wraps, this problem is exacerbated and the delay experienced will provide an unsatisfactory service.

Even if an organization has a right-of-way agreement, (such as power companies, waterways, railways etc) acquiring, installing and maintaining single mode fibre is an expensive undertaking. It is possible that public carriers could supply such fibre, however, it would still remain a very costly exercise.

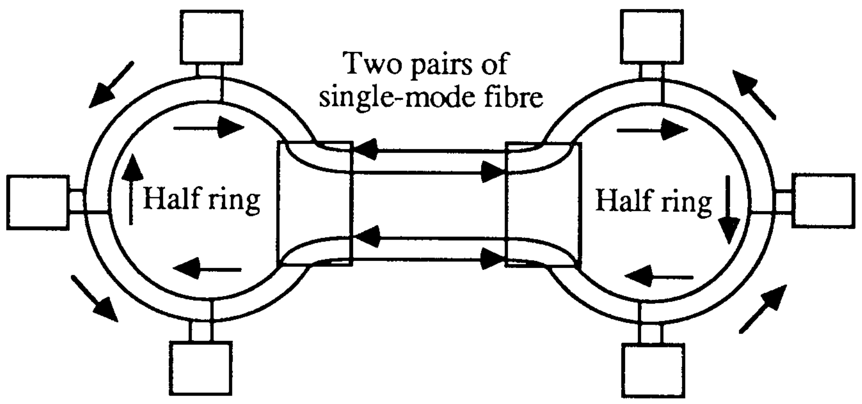


Figure 2.5 FDDI LANs interconnected via single-mode fibre.

The above physical layer interconnection method sends all data, regardless of destination, over the entire ring. This means that local traffic will experience the same delay as traffic destined for some remote LAN. In order to overcome this problem, devices can be inserted between the LANs which only forward traffic which is destined for a remote LAN. In this way, lower speed links can be used to interconnect the remote LANs. Using these filtering devices each LAN remains autonomous, and the bandwidth inefficiency problem is relieved. Additionally, it produces an overall increase in network performance when most of the traffic is local.

There are several different wide area networking technologies which can be coupled to filtering devices to allow inter-site LAN connection.

2.3.2 Interconnection by Public Data Networks

Public data networks (PDNs) are very commonly used for interconnecting todays CSMA/CD and token ring LANs. They are almost always based on the connection oriented X.25 set of standards. British Telecoms Packet Switched Stream (PSS) is an example of such a PDN. Charging for PDNs is based on the number of packets that a user sends over the network [89]. The use of PDN's for interconnecting FDDI LANs may be justified if the amount of traffic emanating from the LAN is small.

2.3.3 Interconnection by Circuit Switched Links

Circuit switched, or dial-up-lines are simply telephone lines coupled to a modem. These lines can be called up when required such as during periods of congestion on the main links. The charging for the use of these lines follows the standard telephone system. These lines are cost effective for transmissions that have a large number of bits per call [89].

2.3.4 Interconnection by Leased Lines

Dedicated lines leased from the public telephone company can offer a variety of speeds for LAN interconnection. The most commonly used speeds for such interconnection are the 64 kbps (Kilostream) and the 2.048 Mbps (Megastream) links available from British Telecom. Charging for such networks is on a per link basis and will be constant regardless of the number of packets sent down the line. Leased lines are the most common method for interconnecting LANs. They allow almost unlimited distance between remote LANs and give the organization using them tremendous freedom in designing the network [89].

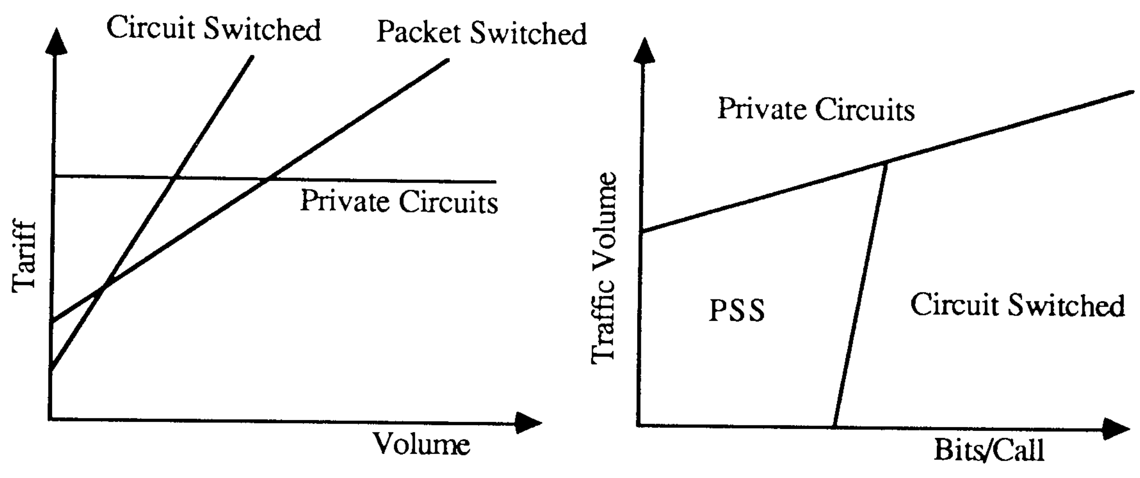


Figure 2.6 Tariff vs volume and volume vs bits/call for various wide area networks.

The choice between the three WAN technologies is based on the following considerations; tariff, traffic volume and bits per call. The relationships between these parameters and the three technologies are shown in figure 2.6. The technologies are shown pictorially in figure 2.7 interconnecting FDDI LANs.

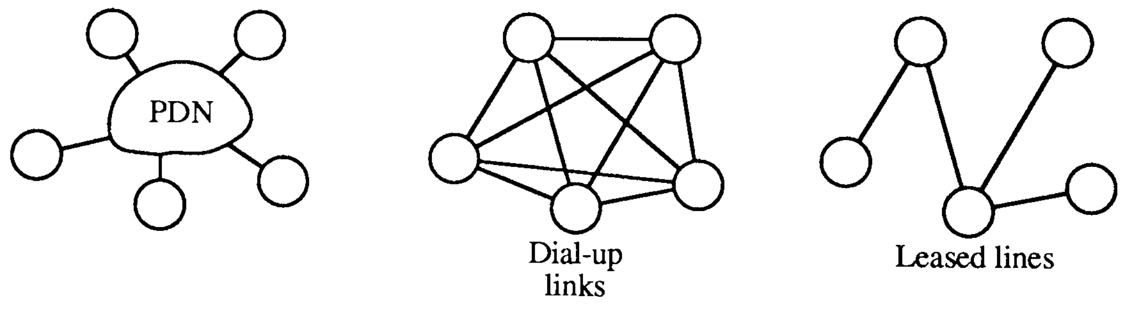


Figure 2.7 PDN, dial-up links and leased lines interconnecting FDDI LANs.

FDDI is capable of generating a great deal of inter-LAN traffic. Additionally, a large network may have LANs spread over hundreds or even thousands of miles. For the above reasons it is justifiable to utilize leased lines from the public telephone companies. Using leased lines has become the de facto standard for interconnecting todays corporate LANs.

2.3.5 B-ISDN

The above methods of LAN interconnection (with the exception of the single mode fibre approach) are considered the traditional methods of remote LAN interconnection. Indeed, many large networks may employ all three methods of interconnection. However, the public carriers are looking towards implementation of the Broadband Integrated Services Digital Network (B-ISDN). This is a cell based fast packet switching technology which can support services as diverse as real time telephony and video transmission on the same high speed link.

The transition to full B-ISDN will not take place overnight, an evolutionary process is expected [10]. This process will generate solutions to the LAN internetworking problem which may rival those presented in the previous section. A brief overview of these intermediate technologies is in order.

2.3.6 Interconnection by MANs

The IEEE 802 committee has standardized the Distributed Queue, Dual Bus (DQDB) technology as 802.6 for use as a metropolitan area network [11,12]. Metropolitan area networks (MANs) are larger than local area networks but smaller than wide area networks. They use the same medium access control schemes as LANs and cover the area of a large city. MANs are operated by the public telephone carrier and allow subscribers to the service to communicate with one another. The disadvantage of such a system is that it is limited in geographical coverage. Charging for the system is likely to be on a per packet basis as it is with the PSS.

2.3.7 Interconnection by Frame Relay

Frame relay is a new ISDN packet mode bearer service provided by the public carriers. It is a data link layer technology which provides connection-oriented service to the user at speeds up to 768 kbps [13,14]. This is simply the carriers' answer to updating the PDN's that are in common use. However, frame relay provides much higher speeds and since it operates at the data link layer (as opposed to X.25's network layer) the amount of processing required at each switching node is much lower. Frame relay offers a far better solution to the internetworking problem than todays PDNs.

2.3.8 Interconnection by SDH

To overcome the expense of single mode fibre, the new 155.52 Mbps synchronous digital hierarchy (SDH) links can be used. The entire FDDI symbol stream is mapped directly into the SDH synchronous payload envelope [9]. The carrier is then responsible for regenerating the FDDI signal which will allow an almost unlimited distance between FDDI LANs. This is shown in figure 2.8. The SDH links are bidirectional, so each one replaces two unidirectional single-mode fibres. However, although the distance between LANs can be greater, the problem of token latency remains. Links slower than SDH lines such as the Kilostream and Megastream links can not be used in this physical layer configuration since they would reduce the performance down to an unacceptable level.

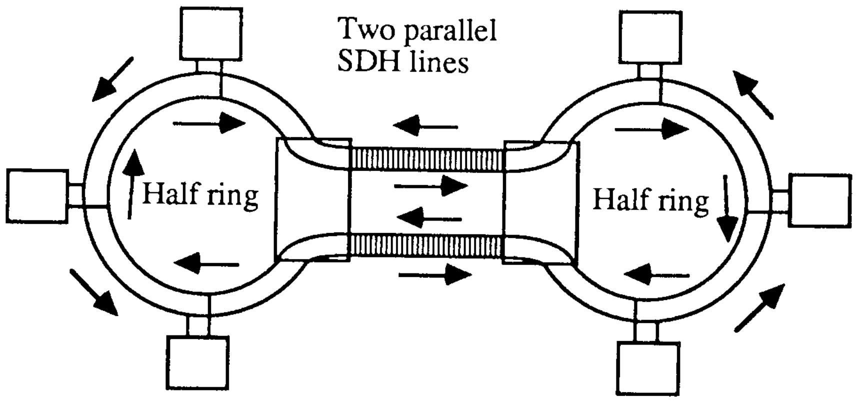


Figure 2.8 FDDI LANs interconnected via SDH links.

Even when the full B-ISDN has been implemented, it is anticipated that many users will still require leased lines for some applications. Frame relay and DQDB are seen by many as simply stop-gap solutions before full B-ISDN implementation. It is for this reason that some users are embracing the new technologies with something less than enthusiasm.

It is therefore assumed in this thesis that the remote FDDI LANs are interconnected using leased lines.

The above interconnection methods are summarized in table 2.1.

Link type	Distance between LANs	Tariff	Suitable traffic loading
Single-mode fibre	metropolitan area	per link	high
SDH lines (physical layer)	unlimited	per link	high
Circuit-switched lines	unlimited	per call	low
Public data networks	unlimited	per packet	medium
Metropolitan area networks	metropolitan area	per packet	medium
Frame relay	unlimited	per packet	medium
Leased lines	unlimited	per link	high

Table 2.1 Summary of various wide area networking technologies.

2.4 LAN Interconnection - Devices

In order to connect FDDI LANs to the wide area network technology, relays which forward packets from one LAN to another are needed. These devices can operate at any layer of the OSI model. A few of the more common device names and the layer that they relate to are shown below [19].

repeater : physical layer
bridge : data link layer
router : network layer

The above terminology is not universally accepted. Routers are often termed Interface Message Processors (IMP), intermediate systems or even gateways. However, in this theses the term router will be used to indicate a layer three device.

In general the more layers of the OSI model that are used to achieve interconnection, the greater the complexity of the device. However, with the increased complexity comes the ability to cope with a greater number of differences between connecting networks.

Repeaters are the simplest devices that can be used for interconnecting networks. They take in all bits from the input port and simply copy them, regardless of destination to the output port at a higher signal level. Since there is no processing to be done on packets as they pass through the repeater, packet delay through the device is very low. Repeaters can be used for interconnecting FDDI LANs by single-mode fibre and SDH links as discussed in the previous section. This method was shown to suffer problems with token latency.

Instead of forwarding all traffic regardless of destination as repeaters do, bridges and routers can be used to filter out intra-LAN traffic from inter-LAN traffic and forward only the latter. There is great debate over which technology offers a better solution for interconnecting LANs [15,16,17,18,20]. This section will briefly examine the issues and suggest which technology is better for use in a network of FDDI LANs interconnected with leased lines.

2.4.1 Bridges

The IEEE 802 committee has standardized two types of bridge. The spanning tree bridge and the source routing bridge. Most bridges on the market today, if not all, correspond to one of these two standards. Bridges by definition, operate at either the Logical Link Control (LLC) sublayer or the Medium Access Control (MAC) sublayer of the data link layer (figure 2.9). Both of the IEEE 802 bridges operate at the MAC layer.

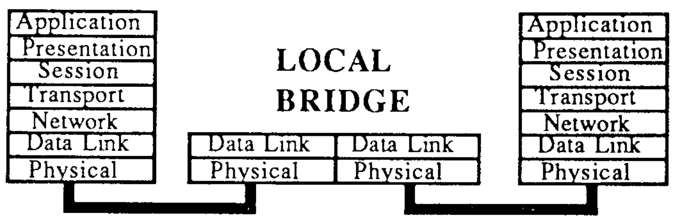


Figure 2.9 OSI layers of a local bridge.

2.4.1.1 Spanning Tree Bridge

The spanning tree bridge or transparent bridge accepts every frame on all the LANs to which it is attached [1,22]. When a frame arrives, a fairly simple routing algorithm is performed to determine which LAN a frame should be retransmitted on. The routing procedure is as follows:

1. If the destination and source LANs are the same, discard the frame.
2. If the destination and source LANs are different, forward the frame.
3. If the destination LAN is unknown, use flooding.

(If a frame is 'flooded' it is sent out on all the available links accept for the one on which it was received. Flooding will be discussed further in the next chapter.)

This routing procedure does have its' disadvantages. If two bridges are used from one LAN to another, to improve reliability, loops can occur. This can be seen in figure 2.10. If each bridge receives a frame, with an unknown destination, from LAN 1 it is flooded onto LAN 2. Bridge 1 and 2 will then forward each others frames back onto LAN 1 and the cycle begins again.

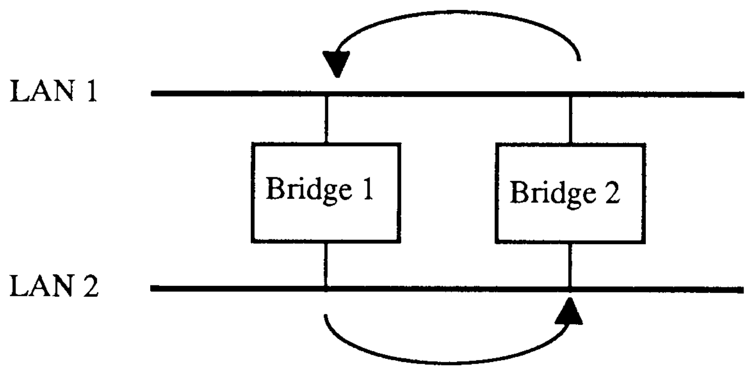


Figure 2.10 Looping of packets in a spanning tree bridge.

The solution to this problem is for each bridge to communicate with each other and overlay the actual topology with a spanning tree that reaches every LAN. Since a unique path exists from each source to destination, loops are avoided. Bridges which are not in use are 'blocked' and are not used to transmit traffic. These bridges can become 'unblocked' if a link or bridge failure occurs and a new spanning tree of the network is then calculated. Following this procedure, operation of the network can then continue using the new topology (figure 2.11).

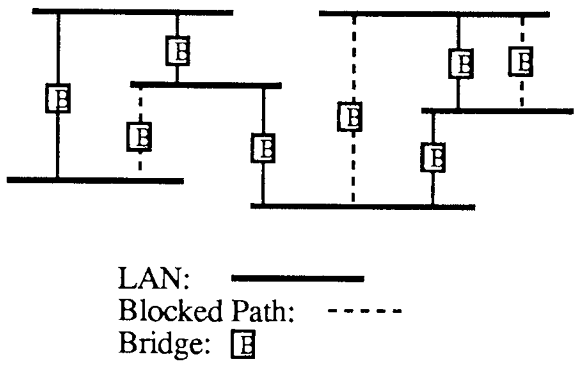


Figure 2.11 Spanning tree bridged network.

Installation of transparent bridges requires no hardware or software changes, no setting of address switches and no downloading of routing tables. It is an excellent technology for the interconnection of local area networks around a campus. However, more relevant to this section is how these devices perform when connected to a series of leased lines.

Spanning tree bridge technology can be used in the wide area network environment with a slight modification. Instead of having a single bridge between two LANs, each network has what is termed a 'half bridge' or remote bridge (figure 2.12) [21]. These operate very much as described above. However the shortcomings of the spanning tree bridge become quite acute in a wide area network.

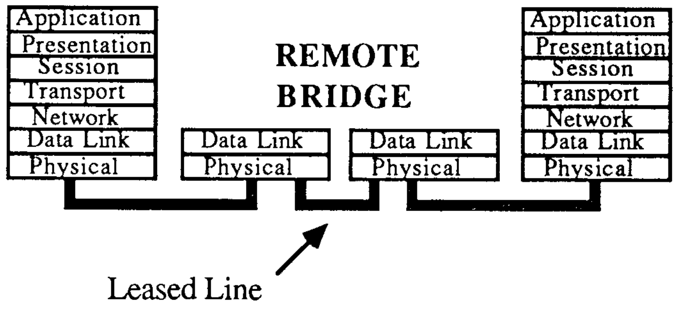


Figure 2.12 OSI layers of a remote bridge.

One of the biggest problems with the device is its' inability to cope with multiple routes between destinations. If the links between the LANs in figure 2.11 are now considered to be wide area network leased lines the problem begins to emerge. As discussed previously, leased lines cost money whether they are carrying any traffic or not. Therefore, using spanning tree bridges with certain links in blocked mode is uneconomical and does not allow for the optimal use of links.

2.4.1.2 Source Routing Bridge

As discussed above, spanning tree bridges do not make optimal use of the bandwidth, since they only use a subset of the topology. It is for this explicit reason that the 802 committee produced its second bridging standard for the source routing bridge (figure 2.13).

Very simply, source routing requires that the sender of each frame knows whether or not the destination is on its own LAN. If the frame is destined for a different LAN, the source machine must set the high order bit of the destination address to a '1' otherwise it must be left as a '0'. In addition, it includes in the frame header the exact path that the frame is to follow [23,24].

To find the route that the frame must follow is an integral part of the source routing algorithm. If a particular destination is unknown, the source transmits a broadcast frame asking where it is. This so called 'discovery frame' is copied by every bridge so that is reaches every LAN on the network. When the destination is found it sends a reply back to the source via every bridge that was used to construct the initial path. The sending station is then presented with a multitude of paths with the result being that the sender chooses the best route.

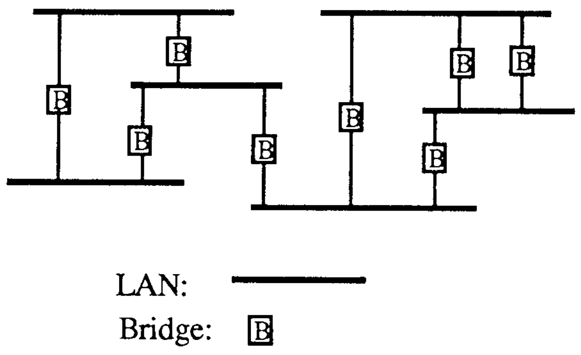


Figure 2.13 Source routing bridged network.

The use of source routing bridges in interconnected leased line WAN certainly looks appealing particularly with it 'best route' mechanism. Unfortunately, even with this technique there is an underlying problem. This difficulty stems from the use of the discovery frames.

For each bridge on the network one discovery frame is produced. If a network exists with LANs all interconnected by three bridges (not an unusual situation) then the following situation will result: The three bridges will each forward the discovery frame to the next LAN producing three more discovery frames. The next bridge will produce nine frames and so on up until the final LAN is reached. By the time it reaches the N^{th} LAN, 3^N frames will be circulating. If 10 sets of bridges are crossed, almost sixty thousand frames will be injected into the final LAN causing tremendous congestion. This situation also develops with the spanning tree bridge, but since only one route is available the frames increase on a linear basis, not on an exponential one [1].

For LANs interconnected around a campus site, the spanning tree bridge is probably a better performer than the source routing bridge. The ability of the bridge to only use a subset of the topology is not a problem in such a network since redundant links cost nothing following installation.

Although the spanning tree bridge may be better than the source routing bridge in a local setting, neither is truly satisfactory in an environment of interconnected LANs over leased lines [25,26].

2.4.2 Routers

Routers operate at the network layer of the OSI model (figure 2.14). This means that they must utilize a protocol above the basic 802.2 LLC protocol. Some of the most commonly used network layer protocols are listed below:

1. ARPA IP
2. DNA network layer
3. XNS network layer
4. ISO 8473

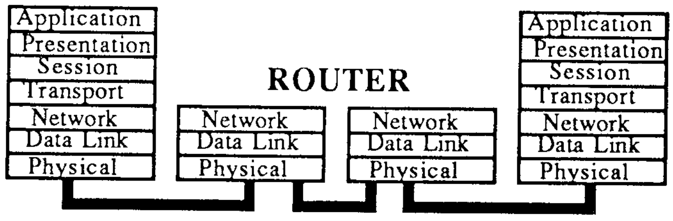


Figure 2.14 OSI layers of a router.

One of the biggest problems encountered in an extended LAN is that of addressing. In a bridged network, addressing must be common across all links or some simple mapping must exist. This in turn means there is a practical limit on the total network size that can be supported. This problem does not occur in a router network since all stations have agreed on a common network layer address space. Addressing can, therefore, be done hierarchically which means an almost limitless size network can be supported [17,18].

Two problems with a bridged network were discussed in the previous section. These are the problems of inefficient use of links (spanning tree bridge) and the flooding of discovery frames to unknown destinations (source routing). The router overcomes these problems by utilizing network layer protocols which provide an active handshaking protocol between routers and end stations. This protocol allows routers to determine with greater precision, the location of a station. It is for this reason that routers can use an arbitrary network topology and can select routes based on a variety of metrics.

In a bridged network, packets which are larger than a link permits will not be delivered to a station on that link. This is because data link layer protocols do not contain sufficient information to do packet reassembly. A solution to this problem is to specify a maximum packet size for the entire network. In contrast, most network layer protocols provide a fragmentation and reassembly mechanism which rules out the need for any network wide packet size limitation.

For interconnecting LANs across leased lines to another distant LAN routing technology is very appealing. It is very efficient at using bandwidth, implements congestion and flow control, allows variable packet sizes and can be used in large networks. Unfortunately, due to the extra processing requirements, the router can only handle about one third the frame throughput rate of a bridge and is sometimes the cause of the bottle-neck between connecting LANs. However, router technology is rapidly advancing and the higher throughput of the new breed of routers is making them more attractive all the time [15]. Bridges and router devices are summarized in table 2.2.

	Router	Spanning-Tree Bridge	Source Routing Bridge
OSI Layers	1,2,3	1,2	1,2
Protocol Dependent	yes	no	no
Different Frame Size support	yes	no	no
Load Balancing	yes	no	no
Multiple Routes	yes	no	no
Address Change	yes	no	no
Transparent	no	yes	no
Fault Tolerant	yes	yes	yes
Error Checking	good	poor	poor
Throughput	low	high	high
Flow Control	good	poor	poor
Routing	complex	simple	simple
Type of Service	yes	no	no
Suitable for Large WAN	yes	no	no

Table 2.2 Internetworking devices compared.

It is likely that in an actual network using FDDI as a backbone LAN that all three relaying devices will be used. The most common uses for these devices are shown in figure 2.15. Repeaters are shown connecting some CSMA/CD segments which are in turn connected via a bridge to the backbone. A token ring LAN and a bridged CSMA/CD network are also bridged to the FDDI. Finally, routers are used to connect the FDDI LANs to remote sites via leased lines [27].

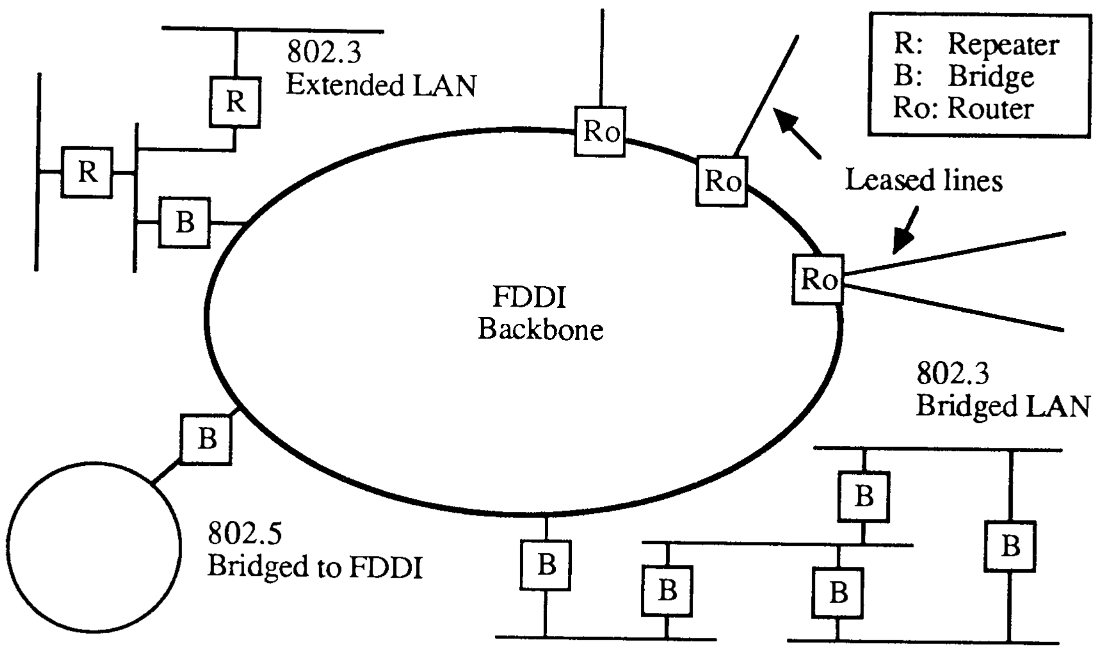


Figure 2.15 Repeaters, bridges and routers in an FDDI network.

2.5 Conclusion

FDDI represents the next generation of high speed local area networks which has been designed to meet the needs of several different networking environments. Even with the 100 km distance that FDDI can support, users will want to connect remote FDDI sites together.

Interconnection can be achieved in a variety of different ways, the selection of which depends on many factors. Leased lines offer one of the most cost effective and flexible solutions to the FDDI internetworking problem. Leased lines are expected to be in use even after implementation of the B-ISDN.

In order to connect FDDI LANs to leased lines, bridges or routers can be used. Bridges are an excellent technology for interconnecting LANs around a single campus site. For a large internetwork with multiple links they are less satisfactory; routers offer the best solution for such a network.

The remainder of this thesis will examine routers in a network composed of interconnected FDDI LANs using leased lines.

Chapter 3

Overview of Routing Algorithms

3.0 Introduction

A router, as its name suggests, requires a routing algorithm of some description in order to forward packets from one LAN to another. This routing function is one of the principle tasks of the OSI network layer. In a single LAN, this network layer is minimal since the routing problem is trivial. Packets are simply placed onto the medium, no routing is required. Routing algorithms become important when there is a choice of paths available from a source to a destination.

Routing algorithms are mainly used in the traditional wide area network, however they become applicable to LANs when they are interconnected via routers across leased lines. In this context there is little difference between a wide area network and a local area network. Therefore, it is necessary to look at the switching mechanisms used in wide area networks to see which are suitable for implementation in a LAN internetwork.

There are many algorithms in existence both in the literature and in actual network operation [32]. In order to reduce the selection of potential algorithms it is necessary to examine the type of environment that the algorithm is expected to operate in. This will involve, among other things, looking at characteristics of the LAN being interconnected, the four routing procedures that make up an algorithm and finally how adaptive the algorithm should be. From this examination it is possible to specify some of the attributes that a routing algorithm should possess for an interconnected FDDI LAN environment. Only after this framework specification is it possible to begin designing a suitable algorithm.

3.1 The Ideal Routing Algorithm

Bell and Jabour define the attributes of an ideal routing algorithm as follows [31]:

-Correctness: The algorithm must work.

-Computational simplicity: The algorithm must use a minimum amount of processing capacity at each router so as not to unduly increase packet delay.

-Adaptiveness to changing traffic and topologies or robustness: The algorithm must be able to adapt to changing levels of traffic flow through the network and to find alternative routes when routers and/or lines fail or come back into service.

-Stability: The algorithm must be able to converge to an acceptable solution without excessive oscillation while adapting to changing traffic and topology.

-Fairness: The algorithm must be equitable to all users.

-Optimality: The routing algorithm should be able to provide the 'best' route that minimizes mean packet delay and maximizes throughput.

The attributes of fairness and optimality are often contradictory goals as can be seen in figure 3.1. Suppose that there is enough traffic between A and A', between B and B' and between C and C' to saturate the horizontal links. To maximize the total flow, the X to X' traffic should be shut off altogether. Unfortunately, this is somewhat unfair to X and X'. Some compromise between global efficiency and fairness to individual connections is needed [1].

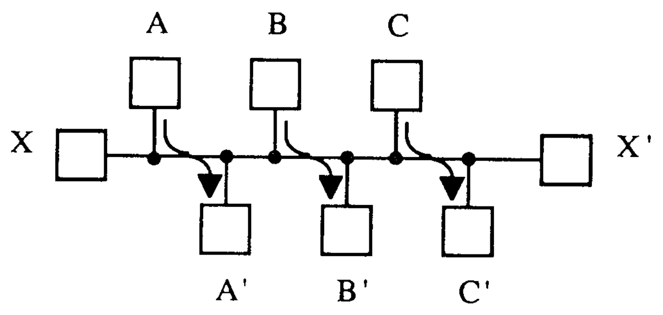


Figure 3.1 The trade off between fairness and optimality

The effects of good and poor routing strategies on average packet delay (quality of service) and throughput (quantity of service) are shown in figure 3.2 [30].

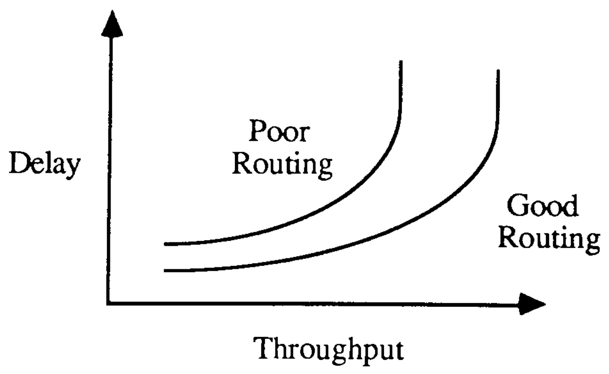


Figure 3.2 The effects of poor and good routing on delay and throughput.

It is obvious that it is not possible to produce an ideal algorithm. There are many trade offs that the network designer can make in order to optimize one thing or another. As such, they can only be used as points to keep in mind during the design process.

3.2 Inter-Network Characteristics

The choice of routing algorithm is highly dependent on the type of inter-network that the algorithm is expected to operate in [29]. To emphasize this, a routing algorithm for a mobile network in a hostile environment where nodes and links can at any moment be destroyed is very different from a routing algorithm for a computer communications network interconnected by reliable links.

A network comprised of interconnected FDDI LANs using leased lines would exhibit the following characteristics:

-Connection-oriented transport layer, connectionless network layer.

Most local area networks in use today utilize a connection-oriented transport layer and a connection-less network layer in the end system. Connection-oriented service is very similar to what happens in a public telephone network. The customer first dials a number to set up a connection. The connection is accepted and the two parties exchange data by talking. Finally the connection is broken. The two users are presented with the illusion of a dedicated, point to point channel that always delivers information in the order it was sent [1].

Connectionless service, on the other hand, is like the postal system. Each letter carriers the full destination address, and is totally independent of any letters that were sent before it. For this reason letters don't always arrive in the order that they were sent. If the postman accidently drops a letter, the postal system does not time out and send a duplicate. It is up to the users to sort the letters in order and re-send a letter if it doesn't arrive.

On a LAN, the transport layer in a host (say host A) sets up a connection with its peer process in the destination host (host B). Packets from host A travel down the transport layer to the physical medium gaining additional headers as it traverses each layer of the OSI model. This is sent across the medium to host B where the opposite occurs. Each layer in host B strips off the appropriate header as the packet passes up through the layers. If the packet is lost anywhere along the route, the transport layer times out and sends a duplicate. Likewise, if the packets don't arrive in order it is up to the transport layer to sort them out. This service ensures that packets which are passed up to the session layer are in order, error free without any duplications. One of the best examples of a connection-oriented transport layer and a connection-less network layer service is the well known TCP/IP (Transmission Control Protocol / Internet Protocol) protocol suite.

The type of service used by the transport and network layer often dictate the types of service used above and below these layers. Figure 3.3 shows the services commonly used by the other layers.

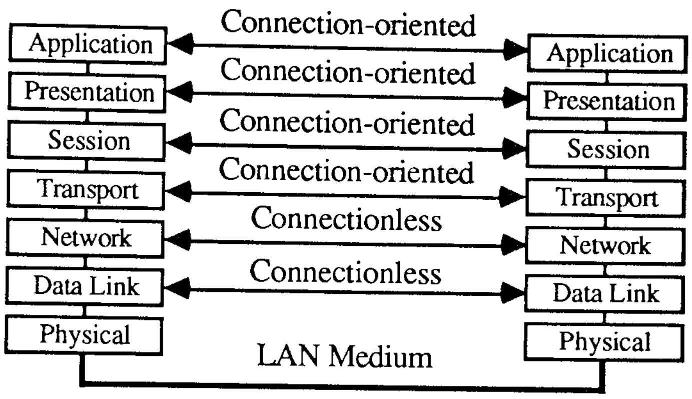


Figure 3.3 Commonly used LAN service types.

There are basically two different ways of arranging the link between remote LANs, one using connections the other operating connectionless. These methods are referred to as virtual circuits and datagrams respectively. It must be stressed that this is relevant to the internal operation of the subnet and is an independent issue to the type of service provided by the network layer. For example, it is possible to use a connectionless network layer coupled with virtual circuits. This may be used when very high reliability of the link is required. There is great debate over whether the network layer should provide a connection-oriented or connectionless service and whether datagrams or virtual circuits should be used within the subnet. However, the most common and simplest method for interconnecting LANs that use a connectionless network layer is to use datagrams within the subnet. Using datagrams means that the intermediate system can route successive packets independently. The transport layer responsibility has now transferred from coping with a single LAN to having to cope with an entire network. This can be seen in figure 3.4.

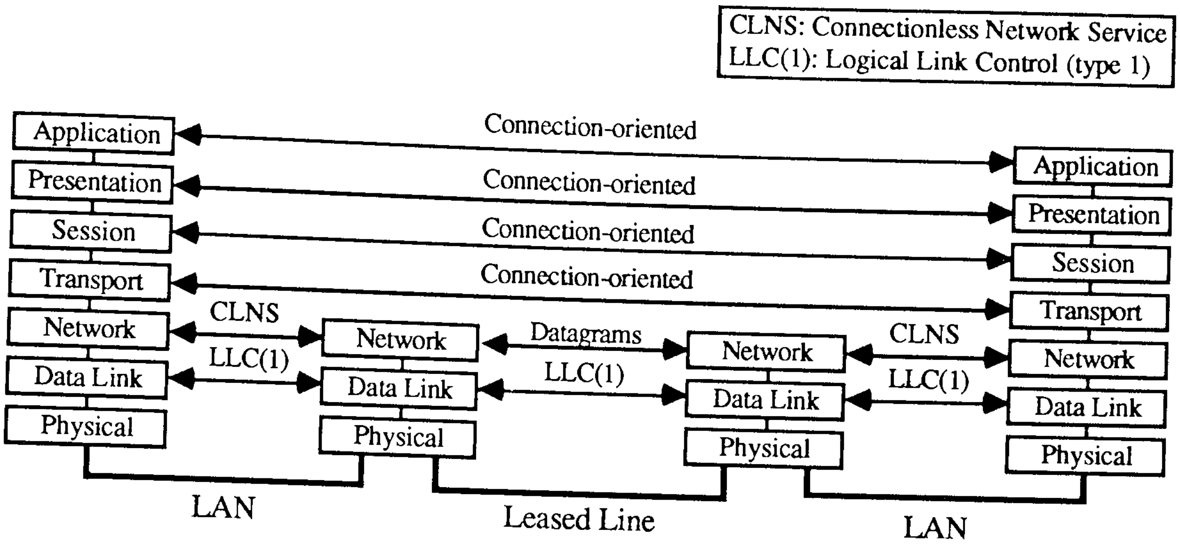


Figure 3.4 Service classes commonly used across a LAN internetwork.

- Links: In this study the links connecting the FDDI LANs are assumed to be lines leased from the public telephone company. These lines are usually very reliable and the failure rate is extremely low. The 2.048 Mbps Megastream links and the 64 kbps Kilostream links from British Telecom run at speeds approximately 50 and 1560 times slower respectively than an FDDI LAN. A Megastream link is the equivalent of 32 Kilostream links but only costs ten times as much.

-Topology: The topology of a LAN internetwork very often develops randomly. Links are invariably added between LANs with no global structure in mind [38]. The topologies also tend to be very stable in that links and routers are added and taken away from the network quite infrequently. Due to fairly high cost of the leased lines, the number of links used is normally kept to a minimum. For this reason, the resulting topologies are generally sparsely connected.

-Traffic loading: A rule of thumb for interconnected LANs is that, on average, 10% of the traffic is inter-LAN and the remaining 90% is destined for local stations [15]. If the number and speed of the links are chosen to minimize the cost and still provide a satisfactory throughput for this average load, then there is very strong possibility that periodically more inter-LAN traffic will arrive at the router than the network is able to handle. It is therefore extremely important that the routing algorithm is able to cope during times of heavy traffic loading.

- LAN traffic characteristics: FDDI can support a wide variety of traffic types, ranging from file transfers to packet voice and video transmissions, each one having different delay requirements [4]. File transfers would consider accurate delivery more important than a timely delivery. This is in contrast to packet voice and video where delay is critical and accuracy is less important. Traffic of this nature tends to arrive at a station in bursts. This means that during the course of a call several packets may arrive all at once followed by no arrivals followed by another burst of traffic.

- Routers: The routers in use today are far more reliable and have a lower failure rate than those used in the past. In addition, with the tremendous drop in cost, router memory is no longer the scarce resource that it once was. Router throughput, due to faster processor speeds, is also rapidly increasing.

Having defined the type of environment that the algorithm is expected to operate in, it is necessary to examine routing algorithm characteristics.

3.3 Routing Algorithm Characteristics.

Routing algorithms have many characteristics which are often used for classification purposes. There are many different methods of classifying routing algorithm; a partial list is shown below.

Centralized vs distributed control: In a centralized network, a control centre is responsible for gathering information about the entire network and constructing the routing tables. These tables are then downloaded to every switching node which are then used for packet forwarding. Centralized networks tend to be used when there is a single mainframe computer which many users wish to communicate with via dial-up links. The major difficulty with such schemes is the reliance on a single control centre to handle the network. If the control centre fails, the entire network may go down. This is not a problem in distributed networks where every switching node is responsible for making its own decisions. A router failure can be isolated and network operation can continue. Distributed networks are, therefore, far more robust than their centralized counterparts [1,39]. Since LANs are by their very nature distributed systems, there is little to be gained by employing a centralized routing algorithm.

Non-adaptive vs adaptive: A routing scheme may be non-adaptive in that packets always take the same route from source to destination regardless of the prevailing network conditions. These routing decisions are made at the network set-up time and not changed thereafter. Adaptive routing schemes allow the decisions about packet forwarding to change with time. Adaptive algorithms can be further differentiated on the basis of 1) frequency of adaptation, (ie by packet, by message, by session or by duration of network configuration and 2) type of adaptation (ie to topological changes to average network delays or to local queue lengths) [32,39]. Algorithm adaptivity is a very big issue that will be covered in depth later in this chapter.

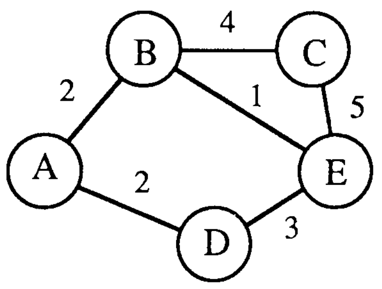
Global vs local information: The information that is used to make a routing decision can be local, global or a combination of both. Local information consists of the status of the outgoing links the associated queue sizes or the number of virtual circuits running through a router. Global information consists of network topology and the network status such as average link delays and nodal congestion. The choice between global and local information will be discussed in the adaptive routing algorithm section.

Topological Dependence: Some routing algorithms have been designed for a particular topology [40,90,91,92,93,94]. Such regular structures are common in virtual circuit wide area networks, less so in datagram WANs. In keeping with the assumption that most LAN internetworks grow randomly, only algorithms that are not topology dependent will be examined.

Therefore algorithms that are distributed, adaptive in some way and do not require a specific network topology will be considered. Before looking at the adaptive algorithms, it is necessary to discuss the four procedures that make up a routing algorithm.

3.4 The Four Routing Procedures

Most distributed routing algorithms require that a database is maintained at every router in the network which lists each destination, the distance to that destination and the next router to forward the packets toward. This can be seen in figure 3.5.



Routing Table for Node A

destination	distance	next node
B	2	B
C	6	B
D	2	D
E	3	B

Figure 3.5 Routing table for a simple network.

Maintenance of routing table databases involves four algorithms that work more or less independently and yet support each other by exchanging services or information [31]. These four algorithms form a feedback loop as shown in figure 3.6. The diagram appears to suggest that the different procedures are well defined and discrete, however, like the

OSI model, in a real routing algorithm functions of one procedure very often blend into those of another.

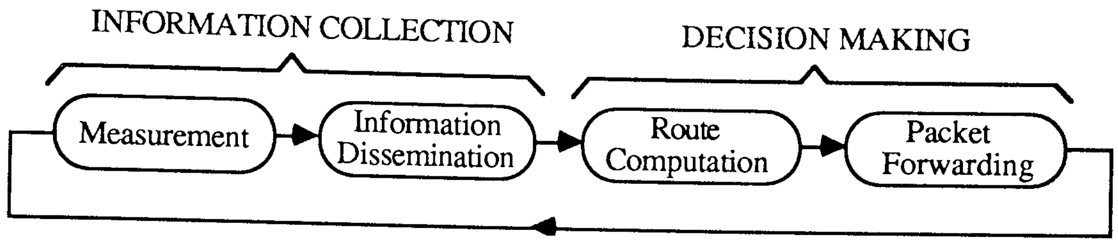


Figure 3.6 The four procedures that make up a routing algorithm.

The distance measurement procedure monitors and collects certain network parameters according to the routing metric used. This collected information is then distributed over the entire network by the information dissemination procedure. In each router, the route computation procedure then constructs the routing table based on the received information. The packet forwarding procedure actually routes the traffic to the next router based on the routing table. Papers dedicated to each of the independent procedures have appeared in the literature [36,41,59].

3.4.1 Measurement

There are many metrics or costs that can be used by the measurement procedure in order to define a path from a source to a destination. These include such things as the number of hops, link speed, link security, link reliability, link expense, link error rate and link delay to name but a few. The number of hops metric refers to the number of routers that must be traversed when forwarding a packet from a source to a destination. In some networks minimizing the hop count results in a lower packet delay. The numbers next to the links in figure 3.5 represent the cost of using those links. The metric in this network is undefined and could be any one of the above (with the exception of the number of hops metric).

Link costs can either be defined by the network manager, or the router itself may be responsible for determining the cost of the link. The latter is often true in the case of using network delay as the routing metric. In order to determine the link delay, a high priority packet is sent out from a router across a link to an adjacent router with the time it was sent stamped on it (time A). When the neighbouring node receives the packet it sends it back immediately to the initiating router with the time received within the packet (time B). The delay for that particular link is simply time B - time A.

3.4.2 Dissemination

Having measured the cost of the link according to the metric, it is necessary to inform all other nodes of this information. This can be carried out using either a co-ordinated update procedure (CUP) or an independent update procedure (IUP) [58]. Using a CUP nodes update their routing tables in a strictly controlled manner. This is used to reduce a few of the negative affects that some route computation procedures can produce. An IUP is a complete free-for-all, nodes may update their tables whenever they so desire. The use of a CUP or an IUP is highly dependent on the route computation procedure, both of which will be covered fully in subsequent chapters.

3.4.3 Route Computation

Route computation is probably the most important of the four procedures that make up a routing algorithm. Very often this procedure is specified first and the remaining three procedures are thrown in as an after thought.

The route computation process can be accomplished in a variety of ways. Two of the most popular methods are bifurcation and shortest path algorithms [29].

Bifurcated routing is used when the network-wide average time delay is to be minimized. It is a highly complex technique which utilizes traffic flow models and uses methods such as gradient projection and flow deviation to calculate the traffic flow [42,43,44]. These algorithms are best used in networks where the traffic flows correspond to some well known pattern. These algorithms have been implemented in only a very few communication networks [30].

Shortest path algorithms on the other hand are widely used in computer communications networks. They are much simpler and have much more modest goals. They are used in a wide variety of networks regardless of whether they are centralized, distributed, adaptive, non-adaptive etc. [32,33,34,52].

Routers can be configured in a LAN internetwork in a variety of different ways. Some routers are capable of handling only a single low speed link, whilst other more complex routers may be able to support several high speed links. This can be seen in figure 3.7.

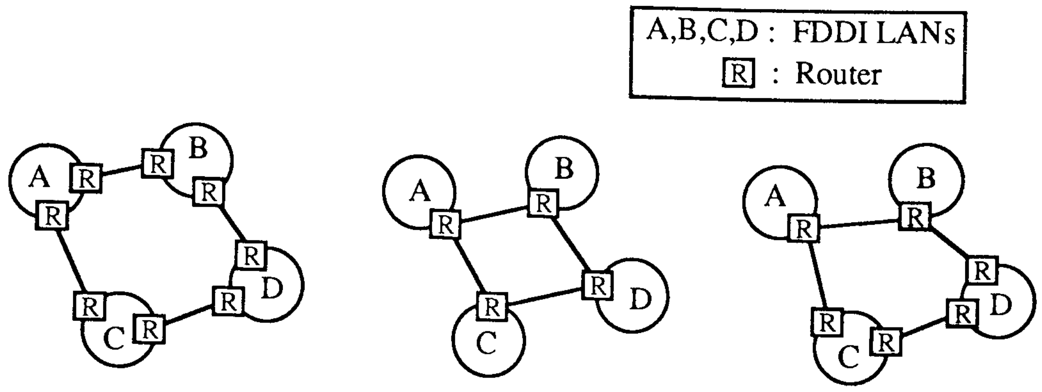


Figure 3.7 Possible router arrangements on a LAN.

However, regardless of how the routers are set up, it is possible (within the context of graph theory) to ignore the actual physical implementation of the routers on the LAN and treat them as a single packet switching device. This can be seen in figure 3.8, which could represent any one of the networks shown in figure 3.7.

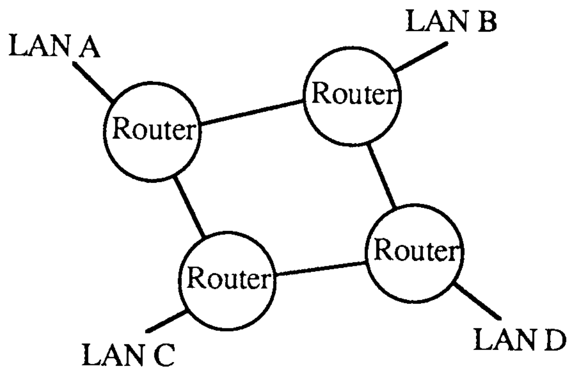


Figure 3.8 The combined router representing all routers on a single LAN.

The network then becomes a directed graph as shown in figure 3.9. The vertices of the graph represent the 'combined router' on the LAN and the arcs represent the links connecting the LANs. From a graph theory point of view, the LANs can be completely ignored.

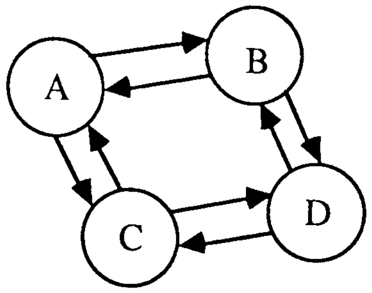


Figure 3.9 Directed graph representing a LAN internetwork.

If each arc has a weighting (the metric) associated with it then it is often desirable to find the minimum length path from one vertex to another. This is termed the shortest path problem and has been the subject of much research. Two of the most common techniques to find the shortest path in a network are the Bellman-Ford and the Dijkstra algorithms [45,46,47]. Regardless of what algorithm is used, all shortest path algorithms converge to the same solution for a given network [48,49,50,51]. The difference lies in the iterative process used to converge to the final solution. This can be seen in figure 3.10, which shows the Bellman-Ford algorithm and the Dijkstra algorithm converging to the shortest paths using different methods.

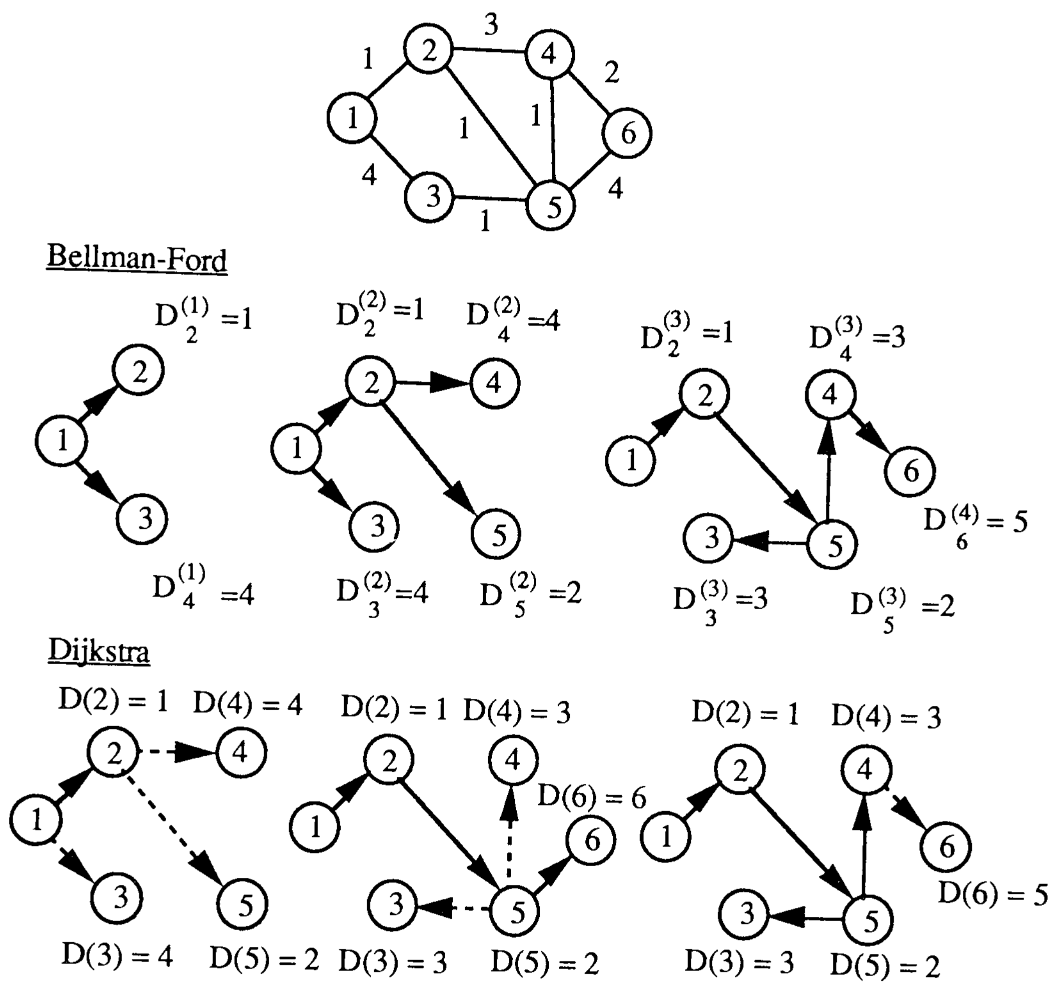


Figure 3.10 Bellman-Ford and Dijkstra's shortest path algorithms shown converging in a simple network.

The Bellman-Ford algorithm is often termed a distance-vector algorithm. Networks that use distance vector algorithms require routers to have knowledge of their neighbouring nodes only. Dijkstra's algorithm, on the other hand is a link-state algorithm which necessitates that all routers store the entire topology database. Distance vector and link state algorithm will be the subject of subsequent chapters.

3.4.4 Packet Forwarding

Most networks that run one of the above algorithms simply forward the packets along the shortest path towards the destination [37]. The forwarding database is the 'Next Node' column in figure 3.5. For a packet from node A destined for node C the next node to send a packet towards is node B. This is one area in particular where very little research has been done.

A distance-vector or link-state shortest path algorithm will be used to carry out the route calculation procedure. The three remaining procedures will remain undefined until further consideration of the shortest path algorithm in subsequent chapters.

3.5 Algorithm Adaptivity

Routing algorithms can either be non-adaptive or adaptive. Non-adaptive algorithms constrain packets to follow the same route from source to destination regardless of the prevailing network conditions. Adaptive algorithms, on the other hand, can adjust the route taken based on gathered information concerning packet delays, link utilizations, node and link operational status etc. Adaptive algorithms are far more robust and can provide a higher network performance than their non-adaptive counterparts. These can be differentiated on the basis of how frequently new routes are selected (ie by packet, by session, by message, by topology) and by what they adapt to (ie local queue lengths, average network delay, topology).

The degree of routing algorithm adaptation varies greatly and is a very important design issue. Some algorithms adapt only when a topological change occurs, whilst others attempt to adapt to changing network delays. These obviously represent the two extremes of what an adaptive algorithm is capable of providing. Each time that an algorithm decides that it is time to adapt (to topology, delay etc) the four procedures in the previous section must be carried out. In designing an adaptive routing algorithm it is necessary to strike some balance between the completely adaptive algorithm and one that adapts very infrequently.

Routing has changed considerably since the first algorithms suggested by Baran in 1962 [35]. In current WANs and LANs interconnected by leased lines, the transmission facilities are expensive relative to computer processing. Complex routing and flow control algorithms have evolved to efficiently use the transmission facilities. In earlier networks processing was more expensive and much simpler mechanisms were used [37]. Many of these early algorithms didn't include all four of the procedures discussed in the previous section. However, there are characteristics of these algorithms which make them relevant to todays networks. A brief overview of some of these well known early algorithms follows.

The simplest routing mechanism ever developed was random routing. In this algorithm packets that arrive at a router are sent out on any outgoing link chosen at random. This selection is made with complete disregard as to the packets final destination. It is a simple, but inefficient, algorithm since packets take unnecessarily long paths.

Another simple routing mechanism is the so-called flooding algorithm. Using flooding, a router would send a received packet out on all the links that it was connected to. This was very wasteful of bandwidth but it made the algorithms very simple indeed. Since all possible paths were taken, it ensured that packets always took the shortest route to the destination.

In hot-potato routing, when the shortest path link to the destination was busy, any one of the available links is selected at random. This technique can adapt rapidly to changes in load or topology and is only dependent on the local information available at each router [1,39].

A more advanced algorithm than the above three was the shortest-queue plus bias algorithm. This algorithm required that each router calculate the shortest path to each destination in terms of number of hops. Packets were then forwarded based on the local queue lengths of the outgoing links. This algorithm used global information for changes in topology and local information for adapting to traffic fluctuations [53].

All through the 70's many networks were attempting to become more and more adaptive taking advantage of the increasing cost reduction of computer processing. This is best illustrated by the original ARPANET routing algorithm which represents the ultimate in adaptive routing algorithms The algorithm measured the delay on each of its links every 2/3 of a second and sent this information to its adjacent routers. The shortest delay paths were calculated from this information and packets forwarded towards the destination based on these routes [37].

The frequency with which a routing algorithm adjusts its routes to network delay is reflected in the amount of control information transmitted by the routers in order to keep the routing tables up to date. This information may be transmitted on a per packet, per message, per session or per topology basis. In addition it may be sent periodically or aperiodically based on some threshold value, such as the number of packets processed by the node. Figure 3.11 shows the additional control information transmitted by the various techniques. The original ARPANET algorithm adapted by packet on a periodic basis and generated the most control information. The shortest-queue plus bias and the hot-potato algorithms only require control information when there is a topology change. Flooding and random routing require no control information at any time, regardless of topological changes. For this reason they are not shown in the figure.

The adaptive by message and the adaptive by session algorithms for virtual circuit networks are also shown for comparison. The latter are used to show that as the frequency of updates decreases so too does the amount of control information. It is important to point out that a datagram network has no concept of what a session is or what constitutes a message. In a datagram network, only the transport layer has any notion of what these terms mean. For this reason, it is only possible to have a datagram network adapt on a per packet basis or when there is a topology change.

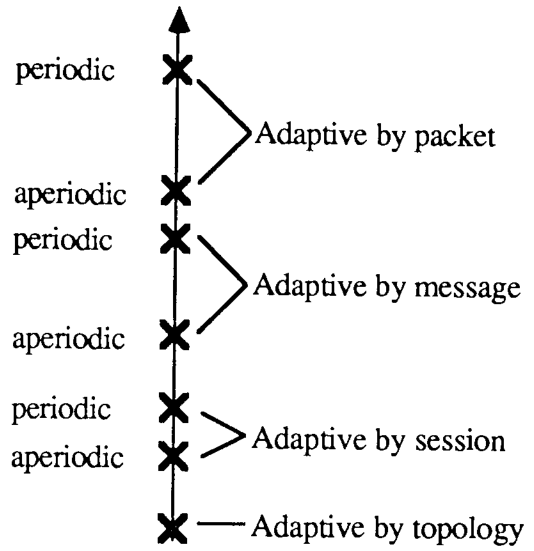


Figure 3.11 The increasing number of control messages required for various adaptive routing schemes.

The data packets transmitted by the various routing mechanism use the links with a varying amount of efficiency. The least efficient is the flooding mechanism that sends packets out on all links. Random routing is slightly more efficient since only a single outgoing link is used. Hot-potato is more efficient again since, during times of low congestion, the shortest route is taken. The most efficient of the algorithms is the shortest-queue plus bias and the adaptive by packet algorithms, which always attempt to take the shortest path from source to destination. The five schemes relative to each other are shown in figure 3.12.

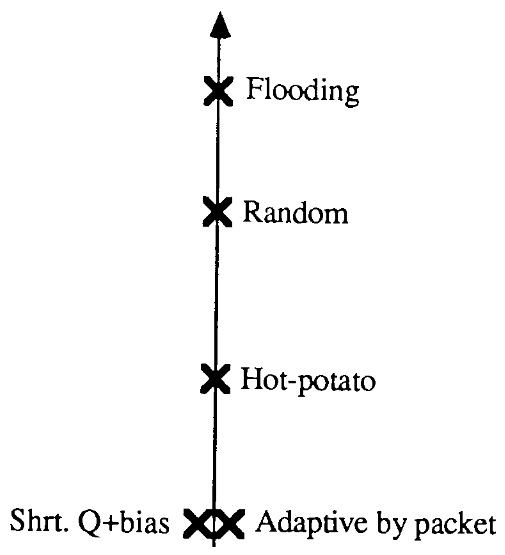


Figure 3.12 Increasing amount of data bits sent by various adaptive routing algorithms.

One of the main constraints to throughput in a network is the amount of processing performed by the routers. For this reason it is necessary to look at the processing burden imposed by the five routing algorithms. Processing can be done either at the edge of the network (hosts) or within the network itself (routers). The source rate is defined as the speed at which the hosts must perform their processing function in order to keep up with the packet arrival rate. In a datagram network this refers to the function performed by the transport layer to sort packets, send acknowledgements etc. The network rate is the processing speed required by the intermediate routers on the path between the source and destination. At the network rate, routing mechanisms must maintain queues, select the best route, perform measurements on the network and change the route in adaptive schemes. In order to reduce the delay imposed by the routers it is desirable to restrict the network rate to a low value [39]. The various routing schemes are shown in figure 3.13.

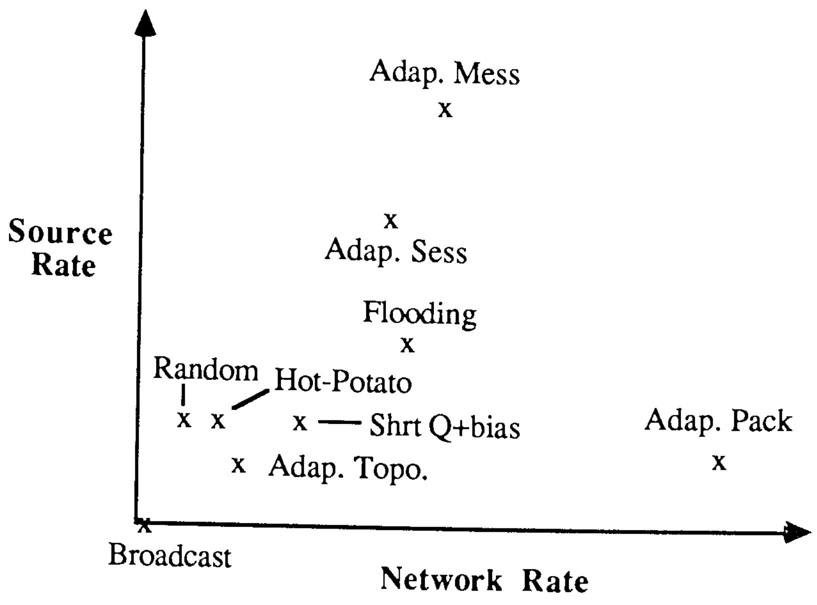


Figure 3.13 Source rate vs network rate processing requirements.

A broadcast LAN is shown at the origin of the figure for comparison. LANs require the least amount of processing since there is no route selection to make and packets never arrive out of order. Adaptive by packet algorithms require the most amount of processing at the network rate in order to maintain queues and make routing decision at each node. Algorithms that adapt to network delay by message and by session also maintain these queues but the frequency of adaptation is much lower. For datagram networks, flooding requires the highest source processing rate in order for the transport layer to sort out the duplicate packets that have arrived. Random, hot-potato, shortest-queue plus bias and algorithms that adapt only when a topology change occur require the least amount of network rate processing.

By combining the graphs for control information from figure 3.11, the amount of extra data packets from figure 3.12 and the source rate vs network rate graph of figure 3.13 a bandwidth inefficiency vs network rate graph can be produced (figure 3.14). This sums up all the above discussion in one graph. It may appear that those schemes that are closest to the origin such as adaptive by topology algorithms will provide the best algorithm. However, this graph does not give any indication of how well the algorithm will perform in terms of delay and throughput. It merely serves to indicate what schemes use the bandwidth efficiently and require the lowest amount of processing at the network rate.

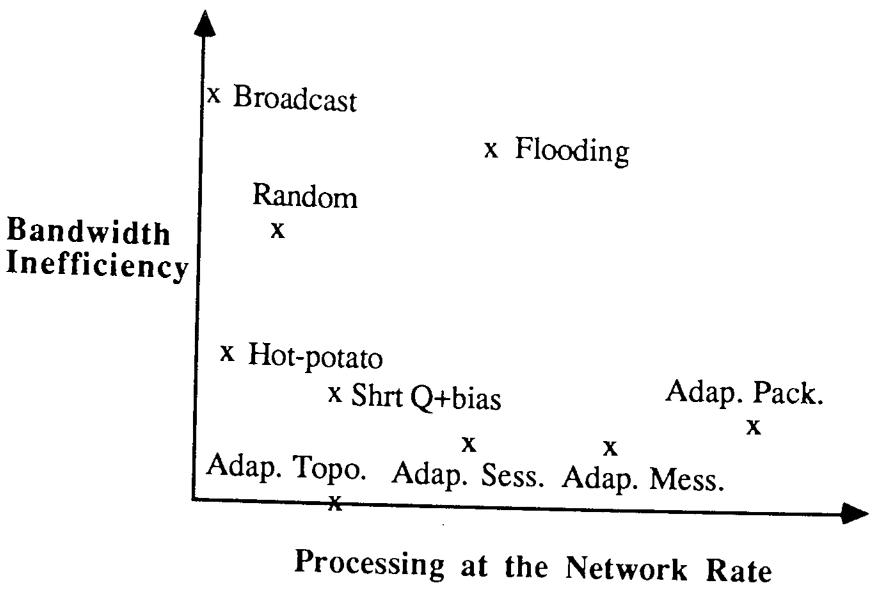


Figure 3.14 Bandwidth inefficiency vs network rate processing.

The most promising schemes appears to be adaptive routing algorithms that use global information for long term adaptivity to topology and local information regarding queue lengths for traffic fluctuations. This suggests the use of algorithms such as hot-potato and shortest-queue plus bias for use in a datagram network.

3.6 Conclusion

There are many different routing algorithms available for the network designer to choose from. The selection of which is dependent on a variety of factors which this chapter has outlined for one particular type of network. For a network of interconnected FDDI LANs over leased lines the following goals for a routing algorithm are desirable:

-distributed

-able to route datagrams

-independent of topology

-adapts to topology changes on a global basis

-adapts to traffic fluctuations on a local basis

-uses a shortest path algorithm for route computation

-able to cope with heavy traffic loading

-able to handle bursty traffic sources

It must be remembered that the above are in addition to the goals of the ideal routing algorithm discussed in section 3.2.

The above goals provide a framework for examining routing algorithms within the literature. This does not mean, however, that routing algorithms will be rejected if they do not meet all of the above criterion. Algorithms can usually be modified in some way to cope with different network environments.

The next chapter will examine the Bellman-Ford, distance vector algorithm for use in an interconnected FDDI environment.

Chapter 4

Distance-Vector Routing Algorithms

4.0 Introduction

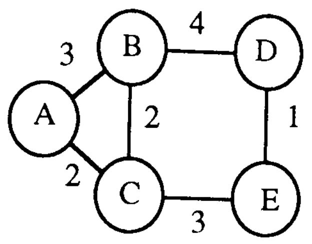
The distance-vector algorithms discussed in this chapter are all based on the work of three people, R. Bellman, L.R Ford and D.R Fulkerson [45,46,47]. These researchers developed and modified the basic algorithm back in the late 1950's and early 60's. For this reason, the modified algorithm is referred to in the literature as 'Bellman-Ford', 'Ford-Fulkerson' or just simply 'Ford'. These are all effectively the same algorithm. In this thesis the algorithm is referred to as the Bellman-Ford algorithm.

The first implementation of a Bellman-Ford algorithm in a communications network was by the Advanced Research Projects Agency division of the U.S Department of Defence in 1969 (the so-called ARPANET) [37]. Many of the algorithms developed for the ARPANET were subsequently used in other computer communication networks. The ARPANET algorithm has gone through many changes in its 20 year history. This chapter will discuss the original ARPANET algorithm and various approaches used to improve upon the convergence of the Bellman-Ford algorithm.

4.1 Overview of the Bellman-Ford Shortest Path Algorithm

The basic Bellman-Ford algorithm first implemented in the ARPANET (circa 1969) operates as follows [53]:

Each node holds two tables, a distance table and a route table. The distance table of a given node (say node A) contains a list of all the other nodes in the network and the distances to those nodes via A's neighbouring nodes. The routing table stores the value of the minimum cost to each destination along with the corresponding next node. The distance and routing tables for a sample network are shown in figure 4.1. Note that these two tables would be stored at node A.



Distance table

	B	C
A	-	-
B	3	4
C	5	2
D	7	6
E	8	5

Routing table

-
3B
2C
6C
5C

Figure 4.1 A simple five node communications network and the associated distance and routing tables for node A.

When a change in link cost occurs, the two nodes next to the link recalculate their distance tables (route computation procedure). If there is a new minimum cost from source to destination (ie the route table has changed) then this information is immediately sent to all adjoining nodes (route dissemination procedure). This is then repeated throughout the network until the algorithm converges to the new minimum cost paths. A complete example of the ARPANET algorithm converging to shortest paths after a link failure is shown in appendix A.

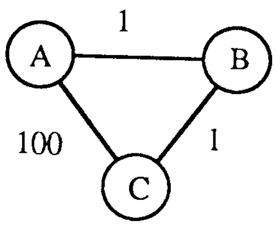
There are several advantages to the basic Bellman-Ford algorithm. First of all the algorithm is incredibly simple. It does not require global information in order to converge the algorithm, it only needs to know who its neighbours are and the cost of the adjoining link connecting them. From this information it is possible to work out the shortest paths to all nodes. This means that the nodal storage requirement in terms of memory needed at each node is very low.

Secondly, the algorithm is truly distributed in that each node benefits from the computation done by the neighbours preceding it. This reduces the amount of processing that needs to be done on the routing information that arrives at each node.

The disadvantages of the basic Bellman-Ford algorithm are extremely unpleasant. It suffers terribly from two problems when cost increases occur; routing table looping and what is known as the 'bouncing effect'. The bouncing effect is also known in the literature as 'counting', 'ping-ponging' and 'reverberation'[37,60,62,63].

These two concepts are best understood by a network example. Consider figure 4.2. Under static conditions, the routing tables point to the shortest paths between source and destination. However, consider what happens when link AB fails. Node B immediately assumes the second best path to A is 3 hops away via node C. Node B does not know that the path from C to A actually loops back via B. As such a so called 'routing table loop' is formed.

Looping is wasteful of network resources and causes excessive delay for the data packets.



Node A

	b	c	
a	-	-	-
b	1	101	1b
c	2	100	2b

Node B

	a	c	
a	1	3	1a
b	-	-	-
c	3	1	1c

Node C

	a	b	
a	100	2	2a
b	101	1	1b
c	-	-	-

Figure 4.2 The problems of looping and vector bouncing.

The second problem with these algorithms is the bouncing effect. Assume again that link AB fails. B sets its routing table for destination A to 3. Since this has changed the route table, B sends this information to all its neighbours, namely node C. Node C takes this value and adds the link cost between node B and C which results in a count of 4. Node C detects a change in its route table and sends this information to nodes A and B. Node B then repeats this procedure again. This is an example of the bouncing effect. This problem will not cease until the value exceeds 100, whereupon the nodes realize the best path from C to A is actually via node A [65].

Now consider what happens if during this counting procedure between nodes B and C link AC fails. It becomes obvious that this counting will never cease since there is no upper bound on the value to count to. In the previous example the nodes only had to count up to 100 before realizing that a better path was available. In this example no other paths exist, the nodes will count forever. This is the so-called counting to infinity problem. The counting to infinity problem is a special case of the bouncing effect.

The above problems were particularly acute in the ARPANET which attempted to adapt to network delays every 2/3 of a second. This required update vectors to be passed back and forth between nodes. During times of link cost increases, it become apparent that a large amount of packet looping was taking place.

Recall that in this study, to reduce nodal processing at the network rate, an algorithm will be designed that adapts to traffic fluctuations based on local information only. However, if a link or node failure occurs, then it is essential that the algorithm converges as quickly as possible to the shortest paths.

It should be noted at this point that loop freedom means no data packet looping during the algorithm convergence interval. Looping of data can never be completely avoided as demonstrated by figure 4.3. In that figure a packet is shown on its way to destination 5 via node 4. Just as it reaches 4, link 4-5 fails. Obviously the packet must either be returned to node 3 or dropped at node 4. Here we take an algorithm to be loop free if it does not have routing table loops [61].

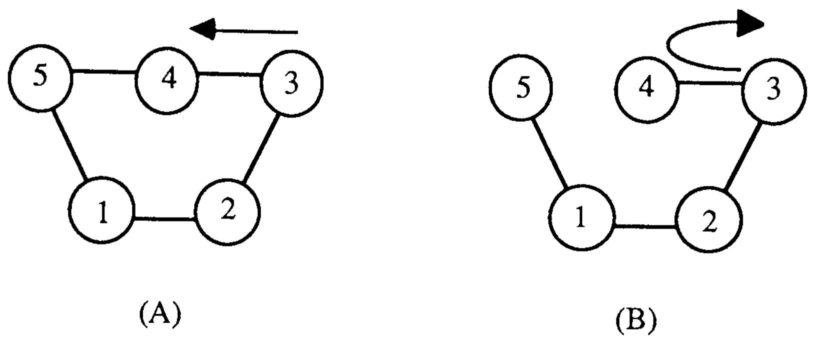


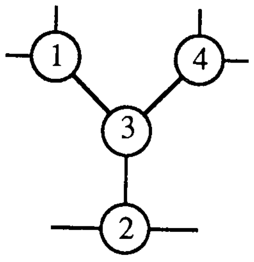
Figure 4.3. Packet looping during link failure (A) before failure (B) after failure.

The problems of looping, vector bouncing and the counting to infinity problem have been known for some time. The remainder of this chapter will examine the various approaches suggested in the literature for dealing with these problems. These approaches involve modifications to the standard information dissemination procedure and the route computation procedure; the reason being that the above problems are related to the way in which the routing tables are updated. The measurement procedure and packet forwarding procedure are of little interest since they are not involved in the routing table updates. Some of the methods discussed have been implemented in computer communications networks whilst others are discussed in research papers.

4.2 Previous Work on the Bellman-Ford

One of the first attempts to reduce the looping problem was by the ARPANET community [37]. They realized that routing table loops occurred when nodes were using stale information about the network. In order to alleviate this problem they introduced the idea of hold-downs. Using this method, a node would continue to use the best route to a given destination for some time period after a cost increase had occurred. This time period was approximately two seconds which allowed surrounding nodes the chance to purge any out-of-date information before allowing them to accept new information. This method was found to respond quickly to cost decreases ('good news') but less satisfactory to cost increases ('bad news').

Cegrell [55] presented a very simple solution to the two node looping problem, termed split-horizon. Using this method, a node sends only relevant information to adjoining nodes. It does not send the minimum cost information to a node if that node is the one to be used for routing packets. It instead sends the next best path information. This can be seen in figure 4.4.



Distance table at node 3

	1	2	4
1	5	10	10
2	20	5	15
3	0	0	0
4	15	15	5

Figure 4.4 Distance table for sample network.

Using split horizon, node 3 sends (10,5,0,5) to node 1. This allows node 1 to get some idea of what the network looks like beyond node 3. The original ARPANET algorithm sent all adjoining nodes the minimum values in its tables (ie 5,5,0,5) regardless of which nodes they referred to. The split-horizon mechanism was first implemented in the TIDAS network of the Swedish power system. It has subsequently been used in many networks and by manufacturers of routers for interconnected LANs.

Naylor [54] uses the split horizon method to reduce the chances of local loops and a loop-checking mechanism in order to reduce the incidence of multi-node loops. This involves sending a control packet along the new updated route. If the control packet is acknowledged by the receiver then the new route is permitted. If, however, the packet is not acknowledged or it returns to the sender then the route is ignored.

The routing algorithm developed for the MERIT computer network by Tajibnapis [56] uses a hop count metric to differentiate between potential paths. Hop count refers to the number of links that must be traversed from a source to destination. The MERIT network only adapts to changes in topology and does not try to route traffic based on a delay metric. By using a minimum hop metric, it is possible to reduce the counting to infinity problem since every node knows how many nodes are in the total network. If a path is specified which has a greater value than the total number of nodes in the network then the path must contain a loop and is ignored. Therefore the counting problem is still present but it ceases after a certain value.

One of the main reasons that the previous algorithms develop routing table loops is the method in which updates are processed. A node, upon detecting a cost change, updates its table, selects a new node and then forwards a cost increase message to its neighbours. These algorithms are termed independent update procedures (IUP). Merlin and Segall [57] introduced the concept of a co-ordinated update procedure (CUP) in order to reduce the looping problem. This algorithm guarantees routing table loop freedom at all times by restricting the manner in which tables are updated. This is accomplished by defining each node as the SINK to a tree comprised of all the other nodes. The SINK trees for figure 4.1 are shown in figure 4.5. A node detecting a change in the cost directed towards a destination would send this information to the SINK concerned. When the SINK receives this information it sends out a freeze message up the tree until the end of the tree is reached. A freeze message informs nodes that a change in link cost has occurred and they are not permitted to modify their routing tables in any way. The final node of the SINK tree is frozen and permitted to select a new next node to the destination, following this selection it is then unfrozen. This process continues back down the tree towards the SINK. It is a very complex protocol which requires large numbers of control messages and computation. It does, however, guarantee loop freedom even during topological changes.

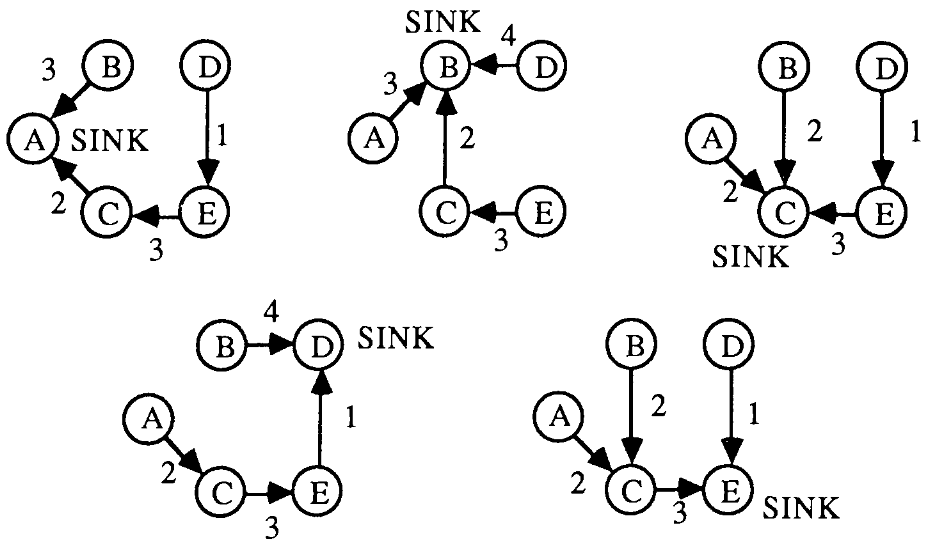
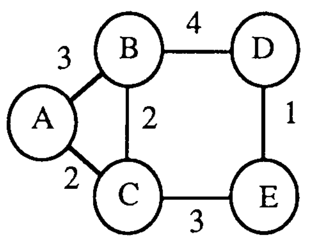


Figure 4.5. The SINK trees of figure 4.1.

Jaffe and Moss [58] extended this idea of the CUP. Instead of requiring the SINK to perform the calculation every time a link change occurred, Jaffe and Moss found that looping only occurred with Bellman-Ford algorithms during a link cost increase. They therefore used a standard Bellman-Ford algorithm for link cost decreases (IUP) and a CUP for link cost increases. This made the algorithm much simpler than the previous Merlin algorithm. This algorithm was designed for virtual circuit networks where loop-freedom is much more essential than with datagram networks.

In Hagouel's [61] PhD thesis he introduced two algorithms to cope with the looping problem. Algorithm A is similar to the Tajibnapis algorithm but allows for link cost values greater than one. He reduced the looping problem by using two flags Set and Reset. If a cost increase occurs then the flag is changed to Set and sent to all neighbours. The purpose of the flag is to trigger alternative messages from the nodes receiving that message. This avoids an endless loop that will result if D's message does not affect C's route table.

Hagouel's algorithm B [61] used an IUP method but introduced the idea of maintaining at each node a source tree. These trees are exactly analogous to Merlin's SINK trees. Using this method the entire route from source to destination can be defined. The nodes in this method not only store the next node to the destination they also store the identification of the node just before the destination. This can be seen from the table that would be stored at node A from figure 4.6 below.



Dest.	Cost	Next	Final
A	-	-	-
B	3	B	A
C	2	C	A
D	6	C	E
E	5	C	C

Figure 4.6 Data structures used for source trees.

By tracing back from the destination using the final node column it is possible to build up the entire path from the source. For example to get from node A to node D the path D,E,C,A can be used. This method helps reduce the problem of looping since the entire path is known from source to destination. Hagouel's method requires a known upper bound on the length of the shortest paths, however, which is somewhat restricting.

The work of Cegrell, Tajibnapis, Merlin and Hagouel form the basis for many of the other algorithms left in this section. The remaining algorithms will therefore not be discussed in depth. An example of each of these algorithms shown converging following a link failure is shown in Appendix A.

Sloman and Andriopoulos [60] used a Tajibnapis type algorithm and modified it to try and cope with the problem of looping and bouncing. The method suggested is basically a concatenation of split horizon, hold downs and the broadcasting of a failure message to all nodes. The algorithm is designed specifically for interconnected LANs and utilizes a minimum hop metric. The algorithm only adapts to topology changes and no attempt is made at link delay measurements.

Schwartz [29] introduced the predecessor algorithm for coping with the problem of two node looping. It is similar to the split horizon method and can only reduce the chances of two node looping. It cannot guarantee loop freedom during a topology change.

Garcia [66,97] used Hagouel's source tree mechanism to calculate the shortest paths to each destination. These shortest paths are then broadcast to neighbouring nodes. Using this algorithm the bouncing effect is somewhat reduced.

Shin and Chen [67,69] used a similar method to Garcia [66] whereby the sequence of nodes making up the shortest paths were broadcast to all nodes. In addition, this was coupled to the Cegrell split horizon method. Like the Garcia [66] approach, it is somewhat inefficient.

Chen [67] used the source trees coupled with the idea of searching for new paths only from a set of paths which were 'simple'. Simple paths by definition do not contain loops, be they two node or multi-node loops.

Rajagopalan and Faiman [65] used source trees and the optimality principle in order to ensure that only optimal shortest path routes were used. Obviously paths with loops in them were considered sub-optimal. This is not too dissimilar to Chen [67].

Garcia [72] used a co-ordinated update scheme in order to reduce the problems of bouncing and packet looping. In addition to a CUP, nodes are only permitted to choose a new neighbour if the path presented offers an equal cost or lower cost than what the table contained prior to the update.

Awerbuch [68] introduced a method which employs a dynamic synchronizer. This is similar to using a global clock which is distributed in each of the nodes. This is claimed to be the fastest implementation of the Bellman-Ford algorithm.

Humblet [70] used Hagouel's source trees to reduce the looping problem. The method presented does not guarantee routing table loop freedom during topology changes.

The various mechanisms used to reduce the occurrence of routing table loops are summarized in table 4.1.

Algorithm	Basic method used to reduce routing table loops.
McQu[37]	Hold downs
Cegr[55]	Split horizon
Nayl[54]	Loop checking packet
Taji[56]	Maximum number of hops
Merl[57]	Internodal co-ordination
Jaff[58]	Internodal co-ordination
Hagoa[61]	Set and reset flags
Hagob[61]	Source trees
Slom[60]	Hold downs, split horizon and broadcast of failure reports
Schw[29]	Split horizon
Garc[66]	Broadcast of shortest path trees to neighbors
Shin[63]	Broadcast of shortest path trees to neighbors
Garc[97]	Include next node information in updates
Raja[65]	Source trees and optimality principle
Chen[67]	Source trees and simple paths
Garc[72]	Internodal co-ordination
Awer[68]	Global clock (dynamic synchroniser)
Humb[70]	Source trees

Table 4.1 Summary of methods used to reduce routing table loops in Bellman-Ford algorithms.

Table 4.2 summarizes the information distribution and the distance measurement procedures of the various routing algorithms. The route computation procedure for every algorithm is, of course, the Bellman-Ford and packets are forwarded along the shortest paths.

Algorithm	Information Distribution				Distance Measurement	
	Updates				Link Cost	
	IUP	CUP	Event	Periodic	Unity	Any
McQu[37]	●			●		●
Cegr[55]	●			●		●
Nayl[54]	●			●		●
Taji[56]	●		●		●	
Merl[57]		●	–	–		●
Jaff[58]		●	–	–		●
Hago[61]	●		–	–		●
Slom[60]	●		●		●	
Schw[29]	●		–	–		●
Garc[66]	●		●		●	
Shin[63]	●		–	–		●
Garc[97]	●			●		●
Raja[65]	●		–	–		●
Chen[67]	●		–	–		●
Garc[72]		●	–	–		●
Awer[68]		●	–	–		●
Humb[70]	●		–	–		●

Table 4.2 Information dissemination and measurement processes for the Bellman-Ford algorithms.

4.3 Algorithm Discussion

A variety of different methods have been presented in the literature to cope with the problem of routing table loops in the Bellman-Ford algorithm. These vary in complexity and how well they cope with the looping problem. The simplest algorithm is the IUP, minimum hop algorithm of Tajibnapis which can halt the counting to infinity problem. By far the most complex is the Merlin-Segall co-ordinated update scheme which guarantees loop freedom even during times of topological changes.

The algorithms that show the most promise are those based on Hagouel's source trees. These represent the middle ground, in terms of complexity, between the Cegrell and Merlin-Segall algorithms. However, it has been shown by Garcia [72] that these algorithms do not stop the counting problem, but simply reduce it.

Some researchers have, after presenting a new Bellman-Ford algorithm, even suggested that the Dijkstra algorithm is preferable if enough memory is available at each node. For a

LAN internetwork this will certainly be the case. Bellman-Ford algorithms provide satisfactory performance for networks where only minimal memory is available at each node.

In this way the Bellman-Ford algorithm has degenerated into nothing more than an interesting puzzle. Humblet states " ..we prefer the topology broadcast algorithm if enough memory is available" [70] and Garcia says " If enough memory is available at each node then the Dijkstra algorithm offers a better solution to the routing problem" [71]. Humblet also says "many modern networks avoid much of the problem by broadcasting the whole topology to all nodes. It is still interesting to try to modify the basic Bellman-Ford algorithm to prevent looping". The purpose of this work is not to find some interesting solution, but to develop a better algorithm for interconnected LANs.

4.4 Conclusion

An overview of the ubiquitous Bellman-Ford routing algorithm has been presented. This is a very simple algorithm which suffers severely from the cost dependent looping problem. Many researchers have attempted to address this problem with varying degrees of success. The problems can be eliminated completely with some techniques but the algorithm complexity makes these prohibitive for actual network implementation. Many researchers have come to the conclusion that if enough memory is available at each node, and the time between routing table updates is long, then the topology broadcast algorithm (Dijkstra) offers a far superior solution to the traditional Bellman-Ford algorithm. For this reason, this thesis will not consider the use of Bellman-Ford algorithms for use in an interconnected FDDI LAN environment.

In the next chapter, link-state algorithms will be discussed and a new routing algorithm presented.

Chapter 5

Link-State Routing Algorithms

5.0 Introduction

Many of the problems associated with the Bellman-Ford algorithm can be alleviated to a large extent by using a link-state algorithm. As the name suggests, these algorithms require knowledge about the state, or cost, of all links in the network. One of the most efficient link-state algorithms is Dijkstra's shortest path algorithm. A modified form of Dijkstra's algorithm has been used successfully in the ARPANET to replace the original Bellman-Ford routing algorithm.

Far less work has been done on Dijkstra routing algorithms for computer networks than on the Bellman-Ford algorithms. The reason for this is that the algorithm has far fewer "interesting" side effects than the Bellman-Ford. Most research on the Bellman-Ford algorithms has involved reducing the oscillation and looping problems and paid little attention to the actual performance of the algorithm in terms of network throughput and packet delay. Since these problems are less severe in the Dijkstra algorithm, it is possible to ignore them and concentrate on improving the network delay and throughput performance.

The problem of routing table looping is still present during times of link failure in networks that use a Dijkstra algorithm. In order to combat this problem, Garcia has proposed a co-ordinated update scheme in order reduce the looping problem [72]. However, this method does produce a very complex algorithm. McQuillans experience with the ARPANET has shown that the amount of looping produced by the Dijkstra algorithm is so minimal as not to be important [73]. Loop freedom may be essential in a virtual circuit network where the first packet sets up the call for all remaining packets, but less so in a datagram network.

This chapter will give a brief overview of the operation of the Dijkstra algorithm, the ARPANET algorithm and discuss the ISIS routing protocol developed by the ISO OSI committee. A new algorithm, termed DSPS (down stream path splits), which exploits information discarded by the ISIS routing algorithm is presented along with a performance comparison.

5.1 Overview of the Dijkstra Shortest Path Algorithm

Dijkstra's shortest path algorithm forms the basis of many link-state algorithms used in computer communications networks. It is computationally the most efficient shortest path algorithm known today [48]. Its worst case computational requirements are considerably less than those of the Bellman-Ford algorithm.

This algorithm requires that all arc lengths are positive. Fortunately this is the case for computer communications networks. The Dijkstra algorithm finds the shortest paths from a source to all other nodes. In order to do this it requires status information about the entire network topology, hence the name link-state algorithm. The objective is to find the shortest paths from one node to all other nodes. The algorithm does this in a step by step fashion starting at the source and finishing when the furthermost node is reached. By the K^{th} step the shortest paths to the k nodes closest to the source have been calculated. These are then defined to be within the set N. A simple flow chart shows the operation of this algorithm (figure 5.1).

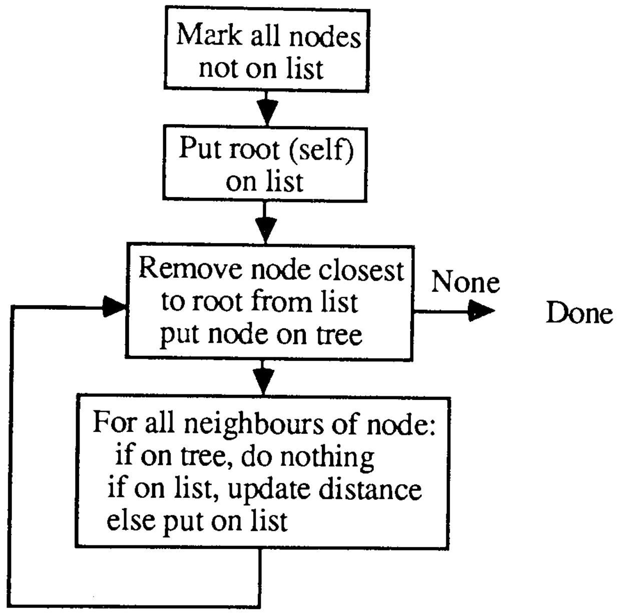

Mark all nodes not on list
Put root (self) on list
Remove node closest to root from list put node on tree
None
Done
For all neighbours of node: if on tree, do nothing if on list, update distance else put on list

Figure 5.1 Simple flow chart for Dijkstra's algorithm.

More formally the algorithm can be described thus.

Let D(v) be the distance (sum of the link weights along a given path) from source A to node v. Let l(i,j) be the cost between node i and node j. There are then two parts to the algorithm: an initialization step and a step to be repeated until the algorithm terminates.

1. Initialization. Set N= {1}. For each node v not in n set D(v) = l(A,v). Infinity (∞) is used to represent nodes not connected to A.

2. At each subsequent step. Find a node w not in N for which D(w) is a minimum, and add w to N. Update D(v) for all nodes remaining that are not in N by computing:

D(v) <- Min[D(v), D(w) + l(w,v)]

Step 2 is repeated until all nodes are in the set N.

An example of the Dijkstra algorithm in operation for the network shown in figure 5.2 is shown in figure 5.3. In this diagram the algorithm is calculating the shortest path trees for node A to all other nodes. The number of iterations required to converge the algorithm is equal to the number of nodes in the network.

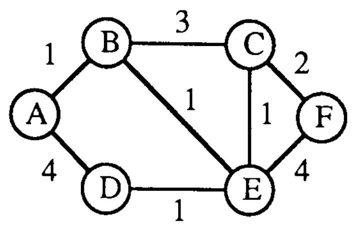


Figure 5.2 Simple six node network with multi-cost links.

Table 5.1 shows the various steps in the calculation. The distances that are circled represent the point at which the shortest path to a particular node has been found.

Step	N	D(B)	D(C)	D(D)	D(E)	D(F)
Initial	(A)	1	∞	4	∞	∞
1	(A,B)	$\textcircled{1}$	4	4	2	∞
2	(A,B,E)	1	3	3	$\textcircled{2}$	6
3	(A,B,C,E)	1	$\textcircled{3}$	3	2	5
4	(A,B,C,D,E)	1	3	$\textcircled{3}$	2	5
5	(A,B,C,D,E,F)	1	3	3	2	$\textcircled{5}$

Table 5.1 The iterations required to build up the shortest paths.

This building up of the shortest path tree from node A can be seen graphically in figure 5.3.

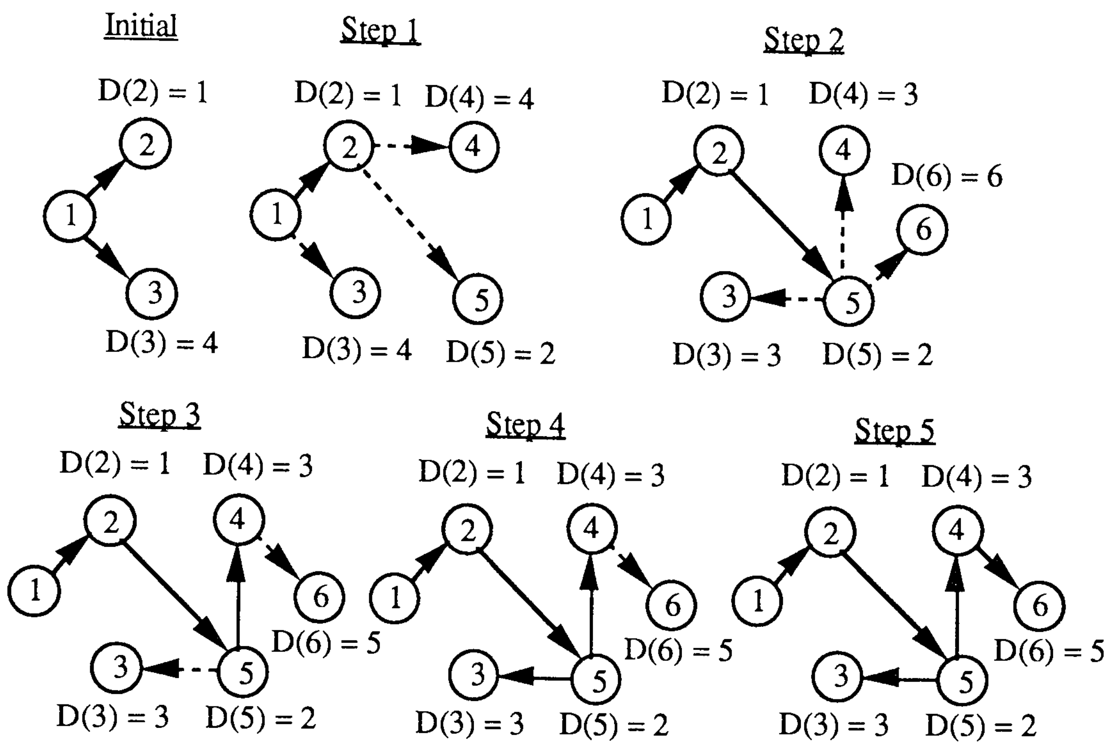


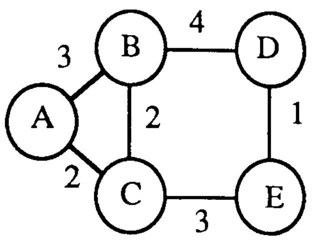
Figure 5.3 Graphical representation of Dijkstra's algorithm for 6 node network.

The Dijkstra algorithm can easily be implemented in a network with a central node having knowledge about the entire network. This node then calculates the shortest path trees for all nodes to every other node. Having done the calculation the central node then informs all network switching nodes of the routes they should use to forward packets. The ARPANET has implemented a modified form of the Dijkstra algorithm for operation in a distributed network environment.

5.2 The ARPANET Routing Algorithm

The original ARPANET routing algorithm was based on the Bellman-Ford routing algorithm. However, after ten years of operation and many different attempts at controlling the problems associated with that algorithm it was eventually scrapped completely in 1979 in favour of a modified form of Dijkstra's algorithm [73].

In order for Dijkstra's algorithm, which is inherently centralized, to operate in a distributed network, each node must have global topology information. Only then can the algorithm correctly calculate the shortest paths. This can be seen in figure 5.4, which shows an arbitrary network, the topology database and the forwarding table for node A.



Global topology database stored at every node.

	A	B	C	D	E
A	0	3	2	0	0
B	3	0	2	4	0
C	2	2	0	0	3
D	0	4	0	0	1
E	0	0	3	1	0

Routing table for node A

Dest.	Dist.	Next
A	0	-
B	3	B
C	2	C
D	6	C
E	5	C

Figure 5.4 Topology database and routing table for ARPANET node.

The discussion of the ARPANET will cover the four routing procedures that make up the routing algorithm.

5.2.1 Route Computation Procedure.

The shortest path first (SPF) routing algorithm of the ARPANET was so-called because of the way in which the shortest path trees are built up at each node. Based on Dijkstra's algorithm each node of the network maintains the entire network topology database in order to calculate the shortest paths. In this way Dijkstra's algorithm can be operated in a distributed environment. It is not truly distributed in the sense that each node benefits from the calculation made by its predecessors. Each node runs the calculation completely independently of all other nodes. In the literature it is sometimes referred to as a centralized/distributed algorithm.

The SPF algorithm is slightly different from the original Dijkstra algorithm in that changes in network link weights can be handled without recalculating the entire shortest path tree. This modification allows changes to the tree to be computed incrementally, without having to recalculate those parts of the tree that do not change. This is an important consideration since the algorithm attempted to adapt to traffic delay and would have required large amounts of processing in order to run the algorithm each time a new delay packet arrived.

5.2.2 Measurement Procedure

The goal of the ARPANET community has always been to send packets on the least delay routes. The new algorithm was no exception. Each node measures the actual delay of each packet flowing over each of its outgoing lines and calculates the average delay every 10 seconds. If this delay is significantly different from the previous delay it is reported to all the other node. The 10 second measurement time is a far cry from the original Bellman-Ford algorithm's 128 ms measurement time. It was found that 128 ms was too short an interval to obtain an accurate estimate of link delay.

In order to measure the delay on a link, a node sends packets to its neighbours with the arrival time stamped on it. When the first bit of the packet is transmitted to the next node, the packet is stamped with its 'sent time'. When the acknowledgement for the packet is received, the arrival time is subtracted from the sent time. To this difference are added the propagation delay of the line and the packet's transmission delay. The result is the total delay for that link.

5.2.3 Information Dissemination Procedure.

The updating procedure for networks that use Dijkstra's algorithm must be very reliable to ensure that all nodes have identical databases. If the databases at each node are different from one another then routing table loops may form. In order to ensure that all nodes have the same database, the cost changes are flooded to all nodes. When a node receives an update, it first checks to see if it has processed that update before. If so, the update is discarded. If not, it is immediately forwarded to all adjacent nodes. In this way the update flows quickly throughout the network.

Appendix B shows the control messages required for the ARPANET routing algorithm following a link failure. Note how few messages are required compared to the Bellman-Ford type algorithms of Appendix B.

5.2.4 Packet Forwarding Procedure

The packet forwarding procedure of the ARPANET algorithm is very straight forward. Packets that arrive at a node are simply sent out on the least delay path towards the destination.

5.3 Performance of the ARPANET Routing Algorithm

Although the new ARPANET algorithm offers many advantages over the original Bellman-Ford algorithm, it has been shown to possess some unsatisfactory characteristics. The algorithm works very well during times of low traffic loading and slowly varying traffic levels. However, with the increase in the amount of traffic and its burstiness the quality of the routes often deteriorates. Under such circumstances the algorithm tends to become unstable and often results in oscillation of routes [77,78]. These problems are related to the attempt by the algorithm to adapt continually to changes in network traffic levels. Algorithms which are this adaptive place place a large processing burden on the nodes and produce excessive control packet overhead. These points were discussed in chapter 2. Therefore, great potential exists for a modified form of the SPF algorithm which does not attempt to adapt to global traffic fluctuations. Such an algorithm does exist.

5.4 Overview of the ISIS Intra-Domain Routing Protocol

The ISO is in the process of standardizing the Intermediate System to Intermediate System Intra-Domain Routing Protocol (ISIS) [75]. ISIS is used for connectionless network layer services and is therefore relevant to this study. An intermediate system (IS) is another name for a router and an end system (ES), in OSI-speak, is a host on a LAN. The ISIS routing algorithm is designed explicitly for routing traffic in interconnected LANs.

The ISIS routing algorithm has built on the experience of the ARPANET research network. It has taken the best components from that work and combined them into a comprehensive routing strategy. The standards bodies are often criticized for standardizing things which have become obsolete or are of little use. The ISIS algorithm does not fall into this category and is being enthusiastically supported by all the major networking manufacturers [74].

The intra-domain ISIS routing protocol is intended to support large routing domains consisting of combinations of many types of subnetworks. This includes point to point links, multi point links, X.25 subnetworks and broadcast subnetworks. Some of the goals of the routing algorithm as detailed in the standards document are :

> Adaptability. It adapts to topological changes within the routing domain, but not to global traffic changes, except potentially as indicated by local queue lengths. It splits traffic on multiple equivalent paths.
>
> Efficiency. It is both processing and memory efficient. It does not create excessive routing traffic overhead.
>
> Stability. It stabilizes in finite time to good routes, provided no continuous topological changes or continuous database corruptions occur.

ISIS is best examined by looking at the four routing procedures that make up the routing algorithm. The names in brackets are those used in the ISIS document.

5.4.1 Route Computation Procedure (Decision Process)

The route computation procedure is a standard SPF algorithm as used in the ARPANET. However it is different in two ways. First of all the algorithm in the standard is defined for complete recalculation only and is not defined for incremental changes to the shortest path tree. This is reasonable since the number of times that the algorithm has to run in comparison to the ARPANET is very low.

The second significant point about the algorithm is that it allows the use of equal cost paths between a source and destination. The ARPANET algorithm only permits a single cost path to be calculated between a source/destination pair.

5.4.2 Measurement Procedure

Link costs are not actually measured as they are in the ARPANET but are defined by system management. Therefore the measurement procedure is effectively moot. The ISIS protocol allows the use of four different metrics that can be used for routing traffic. These are link speed, monetary cost, security and delay. The link speed metric is the default metric which all ISIS routers must support, the remaining three metrics are optional.

5.4.3 Information Dissemination Procedure (Update Process)

The information distribution procedure is exactly like that of the ARPANET. It uses flooding to ensure that all nodes have the same databases throughout the network. The time between information updates is a variable which is set by the network designer. A recommended value in the standard is specified as 15 minutes. Note that the purpose of forwarding updates is to ensure that no corruption of any of the intermediate systems databases has occurred, it's function is not to adapt to global traffic loadings.

5.4.4 Packet Forwarding Procedure (Forwarding Process)

Packets are forwarded along the shortest path trees. In the case of more than one equal cost path, the local queue lengths of the outgoing links are examined and the one with the shortest length is used as the link to forward the packet on. In this way, the algorithm adapts to traffic fluctuations on a local basis only.

The ISIS routing algorithm is a very well thought out algorithm. It encompasses most of the features required for interconnecting FDDI LANs as discussed in chapter 3. It is an SPF algorithm which routes packets based on local information only and uses global topology information for long term topology changes. It is independent of topology and can cope with the bursty traffic sources that are often found in LANs. Although it copes with high traffic loading better than the ARPANET, this is an area where the algorithm could benefit from some improvement.

5.4.5 Operation of ISIS

In order for the ISIS routing algorithm to operate two databases must be maintained. These are PATHS and TENT [75].

PATHS is an acyclic directed graph of shortest paths from the system performing the calculation. It is stored as a set of triples of the form <N,d(N),{Adj(N)}> where :

N is a system identifier

d(N) is N's distance from S (ie the total metric value from N to S)

{Adj(N)} is a <u>set</u> of valid adjacencies that S may use for forwarding to N.

When a system is placed on PATHS, the path(s) designated by its position in the graph is guaranteed to be a shortest path.

TENT - This is a list of triples of the form <N,d(N),{Adj(N)}>, where N, d(N) and {Adj(N)} are as defined above for PATHS.

TENT can be thought of a as a tentative placement of a system in PATHS. In other words, the triple <N,x,{A}> in TENT means that if N were placed in PATHS, d(N) would be x, but N cannot be placed on PATHS until it is guaranteed that no path shorter than x exists.

<u>5.4.6 Algorithm Steps</u>

The basic algorithm, which builds PATHS from scratch, starts out by putting the system doing the computation on PATHS (no shorter path to SELF can possibly exist). TENT is then pre-loaded with the systems adjacencies. Note that a system is not placed in PATHS unless no shorter path to that system exists. When a system is placed in PATHS, the path to each neighbour M of N through N is examined as the path to N plus the link from N to M. If <M,*,*> is in PATHS this new path must be longer and thus ignored.

If <M,*,*> is in TENT and the new path is shorter the old entry is removed from TENT and the new path is placed in TENT. If the new path is the same length as the one in TENT, N is added as a potential parent in the triple tent. If M is not in TENT, then the path is added to TENT.

Next the algorithm finds the triple <N,x,{Adj(N)}> in TENT with minimal x.

N is placed in PATHS. It is known that no path to N can be shorter than x at this point because all paths through systems already in PATHS have already been considered and paths through systems in TENT will have to be greater than x because x is minimal in TENT

When TENT is empty, PATHS is complete. The steps used in the ISIS calculation are shown below.

```
Step 0: Initialize TENT and PATHS to 0
        a. add (self,0,0) to PATHS
        b. Pre-load TENT with the local adjacency database
        c. got to step 2

Step 1: With reference to the system just placed in PATHS
        a. compute dist(P,N) = d(P) + metric(P,N) for each neighbour
                N of the system P.
        b. If dist(P,N) > MaxPathMetric, do nothing
        c. If <N,d(N),{Adj(N)}> is in PATHS, do nothing
        d. If a triple <N,x,{Adj(N)}> is in TENT, then
                i. if x=dist(P,N) then Adj(N) <= {Adj(N)} U Adj(P)
                ii. if x < dist(P,N), do nothing
                iii. if x > dist(P,N), remove <N,x,{Adj(N)}> from TENT
                        and add <N,dist(P,N),{Adj(P)}>
        e. if no triple is in TENT, then add <N,dist(P,N),{P}> to TENT

Step 2: If TENT is empty, stop, else
        a. find the element <P,x,{Adj(P)}>, with minimal x.
        b Remove <P,tentlength,{Adj(P)}> from TENT
        c. Add <P,d(P),{Adj(P)}> to PATHS
        d. Go to step 1
```

For node A of the network shown in figure 5.5 the algorithm would generate the following steps.

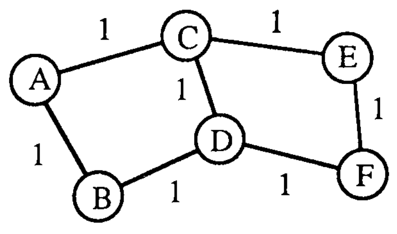


Figure 5.5 Simple six node network with equal cost links.

STEP 0:	TENT and PATHS are set to 0
STEP 0a:	PATHS contains (A,0,0)
STEP 0b:	TENT contains (B,1,B) and (C,1,C)

Step 2a:	Minimum of TENT is (B,1,B)
Step 2b:	(B,1,B) is removed. TENT contains (C,1,C)
Step 2c:	PATHS contains (A,0,0) and (B,1,B)
Step 1a:	dist(B,D) = d(B) + metric(B,D) = 3
Step 1e:	TENT (C,1,C) and (D,2,B)
Step 2a:	TENT minimum is (C,1,C)
Step 2b:	TENT becomes (D,2,B)
Step 2c:	PATHS becomes (A,0,0), (B,1,B) and (C,1,C)
Step 1a:	dist(C,D) = d(C) + metric(C,D) = 2
Step 1d:	TENT becomes (D,2,B,C)
Step 1a:	dist(C,E) = d(C) + metric(C,E) = 2
Step 1e:	TENT becomes (D,2,B,C) and (E,2,C)
Step 2a:	TENT minimum is (D,2,B,C)
Step 2b:	TENT contains (E,2,C)
Step 2c:	PATHS contains (A,0,0), (B,1,B), (C,1,C) and (D,2,B,C)
Step 1a:	dist(D,F) = d(D) + metric(D,F) = 3
Step 1e:	TENT contains (E,2,C) and (F,3,B,C)
Step 2a:	TENT minimum is (E,2,C)
Step 2b:	TENT contains (F,3,B,C)
Step 2c:	PATHS contains (A,0,0), (B,1,B), (C,1,C), (D,2,B,C) and (E,2,C)
Step 1a:	dist(E,F) = d(E) + metric(E,F) = 3
Step 1d(i):	TENT contains (F,3,B,C,C)
Step 2a:	TENT minimum is (F,3,B,C,C)
Step 2b:	TENT is empty.
Step 2c:	PATHS contains (A,0,0), (B,1,B) (C,1,C), (D,2,B,C), (E,2,C) and (F,3,B,C,C)
Step 1a:	The algorithm is effectively finished.

At the end of the computation the basic ISIS algorithm, as detailed in the standards document, will contain the following set of triples in PATHS for routing packets from source A (figure 5.6).

PATHS DATABASE		
Dest.	Dist.	Adj.
A	0	0
B	1	B
C	1	C
D	2	B,C
E	2	C
F	3	B,C,C

Figure 5.6 PATHS database produced for node A by ISIS standard.

Therefore if traffic arrived at node A destined for, say, node D (figure 5.5) two different paths could be used to get there, one via adjacency B and one via adjacency C. Both adjacencies offer equidistant paths. What is extremely interesting about the basic algorithm is the fact that to get to node F the adjacency C has appeared twice in the calculation. Obviously in a real implementation of the ISIS algorithm there would be a checking mechanism in step 1a in order to stop duplicate adjacencies being placed in TENT. However, in the basic algorithm there is no such check and the duplicate adjacency appears finally in the PATH database.

An actual ISIS implementation would have at some stage discarded this duplicate, since there appears to be little point in keeping it. This would most likely be done during step 1. The resulting ISIS PATHS database would then contain the following (figure 5.7):

PATHS DATABASE		
Dest.	Dist.	Adj.
A	0	0
B	1	B
C	1	C
D	2	B,C
E	2	C
F	3	B,C

Figure 5.7 PATHS database after removing duplicate adjacencies.

However, this duplicate represents some extremely useful information which should not be discarded so easily. A duplicate adjacency in the PATHS database means that after having gone through that adjacency another equal cost path is available to get to a destination. In other words a split occurs down tree (or down stream) from the node doing the calculation. So although the adjacencies C and B offer equal cost paths from A to F, routing traffic via C is probably a better path to take since having gone through node C, a choice of routes is again made available to the packet. Node C has the choice of sending the packet to F via node E or node D. If the packet was initially sent via node B, no such choice would be permitted further down the tree.

5.5 The Down Stream Path Splits Routing Algorithm

In this thesis, the duplicate adjacencies will be referred to as down stream path splits (DSPS). A new algorithm which exploits these down stream path splits is discussed in this section. The DSPS algorithm requires modification to the route computation and packet forwarding procedures of the ISIS algorithm, it uses the same distance measurement and information distribution procedure that the ISIS algorithm uses.

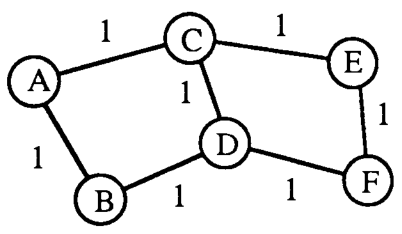
5.5.1. Route Computation Procedure

In order to calculate the number of DSPS for each adjacency the following modification can be made to step 1d of the ISIS route computation procedure

```
i if x=dist(P,N) and Adj(P) E {Adj(N)} then increment
        DSPS.COUNT(N,Adj(P))
ii. if x=dist(P,N) then Adj(N) <= {Adj(N)} U Adj(P)
iii. if x < dist(P,N), do nothing
iv. if x > dist(P,N), remove <N,x,{Adj(N)}> from TENT
        and add <N,dist(P,N),{Adj(P)}>
```

The DSPS of a particular adjacency for a given destination can be stored next to the adjacency information in the routing table. This can be seen in figure 5.8.

The pseudo-code for the DSPS route computation procedure is shown in appendix C.



Routing table stored at node A

Destination	Distance	Adjacency	DSPS
A	0	-	-
B	1	B	0
C	1	C	0
D	2	B,C	0,0
E	2	C	0
F	3	B,C	0,1

Figure 5.8 Routing table with DSPS for simple six node network.

5.5.2 Packet Forwarding Procedure

The forwarding of packets towards nodes which offer down stream path splits results in the following advantages:

1. Packets can adapt more easily to traffic fluctuations.
2. The chances of the packet being dropped by the network are much reduced.

The effects of dropping a packet at a node due to a full buffer may cause the transport layer to time out and retransmit the packet. This not only increases the average network packet delay but also wastes resources in retransmitting a packet.

Although sending packets towards adjacencies that offer DSPS is a good idea it should not be used with the complete exclusion of all other equal cost paths. To do so would cause unnecessary congestion. It is therefore necessary to modify the ISIS forwarding algorithm to allow the DSPS of an adjacency to be taken into account whilst not neglecting other equal cost paths.

If two equal cost paths are available to a node, the ISIS forwarding algorithm chooses between the two based on the outgoing queue lengths. Whichever offers the shortest queue is then used for forwarding the packet. In order to bias the packets towards nodes which have DSPS it is necessary to modify the basic ISIS forwarding algorithm. The pseudo-code for the DSPS forwarding procedure is shown in appendix C.

If an adjacent node offers a single DSPS, then the forwarding algorithm considers the queue to that adjacency as having one less packet in its outgoing queue than if it offered no DSPS. Likewise, if two DSPS were offered by an adjacency, two less packets would be considered in the queue. In this way a sort of credit system is in operation which allows equal cost adjacencies to be considered even if they do not provide the shortest outgoing queue length. It allows the forwarding algorithm to send packets towards nodes which give the packet maximum flexibility. The following simple equation shows how queue lengths and down stream path splits are compared.

[Queue.length(Adj_i) - DSPS(Adj_i)] cf. [Queue.length(Adj_j) - DSPS(Adj_j)] ...
cf. [Queue.length(Adj_N) - DSPS(Adj($_N$)]
where N is MaxPathSplits

If this equation results in equal values for each adjacency then the router chooses the first adjacency listed in the forwarding table. This is a fairly minor point, and the algorithm could easily be modified to break ties by selecting nodes based on which adjacency offers more DSPS's, in a round-robin fashion or perhaps even randomly.

The following discussion of the forwarding procedure is with respect to the network shown in figure 5.8. It is assumed throughout the discussion that the packet under consideration begins at node A and is destined for node F.

When the output queues to nodes B and C are 5 and 6 respectively the DSPS algorithm still chooses C as the next node. This is due to the one down stream path split offered by that node, effectively considering the queue lengths as equal. In this situation the forwarding algorithm would choose node C since it offers a DSPS.

If the length of any of the outgoing queues is equal to the maximum buffer size allowable then the DSPS algorithm must select the next best path. This effectively suspends the credit system of DSPS when queues become equal to the maximum buffer length. For example, if the maximum buffer size is limited to 10 packets and queue lengths for nodes B and C are 9 and 10 packets respectively, then node B must be selected as the next node. If node C is selected (using DSPS credits) then the packet will be instantly dropped by the buffer, since it is at maximum length. This would drop the packet needlessly since node B offers a queue length less than the maximum. This example is shown in table 5.2. Therefore, the forwarding algorithm must take into account the maximum buffer size of the outgoing queues.

Table 5.2 lists the next nodes that the ISIS and DSPS algorithms would choose with various queue lengths for nodes B and C. The next node selection assumes that the packet starts at node A and is destined for node F.

Queue length for node B	Queue length for node C	DSPS for node B	DSPS for node C	Next node (ISIS)	Next node (DSPS)
2	4	0	1	B	B
4	3	0	1	C	C
5	5	0	1	B	C
5	6	0	1	B	C
9	10	0	1	B	B

Table 5.2 Next node chosen by ISIS and DSPS for different queue lengths.

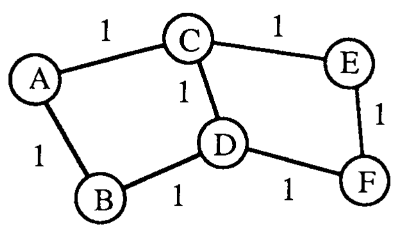
This completes the introduction to the DSPS algorithm. The performance of the DSPS algorithm relative to that of ISIS is the subject of the next section.

5.6 Performance of ISIS and DSPS Routing Algorithms

In order to ascertain the performance of the new DSPS routing algorithm over the ISIS it is necessary to resort to computer simulation since mathematical evaluation would be far too complex. Mathematics can only be used on the very simplest networks, even then the complexity is somewhat prohibitive. The DSPS algorithm and the ISIS algorithm were simulated using the SIMSCRIPT II.5 simulation language. The complete simulation model will be described fully in chapter 8.

One of the best parameters for comparing routing algorithms is network power. Power is defined as throughput divided by average network delay [78]. Since throughput and delay are often in contention, power is a good measure of performance. Power is plotted against the arrival rate at each of the nodes. If for example the arrival rate is shown as 20 packets per second, then each node in the entire network will be generating 20 packets every second.

The first network to be simulated was the six node network which has appeared throughout this chapter. In this network there are a total of four down stream path splits. These are shown in figure 5.9 below. The simulated network assumes 64 kbps links throughout, however, the results are just as valid if 2.048 Mbps Megastream links, or indeed any link speed was used. By defining all links as equal, the weight assigned to each link is effectively one. This will be true of all the networks simulated in this chapter.



Source	Dest.	Next	DSPS
Node A	F	C,B	0,1
Node B	E	A,D	0,1
Node E	B	C,F	1,0
Node F	A	D,E	1,0

Figure 5.9 Nodes that offer DSPS in the six node network.

The power generated by the ISIS and the DSPS algorithm are shown in figure 5.10.

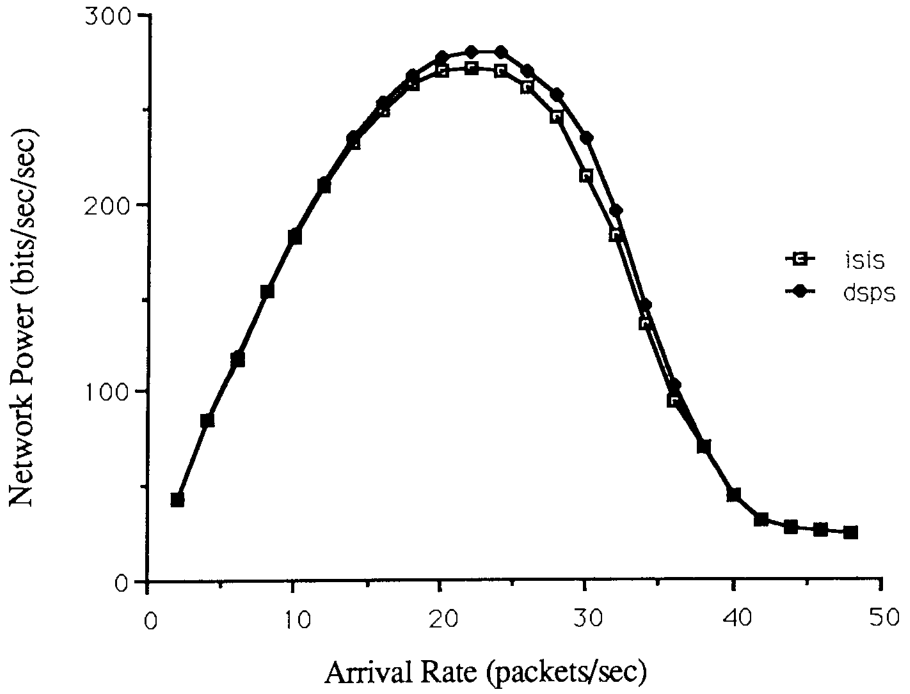


Figure 5.10 Power against arrival rate for ISIS and DSPS (6 nodes, 7 links)

The performance improvement of the DSPS algorithm over the standard ISIS algorithm may not be fully appreciated from the graph in figure 5.10! In order to illustrate the gain made by DSPS, figure 5.11 shows a graph of the percentage improvement of the DSPS algorithm over the ISIS plotted against arrival rate.

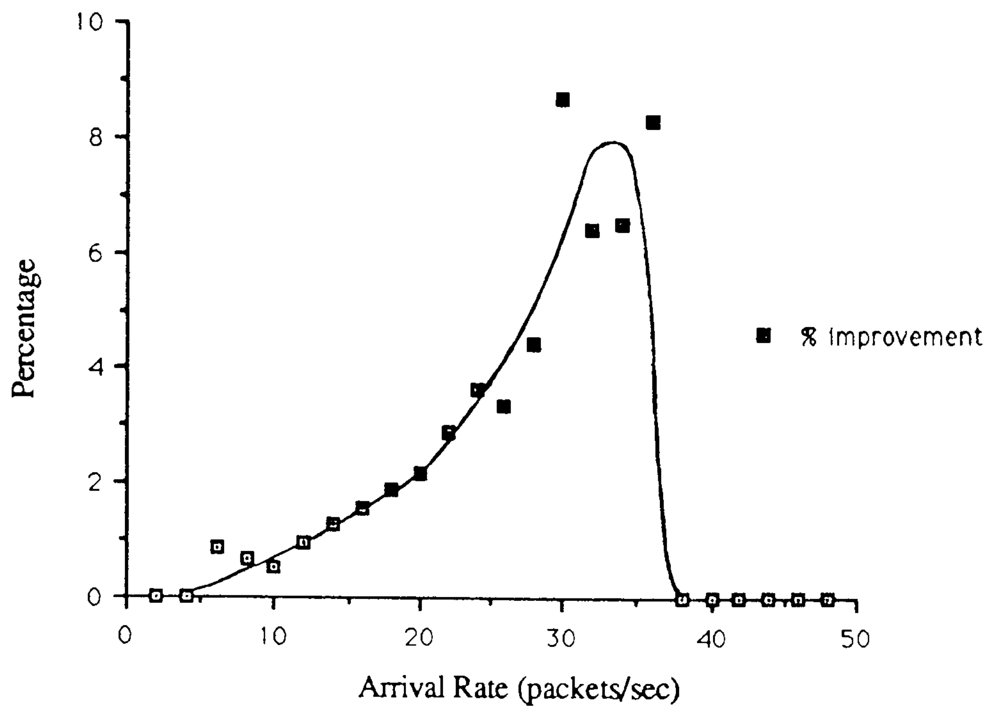


Figure 5.11 Percentage improvement of DSPS over ISIS (6 nodes, 7 links).

The next three power graphs are for 10 and 15 nodes respectively. In these graphs the total number of different DSPS's in all of the nodes' routing tables are shown. It is very important to have different values of down stream path splits in a network. For example, if a node has two adjacencies X and Y with down stream path splits of 3 and 3 respectively then the DSPS values effectively cancel each other out and the choice between the two reverts to examining queue lengths only. A network may offer many down stream path splits, but if they all cancel each other out then the performance of the network degrades to that produced by the ISIS algorithm.

Both of the remaining networks have been built up from a spanning tree (number of nodes - 1 links) with links added at random (recall from chapter 3 that most datagram networks grow randomly). Tables 5.3 and 5.4 shows the number of different DSPS's that were produced for a 10 and 15 node randomly generated network respectively.

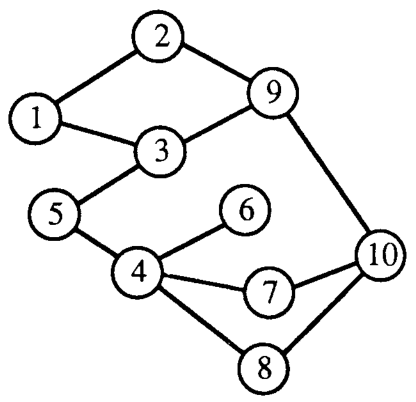
Number of different DSPS		Number of links									
		10	11	12	13	14	15	16	17	18	19
Random number seed	1	0	0	5	2	0	4	5	2	2	2
	2	0	2	2	5	3	5	3	2	2	2
	3	0	5	2	0	0	1	2	2	3	4
	4	0	5	10	3	10	9	11	9	7	7
	5	0	0	0	2	3	5	6	3	2	2
	6	0	0	0	0	1	3	4	4	3	4
	7	0	0	0	2	5	6	6	4	4	5
	8	0	4	2	2	2	5	4	6	6	4
	9	0	0	5	0	0	2	2	3	0	0
	10	0	0	0	0	3	6	6	9	9	7

Table 5.3 Number of different DSPS for randomly generated 10 node network.

Number of different DSPS		Number of links														
		15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Random number seed	1	0	0	0	0	4	7	8	9	21	12	6	12	10	11	12
	2	0	0	0	0	0	0	3	0	11	15	17	16	18	23	21
	3	0	0	5	14	18	18	20	28	27	34	26	22	20	18	14
	4	0	0	9	7	0	0	12	17	17	14	9	13	14	14	12
	5	0	0	0	0	3	2	8	3	2	5	13	12	13	12	14
	6	0	0	1	1	5	15	12	21	13	12	13	10	10	12	13
	7	0	0	10	6	6	6	10	7	9	9	14	12	13	16	17
	8	0	0	0	0	0	0	2	6	20	25	14	12	10	14	12
	9	0	4	6	6	9	8	9	25	20	19	18	21	22	23	22
	10	0	8	9	4	4	4	8	13	12	18	16	22	22	27	25

Table 5.4 Number of different DSPS for randomly generated 15 node network.

The simulation results for the ten node network shown in figure 5.12 are shown in figure 5.13. This network has 12 links and 10 different DSPS (random number seed 4).



Source	Dest.	Dist.	Next	DSPS
Node 1	7 8	4 4	2,3 2,3	0,1 0,1
Node 2	4 6	4 5	1,9 1,9	0,2 0,2
Node 4	2	4	5,7,8	1,0,0
Node 5	10	3	3,4	0,1
Node 7	1	4	4,10	0,1
Node 8	1	4	4,10	0,1
Node 9	4 6	3 4	3,10 3,10	0,1 0,1

Figure 5.12 Nodes that offer different DSPS in the 10 node network.

The power against arrival rate graph for the above 10 node network is shown in figure 5.13.

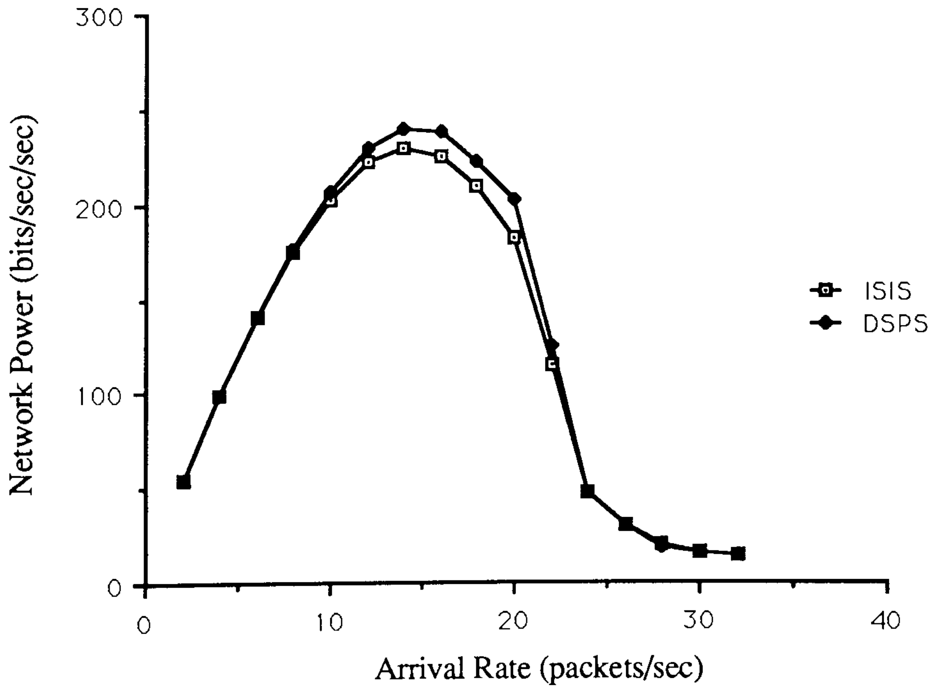


Figure 5.13 Power against arrival rate for ISIS and DSPS (10 nodes, 12 links)

The percentage improvement graph for the 10 node network is shown in figure 5.14.

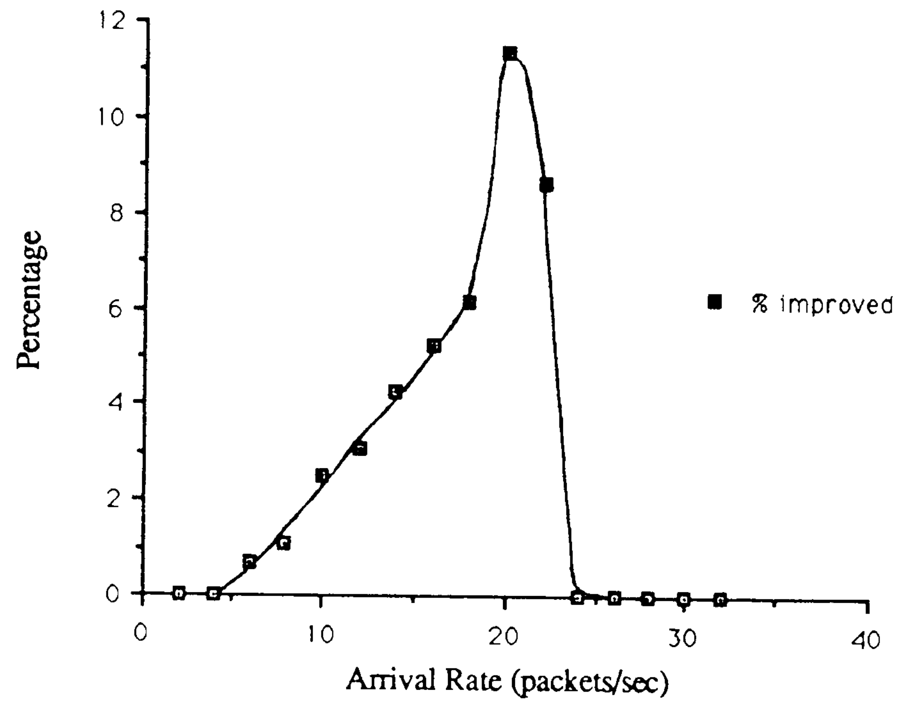


Figure 5.14 Percentage improvement of DSPS over ISIS (10 nodes, 12 links)

The power against arrival rate graph for a 15 node network is shown in figure 5.15.

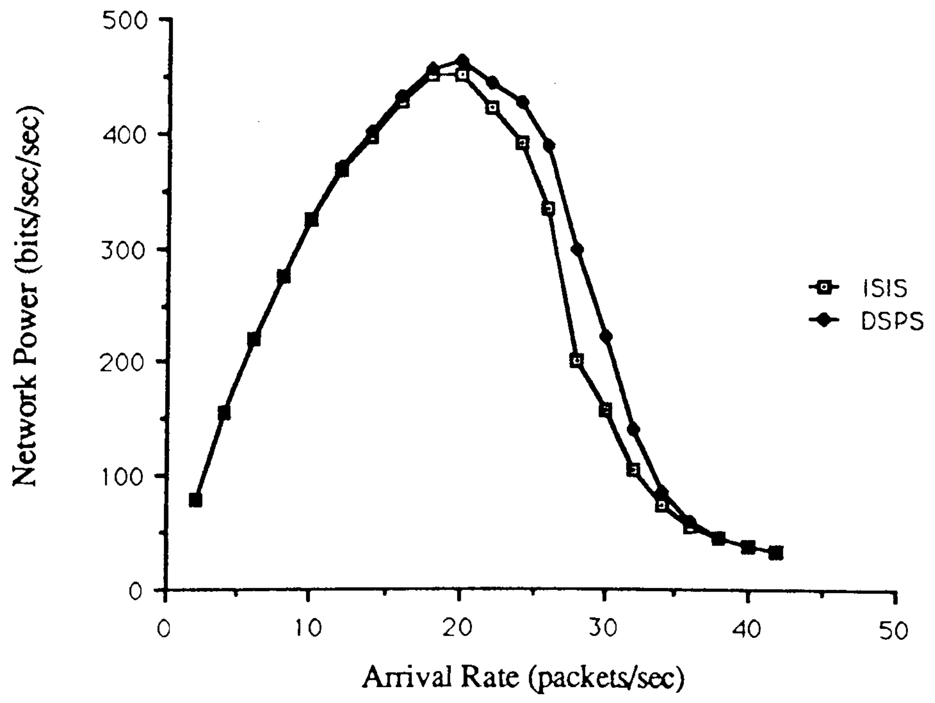


Figure 5.15 Power against arrival rate for ISIS and DSPS (15 nodes, 22 links)

The percentage improvement graph for the 15 node network is shown in figure 5.16.

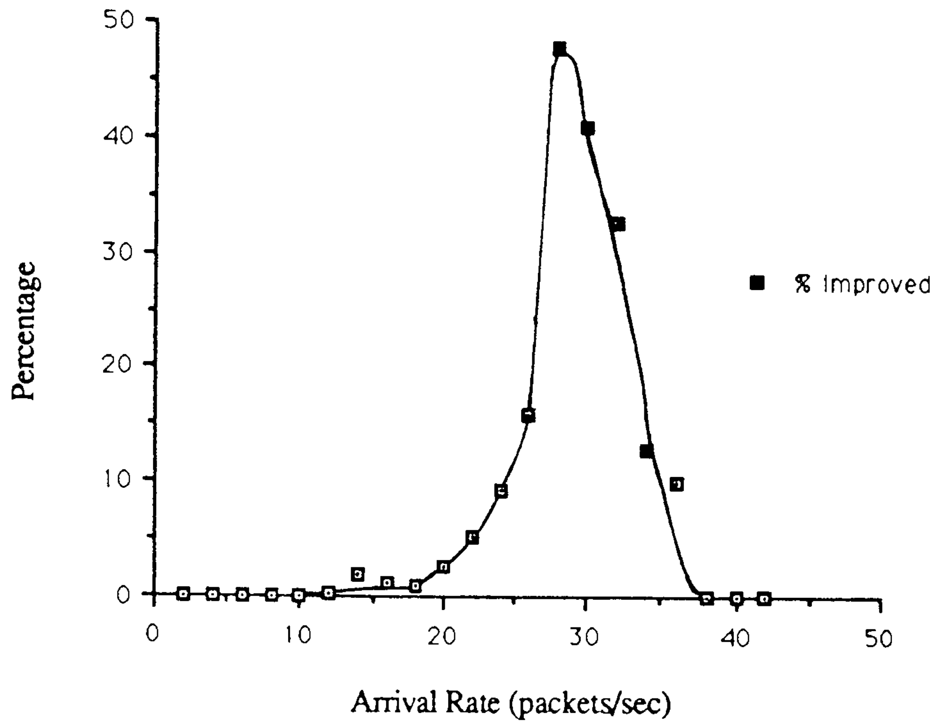


Figure 5.16 Percentage improvement of DSPS over ISIS (15 nodes, 22 links)

Further simulation results are shown in appendix D.

5.7 Discussion

There are four different regions of the power/arrival rate graphs which can be identified. These regions are shown in figure 5.17.

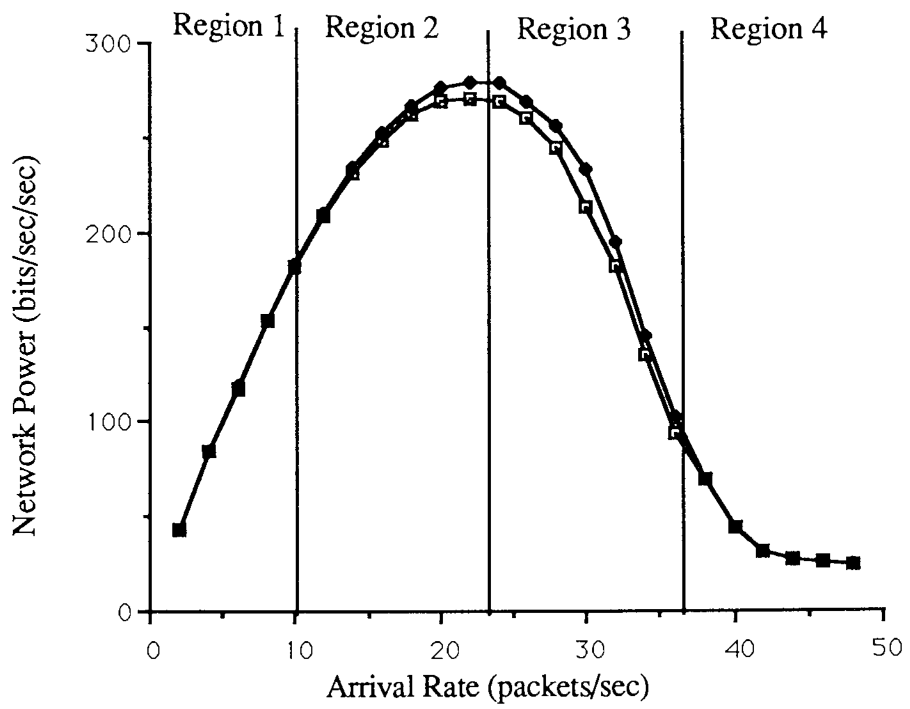


Figure 5.17 The four regions of power against arrival rate graphs.

The first region represents low traffic loading on the network. In this region the average queue lengths are very low and the effect of sending a packet toward a node that offers DSPS and one that doesn't is negligible. There is almost no performance gain made by the DSPS algorithm over the ISIS in this region. As an aside, simulation results have shown that even the ARPANET which operates most effectively in this area, provides no significant gain over DSPS or ISIS [96].

In the second region queue lengths at each node are increasing and the effects of the DSPS algorithm can be seen. The maximum value that the power graph attains in this region represents the optimal arrival rate for that network. In this region, almost no packets are dropped due to insufficient buffer space.

The third region of the graph represents the reduction in power that an increasing arrival rate has on the network. More packets are being dropped due to buffer overflow. The DSPS algorithm provides the maximum amount of performance gain in this region. Region three represents the upper bound for packet arrival rate that a network should be operated within.

The fourth and final region is the convergence of the two plots. This occurs because the arrival rate at each node is so high that queues are becoming saturated. In this case, (recall from the example given in section 5.5.2) the DSPS algorithm is effectively suspended and the shortest outgoing queue is selected. In this region there is no difference between the two algorithms. It is most unlikely that a network would be operated continuously in this region since the performance degradation is so very high due to the large number of transport layer retransmissions.

A complete discussion of power graphs will be included in chapter 8.

The performance of the DSPS algorithm is directly related to the number of different down stream path splits that exist in the network. This depends on two factors, link weighting and topology.

If link weights are all set to the same value, then the only factor to be concerned with is the topology. The effects of topology on the number of different down stream path splits can be quite drastic. This can be seen by examining table 5.3 for a ten node network. Using a random number seed of four and 12 links results in 10 different DSPS's. If a single link is added between nodes 1 and 5 the number of different DSPS in the network drops from 10 down to 3. If, in addition to the added link between 1 and 5, a link is also added between nodes 2 and 8 the number of DSPS increases back to 10.

It must be remembered that all of the topologies generated in this study are completely random. There has been no attempt to define a topology with a large number of different down stream path splits. This is in keeping with chapter 3, which states that most datagram networks grow randomly. However, of the randomly generated networks, those that offered the largest number of DSPS were chosen for the simulation runs.

There are certain types of network topology where usage of the DSPS algorithm will not produce any increase in network performance. In addition to a network that offers no different DSPS's there are three types of network where absolutely no performance gains can be made by implementing the DSPS algorithm. In all of these topologies, no down stream path splits exist.

-The network cannot be a spanning tree. More links than the number of nodes minus one must exist in the topology in order for different DSPS's to be present.

-The network cannot be fully connected. If every node in a network has links to every other node in the network then it is described as being fully connected. In such a topology, there are no equal cost paths, hence no possibility of DSPS.

-Six node minimum. In order for down stream paths to exist, there must be at least six nodes in the network. This has been found to be the minimum number that will generate different down stream path splits.

It is very unlikely that in a corporate network a tree topology would exist, since a single link would disconnect the network. Invariably other links are included for reliability. On the other extreme, fully connected nodes are very robust, but they are prohibitively expensive for anything other than the smallest of networks. Most practical networks are in between these two extremes. The most constraining topology problem is the requirement that more than six nodes are present in the network. However, this is not a severe problem since many corporate networks have tens and even hundred of nodes.

In all the simulation runs, with various topologies, the DSPS algorithm provided either a better performance than the ISIS or an equal performance. At no time was the DSPS algorithm outperformed by the ISIS.

It is simply not possible to exhaustively simulate every topology with X number of links and determine how often the DSPS algorithm outperforms the ISIS algorithm. What the simulation does provide, however, is the ability to make the following generalization:

If different down stream path splits exist in a network then the performance provided by the DSPS algorithm will be greater than that produced by the ISIS algorithm.

Although all the networks in this chapter had equal link weights, this does not mean that the algorithm cannot work in an environment where link costs are different. It simply means that the chances of having down stream path splits in the network are somewhat reduced.

One of the most appealing aspects of running the DSPS algorithm in a router is that it does not require other nodes in the network to understand the concept of down stream path splits. In addition, it sends and receives the exact same link-state information that the standard ISIS uses. What this means, of course, is that DSPS routers can work quite happily alongside ISIS routers in the same network. However, those network where DSPS routers are installed can attain a higher performance than their ISIS counterparts!

The ISIS standard specifies that any SPF algorithm can be used so long as the router still conforms to the PICs (protocol information characteristics) proforma in the document. The DSPS algorithm unquestionably corresponds to this standard.

A summary of the advantages that the down stream path splits algorithm offers are listed below:

- Very simple.
- Uses information discarded by ISIS.
- Improves network performance.
- Only needs small modifications to the route computation and forwarding procedures.
- Can be used with any router running ISIS.
- No additional control packet overhead.

The DSPS algorithm operates best in the following environment.

- Topology is neither fully connected, nor a spanning tree.
- All links have the same weighting.
- More than 5 nodes in the network.

The disadvantages of DSPS are:

- Requires up to one third more storage space.
- Tiny amount of additional processing.

There is little doubt that the if different down stream path splits exist in a network then the performance of that network is improved by implementing the new algorithm. In general, the greater the number of different DSPS's the greater the performance improvement. This performance gain has been shown to be as high as 50% in some of the simulated networks.

5.8 Improvements and Other Uses for DSPS

All of the simulated networks assume that for each DSPS offered by an adjacency, a single credit is given for the queue length. Although not simulated here, it may prove worthwhile to increase the size of the DSPS credit from one up to some value depending on the size of the network. For example, an adjacency with two DSPS could be given four credits to be used against the queue length. This however, is very much dependent on the buffer sizes and packet lengths.

Another use of the down stream path splits concept which does not involve modification to the ISIS forwarding procedure or require permanent storage of the DSPS values is in the selection of equal cost paths.

The ISIS algorithm allows a maximum of 32 equal cost paths for each destination to be stored in the forwarding database [75]. The default value for the variable `MaximumPathSplits` is two. If the SPF algorithm is run and more equal cost routes appear in the calculation than `MaximumPathSplits`, ISIS must prune the selection by removing these excess adjacencies. The choice between adjacencies is based on the identification number of the particular adjacency. Whichever ID is lowest, will be retained in the forwarding database.

This is a very arbitrary selection process, which does not take into account the quality of the routes provided. A far better method of pruning would be to store those adjacencies that offer the greatest number of down stream path splits. This will ensure that the best of the equal cost routes is maintained in the forwarding database. Implementing this modification to a standard ISIS router would be an extremely simple operation.

5.9 Conclusion

Dijkstra's shortest path algorithm represents one of the most efficient shortest path algorithms known today. This algorithm in its distributed form has been successfully implemented in the ARPANET. The ARPANET's new SPF algorithm is far better than the original Bellman-Ford algorithm which suffered badly during times of link and node failure. The new ARPANET algorithm has been found to operate very well under lightly loaded traffic conditions but less well under heavy traffic loading. It is also prone to traffic oscillation during periods of high loading. These problems with the ARPANET are attributed to the constant updating required by the algorithm in order to adapt to traffic fluctuations throughout the network.

The OSI ISIS intra-domain routing protocol is based on the ARPANET's SPF algorithm. ISIS alleviates many of the flaws associated with the SPF algorithm by not requiring nodes to adapt to traffic on a global basis. The only traffic adaptation that does occur is made locally based on outgoing queue lengths. Most computer network manufacturers' in their move towards open systems will implement the ISIS protocol.

The down stream path split algorithm is a very simple extension of the ISIS algorithm that utilizes information which is completely disregarded by that algorithm. This algorithm requires additional storage for the DSPS values and introduces a minimal amount of additional processing. However, the algorithm provides an increase in network power which can be as high as 50% over that produced by the ISIS algorithm. A manufacturer of ISIS routers can easily implement the DSPS algorithm and provide it as one of the options available to the end user.

The next chapter presents several new algorithms which allow alternative paths to be used during times of network congestion.

Chapter 6

Alternative Path Routing Algorithms

6.0 Introduction

The ISIS algorithm and the ARPANET's SPF algorithm are fundamentally single path routing algorithms. This means that a single cost path is available for sending packets from a source to destination. During times of heavy traffic loading on the network, these single paths can become heavily congested. It is desirable to examine algorithms that can increase the performance at these levels of traffic.

Alternative path algorithms allow the use of higher cost paths when the main shortest path is blocked or congested. These algorithms are more complex and require larger nodal memory stores than the basic shortest path algorithms, but have the potential to produce sizable performance increases.

This chapter presents three new alternative path algorithms which use the ISIS intra-domain routing protocol as a basis. The algorithms are completely independent of the DSPS algorithm discussed in the previous chapter. The new algorithms use the same link state information that a standard ISIS router would use. In addition, they require modification only to the packet forwarding and route computation procedures of the ISIS algorithm.

6.1 Overview of Alternative Path Algorithms

The maximum flow between two points in a network represents the dynamic range of traffic that the network is able to handle. According to the max-flow min-cut theorem, the maximum flow between any two arbitrary nodes in a network is equal to the capacity of the minimum cut separating those two nodes. This well known theorem can be used to highlight the differences between a shortest path algorithm and an alternative path algorithm [76,77].

In the network below (figure 6.1), the arc weights represent the carrying capacity of the links in units of traffic. Therefore the higher the value of the weight, the greater the capacity of the link. Using the max flow min cut theorem, the maximum flow that can be attained from node A to node G can be found by cutting links BD and CE. The sum of these two minimum cuts gives a maximum flow of four units..

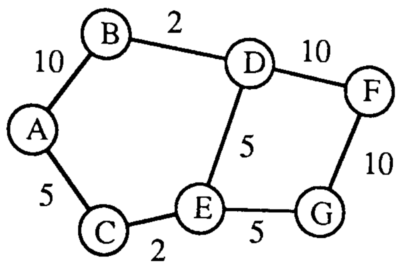


Figure 6.1 Simple network with maximum flow as link cost.

Therefore it is theoretically possible for the network to handle an arrival rate of four units of traffic at node A destined for node G. Two of these units can be sent via nodes ABDFG and the other two units can be sent via nodes ACEG (figure 6.2).

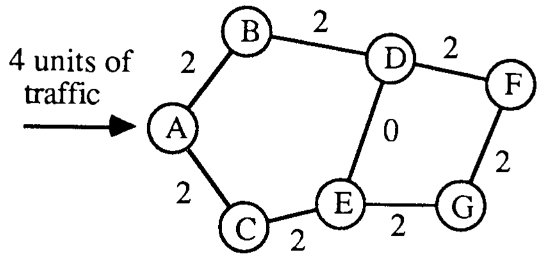


Figure 6.2 Simple network with distribution of 4 units of traffic.

In order for an algorithm to calculate the shortest path trees in a network, the carrying capacities are inverted and normalized against the highest speed link. In this way lower cost values represent higher speed links. The link weights of figure 6.1 are shown normalized in figure 6.3.

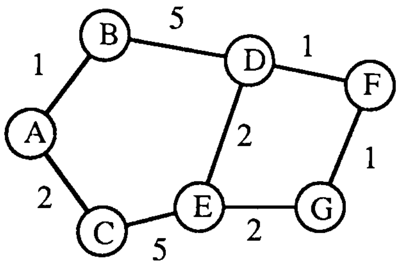


Figure 6.3 Simple network with link costs in form suitable for shortest path calculation.

A shortest path algorithm will calculate a cost of eight as the minimum path length between nodes A and G. Data packets that arrive at node A destined for node G will be routed along the path ABDFG. This path has a maximum carrying capacity of 2 units of traffic due to link BD. An alternative path algorithm, on the other hand, calculates not only the shortest path ABDFG but also a second path ACEG at a cost of nine. Using this alternative path during times of congestion will allow the flow rate between nodes A and G to approach that of the theoretical value.

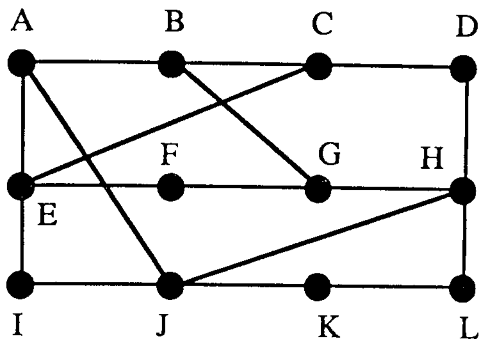
The use of alternative paths may be particularly important for time insensitive traffic such as file transfers. It may be far better to send these packets down sub-optimal links than to cause a transport layer retransmission if they are dropped.

In general, alternative path algorithms tend to be more complex than their single path counterparts. Additionally they require more memory at the switching node in order to store the alternative paths. On the other hand algorithms that provide alternative paths can generate a higher network performance in the face of congestion than algorithms that are restricted to using only the shortest paths.

The idea of providing alternative paths in the face of congestion is not particularly new. However, this chapter is concerned with designing an improvement on current alternative path routing algorithms which can be implemented in an ISIS router.

6.2 Previous Work on Alternative Path Algorithms

Tanenbaum discusses one very simple method for providing multiple routes from a source to a destination [1]. Each router maintains a table with one row for each possible destination. A row gives the best, second best, third best etc. outgoing line for that destination, together with a relative weight. Before forwarding a packet, a router generates a random number and then chooses among alternatives, using the weights as probabilities. The tables are worked out manually by the network operators, loaded into the routers and not changed thereafter. This is shown in figure 6.4.



Destination	First choice		Second choice		Third choice	
A	A	0.63	I	0.21	H	0.16
B	A	0.46	H	0.31	I	0.23
C	A	0.34	I	0.33	H	0.33
D	H	0.50	A	0.25	I	0.25
E	A	0.40	I	0.40	H	0.20
F	A	0.34	H	0.33	I	0.33
G	H	0.46	A	0.31	K	0.23
H	H	0.63	K	0.21	A	0.16
I	I	0.65	A	0.22	H	0.13
-						
K	K	0.67	H	0.22	A	0.11
L	K	0.42	H	0.42	A	0.16

Figure 6.4 Network and associated table for Tanenbaum routing procedure.

This method, although simple, does not permit adaptation to traffic changes. If traffic levels are very low, the algorithm may choose to send a packet on a sub-optimal path to the destination. In addition caution must be exercised in setting up the routing tables to avoid routing table loops. For example, if node J forwards a packet to destination A via H, node H must guarantee that the packet does not get returned to J. Various extensions to the basic algorithm exist to take care of these problems.

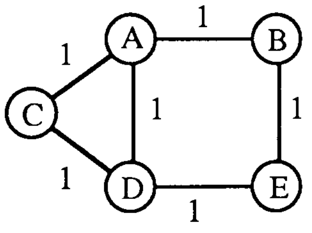
All of the remaining algorithms in this section are designed to operate in a network which allows packets to adapt to global traffic changes.

Nelson developed the NELHNET routing strategy [78]. This has been shown to provide a higher network performance than the ARPANET's SPF algorithm [77]. The algorithm was designed to operate in a tactical network in which both the topology and traffic patterns change within relatively short periods of time.

The NELHNET algorithm requires global topological information in order for the minimum hop paths to be calculated. The algorithm used to calculate the minimum hop and minimum hop plus one paths is not Dijkstra's algorithm but a more efficient mechanism which is only suitable for networks with equal cost links. The choice between using the minimum hop path or the minimum hop plus one path is based on the network traffic delays. In order to find the delays in the network it uses a similar technique to the

ARPANET algorithm to calculate the nodal delays. Although the delay calculation may be similar, the information distribution procedure is very different. In the ARPANET, the packet delays are averaged over a ten second period and this information is then flooded to all other nodes. The NELHNET strategy requires that the data packets include the latest delay information in the packet header. Nodes then recalculate the least delay path and are permitted to select either the minimum hop or minimum hop plus one route. This method distributes over time the intensive calculation procedure of the ARPANET.

The paper presents three algorithms. The first (NELHNET A) only selects paths from the minimum hop set. The second (NELHNET B) selects routes from the minimum hop and minimum hop plus one sets at the input node only. The third algorithm (NELHNET C) permits selection from both sets at any node. The final algorithm requires an extra bit of overhead in order to guarantee that the selection from the alternative path set is only performed once. If selection from the alternative path set occurs more than once, packet looping will be produced. Figure 6.5 shows the routing table required for the NELHNET routing strategies.



	Minimum		Min. + 1	
Dest.	Dist.	Next	Dist.	Next
A	0	0	0	0
B	1	B	-	-
C	1	C	2	D
D	1	D	2	C
E	2	B	3	C

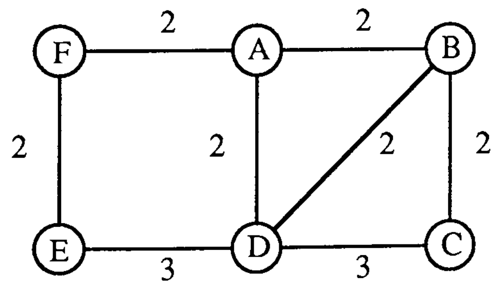
Figure 6.5 Network and associated routing table for NELHNET algorithm.

NELHNET algorithm A would not be used since NELHNET B offers better performance with no increase in packet overhead. An example of the NELHNET's algorithm B operating in the network of figure 6.5 is as follows. If a packet is destined from node A to node D and the delay on that path is greater than the delay experienced on the path ACD then the latter path can be used. NELHNET C is an extension of NELHNET B and operates as follows. A packet from node B destined for node C would take the shortest path via node A. If the link AC is experiencing congestion, then the alternative path AD can be used to get the packet to node C. This route takes only one hop more than the minimum. A one bit flag in the packet header is then set to ensure that future nodes only route the packet using the minimum hop set. If this header bit is not set, then node D could possibly route the packet back up link DA, if link DC was experiencing congestion.

Wang has developed the algorithm he terms Shortest Path First with Emergency Exits (SPF-EE) which has been designed to provide an improvement to the ARPANET's SPF algorithm. This algorithm is not restricted to the use of unity cost links as the NELHNET algorithms are.

Using this algorithm, packets are sent down the shortest path routes during low congestion and are allowed to take alternative paths, or emergency exits, during times of heavy congestion. The algorithm uses the same distance measurement procedure and information updating procedure as the ARPANET algorithm. In this way the algorithm attempts to adapt to traffic fluctuation on a global basis. The SPF-EE algorithm requires modification to both the route computation and the packet forwarding procedures. The other procedures are the same as the ARPANET's.

The algorithm that is used to calculate the alternative paths (AP) is very simple. After having calculated the shortest paths, it then derives the routing trees for each of its neighbouring nodes. The paths for each destination via these neighbours is then stored in the alternative path database. The computation complexity of this algorithm is $O(\mathrm{D}n^2)$, where O means on the order of, D is the average degree of the network and n is the number of nodes in the network. The simplicity of the AP algorithm means that it is incapable of finding all the available alternative paths (figure 6.6) from the node doing the calculation. Note that node F can be reached from node A via node D with a cost of 7, the SPF-EE alternative path algorithm is unable to find this path.



	Shortest Dist.	Shortest Next	Alternative Dist.	Alternative Next
A	0	0	0	0
B	2	B	4	D
C	4	B	5	D
D	2	D	4	B
E	4	F	5	D
F	2	F	-	-

Figure 6.6 Network and associated routing table for Wang algorithm.

In order to find all of the alternative paths in a network the SPF-EE algorithm must resort to what Wang terms reverse alternative paths (RAP). In order to find a RAP during times of congestion, an initiating node sends a query message concerning a particular destination to each of its neighbouring nodes. The neighbours check their routing tables for this destination to see if they can provide a suitable alternative path (exit) for the packet. If an exit is found then the initiating node is informed, otherwise the neighbouring nodes send query messages to their neighbours. This process repeats until an exit is found or the number of neighbours to be checked reaches a defined limit. The RAP for the network of figure 6.6 is shown in figure 6.7.

Source	Destination	Intermediate Hops	Exit
A	F	D	E

Figure 6.7 RAP table required for Wang algorithm.

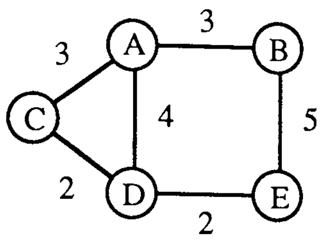
Although this method will eventually find all the alternative paths if they exist, it is a very inefficient and cumbersome method. While nodes are attempting to establish RAPs the data packet has to sit at the initiating node and wait. Even for time insensitive traffic such as file transfers this is totally unacceptable.

In addition this method may lead to packet looping. To quote Wang "It is interesting to note that in some cases the packet may be sent backwards to the nodes it just came from and then take a different path". From this statement it can be concluded that the use of the SPF-EE algorithm does not provide a feasible solution to the alternative path problem for interconnected FDDI LANs.

In order to calculate all routes from a source to a destination, regardless of cost, a k shortest path algorithm can be used. These algorithms generally require global topological information in order to find the k best paths. If the number of alternative paths required is large (ie large k) then the amount of processing required by the algorithm is quite extensive.

The French DATAPAC network [33,79] utilizes such an algorithm. In this network a dedicated central controller collects information about the delays on all links and calculates the k shortest paths for every source/destination pair. The routes to be used are then forwarded to all of the nodes of the network. This algorithm is also distributed in that the individual nodes may select which of the k paths to send a packet down based on the outgoing queue lengths. The routers can choose one of the alternative paths if congestion occurs on the shortest path. The central controller must run the k shortest path algorithm quite frequently in order to keep track of the network wide delays.

The routing table required for this algorithm is shown in figure 6.8 below.



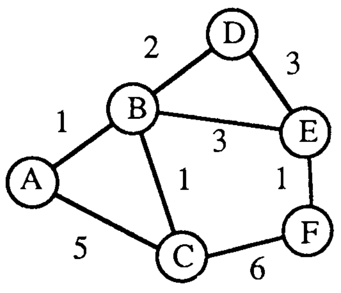
	Shortest		1st Alt.		2nd Alt.	
Dest.	Dist.	Next	Dist.	Next	Dist.	Next
A	0	0	0	0	0	0
B	3	B	11	D	12	C
C	3	C	6	D	12	B
D	4	D	5	C	10	B
E	6	D	7	C	8	B

Figure 6.8 Network and associated routing table for DATAPAC algorithm.

6.3 k Shortest Path Algorithms

Of the above methods for calculating alternative paths, the k shortest path algorithms appear to be the most flexible. These algorithms can calculate the best, second best, third best etc paths in a network without any restrictions on link costs. There are basically three different versions of this algorithm. In the first version, no k shortest path is allowed to contain repeated nodes. A second version of the problem allows the k shortest paths to contain repeated nodes. Much research has been done on versions one and two of this algorithm [80,81,82,83]. A newer implementation, version three, proposed by Topkis requires only that the initial links of the alternative paths are distinct from those of the shortest paths [79].

These three versions of the k shortest path and how they relate to a simple network are shown in figure 6.9. In this figure the shortest paths and the alternative paths from node A to node F are shown.



	Shortest path	Cost	Alternative path	Cost
Version 1	ABEF	5	ACF	11
Version 2	ABEF	5	ABDEF	7
Version 3	ABEF	5	ACBEF	10

Figure 6.9 Shortest and alternative paths for three versions of the k shortest path problem.

Algorithms that find completely disjoint paths are satisfactory when there are a large number of links in the network to choose from. However, this severely limits the number of permitted alternative paths if the network is sparsely connected. If a router has global information on a network's delays, then it is in a suitable position to specify that one path is superior to another path. Version one of this algorithm may be satisfactory for such a network. If, however, only local information on the outgoing queue lengths is available to the router, then it is not really in a position to completely disregard one entire path over another. A router with only local information is only able to claim that one outgoing link offers better performance than some other outgoing link. Finally, algorithms that attempt to find the k shortest disjoint paths in a network require a large amount of computational effort.

The second type of algorithm which allows repetition of nodes is also not completely satisfactory. It is possible that when using this algorithm the alternative path contains the same initial link as that of the shortest paths. If the queue for a link exceeds some threshold, then the router may select a path from the alternative routing set. If the initial link of the alternative path set is the same as that from the shortest path set then no gain is made and the packet may become rejected, depending on the threshold value.

The simplest of the three versions of k shortest path algorithm is the third type. These algorithms require only that the initial link is different from that of the shortest path. The algorithm is unconcerned with whether or not succeeding links are disjoint from the shortest path set or not. This algorithm is ideally suited for a routing algorithm which only adapts to traffic changes on a local basis. The computational time for this type of algorithm can be far lower than both version one and two. This type of k shortest path algorithm will be used as the basis for the new algorithms.

6.4 New Alternative Path Algorithms

The ISIS routing algorithm is fundamentally a single path algorithm and requires modification if it is to be used to calculate an alternative path set. Two new routing algorithms based on a modified ISIS are presented in this chapter. Algorithms A and B allow the alternative routing set to contain minimum cost plus one paths. Algorithm A only permits selection of these paths at the initial node, whereas algorithm B allows selection at either the initial node or somewhere along the path. Algorithm C allows the alternative path set to take any value greater than the shortest path set. Selection from this set is only permitted at the initial node.

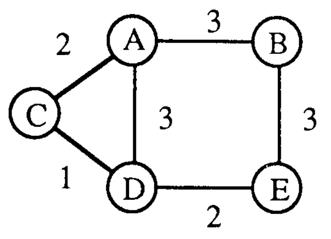
All three algorithms allow selection from the alternative path set based on the outgoing queue lengths of the adjoining links. No attempt is made to adjust routes according to the global traffic conditions in the network. This is the same traffic adaptation policy that the standard ISIS algorithm permits when choosing between two equal cost paths.

The route computation procedures for the three new algorithms are very similar and are discussed below.

6.4.1 Route Computation Procedure - Algorithms A,B and C

It is possible to build a k shortest path algorithm out of the standard ISIS shortest path algorithm. This can be achieved by using multiple runs of the standard algorithm. Using this method it is possible to calculate all three versions of the k shortest path problem. However, in this chapter only version three of the algorithm will be discussed.

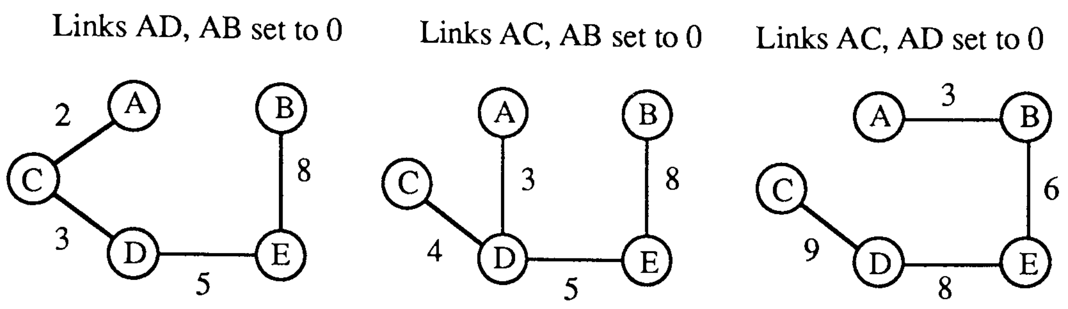
In order to calculate alternative paths with disjoint initial links, the cost to each adjacency of the source node is temporarily set to zero. Figure 6.10 shows a simple network and the topology database with all links included.



	A	B	C	D	E
A	0	3	2	3	0
B	3	0	0	0	3
C	2	0	0	1	0
D	3	0	1	0	2
E	0	3	0	2	0

Figure 6.10 Simple network and topology table with all links present.

By temporarily masking out all but one of a node's adjacencies in the topology database and running the algorithm, the shortest paths via that adjacency can be found. After having found the shortest paths via that link, the link is reinstated into the topology database and the next link is removed. This is done for each adjacency of the source. Figure 6.11 shows the shortest path trees rooted at node A (from network of figure 6.10) for each of its three adjacencies.



Dest.	Dist.	Next
A	0	0
B	8	C
C	2	C
D	3	C
E	5	C

Dest.	Dist.	Next
A	0	0
B	8	D
C	4	D
D	3	D
E	5	D

Dest.	Dist.	Next
A	0	0
B	3	B
C	9	B
D	8	B
E	6	B

Figure 6.11 Resulting shortest path tables after masking adjacencies.

The shortest paths that are generated by each of these runs can then be sorted to find the shortest path, the 2nd best, third best etc. paths to each destination. Since the algorithm is basically Dijkstra's algorithm with masked links, the alternative path set is guaranteed to be loop free. The final routing table is shown in figure 6.12. In this diagram only one of the alternative paths is shown.

	Shortest Path		Alternative Path	
Dest.	Dist.	Next	Dist.	Next
A	0	0	0	0
B	3	B	8	C,D
C	2	C	4	D
D	3	C,D	8	B
E	5	C,D	6	B

Figure 6.12 Final routing table showing shortest and alternative paths.

The route computation procedures of the three algorithms are slightly different when it comes to storing the alternative paths. Algorithms A and B must only store alternative paths if they offer a cost to the destination which is one greater than that offered by the shortest path. Algorithm C can store any value in the alternative path set. Therefore, in the above example, the alternative path routing set is shown for algorithm C. For algorithms

A and B the routing table is shown in figure 6.13. In order to reduce the amount of nodal storage space required, the three algorithms only permit one alternative path to be stored.

	Shortest Path		Alternative Path	
Dest.	Dist.	Next	Dist.	Next
A	0	0	0	0
B	3	B	-	-
C	2	C	-	-
D	3	C,D	-	-
E	5	C,D	6	B

Figure 6.13 Routing table for algorithms A and B.

The route computation procedure pseudo-code for the three algorithms is shown in appendix E.

The initial link disjoint algorithm developed by Topkis [79] is somewhat more efficient than the above route computation procedure. In his algorithm each shortest path calculation exploits information generated by the previous iteration. In this way the computational complexity is only $O(\text{kn}^2)$, where k is the number of shortest paths required. When k is one, the complexity is equal to that of the Dijkstra shortest path algorithm, namely $O(\text{n}^2)$. The worst case computational complexity of algorithm A and B is $O(\text{dn}^2)$, where d is the number of adjacencies of the node doing the calculation.

Therefore if the number of paths required (k) from a source is less than the number of adjacencies of that source (d) then the Topkis algorithm, in terms of efficiency, will outperform the route computation procedure of algorithms A and B. However, when k=d the computational complexity of the algorithms is the same. For example, if a node has three neighbours and three paths were required, then the computation complexity of the algorithms will be exactly the same. However, if a node has three neighbours and only two paths are required, the Topkis algorithm is more efficient.

This may suggest that the above algorithm should be used for calculating the alternative paths in the network. If it is used, the forwarding procedures for algorithms A and B (discussed in the next section) will remain the same.

However, there is still a case for the route computation procedure discussed in this section. Topkis's algorithm has been designed for use in the DATAPAC network where the algorithm must be run each time a new network delay estimate arrives. This could potentially be every couple of seconds. An environment such as this requires that the most efficient algorithm available is used in order to minimize the required processing time. The ISIS algorithm makes absolutely no attempt to adjust to global traffic levels, and the only time the algorithm is run is when a topology failure occurs. Since this is the case, the speed that the Topkis algorithm provides may be unnecessary. In an interconnected LAN environment, where links and nodes fail quite infrequently, the computation procedures of algorithms A and B may be satisfactory.

Another argument in favour of the new algorithms over the Topkis algorithm is that the former requires no modification to the standard ISIS routing procedure. The algorithm is called by a subroutine which simply modifies the topology database before calling the ISIS algorithm.

6.4.2 Packet Forwarding Procedure (Algorithm A)

The packet forwarding procedure of a routing algorithm is responsible for sending a packet from one node to the next. In the case of a shortest path algorithm this is simply the next node listed in the routing table. If the algorithm permits equal cost path splitting, as ISIS does, then the forwarding procedure can make the selection based on the outgoing queue lengths.

For an alternative path algorithm, the forwarding procedure can use the secondary paths under a variety of conditions. Some of these include:

- queue lengths exceeding some threshold
- during link or node failure
- randomly

In this forwarding procedure, the alternative paths will be used when the queue length to the preferred path exceeds some threshold. The effects of varying the threshold value will be shown in the simulation results. In addition, the use of the alternative paths has been limited to traffic which is <u>not</u> time sensitive. It is often better to drop time sensitive traffic, such as real time packet voice, than to route it down alternative higher cost paths.

The forwarding procedure for algorithm A requires that the data packets contain a one bit flag in the packet header. When set, this flag indicates that an alternative path has been selected somewhere on the path. When a node receives a packet the forwarding procedure checks this bit and, if set, the next node selection must <u>only</u> be made from the shortest path group.

If the threshold exceeds some value and multiple equal cost alternative paths exist, then the algorithm chooses the alternative path with the lowest outgoing queue length. This is the same technique used for selecting between two equal cost shortest paths. Algorithms B and C also use this method to select between equal cost alternative paths.

6.4.3 Packet Forwarding Procedure (Algorithm B)

The forwarding procedure of algorithm B is similar to that of algorithm A. However, the choice of the alternative paths is limited to selection at the initiating node only. This reduces the overhead required by the algorithm.

6.4.4 Packet Forwarding Procedure (Algorithm C)

The forwarding procedure of algorithm C is similar to that of algorithms A and B in that alternative paths are used when the number of packets in the queue for the primary path exceeds some threshold. The selection of these paths is restricted to the initial node only.

If the alternative paths for algorithm C are allowed to take any value greater than those of the shortest paths, then it is not possible to allow selection from the alternative path set to occur arbitrarily in the network. This selection can only be made at the initial node, even if a one bit flag is included in the packet header as in algorithm A. This can be seen from the example below.

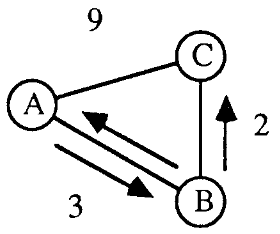


Figure 6.14 Possible packet looping in algorithm C.

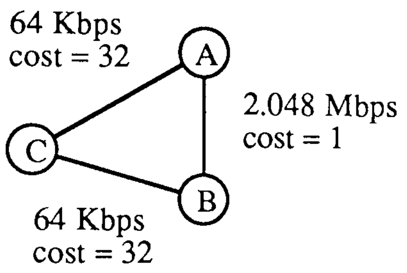
Assume that a packet begins at node A (figure 6.14) and is destined for node C. The shortest path from A to C is a cost of 5 via node B. If, when the packet arrives at B, congestion is encountered on link CB and assuming selection is permitted from the alternative paths, node B will send the packet back to node A on link BA. Having set the bit to a one in the packet header (alternative path chosen flag), the selection at node A must be from the shortest path set. This packet is then sent back to node B where the selection of the next node must also be made from the shortest path set. Only after the loop AB, BA, AB does the packet arrive at the destination node C.

The forwarding procedure pseudo-code for algorithms A, B and C is shown in appendix E.

6.5 Summary - Algorithms A and B

Algorithms A and B are both based on the NELHNET routing strategy. However, there are some important differences between the two.

The route computation procedure of NELHNET has been designed to operate in a network where link costs are all the same. Algorithms A and B allow the link costs to take on any value. The alternative paths are then limited to the shortest path plus one routes. This is achieved using multiple runs of the ISIS shortest path algorithm.



	Shortest		Alternative	
Dest.	Dist.	Next	Dist.	Next
A	0	0	0	0
B	1	B	-	-
C	32	C	33	B

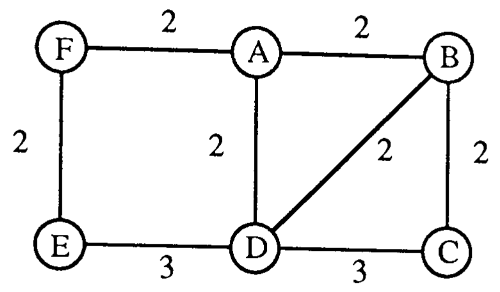
Figure 6.15 Simple multi-cost network and associated routing table for algorithms A and B.

In a network which supports links of multiple costs, the highest speed link can be defined as having a cost of unity. All other lower speed links in the network can have their costs related to the highest speed link by dividing the former by the latter. For a network with 2.048 Mbps and 64 kbps links this would result in costs of 1 and 32 respectively. This can be seen in figure 6.15.

NELHNET also attempts to route packets based on the global traffic delay. This requires precious nodal processing time in order to calculate paths and update neighbours. Algorithms A and B are based on the ISIS protocol which only attempts to adapt to traffic locally depending on the outgoing queue lengths.

6.6 Summary - Algorithm C

Algorithm C is capable of finding initial link disjoint alternative paths in the network if they exist. This is in contrast to the SPF-EE algorithm developed by Wang which requires the use of RAPs in order to find all the alternative paths. For the network shown in figure 6.16, the SPF-EE algorithm developed by Wang is incapable of finding the route to F via node D without having to resort to RAPs. Multiple runnings of the ISIS routing algorithm can easily find these paths. It is very wasteful of resources if the node must go searching for an alternative path, particularly when global topological information is available to the node.



	Shortest Dist.	Shortest Next	Alternative Dist.	Alternative Next
A	0	0	0	0
B	2	B	4	D
C	4	B	5	D
D	2	D	4	B
E	4	F	5	D
F	2	F	7	D

Figure 6.16 Simple network and associated routing table for algorithm C.

The k shortest path algorithm developed for the DATAPAC network is a centralized approach to finding alternative paths [33]. This network attempts to adapt to traffic fluctuations on a global basis and requires large processing power at the central node. Using multiple runs of the ISIS algorithm, the k shortest path problem can be used in a distributed environment where traffic adjustments are made on a local basis.

Table 6.1 summarizes the differences between the three algorithms and those discussed in the literature.

Strategy	Traffic adaptation	Link costs		Route computation procedure	Point of alt. path selection
		Shortest	Alternative		
NELHNET	global	hops	hops+1	Min. hop algorithm	any node
DATAPAC	global/local	any	any	K-Shortest algorithm	initial node
Algorithm A	local	any	shortest+1	Multiple ISIS runs	any node
Algorithm B	local	any	shortest+1	Multiple ISIS runs	initial node
Algorithm C	local	any	any	Multiple ISIS runs	initial node

Table 6.1 Comparison of literature algorithms with algorithms A, B and C.

6.7 Performance of ISIS and Alternative Path Algorithms

In order to compare the new algorithms with the ISIS protocol it is again necessary to resort to simulation. The effect of topology on the routing algorithms will be seen to be a critical parameter which must be considered when selecting an algorithm. It is not possible to examine every different topology possible for a given network size. All of the networks in this chapter have been randomly generated and those topologies which produced large numbers of alternative paths have been used for the simulation runs.

It must be emphasized that although many of the networks in this chapter are shown with equal cost links, the three algorithms can work with any network regardless of link costs.

6.7.1 Algorithms A and B

For each network, two different forwarding procedures are simulated. The first allows the alternative paths to be used only when the outgoing queue length to the preferred path exceeds the maximum buffer size. The threshold in this case is defined as the maximum buffer length. Therefore, alternative paths are only used when a packet is in imminent danger of being dropped by the network. The second forwarding procedure takes the other extreme and forwards the packets along either the shortest path or the alternative path depending on which offers the shortest queue length. If the preferred path has a queue length of five and the alternative has a queue length of four, then the alternative path will be chosen. This forwarding procedure will be termed low threshold. A real implementation of an alternative path algorithm would probably be within these two extremes.

The first network to be simulated is the five node network shown below (figure 6.17). This network offers a total of 18 alternative paths with a cost one greater than the minimum. For this network, algorithm C also produced the exact same 18 alternative paths.

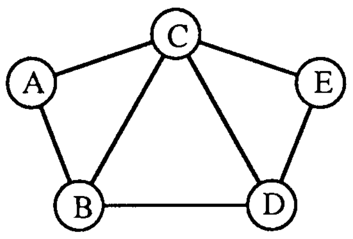


Figure 6.17 Simulated five node network.

The power produced by algorithms A and B for the different types of forwarding procedure are shown in figure 6.18. A high threshold setting is denoted by HT and a low setting by LT.

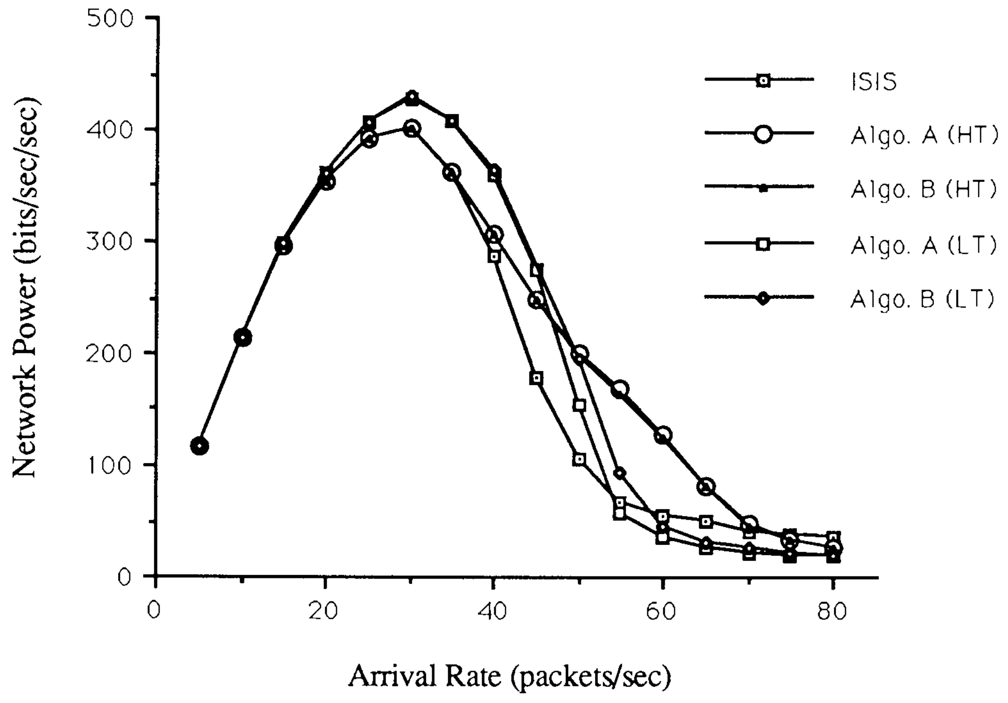


Figure 6.18 Power against arrival rate (5 nodes, 7 links)

The percentage gain of these algorithms over the ISIS are shown in figure 6.19.

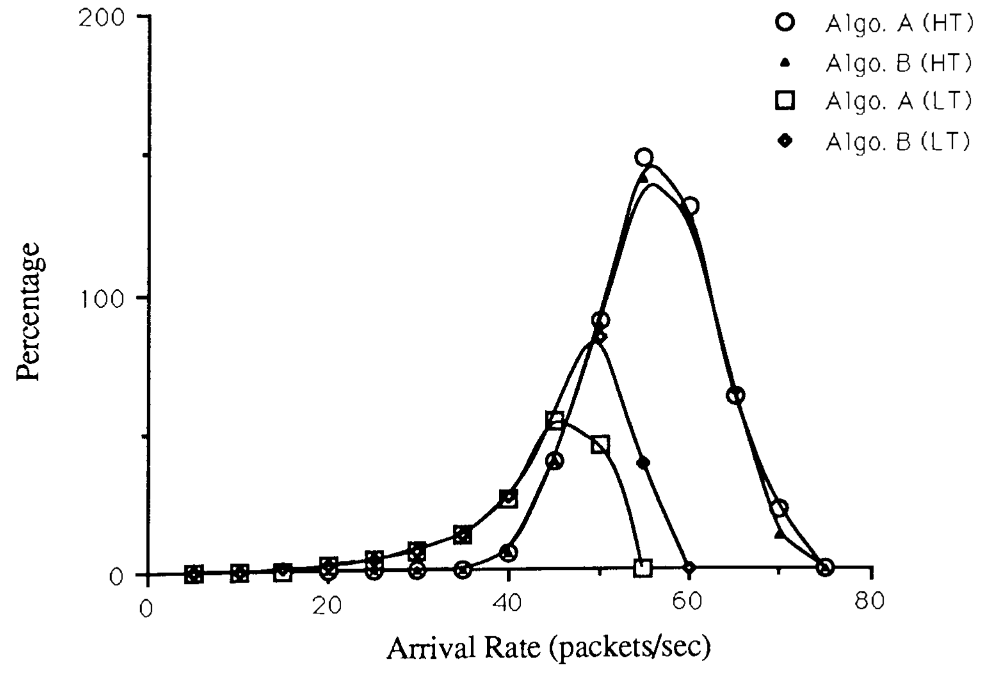


Figure 6.19 Percentage gain of algorithms A and B over ISIS (5 nodes, 7 links)

For a 10 node network the following power graph was produced (figure 6.20).

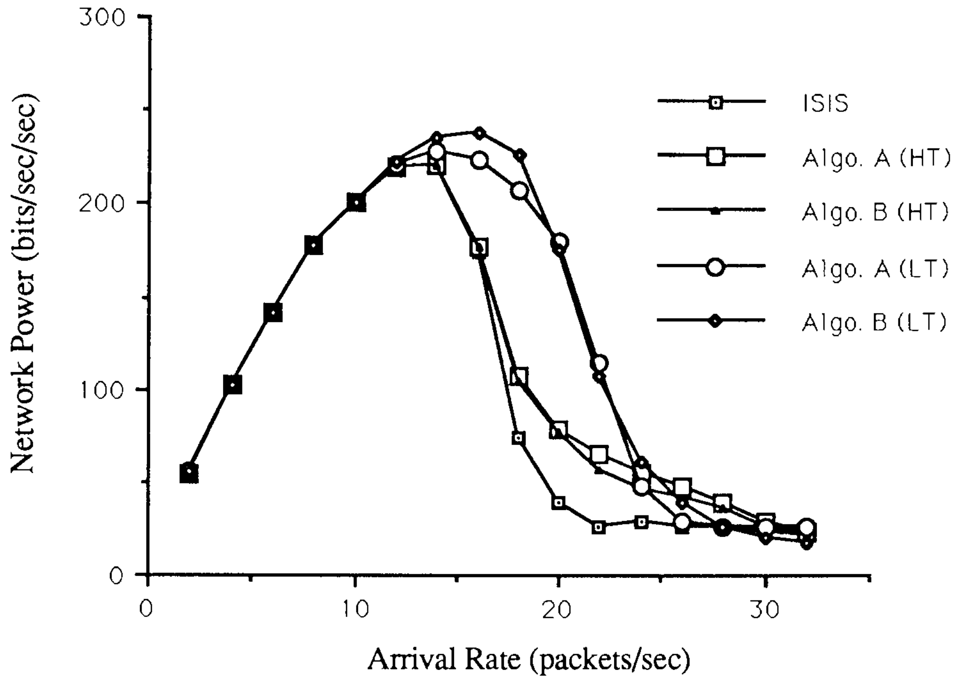


Figure 6.20 Power against arrival rate (10 nodes, 14 links)

The percentage improvement offered by Algorithms A and B over the standard ISIS protocol are shown in figure 6.21.

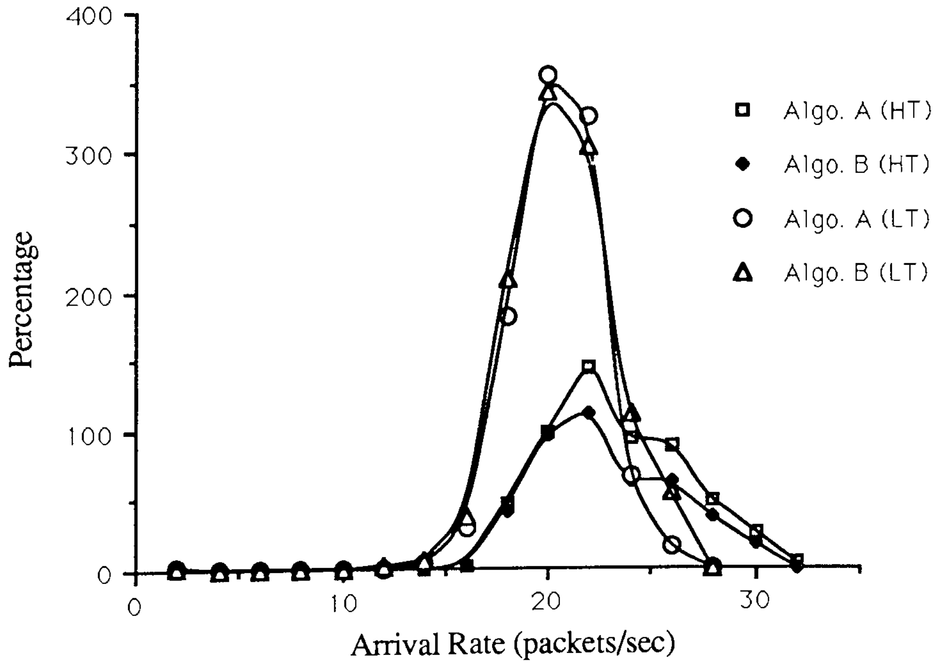


Figure 6.21 Percentage gain of algorithms A and B over ISIS (10 nodes, 14 links)

Figure 6.22 shows the simulation results for a 15 node network.

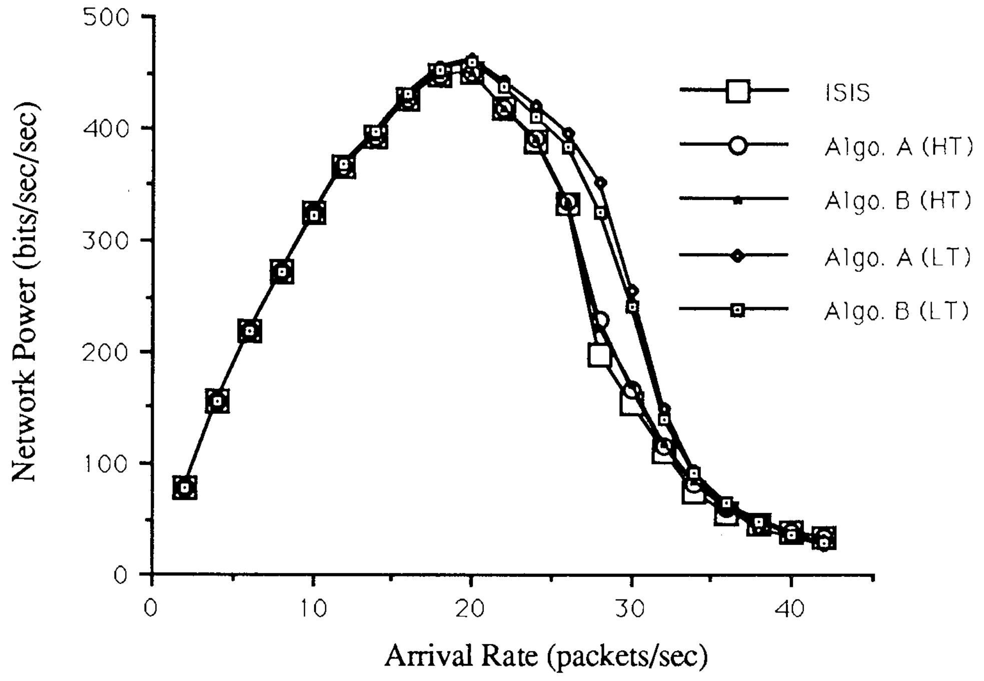


Figure 6.22 Power against arrival rate (15 nodes, 22 links)

The percentage improvement of algorithms A and B over the ISIS are shown in figure 6.23 below.

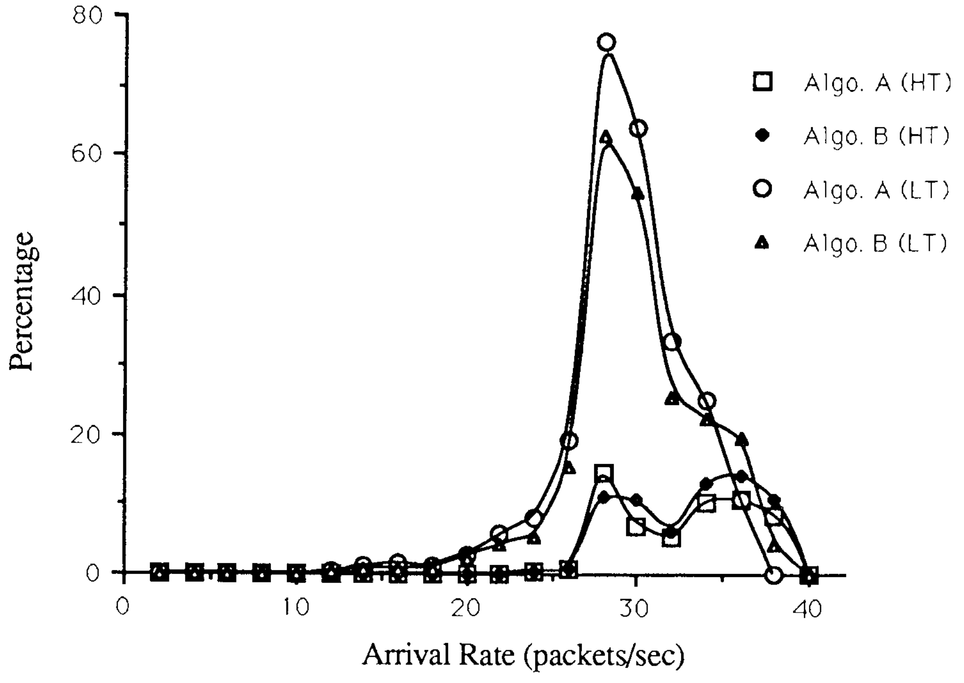


Figure 6.23 Percentage gain of algorithms A and B over ISIS (15 nodes, 22 links)

6.7.2 Algorithm C

The simulation results for algorithm C are shown in this section. The threshold value for all the simulations is set to the maximum buffer length. This means that selection of the alternative paths occurs only when the buffer to the preferred link is at the point of overflowing. It would be unwise to choose between the shortest path and the alternative path based on which offered the shortest outgoing queue length as in the previous simulations for algorithms A and B; the reason being that the value of the alternative path set may be much greater than the shortest paths since they are not limited to a cost of one greater than the minimum as they are in algorithms A and B. If these paths were selected based on shortest queue length, it is inevitable that the performance of the network would deteriorate.

The six node network in figure 6.24 is used for the first simulation run. In this network, algorithm C finds a total of 26 alternative paths. Note that the alternative path routing set for algorithms A and B would be empty for this network.

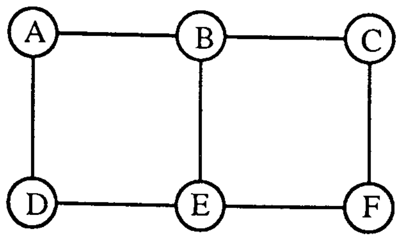


Figure 6.24 Six node network used for simulation.

The amount of power generated by the ISIS algorithm and algorithm C for this network is shown in figure 6.25.

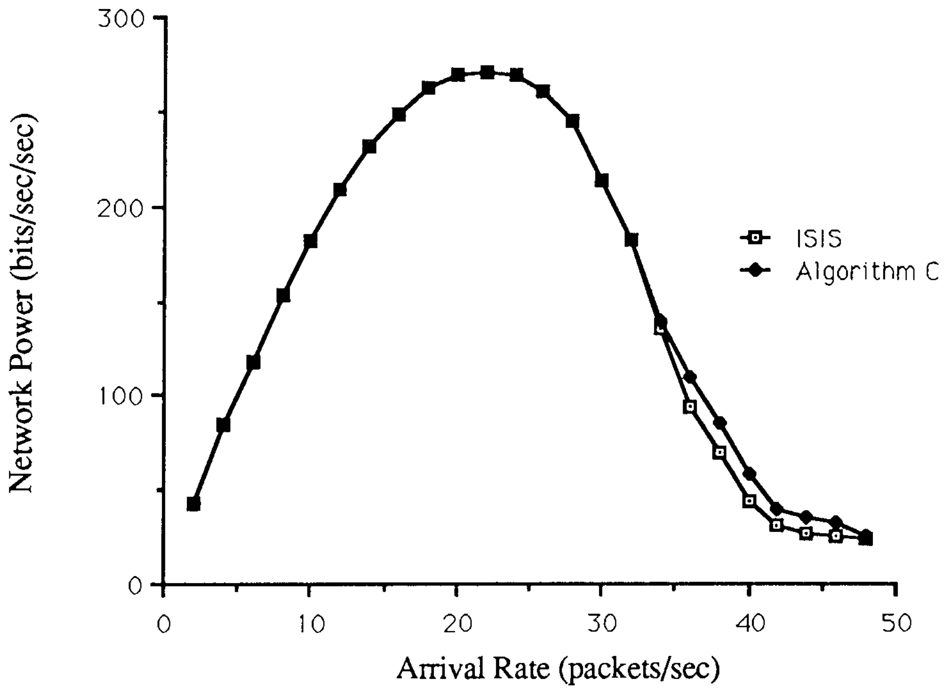


Figure 6.25 Power against arrival rate (6 nodes, 7 links).

Figure 6.26 shows the percentage gain that algorithm C provides over the ISIS algorithm.

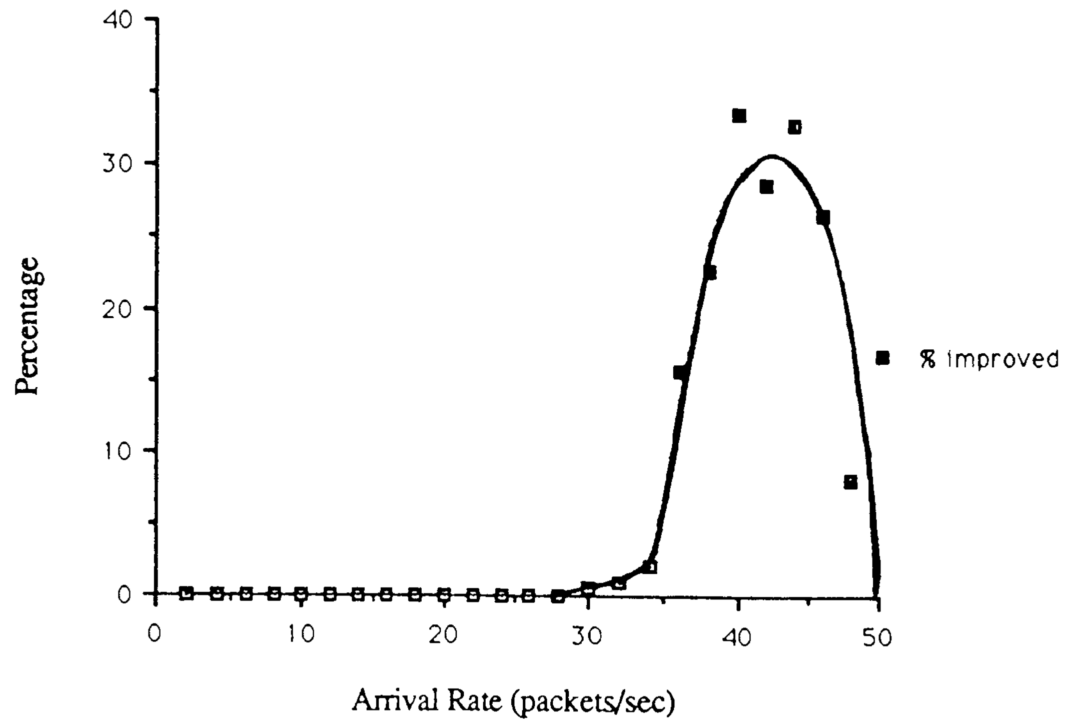


Figure 6.26 Percentage gain of algorithm C over ISIS (6 nodes, 7 links)

For a ten node network, the following simulation results were produced (figure 6.27).

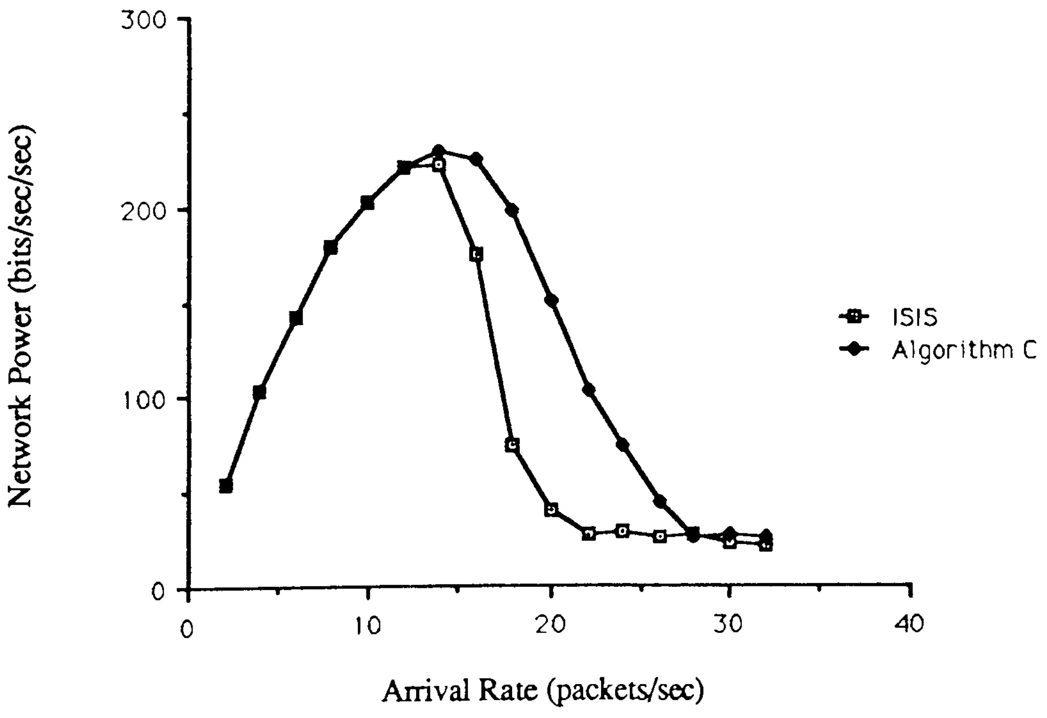


Figure 6.27 Power against arrival rate (10 nodes, 12 links)

The percentage gain provided by algorithm C for this network is shown below (6.28).

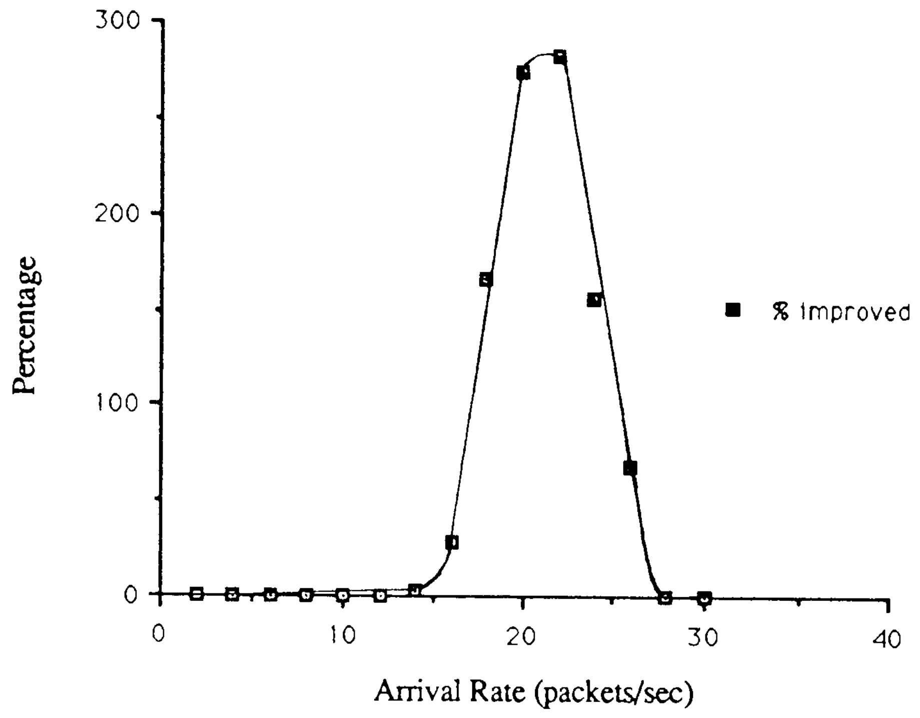


Figure 6.28 Percentage gain of algorithm C over ISIS (10 nodes, 12 links).

The simulation results for a fifteen node network are shown below in figure 6.29.

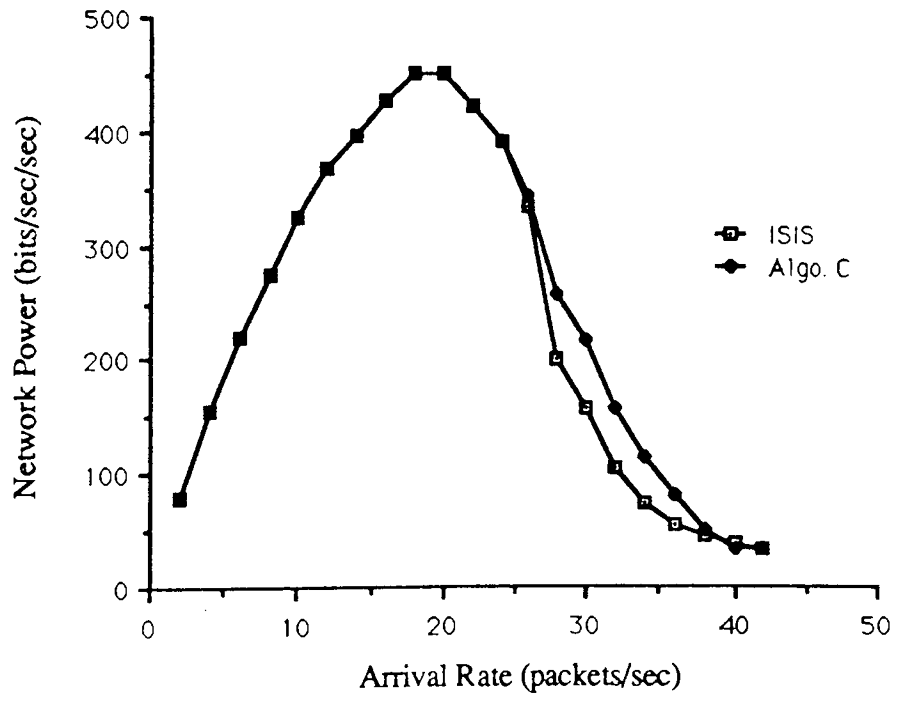


Figure 6.29 Power against arrival rate (15 nodes, 22 links)

The percentage gain generated by the new algorithm is shown in figure 6.30.

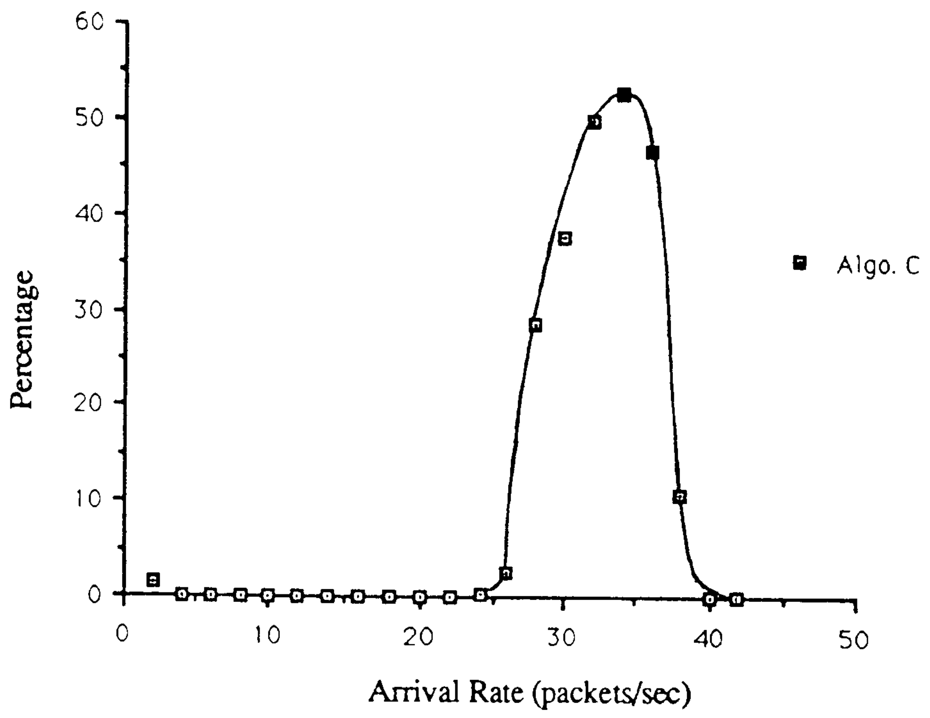


Figure 6.30 Percentage gain of algorithm C over ISIS (15 nodes, 22 links).

Additional simulation results for all three algorithms are shown in appendix F.

6.8 Discussion

The purpose of all three algorithms is to improve the network performance in regions three and four of the power graphs. Recall from chapter 5 that region three represents the rolling off of the power graph from the optimal arrival rate. Region four is the area where packets are being dropped and transport layer retransmissions become more apparent.

All three of the algorithms are capable of providing some sort of performance gain in the desired regions. The amount of gain is highly dependent on the topology and the forwarding procedure. Due to the topology dependence it is unnecessary to compare the performance of algorithms A and B with that of algorithm C. For some topologies algorithm A and B generated alternative paths which were exactly the same as algorithm C. In such a case, no gain difference would be seen between algorithm B, with a high threshold value and algorithm C. In addition, some of the topologies generated resulted in no alternative paths for algorithm A and B, whereas algorithm C would generate many alternative paths.

The alternative path algorithms discussed in this chapter can operate with any topology with more than two nodes. This differs from the DSPS algorithm which must have at least six nodes before any different down stream path splits exist.

The packet forwarding procedure for algorithms A and B can be optimized, or tuned, for each network to allow the alternative paths to be taken during times of congestion. Congestion is a relative term and can be defined in the forwarding procedure by setting the threshold value. Since the alternative paths for algorithms A and B are only one greater than the minimum, it is possible to set the threshold value very low and improve network performance. This is not possible for algorithm C.

The simulation results show the effects of setting the threshold level to a high and low value. Setting a high threshold level improves performance in the top end of region three and all of region four. A low threshold value can improve the performance across all of region three and some of region four. However, for the low threshold setting, the simulation results show that a degradation of performance occurs under very heavy loading. In addition, setting a low threshold value will increase the amount of processing required for each packet. The reason for this is that queue lengths for both the shortest path and the alternative paths must be checked to ascertain which offers the shortest queue. For high value threshold setting, the queue lengths for the alternative paths are only checked when the outgoing queue to the shortest path reaches the maximum buffer length. It is therefore necessary to strike a balance between performance and router processing when setting the threshold value in the packet forwarding procedure.

It is interesting to note that in the simulation results for five nodes, algorithm A actually performed less well than algorithm B for low threshold settings. However, under high threshold values, algorithm A outperformed algorithm B.

Algorithm C is permitted to use any alternative path if it exists. In the simulation runs, no upper bound was placed on the value that the alternative paths could take. If the shortest path offered a cost of 10 to a destination, it is feasible that the alternative path could offer a cost many times this value. In an actual implementation it would be wise to limit the values permitted for these alternative paths. This is particularly true in small networks. If large value alternative paths are used, the increased average delay caused by using these paths may actually be greater than retransmitting the packet.

All three algorithms require up to one third additional memory storage space for the alternative paths. Algorithm A also requires that one bit of overhead is included in the packet header to indicate when an alternative path has been selected. The actual overhead of one bit is quite trivial, but it does require that all nodes in the network understand what is meant by this bit. If this routing algorithm is run in a network with a large number of ISIS routers, then in order for algorithm A to work properly, not only must the ISIS routers store alternative paths but they must also recognize what the one bit flag in the packet header signifies. Algorithms B and C make no such demand on the other routers.

Apart from the complexity and the additional storage space required by the algorithms the following table (table 6.2) lists some of the advantages and disadvantages of the three algorithms.

	Advantage	Disadvantage
Algorithm A	-Selection of alt. path anywhere in network. - Threshold tuning.	- 1 bit overhead for flag. - Requires other routers to understand flag. - Alt. paths are limited in cost.
Algorithm B	- Threshold tuning.	- Alt. paths are limited in cost.
Algorithm C	- Alt. paths can have any cost.	- Performance improvement limited to high traffic load.

Table 6.2 Advantages and disadvantages of algorithms A,B and C.

The selection of a particular algorithm is highly dependent on the network topology. If there are a large number of minimum cost plus one paths in a network, then algorithm A will provide the best performance. Algorithm B will perform quite satisfactorily in the same network without the one bit of overhead. Finally, algorithm C is ideally suited for networks which have very few minimum cost plus one paths.

6.9 Conclusion

The ISIS routing algorithm is essentially a single path algorithm which generates a performance far below that of the theoretical potential. Alternative path algorithms, whilst being more complex, are capable of approaching this potential. The new algorithms require approximately one third more storage space than the standard ISIS algorithm. This increase in storage is common for all alternative path algorithms. There is also a small amount of additional processing required during times of congestion to select the alternative paths.

Previous work on alternative path algorithms has centred around schemes designed to react to network-wide traffic delays. The three new algorithms presented in this chapter are all based on the ISIS routing protocol and, like that algorithm, only attempt to adapt to traffic on a local basis.

Multiple runs of the ISIS shortest path algorithm are used to calculate a selection of alternative paths. This is perhaps not the most efficient method of calculating the k shortest paths, it is however, the most flexible. It is capable of generating the shortest path and alternative paths regardless of cost. It can produce initial link disjoint alternative paths or even path disjoint alternative routes if required. The latter can be controlled by defining the initial conditions before running the ISIS routing algorithm. Finally, it reqɹires no modification to the actual ISIS route calculation procedure, only a short subroutine to call this procedure.

It is possible to tune the algorithms for a particular network implementation by adjusting the threshold value in the packet forwarding procedure. Even without tuning the algorithms for each network, power gains as high as 150% over the ISIS algorithm have been achieved.

The three algorithms presented are far simpler than the alternative path algorithms proposed in the literature. They can all be used with an ISIS router without significant modification to the existing routing procedures.

The next chapter will examine the performance improvement provided by an algorithm that allows both down stream path splits and alternative paths.

Chapter 7

Combination Routing Algorithms

7.0 Introduction

The Down Stream Path Splits algorithm offers an improvement on the performance of the ISIS algorithm during times of medium to high congestion levels. Under high traffic loading the down stream path splits credit system is effectively suspended and performance of the algorithm is equivalent to that of the ISIS.

The alternative path algorithms A, B and C are most effective during times of high to very high traffic levels. During low to medium levels of traffic, the alternative path algorithms only select the shortest paths to a destination and no improvement is made over the ISIS.

In order to develop an algorithm which provides an increase in performance across a wide spectrum of traffic arrival rates, it is possible to combine the DSPS algorithm with the alternative path algorithms. This will allow the resulting algorithm to use the DSPS credit system during times of medium/high loading and the alternative path sets during high/very high traffic loading.

This invariably results in a more complex algorithm than those presented in the previous chapters. In addition, more router memory is required in order to store the two sets of information. However, the performance gain achieved by these algorithms may justify the complexity and memory increases. Even after combining these two types of algorithm they are still far simpler than many of the routing algorithms proposed in the literature [30,42,72].

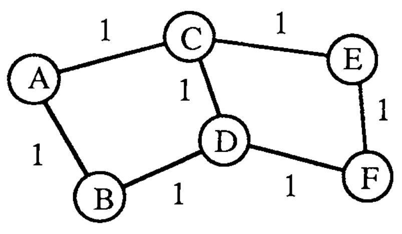
The combined algorithms in this chapter will have the same names as those in chapter 6, but with the extension of + DSPS.

7.1 Route Computation Procedure - Algorithms A+DSPS, B+DSPS and C+DSPS

The route computation procedures for the three combination algorithms are all very similar. They are all based on a concatenation of the DSPS algorithm and the alternative path computation algorithm.

The route computation procedure for the down stream path splits algorithm is embedded in the standard ISIS calculation. When the ISIS shortest path algorithm is performed the down stream path splits for the shortest paths to each destination are produced. Therefore, when the algorithm is run multiple times in order to find the alternative paths, the down stream path splits for each link are simply a by-product of the calculation. For every alternative path produced by the algorithm, there will be an associated DSPS value. These down stream path splits are then stored next to each of the shortest paths generated. The complexity required for calculation of the algorithm is no greater than that for the alternative path algorithms, namely $O(dn^2)$.

Algorithms A+DSPS and B+DSPS again allow the storage of alternative paths, if they offer a cost one greater than the minimum. Algorithm C+DSPS allows the storage of alternative paths regardless of cost. A routing table for a six node network running algorithm C+DSPS is shown in figure 7.1.



Routing table stored at node A

	Shortest			Alternative		
Dest.	Dist.	Adj.	DSPS	Dist.	Adj.	DSPS
A	0	-	-	0	-	-
B	1	B	0	3	C	0
C	1	C	0	3	B	-
D	2	B,C	0,0	-	-	-
E	2	C	0	4	B	1
F	3	B,C	0,1	-	-	-

Figure 7.1 Simple network and routing table for algorithm C + DSPS.

Note that the alternative path and alternative DSPS tables would be empty if the network was running algorithm A+DSPS or B+DSPS.

The pseudo-code for the route calculation procedures for the three algorithms is shown in appendix G.

7.2 Forwarding Procedures

The forwarding procedures of the three algorithms are all very similar to their uncombined counterparts of chapter 5 and 6.

The number of down stream path splits offered by a path is taken into account when selecting two equal cost routes to a destination. This credit system effectively biases packets towards nodes which offer a choice of routes further down the tree. The credit system is suspended when queues to a link exceed the maximum buffer length.

At the point of suspension of down stream path splits, the alternative paths can be considered.

7.2.1 Packet Forwarding - Algorithm A+DSPS

Algorithm A+DSPS allows the selection of an alternative path to take place once anywhere in the network. To inform other nodes in the network that an alternative path has been chosen, a one bit flag in the packet header is required.

7.2.2 Packet Forwarding - Algorithm B+DSPS

Algorithm B+DSPS permits selection from the alternative path set only at the initiating node. It therefore requires no additional packet overhead. Algorithm A+DSPS and algorithm B+DSPS can be tuned to the network environment by allowing two different forwarding procedures. The first only uses alternative paths when the outgoing queue to the shortest path reaches the maximum buffer length. The second procedure compares the length of the queue for shortest path with that of the alternative path and whichever is the shorter is selected for packet forwarding. Only the first method of packet forwarding will be discussed in this chapter.

7.2.3 Packet Forwarding - Algorithm C+DSPS

Algorithm C+DSPS is similar to algorithm B+DSPS in that alternative paths are only used at the source node. Alternative paths are used when the shortest path queue is equal to the maximum buffer length and the packet is in imminent danger of being dropped.

An example of the forwarding procedure for several queue lengths is shown in table 7.1. This table could relate to any one of the three algorithms (assuming that the flag for algorithm A+DSPS has not been set). In this example, the node running the algorithm has three neighbours, A, B and C. The adjacencies A and B offer equal cost shortest paths with down stream path splits and an alternative path is offered by adjacency C. For queue lengths below the maximum, the down stream path splits credit system is in operation. When the queue length hits the maximum value of ten, the node is permitted to utilize the alternative path. When the alternative path queue length also hits the maximum buffer size, then the selection of the next node is irrelevant since the packet will inevitably be dropped by the router.

Shortest Path				Alternative Path	Next Node
Queue lengths		DSPS		Queue length	
A	B	A	B	C	
6	7	0	2	x	B
10	10	0	2	5	C
10	10	0	2	10	x

x = dont care

max. buffer length = 10

threshold = max. buffer length

Table 7.1 Next node selection for various queue lengths.

The pseudo-code for the three forwarding procedures is shown in appendix G.

7.3 Performance of ISIS and Combination Algorithms

In the simulation results for this section the number of alternative paths that can be stored have been limited to one. In addition, the down stream path splits for the alternative paths have been neglected. This saves on a large amount of storage space for the simulation runs. If nodal storage space is not at a premium then the alternative path down stream path splits can be stored and taken into account if required. This may result in an increase in performance if many alternative paths are available.

The networks used in the simulation results are the same as those used in chapter 5 for the down stream path splits algorithm. These were chosen for the large number of DSPS that are generated for those topologies.

Simulation results are shown for the combined algorithms and the ISIS algorithm. They are not shown compared with the simulation runs of DSPS from chapter 5 nor with the alternative path algorithms of chapter 6. The reason for this is that the combined algorithms are coincident with the DSPS plot for part of the graph and the alternative path plot for another part of the graph. Percentage improvement graphs are also not shown for the same reason.

The first network simulated is the same six node network that appeared in chapter 6. This network was simulated for algorithm C+DSPS and offers a total of 4 DSPS and 22 alternative cost paths (figure 7.2). The simulation is not performed for algorithms A+DSPS and B+DSPS for this network because these algorithms have an empty alternative path routing set. The five node network of chapter 6 was not simulated in this chapter for algorithm A+DSPS or B+DSPS because no down stream path splits exist in that network.

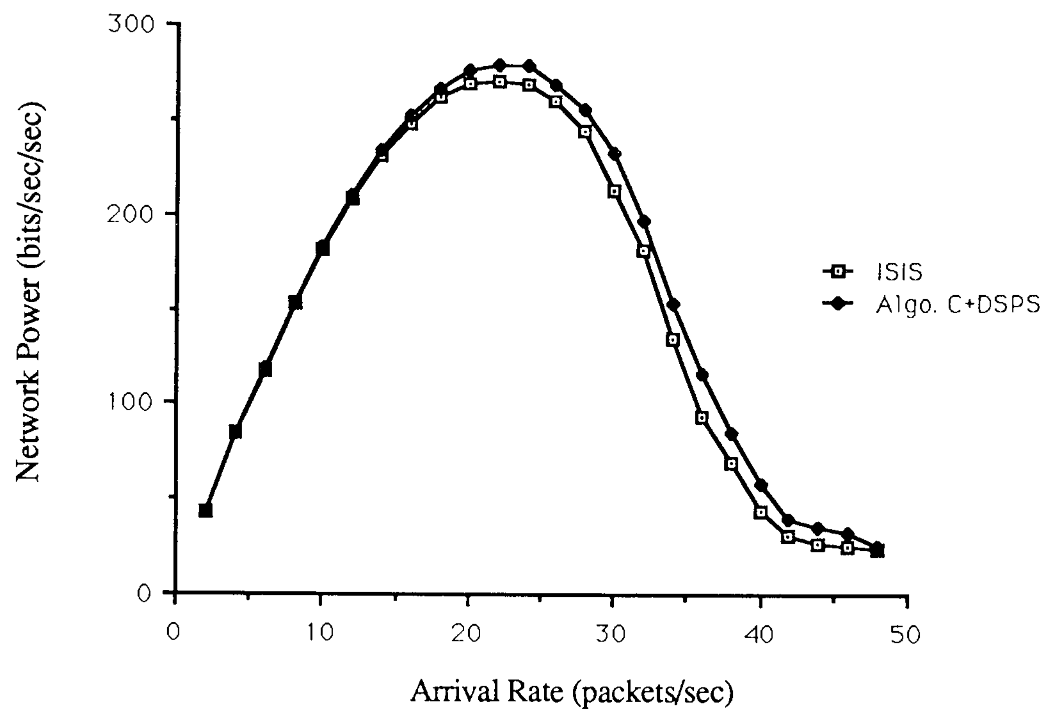


Figure 7.2 Power against arrival rate for algorithm C+DSPS and ISIS (6 nodes, 7 links)

For the larger networks with ten or more nodes, all three combination algorithms are plotted on one graph. The reason for this is that the larger networks are capable of producing a quantity of DSPS and alternative path sets which suit all three types of alternative path algorithm. For smaller topologies, it is somewhat rare for all three criteria to exist in one network. This combining of plots was not done in chapter six, because topologies were chosen to provide the maximum number of alternative paths for each type of routing algorithm. This led to topologies where all the alternative paths generated had a cost one greater than the minimum (for algorithms A and B) or topologies where alternative paths costs were unlimited (algorithm C). In this chapter this goal is relaxed.

The simulation results for the 10 node network is shown in figure 7.3 below. This topology produced 10 DSPS, 55 alternative paths with a cost one greater than the minimum and 71 unlimited cost alternative paths. The threshold value for Algorithms A+DSPS and B+DSPS are set equal to the maximum buffer length.

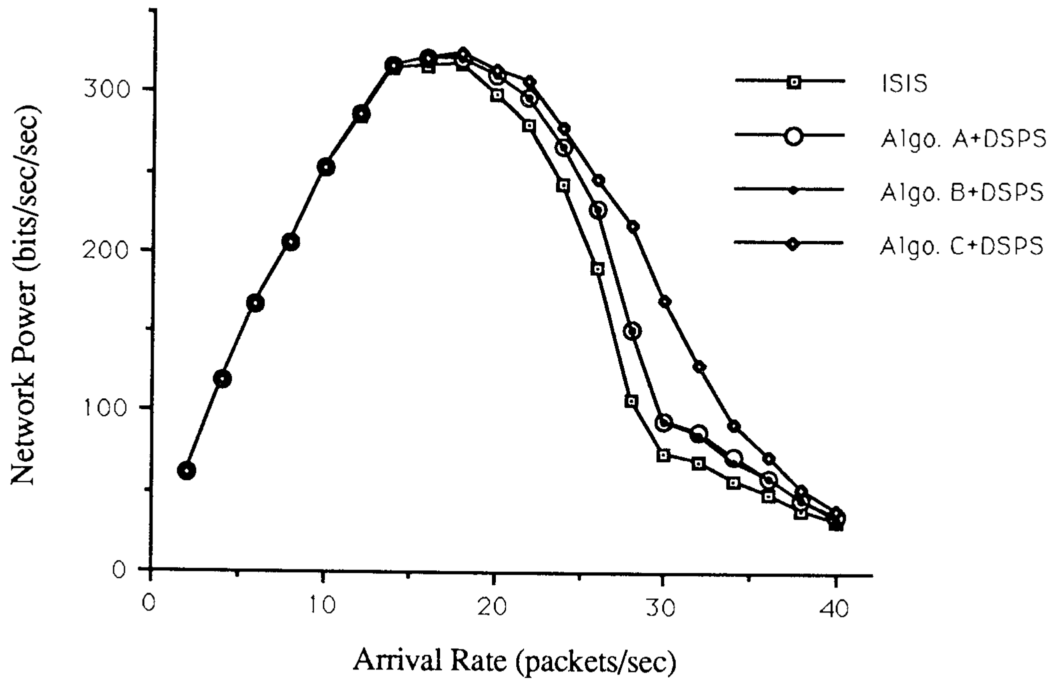


Figure 7.3 Power against arrival rate (10 nodes, 14 links)

The simulation results for the 15 node network are shown in figure 7.4 below. This topology produced 36 DSPS, 79 alternative paths with a cost one greater than the minimum and 169 unlimited cost alternative paths.

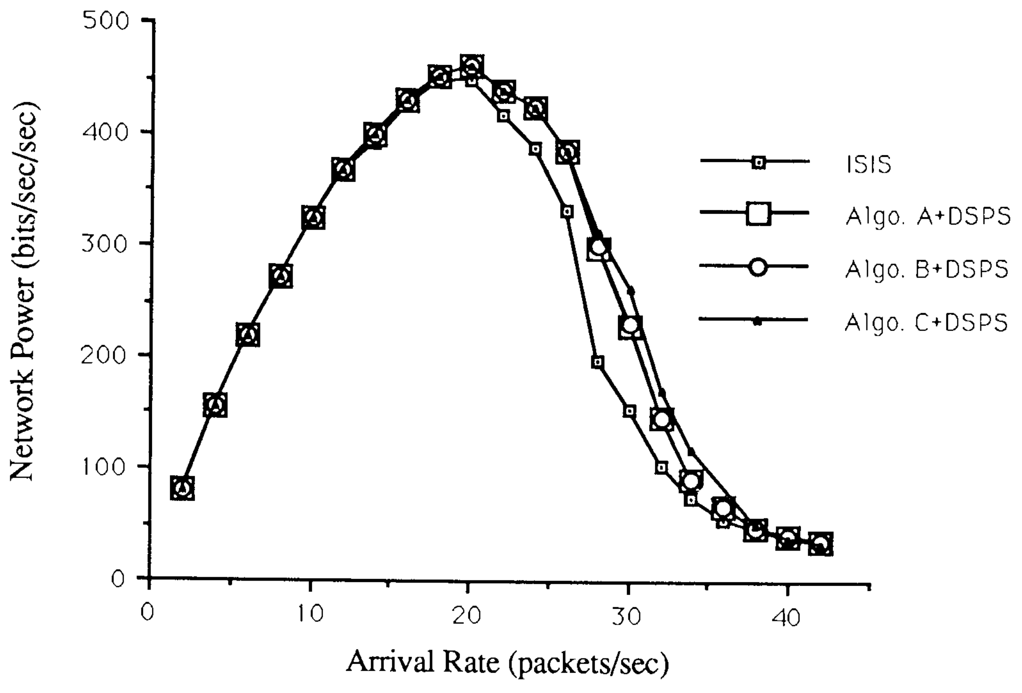


Figure 7.4 Power against arrival rate (15 nodes, 22 links)

Additional simulation results are shown in appendix H.

7.4 Discussion

Combining the alternative path algorithms with DSPS produces a very flexible algorithm. These algorithms are capable of providing a performance gain over a wide range of traffic loadings. In addition, since they can use both down stream path splits and alternative paths they are not as topology dependent as the other algorithms presented in chapters 5 and 6.

Of 450 randomly generated topologies, 94 contained no down stream path splits, 26 contained no alternative paths with a cost of one greater than the minimum and 26 contained no alternative paths whatsoever. Of these only 11 contained no DSPS or alternative paths at any cost. Therefore of the 450 topologies, only 11 would not benefit from the combination algorithms. The topologies produced for the simulations ranged from very sparsely connected (one link more than a spanning tree) to highly connected networks.

Algorithm C, which is limited to performance increases during high traffic arrival rates, benefits tremendously by adding the DSPS algorithm. Algorithms A and B also show an increased performance with the addition of down stream path splits.

7.5 Conclusion

The down stream path splits algorithm provides an increase in performance over the ISIS algorithm during medium to high traffic levels. The alternative path algorithms are able to produce an increase in performance over the standard algorithm during high to very high traffic levels.

A combination of these two independent techniques results in an algorithm which provides an increase in performance over a wide range of traffic levels. In addition, the algorithm is less topology sensitive than the previous approaches since it exploits two different types of network information.

The disadvantage of the algorithms is the extra storage space required for both the alternative path routing set and the down stream path splits. The processing burden on the router is constant throughout all the possible traffic arrival rates. At low to high load, down stream paths splits are being considered for every packet and at high to very high loads the alternative path sets come in to play. However, even this processing burden dwindles into insignificance compared to algorithms which attempt to track global packet delays.

The next chapter discusses the simulation model and provides a comment on the shape of the power graphs.

Chapter 8

Simulation Model

8.0 Introduction

In order to ascertain the benefits of various routing schemes it is necessary to resort to computer simulation. Mathematical analysis soon becomes intractable for anything but the simplest of models.

The main components of the simulation model are the transport layer running in the end systems, the network layer in the intermediate system and the physical link connecting the remote LANs together. These three layers have the most effect on the end to end packet delay. The various protocols required for the operation of these layers are included in the discussion.

This chapter gives an overview of the simulation model built to test the various routing algorithms. In addition, this chapter provides a short discussion on the power graphs that have been used throughout this thesis.

8.1 Queuing Theory

The performance of packet switched networks can be analysed using queuing theory. Queuing arises very naturally in the study of such networks; packets that arrive at a node must wait in a buffer before being transmitted on the appropriate outgoing links. This section will only give a very brief comment on the queuing theory principles relevant to this chapter. These are standard principles that can be found in any book on queuing theory [29].

The packets arrive at a node randomly, at an average rate of λ packets/second (figure 8.1). They queue up for service in the buffer and are then served, following some specified service discipline, at an average rate of μ packets/second. The server in this case is the transmission medium connecting the nodes together. A transmission link handling 1000-bit packets and transmitting at a rate of 2400 bps would be capable of transmitting at a rate of μ=2.4 packets/second.

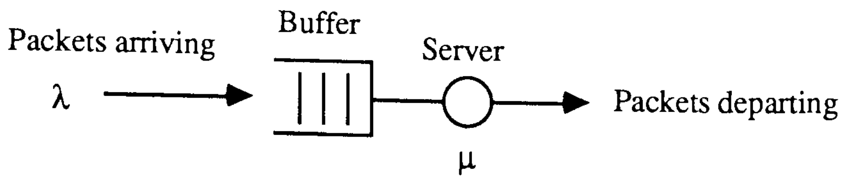


Figure 8.1 Model of simple single server queue.

One of the most commonly used statistics for customers arriving at a queue is the Poisson arrival process. This in turn, suggests the use of an exponential service time. These two disciplines form the basis of one of the simplest queues, the so-called M/M/1 queue. This is a single server queue, with Poisson arrivals, exponential server-time statistics, and first-in-first-out service. The symbol M is used to denote the Poisson process or the equivalent exponential distribution. The final integer represents the number of servers available to the queue.

There are many equations which can be derived from the basic M/M/1 queuing model. These include average packet delay through the queue, throughput and link utilization to name but a few. These equations can be applied to a network of interconnected queues as found in a packet switching network. Various theoretical values of average network delay and throughput can then be calculated. Unfortunately, the complexity and simplifying assumptions required for these calculations increase to such a level as to make mathematical analysis impractical. Therefore, for large networks it becomes necessary to resort to computer simulation.

Although queuing theory is not used to analyse the networks, some of its concepts will be drawn upon later in this chapter.

8.2 Simulation Model

The simulation model uses the SIMSCRIPT II.5 simulation language and was run on SUN workstations [88]. Even with these very fast machines, some of the simulation runs took several days to complete.

Simulation can provide a better insight into a model's performance than mathematical analysis will permit [84,85]. With its many simplifying assumptions mathematical analysis becomes far too theoretical and moves away from a practical implementation of the problem. The simulation model in this study is very flexible and permits not only the analysis of routing algorithms but also the effects of changes in topology, congestion control and transport layer protocols.

There are two different types of computer simulation that can be used, "continuous simulation" and "discrete-event simulation". Continuous simulation describes systems by sets of equations to be solved numerically. These may be algebraic or differential equations, usually with time as the independent variable. This type of simulation is commonly used for the modelling of fluid-flow and hydraulics problems.

Discrete-event simulation assumes that events occur at discrete points in time rather than continuously over time. Customers arrive and change the state of the system instantaneously. Most queuing models are analysed using discrete event simulation.

8.2.1 Model Overview

The simulation model uses a connection oriented transport layer and connection-less network and data link layers. The layers above the transport layer have no bearing on the packet delay and are therefore ignored. The simulation model allows a 'birds eye' view of the network under consideration. Packets from any source to any destination can be traced as they make their way through the various queues of the network. This ensures that the algorithms and simulation model are working correctly.

No attempt is made to model the physical layer of the FDDI LANs in terms of token rotation times, token holding times etc. FDDI is only used as a source of packets, and the delay between the host and the router on the LAN is unimportant. Router processing speed is also neglected since this is dependent on the type of router employed. Links are assumed to have no propagation delay. The number of end systems on the LAN is not modeled as such, but is represented by an increased packet arrival rate at the router.

An FDDI LAN may have several routers on a ring in a variety of different configurations. These routers may be supporting only a single link or a variety of links depending on the processing speed of the router. For this reason it is assumed that only one router is on each LAN. Figure 8.2 shows an FDDI LAN coupled to a single router serving three outgoing leased lines.

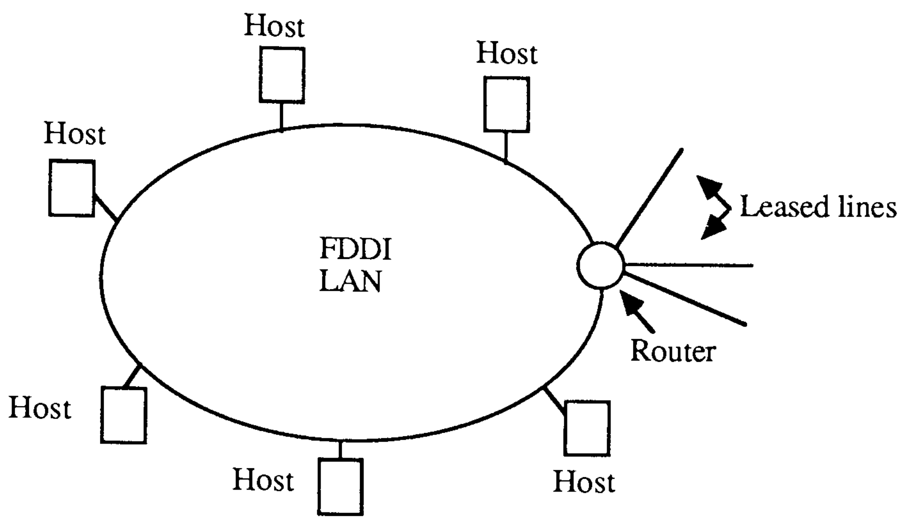


Figure 8.2 FDDI LAN connected to leased lines via single router.

The simulation model is not very different from that of a highly bureaucratic organization. This analogy assumes that a customer must pick up a series of forms from a number of different offices within a building. The comments contained within the brackets relate to interconnected LANs and will be discussed in the relevant sections of this chapter.

Upon arriving at the first office (router on source LAN) an assistant must be informed of a customers requirements. The assistant consults a list and is able to tell the customer which queue within the office to join in order to get the correct form (routing algorithm). If the desired queue is too long, then the assistant may reject the customer altogether (buffer overflow). However, it may be possible for the customer to bribe the assistant to allow him into the queue (contains acknowledgement). Having gained access to the queue, the customer must then wait to be served by the person behind the desk. The server gives him the required form and the customer then makes his way to the next office where he confronts yet another assistant. This assistant carries out the same operation as the first assistant. This procedure continues until the customer has all the forms in his possession (router on destination LAN). If at any time during the proceedings the customer is thrown out of the offices due to lengthy queues he is permitted to come back and try again the next day (packet retransmission). If the customer is dropped often enough, however, he will eventually give up completely (maximum number of retransmissions).

For the small network shown below (figure 8.3), a packet from node A to node E takes the path ABCE. The simulation model allows every step along this path to be closely monitored.

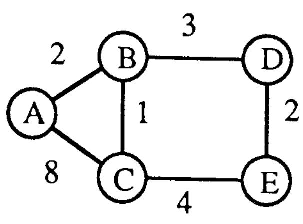


Figure 8.3 Simple five node network with associated link weights.

These steps are as follows:

1. packet arrives at node A destined for node E.
2. routing table is checked and packet placed in queue for link AB.
3. packet is serviced by link and arrives at node B.
4. routing table is checked and packet placed in shortest queue for link BC.
5. packet is serviced by link and arrives at node C.
6. routing table is checked and packet placed in queue for link CE.
7. packet is serviced by link and arrives at node E.
8. routing table is checked and no forwarding is needed.

The various queues and routing tables at each node for this network are shown in figure 8.4.

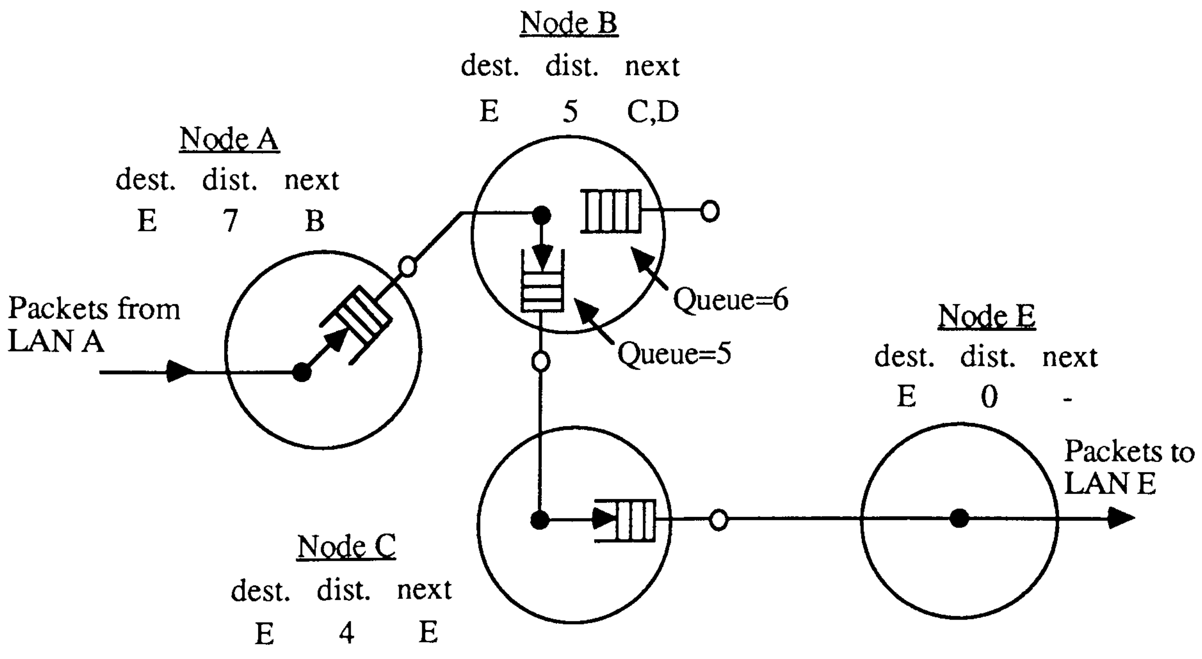


Figure 8.4 Routing tables and queues for a packet from node A to node E.

8.2.2 End Systems

The end systems (ES, or hosts) on a LAN contain all seven layers of the OSI model. However, only the lower four layers of the model are really relevant to end to end packet transmission (figure 8.5) [1].

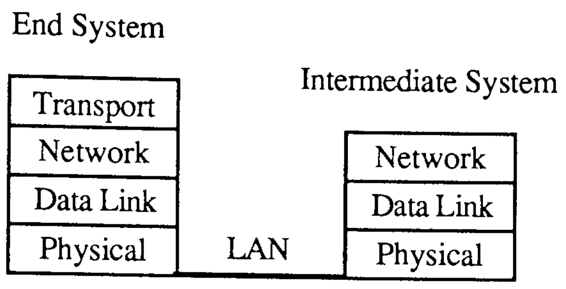


Figure 8.5 End system and intermediate system layers relevant to end-to-end transmission.

The transport layer is responsible for setting up packet transmission from one end system to another end system. It also takes care of any dropped packets and sends acknowledgements. The network layer in an end system is often quite thin. One of the basic functions of the network layer is to route packets. However, routing on a single LAN is a trivial operation since no choice in routes is available to the packet. The data link layer in an end system allows stations to gain access to the medium and ensures that packets arrive at the destination uncorrupted. Finally the physical layer actually moves the bits around the local area network.

The delay a packet experiences in the end system to intermediate system connection on the LAN is negligible compared to the delay experienced across a leased line. For example a 1000 bit packet on a 100 Mbps FDDI ring may take 10 microseconds to traverse. On a leased line running at 2.048 Mbps the delay increases to 488 microseconds. Therefore, the simulation model does not attempt to model the network layer through physical layer of the end systems on the LAN. The only layer of the end system that is of importance is the transport layer.

8.2.2.1 Traffic Characteristics

The type of traffic that is simulated is taken from an actual FDDI LAN implementation [95]. This assumes that there are only two different packet sizes; one is 512 bytes and the other is 100 bytes. The longer packets represent file transfers and other traffic intensive applications. The shorter ones can represent voice packets. Only 35 % of the traffic uses the long packets and these are assumed to require transport layer retransmissions if no acknowledgement is received by the sending station. No retransmissions are required for the remaining 65 % of the traffic.

The traffic statistics are 'bursty Poisson'; each burst consisting of up to five packets. However, this traffic pattern is only valid during times of low traffic loading. During high levels of traffic, the pattern is no longer strictly Poisson since transport layer retransmissions become more prevalent.

Using the bursty Poisson arrival process and the packet lengths discussed earlier, the average packet length in the network turns out to be 1953 bits. This is again only valid for low loading. The transport layer acknowledgements are limited to 136 bits.

The random number generator in the simulation model produces a source/destination pair each time a new packet is required. Each source/destination pair in the network is assumed to be equally likely. The rate of production of these pairs is essentially the arrival rate of packets to the system. The arrival rate is defined as the number of new packets per second arriving at every router in the network, it does not take into account packets that have been retransmitted by the transport layer.

8.2.2.2 Transport Layer (End System)

The transport layer runs in the hosts and is responsible for retransmitting dropped packets and sending acknowledgements for packets received. It is also responsible for end-to-end flow control and setting up and tearing down connections. These final two concepts are not used in the simulation model.

Use of transport layer retransmissions are very important if the simulation is to accurately reflect routing algorithm performance. This is particularly true at the higher traffic loadings. If retransmission of dropped packets is not provided, then a routing algorithm has little incentive to find better routes for the packet. The packets are simply dropped somewhere in the network and the packet is not counted in the delay statistics. This effectively reduces the average global packet delay of the network. The routing algorithm is basically being rewarded for dropping packets. It would be possible to draw a series of graphs showing how many packets were dropped, how many arrived etc. This method has been used in a variety of simulations.

A far more elegant solution is to use transport layer retransmissions in order to take into account the dropped packets without having to resort to multiple graphs for the same topology and algorithm. In a real network, transport layer retransmissions are used anyway so there is little point in ignoring them. It is assumed in the simulation model that the transport layer is willing to retransmit a dropped packet up to five times before giving up completely. This is referred to as the maximum number of retransmissions.

The transport layer protocol devised for the OSI model has five different variations, each one caters for a different type of network. The simplest is TP class 0 which basically handles the call set-up and tear down procedures and nothing more. This type of service is usually implemented in very reliable virtual circuit networks. However, it may be useful for packet voice conversations over a datagram network, in this case no retransmissions or acknowledgements are required. The most complex of the five OSI transport layers is TP class 4. This is for networks which use an unreliable datagram service, but still require a reliable transport level service [1]. Therefore, the simulation model is essentially using TP class 0 for the 100 byte voice packets and TP class 4 for the 512 byte file transfer packets.

The packet delay across a network is measured as the time a packet arrives at the starting node (Creation Time) and ends at the destination node (Finish Time). The delay is then simply Finish Time-Creation Time. If the packet is dropped and retransmitted some time later, Creation Time remains the same, it does not take on a new value. In this way the transport layer retransmissions are included in the delay average for the network.

A satisfactory value for the transport layer retransmission time can be calculated by examining a simple network. In the network shown below (figure 8.6), the maximum delay experienced by a packet is a function of the the link speeds, buffer lengths, packet sizes and the maximum number of links between two points.

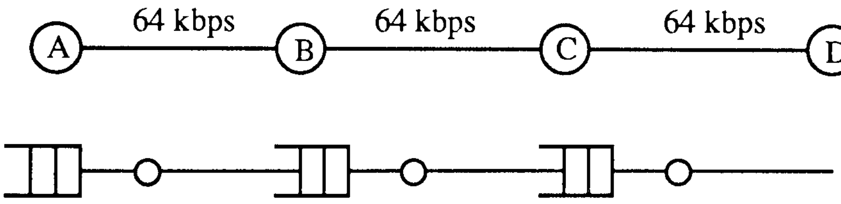


Figure 8.6 A four node network and its associated queues.

Assume that the links speeds are 64 kbps, buffers can hold 50 packets and each packet is 1000 bits long. The maximum delay that a packet could experience at each individual hop is simply (50 x 1000)/64000. This assumes that every buffer in the network is full. Therefore the total maximum delay from node A to node D is simply the number of links in the path multiplied by the hop delay (figure 8.6). This comes out to 2.3 seconds from node A to D. For a packet to go in both directions the time is 4.7 seconds. Therefore the transport layer at node A must not time out and send a retransmission in under 4.7 seconds. The equation below generates a suitable transport layer timeout value for the worst case scenario.

$$\left[\frac{\text{Maximum buffer length X Average packet size (bits)}}{\text{Link speed (bits/sec)}}\right] \text{ X } \begin{matrix}\text{Maximum number}\\\text{of hops}\end{matrix} \text{ X } 2$$

This equation does not take into account the router processing delay or the link propagation delay. It also assumes that all routers have the same buffer lengths.

Acknowledgements for packets received by a host are piggybacked onto outgoing data packets. This reduces both the number of packets in the network and the overhead required. However, the transport layer is not permitted to wait indefinitely for a data packet going in the opposite direction to piggyback the ACK on. If it does so, the sending station will obviously assume that the packet has been lost and retransmit the packet, negating any positive benefit. An acknowledgement timeout is required. If this ACK timeout is set to one second, then the transport layer timeouts must be increased by the same amount (5.7 seconds in the above example). This method can cope with the worst case packet delay. If a packet takes 2.3 seconds to arrive at the destination, then the receiving station is permitted to wait for up to one second for a packet going in the opposite direction. If no packet is forthcoming then after one second an explicit ACK must be sent. If this ACK takes 2.3 seconds to arrive at the destination, then the total round trip delay is only 5.6 seconds. This stops the transport layer from timing out and sending a retransmission.

The simulation model only uses retransmissions and acknowledgements for the long packets which make up 35 % of the traffic. Therefore, in order to send a retransmission if an ACK is not received by a node, a copy of all unacknowledged packets must be maintained by the simulation model. For a 15 node network under heavy traffic loading using 2.048 Mbps links the simulation model had to store 45000 packets.

The timeouts used in the simulation for retransmission and acknowledgements were as follows.

5, 6 nodes :	retransmission - 9 seconds
	acknowledgement - 1.8 seconds
10 nodes :	retransmission - 15 seconds
	acknowledgement - 3 seconds
15 nodes :	retransmission - 30 seconds
	acknowledgement - 6 seconds

8.2.3 Intermediate Systems

The intermediate systems, or routers, are modeled as a set of buffers used to store the incoming packets. Packets wait in these buffers until being serviced by the transmission medium. Every outgoing link has a separate queue feeding it. Figure 8.7 shows the internal workings of an intermediate system with two outgoing links.

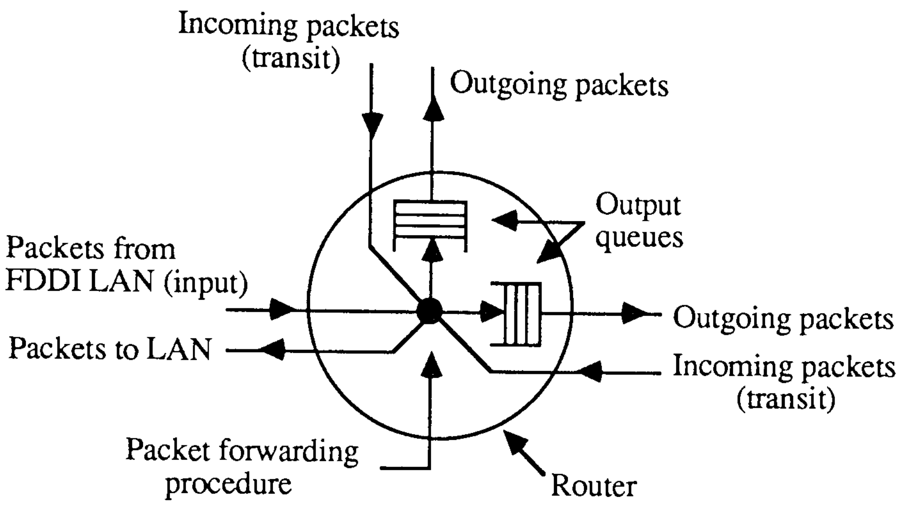

Incoming packets (transit)
Outgoing packets
Output queues
Packets from FDDI LAN (input)
Packets to LAN
Outgoing packets
Incoming packets (transit)
Packet forwarding procedure
Router

Figure 8.7 Internal workings of an intermediate system.

8.2.3.1 Network Layer (Intermediate System)

A network layer operating connectionless is responsible for both the routing algorithm and congestion control algorithm. The routing algorithms require that a table is maintained for each node in the network. For the simulation model this is an N x N x 12 matrix, where N is the number of nodes in the network. This matrix is consulted each time a packet arrives at the node in order to determine which queue the packet should be placed in; the selection of which is dependent on the routing algorithm under test.

The congestion control algorithm used in the simulation is termed input buffer limiting. In this algorithm, transit packets through the router are given access to more of the available buffer space than are input packets; the theory being that transit packets have used up more of the networks resources than input packets and to drop them would more wasteful than dropping input packets.

Before being dropped by a full buffer all packets are checked to see if they contain an acknowledgement. If the packet does contain an ACK additional buffer spaces can be allocated to the packet. Some congestion control schemes will still drop the data portion of the packet and retain the acknowledgement. Other schemes reward the data packet for containing an ACK and allow the entire packet to be stored and forwarded. The latter congestion control scheme is used in this simulation model [29,86,87]. The buffer sizes for the simulation runs were as follows. Note that these buffer lengths are defined for each outgoing link of the node.

5,6 and 10 node networks:	Input packets 20 (+2 if packet contains an ACK)
	Transit packets 22 (+2 if packet contains an ACK)
15 node networks:	Input packets 30 (+3 if packet contains an ACK)
	Transit packets 40 (+3 if packet contains an ACK)

These are extremely small values for buffer lengths. In an actual router, the buffer may be able to hold as many as 1000 packets. However, increasing the buffer sizes means that the transport layer retransmission time must also be increased. This in turn requires that more packets must be stored for possible retransmission by the transport layer. It was found that the computers running the simulation programs were unable to store the vast number of packets required if buffer lengths were increased.

The congestion control algorithm was not tuned to the various networks topologies under consideration. If it had been, the power graphs would probably achieve a far greater performance level. The effects of changing the input packet/transit packet ratios are discussed in Schwartz [29].

8.2.3.2 Data Link Layer (Intermediate System)

The data link layer has a much more modest goal than that of the network layer. It's task is to ensure delivery of a frame from one node to its adjoining node. It is not capable of ensuring uncorrupted transmission across an entire network. The OSI has standardized three different types of logical link control (LLC) sublayer. This sublayer can provide either connectionless (called type 1) or connection-oriented (type 2) service to the network layer. It also provides acknowledged connection-less service (type 3) [1].

When connectionless (type 1) service is used, the LLC sublayer accepts a packet from the network layer and uses a best efforts attempt to send it to the destination.There is no acknowledgement and no guarantee of delivery. This level of service is very basic and assumes higher layers will deal with acknowledgements and retransmitting dropped packets. It is ideally suited for use with connection-oriented transport layer and a connectionless network layer. LLC type 1 is used in the simulation model.

8.2.3.3 Physical Layer (Intermediate System)

The intermediate system actually has two physical layer protocols. One which connects the router to the LAN the other connects the router to the leased line. This discussion will only deal with the latter.

The physical layer takes into account the link speeds and the topology of the network. The link speed has been set to 64 kbps which is the speed offered by British Telecoms Kilostream leased line service. The higher speed links of 2.048 Mbps were not used for the simple reason that more packets would be circulating within the network. This would increase the amount of time taken for the simulations to finish running. Even with the fairly low speed links many of the simulations took several days to complete.

All the topologies used in the simulation were built up from a spanning tree. After running the simulation for a set time an additional link was randomly added to the tree. The new link was not attached to any node that already had more than five links emanating from it. This ensures that links are evenly spread throughout the network. Routing tables were then recalculated and the simulation continued. This occurred until the number of links was equal to twice the number of nodes in the network. The topologies generated by this algorithm ranged from the very sparsely connected to well connected graphs.

8.3 Model Verification

The question inevitably arises of how to ensure that the model is correct. The most complex component of the simulation model is the concept of time and maintaining customers in the queues. This is handled by the SIMSCRIPT II.5 simulation language. The remainder of the simulation is effectively divorced from these two concepts.

The routing tables are produced by running a shortest path algorithm and it is a simple matter to ensure that they are correct for the topology under consideration. The concept of retransmissions and acknowledgements are easily monitored by tracing packets as they arrive and depart from any node in the network. These algorithms are either correct or incorrect, there is no middle ground between the two. The actual operation of the network is very straightforward and cannot be faulted. It is, therefore, highly unlikely that the operation of the model is incorrect.

What is open to scrutiny, however, are the parameters that effect the performance of the model. These parameters include the traffic characteristics, the retransmission timeout values, ACK timeout values, packet lengths, buffer sizes and link speeds to name but a few. For a real network these parameters take on a wide range of different values and it is simply not possible, or worthwhile, to simulate all the possible combinations.

It must be stressed that simulation is not necessary to prove the correctness of the algorithms. The purpose is simply to prove that some performance gain is possible when the algorithms are implemented. The simple concept of sending packets towards nodes where they are less likely to be dropped, as in DSPS, is a common sense idea and does not require rigorous mathematical proofs and/or simulation to prove it. The same is true for the alternative path algorithms. Simulation is only used to measure the performance gain offered by the algorithms, not prove them correct. Therefore, even if the simulation model is completely wrong, it does not necessarily mean that the algorithms are incorrect.

8.4 Power Graphs and the Simulation Model

The power graphs that have appeared in chapters 5,6 and 7 can now be discussed in relation to the simulation model.

Power is made up of two components, network throughput and average network delay. Power is a very good measure of a network performance since these two values are often in contention [78]. Each one of these two components will be examined individually for a single queue.

The simulated networks are simply a large number of interconnected single server queues. Recall that for every link emanating from a router, a single queue is provided for the incoming packets. Therefore, it is possible to examine the basic M/M/1 queue model to gain some understanding as to why power graphs take on the shape that they do.

In a single M/M/1 finite queue, the normalized throughput is given by the equation [29]:

$$\text{Normalized throughput} = \rho(1-\rho^N)/(1-\rho^{N+1})$$

where ρ is the normalized load and N is the buffer length

A sketch of normalized throughput against normalized load is shown in figure 8.8. The throughput of the queue eventually levels off and approaches the throughput capacity as ρ increases. This occurs beyond the point where the normalized load, ρ, equals one. At the point ρ equals one the normalized throughput is equal to $N/(N+1)$.

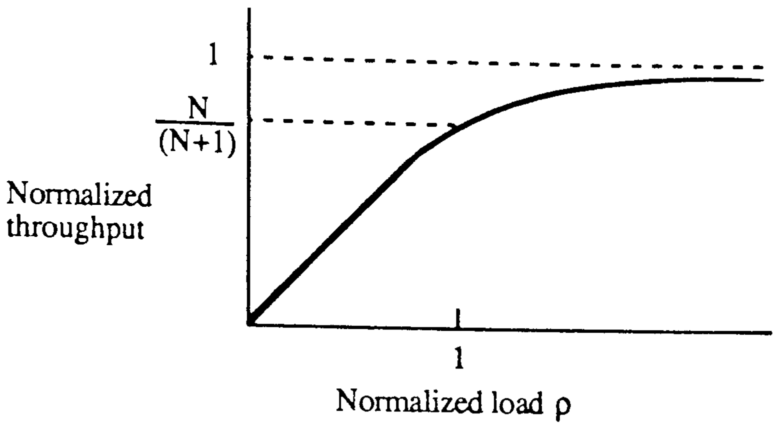


Figure 8.8 Throughput vs normalized load for M/M/1 queue.

In an M/M/1 queue, the average delay experienced by customers waiting for attention by the server rises rapidly as the arrival rate increases. For an infinite queue the average delay, E(T), that a customer must wait is given by the following equation [29]:

$$E(T) = 1/\mu/(1-\rho)$$

This equation is shown plotted in figure 8.9.

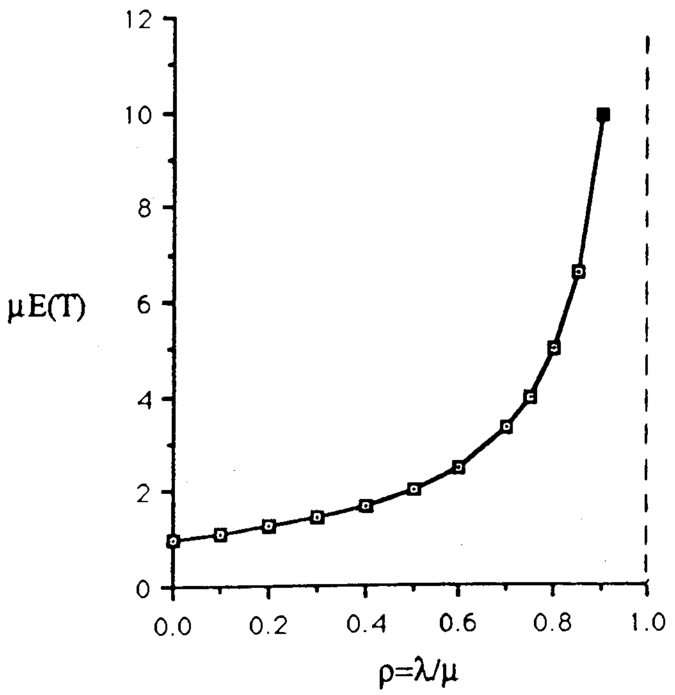


Figure 8.9 Average delay against arrival rate M/M/1 queue

In a real network, buffers do not have infinite length so the delay experienced by a packet waiting for service will always be finite. It is possible to calculate the maximum delay that a packet will experience in a network by multiplying the transport layer retransmission timeout by the maximum number of retransmissions. If the retransmission interval is 10 seconds and the transport layer tries 5 times before giving up, the maximum delay that a packet could possibly suffer is simply 50 seconds.

Therefore the average delay against arrival rate graph for a network will not approach infinity with increasing arrival rate but will level off to some value as shown in the sketch of figure 8.10.

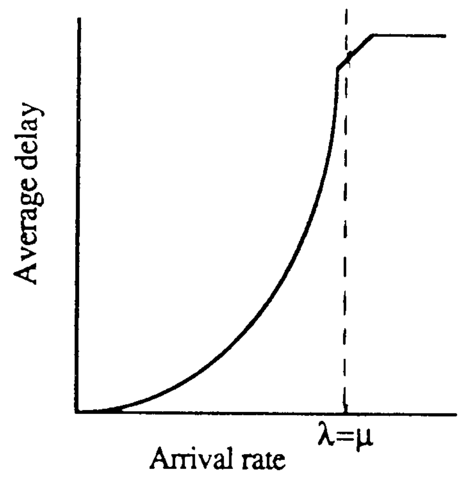


Figure 8.10 Sketch of average delay against arrival rate for a real network.

The power graph is simply a combination of the throughput and delay graphs. Both of which reach a steady state value with an increasing packet arrival rate. It follows from this that the power graph will also reach a steady state value. The graph should rise, hit some maximum and then roll off to this steady state value. Power graphs will always take this shape regardless of the routing algorithm used.

The simulation model uses finite buffer lengths and does not use exponential packet sizes as in the above theory. However, the simulation results still produce graphs which are not dissimilar to those discussed previously. A simulation was performed on the four node network shown below (figure 8.11) and graphs of average network delay and throughput were plotted. The ISIS routing algorithm was used in this network.

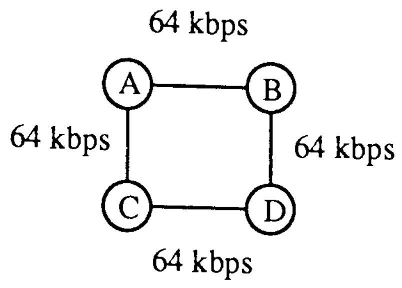


Figure 8.11 Simple four node network.

The delay and throughput graphs for this network are shown in figure 8.12 below.

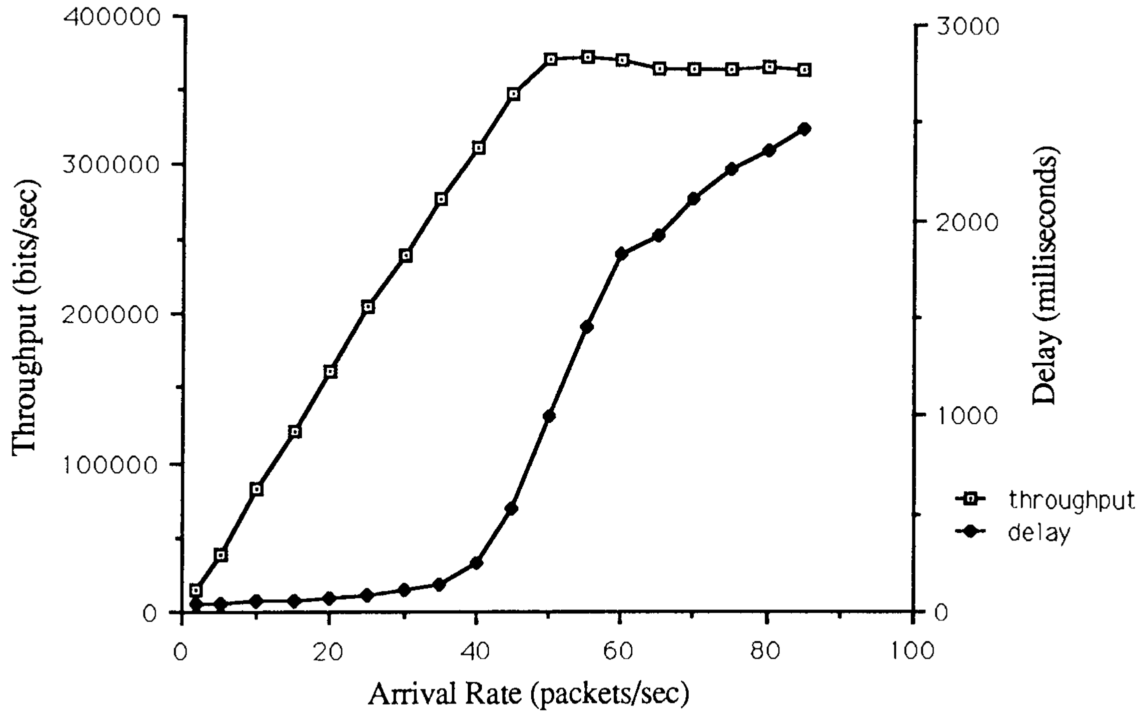


Figure 8.12 Throughput and delay curves for four node network.

If it is assumed that each link is capable of providing the full 64 kbps then the theoretical maximum carrying capacity of this network is 512 kbps (ie 8 links each at 64 kbps). The simulation produces a maximum throughput of 390 kbps which represents an efficiency of 76%. This efficiency can be increased if the transport layer and the congestion control protocols are tuned for this particular network.

The delay graph does not flatten out as it does in the theoretical graph because the number of retransmissions is limited to only 35% of the traffic. If 100 % of the traffic required acknowledgements then the graph would flatten out to a steady state value.

Dividing throughput by delay for the four node network results in the characteristic power graphs that have appeared in chapters 5,6 and 7 (figure 8.13)

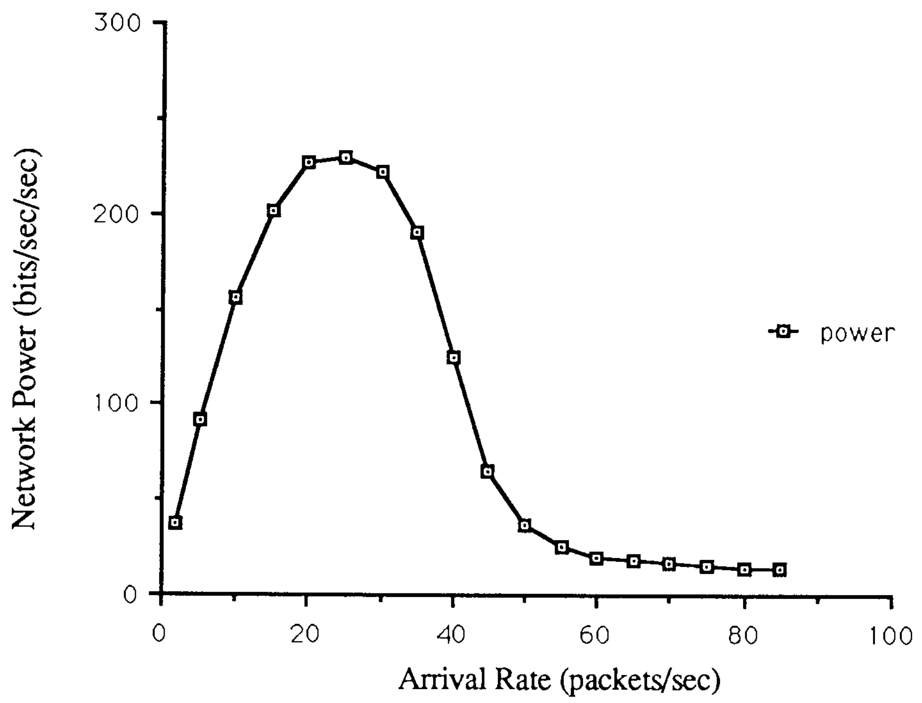


Figure 8.13 Resulting power graph for four node netwo.·k.

8.5 Simulation Variables

The simulation model can be used as a tool to test not only routing algorithms, but a variety of other protocols and topologies.

Varying the transport layer retransmission policy results in changes in the power graphs under high traffic loading. Three values of timeout were used, 6, 10 and 15 seconds. This can be seen in the graph shown below (figure 8.14). These results were taken from the four node network of figure 8.11.

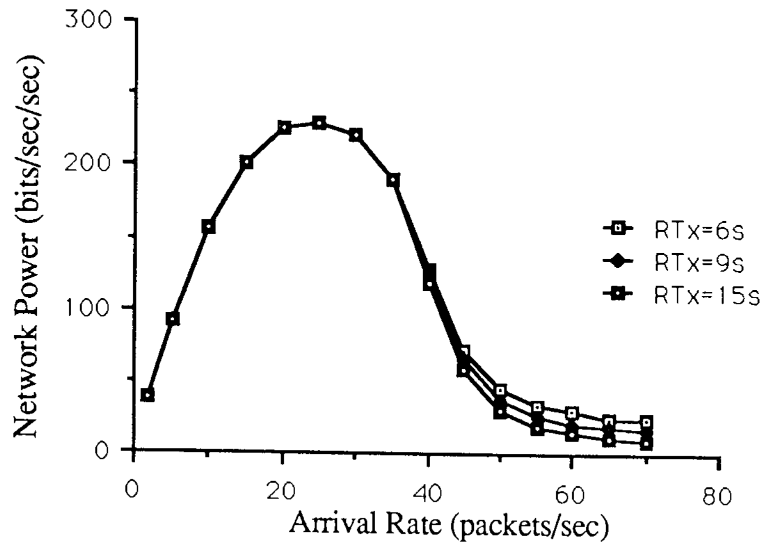


Figure 8.14 Effects of transport layer timeouts on power graphs.

Adjusting a routers buffer length also has an effect on the power graph (figure 8.15). In the same network with a retransmission timeout of 9 seconds and buffer lengths of 5,10 and 20 packets the following results were obtained.

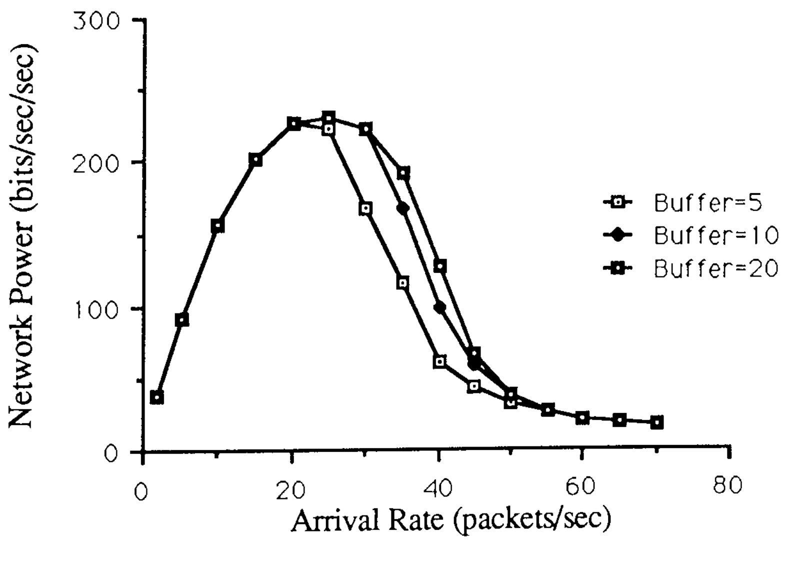


Figure 8.15 Effects of buffer lengths on power graphs.

Note that using a buffer length of 5 packets results in a very fast roll off of the power graph. Recall that this occurred when the 15 node networks in chapters 5, 6 and 7 were simulated. This suggests that the buffer lengths for those networks were far too small for the size of network being modeled, nonetheless the simulation results for the algorithms are still valid. It was not possible to simulate increased buffer sizes for the larger networks. If larger buffer sizes are used then the transport layer retransmission time must be increased accordingly. If this timeout is increased then the number of packets which must be retained by the simulation program also increases. This puts excessive strain on the computers that are running the simulations. As an example, a simulation run of a 30 node network executed on the SUN server took so much processing time and memory that no other users could gain access to the system. Following this, simulation of large networks was abandoned

In addition to adjusting the transport layer retransmission timeout and the buffer sizes, there are many other values in the simulation model which can be varied. Some of these include:

- congestion control algorithm input/transit packet ratio
- packet sizes
- traffic characteristics (includes source/destination pairs, burstiness etc)
- link speeds
- acknowledgement timeout
- topology

This work has fixed many of the above values in order to limit the number of different combinations and permutations that are produced.

8.6 Conclusion

Due to the complexity associated with mathematical analysis it is often necessary to resort to computer simulation for the modelling of packet networks. The simulation model assumes that events occur at discrete points in time. This is a common method used for modelling queuing networks.

The model gives a 'bird's eye' view of the network and allows all packets from any source to any destination to be monitored as they make their way through the network. All forwarding decisions made by routing algorithms can also checked to ensure that the model is working correctly.

The simulation model is fairly comprehensive and incorporates the transport layer of the end systems and the network through physical layers of the intermediate systems. Simplifying assumptions about router processing speed, intra-LAN routing protocols and link propagation delay have been made in order to reduce the complexity of the model. These assumptions are all implementation specific and can be ignored without the loss of generality.

The power graphs produced by the simulation model are not dissimilar to those produced in theory. The graphs have a characteristic shape which is common to all routing algorithms. All power graphs, regardless of the routing algorithm implementation, will eventually converge to a common value with increasing traffic arrival rates.

The simulation program involves a large number of different variables and protocols which can all be modified for the network under consideration. In this way the model can be used as a testbed to allow different topologies to be randomly generated or user defined, buffers can be lengthened and shortened to test various congestion control schemes, packet lengths, arrival rates and link speeds can all be adjusted, routing algorithms can be changed, retransmission and acknowledgment timeouts varied. Due to the large number of variations, the routing algorithms were tested with many of the above variables fixed for the duration of the simulation.

Chapter 9

Conclusion

9.0 Introduction

The Fibre Distributed Data Interface is a new high speed local area network capable of supporting a large number of users over a 100 km distance. Traffic on this LAN is expected to be as diverse as file transfers, packet voice and video. FDDI is being embraced as the next generation of local area network and is the only fibre optic LAN going through the standardization process.

As FDDI LANs proliferate, the desire to interconnect remote FDDI installations arises. These LANs can be connected together in a variety of different ways. One of the most cost effective technologies which is widely available today is the leased line. The most commonly used line speeds are the 64 kbps Kilostream and the 2.048 Mbps Megastream links available from British Telecom. The number and speed of these links used for the internetwork depends on the anticipated quantity of inter-LAN traffic.

In order to connect an FDDI LAN to a leased line a gateway device is required. These devices can operate at any layer of the OSI model. Two of the most common devices for connecting LANs to leased lines are the layer 2 bridge and the router which operates at layer 3. There is much discussion over which provides a better service to the LAN. Bridges offer higher throughput than the router but there are topology restrictions. Routers, on the other hand, can cope with any network regardless of size and topology. It is widely felt that routers are better for large networks with many links, whereas bridges are more suitable for connecting LANs around a single campus.

A router, as its name suggests, requires a routing algorithm of some description in order to send packets from one LAN to another. There are many different routing algorithms both in the literature and in actual network operation. The selection of a suitable routing algorithm is highly dependent on the network environment that the algorithm is expected to operate in.

An FDDI LAN will provide bursty and potentially large amounts of inter-LAN traffic. A rule of thumb for the ratio of intra-LAN traffic to inter-LAN traffic is 90/10. If the links connecting the LANs together are chosen with this figure in mind the potential for generating a very large burst of traffic is extremely high.

It is undesirable for a routing algorithm to adapt to traffic changes on a global basis. This is particularly true if the traffic is expected to be bursty or at times very high. The ARPANET's experience with such algorithms has shown that they work well under lightly loaded conditions but tend to oscillate when the traffic loading gets high. Algorithms therefore should adapt to traffic fluctuations on a local basis only.

One of the most common routing algorithms in use today is the shortest path algorithm developed by Bellman-Ford. This is an extremely simply distance-vector algorithm which only requires next node information in order to make routing decisions. It also requires a minimal amount of storage space. Many computer networks which require a routing algorithm have been based on the Bellman-Ford algorithm. However, its simplicity is also its downfall. When links and nodes fail in the network, the routing tables take a long time to converge. During this interval, routing table loops may form and cause packets to return to a node that they have originally visited. This is very unsatisfactory for LANs where the traffic loading may be very high. Many different techniques have been proposed which attempt to curtail this routing table looping problem. However, they begin to detract from the main advantage of the algorithm, namely simplicity. Therefore the Bellman-Ford algorithm is unacceptable for an interconnected FDDI network.

A shortest path algorithm developed by Dijkstra requires that knowledge about the entire network is known before the calculation proceeds. This link-state algorithm has been used successfully in the distributed ARPANET routing scheme and converges much faster than the Bellman-Ford type algorithms. The ARPANET's SPF algorithm still suffers from routing table loops but to a much lesser degree than the Bellman-Ford. The ARPANET experience has shown that the amount of packet looping associated with the SPF algorithm is almost negligible.

The OSI is in the process of standardizing a version of the SPF algorithm which is detailed in the ISIS standards document. This is a well thought out algorithm which adapts to traffic fluctuations on a local basis and does not require vast amounts of processing. It is likely that this algorithm will be implemented by many computer manufacturers due to its simplicity.

The new algorithms in this thesis are all based on the ISIS standard. They use exactly the same link state information as the ISIS algorithm and adapt to traffic fluctuations based on local information only. The algorithms only require modification to the route calculation and packet forwarding procedures of the ISIS algorithm.

9.1 Review of the New Routing Algorithms

The ISIS algorithm discards some very valuable information during the route calculation procedure. This information is in the form of duplicate adjacencies which indicate that further down stream from the node doing the calculation, a choice of equal cost routes is permitted. This information is exploited by the Down Stream Path Splits (DSPS) routing algorithm. The number of down stream path splits for a particular destination are used as credits against the outgoing queue length. In this way, packets are biased towards nodes which offer a choice of equal cost paths to the destination. During times of very high traffic loading, the DSPS credit system is effectively suspended.

Multiple runs of the ISIS algorithm can be used to generate the k-shortest paths in a network by masking and unmasking a node's adjacencies. Algorithms A, B and C use this method to generate both the shortest paths and a set of alternative paths which can be used during times of network congestion. Algorithms A and B only store alternative paths if they offer a cost one greater than the shortest path. Algorithm C is permitted to store alternative paths of any cost. Algorithm A can select an alternative path at any node in the network. Once selection is made from the alternative path group a 1-bit flag is set in the packet header. This informs subsequent nodes that selection must be made from the shortest path set. Algorithms B and C are only permitted to select an alternative path at the initial node. The alternative paths are used when the queue lengths to the primary path exceed some threshold defined in the packet forwarding procedure.

The DSPS algorithm is coupled with algorithms A,B and C to form the combination algorithms. These are termed algorithm A+DSPS, B+DSPS and C+DSPS respectively. These algorithms use the down stream path splits credit system during times of medium to high traffic levels and the alternative path set during times of congestion.

9.2 Comparison of the New Routing Algorithms

All seven of the algorithms presented in this thesis are capable of providing a performance increase over the standard ISIS algorithm. The amount of gain, as has been stressed repeatedly, is highly dependent on the network environment that the algorithm is expected to operate in. For this reason it is not possible to compare the algorithms to determine which one of them is 'best', but some generalizations can be made.

The DSPS algorithm is extremely simple and requires only minor modifications to the route calculation and packet forwarding procedures. The complexity of the algorithm is no more than that of the ISIS algorithm, namely O(n2). The algorithm does require a tiny amount of additional processing for each packet in order to use the DSPS credit system. In addition, up to one third extra router memory is required in order to store the DSPS values. The algorithm is suitable for all but the very smallest of networks where the number of DSPS is limited. In networks which do possess DSPS, simulation results have shown gains as high as 50% over the basic ISIS algorithm can be achieved. For such a simple algorithm this is quite astounding.

Even if the DSPS algorithm is not implemented in its entirety, the concept should still be used in deciding which adjacencies to keep if the ISIS router has a limit on the number that it may store (MaxPathSplits). Those adjacencies which offer the largest number of down stream path splits should be retained over those which offer very few.

The three alternative path algorithms all require one third more router storage space than the ISIS algorithm. Algorithm A is the most flexible in that it can adapt to congestion in the network more easily than the other two. The one bit of overhead required for this algorithm is trivial in terms of introducing packet delay into the network. However, what is not trivial is the requirement that all other routers in the network understand the significance of this flag. Algorithms A and B can both set low threshold values which allow the alternative paths to be used with a greater frequency. This effectively increases the performance of these algorithms between the medium to high traffic ranges. Algorithm C is similar to algorithm B, but should not use low threshold values. The reason for this is the potentially large cost difference between the shortest paths and the alternative paths. Therefore this algorithm is restricted to providing performance gains during times of heavy traffic loading only.

The combination algorithms all require up to 2/3 more storage space than the ISIS algorithm. These algorithms provide a performance gain over a wide range of traffic loadings. They are also less susceptible to the effects of topology since they exploit two types of topology information. Algorithm A+DSPS is the most flexible and powerful of the seven algorithms developed, but unfortunately it still requires the 1-bit of overhead. The amount of gain provided by algorithm B+DSPS is no better than algorithm B with a low threshold setting. The latter requires less storage space and is less complex. Algorithm C+DSPS outperforms the basic algorithm C since it uses the DSPS credit system during medium to high load.

In general, for networks with many equal cost links, a tuned version of algorithm B appears to be the most promising. For networks with many different cost links, algorithm C+DSPS should be used. Both of these algorithms can operate independently from the other routers in the network. In addition, they can both cope with a wide range of different traffic arrival rates. Table 9.1 summarizes the seven algorithms.

One of the main attractions of algorithms that provide performance gains over another algorithm is the efficient use of links. Not only is it possible to approach some theoretical value for throughput and delay, but the number of links required for a particular implementation may be fewer. For example a network running ISIS may require 10 links in order to guarantee a certain performance value, the same performance may be achieved using one of the new algorithms coupled with say nine or even eight links. This is not only more efficient, but it also saves large amounts of money on leased lines.

Although leased lines are used in the simulation study, this does not mean that the algorithms are suitable for this type of installation only. The algorithms will work in any type of network that the ISIS protocol can be used in. This includes X.25 networks, point-to-point links, broadcast networks etc. The algorithms can be used in virtual circuit networks by only permitting the call set-up packets to take the DSPS routes or alternative paths. Packets in a virtual circuit network are then constrained to follow this route. This differs from datagram network in which all packets are routed independently.

Algorithm	Traffic range	Add.mem. required	Additional requirements
DSPS	med. -> high	1/3	Topology must have more than six nodes but less than fully connected.
Algorithm A	high -> v.high	1/3	Other routers in network must recognise 1-bit flag.
Algorithm B	high -> v.high	1/3	None.
Algorithm C	high -> v.high	1/3	None.
Algorithm A+DSPS	med. -> v.high	2/3	Other routers in network must recognise 1-bit flag.
Algorithm B+DSPS	med. -> v.high	2/3	None.
Algorithm C+DSPS	med. -> v.high	2/3	None.

Table 9.1 Comparison of the seven new routing algorithms

9.3 Conclusion

The objective of this work has been to develop practical routing algorithms for interconnected FDDI LANs. The new algorithms are all based on the ISIS routing algorithm which is being developed by the ISO standards body. This standard is being implemented by many manufacturers of network routing equipment.

The new algorithms require modification to the ISIS packet forwarding and the route computation procedures. All other procedures remain unaltered.They are designed for implementation within an ISIS internetwork with the minimum amount of modification. They send and receive exactly the same link state information as a standard ISIS router and adapt to traffic fluctuations on a local basis only. All of the algorithms are fairly simple and do not require a large amount of router processing in order to operate. Some of the new algorithms offer a performance increase as high as 350 percent over the ISIS algorithm.

The algorithms developed are simple, highly practical and can provide an increased performance over the ISIS standard with a minimum amount of additional processing.

References

1. Tanenbaum, A. S.: 'Computer Networks' (Prentice-Hall, Englewood Cliffs, NJ, 1989)

2. Stallings, W.: 'Local Networks' (Macmillan, New York, NY 1987)

3. Zimmermann, H.: 'OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection', *IEEE Trans. on Comms.*, 1980, Vol. COM-28, No. 4, pp 425-432

4. Ross, R.E.: 'An Overview of FDDI: The Fibre Distributed Data Interface', *IEEE J. on Sel. Areas in Comms.*, 1989, Vol. COM-7, No. 7, pp. 1043-1051

5. Joshi, S.P.: 'High-Performance Networks: A Focus on the Fiber Distributed Data Interface (FDDI) Standard' *IEEE Micro*, 1986, pp. 8-14

6. Ross, F.E., Hamstra, J.R. and Fink, R.L.: 'FDDI - A LAN Among MANs' *Computer Comms. Rev.*, 1990, pp. 16-31

7. Ross, F.E.: 'Rings are 'Round for Good!' *IEEE Network,* 1987, Vol. 1, No. 1, pp. 31-38

8. Schneidewind, N.F. 'Interconnecting Local Networks to Long-Distance Networks', *IEEE Computer*, September 1983, pp. 15-24

9. Lang, L.J., and Watson, J.: 'Connecting Remote FDDI Installations with Single-Mode Fiber, Dedicated Links or SMDS', *Computer Comms. Rev.*, 1990, pp. 72-82

10. Materna, B., Vaughan, J.N. and Britney, C.W.: 'Evolution From LAN and MAN Access Networks Towards the Integrated ATM Network', *Proc. IEEE GLOBECOM*, 1989, pp. 1455-1461

11. Clapp, G.H.: 'LAN Interconnection Across SMDS' *IEEE Network*, September 1991, pp. 25-32

12. Hemrick, C.F., and Lang, L.J.: 'Introduction to Switched Multi-Megabit Data Service (SMDS), an Early Broadband Service', *Proc. ISS*, 1990, pp. 1-8

13. Cavanagh, J.P.: 'Applying the Frame Relay Interface to Private Networks', *IEEE Comms.*, March 1992, pp. 48-64

14. Lamont, J. and Hui, M.H.: 'Some Experience with LAN Interconnection via Frame Relaying', *IEEE Network*, September 1989, pp. 21-24

15. Seiffert, W.M.: 'Bridges and Routers' *IEEE Network*, January 1988, pp. 57-64

16. Salwen, H., Boule, R. and Chiappa, J.N.: 'Examination of the Applicability of the Router and Bridging Techniques', *IEEE Network*, January 1988, pp. 77-80

17. Sunshine, C.A.: 'Network Interconnection and Gateways' *IEEE J. on Sel. Areas in Comms.*, 1990, Vol. COM-8, No. 1, pp. 4-11

18. Bosack, L. and Hedrick, C.: 'Observations, Comparisons and Choosing - Problems in Large LANs', *IEEE Network*, January 1988, pp. 49-56

19. Perlman, R., Harvey, A. and Varghese, G.: 'Choosing the Appropriate ISO Layer for LAN Interconnection', *IEEE Network*, January 1988, pp. 81-85

20. Benhamou, E.: 'Integrating Bridges and Routers in a Large Internetwork', *IEEE Network*, January 1988, pp. 65-71

21. Hart, J.: 'Extending the IEEE 802.1 MAC Bridge Standard to Remote Bridges' *IEEE Network*, January 1988, pp. 10-15

22. Backes, F.: 'Transparent Bridges for Interconnection of IEEE 802 LANs', *IEEE Network*, January 1988, pp. 5-9

23. Hamner, M.C. and Samsen, G.R.: 'Source Routing Bridge Implementation', *IEEE Network*, January 1988, pp. 33-36

24. Dixon, R.C. and Pitt, D.A.: 'Addressing, Bridging and Source Routing', *IEEE Network*, January 1988, pp. 25- 31

25. Soha, M. and Perlman, R.: 'Comparison of Two LAN Bridge Approaches', *IEEE Network*, January 1988, pp. 37-43

26. Zhang, L.: 'Comparison of Two Bridge Routing Approaches', *IEEE Network*, January 1988, pp. 44-48

27. Fink, R.L.: 'FDDI as a Backbone for Large Campus Ethernet Networks', *Proc. EFOC/LAN '87*, October 1987, pp. 65-68

28. Hammond, J.L. and O'Reilly, P.J.P.: 'Performance Analysis of Local Computer Networks' (Addison-Wesley, Reading, MA, 1988)

29. Schwartz, M.: 'Telecommunication Networks - Protocols, Modeling and Analysis', (Addison-Wesley, Reading, MA, 1987)

30. Bertsekas, D. and Gallager, R.: 'Data Networks', (Prentice-Hall, Englewood Cliffs, NJ, 1987)

31. Bell, P.R. and Jabbour, K.: 'Review of Point-To-Point Network Routing Algorithms', *IEEE Comms.*, January 1986, Vol. 24, No. 1, pp. 34-38

32. Schwartz, M. and Stern, T.E.: ' Routing Techniques Used in Computer Communication Networks', *IEEE Trans. on Comms.*, 1980, Vol. COM-28, No. 4, pp. 539-551

33. Sproule, D.E., and Mellor, F.: 'Routing, Flow and Congestion Control in the Datapac Network', *IEEE Trans. on Comms*, 1981, Vol. COM-29, No. 4, pp. 387-391

34. Tymes, L.W.: 'Routing and Flow Control in TYMNET', *IEEE Trans. on Comms.*, 1981, Vol. COM-29, No. 4, pp. 392-397

35. Baran, P.: 'On Distributed Communication Networks' *IEEE Trans. on Comm.* Sys., 1964, pp. 1-9

36. Lai, W.S.: 'Packet Forwarding', *IEEE Comms.*, 1988, Vol. 26, No. 7, pp. 8-17

37. McQuillan, J.M., Falk, G. and Richer, I.: 'A Review of the Development and Performance of the ARPANET Routing Algorithm', *IEEE Trans. on Comms.*, 1978, Vol. COM-26, No. 12, pp. 1802-1810

38. Chlamtac, I. and Franta, W.R.: 'Rationale, Directions and Issues Surrounding High Speed Networks', *Proc. IEEE*, 1990, Vol. 78, No. 1, pp. 94-120

39. Maxemchuk, N.F. and El Zarki, M.: 'Routing and Flow Control in High-Speed Wide-Area Networks', *Proc. IEEE,* 1990, Vol. 78, No. 1, pp. 204-221

40. Antonio, J.K., Huang, G.M. and Tsai, W.K.: 'A Fast Distributed Shortest Path Algorithm for a Class of Hierarchically Structured Data Networks', *Proc. of IEEE INFOCOM*, 1989, pp. 183-192

41. Glazer, D.W. and Tropper, C.: 'A New Metric for Dynamic Routing Algorithms', *IEEE Trans. on Comms.*, 1990, Vol. COM-38, No. 3, pp. 360-367

42. Fratta, L., Gerla, M. and Kleinrock, L.: 'Flow Deviation Method: An Approach to Store and Forward Communication Network Design', *Networks*, 1973, Vol. 3, pp. 97-133

43. Gavish, B. and Hantler, S.L.: 'An Algorithm for Optimal Route Selection in SNA Networks', *IEEE Trans. on Comms.*, 1983, Vol. COM-31, No. 10, pp. 1154-1160

44. Gallager, R.G.: 'A Minimum Delay Routing Algorithm Using Distributed Computation', *IEEE Trans. on Comms.*, 1977, Vol. COM-25, No. 1, pp. 73-84

45. Bellman, R.: 'On a Routing Problem', *Quarterly of App. Math.*, 1958, Vol. 16, No. 1, pp. 87-90

46. Dijkstra, E.W.: 'A Note on Two Problems in Connection with Graphs', *Num. Math.*, 1959, Vol. 1, pp. 269-271

47. Ford, L.R. and Fulkerson, D.R.: 'Flows in Networks' (Princeton University Press, Princeton, NJ, 1962)

48. Ahuja, R.K. et al : 'Faster Algorithms for the Shortest Path Problem', *J. of the Ass. for Comp. Mach.*, 1990, Vol. 37, No. 2, pp. 213-223

49. Deo, N. and Pang, C.: 'Shortest-Path Algorithms: Taxonomy and Annotation', *Networks,* 1984, Vol. 14, pp. 275-323

50. Pape, U.: 'Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem', *Math. Prog.*, 1974, Vol. 7, pp. 212-222

51. Dial, R. et al : 'A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees', *Networks*, 1979, Vol. 9, pp. 215-248

52. Rosenberg, J.L., Gruchevsky, S.A. and Piscitello, D.M.: 'Adaptive Routing in Burroughs Network Architecture', *Proc. SIGCOMM '88*, 1988, pp. 173-184

53. Fultz, G.L. and Kleinrock, L.: 'Adaptive Routing Techniques for Store and Forward Computer-Communication Networks', *Proc. of IEEE ICC '71*, 1971, pp. 39.1-39.8

54. Naylor, W.E.: 'Loop-Free Adaptive Routing Algorithm for Packet Switched Networks', *Proc. of Data Comms. Symp.*, 1975, pp. 7.9-7.15

55. Cegrell, T.: 'A Routing Procedure for the TIDAS Message-Switching Network', *IEEE Trans. on Comms.*,1975, Vol. COM-23, No. 6, pp. 575-585

56. Tajibnapis, W.D.: 'A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network', *Comm. of the ACM*, 1977, Vol. 20, No. 7, pp. 477-485

57. Merlin, P.M. and Segall, A.: 'A Failsafe Distributed Routing Protocol', *IEEE Trans. on Comms.*, 1979, Vol. COM-27, No. 9, pp. 1280-1287

58. Jaffe, J.M. and Moss, F.H.: 'A Responsive Distributed Routing Algorithm for Computer Networks', *IEEE Trans. on Comms.*, 1982, Vol. COM-30, No. 7, pp. 1758-1762

59. Johnson, M.J.: 'Analysis of Routing Table Update Activity After Resource Failure in a Distributed Computer Network', *Proc. SIGCOMM '83*, 1983, pp. 14-20

60. Sloman, M.S. and Andriopoulos, X.: 'A Routing Algorithm for Interconnected Local Area Networks', *Computer Networks and ISDN Systems*, 1985, Vol. 9, pp. 109-130

61. Hagouel, J.: 'Issues in Routing for Large and Dynamic Networks', PhD Thesis, Graduate School of Art and Sciences, Columbia University, 1983

62. Garcia-Luna-Aceves, J.J.: 'A New Minimum-Hop Routing Algorithm', *Proc. IEEE INFOCOM '87*, 1987, pp. 170-180

63. Shin, K.G. and Chen, M.S.: 'Performance Analysis of Distributed Routing Strategies Free of Ping-Pong Type Looping', *IEEE Trans. on Comp.*, 1987, Vol. C-36, No. 2, pp. 129-137

64. Daneshrad, B and Morgera, S.D.: 'Application of Stochastic Automaton Theory for Routing in a Packet-Switched Network', *Proc. IEEE MILCOM '89*, 1989, pp. 11.1.1-11.1.5

65. Rajagopalan, B. and Faiman, M.: 'A New Responsive Distributed Shortest-Path Routing Algorithm', *Proc. SIGCOMM '89*, 1989, pp. 237-246

66. Garcia-Luna-Aceves, J.J.: 'A Minimum-Hop Routing Algorithm Based on Distributed Information', Comp. Net. and ISDN Sys., 1989, Vol. 16, No. 5, pp. 367-382

67. Cheng, C. et al : 'A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect', *Proc. SIGCOMM '89*, 1989, pp. 224-236

68. Awerbuch, B.: 'Shortest-Paths and Loop-Free Routing in Dynamic Networks', *Proc. SIGCOMM '90*, 1990, pp. 177-187

69. Shin, K.G. and Chen, M.S.: 'Minimal Order Loop-Free Routing Strategy', *IEEE Trans. on Comp.*, 1990, Vol. 39, No. 7, pp. 870-881

70. Humblet, P.A.: 'Another Adaptive Distributed Shortest Path Algorithm', *IEEE Trans. on Comms.*, 1991, Vol. 39, No. 6, pp. 995-1003

71. Zaumen, W.T. and Garcia-Luna-Aceves, J.J.: 'Dynamics of Distributed Shortest-Path Algorithms', *Proc. SIGCOMM '91*, 1991, pp. 31-42

72. Garcia-Luna-Aceves, J.J.: 'A Unified Approach to Loop-Free Routing Using Distance Vectors or Link States', *Proc. SIGCOMM '89*, 1989, pp. 212-223

73. McQuillan, J.M., Richer, I. and Rosen, E.C.: 'The New Routing Algorithm for the ARPANET', *IEEE Trans. on Comms.*, 1980, Vol. COM-28, No. 5, pp. 711-719

74. Perlman, R.: 'A Comparison Between Two Routing Protocols: OSPF and ISIS', *IEEE Network*, September 1991, pp. 18-24

75. 'Information Technology, Telecommunications and Information Exchange Between Systems, Intermediate System to Intermediate System Routing Information Exchange Protocol for Use in Conjunction With ISO 8473', *ISO 10589*, 1990

76. Wilson, R.J.: 'Introduction to Graph Theory', (Longman, New York, NY, 1985)

77. Wang, Z. and Crowcroft, J.: 'Shortest Path First with Emergency Exits', *Proc. SIGCOMM '90*, 1990, pp. 166-176

78. Nelson, D.J., Sayood, K. and Chang, H.: 'An Extended Least-Hop Distributed Routing Algorithm', *IEEE Trans. on Comms.*, 1990, Vol. 38, No. 4, 520-528

79. Topkis, D.M.: 'k Shortest Path Algorithm for Adaptive Routing in Communications Networks', *Proc. IEEE ICC '86*, 1986, pp. 108-113

80. Lawler, E.L.: 'A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem', *Man. Science*, 1972, Vol. 18, No. 7, pp. 401-405

81. Yen, J.Y.: 'Finding the k Shortest Loopless Paths in a Network', *Man. Science*, 1971, Vol. 17, No. 11, 712-716

82. Shier, D.R.: 'On Algorithms for Finding the k Shortest Paths in a Network', *Networks*, 1979, Vol. 9, pp. 195-214

83. Sidhu, D., Nair, R. and Abdallah, S.: 'Finding Disjoint Paths in Networks', *Proc. SIGCOMM '91*,1991, pp. 43-51

84. LaRue, W.W., Frost, V.S. and Shanmugan, K.S.: 'Some New Efficient Techniques for the Simulation of Computer Communications Networks', *Proc. IEEE INFOCOM '87*, 1987, pp. 152-158

85. Chiarawongse, J. et al : 'A Simulation Model for a Large Interconnected Local Area Network by Decomposition Techniques', *Proc. IEEE INFOCOM '87*, 1987, pp. 64-72

86. Gerla, M. and Kleinrock, L.: 'Congestion Control in Interconnected LANs', IEEE Network, 1987, Vol. 2, No. 1, pp. 72-76

87. Jain, R.: 'Congestion Control in Computer Networks: Issues and Trends', IEEE Network, 1990, Vol. 5, No. 5, pp. 24-30

88. Russell, E.C.: 'Building Simulation Models with SIMSCRIPT II.5' (CACI Products Co., La Jolla, CA, 1989)

89. Saad, S. and Schwartz, M.: 'Input Buffer Limiting Mechanisms for Congestion Control', *Proc. IEEE ICC '80*, 1980, pp. 23.1.1-23.1.5

90. Maxemchuk, N.F.:'Regulat Mesh Topologies in Local and Metropolitan Area Networks', *AT&T Technical Journal*, 1985, Vol. 64, No. 7, pp. 1659-1785

91. Maxemchuk, N.F.: 'Routing in the Manhattan Street Network', *IEEE Trans. on Comms.*, 1987, Vol. COM-35, No. 5, pp. 503-512

92. Chung, T.Y., Rai, S. and Agrawal, D.P.: 'Doubly Connected Multi-dimensional Regular Topologies for MANs and LANs' *Proc. IEEE INFOCOM '88*, 1988, pp. 551-557

93. Robertazzi, T.G.: 'Toroidal Networks', *IEEE Comms.*, 1988, Vol. 26, No. 6, pp. 45-50

94. Dowd, P.W. and Jabbour, K.: 'A Unified Approach to Local Area Network Interconnection', *IEEE J. on Sel. Areas in Comms.*, 1987, Vol. SAC-5, No. 9, pp. 1418-1425

95. Jain, R.: 'Peformance Analysis of FDDI Token Ring Networks: Effects of Parameters and Guidelines for Setting TTRT', *IEEE LTS*, 1981, Vol. 2, No. 2, pp. 16-22

96. Warford, I.J. and Brewster, R.L.: 'Improved Routing Algorithm for Interconnected LANs', *IEE Electronics Letters*, 1992, Vol. 28, No. 14, p. 1308

97. Garcia-Luna-Aceves, J.J.: 'A Fail-Safe Routing Algorithm for Multi-Hop Packet Radio Networks', *IEEE INFOCOM '86*, 1986, Miami, Fla., pp.434-443

Publications

1. Warford, I.J. and Brewster, R.L.: 'Improvements on the ISIS Routing Algorithm', Submitted to *IEE Electronics and Communications and Communication Engineering Journal.*

2. Warford, I.J. and Brewster, R.L.: 'Improved Routing Algorithm for Interconnected LANs', *IEE Electronics Letters*, 1992, Vol. 28, No. 14, p. 1308

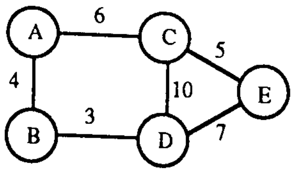
3. Warford, I.J. and Brewster, R.L.: 'A Routing Algorithm for Interconnected FDDI LANs', *Eighth UK Teletraffic Symposium*, 10-12 April 1991, Nottingham

4. Warford, I.J. and Brewster, R.L.: 'Routing in Interconnected FDDI LANs', *Third IEE Conf. on Telecomms.*, 17-20 March 1991, Edinburgh

Appendix A

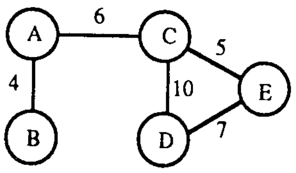
Control messages required for the original ARPANET routing algorithm after a link failure. (Assuming that each step occurs synchronously)

Before link failure



A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	10	4A	A	6	17	16	6A	A	7	16	18	7B	A	11	14	11A
B	4	16	4B	B	0	0	0	B	10	13	15	10A	B	3	20	17	3B	B	15	10	10D
C	14	6	6C	C	10	13	10A	C	0	0	0	0	C	13	10	12	10C	C	5	17	5C
D	7	16	7B	D	11	3	3D	D	13	10	12	10D	D	0	0	0	0	D	15	7	7D
E	14	11	11C	E	14	10	10D	E	17	17	5	5E	E	13	15	7	7E	E	0	0	0

Assume that link BD fails. B and D both set column vectors to high values.



A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4A	A	6	17	16	6A	A	99	16	18	16C	A	11	14	11A
B	4	16	4B	B	0	0	0	B	10	13	15	10A	B	99	20	17	17E	B	15	10	10D
C	14	6	6C	C	10	99	10A	C	0	0	0	0	C	99	10	12	10C	C	5	17	5C
D	7	16	7B	D	11	99	11A	D	13	10	12	10D	D	0	0	0	0	D	15	7	7D
E	14	11	11C	E	14	99	14A	E	17	17	5	5E	E	99	15	7	7E	E	0	0	0

Node B sends (BD,11) and (BE,14) to A.
Node D sends (DA,16) and (DB,17) to C and E.

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4A	A	6	26	16	6A	A	99	16	18	16C	A	11	23	11A
B	4	16	4B	B	0	0	0	B	10	27	15	10A	B	99	20	17	17E	B	15	24	15C
C	14	6	6C	C	10	99	10A	C	0	0	0	0	C	99	10	12	10C	C	5	17	5C
D	15	16	15B	D	11	99	11A	D	13	10	12	10D	D	0	0	0	0	D	15	7	7D
E	18	11	11C	E	14	99	14A	E	17	17	5	5E	E	99	15	7	7E	E	0	0	0

Node A sends (AD,15) to B and C. Node E sends (EB,15) to C and D.

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4A	A	6	26	16	6A	A	99	16	18	16C	A	11	23	11A
B	4	16	4B	B	0	0	0	B	10	27	20	10A	B	99	20	22	20C	B	15	24	15C
C	14	6	6C	C	10	99	10A	C	0	0	0	0	C	99	10	12	10C	C	5	17	5C
D	15	16	15B	D	19	99	19A	D	19	10	12	10D	D	0	0	0	0	D	15	7	7D
E	18	11	11C	E	14	99	14A	E	17	17	5	5E	E	99	15	7	7E	E	0	0	0

Node B sends (BD,19) to A. Node D sends (DB,20) to C and E.

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4A	A	6	26	16	6A	A	99	16	18	16C	A	11	23	11A
B	4	16	4B	B	0	0	0	B	10	30	20	10A	B	99	20	22	20C	B	15	27	15C
C	14	6	6C	C	10	99	10A	C	0	0	0	0	C	99	10	12	10C	C	5	17	5C
D	23	16	16B	D	19	99	19A	D	19	10	12	10D	D	0	0	0	0	D	15	7	7D
E	18	11	11C	E	14	99	14A	E	17	17	5	5E	E	99	15	7	7E	E	0	0	0

Node A sends (AD,16) to B and C

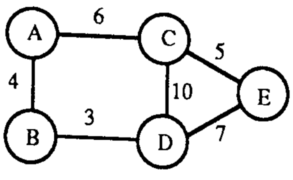
A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4A	A	6	26	16	6A	A	99	16	18	16C	A	11	23	11A
B	4	16	4B	B	0	0	0	B	10	30	20	10A	B	99	20	22	20C	B	15	27	15C
C	14	6	6C	C	10	99	10A	C	0	0	0	0	C	99	10	12	10C	C	5	17	5C
D	23	16	16B	D	20	99	20A	D	22	10	12	10D	D	0	0	0	0	D	15	7	7D
E	18	11	11C	E	14	99	14A	E	17	17	5	5E	E	99	15	7	7E	E	0	0	0

Node B sends (BD,20) to A

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4A	A	6	26	16	6A	A	99	16	18	16C	A	11	23	11A
B	4	16	4B	B	0	0	0	B	10	30	20	10A	B	99	20	22	20C	B	15	27	15C
C	14	6	6C	C	10	99	10A	C	0	0	0	0	C	99	10	12	10C	C	5	17	5C
D	24	16	16B	D	20	99	20A	D	22	10	12	10D	D	0	0	0	0	D	15	7	7D
E	18	11	11C	E	14	99	14A	E	17	17	5	5E	E	99	15	7	7E	E	0	0	0

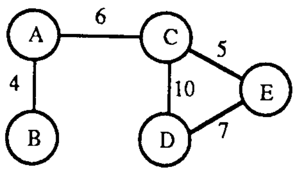
Control messages required for Cegrell's routing algorithm after a link failure.

Before link failure



A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	10	4	A	6	17	16	6	A	7	16	18	7	A	11	14	11
B	4	16	4	B	0	0	0	B	10	13	15	10	B	3	20	17	3	B	15	10	10
C	14	6	6	C	10	13	10	C	0	0	0	0	C	13	10	12	10	C	5	17	5
D	7	16	7	D	11	3	3	D	13	10	12	10	D	0	0	0	0	D	15	7	7
E	14	11	11	E	15	10	10	E	17	17	5	5	E	13	15	7	7	E	0	0	0

Assume link BD fails. B and D set the column vector to 99.



A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4	A	6	17	16	6	A	99	16	18	16	A	11	14	11
B	4	16	4	B	0	0	0	B	10	13	15	10	B	99	20	17	17	B	15	10	10
C	14	6	6	C	10	99	10	C	0	0	0	0	C	99	10	12	10	C	5	17	5
D	7	16	7	D	11	99	11	D	13	10	12	10	D	0	0	0	0	D	15	7	7
E	14	11	11	E	15	99	15	E	17	17	5	5	E	99	15	7	7	E	0	0	0

Node B sends (99,0,99,99,99) to A.
Node D sends (18,17,12,0,7) to C and (16,20,10,0,15) to E.

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4	A	6	28	16	6	A	99	16	18	16	A	11	23	11
B	4	16	4	B	0	0	0	B	10	27	15	10	B	99	20	17	17	B	15	27	15
C	99	6	6	C	10	99	10	C	0	0	0	0	C	99	10	12	10	C	5	17	5
D	99	16	16	D	11	99	11	D	13	10	12	10	D	0	0	0	0	D	15	7	7
E	99	11	11	E	15	99	15	E	17	17	5	5	E	99	15	7	7	E	0	0	0

Node A sends (0,16,6,16,11) to B and (0,4,99,99,99) to C.
Node C sends (16,15,0,10,5) to A, (6,10,0,12,5) to D and (6,10,0,10,17) to E
Node E sends (23,27,17,7,0) to C and (11,15,5,15,0) to D.

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4	A	6	28	28	6	A	99	16	18	16	A	11	23	11
B	4	21	4	B	0	0	0	B	10	27	32	10	B	99	20	22	20	B	15	27	15
C	99	6	6	C	10	99	10	C	0	0	0	0	C	99	10	12	10	C	5	17	5
D	99	16	16	D	20	99	20	D	99	10	12	10	D	0	0	0	0	D	15	7	7
E	99	11	11	E	15	99	15	E	99	17	5	5	E	99	15	7	7	E	0	0	0

Node A sends (0,21,6,16,11) to B and (0,4,99,99,99) to C.

Node B sends (99,0,99,99,99) to A.
Node C sends (28,27,0,10,5) to A, (6,10,0,12,5) to D and (6,10,0,10,17) to E.
Node D sends (18,22,12,0,7) to C and (16,20,10,0,15) to E.

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4	A	6	28	28	6	A	99	16	18	16	A	11	23	11
B	4	21	4	B	0	0	0	B	10	27	32	10	B	99	20	22	20	B	15	27	15
C	99	6	6	C	10	99	10	C	0	0	0	0	C	99	10	12	10	C	5	17	5
D	99	16	16	D	20	99	20	D	99	10	12	10	D	0	0	0	0	D	15	7	7
E	99	11	11	E	15	99	15	E	99	17	5	5	E	99	15	7	7	E	0	0	0

Node A sends (0,4,99,99,99) to C and (0,33,6,16,11) to B
Node C sends (28,32,0,10,5) to A, (6,10,0,12,5) to D and (6,10,0,10,17) to E

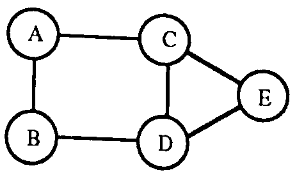
A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4	A	6	28	28	6	A	99	16	18	16	A	11	23	11
B	4	38	4	B	0	0	0	B	10	27	32	10	B	99	20	22	20	B	15	27	15
C	99	6	6	C	10	99	10	C	0	0	0	0	C	99	10	12	10	C	5	17	5
D	99	16	16	D	20	99	20	D	99	10	12	10	D	0	0	0	0	D	15	7	7
E	99	11	11	E	15	99	15	E	99	17	5	5	E	99	15	7	7	E	0	0	0

Node A sends (0,4,99,99,99) to C and (0,38,6,16,11) to B.

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	0	0	0	A	4	99	4	A	6	28	28	6	A	99	16	18	16	A	11	23	11
B	4	38	4	B	0	0	0	B	10	27	32	10	B	99	20	22	20	B	15	27	15
C	99	6	6	C	10	99	10	C	0	0	0	0	C	99	10	12	10	C	5	17	5
D	99	16	16	D	20	99	20	D	99	10	12	10	D	0	0	0	0	D	15	7	7
E	99	11	11	E	15	99	15	E	99	17	5	5	E	99	15	7	7	E	0	0	0

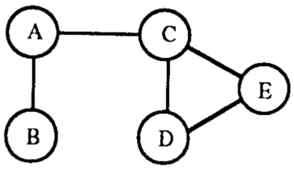
Control messages required for Tajibnapis's algorithm after a link failure.

Before link failure.



A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	-	-	-	A	1	3	1A	A	1	3	3	1A	A	2	2	3	2B	A	2	3	2C
B	1	3	1B	B	-	-	-	B	2	2	3	2A	B	1	3	3	1B	B	3	2	2D
C	3	1	1C	C	2	2	2A	C	-	-	-	-	C	3	1	2	1C	C	1	2	1C
D	2	2	2B	D	3	1	1D	D	3	1	2	1D	D	-	-	-	-	D	2	1	1D
E	3	2	2C	E	3	2	2D	E	3	2	1	1E	E	3	2	1	1E	E	-	-	-

Link BD goes down.



A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	-	-	-	A	1	5	1A	A	1	3	3	1A	A	5	2	3	2C	A	2	3	2C
B	1	3	1B	B	-	-	-	B	2	2	3	2A	B	5	3	3	3C	B	3	2	2D
C	3	1	1C	C	2	5	2A	C	-	-	-	-	C	5	1	2	1C	C	1	2	1C
D	2	2	2B	D	3	5	3A	D	3	1	2	1D	D	-	-	-	-	D	2	1	1D
E	3	2	2C	E	3	5	3A	E	3	2	1	1E	E	5	2	1	1E	E	-	-	-

B sends a Netchange message: [BD,3] and [BE,3] to A.
D sends a Netchange message: [DB,3] to C and E.

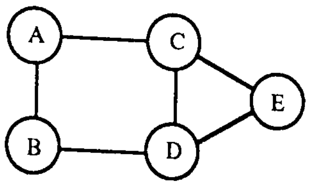
A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	-	-	-	A	1	5	1A	A	1	3	3	1A	A	5	2	3	2C	A	2	3	2C
B	1	3	1B	B	-	-	-	B	2	4	3	2A	B	5	3	3	3C	B	3	4	3C
C	3	1	1C	C	2	5	2A	C	-	-	-	-	C	5	1	2	1C	C	1	2	1C
D	4	2	2C	D	3	5	3A	D	3	1	2	1D	D	-	-	-	-	D	2	1	1D
E	4	2	2C	E	3	5	3A	E	3	2	1	1E	E	5	2	1	1E	E	-	-	-

E sends a Netchange message: [EB,3] to C and D

A	B	C	R	B	A	D	R	C	A	D	E	R	D	B	C	E	R	E	C	D	R
A	-	-	-	A	1	5	1A	A	1	3	3	1A	A	5	2	3	2C	A	2	3	2C
B	1	3	1B	B	-	-	-	B	2	4	4	2A	B	5	3	4	3C	B	3	4	3C
C	3	1	1C	C	2	5	2A	C	-	-	-	-	C	5	1	2	1C	C	1	2	1C
D	4	2	2C	D	3	5	3A	D	3	1	2	1D	D	-	-	-	-	D	2	1	1D
E	4	2	2C	E	3	5	3A	E	3	2	1	1E	E	5	2	1	1E	E	-	-	-

Control messages required for the Merlin-Segall routing algorithm after a link failure.

Before link failure.



A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	3	A1	A	1	3	3	A1	A	2	2	3	B2	A	2	3	C2
B	1	3	B1	B	0	0	00	B	2	2	3	A2	B	1	3	3	B1	B	3	2	D2
C	3	1	C1	C	2	2	A2	C	0	0	0	00	C	3	1	2	C1	C	1	2	C1
D	2	2	C2	D	3	1	D1	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	3	2	C2	E	3	2	D2	E	3	2	1	E1	E	3	2	1	E1	E	0	0	00

Assume link BD fails

T1: B begins update procedure. B sends B=0 to A.
D sends DB=99 to C and E.
T1: B sends REQ to A to begin update cycle.
D sends DA=99 to C and E.
T1: D begins update procedure. D sends D=0 to C and E.
B sends BD=99 to A.
T1: D sends REQ to E to begin update cycle.
B sends BE=99 to A.

A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	99	A1	A	1	99	3	A1	A	99	2	3	99	A	2	99	C2
B	1	3	B1	B	0	0	00	B	2	99	3	A2	B	99	3	3	99	B	3	99	99
C	3	1	C1	C	2	99	A2	C	0	0	0	00	C	99	1	2	C1	C	1	2	C1
D	99	2	C2	D	3	99	99	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	99	2	C2	E	3	99	99	E	3	2	1	E1	E	99	2	1	E1	E	0	0	00

T2: A sends AB=1 to C.
T2: A begins update procedure. A sends A=0 to C and B.
T2: C sends CD=1 to A and E.
E sends ED=1 to C.
T2: E begins update procedure. E sends E=0 to C and D.

A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	99	A1	A	1	99	3	A1	A	99	2	3	99	A	2	99	C2
B	1	3	B1	B	0	0	00	B	2	99	3	A2	B	99	3	3	99	B	3	99	99
C	3	1	C1	C	2	99	A2	C	0	0	0	00	C	99	1	2	C1	C	1	2	C1
D	99	2	C2	D	3	99	99	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	99	2	C2	E	3	99	99	E	3	2	1	E1	E	99	2	1	E1	E	0	0	00

T3: C sends CB=2 to D and E.
T3: B selects A as the preferred neighbor to A and informs A.
C sends CA=1 to E and D.
T3: A sends AD=2 to B.
T3: C sends CE=1 to D and A.
D sends DE=1 to C.

A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	99	A1	A	1	99	3	A1	A	99	2	3	99	A	2	99	C2
B	1	3	B1	B	0	99	00	B	2	99	3	A2	B	99	3	3	99	B	3	99	99
C	3	1	C1	C	2	99	A2	C	0	0	0	00	C	99	1	2	C1	C	1	2	C1
D	99	2	C2	D	3	99	99	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	99	2	C2	E	3	99	99	E	3	2	1	E1	E	99	2	1	E1	E	0	0	00

T4: D sends DB=3 to E.
T4: E sends EA=2 to D.
T4: B selects A as its preferred neighbor to D and informs A.
T4: A sends AE=2 to B.

A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	99	A1	A	1	99	3	A1	A	99	2	3	99	A	2	99	C2
B	1	3	B1	B	0	0	00	B	2	99	3	A2	B	99	3	3	99	B	3	4	99
C	3	1	C1	C	2	99	A2	C	0	0	0	00	C	99	1	2	C1	C	1	2	C1
D	99	2	C2	D	3	99	A3	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	99	2	C2	E	3	99	99	E	3	2	1	E1	E	99	2	1	E1	E	0	0	00

T5: E chooses C as the preferred neighbor to B and informs C and D.
T5: D selects C as its preferred neighbor to A and informs C and E.
T5: A selects C as its preferred neighbor to D and informs C.
T5: B selects A as its preferred neighbor to E and informs A.

A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	99	A1	A	1	99	3	A1	A	99	2	3	C2	A	2	99	C2
B	1	3	B1	B	0	0	00	B	2	99	3	A2	B	99	3	3	99	B	3	4	C3
C	3	1	C1	C	2	99	A2	C	0	0	0	00	C	99	1	2	C1	C	1	2	C1
D	99	2	C2	D	3	99	A3	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	99	2	C2	E	3	99	A3	E	3	2	1	E1	E	99	2	1	E1	E	0	0	00

T6: D chooses C as the preferred neighbor to B and informs C.
T6: E selects C as its preferred neighbor to A and informs C.
T6: C selects D as its preferred neighbor to D and informs D and E.
T6: A selects C as its preferred neighbor to E and informs C.

A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	99	A1	A	1	99	3	A1	A	99	2	3	C2	A	2	99	C2
B	1	3	B1	B	0	0	00	B	2	99	3	A2	B	99	3	3	C3	B	3	4	C3
C	3	1	C1	C	2	99	A2	C	0	0	0	00	C	99	1	2	C1	C	1	2	C1
D	99	2	C2	D	3	99	A3	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	99	2	C2	E	3	99	A3	E	3	2	1	E1	E	99	2	1	E1	E	0	0	00

T7: C chooses A as the preferred neighbor to B and informs A.
T7: C selects A as its preferred neighbor to A and informs A.
T7: E selects D as its preferred neighbor to D and informs D
T7: C selects E as its preferred neighbor to E and informs E and D.

A	B	C	PnD	B	A	D	PnD	C	A	D	E	PnD	D	B	C	E	PnD	E	C	D	PnD
A	0	0	00	A	1	99	A1	A	1	99	3	A1	A	99	2	3	C2	A	2	99	C2
B	1	3	B1	B	0	0	00	B	2	99	3	A2	B	99	3	3	C3	B	3	4	C3
C	3	1	C1	C	2	99	A2	C	0	0	0	00	C	99	1	2	C1	C	1	2	C1
D	99	2	C2	D	3	99	A3	D	3	1	2	D1	D	0	0	0	00	D	2	1	D1
E	99	2	C2	E	3	99	A3	E	3	2	1	E1	E	99	2	1	E1	E	0	0	00

T8: A chooses B as the preferred neighbor to B and informs B.
T8: A terminates the cycle.
T8: D terminates the cycle.
T8: D selects E as its preferred neighbor to E and informs E.

A	B	C	PnD
A	0	0	00
B	1	3	B1
C	3	1	C1
D	99	2	C2
E	99	2	C2

B	A	D	PnD
A	1	99	A1
B	0	0	00
C	2	99	A2
D	3	99	A3
E	3	99	A3

C	A	D	E	PnD
A	1	99	3	A1
B	2	99	3	A2
C	0	0	0	00
D	3	1	2	D1
E	3	2	1	E1

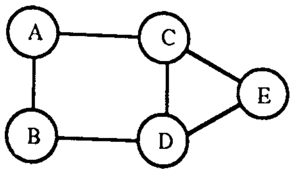
D	B	C	E	PnD
A	99	2	3	C2
B	99	3	3	C3
C	99	1	2	C1
D	0	0	0	00
E	99	2	1	E1

E	C	D	PnD
A	2	99	C2
B	3	4	C3
C	1	2	C1
D	2	1	D1
E	0	0	00

T9: E terminates the cycle.
T9: B terminates the cycle.

Control messages required for the Jaffe-Moss algorithm routing algorithm after a link failure.

Before link failure.



A	NnD	C	B
A	-	-	-
B	B1	3	1
C	C1	1	3
D	B2	2	2
E	C2	2	3

B	NnD	A	D
A	A1	1	3
B	-	-	-
C	A2	2	2
D	D1	3	1
E	D2	3	2

C	NnD	A	D	E
A	A1	1	3	3
B	A2	2	2	3
C	-	-	-	-
D	D1	3	1	2
E	E1	3	2	1

D	NnD	B	C	E
A	B2	2	2	3
B	B1	1	3	3
C	C1	3	1	2
D	-	-	-	-
E	E1	3	2	1

E	NnD	C	D
A	C2	2	3
B	D2	3	2
C	C1	1	2
D	D1	2	1
E	-	-	-

Assume that link BD fails:

T1: D sets C(D,A,B)=99
D sets C*(D,A)=99
D sends MSG(A,99,1) to C and E
D enters freeze state for des A
B sets C(B,A,D)=99

T1: D sets C(D,B,B)=99
D sets C*(D,B)=99
D sends MSG(B,99,1) to C and E
D enters freeze state for B

T1: B sets C(B,C,D)=99
D sets C(D,C,B)=99

T1: B sets C(B,D,D)=99
B sets C*(B,D)=99
B sends MSG(D,99,1) to A
B enters freeze state for D

T1: B sets C(B,E,D)=99
B sets C*(B,E)=99
B sends MSG(E,99,1) to A
B goes into freeze state
D sets C(D,E,B)=99

A	NnD	C	B
A	-	-	-
B	B1	3	1
C	C1	1	3
D	B2	2	2
E	C2	2	3

B	NnD	A	D
A	A1	1	99
B	-	-	-
C	A2	2	99
D	D99	3	99
E	D99	3	99

C	NnD	A	D	E
A	A1	1	3	3
B	A2	2	2	3
C	-	-	-	-
D	D1	3	1	2
E	E1	3	2	1

D	NnD	B	C	E
A	B99	99	2	3
B	B99	99	3	3
C	C1	99	1	2
D	-	-	-	-
E	E1	99	2	1

E	NnD	C	D
A	C2	2	3
B	D2	3	2
C	C1	1	2
D	D1	2	1
E	-	-	-

T2: C receives MSG(A,99,1) from D
C sets C(C,A,D)=99
C sends ACK(A) to D
E receives MSG(A,99,1) from D
E sets C(E,A,D)=99

E sends ACK(A) to D

T2: C receives MSG(B,99,1) from D
C sets C(C,B,D) =99
C sends ACK(B) to D
E receives MSG(B,99,1) from D
E sets C*(E,B)=99
E sends MSG(B,99,1) to C and D
E enters freeze state for B

T2: A receives MSG(D,99,1) from B
A sets C*(A,D)=99
A sends MSG(D,99,1) to B and C
A enters freeze state

T2: A receives MSG(E,99,1) from B
A sets C(A,E,B)=99
A sends ACK(E) to B

A	NnD	C	B	B	NnD	A	D	C	NnD	A	D	E	D	NnD	B	C	E	E	NnD	C	D
A	-	-	-	A	A1	1	99	A	A1	1	99	3	A	B99	99	2	3	A	C2	2	99
B	B1	3	1	B	-	-	-	B	A2	2	99	3	B	B99	99	3	3	B	D99	3	2
C	C1	1	3	C	A2	2	99	C	-	-	-	-	C	C1	99	1	2	C	C1	1	2
D	B99	2	2	D	D99	3	99	D	D1	3	1	2	D	-	-	-	-	D	D1	2	1
E	C2	2	99	E	D99	3	99	E	E1	3	2	1	E	E1	99	2	1	E	-	-	-

T3: D receives ACK(A) from C and E
D updates Nn(D,A)=C
D updates C*(D,A)=2
D sends MSG(A,2,0) to C and E

T3: D receives ACK(B) from C
C receives MSG(B,99,1) from E
C sets C(C,B,E) to 99
C sends ACK(B) to E
D receives MSG(B,99,1) from E
D sets C(D,B,E) to 99
D sends ACK(B) to E

T3: B receives MSG(D,99,1) from A
B sets C(B,D,A) = 99
B sends ACK(D) to A
C receives MSG(D,99,1) from A
C sets C(C,D,A)=99
C sends ACK(D) to A

T3: B receives ACK(E) from A
B updates Nn(B,E)=A
B updates C*(B,E)=3
B sends MSG(E,3,0) to A

A	NnD	C	B	B	NnD	A	D	C	NnD	A	D	E	D	NnD	B	C	E	E	NnD	C	D
A	-	-	-	A	A1	1	99	A	A1	1	99	3	A	C2	99	2	3	A	C2	2	99
B	B1	3	1	B	-	-	-	B	A2	2	99	99	B	B99	99	3	99	B	D99	3	2
C	C1	1	3	C	A2	2	99	C	-	-	-	-	C	C1	99	1	2	C	C1	1	2
D	B99	2	2	D	D99	99	99	D	D1	99	1	2	D	-	-	-	-	D	D1	2	1
E	C2	2	99	E	A3	3	99	E	E1	3	2	1	E	E1	99	2	1	E	-	-	-

T4: C receives MSG(A,2,0) from D
C updates C(C,A,D) to 3
E receives MSG(A,2,0) from D
E updates C(E,A,D) to 3

T4: E receives ACK(B) from C and D
E sends ACK(B) to D
E updates Nn(E,B) =C
E updates C*(E,B)=3
E sends MSG(B,3,0) to C and D
T4: A receives ACK(D) from B and C
A sends ACK(D) to B
A updates Nn(A,D)=C
A updates C*(A,D)=2
A sends MSG(D,2,0) to B and C
T4: A receives MSG(E,3,0)
A updates C(A,E,B)=4

A	NnD	C	B	B	NnD	A	D	C	NnD	A	D	E	D	NnD	B	C	E	E	NnD	C	D
A	-	-	-	A	A1	1	99	A	A1	1	3	3	A	C2	99	2	3	A	C2	2	3
B	B1	3	1	B	-	-	-	B	A2	2	99	99	B	B99	99	3	99	B	C3	3	2
C	C1	1	3	C	A2	2	99	C	-	-	-	-	C	C1	99	1	2	C	C1	1	2
D	C2	2	2	D	D99	99	99	D	D1	99	1	2	D	-	-	-	-	D	D1	2	1
E	C2	2	4	E	A3	3	99	E	E1	3	2	1	E	E1	99	2	1	E	-	-	-

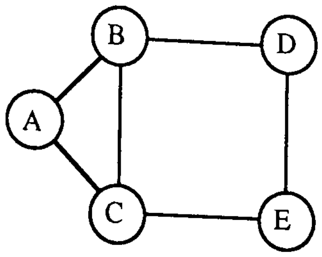
T5: D receives ACK(B) from E
D updates Nn(D,B)=C
D updates C*(D,B)=3
D sends MSG(B,3,0) to C and E
C receives MSG(B,3,0) from E
C updates C(C,B,E)=4
D receives MSG(B,3,0)
D updates C(D,B,E) to 4
T5: B receives ACK(D)
B receives MSG(D,2,0) from A
B updates Nn(B,D)=A
B updates C*(B,D) =3
B sends MSG(D,3,0) to A
C receives MSG(D,2,0) from A
C updates C(C,D,A) to 3

A	NnD	C	B	B	NnD	A	D	C	NnD	A	D	E	D	NnD	B	C	E	E	NnD	C	D
A	-	-	-	A	A1	1	99	A	A1	1	3	3	A	C2	99	2	3	A	C2	2	3
B	B1	3	1	B	-	-	-	B	A2	2	99	4	B	C3	99	3	4	B	C3	3	2
C	C1	1	3	C	A2	2	99	C	-	-	-	-	C	C1	99	1	2	C	C1	1	2
D	C2	2	2	D	A3	3	99	D	D1	3	1	2	D	-	-	-	-	D	D1	2	1
E	C2	2	4	E	A3	3	99	E	E1	3	2	1	E	E1	99	2	1	E	-	-	-

T6: C receives MSG(B,3,0) from D
C updates C(C,B,D)=4
E receives MSG(B,3,0) from D
E updates C(E,B,D)=4
T6: A receives MSG(D,3,0) from B
A updates C(A,D,B) to 4

A	NnD	C	B	B	NnD	A	D	C	NnD	A	D	E	D	NnD	B	C	E	E	NnD	C	D
A	-	-	-	A	A1	1	99	A	A1	1	3	3	A	C2	99	2	3	A	C2	2	3
B	B1	3	1	B	-	-	-	B	A2	2	4	4	B	C3	99	3	4	B	C3	3	4
C	C1	1	3	C	A2	2	99	C	-	-	-	-	C	C1	99	1	2	C	C1	1	2
D	C2	2	4	D	A3	3	99	D	D1	3	1	2	D	-	-	-	-	D	D1	2	1
E	C2	2	4	E	A3	3	99	E	E1	3	2	1	E	E1	99	2	1	E	-	-	-

Control messages required for the Hagouel routing algorithm after a link failure.
Before link failure.



A	R	S	B	R	S	C	R	S	D	R	S	E	R	S
A	-	-	A	1	A	A	1	A	A	2	B	A	2	C
B	1	B	B	-	-	B	1	B	B	1	B	B	2	C
C	1	C	C	1	C	C	-	-	C	2	B	C	1	C
D	2	B	D	1	D	D	2	B	D	-	-	D	1	D
E	2	C	E	2	C	E	1	E	E	1	E	E	-	-

Assume link CE fails

A	R	S	B	R	S	C	R	S	D	R	S	E	R	S
A	-	-	A	1	A	A	1	A	A	2	B	A	10	C
B	1	B	B	-	-	B	1	B	B	1	B	B	10	C
C	1	C	C	1	C	C	-	-	C	2	B	C	10	C
D	2	B	D	1	D	D	2	B	D	-	-	D	1	D
E	2	C	E	2	C	E	10	E	E	1	E	E	-	-

C sends (set,C,E,10) to A and B
E sends (set,E,A,10), (set,E,B,10), (set,E,C,10) to D

A	R	S	B	R	S	C	R	S	D	R	S	E	R	S
A	-	-	A	1	A	A	1	A	A	2	B	A	10	C
B	1	B	B	-	-	B	1	B	B	1	B	B	10	C
C	1	C	C	1	C	C	-	-	C	2	B	C	10	C
D	2	B	D	1	D	D	2	B	D	-	-	D	1	D
E	10	C	E	10	C	E	10	E	E	1	E	E	-	-

A sends (set,A,E,10) to B. B sends (set,B,E,10) to A and D
D sends (reset,D,A,2), (reset,D,B,1), (reset,D,C,2) to E

A	R	S	B	R	S	C	R	S	D	R	S	E	R	S
A	-	-	A	1	A	A	1	A	A	2	B	A	3	C
B	1	B	B	-	-	B	1	B	B	1	B	B	2	C
C	1	C	C	1	C	C	-	-	C	2	B	C	3	C
D	2	B	D	1	D	D	2	B	D	-	-	D	1	D
E	10	C	E	10	C	E	10	E	E	1	E	E	-	-

D sends (reset,D,E,1) to B

A	R	S	B	R	S	C	R	S	D	R	S	E	R	S
A	-	-	A	1	A	A	1	A	A	2	B	A	3	C
B	1	B	B	-	-	B	1	B	B	1	B	B	2	C
C	1	C	C	1	C	C	-	-	C	2	B	C	3	C
D	2	B	D	1	D	D	2	B	D	-	-	D	1	D
E	10	C	E	2	C	E	10	E	E	1	E	E	-	-

B sends (reset,B,E,2) to A and C

A	R	S	B	R	S	C	R	S	D	R	S	E	R	S
A	-	-	A	1	A	A	1	A	A	2	B	A	3	C
B	1	B	B	-	-	B	1	B	B	1	B	B	2	C
C	1	C	C	1	C	C	-	-	C	2	B	C	3	C
D	2	B	D	1	D	D	2	B	D	-	-	D	1	D
E	3	C	E	2	C	E	3	E	E	1	E	E	-	-

A sends (reset,A,E,3) to C
C sends (reset,C,E,3) to A

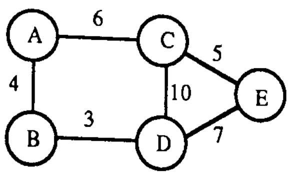
A	R	S	B	R	S	C	R	S	D	R	S	E	R	S
A	-	-	A	1	A	A	1	A	A	2	B	A	3	C
B	1	B	B	-	-	B	1	B	B	1	B	B	2	C
C	1	C	C	1	C	C	-	-	C	2	B	C	3	C
D	2	B	D	1	D	D	2	B	D	-	-	D	1	D
E	3	C	E	2	C	E	3	E	E	1	E	E	-	-

Algorithm converges.

Appendix B

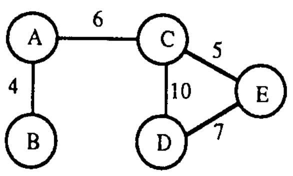
Control messages required for the new ARPANET routing algorithm after a link failure. (Assuming that each step occurs synchronously)

Before link failure



A	R	B	R	C	R	D	R	E	R
A	0	A	4A	A	6A	A	7B	A	11C
B	4B	B	0	B	10A	B	3B	B	10D
C	6C	C	10A	C	0	C	10C	C	5C
D	7B	D	3D	D	10D	D	0	D	7D
E	11C	E	10D	E	5E	E	7E	E	0

Assume that link BD fails. B and D both change their topology databases.



A	R	B	R	C	R	D	R	E	R
A	0	A	4A	A	6A	A	16C	A	11C
B	4B	B	0	B	10A	B	20C	B	10D
C	6C	C	10A	C	0	C	10C	C	5C
D	7B	D	20A	D	10D	D	0	D	7D
E	11C	E	15A	E	5E	E	7E	E	0

Node B sends BD=99 to A.
Node D sends DB=99 to C and E.

A	R	B	R	C	R	D	R	E	R
A	0	A	4A	A	6A	A	16C	A	11C
B	4B	B	0	B	10A	B	20C	B	25C
C	6C	C	10A	C	0	C	10C	C	5C
D	11C	D	20A	D	10D	D	0	D	7D
E	11C	E	15A	E	5E	E	7E	E	0

Node A sends BD=99 to C.
Node C sends BD=99 to A and E.
Node E sends BD=99 to C.

Algorithm converges.

Appendix C

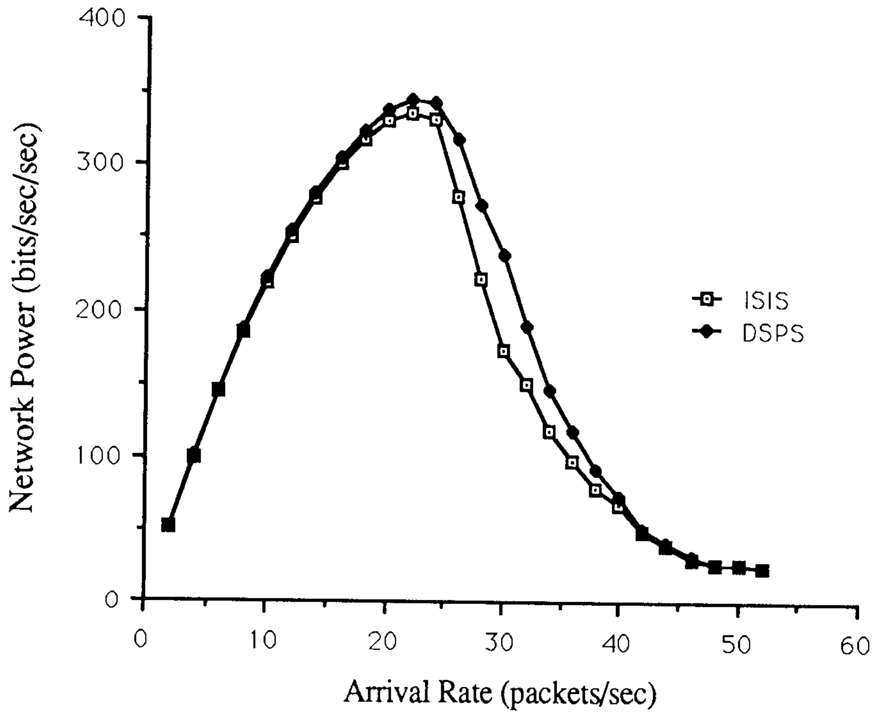
Pseudo-code for the shortest path and forwarding procedures for the DSPS routing algorithm.

```
PROCEDURE SHORTEST PATHS (DSPS)
BEGIN
  TENT := [0]; PATHS := [0]
  place (SELF,0,0) in PATHS
  FOR each Adj of SELF DO
    place <P,x,{Adj(P)}> in TENT
  END;
  WHILE TENT is nonempty DO
    get <P,x,{Adj(P)}> from TENT with x = minimum
    remove <P,x,{Adj(P)} from TENT
    add <P,x,{Adj(P)} to PATHS
    FOR each P, P ∈ Adj(P), DO
      FOR each N of P DO
        dist(P,N) = d(P) + metric(P,N)
        IF <N,d(N),Adj(N)> ∉ PATHS THEN
          IF <N,x,Adj(N)> ∈ TENT THEN
            IF x = dist(P,N) THEN
              IF Adj(P) ∈ Adj(N) THEN
                dsps(P) = dsps(P) + 1
                place dsps(P) in <N,x,{dsps(N)}>
              ELSE
                place Adj(P) in Adj(N)
              END;
            ELSE IF x > dist(P,N) THEN
              remove <N,x,{Adj(N)}> from TENT
              remove <N,x,{dsps(N)}> from TENT
              put <N,dist(P,N),{Adj(P)}> in TENT
            END;
          ELSE
            place <N,dist(P,N),{P}> in TENT
          END;
        END;
      END;
```

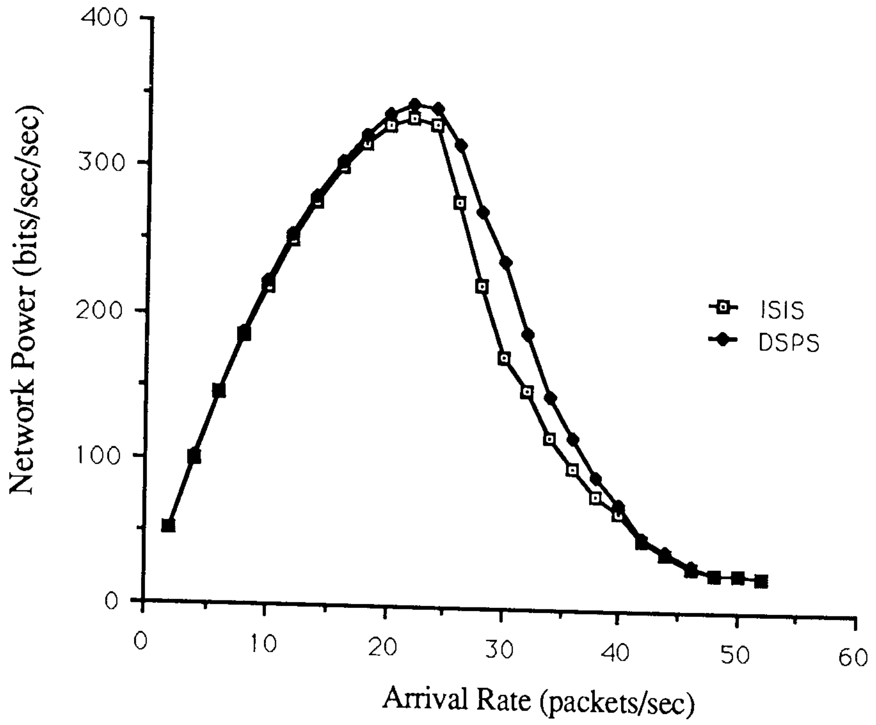
```
        END;
      END;
    END; Shortest Paths (DSPS)


PROCEDURE FORWARD (DSPS)
(given Dest, yielding adj)
BEGIN
  IF NextNodeTable[Dest.Adj].count > 1 THEN
    minQueue := MaxUnsigned;
    FOR each Adj of NextNodeTable[Dest] DO
      IF QueueSize(NextNodeTable[Dest.Adj]) < MaxBufferSize THEN
        dspsCount := DspsTable[Dest.Adj];
        IF minQueue > QueueSize(NextNodeTable[Dest.Adj]) - dspsCount THEN
          minQueue := QueueSize(NextNodeTable[Dest.Adj]) - dspsCount;
          adj := NextNodeTable[Dest.Adj];
        END;
      END;
    END;
  ELSE
    adj := NextNodeTable[Dest.Adj];
  END;
END; Forward (DSPS)
```

Appendix D

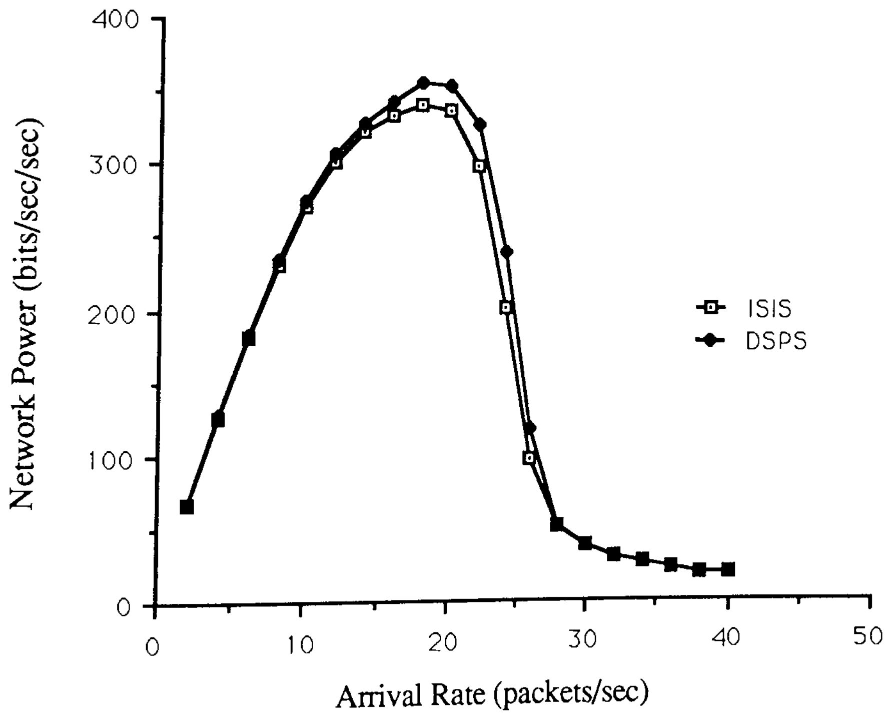
Additional simulation results for the ISIS and DSPS routing algorithms.



Power against arrival rate for ISIS and DSPS (8 nodes, 14 links)



Power against arrival rate for ISIS and DSPS (12 nodes, 14 links).



Power against arrival rate for ISIS and DSPS (14 nodes, 19 links)

Appendix E

Shortest path, alternative path and forwarding procedures for routing algorithms A, B and C.

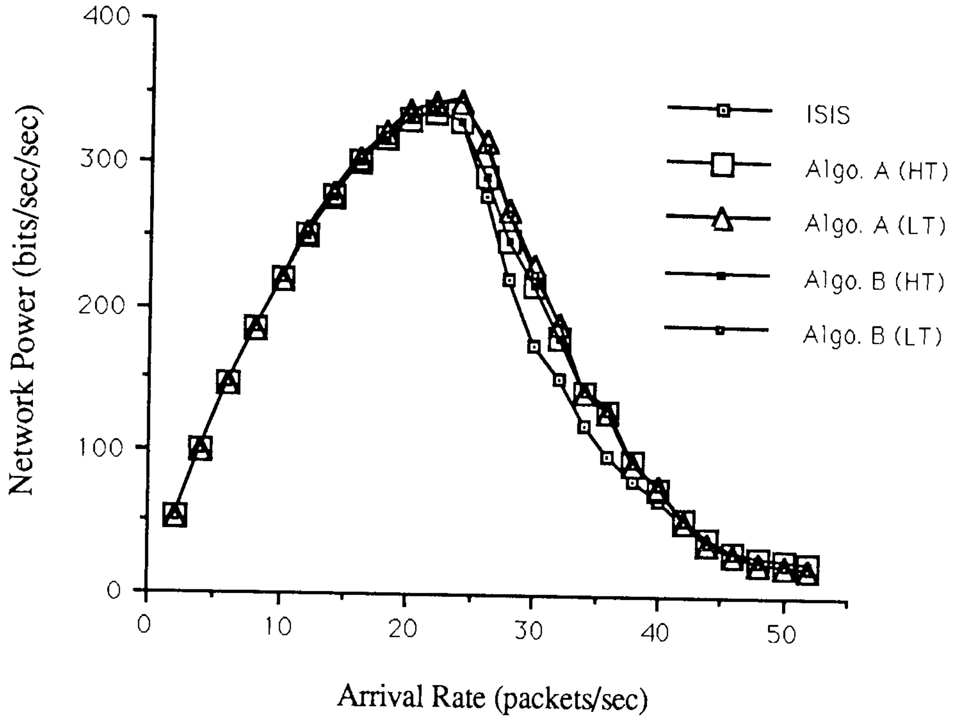
```
PROCEDURE SHORTEST PATHS (Algorithms A,B and C)
BEGIN
  TENT := [0]; PATHS := [0]
  place (SELF,0,0) in PATHS
  FOR each Adj of SELF DO
    place <P,x,{Adj(P)}> in TENT
  END;
  WHILE TENT is nonempty DO
    get <P,x,{Adj(P)}> from TENT with x = minimum
    remove <P,x,{Adj(P)} from TENT
    add <P,x,{Adj(P)} to PATHS
    FOR each P, P ∈ Adj(P), DO
      FOR each N of P DO
        dist(P,N) = d(P) + metric(P,N)
        IF <N,d(N),Adj(N)> ∉ PATHS THEN
          IF <N,x,Adj(N)> ∈ TENT THEN
            IF x = dist(P,N) THEN
              place Adj(P) in Adj(N)
            ELSE IF x > dist(P,N) THEN
              remove <N,x,{Adj(N)}> from TENT
              put <N,dist(P,N),{Adj(P)}> in TENT
            END;
          ELSE
            place <N,dist(P,N),{P}> in TENT
          END;
        END;
      END;
    END;
  END;
END Shortest Paths (Algorithms A,B and C)
```

```
PROCEDURE ALTERNATIVE PATHS (Algorithms A,B and C)
BEGIN
  IF Adj.count > 1 THEN
    PATHS[index] := [0];
    index := 1;
    FOR each Adj of SELF DO
      temp := Adj[index];
      Adj[index] := 0;
      SHORTEST PATHS
      PATHS[index] := PATHS;
      Adj[index] := temp;
      index := index+1;
    END;
    FOR each destination DO
      FOR each k DO                              (* algorithm C only *)
        FOR each index DO
          IF PATHS[index] = min, THEN
            temp := index
            place triple in SP[1]                  (* algos A and B only *)
          ELSE IF PATHS[index] = min + 1 THEN    (* algos A and B only *)
            place triple in SP[2]                  (* algos A and B only  *)
          END;
        END;
        place triple in SP[k]                    (* algorithm C only *)
        set triple in PATHS[temp] to MaxUnsigned
      END;                                       ( * algorithm C only * )
    END;
  ELSE
    SHORTEST PATHS
  END;
END; Alternative Paths (Algorithms A,B and C)
```

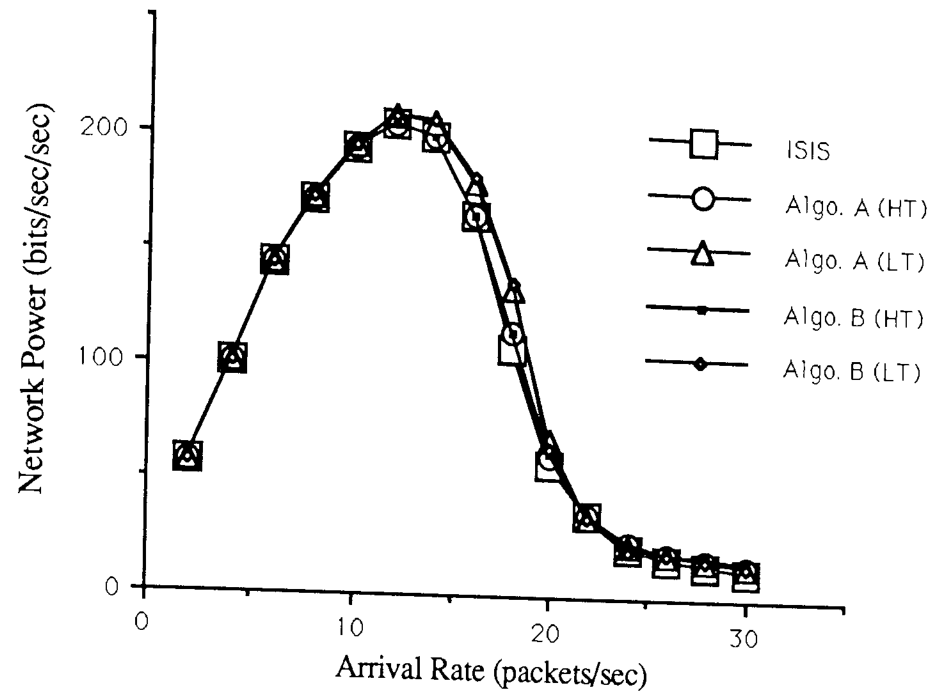
```
PROCEDURE FORWARD (Algorithms A,B and C)
(given Dest, yielding adj)
BEGIN
  IF NextNodeTable[Dest.Adj].count > 1 THEN
    minQueue := MaxUnsigned;
    FOR each Adj of NextNodeTable[Dest] DO
      IF minQueue > QueueSize(NextNodeTable[Dest.Adj]) THEN
        minQueue := QueueSize(NextNodeTable[Dest.Adj]);
        SPadj := NextNodeTable[Dest.Adj];
      END;
    END;
  ELSE
    SPadj := NextNodeTable[Dest.Adj];
  END;
  IF Flag= False THEN                 (* Algorithm A only *)
  IF input packet THEN                (* Algorithms B and C only*)
    IF QueueSize(SPadj) > Threshold
      IF AltNextNodeTable[Dest.Adj].count > 1 THEN
        minQueue := MaxUnsigned;
        FOR each Adj of AltNextNodeTable[Dest] DO
          IF minQueue > QueueSize(AltNextNodeTable[Dest.Adj]) THEN
            minQueue := QueueSize(AltNextNodeTable[Dest.Adj]);
            Altadj := AltNextNodeTable[Dest.Adj];
          END;
        END;
      END;
      IF QueueSize(Altadj) < QueueSize(SPadj) THEN
        adj := Altadj
      ELSE
        adj := SPadj
      END;
    END;
  ELSE
    adj := SPadj
  END;          (*Algorithms B and C only *)
  END;          (*Algorithm A only *)
END; Forward (Algorithms A,B and C)
```

Appendix F

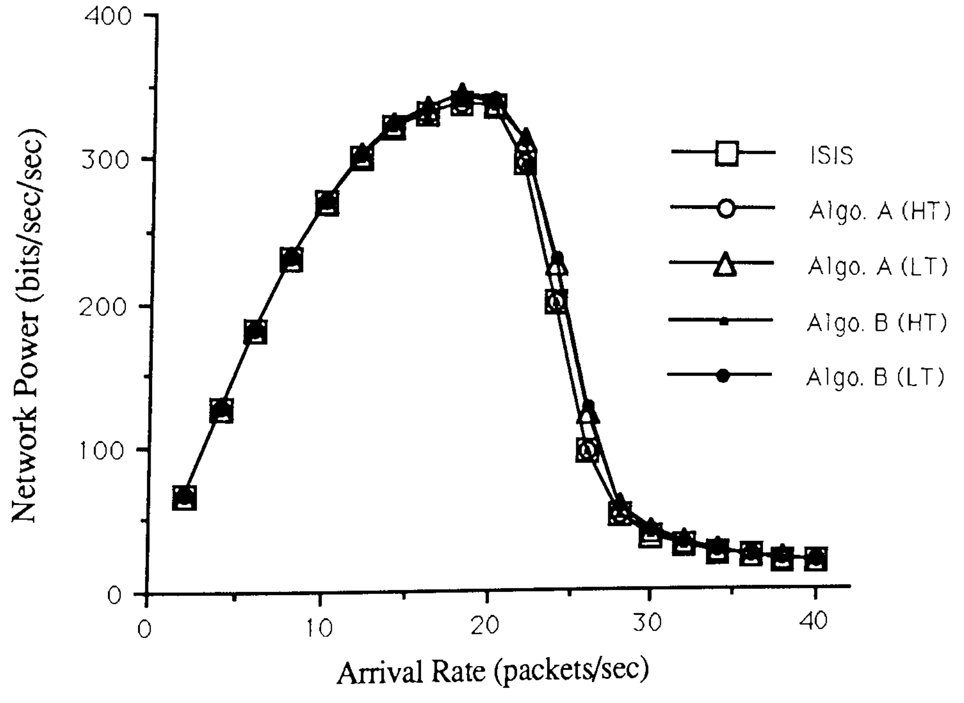
Additonal simulation results for ISIS and algorithms A,B and C.



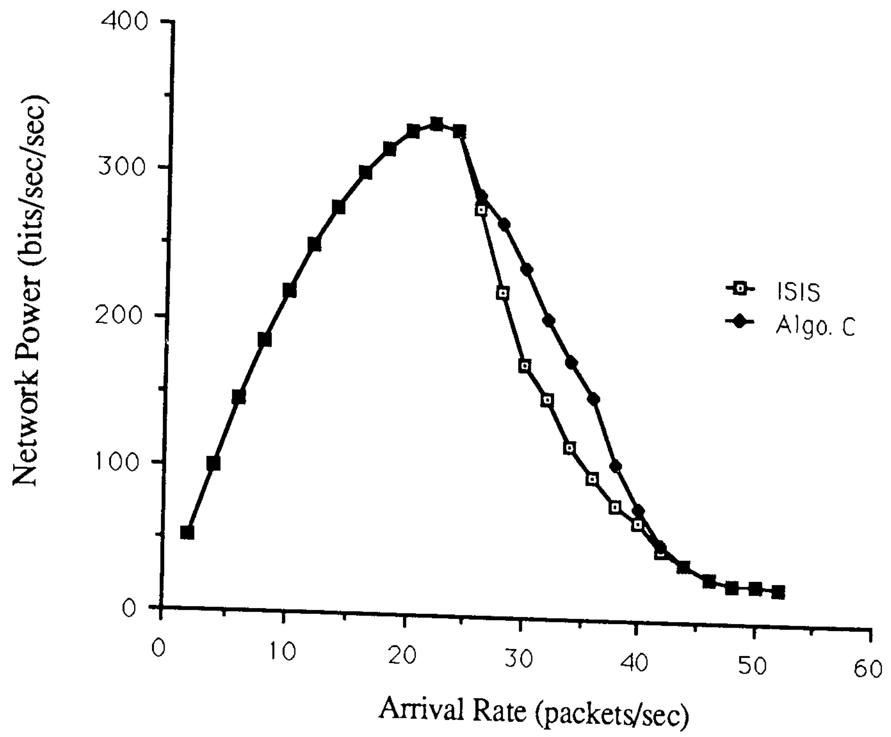
Network power against arrival rate for ISIS and algorithms A and B
(8 nodes, 11 links)



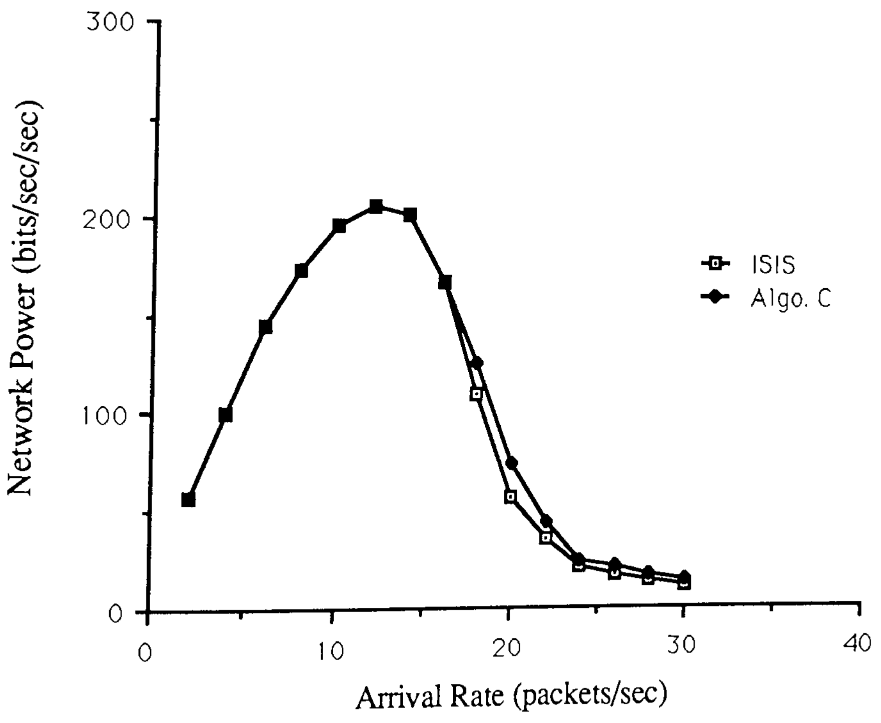
Network power against arrival rate for ISIS and algorithms A and B (12 nodes, 14 links)



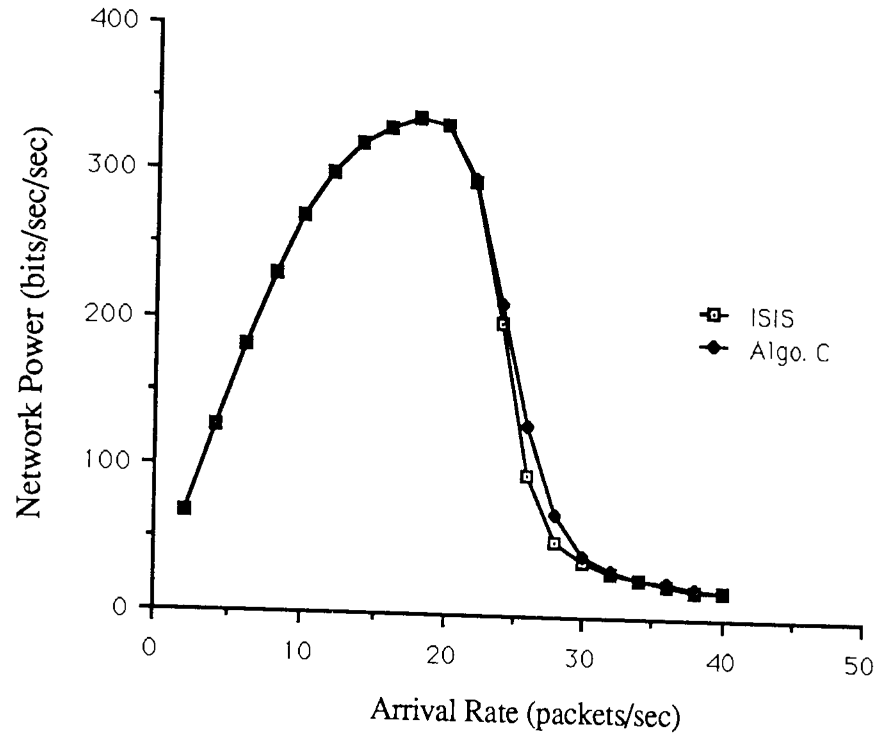
Network power against arrival rate for ISIS and algorithms A and B (14 nodes, 19 links)



Network power against arrival rate for ISIS and algorithm C
(8 nodes, 11 links)



Network power against arrival rate for ISIS and algorithm C
(12 nodes, 14 links)



Network power against arrival rate for ISIS and algorithm C
(14 nodes, 19 links)

Appendix G

Shortest path, alternative path and forwarding procedures for routing algorithms A+DSPS, B+DSPS and C+DSPS.

```
PROCEDURE SHORTEST PATHS (Algorithms A,B and C + DSPS)
BEGIN
   TENT := [0]; PATHS := [0]
   place (SELF,0,0) in PATHS
   FOR each Adj of SELF DO
      place <P,x,{Adj(P)}> in TENT
   END;
   WHILE TENT is nonempty DO
      get <P,x,{Adj(P)}> from TENT with x = minimum
      remove <P,x,{Adj(P)} from TENT
      add <P,x,{Adj(P)} to PATHS
      FOR each P, P ∈ Adj(P), DO
         FOR each N of P DO
            dist(P,N) = d(P) + metric(P,N)
            IF <N,d(N),Adj(N)> ∉ PATHS THEN
               IF <N,x,Adj(N)> ∈ TENT THEN
                  IF x = dist(P,N) THEN
                     IF Adj(P) ∈ Adj(N) THEN
                        dsps(P) = dsps(P) + 1
                        place dsps(P) in <N,x,{dsps(N)}>
                     ELSE
                        place Adj(P) in Adj(N)
                     END;
                  ELSE IF x > dist(P,N) THEN
                     remove <N,x,{Adj(N)}> from TENT
                     remove <N,x,{dsps(N)}> from TENT
                     put <N,dist(P,N),{Adj(P)}> in TENT
                  END;
               ELSE
                  place <N,dist(P,N),{P}> in TENT
               END;
            END;
         END;
```

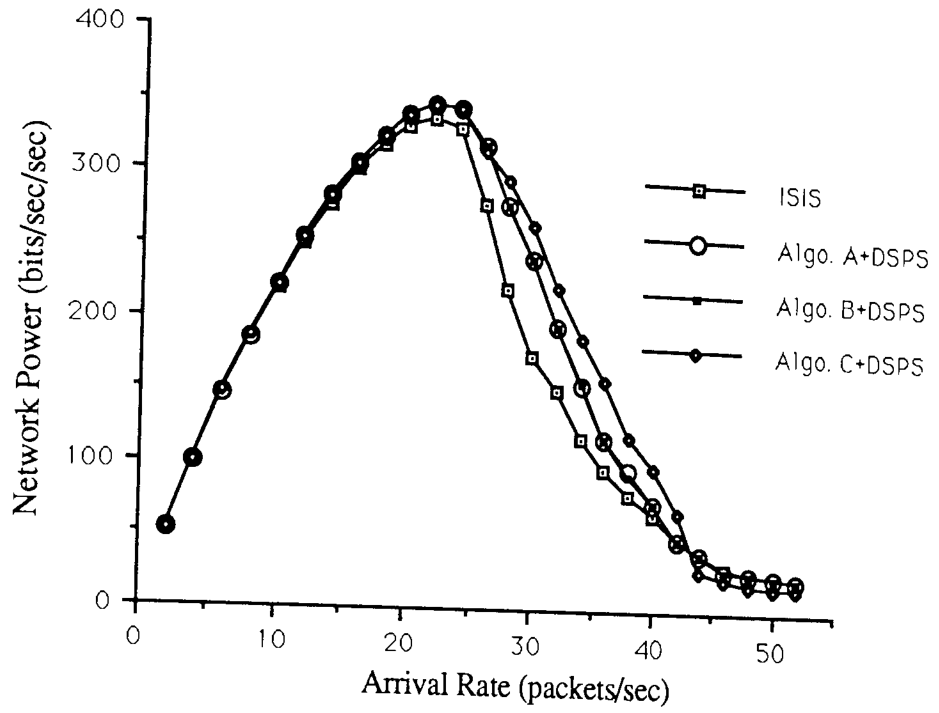
```
        END;
      END;
    END; Shortest paths (Algorithms A, B and C +DSPS)


    PROCEDURE ALTERNATIVE PATHS (Algorithms A,B and C plus DSPS)
    BEGIN
      IF Adj.count > 1 THEN
        PATHS[index] := [0];
        index := 1;
        FOR each Adj of SELF DO
          temp := Adj[index];
          Adj[index] := 0;
          SHORTEST PATHS
          PATHS[index] := PATHS;
          Adj[index] := temp;
          index := index+1;
        END;
        FOR each destination DO
          FOR each k DO                            (* algorithm C only *)
            FOR each index DO
              IF PATHS[index] = min, THEN
                temp := index
                place triple in SP[1]                (* algos A and B only *)
              ELSE IF PATHS[index] = min + 1 THEN    (* algos A and B only *)
                place triple in SP[2]                (* algos A and B only  *)
              END;
            END;
            place triple in SP[k]                  (* algorithm C only *)
            set triple in PATHS[temp] to MaxUnsigned
          END;                                     ( * algorithm C only * )
        END;
      ELSE
        SHORTEST PATHS
      END;
    END; Alternative Paths (Algorithms A,B and C + DSPS)
```

```
PROCEDURE FORWARD (Algorithms A,B and C + DSPS)
(given Dest, yielding adj)
BEGIN
  IF NextNodeTable[Dest.Adj].count > 1 THEN
    minQueue := MaxUnsigned;
    FOR each Adj of NextNodeTable[Dest] DO
      IF QueueSize(NextNodeTable[Dest.Adj]) < MaxBufferSize THEN
        dspsCount := DspsTable[Dest.Adj];
        IF minQueue > QueueSize(NextNodeTable[Dest.Adj]) - dspsCount THEN
          minQueue := QueueSize(NextNodeTable[Dest.Adj]) - dspsCount;
          SPadj := NextNodeTable[Dest.Adj];
        END;
      END;
    END;
  ELSE
    SPadj := NextNodeTable[Dest.Adj];
  END;
  IF Flag= False THEN                  (* Algorithm A only *)
  IF input packet THEN                 (* Algorithms B and C only*)
    IF QueueSize(SPadj) > Threshold
      IF AltNextNodeTable[Dest.Adj].count > 1 THEN
        minQueue := MaxUnsigned;
        FOR each Adj of AltNextNodeTable[Dest] DO
          IF minQueue > QueueSize(AltNextNodeTable[Dest.Adj]) THEN
            minQueue := QueueSize(AltNextNodeTable[Dest.Adj]);
            Altadj := AltNextNodeTable[Dest.Adj];
          END;
        END;
      END;
      IF QueueSize(Altadj) < QueueSize(SPadj) THEN
        adj := Altadj
      ELSE
        adj := SPadj
      END;
    END;
ELSE
  adj := SPadj
END;          (*Algorithms B and C only *)
```

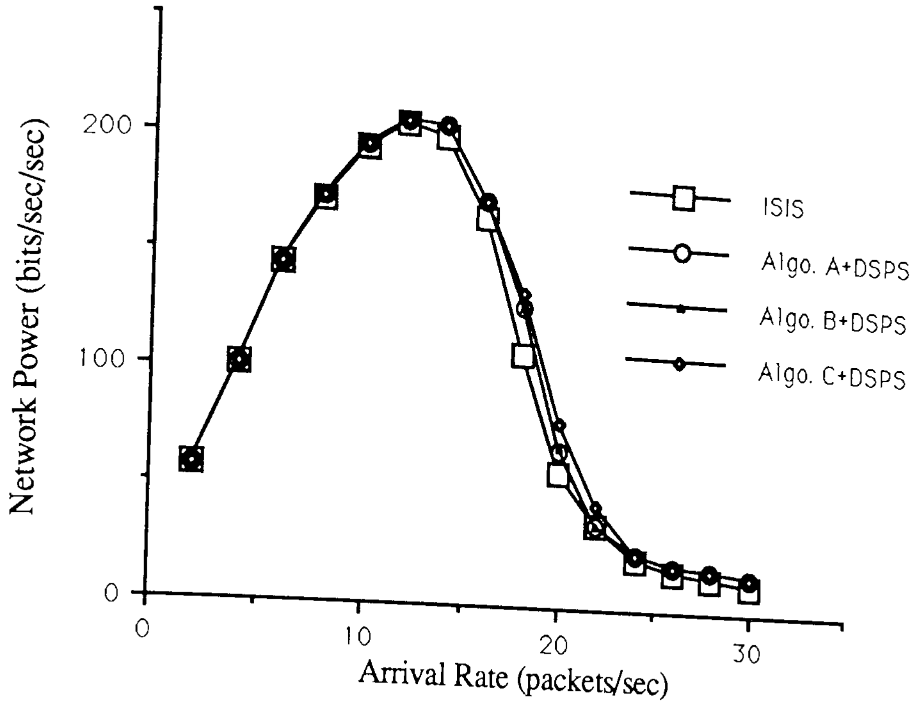
```
    END;          (*Algorithm A only *)
END; Forward (Algorithms A,B and C + DSPS)
```

Appendix H

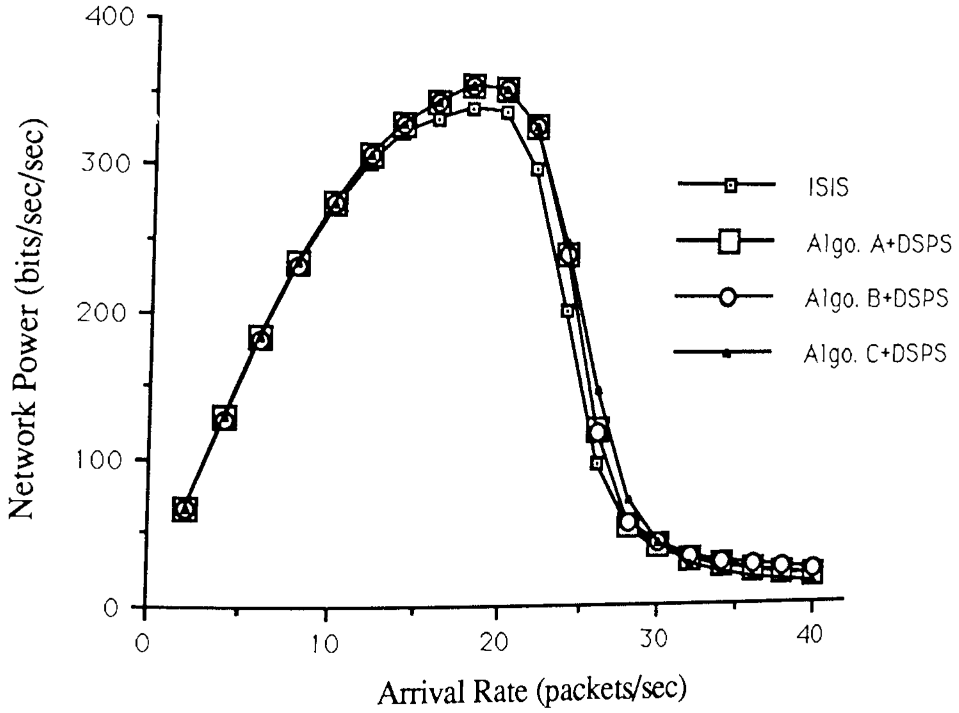
Additional simulation results for ISIS and algorithms A+DSPS, B+DSPS and C+DSPS.



Power against arrival rate for ISIS and algorithms A+DSPS, B+DSPS and C+DSPS (8 nodes, 11 links).



Power against arrival rate for ISIS and algorithms A+DSPS, B+DSPS and C+DSPS (12 nodes, 14 links).



Power against arrival rate for ISIS and algorithms A+DSPS, B+DSPS and C+DSPS (14 nodes, 19 links).