Simulation and analysis of adaptive routing and flow control in wide area communication networks

Nichols, S. J.

SIMULATION AND ANALYSIS OF ADAPTIVE ROUTING AND FLOW CONTROL IN WIDE AREA COMMUNICATION NETWORKS

S. J. Nichols, B.Sc.

School of Engineering and Applied Science,
Durham University,
South Road,
Durham DHI 3LE,
UK.

A thesis submitted in partial fulfillment of the Requirements of the University of Durham for the Degree of Doctor of Philosophy (Ph.D.).

March 1990

Abstract

Simulation and Analysis of Adaptive Routing and Flow Control Wide Area Communication Networks

S.J. Nichols

This thesis presents the development of new simulation and analytic models for the performance analysis of wide area communication networks. The models are used to analyse adaptive routing and flow control in fully connected circuit switched and sparsely connected packet switched networks. In particular the performance of routing algorithms derived from the L_{R-I} linear learning automata model are assessed for both types of network.

A novel architecture using the INMOS Transputer is constructed for simulation of both circuit and packet switched networks in a loosely coupled multi–microprocessor environment. The network topology is mapped onto an identically configured array of processing centres to overcome the processing bottleneck of conventional Von Neumann architecture machines.

Previous analytic work in circuit switched work is extended to include both asymmetrical networks and adaptive routing policies. In the analysis of packet switched networks analytic models of adaptive routing and flow control are integrated to produce a powerful, integrated environment for performance analysis

The work concludes that routing algorithms based on linear learning automata have significant potential in both fully connected circuit switched networks and sparsely connected packet switched networks.

Declaration

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

In the course of this research the following were included in an approved programme of advanced studies:

1. Advanced lectures and examination in the subjects of Digital Systems, Digital Signal Processing and Semiconductor Devices, October 1986–July 1987.
2. A working visit to the Network Services Division of Reuters Int., London, September–October 1988.

Simon Nichols, March 1990

Copyright

Acknowledgements

First and foremost I would like to express my thanks and appreciation to my supervisor Professor Philip Mars for his guidance, encouragement, enthusiasm and support throughout this project.

Many people have contributed to this work through their discussions, support and encouragement. My thanks to all those who have helped, notably Dr. Ian Mc Phee during his temporary lectureship at Durham University, Dr. Frank Kelly and Richard Gibbens, both of Cambridge University Statistical Group, not forgetting all my friends, colleagues and aquaintances who have made my time at Durham a thoroughly enjoyable experience.

My thanks also to Brian Lander and his staff at Durham University Computer Centre whose services enabled much of this work to take place. Finally I would like to thank the Science and Engineering Research Council and Reuters International for their financial support.

This work is dedicated to my parents

Table of Contents

Chapter 1 Introduction

Chapter 2 The Design of a Distributed Simulation Environment For Wide Area Communication Networks

Chapter 7 Conclusions And Further Work

Chapter 1
Introduction

1.1 Communication Networks

Whenever a large number of geographically, or even locally, distributed centres wish to communicate with each other, a direct link between each party quickly becomes impractical. To provide such a communication facility networks are constructed, the nodes of which provide centres for users of the network to connect terminal equipment. A classical example of such terminal equipment is the humble telephone with which two suitably equipped sites can communicate in any part of the world. Because initially all network traffic was voice based, the first telephone networks using analogue technology, were designed specifically for that purpose. The basic characteristic of these networks is that a dedicated frequency band is reserved for the duration of any call between two users. With the advent of modern technology digital techniques have been introduced into the design of the network switching nodes. This has expanded the range of applications for networks. Previously binary data could only be transmitted using multiplexers/demultiplexers (MODEMs) at the terminating connections over standard channels. Digital switches allow much greater capacity channels to be allocated and provide a greater range of potential services to the users. Today it is possible to transmit both voice and data based traffic over the same network. However the basic characteristic of dedicated bandwidth remains unchanged and the technique is given the name circuit switched to reflect this fundamental idea.

This change to digital technology has led to another, more fundamental, change in networks and the transmission of data across them. Since binary data in particular does not have to be transmitted in real–time, dedicated bandwidth no longer has to be made available at all times. Instead bandwidth can be made available dynamically on request by the data itself. If local bandwidth is temporarily unavailable the data can be stored locally within each node. These networks were originally termed message switched, the key feature being that the message was retained as a single stream through the network. More recently the technique of dividing the message into a number of smaller basic transmission units called packets, before passing it over the network, has been widely used and the term packet switched used to describe the resulting networks.

Natural evolution in this digital environment has led to attempts to combine circuit and packet switched networks to give a single medium for voice or data transmission, matching the most efficient manner of transmission to the needs of the traffic. This involves the ability to both assign bandwidth for the duration of a circuit orientated session with the flexibility to dynamically assign the remainder of the link capacity to packet based transfers. Systems of this type fall under the broad title of Integrated Services Digital Networks (ISDN). More recently the use of optical fibre communication link technologies has led to intensive research into a new generation of networks known as broadband ISDN. The enormous data rates now possible suggest a whole host of new applications such as real–time video transmission and new generations of sound and data transmission capabilities.

1.2 Adaptive Routing

To provide and manage the complex needs of network users sophisticated software techniques have had to be developed which shield the users from the rigours of the particular technology and transmission media being used to transmit the information between the source and destination sites. This has inevitably led to the definition of standards for network architectures to introduce consistency and define interfaces for the design of network access equipment. However these standards are not definitive and allow a degree of freedom in the design of many of the key network features. One important example of such a feature is the routing strategy within the network by which information at a source or traffic originating node is directed to its destination.

The path by which information is routed over a network can have a profound affect on the performance of the network. If the network is considered as a limited pool of distributed transportation components, the efficiency in the utilisation of these components depends on the distribution of the traffic over each of the sites. Invariant routing techniques, formulated using projected traffic forecasts may be simple to implement but suffer from serious drawbacks. Sustained traffic fluctuations or component failures may compromise network performance by overloading sections of the network while leaving other areas relatively lightly loaded. Equally, static routing techniques, if they are to remain efficient necessitate, by definition, regular recalculation off line at each major event in the networks lifetime, such as new subscribers or component installation.

Alternatively an algorithm which can automatically adapt to such changes can be proposed. This is achieved by modifying the routing tables, based on current network status in accordance with some performance index. In this way traffic patterns will automatically be adjusted in the event of a new traffic injection or capacity modification by nothing more than the secondary effect of the accompanying change in performance in the affected areas. Of particular interest in this work is the potential use of simple distributed routing strategies based on a class of algorithms known as linear learning automata. In this thesis investigations into their suitability for implementation in both circuit and packet switched networks are carried out with comparisons of their performance against other popular strategies and mathematical bounds.

1.3 Network Performance

To investigate the potential benefits in the introduction of adaptive routing strategies and compare and contrast different adaptive policies, a performance environment is required. This can be provided using one of two techniques. The most elegant method is the formulation of mathematical models of the communication networks for each strategy. These can then be solved to give exact solutions for predicted network behaviour, subject to the approximations invariably made in the models. When the solutions to these models prove too computationally intensive or the approximations made are considered unrealistic another technique must be used.

The second method of performance evaluation is the construction of a discrete simulation model of the communication networks. This has the advantage that a more faithful reproduction of the target network can be constructed. Alternatively it does not produce the exact performance figures of a mathematical model and a mean figure and confidence interval must be extracted from multiple measurements using the simulation model. This unfortunately leads typically to long and therefore expensive execution times for these simulation models. As the size of the network, the traffic intensity over the network and the complexity of the simulation models all increase conventional simulation becomes increasingly costly. The basic problem stems from an inherent mismatch between the sequential, Von Neumann processor architecture and the highly parallel simulation problem. In this work a way forward is suggested using a new multi–processor simulation environment designed to exploit this parallelism.

1.4 Outline of Thesis

Following this chapter the rest of the thesis is arranged into six further chapters 2 to 7. Chapter 2 discusses the design and implementation of a new multi–processor simulation environment for wide–area communication networks. Initially previous work in this field is outlined and discussed with particular attention to limitations and bottlenecks in the proposed architectures. The heart of the new environment the Transputer and its associated programming language Occam are then introduced. Building on both the previous work and the Transputer as a parallel processing component, the hardware and software specification of the new simulation environment are then presented. The key concept is the mapping of the network topology onto an identically configured array of processing components. The design of a nodal architecture for the software of such a processing component is presented. In particular the steps taken to prevent such code deadlocking are discussed. Special problems associated with distributed simulation are identified and the solutions used in the work presented. These include synchronisation and access to the simulation. Finally the range of ways in which a network topology can be mapped onto the processing surface are investigated with specific examples.

Chapter 3 concentrates on circuit switched networks and the performance modelling of routing strategies in fully connected networks, an important subset of the possible topological configurations. The impact of VLSI and digital technology on circuit switched networks is explained, concentrating on the areas relevant to this work, specifically signalling and network architecture. The advances in these areas have allowed much greater flexibilty in the way in which a path through the network can be established. To take advantage of this a number of different mechanisms for path selection have been proposed or implemented. To clarify these schemes a classification system is introduced and the algorithms identified within its structure. In addition the concept of flow control is introduced to avoid excessive use of inefficient paths and the relationship between routing and flow control discussed within the selection process of a path through the network. Following this the simulation and analytic models that are used to analyse some of the more interesting schemes both with and without flow control are presented. The simulation model, based on a simple signalling protocol, is designed specifically for implementation in the environment discussed in chapter 2. The analytic work presents new work based on a well known simple, symmetrical, Poisson model. Extensions to the model allow the analysis of adaptive strategies with well defined mathematical behaviour for asymmetrical trunk and traffic distributions for both single and multiple traffic sources.

Chapter 4 reports on the application of the analytic and simulation models to fully connected circuit switched networks. Initially simple, analytic, single source experiments are carried out to investigate the equilibrium behaviour of a range of strategies, including some of the learning automata based algorithms, to highlight the differences or similarities in behaviour. Having established this base more complex analytic models, involving the interaction of traffic sources are examined. Criteria for the structured introduction of asymmetry are outlined to facilitate the comparison of different networks. In addition the accuracy of the analytic model itself is examined, based on the underlying assumptions implicit in its formulation, and criticisms made on its validity over an important range of operational values. Both simulation and analytic models are then applied to networks with and without flow control using different strategies. Two main features are examined in each case, performance and instability. Performance analysis uses average blocking probability and individual grade of service as indexes. The instability work is based on previous analytic work which has suggested the existence of multiple solutions for the performance of fully connected, alternatively routed networks over a certain range of traffic loads.

Chapter 5 discusses the development of packet switched networks and the analytic work done on the performance evaluation of different routing strategies used by these networks. The basic mechanisms of a packet switched network are defined and discussed in the context of the evolution of layered architectures to provide a transparent transmission medium across the network for user applications. Describing the functions of the lower layers, those most important to the network rather than the specific application, the summary concentrates on the network layer, that part of the architecture responsible for routing and flow control. As in circuit switched networks a plethora of different approaches to the routing problem exist. A classification of routing strategies is carried out to categorise the major approaches and techniques. This is followed by a comprehensive and critical overview of existing and proposed routing techniques. Whenever possible analytic models which have been developed to investigate strategies are summarised. Analagous with circuit switched networks, packet switched networks also need flow controls to limit entry of traffic on a network. Following a brief classification of the different positions in which flow control can be administered in the architecture, a survey of the main techniques is performed. Once again a summary of mathematical models is included and the major findings reported. Finally in this chapter the specification for the packet switched simulation package constructed for execution on the multi–processor environment is presented. It is provided in the form of a series of flow diagrams outlining the major component functions of the network architecture reproduced by the simulation.

In chapter 6 a series of experiments are presented on the application of a selection of routing strategies and flow control procedures in packet switched networks. A number of the analytic models for the performance evaluation of these strategies, described in the previous chapter, are implemented using iterative techniques based on shortest path routing and window based flow control schemes. These models are first applied to a series of introductory experiments using a simple network configuration. The results are compared with simulations of the same network configurations using the model also detailed in the previous chapter. This enables the validity of the mathematical assumptions made in the formulation of both the generalised network model and the additional assumptions made in the analysis of some of the adaptive work to be evaluated by comparison of the two sets of results. A more complex and representative network is then presented and a series of detailed experiments formulated to investigate the performance of learning automata routing and delay based flow control. Optimum routing and flow control models are used to provide an upper bound, while traffic insensitive procedures provide a deterministic lower bound on the performance of the strategies. Finally a special case of network is analysed, using an asymmetric traffic matrix to investigate the performance of an international network under the same variety of adaptive and non–adaptive policies.

Finally chapter 7 draws together the major findings and conclusions from this work. The chapter is broken into three major areas, the design and implementation of the multi–processor simulator on an array of suitable configured Transputers, the application of dynamic routing and flow control in fully connected circuit switched networks and the application of the same class of strategies to sparsely connected packet switched networks. In addition, based on the findings of the work done, future avenues of promising work are suggested, existing bottlenecks and unresolved problems are identified and the results of this work are discussed in relation to their applicability to the performance of real networks.

Chapter 2
The Design of a Distributed Simulation Environment for the Performance Evaluation of Wide Area Communication Networks

2.1 Introduction

This chapter presents the design of a dedicated simulation environment for the performance evaluation of wide area communication networks. The architecture of the simulator exploits the parallelism available in network simulation to maximise the execution rate of the simulation models. To do this the simulator adopts a multi–processor approach. Multi–processor architectures have already attracted considerable attention for such work and some of the more notable attempts to construct such simulators are first outlined. This takes the form of a short review section which includes some comments on each authors approach, its applicability and limitations. A new tool for parallel simulation, the Transputer, is then introduced and the special features which suit it for such a task are discussed along with the form of its natural programming language, Occam. Following on from these summaries the hardware and software structures of the simulation environment and operator interfaces are outlined. Included in these sections are discussions on some of the major difficulties in writing a simulation package for execution in a distributed environment. Constructions which were adopted to solve problems such as synchronisation and data access are presented along with additional requirements to overcome inherent hardware limitations. Finally the variety of ways in which the software can be organised on the available hardware are presented. This flexibility arises from the power of the Transputer to act as a building block for parallel processes and allows the operator a number of options. The option of which software and hardware configuration to adopt depends on the availability of Transputer boards and the requirements of the operator.

2.2 Distributed Simulation Architectures

2.2.1 Review of Previous Architectures

It is generally recognised that the evaluation of communication systems by analytic means alone is inadequate for the majority of criteria sought in modern, large complex networks[1]. Fortunately digital simulation techniques provide an alternative method by which performance figures can be derived. However conventional simulation programming environments provide a poor base on which to model highly parallel network environments. The limitations imposed by Von Neumann architectures on the performance of simulation models for communication networks has led to interest in the development of alternative, dedicated environments. Early work in the closely related area of road traffic analysis led to the construction of special purpose hardware simulators[2] using digital circuitry to implememt the various elements of the queueing system. However in recent years there has been considerable advances in microprocessor technology and software tools, languages and techniques. These advances combined with the flexibility of software modelling has led to the majority of work concentrating on the development of distributed arrays of processors which can share the simulation workload. A number of different approaches have been investigated for the construction of these distributed environments to optimise performance and/or flexibility. In this section some of the major architectures which have already been developed are discussed and attempts are made to assess their relative strengths and limitations.

Before considering architectures developed specifically for the purpose of network simulation it is worth mentioning a more general approach proposed and adopted by M. Papazoglou. In his paper[3] he outlines a 'parallel Simula machine' for the concurrent execution of processes defined within Simula–67. Such instances of parallelism are determined by a pre-processing stage. The machine then operates a master–slave topology with a central microprocessor controlling a series of satellite processors with both common and private memory. A complex arrangement of interrupt controllers and bus structures allows process interaction with other satellite processes and the central controller. The master processor synchronises the processes, allocating tasks to them and adjudicating when contention arises between two or more of the tightly coupled microprocessors.

Although no performance figures are quoted, the method of farming out processes and the use of both bus and shared memory architectures in an interrupt driven environment shows promise if a sufficiently intelligent kernel accompanies the environment hardware to direct the processes. However for the specialised task of network simulation, a more specialised approach to the architecture question has been developed by many authors, creating a 'test–bed' for evaluation by simulation.

Within the subject area of 'test–bed' design two distinctly different approaches have received attention. In the first approach the act of simulating the network is divided into major tasks. Special purpose units are constructed for the implementation of each of these tasks. These tasks are then undertaken by programming each unit in languages not dissimilar to special purpose simulation languages like GPSS and SIMSCRIPT. These modules and their interconnection define the machine architecture. This approach has been adopted specifically in the development of two special purpose simulators for the analysis of queueing networks at the Technical University of Aachen[4,5]. The first simulator divides the simulation into the tasks of random number generation, queueing network management and statistical calculation of the network state. These modules are arranged in a pipeline and discrete time techniques used for the simulation clock allowing the definition of a simple language for network construction and definition. The construction of this specialised environment allows the efficient analysis of simple queueing networks, although for more general problems the architecture lacks flexibility and for large systems the queueing module would doubtless produce a processing bottleneck. In its favour, as the authors point out, it provides a cheap alternative to expensive mainframe computation for this class of problem with comparable performance.

The second architecture developed at Aachen, some time later, divides network simulation into a similar set of tasks, but implements each task in a more distributed fashion. The simulator now contains arrays of special units, each array dedicated to the task of either list processing, random number generation or statistical analysis arranged in a mesh structure and controlled from a single simulation execution unit connected to all list processing modules. Each module is implemented using a specialised processor for each particular task, connected to other modules between which data streams arise during execution of simulations. The structure represents a hybrid between the Simula-67 special purpose machine mentioned earlier and the previous network analysis environment developed at Aachen. The decomposition of tasks continues to divide the simulation into random number generation, list processing and statistical analysis but now uses a single processor to control the progress of the simulation and arrays of interconnected special purpose modules to which tasks are delegated. Performance evaluation of the architecture, implemented in TTL, NMOS, etc. technology produced approximately 0.5 MIPS which compared favourably with figures achieved on a mainframe for a fraction of the cost. Interestingly the author claims at the end of the article that improvement in performance depended not now on the exploitation of further parallelism within the simulation but on the clock rates and transmission rates of available technology. Further performance improvements up to 10 MIPS were suggested if the clock rate could be increased by an order of magnitude.

The second major approach seeks to exploit the parallelism in communication network simulation by constructing arrays of connected microprocessors which each undertake the task of simulating a single node of the network. In this way an N node network is modelled by N coupled microprocessors. The control and interconnection of these processors has been tackled in a number of different ways enabling the construction of cheap, useful, 'test–bed' networks for distributed simulation. R.Kain et al[6] developed just such a structure as a tool for distributed system design. Serial links connect nodal microprocessors in a topology identical to the network under consideration. Each of the nodes is also attached to a single control node by a shared bus. This master node controls the timing, network modification and result gathering via the bus.

Constructed using 'off–the–shelf' components and modules the system has been designed to maximise its flexibility and minimise cost rather than to optimise performance. Typically the basic unit is a 6502 processor board and nodes are built up using an intra–node bus to connect serial interface boards and additional memory.

A similar master/slave architecture has also been developed for traffic simulation by A. Toda et al[7]. NEWTS (Network Traffic Simulator) uses a two layer hierarchical common bus structure to connect the individual microprocessors. This enables a degree of concurrency in communication over the bus structure as each local bus can operate independently, allowing the transmission of information between two nodes attached to the same bus. In addition communication between two nodes on different buses can also take place via the global bus. However contention still arises when more than one pair of nodes wishes to access the global bus. Seperate control lines to and from each node lead to a master node which implements the system clock using a simple stop–and–wait protocol.

Finally an interesting variation on the interconnection problem is suggested by M. Geary[8] using a grid array of nodal processors each connected by bi–directional links to their four neighbours. Each processor has a central CPU and four surrounding buffers with a bus structure which can route transmissions to, from and around the CPU. Using the buffers to queue calls any message can be transmitted across the grid from any node to any other node allowing a great deal of flexibility in the topologies that can be investigated with appropriate routing tables.

2.2.2 The Transputer and Occam — An Overview

In 1985 Inmos launched its attempt to fulfil Barrons prophesy made almost a decade earlier[9]. In his discussion paper on distributed computing he made the observation that the effective implementation of distributed techniques would require a radical change in the architecture of computers, the design of their programming languages and the style of programming. Inmos's offering, the Transputer, developed in asssociation with its natural programming language Occam, allow the design and implementation of multiprocessor solutions for a wide range of applications from image processing to distributed databases, [10–14,17]. The rest of this section describes the design of the Transputer, concentrating on the features which have been included to optimise it for use as a component in a multiprocessor array, and the language which allows such systems to be easily realised.

The Transputer family currently consists of the T2, T4 and T8 series chips. These chips consist of a 16 bit processor, a 32 bit processor and a 32 bit processor with floating point unit respectively. More detailed information can be found in product overviews[15] and other Inmos reference material[16]. Each Transputer is a complete microprocessor implemented as a VLSI chip using 1.2 micron technology[17]. The chip integrates processor, a small amount of internal memory, external serial communication links and a comprehensive set of signal lines to its external interface A block diagram identifying these major sections and the main bus connections between them is shown in figure 2.1 The external memory interface can be used to access up to 4 Giga bytes of external RAM and comes with a set of configurable timing signals.

To enable efficient communication between Transputers each chip has four, high speed, autonomous, link interfaces[18]. A link consists of two unidirectional connections which provide serial point to point communication between Transputers using DMA with processor cycle stealing to minimise overheads. Communications over a single link consist of asynchronously transmitted byte transfers. Confirmation of each bytes arrival is provided by small acknowledgement packets which are interleaved with data packets on the same link in the opposite direction.

As well as providing links for concurrent communication with other Transputers, allowing communication between concurrently executing code, a single Transputer also has the ability to internally emulate the execution of concurrent code. A microcode driven scheduler maintains a list of both active and inactive processes where each process requires concurrent execution. Each active process is then time–sliced to share the processor and remains on the active list until it is required to communicate with another, unready process or is placed in a wait state by a specific delay action when it is made inactive. In this way processes waiting for communication or in specially constructed delay states do not waste processor time and remain idle until reactivated. The microcode instruction set itself is very compact and has been compared to RISC architectures. The chip supplies only three arithmetic registers in the form of an evaluation stack, removing the need of operators to define registers and allowing a large range of operations to be executed using single byte instructions.

To facilitate the programming of multiprocessor systems and the Transputer in particular, a new parallel language was developed. The design of the new language, Occam, was based on C.Hoares concept of Communicating Sequential Processes or CSP[19–21] which provides a model for concurrency and communication in a concurrent environment. The basic element of the language is the process. A process simply performs a series of actions and then terminates. The design of Occam and the Transputer were linked so the Transputer could directly implement the concept of an Occam process. Simple processes are then combined by structures known as constructors into larger processes which themselves are linked by further constructors to form larger processes still and so on building into a hierarchical program structure. The language is introduced in a paper by its designer D.May[22] and largely reprinted with some additions to highlight its use as a design formalism in a later joint paper[23]. The present state of the language, Occam2, is outlined in an introductory tutorial available from Inmos which includes a formal language definition[24].

It is possible to divide Occam into two parts, those features which the language uses to define parallelism and communication between parallel processes, and those which are used to direct sequential processes. Before highlighting the primitive (i.e. fundamental) processes and constructors which allow Occam to support concurrent code execution, it is useful to outline the range of constructs which are more familiar to the programmer from sequentially based languages. As well as supporting assignment (the first primitive process), Occam supports conventional sequential code execution by the use of a SEQ construct and branching for conditional execution of processes by using the IF construct, with boolean conditions preceding each process. The structure of these constructs is shown in figure 2.2(a). The SEQ construct executes the following processes from top to bottom and then terminates. The IF construct executes the process associated with the first boolean statement which is evaluated as true and then terminates. Again the order of evaluation is top to bottom.

Equivalent constructs for the execution of concurrent processes can be identified, but first the concept of a channel must be introduced. A channel is a one way, point to point communication link between two parallel processes. Two primitive processes are used to communicate over the channel, the output process which offers information to the channel and the input process which accepts it. Communication between the two processes is also synchronised, taking place only when both processes are ready. The two constructs analagous to the SEQ and IF constructs, using channels are the PAR construct and the ALT construct respectively. The PAR construct, short for parallel, defines each of the following processes to be concurrent with the other processes in the construct. When all component processes of the PAR construct have terminated the PAR construct terminates. Communication between processes, for synchronisation, data transfer or both is carried out exclusively by channels as variables may not be shared between components of a parallel process. The ALT construct, short for alternative, has a number of channel input processes with a further process associated with each input. The first channel to synchronise with its corresponding output process completes its communication, its associated process is then executed and the whole process terminated. The analogy between the channel input process and the boolean construct in the IF process is evident. The form of the concurrent constructs are outlined in figure 2.2(b).

In addition each of the constructs can be used to form a construct replicator, equivalent to a list of processes with suitable subscripts under the constructor. This is particularly useful for the manipulation of arrays of variables or channels. Finally a WHILE statement is included to provide a loop construct with termination on the evaluation of a boolean flag as false, similar to the PASCAL command.

2.3 Simulator Software Organisation

2.3.1 Environment Overview

The design of the simulation environment was based on a simple six stage conceptual model. The first stage is initialisation of the system. The next three stages cooperate to modify, simulate and retrieve information for the duration of the simulation. Finally the information retrieved must be stored for future analysis and possibly used at that time for on–line reporting. During the initial discussions on the development of the simulation environment, in addition to this basic breakdown, it was decided an additional important consideration should be the ability of the final product to be used both as a research tool and also as an instrument for network investigation. In its second capacity an important attribute the environment must possess is the flexibility to allow an operator to construct and analyse conditions of interest without detailed knowledge of the products internal implementation or operation. To accomplish this input and output interfaces were constructed around the main simulation process. These interfaces can then pass information to and from the simulation in a formally defined format, freeing that portion of the code from direct communication with the operator. The interfaces can then be separately designed to present and accept information to and from the operator with as much flexibility as desired. Figure 2.3 shows how the conceptual model is mapped onto the two interface processes and the enclosed simulation process by the definition of each process as one of three components of a PAR construct. It also shows the major interprocess communication during the simulations progress and the final post processing stage carried out external to the environment. This is done to take advantage of various graphical and analytic software readily available.

Based on the preceding description, each of the interface processes can be further broken down into the areas of initialisation, the period during simulation and the period immediately after simulation. The input interface, has a number of processes which cope with the demands of the operator and the environment during each of these periods. The first process allows the operator to construct a network topology and furnish all the necessary information for the particular network simulation desired, i.e. capacities, traffic, operation parameters, protocols etc. When the simulation definition stage is complete this process then transfers the portion of information required by the result processing stage, for its on-line calculations, to the output process. It then uses the operators definition of the network topology to configure the simulator hardware into the necessary simulation topology (see section 2.4) and loads each Transputer card in the topology with its relevant block of code, according to its function in the simulation (see section 2.6). Finally when the simulation software has been successfully exported to its hardware the the process initialises the simulation and terminates.

Two processes, running concurrently, make up the second component of a sequential construction within the input process. The first of these processes undertakes the control and implementation of modifications to the simulation process. It has two major functions, first to interact with the operator allowing him to define a set of interruptions in the simulation through which modifications can be introduced, and secondly the transmission of these interruptions at the appropriate time to the simulation process. The modifications range from a simple pause in the simulation, through extraction of results, to the modification of traffic levels through the network or even variation in network topology by the simulation of component failure. Injection of these modifications to the network is done by periodically synchronising with the simulation process at the time scheduled for the next interruption and transmitting modifications labelled for that time.

The second process runs concurrently with the modification process. It accepts input directly from the keyboard and passes it to the modification process. However it also communicates with the result processor both during and after the simulation. This is necessary because the operator's input has to be divided between the modification process residing in the input interface and the on-line analysis residing in the output interface. It acts as a simple input demultiplexor, accepting input and forwarding it to one of these two processes according to the purpose of the instructions.

The output processes, the complement of the input process, can be considered as a combination of three simple processes arranged in the same form as the input process The initial process collects data on the particular network under investigation. This allows a number of network parameters to be defined for subsequent use in the calculation of network statistics. The remaining functions can then be most simply divided into the two main areas of data collection and manipulation. When the first process receives data from the simulation process in the form of a result dump it is filed for future analysis and a copy passed to the second process. This second process then performs any on-line processing required and together with instructions relayed through the input interface to the output interface, maintains a graphical and numerical display of the simulations progress up to its last reported state.

Finally, between these two processes is the simulation process. In the next section this process is considered in more detail. In particular a second layer of concurrency is introduced within this process to optimise execution time.

2.3.2 Simulation Process

The simulation process which lies at the heart of the software model interfaces to the input and output processes via single channels into and out of the process respectively. Internally the simulation process models a N node network by the construction of N identical parallel processes known as nodal processes. These processes are connected by pairs of channels to allow two way communication and organised so that the resulting topology reproduces the topology of the network defined by the operator. Figure 2.4 shows the simulation process as it appears to the interfaces and its internal decomposition into the nodal processes and connecting channel structure for the simple example of a 5 node fully interconnected network. Two of the nodes in the network have a single additional channel each. These map onto the external input and output interface channels and act as communication gateways for the network.

In general sequential simulation times are functions of both network size and average traffic density at a node. By dividing the simulation process into N parallel processes, a degree of concurrency equal to the size of the network has been produced. If fully exploited by the accompanying hardware the execution times of network simulations using this structure should theoretically be independent of network size. The rate of progress the simulator makes should then be dictated by the most heavily loaded nodal process, as other processes cannot proceed at a greater rate than the slowest member of the structure if synchronisation is to be preserved. In reality the simulation rate is also dependant on the availability and fundamental limitations of the hardware on which the nodal processes are implemented and the full benefit of such a division is not always available. This is discussed in a later section, after the hardware environment has been introduced.

2.3.3 Nodal Process

Each of the nodal processes, configured in the software model to reproduce the desired network topology, replicates an identical process which executes a data processing algorithm used to simulate the operation of a network node. In addition it also contains two other processes which allow it to communicate with neighbouring nodal processes to accept, forward and transmit information through the network. The first process allows initialisation information, defining each nodes characteristics and local traffic details, to pass through and be accepted by each node. The second process accepts, implements and forwards modification information which may introduce changes into the network model and request network status updates. The relationship between the three processes is shown in figure 2.5. After each node has received its own initialisation information it alternately executes either the simulation process or the result access process, modelling the network or accessing the present network status for modification or extraction.

We now consider the core process within each of the nodal process which executes the call processing algorithm. The process has to meet two major conditions. First the process must implement the data transfer protocols defined for the connection, transmission and termination of calls across the network, according to the networks type and operational parameters. In addition each process must also support concurrent communication between themselves and their neighbouring processes in such a way as to avoid the introduction of conditions where two or more processes could deadlock. Figures 2.6(a) and 2.6(b) shows how the implementation of code in a single sequential process or the division of the process into three separate parallel processes, separating the input and output processes from the data processing stage, fail to meet the second of these requirements. In each case the inherent synchronisation of channel communication provides suitable conditions for deadlock between two neighbouring nodal processes. In the case of a single sequential process, two nodes may both be either waiting for input or trying to output to each other. Deadlock similarly occurs in the parallel implementation when two nodes wish to communicate to each other but find their reciprocal input process on each others node already occupied. The inability to complete the communication prevents these processes from accepting further information and leads to local deadlock. Both examples can easily be expanded to to larger examples involving a larger number of processes.

A solution to deadlock avoidance was constructed by adopting a nodal architecture of the form shown previously in figure 2.5. The simulation process has four parallel processes, arranged to form three subsidiary processses which communicate with the fourth through an ALT structure. The fourth process maintains the nodal data structure. The three interface processes control input to the process, transmission from the process and the internal generation of traffic for injection into the network. The input and output also multiplex and demultiplex calls from and to neighbouring nodes over input and output channels. Each of the three processes competes for the attention of the central process via the ALT structure if it is active, i.e. still has processing to perform. On selection by the central process data transfer takes place between the two processes. In the case of the input and generate processes, data is transferred into the central process, while the output process accepts data from the central process after initial exchange of an access token from the output to the central process. In this way if one of the input or output processes is waiting to synchronise over an external channel, the other two interface processes are free to communicate with the central process independent of the state of the third processes progress.

2.4 Simulator Hardware Environment

The simulation hardware can be separated into three component sections with two interfaces linking them together. The connection arrangements of the three components are shown in figure 2.7. The operator interface, an IBM PC–AT is linked to the simulator, a programmable array of Transputer cards, to form a bidirectional data path. In addition the simulator drives a separate high resolution graphical display unit which can be used to display information about the simulators progress. Returning to the operator interface, the PC–AT performs a number of functions throughout the simulation. Initially it holds the compiled simulation code on its hard disk and executes a kernel program to allow access to the code and the simulator environment. Functions within the kernel allow the operator to load the simulators component processes from the hard disk and allow certain of these processes access to keyboard input and the ability to direct output to the screen. Both the keyboard anf the screen are effectively treated as parallel processes running in parallel with the simulator. Furthermore the kernel also allows the simulator access to the DOS filing system to create files in DOS format and store data within them. These files then become accessible on termination of the kernel process.

The section of the hardware responsible for the execution of the simulation code can be further divided into two areas, a front end and a configurable array of Transputer cards linked by programmable crossbar switches as shown in figure 2.8. The front end is made up of several Transputer cards, in a fixed topology, including the special purpose INMOS B004 and B007 cards. The B004 card resides in one of the PC's expansion slots and contains a Transputer, one of whose links is interfaced via a serial–to–parallel converter to the IO bus of the host, providing the link used by the kernel program and the simulator. The B007 card contains a Transputer card onto which is memory mapped $\frac{3}{4}$ of a MByte of video RAM which drives a video controller which drives the display unit mentioned earlier. These cards and their neighbours in the fixed front end execute the input and output interface processes, leaving the actual simulation process to be executed on the second section.

Communications between the two sections of the simulator hardware are carried out over two further links connecting the fixed front end to a C004 switch, the first to control it and the second to send and receive data to and from it. The INMOS C004 switch is a 32 input by 32 output crossbar switch[25]. Programmable from a Transputer link it allows any of its 32 inputs to be uniquely connected to any of its 32 outputs, preserving the concept of point–to–point communication introduced by Occam and the normal connectivity of Transputer links by retaining a 1 : 1 mapping over the input and output sets. One C004 has sufficient inputs and outputs for a maximum of only 8 Transputers (4 links on each Transputer, with an input and an output on each link). So in order to provide sufficient switching capability to support 32 Transputers, a second set of 4 C004's are arranged to form the second layer of a two layer hierarchy using the primary C004 as a master controller.

Four links from the master C004 are used to control the operation of the slave C004s by forwarding instructions from the front end, through the master switch. Another set of four links connect the master C004 to each slave C004 to provide a data path between the front end and every slave switch. The remaining input and output pins of each slave are connected to the output and input pins of an array of Transputers, each of which sits in a card with 1 Mbyte of external RAM. The links on each of these Transputers are paired, links 0 and 1 forming one pair and links 2 and 3 the other pair. An input from each pair and its partners output are fed into each of the slaves, each of the four slaves accepting the same duo from each Transputer. This allows the C004s to be programmed in such a way that any Transputer in the array can be configured to communicate directly with any other Transputer (including itself) over either of the two pairs of links. For example a Transputer using link 1 can communicate with any Transputer over its link 0. Compare this with full connectivity where any Transputer could communicate with any other over any link pairings and this may seem restricted, however this arrangement provides an economic, flexible approach using the minimum of switching hardware.

2.5 Distributed Constructs

2.5.1 Master–Slave Constructs

When connecting a network of Transputers into a topology suitable for the mapping of an arbitrary network model a major consideration is the availability of the point–to–point serial links. At present each member of the Transputer family provides only four links. This poses a problem for highly connected networks where the number of immediate neighbouring nodes is considerably larger than the number of serial links provided. The solution lies in the formation of Transputer modules incorporating two or more Transputers in a linear chain. This provides a total of $2 + (2 \times n)$ links for an n Transputer chain. The link availability of a single Transputer and chains of two and three Transputers are compared in figure 2.9. In any one module, one of the Transputers is declared the master. This Transputer controls the execution of the nodal process and contains the bulk of the code to implement it. The remaining Transputers are referred to as slaves and execute small processes which multiplex incoming calls down a single link toward the master Transputer. In the opposite direction they also accept outward bound calls from the same link, demultiplex them and transmit them to neighbouring processes. Distributing the input and output processes over several Transputers in this way effectively provides the nodal process with the link availability it needs. In addition it does so without increasing the load on the master processor above that which would normally be applied, for the equivalent traffic density, over a network which could be accomodated by single Transputer nodal processes. However, it does tie up Transputer cards with the relatively menial task of forwarding messages, squandering much of its processing ability.

2.5.2 Interface Access

An important function of the simulation environment is its ability to interact with the operator as it progresses through the simulation run. This enables the operator to control the rate at which the simulation reports on its progress, access the results produced so far and introduce modifications. To do this the interface processes and the simulation process must establish a protocol to introduce, extract and incorporate information into and out of the simulation process at the correct time and place. This is done by first establishing a path through the simulation process. The path consists of a chain of nodal processes, linked at one end to the input process and linked at the other end to the output process, which passes through every nodal process once and only once. Using this path in conjunction with the access process on each nodal process, data can be introduced at the input process, travel down the chain and be finally received at the output process. At the end of every interruption of this form, the final data stream contains the time of the next interruption which provides the information necessary for each nodal process to synchronise its termination for the next interruption.

2.5.3 Sychronisation

An essential part of any simulation is the construction of a clock mechanism which can be used to schedule future events in the correct sequence. Broadly speaking clock mechanisms for discrete simulation techniques fall into one of two mechanisms, asynchronous discrete event and synchronous discrete event. Each mechanism characteristically contains list structures with forthcoming events and the times at which they are scheduled, this is known as the 'event list'. In an asynchronous mechanism the time at which any one event occurs is referred to as an event epoch and no two event epochs are allowed to occur simultaneously. The simulation typically jumps from epoch to epoch, skipping over the intervening periods in which no event has been scheduled. The alternative clock mechanism, using a synchronous discrete event mechanism divides simulation time into incremental steps. All event epochs falling within a step are grouped together and are assumed to have taken place simultaneously. The simulation progresses by incrementing the simulation clock at each step and scanning the list structures for scheduled events within that step.

In general an asynchronous discrete event approach is favoured in conventional simulations because of limitations in the flexibility and performance of synchronous implementations under certain conditions. In particular priority problems can arise when the order in which events falling in the same incremental slot are executed affects the course of the simulation. The limiting factor on the simulators performance arises as a product of attempts to increase the accuracy of the simulation by reducing the size of the incremental step. Increasingly this wastes processing time scanning data structures in which no events are currently scheduled, in direct contrast to the asynchronous approach which skips over such periods to the next epoch. In their favour synchronous schemes allow easy implementation of structures involving constructions of the '...if not done by...then do...instead' form, by simple checking of the data structure on each increment until either the condition is fulfilled or a sufficient time has elapsed for the alternative event to occur. Purely asynchronous structures have more difficulty in implementing such conditional statements as the clocks progression is only between event epochs into which constructs involving alternative action after specific delays do not easily fit.

The implementation of such clock mechanisms in conventional simulations running on machines using sequential architectures requires only a global variable location in which to store its current value. When constructing equivalent clock mechanisms in a distributed environment such a convenient implementation is not available and an alternative approach must be adopted. Equally in the absence of a central data structure each independent process must maintain its own local event list, communicating with neighbouring processes to maintain synchronisation across the distributed system.

An asynchronous discrete event mechanism in a distributed network simulation could require that each communication between processes carry an explicit time–stamp identifying the time of their arrival. These time–stamps can then be used to schedule the event associated with each arrival, interleaving them correctly with communications from other processes and locally generated events. Such a system works, however, only if each process generates sufficient communications to each of its neighbouring processes. Idle processes or those processes which are required to communicate only infrequently with some of their neighbours can fall out of synchronisation with one or more of those neighbours. A neighbouring process will lose synchronisation if it finds itself without a time–stamped communication to schedule from a process with a direct connection. Unfortunately the guilty process may have no communication to offer and no idea as to when the next event may be generated by it, this being dependant on the arrival of an event from its own neighbouring processes. A more complex interrogative protocol would have to be adopted to find the oldest event in the network for execution to ensure no event occurred out of synchronisation. This of course destroys the very parallelism which we wish to exploit.

To overcome this a standard time unit can be declared. Synchronisation is then maintained if each process is required to generate a null communication every standard unit of time over each link not already involved with communications generated by its event list. Each process then always has, in its event list, the next scheduled arrival over each link. In return for a solution to the scheduling problem the simulation has to be content with a limitation that is introduced as a consequence of this definition of a standard unit. Because each process only checks for a communication once every standard unit, an event which can be generated and communicated completely within a single unit on an otherwise idle connection will not be executed at the correct time on the neighbouring process. Instead its arrival time will be approximated by the arrival time of the next scheduled communication. Out of this arises the usual trade–off between accuracy and overhead in determining the size of the standard time unit.

A synchronous discrete event mechanism also requires a communication overhead to maintain synchronisation. Each node again has an event list of future operations and their scheduled time for implementation. When the simulation is active each nodal process carries out all the operations scheduled within that time interval and then increments their clocks and re-examine their event lists for the next series of operations. Neighbouring nodes inform each other of such increments by the transfer of special communications at the end of the stream of data packets associated with each increment. A simple protocol to implement such a scheme is described in figure 2.10(a) Result and modification access simply consists of comparing each nodes simulation clock against the value assigned to the next interruption at each increment.

This mechanism will maintain synchronisation over the network as long as nodes are not considered to be transparent and generate further transmission events within the same time interval as the original event. In such a case a receiving node may receive a communication from an initiating node and generate its own communication, where that communication was scheduled for transmission in the same time interval as the original communication. If the receiving node has already scanned its event lists for scheduled events and found none, prior to this arrival, it may already have decided that it has no transmissions to complete in this interval and informed its neighbours of this fact. It cannot then send the communication, created by the subsequent arrival, in the correct interval. Neither however could it wait indefinitely for such an arrival as it may not have existed and the node may indeed have no further processing to complete within the interval.

To limit the frequency of such occurrences a simple protocol can be implemented across each pair of channels connecting the nodal processes which postpones the actual clock increment until the protocol has been complete. The protocol defines the number of pairs of end–of–increment transmissions that must be received without an interleaved data communication before a link is said to be safe to close for the remainder of that increment. Thus if a node scans its data structure and finds no more events, it can transmit communications to its neighbours informing them of the fact knowing that if it later receives a communication from one of them, that link will automatically be re–opened for a possible reply. In addition if the node forwards the message it may find a link in a similar position and be able to transmit it straight away. Obviously the greater the number of re–attempts defined in the protocol the greater the probability of catching such sequences of events, but correspondingly the greater the overhead associated with each increment of the clock.

Which of the two approaches to adopt depends on the specific requirements of the simulation. A packet switched simulation, if it is to be meaningful, must produce a high degree of accuracy so that the delay a packet experiences can be assessed and compared to many similar delays incurred by other packets. The circuit switched package has a greater degree of flexibility as the major criteria under investigation is path availability. Each call experiences either the presence or absence of a resource. Factors such as set–up times and packet interleaving are less critical and can therefore be modelled more generally. For this work an asynchronous clock mechanism, ensuring a highly accurate simulation environment was chosen for the packet switched simulator. The obvious choice for the standard unit of time is related to the transmission time across the network links as shown in figure 2.10(b). The more general synchronous approach was used in the circuit switched package, relying more on the statistics of bulk arrivals than the elegance of individual treatment.

2.6 Network Partitioning

The software has been constructed to allow a degree of flexibility in the way it is mapped onto the available hardware. This allow a number of options, or modes, which the operator can select from to tailor the simulations performance and output to his individual needs. As mentioned before, a single Transputer has the ability to successfully support any number of parallel processes by using its microcode scheduler and a time–slicing arrangement. Indeed this is a necessary function the Transputer must support to execute even a single nodal process successfully. Thus given M Transputer cards on which to model the network and an N node network, the allocation of nodal processes can be undertaken in a number of different ways to create a variety of simulation processes with varying characteristics.

First, if there are sufficient Transputer cards, the M cards can be grouped into N master-slave combinations (if slaves are necessary) to produce a sufficient number of modules with sufficient connectivity to completely recreate the network topology in hardware. Each Transputer will execute code for a single master or slave process and all topological links will be mapped onto hardware links between Transputers. An example of such a network is shown in figure 2.11, showing an array of processes mapped onto a hardware array of identical topological arrangement. In this configuration the simulation will proceed at its maximum execution rate, exploiting the full parallelism of the network. Unfortunately, although this method produces the greatest performance, for large networks or networks with densely connected nodes, the number of Transputer cards required to reproduce the entire network or to act as slaves to produce the required connectivity for each module may greatly exceed the number available.

In cases of insufficient or impractical hardware requirements for a one–to–one mapping of hardware modules to nodal processes, a solution lies in reducing the parallelism of the problem by network partitioning. Here the network is divided into node clusters of one or more nodal processes. Each cluster is then implemented as a number of nodal processes running in parallel on each Transputer module, made up as before. To retain the entire network topology, local switching processes on each module act as interfaces. Each switching process multiplexes software channels from each nodal process onto the hardware links of the Transputer module with an extended data protocol enabling the messages ultimate destination to be determined on arrival at an appropriate cluster.

Extending the idea of network partitioning to its logical conclusion it is also possible to run the entire N node simulation on a single Transputer. In this case N nodal processes will run in parallel with a single topology generator which maps the software channels from each nodal process onto each other to once again create the desired topology. For a single Transputer there is obviously no parallel execution of code but concurrency of a different form can be introduced. Assuming each of the M Transputer cards has sufficient memory, each card can execute the same network model on each card with independent random number seeds to produce M independent result files of the same model at the same time. A mapping of this form is shown in figure 2.12 for the same network as the previous example. This has obvious benefits in the subsequent analysis when calculating such parameters as variance and confidence intervals for the simulation results. Solutions of an intermediate nature are also possible where a network of Transputers could be configured to run a number of independent simulations, where each simulation is partitioned over a section of the network. The resulting performance would produce a trade–off between execution rate and result accuracy.

The decision on the way the available processing power should be organised is dependant entirely on the operators requirements. If for example the simulator was being used as a network design aid or as part of a real time decision support system where the order of magnitude is more important than the confidence implied by the result then the most profitable option would most probably lie in maximising the parallelism and so performance. In contrast, many aspects of research into network performance often requires multiple repetitions of identical simulation models (with independent seeds) to reduce statistical noise which may swamp any single set of results. These areas include topics such as the interaction of routing and flow control, transient behaviour on component failure or traffic fluctuation, parameter optimisation and network instability effects with no overload protection. In these cases the simultaneous execution of independent simulations can provide the ideal medium for such investigations.

2.7 Summary

In this chapter the major design principles in the construction of a high speed simulation environment are extracted from a review of previous architectures. A hybrid of these principles is then applied to the construction of a new environment using the Transputer as a parallel component, programmed in its associated language Occam. Design features arising from the use of a loosely coupled array of processing elements to simulate communication networks are discussed and solutions outlined to problems such as access and synchronisation. Finally the variety of modes in which the simulation engine can be configured are presented and the advantages of each orientation explained in terms of the trade–off between performance and accuracy. In the next chapter analytic and simulation models are presented for the performance analysis of fully connected circuit switched networks. The simulation model is designed in such a way that it can use the Transputer surface just descibed to optimise execution times.

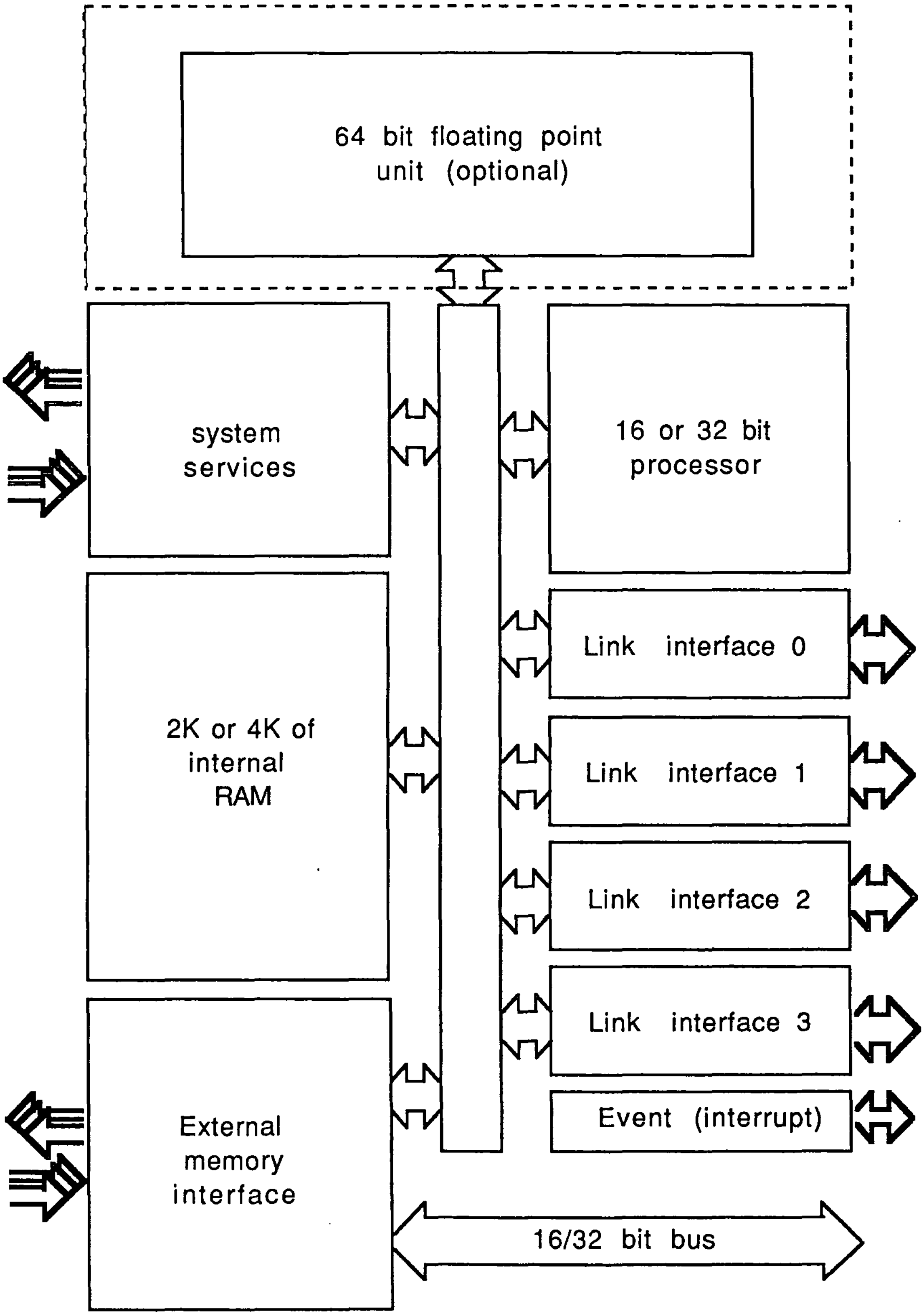


Figure 2.1 Schematic of the INMOS Transputer

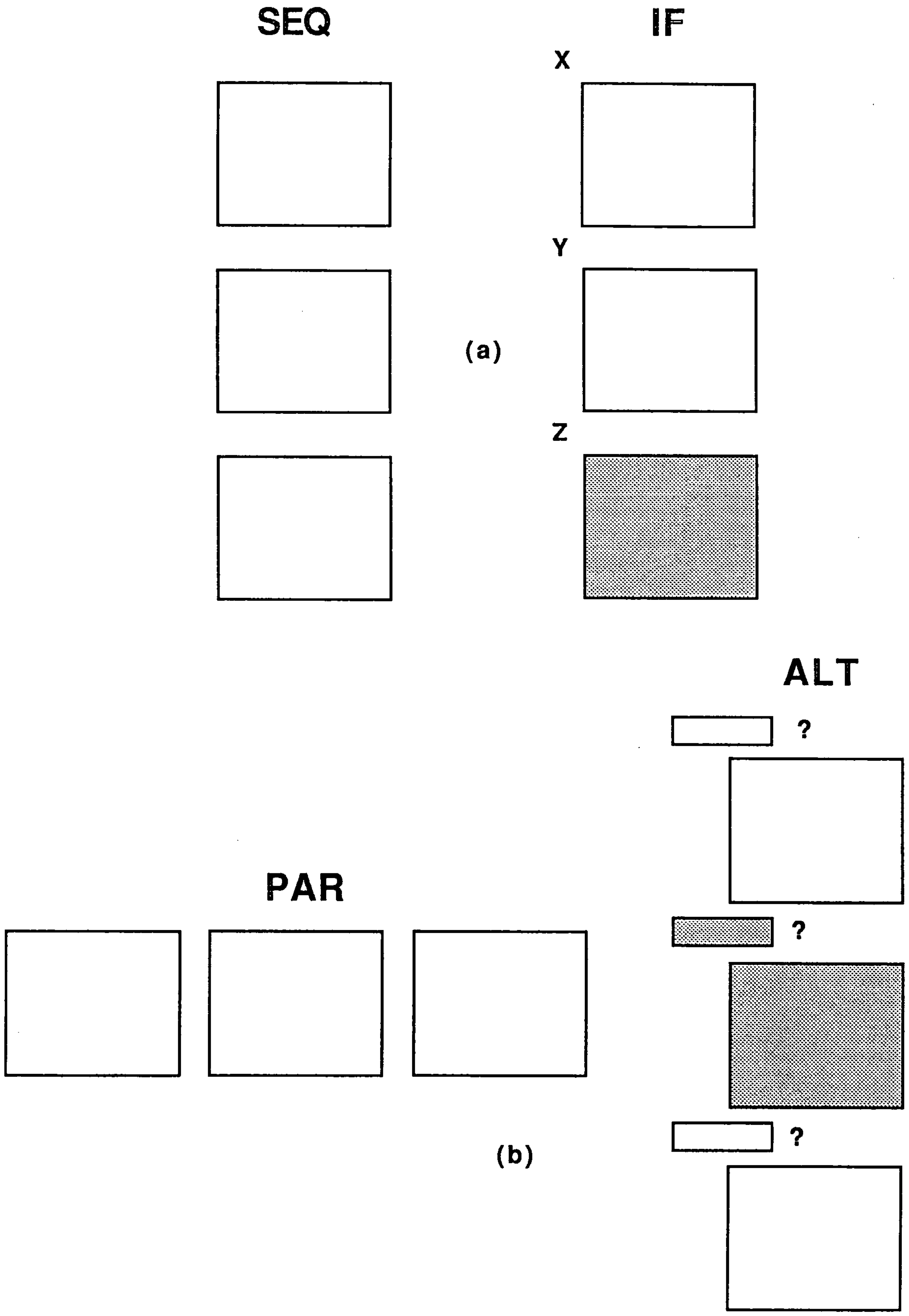


Figure 2.2 : Occam Constructs

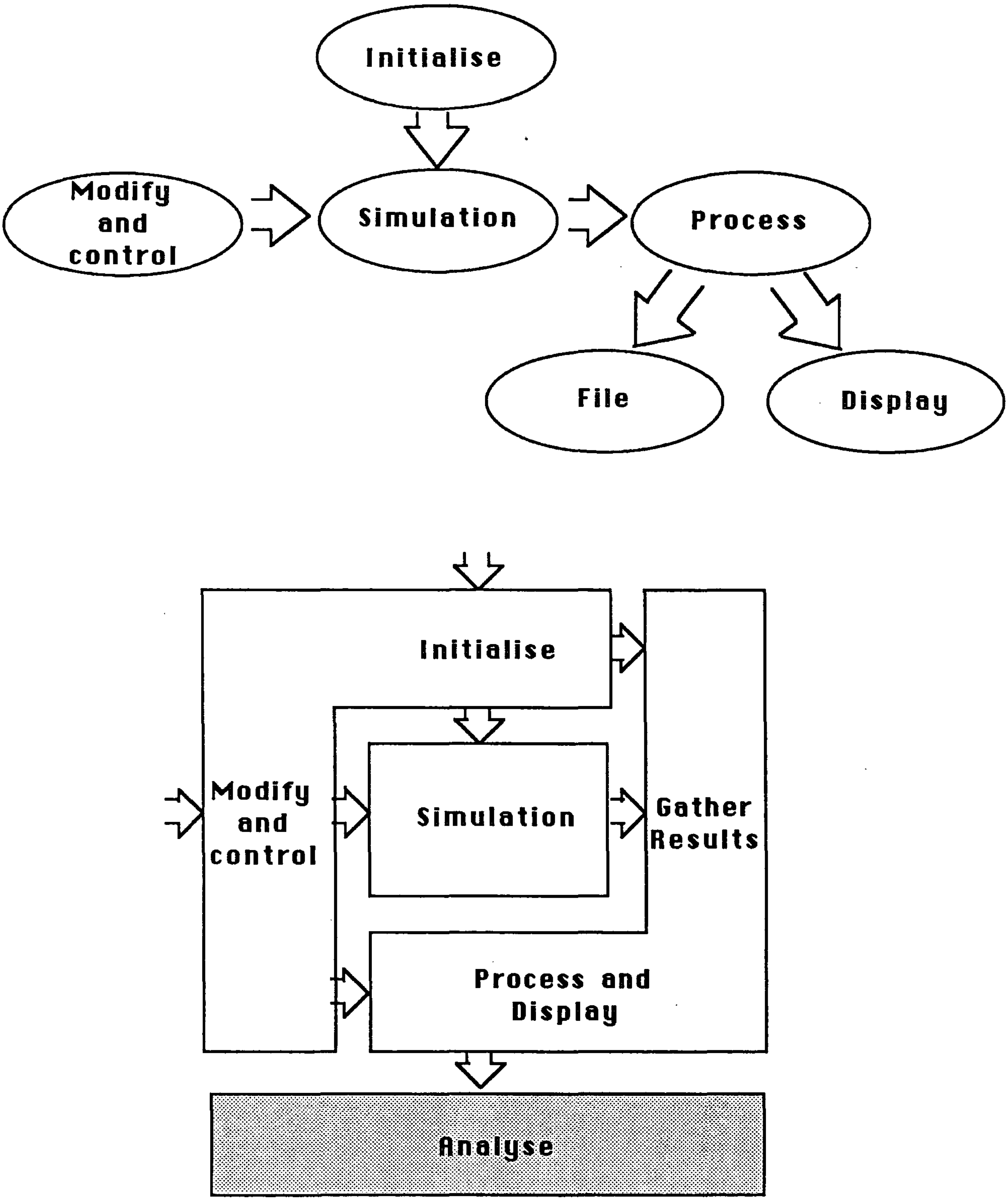


Figure 2.3: OCCAM Mapping

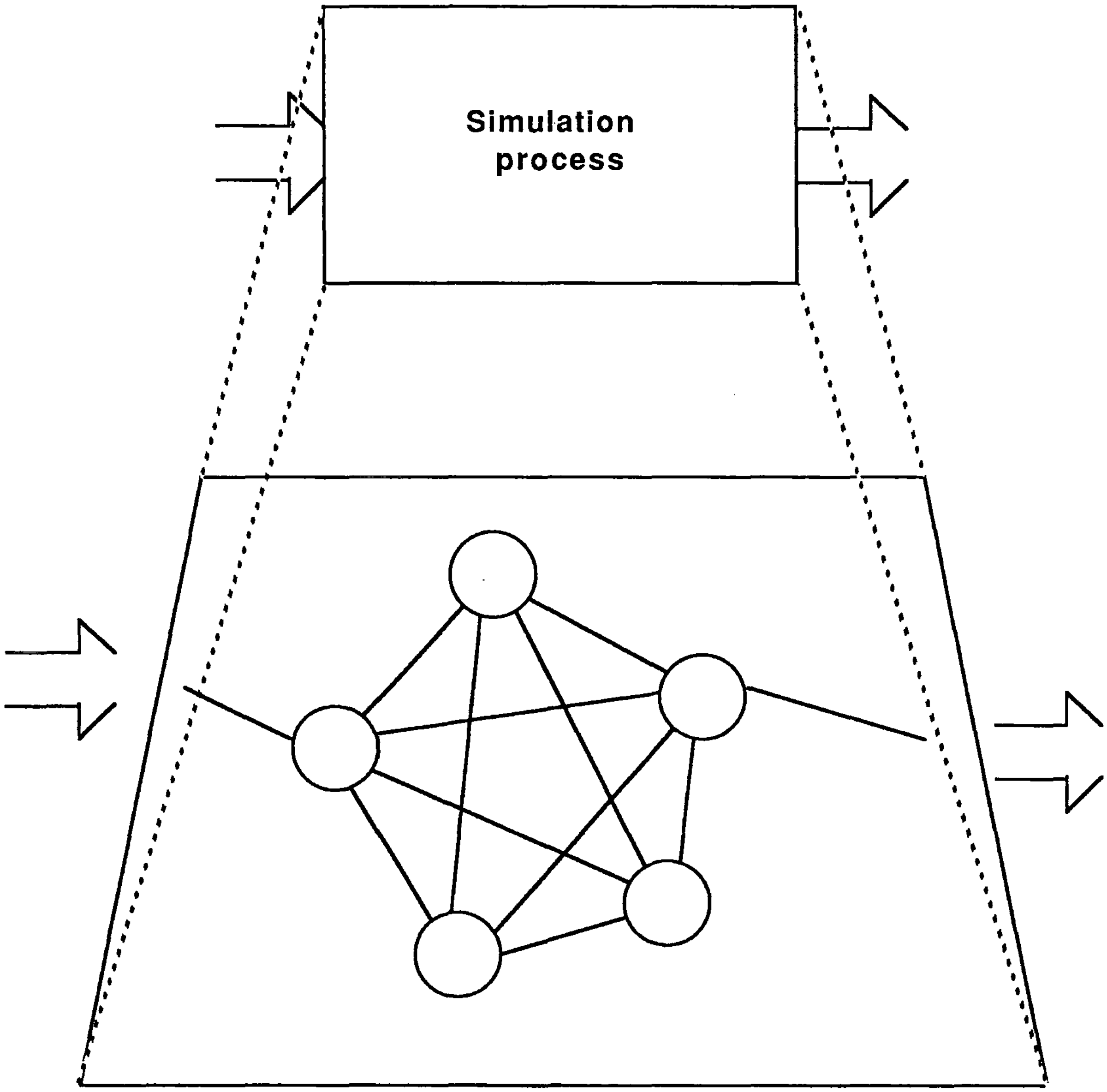


Figure 2.4 Simulation process decomposition

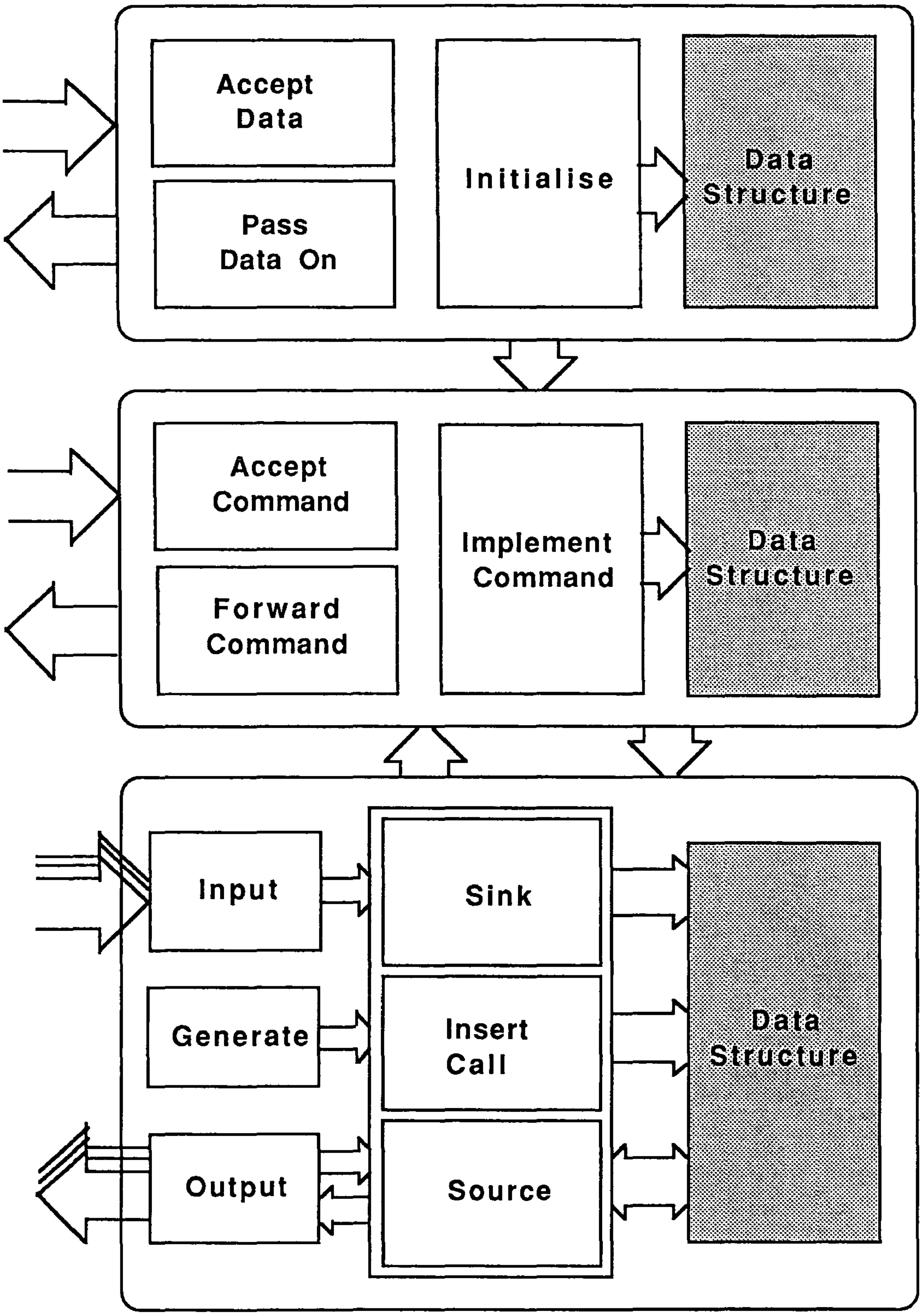
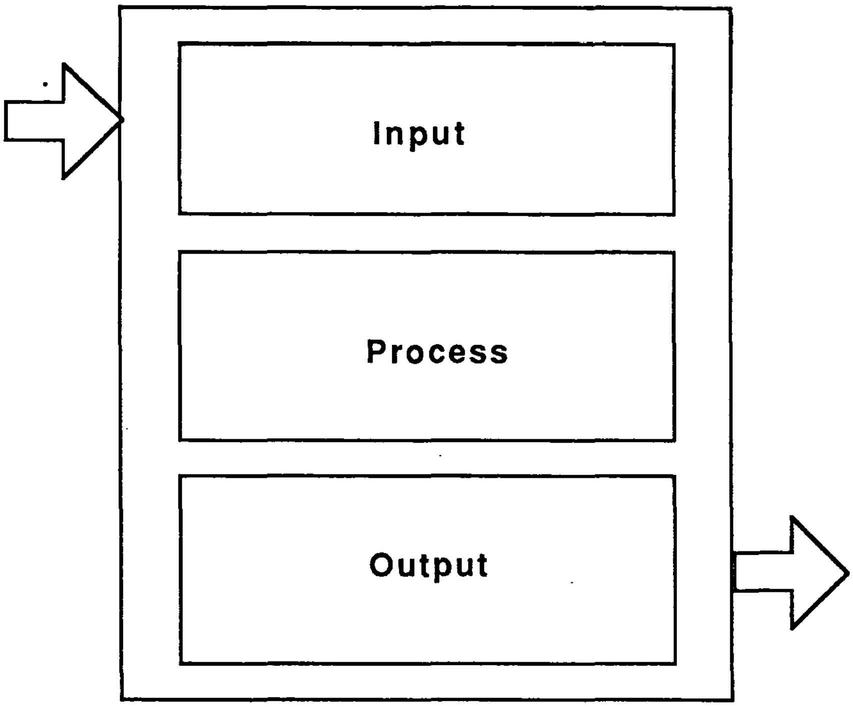


Figure 2.5 Nodal Structure



Sequential Nodal Process

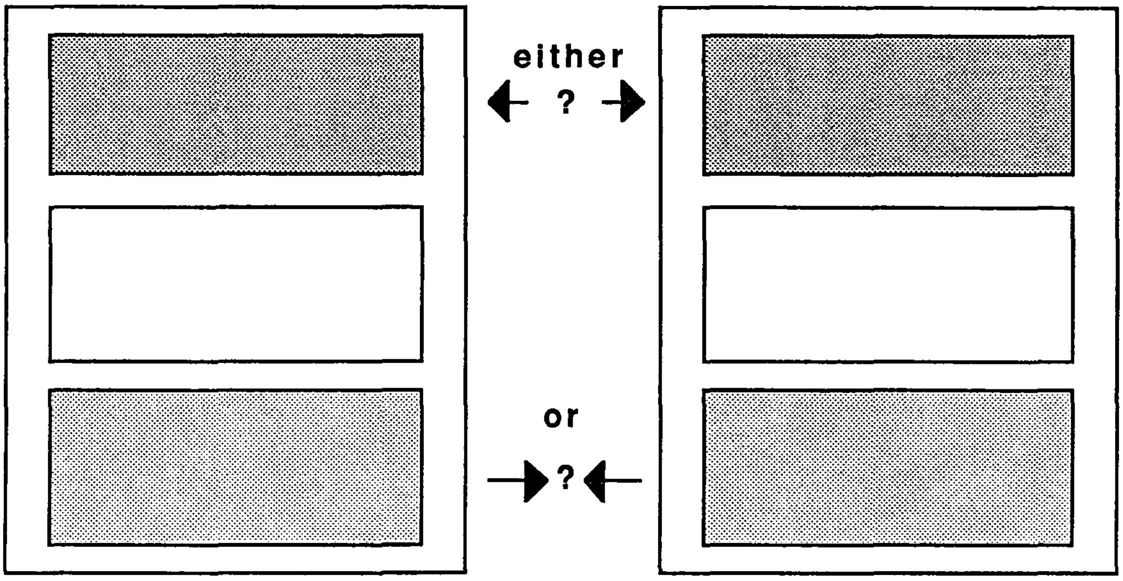
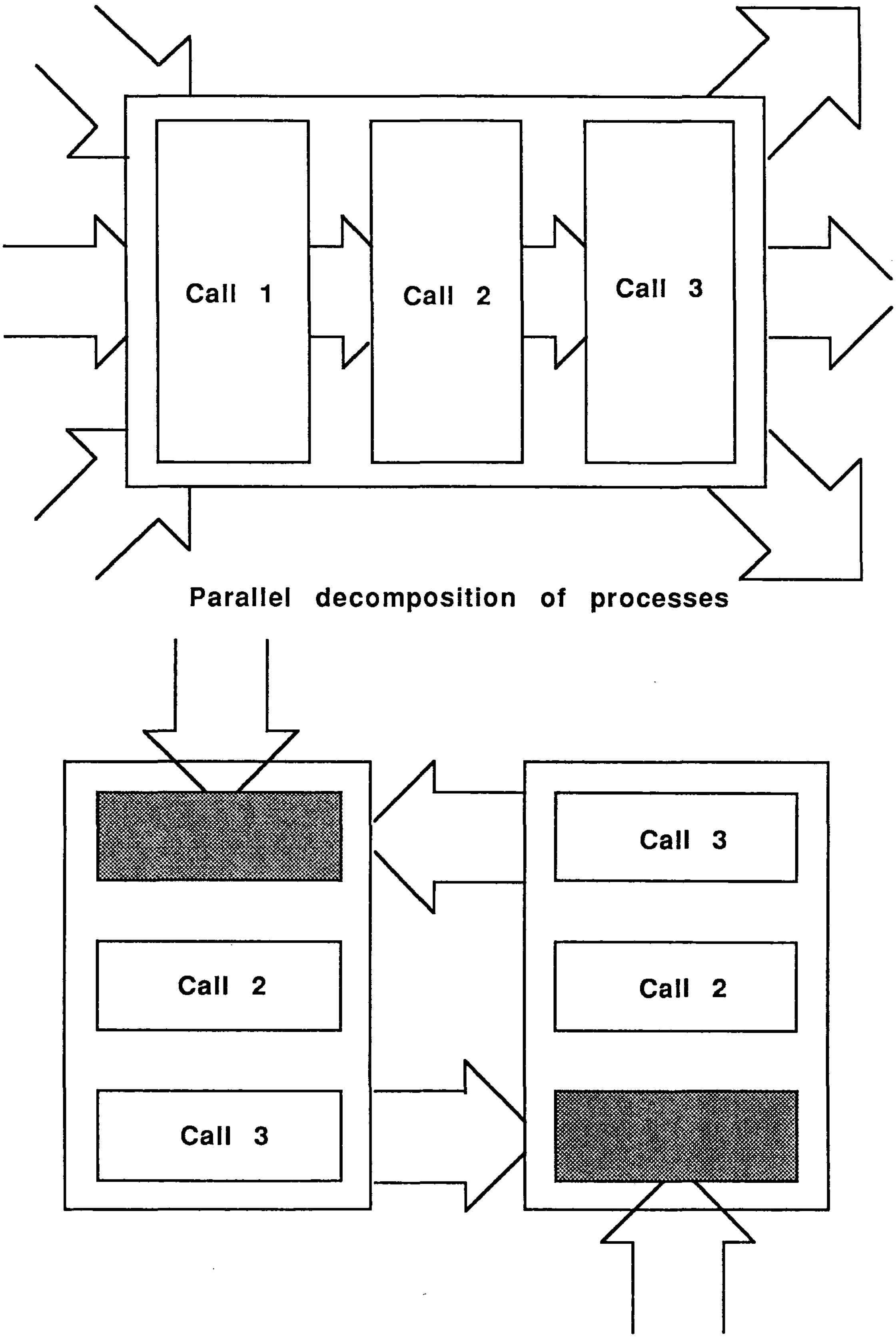


figure 2.6(a) Sequential deadlock



Parallel decomposition of processes

Figure 2.6(b) Parallel Deadlock

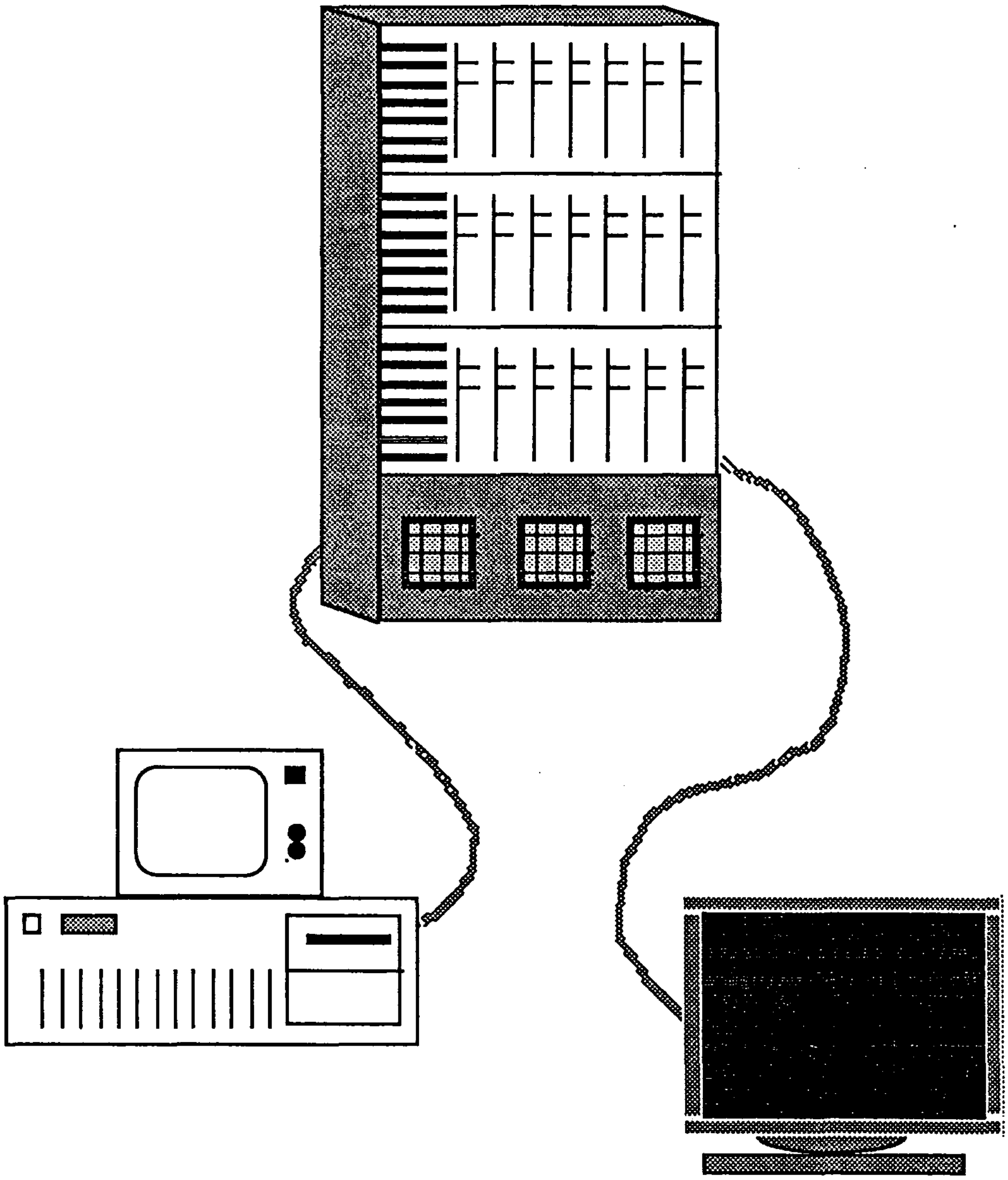


Figure 2.7 External View

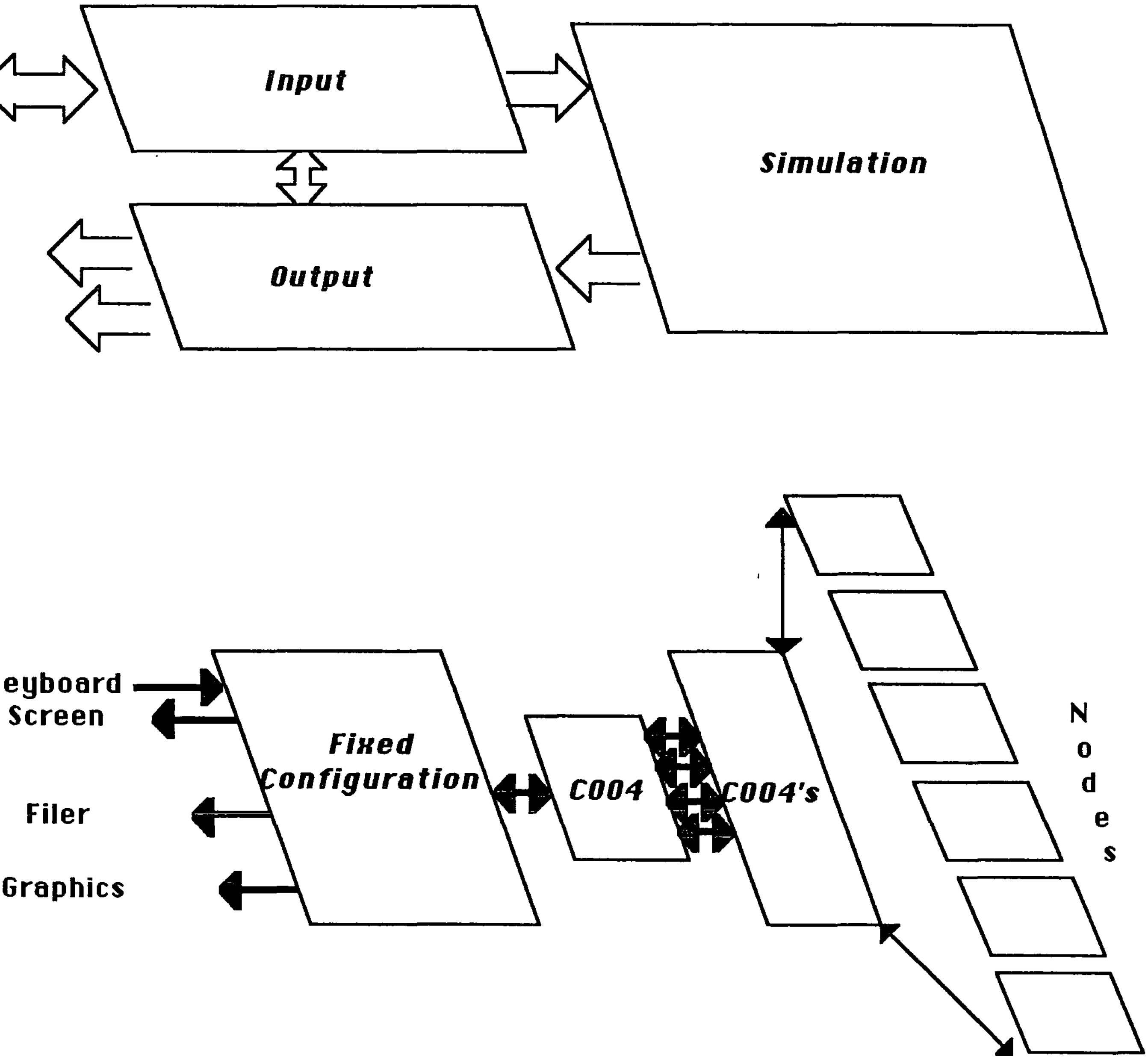


Figure 2.8 Hardware Mapping

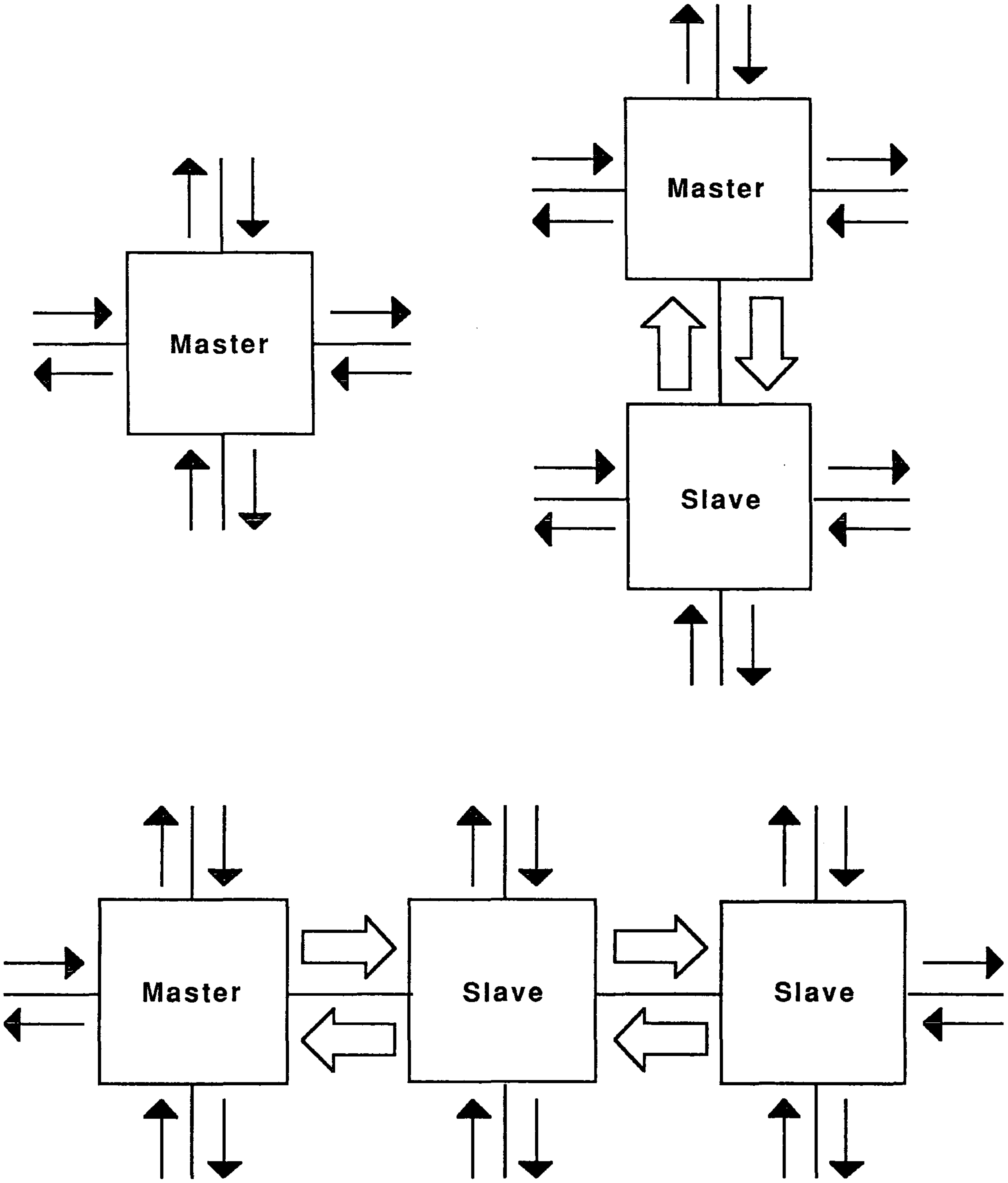
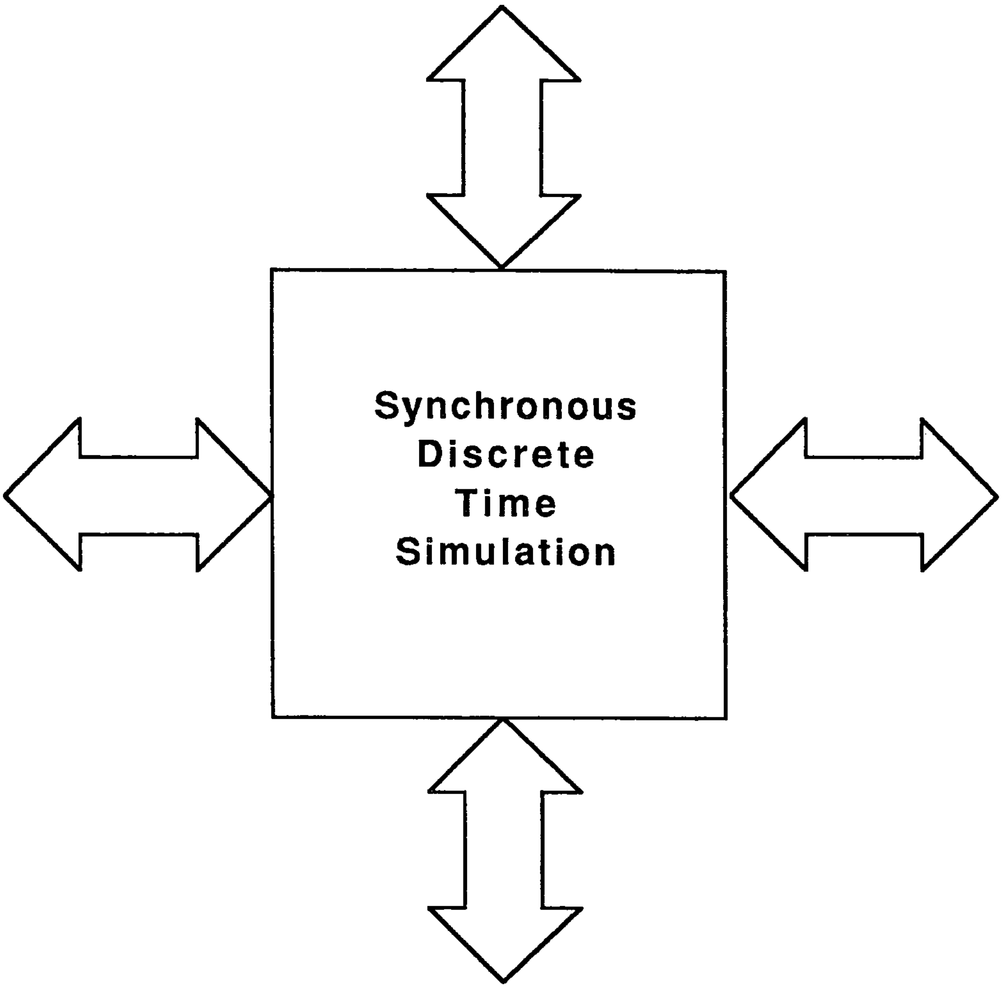


Figure 2.9 1,2 and 3 Transputer modules



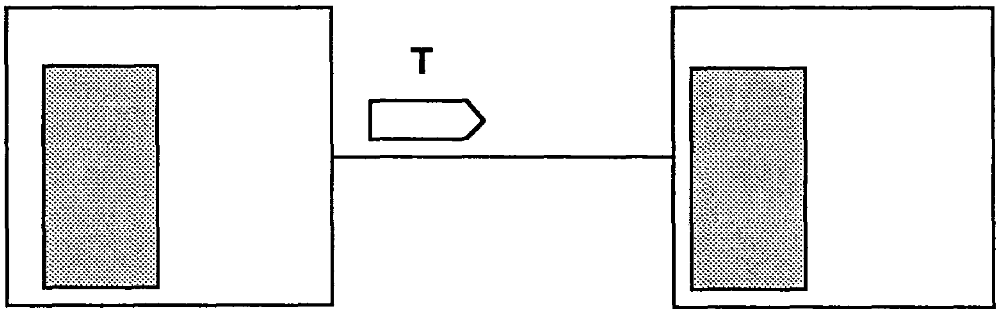
PROTOCOL OUT:

```
    {0..N} Data Packet + Finish Packet
```

PROTOCOL IN:

```
        IF Finish Packet
            IF ALL FINISHED
                TIME := TIME + 1
            ELSE
                NOTE LINK FINISHED
        ELSE
            PROCESS DATA
```

Figure 2.10(a): Synchronous Discrete Time Model



- T = transmission time of fundamental data unit between nodes
- transmission at time t reaches neighbouring node at t+T
- events at a node cannot affect events at a neighbouring node within time T of generation
- to retain synchronization a timing packet need only be sent every T seconds if no other communication is scheduled

IMPLEMENT AS ASYNCHRONOUS DISCRETE EVENT SIMULATION WITH MAXIMUM INTER-EVENT TIME T

Figure 2.10(b) Asynchronous Discrete Time Model

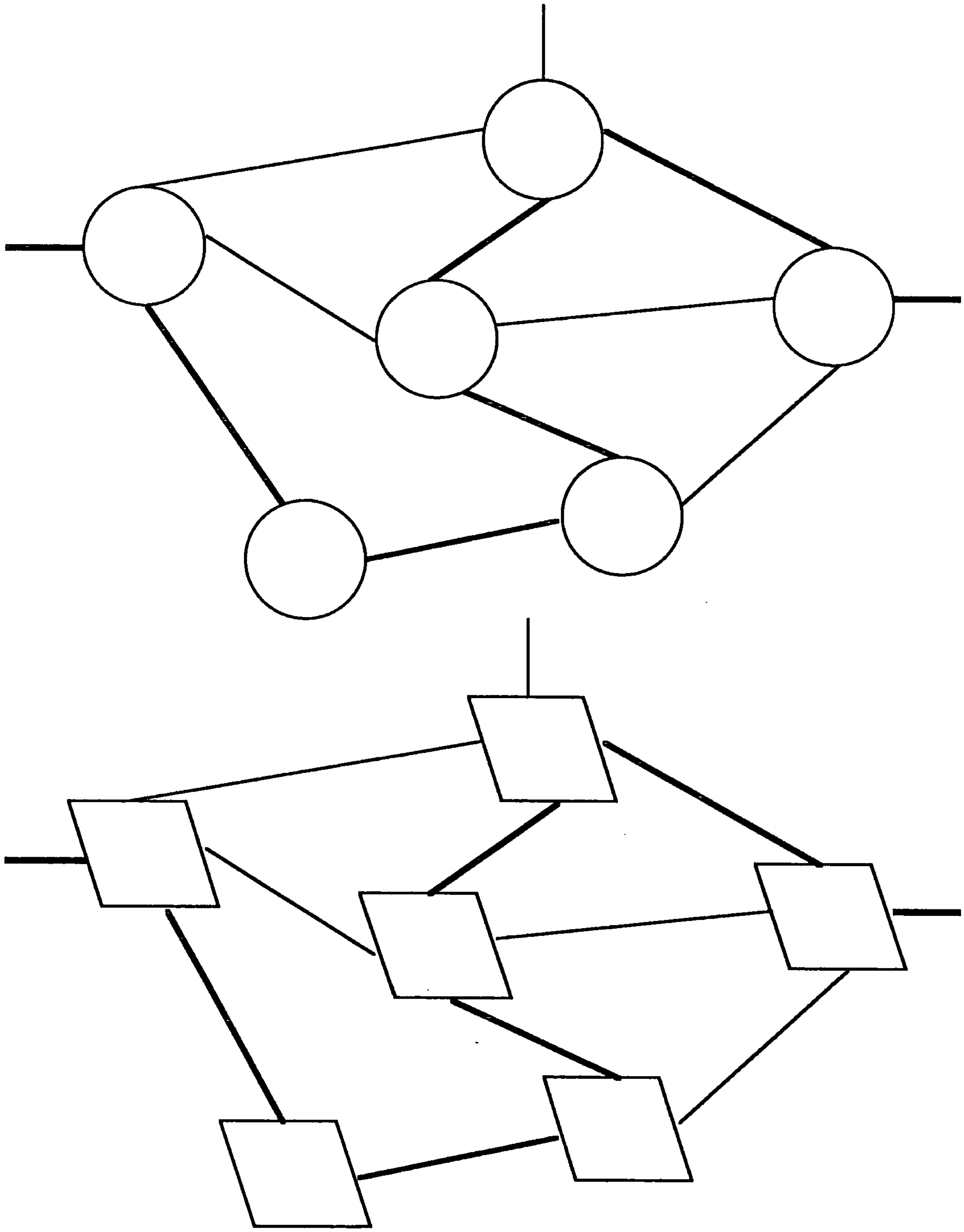


Figure 2.11 1:1 Mapping

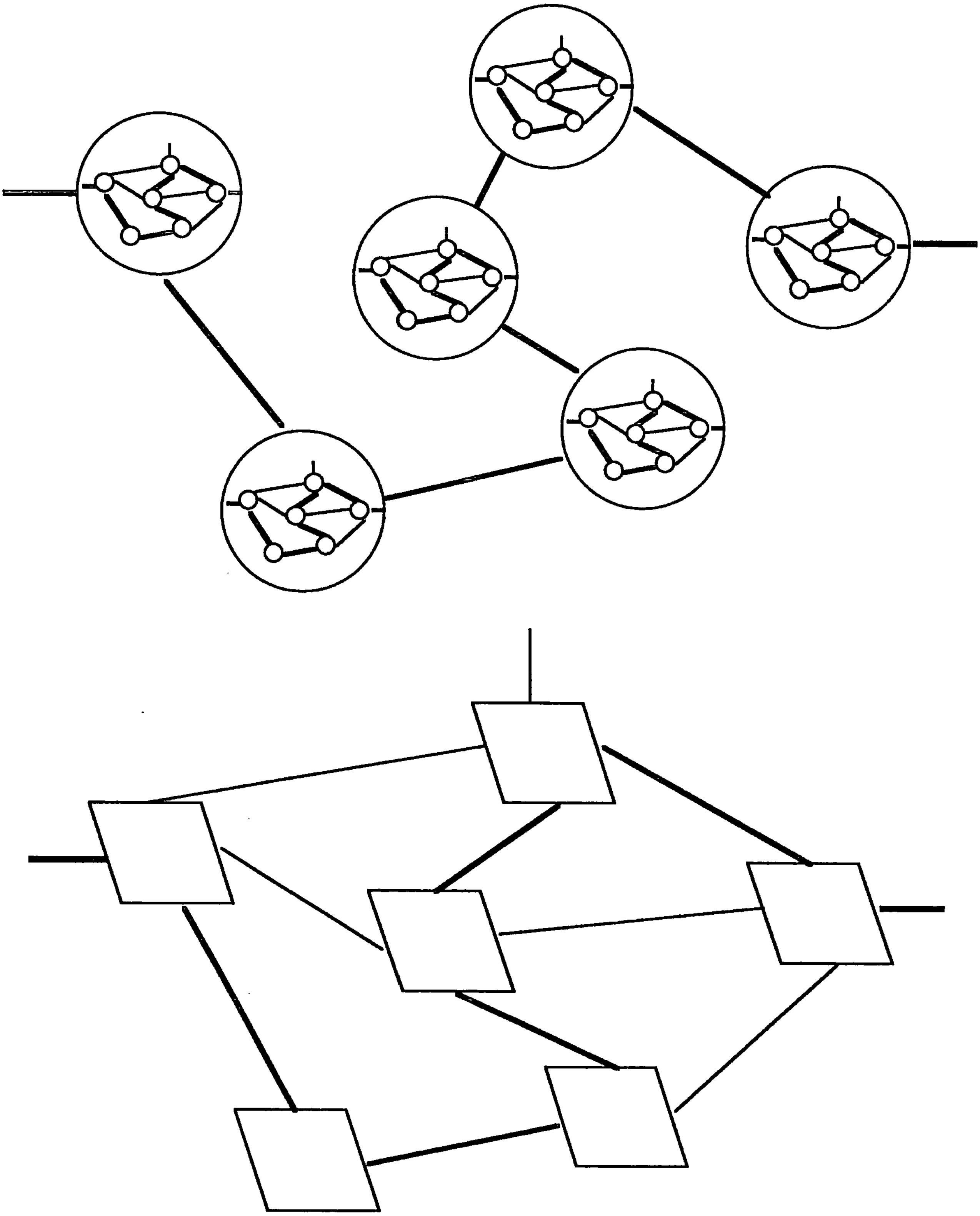


Figure 2.12 N:1 Mapping

Chapter 3
Analytic and Simulation Models For Dynamic Routing Strategies in Fully Connected Circuit Switched Networks

3.1 Introduction

Since their introduction on a commercial scale telephone links, using circuit switched technology, have provided the primary source of point to point communications in the world. The obvious impracticalities of a permanent, dedicated telephone link between every pair of subscribers quickly lead to the construction of networks to concentrate and transmit calls practically and efficiently between communicating exchanges. Technological limitations led to the design of hierarchically structured networks in the 1950's which enabled the growing networks to implement crude alternative routing strategies, allowing multiple paths for call transmission and networks engineered to provide a predictable grade of service (GOS).

Today, with the development of digital techniques, semiconductor technology leading to LSI and VLSI implementation of complex transistor circuits and software engineering many of the previous limitations have been removed. Modern switching exchanges are now capable of performing a multiplicity of functions unparalleled in the history of telecommunications. Such advances have led to a revolution in the design of circuit switched networks, both in architecture and management. Not least among these advances has been work generated from interest in new forms of routing algorithms, themselves only made feasible by these same advances, to maximise the performance of the new network architectures. An important class of these new routing strategies use adaptive algorithms which seek to optimise their behaviour by modifying their route selection according to information on the state of the network.

In this chapter adaptive routing and its place in the newly emerging network architectures is reviewed. Analytic and simulation models are then introduced to examine the behaviour of a wide variety of strategies in fully connected networks, where every node is connected to every other node by a full duplex connection. Section 2 identifies the major innovations which have brought about these changes and describes the effects they have had on the type and way in which information is carried across the network. Section 3 describes the routing strategies which have been developed to route calls across the networks. A classification system is introduced to categorise strategies according to their method of operation. Finally some additional techniques which have been developed to protect networks components from degradation under overload, and which are used in conjunction with these routing strategies are also mentioned. Section 4 outlines the specification of the circuit switched simulation package which was used to analyse the comparative performance of these routing strategies. It was designed to incorporate a number of features which allow it to model the processing capability of the nodal exchanges, giving it a degree of flexibility in its operational characteristics. Finally in section 5 analytic models, based on Markov chain theory, are derived to calculate the performance of different routing strategies for the case of single and multiple traffic sources in a network.

3.2 Evolution of Circuit Switched Networks

3.2.1 Network Architecture

Historically circuit switched telecommunication networks employed electromechanical switching exchanges arranged in a hierarchical structure[26]. This guaranteed loop free connection paths under a routing strategy which used a fixed sequence of routing alternatives at each node to form a path through the network. This limitation in route selection was a product of the lack of available network status information and the hardware limitations imposed by the hard wired switching arrays used in these exchanges. In addition the administration of a hierarchical network and the engineering of trunk capacities for peak traffic conditions were found to be relatively simple in comparison to non-hierarchical networks. An example of such a structure, based on the old AT&T toll network is reproduced in figure 3.1. It shows a section of a five layer hierarchical network connecting two end offices. Two distinct classes of link can be identified. The dashed connections show some of the possible high usage links which may connect exchanges and which form preferred routes between the two ladders of the hierarchy along which a call can be routed. The solid lines indicate final trunk groups and together form the route a call would attempt last before finally being blocked by the network. Some or all of the high usage trunk groups exist between all end offices depending on their relative locations geographically and their importance.

A call would first attempt the direct path between the two end offices (if such a link existed). If that path is unavailable it is routed up to the next exchange in the hierarchy where there may be further high usage trunk groups. The call will attempt to find a path on one of these in order of ascending connection up the hierarchy. If these paths are unavailable the call progresses to the next exchange up the hierarchical ladder. This process of climbing the hierarchy continues until the call is successfully routed over a high usage trunk group or a final trunk group between two exchanges is unavailable whereupon the call is blocked. On traversing the hierarchy to the destination ladder the call is routed down the ladder, making use of any high usage trunk groups, to the end offfice. Again the call may be blocked at any stage if a final trunk group is unavailable when the call tries to acquire a link on it.

Advances in signalling and switching systems and the rapid growth of teletraffic have lead to a good deal of research into network efficiency. It has been demonstrated[27,28] that considerable performance benefit can achieved by replacing some of the layers in the hierarchical network by a densely connected mesh structure with an appropriate routing policy. This allows the network to make effective use of spare capacity within the network caused by stochastic traffic fluctuations, forecast errors and non-coincidence of peak traffic across the network (especially in continental and world-wide networks). A hierarchically structured network is unable to do this because of its inherently static nature where each area must be dimensioned for peak traffic levels as it is unable to share its resources with the rest of the network. In comparison consider the small mesh network shown in figure 3.2. In the network two exchanges A and B (which may serve an area with a hierarchically based network beneath them) wish to communicate. The most direct path would be attempted first. If the call was blocked an alternative path will then be attempted using one or more tandem nodes in the network. The details of the path selected will vary with the routing policy adopted but the essential feature is the equal importance of each node to each of the other nodes and the ability this gives it to share transmission bandwidth to create a path between the originating and destination nodes.

3.2.2 Switching Exchanges and Call Connection

The switching exchanges in circuit switched networks have to deal with the closely coupled tasks of providing dedicated bandwidth on the transmission links over which the calls are routed in and out of the exchanges and providing a dedicated connection between these two links. Before the arrival of modern digital telephone exchanges such a service was provided by frequency division multiplexing the transmission link bandwidth into separate frequency bands and connecting them via cross–point switching arrays using relays or semiconductor connections. Figure 3.3(a) shows how the transmission bandwidth is divided between two exchanges on a connecting trunk. In this example N channels are carried in a frequency band of fHz beginning at fsHz. Considerable research went into the formulation of cascaded cross–point switching arrays to provide non-blocking and low blocking switching array combinations for large exchanges. In contrast modern telephone exchanges contain dedicated processor controlled systems and the transmission information is digitised for transmission using time division multiplexing. Calls are sampled and transmitted in pre-defined order to form a transmission frame. A typical frame is shown in figure 3.3(b) showing 24 channels transmitted in a 125μsec frame. This is the standard format used in America, Canada and Japan providing a bandwidth of 64kps to each channel. Switching from input to output channels can be simply achieved by reading the data frames into local memory at each exchange and reading back out the relevant sample into a suitable slot in the output frame. In switching networks such as the AT&T ESS No4^{TM} a combination of digital and analogue switches are cascaded to give large handling capacities.

The introduction of digitisation and the installation of sophisticated switching exchanges in modern circuit switched networks also enables the network to offer new functions not previously available with the previous analogue systems. Functions such as call forwarding, automatic re–dial, priority access and free–phone services can all be coded into the call connection signalling at the inception of a call. In addition the use of high frequency digital transmission allows the efficient transmission of digital data over the telecommunication networks. Although voice traffic stills accounts for the majority of the service, the ability of the network to handle digital data is greatly enhanced and transmission rates of 64kps plus the growth in the use of optical fibres provide greater incentives for the network to be used for these purposes. Modern systems can provide order of magnitude improvements over the analogue 14.4kps or 9.6kps bandwidth links.

As was briefly mentioned above the increasing power and sophistication of the switching exchanges has lead to changes in the manner in which a call is established between two parties wishing to communicate across the network. In any circuit switched call a dedicated path must first be established over which the call can take place. This is described as a circuit. The circuit may take a direct path between exchanges or may pass through one or more intermediate exchanges. However at each stage a channel between the two communicating exchanges must be secured for a successful connection. In all circuit switched networks this is done by the transmission of control messages between the exchanges. Conventionally these messages are transmitted on the same channels as are to be used to transmit the information carried over the network on successful completion of the connection. This form of call connection is known as in–band signalling. With the digitisation of the telephone networks, however, a second method of call connection has been developed, this is known as common channel signalling (CCS) or common channel interoffice signalling (CCIS). In CCS the connection information is carried on dedicated channels (time slots in each frame) or on a separate CCS network alongside the circuit switched network. As the information carried by the CCS network is in the form of short packets of information it can use packet switched technology. This greatly reduces call connection times and increases the network signalling capacity as the data can be efficiently transmitted over dedicated, high bandwidth links. A great deal of network flexibility is also introduced with CCS methods as additional information can also be carried over the signalling system in digitised form, enabling the implementation of the advanced network functions mentioned earlier, network management operations and advanced routing techniques.

3.3 Routing Strategies in Circuit Switched Networks

3.3.1 Classification of Routing Strategies

With the advent of non-hierarchical networks the fixed sequence of routes via high usage trunk groups, followed by the final trunk group engineered to give the required grade of service, is no longer the only option open to network designers. The upper layers of the hierarchical networks are now being replaced by densely connected mesh networks with peer nodes requiring a new type of routing. Let a path between two nodes be defined as a collection of trunk groups which connect two nodes, possibly via one or more intermediate nodes. Then any routing strategy in a non-hierarchical structure must provide one or more paths through the network for calls in that network from the calls origin to the destination and some method for selecting amongst them.

Each of these paths can be considered as a route. The way in which these routes are determined by a particular strategy can be used to classify the strategy. This enables us to build up an understanding of the fundamental similarities and differences between algorithms under comparison. The following classification is adapted from Grandjean's paper on the introduction of non-hierarchical routing in telecommunication networks[29] and the approach of Gerla[30]. A diagrammatic breakdown of the classification of a routing mechanism of operation is given in figure 3.4. Fundamentally a routing strategy is said to be *dynamic* if the mechanism by which the routes are selected allows the selection to change its form according to some function of a variable or set of variable parameters. If a routing strategy does not meet this criteria it is said to be *fixed.* Dynamic routing strategies can be further sub-divided into *adaptive* and *time-dependent* schemes. In a time dependent strategy the selection of routes through the network, or the criteria by which they are judged, uses time as the variable parameter. In a purely adaptive strategy the decision of route selection depends on information about the state of the network available to the strategy at the time the route selection is made. Adaptive routing can be divided again into schemes which use information on the global state of the network to make their decisions and those who base their decisions on locally gathered information. This may entail neighbouring nodes exchanging special 'routing'packets to acquire the information necessary to formulate their selections. The special case where no information of this type is exchanged, each node making their decisions on information contained exclusively within the node is known as an 'isolated' routing algorithm. In general routing strategies using global information use special purpose network routing centres in which to gather and process network information and are referred to as *centralised*. The alternative, which allows each node to make its own routing decisions is known as a *distributed* routing strategy. An interesting combination of these two extremes is the *hybrid* routing policy which combines the two forms of algorithm[31]. Here a centralised routing centre would periodically inform each node of a selection of actions which were most favourable. Each node would then be free to choose from amongst selections as it saw fit until the next update. A possible implementation derived from work in packet switched networks is described in work done by Girard and Hurtubise[32]. Policies of these form combine information from a centralised global optimising algorithm with a distributed component allowing each node freedom to react to traffic fluctuations between updates.

Finally one important distinction between routing strategies not covered by this classification is the mode of path selection, which is defined as deterministic or stochastic. A routing strategy is said to be *stochastic* if the choice of a route is governed by a probabilistic method. The complement of stochastic routing is *deterministic* routing where the selection of routes is made in a pre-defined order.

As an aside from the classification of routing strategies, but worthy of mention as a distinction in operation, is the method of control used during the call connection phase. In originating office control (OOC) the origin node ensures a path has been established to the destination node before routing the call. This can be done using CCS with control being returned to the origin node in the event of blocking at a tandem node. In sequential office control (SOC) the path is established on a node-by-node basis. If a call at a tandem node is unable to locate a route to the destination the call is blocked at the tandem node and lost.

3.3.2 Review of Current Routing Stategies

In this section the routing strategies currently employed by the major telecommunication companies are described and the research they and others have undertaken to further the understanding of routing in circuit switched networks summarised. All the algorithms described here are for implementation on non-hierarchical networks and can be classified under the scheme described in the previous section.

Fixed routing

Fixed routing describes any routing policy which is invariant to external influences and can be either deterministic or stochastic. A deterministic, fixed routing policy is also known as automatic alternative routing (AAR) and is a direct analogy of a fixed hierarchical routing strategy. In this strategy the call first attempts the direct link and, failing that, attempts one or more routes via tandem nodes in a fixed sequence. If a path has still not been found at the exhaustion of the sequence the call is blocked. An example of such a strategy is implemented on the European AUTOVON network which forms part of the worldwide Defence Communication System (DCS). This network uses OOC with spill forward which enables tagged tandem nodes to act as origin nodes to increase the range of paths available for calls[33]. The task of optimising a network using fixed, deterministic routing for forecast traffic is discussed in a paper by A.Girard[31], who defines the problem for a fixed network configuration and introduces heuristic methods for accelerating the calculation time for the iterative solution.

The second form of fixed routing, fixed stochastic routing, uses an invariant probability distribution to select an alternative path if a call is blocked on the direct path. If all outgoing links have equal probability of selection, regardless of network parameters, the strategy is known as random routing. If however the distribution of selection probability is modified according to the trunk capacity the stochastic strategy falls into a category broadly labelled as proportional routing. An example of this strategy occurred in the Bell-Northern toll network during the transition from analogue to digital exchanges[27] to route overflow traffic from an unintelligent analogue node to an intelligent digital node for selection of a route to the calls destination.

Time-Dependent Dynamic Routing

Dynamic Non-Hierarchical Routing (DNHR) is AT&T's dynamic routing strategy implemented in its intercity toll telecommunication network. It is substantially a centralised, time-dependent, deterministic, dynamic routing policy and is used to route calls in a densely connected mesh network of 'tandem offices'. Each tandem office is then serviced by many smaller exchanges using hierarchical routing which concentrate the traffic for destinations across the network and feed it to their local tandem office. Within the DNHR section of the network calls are routed between tandem nodes using direct links or tandem paths with crank–back (OOC) using stored program control (SPC) exchanges utilising CCIS. These will contain No4 ESS exchanges or those with similar specifications. The alternate routes to each destination are selected in fixed order and the call only blocked if all alternatives are exhausted. The algorithm varies from a fixed, deterministic policy by dividing the day into 10 periods and providing a separate sequence of alternate paths for each period. In addition further alternate paths are provided by a real time algorithm on exhaustion of the sequence. These enable the network to take advantage of the stochastic fluctuations in traffic on the network and are only provided if sufficient spare capacity is detected on the outgoing links. The sequences of alternate links are calculated from the solution of a large linear programming algorithm designed to provide optimal routing patterns to minimise network cost for forecast traffic in each time period. This allows the network to take advantage of the non-coincidence of peak traffic in the network and utilise spare capacity[34]. The program is run once a year for the main network design and weekly to correct forecast errors with solutions which not only meet the grade of service (GOS) required but also represent least cost deviations from the present network state[35].

Adaptive Dynamic Routing

In adaptive, dynamic routing it is the state of the network itself rather than any external factors which control the selection of alternative routes between origin-destination pairs. Bell-Northern have developed an adaptive, dynamic, centralised, stochastic routing strategy in which a central network processor collects data from the network nodes and transmits a suggested selection of tandem nodes for the call overflowing its direct link. A weighting factor is also transmitted for each suggested tandem node informing the origin node of the expected number of free links on the path incorporating the tandem node. If the direct link between two nodes is unavailable a route through a tandem node is selected by the origin node. The call is controlled by a SOC process and if blocked on either link the call is lost. The probability of a tandem node being selected is proportional to the expected number of free links on that path. The predicted number of free links is calculated by linear extrapolation of the most recent data on the state of each path using the total arrival rate of traffic to each node, the number of free trunks and the average call holding time.

While the central processor allows the best potential paths to be identified for each traffic source it suffers from the vulnerability of central control and a delay in processing and transmitting the data back to the network. The following routing strategies adopt a distributed, stochastic approach in which a node makes its own routing decisions based on local information. They have the advantage of being simple and effective, requiring only a small amount of local processing and memory to store the state information. The first strategy is the stochastic learning automata approach. In this strategy each routing alternative or action is given a selection probability. If the direct link is inaccessible, one of the actions is stochastically selected. Depending on the outcome of the call attempt using the selected action, the probability of that action is updated. Reward and penalty schemes exist to increase and decrease the probability of selecting the action again, according to various formulae. Different combinations of reward and punishment schemes have been developed. In this work we focus on the extensive work has that has been done on the application of stochastic learning automata with linear update mechanisms in hierarchical networks[36] and small mesh networks[37,38] where it compared favourably with fixed, deterministic and fixed, stochastic (random) strategies, especially under focused overload and failure conditions. A summary of automata theory is outlined in Appendix A.

A second strategy falling into the same category has been developed by

R.Gibbens and F.Kelly at Cambridge University[39] for British Telecom's trunk network with the introduction of their new digital exchanges. Dynamic Alternative routing (DAR) attempts the direct link first, and if blocked then attempts to route the call via a tandem node, the identity of which it has stored in local memory. If the call is blocked on either link of this alternative route the call is lost and the identity of the tandem node is re–selected stochastically from among all the possible tandem nodes. If the call is successfully completed on the alternative path the identity of the tandem node is retained and subsequent calls overflowing the direct link attempt the same alternate path until a call is blocked. Simulations on small overloaded networks have been carried out[40] but no direct comparisons against other dynamic strategies are available in the published literature.

3.3.3 Network Protection Mechansims

When a network becomes overloaded, that is attempts to carry traffic in excess of its engineered load, its performance can deteriorate. This can occur as a result of trunk blocking, switch blocking or a combination of the two. Trunk blocking is the inability of the call to find a path through the network from the origin to the destination. Mathematical analysis of non-hierarchical, symmetrical[39,41,42] and small non-hierarchical, asymmetrical networks with fixed deterministic routing strategies[43], under overload conditions, produce multiple solutions suggesting bi–stable blocking behaviour with high and low congestion states. Further work on simulation models of real engineered networks[43] found no evidence of the manifestation of this behaviour but did find high blocking probabilities in networks using alternative routing at loads exceeding their engineered specification. This behaviour arises from the competition between directly and alternately routed traffic for transmission bandwidth. An alternately routed call, because it occupies more than one link in the network, ties up more network resources than its direct counterpart. When the network contains spare capacity this is acceptable and network blocking is reduced. However, under overload conditions the spare capacity is not available and the additional expense of alternatively routed calls compounds the blocking problem in the network. In overload two states, corresponding to the mathematically observed states of congestion can be envisaged. Low congestion would occur if there is little alternative routing, the majority of the calls accepted by the network being routed directly, efficiently utilising the network resources. States of higher congestion are possible when there are sizeable proportions of alternative routing in the network and much of the trunk capacity is used for less efficient multiple link calls. The proposed solution is the use of a trunk reservation parameter (TRP) to reserve a number of channels on each link for directly routed traffic only. With the selection of a suitably sized value for this trunk protection scheme the bi–stable behaviour diasppears from the model and lower blocking probabilities are generated in the analytic and simulation models for overload conditions.

The second cause of deterioration in network performance which occurs some time after the network enters a congested state is caused by calls being blocked at the switching exchanges as the demand for call connection exceeds the capacity of the switching exchanges processing power[44]. In this form of blocking the congested state of the network causes increasing delays in the time taken to process a call and a reduced probability of success. This causes multiple attempts at call connection from the end offices compounding the problem and eventually leading to the release of common-control components in the exchanges, necessary for call connection, due to built in time-outs and the subsequent loss of calls. Here dynamic overload control (DOC) can be used to control the amount of traffic offered to a node, causing the busy tone to be automatically returned to a proportion of the calls into the node, without attempting call connection. Heanschke, in his paper on network management in the U.S.[44] describes TRP and DOC in the AT&T network and describes a two-level implementation. Calls with a low completion probability based on real-time measurements are labelled as hard-to-reach (HTR). These labels are distributed to affected parties by CCIS signalling methods and trunk reservation and DOC are selectively applied to these destinations when certain congestion levels are reached. The remainder of the network is not subject to the same restrictions until a second, higher, level of congestion develops.

3.4 Simulation Model Specification

3.4.1 Network Characteristics

The definition of a nodal model is complicated by the differences in the way that nodal exchanges of circuit switched exchanges can approach the task of call connection using different technologies. In order to develop a simulation model capable of mimicing the effects of these different approaches a generalised structure was developed for the nodal design which incorporates a number of variable features. By careful choice of these variables it is possible to tailor the model to reflect the processing capabilities of the exchanges in a range of networks.

The structure of the nodal model is shown in figure 3.5. The processing centre at the heart of the model implements the call connection protocol. Its performance is determined by a number of parameters and the availability of the processing centre is determined by the state of the 'sender' pool. There are three parameters which characterise the processing center; the 'acquisition time', the 'release time' and the 'maximum waiting time'. The acquisition time defines the delay a new call connection message or 'initiate' packet experiences on arrival at the processing centre before it can be retransmitted out on the next link of its circuit. The release time is defined as the additional processing overhead the node incurs when an initiate packet acknowledges its safe reception at its tandem or destination node. These delays represent the time taken by the exchange in the assignment and subsequent processing time required by a common control component, or sender, at the initiation of a new call and its release when it is no longer required.

If a sender is not available the initiate message is stored in a FIFO queue until one is released. To prevent queues building up during times when the node is overloaded a third parameter, the maximum waiting time, defines the maximum period a message spends in the queue before it is automatically rejected. Other connection transmissions do not require a sender to be available as the bulk of the necessary processing for the call connections are considered to have been carried out and these messages are processed independently of the availability of the common control components

Connection Protocol

To model the exchange of call connection messages involved in establishing and clearing of calls in a circuit switched network, the simple protocol in figure 3.6 was adopted. A node acting as the origin for the call attempts to establish a circuit to the destination node by transmitting an 'initiate' packet. The receiving node transmits an 'ack' back and if the receiving node is the destination follows it with a 'accept' packet. If the node is a tandem node on a two link path, as shown, the tandem node attempts to retransmit the 'initiate' packet on its link to the destination. If it is unsuccessful it transmits a 'reject' packet, but if it is successful it receives an 'ack' followed by an 'accept' packet which it retransmits to the origin node and the connection is established. At a later date the origin node transmits a 'finish' packet which terminates the connection.

Traffic Statistics

The arrival rate of each source is modelled as a Poisson process, where the probability $P_n(t)$ of n calls arriving in a time interval t is given by

$$P_n(t) = \frac{(\lambda t)^n}{n!}e^{-\lambda t}$$

where λ is the average arrival rate. This leads to a probability density function for the inter–arrival time x, $f_X(x)$, defined by

$$f_X(x) = \lambda e^{-\lambda x}$$

This is the memoryless exponential function which is also used to describe the distribution of the hold time or call duration t, $g(t)$ where

$$g(t) = \frac{1}{\mu}e^{-t/\mu}$$

and where $1/\mu$ is the average call duration. This form of traffic arrival and service statistics use the same mathematical functions as the analytic models discussed in the next section, allowing direct comparison of the simulation and analytic results, within the limits of the other analytic approximations.

3.4.2 Software Implementation

We can now combine the connection protocol with the model of the node processing centre to give a complete specification of the software model used to simulate circuit switched networks. For clarity the protocol skeleton is broken down into three sections in figure 3.7(a) - (c) corresponding to the procedure for a node acting as an origin, tandem or destination node. Each of the nodes contains code to assume any one of these identities according to the traffic source. In the flow diagrams a number of well defined states can be identified in which the node may find itself at any one time. These states are represented by labelled boxes. The algorithm moves from state to state on the fulfillment of the Boolean conditions, shown in bold type, or the arrival of a packet. The states Waiting, Routing, Sender and Progressing all have timing variables associated with them, these being the maximum waiting time, acquisition time, release time and hold time respectively which were defined earlier. Overtime is then defined as a Boolean variable which is always false when any of these states is entered and is only set to true after the node has resided in that state for a period of time equal to its assigned timing variable.

Beginning with the origin node, each call request is dealt with in the following manner. Initially the node is said to be in a Null state before the request is generated. On generation the node enters the queue waiting for attention from the processing centre and if a sender is available within the allowed time interval the call proceeds to the routing stage. If no sender is made available the call is blocked and the node returns to the Null state. Assuming the Routing state has been successfully reached, but a free trunk is not found the node moves into a Sender state, and after an appropriate delay moves back to a Null state with the release of a sender. On discovery of a free trunk however, the node enters a Free state with the transmission of an initiate packet after the acquisition delay. An incoming ack packet causes an extra Sender state which behaves in exactly the same way as the Sender state described earlier, for the case of failure of the routing algorithm to find a free trunk. The node then continues to wait for a further packet indicating the overall failure or success of the attempt to establish a circuit and on reception clears the call entry or moves to the Progressing state respectively. On successful call connection and expiry of a period of time equal to the hold time for the call a finish packet is dispatched along the circuit and the call cleared.

Nodes used as tandems in a two-link path undergo transitions from Null to Waiting states on reception of an initiate packet. Again the absence of a sender in the defined interval causes rejection of the call, with the origin node being informed by an ack, reject packet sequence. Successful acquisition of a sender is reported by a single ack transmission and the node enters the Routing stage. Failure to find a free trunk on the destination link causes the transmission of a reject packet and a sender state to be introduced. If a trunk is found then an initiate packet is transmitted and a Free state entered to await an ack packet, whereupon an independent Sender state is created. The Free state is then resumed to wait for a second packet informing the node of the circuits completion. For a two-link path this will always be an accept packet as the destination node is considered non-blocking. However in a network where multiple path routing is allowed a reject packet could be returned if the circuit was prevented from reaching its destination by the unavailability of a link on a trunk further down the chain. The node then remains in a Free state until a finish packet arrives, is retransmitted, and clears the call.

The protocol followed by a node acting as destination to a traffic source follows a simple protocol involving a fixed sequence of received and transmitting packets. On receiving an initiate packet the node transmits ack and accept packets to accept the call and enters a Free state. A sender is not required as there is no further network processing to be done, the call has reached its destination. The node then remains in the Free state until a finish packet clears the call entry.

Calls are prevented from interfering with each other by the assignment of unique identifiers to each call at each of the exchanges which make up the nodes of their circuit. These identifiers are carried by the connection messages and used as subscripts to identify the correct data items at each node in a similar way to the establishment of Virtual Circuits in packet switched networks. The packet format is reproduced in figure 3.8. Packets initiating new call connections inform nodes in the circuit of the address, at the previous node, where data relevant to the call can be found. The local acknowledgement packets carry the return address back to these nodes to complete a two-way path. This path is then used to direct the packets informing the circuit of the success or failure of each connection and the termination packet used to clear the call. The model presented here has been implemented using the OCCAM language on an IBM PC–AT with a B004 expansion card using the D700C Transputer development system from Inmos Ltd. A copy of the source code developed and used to carry out the subsequent simulation experiments presented in the next chapter has been included on the diskette accompanying this work. Full instructions on how to retrieve this code are detailed in Appendix B.

Data Structure

The data structure for each node maintains a complete picture of the nodes present state plus information on past events to reconstruct desired results. The structure can be separated into the three main component areas identified in figure 3.9, Call Progress, Node State, and Node History. The Call Progress section uses a series of linked lists to store each currently active calls progress. There is a separate list for each state a call may enter during its attempt to establish and hold a circuit. Node State variables keep track of link and sender availability for new calls and in the case of dynamic routing algorithms, the present routing strategy state. Finally the History section of the data structure stores a summary of the nodes response to traffic since the beginning of the simulation, especially where and how successfully calls have been routed. The links between the various components of the data structure indicate the main paths of data manipulation. The Call Progress is central to these manipulations requesting, accessing and modifying the nodal resources stored in the Node State data area and updating the routing algorithm in the case of adaptive routing strategies In contrast the History section passively collects statistics from the other sections of the data structure for extraction and use in the calculation of network performance.

3.5 Analytic Models of Dynamic Routing

3.5.1 Model Assumptions

In order to make the mathematical models for the behaviour of call blocking in circuit switched networks mathematically tractable it is necessary to make a number of assumptions about the network and traffic characteristics. The following assumptions are made in virtually all analytic treatments of the problem[33] and enable the formulation of the models which follow them.

1. Call arrivals for each origin-destination pair have a Poisson distribution.
2. Call holding times for each origin-destination pair have a negative exponential distribution
3. Link blocking probabilites are statistically independent. This is obviously an approximation when using paths of more than one link but allows each link of the network to be considered by a simple Markov model by decoupling it from the states of other links in the network.
4. Nodes are non-blocking. In the model we assume that blocking is caused only by the inability of the routing algorithm to find an available path through the network and is in no way constrained by the processing ability of the node.
5. Blocked calls are not reattempted. This preserves the Poisson statistics of the call arrival function.
6. The network is in statistical equilibrium. This allows the Markov models to be applied to the network.
7. Call connection times are considered to be negligible and a path is established instantaneously with no nodal processing delays. Again this allows a simple model to be constructed for call occupancy, ignoring the connection times which are small compared to the holding time of the call.
8. Overflow traffic is also considered to have a Poisson arrival distribution function. Again this is a mathematical approximation as the actual overflow traffic will not have a Poisson distribution. However this simplification allows the total traffic offered to a node to be considered as a single Poisson arrival stream with mean equal to the sum of the direct and overflow arrival rates.

3.5.2 Single Source Model

In general the analysis of dynamic routing algorithms is difficult because of the manner in which they stochastically divide the traffic overflowing the direct link amongst the alternative paths according to the network occupancy. The calculations involved in investigating their behaviour must therefore be solved iteratively as the two sets of variables, network occupancy and traffic division over the alternative paths, are so closely linked with each other. In order to explore the operation of different routing strategies in the simplest environment it is instructive to simplify the network to a point where the analysis becomes relatively straight forward. This can be accomplished by carrying out studies on a network with a single origin-destination pair and a number of alternative paths of fixed capacity to the destination.

By the introduction of the assumptions outlined in the previous section the blocking probability on any path in the network can be calculated by the application of the Erlang-B formulae[45], $E(T, C)$, which defines the blocking probability when T Erlangs of traffic are offered to a link of trunk capacity C and is given by the formulae,

$$E(T, C) = T^C/C! \Big/ \sum_{i=0}^{C} T^i/i! \tag{3.1}$$

This formulae comes from the treatment of the link occupancy as a Markov chain birth-death model with state independent arrival rate T. Now returning to the simple model, consider A Erlangs of traffic offered to an originating node for transmission across a network with trunk capacities on its direct and M alternate paths of N and $N_1 \dots N_M$ respectively. The overflow traffic from the direct link will be given by A' where

$$A' = E(A, N)A.$$

This traffic is then divided amongst the M paths according to the particular algorithm such that over any alternative path, p_m, a traffic intensity of A'_m Erlangs is offered where

$$A'_m = \alpha_m A'$$

and

$$\sum_{m=1}^{M} \alpha_m = 1 \quad \alpha_m \geq 0, \quad m = 1 \ldots M$$

giving a path blocking probability of $E(A'_m, N_m)$ for each path and an end-to end blocking probability, B, of

$$B = \sum_{i=1}^{M} E(A'_m, N_m)\alpha_m \tag{3.2}$$

3.5.3 Multiple Source Model

In a network with multiple sources using a routing strategy which offers alternate paths for calls blocked by the direct trunk group there will be competition between directly and alternately routed calls on some or all trunks of the network. In these cases the total traffic offered to a link will be a combination of the directly offered load and the overflow traffic from the other parts of the network, sharing the transmission bandwidth. The analysis of the problem, introduced here for the dynamic adaptive routing strategies, uses an extension of models used in the investigation of symmetrical[41,42] and asymmetrical networks with fixed, deterministic routing[43] and limits alternate routing to two link paths through a single tandem node. The notation follows Yum's treatment and is presented in full, repeating some of the earlier work for completeness.

Consider an M node fully interconnected network with trunk capacities, $N^{i,j}$, between any two nodes i and j such that $N^{i,j} = N^{j,i} \geq 0$ for all $i, j = 1 \ldots M, \quad i \neq j$. In addition let each link have a trunk reservation parameter, $r^{i,j} = r^{j,i} \geq 0$ for all $i, j = 1 \ldots M, \quad i \neq j$ such that overflow traffic is rejected if there are $r^{i,j}$ or fewer trunks available on link (i,j). Now let the directly offered traffic intensity between each of the nodes in the network be $A^{i,j}$ erlangs. Two different types of traffic can be identified as candidates for each link, labelled $A^{i,j}_{T,1}$ and $A^{i,j}_{T,2}$. $A^{i,j}_{T,1}$ is defined as the sum of the directly routed traffic $A^{i,j}$ and the overflow traffic offered to the link (i,j) as part of a tandem path from the other network traffic sources. This traffic is offered to the link if there are more than $r^{i,j}$ free links available on the trunk group (i,j). $A_{T,2}^{i,j}$ is just the directly offered traffic $A^{i,j}$ which is offered if there are $r^{i,j}$ or fewer free trunks.

From these definitions a simple Markov chain model of each trunk group can be formulated. The subscript for each trunk group is dropped for the first part of this derivation. Now forming a birth death model where each state in the chain reflects the trunk group occupancy, the birth rate of calls offered to the trunk group is state dependent (compare with to the state independent Erlang-B model) and in state i is defined by b_i where

$$b_i = A_{T,1}\mu_1 \quad \text{for} \quad i = 0 \ldots N - r - 1$$

$$b_i = A_{T,2}\mu_1 \quad \text{for} \quad i = N - r \ldots N - 1$$

and the death rate, d_i, is given by

$$d_i = i\mu_1 \quad \text{for} \quad i = 1 \ldots N$$

where $1/\mu$ is the average holding time of a call. The model is reproduced diagrammatically in figure 3.10 with the Erlang-B model for comparison. Assuming statistical equilibrium (assumption 6) the probability of n trunks being occupied, P_n is,

$$P_n = P_{n-1}\frac{b_{n-1}}{d_n} = P_0\frac{\prod_{i=0}^{n-1} b_i}{n!\mu^n} \tag{3.3}$$

where

$$\sum_{i=0}^{N} P_i = 1. \tag{3.4}$$

Calls will be blocked if the link is found in state P_N i.e. no available trunk groups, so if y is defined as the blocking probability on a link then

$$y = P_N = P_0\frac{A_{T,1}^{N-r} A_{T,2}^{r}}{N!} \tag{3.5}$$

Similarly the probability of a link accepting overflow traffic is the probability that it is has no more than $N - r - 1$ occupied trunk groups, which is the probability of finding it in one of the states $P_0 \ldots P_{N-r-1}$. So if we define x_r as the probability of accepting overflow traffic with a TRP of r then

$$x_r = \sum_{i=0}^{N-r-1} P_i = \sum_{i=0}^{N-r-1} \frac{A_{T,1}^i}{i!} P_0 \tag{3.6}$$

where P_0 can be calculated by summing the terms defined in eqn(3.3) with respect to eqn(3.4) and is given by

$$P_0 = \left[\sum_{i=0}^{N-r} \frac{A_{T,1}^i}{i!} + \sum_{i=N-r+1}^{N} \frac{A_{T,1}^{N-r} A_{T,2}^{i-(N-r)}}{i!} \right]^{-1}$$

The difference between offered traffic and overflow traffic now requires some distinction. The overflow traffic on a link (i,j) is the summation of the traffic overflowing each direct trunk group in the network and alternatively routed over a path containing the link (i,j). However the traffic offered to a link (i,j) is only that fraction of the overflow traffic which is capable of completing the tandem path to its destination. Because call set-up time is negligible (assumption 7), traffic blocked on the second link of the alternative path, at the tandem node, can be ignored and the offered traffic considered as only that fraction of the traffic which will complete its connection by successful trunk acquisition in this link. As an example if X erlangs of traffic overflowed from link (i,k) onto (i,j) as a link in its alternative path, then only $Xx_r^{j,k}$ Erlangs of traffic would be considered as offered to the link (i,j), the remaining $X(1 - x_r^{j,k})$ erlangs being blocked at the tandem node j in the alternative path from i to k.

Returning to superscripted variables, we can now formulate an expression for $A_{T,1}^{i,j}$, the traffic offered to link (i,j) if it has more than $r^{i,j}$ free trunks. Let the proportion of traffic overflowing link (l,m) and offered to the tandem path containing node n be defined as α^{lmn} where,

$$\alpha^{lmn} = \alpha^{mln} \geq 0 \quad n, m, l = 1 \ldots M \quad n \neq m \neq l$$

and

$$\sum_{\substack{n=1 \\ n \neq m,l}}^{M} \alpha^{lmn} = 1$$

and similarly let the blocking probability and the probability of accepting overflow traffic onto link (l,m) be defined such that

$$y^{lm} = y^{ml} \quad m, l = 1 \ldots M \quad l \neq m$$

and

$$x^{lm} = x^{ml} \quad m,l = 1 \ldots M \quad l \neq m$$

Now consider a node k in the network and the traffic offered to link (i,j) due to traffic overflowing direct paths between node k and these nodes. Two terms arise from node k. One term arises due to traffic being blocked on link (i,k) and routed over (i,j) and (j,k). The overflow traffic is given by $A^{ik}y^{ik}\alpha^{ikj}$ and the traffic offered to the link (i,j) is then given by $A^{ik}y^{ik}\alpha^{ikj}x^{jk}$. Similarly direct traffic blocked on link (j,k) causes the second term, giving an offered traffic of $A^{jk}y^{jk}\alpha^{jki}x^{ik}$ erlangs on link (i,j). The traffic paths producing these three terms are illustrated in figure 3.11 for a single alternate node Summing over all the nodes in the network adding the direct traffic over the link the traffic $A^{i,j}_{T,1}$ is given by

$$A^{i,j}_{T,1} = A^{i,j}_1 + \sum_{\substack{k=1 \\ k \neq i,j}}^{N} A'^{jk}_1 y'^{jk} x'^{ik} \alpha'^{jki} + \sum_{\substack{k=1 \\ k \neq i,j}}^{N} A'^{ik}_1 y'^{ik} x'^{jk} \alpha'^{ikj} \tag{3.7}$$

Solving the series of equations defined by eqn(3.7) for each link in the network by iteration allows the end-to-end blocking probability for each link, $EEBP^{i,j}$ to be calculated as,

$$EEBP^{i,j} = y^{i,j} \sum_{\substack{k=1 \\ k \neq i,j}} \alpha^{ijk}(1 - x'^{ik} x'^{jk}) \tag{3.8}$$

from which the total network blocking probability, BP, can be calculated as

$$BP = \frac{\sum_{\substack{i,j \\ i<j}} EEBP^{i,j} A^{i,j}_1}{\sum_{\substack{i,j \\ i<j}} A^{i,j}_1}. \tag{3.9}$$

3.5.4 Application to Dynamic Routing Strategies

By using the single and multiple source models, and in particular by manipulation of the α parameters it is possible to apply the model to a number of different algorithms to produce numerical solutions for comparison both with each other and with simulation results using the same routing strategies. In this section the conditions for the application of this model are defined for the range of algorithms studied. In the single source model $\overline{\alpha}$ is a vector of size M with components,α_m, which defines the probability of selecting alternative path m. In the multiple source model $\overline{\alpha}$ is a three dimensional matrix, $M \times M \times M$ whose components α^{ijk} define the probability of selecting node k as a tandem node for traffic overflowing from link (i,j).

Random Routing

This is the simplest routing strategy which can be modelled using the formulae derived in the previous sections. In this routing strategy each path has an equal chance of selection to carry traffic overflowing a direct path. For a single source model, random routing is analysed by assigning each of the traffic splitting parameters α_m in an M node network, the value $1/M$. Equally simple allocation of overflow traffic in the case of a multiple source M node network is modelled by defining the matrix elements of $\overline{\alpha}$ as

$$\alpha^{ijk} = \begin{cases} 1/(M-2) & \text{for all } i,j,k \text{ where } i \neq j \neq k \\ 0 & \text{otherwise} \end{cases}$$

Proportional Routing

In this strategy the probability of selecting a path is directly proportional to the number of free trunks on this path. For a single source with no interfering traffic the problem is trivial and similar to random routing. However when there are multiple sources with alternative routing which will interfere with each other the problem is less trivial. It can easily be seen that a global knowledge of the network is required to calculate each paths availability through the network for all alternative routes through tandem nodes. Here we develop equations which allow us to solve the analytic models for both cases.

In the single source model, for M alternative paths with trunk capacities $M_1 \ldots M_M$, the traffic is divided stochastically with probability of selecting path m, α_m given by,

$$\alpha_m = \frac{M_m}{\sum_{i=1}^{M} M_m} \quad m = 1 \ldots M$$

In the case of multiple sources the components of $\overline{\alpha}$ can be calculated in the following manner. For each link (i,j),first calculate the average trunk occupancy, $T^{i,j} = T^{j,i}$, given by

$$T^{i,j} = \sum_{i=0}^{N^{i,j}} iP_i$$

where P_i is the probability of exactly i trunks being occupied defined by eqn(3.3). The spare capacity for this link is then given by $S^{i,j}$, where

$$S^{i,j} = N^{i,j} - T^{i,j} \quad S^{i,j} = S^{j,i}$$

The spare capacity over each path from i to j via k, SP^{ijk}, is then given by

$$SP^{ijk} = \min(S^{i,k}, S^{j,k})$$

Finally the proportion of alternatively routed calls via node k, overflowing the direct link (i,j) is given by

$$\alpha^{ijk} = \frac{SP^{ijk}}{\sum_{\substack{k=1 \\ i,j \neq k}}^{M} SP^{ijk}}$$

This gives an expression for $\overline{\alpha}$ in terms of the traffic flowing in the network, which can be substituted into the series of equations defined in eqn(3.7). The mathematical solution applying the model to proportional routing is then the solution to these modified equations.

This strategy encapsulates an idealistic form of the adaptive, centralised Bell-Northern algorithm[27]. It uses a centralised routing processor which enables it to gather all the information it requires to implement a global strategy. As can be seen for the model above this is necessary as the selection probabilities depend on the entire network state. Extrapolation of the network state, as is done in the network algorithm to allow for the processing and transmission delays is not required here as the network is in equilibrium and this solution could either be thought of as the performance of the algorithm under continuous call–by–call update with zero delay or, as is the case for calculation of mathematical solutions to all of these models, the network operating in statistical equilibrium with no traffic fluctuations.

Learning Automata

Simple analytic analysis of linear stochastic learning automata[46] backed up by extensive simulation studies[47] have shown that in steady state conditions, if each automata actions is defined as the selection of an alternative path, the algorithms act in such a way as to converge to give a probability distribution which is a function of the blocking probabilities on those paths. In particular the linear reward-inaction (L_{R-I}) strategy and the linear reward-penalty (L_{R-P}) strategy have been extensively studied. These algorithms update their probability distributions on a successful action and an unsuccessful action respectively, according to the formulae

$$P_i(n+1) = P_i(n) + \sum_{j \neq i}(1-A)P_j(n), \qquad 0.0 < A < 1.0 \tag{3.10}$$

with

$$P_j(n+1) = AP_j(n) \tag{3.11}$$

for a successful action, i, where A is known as the reward parameter and the summation is over the set of actions, and

$$P_i(n+1) = BP_i(n) \tag{3.12}$$

with

$$P_j(n+1) = P_j(n) + \frac{(1-B)P_i(n)}{L-1}, \qquad 0.0 < B < 1.0 \tag{3.13}$$

for an unsuccessful action i where B is the penalty parameter and L is the number of actions. In each case $P_i(n)$ is defined as the probability of selecting action i at stage n. These forms of stochastic learning automata are found to converge to equalise the blocking probabilites on each path and the blocking rates on each path for (L_{R-I}) and (L_{R-P}) respectively.

Using these relationships it is easy to construct a set of equations for the single source models which can then be solved by numerical iteration to find the proportion of traffic flowing over each path. For M alternative paths and overflow traffic of A Erlangs a series of equations of the form

$$E(\alpha_i A, M_i) - E(\alpha_{i+1} A, M_{i+1}) = 0, \quad i = 1 \ldots (M-1)$$

can be solved to find the probability distribution over the alternative paths for the L_{R-I} algorithm with the constraint, as always, that $\sum_{i=1}^{M} \alpha_i = 1$.

For the same case the L_{R-P} converges to give a steady state probability distribution which can be found by the solution, to the series of equations

$$E(\alpha_i A, M_i)\alpha_i - E(\alpha_{i+1} A, M_{i+1})\alpha_{i+1} = 0, \quad i = 1 \ldots (M-1)$$

with $\sum_{i=1}^{M} \alpha_i = 1$.

In the multiple source model the algorithm at each node will again equalise the blocking probability or rate for each traffic source. This can be expressed by a set of equations relating the blocking probabilities and rates over each alternate path, for each source. For L_{R-I} the equations are of the form

$$(1 - x'^{ik} x'^{jk}) = K_{i,j} \quad \text{for } i, j, k = 1 \ldots M, k \neq i \neq j.$$

and for L_{R-P} the equation take the similar form

$$\alpha^{ijk}(1 - x'^{ik} x'^{jk}) = K_{i,j} \quad \text{for } i, j, k = 1 \ldots M, k \neq i \neq j.$$

where **K** is two dimensional matrix $M \times M$ whose components $K_{i,j}$ are independent constants in each case.

Dynamic Alternative Routing

A simple analysis of DAR[39] has shown that it acts to equalise the blocking rates on each alternative path, for each traffic source. This means the solution to the analytic model for DAR is identical to the solution for L_{R-P}. It also suggests that L_{R-P} and DAR should, in steady state conditions, perform equally well in networks

Optimum Routing

This is not a real routing strategy but an attempt to formulate, within the limits of the model, the optimum traffic splitting probability distribution for the single source models. Optimum in this context refers to the minimisation of the overall blocking probability of the network with no regard for individual GOS constraints.

For the single source model the optimum routing probability distribution can be found by solving

$$\min_{\overline{\alpha}}\Big(\sum_{m=1}^{M} \alpha_m E(\alpha_m A, N_m)\Big)$$

i.e. minimising the total blocking probability,B, over the probability distribution $\overline{\alpha}$

Calculating the optimum routing strategy for the multiple source model is more complex because of the larger number of degrees of freedom in the variable space and its non–linearity, but can be simply stated as finding

$$\min_{\overline{\alpha}}\Big(\frac{\sum_{\substack{i,j\\ i<j}} EEBP^{i,j} A_1^{i,j}}{\sum_{\substack{i,j\\ i<j}} A_1^{i,j}}\Big).$$

where the expression to be minimised is just the overall blocking probability of the network, BP, as defined in eqn(3.9).

3.6 Summary

Recent developments in circuit switched technology, which have allowed the implementation of advanced routing policies to be considered, have been discussed. The most important of these emerging policies are then described and categorised. In order to evaluate the affects of such routing strategies on the performance of networks in this category both simulation and analytic models are then formulated for their investigation. The analytic models allow network blocking probabilites to be calculated for each strategy under classical mathematical approximations. The simulation models, using the same statistical distributions as the analytic models, include several user definable parameters to allow the model to be tailored to model specific classes of switching exchanges. In the next chapter the analytic and simulation models introduced here are applied to both single and multiple source networks for the comparative performance evaluation of a number of fixed and adaptive routing strategies.

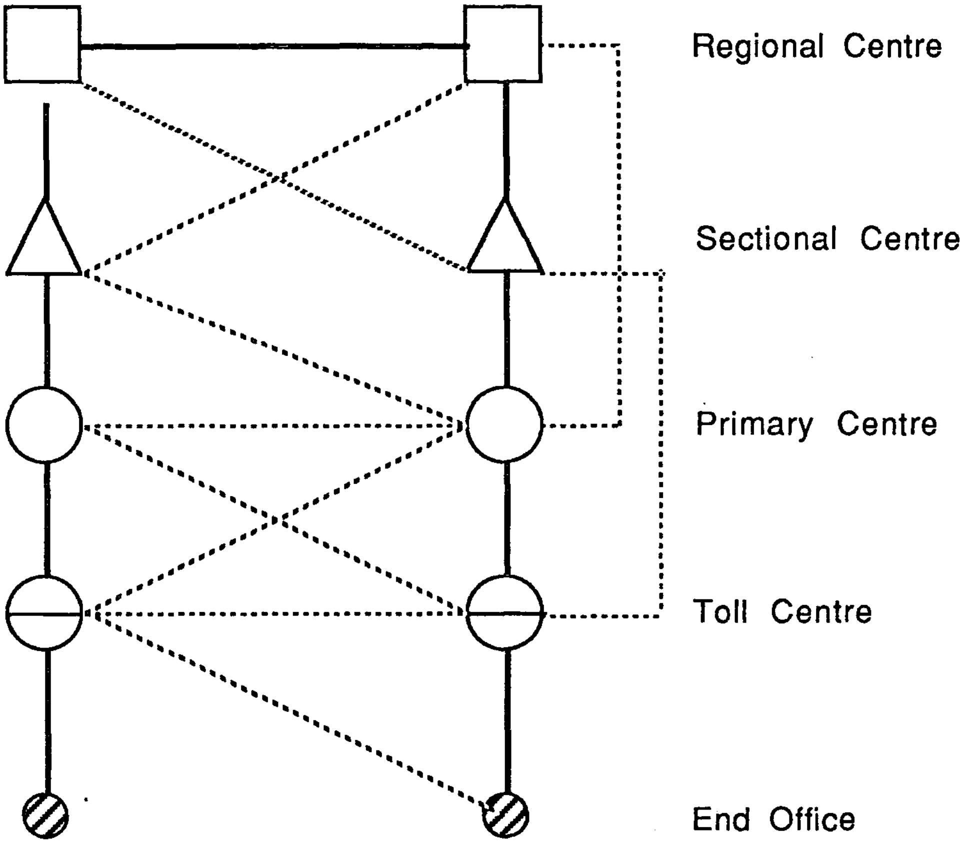


Figure 3.1 Hierarchical Network Structure

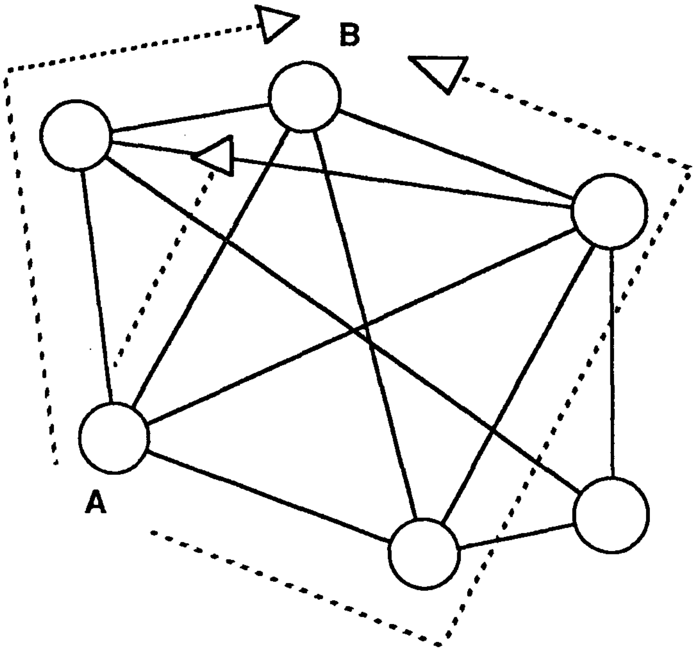


figure 3.2 Alternate routing in a non-hierarchical network

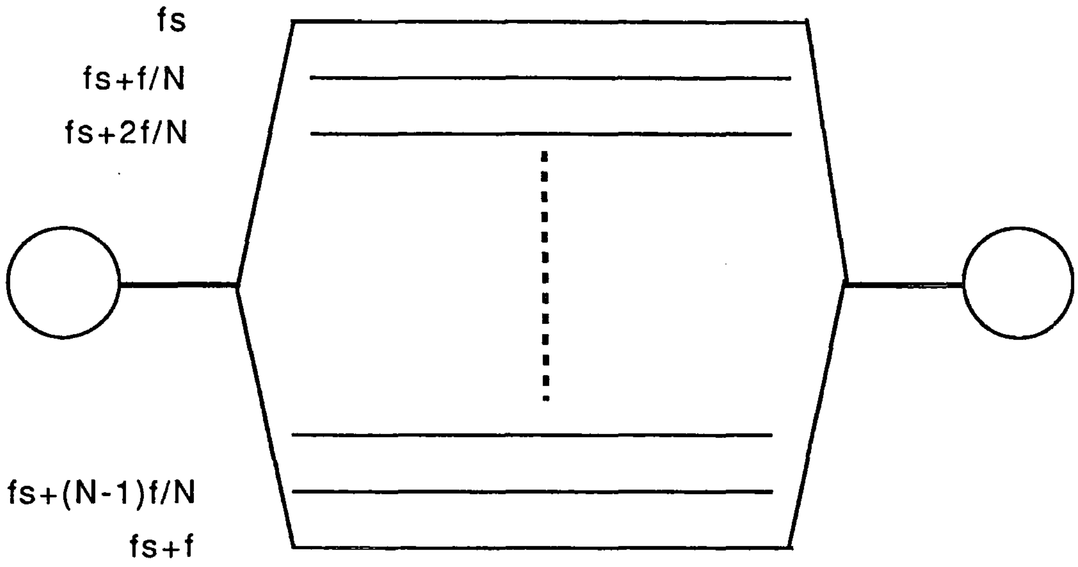


Figure 3.3(a) Schematic of Frequency Division Multiplexing

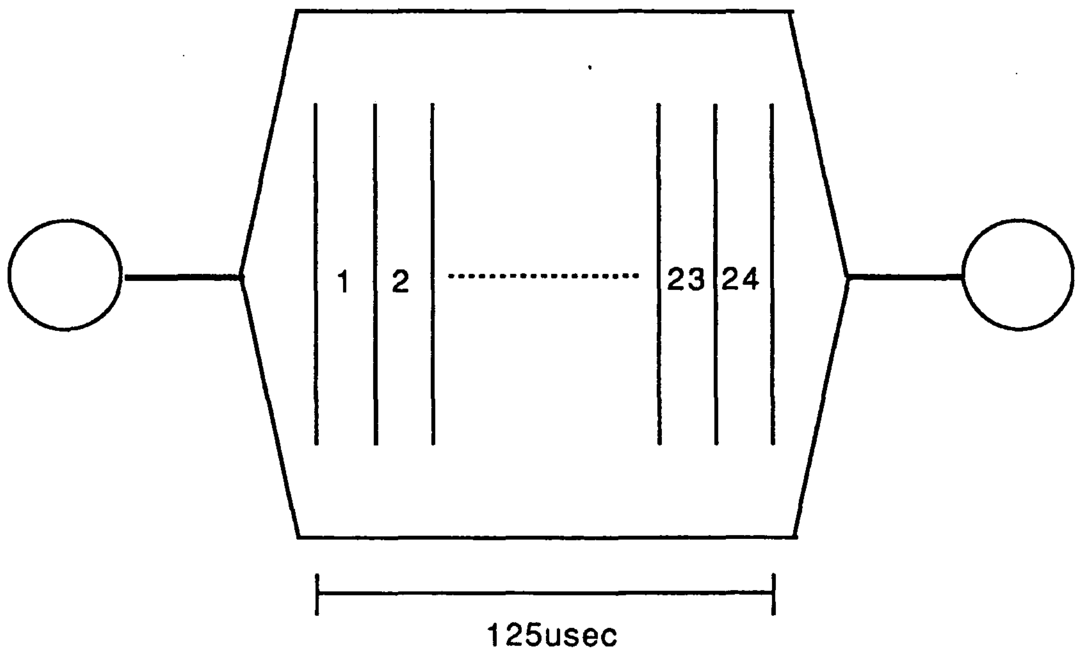


Figure 3.3(b) Frame format for Time Division Multiplexed Channel

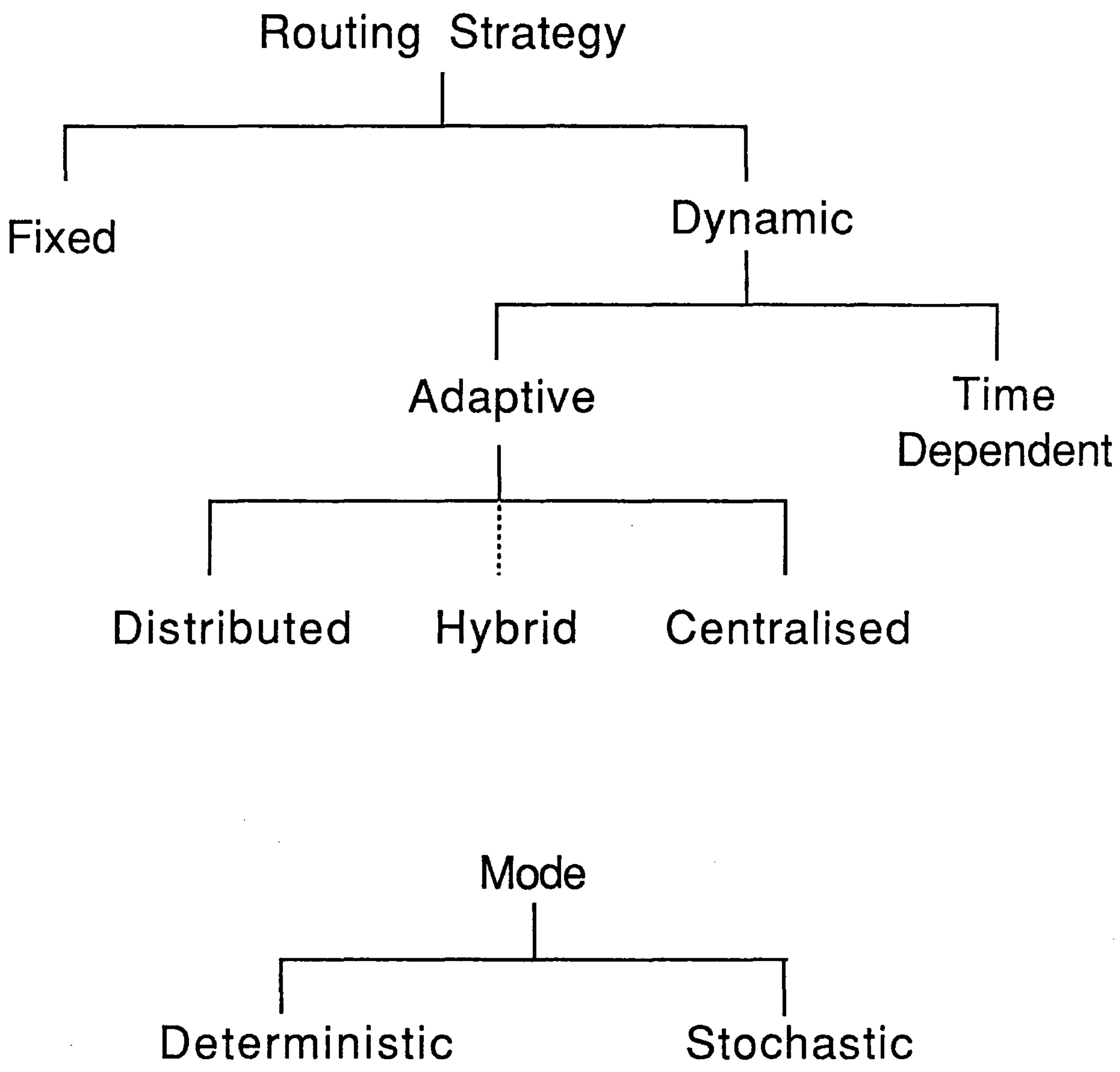


Figure 3.4 Classification of routing strategies

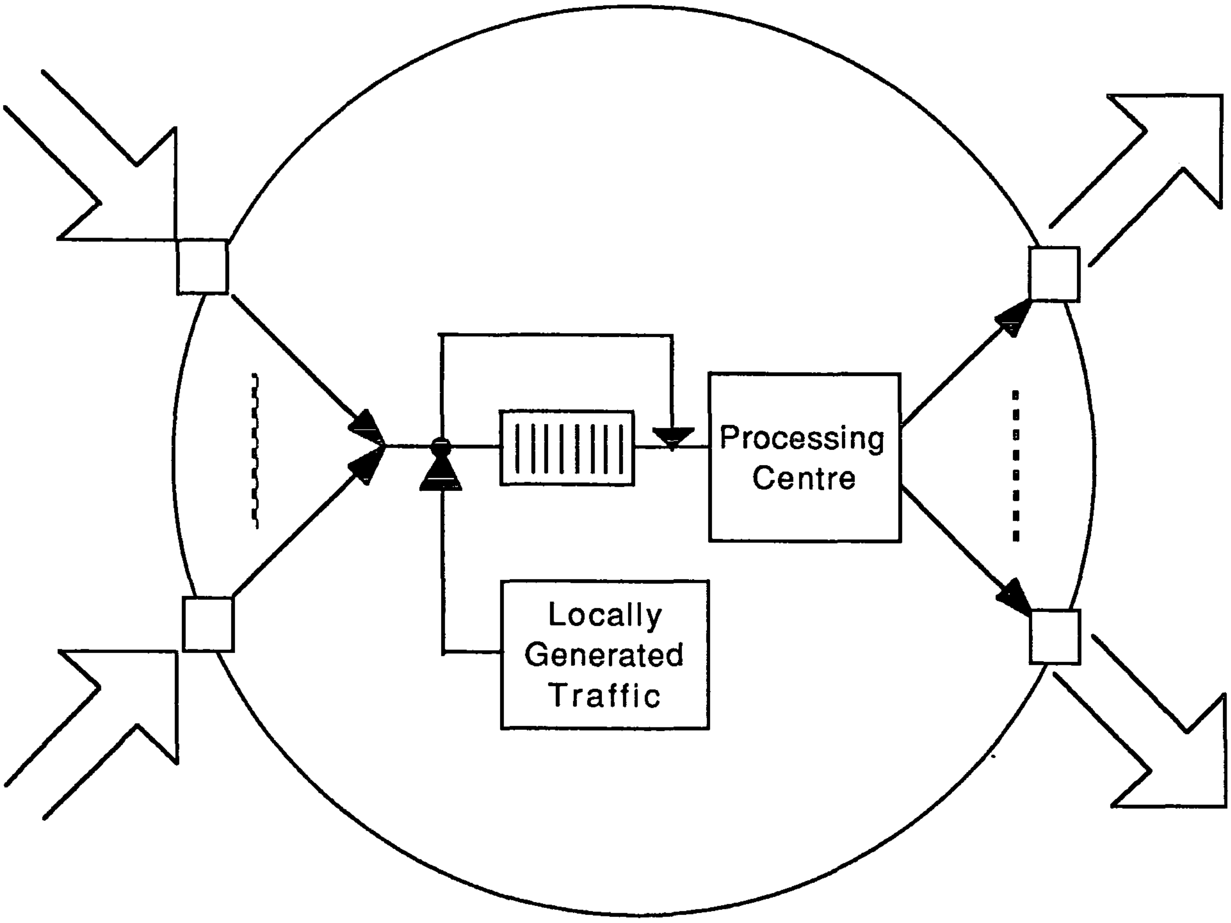
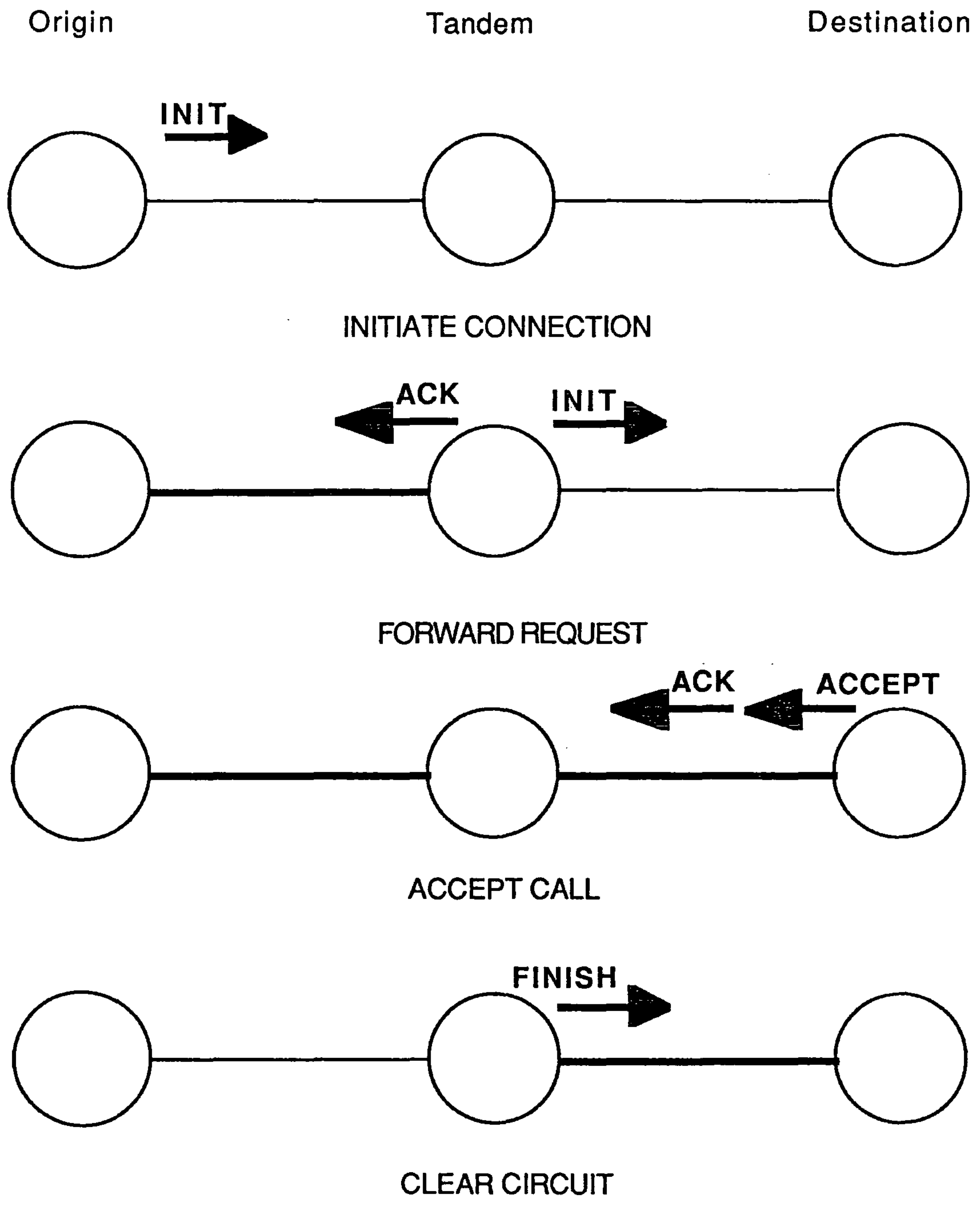


Figure 3.5 Nodal Model



INITIATE CONNECTION

FORWARD REQUEST

ACCEPT CALL

CLEAR CIRCUIT

Figure 3.6 Four stages in the connection protocol

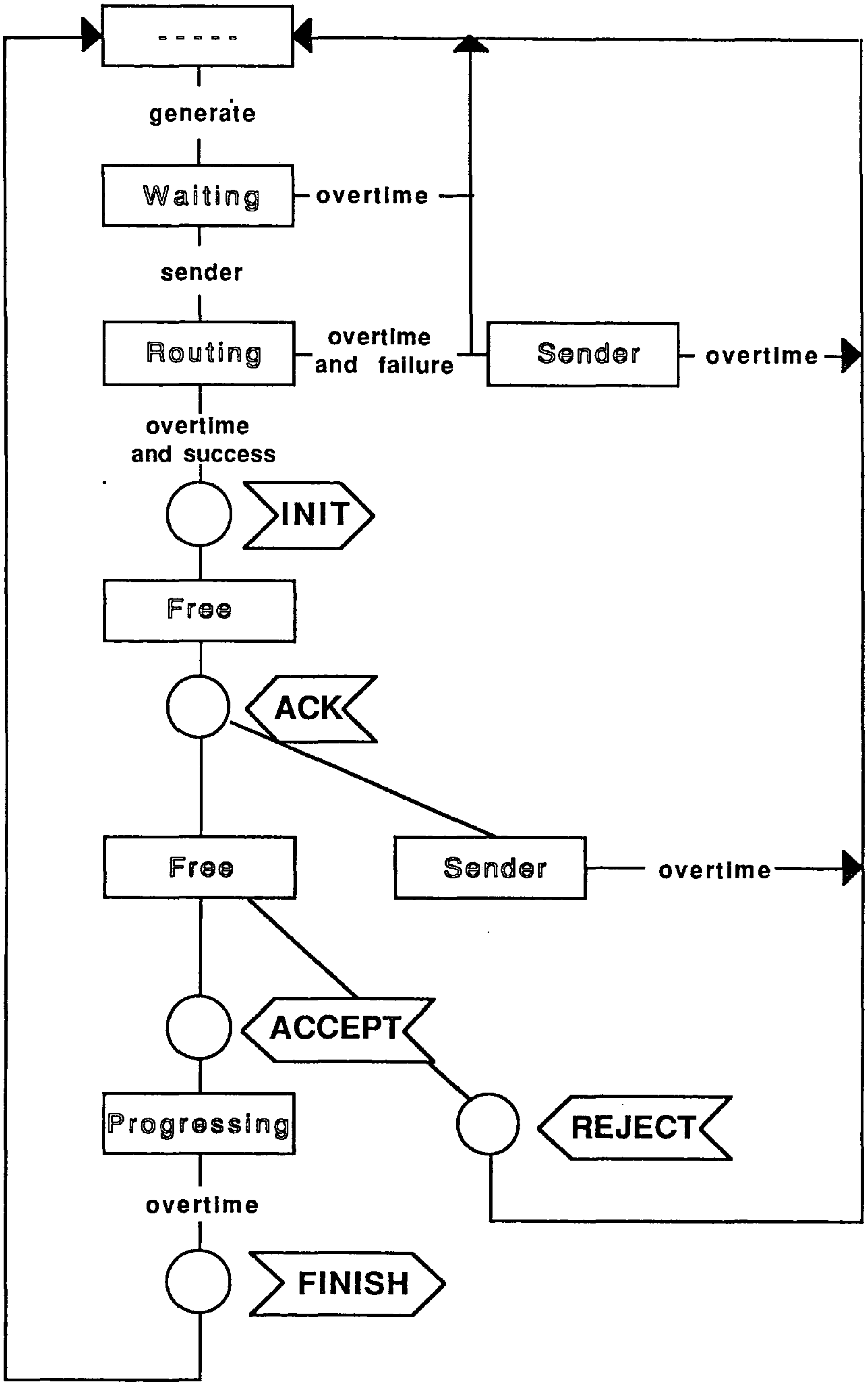


Figure 3.7(a). Origin Flow Diagram

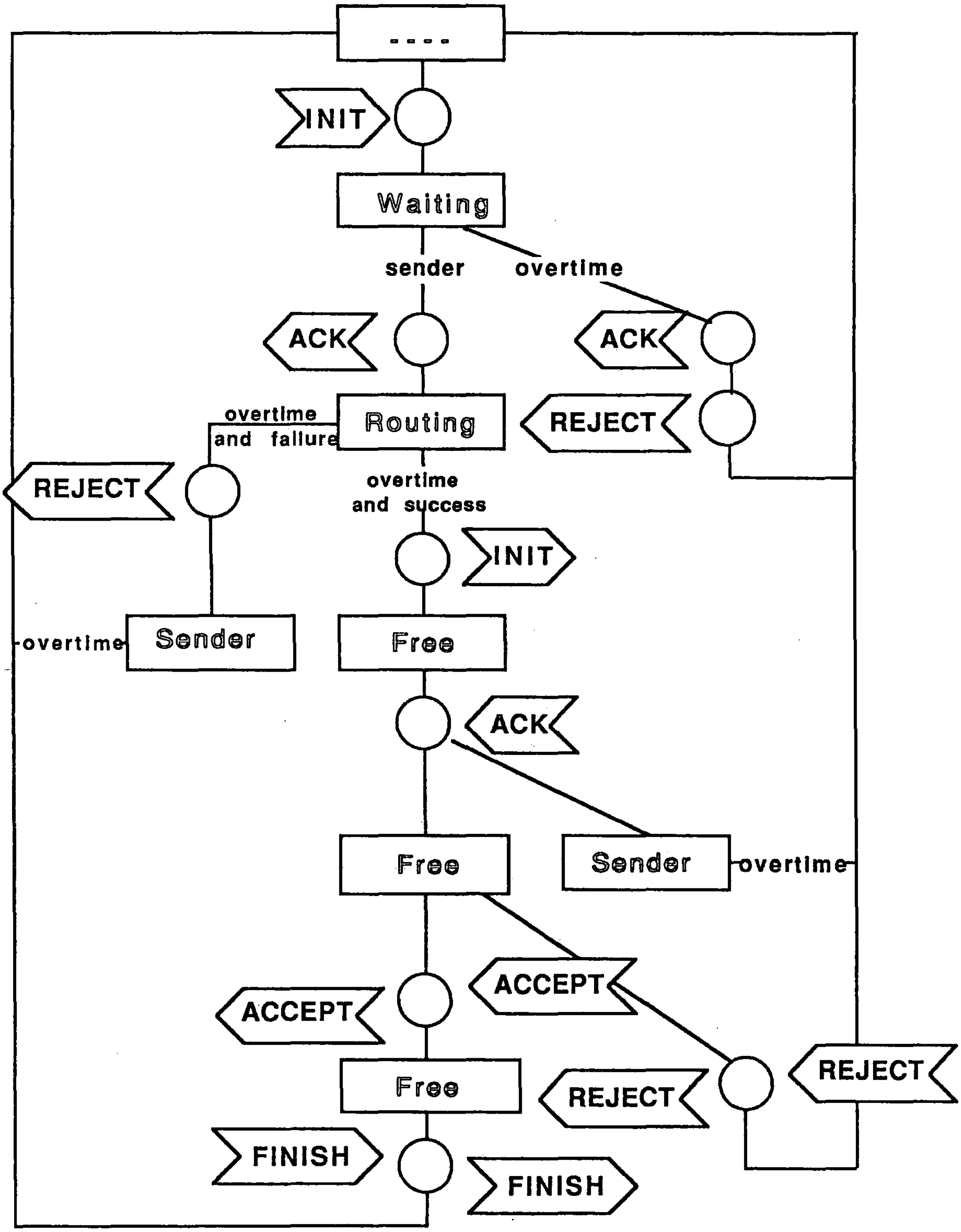


Figure 3.7(b) En-Route Flow Diagram

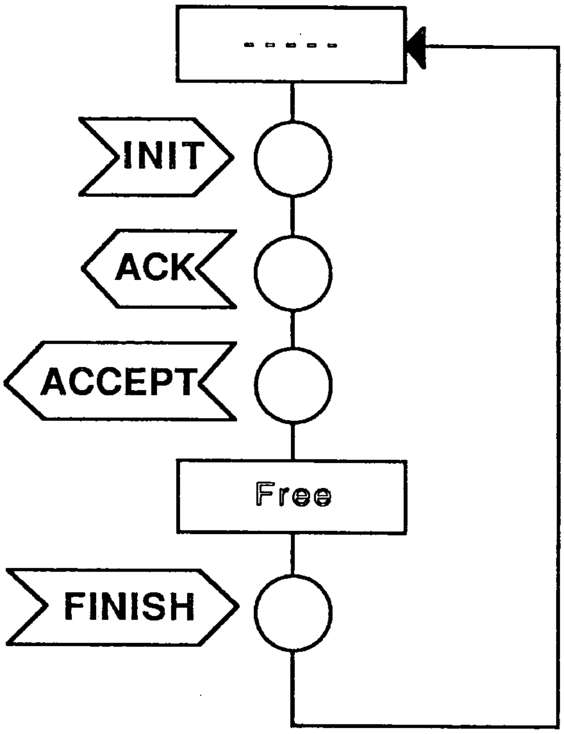
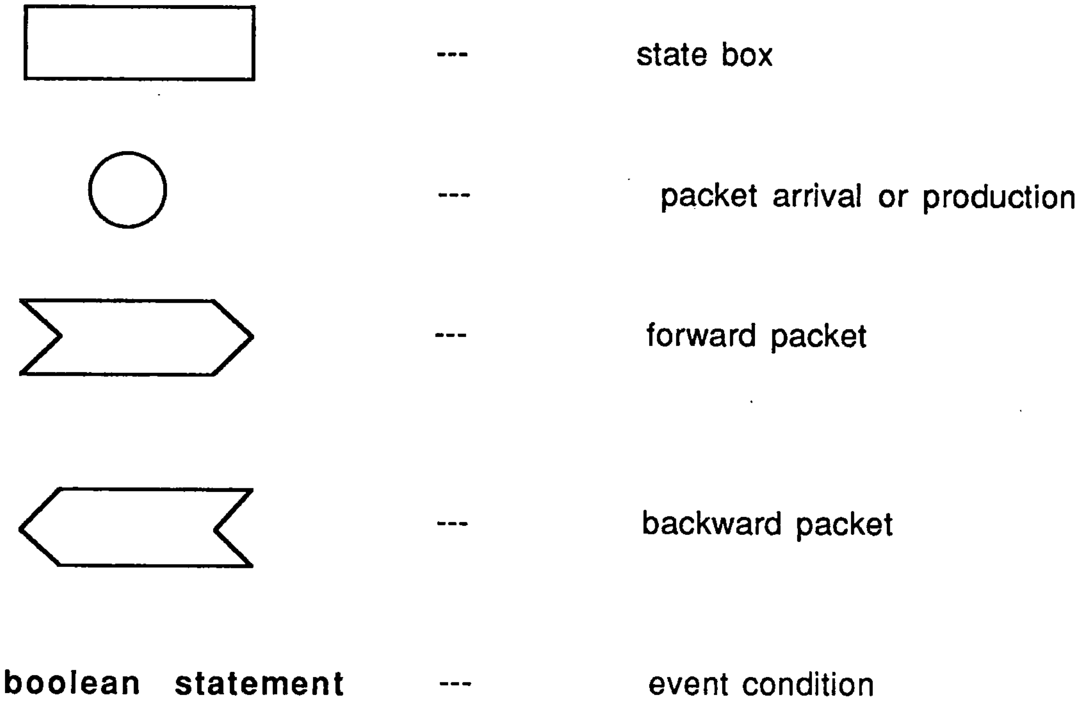


Figure 3.7(c) Destination Flow Diagram


--- state box
--- packet arrival or production
--- forward packet
--- backward packet
boolean statement --- event condition

Packet Formats

Initiate	1	O	D	BBBB
Acknowledge	2	FFFFF		BBBB
Accept	3	O	D	BBBB
Reject	4	O	D	BBBB
Finish	5	O	D	FFFFF

Index:	O	-	Origin
	D	-	Destination
	B	-	Address at previous node
	F	-	Address at Next Node

Figure 3.8 Packet Formats

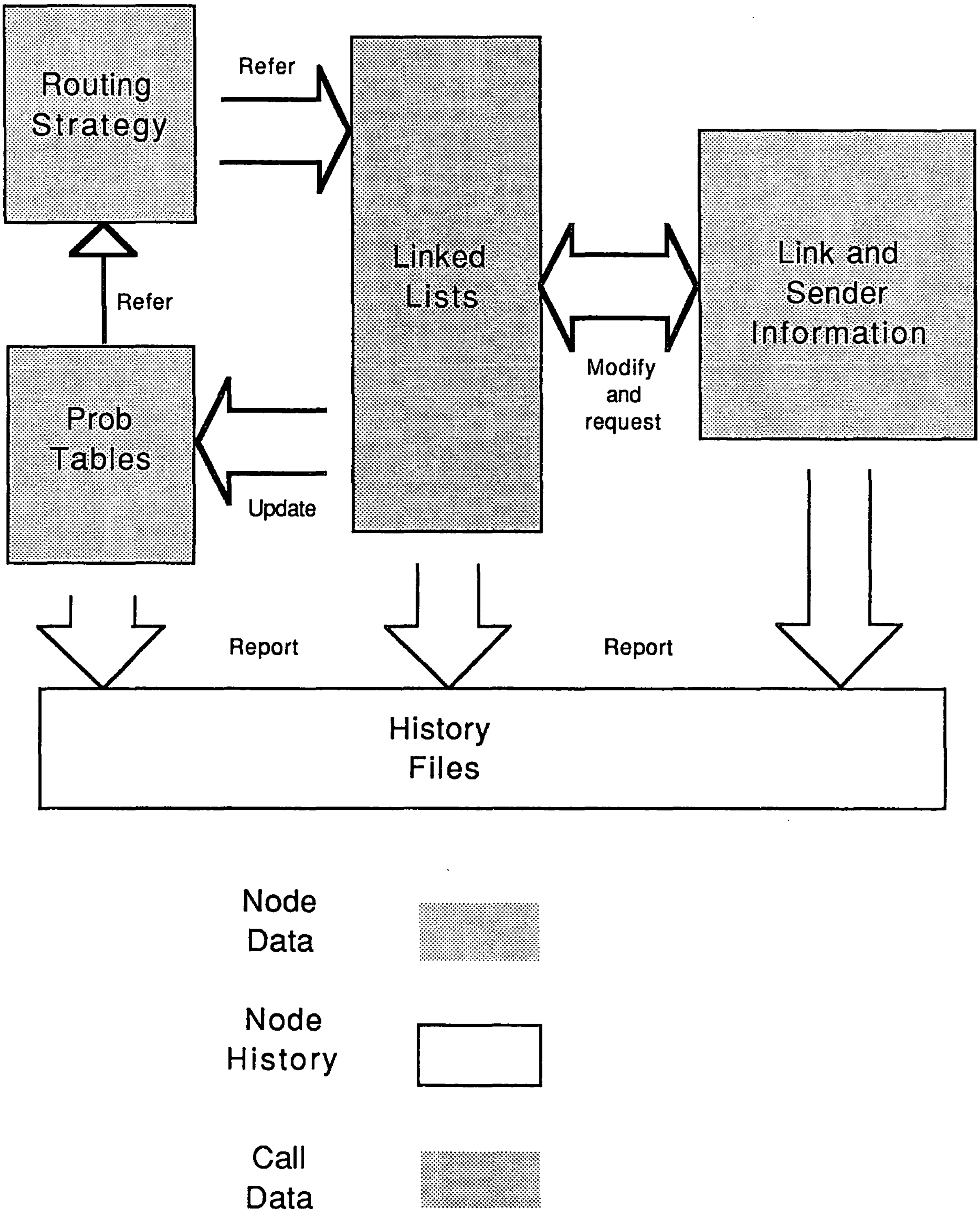
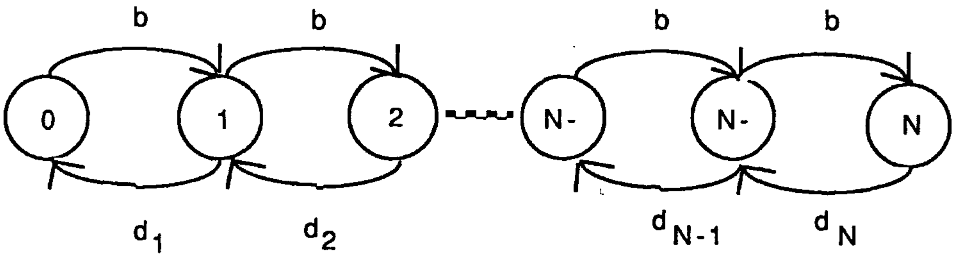


Figure 3.9 Data Structure

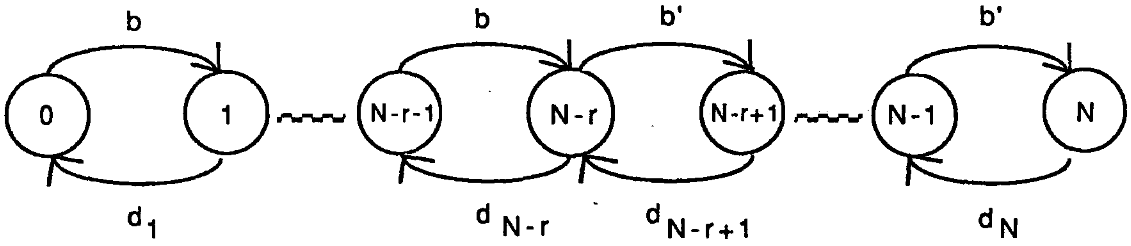
ERLANG-B FORMULAE MODEL



Birth rate, b = Arrival rate of offered traffic, A.u

Death rate, d_i = i.u

STATE DEPENDENT MODEL WITH TRP OF r TRUNKS



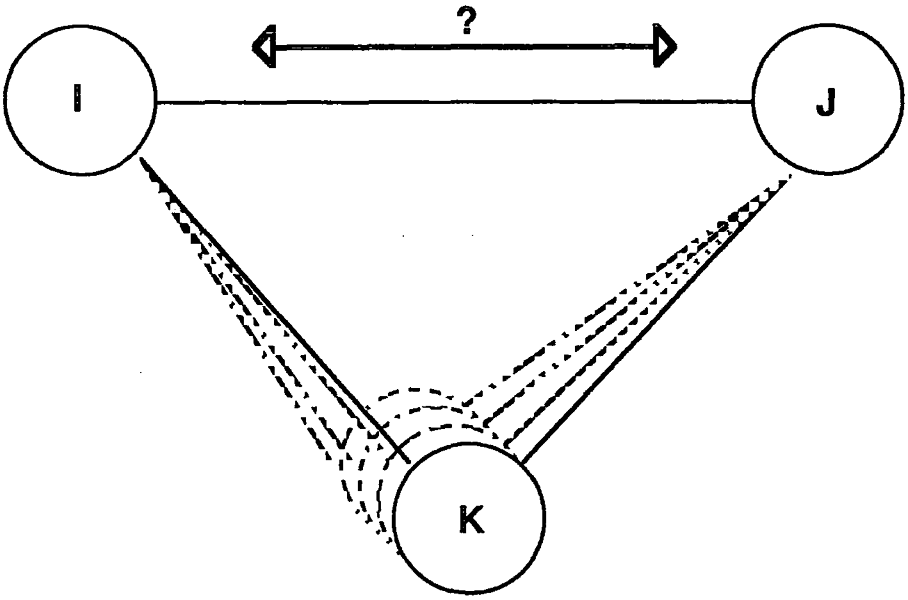
Birth rate, b = Arrival rate of direct and overflow traffic, $A_{T,1}$.u

b' = Arrival Rate of direct traffic only, $A_{T,2}$.u

Death rate, d_i = i.u

Where 1/u is the arerage call hold time

Figure 3.10 Markov Chain Models for Trunk Occupancy States



Traffic terms

1. Direct Traffic
2. Overflow traffic from i <-> k
3. Overflow traffic from j <-> k

$$\underbrace{\begin{array}{l} A(\text{total}) = A(\text{direct}) + \Sigma\ (i,k)\ \text{terms} + \Sigma\ (j,k)\ \text{terms} \\ E(A(\text{total}),C) = \text{Overflow on } (i,j) \end{array}}_{\text{For each link in the network}}$$

figure 3.11 Sources of traffic over link (i,j)

Chapter 4
Comparative Analysis of Learning Automata Performance as a Dynamic Routing Strategy in Fully Connected Networks

4.1 Introduction

In the previous chapter both analytic and simulation models were developed for the analysis of fully connected circuit switched networks using a variety of adaptive routing algorithms. In this chapter the models are applied to a series of problems for comparative performance analysis of the strategies under a variety of network conditions. The initial work involves single source problems to demonstrate the fundamental techniques used by each of the strategies in establishing stable traffic patterns. The next section establishes a framework for the generation of network models of different sizes and varying topological and traffic asymmetry. The assumptions applied to the analytic solution of these network models are then analysed using a simple example of two interfering traffic streams to establish criteria for the limitations of such models

The remainder of the chapter then draws upon the initial work and the multiple source models to investigate both the performance and potential instability of dynamic routing algorithms, both with and without the application of a trunk reservation parameter. Two major examples are examined in detail. The first example examines the effect of unequal traffic loading over a symmetrical network. The second example investigates the performance of a balanced traffic matrix over an increasingly asymmetrical traffic matrix

4.2 Single Source Experiments

4.2.1 Two Path Problem

In this section the simplest possible routing problem is examined in an effort to develop an understanding of the fundamental processes involved in each of the routing strategies under examination. The problem consists of a single traffic source generated at a particular node for transmission to a neighbouring node. Each call can be routed to the destination over one of two possible paths. Given the intensity of the traffic source and the trunk capacity of each link it is then possible to calculate the division of traffic over each path using the simple Erlang-based models developed in the previous chapter. From the traffic allocation the performance of each strategy can then be calculated. Initially each of two alternative paths were given the same capacity, 40 trunks, and the applied load was 65 Erlangs. This load is engineered to give a blocking probability of 1% if it were applied to a single link of 80 trunks, this being the sum of the two path capacities. Figure 4.1(a) is a plot of values calculated from the application of the models to this problem. It shows the total blocking probability for traffic at the origin node together with the blocking probabilities and blocking rates over each of the two paths, plotted against the probability of selecting path one, p (and therefore a probability of selecting path two of 1 – p).

Clearly the minimum blocking probability and equalisation of both the blocking probabilities and rates over each path occurs at a selection probability of 0.5 for each path. For this problem Random and Proportional routing also produce this traffic division, the former by definition and the latter by assignment. In addition the graph shows a well defined minimum value for the total network blocking function, increasing sharply away from the equal division of traffic.

The problem is now modified by the introduction of asymmetry into the allocation of trunk capacity over the two paths. Retaining the total trunk capacity and generated traffic, the trunk capacities of the two paths are altered to 30 and 50 trunks for paths 1 and 2 respectively and the models used to recalculate the same parameters as in the previous problem. Figure 4.1(b) re–plots the same five functions against the probability of selecting path 1, the path of smaller capacity. The total blocking probability of the incident traffic now reaches its minimum value at a new path selection probability and, as before, increases sharply to either side of the value. However, unlike the first example the point at which the blocking rates and blocking probabilities equalise are no longer equal and no longer coincide with the optimum selection probability, which lies between them.

To emphasise the differences in the two problems each of the trunk allocations was repeated for a new traffic source of 135 Erlangs, representing an overload of 100% from the engineered value. Figures 4.2(a) and 4.2(b) plot the results of the models for the symmetric and asymmetric cases. The point of minimum blocking and equalisation of individual blocking and blocking rates over each path coincide once again in figure 4.2(a). The divergence from this equalisation is clearly visible in the overloaded asymmetric case where the two functions over each path equalise once again to either side of the optimum selection probability, but at different values than for the case of an engineered load. In addition the minimum function is seen to be much less sharply defined, suggesting at this traffic level a wide range of traffic division probabilities would yield similar overall blocking probabilities.

Having established that the adaptive routing policies diverge from the optimum selection for asymmetrical trunk allocation, the next problem seeks to look into the way in which the selection probabilities differ and the effect this has on the blocking experienced by the traffic source. Returning to the asymmetrical trunk allocation of the previous problem, the path selection probability of the major trunk group is calculated for each of the routing policies studied over a range of traffic intensities from 1 to 135 Erlangs and plotted in figure 4.3. Random and Proportional routing are invariant in their division of the traffic and route calls over the major trunk group with probabilities of 0.5 and 0.625 (5/8) respectively. In comparison the two adaptive policies LRI and DAR follow closely the optimum path selection but fall away to either side as the traffic intensity increases. At large overload the selection probability of path 1 for each of the three adaptive strategies, LRI, DAR and the optimum routing policy begin to flatten out. The LRI plot forms the steepest gradient followed by optimum routing and then DAR. Each of the routines tends asymptotically toward a selection probability of 0.5 as T approaches infinity, but do so in accordance with their governing equations. LRI, which equalises blocking, quickly falls towards 0.5 with increasing traffic as the blocking probability over each path quickly rises towards 1.0 with traffic overload in accordance with the Erlang function. The blocking rates over each path are more sensitive to traffic levels over different paths and capacities and therefore approaches equal selection probabilities less quickly with increasing traffic.

Finally the routing probabilities can be converted into blocking probabilities to compare the invariant, adaptive and optimum routing strategies for this problem over the same range of traffic arrival rates. Figure 4.4 shows the blocking probability for each routing strategy applied to the asymmetrical network. Random routing aside, which behaves poorly at anything other than low traffic intensities, the performance of all the other strategies are almost indistinguishable. For this simple two path problem the introduction of any of the adaptive algorithms would seem to provide an admirable, near optimum, solution.

4.2.2 Multiple Path Models

An extension of the model described in the previous section can be formed by simply increasing the number of possible paths between the origin and destination. The probability distribution over the available trunk groups and the resulting performance of the routing strategy can be determined using the simple models outlined in the previous chapter. In the following experiment a pool of 360 trunks was divided between a total of N paths, for values of N from 2 to 10. The assignment of trunks in each case was made as asymmetrical as possible to create the most challenging environment for the routing strategies. A load of 336 Erlangs was then introduced, which was calculated to give a blocking probability of 1% over a single link of 360 trunks. The importance of selecting an intelligent probability distribution for traffic allocation over the set of paths can be demonstrated by comparing the performance of the optimal routing strategy over a range of traffic loadings from the engineered load (336 Erlangs) up to a 10% traffic overload in figure 4.5 with the same load applied to the models using random routing shown in figure 4.6. In each case the blocking probability using a single path is also plotted as a lower bound.

Proportional routing, LRI and DAR strategies are compared with optimum routing at the engineered load in figure 4.7. The plot shows the degradation in performance produced by the adoption of each strategy in comparison to the optimum models performance for each of the divisions of trunk capacity from 2 to 10. Proportional routing produces a consistent performance over this range while DAR and LRI become progressively worse as N increases. However for even the worst case of $N = 10$ all the routing strategies provide a performance that lies within 0.002 of the optimum solutions blocking probability. Once again it appears that the introduction of any intelligent routing algorithm with an appreciation of trunk capacities produces a near optimum performance for the problem of multiple path selection.

4.3 Multiple Source Models and Limitations

4.3.1 Network Models

In order to generate meaningful results on the application of dynamic routing strategies to circuit switched networks it is first necessary to define a set of test networks over which the strategies can be applied. These test networks will then provide a structured set of conditions over which each of the routing strategies can be assessed. Three areas of interest where identified; traffic distribution over the network; trunk distribution over the network and the size of the network itself. All networks within these three areas can be thought of as derived from the same basic reference network shown in figure 4.8(a), four nodes, fully connected and symmetrical with respect to traffic and trunk capacity. Each pair of nodes is connected by a trunk with capacity C and generates a traffic of intensity T Erlangs between them. From this primary reference a series of secondary reference networks can be built up of larger size. Each of these reference models is also a fully connected network with a number of nodes equal to an integer multiple of the number of nodes in the primary reference model. As before each link in the network has a capacity C and traffic between each node pair of T Erlangs. Both the primary and secondary reference models can then be used to define network models to investigate the effects of each of the other two axes of freedom, traffic and trunk allocation. For the following work C was given a value of 500 trunks and T then calculated to be 474 Erlangs, the traffic intensity required to produce a blocking probability of 1% over the direct path between two nodes.

The first area of interest, traffic distribution, is concerned with the effect of different traffic patterns on a network with fixed trunk capacities. A symmetrical network trunk distribution is used for all experiments involving traffic distribution to avoid complications involving additional effects due to asymmetrical trunk distribution. To define the traffic patterns the links in the network are divided into two groups, labelled '+' and '–'. The labelling is carried out in the following manner for each reference network. A list is made in which every link of the network appears labelled by their connecting nodes. The list is sorted using the identity of the smallest connecting node into ascending order. Links are then assigned '–' or '+' labels alternatively. The process is demonstrated for a four node network in figure 4.8(b), beginning with a '–' label on the (1,2) trunk. Having assigned each link a label, a set of network models are calculated with asymmetrical traffic patterns based on these labels. An asymmetry parameter, ϱ, is defined which modifies the traffic on each link according to its label. Links with a '+' are assigned a traffic source of $(1.0 + \varrho).T$ Erlangs while sources whose direct path lies over a link with a '–' label generate a reduced traffic arrival rate of $(1.0 - \varrho).T$. Clearly ϱ takes values over the range $0.0 \geq \varrho \geq 1.0$ for the generation of meaningful traffic arrival rates. This retains the same global traffic density across the network but provides a simple way of redistributing the traffic over each link. In the same way it is equally simple to form a set of test networks to investigate the effects of asymmetrical trunk distribution over a network. Given an asymmetrical trunk parameter ϑ, the trunk capacity of each link in the network can be redefined as $\lfloor(1.0 \pm \vartheta).C\rfloor$ for '+' and '–' labelled links. To avoid asymmetrical traffic affects interfering with the effects of the redistribution of trunk capacity, the traffic over each link is then redefined as the value which would give the same blocking probability as T Erlangs of traffic produces over C trunks.

Having decided to adopt this policy for the generation of network models, an important step can then be made to simplify the analysis and understanding of the generated results. Consider each of the labelled links in the four node network in figure 4.8(b). If we assume that the overflow traffic from each of the sources is routed independently of all the other traffic sources, then all sources with the same environment will react in the same way. By application of equation (3.7) to the network it can be shown that the set of equations defining the traffic over the network reduce to a single pair of non-linear equations. If labelling begins with a '–' link the total traffic over each type of link is given by

$$T(-) = T_- + 2T_- y_- x_- p_{--} + 2T_+ y_+ x_+ p_{-+}$$

where $T(-)$ is the traffic over each link with a '−' label and

$$T(+) = T_+ + 2T_- y_- x_+ p_{++} + 2T_+ y_+ x_- p_{-+}$$

where $T(+)$ is the traffic over each link with a '+' label. Alternatively if labelling begins with a '+ ' link the total traffic over each type of link is given by the formulae

$$T(-) = T_- + 2T_+ y_+ x_- p_{--} + 2T_- y_- x_+ p_{-+}$$

and

$$T(+) = T_+ + 2T_+ y_+ x_+ p_{++} + 2T_- y_- x_- p_{-+}$$

In each case T_+ and T_- are the directly routed traffic arrival rates over each type of link and both x and y have the same meaning as in the original formulation with subscript substitution. p represents the probability of selecting an alternative path of a type consisting of two links described by its dual subscript. By inspection the value of p_{-+} must be $1/(N-2)$, where N is the number of nodes in the network, as all overflow traffic from a '+' link must flow over one of $(N-2)$ identical paths each with probability p_{-+}.

All that remains is to calculate the remaining traffic splitting probability p_{++} and p_{--} using the equilibrium conditions of each of the network routing strategies. More important than the simplification of the four node model just described are the implications this simplification in calculation has for the analysis of larger networks. The mathematical computation for a four node network and a dynamic routing strategy lies within the storage and computational ability of reasonably powerful computers. However as the size of the networks increase the resources required to calculate the full model quickly exceed their availability. However for larger networks whose traffic and trunk asymmetry are defined in the same manner as those above these equations will also provide the solutions to those routing probabilities, from which the network performance can be calculated. To show this consider a network of size N, equal to an integral number of primary reference models. In this case the multiplier 2 will be replaced by $N-2$ in each of the terms above. Equally p_{-+}, presently 0.5, will be replaced by $1/(N-2)$ cancelling out the change in the multiplier in the terms in which it appears. p_{++} and p_{--} will have new values, but if these are taken to be the values calculated for the four node network, divided by the multiplication factor appropriate for the network, the multiplier and new probabilities will cancel out again. This arrangement will therefore lead to the same overall traffic over each link and from that the same blocking values in x and y, leading directly to a solution for the equations and constraints and therefore valid routing configurations. The results generated here can therefore be directly applied to larger networks without the computational effort of evaluating the larger topologies.

4.3.2 Model Limitations

In the formulation of the multiple source model for fully connected networks in the previous chapter a number of assumptions were introduced in order to make the model mathematically tractable. In general none of these assumptions are strictly true for a real network, however the magnitude of the approximations created by these assumptions will vary.

The mathematical approximations can be divided into one of two broad categories. The first category contains those approximations which are of little significance to this work or which, while limiting the solution, provide a valid solution for an equivalent real network under the same conditions. These assumptions are that of Poisson traffic arrival rates, exponential service call hold times, no blocking due to insufficient nodal resources, no multiple call connection requests and instantaneous connection times. The second category of approximations include those which again are necessary for tractability but which are clearly untrue and which cannot be as easily dismissed as acceptable. Within this category lie the approximation of independent blocking on each link of the network and the approximation that traffic overflowing direct links onto tandem paths does so with a Poisson distribution. The former approximation is invalid because of the presence of alternatively routed calls which acquire and release links at the same time, introducing a component of weak interaction between each link and every other with which it is able to form an alternative path. The second more important approximation of Poisson overflow traffic has been mathematically demonstrated to be invalid[48] but is necessary to remove constraints on the mathematical derivation of blocking models.

In comparison the simulation model of a circuit switched network does not need to conform to the assumptions imposed on its analytic counterpart. The model has a number of degrees of freedom which can be exploited to create a range of environments. On the one hand this ability can be used to create an environment which is closely related to the analytic model, or alternatively the simulation can introduce more realistic components and more closely emulate a real network environment. This ability to control the degree of approximation, and more importantly in this example to implement many of the mathematical restrictions forms a powerful tool for the investigation of the effects of approximations on the accuracy of the mathematical models.

In order to investigate the effects of the necessary but unrealistic approximation of Poisson overflow traffic outlined above, at various levels of overflow, the simple problem outlined in figure 4.9 was devised. In the problem two traffic sources at nodes X and Y route traffic to destination Z. Node Y attempts to route its calls directly to node Z and if unsuccessful drops the call and records it as blocked. The second traffic source at node X also tries to route traffic directly to Z, however if this is unsuccessful it then attempts the path XYZ. The result is competition between direct and overflow traffic over the link connecting nodes Y and Z. In the experiment the capacities of the links between nodes X and Y and between nodes Y and Z was fixed at 500 trunks and the traffic arrival rate at node Y for node Z was 500 Erlangs. In contrast the trunk capacity of the link between nodes X and Z was varied from 0 to 500 trunks. For each value of the trunk capacity between X and Y, the directly offered traffic over the link was calculated to give an overflow intensity of 5 Erlangs or 1% of the directly offered traffic between Y and Z. This generates overflow traffic onto the alternative path XYZ with probabilities between 1.0 for zero capacity on link XZ down to 0.01 for a capacity of 500 trunks. In the mathematical model, assuming Poisson statistics, the two traffic sources should combine and act as a single source to give a blocking probability for both sources of 0.042 for all overload probabilities.

The actual simulation results of blocking probability for overflow traffic on the link connecting Y and Z are plotted against percentage overflow in figure 4.10 with 90% confidence intervals. For large levels of overflow traffic the simulated points compare favourably with their predicted behaviour, demonstrating a blocking probability comparable with that of the major, direct, traffic source between Y and Z. However as the overflow traffic is made up of a smaller percentage of the directly routed traffic from X to Z the blocking probability it experiences begins to rise sharply. For overflow percentages below 15% the two traffic sources show markedly different blocking characteristics. This rise in blocking for the overflow traffic is due entirely to its arrival characteristic. Rather than the overflow from the direct link XZ being statistically independent, it tends to arrive in bursts whenever the direct link is saturated, separated by pauses as the arrival rate dips below the level necessary to fill the link. This leads to only a small number of any one burst having the chance to acquire an alternative path and the almost certain rejection of those arriving later in any burst. This behaviour leads to a higher than predicted blocking probability, under the assumption of independent arrivals, which increases as the traffic is increasingly concentrated into short bursts.

4.4 Routing in Unprotected Networks

4.4.1 Routing Algorithm Performance

In this section the analytic and simulation based models that have been developed are used to examine the performance of routing strategies under various asymmetrical conditions. The first model examines the performance of the dynamic routing strategies over a four node, fully connected network with traffic imbalance using an asymmetrical traffic parameter over the range $0.0 < \varrho < 0.5$. In this example labelling began with a '+' link. Figure 4.11(a) shows the theoretical blocking probability predicted by the analytic models for random, proportional, DAR/LRP and LRI strategies. The symmetrical reference model used to generate the traffic patterns assigned each link a capacity of 500 trunks and each traffic source an arrival rate of $1.1T$ where T Erlangs is sufficient to generate a blocking probability of 1% using direct routing.

Random, proportional and DAR strategies all exhibit similar blocking behaviour, the value increasing steadily as the traffic distribution is made progressively more unbalanced. Further, there is no real gain in using either of the more intelligent algorithms in preference to the simple random strategy until the traffic asymmetry has reached a significant value. LRI on the other hand displays a fundamentally different behaviour which can be separated into two distinct regions. For a low value of ϱ, analysis of LRI suggests that it will produce a performance exactly equal to that of any the routing strategies for the case of a completely symmetrical traffic distribution. Beyond this region it is found that no valid solution could be found for the allocation of traffic over alternative paths that would satisfy the constraints laid down by the mathematical model. The criteria for the model was then changed to find the solution which would minimise the deviation from the constraints rather than satisfy them and the results using this criteria form the second region which is labelled *LRI Routing.* This represents a 'best fit' solution under highly asymmetrical conditions and produces noticeably better average blocking performance than its counterparts over the same region.

Figure 4.12(a) plots the individual end–to–end blocking probabilities, or grade of service (GOS) that each of the sources in the model experiences for specific traffic parameter values. Again random, proportional and DAR strategies show similar behaviour, their traffic sources experiencing one of two possible grades of service. Links labelled '+' with increasing direct traffic show an increasing blocking probability in comparison with those labelled '–' from which direct traffic is removed. In each case the divergence increases with ϱ. LRI, while in the region where it is able to satisfy its analytic constraints, produces identical GOS values for each source, obviously equal to the average blocking probability. Beyond this region the grade of service on links with different labels does diverge in the same way as the other strategies although the difference remains considerably smaller.

In comparison with the analytic work, figures 4.11(b) and 4.12(b) show the simulated results for the same network model over the same range of traffic asymmetry. Both average blocking and individual blocking values show a good agreement with the predicted values at high values of the traffic asymmetry parameter, but return lower blocking probabilities for the more symmetrical network configurations. The form of the plots is similar throughout, with the exception of LRI routing near the point where the mathematical model is unable to find an exact solution as the asymmetry parameter is increased.

The first deviation from the analytic predictions can be explained by reference to the single source experiment discussed earlier, where it was shown that the assumption of overflow traffic having a Poisson distribution leads to an over estimation of the success of alternatively routed traffic for small overflow probabilities. As there is no network protection in these networks alternative routing will generally lead to interference and high blocking probabilities. Over–estimation of this effect by the analytic model will lead to it predicting a higher blocking performance than actually materialises. As the asymmetry is increased the resulting traffic imbalance will lead to greater overflow from the larger traffic sources, the main contributors to alternatively routed traffic, and a closer approximation to the conditions defined in the analytic model. The second area of interest, where LRI rises unexpectedly can be explained by reference to the algorithm itself. In this area several of the sources divide the traffic predominantly over one of the two available paths at equilibrium in the analytic model. However the algorithm LRI does not have a punishment component, only a reward for successful completion of a call over a tandem path. If therefore the algorithm rewards a path to the point where it is selected with probability 1.0, the algorithm cannot then continue to adapt, it is effectively locked in to that path. In the case outlined here, the algorithm overshot its predicted value, due to statistical fluctuations, became locked at unity and was unable to recover, giving a different GOS. One solution is always to retain a very small component of punishment, retaining the ability to recover from deterministic selections. However this can lead to a change in behaviour to a DAR/LRP based mechanism. A second solution might be to introduce a threshold to the selection probability over any one path. This would ensure the remaining paths always had some chance of route selection.

A second experiment was then formulated using the same reference network and labelling technique, with a trunk asymmetry parameter over the range $0.0 \leq \vartheta \leq 0.5$ to define a series of networks with increasing topological imbalance. In this case labelling began with a '−' link. The trunk capacity of each link in the reference network was 500 trunks and the traffic sources were $1.1T$ where T was again engineered to give a blocking probability of 1%. As the trunk capacities were modified the traffic for each value was recalculated to maintain the reference value of T for each link which would produce 1% blocking using a simple direct routing policy. The mathematical models of the previous section were used once again to calculate analytic solutions for each of the routing strategies. The average blocking probability for each strategy over the range in which the trunk asymmetry parameter was investigated is plotted in figure 4.13(a). In contrast to the previous example involving traffic asymmetry, random, proportional and DAR strategies all show improved average blocking probabilities as trunk asymmetry increases. Random routing actually outperforms the two dynamic strategies, with an increasingly superior average value as asymmetry increases. LRI maintains a constant performance while the model is able to satisfy the mathematical constraints. When this becomes impossible the criteria was again changed to calculate the probability distribution and the resulting blocking probability which provided the minimum deviation from the constraints. These values, plotted as the LRI routing curve fall away in a similar manner to the other strategies, while maintaining a higher average blocking probability.

Figure 4.14(a) shows the end–to–end blocking probabilities for each of the sources in the model. The pattern is similar to the traffic asymmetry experiment with two clearly defined, different grades of service for traffic directly routed over '+' and '−' links for all strategies except LRI. Traffic routed over '+ ' links see a lower blocking probability than their '−' counterparts. For LRI, in the region where the model is able to produce a solution which satisfies the strategies equilibrium criteria, the traffic is routed to achieve a consistent grade of service for each source. Where this relationship breaks down the familiar split in GOS appears but at a significantly reduced magnitude in comparison with the other strategies.

Simulation results for the average blocking probability and individual GOS for each of the traffic sources are shown as points in figures 4.13(b) and

4.14(b) with 90% convergence intervals. Figure 4.13(b) also reproduces the analytic results as curves for comparison. The simulation points show the same basic behaviour as predicted but once again return a lower blocking rate than suggested by the mathematical models.

The results from both the traffic and trunk based experiments can be explained in terms of the simplified formulae derived from examination of the labelling technique for the introduction of asymmetry. As was noticed earlier, if each source acts independently, that is without some sort of co–operation with the other sources in the network, overflow traffic from half of the links has no real choice in its division over the alternative routes and calls are routed equally between the possible paths. This leaves only the overflow traffic from the other links with a choice between a path consisting of either two '+' links or two '–' links.

For the case of traffic asymmetry the alternative path involving two '–' links appear more and more attractive as the direct traffic is reduced over them. However in this overloaded network a significant proportion of the traffic is still routed over via the overcrowded path involving the two '+'links. This causes the overall depreciation in average performance. This effect is shown clearly by the results in that despite the fact that '–'links in the network shows a rapid decrease in blocking probability with traffic asymmetry, the average performance is dominated by the traffic over the '+' links.

Examination of the LRI results and the routing probabilities show a fundamentally different type of behaviour in the allocation of traffic to alternative paths which cannot be explained by this approach. No solution can be found using the simplified models which will fulfil the mathematical constraints of the L_{R-I} model and the full four node model must be invoked. The equalisation of GOS, and therefore traffic, for the topologically symmetrical network, is accomplished by 'co–operative' rather than 'competitive' action. The division of overflow traffic from '–' links does not show the same impotence as that found with each of the other strategies. Instead there is no repeated pattern and each source divides the overflow traffic from its direct path in such a way as to equalise its GOS with all of the other sources in the network. Secondly the algorithm directs traffic over the underused path with a greater selection probability than its rivals. At a certain point in the introduction of traffic asymmetry the mathematical model for this action can only be solved by the introduction of values for some routing probabilities outside of the range $(0, 1)$, and so with no realisable implementation. Instead the 'engineered' solution calculates the minimum perturbation from the mathematical solution within the range of realisable probabilities. Traffic overflowing from a '–' link is routed in a deterministic manner over the preferred route, i.e. involving two '–' links, with probability 1.0. The remaining sources then route their traffic to achieve equalisation of individual blocking over each path. This is achieved by similar traffic splitting by each of the sources involved.

The explanation of the results for the asymmetrical trunk experiment follow a similar outline. The effect of shifting trunk capacity from '–' to '+' links can be examined by reference to the simplified model for each type of link, given an independent, competing routing strategy. Overflow traffic from a '+' link is faced with decision between two equivalent paths and always directs equal proportions along each path. In this experiment it is overflow traffic from the smaller link capacity trunks which are faced with a choice of paths between two high capacity '+' links or two low capacity '–' links. Random routing, which divides the traffic equally between each of the two paths, achieves the best average performance but, as the GOS measurements show, does so at the expense of traffic routed over the links of smaller capacity, which are given increasingly poor service as the asymmetry increases. DAR and proportional routing, recognising the different paths, route more traffic down the path with greater capacity. This causes more traffic on '+' links, the dominant traffic carrying trunks, and so a poorer average performance. However the difference in GOS between the different types of links is reduced in compensation, giving a fairer overall service.

Once again the LRI routing strategy approaches the problem from a global rather than local viewpoint, and while possible gives each source equal GOS over the network, allocating least traffic to the overcrowded '–' links. The equality of service eventually becomes impossible as the solution to the probability distribution lies outside its allowed solution set. In these cases the engineered solution results in the split in GOS and the accompanying drop in average blocking probability as the majority traffic carrying links experience fewer overflow calls than the strict solution to the routing strategy would assign to them. These 'engineered' solutions involve a similar distribution of traffic as in the previous experiment. Those sources which cannot achieve equalisation of blocking probability opt for deterministic routing down the most favourable path, leaving the remaining sources with an opportunity to equalise their blocking. In these cases each source again divides its traffic in a similar fashion.

Another feature of routing using the LRI strategy, contributing to its more balanced performance, is the severity of the division of traffic over alternative paths. It consistently adopts a heavier weighting to more favoured paths than the other strategies, demonstrated by the adoption of deterministic routing at high levels of asymmetry in both experiments. This is consistent with the single source results which showed LRI as being the most responsive to changes in traffic density, quickly rejecting paths as the blocking increased. Indeed the policy over reacts in comparison to the optimum solution and further over–selects the favoured path in comparison to then less responsive DAR/LRP policy which acts upon the less severe function of blocking rate for a given traffic density.

4.4.2 Network Instability

The classical instability characteristic reported by Krupp among others[41,43] for a symmetrical network with alternative routing is reproduced in figure 4.15. The plots show blocking probabilities associated with the analytic solution to the traffic assignment problem over a region of traffic arrival rates between 97% and 100% of a load engineered to give a blocking probability of 1% using direct routing. Within the region of instability the blocking probability clearly has up to three distinct solutions. On either side of this region the model returns a single monotonically increasing solution for increasing traffic arrival rates. Interestingly simulation results for the same network also plotted on the graph show no signs of instability or sudden increase in blocking over this region. They suggest a much lower, monotonic, steadily increasing blocking rate. This can again be attributed to the mismatch in actual and assumed behaviour of overflow traffic over this region.

Using the analytic model figure 4.16 traces the relationship between the call blocking probability and the traffic intensity, over the region of instability identified by the symmetrical model, as the traffic distribution is made progressively more asymmetrical. Figure 4.16(a) shows the instability behaviour over the same region of traffic arrival rate as the symmetrical model for each of the routing strategies random, proportional, DAR and LRI with a traffic asymmetry factor of 0.05. For each strategy the region over which instability is present has been reduced. In addition the region over which multiple solutions exist has moved. The upper and lower bounds have retreated in each strategy to lower arrival rates, the upper bound showing the more significant reduction. The plot also shows the strategies beginning to diverge. Random routing has retreated by the greatest factor, followed by proportional routing and DAR which show very similar characteristics, and finally LRI which retains the greatest instability region.

The traffic asymmetry parameter was increased to 0.1 and the analytic results plotted in figure 4.16(b). The blocking probability of the network using random routing no longer displays any form of instability over this region, increasing monotonically as the arrival rate increases. Extending the region over which the model was investigated showed no signs of the reappearance of instability at lower traffic intensities. DAR and proportional strategies remained almost inseparable in terms of their performance and retained the narrowest of instability bands which had further retreated to 97% of the engineered workload. LRI showed a similar narrow band of instability near the original lower bound in the symmetrical model. Increasing the asymmetrical parameter to 0.2 eliminates all forms of instability as shown in figure 4.16(c). Examining the performance of each of the routing strategies for traffic arrival rates values below those plotted confirmed that the instability effect did not return at a lower point on the performance curve. Instead each of the strategies showed a progressive decrease in blocking with reduction in traffic intensity down to a point of inflection where the gradient smoothly levelled off.

In a complementary experiment figure 4.17 traces the effects of introducing trunk asymmetry into the network over the region of instability identified for a symmetrical network. Figures 4.17(a) to 4.17(c) show the instability of networks constructed using asymmetrical trunk parameters of 0.2, 0.4 and 0.6 respectively.

The sequence shows a general reduction in the region of instability for each of the routing algorithms as the asymmetry increases. Random routing is the strategy most affected by the topological variations, and has all but lost any unstable behaviour at $\vartheta = 0.6$. DAR and proportional routing are affected to a lesser extent and follow each other closely in each of the plots. Only LRI is comparatively unaffected by the topological variations and retains a significant region of instability up to this point. In this experiment the reduction in the region over which instability is found is caused entirely by an increase in the lower bound at which the behaviour begins, the upper bound remaining constant throughout the topological variations. Finally with an asymmetrical trunk factor of 0.8 the unstable behaviour disappears as figure 4.17(d) demonstrates. All the routing algorithms show well behaved functions with a knee in their performance curves at the point representing the upper bound for instability. Interestingly the performance curves show the opposite behaviour to those of the highly asymmetric traffic curves. For highly asymmetric trunk topologies the lower curve of the unstable region dominates, whereas for traffic distributions the higher valued curve replaces the unstable region.

Notably the LRI algorithm undergoes a drastic change in behaviour between ϑ equal to 0.6 and 0.8. Its large unstable region has disappeared and been replaced by a shallow curve which outperforms the other strategies, having been the most expensive policy. Examination of the routing probabilities reveals that the extreme topological variation of the network does not allow the LRI mathematical constraints to be satisfied over this region. Further the actual solution is so far removed from any real implementation that the engineered solution leads to a strategy in which alternative paths are deterministically selected by each of sources, and it is this form which leads to the performance displayed in the final plot.

4.5 Routing In Protected Networks

4.5.1 Trunk Reservation Parameter Selection

The addition of a trunk reservation parameter to each link of the network results in two major advantages. First it prevents the routing of calls blocked on their direct path down alternative paths which would be better employed carrying direct calls. Secondly it eliminates the unstable behaviour suggested by analytic models at certain levels of traffic. The natural question which arises is how to select the value of the trunk reservation parameter (TRP). R. Gibbens suggested criteria for TRP selection[49] based on guaranteed levels of protection for each link or optimal traffic throughput based on the maximisation of a cost function for known levels of direct and overflow traffic. The actual selection is a simple trade–off between these two factors, protection and performance. A low value for the TRP will allow alternative paths to be used for directly blocked calls and maximise performance when those paths are made up of idle capacity. A high TRP will limit allocation of those same paths, reducing throughput if the network is lightly loaded but providing much needed protection if that capacity could also be used by direct traffic. The value chosen should allow sufficient flexibility to take advantage of spare capacity caused by unbalanced traffic patterns but discourage alternate routing over overloaded or busy sections of the network.

A slight complication arises in the selection of a TRP for a network when the trunk capacities in the network are unequal. To simplify this problem the network TRP is defined as the percentage of each links capacity set aside for direct traffic only, once the rest of the link is full. In this way a single number can be used to describe the protection applied to any network. Using this notation the effects of a variety of network TRPs have been evaluated over a series of traffic overload values at the two extreme topological configurations already introduced. Figure 4.18(a) shows the blocking over a fully symmetrical four node network with trunk capacities of 500 on each link when subjected to traffic overloading of up to 10% above the engineered value, T Erlangs on each link, where T Erlangs gives a blocking probability of 1% using direct routing. Figure 4.18(b) shows the results for an asymmetrical network subjected to the same series of traffic overload values over each link. In this example the trunk capacities in the network were 250 and 750 on alternate links, an asymmetrical trunk factor of 0.5. Both plots have a very similar form, clearly showing the advantage of a using a TRP over an unprotected network with an alternative routing strategy. After the initial fall each of the curves level out at the same blocking probabilities that direct routing would produce as the larger values of the TRP increasingly discourage any alternative routing. For larger overload values the value of the TRP is irrelevant above a certain value but for the lower traffic levels there is a minima corresponding to improved performance for a range of values where the blocking probability falls below that of fixed routing alone. From these curves a value for the network TRP can be selected based on forecast or projected imbalances in traffic to optimise potential performance gains.

4.5.2 Routing Algorithm Performance

Using the plots in figure 4.18 and the preceding discussion a trunk reservation parameter of 0.03 was selected. This level was felt to combine a judicious level of protection against traffic overload while retaining sufficient flexibility to develop alternative routing in the presence of asymmetrical traffic distributions. The basic models of traffic and trunk distribution outlined in section 4.3.1 were reformulated with the addition of this TRP to examine the effects, both on the performance of the dynamic schemes and their characteristics. Figure 4.19(a) shows the results of the analytic model applied to a four node network with C set to 500 trunks, T engineered to give a direct blocking probability of 1% and a traffic asymmetry parameter range of $0.0 \geq \varrho \geq 0.5$. The performance of a direct routing strategy is included for comparison with the alternative routing algorithms. The alternative strategies clearly give superior average performance in comparison to the direct policy for asymmetrical traffic and the benefit of such a strategy grows with the asymmetry. In addition the dynamic strategies outperform random routing consistently, although the difference is not substantial in comparison, demonstrating the gain in intelligent stochastic rather than entirely stochastic path selection.

Figure 4.20(a) shows how this performance is split between the two types of trunk group, with a clear division in the GOS of the two components which make up the network. Traffic routed directly over the lightly loaded or '–' links have excellent blocking characteristics with minimal blocking. Traffic over overloaded links are able to benefit from the opportunity to attempt alternative routes, subject to link availability and the TRP over that path. Fixed routing on the other hand suffers a rapidly increasing blocking probability over the overloaded links and the average performance deteriorates. In this case the TRP successfully allows alternative routing to improve the performance of the network when faced with spare capacity.

The major change in the form of the solutions for traffic asymmetry with a TRP in comparison to the unprotected network occurs in the behaviour of the LRI algorithm. There is no longer a region where the algorithm equalises GOS and all solutions must instead be engineered to produce the minimum perturbation from this ideal behaviour. In each case the alternatively routed traffic overflowing from the major traffic sources is routed deterministically over the path consisting of two '$-$' links. The other traffic sources divide their considerably smaller overflow traffic evenly over the two paths available to them, just as the other dynamic algorithms do. This is similar to *LRI Routing* which occurs in the unprotected network when the algorithm is unable to divide traffic is such a way as to satisfy the mathematical constraints.

Figure 4.19(b) and figure 4.20(b) plot the simulated results for the average performance and individual blocking probabilities for the same networks. The agreement between mathematically predicted and simulated results is extremely good. For this series of network configurations the assumption of overflow traffic with a Poisson distribution appears to be accurate. This is justified when it is considered that the models do involve large overflows, especially from the dominant '$+$' links. The effect of a TRP may also have helped to smooth out peaks of overflow traffic at the lower levels and present alternative paths with more statistically independent overflow components.

Figure 21 shows analytic and simulated results for a four node, fully connected network with initial trunk capacity C, equal to 500, and a traffic intensity over each link of $1.1T$ modified by a topological asymmetry parameter over the region $0.0 \leq \varrho \leq 0.5$ with the addition of a TRP of 3%. over each link. The plot shows how despite the increasing topological imbalance the TRP prevents major changes in network performance. Each of the adaptive strategies maintains a performance only marginally worse than that of fixed routing in the generally overloaded environment. The simulated points are within 1% of the analytic predictions suggesting that the alternatively routed traffic and direct traffic are modelled more accurately with the TRP once more. Figures 22(a) and 22(b) plot the individual GOS of direct traffic sources over '+ 'and '–' links. The GOS throughout the series of points is limited to a range of 2%, demonstrating how effective trunk reservation prevents excessive routing on alternative paths and divergence in the GOS of the two types of link.

Interestingly LRI behaves in a similar manner as outlined for traffic asymmetry, with deterministic routing down the preferred path by half of the traffic sources and similar traffic splitting to equalise blocking over alternative paths by the remaining sources.

4.5.3 Network Instability and TRP

Previous authors have already pointed out that the introduction of a TRP not only improves the performance of a routing strategy using alternative paths through a network, but also eliminates instability effects. This is clearly demonstrated in figure 4.23 where the instability curve of figure 4.15 is reproduced along with curves calculated over the same region and network configuration with a TRP. The second curve on the plot shows the predicted blocking behaviour for a TRP of 0.01 ,in which the last 1% of the trunk capacity of a link is reserved for direct traffic only. The plot shows no sign of instability and returns a much lower blocking probability than the average blocking probability of the unprotected network for traffic intensities above the lower limit of the onset of instability. The final curve on the plot shows the results of the mathematical model for a trunk reservation actor of 0.002. This corresponds to only the final trunk in each link of 500 trunks being reserved for direct traffic. Instability again disappears completely and is replaced by a smooth curve which follows the same form as the more protected networks performance plot, diverging to a higher blocking probability as the traffic intensity increases.

The fact that the reservation of a single channel in a large trunk group can make so much difference to the performance of a network can be explained by examining the way in which paths are allocated in the network. If the link is being highly utilised, at any one time there may only be a small number of the channels in any one trunk group available to carry a new call. It is the competition for these free channels which determines the performance of the network. Reservation of the last trunk for direct calls ensures that as the trunk group spends a greater proportion of the time full, these direct calls will receive increasingly preferential treatment over alternatively routed traffic.

4.7 Conclusions

To explain the way in which each of the routing strategies behaves in an unconstrained environment we return to the conditions under which each is governed. Random routing directs traffic equally over all the alternative routes at each node. Each node selection is isolated and does not react to trunk or traffic imbalances. In this work it represents an upper bound, or least intelligent option.

Proportional routing assigns traffic according to trunk availability, calculated using the set of probabilities P_0 to P_N defined in eqns(3.3) and (3.4). DAR uses not trunk availability but trunk blocking rate as criteria for traffic splitting, defined as the blocking probability over a link multiplied by the probability of using that link. Both strategies involve very different ways of selecting routing probabilities but produce very similar solutions under asymmetrical conditions. In both cases the routing policy at each node makes identical routing decisions as its network neighbours faced with the same environment. From the single source models in figures 4.1 and 4.2 blocking rate can be seen to be a convex function of selection probability. Similarly the plot of normalised spare capacity vs selection probability is also found to be convex. In the multiple source model this produces an array of constraints based on convex functions over the multidimensional routing space. This suggests that any solution will be unique and therefore by definition requires identically situated sources to behave in an identical manner to prevent multiple solutions by simply swapping indicies in the network.

LRI routing involves the equalisation of blocking over each of the alternative paths to a destination from each node. The blocking function is a strictly increasing but non–convex function of selection probability. Therefore it will form a non–convex set of constraint equations and the argument outlined for DAR and proportional routing cannot be applied. Multiple solutions can and do exist in the solutions to the network models used. In each case a new and equally valid solution can be formed by relabelling the nodes in the model. Configurations can be found which maintain identical traffic levels over each link while the selection probabilities of each of the sources over alternative paths are re–arranged. However a full explanation as to why the equalisation of the individual blocking probabilities for each source leads to an equalisation of the GOS of each source remains an open problem. Despite this the result remains one of the most important conclusions of this work.

The behaviour of the strategies when a TRP is applied is also quite different. Both proportional routing and DAR rearrange themselves according to their new environment, splitting the overflow traffic over the available alternate pathways in the same way as in the case of the unconstrained network. In each case the function used to select the traffic distribution over the alternative paths for each source, i.e. spare capacity or blocking rate, covers a range of values for the variable of path selection probability sufficient to allow an exact solution. In contrast the blocking probability over a link is relatively unaffected by the selection probability of a single traffic source using that link as an alternative route. In consequence LRI is often unable to equalise blocking over all its paths for all its sources, especially those with very different alternative paths to a destination and this gives rise to the *LRI Routing* solutions. The imposition of a TRP further inhibits this ability, by inhibiting the traffic and therefore the influence any one source may have on a link. Again this drives the LRI strategy to deterministic routing for traffic sources in which the maximum allowed traffic overflow is insufficient to equalise blocking, but in this instance the asymmetry required to initiate such action is very small.

The work on instability has two important results. First, it is clear from both the analytic and simulation models that the addition of even a small trunk reservation results dramatically improves the performance of the network. However, without such a device instability is predicted by the analytic models of all the dynamic strategies to varying degrees over a range of traffic and trunk distributions. In contrast the simulation models for a four node network over the affected regions suggest only a gradual increase in blocking producing a significantly better performance. This and other work at higher traffic intensities suggest that the analytic assumptions are not valid over these regions for small networks. For further work in this area the reader is referred to the recent thesis of N. Eshragh from Durham University[50]. In the next two chapters routing and flow control over another form of network is considered, a packet switched network, where traffic is divided into small segments and transmitted independantly. Although their basic function is very similar, the way in which each type of network operates is very different and the two are considered quite separately. Chapter 5 develops analytic and simulation based tools for the performance evaluation of generalised packet switched networks which are then applied to specfic examples in chapter 6.

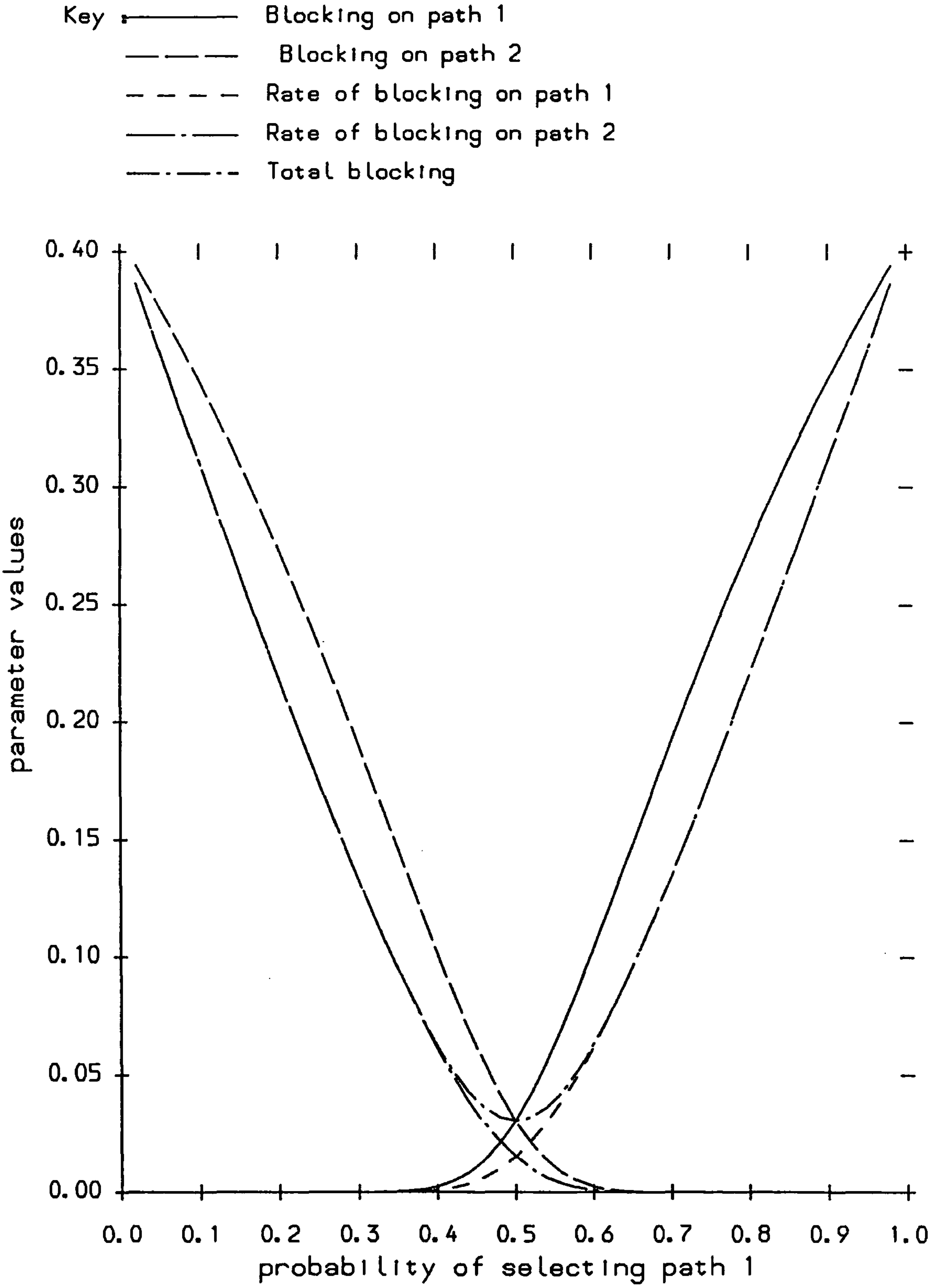


figure 4.1 (a)

performance criteria in a symmetrical network

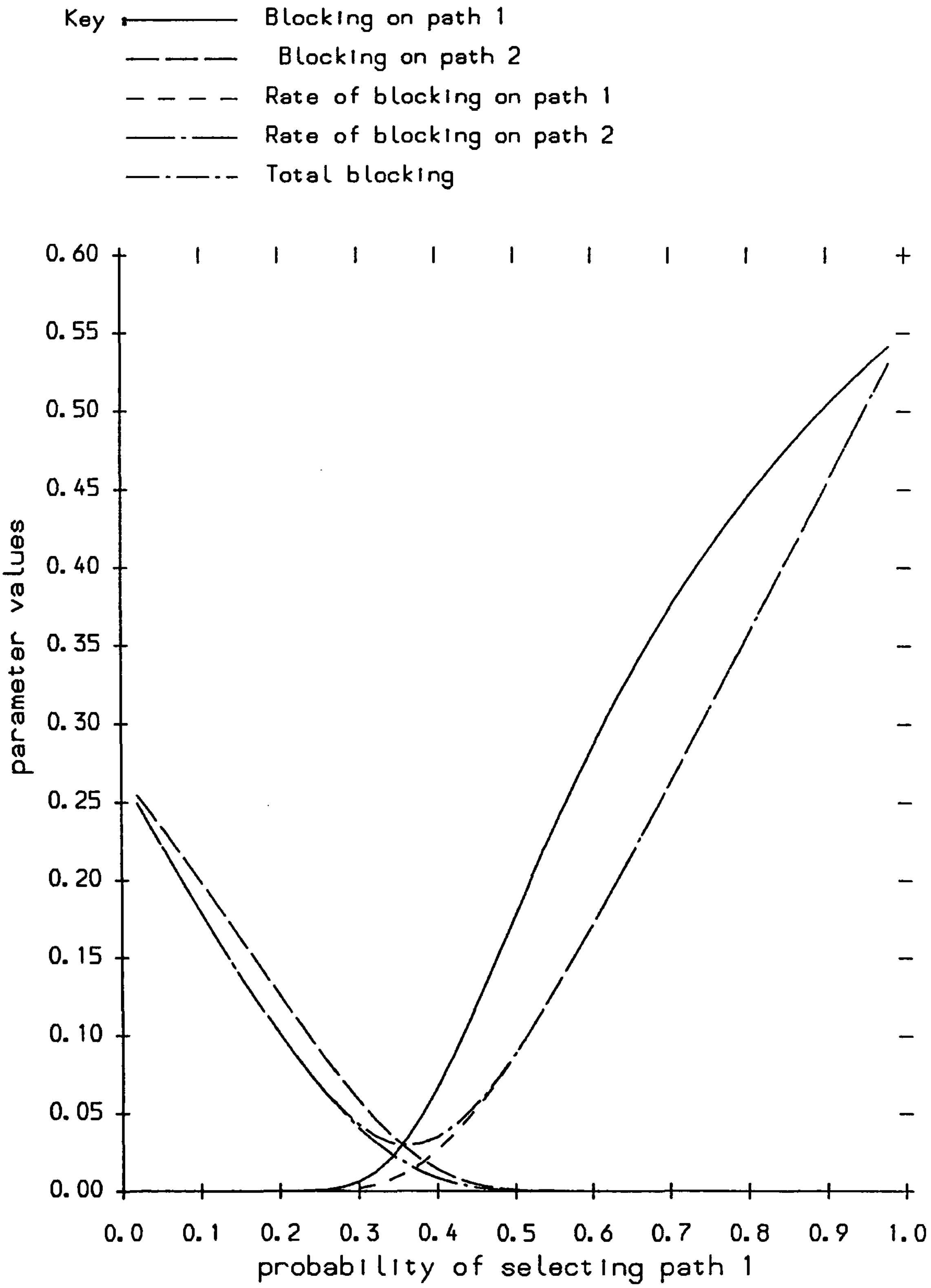


figure 4.1 (b)

performance criteria in an unsymmetrical network

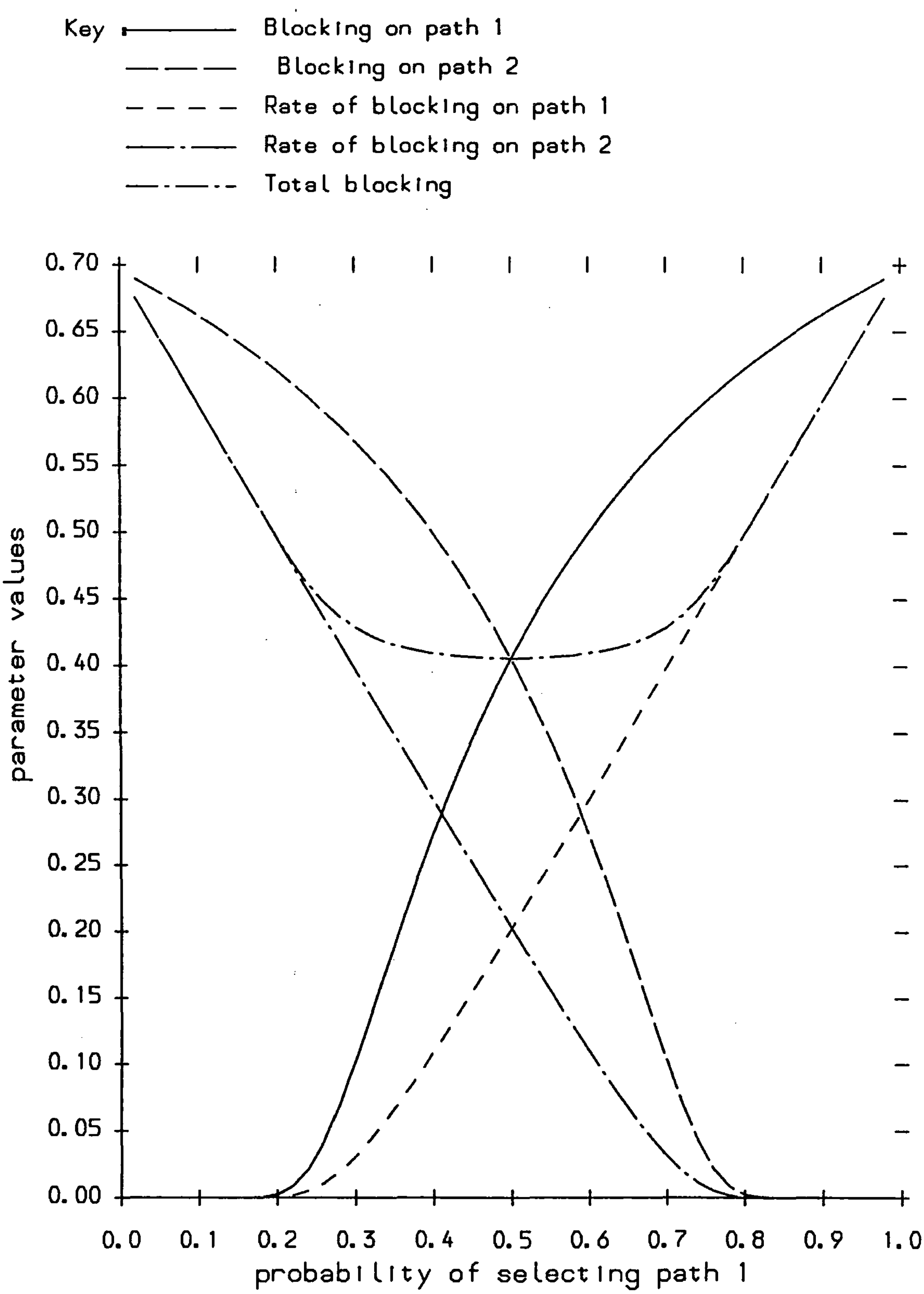


figure 4.2(a)

performance criteria in an overloaded symmetrical network

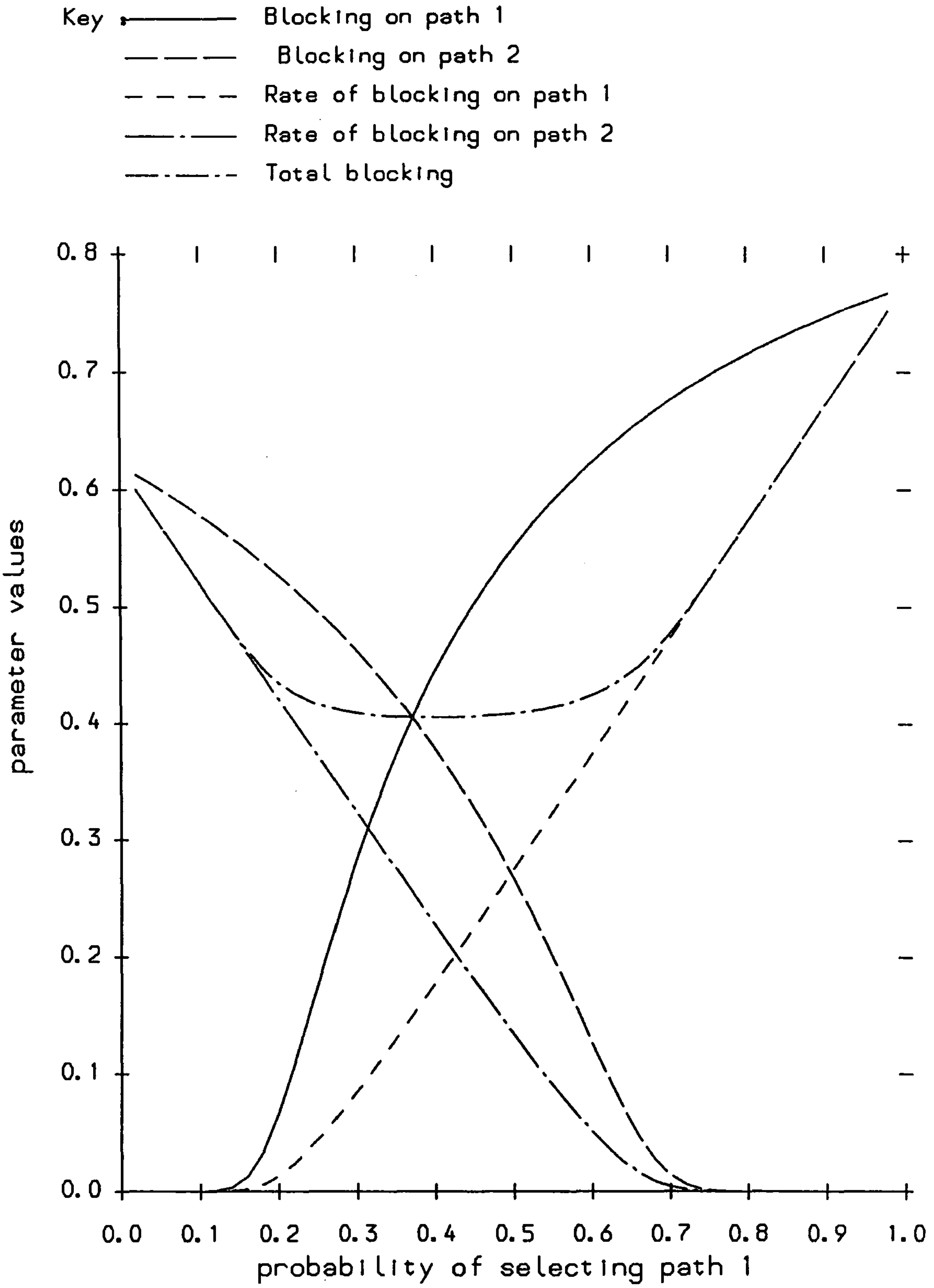


figure 4.2(b)

performance criteria in an overloaded unsymmetrical network

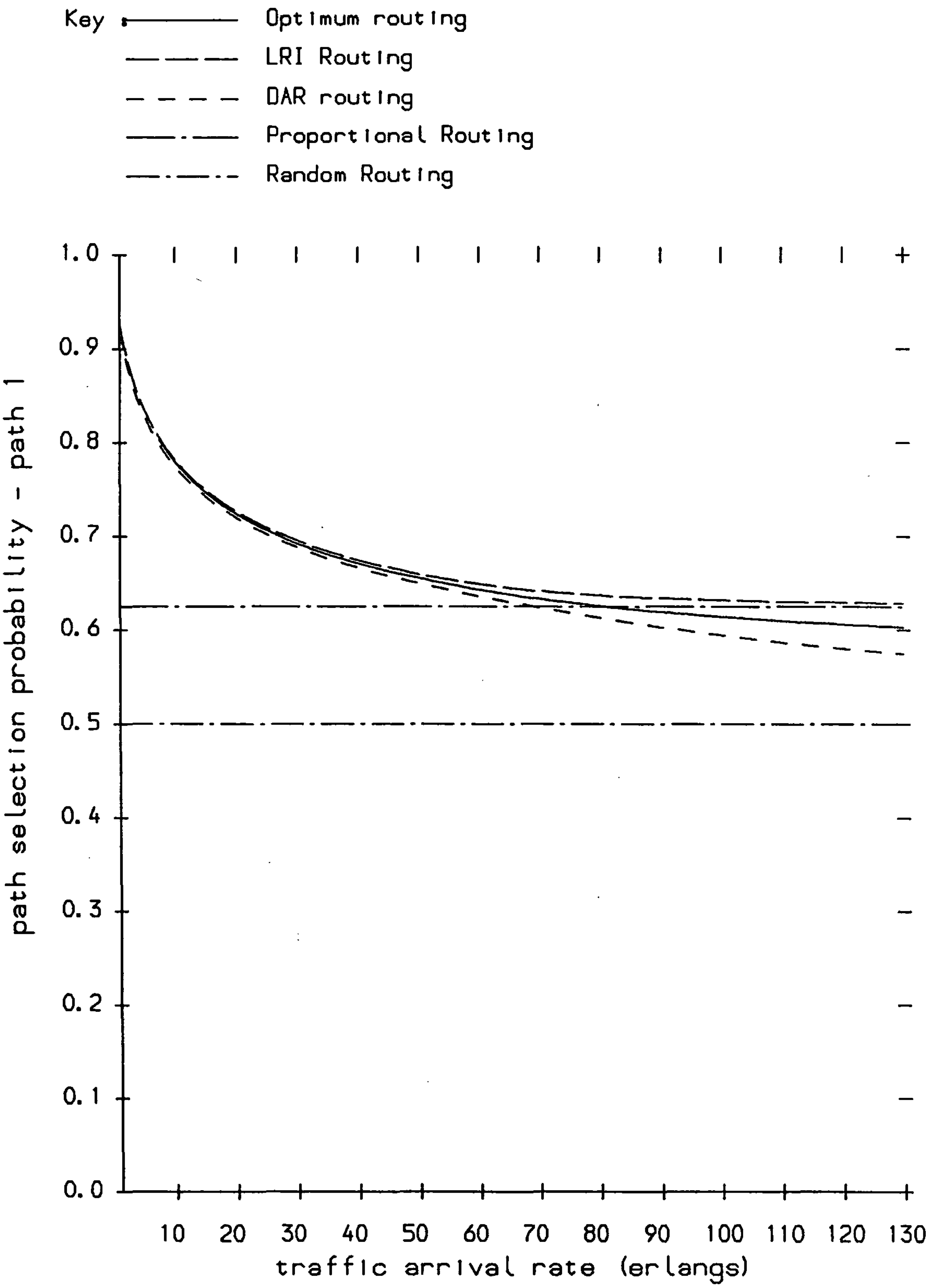


figure 4.3

path selection probability as function of traffic

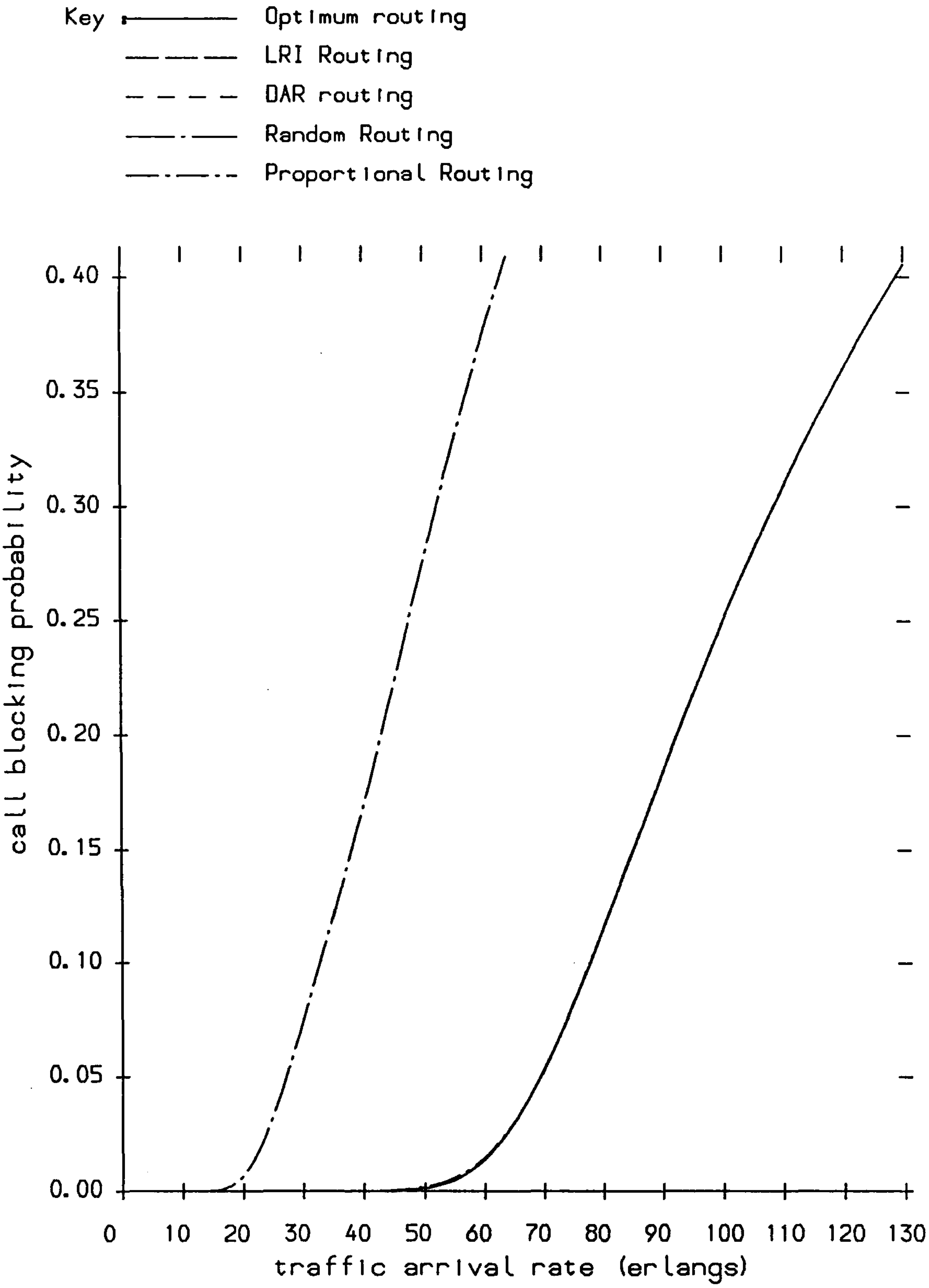


figure 4.4

single traffic source performance

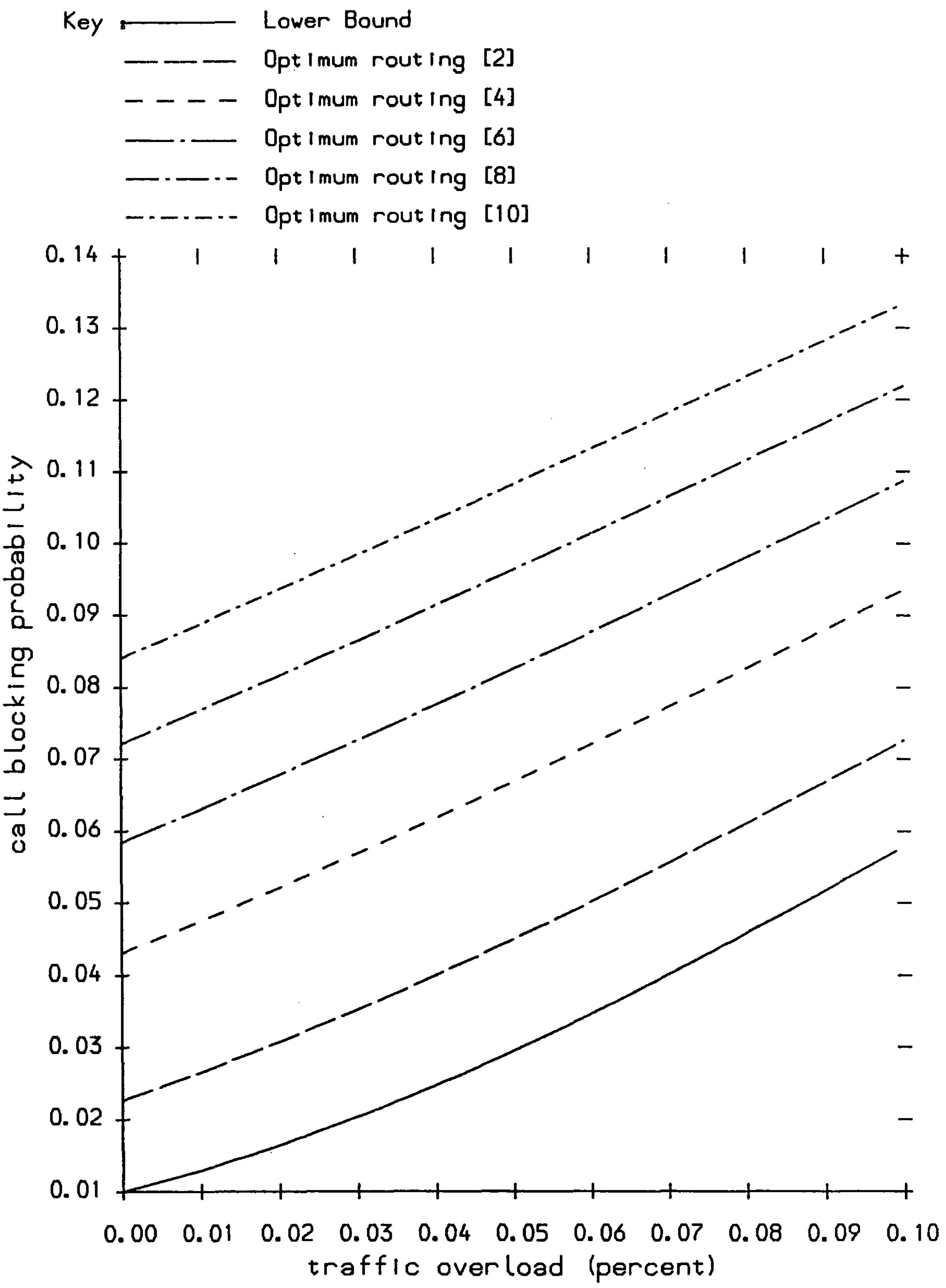


figure 4.5

optimum traffic distribution over multiple paths

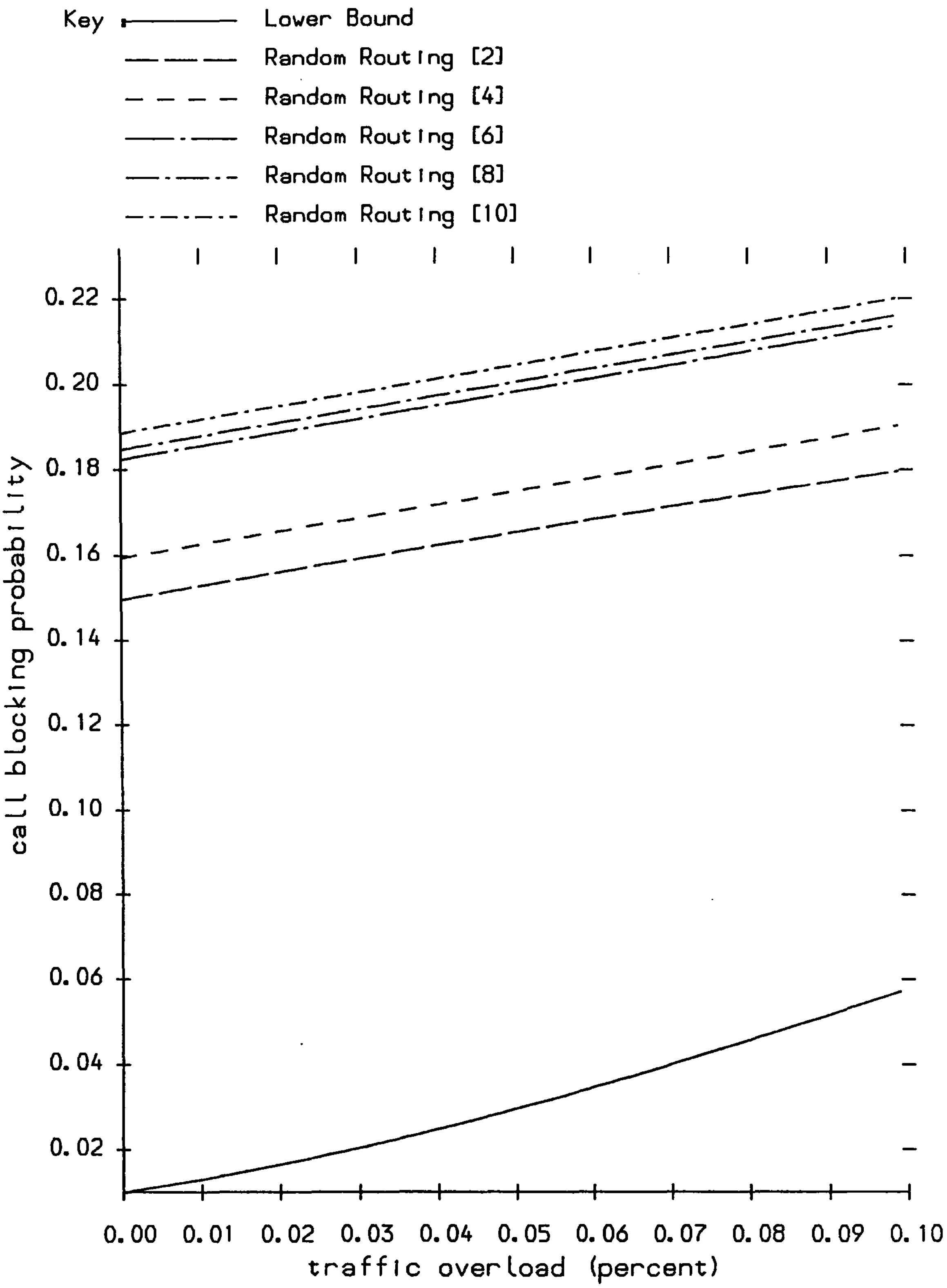


figure 4.6

randon traffic distribution over multiple paths

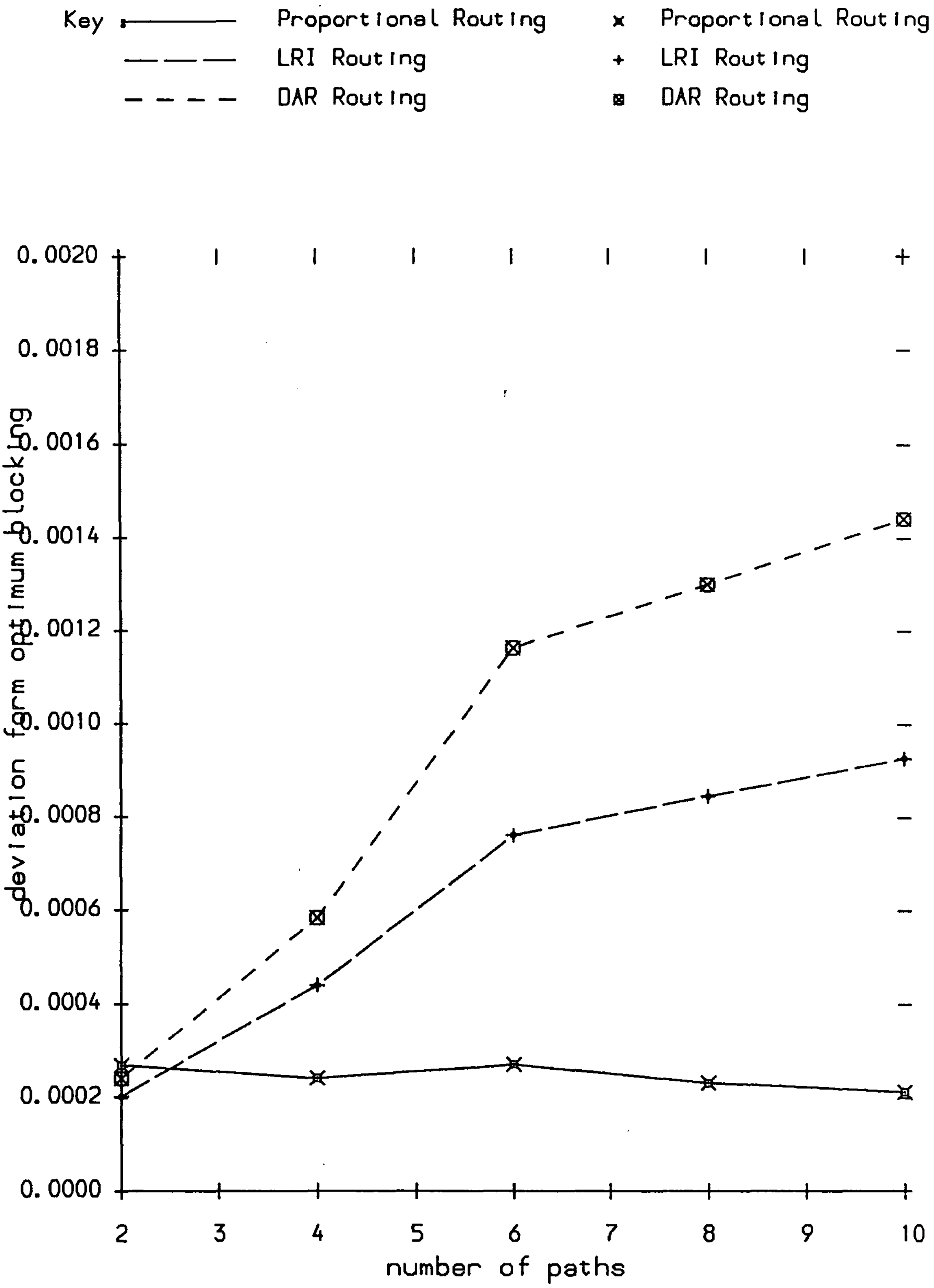


figure 4.7

sub-optimal performance of adaptive strategies

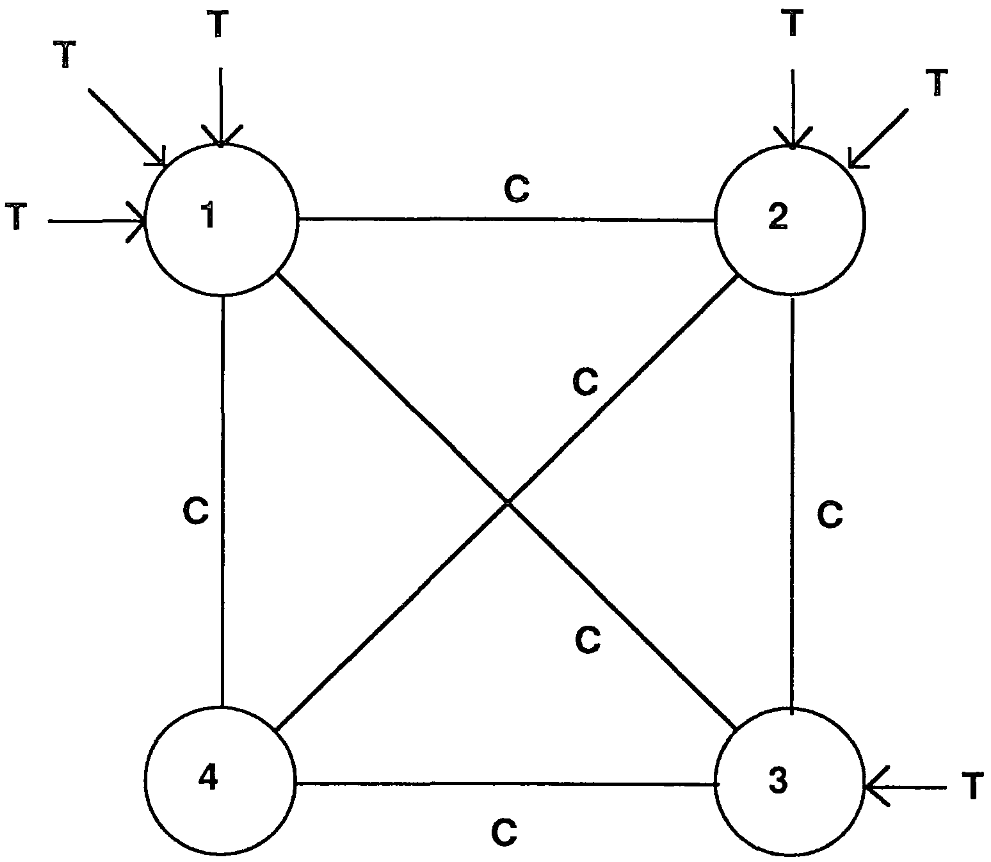


figure 4.8(a)

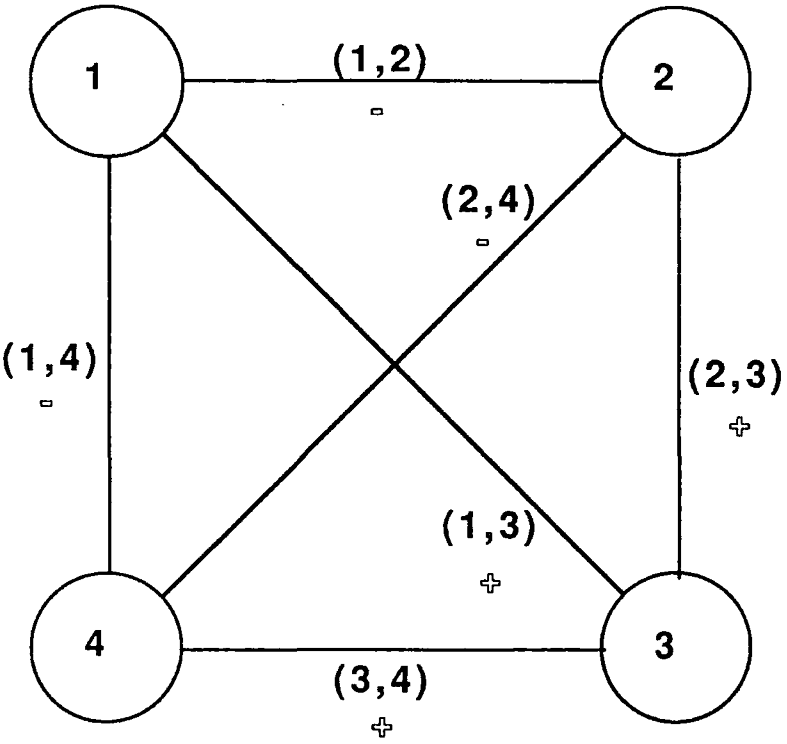


figure 4.8(b)

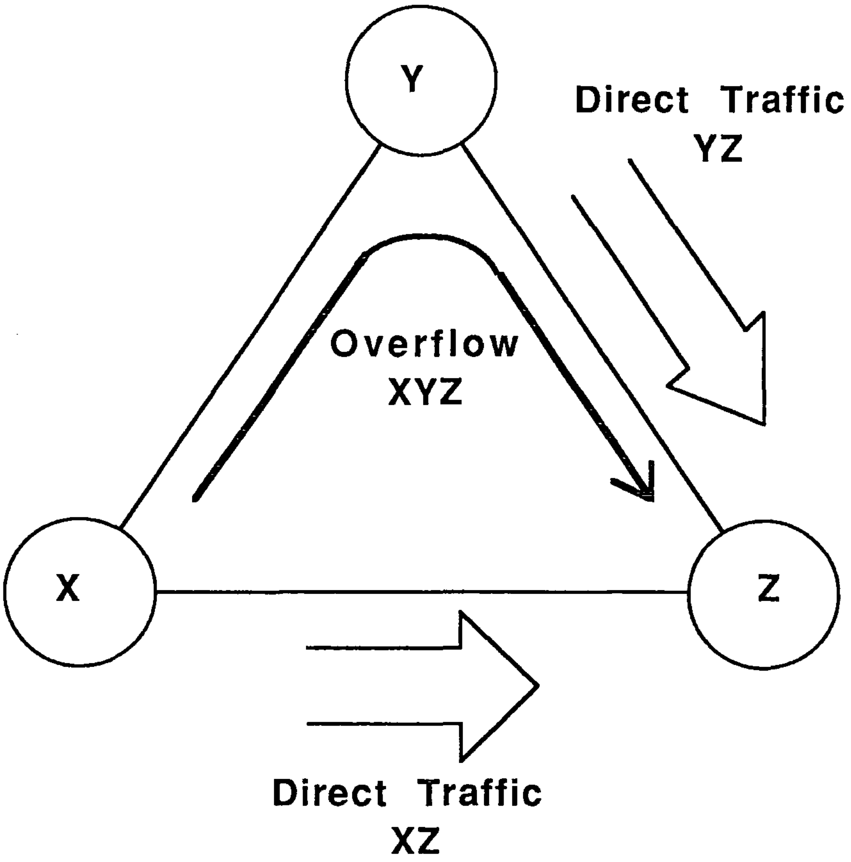


Figure 4.9

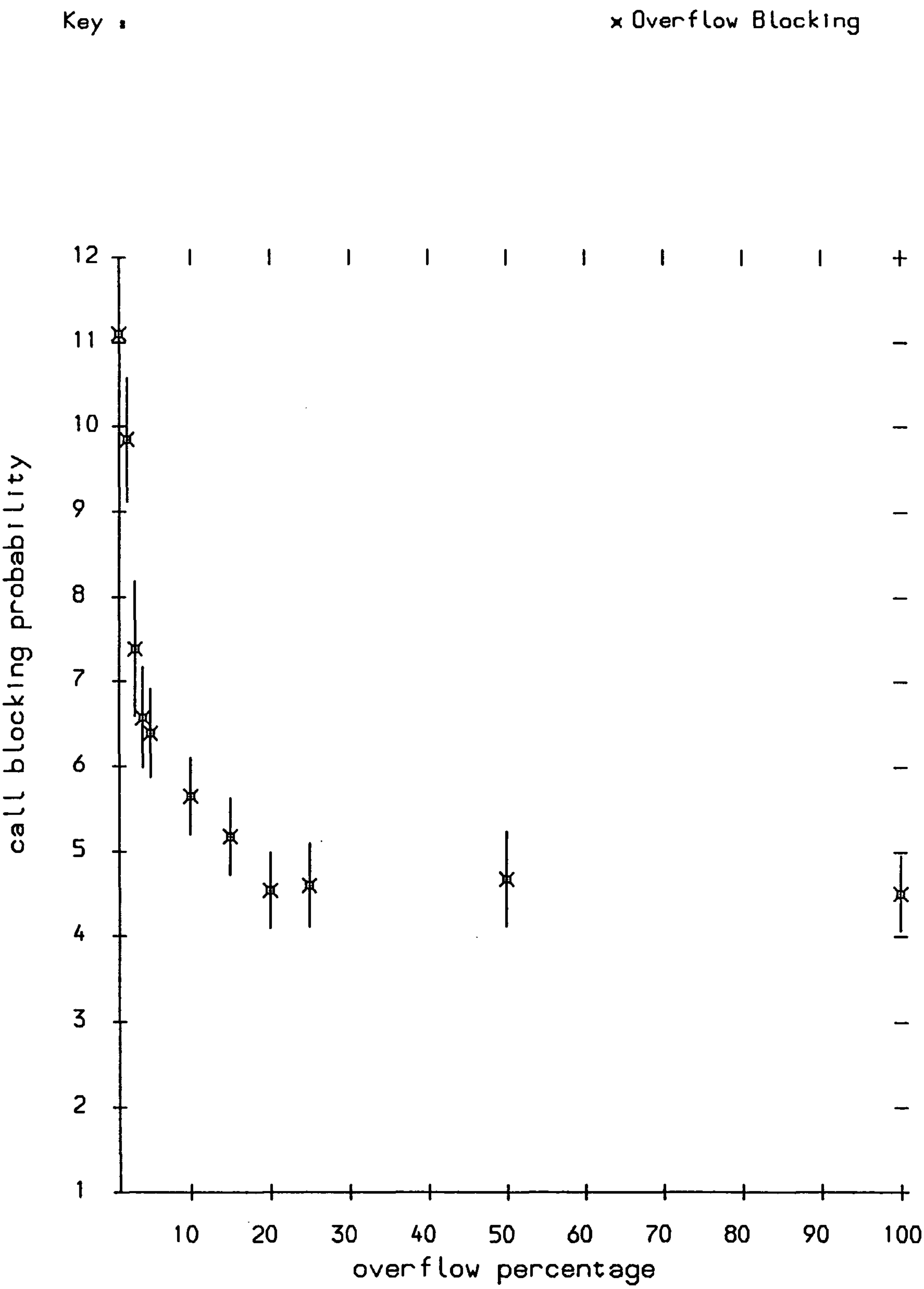


figure 4.10

blocking probability of overflow traffic

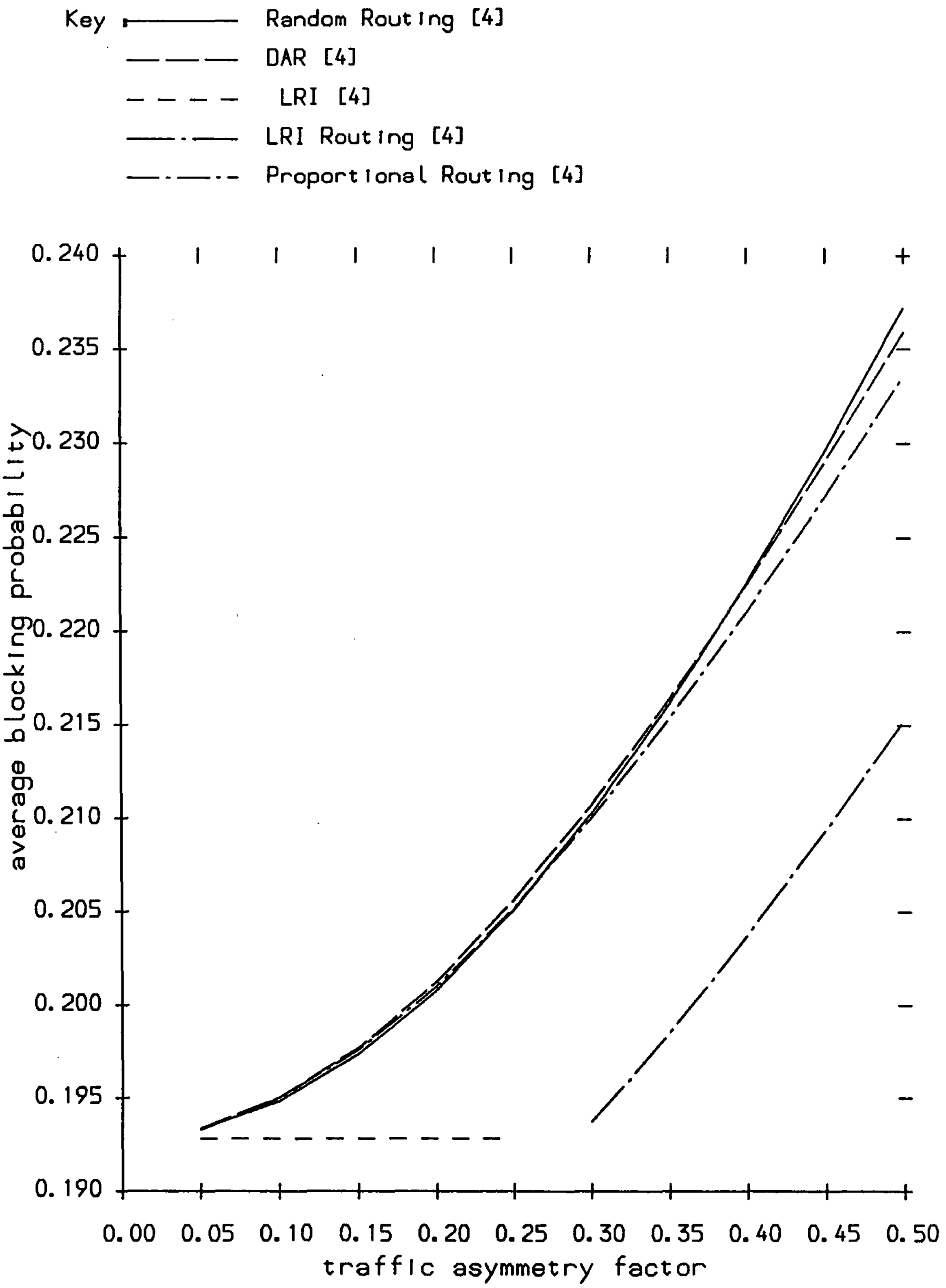


figure 4.11(a)

analytic results for traffic asymmetry

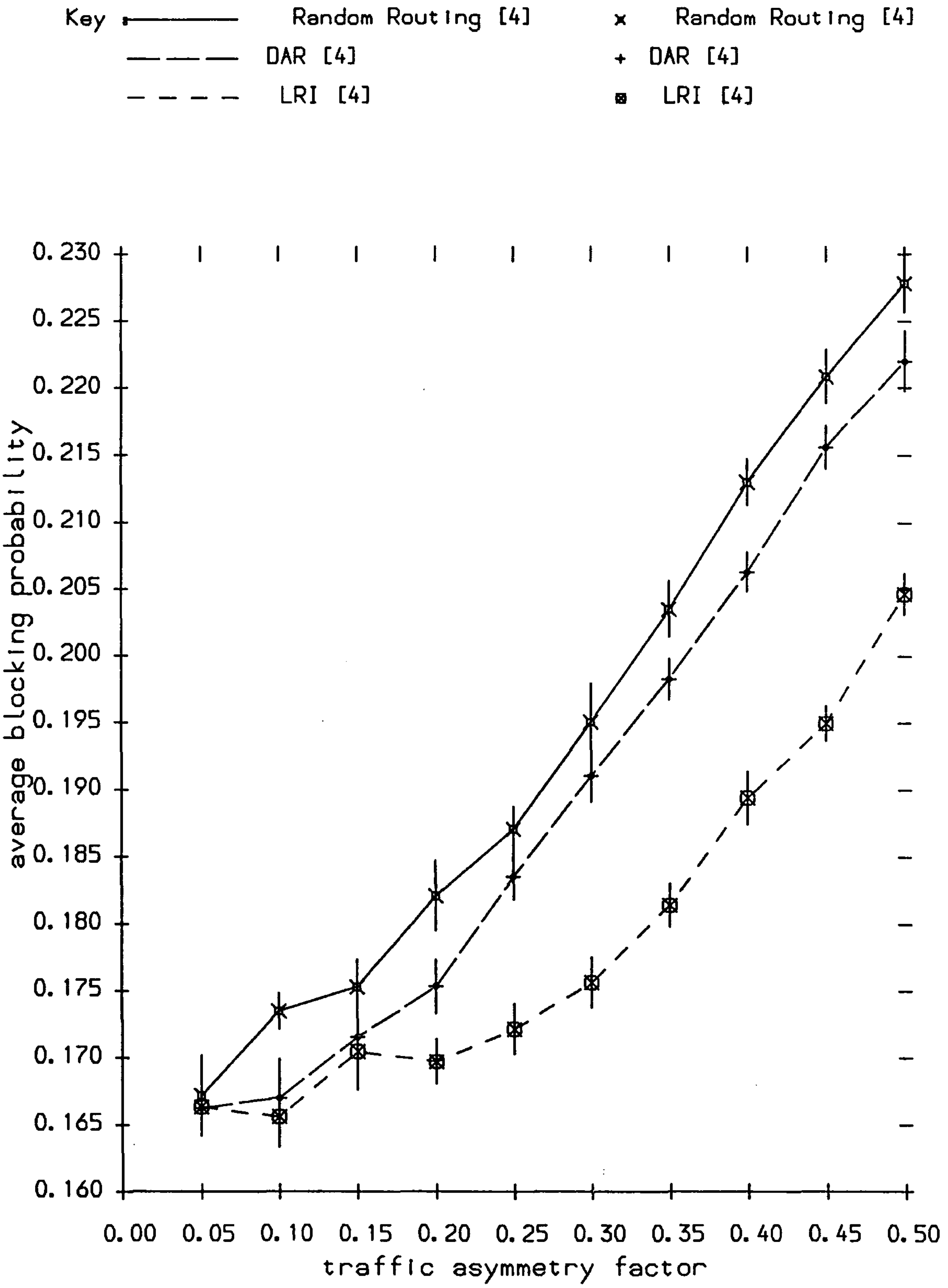


figure 4.11 (b)

simulated results for traffic asymmetry

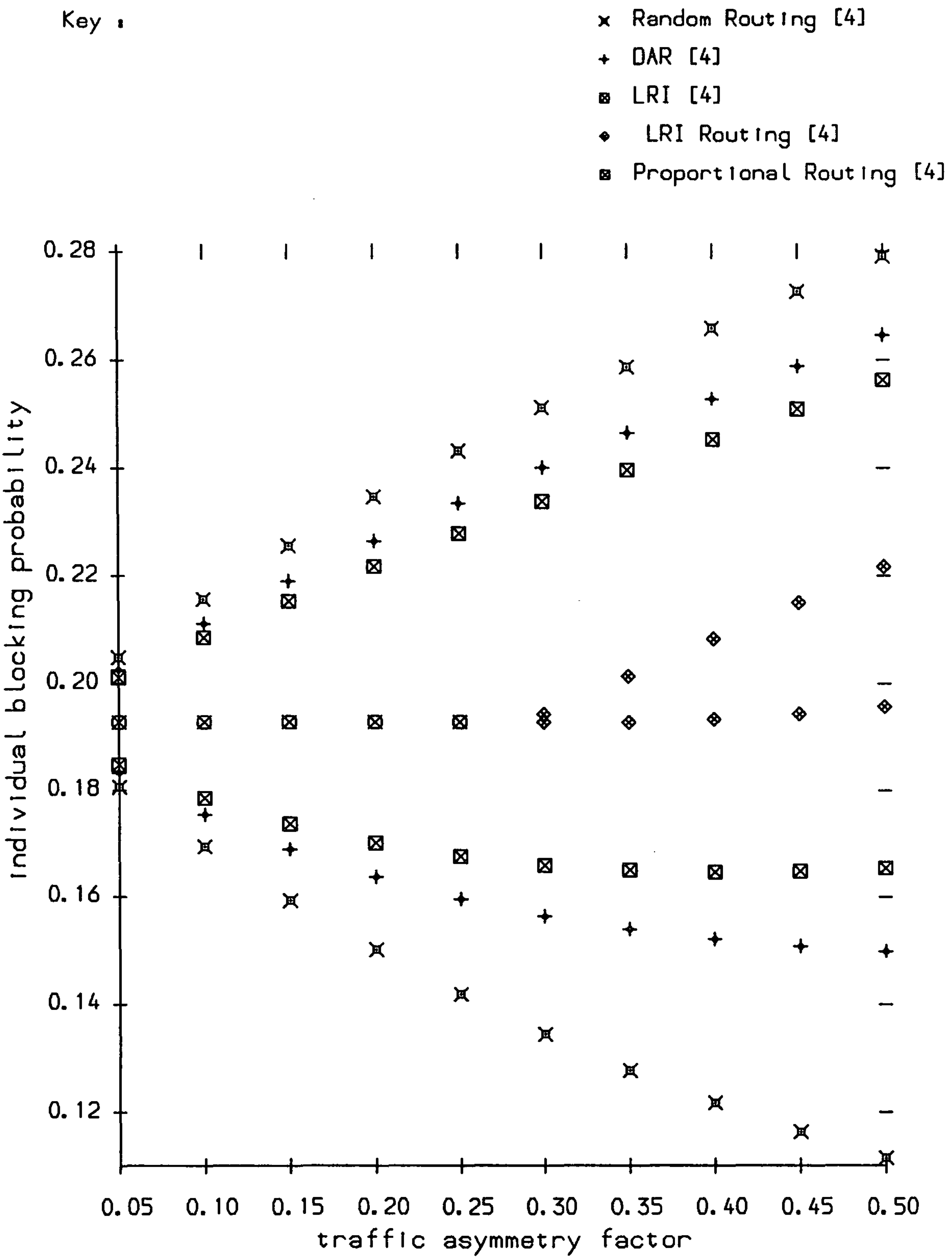


figure 4.12(a)

analytic GOS results for traffic asymmetry

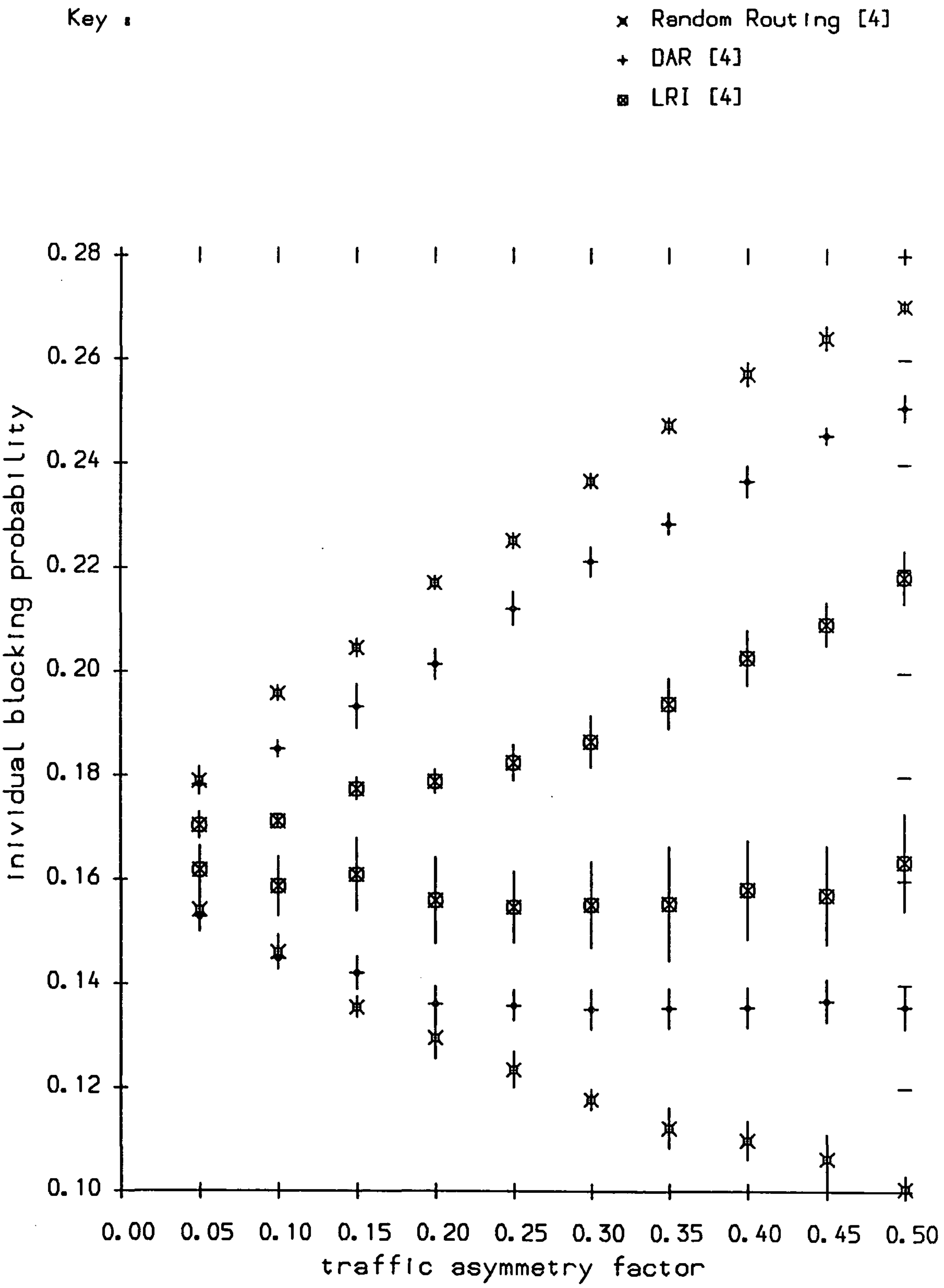


figure 4.12(b)

simulated GOS results for traffic asymmetry

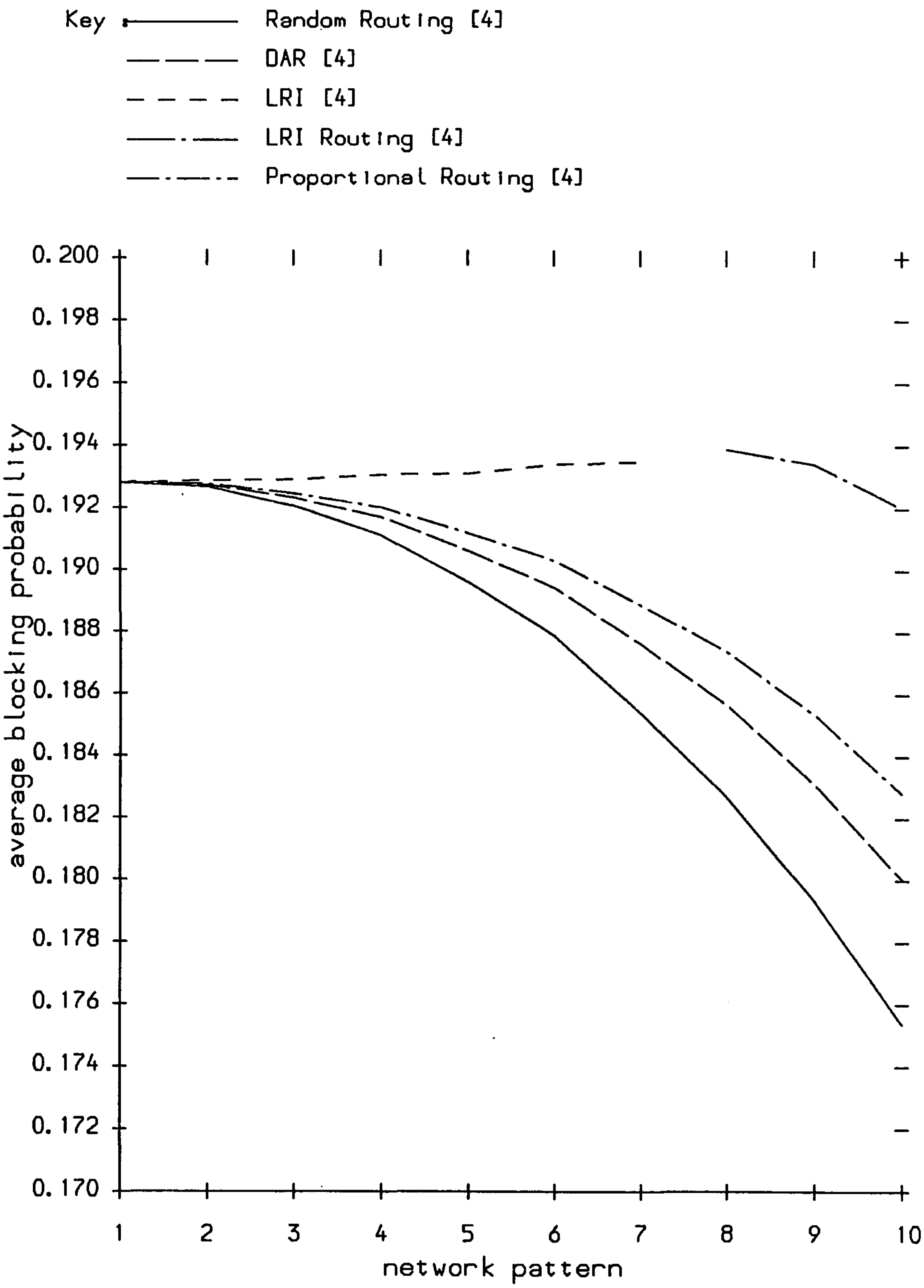


figure 4.13(a)

analytic results for trunk asymmetry

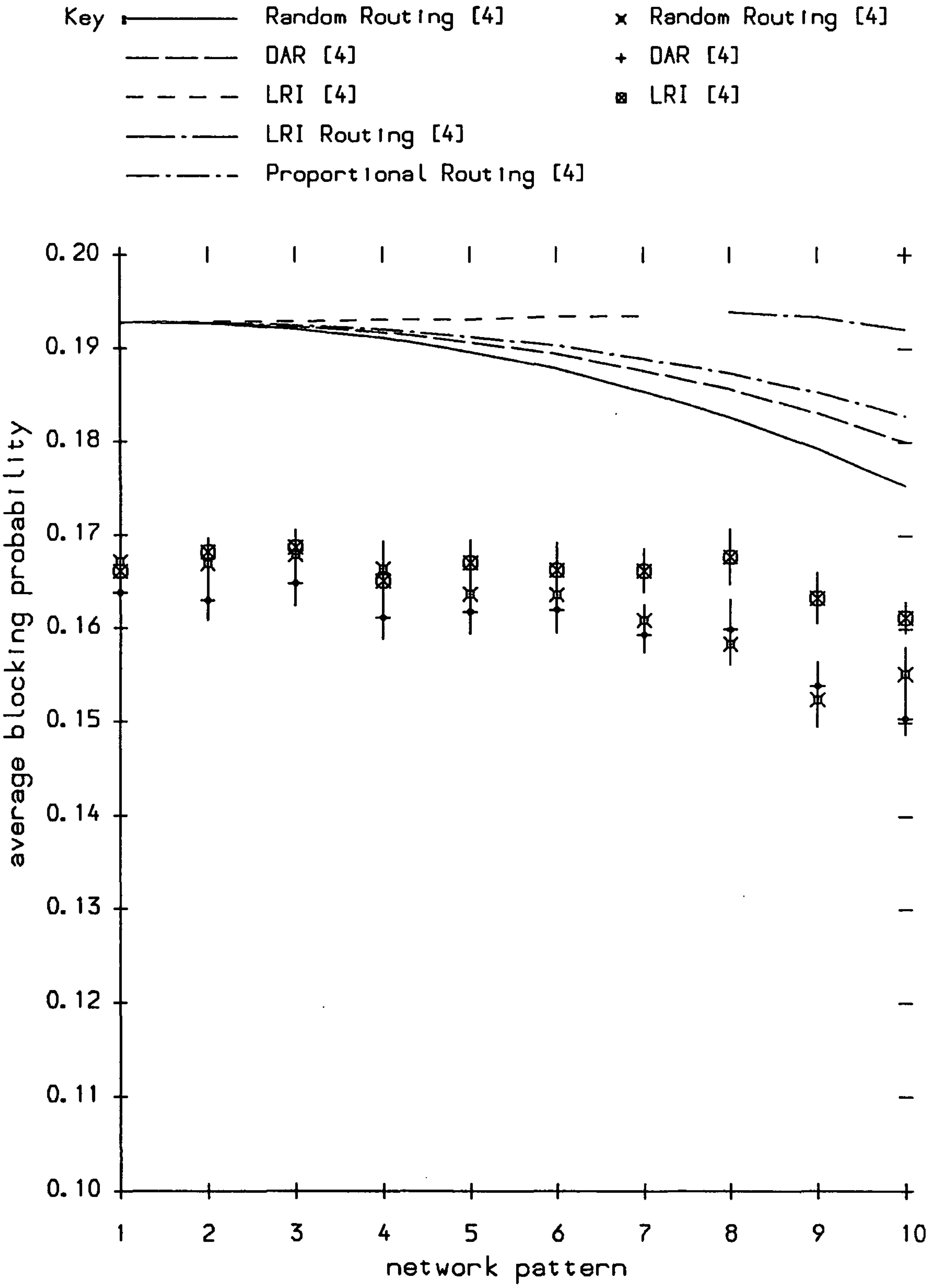


figure 4.13(b)

simulated results for trunk asymmetry

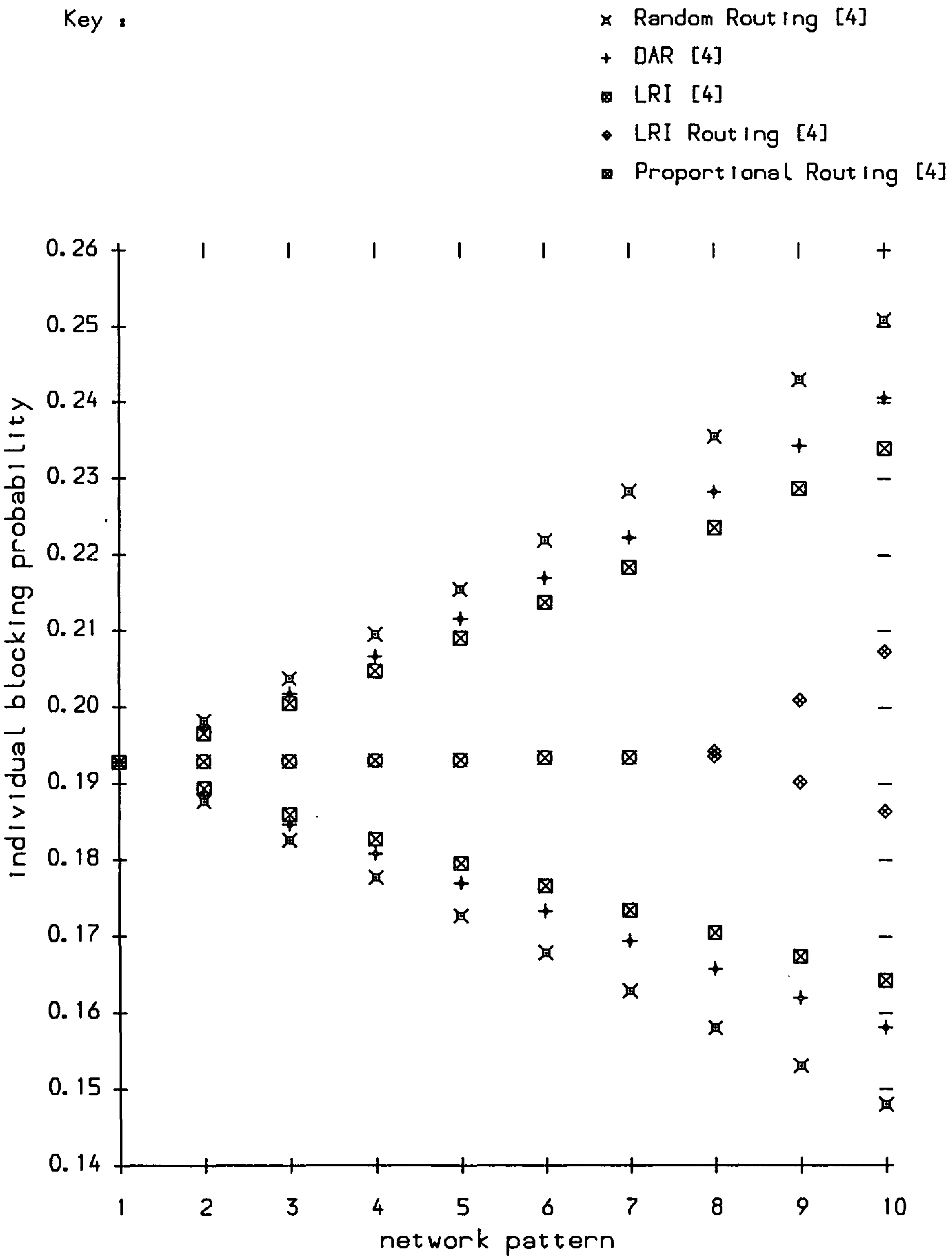


figure 4.14(a)

analytic GOS results for trunk asymmetry

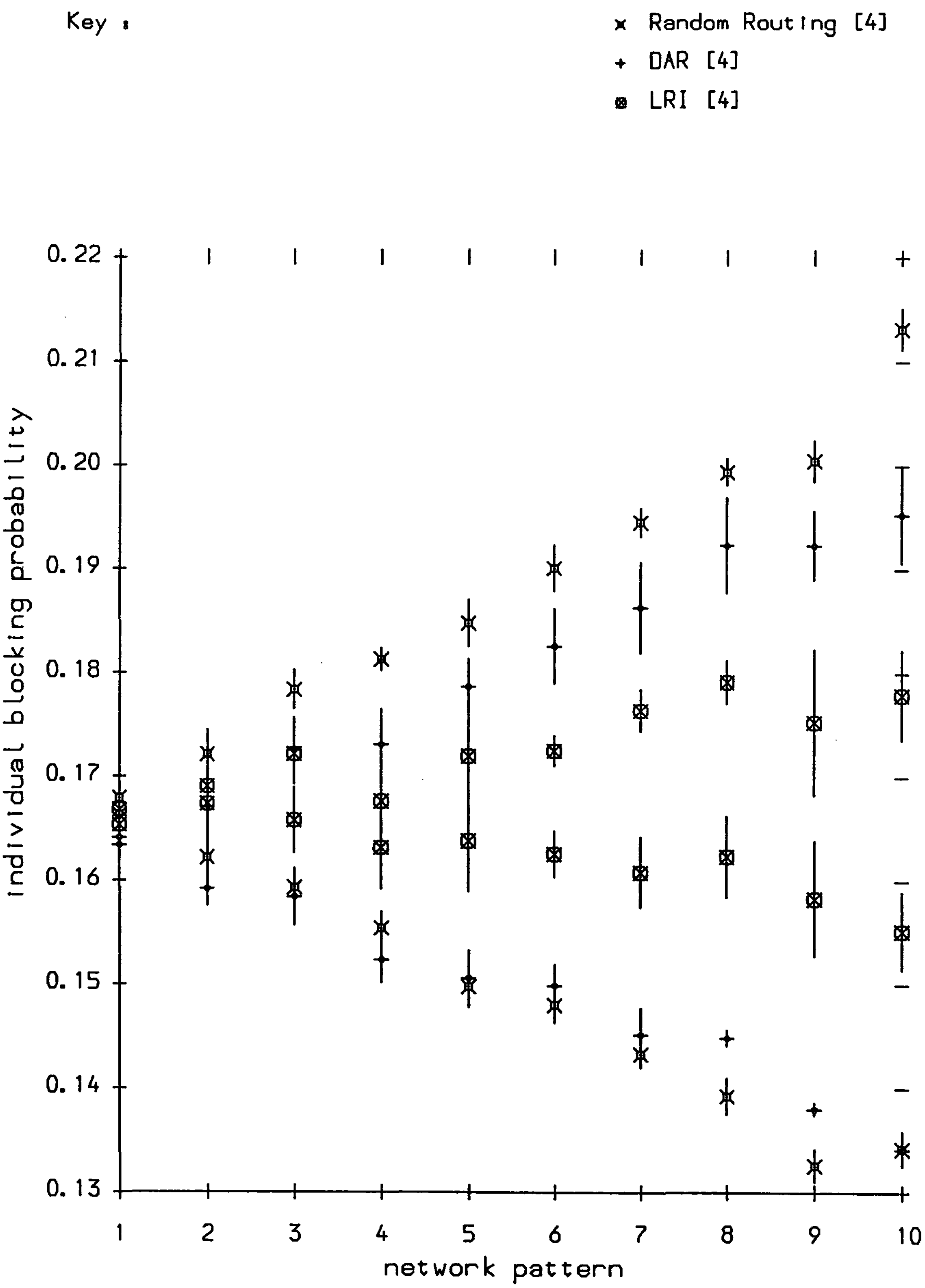


figure 4.14 (b)

simulated GOS results for trunk asymmetry

Key :

× Analytic Model

+ Simulated Network [4]

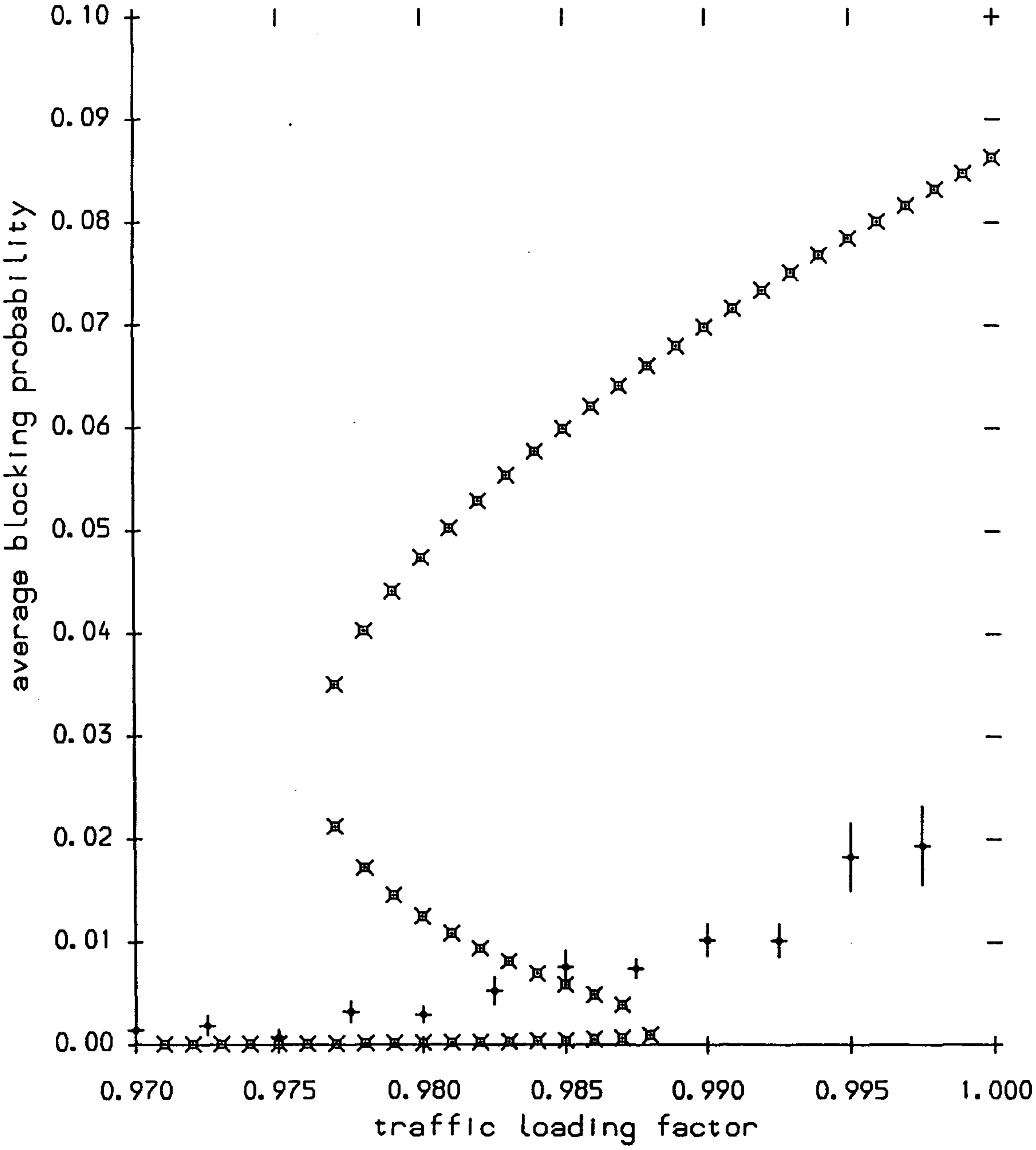


figure 4.15

instability in a symmetrical network

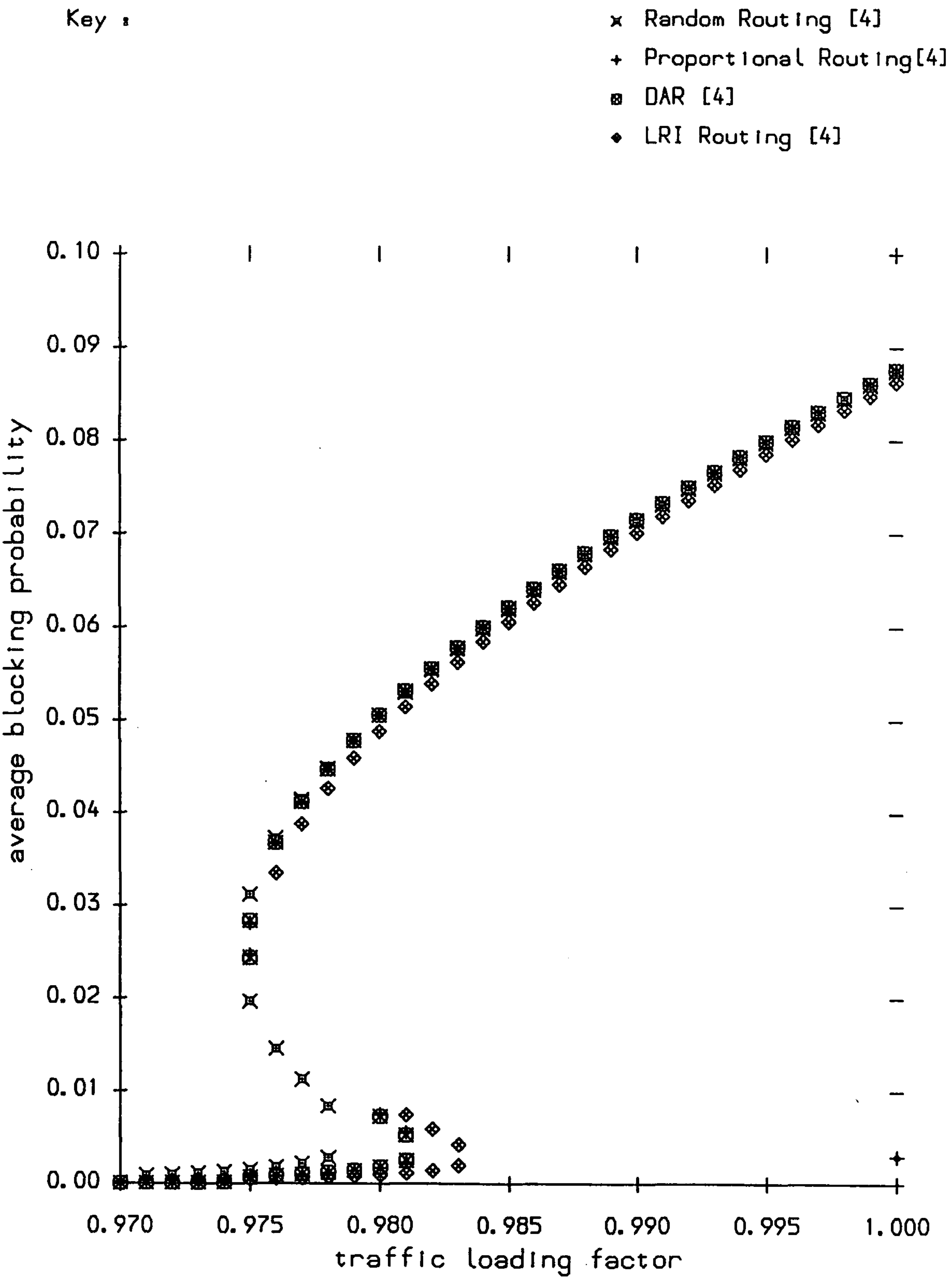


figure 4.16(a)

asymmetrical traffic loading = 0.05

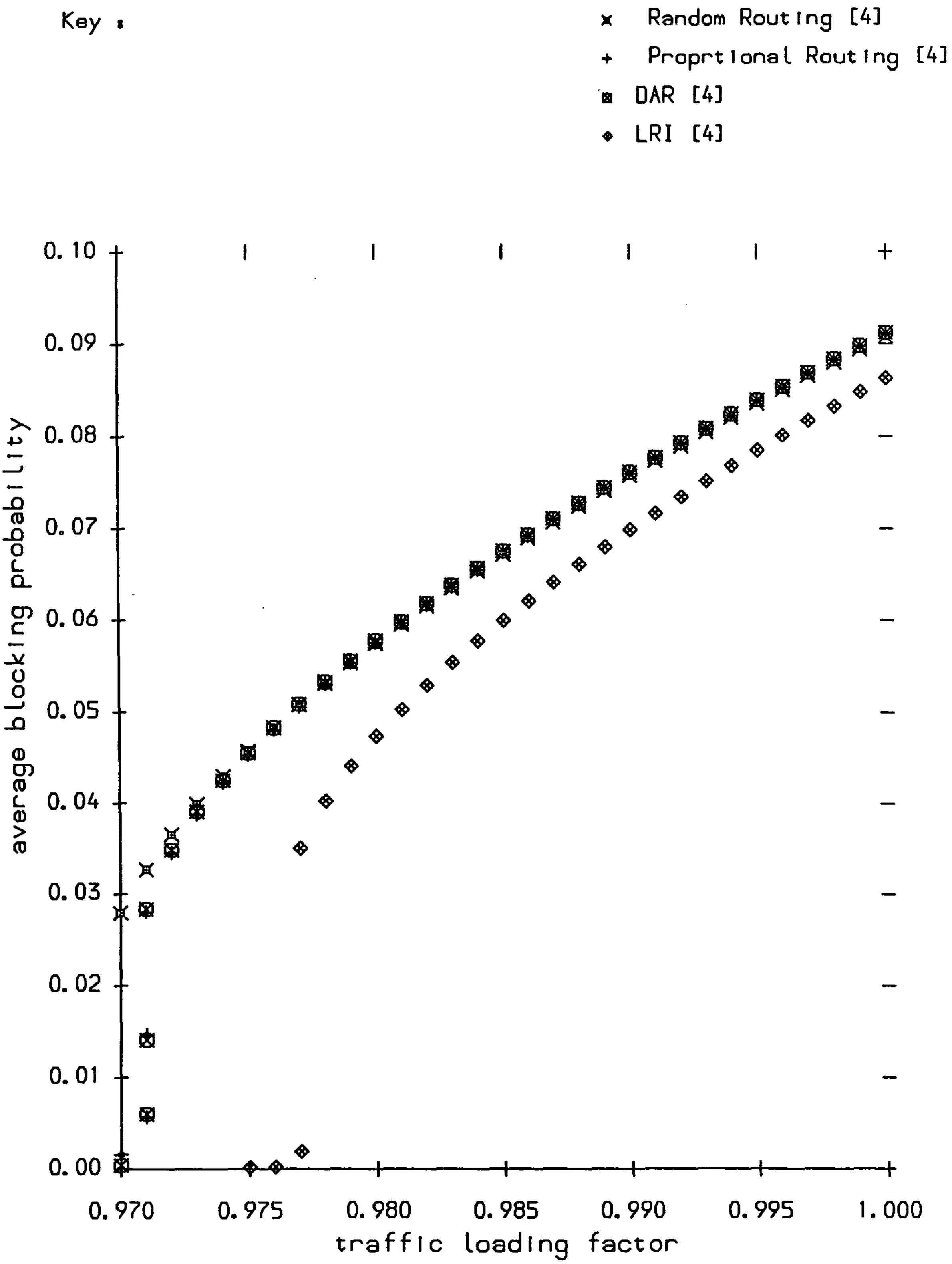


figure 4.16(b)

asymmetrical traffic loading = 0.1

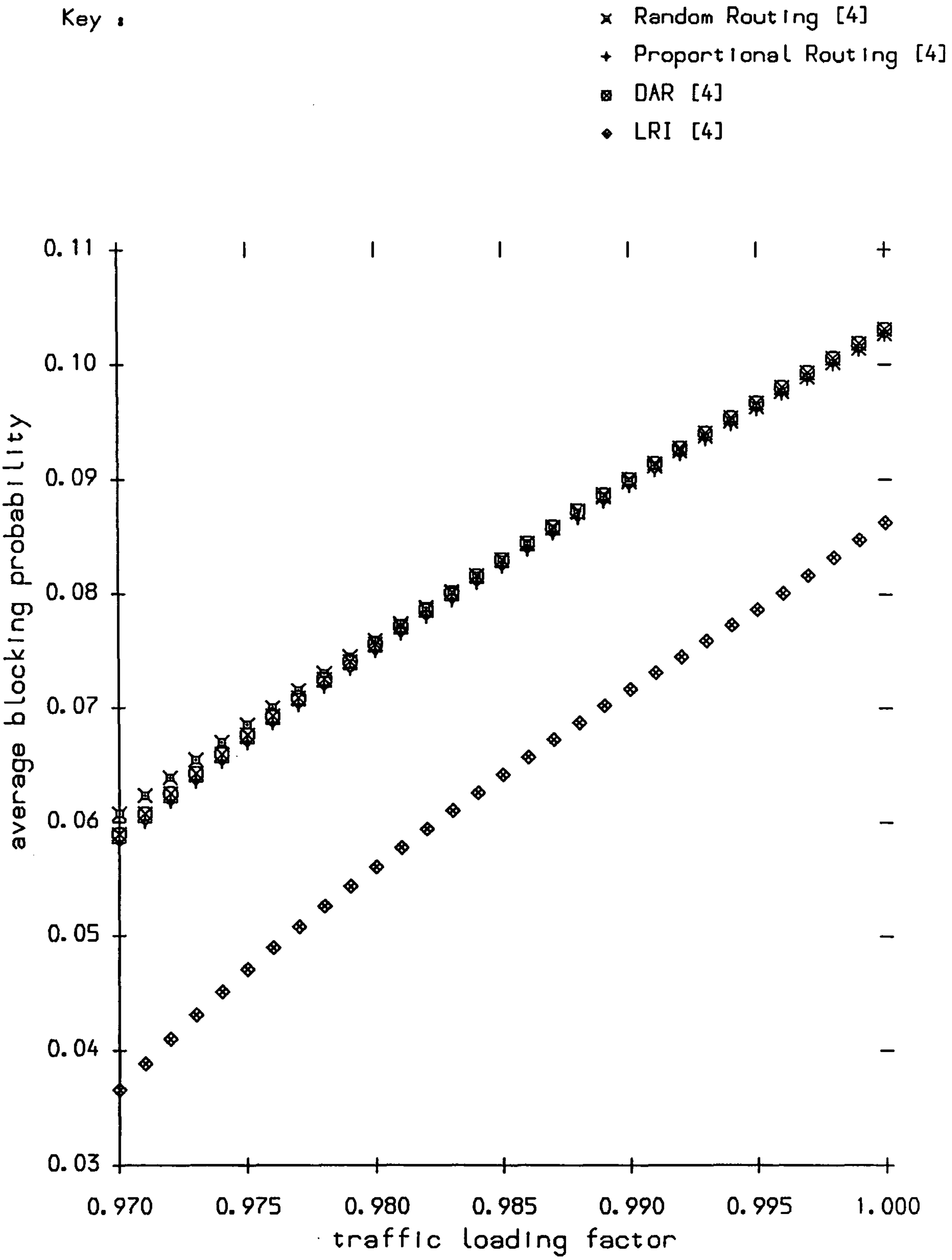


figure 4.16(c)

asymmetrical traffic paramter = 0.2

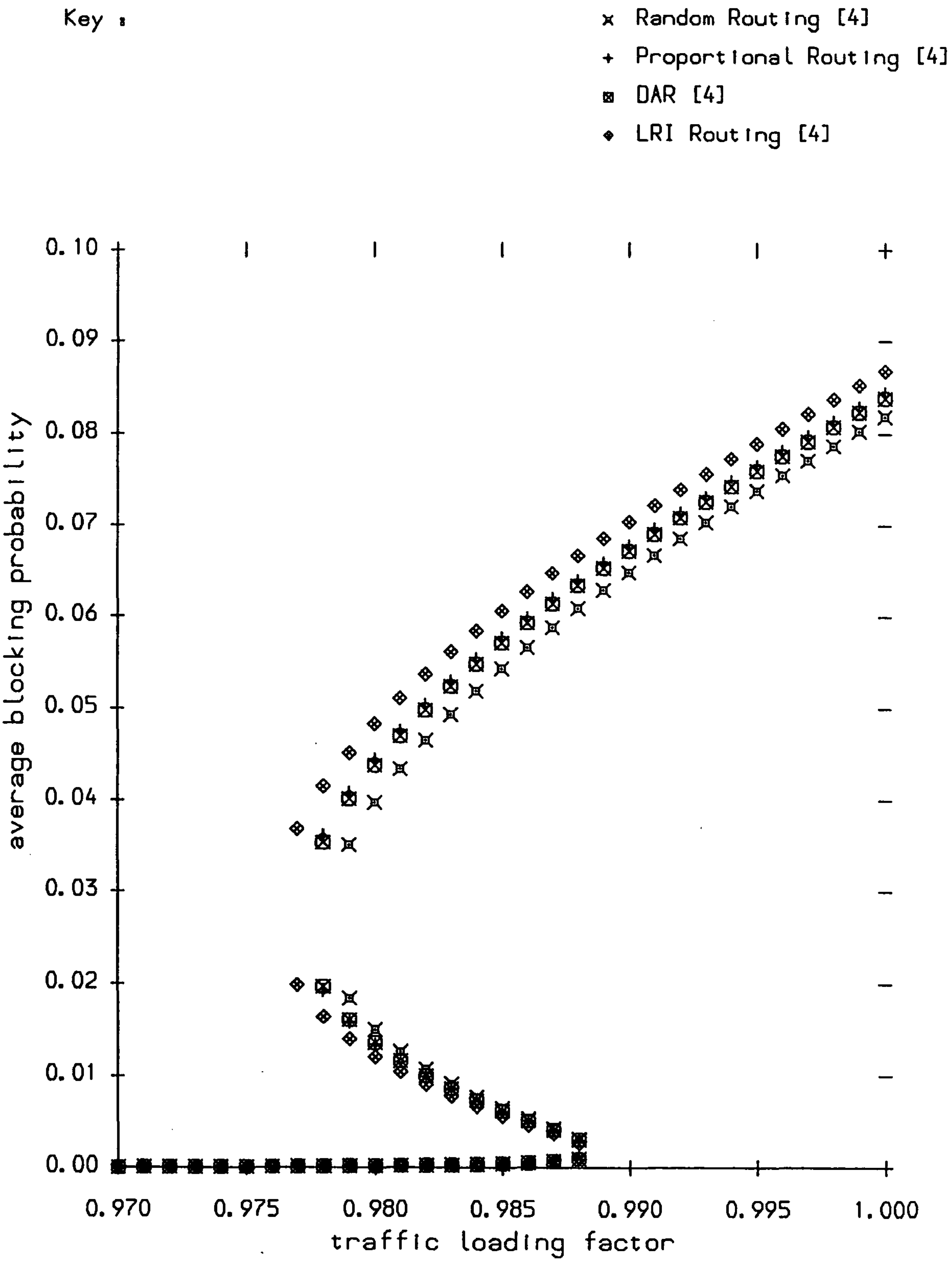


figure 4.17(a)

asymmetrical trunk paramter = 0.2

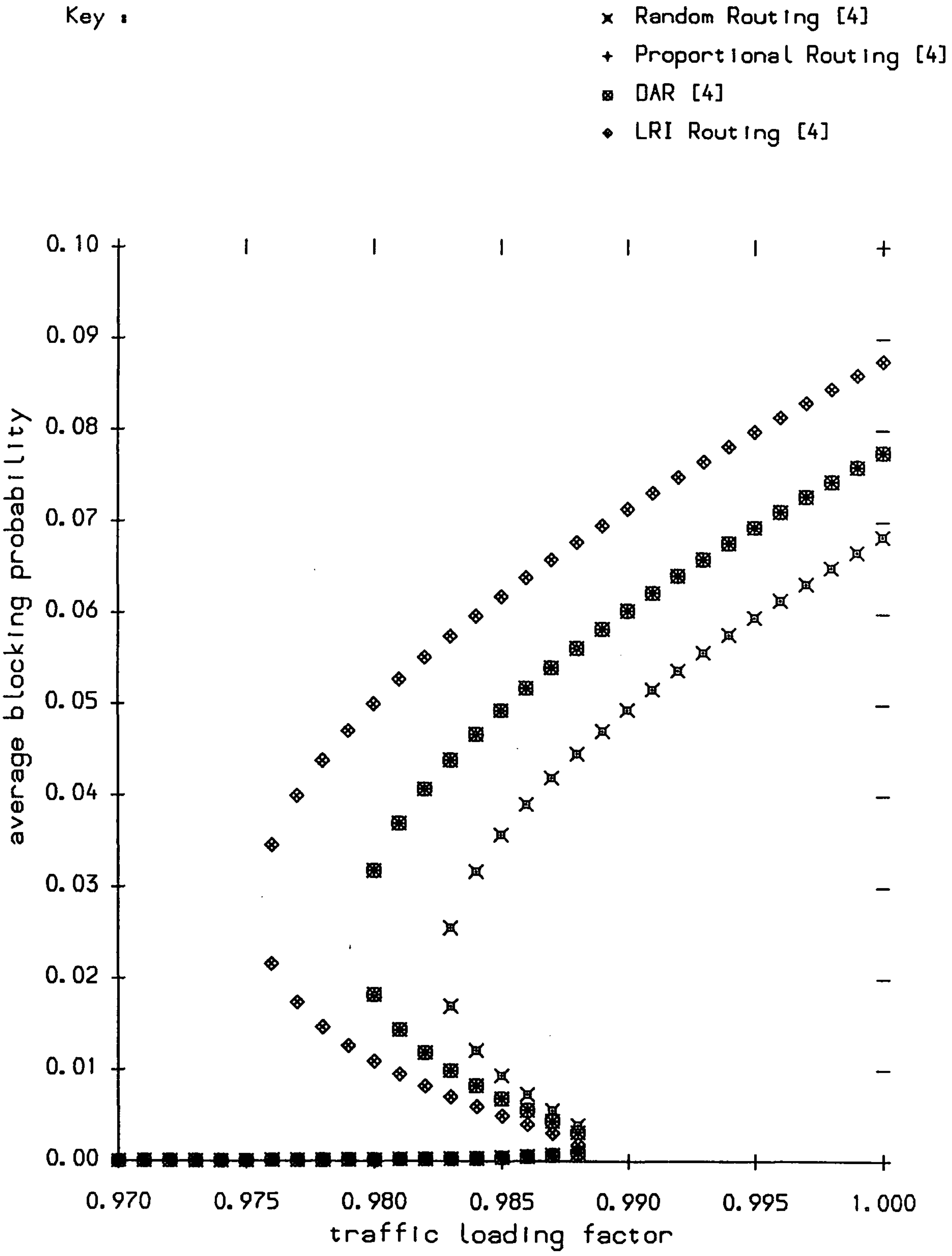


figure 4.17(b)

asymmetrical trunk parameter

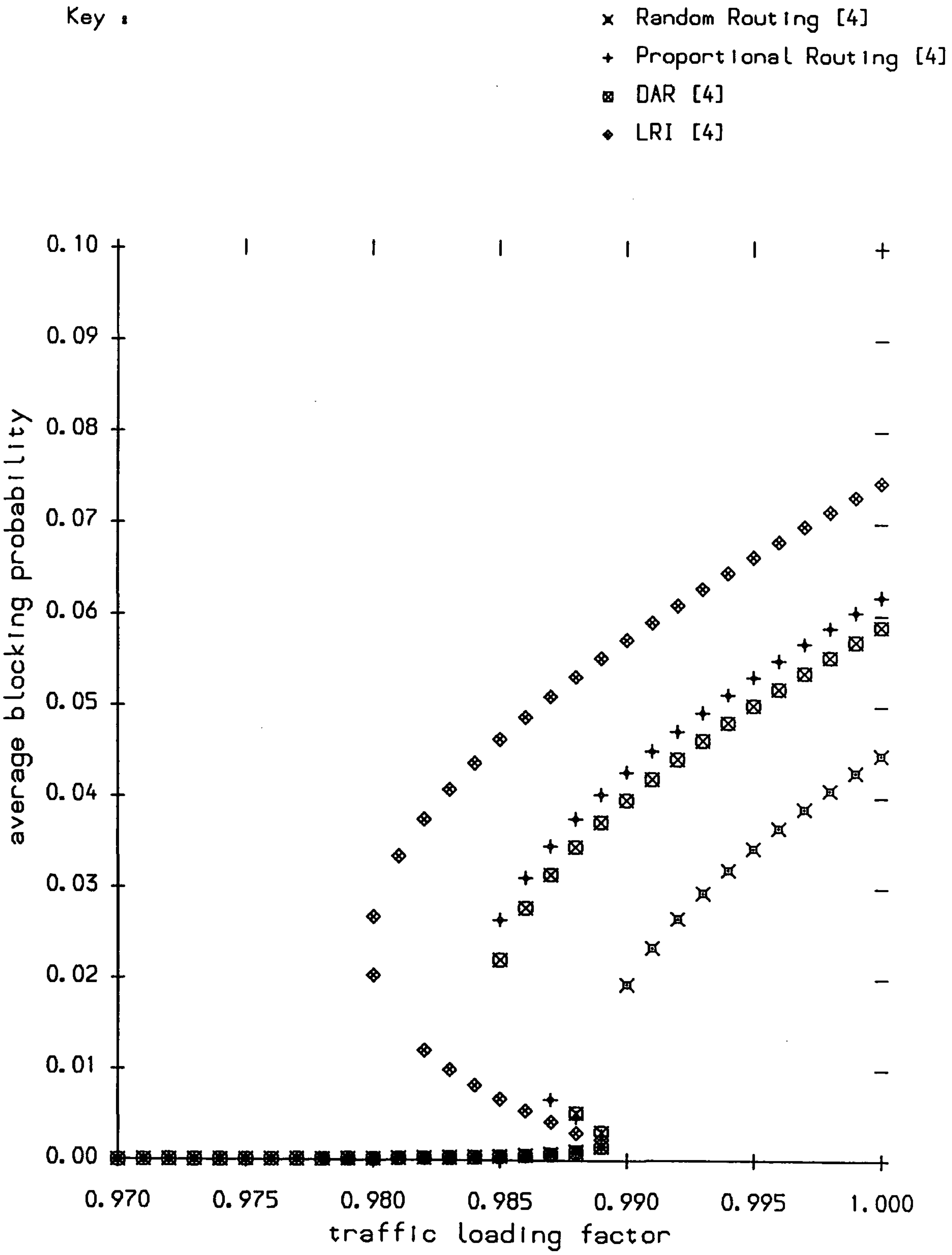


figure 4.17(c)

asymmetrical trunk parameter = 0.6

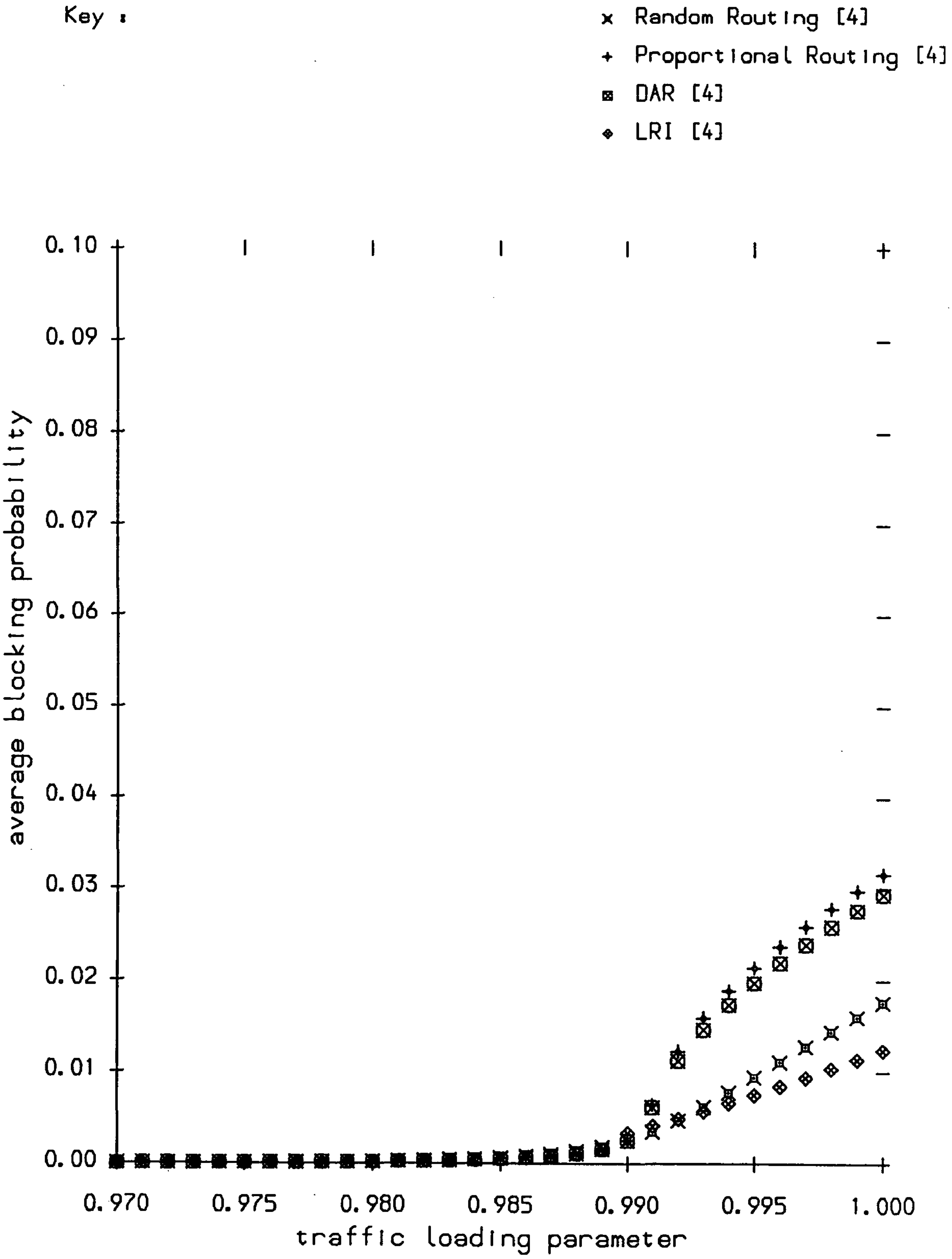


figure 4.17(d)

asymmetrical trunk parameter = 0.8

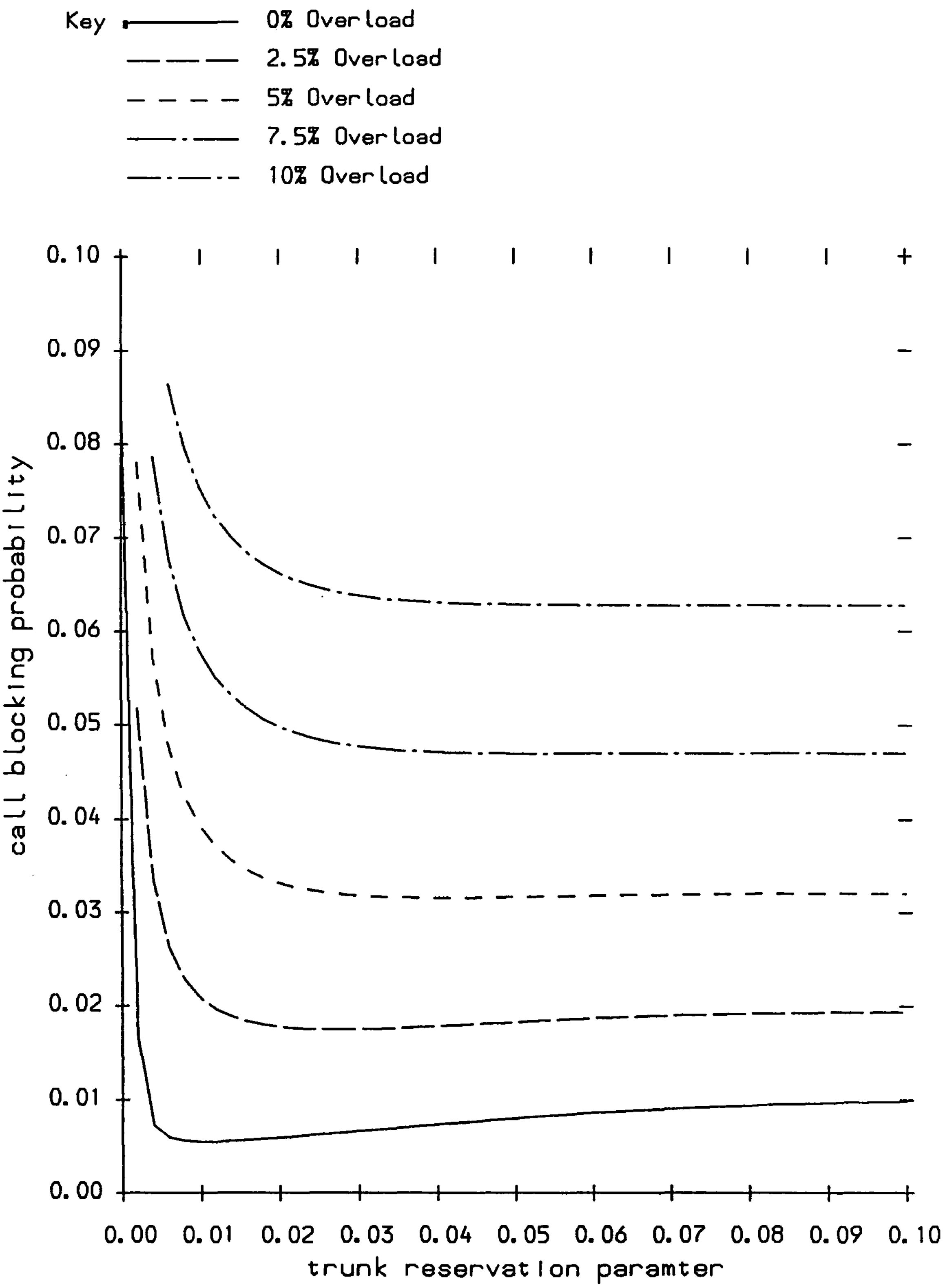


fig4.18(a)

trunk reservation applied to symmetrical network

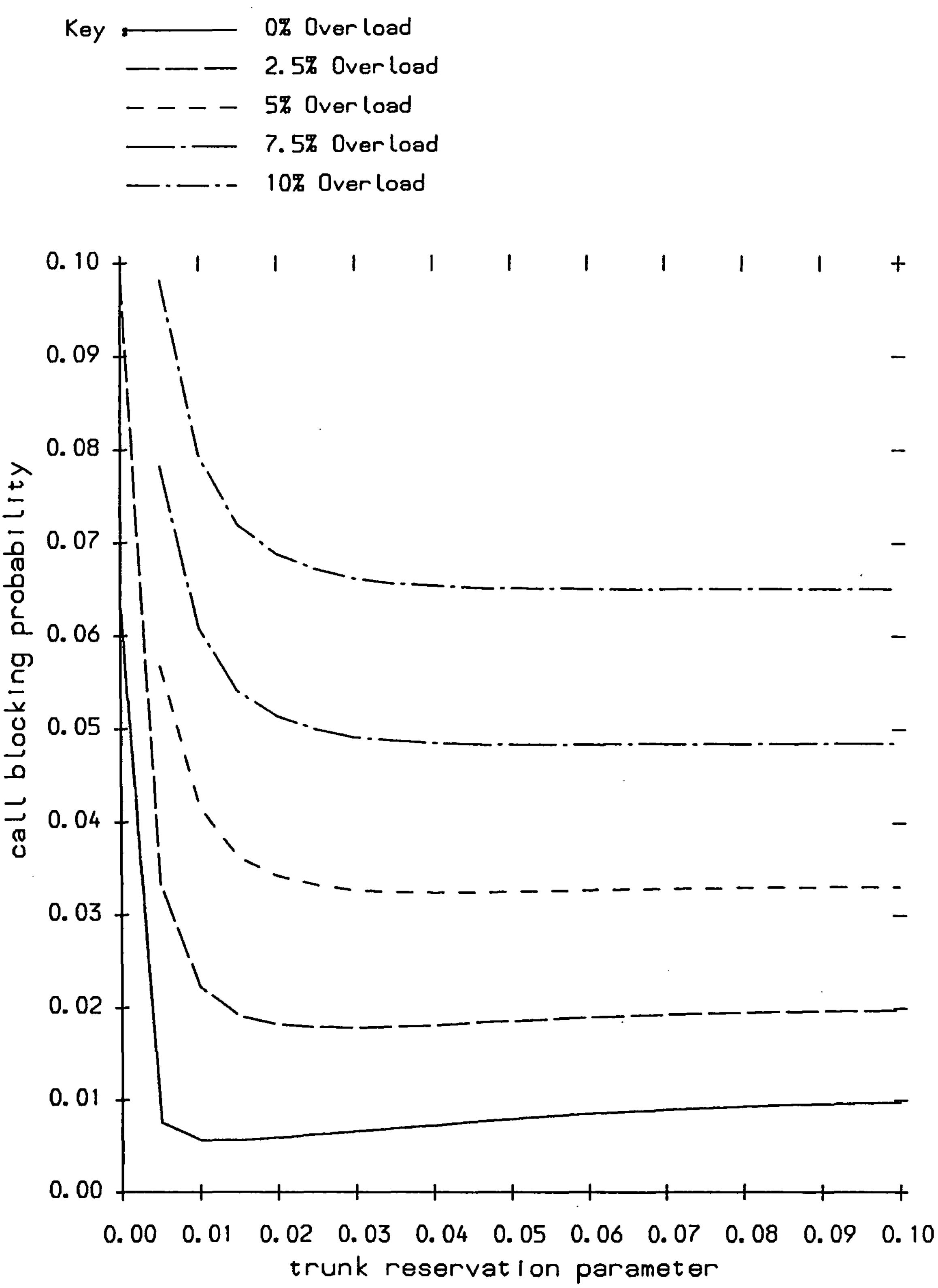


figure 4.18(b)

trunk reservation applied to asymmetrical network

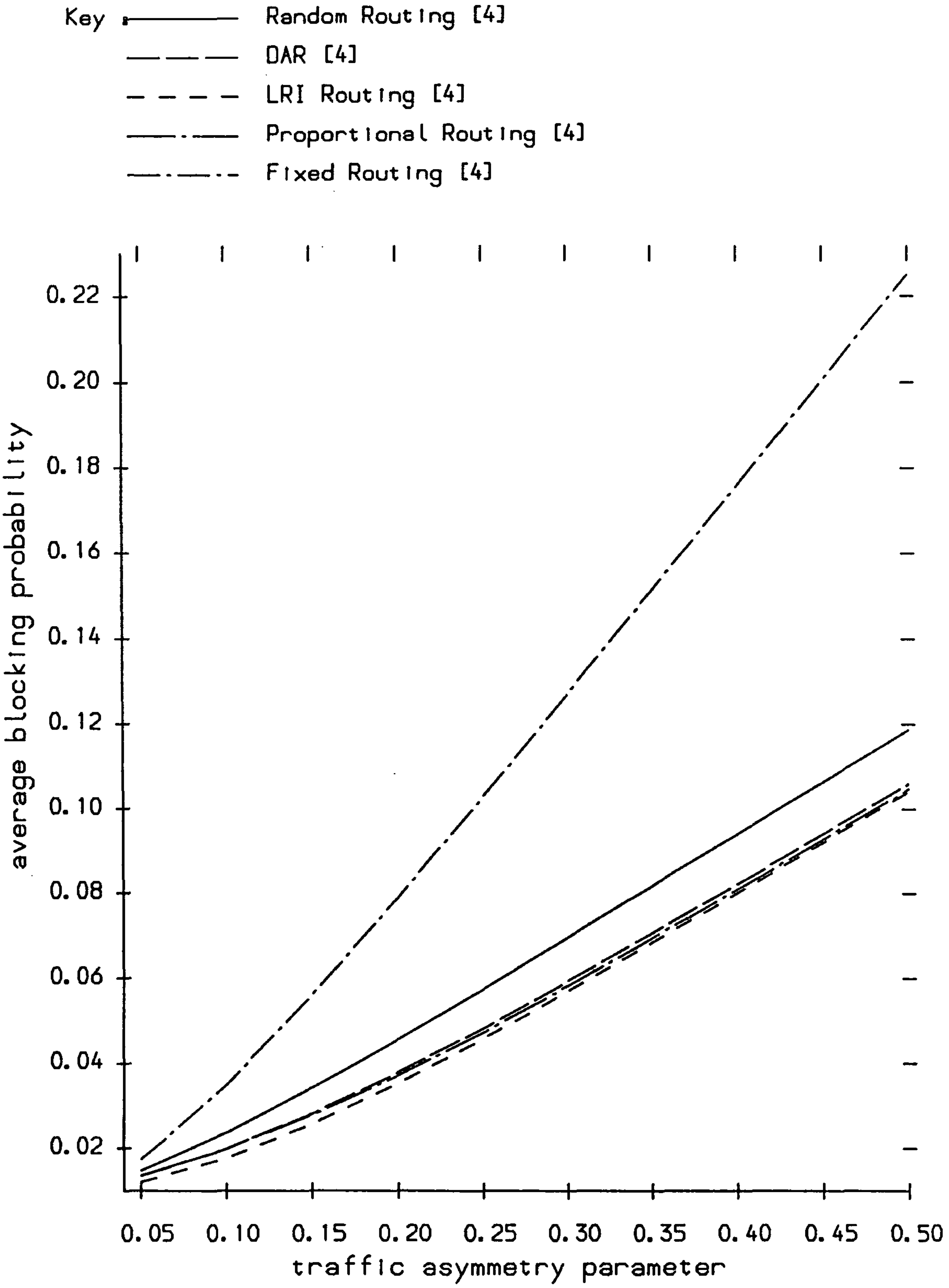


figure 4.19(a)

anlaytic results for traffic asymmetry with TRP

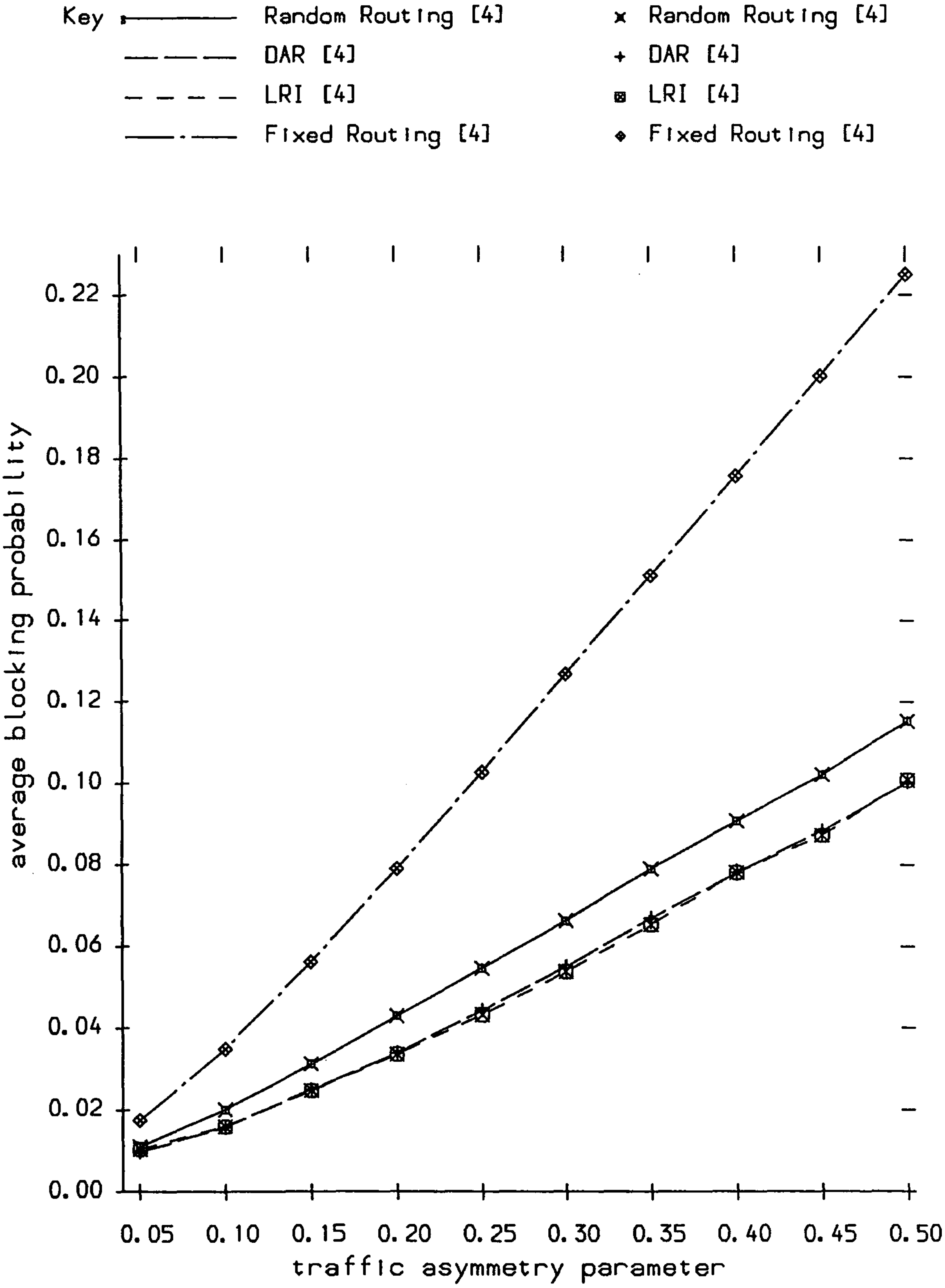


figure 4.19 (b)

simulated results for traffic asymmetry with GOS

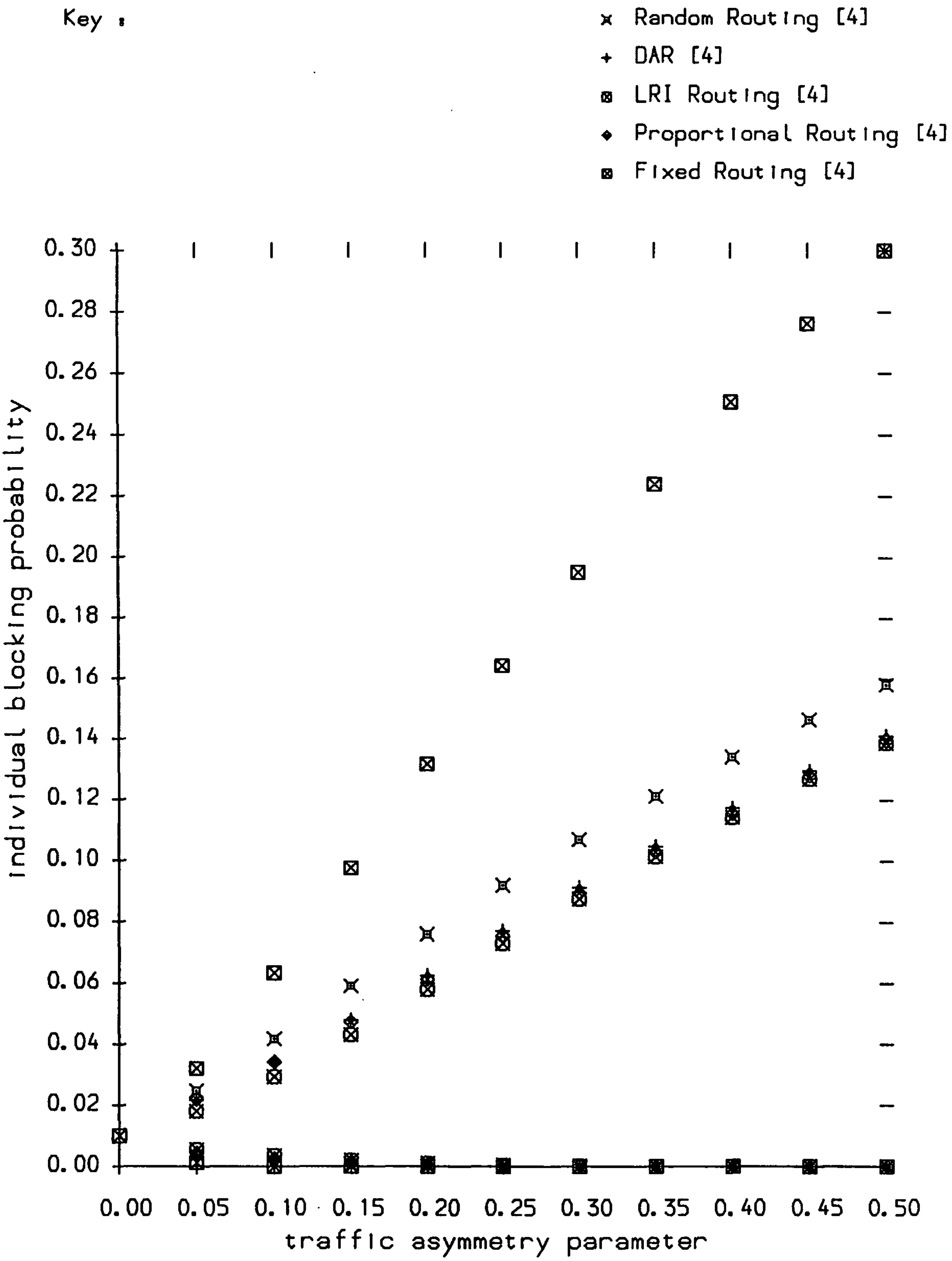


figure 4.20 (a)

analytic GOS results for traffic asymmetry with TRP

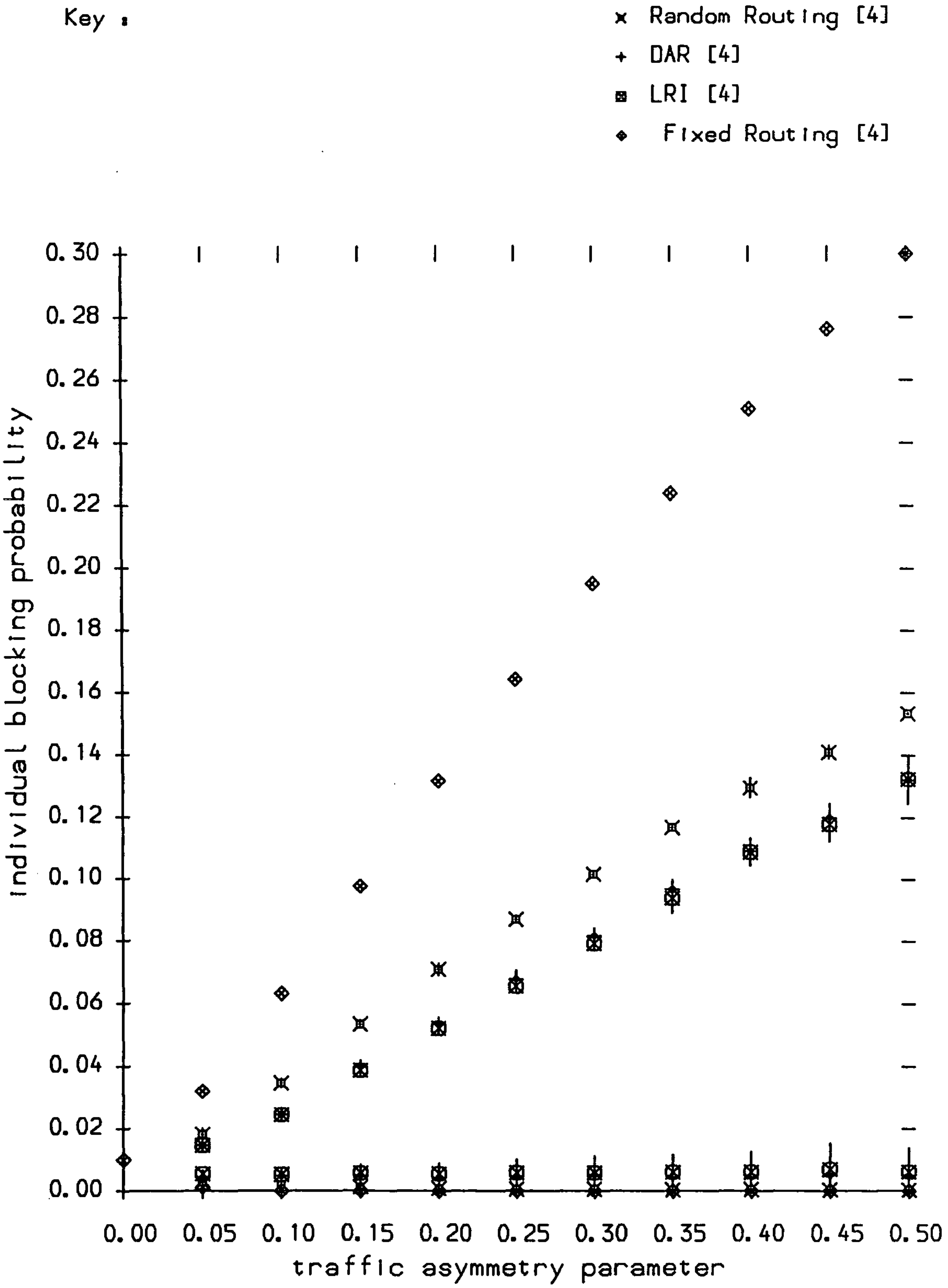


figure 4.20 (b)

simulated GOS results for traffic asymmetry with TRP

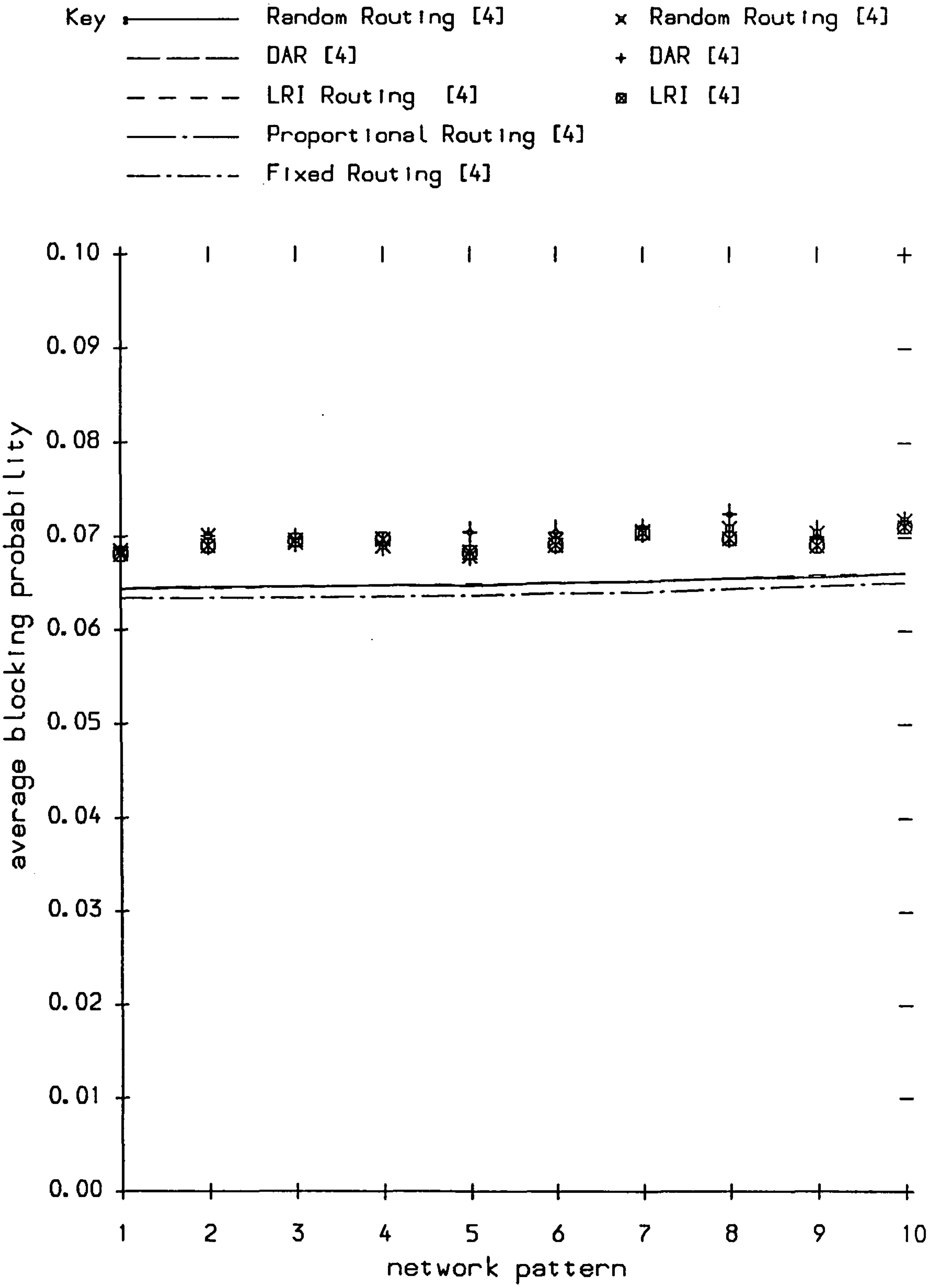


figure 4.21

Joint results for trunk asymmetry with TRP

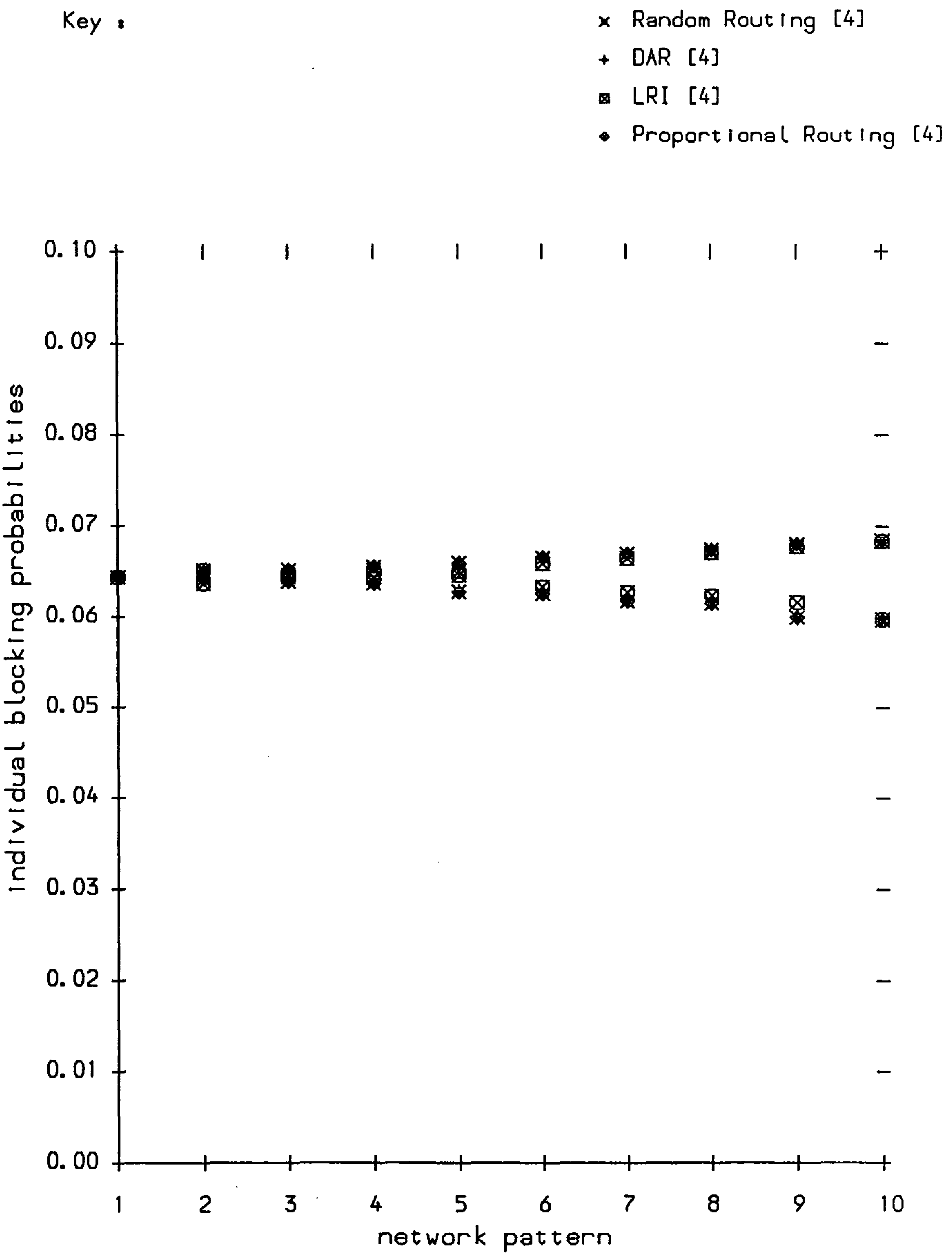


figure 4.22(a)

analytic GOS results for trunk asymmetry with GOS

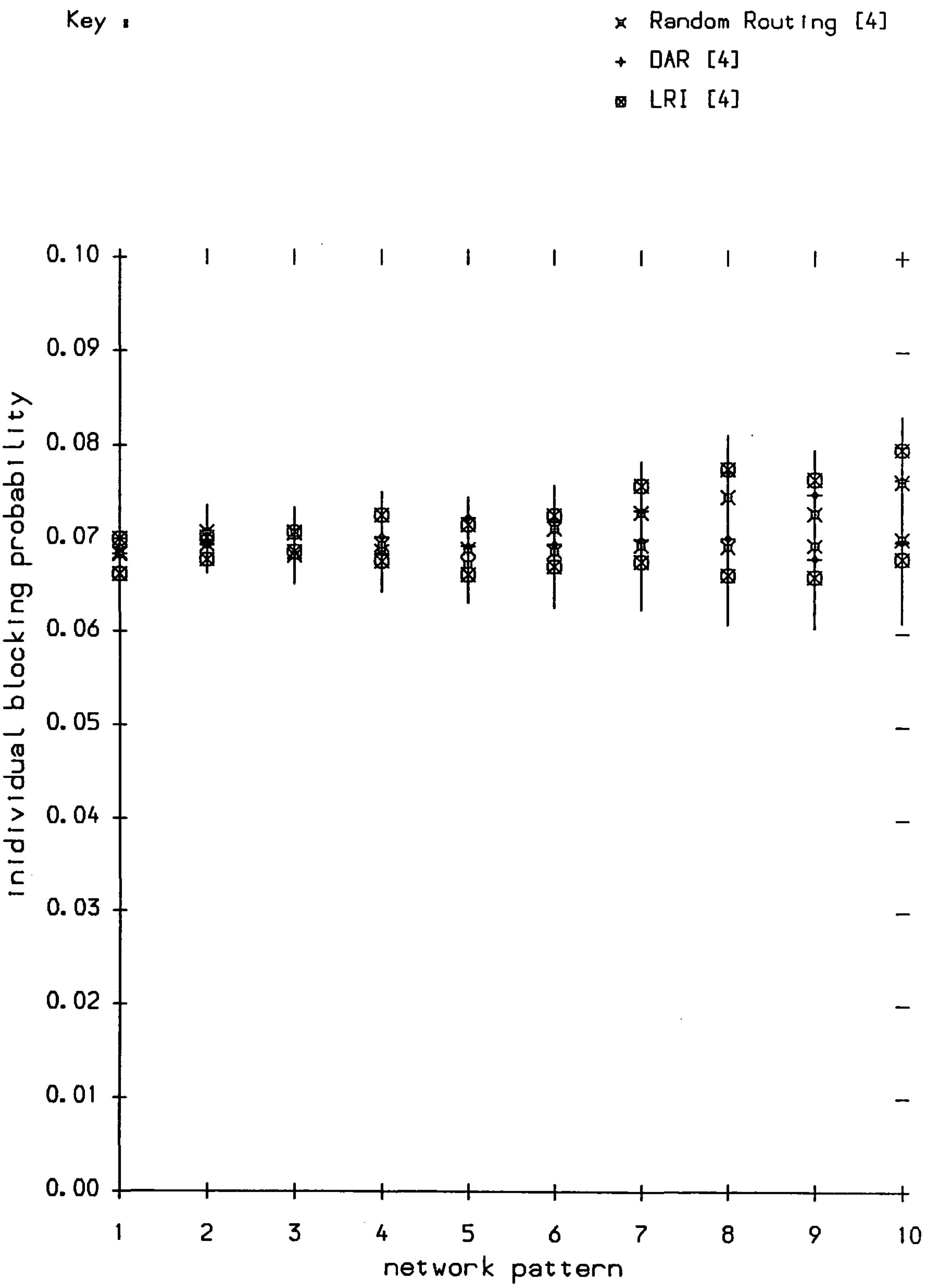


figure 4.22 (b)

simulated GOS results for trunk asymmetry with TRP

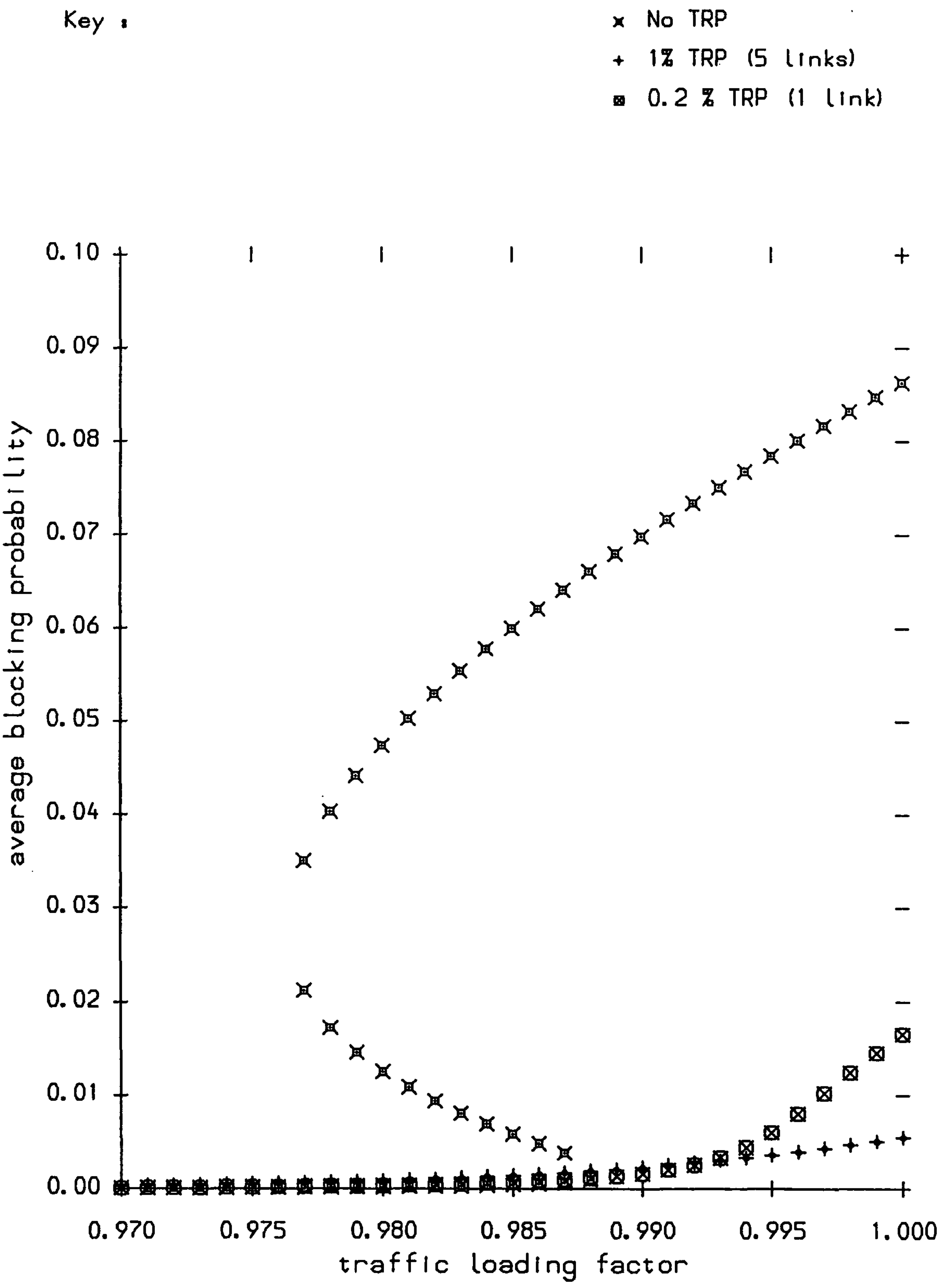


figure 4.23

Instability in a symmetrical network with TRP

Chapter 5
Analytic Modelling Techniques and the Design of a Simulation Environment for the Study Of Packet Switched Networks

5.1 Introduction

In the same way that digital technology revolutionised circuit switched networks, the same advances in software and transmission technology also allowed the development of new ways to transmit information, especially non–real time data. Rather than occupying a dedicated slot for the duration of a conversation between two users, it was realised that data need only occupy bandwidth for the period of time needed to transmit it. In this way data for different destinations could occupy the same bandwidth as and when it was required, rather than occasionally using a piece of bandwidth which had been permanently reserved for its use. Given this dynamic allocation of bandwidth, a routing strategy for this type of network has the task of directing each item of traffic over a series of links to its eventual destination.

In this chapter the workings of such a 'packet switched network' are outlined, describing the way in which technological advances have been used to implement the complex operations necessary for the efficient function of such a facility. The way in which the routing strategy directs traffic over the network is then categorised to identify the different techniques for traffic allocation. This is followed by a critical review detailing many of the routing strategies proposed in the literature, although few have been implemented in real networks to date. The review includes coverage of 'optimum' routing, defined by the minimisation of a delay based performance index, for both centralised and distributed implementation. Following this is a review of flow control procedures in packet switched networks which serve to limit the traffic within the network at any one time, to prevent a deterioration in the average performance. Wherever possible analytic work on the performance of both routing and flow control is summarised and important conclusions included.

The review highlights the way in which many current routing strategies fail to capitalise on potential network performance through inefficient traffic distribution over the network. Theoretical modelling work is presented to show how optimal performance requires the use of adaptive, bifurcated routing and adaptive flow control which react to traffic conditions. Finally the review suggests how simple algorithms, based on delay measurements, may be formulated to provide such adaptive behaviour using decentralised techniques with the minimum of transmission overhead.

The chapter concludes with a description of a simulation model which was constructed to capture the essential elements of a packet switched network and the operation of various routing policies within it. The model was designed to allow implementation on a distributed environment such as the Trnasputer based network simulator mentioned within this work.

5.2 Packet Switched Network Architecture

A packet switched network can be considered as a collection of nodes sparsely connected by a mesh of point to point links. Each node represents either an Interface Message Processor (IMP), a host computer to which customer computer equipment may communicate directly, or a Terminal Interface Processor (TIP) which combines the job of the IMP and customer machine. The bi-directional links between each node are supplied by full duplex communication channels, using either land–line, microwave or satellite technology. An example of such a small network of IMP's is shown in figure 5.1. Associated with each IMP is a host computer, attached to which terminals and other application equipment will communicate locally. These local customers will submit requests to the host for use of the network for some application. The host acts as a multiplexor for all its customers requests and submits the resulting traffic to the interface IMP for transmission across the network. In the same way the host machines acts as demultiplexors when traffic is fed to them from their associated IMP after it has been received from the network, and passes the traffic on to its customers. To complicate the picture further an IMP may have several host computers associated with it and may even act as a gateway to an entirely separate network.

Returning to the the simple problem, suppose that two hosts in the network, A and B, wish to communicate with each other. They are not directly connected and therefore must establish communication via the network. This involves host A first transferring the information it wishes to communicate to IMP A. IMP A will then initiate communication with IMP B over the network, and only then can IMP B transfer the information to its intended recipient, host B. The three distinct stages in this process are shown in figure 5.2 where the protocols adopted for information transfer at the interfaces and within the network are separatly identified. A communication system which allows two customers to transmit error free information between them, over large geographical distances, via a wide area network with multiple intermediate nodes is obviously a complex problem. In order to separate the problem into a manageable number of stages the concept of a layered architecture is introduced into the design of such systems. A layered architecture defines a number of distinct stages in the transfer of information between two locations, arranged in a vertical stack with well defined interfaces between each layer of the architecture and the layers immediately above and below them. Information is passed conceptually down the protocol stack in the node sending the information, to the bottom physical layer where it is transmitted. On reception of the raw data, it passes up the protocol stack through each layer. Each of the layers is responsible for certain features of the transmission process and peer layers (layers at the same vertical height in the stack) communicate with each other through peer protocols at each of the layers. These peer protocols together define completely how information is represented, processed and transmitted between any two points.

One important property of each of these peer layers is that its implementation should be independent of the other layers in the architecture, allowing it to be replaced or modified without affecting the remaining layers. This is accomplished automatically by requiring that any implementation of a layer in the architecture complies with the definition of the interfaces between each of the layers around it. In order that each implementation does not affect the way in which the data is treated, each layer does not modify the information passed down to it from a higher layer, but adds a header to the data it receives before passing it down to the next layer. The header contains information its peer layer requires for the implementation of the particular protocol between them. On each peer layer at the receiving node, the header is removed, the appropriate action taken according to the headers contents and the remainder of the transmission passed up to the next layer in the hierarchy. Figure 5.3 shows how such a transmission is built up as it descends the protocol stack at the transmitting node, through three layers; the User layer, the Network layer and the Data Link layer, before being transmitted over the lowest Physical layer, acquiring headers at each stage. Once at the receiving node the data climbs the stack and the header attached at each stage is removed as it ascends.

Different architectures have been designed for commercial networks, good examples being IBM's Systems Network Architecture (SNA) and DEC's Digital Network Architecture (DNA), however the International Standards Organisiations model for Open Systems Interconnection (OSI)[51] is becoming the accepted standard for network communication. The OSI model defines a protocol stack with seven layers which may be broadly divided into two groups. The first three layers of the OSI architecture known as the network service protocols are designed to transport the information across the network and deliver it to its destination in an ordered fashion. The remaining four layers which form the second, higher group of protocols, are concerned with the end–to–end interactions of the information in which the network is seen as a transparent communication medium, governed by the network service layers. Figure 5.4 represents this layered concept for a pair of host machines A and B communicating over a network via IMP's A and B. The diagram is a direct mapping of a layered architecture onto figure 5.2 which describes the physical path data takes between hosts. Two distinct sets of protocols are defined in the diagram. The first, outer set of protocols, control the transfer of information over the higher layers of the protocol and the network layers over the IMP–host interface. The second set of protocols occur at the network layer only and control the passage of information between the source and destination IMP's over the network. The path the information traverses as shown by a bold line, demonstrating how the layers are traversed form source to sink. The higher layers have been grouped into a single layer representing the peer end–to-end communication for simplicity. These layers can be thought of as collectively providing a traffic arrival rate to the network at a rate which can be accommodated by the hosts at the opposite side of the networks via their peer protocols. This then defines the load which is presented to the lower network layers for transmission.

The networks service group is divided into the three layers; Physical, Data Link and Network in ascending order. The Physical layer is responsible for the transmission of the actual data bits between neighbouring nodes in the network or across the IMP-host interface along a communication link. The communication link is characterised by a number of parameters such as transmission speed, propagation delay and bit error rate. The processing centres at each IMP or host will also have finite memory for data storage and certain response times to service interrupts for packet arrivals which must be considered in the design of interfaces for the physical layer. The major function can be summarised as the translation of data bits into electrical signals, their transmission over a communication channel and their reconstitution back into binary form at the opposite end. These data bits are passed down from and returned to the Data Link layer at the sending and receiving nodes respectively.

The Data Link layer uses *packets* of information, a term first used by the early workers on store–and–forward networks of this type. A packet is a unit of information which is handed down from the network layer for transmission over the communication link. The Data Link layer adds a header and passes it to the physical layer. When it is received at the other end of the communication channel the Data Link layer is able to use the additional information included by the peer layer at the sending station to interpret whether the packet has been successfully transmitted error free, or whether retransmission is necessary. Other information carried by the header includes previously received packet acknowledgements, flow control information, priority information, etc. Error free packets of information are then passed up to the receiving stations Network layer. Parameter selection at this layer can have pronounced affects on the performance of the link. The size of the data packet, flow control restrictions and type of retransmission scheme used can be critical especially if the physical layer characteristics limit the capacity of the link in some way, e.g. a long propagation delay over a satellite link or a noisy link with a high error rate causing frequent packet retransmissions.

The third and highest layer in the network services group, the Network layer combines the joint functions of routing and flow control. The flow control at this level is over the entire network rather than the single link at the Data link layer. The flow control attempts to prevent the destination being overrun with packets and the onset of network performance degradation by the admission of too much traffic to the network at any one time. The routing algorithm guides the traffic between the source and the destination IMP's over the network in the most efficient possible way. Two main routing types can be identified, datagram routing and virtual circuit routing. In datagram routing each packet is treated by the routing algorithm as an individual and routed using only the desired destination held within the Network layer header. Virtual circuit routing recognises packets from a particular customer and establishes a single route over which they are all sent. A small end–to–end acknowledgement packet is generated on the safe arrival of each packet at the destination station. This packet is then transmitted back over the route taken by the packet to the source. This allows packets which fail to arrive a chance to be retransmitted over the whole network. Datagram routing is much simpler and relies on the higher protocol layers to implement this end–to–end acknowledgement function. Each datagram packet carries a header identifying the destination node and a sequence number so that the entire message may be reconstituted at the other end of the network. Local error checking is provided by acknowledgement packets generated at each hop by the transmission protocol at the Data Link layer for both datagram and V.C. traffic.

The Network layer can also be thought of as the multiplexing level where messages for different destinations and arriving from different sources are identified and separated before being passed up to the higher layers at their destinations and labelled before being passed down for transmission to the Data Link layer at originating nodes. The concept of a logical link is used by the OSI model to describe the relationship in th Network layer between packets with the same source and destination and the network layer itself. Each packet over a logical link is considered part of the same session. Thus while the Data Link layers are connected by a single link in figure 5.4 it is more correct to show the Network layers connected by multiple virtual links to represent these multiple logical links.

The additional layers of architecture, represented by the top layer of the figure 5.4 combine to form a complete communication environment which can then be used directly by application packages by interface to the highest layers. The higher layers provide functions such as session control, code conversion and security. Such functions are not required within the network as they relate to the way in which the customers at each end of the network perceive and manipulate the information transported to and from them. The most important of these higher functions from the networks point of view takes place at the Transport layer, the fourth layer of the architecture, which provides a secure end–to–end network connection function for the layers above it. It has the job of breaking down the messages passed to it into the packets referred to in the Network layer. If virtual circuit routing is used the packets from a single message or stream of messages from a single session make up the packets sent over a logical channel. For simpler datagram routing the transport layer may be called upon to provide virtual circuit end–to–end acknowledgement packets and other error recovery procedures.

This simplified model of the communication architecture provides a good representation of the system, but is complicated by the presence of two separate protocol stacks; the host–IMP interface protocol stack and the network IMP to IMP architecture within the network. It is possible to further simplify the model without losing information on the part of the structure we wish to explore, specifically flow control and routing procedures in the network layer of the wide area network. Both these functions are completely defined within the network services layers of the IMP to IMP protocol layers. The IMP–host interface protocol serves only to provide traffic to each IMP of the network at a controlled rate and accept it at the other end. If the assumption is then made that the limiting factor for the acceptance and removal of traffic to and from the network is the network itself, rather than the interface protocols, then modelling the IMP–host interface becomes unnecessary. Introducing this simplification the model can be redefined as shown in figure 5.5. Here only the network service layers of the network are shown with an implied traffic arrival rate at the interface IMP's. The first two layers are unchanged but the third layer has been expanded into two separate sections. The first section covers the routing strategy, where the protocol is responsible for the selection of the outgoing link for each packet arriving at the node. The second section contains the flow control procedures which may be local or, as shown, may extend over the whole network between the source and destination nodes in the network.

5.3 Routing Strategies and Congestion Control

5.3.1 Classification of Routing Strategies

Just as routing strategies in circuit–switched networks were classified in Chapter 3, an analogous division of routing techniques can be identified in algorithms designed for use in packet switched environments. The algorithm is categorised in terms of its method of selection of routes, a route being defined as a path over which traffic is directed from its source node in the network to its destination node, also in the network. To complete the definition, a path consists of the links and intermediate nodes a packet traverses in its journey between the two terminal nodes. The categories into which these routing algorithms are divided are shown in figure 5.6. The structure bears some ressemblence to its circuit–switched counterpart, but also differs in several important ways, reflecting the different natures of the two architectures. An important distinction in the definition of a packet switched routing algorithm's operation is in the way in which a path is selected. In a circuit–switched environment the routing algorithm selected a path which consisted of either a direct connection to the destination node or a tandem path involving an intermediate node. In both cases the path was defined completely by the source node's selection of an outgoing link. In general a packet switched network does not enjoy the connectivity of such networks and forms a much more sparsely connected mesh topology. Many pairs of nodes do not share a direct link and some may be separated by several intermediate nodes over even the route involving the shortest number of hops. If follows therefore that routing cannot be controlled directly from the originating node, but must be implemented as a series of steps, one at each stage of the path to the destination. The task of the routing algorithm is therefore to provide an outgoing link at each stage in the path of each traffic source to each destination.

A routing algorithm is said to be fixed if the paths over which traffic traverse the network and the proportion of traffic flowing over each of the paths are invariant. The routing algorithm is therefore unaffected by both the level of traffic flowing from the source over the prescribed paths or the traffic encountered from other sources flowing throughout the network. Alternatively if the paths or proportion of traffic flowing over them are modified by either of these two factors, the algorithm is termed adaptive.

Adaptive strategies are further divided into three sub–categories, dynamic, quasi–static and hybrid, the latter simply being a combination of the two former techniques. In a quasi–static scheme the key feature is the concept of periodically updating the paths and traffic allocation over them. The modifications at each update are based on measurements taken over the period since the previous update took place. Dynamic strategies operate on a much smaller time interval and typically the routing configuration is re–evaluated for each packet that uses the algorithm to select a path through the network. The hybrid scheme, as was mentioned above, seeks to combine the two operations. Periodic updates using a quasi–static process would form the basis for the major routing decisions affecting traffic flow in the network at each iteration. However some degree of freedom would also be included for the dynamic part of the network to exploit local fluctuations over the interval between each update from the quasi–static process.

Finally the question of how a particular algorithm is implemented can be used to divide algorithms into distributed or centralised form. In a centralised implementation complete paths for each traffic source in the network are formulated at a single location. The necessary information about network status must therefore be gathered from the network and the routing algorithms decisions transmitted back to each node. In a distributed implementation the components of each path through the network, for each traffic source, are calculated by the network components throughout the paths themselves. Within distributed routing two techniques can be identified. If the routing algorithm bases its decisions on information available purely within the node itself, it is termed an 'isolated' policy. The alternative is an algorithm which exchanges information with its nearest neighbours to provide a greater appreciation of the surrounding environment. Dynamic algorithms would seem to fall naturally into the distributed, isolated category as the requirement for a rapid response to local events, like individual packet arrivals, precludes dialogue with other nodes for every decision. Alternatively quasi–static algorithms have the choice of distributed or centralised implementation as the periodic updating could equally possibly be carried out via a central location or across the network.

Another, quite separate, but equally important, way in which algorithms can be divided concerns the form of the paths produced rather than the method of formulation. Specifically, for each source–destination pair an algorithm can generate either a single or multiple paths between the nodes at any one time. Algorithms which generate single paths are known as shortest path strategies. Typically the algorithm assigns a value to each link of the network and then calculates the path, through the network, of least cost for each traffic source. If all links are assigned equal values or values based on the inverse of their transmission speed, then the algorithms used to generate the routing tables are known as minimum hop and minimum delay techniques respectively. Notice how algorithms falling into this category could also be classified as *deterministic* as all the traffic from a particular traffic source follows the same 'shortest' path at any one time. Algorithms which generate multiple paths are known as bifurcated strategies, a generalisation of the strict definition of the term suggesting only a binary division. Each time a route is established, a selection is made at each stage of the network over the set of outgoing links (or a subset of them) to form the path. The selection at each stage is based on a probability distribution over each set of links, the distribution being determined by the specific algorithm. The bifurcated selection procedure is therefore basically *stochastic* in nature.

5.3.2 Routing Techniques In Packet Switched Networks

As the preceding section suggests the actual routing scheme in a store and forward network forms only a very small part of the overall network architecture. The development of advanced architectures for such networks represents a highly complex and integrated feat of engineering. Because of this the design of real networks has tended to concentrate more on the technical standards for component interface and the establishment of protocols for peer communications. Ultimately this has lead to the development of standardised architectures such as SNA, DNA and the emerging ISO and CCITT standards[51–53]. In the earlier stages of development the question of routing was often solved by the adoption of a fixed routing policy, where packets from each source were routed over an invariant path to the destination. Alternative paths were only considered in the event of a failure of some topological component. The SITA network, connecting airline reservation computers with agent's terminals was an example of an early network where just such a policy was implemented[54]. This policy is still used today in software for many small private networks where performance optimisation is sacrificed for simplicity. The best projected routing tables are calculated from predicted traffic levels using an off–line optimisation program to produce the routing tables at each node, usually based on some shortest path criteria. Other fixed policies which should be mentioned include a process known as 'flooding' where a message is transmitted on all outgoing links from a node. This technique can be especially useful when information has to be broadcast throughout a network of some general topology. The additional overhead caused by the reception of multiple identical messages at each node is often offset by the advantages in simplicity and speed of distribution flooding offers. Another useful technique is random routing, where paths are established through the network according to some off–line technique and each assigned a proportion of the traffic. Such a policy provides a useful bound for comparison with adaptive bifurcated strategies, whose selection probability of each path may be modified by the state of the network.

The fixed policy of this type will provide a balanced routing policy for the traffic levels for which it was designed. However such invariant schemes cannot modify their behaviour to react to traffic fluctuations or more seriously, sustained traffic patterns which were not envisaged at the design stage of the network. In these cases simulation studies have produced substantial evidence to substantiate the claim that a more flexible approach is required if the performance of the network is not to be severely compromised[55,56]. Various networks and network architectures have emerged which have developed adaptive strategies which retain the simplicity of shortest path strategies but include the ability to adapt to changes in the network environment. An excellent review paper by Schwartz and Stern[57] outlined the approach of some of better known examples including ARPANET,TYMNET,DATAPAC as well as the SNA and DNA architectures. Similarities are found in the way the routing strategies are formulated in each case. All the strategies use the shortest path concept but each has its own scheme to identify the cost assigned to each link. The functions vary from the simple assignment of a fixed cost based on the capacity of a link up to complex algorithms which try to assign a cost to the link based on the congestion measured over the link.

One of the best documented examples of the development of a network routing policy is that of the ARPANET packet switched network. Originated in 1969 ARPANET was one of the pilot packet switched networks in the United States. Its initial routing algorithm fell into the adaptive, distributed and shortest path category. Every node in the network maintained a delay table which assigned a value for the delay from this node to each destination over each of the possible outgoing links. The delay table was used to construct a minimum delay table and a routing table. These tables stored the minimum delay value to each destination and the link to which that delay was assigned. Every $\frac{2}{3}$ seconds each node would sychronously exchange minimum delay tables with each of their nearest neighbours. Each entry in the received minimum delay tables from the nearest neighbours is added to the local delay from the receiving node to that neighbouring node. This provides the receiving node with a new estimate of the total delay, via the transmitting node, to each of the destinations and the delay table can then be updated with this information. The local value for the delay to each of nearest neighbours was derived from the instantaneous queue size in the output buffer of each link[58]. This implementation suffered from a number of problems concerned with the response of the routing tables to certain network events, especially 'bad news' such as component failure, and caused instability in some routing patterns subjected to traffic fluctuations. The short period of time between table updating combined with the simple method of local delay measurement compounded by the synchronous exchange of information caused a number of identifiable problems.

In addition the distributed nature of the algorithm led to problems in the maintenance of consistent routing tables throughout the network, causing looping and misdirection of packets. The algorithm was modified a number of times before eventually being completely replaced by a new scheme[59]. The new scheme still formulated shortest paths via an adaptive, quasi–static algorithm, but no longer used an entirely distributed approach. Each node in the network maintained a complete database of the network topology and the measured delay associated with each link in the network. Each node then calculated the current shortest path to every other node and formed a routing table directing traffic for each destination over the appropriate outgoing link using a centralised algorithm based on a Dijkstra's technique[60]. The average time to queue and transmit a packet over a 10 second time interval was used to calculate the link metric for the algorithm. If the latest calculated value differed from the last by greater than a threshold value, the new value is transmitted to every node in the network via a flooding algorithm and used to update each node's database. The threshold value was reset to an initial maximum value after each update and decreased linearly to zero after a period of 60 seconds guaranteeing at least one update every minute and allowing more frequent updates to take place in response to considerable traffic fluctuations.

Other authors have continued to refine the original algorithm in an attempt to overcome the problems identified in the initial investigations[61–63]. All of these later algorithms seek to reduce looping problems (by far the most persistent problem) by the introduction of more intelligence into the updating procedures at each node. A comprehensive description of their modifications is given in a paper by Schwartz[64] who analyses their relative performances by calculation of their convergence times and the overhead generated by each process.

Shortest path routing offers a simple methodology which seeks out the single least cost path between two nodes for each source in the network. However consider the following simple problem: over which path does a traffic source route its information when it is presented with a choice of two routes of equal cost? The algorithm must select a single path and yet intuitively common sense tells us the source would be better served by some division of the traffic over the two equally available routes. A variation on shortest path routing, bifurcated shortest path routing, does just this and assigns equal portions of traffic to each path of mimimum cost with improved performance over its restricted alternative. It is a short step to suggest that better performance over the network should be available if all sources were allowed to route their traffic over multiple paths through the network. This step leads us directly to the concept of bifurcated routing strategies. Shortest path algorithms, by definition, calculate the minimum cost path for each source, subject to some metric over the links of the network. The technique can therefore be criticised as myopic in that there is no element of co-operation between the traffic sources. Each source seeks out the shortest path, based on the link metrics, regardless of the path taken by other sources. In fact the solution to the shortest path problem can easily be shown to result in the linear optimisation problem

$$\min \sum_{\substack{all\ links \\ i}} f_i l_i$$

where l_i is the metric for link i and f_i is the total flow assigned to that link. Notice how the penalty function of using any link only increases linearly with traffic allocation. Bifurcated strategies allow the implementation of schemes which allow traffic to be allocated across a wider bandwidth of the network, allowing better utilisation of the network. In particular they allow the solution of more complex and representative network performance criteria. The classic approach used to optimise network performance using this strategy has been the formulation and solution of the objective function given by the minimisation of the average network delay, E(T),

$$E(T) = \frac{1}{\gamma} \sum_{\substack{over\ all \\ links(i,k)}} D_{ik}(f_{ik}), \qquad f_{ik} = \lambda_{ik} C_{ik} / \mu_{ik} \tag{5.1}$$

where $D_{ik}(\cdot)$ is the delay over the link connecting nodes i and k, expressed as a function of f_{ik}, the total traffic flowing over the link in bits per second, calculated from the traffic arrival rate λ_{ik}, the transmission rate in packets per second μ_{ik} and the link capacity in bits per second C_{ik}. γ, the summation of the traffic arrival rates λ_{ik}, is a constant and can be dropped from the objective function.

In addition each traffic source between any two nodes i and j, r_{ij} bits per second, generates a flow conservation equation at node l of an N node network of the form

$$\sum_{m=1}^{N} f_{ml}^{ij} - \sum_{n=1}^{N} f_{ln}^{ij} = \begin{cases} -r_{ij} & \text{if l=i} \\ r_{ij} & \text{if l=j} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where f_{mn}^{ij} is the flow on link (m,n) specifically associated with traffic flowing between nodes i and j where all $f_{mn}^{ij} \geq 0$ and $f_{mn}^{ij} \equiv 0$ if there is no direct link between nodes n and m. This is simply a statement of conservation of flow in, out and around the network through each node. The additional constraint that $f_{ik} < C_{ik}$ where C_{ik} is the capacity of link (i,k) is usually omitted as for the class of function used $D(f_{ik}) \rightarrow \infty$ as $f_{ik} \rightarrow C_{ik}$.

This problem is known as a constrained optimisation, multicommodity flow problem. The object is to find a set of flows f_{mn}^{ij} which minimises the objective function defined in equation 5.1, within the constraints imposed by equation 5.2. A number of approaches have been reported in the literature which have used various numerical techniques to solve this problem[65–68]. Each of these techniques uses a form of the delay function based on Kleinrock's classic formulae derived from the $M/M/1$ queuing model[69]. The model assumes Poisson arrival streams to the network and *independent* exponential service times at each node a packet traverses in the network. This leads to $D_{ik}(f_{ik})$ being defined as

$$D_{ik}(f_{ik}) = \frac{f_{ik}}{C_{ik} - f_{ik}} + f_{ik}t_{ik}$$

where the final term, which is sometimes dropped, represents the delay introduced by a constant propagation delay. More complex models have been formed to include features such as nodal processing[70] but add little to the basic problem of path allocation and shall not be considered here. $D_{ik}(f_{ik})$ is a strictly convex function which ensures that E(T), a combination of convex functions, is also convex. This guarantees that any algorithm which finds a local minimum has also found the single global minimum. The algorithms used to find the minimum fall loosely into one of two categories. The first technique determines necessary and sufficient conditions for optimisation within the variable space defining the problem. The algorithm then modifies the flow on each link iteratively toward the fulfillment of these conditions and so the optimisation of the objective function. The second technique tackles the optimisation directly using gradient techniques subject to the constraints. Some of the more notable algorithms using these two techniques are outlined below for comparison.

Fratta et al[65] show that for the optimum flow allocation, $\underline{f}$, which minimises $D(\underline{f})$, the total delay function, traffic from each source will only flow over paths with the same marginal delay, where that marginal delay is less that that of all other possible paths through the network. Marginal delay over a path π is defined as the sum of the marginal delays over each link in the path k, $k \epsilon \pi$, defined as $\delta D_T / \delta f_k$, the partial derivative of the total delay over the network, D_T with respect to the flow over link k, f_k. Successive iterations converge to produce the optimal flow by the 'flow deviation method'. At each iteration traffic is transferred at each stage from paths with high marginal gain to those with a lower value. Specifically each iteration replaces the set flows f^n by f^{n+1} which satisfies

$$D(f^{n+1}) = \min_{\lambda} D[(1-\lambda)f^n + \lambda\nu] \qquad 0 \leq \lambda \leq 1$$

where ν is the shortest route flow calculated under the metric l_k, $l_k = \delta D_T / \delta f_k$. This results in an algorithm which converges using the steepest descent method. In contrast the gradient projection method of Schwartz et al[66] also used the derivative of the objective function with respect to link flow but incorporates a projection operator. The projection operator is calculated for each iteration of the algorithm and projects the gradient onto the set of constraint hyperplanes, modifying it to eliminate all directions incompatible with the constraints defined by the conservation equations. The set of flows $\underline{f}$ at each iteration is given by the formulae

$$\underline{f}^{i+1} = \underline{f}^{i} - hP\nabla\overline{T}$$

where $\nabla\overline{T}$ is the gradient matrix, P is the projection matrix and h is a scalar constant. h is bounded by the minimum value which would create a negative flow over one or more links after the iteration and is chosen to produce the smallest value of the objective function within that limit. When $P\nabla\overline{T} \equiv 0$ then no feasible move can be made in the direction of minimisation and the optimal solution has been found. Notice how gradients are used not only to find a feasible direction but also, in conjunction with the projection operator to determine optimisation directly, rather than through necessary conditions. Cantor et al[67] cleverly combined the two schemes by noting that any feasible multicommodity flow within the network could be made up of a convex combination of 'extremal' or shortest route flows. The optimal route flow can therefore be generated by a combination of up to NA shortest route flows, where NA is the number of arcs in the network. The metric used to generate such flows is once again the marginal delay with respect to the flow on each link. The problem is now reduced, or decomposed, into one of finding the set of extremal flows and the fraction of each flow which minimises the objective function. The Kuhn-Tucker theorem[71] is used to show that if an extremal flow forms part of the solution set it must produce the same, minimal marginal delay as other shortest flows within the solution set or alternatively a flow f_i is only part of the optimum set if

$$\sum_{i=1}^{NA} l_i f_i = \min_k \{\sum_{i=1}^{NA} l_i \varphi_i^{(k)}\}$$

where φ is the set of extremal flows. On each iteration of the algorithm the optimum combination of NA extremals flows is calculated using the gradient projection method. The marginal link costs for this scheme are then used to calculate the minimum cost flow. If this cost is equal or greater than the extremal flows used in the solution the optimum flow allocation has been found and the algorithm terminates. If not the new extremal flow is added to the solution set of extremal flows and the algorithm continues to its next iteration.

In an alternative approach Gallagher[72] in 1977 published a paper outlining a method which modified not the path flows, but the routing variables at each node directly. The algorithm was also suitable for distributed implementation in a network. Marginal delays with respect to routing variables at each node, for commodities flowing to each destination, were used to formulate conditions for a stationary point using Lagrange multipliers. This required the equalisation of all marginal delays $\delta D_T/\phi_{ik}(j)$ where $\phi_{ik}(j)$ is the probability of routing traffic from i to k, where a direct link (i,k) is present, for traffic destined for $j \neq i$ for all $\phi_{ik}(j) > 0$. In addition this marginal delay must be less than or equal to $\delta D_T/\delta\phi_{ik}(j)$ for links on which $\phi_{ik}(j) = 0$. Or put more concisely

$$\frac{\delta D_T}{\delta\phi_{ik}(j)} \begin{cases} = \lambda_{ij} & \phi_{ik}(j) > 0 \\ \geq \lambda_{ij} & \phi_{ik}(j) = 0 \end{cases}$$

where λ_{ij} is a constant and D_T is the total network delay once more. However this does not guarantee optimisation, as the stationary point could represent a point of inflection and so an additional set of constraints are introduced for all links (i,k) and $i \neq j$,

$$D'_{ik}(f_{ik}) + \frac{\delta D_T}{\delta r_k(j)} \geq \frac{\delta D_T}{\delta r_i(j)}$$

with equality for $\phi_{ik}(j) > 0$, where $D'_{ik}(f_{ik})$ is the marginal delay over link (i,k). This is simply a statement that, at the optimum solution, the marginal delay for a traffic source is minimised and all paths which carry traffic from this source have the same value. All other paths have a higher marginal cost associated with this traffic and are unused. Finally the inequality can be rewritten in the form

$$D'_{ik}(f_{ik}) + \frac{\delta D_T}{\delta r_k(j)} - \min_{m:(i,m)} [D'_{im}(f_{im}) + \frac{\delta D_T}{\delta r_m(j)}] \geq 0 \tag{5.3}$$

In this form it is used in an iterative algorithm developed by Gallagher to find the optimum flow. At each iteration the nodes exchange marginal delays for each destination $\delta D_T/\delta r_k$. From this information and their own locally measured $D'_{ik}(f_{ik})$ a series of values for each outgoing link can be formed from equation 5.3. The magnitude of these values can then be used, via a scaling factor, to reduce the probability of selecting a non–optimal link and increasingly the probability of links which currently have the lowest marginal cost. The scaling factor determines the trade off between speed of convergence and accuracy of the solution. More recently work has been done on functions which use the second derivative of the objective function to scale the descent direction in an effort to improve convergence rates. In these algorithms step size becomes a function of these second derivatives and the original algorithm has been shown to be a special case of these much more general techniques known as projected Newton methods[73]. To finish off this review of techniques for solving the optimal assignment of flow in a packet switched network Stern[68] produced a paper outlining his own algorithm for a distributed solution. The nodes exchanged the same information as in the Gallagher algorithm but Stern's algorithm used relaxation techniques to solve the flow assignment, from which routing tables were generated. One of the advantages claimed by Stern was that his algorithm was asynchronous and therefore more flexible.

The model has also been expanded by several authors to include the problem of capacity assignment. Ng and Fratta et al[65,74] both show that the basic routing model could be modified to determine not only the optimum flow allocation in the network for a given topology, but also the capacity to be assigned to each of the links. Ng formulated a model based on $m - M/M/1$ link queues to minimise the capacity and optimise flow allocation to produce an average message delay below a design threshold. The link queue model ensures that once again the objective function is convex and is justified by the concept of multiple, low capacity trunks between nodes rather than a single high capacity link as in the $M/M/1$ models. Fratta et al use a linear cost function for the capacity assigned to a link and defines a maximum total capacity allocation for the network which introduces an additional constraint function. They then go on to show how the flow deviation technique can be used to calculate the optimum non-bifurcated flow in large, balanced networks. This technique can then be used as a heuristic method for the generation of routing tables for implementation in operational networks which still favour non–bifurcated solutions.

Another similar heuristic method based on minimisation of delay has been suggested by Frank and Chow[75] and compared with the optimum solution, capturing much of the benefit of the exact solution. The algorithm first locates traffic between directly connected nodes and always allocates it to the direct path. The algorithm then considers routing traffic which can reach its destination via multi–hop paths, beginning with a single intermediate node and increasing the path length at each iteration. Traffic is allocated paths according to link utilisation, or residual capacity. Any link which is considered to have been allotted its maximum traffic allocation according to a pre–defined threshold is removed from the possible selections. This continues until all traffic is allocated or no more paths can be found through the network in which case the traffic requirements are unable to be satisfied for the present network and threshold values.

In terms of the applicability of the algorithms described above to actual implementation in a packet switched network, the most important dividing factor is their ability to support centralised or distributed execution. The centralised algorithms necessitate, by definition, a central controller to gather, process and inform each of the nodes of the algorithms solution. Similar arguments to those developed in the discussion on circuit switched routing algorithms can be invoked again. Centralised control introduces vulnerability by placing the responsibility at a central location and generates an overhead, either in cost if separate lines are used to the controller, or in link utilisation if the network itself is used to communicate with the network management. In addition there is also the question of the applicability of the centralised algorithms decisions. In a rapidly changing environment, the delay in collecting, solving and distributing the routing algorithm's solution may produce outdated solutions to the immediate traffic requirements.

The distributed approach offers a more responsive technique as the routing variables can be updated at every iteration of the algorithm, either explicitly or via calculation from the flows at each stage. However there are two inter–related problems which arise from the basic quasi–static nature of the algorithm's formulation. First the algorithm requires an accurate measurement of the marginal delay over each link in order to calculate the modifications to the flows and/or routing variables. Second, in order to operate efficiently the algorithm must converge to the optimum solution and then constantly modify the routing tables to keep in step with changes in the traffic pattern across the network. The problem is immediate and obvious. The more often the algorithm iterates the more quickly it will respond to traffic fluctuations. Failure to do so would clearly lead to incorrect routing patterns and a degradation in performance. However the smaller the time interval between iterations and calculation of the marginal delays, the less accurate the estimation of link utilisation will be, again leading to inaccurate traffic patterns. For successful management of network resources the algorithms require a 'quasi–static environment' in which traffic fluctuates gradually allowing the algorithms time to gather accurate link utilisation measurements and track the traffic patterns.

These limitations in the flexibility of 'quasi–static' strategies leads to the consideration of the final class of algorithm, dynamic routing, most notable for its distributed, continuous monitoring and updating of its local environment. This last property allows it to react quickly to rapid changes in a way more static algorithms may find hard to emulate. In the remainder of this section a number of schemes for such implementations are reported along with some suggestions for others based on existing work. Dynamic routing strategies have been suggested for as long as the problem of routing has been under discussion. Baran[76] suggested the simplest of schemes in his key paper on the development of distributed store–and–forward networks. His 'hot potato' algorithm was based on the method of telegraph operators who routed messages through switching exchanges by hand. The operators technique involved routing the call over the best free link or failing that the first available outward link in the general direction of the destination. For a packet switched network the 'first available link' can be thought of as the link with the shortest queue, and the output set of links can be determined by observing the hop count of packets arriving *at* the node *from* each destination. The basic algorithm can be refined by the addition of constant values associated with each link for a particular destination. This leads to a 'shortest queue + bias' algorithm, which directs the traffic over prefered routes, especially under light or balanced traffic patterns. This idea of using packets travelling the opposite direction was also used by Fultz[55] in his 'backward learning' technique to formulate an algorithm based on delay rather than hop count. The time a packet from a particular source spends in the network before arriving at the node is used as a measure of congestion over the link on which the packet arrives. This is used to calculate a value associated with the delay for a packet travelling to the source. The algorithm was presented in shortest path form where the delay tables formed and updated by each packet are periodically used to update the routing tables for each destination. However exactly the same algorithm could be used in a dynamic algorithm by using the delay tables directly to select the most favourable link for each packet when it requests an outgoing link.

In the same way other isolated algorithms, such as those described by Chou at al[77], could be used as a basis for a dynamic routing policy which continuously update the link selection based on the most recently available information. Chou considered a class of algorithm which can be considered as an extension of the shortest queue + bias scheme. The queue length at each node was used to calculate a value associated with each outgoing link using a metric of the form $a_0 + a_1Q + a_2Q^2$ where Q is the queue size and a_0, a_1 and a_2 are constants where $a_0 \gg a_1 \gg a_2$. By careful selection of the metric coefficients Chou was able to develop algorithms that tended to produce near deterministic behaviour at light loads and progressively more adaptive behaviour as the load increased. Chou concluded from comparative studies between deterministic and adaptive polices of this type that the best routing policy depended on the traffic patterns to which it was subjected. Deterministic policies were superior for balanced networks, but adaptive policies performed equally well in unbalanced networks and networks in which traffic surges were introduced. Furthermore in a chaotic environment, defined as extremely unbalanced, adaptive policies proved more efficient at preventing congestion than their deterministic counterparts.

Rudin[78] suggested combining the global perspective of a centralised scheme with the flexibility of dynamic link selection to produce a hybrid or constrained dynamic algorithm known as 'delta routing'. The centralised part of the algorithm periodically collects statistics on the network performance over the update period and calculates the n shortest paths through the network for each traffic source. The central controller then compares the delay associated with each path for a traffic source. If the delay associated with a path is greater then the path of minimum delay by an amount δ then the path is discarded, otherwise it is retained. The network is then informed of all paths surviving this selection procedure. The δ parameter defines the degree of control the centralised network controller has over the network. If δ is small, the routing strategy will tend to a shortest path, minimum cost routing policy. If δ is made larger, the individual nodes will have more choice in the selection of outgoing links. The selection of links from amongst the allowed subset is controlled by the dynamic part of the algorithm which assigns traffic to the outgoing link with the smallest queue. The algorithm was compared with a scheme utilising the purely centralised part of the algorithm and 'proportional routing' another centralised scheme where the division of traffic over outgoing links is incrementally updated according to reported delays over each path. 'Delta routing' performed well over a range of networks and traffic patterns. Significantly Rudin also noted that the optimum value of *delta* depended on the environment, which agrees with Chou's observation that the usefulness of adaptability is determined by its necessity.

In a later paper Rudin[79] took up the analysis of his dynamic routing again, comparing his delta routing strategy with minimum cost and minimum delay shortest path strategies in simulation studies including window based flow control. Delta routing behaved admirably at intermediate loads but there was a case for its suppression at high loads where non-minimum cost paths lead to inefficient use of resources. It was also demonstrated that flow control can have a profound effect on the performance of routing algorithms in networks and comparisons made without such schemes can be misleading. In conclusion the algorithm has much to recommend it, especially the ability to limit the dynamic content of the algorithm to prevent the inefficient use of network resources, a trait associated with purely dynamic policies. Unfortunately in order to achieve this, a centralised component has had to be introduced into the network and 'optimised' for the environment.

The problem associated with these isolated routing strategies is obviously their lack of appreciation of the global network state. In general the flows generated by such algorithms are inferior to those associated with global policies and tend only to be efficient within their locality. This leads to the generation of longer paths utilising more network resources which can lead to a degradation in performance, especially at high loads when minimum hop paths become increasingly attractive. Another problem is often the algorithm's reliance on instantaneous measurements like queue sizes which are prone to stochastic fluctuations leading to the formation of loops which can only increase congestion. The final algorithm introduced here is fully distributed and isolated in that neighbouring nodes do not communicate routing information explicitly. However the protocol does use information supplied by network protocols to update its route selection. The algorithm uses the delay over each link to the destination as a parameter to update the probability of selecting that link using learning automata at each node[80]. The reported delay over a link i, d_i is used to calculate a normalised penalty parameter p where

$$p = \sqrt{\frac{(d_{min})}{d_i}}$$

and d_{min} is the minimum reported delay, which is continuously updated to keep track of network changes such that

$$d_{min} = \begin{cases} d_i & \text{if } d_i \leq d_{min} \\ (1-\lambda)d_{min} + \lambda d_i & \text{if } d_i > d_{min} \end{cases}$$

The penalty parameter, p, is then used to update the probability distribution, r, over the set of output links according to the functions

$$r_i = r_i + a(1.0 - p)[1.0 - r_i]$$

and

$$r_j = r_j + a(1.0 - p)r_j, \qquad j \neq i$$

where a is a constant, $0 < a < 1.0$, and N is the number of outgoing links. The use of delay to the destination as an update parameter introduces global information to the local decision process which tends to smooth out local stochastic fluctuations but track changes in traffic patterns. Work done on the characteristics of such schemes has suggested that the traffic patterns converge to generate and route traffic through the network over paths of equal delay for each source[65,81]. As Agnew demonstrated[82] in his comparison of equilibrium and optimum routing, the equalisation of delay over paths through a network falls short of the criteria for optimal routing and tends to route too much traffic over some paths. More recently Mason[83] has applied the Kleinrock delay formulae to work done by Dafermos[84] who showed that the solution to the flow over each link of the network for the equalisation of path delays can also be formulated as the optimisation of a multicommodity flow problem.

Dafermos described the equalisation of delay by each source as 'user optimised' rather than 'system optimised'. The cost function over each link, $D_i(f_i)$, is shared by each of the users of the link, leading to a concept of price/unit flow or mean link cost equal to $D_i(f_i)/f_i$ seen by each unit of flow over the link. Equalisation of delay is achieved when all paths carrying traffic for a source have equal mean cost, which is the sum of the mean link costs for each path. Dafermos goes on to show that this condition can be reformulated as an optimisation of a multicommodity flow problem of the same form as the 'system optimisation' problem using a modified link cost function $D_i^*(f_i)$, where

$$D_i^*(f_i) = \int_o^{f_i} d_i(x)dx$$

The mean link cost function for the $M/M/1$ model, d_l, becomes $d_l = 1/(C_l - f_l)$, leading to an objective function for the equalisation problem using the Kleinrock model given by

$$\min_{\underline{f}} \sum_{\substack{all\ links \\ i}} ln \frac{C_i}{C_i - f_i}$$

which can be solved using any of the ways previously discussed. The direct optimisation methods uses the objective function, while the techniques based on iteration to necessary and sufficient conditions use the equalisation of delay (rather than marginal delay) as criteria for optimisation of the function indirectly. Mason calculated the performance of several networks for both optimal and equalising routing policies and found the two schemes produced very similar results at convergence. In conclusion the learning automata scheme, although converging to equalise delays rather than marginal delays, produces a performance very close to that of an optimal strategy, without recourse to any centralised component.

5.3.3 Congestion and Flow Control Techniques

Both circuit and packet switched networks can be considered as a set of distributed resources; transmission links, nodal buffers and processors, all of finite capacity. However the way in which these resources are managed is quite different. In a circuit switched network resources are allocated for the duration of a call. For the case of an overloaded network this method of allocation results in an increase in the probability that a customer will be unable to make a connection over the network. However, once a connection is established the customer will not suffer any further reduction in performance. In a packet switched network resources are shared dynamically between those customers who request their use. When a packet switched network is overloaded customers are not blocked automatically and the resources of the network can be depleted leading to a degradation in the performance seen by all customers using the network. The point at which this takes place marks the onset of congestion, typified by an increase in the delay of traffic through the network and a decline in throughput. The typical response curves of delay and throughput vs. load are shown in figure 5.7 for such an unprotected network.

The network routing algorithm is an important parameter in determining the onset of congestion, generally accepted to be the point at which the maximum occurs in the plot of throughput vs. load. A good algorithm will make good use of available resources, however no matter how efficiently the routing algorithm manages the network, the resources represent a finite quantity. If the traffic level increases it will inevitably reach a point, no matter what scheme is employed, where these resources cannot accommodate the demands made upon them, resulting in congestion. To avoid this problem flow control and congestion avoidance techniques have been developed to limit access to the network as congestion approaches in an effort to prevent the exhaustion of resources. A number of excellent review papers have been published which discuss the various techniques[85–88]. In this section some of the more promising techniques and conclusions generated from these reviews will be discussed and the models they refer to expanded from their original sources to cover some of the analytic work that has been developed to assess the impact of introducing flow control. The remainder of the section will then concentrate on the work done on the interaction between routing algorithms and flow control procedures.

The various techniques can be classified according to a number of criteria, but for this review the Gerla and Kleinrock system is adopted[85]. The congestion control technique is identified by reference to the position in the network architecture at which it operates. The different areas of interest are labelled on a sketch of the network architecture in figure 5.8. Four different areas of interest are identified within the hierarchical structure; hop level, network level, network access, and transport level. A network may use a combination of some or all of these mechanisms to provide congestion but they will be considered in isolation here to clarify their effects.

Beginning at the bottom of the hierarchy, flow control may be exercised between each of the nodes in the network at the Data Link layer of the protocol structure. Control at this level prevents local congestion by management of buffer resources in individual nodes. Irland[89] showed that by selection of appropriate thresholds for the number of buffers that could be allocated to a single output queue at any one time, busy paths can be prevented form 'hogging' buffer capacity to the detriment of other customers. A two dimensional Markov model was developed to calculate the loss probability of traffic under a number of suggested schemes including the optimum buffer allocation, for both balanced and unbalanced traffic patterns. The analytic results suggested that the performance produced by the optimal strategy for each traffic pattern could be approximated by the the adoption of a simple heuristic scheme. The scheme allocated a threshold of $B/\sqrt{N}$ buffers to each output queue where B is the total buffer capacity and there are N output queues.

An alternative approach, rather than limit the allocation of buffers to the output queues, is the use of virtual circuits between adjacent nodes for each source. Notice this requires a virtual circuit implementation at the network level to provide the path for the traffic between the source and destination nodes. The maximum allocation of buffers given to an individual end–to–end virtual circuit at each node in the network is defined by the window size of the virtual circuit over the link between each node and the former node in the path. This technique was studied by Pennotti and Schwartz[90] who developed an analytic model of a single logical link in the network based on the queue in figure 5.9. Two traffic types are identified contributing to the input stream, the traffic associated with the logical link under analysis,λ_1, and the traffic from external logical links entering and leaving each queue, λ_2. The logical link was modelled by a series of finite queues and servers of this type, the buffer sizes given by the size of the window at each stage of the path. The effect of external traffic is introduced by the reduction of the servers capacity by a factor equal to the external traffic arrival rate. Congestion caused by the presence of the link is then measured as the increase in queueing time experienced by the external users due to the prescence of the logical link. The resulting model is reproduced in figure 5.10. The service rate of each stage has been cut to $\mu_n - \lambda_n$ where μ_n and λ_n are the total service rate capability and external traffic intensity of node n respectively. The queueing system can be solved iteratively by considering each queue as an independent finite $M/M/1$ queue of size N_n whose server has a probability of being blocked given by the probability that the succeeding stage is full. If the final server's probability of being blocked is taken as zero then the probability that the buffers at each stage in the chain upstream are full can be calculated. All that is required for this calculation is an assumed throughput λ and the probability that the next stage is full. A new estimate of the throughput can be calculated from the final stage where $\lambda = \lambda_0 p_1$ where λ_0 is the link arrival rate and p_1 is the probability the first queue is full. This value can then be used in the next iteration until the difference between successive iterations is less than some threshold value. Steady state results suggest that such a scheme provides similar protection levels for the external users as an equivalent end–to–end scheme which will be discussed next. However the question of how long such a scheme would take to throttle the source by backward pressure is not investigated.

At the next level in the network hierarchy, flow control can be introduced between the source and destination nodes of the network. This is done by the implementation of an end–to–end window protocol which places a maximum on the number of unacknowledged packets in transit over any virtual circuit at any time. Primarily this prevents the destination being flooded by the source if the ability to accept information arriving is insufficient to cope with the demand, as for example in the case of a slow local loop to the customer. In addition the window protocols also limit global congestion by limiting the number of packets traversing the network at any one time. During times of high network throughput the time for a packet to traverse the network will increase and this will automatically reduce the ability for each source to introduce new packets as the number of acknowledged packets will be equal to the virtual circuits window size for increasing amounts of time. An analytic model was developed by Pennotti and Schwartz for an end–to–end protocol of this type based on the queue in figure 5.9 and using the same concepts of external traffic interference and congestion as previously discussed for the case of hop level flow control. The model formed a closed queueing system of the type shown in figure 5.11 in which N customers cycle around the loop, where N is the window size of the virtual circuit and λ is the arrival rate of traffic for introduction to the network. The balance equations can be solved using a product form solution and used to determine the congestion produced by circuits of varying sizes under various external load conditions. In a later paper by Schwartz[64] the problem was simplified by considering only the end–to–end performance of a homogeneous link. A logical link of M identical queues with service rate μ can then be replaced by its 'Norton equivalent' which is simply a single queue with a state dependent service rate $\mu(n, M)$ where there are n packets in the queue and

$$\mu(n, M) = \frac{n}{n + (M - 1)}\mu.$$

Using the concept of 'power' which is the ratio of throughput to delay, Schwartz showed that for this model, the optimum window size, generating maximum power at a load $\lambda = \mu$, is equal to the number of hops in the link, i.e. M. Interestingly for $\lambda \rightarrow \infty$ the optimum window size was found to be only $M - 1$ and neither of these values proved to be critical for effective flow control. This last point has important repercussions for adaptive and dynamic strategies where effective flow control can be applied without detailed knowledge of path lengths for the virtual circuits through the network.

The next class of congestion control, network access protocols, takes place at the entry point of the network. Although this appears similar to the end–to–end procedures discussed earlier, schemes that fall into this category determine the accessibility of traffic based on some measure of congestion in the network *as a whole.* End–to–end schemes limit flows from specific sources to congested destinations, aiding network congestion as a consequence rather than directly. Two main schemes have been identified in the literature, one global and the other based on purely local congestion levels at each origin node. The global scheme, known as isarithmic control, was developed at the National Physics laboratory in the United Kingdom[91,92]. The scheme introduces a number of 'permits', dispersed throughout the network. To transmit a packet a source node must first obtain one of these permits. The permit is then carried with the packet and released at the destination node, where it can be reused for traffic from that destination. It was also recognised that permits must be prevented from collecting at a few monopolising nodes, starving the rest of the network of access permission. A maximum on the number of permits a node can possess at any one time was proposed. Additional permits arriving at a destination would then be re–distributed over the network. Clearly this scheme imposes a maximum on the number of packets that can be in transit over the network at any one time, however no satisfactory scheduling mechanism has been developed for the dispersion of packets round the network and additional problems that have been highlighted, such as the maintenance of permit integrity, have prevented this scheme evolving from the simulation stage. The second congestion control technique uses the occupancy of the buffers at each node as a measure of traffic congestion. The input of traffic into the node is then regulated as a function of this local congestion. The node recognises two types of traffic, transit traffic from other nodes in the network, and new traffic from local customers trying to gain access to the network. The model of the queue arising from such a division is shown in figure 5.12. Both new and transit traffic compete for the finite buffer space and are blocked with probabilities P_I and P_T respectively. In addition a portion of the transit traffic, P_X leaves the network at the node, having reached its destination. Considering a symmetrical, homogeneous network made up of queues of this form the throughput γ is simply equal to the transit traffic accepted by the queue, given by

$$\gamma = \lambda_T(1 - P_T)$$

For the conservation of traffic as many packets must leave the network as arrive. Equally this must also be true for each of the identical nodes in the network. Therefore

$$\gamma = \lambda_T(1 - P_T) = \gamma_I + \gamma_T(1 - P_T)$$

from which the simple relationship $\gamma_T = \gamma_I/P_T$ can be derived connecting the transit and input throughput rates. If we first assume that no flow control is imposed and both input traffic and transit traffic are treated equally, the model further simplifies to a single offered traffic term λ and a throughput $\lambda(1 - P_B)$ where P_B is the probability that the buffer is full. Plotting this function against load produces the classic throughput function sketched in figure 5.7. Lam and Reiser[93] proposed a strategy to eliminate this downturn in performance by the introduction of the input buffer limit scheme. The scheme limits the number of buffers input traffic can occupy in a queue of buffer capacity N_T, to a maximum value N_I, where $0 < N_I < N_T$. No limit is placed on the number of transit packets in the buffer at any one time and they are allowed to fill all empty buffers whether input traffic is limited or not.

This scheme can be analysed by the formulation of a 2-D birth–death process[94] defining the balance equations satisfied by the probability distribution $p(n_I, n_T)$, where n_I and n_T represent the number of buffers occupied by input traffic and transit traffic respectively. The equations use state–dependent service rate for both transit and input packets to differentiable between the relative amounts of time they each spend being serviced and from this a product form solution can be derived which shows that as N_I is reduced from N_T a downturn in performance is prevented at some value. Tighter control produced the same effect with a degradation in throughput across the range of applied load. Lam and Reiser used a more complex analysis and showed that N_I should be selected to satisfy the design criteria

$$N_I/N < 1/\overline{n} + 1$$

where $\overline{n}$ is the average hop count. This result agrees with the observations of the simpler model. Saad and Schwartz[95] later showed that the ratio of throughput to end–to–end delay could be improved by implementing a different scheme known as additional buffer allocation similar to the concept of trunk reservation in circuit switched networks. In this scheme new input traffic and transit traffic are both accepted up to the point where there are fewer than N_I buffers free where $0 \leq N_I \leq N_T$. While the buffer is in a state where the number free buffers, n, are less than N_I, input traffic is rejected and only transit traffic is allowed access to the node. Thus input traffic is refused entry to the network when

$$0 \leq n_I + n_T \leq N_I.$$

The final layer in the hierarchy of congestion control techniques occurs at the transport layer. This is usually based on a window type mechanism similar to those described for end–to–end control at the network layer. These flow control procedures generally control the rate at which two applications communicate, providing pacing between the sending and receiving customers. Once again they have a secondary effect of controlling input to the network as delays increase by imposing a maximum window size on unacknowledged packets. However the effect at this level is much less pronounced because of the remote position of the control and this secondary function is not usually significant if sufficient control is exercised lower in the hierarchy.

In this section flow control schemes have been outlined and analytic models have been used to formulate performance criteria based on single logical links or local measurements at individual nodes. In the previous section routing algorithms were discussed, categorised and once again models formulated to analyse their behaviour. In each case the influence of one upon the other was not included in the analysis. Routing is discussed in terms of well defined traffic sources between origin and destination nodes, while flow control was analysed using fixed paths and well defined external traffic interfering with the progress of a sources traffic. In a network implementation the routing strategy and flow control interact strongly to define the performance of the network[96]. The routing strategy defines link utilisation from the allocation of traffic across the network. From this buffer occupancy and end–to–end delays result. The flow control algorithms will use these parameters to determine the level of traffic from each source that is allowed entry to the network. This will affect the amount of traffic over each path which will in turn affect the selection of paths and allocation of traffic assigned to them using the adaptive strategy. The importance of this relationship was shown by Rudin, Chou, Gerla[56,77,79] and others who carried out simulations which clearly showed the importance of selecting the correct algorithm and the use of suitable flow control.

More recently there have been a number of attempts to extend the mathematical modelling of routing algorithms to include flow control in the formulation of the objective function. Gerla et al[97] examined the problem of a network whose traffic sources were each throttled by end–to–end windows of fixed size. An iteration procedure using the flow deviation method is used to derive the optimum routing strategy for the minimisation of the average network delay. The complexity of solving the problem in a closed network, necessary to introduce the effects of flow control, is approximated by dividing each iteration into two stages. First the routing tables of the network are frozen and the throughput and end–to–end delay of each session are evaluated using mean value analysis[98], an approximate recursive technique which is computationally feasible for realistic networks in comparison with the exact product form solution which can only be realistically implemented for trivial networks. The end–to–end windows are then removed leaving the open network and each session is replaced with a traffic source equal to the calculated throughput. This is another approximation as the magnitude of the throughput has been retained but the window function limiting the number of packets on the network from one source at any one time has been removed. The flow deviation technique is then used to calculate the new routing pattern, and this is then used in the next iteration until no significant change is observed at each iteration. The results are instructive and can be used to propose a limit on the number of sessions that can operate with a given window size without causing congestion on the network, if they were each to use the full capacity of the window assigned to them.

Gallagher and Golestanni[99] proposed a more dynamic approach to the problem and introduced the concept of dynamic window sizes based on congestion levels as a natural partner to Gallaghers distributed routing algorithm. The objective function for the joint routing and flow control strategy is a summation of the routing cost function used in the optimisation of open networks and a second term representing the cost of rejecting traffic by the introduction of flow control. The objective function $J(\overline{f},\overline{r})$, where f and r represent the vectors of flows around the network and into the network respectively, is therefore given by

$$J(\overline{f},\overline{r}) = \sum_{\substack{all\ links \\ l}} D_l(f_l) + \sum_{\substack{all\ traffic \\ pairs\ (i,j)}} E_{ij}(r_{ij})$$

subject to usual the conservation and capacity constraints and $0 \leq r_{i,j} \leq r^d_{i,j}$ where $r^d_{i,j}$ is the input rate desired by the customer. Applying the Kuhn-Tucker theorem to the objective function once again produces the result that traffic flows only over paths of equal and minimum incremental cost. In addition the solution produces the result that $-E'_{ij}(r_{ij})$, termed the incremental gain for the allocation of traffic r_{ij}, assumes a value as close to the incremental cost of traffic r_{ij}, $D'_{ij}(r_{ij})$, as possible within the limits of the constraints on r_{ij} and is independent of r^d_{ij}, within these limits. To develop an algorithm to solve this problem it is then reformulated as a quasi–static routing problem by assigning the traffic rejected by the flow control to a fictional link and assigning a cost function to this link $D_{l'}(f_{l'})$ where

$$D_{l'}(f_{l'}) = E_{ij}(r^d_{ij} - f_{l'})$$

$D_{l'}(f_{l'})$ is a convex function in $f_{l'}$ compared to $E_{ij}(r_{ij})$ which is a strictly decreasing function of r_{ij}. This reduces the cost function to the form $\sum_l D_l(f_l)$ over all links real and ficticious. The paper goes on to prove that window size can be expressed as a function of the session rates in the network and that from this changes in the size of a session window lead to well defined changes in the rates of sessions in the network. By formulation of J in terms of session rates and routing it demonstrates that the effect of such incremental changes on the objective function can be used in a distributed algorithm to reduce the value of that objective function based on local measurements of the marginal cost of the session. Thaker and Cain[100] covered much of the same ground in their thorough paper on the interaction of routing and flow control procedures. The paper presents results for both simple and bifurcated shortest path routing and compares them with an optimum routing policy with and without window flow control. The effect of flow control was evaluated by the use of Little's formulae to calculate the allowed throughput for each session based on the window size. This simple approximation appears to work well for loads which exceed the allowed throughput. An adaptive window size based on the incremental delay for each session, $\delta D_T/\delta r_{ij}$ is then proposed. A penalty function defining the maximum rate for a session for a given incremental delay is formulated and shown to be of the same form but slightly more flexible than the Gallagher scheme and much more successful than a scheme based a simple marginal delay threshold to turn the traffic on and off. Using an adaptive scheme of this form is shown to have the important attribute of preventing network performance deteriorating as the number of sessions increase, unlike fixed window size schemes. The penalty function is used to produce a formulae for the calculation of the window size for each session based on Littles formulae once more. The window size turns out to be a function of the marginal delay and the end–to–end delay of the session. Finally Thaker and Cain suggest a variation of the scheme where marginal delay is replaced by the simple average delay/hop and results suggest that the performance of this scheme only undergoes a slight degradation when at high loads in comparison to optimal control.

The work outlined in this section allows considerable progress to be made in the analysis of packet switched flow control and routing strategies using analytic methods. However there are many examples where analytic treatment is not applicable or the approximations made insufficient. Many of these problems arise from the assumptions made by the mathematical models to allow tractable problems to be formulated. Alternatively results can be generated by the introduction of simulation techniques to create a more realistic environment for network analysis. In the next section the design of such a packet switched network simulator constructed on the Transputer simulation environment is presented.

5.4 Packet Switched Simulation Model

5.4.1 Layered Nodal Model

The simulation model of the packet switched network node is based on the first three layers of the network protocol architecture outlined in figure 5.5. A physical model was first designed defined the queueing and transmission characteristics of the input and output interfaces and the main data paths between them and the processing areas. Within this physical model the data link and network layers of the layered communication protocol where then designed to control the passage of packets between each of the queueing systems from the source to the destination nodes.

Physical Layer

The major physical components of the network node model are shown in figure 5.13. The node consists of a central area where packets are processed and a number of associated input and output data paths to and from this area. The input paths come from two different sources; transit packets from the incoming links of neighbouring nodes and packets generated from messages originated at that node. A message source generates virtual circuits for transmission across the network which are stored in a group of buffers assigned for local transmission. Packets from these virtual circuits compete both with other and incoming packets for the processors attention on a first–come–first–served (FCFS) basis. The central processing area contains the upper layers of the communication protocols for accepting, processing and redirecting packets from the input paths over the output paths. The output from the processing stage is linked into the output queue for each outgoing link according to its priority. Each queue has a server and a second area of storage beyond the server. When a packet reaches the head of the queue it is held in the server for a time equivalent to its transmission time and then released into the second area. The packet is then stored in the second area for a further period of time equal to the propagation delay over the outgoing transmission link. The packet is then sent to the next node in the path where it is accepted over the appropriate incoming path.

Data Link Layer

The remaining model characteristics depend on the protocols that are chosen for the Data Link and Network layers. The model does not explicitly consider the individual bits that make up packets, rather the protocols are built around the smallest important fundamental building block, which is considered to be the end–to–end acknowledgement packet in the Network layer of a virtual circuit routing policy. The Data Link layer, which provides an error free packet transmission service for these fundamental units of data, between neighbouring nodes, was constructed to be complex enough to cope with corruption of packets over a link to provide a good model for error recovery. However much of the fine detail of real protocols, aimed at unusual conditions or optimization can be ommitted without the loss of a good appreciation of the protocols behaviour.

The Data Link layer chosen is a simplified form of the Go-Back-N strategy outlined in Tanenbaums text on computer networks[101]. The essential features of the strategy include

- acknowledgement packets transmitted to the sending node on successful acceptance of a packet at a receiving node. The acknowledgement packet is sent with the highest priority to minimise delay.
- packets are discarded if there is insufficient buffer capacity for their storage on the output queue selected or if the packet has been corrupted on transmission over the incoming link.
- nack packets, or negative acknowledgements are transmitted at the highest level of priority to the sending node if a packet is discarded. This is the only mechanism used to inform the sending node of a packets failure to be accepted. No timeout scheme is implemented.
- both ack and nack packets are never corrupted by transmission over a link and are considered to be of zero size.
- multiple transmission of unacknowledged packets over each link. This allows the transmitting link to keep sending data packets over the outgoing link without incurring a round trip delay for each packets acknowledgement from the receiving node.
- on reception of an nack packet a copy of the packet that was lost at the receiving node is retransmitted, followed by all the packets that were transmitted following it.
- packets are only accepted in the correct order, i.e. all packets following a lost packet are automatically rejected until the lost packet is resent. The retransmission of all packets following a lost packet preserves the order of transmission across the link. This removes the need for resequencing and simplifies the protocol. The retransmission also introduces a transmission overhead caused by the rejection of correctly transmitted packets

following the lost packet.

Network Layer

The network layer protocol, with its two main components of routing and flow control will vary according to the particular models under analysis. The routing strategy will select an outgoing path for each packet but how the path is selected will depend on the algorithm, type of routing and type of packet. A range of algorithms have been implemented, consisting of

- random routing. The outgoing link is selected at each node from a selection of possible outgoing links which form a subset of the possible outgoing links. The scheme directs the traffic over a number of preferred paths with equal probability of selecting each of the allowed links at each node.
- learning automata. Once again the outgoing link is selected at each node from a subset of the nearest neighbour connections. However in this scheme the probability of selecting each of the outgoing links is modified by the history of the traffic routed through the node. This is accomplished by updating the link probabilities based on the reported delay experienced by packets using each of the links.
- shortest path strategies. Routing strategies using minimum hop and minimum delay techniques are implemented. Minimum hop routing finds the path for each traffic source which minimises the number of intermediate nodes packets have to traverse. Minimum delay routing finds the route to each destination of least value, where the value assigned to each link is the inverse of its transmission capacity.
- bifurcated shortest path strategies and optimal routing.

In addition to the actual algorithm used, both virtual circuit and datagram routing are implemented using each of the strategies. In datagram routing each data packet is routed independently of all the packets and uses the algorithm at each node to direct it to the next node in its path to the destination. No explicit end–to–end acknowledgement takes place, although each packet still generates an acknowledgement packet between each node at the Data Link layer. Virtual circuit routing for a message involves the specific creation of a route to the destination using a special 'Trace' packet. This packet uses the algorithm at each node to traverse the network to the destination. When an end–to–end acknowledgement packet reports the successful creation of such a path, data packets are transmitted to the destination over the path formed by the 'Trace' packet. Each data packet is acknowledged by its own network acknowledgement until the entire message has been successfully sent. Another special packet, a 'Finish' packet is sent down the path to break down the virtual circuit.

The second part of the network layer function, flow control, is implemented with a number of options which can be applied according to the form of the routing technique. The options allow

- a limit on the number of traffic sources from any node at any one time.
- the window size of each traffic source to be defined. This limits the number of unacknowledged packets on the network for each traffic source.
- input buffer limiting and buffer allocation. This allows the number of buffers to be assigned to transit traffic only and the number of buffers to be assigned to any one output link at any one time to be defined.

The first two parameters allow the formulation of virtual circuit routing algorithms using end–to–end acknowledgements to control input to the network. Datagram strategies do not generate end–to–end acknowledgements and so the network flow control has to be controlled via buffer availability, although this form of flow control can also be used for virtual circuit routing strategies as well.

5.4.2 Model Implementation

The implementation of code for the packet switched network simulation uses the architecture defined in chapter 2 The nodal process representing each node in the network is reproduced in figure 5.14. Each nodal process is composed of six major procedures which form four parallel processes, consisting of three satellite processes which compete for the attention of the fourth process. This fourth process controls access to the data structure. The sequence in which a call travels through these processes from its generation to its arrival at its destination is shown in figure 5.15. Initially a message is generated which subsequently generates individual packets. Once a packet has been admitted to the network, a route is selected and the packet inserted on the appropriate output queue where it stays until it is transmitted to the neighbouring node selected by the routing strategy. If a packet is still in transit it goes through the same sequence of processes at the new node. Alternatively if the destination has been reached it will either generates an end–to-end acknowledgement packet if part of a virtual circuit strategy or simply terminate if the network routing strategy is of a datagram type. The protocol layer at which each major operation takes place is shown along side each stage with the title of the operation.

Figures 5.16 to 5.20 analyse each of the processes more closely with respect to the major operations outlined in figure 5.15. Figure 5.16 gives details of the peripheral routines corresponding to the implementation flow control within the network layer and the generation of traffic from the customers and its subsequent introduction into the network. Figures 5.16(a)-(c) describe the process Generate, and routines in the processes Call Origin and Sink which control the end–to-end behaviour of a message send over the network by a virtual circuit. The Generate process, outlined in figure 5.16(a), is initialised by calculation of the first message arrival for each destination node from each node from which traffic originates. The process then introduces the messages in the form of virtual circuits at appropriate times in the simulation. New virtual circuits are subsequently generated according to arrival statistics defined in the simulation. As each new virtual circuit is generated its progress over the network is controlled by the Call Origin process at the originating node and additonal code at the destination node to form a simple communication protocol. The basic form of the two halves of the protocol are presented in figures 5.16(b) and 5.16(c). Together they operate a scheme similar to that outlined earlier in the discussion on network layer protocols. Each V.C. initially transmits a 'Trace' packet. On successful acknowledgement of this packet from the destination node, data packets are transmitted across the network. When all the data packets have been successfully acknowledged a 'Finish' packet is transmitted to terminate the connection. At the destination node, code implementing the other half of the peer protocol receives a 'Trace' packet and generates an end–to–end acknowledgement packet. Subsequent data packets are acknowledged in a similar fashion until the reception of a 'Finish' packet which terminates the connection. When the model is used with datagram routing, each of the virtual circuits become a permanent session between the origin and destination nodes. End–to–end acknowledgements are not generated and so these initialisation procedures are not required. In their place sessions are initially defined prior to the beginning of a simulation and these sessions generate packets continuously as the simulation progresses.

Figure 5.17 shows the two halves of the window flow control mechanism used for virtual circuit studies which make up the protocol layer 3(ii) labelled in figure 5.15. Figure 5.17(a) shows the decision process made on the arrival of each packet of a V.C. If a Trace packet arrives a new window is generated, otherwise the packet is only accepted if there is sufficient room in the window assigned to its parent message. Dropped packets are not re–attempted and are reported in the results generated by the simulation in the form of the percentage of traffic unable to gain entry to the network because of flow control. Figure 5.17(b) simply removes the packet entry from the end–to–end window on successful reception of a network acknowledgement packet. Figure 5.18 completes the implementation of layer 3 of the protocol stack by defining the procedures for both virtual circuit routing (a) and the much simpler datagram technique (b). Again data packets which are unable to continue because of insufficient buffer capacity are dropped. If this occurs at the originating node these packets will be added to those which form the statistics reporting on entry to the network. Otherwise the protocol at the Data Link layer will schedule retransmission of the dropped packets

Figure 5.19 shows the implementation of the Go-Back-N strategy used in the Data Link protocol layer. A new packet, when it is added to a queue generates an local acknowledgement packet if it is within the network. The packet then waits until it reaches the head of the queue. It is held for its combined transmission and propagation delay and transmitted over to the next node. If it is successful a local acknowledgement packet is returned and the packet can be deleted. If the packet is not accepted a 'nack' is transmitted back to the node, which causes the pointer defining the packet at the head of the transmission queue to be re–defined. The pointer is reset to transmit the rejected packet followed by all those which followed it before continuing with the transmission of new packets. Figure 5.20 shows the physical layer processes Input (a) and Output (b) which act as multiplexors and demultiplexors and the Data Link layer operations of the Sink process. The code sorts packets according to their type and position in the network for appropriate action.

A copy of the source code developed to implement this model and used to carry out the subsequent simulation workis included on a diskette accompanying this thesis. The code was written in OCCAM using an IBM PC–AT equipped with a D700C Transputer Development System and a B004 Transputer plug–in card, both from INMOS Ltd. Instructions for recovering the source code from this diskette are detailed in Appendix C.

5.4.3 Implementation of Learning Automata Algorithms

As discussed previously the learning automata algorithms use the delay experienced by traffic routed over an outgoing link to a destination in the calculation of the probability distribution for the selection of those outgoing links. This section outlines the way in which nodes in the network could acquire this information under both Virtual Circuit and Datagram based network protocols.

In establishing a Virtual Circuit through a network an initial packet is first dispatched to determine a route to the destination. For a learning automata routing scheme the next link is selected stochastically according to the probability distribution over the set of outgoing links. When the packet arrives at the destination its arrival is confirmed by the creation of an end–to–end acknowledge packet which is sent over the return path. This interchange of packets which serves to establish a route for the data packets to follow is also sufficient for the implementation of the learning automata algorithms. Each time the initial packet arrives at a node the time of arrival is recorded and the call forwarded as described above. When the end–to-end acknowledgement is generated, it is also time–stamped. When this packet arrives a node in the chain, the delay experienced by the initial packet from this node to the destination is simply the difference between the time–stamp carried by the acknowledgement packet and the recorded time of the initial packets arrival.

Datagram strategies route each packet individually and do not include overhead packets such as the initial or end–to–end acknowledgement packets included in Virtual Circuits. Such features are often implemented at higher layers in the protocol stack for flow control and packet re–assembly functions. However each item is still treated individually for the purposes of routing at the Network layer and this is consequently of little use. To construct a learning automata based routing algorithm it is therefore necessary for each node to communicate packet delays using the local acknowledgement packets generated by the data link protocol layer for each packet arriving at a node. In addition, because these communications are local each node must also maintain an array of average measured delay to each destination node over each outgoing link as well as the minimum average delay at any time, d_{min}, for each destination. To explain how the algorithm works consider a packet arriving at a node i. Its arrival time is recorded, an outgoing link selected and the packet placed on the correct output queue. When the packet is safely received by the next node j, the packet is processed, placed on the next output queue and an acknowledgement packet generated and sent back to node i at high priority. The acknowledgement packet carries the average measured delay from node j to the destination node over the outgoing link selected. When the acknowledgement packet is received by node i this average value is added to the local delay, defined as the time between the data packets arrival at node i and the acknowledgement packets arrival, to give a total figure for the packet delay. In addition the total delay is used to update the average measured delay to the destination over the outgoing link using a weighting formulae of the sort defined in eqn(5.4).

The total delay form each node is then normalised and used to calculate a suitable value for updating the probability distribution over the set of outgoing links as previously described.

5.5 Summary

In this chapter the major operations and performance criteria of a packet switched network have been presented with particular reference to those areas associated with routing and flow control. The current state of analytic modelling has been detailed in both these areas and a simulation model presented which will enable the comparison of predicted and measured performance for a range of different policies. In the next chapter examples of some typical industrial strategies and lower bound calculations are compared with network preformance using learning automata based routing strategies both with and without flow control. A variety of different network configurations are studied to highlight some of the properties of traffic allocation, flow control and the interaction between these functions in packet switched networks.

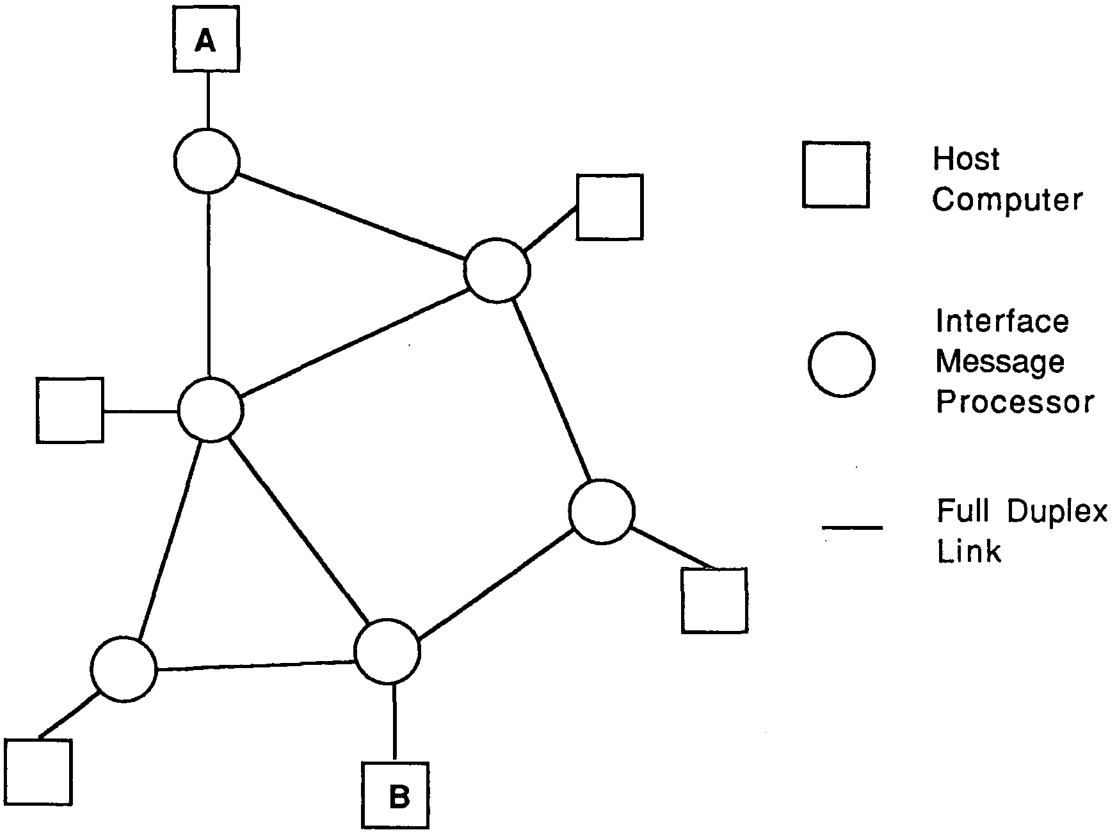


Figure 5.1: Wide Area Communication Network

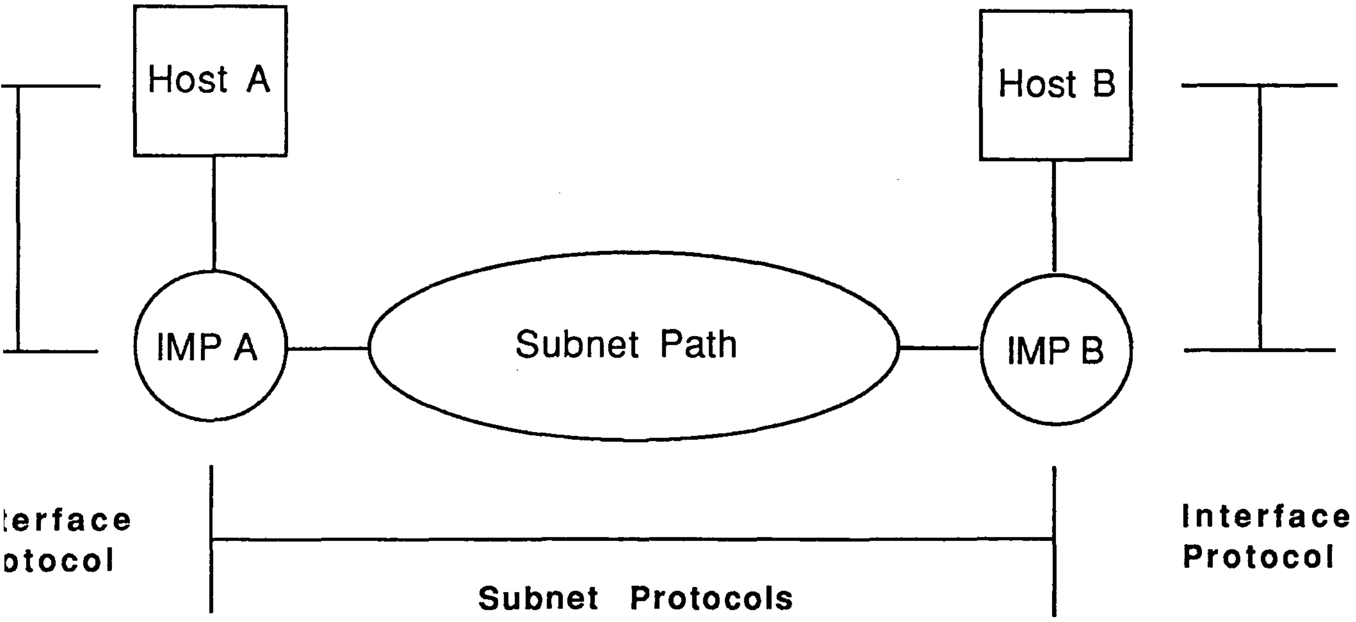
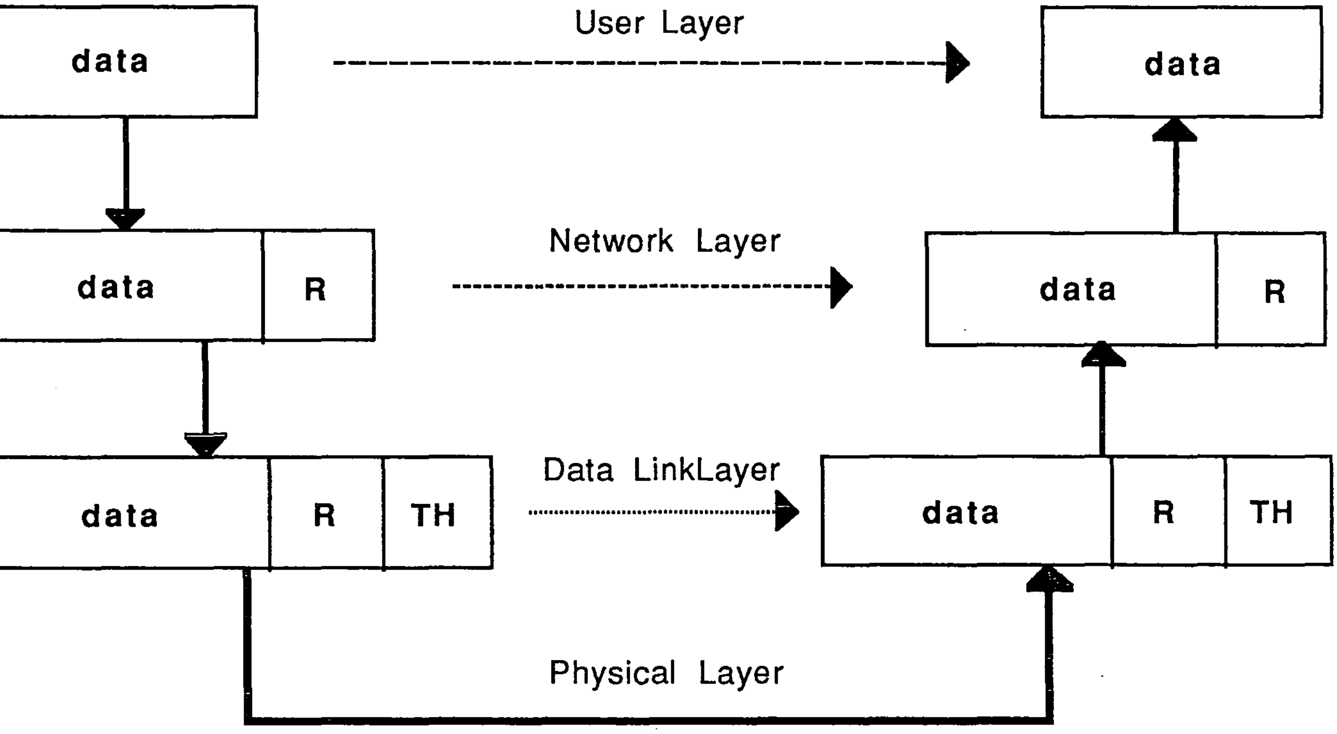


Figure 5.2: Stages In Communcation Between Hosts



R - Routing and flow control information

TH - Transmission header for error detection, status information and link flow control

Figure 5.3 : Packet Formation

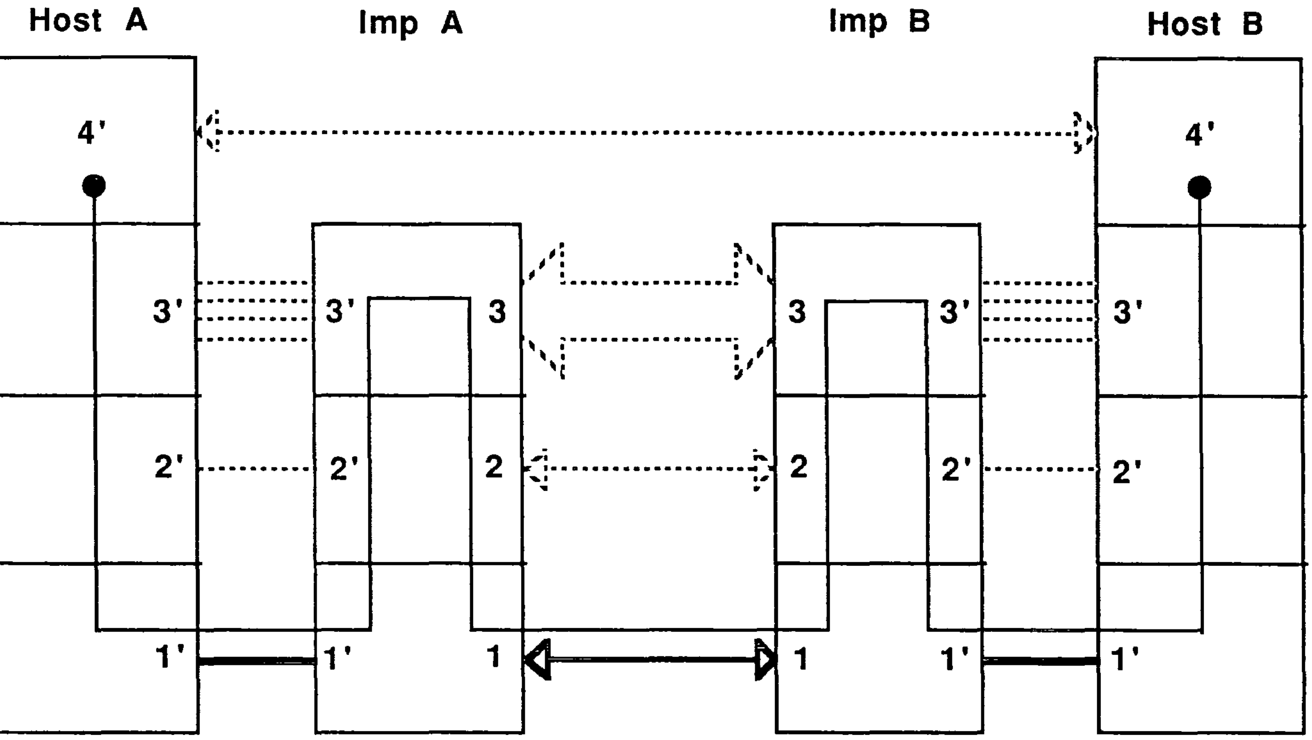


Figure 5.4 : Layered Architecture
Of A Communications Network

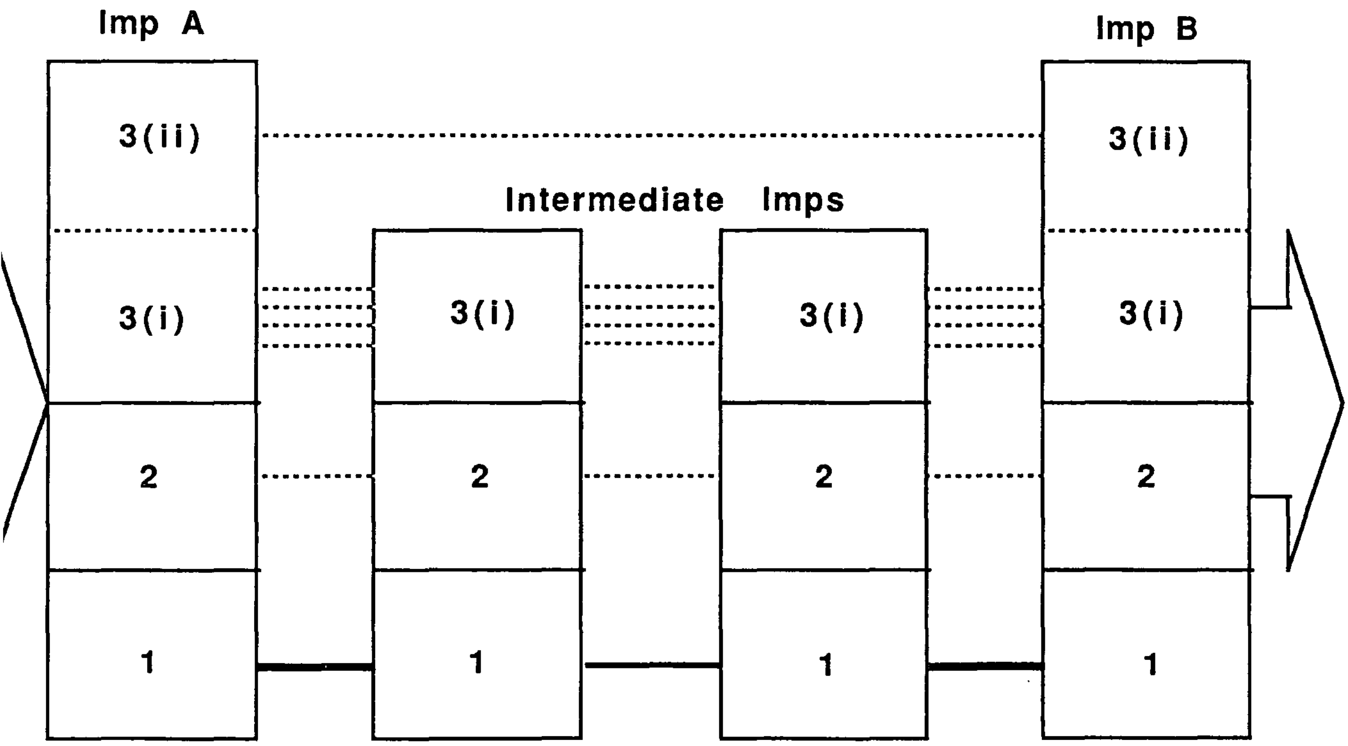


Figure 5.5 : Network Architecture

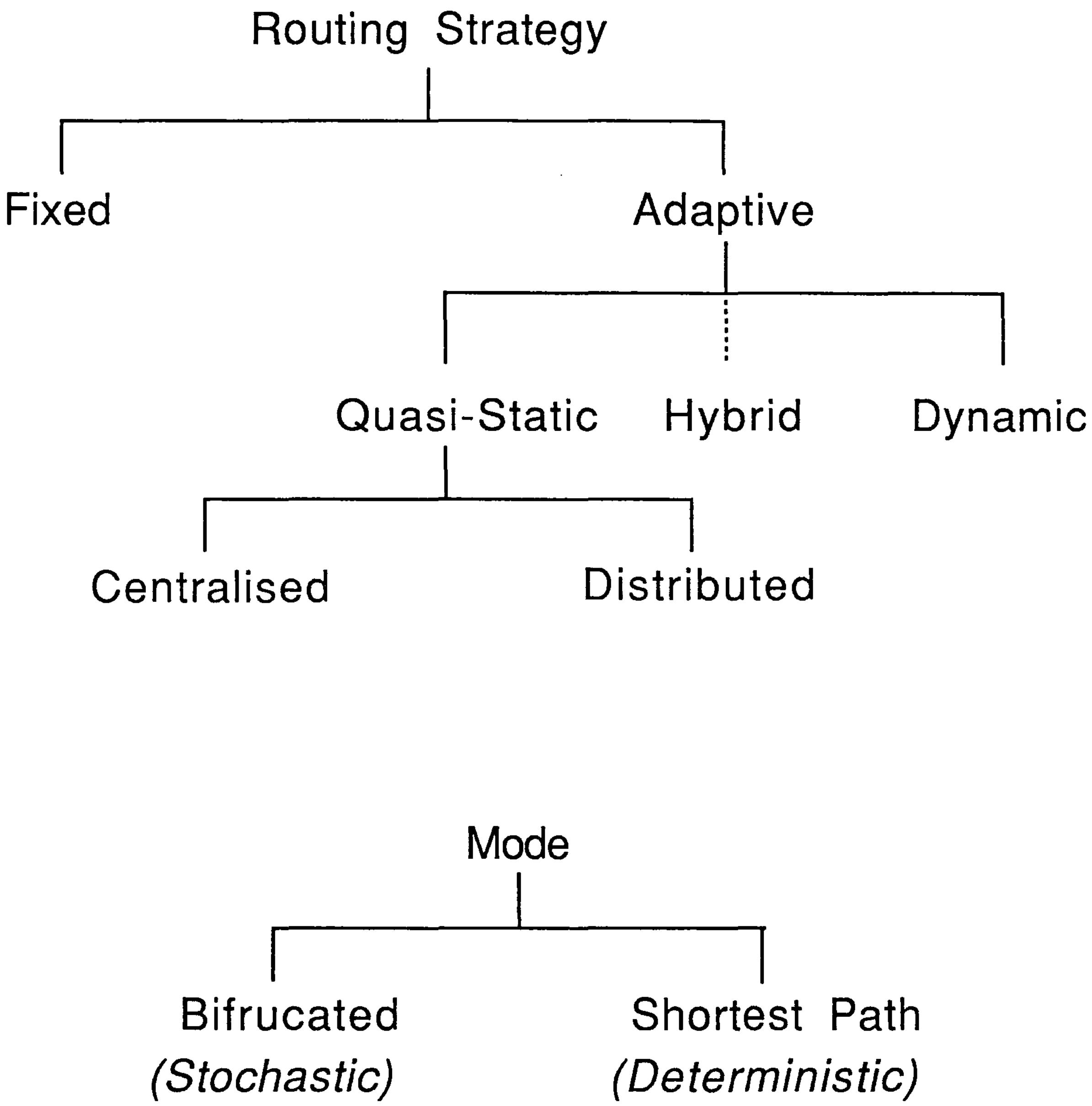


Figure 5.6 Classification of routing strategies

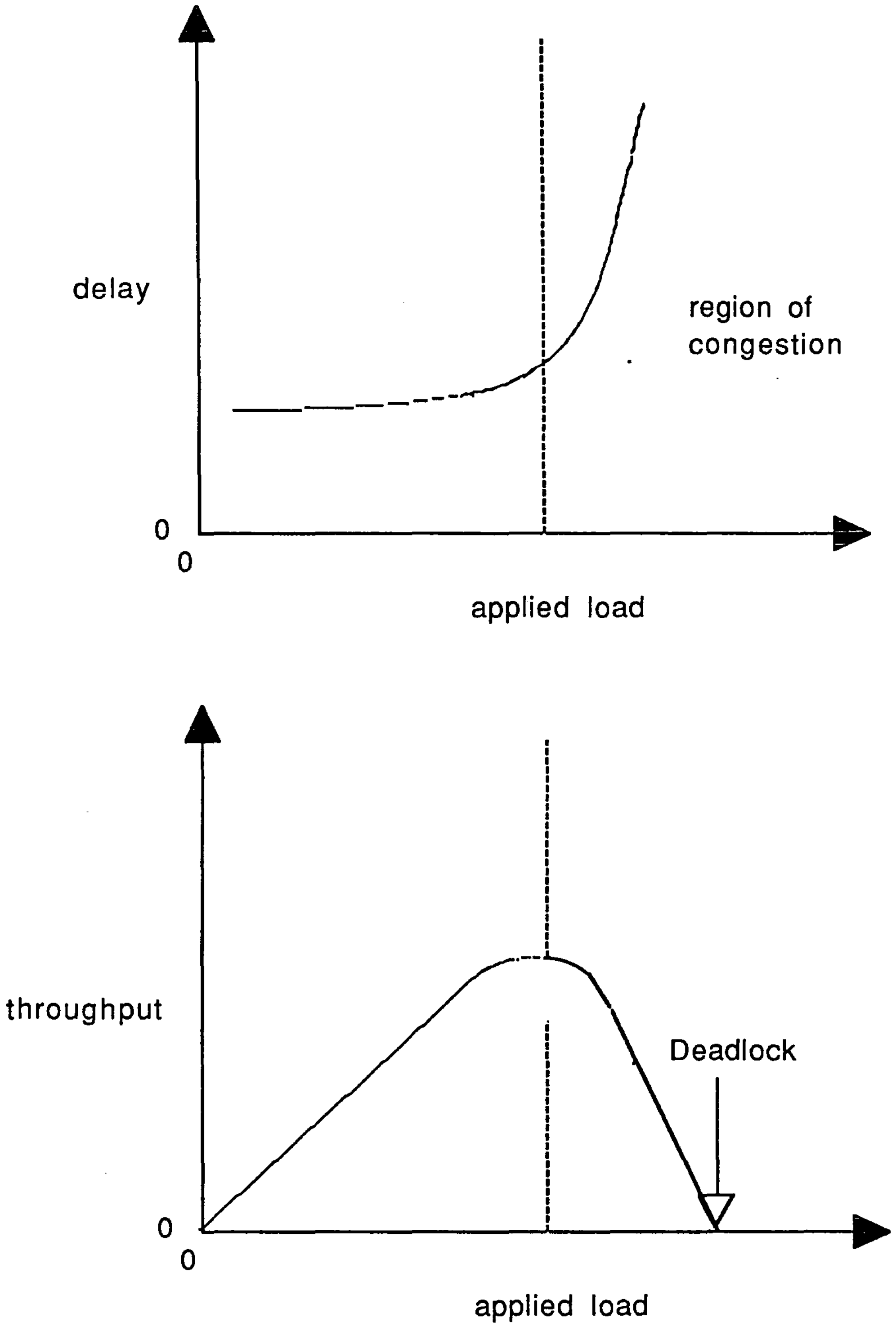


Figure 5.7 Performance curves with insufficient flow control

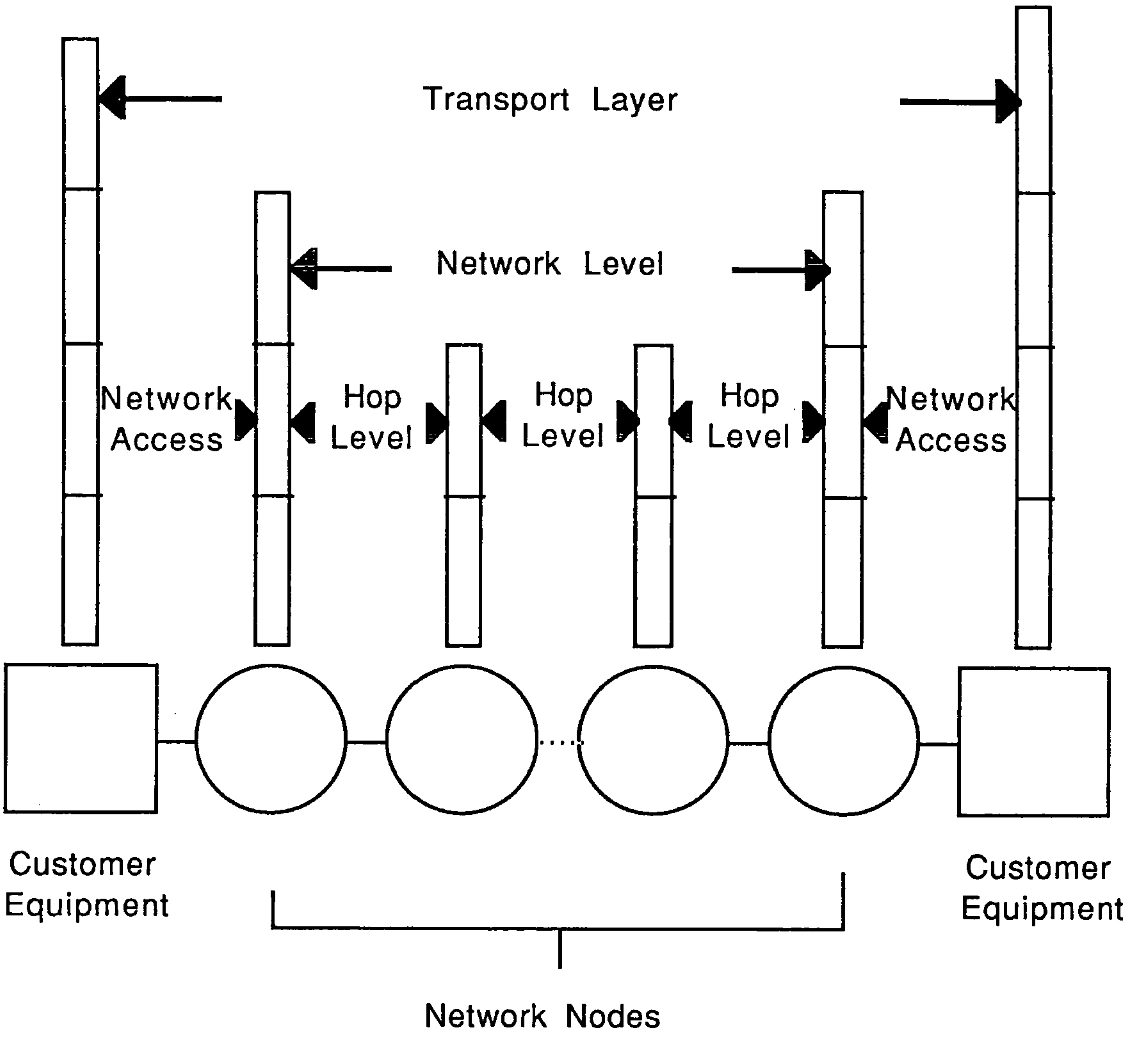


Figure 5.8: Hierarchy of Flow Control

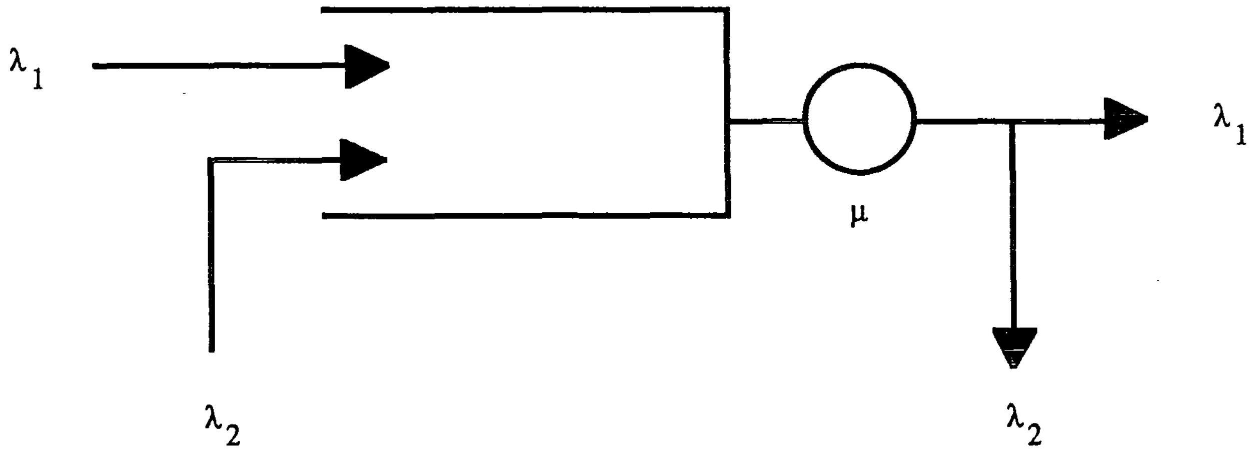


Figure 5.9: Queueing model of a single node in a logical link

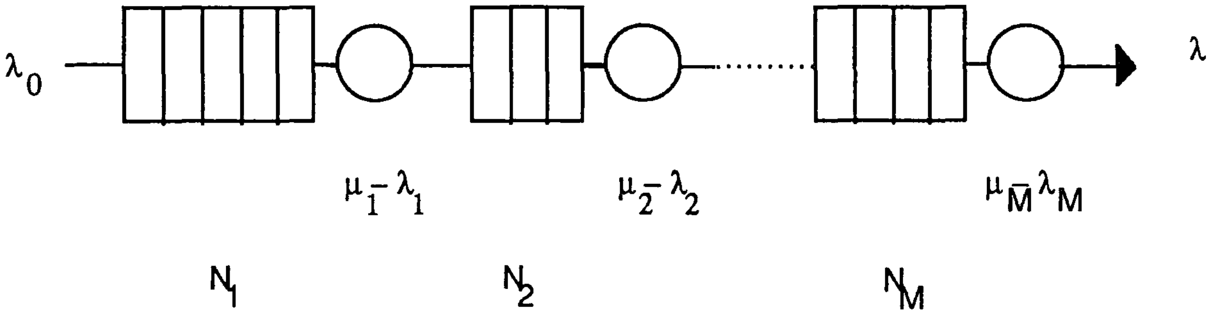


Figure 5.10 M Node logical link model with hop level window flow control

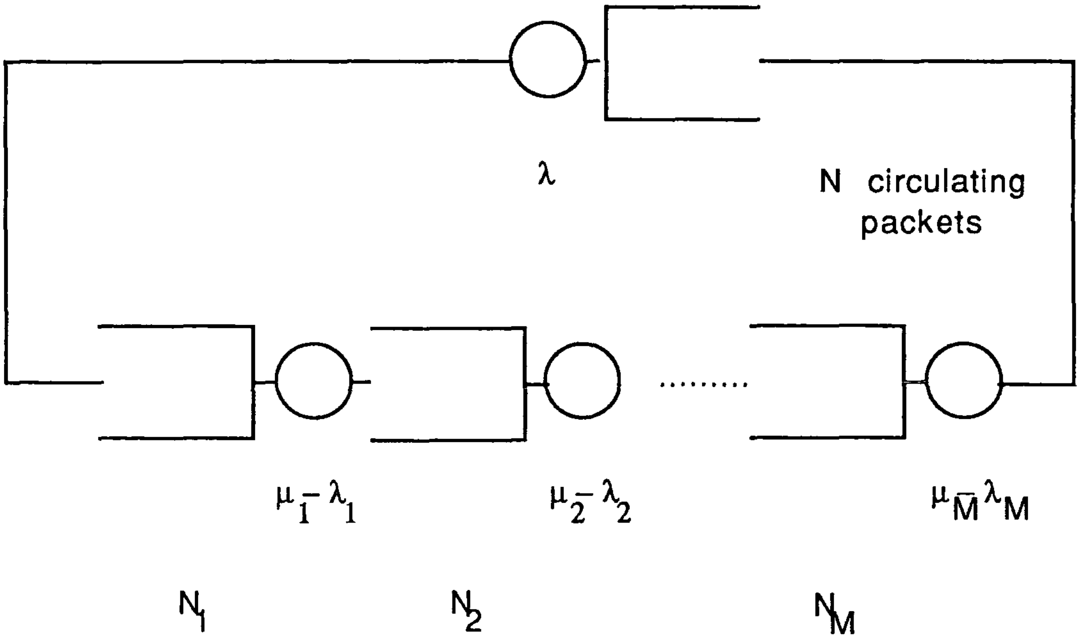


Figure 5.11: Closed Queuing Model of End-to-End Window

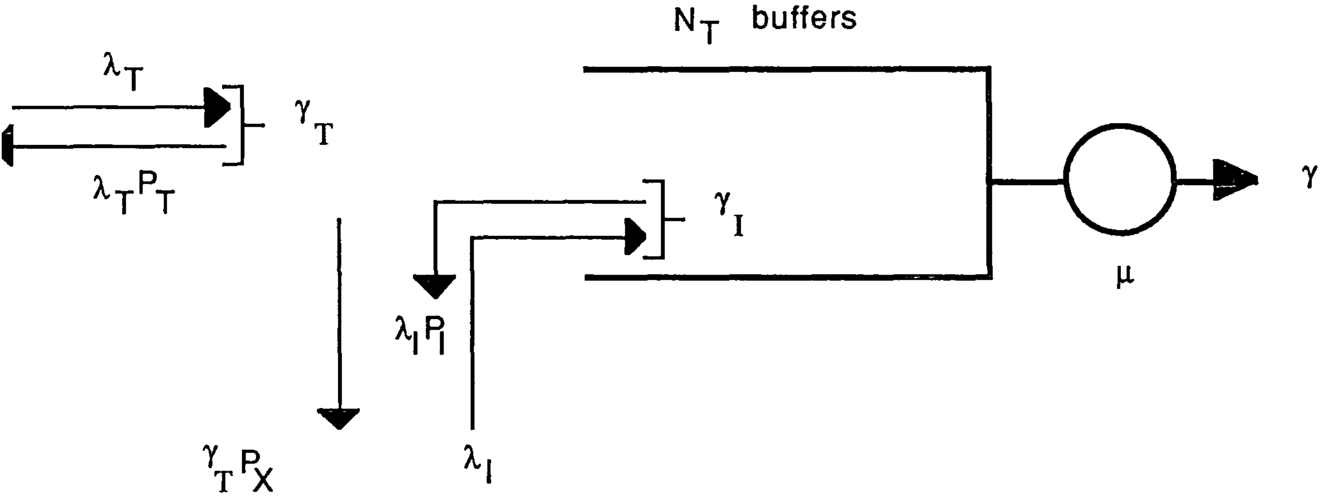


Figure 5.12: Queueing model of single node for buffer analysis

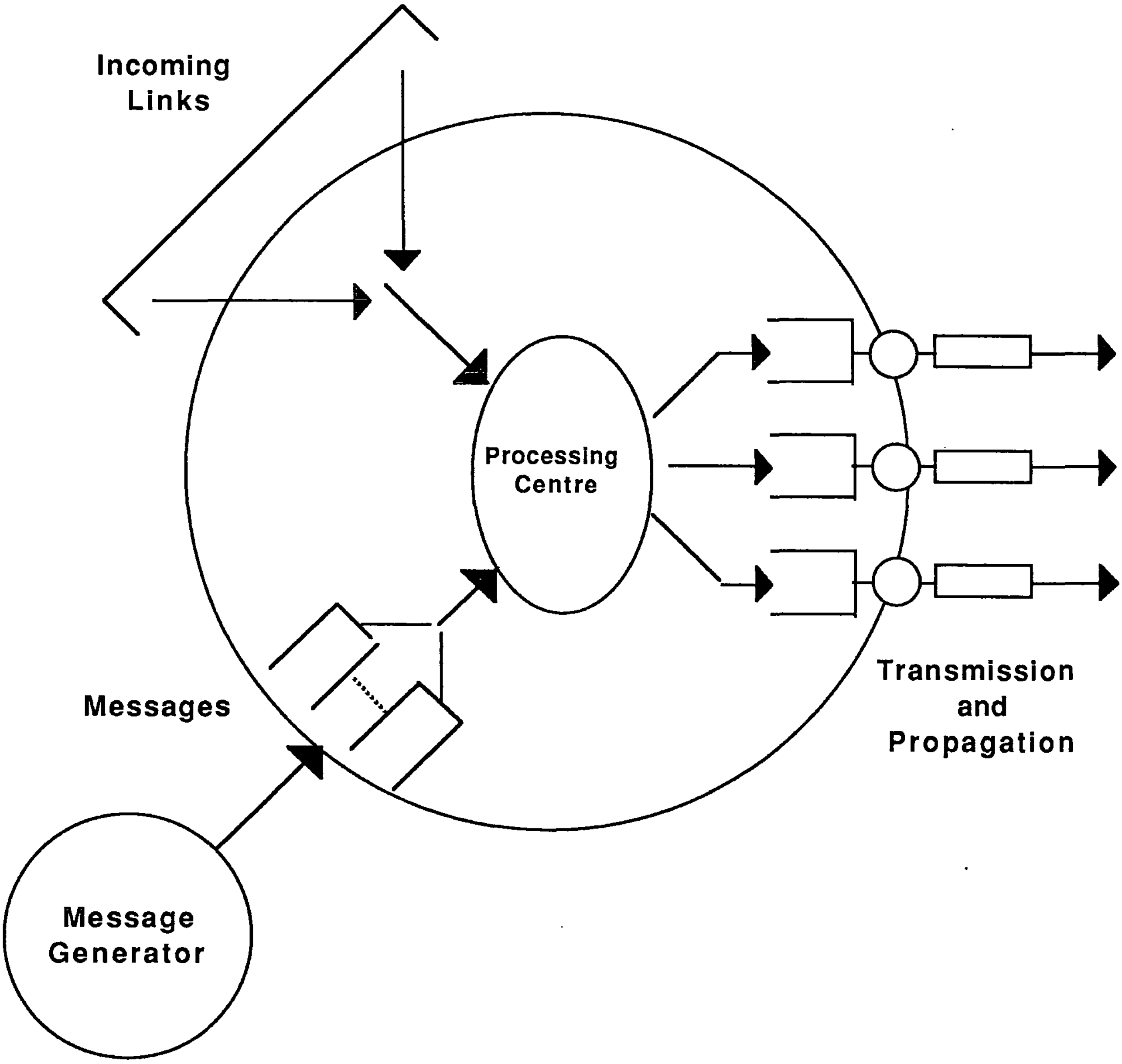


Figure 5.13: Physical Nodal Model

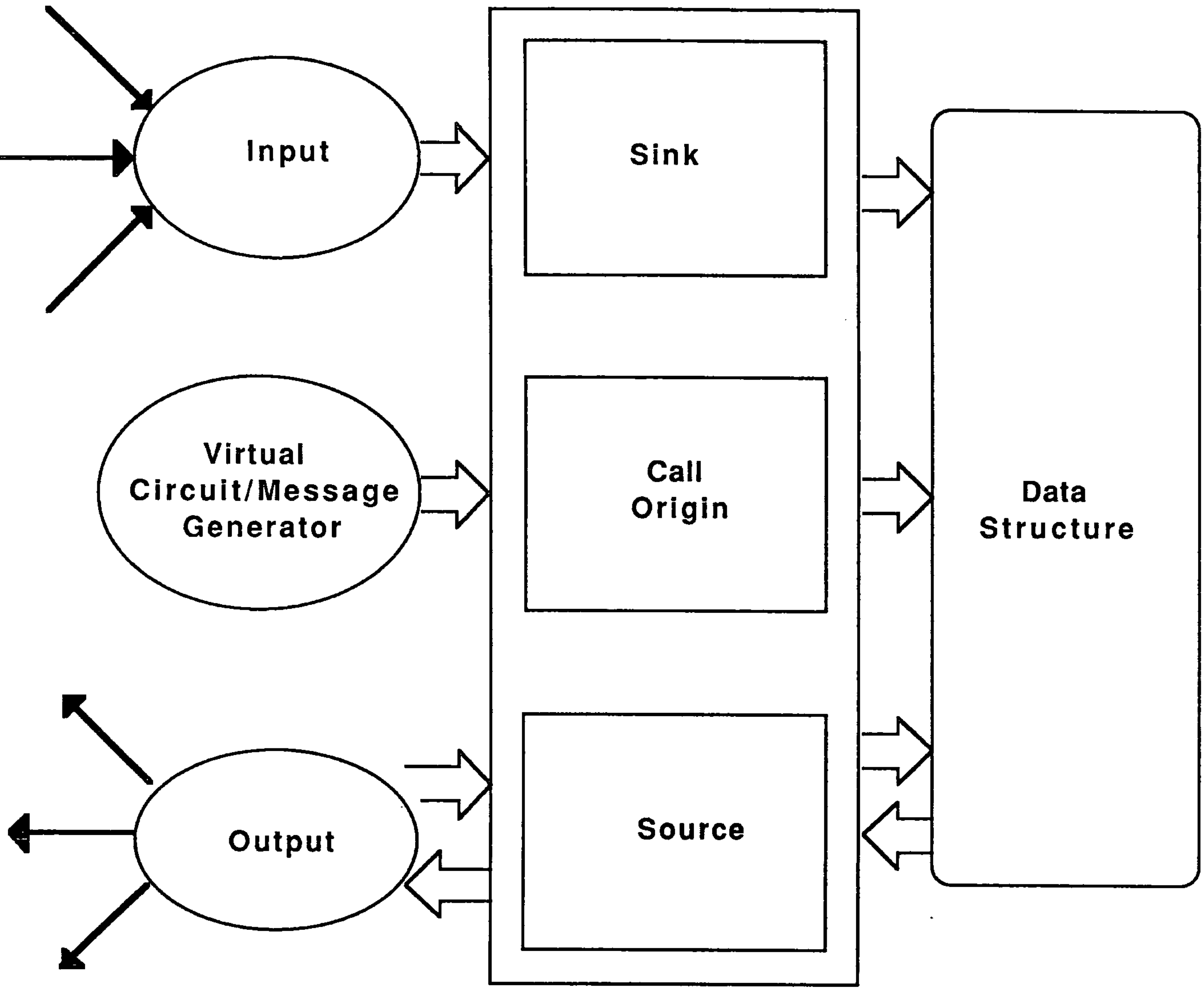


Figure 5.14: Node Structure

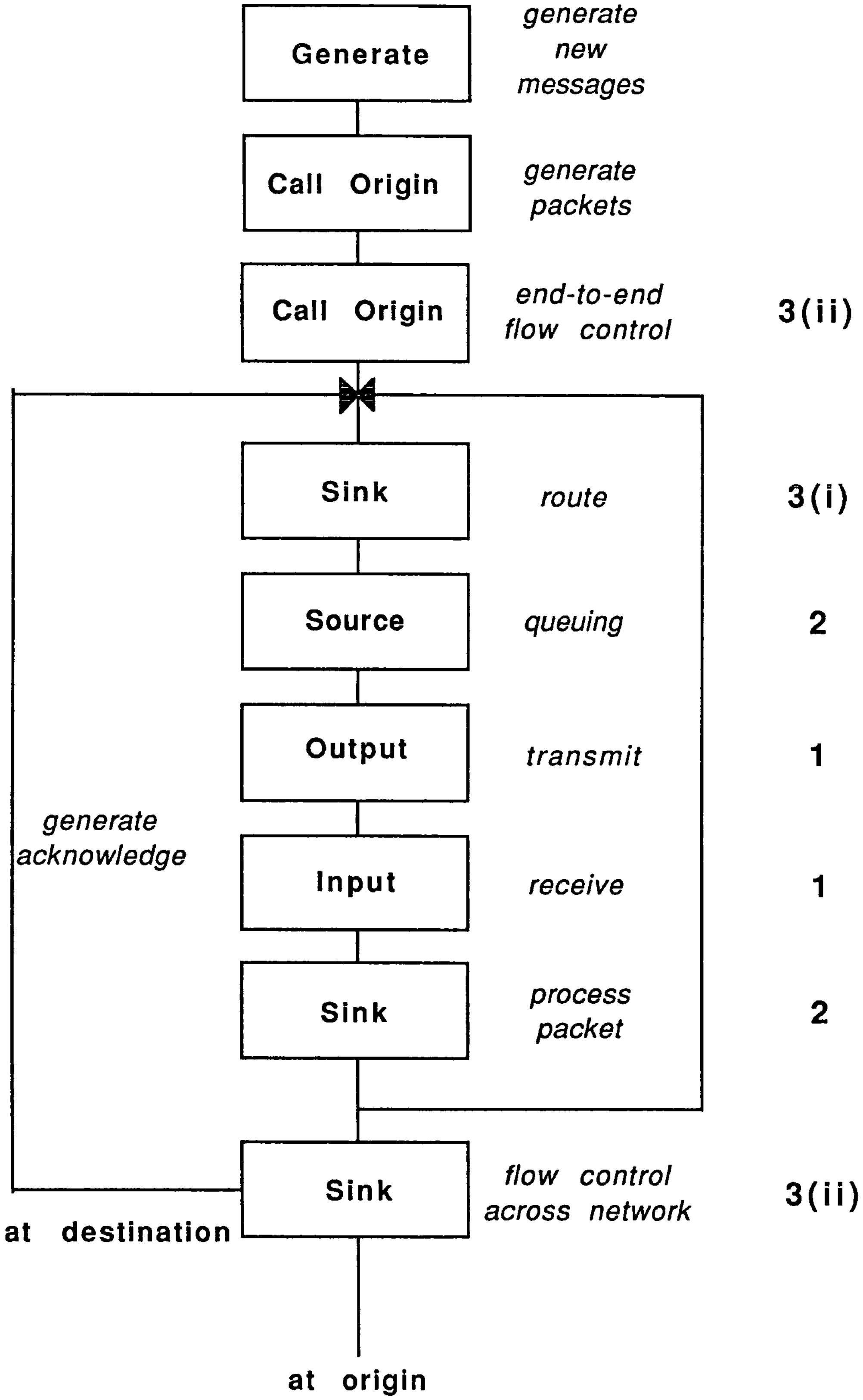


Figure 5.15: Call Progress

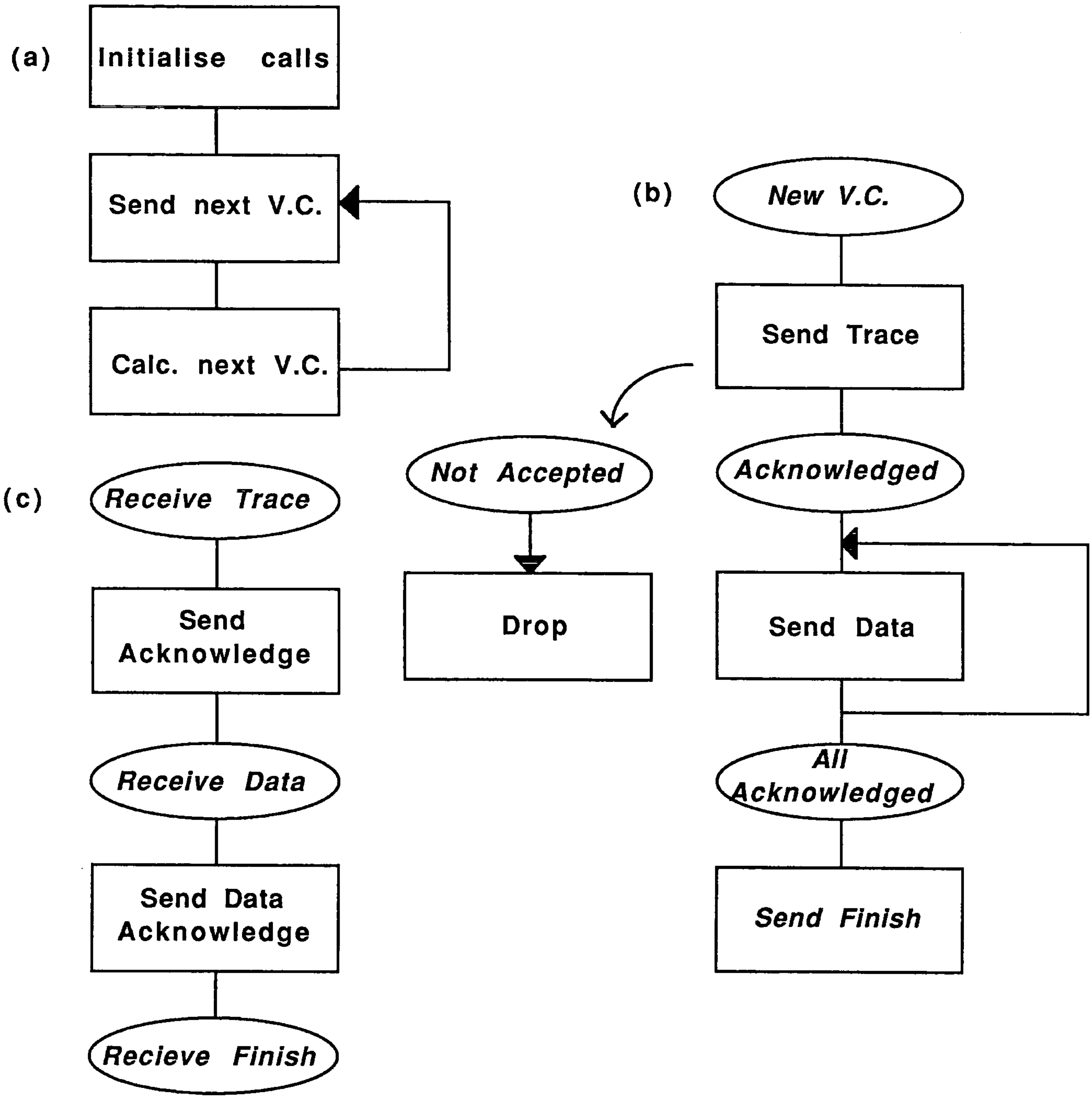


Figure 5.16: Virtual Circuit Control Layer

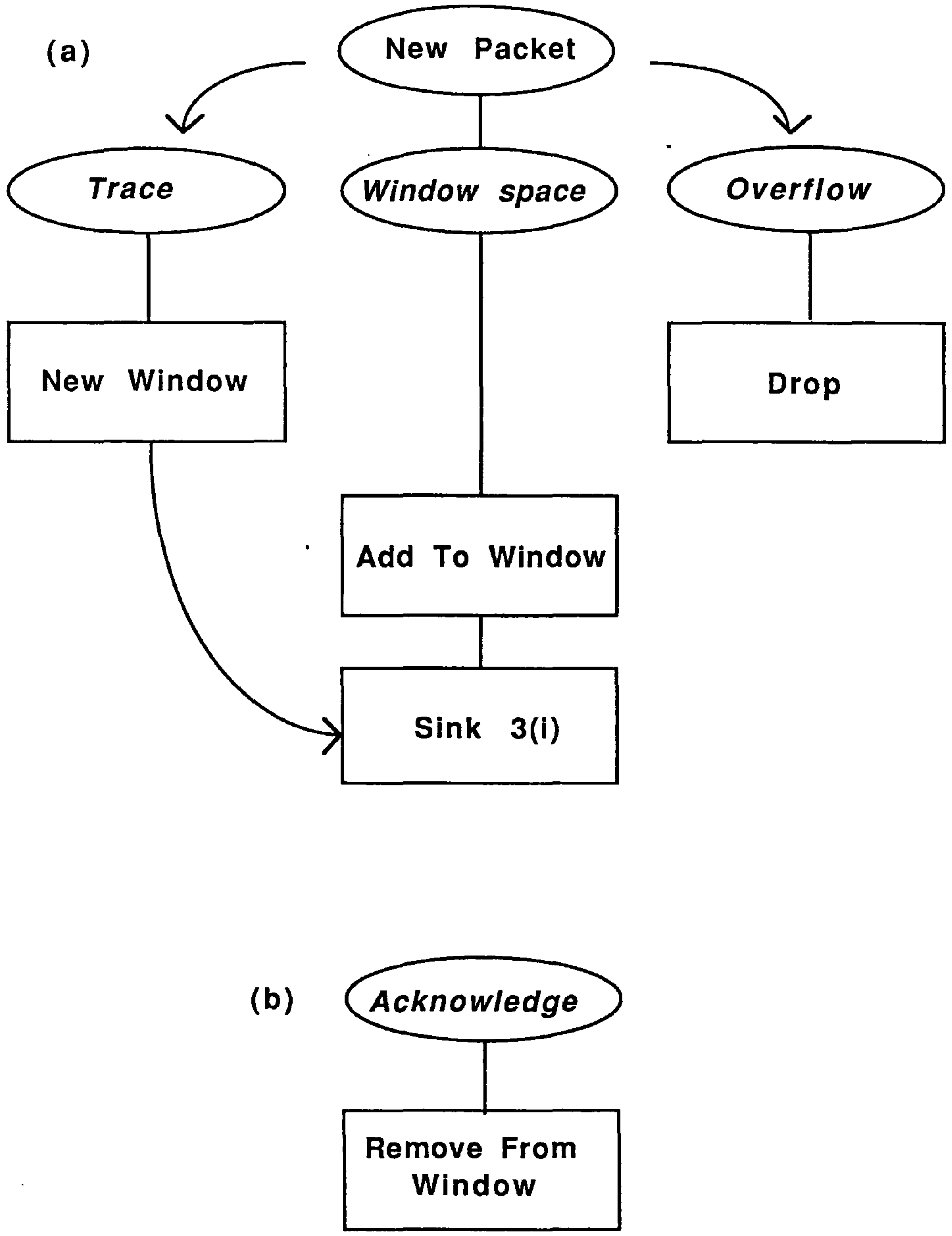


Figure 5.17: Window Control Routines

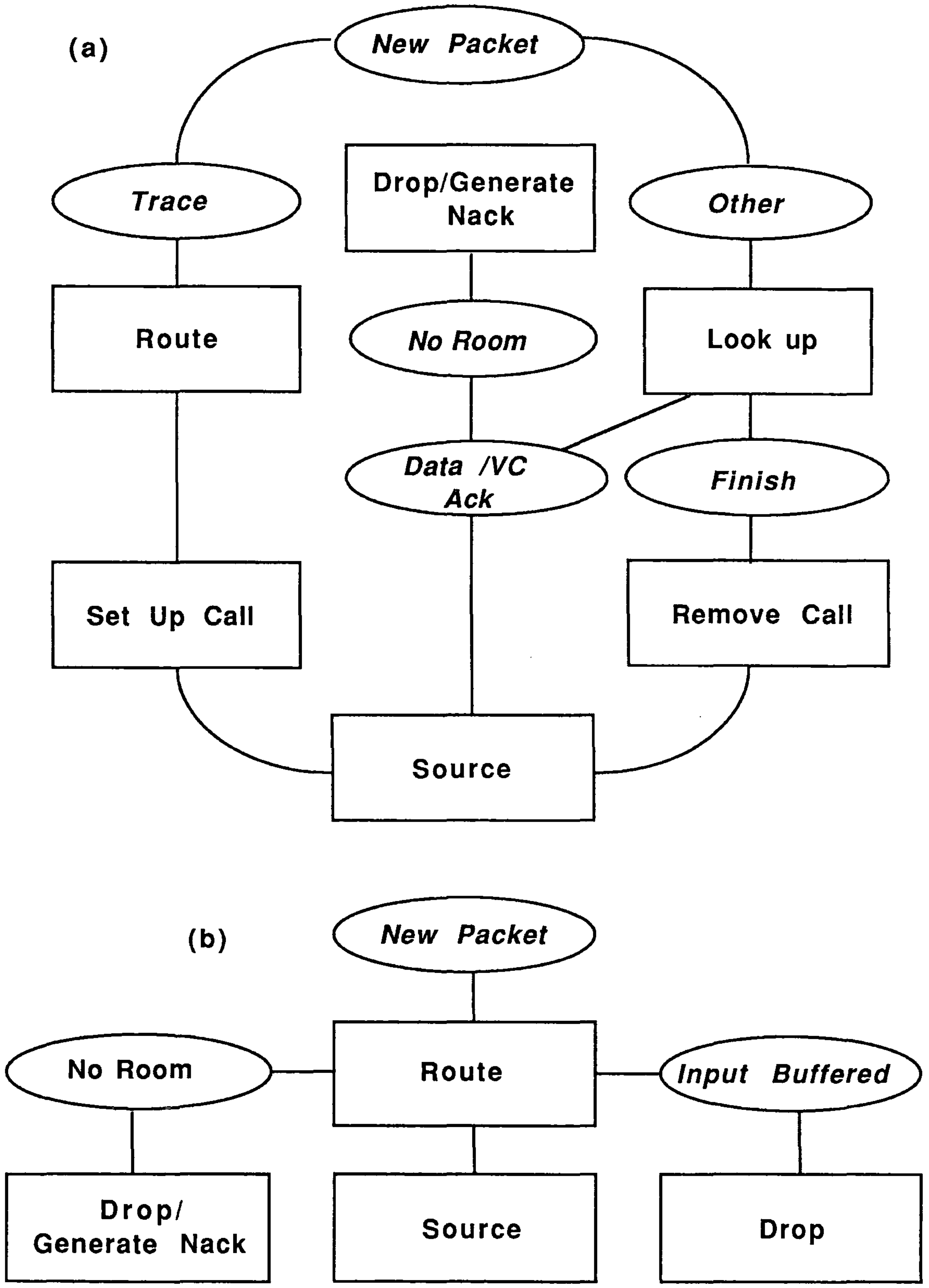


Figure 5.18: Routing Procedures

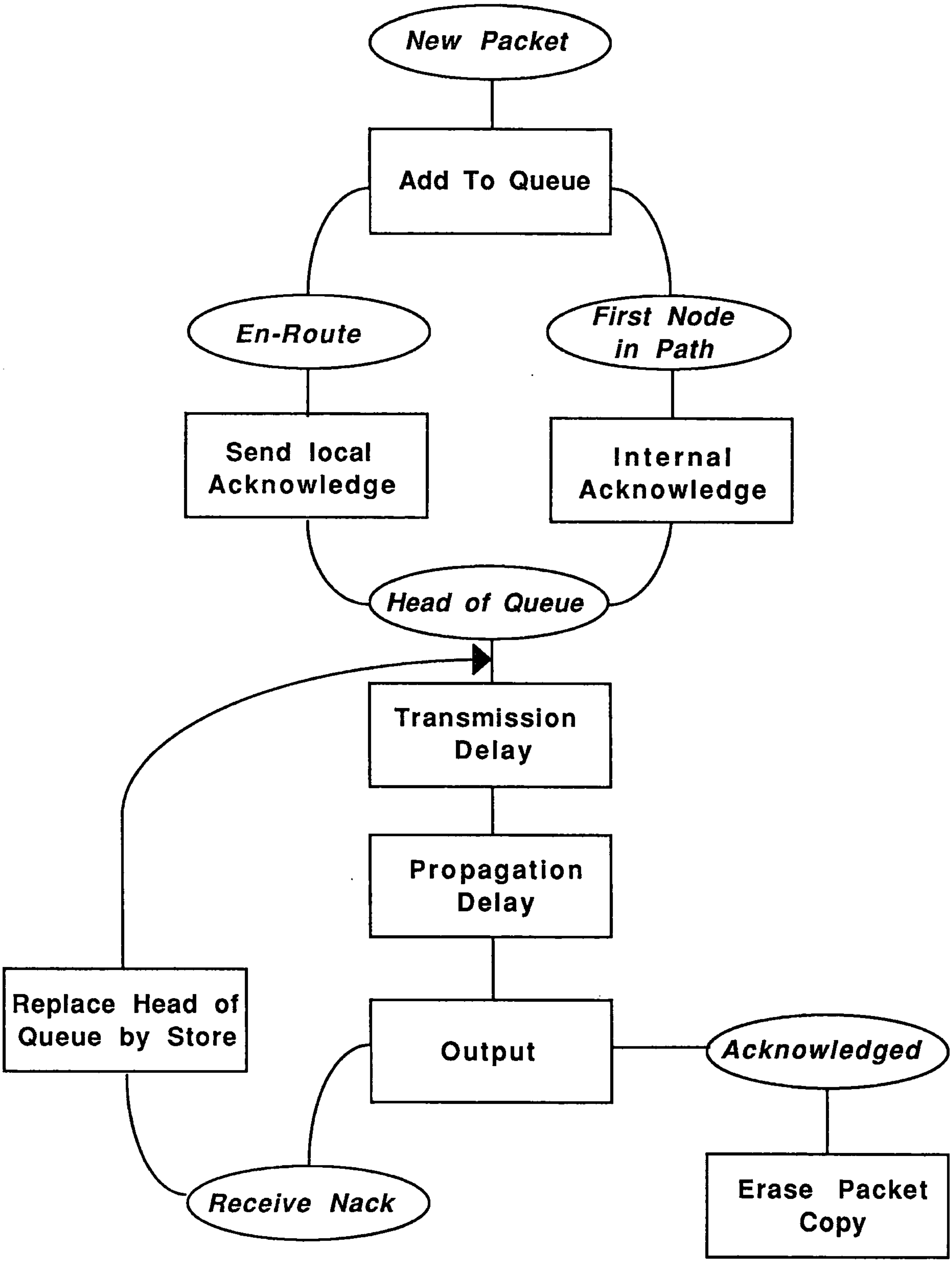


Figure 5.19: Transmission Protocol

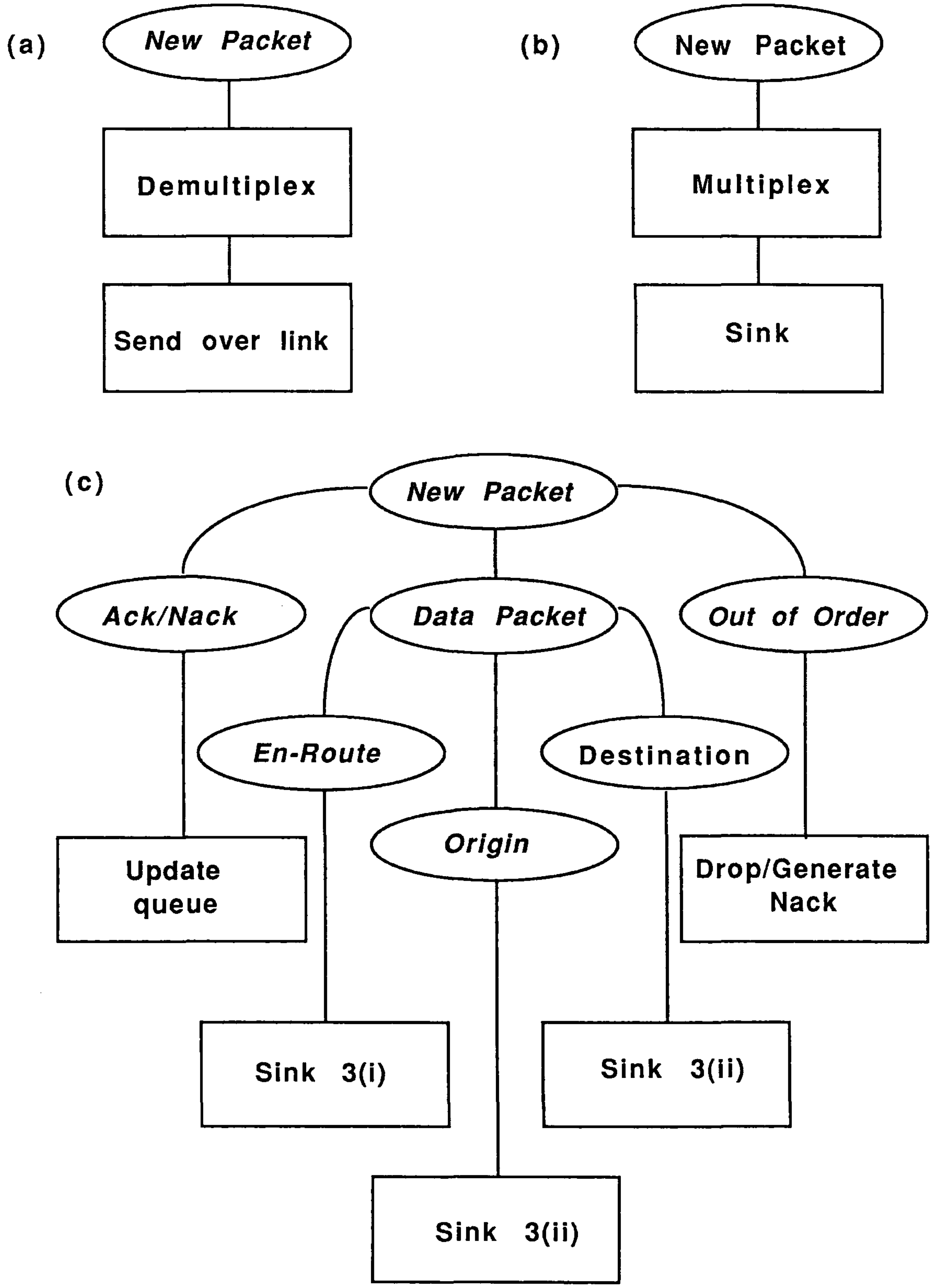


Figure 5.20: Interface And Arrival Processes

Chapter 6
Performance of Learning Automata Based Routing Algorithms in Sparsely Connected Packet Switched Networks

6.1 Introduction

In this chapter a number of experiments are conducted to analyse and highlight some of the features of adaptive routing and flow control in packet switched networks. Analytic results are supported by simulation studies to justify the assumptions inherent in the mathematical models. Work concentrates on the performance of routing and flow control algorithms which use delay as a performance metric and compares the results with optimum routing and flow control which use marginal delay as a performance metric. The delay based algorithm is formulated to route traffic over paths of minimum and equal delay through the network, while the delay based flow control policy uses a window based procedure to limit access to the network using the traffic's average transmission delay. The routing algorithm models the predicted behaviour of the L_{R-I} algorithm which uses network delay as a parameter to update the routing probabilities assigned to each route. The delay based flow control procedure was selected as a complementary algorithm because it uses the same information as the routing algorithm to form its performance metric. This combination of strategies using pure delay is shown to give almost optimal performance for a generalised topology and sub–optimal but extremely good performance for at least one special class of network.

In the next section flow diagrams are presented which outline the algorithms used to calculate the performance of the various routing strategies which are examined in the rest of the chapter. Using algorithms based on the flow deviation method, the adaptive routing policies are solved by iterative techniques based on the repeated calculation of the shortest path strategy which use performance metrics based on link utilisation. The calculation of network performance using various flow control techniques, both fixed and adaptive, are included in the models and criteria formed for the design of suitable penalty functions. A number of introductory experiments are then conducted to examine some of the key features associated with the selection of an appropriate routing function and flow control strategy. The intention of this section is to analyse simple examples to explain the basic functionality of the processes involved in each strategy. A series of more complex examples are then conducted on a ten node, sparsely connected network. The analysis of uncontrolled routing is followed by the analysis of flow control using fixed routing tables before the combination of the two adaptive policies are assessed. Finally the combination of adaptive routing and flow control are analysed over a simplified model of an international network where, due to its geographical distribution, asymmetrical traffic loading occurs which creates regions of spare capacity.

6.2 Analytic Implementations

In order to evaluate the performance of learning automata based routing algorithms in packet switched networks a number of analytic models selected from those detailed in the previous chapter were implemented. In addition a variety of fixed and adaptive window based flow control algorithms were also included to form a suite of possible combinations with the routing strategies.

The simplest routing strategy implemented was shortest path routing using the basic centralised algorithm attributed to Dijkstra[60] outlined in figure 6.1. In the algorithm each of the nodes in turn is designated as the destination node for the remaining nodes in the network. A shortest path tree is then constructed connecting each of the remaining nodes to the destination by the route of least cost. This is done by sequentially adding the node with the minimum cost path to the destination node, using the present partly connected tree. The cost of reaching the destination from each of the nodes not yet in the tree, based on the new nodes connectivity is then updated. These shortest path trees are then used to build routing tables for each destination, from each node. A minor modification to the algorithm, in which all minimum cost paths to a destination are recorded rather than just a one of the possibilities, can be used to produce bifurcated shortest path routing tables using the same basic scheme.

Shortest path and bifurcated shortest path routing strategies provide excellent single and multiple path lower bounds on performance. The obvious upper bound on the performance of a routing strategy is an optimum strategy which minimises the performance index used for the comparison, in this case average delay across the network. Fratta's flow deviation method[65] was used to calculate the optimum allocation of traffic over the network. The basic form of the algorithm, reproduced in figure 6.2, uses the shortest path calculation as a building block. On each iteration the shortest path is calculated using the current delay over each link, due to its present traffic allocation, as a metric. The resulting routing scheme is then combined with the present path allocation in such a way as to minimise the overall objective function until the algorithm converges. By simply changing the metric used to calculate the shortest path algorithm and the objective function to those proposed by Dafermos[84] this same technique can also be used to calculate the routing strategy which equalises the delay of each path taken by each of the traffic sources.

The effect of flow control on the performance of each of the routing strategies can be calculated using an approximate technique described by Thaker and Cain[100] The algorithm is based on Little's formulae, $N = \lambda T$, which relates the average number of packets in the network for each source at any one time, N, to the arrival rate λ and the average network delay, T, each traffic source experiences. Figure 6.3 shows the algorithms implementation for both fixed and dynamic routing policies. If the routing strategy is dynamic, on each iteration the new routing tables are calculated based on the current traffic rates for each source. If fixed routing is being used this section is omitted as the traffic allocation is fixed by the initialisation procedure. The utilisation of each link in the network is then derived and used to calculate the throughput of each traffic source. If the flow control strategy is dynamic then the window size defining the maximum throughput for each traffic source is recalculated using the present throughput and Little's formulae. The traffic from each source is then modified based on the current throughput and the allowed throughput until convergence is achieved.

Three flow control procedures have been implemented. The simplest policy uses a fixed window size for each traffic source on the network. This policy acts as an unintelligent deterministic comparison for the other two dynamic strategies. The remaining policies are derived from the marginal delay $\delta D/\delta r$ and actual delay based policies introduced by Gallagher[99] and Thaker & Cain[100] respectively. Each uses its form of delay as a performance index of the network congestion seen by each traffic source. A penalty function is then constructed defining the maximum allowed level of network congestion, using the appropriate performance index, given any particular capacity allocation for the traffic source. The suggested form of the penalty function, $L(r)$, where r is the traffic level or capacity allocation, is given by the formulae

$$L(r) = \frac{a/c}{(b + r/c)^n} + d/c \qquad 6.1$$

where $L(r)$ is the maximum permissable congestion level for a traffic allocation of r, C is the average link capacity and a,b,c and d are constants defining the shape of the function.

To select an appropriate function for $L(r)$ it is useful to consider the application of flow control to a traffic source over a single link in the network of capacity C for the two extreme cases of high link utilisation and very low loading. Let ρ_m define the maximum link utilisation desired. This figure can then be used to calculate the point on the penalty function, $L(0)$, at which the traffic's capacity allocation is reduced to zero. Secondly let R_o be the zero–load source allocation. This is the allowed throughput rate a traffic source would be assigned if the performance index over the traffic's path was at its minimum possible value; i.e. when there is no traffic over the path. Allocation of traffic over the link by the flow control mechanism will of course always produce a penalty value greater than the minimum possible value and so any traffic source will therefore always be assigned a throughput $r < R_o$, however this simple concept produces a convenient intercept for the definition of the penalty curve.

A further complication arises when this principle is expanded to include traffic which travels over paths of more than a single hop. A crude but simple solution which can be introduced to allow for this feature is to simply multiply both $L(0)$ and $L(R_o)$ by the hop count of the route through the network before computing the constants associated with the penalty function. This is not entirely satisfactory as it removes the protection on each link of the network, which guaranteed its maximum loading. It is replaced by a guarantee that the average link penalty per link over the entire path will not exceed the value associated with the maximum link loading, which is a much weaker statement. However if the performance index used to define the penalty function is sufficiently sensitive to small changes in link utilisation at high loads, a relatively small change in link loading on one of the links in the multi–hop path beyond the maximum proposed should cause a sufficient change in the performance index to still introduce effective flow control.

Another problem arising from the the previous extension to this form of adaptive flow control arises when it is combined with an adaptive routing policy. In this case the path lengths for each of the traffic sources often cannot be defined as there may be several possible paths through the network of different lengths. In this work the minimum hop path is calculated for each traffic source and used as the multiplier. This was felt to give a good approximation as an intelligent routing strategy would naturally route the majority of the network strategy over shortest path routes to minimise network utilisation, all other factors being equal. In a reasonably well connected network, the traffic which was directed over longer paths should not have a hop count considerably larger than the shortest path and would traverse links which were, by definition, less congested than the otherwise more favourable links of the shorter path. Excessive flow control should therefore be avoided despite the increased hop count.

As was mentioned earlier the penalty function can be expressed as a window function by the application of Little's formulae to produce an adaptive window size for traffic allocation to the network. This results in a formulae for the window size, N, of the form

$$N = CT\Big\{\Big[\frac{a}{CL(r) - d}\Big]^{1/n} - b\Big\} \qquad 6.2$$

where T is again the end–to–end delay in transmission across the network.

The optimum flow control strategy of Thaker & Cain, based on the work done by Gallagher and using marginal delay as a performance index, can be expressed in terms of the penalty function as

$$CL(r) = C\frac{\delta D}{\delta r} = \frac{a}{(b + r/C)^n} + d$$

and from the definition of D, the link delay, $\frac{\delta D}{\delta r} = \frac{C}{(C-r)^2}$ which can be used to write down the first equality which must be satisfied, namely

$$CL(0) = \frac{h}{(1-\rho_m)^2} = \frac{a}{b^n} + d. \qquad 6.3$$

By selecting R_o the second equality can be defined as

$$CL(R_o) = h = \frac{a}{b^n + R_o/C} + d$$

or

$$\left\{\frac{a}{h-d}\right\}^{1/n} - b = \frac{R_o}{C}. \qquad 6.4$$

Alternatively if delay is chosen as the performance parameter instead of marginal delay then the penalty function becomes

$$CL(r) = CD = \frac{a}{(b + r/C)^n} + d$$

and from the definition of D once again and ρ_m the first equality becomes

$$CL(0) = \frac{h}{(1-\rho_m)} = \frac{a}{b^n} + d. \qquad 6.5$$

while the second equality is

$$CL(R_o) = h = \frac{a}{b^n + R_o/C} + d \qquad 6.6$$

which is identical to the previous example.

These equalities still leave a degree of freedom in the potential form of the penalty function. In this work the function was simplified by the selection of values of 0 and 1 for d and n respectively. This leads to to a penalty function of a simple reciprocal form shown in figure 6.4. No claim to the optimality or otherwise of this functional form is made, but it was felt that a curve of this form did fulfil the major objective of its design, to quickly reduce traffic as congestion started while still trying to maintain some service until extreme conditions occurred. Selection of suitable values for ρ_m and R_o which lead directly to the derivation of suitable values for a and b for each of the dynamic strategies are left to the next section where a series of introductory experiments are presented.

6.3 Introductory Experiments

In order to investigate some of the key features of both adaptive routing and flow control strategies a series of simple experiments were formulated. In each of the experiments the network link capacities are defined in terms of a reference transmission capacity of C data units/time interval. Similarly traffic arrival rates are also defined in terms of C data unit/time interval. The first of these experiments analysed the performance of different routing strategies applied to a single traffic source between the two nodes 1 and 4 in the simple network configuration shown in figure 6.5. The data can travel over two distinct paths to the destination, via either node 2 or node 3. The path via node 2 consists of two links, each with a transmission capacity $2.0C$, twice that of the two links forming the second path via node 3. This presents each of the routing strategies with an obvious preferred and secondary route to the destination. Shortest path routing and the two adaptive strategies under study, which either equalise marginal delay over each path or equalise delay directly on each path across which data is transmitted, are applied to the network. Figure 6.6 shows the predicted average transmission delay, in time intervals/data unit, produced by each routing strategy as the traffic arrival rate is increased. At some point on each curve the network saturates and the transmission delay increases asymptotically to infinity. The graph also includes simulation results for average packet delay for both shortest path and L_{R-I} routing algorithms. Figure 6.7 plots the power of the network, defined as the ratio of throughput to delay, for each strategy over the same range of traffic arrival rates and includes simulated results for network power derived from the measurement of average delay for both shortest path and the L_{R-I} strategy.

The shortest path strategy, using the inverse of link capacity as a metric, selects the preferred path via node 2 and directs traffic exclusively over these links. In consequence as the traffic arrival rate approaches $2C$ this path saturates and the transmission delay rapidly increases. In contrast the two adaptive strategies have the power to divide the traffic over both paths to minimise their respective objective functions. The resulting delay curves are almost identical and both strategies comfortably handle traffic arrival rates approaching $3C$ before network saturation occurs. At saturation there is a similar sharp increase in the transmission delay of traffic over the network.

In order to examine the different processes involved in the two adaptive routing strategies, which return a very similar delay function, further information can be derived from figure 6.7 which shows the power of the network for each strategy over the same range of traffic arrival rates as the previous delay curve. Power, measured in (data units)^2/(time interval)^2 has been used extensively as a measurement of the tradeoff between the two performance functions of throughput and delay. The plot of power for shortest path routing is useful as a reference for the two adaptive policies. The power of the single path routing strategy increases steadily up to a traffic arrival rate of $1.0C$ and then falls away symmetrically as the traffic arrival rate continues to increase. Finally at a traffic arrival rate of $2.0C$ the power reaches a value of zero, which corresponds to a transmission delay of infinity. Each of the dynamic strategies generate an identical power value for low traffic arrival rates. As the arrival rate increases each of the strategies diverge from the shortest path plot, increase to a maximum value for a traffic arrival rate of $1.5C$ and then also fall away to a value of zero for a traffic arrival rate of $3.0C$.

The significant difference in the two plots of power for the dynamic strategies is the point at which divergence occurs from the shortest path routing curve. This point corresponds to the traffic arrival rate at which the adaptive strategies begin to split the total traffic between the two paths, rather than being routing in a deterministic manner over the preferred route. This point occurs when the unloaded, low capacity path becomes equally attractive to the routing policy as the partially loaded, high capacity path. For this single source, simple network configuration the point at which this occurs can easily be calculated for each strategy. For the Gallagher routing strategy, or optimum strategy which minimises the total average delay over the network, only paths with identical and minimum marginal delay are used over a network. Therefore the low capacity path will begin to be used when the marginal delay of the high capacity path is equal to the marginal delay of the unused low capacity path or

$$\frac{1}{C} = \frac{2C}{(2C - X)^2}$$

where X is the traffic arrival rate. Solving this equality, X has a valid solution in the range $0 \leq X \leq 2.0C$ when

$$X = \sqrt{2}(\sqrt{2} - 1)C \sim 0.56C$$

Equally when the routing strategy selects paths of equal and minimum delay for each traffic source in the network, the routing strategy will begin to select the low capacity path with a finite probability for traffic arrival rates above the point at which the delay over the high capacity path exceeds the delay over the unloaded low capacity path. This occurs when

$$\frac{1}{C} = \frac{1}{2C - X} \qquad \text{or} \qquad X = C$$

which is also the point of maximum power for the transmission of traffic over the high capacity path alone. Expanding on the last point it is simple to show that more generally, the maximum power will occur over a path of any number of links of equal capacity when the traffic over that path is equal to exactly half of the link capacities. This point will be used later to define the adaptive window penalty functions for flow control of the network.

The simulated results show good agreement with the form of both the shortest path and bifurcated L_{R-I} routing strategies. However there are two important ways in which the simulated results diverge from their analytically predicted behaviour. First, all simulated results for shortest path routing and those of L_{R-I} routing below $2.7C$ return an average delay below that predicted, leading to a power greater than calculated. This is due to the assumption in the formulation of the analytic model that the packets are transmitted through two independent M/M/1 queues. But because each packet in the simulation has a fixed size, the two service times are in fact identical for each individual packet. This leads to a close correlation between the inter–arrival time of packets into the second queue and their service times, leading to little queueing in the second stage. In the second departure from analytic predictions, the average packet delays for traffic intensities of $2.7C$ and above using the L_{R-I} algorithm were found to be much higher than suggested from the mathematical modelling. Observations of simulations over these traffic ranges revealed oscillatory behaviour and a tendency to go through periods where the selection became deterministic, effects not observed at lower traffic intensities. The effect was found to be due to the value of the update constant, a, introduced in the definition of the learning algorithm in chapter 5, which had been set at a value of 0.01. The constant was redefined as 0.002 and the experiments rerun. The average delay under these conditions for high traffic arrival rates closely agreed with predicted results. Clearly the previous value for the learning parameter had been sufficient to produce equilibrium at low and medium traffic loads but was too large for arrival rates approaching saturation, when it produced unstable behaviour leading to deterministic link selection.

The second introductory experiment examines the effect of flow control over a two link path for a single traffic source. Each link of the path has a capacity of $2C$ and the effect of various window sizes on the performance of the link as the traffic rate increases are shown in figures 6.8–6.10. The graphs plot the delay, throughput and power generated by the traffic source for window sizes of $2C$,$4C$,$8C$ and $16C$. In each case simulation results are included for the four window sizes on each graph. For each window size the analytic results for average delay and power characteristics at low loading are identical to those produced by the traffic when no flow control is applied. Under these conditions the throughput matches the offered traffic exactly. For all traffic arrival rates above those at which the window begins to constrict traffic the delay is held at a constant value. The effect on the throughput as the traffic arrival rate increases can be clearly seen in figure 6.9 for the various window sizes. The power of the traffic source, the allowed throughput over the delay, is plotted in figure 6.10 along with the plot of power for an unbounded arrival rate for reference. Clearly the optimum window size for power transfer is $2C$ which begins to assert itself at the point of maximum power in the curve. As the window size increases the point at which the flow control asserts itself moves to a greater traffic arrival rate and consequently a lower value of power. Analysing the window using Little's formulae reveals, not surprisingly, that the window size of $2C$ over the link pair produces an average link utilisation of 50%.

The simulation results of delay, throughput and from these power suggest that the application of flow control is not as sharp and well defined as the analytic model suggests. This is by no means unexpected as the model of flow control was made deliberately simple to enable the analytic modelling of multiple source packet switched networks. The simulation work suggests that application of window based flow control results in a rejection of traffic earlier than predicted and to a greater extent, although there is more agreement on the degree of control at higher arrival rates. This results in a reduction of expected throughput which leads to a reduction in the end–to–end delay, compounded by the mathematical assumptions of independent service rate. These two effects combine to produce higher than predicted power over the network for each of the window sizes. However despite this obvious divergence from predicted behaviour, the mathematical models capture the key effects seen in the simulation results. These effects can be summarised as a decline in throughput with increasing incident arrival rate and a levelling off of average packet delay which causes the network power to approach a fixed level determined by the window size of the flow control strategy.

Finally a third experiment was formulated to investigate the fairness in the application of flow control for traffic sources which traverse the network over a different number of links. Using the basis of the second experiment, two additional traffic sources where added to the network. Each of the new traffic sources competed for transmission bandwidth with the original traffic source on one or other of the two links in the original traffic sources path. Two series of experiments were then carried out. In the first all three traffic sources were given equal window sizes. Figure 6.11 plots the analytic curves and simulation results for the lost throughput of both the single hop traffic sources and the double hop source for window sizes of one and two. From the analytic curves it is clear that assigning equal window sizes to traffic which travels over paths with unequal hop counts produces an unequal grade of service. The traffic sources with the lower hop count gain an advantage over those with more remote destinations. In the second set of experiments the traffic source travelling over the two hop path is given a window of twice the size of the single hop sources. The lost throughput of both single and double hop sources are plotted in figure 6.12 along with the simulated results of the same network configurations over a range of traffic arrival rates. These analytic curves suggest a simple remedy to the problem of unfair flow control. In this simple example, basing the window size on a multiple of the hop count produces identical performance for the competing single and double hop traffic sources.

As in the previous experiment the simulation results show that flow control is more active than the analytic models suggest, especially at low traffic arrival rates. In addition the results support the analytic result that equal window sizes for traffic travelling over different path lengths produces unequal performance. Equally importantly it confirms that the two single hop traffic sources introduced in this experiment have similar throughput and therefore delay characteristics. This result infers that the two sources are interfering with the primary traffic source, across both links, to an equal extent. This is a result of the mixing of traffic sources at each queue which increases the validity of the independence assumption made by the mathematical models. The second experiment, where window size is made a multiple of hop count, shows similar results. At low traffic arrival rates flow control is greater than predicted and also unequal, favouring the traffic source over the two links. However as the traffic arrival rate increases the simulation results agree more closely both with the analytic results and each other. In general the results suggest that the analytic models are able to captures the same form of behaviour as the simulation work suggests, especially at high traffic arrival rates.

6.4 Multiple Source Routing and Flow Control

In this section the various routing and flow control strategies which have been identified in the previous work are applied to the 10 node sparsely connected mesh topology reproduced in figure 6.13. Each link has the same capacity, defined simply as $1.0C$. All traffic and capacity measurements in the work are based on this reference value, using the same units defined in the introductory experiments. This topology was selected for a number of reasons. First the topology has been used by Rudin, Chrystall and others as a test bed for the comparative analysis of routing strategies in previous work. This aided in the verification of some of the analytic work by comparison with previously published work. Secondly it is small enough to be analytically tractable and large enough to be interesting as an example of a realistic network configuration. Lastly it fulfilled the criteria sought for the work presented here, namely a sparsely connected network topology with nodes of various connectivity and minimum hop count from the rest of the nodes in the network.

6.4.1. Routing Strategy

In the first example shortest path routing and bifurcated shortest path routing strategies were evaluated for the network using hop count as a metric (equally the reciprocal of capacity would return the same answer in this network as all link capacities are identical). The performance of these routing strategies was then evaluated using a full 90 source traffic matrix in which each source destination pair was assigned an identical traffic intensity. This was carried out for a range of traffic intensities up to the point at which the network saturated and returned an average delay which rapidly approached infinity. In addition the solution to the allocation of traffic, for both optimum performance and for the equalisation of delay over each path used by a source was determined at each traffic intensity up to the point where saturation occurred. The results of these models are presented graphically in figures 6.14 and 6.15. Figure 6.14 plots the average delay over the network for the 90 sources as the traffic arrival rate increases. Shortest path routing clearly gives the least desirable performance and saturates at a traffic arrival rate of $0.09C$. Bifurcated routing performs considerably better, demonstrating the usefulness of multiple path routing and saturates at $0.11C$. Finally both optimal routing and the adaptive routing policy based on delay produce near identical performance curves and route traffic successfully up to a traffic intensity above $0.12C$ for each source. This result identifies the further gain in performance possible by the intelligent selection of traffic division over multiple paths.

Figure 6.15 plots the average network power developed, defined as the average network delay divided by the average throughput. The new performance metric confirms the conclusions of the previous plot, demonstrating again the advantages in traffic bifurcation and the similarity in the treatment of traffic by both the adaptive strategies throughout the traffic range. From the plot it is also possible to identify more easily the point at which deterioration in network performance begins due to traffic loading, i.e. the point at which there is a downturn in the network power developed.

To highlight the way in which each strategy routes traffic through the network, the individual delay of each of the traffic sources where plotted in figure 6.16 for a traffic intensity sufficient to generate an average network delay of 6.0 time intervals/data item in each case. Each of the plots displays the total transmission delay for each source divided by the minimum hop count between the source and destination nodes against a list of source identity numbers. Shortest path has a number of significant spikes, most notably traffic source 57 (origin 5, destination 7). The link connecting these two nodes clearly has a high utilisation and as the traffic increases further it is this traffic source which is instrumental in the saturation of the network. Bifurcated routing produces a more balanced picture with fewer and less severe spikes, a product of traffic splitting. The dynamic routing strategies show an even more balanced picture with very similar delay characteristics over the range of traffic sources. Significantly no single or small group of sources dominate, suggesting an even spread of traffic.

6.4.2. Flow Control

In the second experiment flow control was introduced into the 90 source, 10 node network topology. Each source is assigned a window size limiting the maximum traffic throughput across the network at any time. Both optimal flow control using marginal delay and true delay based flow control algorithms are used to modify the size of each individual traffic source. The introductory experiments demonstrated that maximum power over a link is generated when the utilisation of the link reaches 50%. Therefore the flow control in the network should prevent links exceeding a utilisation of 50% to maximise the power generated by the network. However a generalised network is unlikely to be evenly loaded at high levels of traffic. Inevitably some links will find themselves a member of several paths between different sources and destinations, while others in more remote parts of the network may only be used by a few sources. This may lead to situations where traffic is rejected because one link used to reach the destination is heavily loaded while others are relatively lighted loaded and are some way from generating their maximum power utilisation.

Examining the curve of figure 6.7 it is clear that there is actually a band of link utilisation over which optimal or near optimal power is generated. Beyond this band power falls off increasingly rapidly as traffic intensity moves further from its optimal value. In order to generate a network flow control strategy which would allow as many links as possible to enter the region of high power generation is was define ρ_m as $0.6C$ or 60% utilisation, this being the nominal definition of the boundary of the high performance band. In fact it is easy to show that the power has dropped by a factor of $C^2/100$ from its optimal power level of $C^2/4$, that is 4%. The selection of R_0 is less critical. In general R_0 must be large enough to allow a traffic level which generates a near minimum performance index to flow relatively freely. The general shape of the penalty curve with its shallow curve at high traffic levels ensures that as the performance index rises, flow control will be quickly applied. In this work R_0/C was given a value of 10.0.

Applying the selected values of ρ_m and R_0 to equations (6.3)–(6.6) and assuming values of $n = 1$ and $d = 0$, the two remaining variables, a and b, can be resolved as

$$\frac{a}{b} = X.h \qquad \text{and} \qquad \frac{a}{h} - b = 10$$

where X is the result of inserting ρ_m into the definition of the performance metric defined in equ(6.3) and equ(6.5) for optimum and delay based routing strategies respectively and h is the minimum hop count between the source and destination nodes. Solving these simultaneous equations gives values of

$$a = \frac{X}{X-1}.10h \qquad \text{and} \qquad b = \frac{10}{X-1}$$

Resolving the equation for optimum routing where, $X = 1/(1-\rho_m)^2$ leads to

$$a = 11.9h \qquad \text{and} \qquad b = 1.9$$

Similarly using delay based flow control where $X = 1/(1 - \rho_m)$, a and b assume the values

$$a = 16.67h \qquad \text{and} \qquad b = 6.67.$$

The results of applying both optimal and delay based flow control, using the parameter values calculated above, to the 10 node network topology with a full 90 source uniform traffic matrix are presented in figures 6.17 and 6.18. In both cases bifurcated shortest path routing was used to assign traffic over the network, and plots of the performance of this routing strategy without any flow control are included for comparison. Figure 6.17 shows the average source delay for the three configurations over a traffic arrival rate incident on the network over the range 0.0 to $0.3C$ for each source. The performance curves of both dynamic strategies follow closely the curve of the uncontrolled network up to a traffic intensity of approximately $0.07C$. Beyond this point the delay of the bifurcated routing strategy alone begins to increase sharply. Both dynamic strategies diverge from the uncontrolled routing curve at this point and level out as the traffic intensity continues to increase to levels up to and beyond those at which uncontrolled routing saturates.

Figure 6.18 plots the average power generated in the network by the traffic throughput for each network combination. Bifurcated routing alone reaches a maximum power at a traffic intensity of approximately $0.07C$ and then falls away rapidly as the traffic increases. Both adaptive policies follow the uncontrolled curve to its maximum power and then diverge at the same traffic intensity as in the previous figure. As the traffic increases both the optimal and delay based flow controls develop additional power from the network, above that generated at the peak of bifurcated routing alone. This corresponds to an increase in the number of links whose utilisation is increasing towards the maximum permissable by the application of the flow control procedures. Those links already at the maximum utilisation prevent traffic sources routed over those links from increasing their traffic and thereby reducing the power generated by the link. As the traffic increases fewer links remain under–utilised and more traffic sources are throttled by the imposition of a maximum throughput value less than the traffic incident on the network. As this occurs the power levels off as both the throughput and average delay tend towards fixed values.

The introduction of adaptive flow control demonstrates two major advantages. First it enables the network to cope effectively with incident traffic in excess of its maximum working load. Secondly, by selective entry into the network of those traffic sources whose routes remain relatively uncongested it can increase the average power of the network considerably, in this example by 30%.

To investigate the effect of flow control at the individual source level, the individual delay for each source was recorded at an incident traffic arrival rate of $0.1C$ for each of the three configurations. At this point the unprotected network is nearly saturated and the two flow control strategies are exerting a considerable influence on the average network performance. The delay per hop for each source is plotted against individual source identity in figure 6.19. Bifurcated shortest path routing alone produces several traffic sources with high delay/hop values. In addition there are a wide number of sources which return much lower delay values. This confirms that many of the paths are still under–utilised throughout the network, while the average delay value is a result of a relatively small number of sources, routed over heavily used links. On the application of either flow control procedure, all peaks of value greater than 2.5 time intervals/data unit disappear in an almost identical manner, to be replaced by values of 2.5 or less. Significantly 2.5 time intervals/data unit is the maximum delay/hop defined by the introduction of flow control based on a maximum link utilisation of 60%. Traffic sources with delays less than 2.5 before the application of flow control remain unaffected, their throughput at the present traffic arrival rate being less than the maximum throughput allocated to the source over the network.

Finally figure 6.20 plots the throughput of each traffic source under application of each flow control strategy at the same incident traffic arrival rate of $0.1C$. Both optimum and delay based flow control procedures select the same range of traffic sources to whom access is limited and reduce the throughput to the same degree in all but a small number of cases. In these exceptions delay based routing is marginally less severe, nevertheless the overall pattern is obviously similar. Interestingly, comparing figures 6.20 and 6.19, there is no obvious direct correlation between the selection of the sources to be inhibited, the magnitude of control imposed and the delay of the same source under uncontrolled conditions. Clearly the limiting of one source will result in less traffic over one or more network links, which may affect the performance of a number of other traffic sources using affected links. The argument suggests a complex and highly integrated relationship between link utilisation and the selection of sources to be restrained.

In the next experiment a fixed window flow control strategy was added to the adaptive strategies studied. In this strategy each source is given a fixed window size equal to a global constant multiplied by the minimum hop count between the origin and destination nodes of each source. The experiment was then conducted in two phases. In the first phase all the traffic sources were switched off except for those with origins at nodes 1,10 and 5. This reduced the number of traffic sources from 90 to 27. Optimum and delay based flow control strategies using bifurcated shortest path routing were then applied to the network using the reduced traffic matrix for incident traffic arrival rates of up to $0.3C$ for each source. A fixed window multiplier was then selected which, when applied to the reduced traffic matrix, produced a similar average network delay as the two dynamic strategies for the top end of the traffic range studied. By trial and error this value was found to be approximately $0.4C$.

The full 90 source uniform traffic matrix was then reintroduced and the performance of the fixed window strategy calculated using the same incident traffic intensity of $0.4C$. The average network delay for both the reduced traffic matrix and the full traffic matrix using fixed routing are plotted in figure 6.21. Plots of both dynamic flow control strategies for the two traffic matrices are also included for comparison. Several interesting points can be extracted from this graph. Clearly the application of the fixed flow control strategy to the full traffic matrix results in a greater average delay than the same flow control applied to the reduced traffic matrix. The poor performance of the fixed policy is further highlighted by comparison with the two dynamic flow control policies when they are applied to the full traffic matrix. Here there is a substantial difference in the average delay of network traffic between the fixed and adaptive policies for a wide range of traffic values up to and beyond the point where an unprotected network would saturate.

Additionally, the dynamic flow control procedures applied to the full traffic matrix limit the entry of traffic to the network much more rapidly than when applied to the reduced matrix. The full traffic matrix introduces three times as much traffic as the reduced matrix for the same value of incident traffic intensity for each source. Link utilisation will therefore be much greater for the full traffic matrix and the network will reach its maximum input capacity on the application of flow control more quickly. The reduced traffic matrix will be able to continue to admit traffic from at least some sources over a much greater traffic range and so continue to produce a positive performance gradient as the incident traffic increases. Finally the adaptive flow control policies return a lower average delay for the full traffic matrix than in the reduced case. This is a result of the selection of the active set of sources in the reduced traffic matrix. By including nodes 1 and 10 in the active set many of the longer paths through the network are utilised. Because of this the average end–to–end delay in the reduced traffic matrix will be greater than in the full traffic matrix with its more even spread of path lengths.

Figure 6.22 plots the power generated by the application of the same six network configurations shown in the previous figure over the same traffic range incident on the network. Both adaptive policies return similar powers to each other for the two cases of full and reduced traffic matrices. The power generated by the full traffic matrix is lower than that generated by the reduced traffic matrix as a result of the greater average throughput the reduced number of sources can generate for the same average delay value, given the same network topology and capacities. The two curves of the fixed flow control's performance demonstrate the degradation in performance in comparison to adaptive policies when the traffic matrix is significantly different from the one used to engineer the flow control strategy.

The experiment has highlighted the problem of assigning fixed flow control policies when the number of active sessions over the network may vary significantly. When the number of sessions is equal to the number for which the scheme has been evaluated, near optimal performance can be engineered for high arrival rates. However as the number of sessions varies the efficiency of the network is reduced. As was shown above if the number of sessions increase, the flow control policy is insufficient and average delay increases. Alternately if the number of session reduce, flow control may prove too severe and traffic may be restricted that could have been accomodated. In addition the plots also show that by engineering the fixed policy for worst case conditions, the performance over the range of traffic arrival rates above uncontrolled network saturation but below high overload is inferior to the adaptive strategies, even for the engineered traffic matrix. This is a function of the adaptive policies ability to selectively control sources according to the state of the network rather than apply blanket restrictions which may be inappropriate in conditions less severe than gross overload.

6.4.3. Routing and Flow Control

Having looked at dynamic routing and adaptive flow control in isolation, the performance of combining the two techniques is evaluated in this section. In particular the combination of delay based flow control and a dynamic routing policy which routes traffic over paths of equal and minimum delay is compared with the use of optimum routing and optimum flow control based on the measurement of marginal delay. The performance of both dynamic routing strategies with and without flow control are plotted in figures 6.23 and 6.24. Both adaptive policies return near identical delay and power characteristics without flow control. With the application of their respective flow control policy the delay curve of each policy quickly levels out to give an average delay of approximately 4.5 time intervals/data unit. The optimal combination returns a fractionally lower value at the higher traffic arrival rates which is translated into a higher power level, approximately 3% greater than the purely delay based alternative. Comparing these result with figure 6.17 in which both flow control strategies are applied to bifurcated routing, it is interesting to observe that only a small advantage in performance has been gained by the introduction of an adaptive routing policy of either form.

Two important points can be made from this experiment. The first point is that the application of a dynamic routing strategy and flow control combination based on the performance metric of delay rather than the mathematically optimum metric of marginal delay can give a near optimum performance. This is significant because of the implications for the implementation of such dynamic strategies embedded in real networks. It has been shown that the class of learning algorithms using L_{R-I} techniques can be used to form simple distributed routing algorithms which converge to equalise the delay over the paths selected for the transmission of traffic over the network. In addition it has been shown that combining this form of algorithm with an equally simple delay based flow control procedure provides adaptive flow control. The important connection between these two techniques is that they use the performance metric of delay, a metric readily available within networks already due to timeout features invariably built into transmission protocols. In contrast the performance metric of marginal delay, which involves estimating the derivative of the network delay with respect to the traffic arrival rate is a much more difficult parameter to measure with the same degree of confidence.

The second point clear from experiments on this network is there is only a small improvement in the performance of the network using an adaptive routing strategy over the unintelligent but multiple path algorithm, bifurcated shortest path routing. This is for two major reasons. First the nodes of the network topology are well connected and a significant portion of the traffic travelling over multiple links is able to be divided over two or more paths, spreading the traffic over the available capacity. Secondly a full uniform traffic matrix was used for the comparative performance of the routing algorithms. This limited the spare capacity in the network, reducing the availability of alternative paths for the more intelligent algorithms to discover. This second point is pursued in the next section which introduces a particular class of network for which this is often untrue.

6.5 International Network

In this section a particular type of network is highlighted whose properties make it an interesting candidate for a dynamic traffic–sensitive routing policy.

A typical example of such a network is the international network which spans several time zones or even the entire globe. The key difference between this type of network and a more geographically localised network is that at any one time of the day significant parts of the network may not be generating or receiving traffic, while subscribers to the rest of the network are actively communicating with each other. As an example consider the network in figure 6.25. This network is divided into three regions representing different time zones. Each region has three fully connected nodes, two of which also act as gateways to one of the other regions. The third node in each region is not directly connected to the other two regions and so must communicate over a path via one of the gateway nodes. Once again all trunk capacities are assigned equal capacity which will be referred to as $1.0C$.

To investigate a typical scenario of the type described above, regions 1 and 2 were defined to be active while region 3 was defined as inactive. A traffic matrix was then constructed to reflect this in which nodes 1 to 6 generated uniform traffic to all nodes in regions 1 and 2 except themselves. Nodes 7 to 9 neither generated or received traffic, but were allowed to act as transit nodes in a path between any origin and destination nodes between which traffic was being transmitted.

The average delay for this traffic matrix using shortest path, bifurcated shortest path, optimum and the delay based adaptive routing strategies are presented in figure 6.26 as the traffic arrival rate of the active sources increases. Both the traffic independent algorithms based on shortest paths show similar delay characteristics, saturating at a traffic arrival rate around $0.11C$. Little benefit is gained by the use of the unintelligent bifurcated strategy in the network because of the sparse connectivity between the active regions. Few bifurcated paths can be identified for the set of active traffic sources, leading to a very similar performance profile for the two algorithms. Both the adaptive policies on the other hand are able to take advantage of longer paths through the inactive region, largely unused by the shortest path strategies. In this way the adaptive strategies are able to double the traffic intensity which the network can support before saturation occurs.

Figure 6.27 plots the power generated in the network by each of the routing strategies applied to the traffic matrix. Optimum routing, shortest path routing and bifurcated shortest path routing all follow familiar curves consistent with the previous plot of delay, however the plot of the delay based routing policy shows anomalous behaviour. As the traffic intensity increases the equalising routing policy rises smoothly to a maximum and begins to fall away up to a traffic intensity of approximately $0.13C$. At this point its behaviour changes and it begins to trace a new arc only to be replaced by a third curve. In each case the gradient of the new curve at the crossover point is less severe than the original, improving the overall characteristic curve of power vs. traffic intensity. At each crossover point a major change in the routing tables takes place. At the first crossover nodes 1 and 5 become increasingly utilised as links between nodes 2 and 3 and nodes 4 and 6 become congested. At the second crossover point node 9 is used for the first time as a node in paths through the network as the link between nodes 7 and 8 becomes congested. Because delay is used as a metric, the traffic intensity at which these two major changes in traffic distribution occur do not correspond to the optimum points of changeover, behaviour which produces the smooth optimum curve instead. This is simply a more complex example of the behaviour already seen in the introductory experiments and displayed in the single source plot of figure 6.7.

Two important issues arise out of this simple experiment. First the use of traffic–sensitive dynamic routing is shown to very useful when spare capacity in the network becomes available due to asymmetric traffic loading over the network. Secondly, it has been shown that under conditions where ordinary bifurcated routing strategies are unable to capitalise upon alternative routes, an adaptive routing strategy which uses delay as a metric can find these routes and return a close to optimal performance.

Finally the network was analysed using a variety of routing and flow control combinations. The average delay for optimum routing and flow control, delay based routing and flow control and bifurcated shortest path routing with optimum flow control are presented in figure 6.28. There is a close correlation between the two dynamic routing strategies for traffic loads up to approximately $0.12C$. At this point the delay based curve flattens out before the optimum curve, returning a slightly better average delay for the remaining traffic interval. The optimally controlled bifurcated strategy begins poorly, producing a high delay at low loads. Later it recovers to return the lowest average transmission delay of the three combinations. These results on their own are misleading and must be considered in conjunction with figures 6.29 and 6.30 which plot the power generated in the network and the average source throughput over the same traffic range. The throughput and from this the power of the bifurcated strategy can be seen to be significantly suboptimal. Traffic which enters the network under the bifurcated scheme has a low average delay at high traffic loads, but this is at the expense of network access. The other point these graphs demonstrate is that the introduction of adaptive flow control has drawn the performance of the two adaptive strategies together, despite their differences in routing policies. The flow control has limited the traffic to an extent where the performance of the two strategies are comparable.

6.6 Conclusions

The introductory experiments clearly show the need for traffic bifurcation to successfully utilise the full network capacity. In addition, delay is shown to be a useful but somewhat insensitive index of performance. Power, the ratio of throughput to delay, proves to be not only good at differentiating between routing strategies with similar delay characteristics, but is also of a form in which the magnitude of the index gives a direct representation of the performance. Its usefulness is demonstrated clearly by its ability to not only differentiate between the performance of the two adaptive routing strategies applied to the single traffic source, but also by the amount of additional information that can also be extracted concerning the details of the traffic splitting between the two paths. The experiments in simple flow control confirm the need for tight control to preserve the power generated by the network, and the individual sizing of windows determined by the separation of the nodes communicating over the network, if each traffic source is to be treated as fairly as possible. Additionally these simple experiments identify the link utilisation which generates the maximum power and provide information for the formulation of a suitable penalty function for the restriction of traffic entry into an overloaded network.

The simulation work shows how the concept of independent service rate, the heart of the mathematical assumptions necessary for the routing models formulation, is invalid in single source experiments because of the fixed size of a data packet. However when two or more traffic streams interfere the assumption, although still strictly not valid, provides an excellent approximation. The accompanying simulation work on flow control shows how the simple model for window based mechanisms captures the major effects of limiting traffic accepted into the network. The model proves to be a little simplistic at low arrival rates but provides good agreement as the arrival rate increases and flow control is increasingly applied.

Moving on to full network topologies, the experiments demonstrate a number of interesting points. Obviously the adaptive policies consistently outperform the static shortest path policies, although the bifurcated shortest path policy performs admirably considering its simplicity in the generalised 10 node network. This was helped by the connectivity and even spread of traffic in the models examined. In the international network the difference in performance between the adaptive and static policies increases as the traffic asymmetry increases and the network topology becomes more sparsely connected. Under these conditions the traffic–sensitive adaptive policies are able to exploit unused traffic capacity not considered by the less intelligent alternatives. The second major area of the work considers the impact of flow control on network performance and demonstrates the importance of effective access control as traffic rates increase. The work demonstrates how network performance can be enhanced by the selective admission of traffic from the incidence matrix to utilise the full capability of the network. Just as in the case of selection of a suitable routing strategy, the use of adaptive flow control is shown to be a powerful tool for maximising network performance over a wide range of traffic levels, independent of the number of active sessions. Finally the work has demonstrated that a routing strategy that directs traffic over paths of equal and minimum delay returns a performance very similar and in some cases inseparable from that attained by the use of the mathematically optimal strategy. Evidence suggests that the L_{R-I} strategy may be such a policy. Used in conjunction with delay based flow control, the natural partner to such a strategy, the two adaptive strategies produce a combination with a performance again comparable to that optimally attainable. But, unlike its optimal companions, these delay based strategies are implementable using current protocols and measurements already available within existing packet switched architectures.

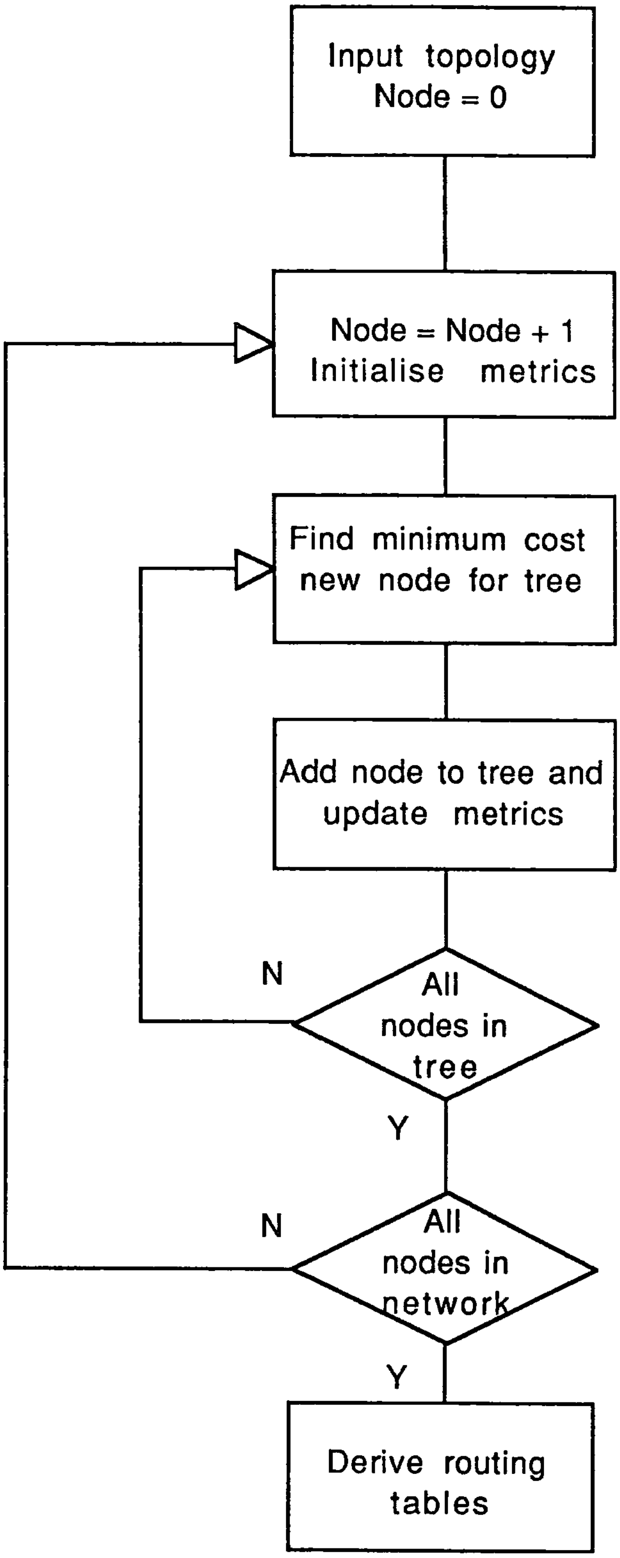

Input topology
Node = 0
Node = Node + 1
Initialise metrics
Find minimum cost new node for tree
Add node to tree and update metrics
N
All nodes in tree
Y
N
All nodes in network
Y
Derive routing tables

Figure 6.1

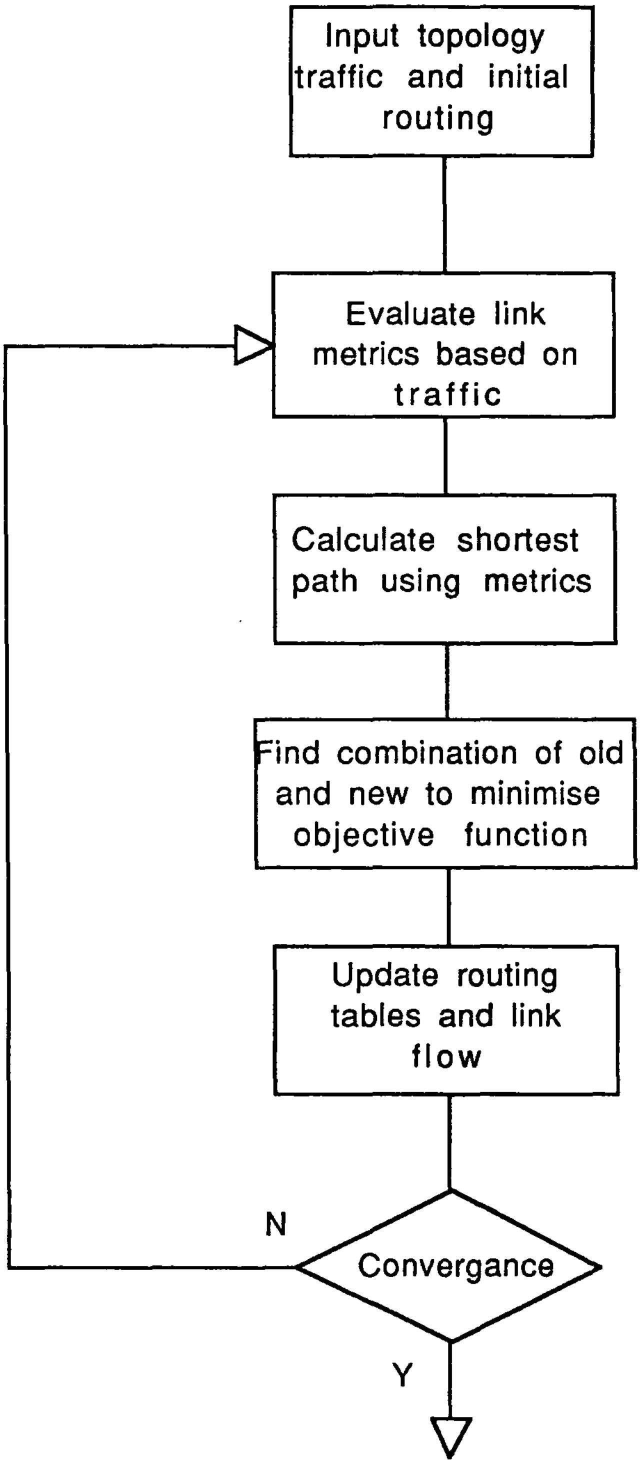

Input topology traffic and initial routing
Evaluate link metrics based on traffic
Calculate shortest path using metrics
Find combination of old and new to minimise objective function
Update routing tables and link flow
N
Convergance
Y

Figure 6.2

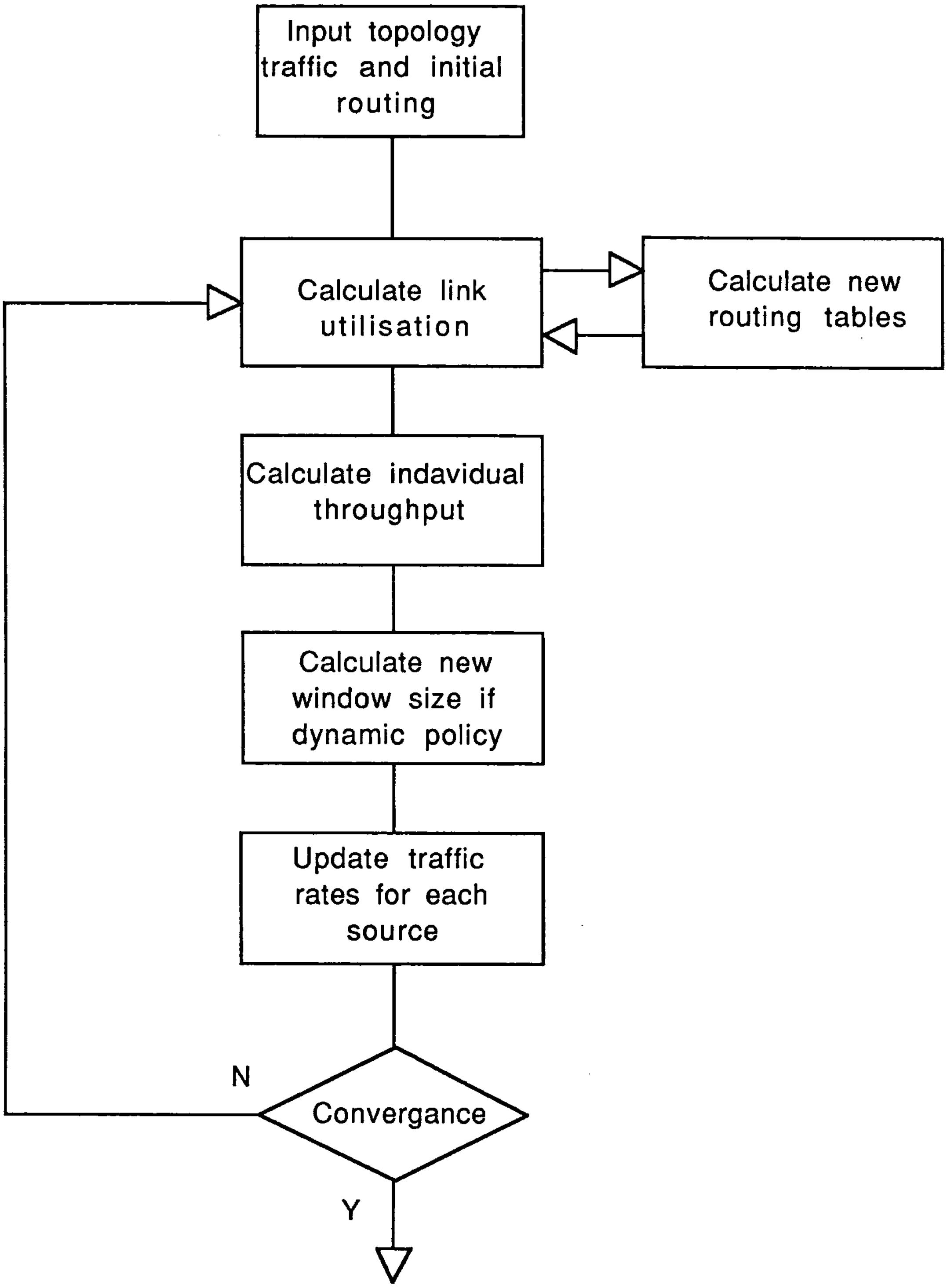

Input topology traffic and initial routing
Calculate link utilisation
Calculate new routing tables
Calculate indavidual throughput
Calculate new window size if dynamic policy
Update traffic rates for each source
N
Convergance
Y

Figure 6.3

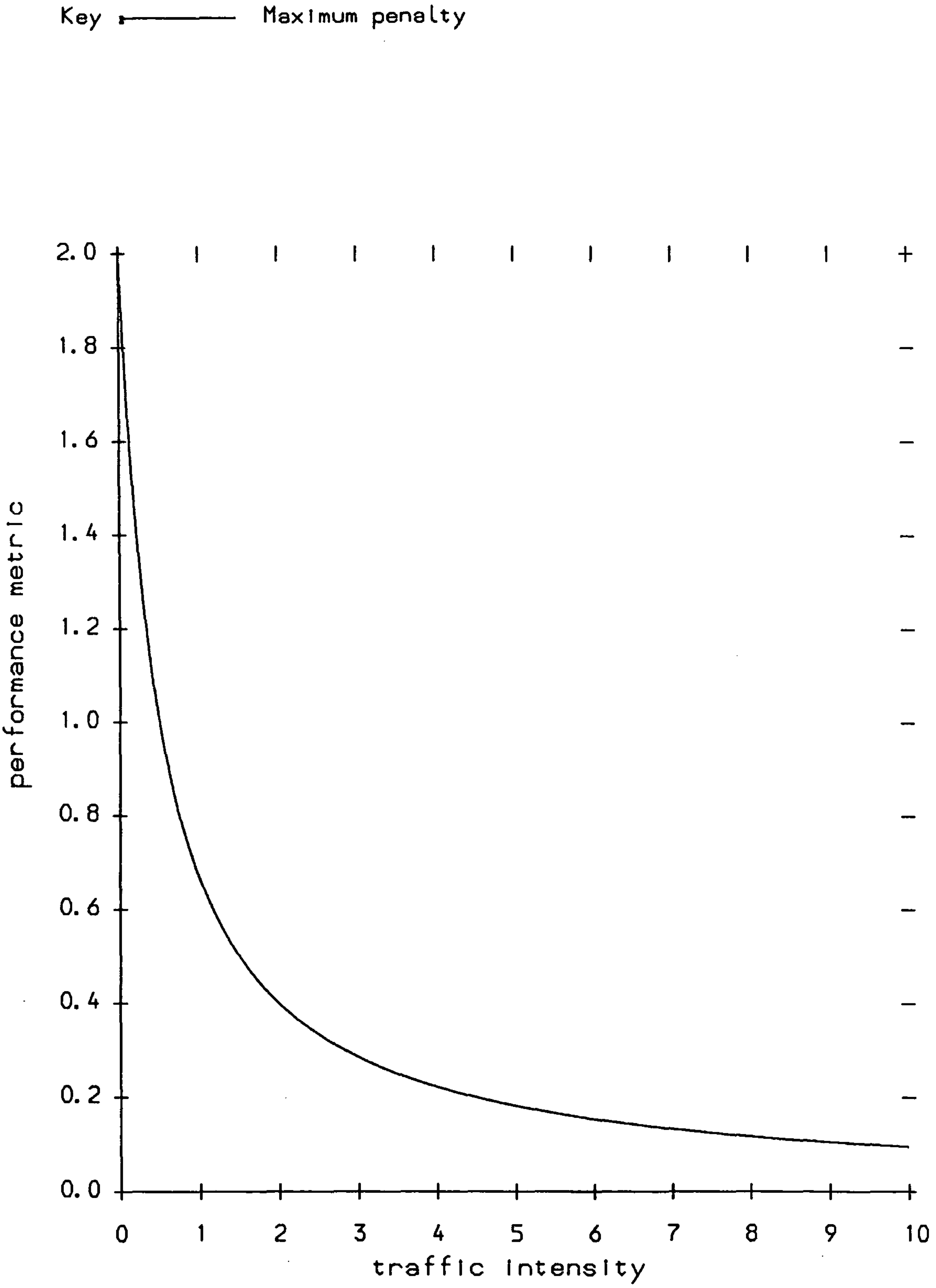


figure 6.4

pemalty function of the form L(r) = 1.0/(0.5+r)

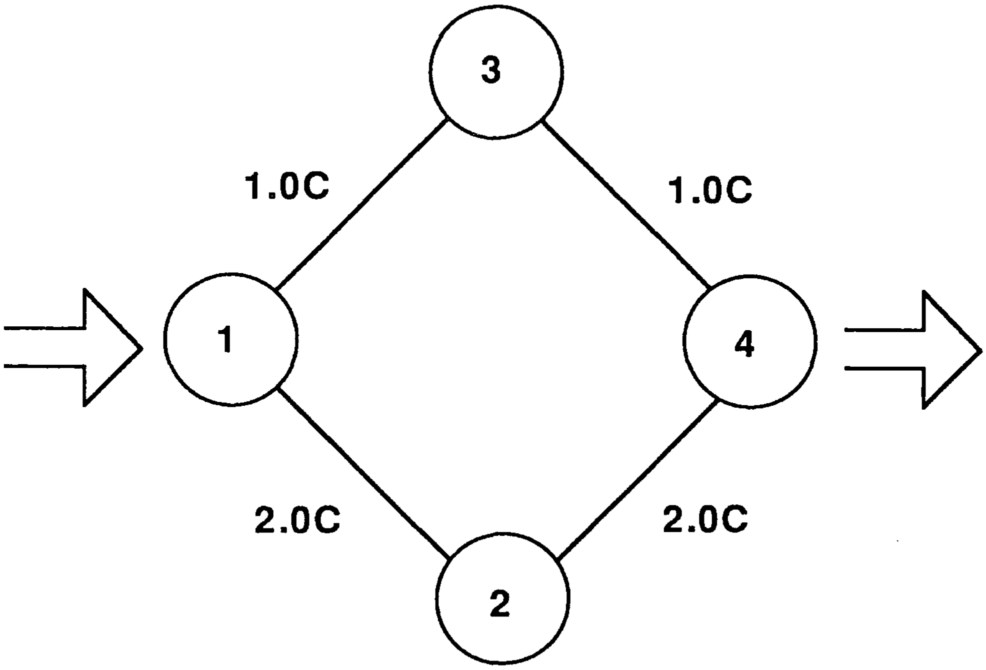


Figure 6.5: Simple four node network

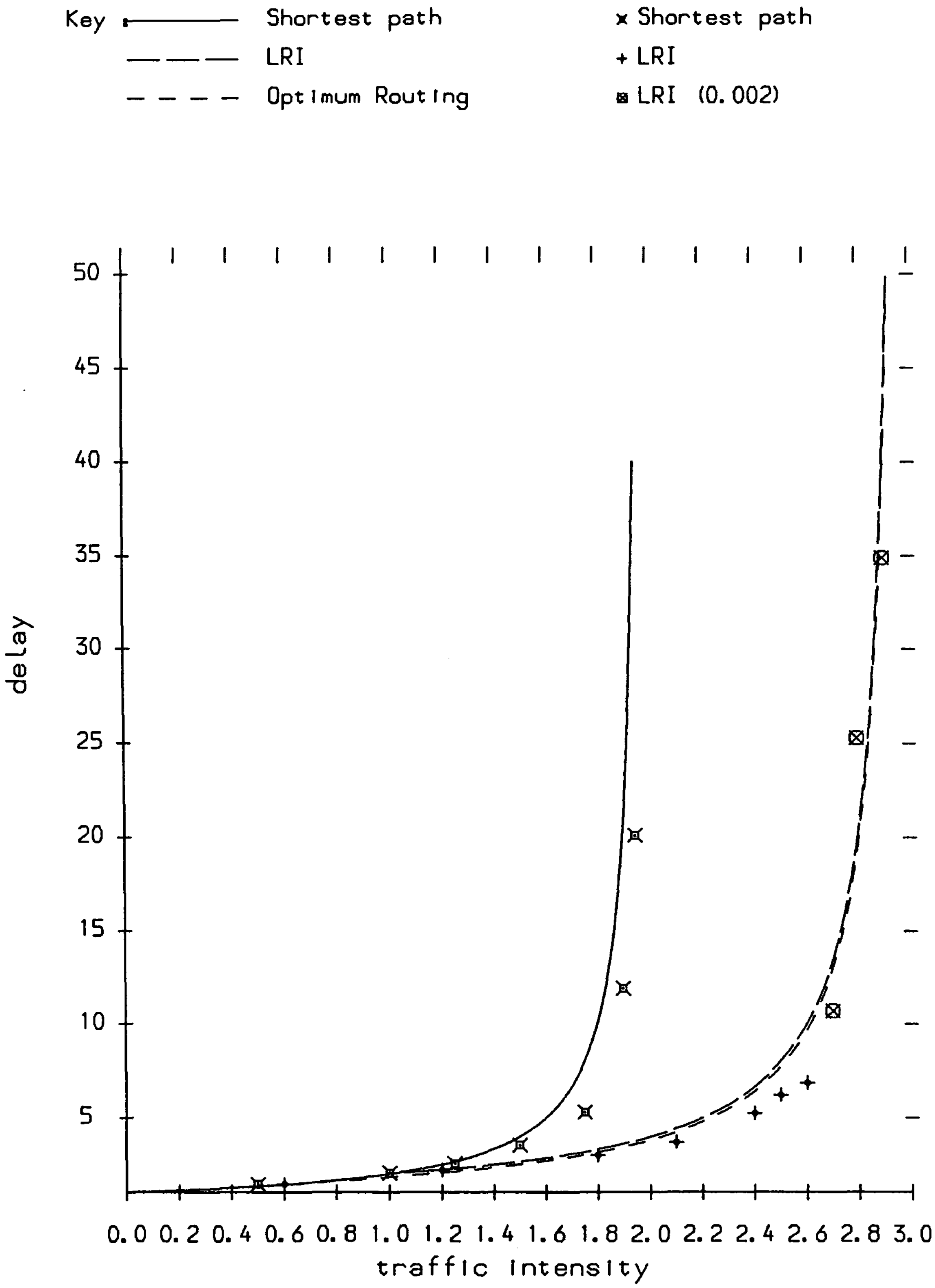


figure 6.6

analytic and simulated single source delay

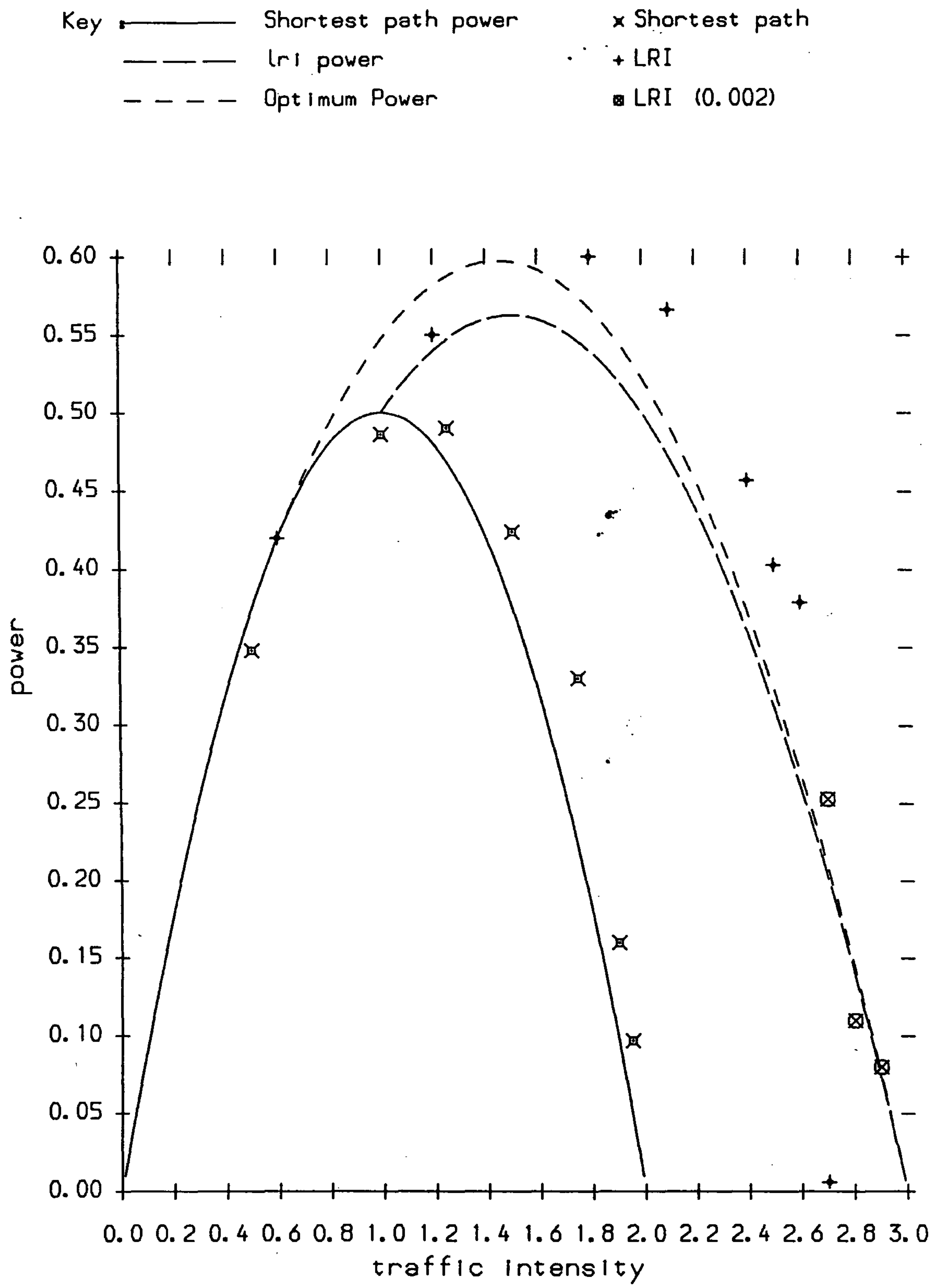


figure 6.7

analytic and simulated single source power

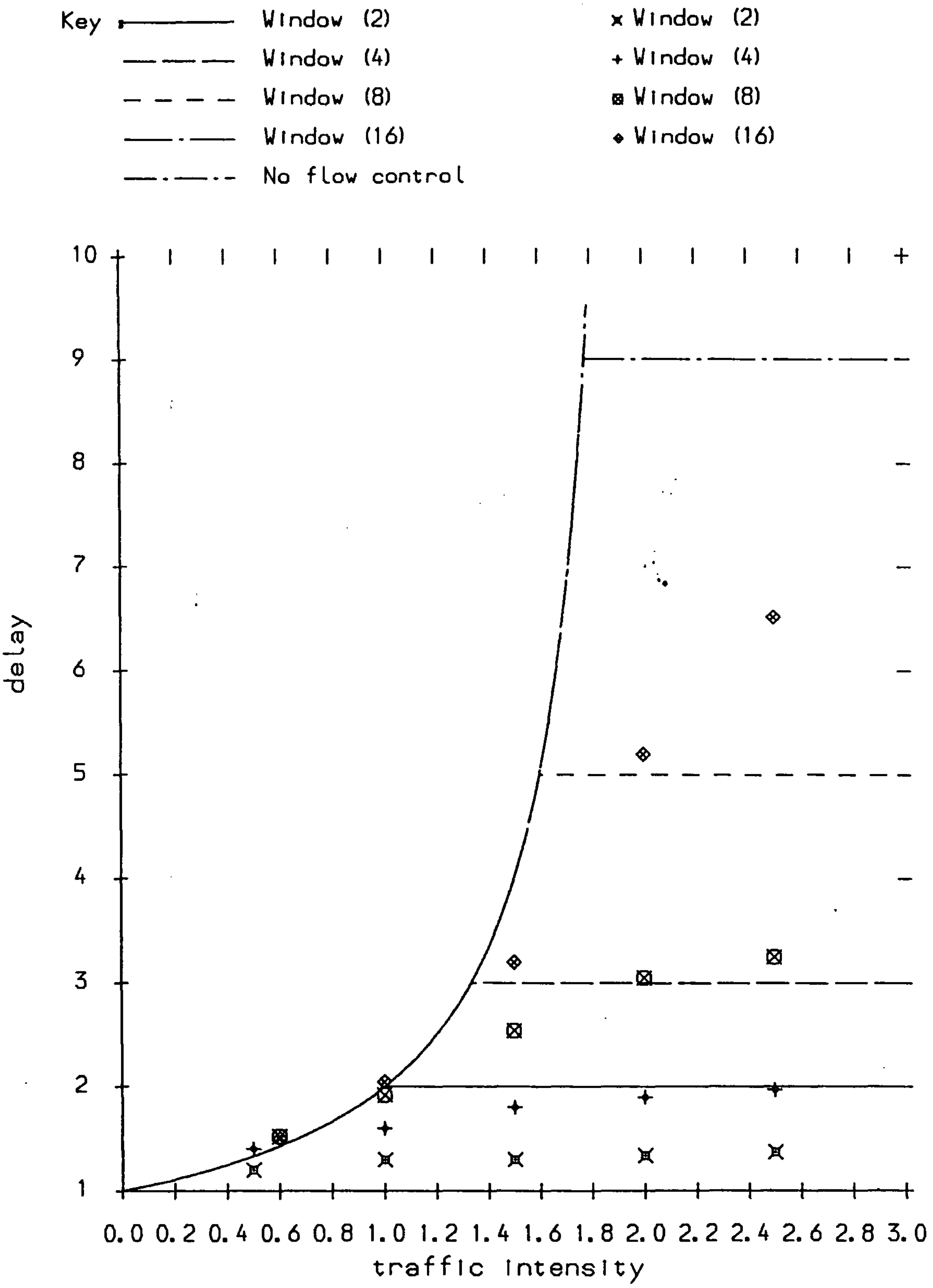


figure 6.8

analytic and simulated results foe single source flow control

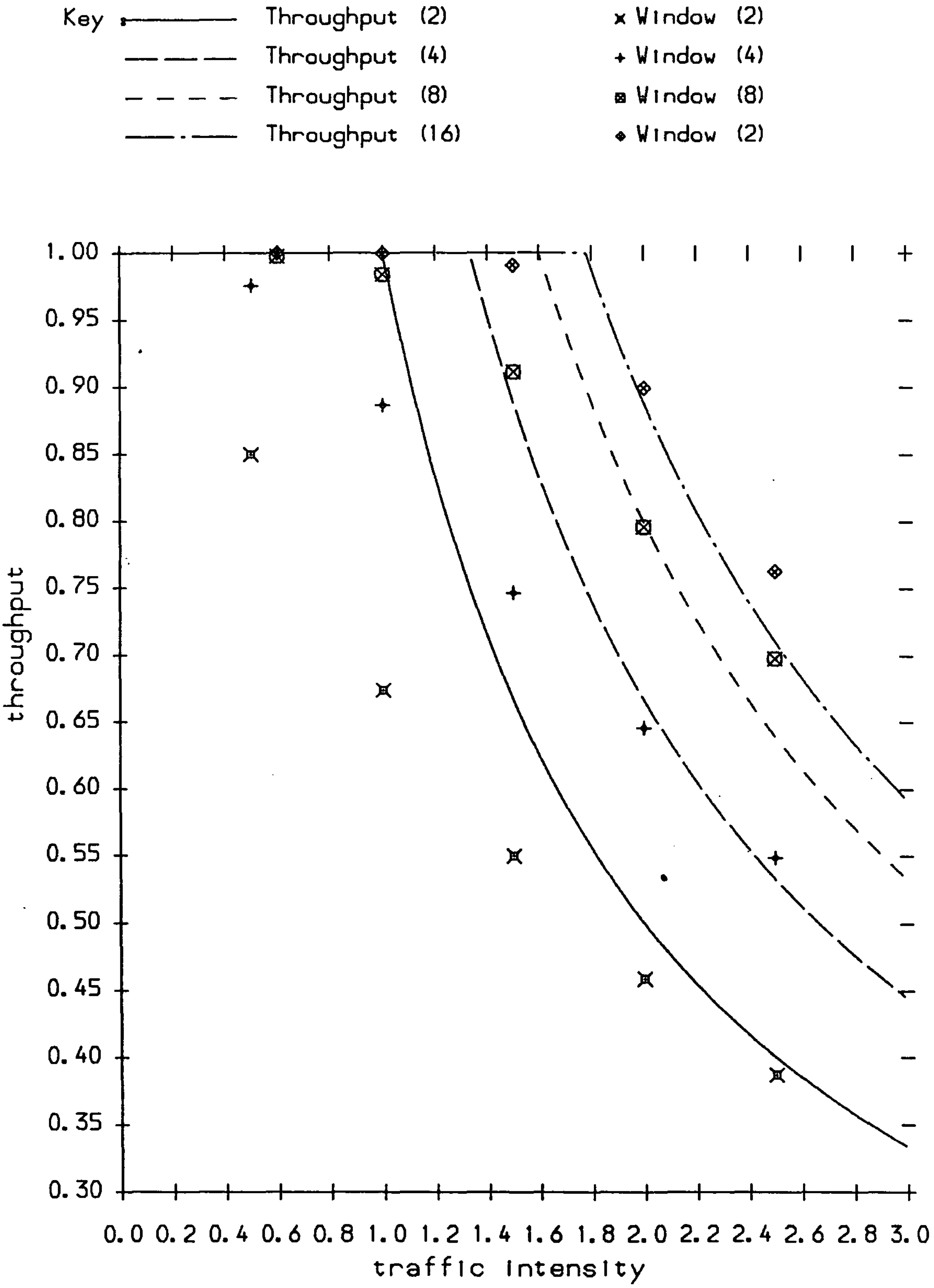


figure 6.9

analytic and simulated throughput with flow control

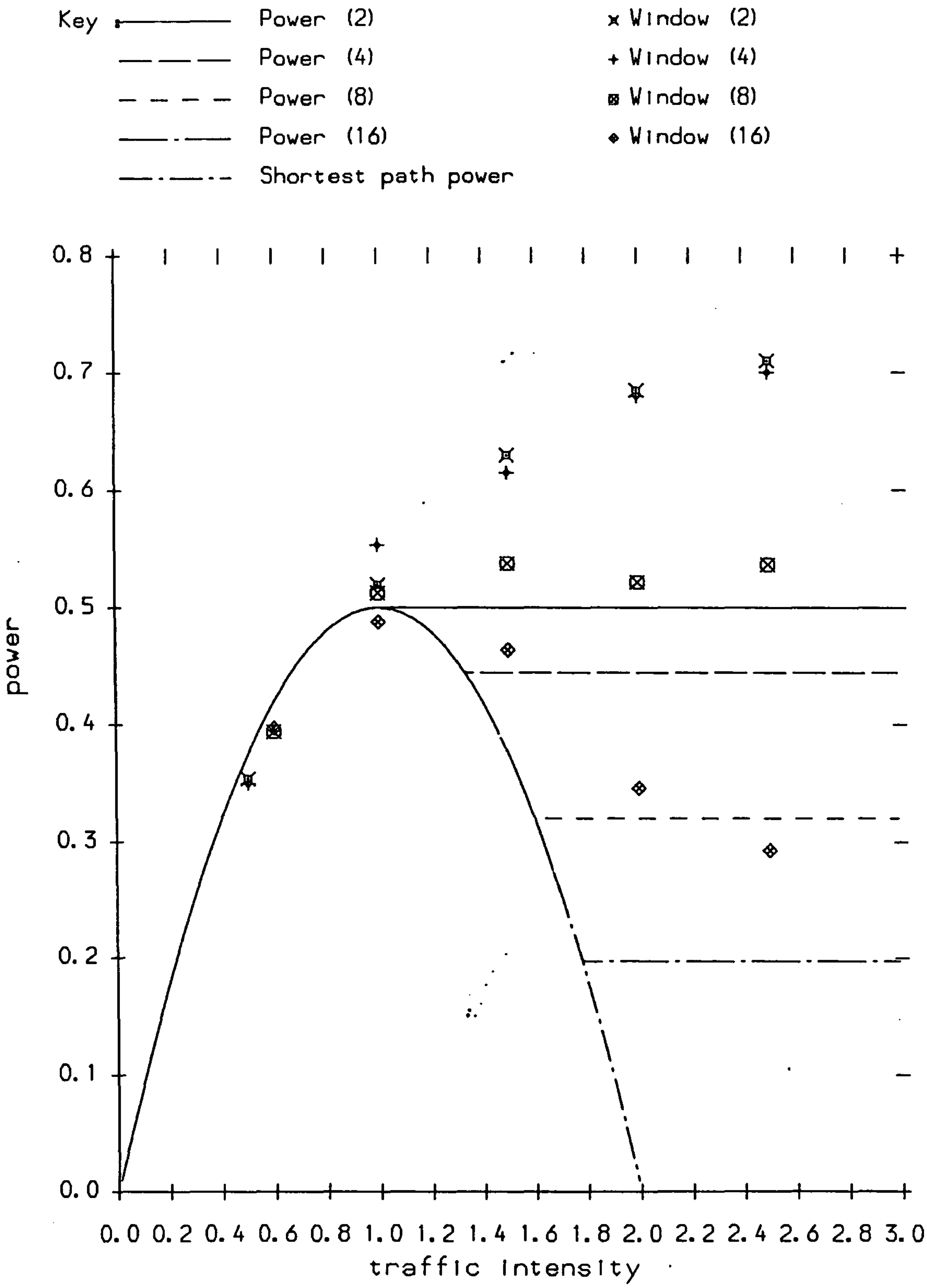


figure 6.10

analytic and simulated power with flow control

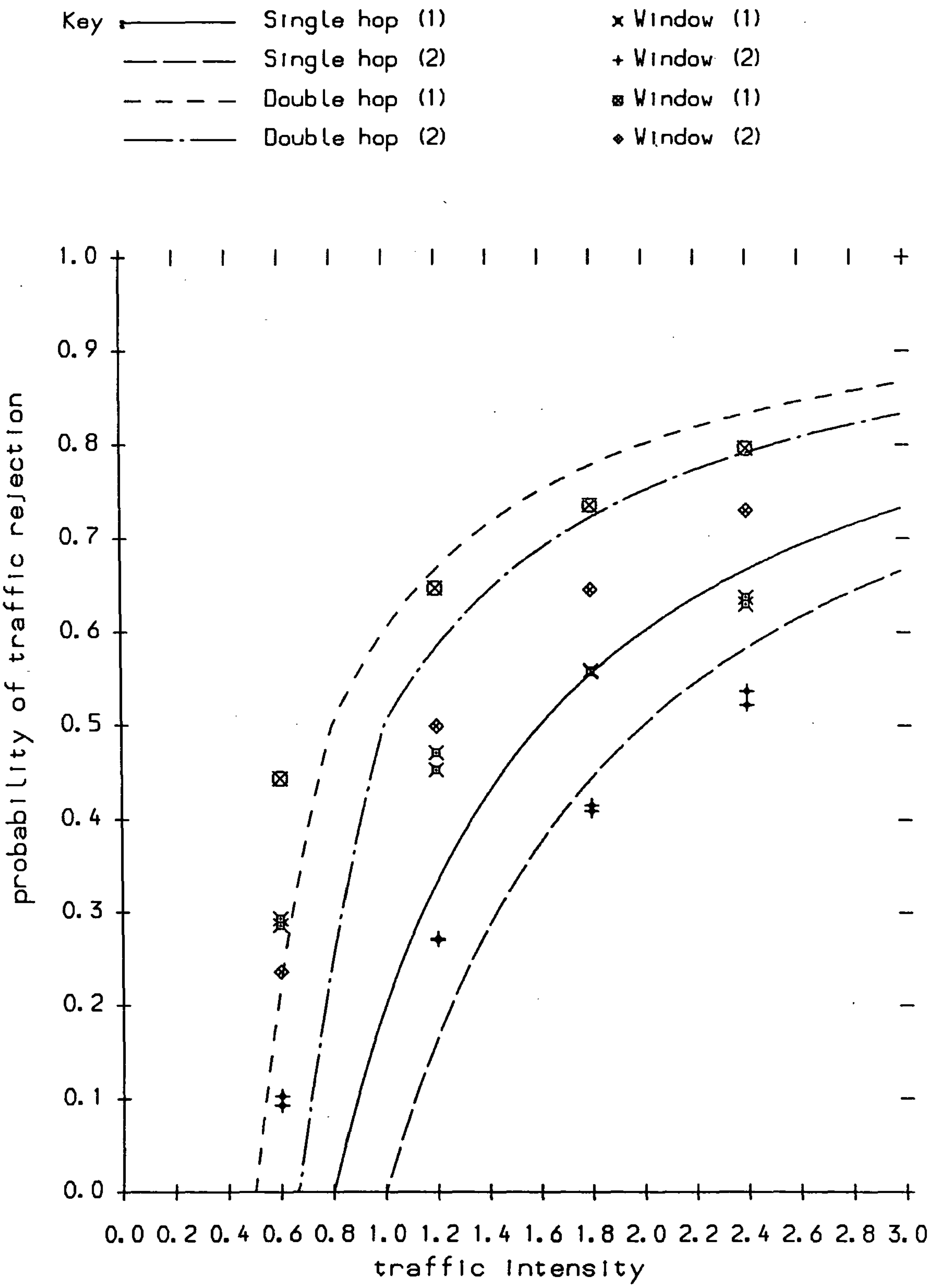


figure 6.11

lost throughput using equal window sizes

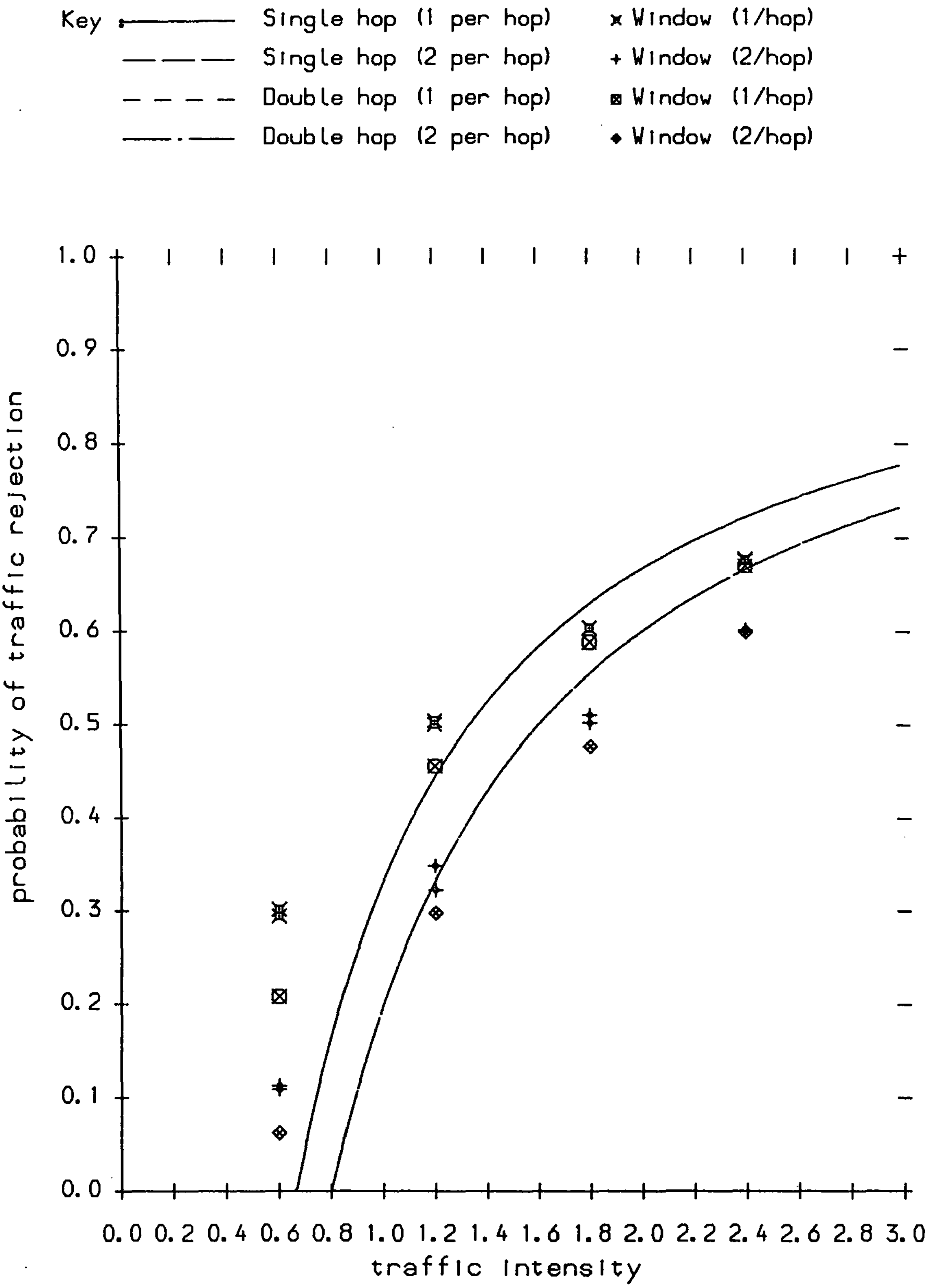


figure 6.12

lost throughput using weighted window sizes

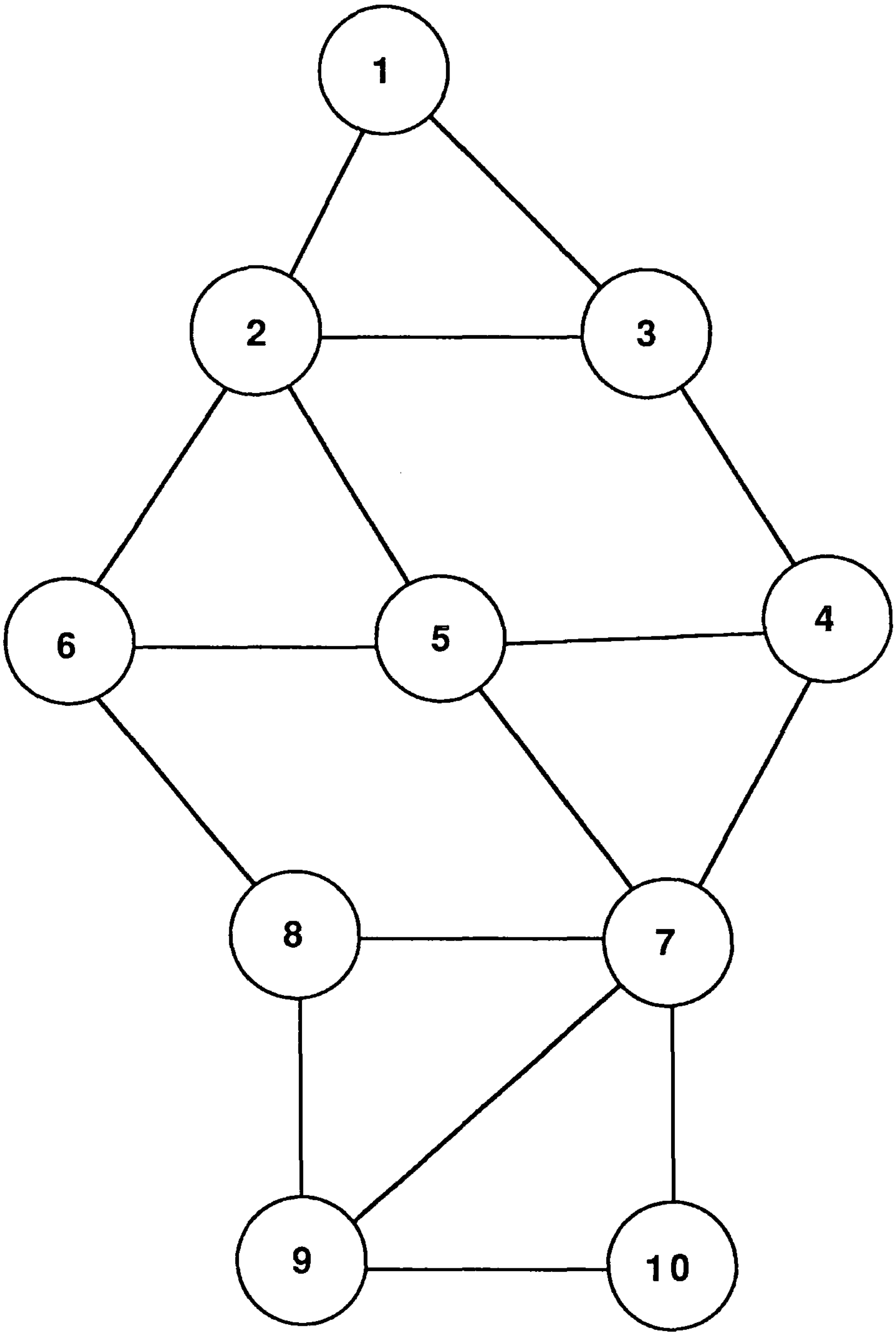


Figure 6.13 : 10 node test network

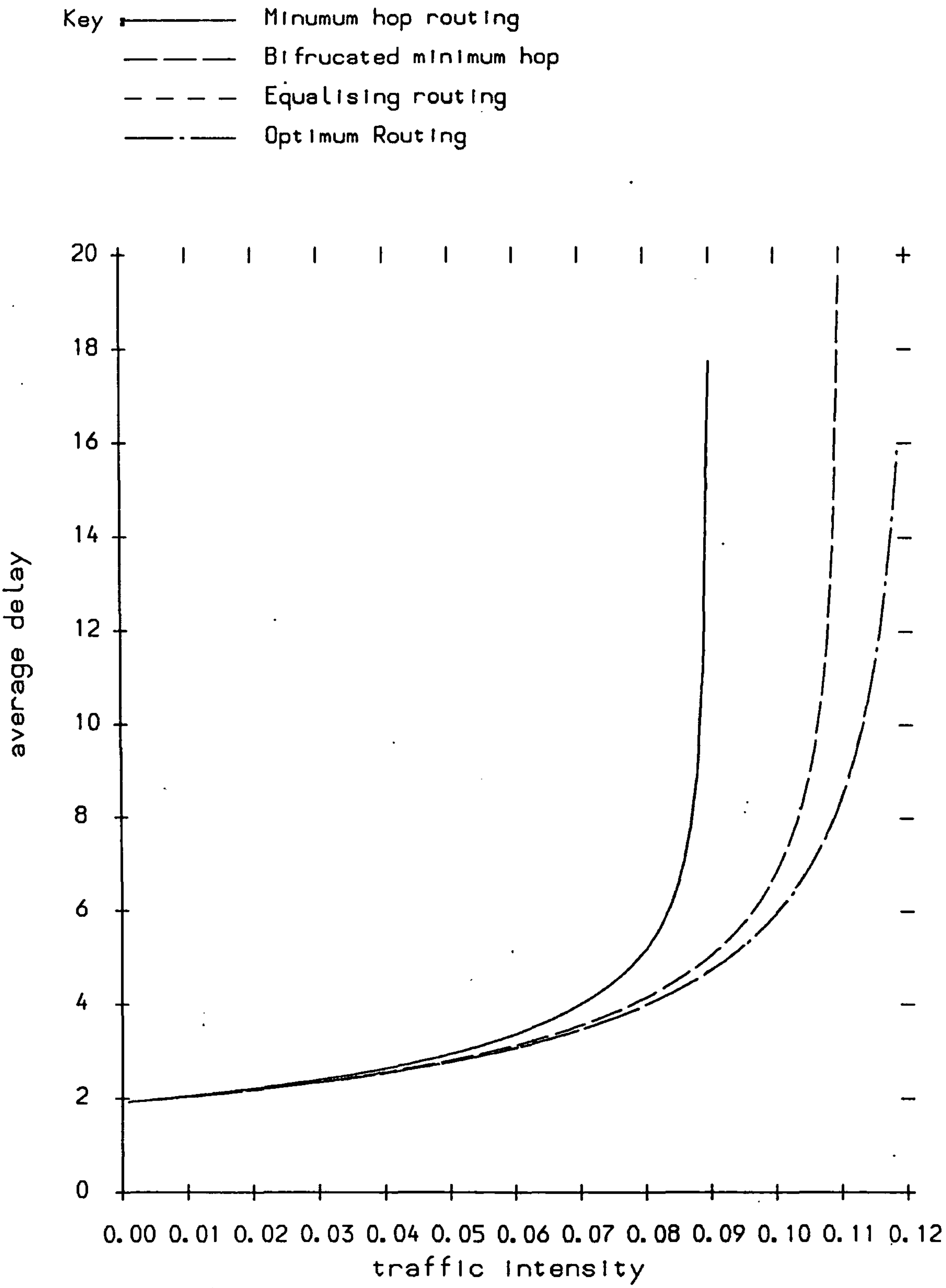


figure 6.14

average delay in 10 node network

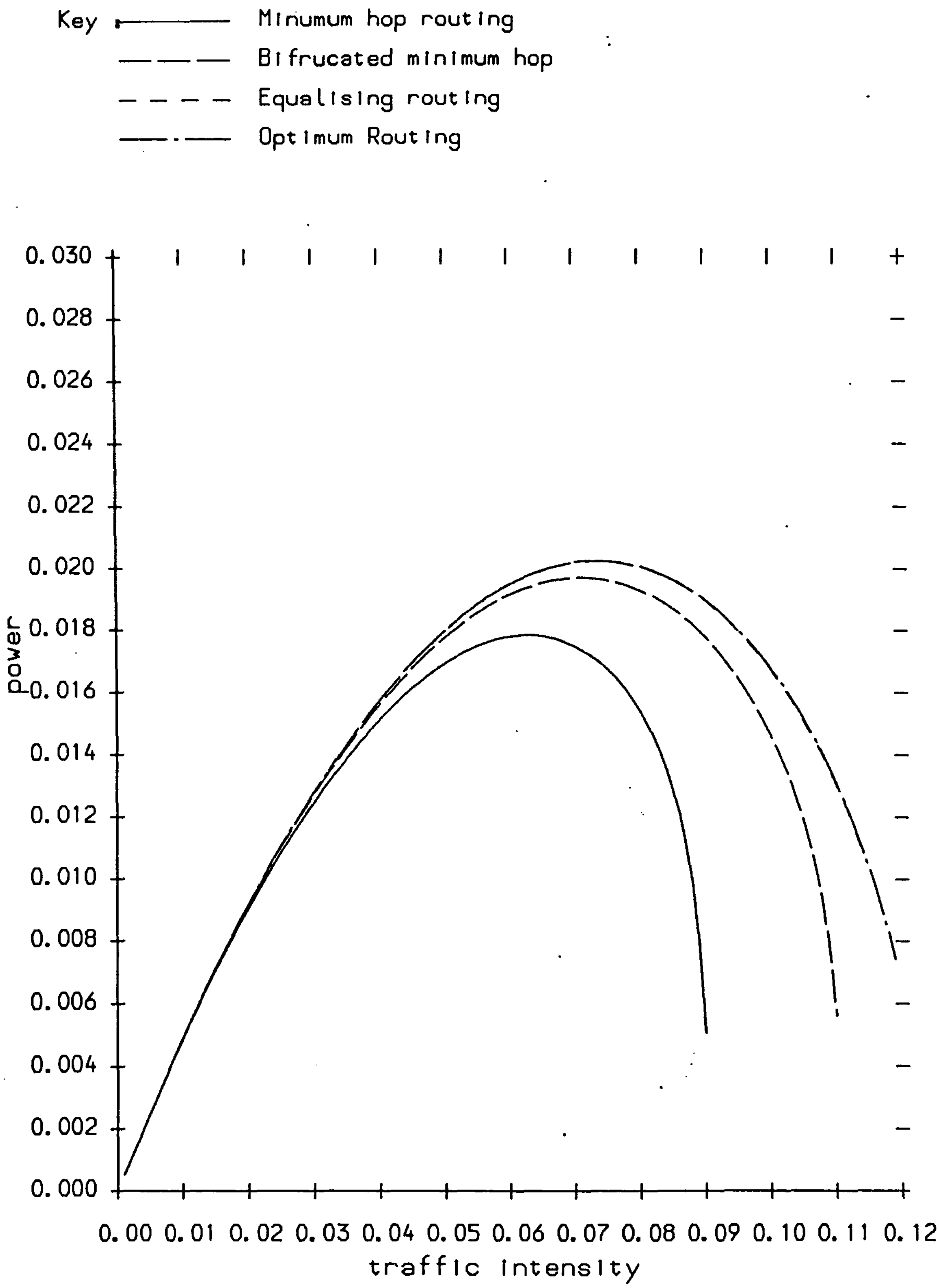


figure 6.15

power in 10 node network

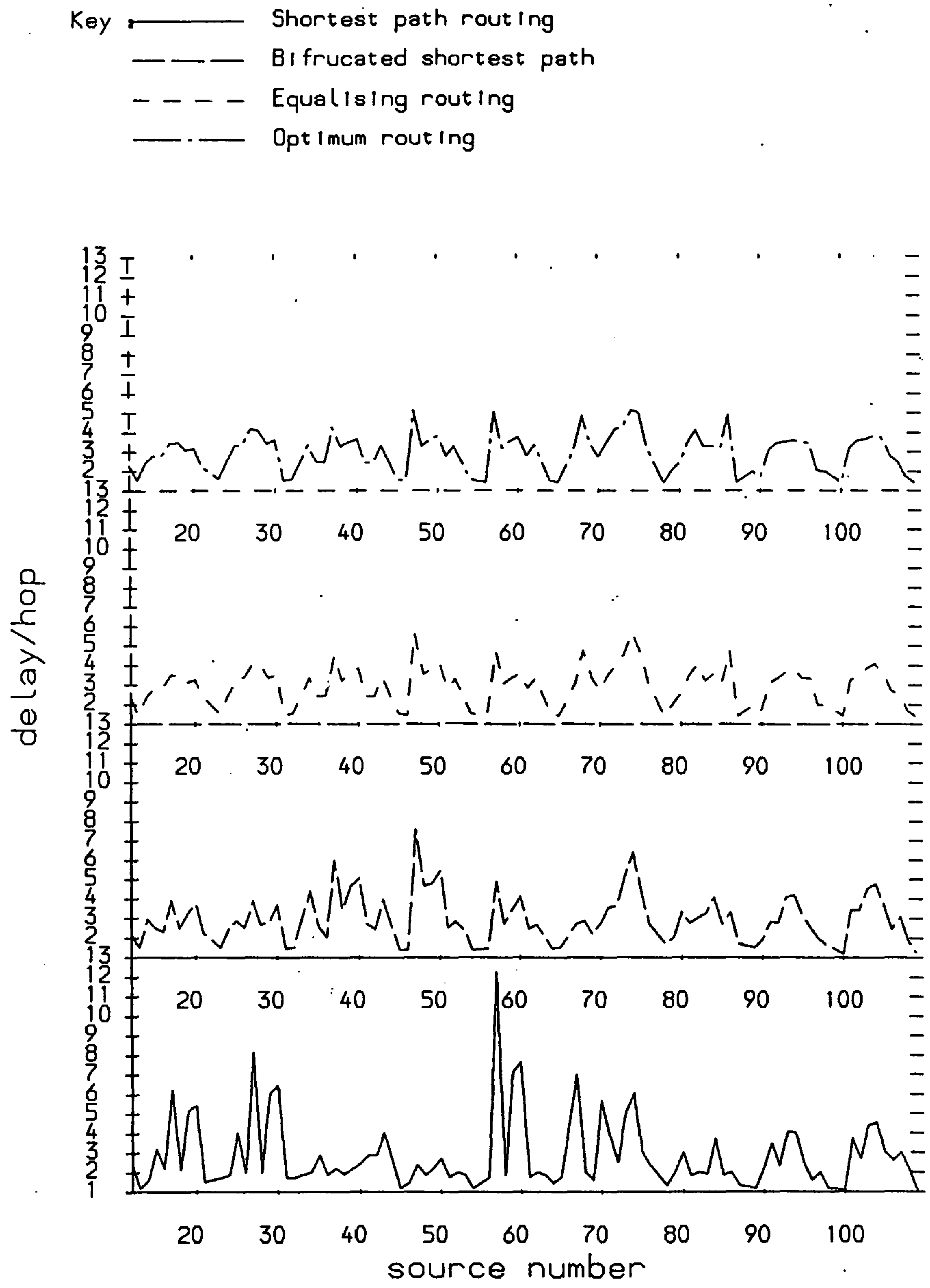


figure 6.16 : individual source delay/hop

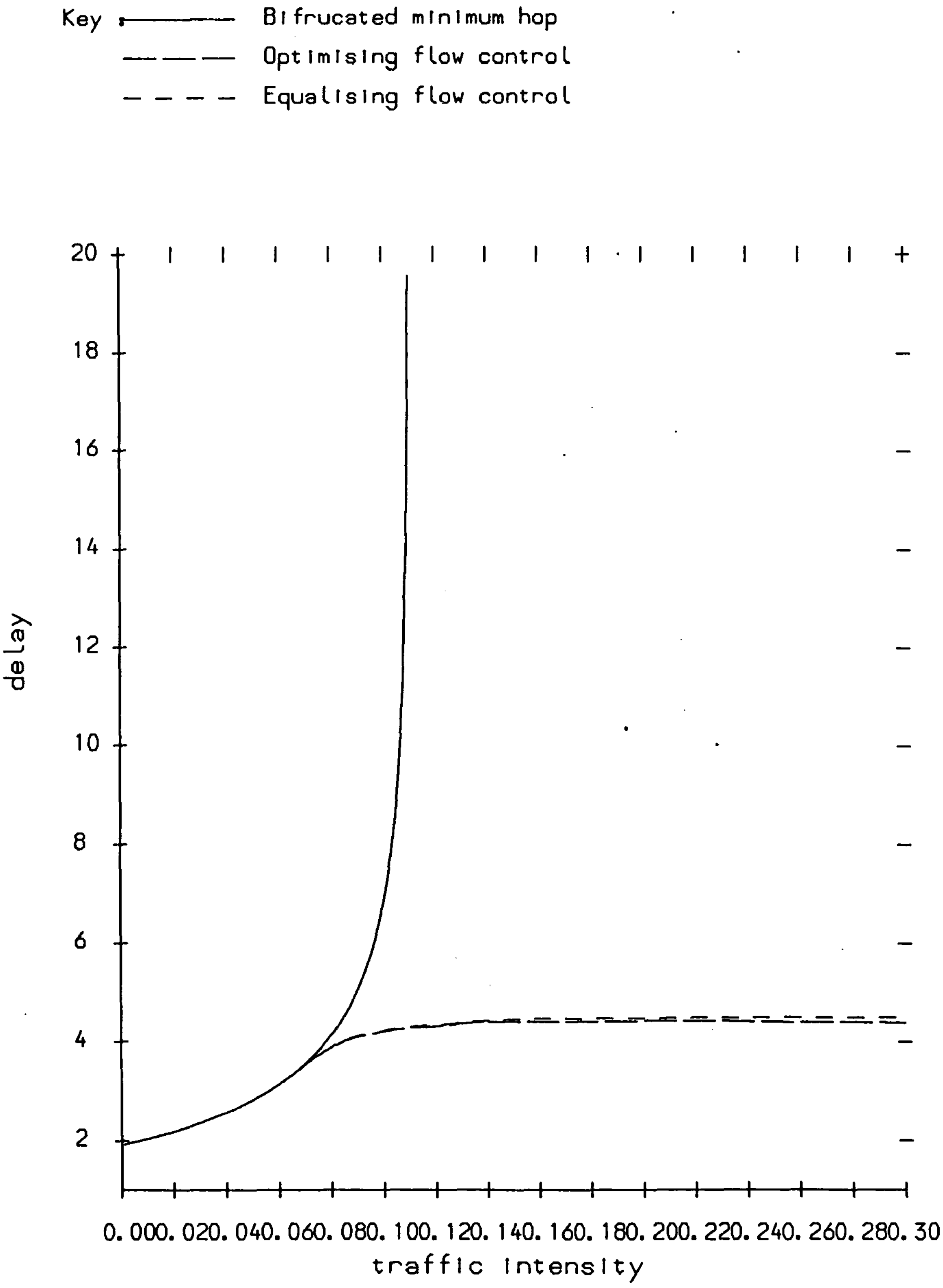


figure 6.17

adaptive flow control with bifrucated routing

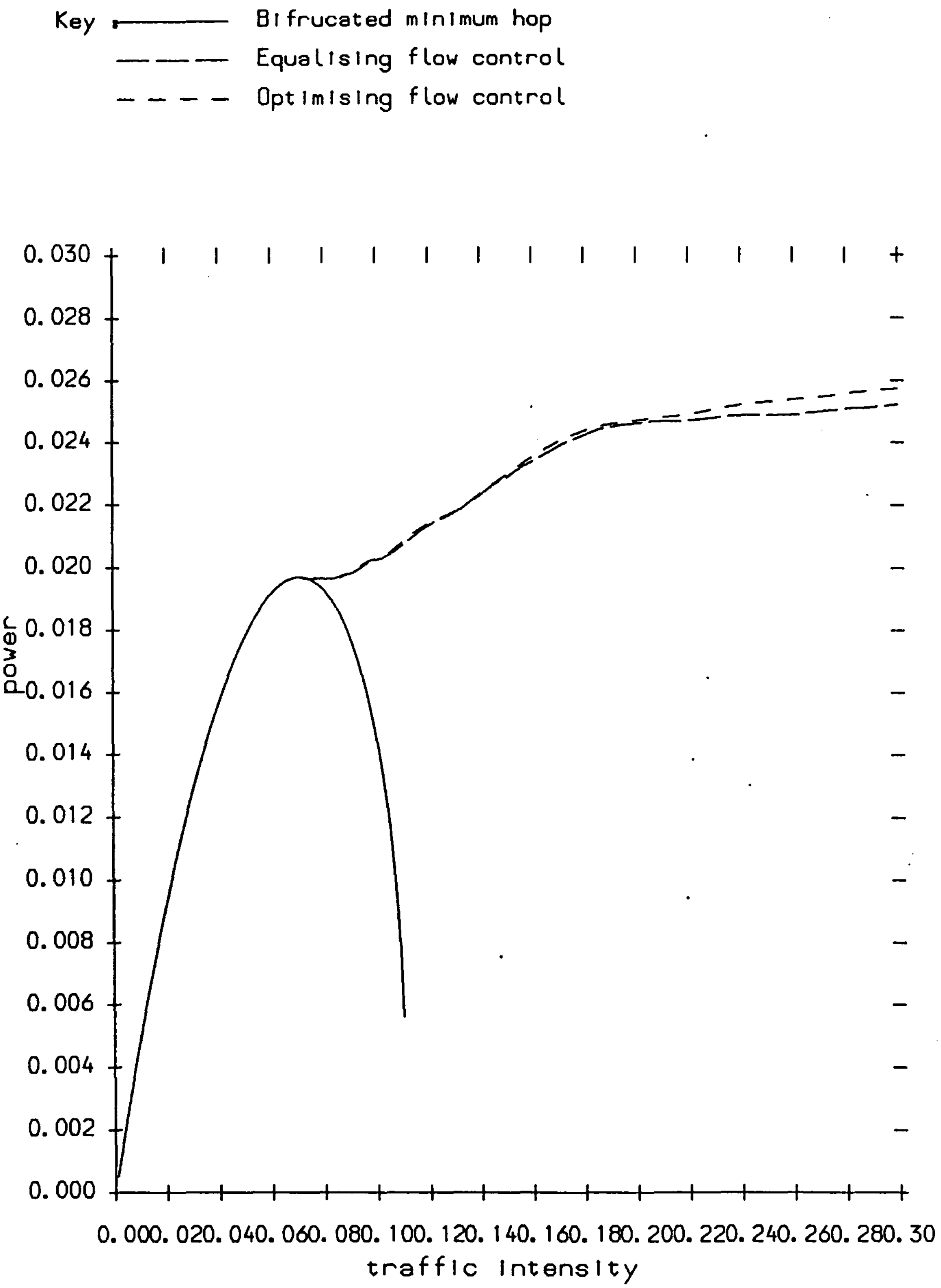


figure 6.18

power with adaptive flow control and bifrucated routing

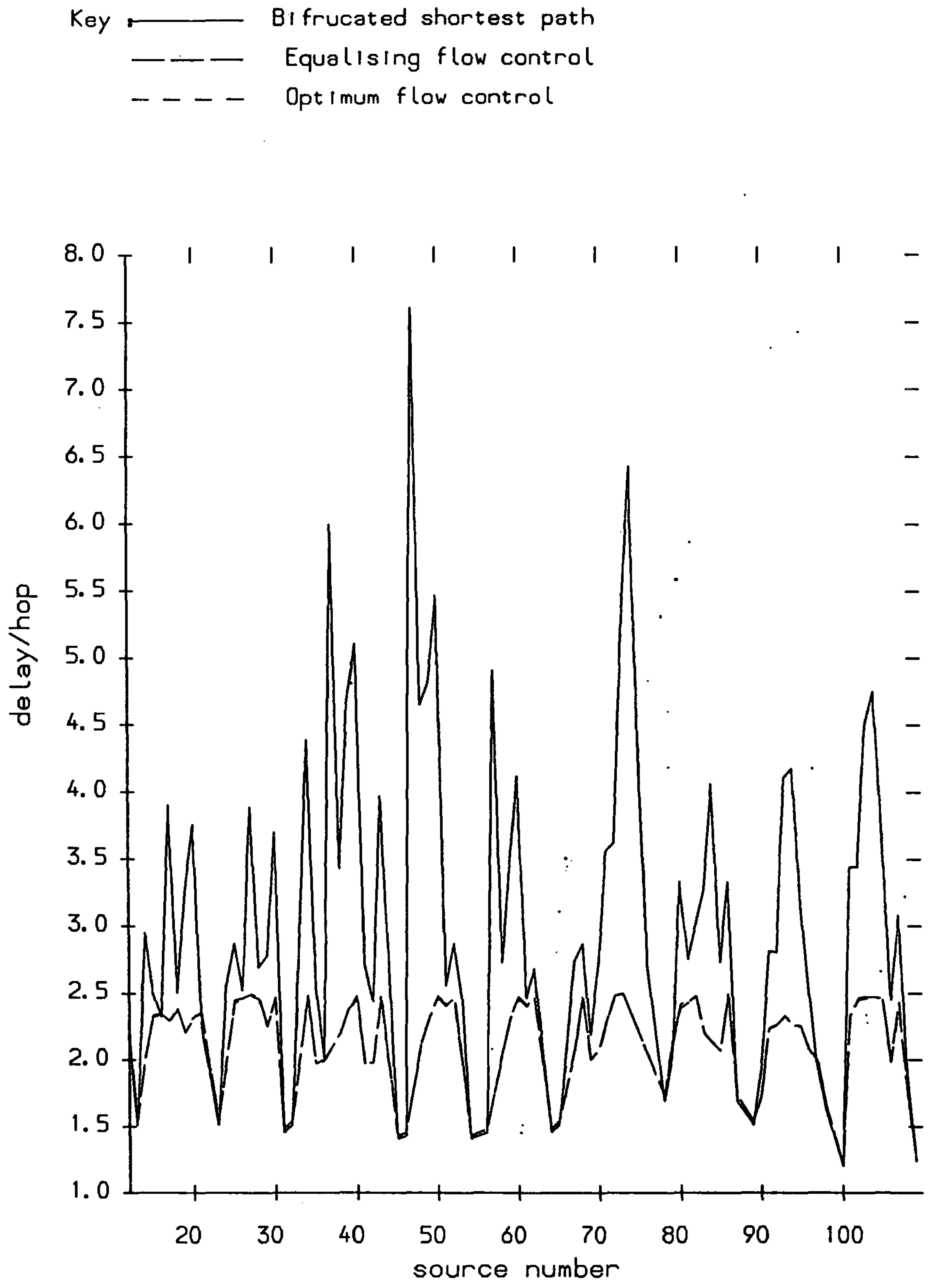


figure 6.19

individual delay with adaptive flow control

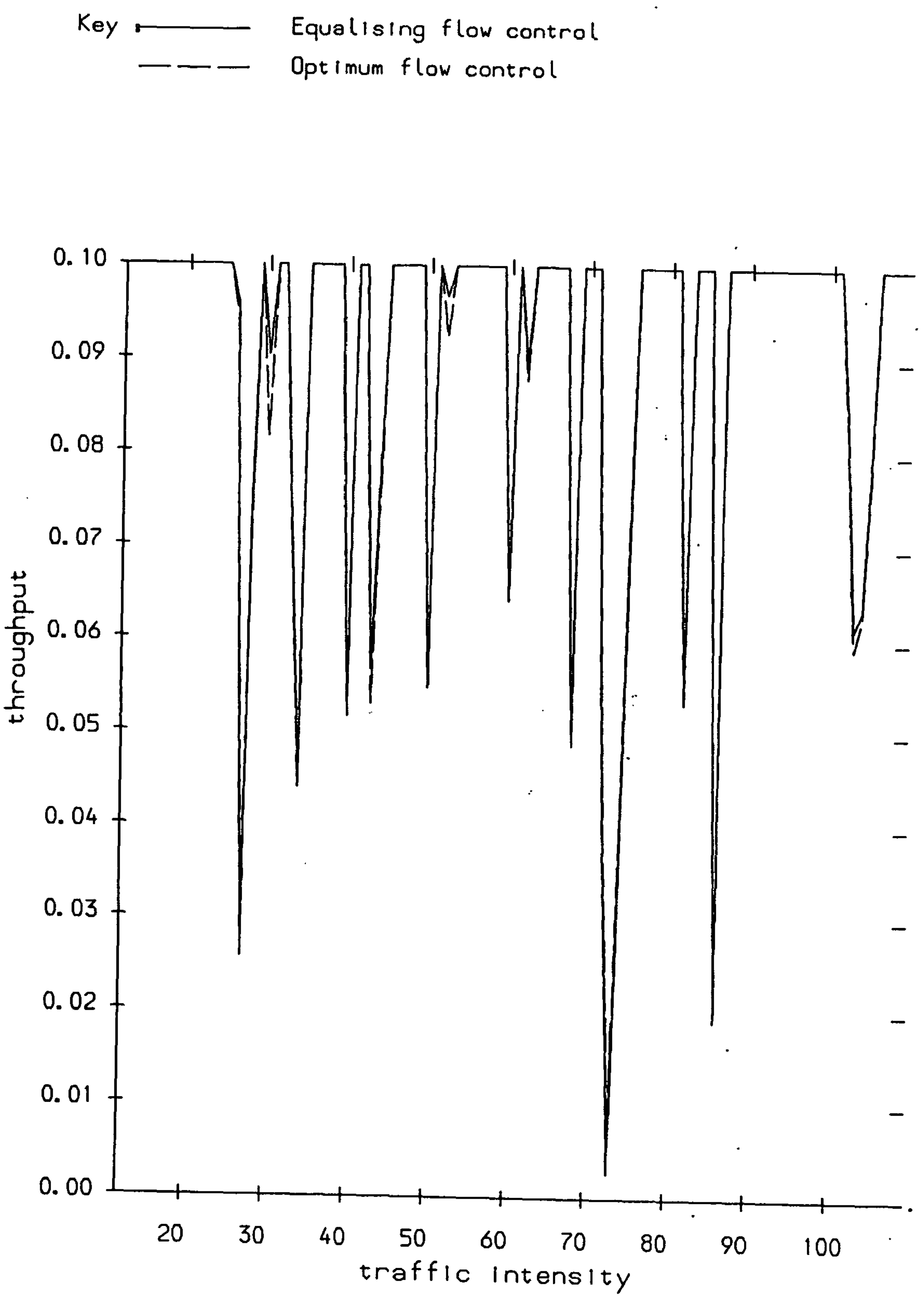


figure 6.20

individual throughput with adaptive flow control

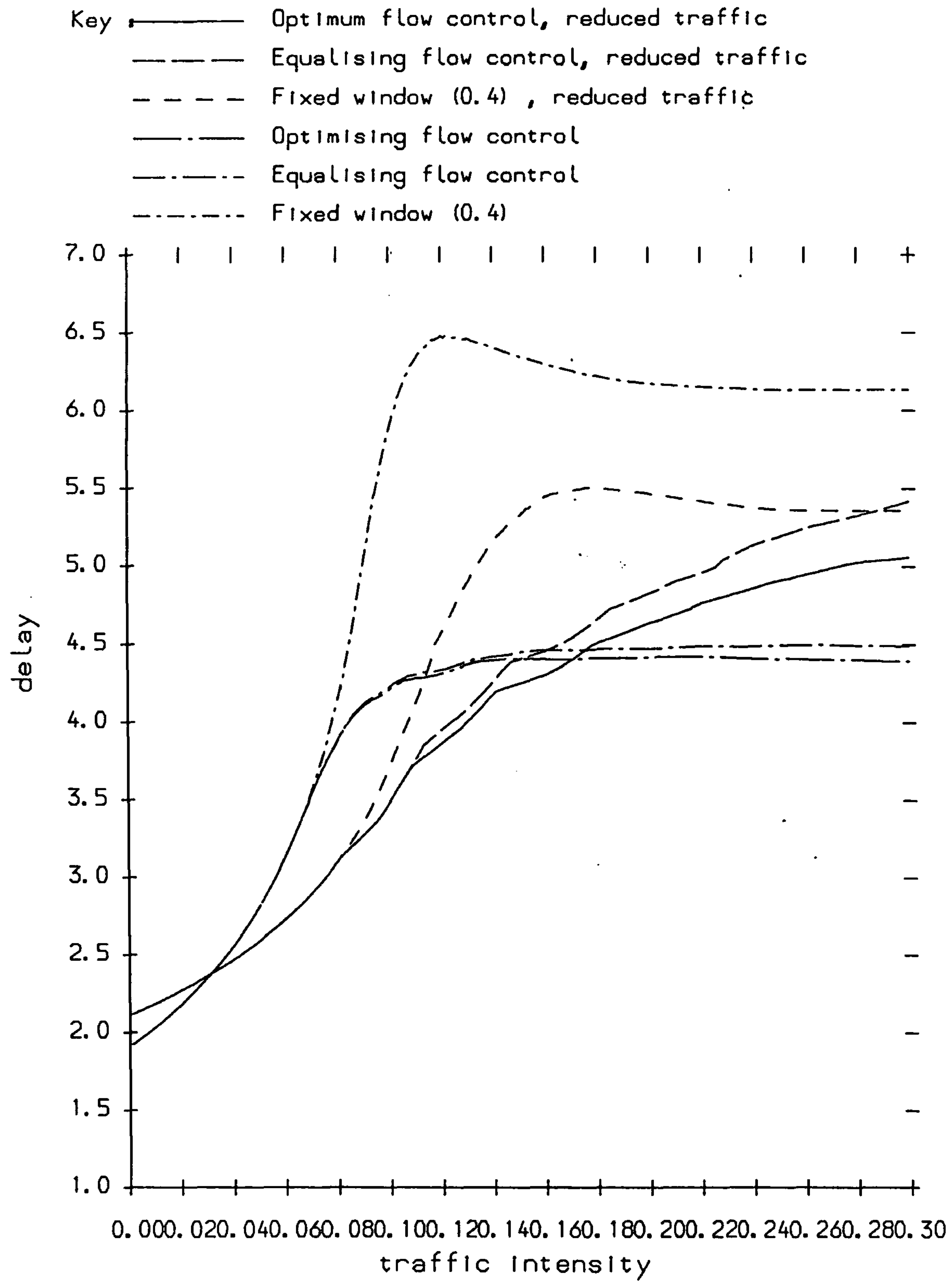


figure 6.21

comparison of fixed and adpative flow control policies

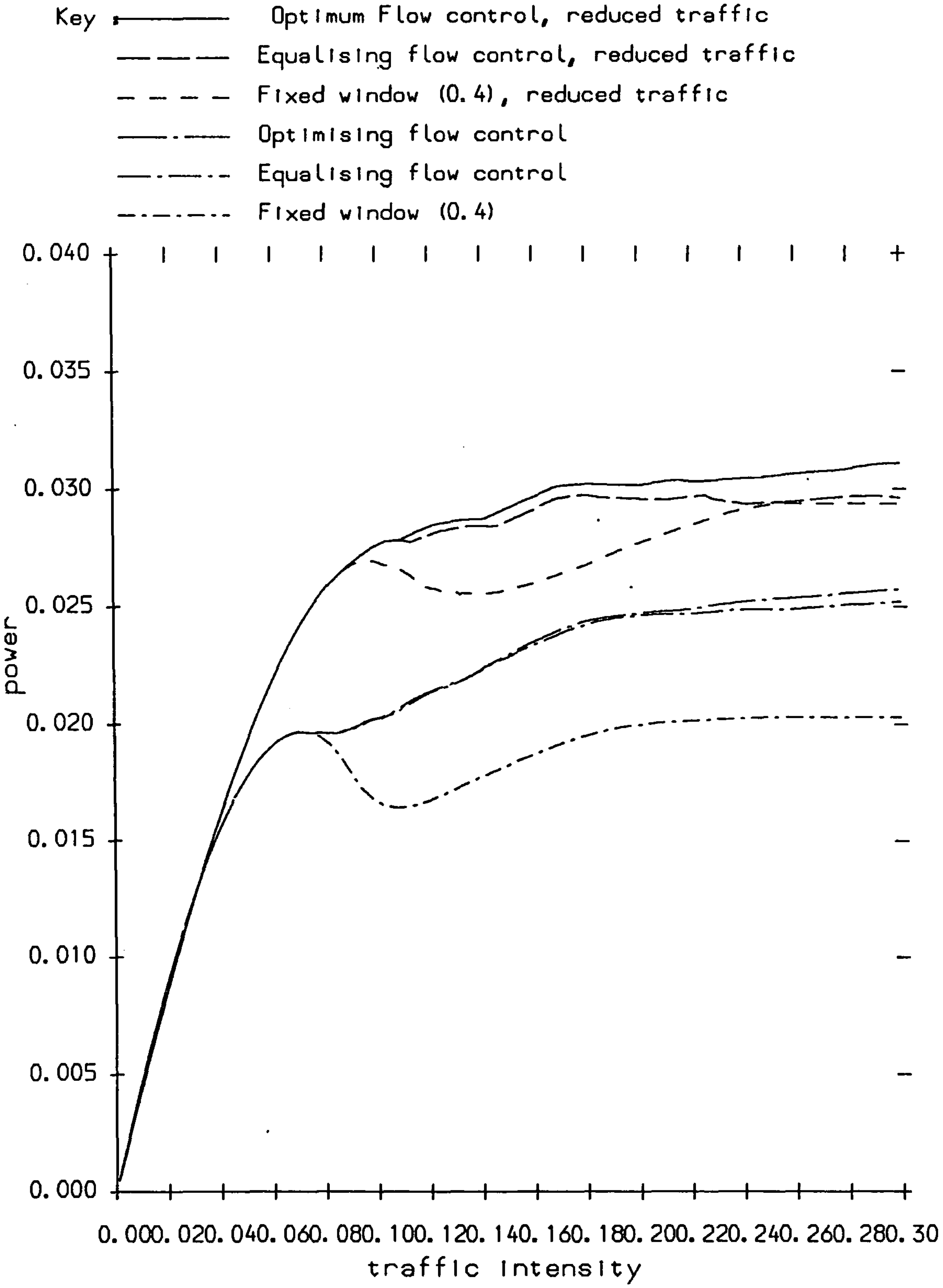


figure 6.22

network power using fixed and adaptive flow control

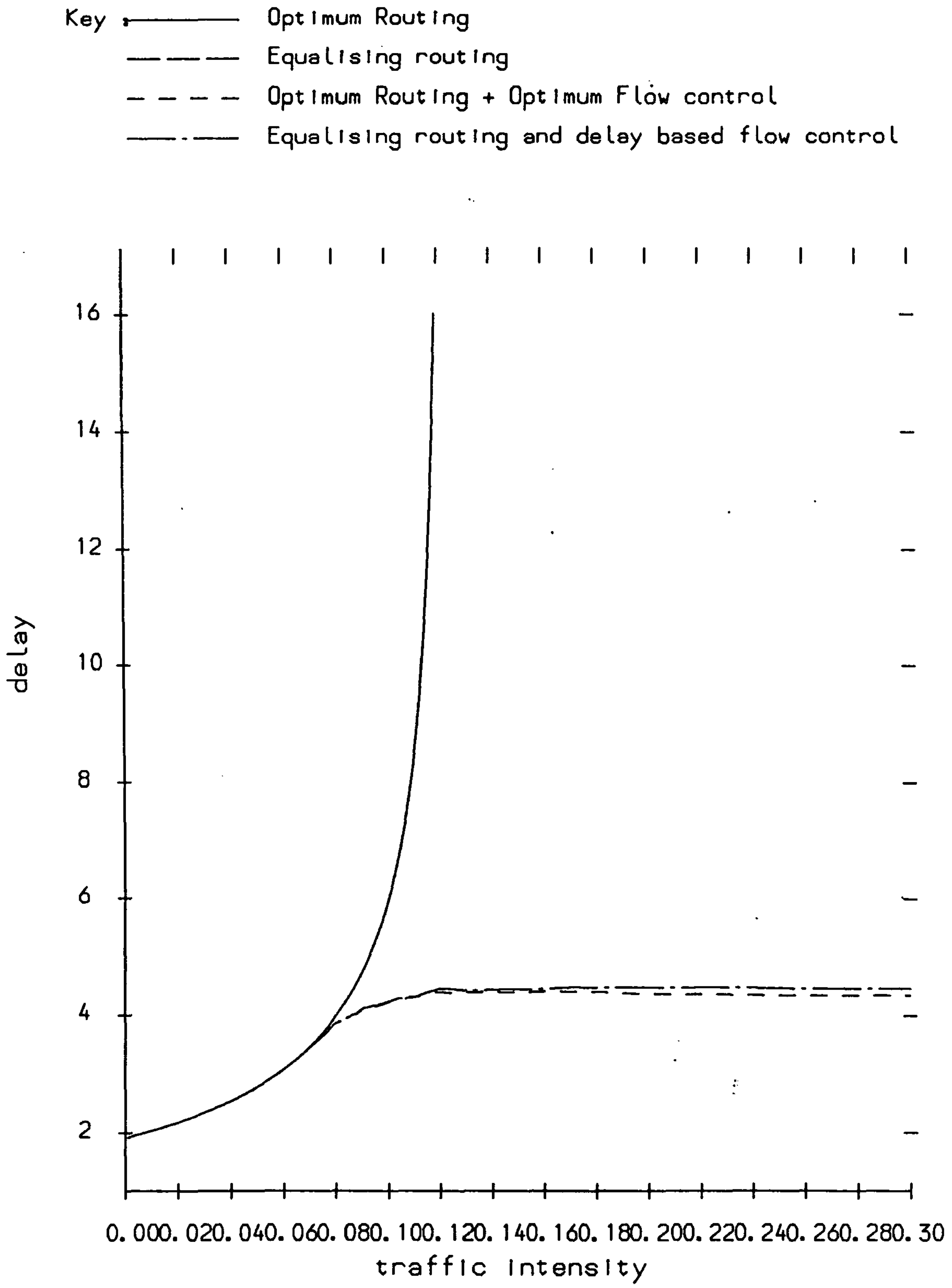


figure 6.23

delay using adaptive routing and adaptive flow control

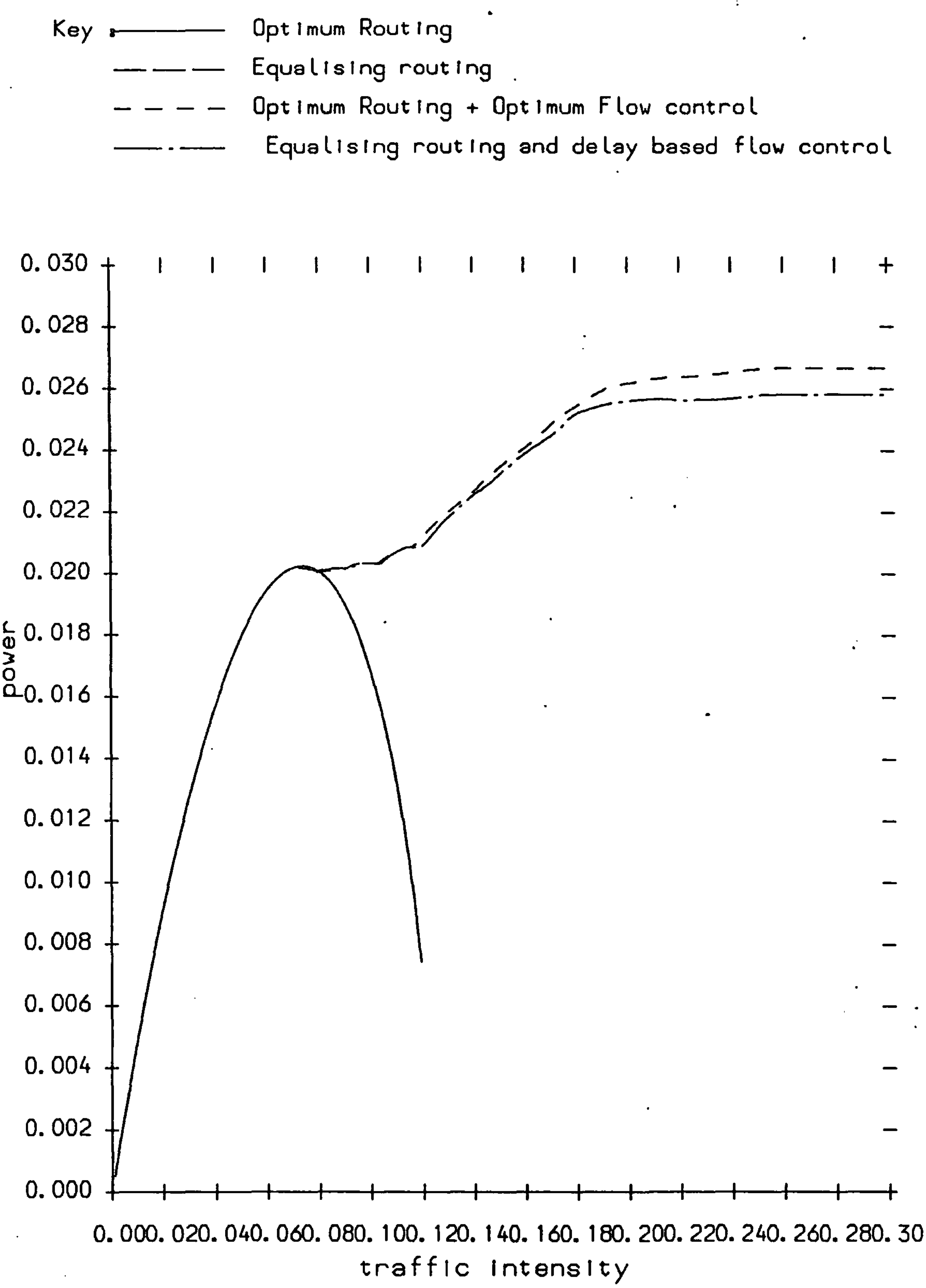


figure 6.24

power with adaptive routing and adaptive flow control

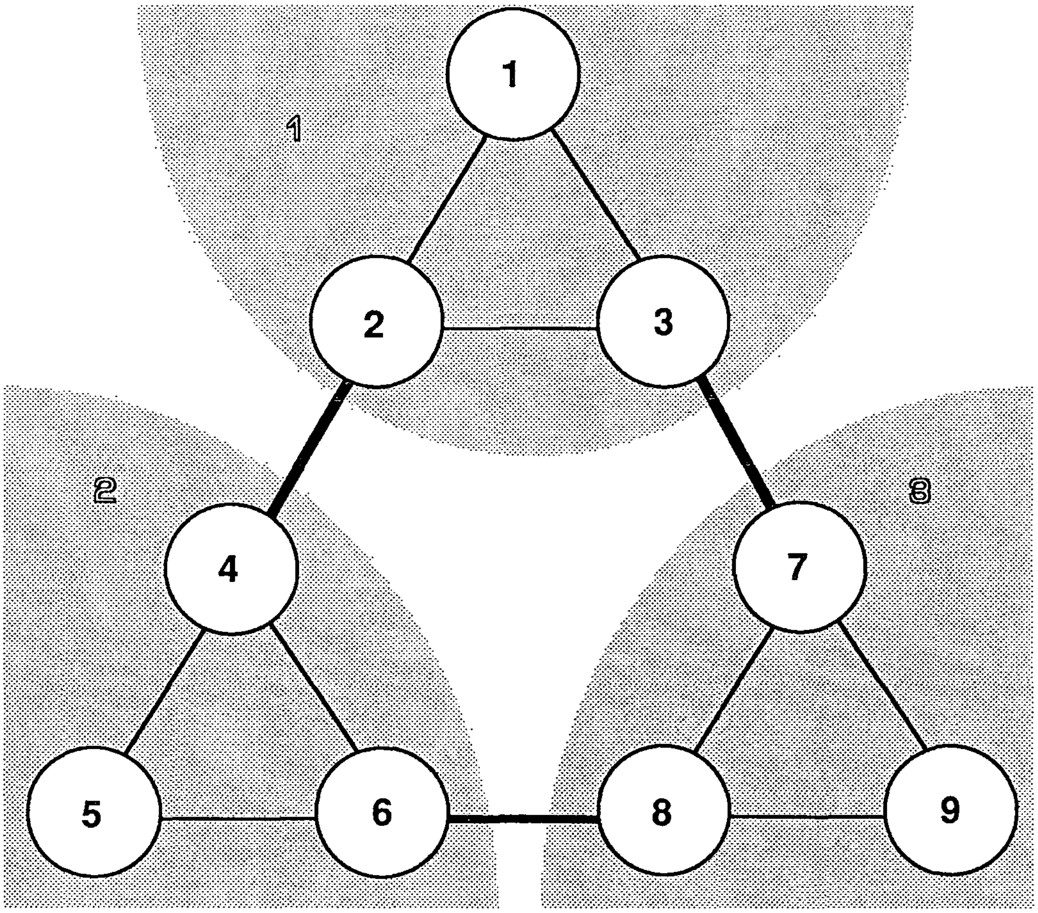


Figure 6.25 : time varying network

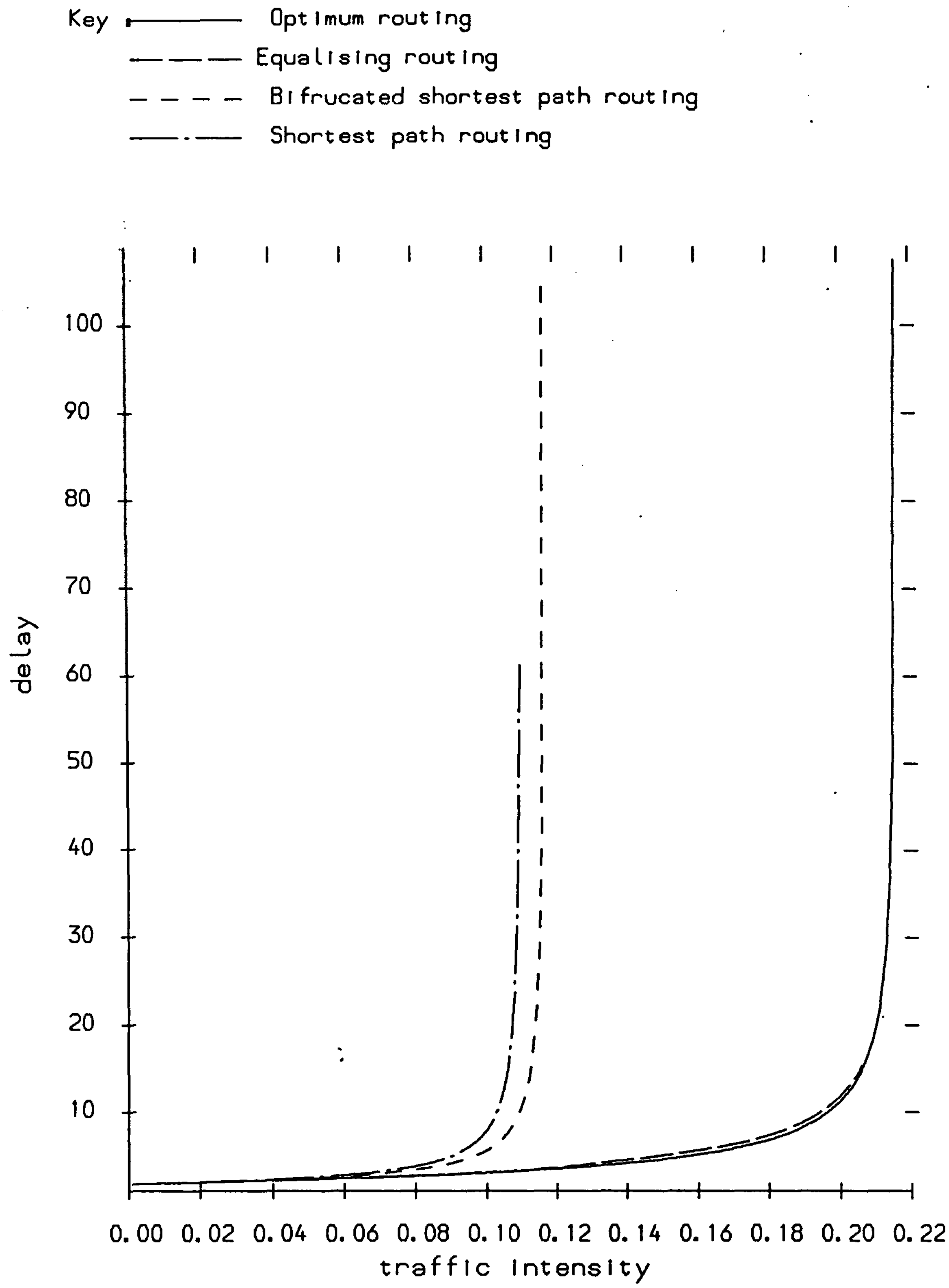


figure 6.26

delay in an international network

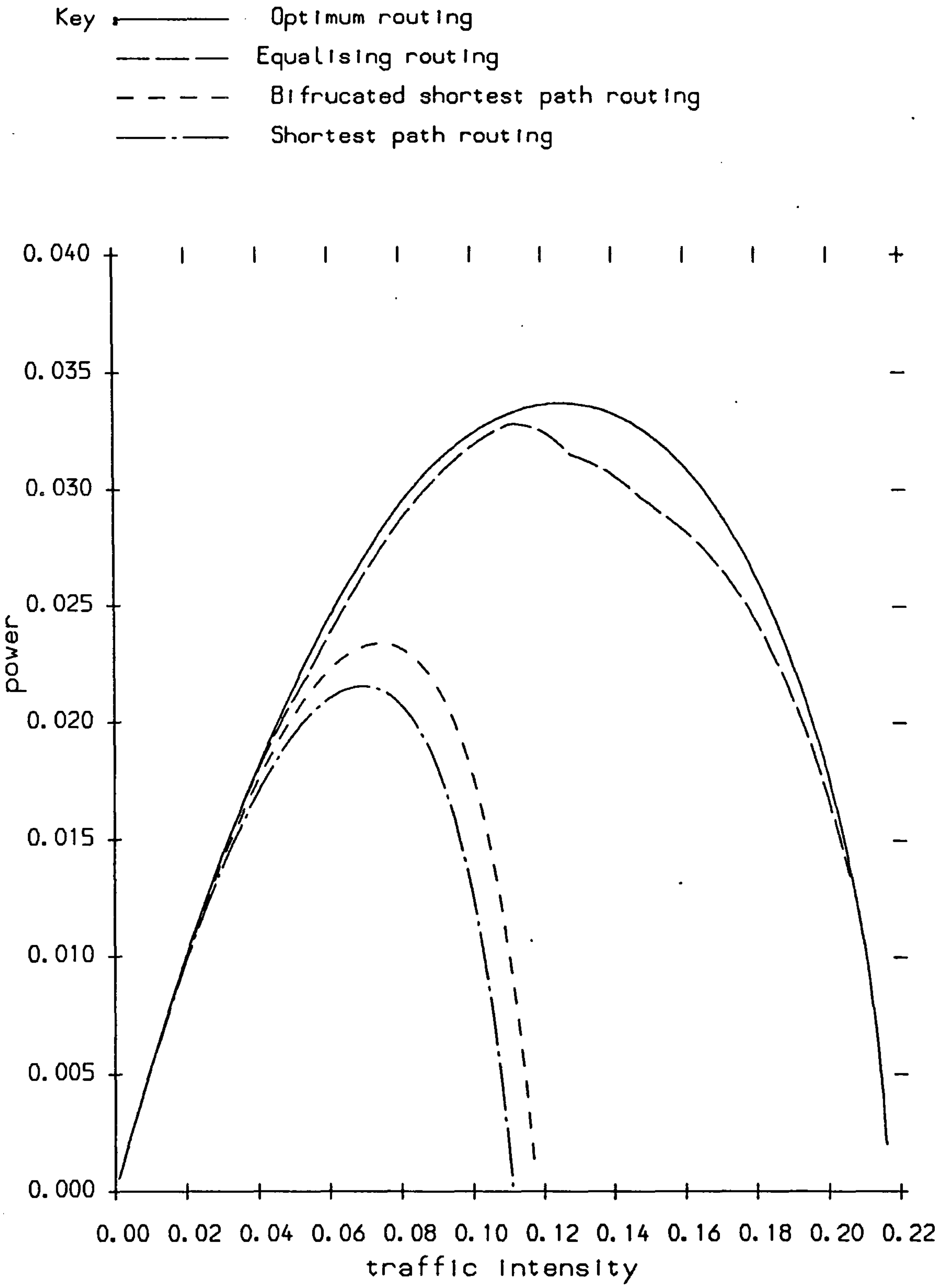


figure 6.27

power in an international network

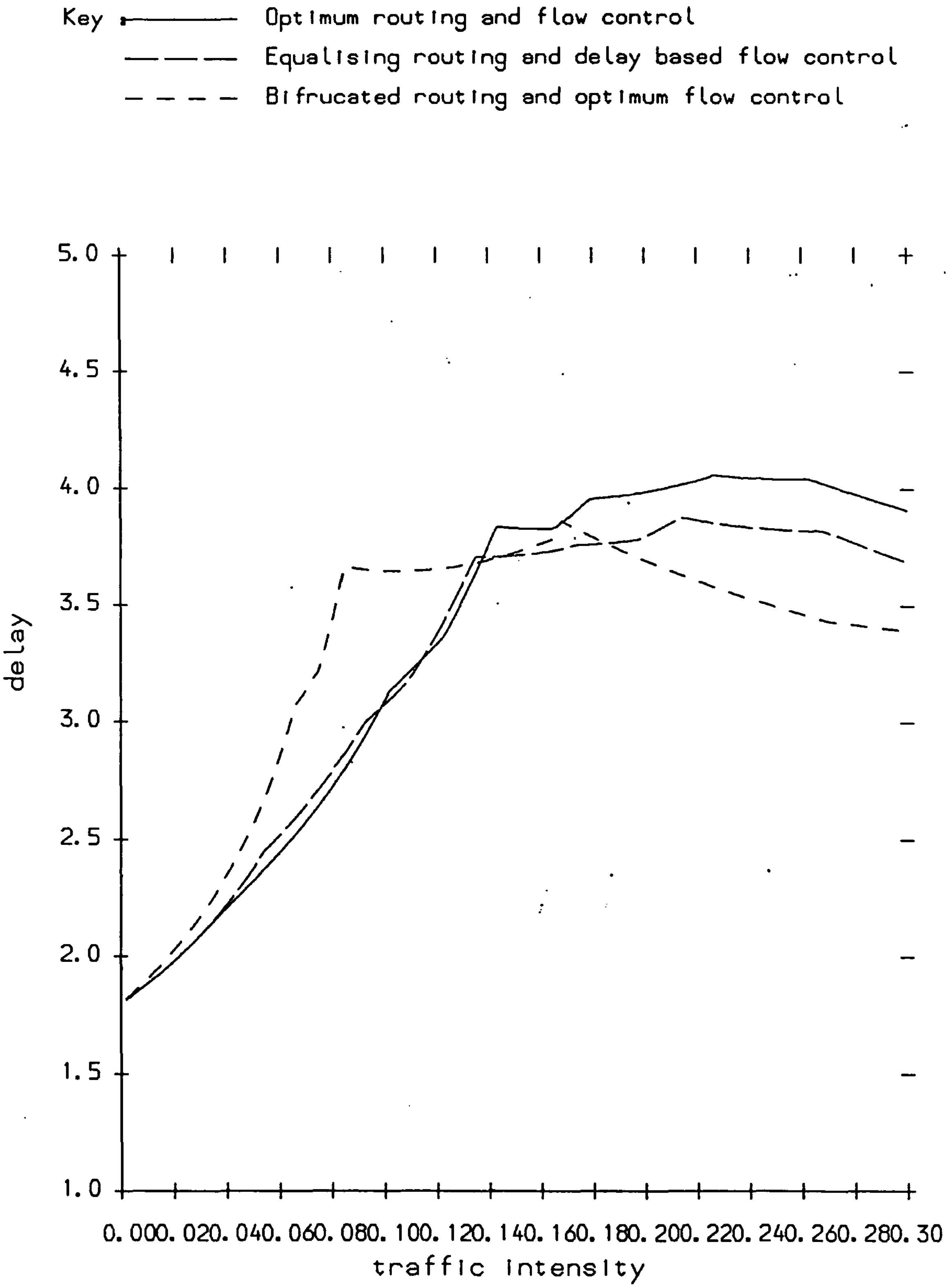


figure 6.28

delay in an international network with flow control

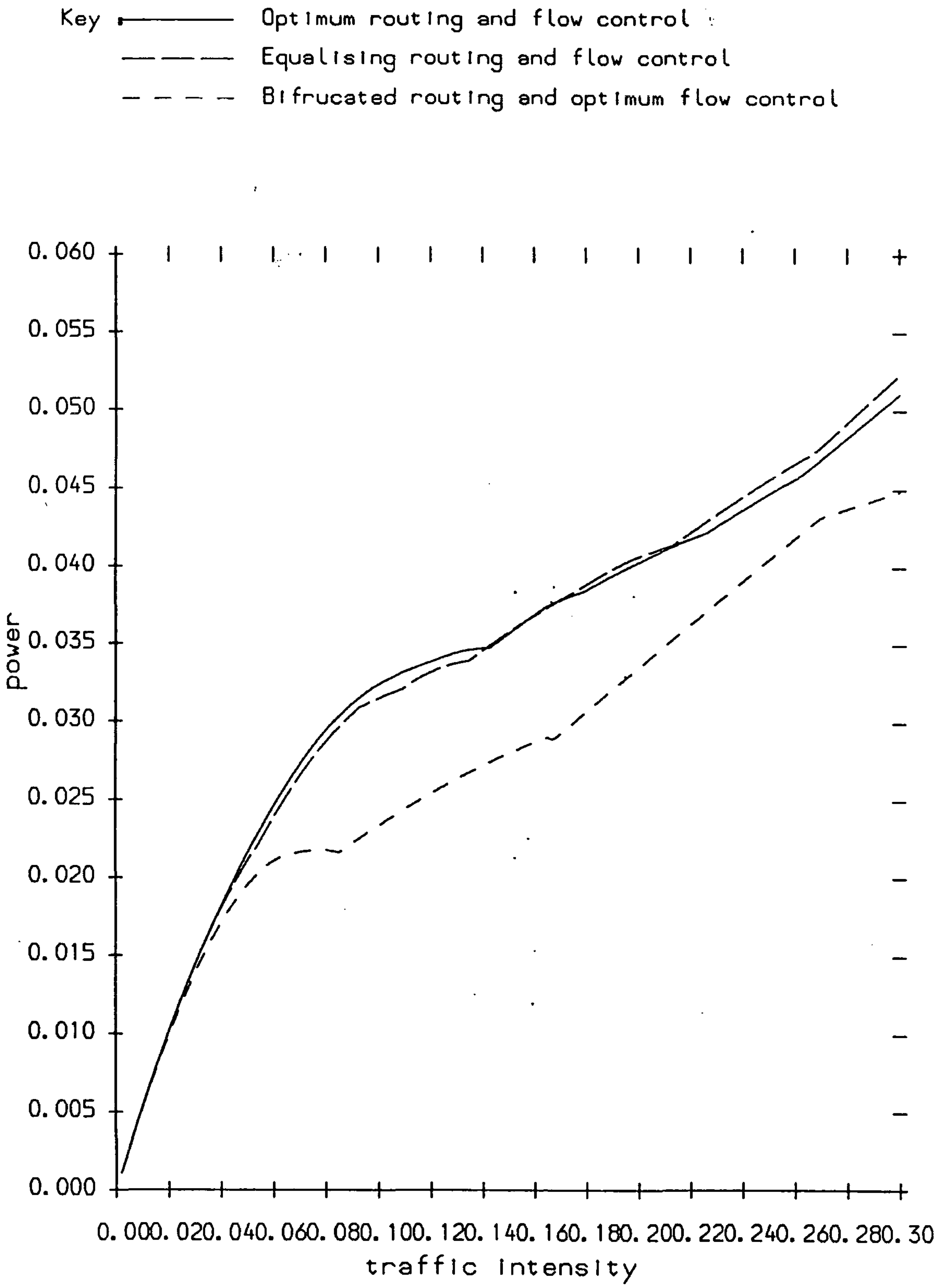


figure 6.29

power in an international network with flow control

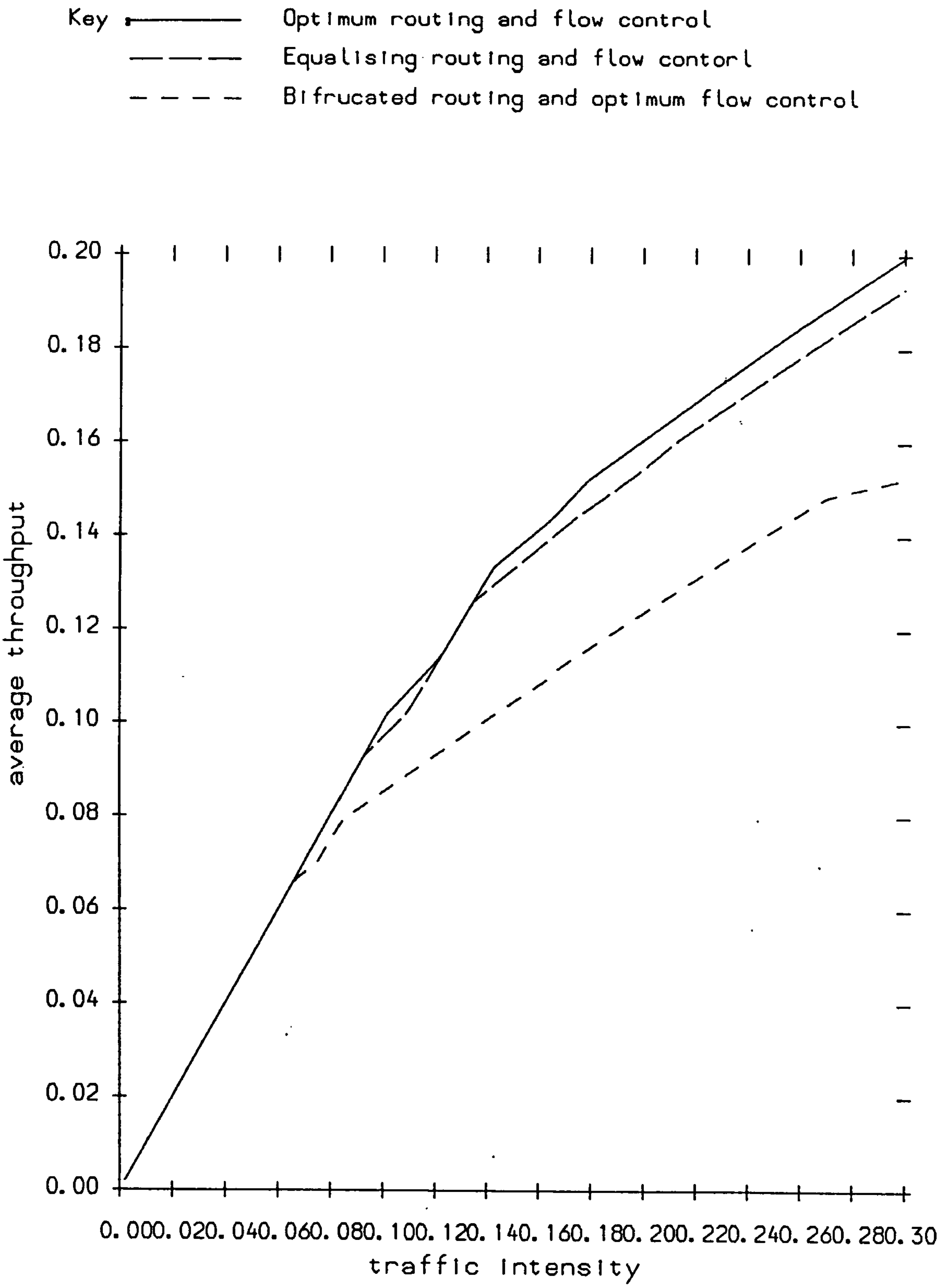


figure 6.30

throughput in an international network with flow control

Chapter 7
Conclusions and Suggestions for Further Work

7.1 Introduction

This chapter, like the detailed description of the work carried out which precedes it, can be split into three distinct sections. The first section summarises the major points of interest arising from the design and construction of the multi–processor simulator. The important differences and difficulties of simulating in a parallel environment in comparison with a sequential alternative are highlighted and the solutions adopted outlined. The major limitations and dependencies on the performance of the simulator are discussed with some comments on the optimisation of execution times for individual simulations and definitions of suitable indexes of performance for the comparison of code running on the simulator. In the second section the work done in modelling adaptive routing in fully connected circuit switched networks is summarised. Particular attention is given to the new results for the behaviour of multiple learning automata acting as routing controllers for the selection of alternatively routed traffic through the network, and their implications from the customers viewpoint. Suggestions are then made for the continued development of work in this area to consider features such as transient response to network changes and to broaden the range of networks to which the algorithm could be applied. Finally the work performed on the modelling of packet switched networks is summarised, the major conclusions drawn from the networks investigated re–iterated and areas of further work identified.

7.2 Multi–processor Simulation

In an attempt to overcome the simulation bottleneck in the modelling of wide area communication networks through simulation, given the analytic intractability of many networking problems, a multi–microprocessor solution was identified as a potential solution. By the effective utilisation of an array of processing elements, the conventional limitations of a single processors capability could be circumvented. This proposal was further strengthened by the appearance of the INMOS Transputer, a novel high speed processor which was designed specifically to communicate efficiently with other processing elements of the same family, in an arbitrary topological configuration. From studies of previous work in multi–processor architectures and by analysis of the problem for which the environment was to be designed, a hardware configuration was developed for the simulation environment. The key element in the design consisted of an array of Transputers each linked into an array of configurable cross–point switches. The switches could be dynamically programmed to connect the Transputers into a wide variety of topologies and produced a flexible medium into which the simulation code could be exported. A number of different ways were examined in the literature for the optimum distribution of the simulation code between the processors. The method chosen, as potentially the most profitable, was the allocation of one parallel task to each of the communication nodes in the network. At most therefore, the code for the simulation can be distributed amongst the same number of processing centres as there are nodes in the network. However this allocation also allows the network to be partitioned over a smaller number of processing centres, with each centre responsible for one or more parallel tasks. In its most concurrent state the code requires the simulation hardware to be configured in the same topological pattern as the original network. Communication between nodes in the simulation then takes place exclusively over the Transputers high speed, serial, point–to–point communication links.

The distribution of a network simulation over an array of distinct, loosely coupled microprocessors introduces a number of problems not usually encountered in a sequential equivalent, which uses only a single process on a single processor. The major areas identified from the work were access to the simulation when under execution, synchronisation between processes to maintain event ordering and the development of a software architecture within each process to avoid deadlock developing as the processes communicated. Access was achieved by the adoption of a 'snapshot' technique. The simulation code within each process runs up to a pre–arranged point where access was then allowed through a linear chain of communication links which linked each node in the network. Synchronisation was maintained by one of two methods, comparable to synchronous and asynchronous clock mechanisms in sequential simulations. Each generates a transmission overhead to synchronise the nodes at either end of every link, which becomes more significant as the traffic loading on that link becomes lighter. Deadlock between two processes is avoided by separating the input and output routines which initiate transmission over the communication links from the simulation code which implements the model of a switching exchange. The interface routines generate the equivalent of software interrupts when they wish to access or be accessed by the model code, and wait to be serviced.. The model code then consists of a number of subroutines which are executed whenever the relevant interrupt is generated.

As previously mentioned the fundamental reason for the adoption of a multi–processor architecture was to reduce the execution time of wide area communication network simulations. Conceptually the approach adopted offered the advantage of producing simulations which were independent of the size of the network itself and instead a function only of traffic density and the distribution of traffic over the network. However the extent to which this performance is realised is dependent on the relative loadings of the processor and communication links. If the processor is heavily utilised in relation to link communication, termed processor limited, such communication is transparent as cycle stealing DMA techniques are employed. Under these conditions the CPU processing rates of the network nodes may be used to form a summation defining the total processing power going into the simulation. Alternatively if the reverse is true and the process is communication limited, each processor will be idle for a percentage of the simulation while it waits for the next transmission. In this case full CPU utilisation is not taking place and the execution times will fall below that expected. The question of whether a simulation is processor or communication limited depends on a number of factors including processor and communication link speeds, model complexity, data transfer demand and synchronisation overheads. Work is currently in progress to evaluate the effect of each of these constituents, which it is hoped will lead to the design of a second generation of simulator which will further improve simulation execution times.

7.3 Circuit Switched Networks

The work done involving the performance analysis of fully connected circuit switched networks can be divided into two major components which can be treated separately. The first component covers the development of analytic models for the prediction of network performance under a range of fixed and dynamic algorithms. The second component consists of the conclusions drawn by application of the models to specific network configurations and the relative performance of the routing algorithms under these conditions.

By extending previous analytic modelling on purely symmetrical fully connected networks, including trunk reservation, a model was formulated to calculate the performance of an asymmetrical network of arbitrary traffic and trunk capacity for fixed routing strategies such as random routing or automatic alternative routing. The model allowed the calculation of the major parameters of interest in the network, trunk utilisation and individual traffic source blocking probabilities. This basic model was then extended to produce further models for a number of adaptive routing policies, namely DAR, L_{R-I} and proportional routing, the latter being derived from the Bell–Northern global routing strategy. For each strategy a set of equations are formulated for each traffic source connecting the distribution of overflow traffic over alternative paths with a performance measure of the success for each selection. The performance measure, different in each case, is a result of the algorithm directly or of mathematically predicted behaviour.

Analysis of a simple four node network without a trunk reservation parameter, using the analytic models, divides the routing algorithms into two categories. The first category labelled 'competitive' includes DAR, random routing and proportional routing. In each algorithm the routing process dividing traffic for each source acts independently of all other routing processes in the network, reacting only to trunk utilisation. The second category, into which the L_{R-I} algorithm falls, labelled 'co–operative' routing, exhibits different behaviour. No independent solution can be found which satisfies the mathematical constraints. Instead the network routing processes act together to produce a solution in which each individual process is a component. The results further show that in acting in this manner the algorithm serves to equalise, or minimise the deviation in, the grade–of–service of each individual traffic source. This is unique in the algorithms studied in this work. The 'competitive' algorithms make no attempt to introduce fairness in this way, consequently the GOS for each source is determined by the position of that source in the network. This often leads to large differences between individual GOS and the average value of the network. Simulation results agree broadly with the conclusions drawn from the analytic models although there are numerical disagreements which derive from the validity of some of the approximations in the mathematical models.

As is widely known, the addition of a suitable trunk reservation parameter greatly improves the performance of the network under overload conditions, This is achieved by the reduction of bandwidth available to alternatively routed calls. This does not change the fundamental behaviour of the algorithms, but in particular reduces the ability of the L_{R-I} algorithm to reach its equilibrium, maximising fairness in the network. In the examples of structured asymmetry studied all of the adaptive algorithms displayed very similar behaviour under tight flow control and again simulation studies of the same network configurations produced the same broad agreement, strengthening the case for the validity of the models for comparative analysis.

The third area of study, instability in the blocking probability of the network gave interesting but conflicting results. The analytic work suggests that there exists a well defined band of instability for symmetric and a range of asymmetric network configurations. However the region over which this behaviour occurs is one in which the mathematical assumptions made by the model are not valid. Simulation results for the four node symmetrical network suggest only a gradual rise in blocking probability over the region concerned, with no clear evidence of unstable behaviour. This does not mean that predicted instability does not occur, merely that the analysis of the simulation done in this work did not find proof of its presence.

The work has presented both analytic and simulation based modelling of the behaviour of a number of adaptive routing algorithms for fully connected circuit switched networks in dynamic equilibrium. In particular the behaviour of a network using a routing algorithm based on learning automata theory has been explored. New results have been discovered into the algorithm's method of operation which suggests direct benefit network customers by attempting to equalise service. Two areas of directly related work in circuit switched networks suggest themselves for further investigation. The first is the study of dynamic routing algorithms, such as these studied, in a transient environment. Important questions such as the ability of an algorithm to track traffic fluctuations and react to step changes in network patterns require detailed investigation to determine the factors affecting the convergence rate and methods for its optimisation. In considering the L_{R-I} algorithm, one key factor is the reward parameter used to update the probability distribution on each successfully routed call. There is an obvious trade–off in using a fixed value between accuracy and convergence speed and suggestions have been made for the use of an adaptive step size, however this area is still open for much further study.

The second area of study is in the development of a suitable adaptive routing algorithm for sparsely connected circuit switched networks of arbitrary topology. In a network there is often no direct link between the originating and destination nodes in the network and the problem becomes one of finding a sensible multi–hop path through the network. Algorithms for either path–based or link–based algorithms incorporating learning automata theory can be formulated based on the same concepts as their fully connected relations. The analytic complexity of networks of this type is formidable and the initial path in the analysis of adaptive algorithms within this sphere may well lie in the further development of the simulation environment to encompass networks of this type and allow such studies to be undertaken.

7.4 Packet Switched Networks

In the work covering packet switched networks attention was focused on the application of an adaptive routing algorithm based on learning automata theory to sparsely connected network topologies. The work then went on to consider the interaction and overall performance of the algorithm with the addition of a simple adaptive flow control policy to the routing strategy. Much of the work consisted of analytic modelling of the packet switched networks with simulation used as a tool to validate the assumptions made in both the queueing models and the models used to predict the behaviour of the learning automata. To construct the models work was brought together from a several authors and integrated into a number of related mathematical models covering a wide range of routing and flow control mechanisms including fixed, adaptive and optimum policies. This powerful modelling capability was then applied to generalised examples of typical packet switched networks for the comparative analysis of network performance under the different strategies.

The network modelling demonstrated that, for the networks studied, the use of a learning automata based routing algorithm gave almost identical performance to the mathematically optimum strategy. Further when it was combined with a simple delay based flow control strategy, the resulting performance was, once again, almost equivalent to the combination of optimal routing and optimal flow control. The advantage of such a combination over its optimal counterpart is that both parts of the overall strategy are easy to implement using trivial numerical calculation in an entirely distributed environment. The information required for the implementation of the algorithms is already present, or can be easily embedded, in existing network protocols. Simulation studies included in the work support the models validity and suggest that the comparisons made between the various mathematical models can be translated into real measures of network performance.

As with the work on fully connected circuit switched networks, these results cover only the behaviour of packet switched networks in dynamic equilibrium. Given the highly statistical nature of packet switched traffic characteristics, it is important that both routing and flow control algorithms can react quickly to changes in traffic distribution and magnitude. Once again the behaviour of the learning automata will depend critically on the selection of the learning parameter embedded within the algorithm. However another possibility for the enhancement of such transient conditions is attracting increasing interest. A hybrid strategy, incorporating a centralised component with a global knowledge of the network state, could provide invaluable information to the distributed portion of the routing algorithm, accelerating the convergence of an adaptive algorithm to major changes. Typically the centralised component would present each node with a selection of routes to each destination, these being a subset of all those possible. If this is done periodically, either synchronously or asynchronously, this could optimise much of the initial work in finding the viable subset of routes for a given network configuration. Once this was accomplished the distributed portion of the algorithm could optimise the network performance by selecting the division of traffic over each of the preferred routes.

Finally the work done on packet switched networks provides a basis for research into the emerging broadband ISDN network architectures. Work is already progressing in this area at the University of Durham, in conjunction with both British Telecom Research Laboratories and several European partners in a RACE initiative. The work centres on the use of Transputer arrays for the simulation and emulation of Asynchronous Transfer Mode (ATM) transmission within B–ISDN. This field promises to be an exciting area of research in the near future and one in which Durham is well positioned to make a significant contribution.

References

[1] H.T. Mouftah and K.S. Shanmugan, Computer Aided Techniques for Communications Systems Engineering, IEEE Communications, Vol. 25, No. 7, July 1987, pp 48-54.

[2] M.G. Hartley (Editor), Digital Simulation Methods, IEE Monograph Series 15, Peter Peregrinus Ltd., 1975, ISBN: 0-901223-50-6.

[3] M.P. Papazoglou et al., Designing a Parallel Simula Machine, Computer Design, October 1983, pp 125-132.

[4] R. Lehnert, A Special Processor for Fast Simulation of Queueing Networks, ITC-9, Spain, 1979.

[5] M.Barel, A Flexible High-Performance Multiprocessor for Data Network Simulation, ITC-10, Session 3.3, Paper 9, 1982.

[6] R. Kain et al., CHIMPNET: A Networking Testbed, Computer Networks 3, December 1979, pp 447-457.

[7] A.Toda et al., A Parallel Processing Simulator for a Network System Using Multimicroprocessors, Micoprocessors and Microsystems, Vol. 6, No. 1, January/February 1982, pp 15-20.

[8] M.J. Geary, The hardware Development of a Multi-Microcomputer Network Simulator, Modelling and Simulation on Microprocessors, San Diego.CA.USA, 1983, pp 140-144.

[9] I.M. Barron, Intercommunication Within Distributed Systems, Small Systems Software, Vol. 1, No. 2, pp 6-10.

[10] R. Taylor, Graphics with the Transuter, Compurt Gaphics '84, 1984.

[11] T. Mano et al., OCCAM To CMOS. Experimental Logic Design Support System, Computer Hardware Description languages, Eds. C.J. Koomen and T. Moto-oka, North-Holland, 1985.

[12] D.S. Broomhead et al., A Practical Comparison of the Systolic and Wavefront Array Processing Architectures, 2nd Proc. IEEE Conf. on Acoustics, Speech and Signal Processing, March 1985, pp 8.7.1-8.7.4.

[13] D. Fay, Working with OCCAM: A Program for Generating Display

Images, Microprocessors and Microsystems, Vol. 8, No. 1, January/February 1984, pp 3-15.

[14] P. Wilson, Highly Concurrent Systems Using the Transputer, Proc. Northon '84, Seattle, October 2-4, 1984, pp 13/2 1-11.

[15] —, IMS T800, IMS T400 and IMS T200 Transputer Product Overviews, Inmos Ltd.

[16] —, Transputer Reference Manual, Inmos Ltd.

[17] C. Whitby-Stevens, The Transputer, 12th Int. Symposium on Computer Architecture, Boston, June 17-19, 1985. pp 292-300.

[18] R. Taylor, Transputer Communication Link, Microprocessors and Microsystems, Vol. 10, No. 4, 1986, pp 211-215.

[19] C.A.R. Hoare, Communicating Sequential Processes, Comms. of the ACM, Vol. 21, No. 8, August 1978. pp 666-677.

[20] D. Fay, Comparison of CSP and the Programming Language OCCAM, Australian Computer Science Communications, Vol. 6, No. 1, 1984.

[21] D. May and R. Shepherd, The Transputer Implementation of Occam, Proc. of the Int. conf. on 5th Generation Computer Systems, 1984..

[22] D. May, OCCAM, SIGPLAN Notices, Vol. 18, No. 4, April 1983.

[23] D. May and R. Taylor, OCCAM - An Overview, Microprocessors and Microsystems, Vol. 8, No. 2, March 1984, pp 73-79.

[24] D. Pountain, A Tutorial Guide to OCCAM Programming, Inmos Ltd.

[25] —, IMS C004 programmable Link Switch (Preliminary Data), Inmos Ltd.

[26] J.J. Pilliod, Fundamental Plans for Toll Telephone Plant, Bell Sytems Technical Journal, Vol 31, 1952, pp832-850.

[27] E. Szybicki, A.E. Bean, Advanced Traffic Routing in Local Telephone Networks; Performance of Proposed Call Routing Algorithms, ITC-9, 1979.

[28] G.R. Ash, A.H. Kafker, K.R. Krishman, Intercity Dynamic Routing Architecture and Feasibility, ITC-10, Session 3.2, Paper 2, 1982.

[29] C. Grandjean, Call Routing Strategies in Telecommunication Networks, ITC-5, June 1967, pp 261-269.

[30] M. Gerla, Deterministic and Adaptive Routing Policies in Packet-Switched Computer Networks, Proc IEEE 3rd Data Commission Sym-

posium, November 1973, pp 23-28.

[31] A. Girard and Y. Cote, Sequential Routing Optimization for Circuit Switched Networks, IEEE Trans. on Comms., COM-32, No 12, Dec 1974, pp 1234-1242.

[32] A. Girard and S. Hurtubise, Dynamic Routing and Call Repacking in Circuit Switched Networks, IEEE Trans. on Comms., COM-31, No 12, Dec 1983, pp 1290-1294.

[33] P.M. Lin, B.J. Leon, C.R. Stewart, Analysis of Circuit Switched Networks Employing Originating-Office Control with Spill-Forward, IEEE Trans. on Comms., COM-26, No 6, June 1978, pp 754-765.

[34] G.R. Ash, R.H. Cardwell, R.P. Murray, Design and Optimization of Networks with Dynamic Routing, Bell System Technical Journal, Vol 60, No 8, Oct 1981, pp 1787-1820.

[35] G.R. Ash, A.H. Kafker, K.R. Krishman, Servicing and Real-Time Control of Networks with Dynamic Routing, Bell System Technical Journal, Vol 60, Oct 1981, pp 1821-1845.

[36] P. Mars and M. Chrystall, Real-Time Telephone Traffic Simulation Using Learning Automata Routing, Yale University Technical Report, S & IS No 7907, November 1979.

[37] K.S. Narendra, P. Mars and M. Chrystall, Simulation Study of Telephone Traffic Routing Using Learning Algorithms – Part II, Yale University Technical Report, S & IS No. 7907, October 1979.

[38] K.S. Narendra and P. Mars, The Use of Learning Algorithms In Telephone Traffic Routing – A Methodology, Automatica, Vol. 19, No. 5, 1983, pp 495-502.

[39] R. Gibbens, F. Kelly and P.B. Key, Dynamic Alternative Routing – Modelling and Behaviour, ITC 12, June 1988, Turin, Italy.

[40] R.Gibbens, Some Aspects of Dynamic Routing In Circuit Switched Telecommunication Networks, Rayleigh Prize Essay, University of Cambridge, January 1986.

[41] R.S. Krupp, Stabilization of Alternate Routing Network, IEEE Int. Commun. Conf., Philadelphia, PA, 1982, No 3I.2.

[42] T.G. Yum and M. Schwartz, Comparison of Routing Procedures for Circuit–Switched Traffic in Nonhierarchical Networks, IEEE Trans. On Comms., COM-35, No 5, May 1987, pp 535-544.

[43] J.M. Akinpelu, The Overload Performance of Engineered Networks with Nonhierarchical and Hierarchical Routing, AT&T Bell Laboratories Technical Journal, Vol. 63, No. 7, September 1984.

[44] D.G. Haenschke, D.A. Kettler and E.Oberer, Network Management and Congestion in the U.S. Telecommunications Network, IEEE Trans. On Comms., COM-29, No. 4, April 1981.

[45] L. Kleinrock, Queueing Systems. Volume I:Theory, John Wiley and Sons, New York, 1975.

[46] K.S. Narendra and M.A.L. Thathachar, On the Behaviour of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing, IEEE Trans. On Systems, Man and Cybernetics, SMC-10, No. 5, May 1980, pp 262-269.

[47] M.S. Chrystall, Adaptive Control of Communication Networks Using Learning Automata, Ph.D Thesis, Robert Gordon's Institute of Technology, Aberdeen, March 1982.

[48] R.I. Wilkinson, Theories For Toll Traffic Engineering in the U.S.A., Bell System Technical Journal., Vol. 35, No. 2, pp 421–514, 1956.

[49] R. Gibbens, Dynamic Routing in Circuit Switched Networks: The Dynamic Alternative Routing Strategy, Ph.D Thesis, University of Cambridge, July 1988.

[50] N. Eshragh, Dynamic Routing in Non—Hierarchical Circuit Switched Networks, Ph.D Thesis, University of Durham, December 1989.

[51] H. Zimmermann, OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection, IEEE Trans. On Comms., COM-28, No. 4, April 1980, pp 425–432.

[52] Systems Network Architecture, Special Issue, IBM Systems Journal, Vol 22, No. 4, 1983, 295–466.

[53] S. Wecker, DNA: The Digital Network Architecture, IEEE Trans On Comms., COM–28, No. 4, April 1980, pp 510–526.

[54] M. Schwartz, Computer–Communication Network Design and Analysis, Prentice–Hall, Englewood Cliffs, New Jersey, 1977.

[55] G.L. Fultz and L. Kleinrock, Adaptive Routing Techniques in Store and Forward Computer Communication Networks, Proc. IEEE Int. Conf. on Communications, Montreal, Canada, 1973, pp 23–28.

[56] M. Gerla, Deterministic and Adaptive Routing Policies in Packet-Switched Computer Networks, Proc. IEEE 3rd Data Commission Symposium, Nov. 1973, pp 23–28.

[57] M. Schwartz and T.Stern, Routing Techniques in Computer Communication Networks, IEEE Trans. On Comms., COM–28, No. 4, April 1980, pp 539–552.

[58] J.M. McQuillan et al., A review of the Development and Performance of the ARPANET Algorithm, IEEE Trans. On Comm., COM–26, No. 12, Dec 1978, pp 1802–1811.

[59] —, The New Routing Algorithm for the ARPANET, IEEE Trans. On Comms., COM–28, No.5, May 1980, pp 711–719.

[60] E.W. Dijkstra, A Note on Two Problems in Connection with Graphs, Numerical Mathematics, Vol. 1, 1959, 269–271.

[61] M. Schwartz amd T–K. Yum, Distributed Routing In Computer Communication Networks, 21st IEEE Conference On Decision and Control, Orlando, Florida, Dec. 1982, pp 600–603.

[62] T.E Stern, An Improved Routing Algorithm for Distributed Computer Networks, IEEE Int. Symp. on Circuits and Systems, Workshop on Large–Scale Networks and Systems, Houston, Texas, April 1980.

[63] A Fail Safe Distributed Routing Protocol, IEEE Trans. On Comms., COM–27, No. 9, Sept. 1979, pp 1280-1287.

[64] M.Schwartz, Routing and Flow Control in Data Networks, NATO Advanced Studies Institute: New Concepts in Multi–User Communications, Norwich, U.K., August 4–6, 1980, Sijthoof and Norohoof, Netherland.

[65] L. Fratta et al., The Flow Deviation Method: An Approach to Store and Forward Communication Network Design, Networks, Vol. 3, Part 3, 1973, pp 97–133.

[66] M. Schwartz and C.K. Cheung, The Gradient Projection Algorithm for Multiple Routing in Message Switched Networks, IEEE Trans. On Comms., COM–25, April 1976, pp 449–456.

[67] D.G. Cantor and M. Gerla, Optimal Routing in a Packet Switched Computer Network, IEEE Trans. On Comms., COM–23, No. 10, Oct 1974, pp 1062–1068.

[68] T.E. Stern, A Class of Decentralised Routing Algorithms Using Relaxation, IEEE Trans. On Comms., COM–25, No. 10, Oct 1977, pp 1092–1102.

[69] L. Kleinrock, Communication Nets: Stochastic Message Flow and Delay, McGraw–Hill, New York, 1964; reprinted Dover Publications 1972.

[70] B. Meister, H.R. Mueller and H. Rudin, On the Optimization of Message Switched Networks, IEEE Trans. On Comms., COM–20, No. 1, Feb. 1972, pp 8–14.

[71] D. Wismer and R. Chattergy, Introduction to Non Linear Programming, North Holland, 1978.

[72] R.G Gallagher, A Minimum Delay Routing Algorithm Using Distributed Communication, IEEE Trans. On Comms., COM–25, No. 1, Jan. 1977, pp 73–85.

[73] D. Bertsekas and R. Gallagher, Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks, IEEE Trans. On Comms., COM–32, No. 8, August 1984, pp 911–919.

[74] T.M. Ng and D.B. Hoang, Joint Optimization of Capacity and Flow assignment in a Packet Switched Communications Network, IEEE Trans. On Comms., COM–35, No. 2, Feb 1987, pp 202–209.

[75] H. Frank and N. Chou, Routing In Computer Networks, Networks, Vol. 1, 1971, pp 99-112.

[76] P. Baran, On Distributed Communication Networks, IEEE Trans. On Comm. Systems, CS-12, Part 1, Marcg 1964, pp 1–9.

[77] W. Chou et al, The Need For Adaptive Routing in the Chaotic and Unbalanced Traffic Environment, IEEE Trans. On Comms., COM–29, No. 4, April 1981, pp 481–490.

[78] H. Rudin, On Routing and 'Delta Routing': A Taxonomy and Performance Comparison of Techniques for Packet Switched Networks, IEEE Trans. On Comms., COM–24, No. 1, Jan 1976, pp 43–59.

[79] H. Rudin and H. Mueller, Dynamic Routing and Flow Control, IEEE Trans. On Comms., COM–28, No. 7, July 1980, pp 1030–1039.

[80] K. Narendra and R. Wheeler Jr., Routing In Communication Networks–A Case Study of Learning in Large Scale Systems, Large Scale Systems—Theory and Applications, Vol 8, No. 3, June 1985, pp211–222.

[81] M. Chrystall and P. Mars, Learning Automata Routing in Message Switched Communication Networks, Tech. Report No. 8101 , Robert Gordon Institute of Technology, School of Electronic and Electrical Engineering, Feb 1981.

[82] C.E. Agnew, On Quadratic Adaptive Routing Algorithms, Communications of the ACM, Vol. 19, No. 1, Jan. 1976, pp 18–22.

[83] L.G. Mason, Equilibrium Flows, Routing Patterns and Algorithms for Store and Forward Networks, Large Scale Systems, Vol. 8, 1985, North Holland

[84] S.C Dafermos and F.T. Sparrow, The Traffic Assignment Problem for a General Network, Jounal of the Natioal Bureau of Standards–B. Mathematical Science 73b, Vol. 2, 1969.

[85] M. Gerla and L. Kleinrock, Flow Control: A Comparative Survey, IEEE Trans. On Comms., COM–28, No. 4, April 1980, pp 553–567.

[86] L. Pouzin, Methods, Tools and Observations on Flow Control in Packet–Switched Data Networks, IEEE Trans. On Comms., COM–29, No. 4, April 1981, pp 413–426.

[87] M. Reiser, Performance Evaluation of Data Communication Sytems, Proceedings of the IEEE, Vol. 70, No. 2, February 1982, pp 171–196.

[88] M. Schwartz and S. Saad, Analysis of Congestion Control Techniques in Computer Communication Networks, Proc. Symp. on Flow Control in Computer Networks, Versailles, France, Feb. 1979.

[89] M. Irland, Buffer Management in a Packet Switch, IEEE Trans. On Comms., COM–26, No. 3, March 1978, pp 328–337.

[90] M. Pennotti and M. Schwartz, Congestion Control in Store and Forwad Tandem Links, IEEE Trans. On Comms., COM–23, No. 12, Dec 1975, pp 1434–1443.

[91] D.W. Davies, The Control of Congestion in Packet Switched Networks, IEEE Trans. On Comms., COM–20, No. 3, June 1972, pp 546–550.

[92] W.L. Price, Data Network Simulation Experiments at the National Physical Laboratory, Computer Networks, Vol. 1, 1977, p 199-210.

[93] S. Lam and M. Reiser, Congestion Control of Store–and–Forward Networks by Input Buffer Limits—An Analysis, IEEE Trans. On Comms., COM–27, No. 1, Jan. 1979.

[94] M Schwartz, Telecommunication Networks: Protocols, Modelling and Analysis, Addison–Wesley, 1987, ISBN 0-201-16423-X.

[95] S. Saad and M. Schwartz, Input Buffer Limiting Mechanisms for Congestion Control, International Communications Conference, Seattle, June 1980, pp 32.1–23.5.

[96] J.M. McQuillan, Interaction Between Routing and Flow Control in Computer Networks, Proc. Int. Symp. on Flow Control in Computer Networks, Versailles, France, Feb. 1979, pp 63–75.

[97] M. Gerla and P.O. Nilsson, Routing and Flow Control Interplay in Computer Networks, Proc. Int. Conf. on Computer Comms., Altanta, Oct 1980, pp 84–89.

[98] M. Reiser, A Queueing Network Analysis of Computer Communication Networks with Window Flow Control, IEEE Trans. On Comms. COM–27, No. 8, August 1979, pp 1199–1209.

[99] R.G. Gallagher and S.J. Golestanni, Flow Control and Routing Algorithms for Data Networks, Proc. Int. Conf. on Comp. Comms., Atlanta, Oct. 1980, pp 779-784.

[100] G. Thaker and J. Cain, Interactions Between Routing and Flow Control Algorithms, IEEE Trans. On Comms., COM–34, No. 3, March 1986, pp 269–277.

[101] A. Tanenbaum, Computer Networks, Prentice–Hall, 1981.

Appendix A
Learning Automata

Classical control theory assumes that a complete mathematical model can be formulated for the process whose outputs are to be monitored. The model is then used to control the inputs in some fashion. Uncertaintly can be managed by the use of stochastic control theory only if the probability of the uncertainty can be characterised. If such characteristics cannot be formulated then control must be implemented by observation and deduction of the system in operation. This can be thought of as a learning process, defined as a relatively permanent change in behaviour, based on past experience. Mathematically it is interesting to consider the problem as one of optimization of a function which is not explicitly known. One approach to the solution, using stochastic automata, attempts to find the optimal action out of the set of all allowed actions and works in the following manner. Initially each allowed action is assigned an equal probability of selection. One of those actions is then randomly selected and the response of the environment is observed. Based on this response the selection probabilities of each of the actions are then updated. This process of selection and the subsequent updating of the selection probabilities is then repeated on the selection of every action. Stochastic automata, operating in this way to satisfy some performance index, are also known as learning automata.

Stochastic Automata

A stochastic automata is a sextuple $\{x, \phi, \alpha, p, A, G\}$ where x is the input value, ϕ is a set of s internal states, α is a set of r outputs or actions with $r \leq s$, p is a probability vector governing the choice of internal states at each stage, i.e. at stage n, $p(n) = \{p_1(n), \ldots, p_s(n)\}$. A is an algorithm known as a reinforcement scheme which maps $p(n) \rightarrow p(n+1)$ and $G : \phi \rightarrow \alpha$ is the output function, mapping the internal states onto to the action set. Often there is a deterministic one–to–one mapping between ϕ and α in which case the two become synonymous. This arrangement is shown schematically in figure A.1.

The environment can be considered as a similar black box with an input vector at a point n, $\alpha(n)$, where $\alpha(n) = \{\alpha_1(n),\ldots,\alpha_r(n)\}$ and an output value x as shown in figure A.2. x lies in the range (0,1) and can either take purely binary values (0 or 1, true or false, yes or no, etc) in which case it is referred to as a P–model; discrete quantised values, when it is known as a Q–model; or any value in the range, when it is known as a continuous or S–model. Within each of these models the probability of a particular output, given an input i is given by c_i the penalty probability. For a simple Q–model these penalty probabilities are random variables, but for the more complex models they take the form of distribution functions for each input. Further if c_i are independent of n, the environment is said to be stationary, otherwise it is known as non–stationary.

Figure A.3 shows the connection between the two environments. The actions of the automata form the inputs to the environment and the response of the environment is fed into the automaton to update the internal states via the reinforcement algorithm.

Behaviour

In this section some of the basic definitions are outlined for the analysis and classification of learning schemes based on the automaton. Given that the basic operation of the automaton is to update the action probabilities on the response of the environment to a particular action, a useful quantity to examine is the áverage penalty' which such an automaton can be expected to receive. At a time n, if an action $\alpha_i(n)$ is selected with a probability $p_i(n)$, the average penalty, $M(n)$ is given by

$$M(n) = E\{x(n) \mid p(n)\} = \sum_{i=1}^{r} p_i(n)c_i$$

For a completely random response where each possible action is selected with equal probability the average penalty is given by M_O where

$$M_O = \frac{\sum_{i=1}^{r} c_i}{r}$$

A process of learning can be said to have occurred if the average penalty incurred by an automaton is asymptotically less than M_O. In this case the learning automaton is said to be expedient. Formally

Def 1: A learning automaton is called expedient if

$$\lim_{n\rightarrow\inf} E[M(n)] < M_O$$

More desirable then merely outperforming a random environment would be an automaton which minimised the average penalty. This is termed optimal behaviour and is defined as

Def 2: A learning automaton is called optimal if

$$\lim_{n\rightarrow\inf} E[M(n)] = \min_i c_i \equiv c_l$$

This suggests that the automaton would converge to the action with the minimum penalty with probability one. If this action is not desirable for reasons other than performance a sub–optimal scheme would have to be used. An important class of sub–optimal scheme is the ε–optimal learning automata, defined as

Def 3: A learning automaton is called ε–optimal if

$$\lim_{n\rightarrow\inf} E[M(n)] < c_l + \varepsilon$$

for any $\varepsilon > 0$ by suitable choice of reinforcement scheme paramters. This suggests such a scheme can be made asymptotically as close to an optimal scheme as desired without the selection of any one action in a deterministic manner.

Other useful properties of a learning automata scheme would be a monotonic decrease in the value of $E[M(n)]$ with time and the ability to produce a desirable performance over the whole range of possible c_i rather than just a certain range (in which case the scheme is referred to as conditional). This leads to the final definition of a scheme which can be said to be absolutely expedient.

Def 4: A learning automata is absolutely expedient if

$$E[M(n+1) \mid p(n)] < M(n)$$

for all n,$p(n)$ and all c_i. If $M(n) \leq M_O$ then absolute expediency implies expediency. Further it can be shown that absolute expediency implies ε–optimality in a stationary random environment.

Reinforcement

The reinforcement scheme is the means by which the environmental responses are used to update the selection probabilities. In general terms the scheme can be represented by an operator T, where

$$p(n+1) = T(p(n), \alpha(n), x(n))$$

The scheme itself can be classified in terms of its behaviour using the definitions above and in terms of the form of the functions used in the scheme itself, in the simplest case linear or non–linear. For any reinforcement scheme there are three basic functions it can perform, reward, punishment and inaction. According to the particular scheme an action $\alpha_i(n)$ will modify the action probability $p_i(n)$ according to the response of the environment $x(n)$. Whether a particular scheme rewards, punishes or leaves unchanged the probability selection may depend on the value of the response. In general for an action α_i at n the probability vector $p(n+1)$ will be defined by a series of equations

$$\begin{aligned} p_j(n+1) &= p_j(n) - f_j(p(n)) \quad \text{for reward} \\ p_j(n+1) &= p_j(n) + g_j(p(n)) \quad \text{for punish} \\ (j &\neq i) \end{aligned}$$

and

$$\begin{aligned} p_i(n+1) &= p_i(n) + f_i(p(n)) \quad \text{for reward} \\ p_i(n+1) &= p_i(n) - g_i(p(n)) \quad \text{for punish} \end{aligned}$$

where both $f_j(\cdot)$, the reward function and $g_j(\cdot)$, the punishment function map $p_k(n)\varepsilon(0,1)$ onto $p_k(n+1)\varepsilon(0,1)$ for all $k = 1 \ldots r$. If either the reward or punishment sections of the reinforcement algorithm wish to leave the selection probabilities unchanged (i.e. inaction) then $f(\cdot)$ or $g(\cdot)$ is simply replaced by zero.

Convergence

Two types of convergence behaviour have been identified for the learning automata. In the first case the sequence of action probabilities, $p_i(n)$, generated from the ergodic Markov process converges to give a distribution function over all points of continuity. These are expedient schemes with no absorbing barrier.

In the second type of convergence scheme, seen in ε–optimal automaton, each of the action probabilities converge to a limiting random variable with probability one. These schemes result in an ergodic Markov process with two or more absorbing barriers, only one of which actually corresponds to the minimum penalty probability. One can only then say that the automata will converge to the desired penalty probability with a positive probability.

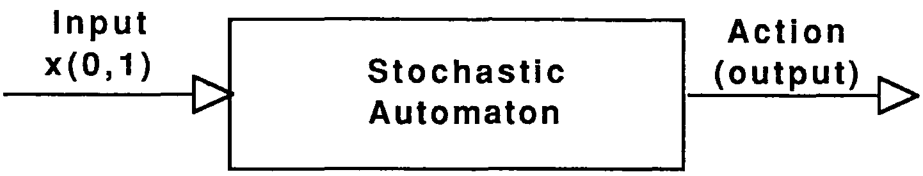


Figure A.1

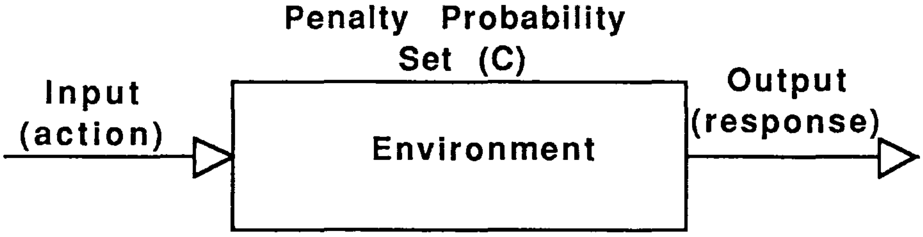


Figure A.2

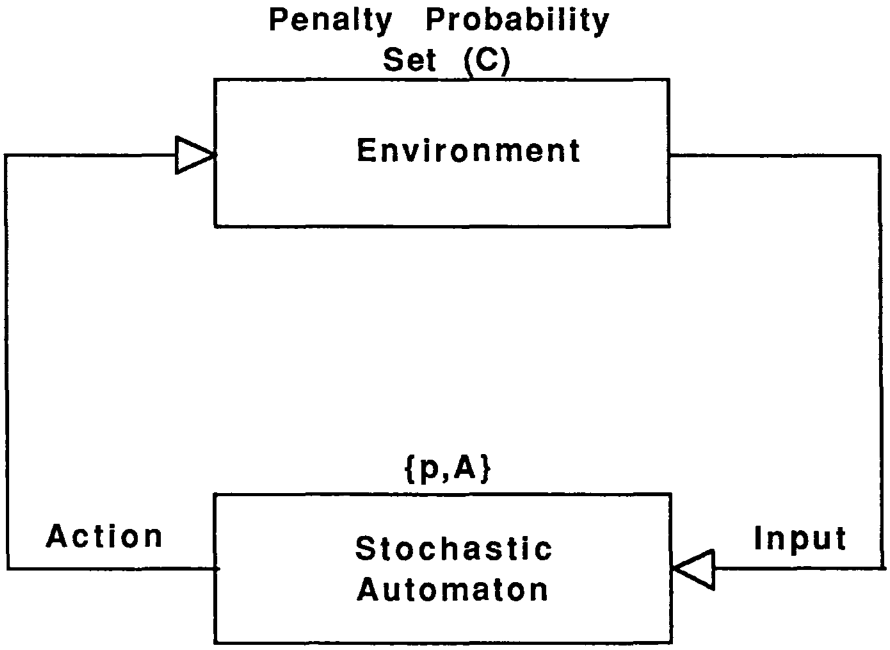


Figure A.3

Appendix B
Recovery of Circuit Switched Code

The code written for the implementation of the circuit switched algorithm for each node of a full connected network has been copied onto the diskette included with this thesis. The code is written in Occam2 using the D700C Transputer development system compiler from INMOS Ltd. A full version of the code is provided, except for the subroutines called directly from libraries supplied with the development system. Where these have been used a short comment is included within the code defining the subroutines function. The software is available in two forms, DOS file format and in TDS file format. The listing in DOS file format is in the directory **\CIRCUIT\DOS** in a single files called **CIRCUIT.LIS**. The file can be examined using any conventional screen editor or listed sequentially to the screen by inserting the diskette in drive a: and typing

A: <CR>

TYPE A:\CIRCUIT\DOS\CIRCUIT.LIS || MORE <CR>

from the > prompt. Similarly a hard copy can be made by directing the file to a printer attached to the PC directly or over a network. The instructions for this will be dependent the configuation of the particular machine being used. The version of the code in TDS format is in the directory **\CIRCUIT\TDS** in a series of files of the form *filename*.**TSR**. The code can be examined by accessing it through the file **CIRCUIT.TSR** via a transputer development system (TDS). The advantage of this method is that the fold structure used to develop the code is maintained and the operator is able to move around within the file and examine it with greater ease than via a standard sequential listing.

Appendix C
Recovery of Packet Switched Code

The code written for the implementation of the packet switched algorithm for each node of a sparsely connected network has been copied onto the diskette included with this thesis. The code is written in Occam2 using the D700C Transputer development system compiler from INMOS Ltd. A full version of the code is provided, except for the subroutines called directly from libraries supplied with the development system. Where these have been used a short comment is included within the code defining the subroutines function. The software is available in two forms, DOS file format and in TDS file format. The listing in DOS file format is in the directory **\PACKET\DOS** in a single file called **PACKET.LIS**. The file can be examined using any conventional screen editor or listed sequentially to the screen by inserting the diskette in drive a: and typing

```
A: <CR>
TYPE A:\PACKET\DOS\PACKET.LIS || MORE <CR>
```

from the > prompt. Similarly a hard copy can be made by directing the file to a printer attached to the PC directly or over a network. The instructions for this will be dependent the configuation of the particular machine being used. The version of the code in TDS format is in the directory **\PACKET\TDS** in a series of files of the form *filename*.TSR. The code can be accessed through the single file **PACKET.TSR** via a transputer development system (TDS). The advantage of this method is that the fold structure used to develop the code is maintained and the operator is able to move around within the file and examine it with greater ease than via a standard sequential listing.

Presented at 5th U.K. Teletraffic Symposium, University of Aston, July 1988

STUDY OF DYNAMIC ROUTING ALGORITHMS USING A HIGH-SPEED MULTIPROCESSOR SIMULATOR

S.J. Nichols, R.T.Clarke and P.Mars

Abstract

This paper reports on the progress in the development of a high speed transputer based network simulator and presents simulation results for fully connected circuit switched networks. Both hardware and software design strategies are outlined and future work on packet-switched networks described.

1. INTRODUCTION

The analysis of wide area communication networks to determine the performance of dynamic routing strategies rapidly becomes analytically intractable as the complexity of the problem increases. In addition behaviour under° transient conditions such as traffic fluctuations or component failures are difficult to express mathematically. Under such conditions the use of simulation techniques to determine relevant network parameters becomes necessary. The conventional, sequential simulations, running on mainframe computer systems suffer from limitations imposed by the excessive use of CPU time required to achieve the required depth of information and is further compounded by the statistical nature of the results. These problems increase as functions of traffic intensity across the network and size of the network itself leading to detailed simulations of large networks often becoming economically and even physically impossible to implement. In this paper the design of a network simulator is presented which seeks to avoid the mainframe implementation limitations by the introduction of concurrent processing into the simulation environment and discusses the major features of the software design in terms of their implementation requirements over the loosely coupled multiprocessor system.

Previous attempts to develop high performance simulation environments discussed in the literature have introduced pipelined processors[1], distributed environments controlled form a central control unit[2] or hierarchical bus structures connecting processors simulating network nodes[3]. These structures suffer from bottlenecks at some point in their division of the workload or introduce bus contention between processors requiring communication which limits their potential performance. The basis of this new design is to capitalise on the advantages of multiprocessor implementation highlighted by these previous attempts but avoid the degradation in performance caused by insufficient communication bandwidth or unequal division of processing requirements between independant hardware modules.

In the simulator transputers are used to represent the nodes in a communications network and the links are used to mimic the connections between the nodes. By using the computational power provided by each transputer and its serial links, a powerful simulation tool has been constructed from an array of these devices, the code of which is written in the computer language Occam. This language uses all the standard declarations of variable types found in languages like PASCAL and C, with the addition of a channel type which allows

S.J.Nichols, R.T.Clarke and P.Mars are with the School of Engineering and Applied Science, University of Durham.

communication between processes that are executing concurrently. It is these software channels which map onto the transputers serial data links. Programs are constructed from four basic building blocks: SEQ for sequential processing, PAR for parallel processing, IF for conditional processing, and finally ALT which is used for alternative processing. The SEQ construct allows the programmer to define a block of code which will be executed in order. The PAR construct is used to define blocks code which are to be executed concurrently. Logical conditional processing is accomplished by use fo the IF construct where the execution of a block of code is dependent on the result of a Boolean expression. The most complex construct is the ALT. This construct is similar to the IF, however instead of using a logical condition as the test for executing a block of code, a guard is used. A guard being simply an input from a channel, with an optional condition. A block of code in an ALT construct will only be executed if data is received down the channel which forms its guard. This construct is very important to many of the operations of the simulator software.

In many other respects, Occam is like any other computer language. It allows both procedures, functions, loop constructs, matrix manipulation, and logical operations, and also has a full set of standard arithmetic operators. The language is relatively straightforward to use once the programmer has mastered the problems which can arise when programming a parallel problem. All the software which runs on the simulator is coded in Occam.

2. SIMULATOR HARDWARE

The hardware of the network simulator is made up from three basic units (Figure 1), a user front-end and file server, a graphical front-end, and a 'black box' transputer based simulator. It was the aim of the overall design strategy to produce a simulation tool that requires the end-user to have little or no knowledge of the software or hardware on which the simulation is carried out, and this is achieved by careful use of these three basic units.

The user front-end and the file server is an IBM-AT (or compatible) which contains a plug-in transputer board. Its main use outside the simulation environment is as a software development tool, supporting the transputer development system and the Occam compiler. During a simulation run it is used to draw up the network topology graphically, enter initial conditions for the simulation, and allow the user to program modifications into the simulation at run-time. The results from the simulation are stored on the IBM-AT in MS-DOS format files. These MS-DOS files can then be used for post simulation statistical analysis and result processing on a main frame computer system.

The graphical front-end is used to display results on-line, making it possible to investigate transients on a network during a simulation by introducing modifications such as lost-links, failed nodes and traffic fluctuations at run time. The graphic front-end enhances the power and increases the usability of the simulator as a network support tool.

The 'black box' simulator is a reconfigurable array of transputers which are used to model nodes within a communications network. The basic structure of the hardware can be divided into two halves based on the operational tasks which each individual transputer performs (Figure 2). The first half consists of the four devices which perform, control and result handling tasks. There is an IBM plug-in card (IMS B004) which files the results, an IMS B007 graphics card to display results, a T414 transputer which handles the user interface software, and a T800 floating point transputer which generates the on-line results. The other half of the hardware consists of the T414 transputer cards which actually perform the simulation. Between these halves lie the programmable link switches which are controlled by the transputer that handles the user interface software. These switches are used to configure the simulation transputers to the network topology under investigation before the simulation begins. Although the limit on the size of the network which can be simulated is dependent on the arrangement of the link switches and the number of transputers available, it is possible to simulate networks of any conceivable size using this basic structure for the simulator.

These three units are interconnected to form the simulator itself which, together with the software discussed below, produced the simulation results presented in this paper.

3. SOFTWARE DESIGN

The conceptual design of the simulation environment can be separated into six major areas and mapped directly into Occam using the SEQ and PAR programming concepts to produce a three-stage pipeline. By their concurrency the processes allow on-line modification and network parameter calculation to allow the operator to trace and direct the simulation during execution. The transformation is reproduced in Figure 3 identifying the parallel processes and the major points of data exchange between the processes. The central simulation process can then be further divided into a number of concurrent processes each corresponding to a nodal process and connected by channels recreating the network topology. This combination of pipelined and mesh topologies, implemented on dedicated hardware, in a multiprocessor environment, assigning one or more processors to each concurrent process, leads to a powerful and flexible simulation structure which can be expanded to encompass networks of arbitrary topological size and complexity. Connectivity difficulties, which arise due to more links terminating at a node than are available in a single processor mapping, are accommodated by dividing the nodal interface routines into independant processes feeding into a central control process. This is divided in such a way as to ensure the connectivity of each process does not exceed the implementation limit on the processors.

The distributed nature of the network simulation, isolating nodal data structures from both each other and equally the interface processes requires special software constructs to enable the simulation to process effectively. Most important of these constructs are those which enable nodal synchronization within the simulation process and allow access to the nodal processes from the result and modification procedures.

Conventional discrete - event simulators can be divided into those which use synchronous and asynchronous clock mechanisms. Asychronous discrete - event simulators are based on a linked list of events and their time of occurrence. The simulation jumps from event epoch to event epoch inserting new events in chronological sequence within the list as they are created by the events presently being processed. Synchronous schemes divide the simulation period into time slots and process all events occurring in a particular slot as if they occurred simultaneously before moving to the next slot by incrementing the timing mechanism. In general asynchronous techniques are favoured for accurate modelling as the synchronous grouping of events in time slots can produce problems for large slots and reduction in step size can lead to inefficient code. However, for non-critical models and sensible selection of time increment synchronous models are viable, especially if the model ensures no bias toward execution of certain events in preference to others of the same priority within the interval.

The implementation of an asynchronous timing mechanism within a distributed environment is made difficult by the lack of a central data structure in which to store the linked list of forthcoming events. Interaction between local queues, swapping next event times is a possible solution but would require a large transmission overhead and reduce concurrency within the system to virtually nil since only the event at the head of all the combined queues would undergo execution at any time. The synchronous approach, while less flexible for the reasons mentioned above, acts to create concurrency through event epoch approximation and grouping and was used as the clock mechanism in the model. Each nodal process scans their data structures for events falling within the time slot, implements them and then broadcasts flags to each neighbour informing them of the nodal processes clock increment.

In addition the lack of a central data structure also prevents the storage of simulated call information in an easily accessible form and it must be retrieved from the distributed network processors. This information is periodically sampled and the nodal position updated by the same structure. The simulation process on each node is momentarily suspended and a second process activated. This second process allows the flow of information to and from the interface processes controlling the results and network condition respectively, and acts as one link of a linear chain between the interfaces made up of all the nodal processes. The results are passed out as a summary of each nodes history up to the moment of simulation suspension and builds into a picture of the simulations progress the detail of which can be controlled by the operator.

The individual nodal processes must perform the normal functions of such units in a circuit switched simulation with the added complication that they must also support concurrent communication with their neighbours. A continuous block of sequential code with input and output statements buried within it is too inflexible to select interface operations appropriate to its neighbours requirements and would cause lock-up. Similarly, separating the input and output processes into independant parallel processes leads to deadlock conditions under certain circumstances without the provision of shared buffers which are difficult to implement in Occam. The actual structure of the nodal process is reproduced in Figure 4, in which three parallel procedures are connected to a central process by channels and this central process accesses the data structure for each of them. The node operates on a two-level interrupt mechanism. The first layer multiplexes and demultiplexes the input and output channels to and from the network. The second layer feeds requests for process recognition to the central processing section. The generator joins the input and output processes at this point with requests for new calls to the network. When a process is selected by the central process the channel communication is initiated and this is followed by execution of the relevant code segment in the central process for that particular interruption. The structure is implemented in the form of ALT structures and operates in a way similar to conventional interrupt mechanisms with the advantage of simultaneous data transfer over the links.

This summarizes the major details of the simulation package developed for the dedicated multiprocessor hardware. It has been designed to take advantage of the concurrent nodal execution while minimising the problems of distributed data structures in a loosely coupled environment. The code was written throughout in Occam 2 using the beta 2.0 release of the language from INMOS Ltd.

4. SIMULATION RESULTS

To obtain initial results from the simulator a number of simulations have been carried out on the network shown in Figure 5. The aim of these simulations is to show how the routing algorithms LRI and DAR perform under differing traffic conditions. To achieve this four traffic patterns have been devised (Figure 6), which have the following properties:

Pattern A - is underloaded by 2.18% and has a balanced traffic distribution.

Patten B - is overloaded by 2.18% and has a balanced traffic distribution.

Pattern C - is overloaded by 2.18% and has an unbalanced traffic distribution.

Pattern D - is underloaded by 2.18% and has an unbalanced traffic distribution.

These patterns are merged into each other in such a way as to move from traffic pattern A to B, B to C, C to D and D to A. Mixing the traffic patterns at equally spaced intervals. As the traffic is varied from one pattern to another eleven simulation points are obtained which show how the routing algorithms behave. These points are plotted against blocking probability. Each simulation consists of 3/4 million calls and takes approximately 85 minutes on a single 10 MIP T414 transputer and represents $2\frac{1}{4}$ hours of simulation time. The two sample result graphs (Figures 7 and 8) show some of the preliminary results from the simulator for this experiment. The first graph shown in Figure 7 traces the blocking probability between the network traffic distribution B, the overloaded network, and network C, the overloaded and unevenly distributed network. The second graph given in Figure 8 plots the same parameter for a decreasing traffic rate in the most unevenly distributed example. In each case the lower bound on blocking performance is given by the Erlang blocking probability formulae using the summation of both the traffic and trunk capacities to provide the calculation parameters. This bound has been recently shown to outperform a linear programming solution despite its simplicity. This can be attributed to the power of the Poisson/negative exponential model over the deterministic counterpart.

It is also possible to theoretically calculate the exact blocking probability using fixed routing with no alternative paths by simple summation using standard formulae. This is also shown for each traffic range to provide a comparison for its dynamic opponents. The first plot also shows simulation data for the fixed strategy superimposed on the curve and a least squares polynomial curve of the points for comparison and validation of the simulator code.

Finally, each plot shows simulated results for DAR and LRI routing in the network with no TRP. Each set of points are connected by least squares polynomial fitted curves and include 95% confidence interval error bars. The results clearly show that both algorithms are outperformed by the fixed routing when there is no or little spare capacity in the network although the dynamic strategies do not show the same degradation in performance as the fixed routing when the distribution is non-ideal. This results in a narrowing of the gap between the fixed and dynamic strategies which continues as capacity becomes available, but is still insufficient to compensate for the cost of tandem links to route overflow traffic.

5. CONCLUSIONS

The paper has described the hardware and software design strategy for a high-speed multiprocessor network simulator using transputers. Initial results on circuit switched networks have been presented. The simulator is expected to provide high-speed/low-cost simulations for realistic network modelling and convenient sensitivity analysis.

Leading on from the work in circuit switched networks, and using exactly the same hardware system, a packet switched model is being developed for the analysis of flow control and congestion avoidance in these store-and-forward networks. These networks are even more taxing to simulate than their circuit switched counterparts as the time scales involved are orders of magnitude smaller and the transfer of information generally has to be reproduced more accurately as it involves the interlacing of data packets rather than merely the reservation of dedicated trunk capacity. The software currently being constructed implements a simplified form of the first four protocol layers of the OSI model developed by the ISO. The model concentrates on the third layer controlling the routing of packets across the network and flow control within the network. It is hoped to carry out simulation studies into the interaction of dynamic routing strategies and network flow control mechanisms as tools to delay congestion within heavily loaded networks by traffic bifrucation.

6. ACKNOWLEDGEMENTS

The authors wish to thank the SERC for supporting the work described in this paper.

7. REFERENCES

1. KAIN, R.: "CHIMPNET : A Network Testbed", Computer Networks, 3, Dec 1979, pp 447-457.

2. BAREL, M.: "A Flexible High Performance Multiprocessor for Data Network Simulation", ITC-190, Session 3.3 Paper No.9.

3. TODA, A, et al.: "A Parallel Processing Simulator for a Network System using Multimicroprocessors", Microprocessors and Microsystems, Jan/Feb 1982, pp 15-20.

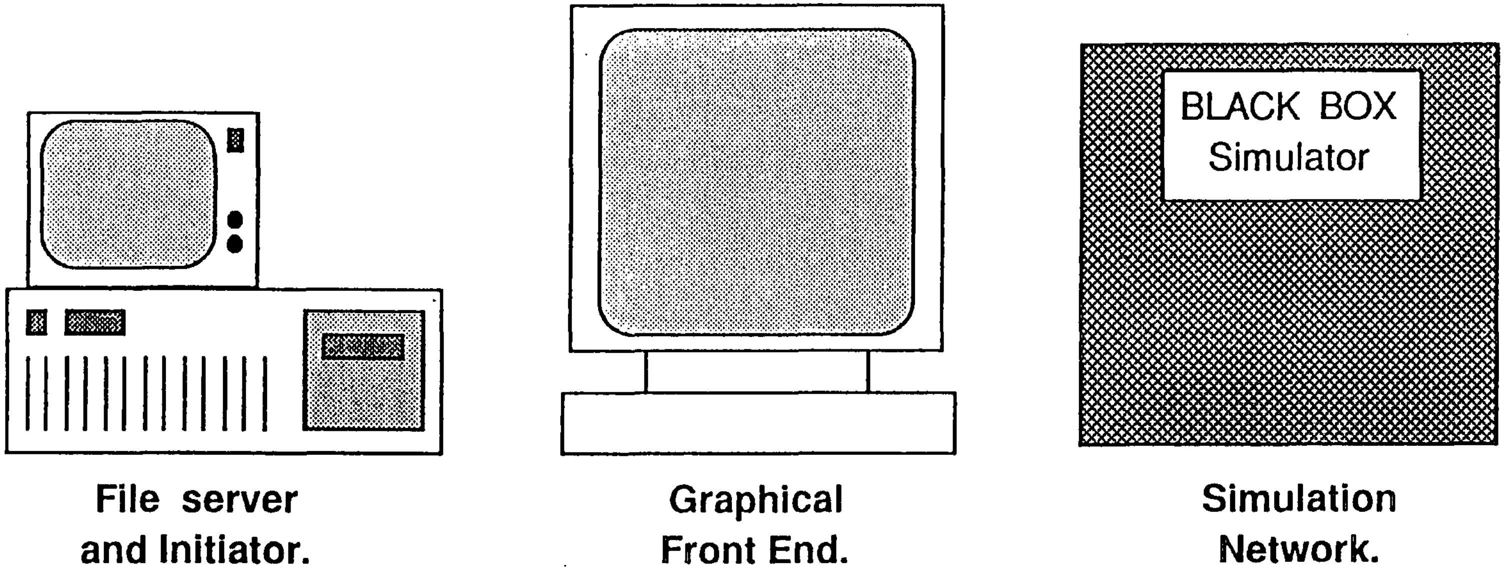


FIGURE 1. HARDWARE SCHEMATIC DIAGRAM

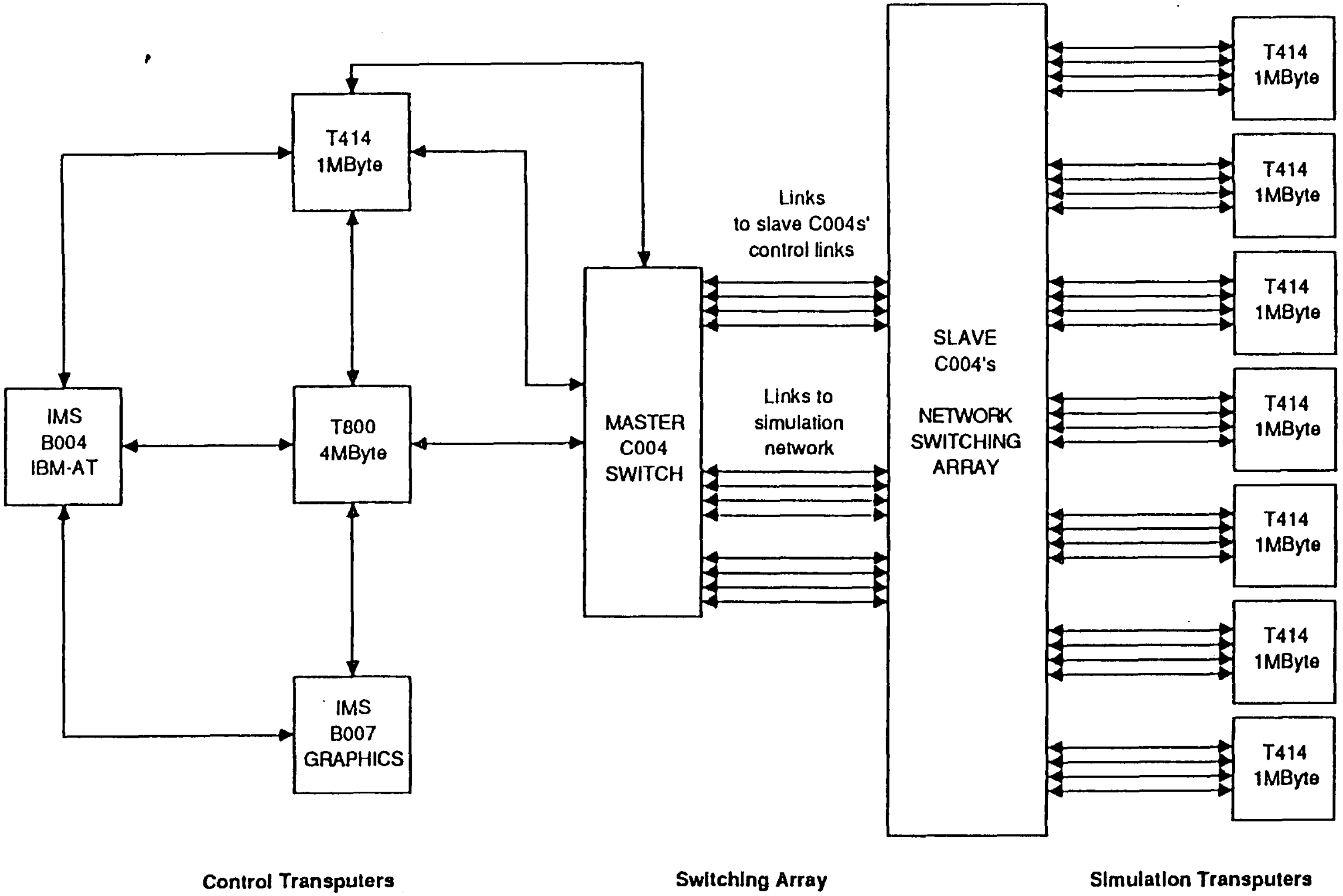
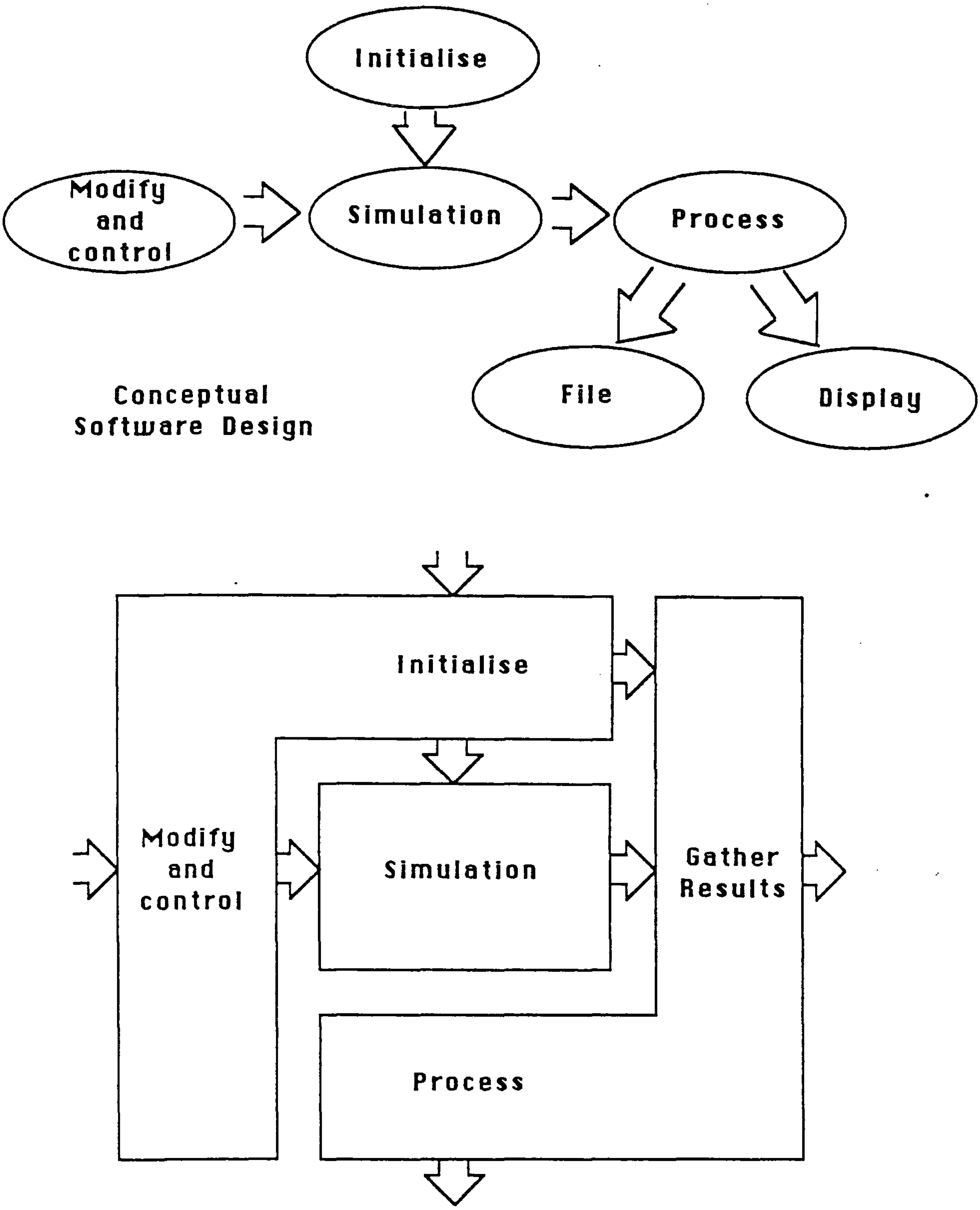


FIGURE 2. HIGH-SPEED TRANSPUTER BASED NETWORK SIMULATOR - HARDWARE CONFIGURATION



Conceptual Software Design

OCCAM Mapping

Figure 3

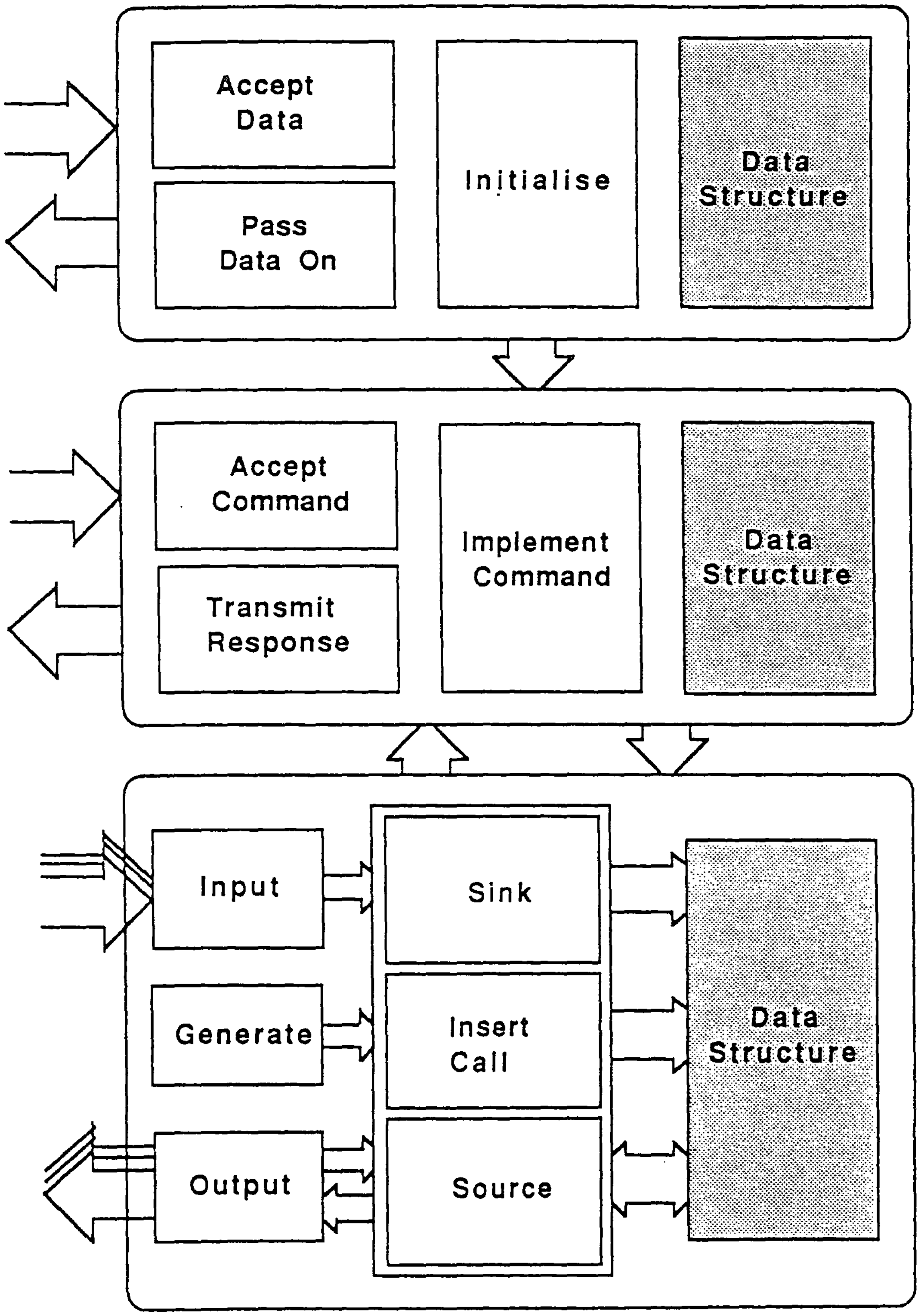

Accept Data
Pass Data On
Initialise
Data Structure
Accept Command
Transmit Response
Implement Command
Data Structure
Input
Sink
Generate
Insert Call
Output
Source
Data Structure

FIGURE 4.

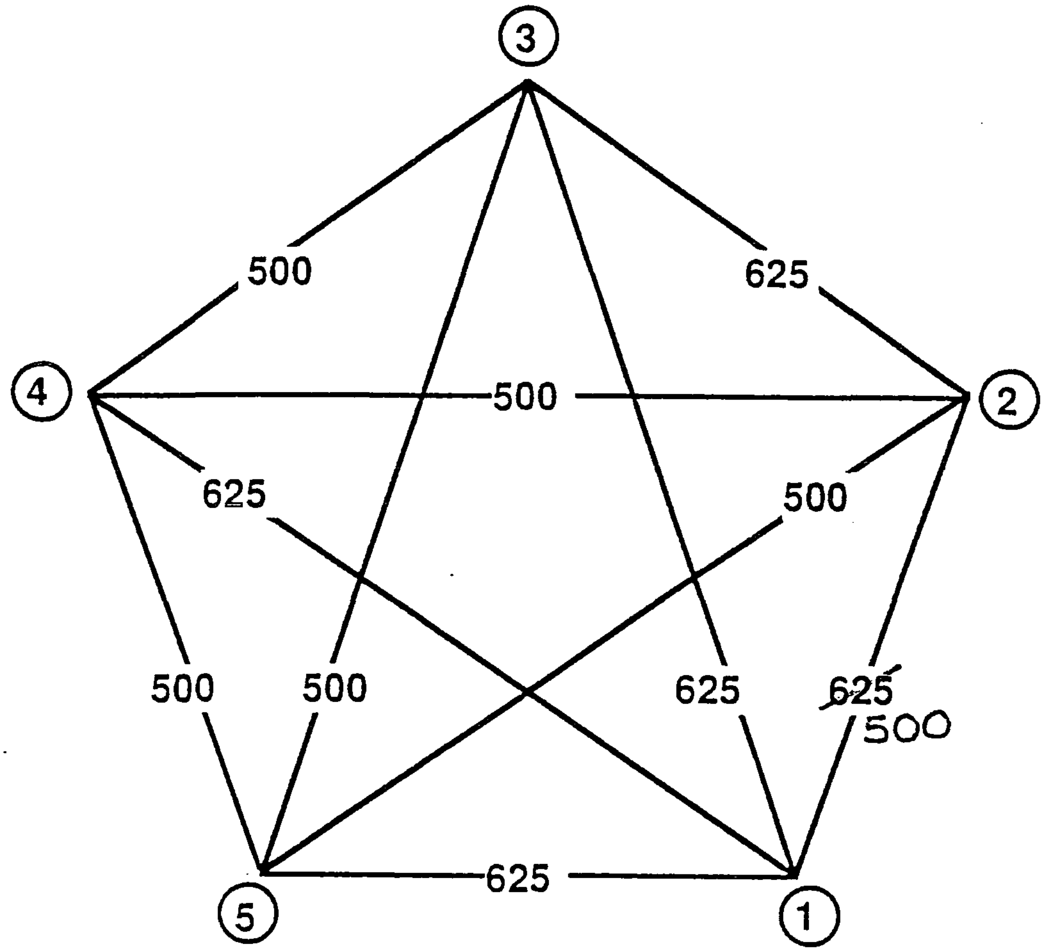
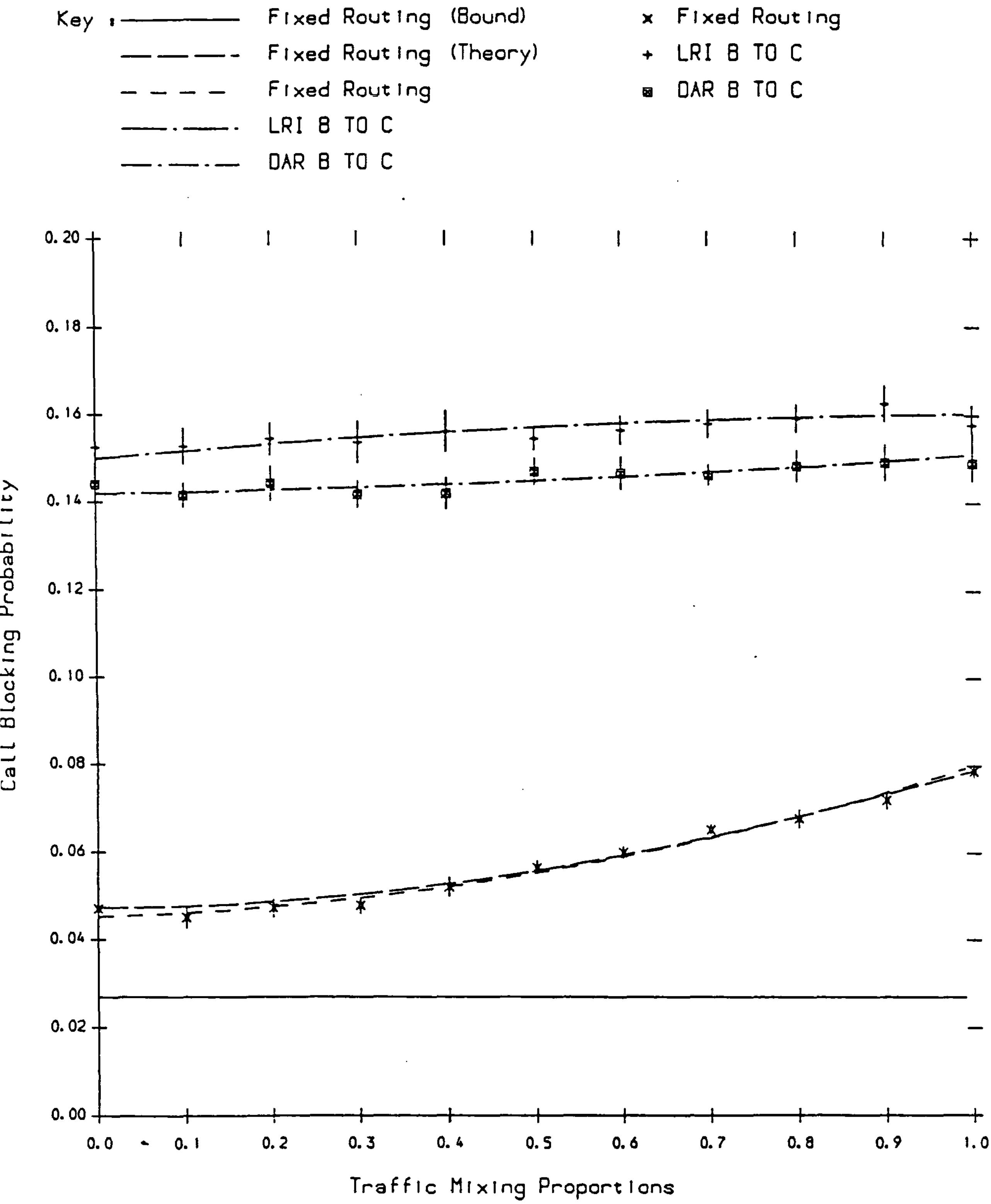
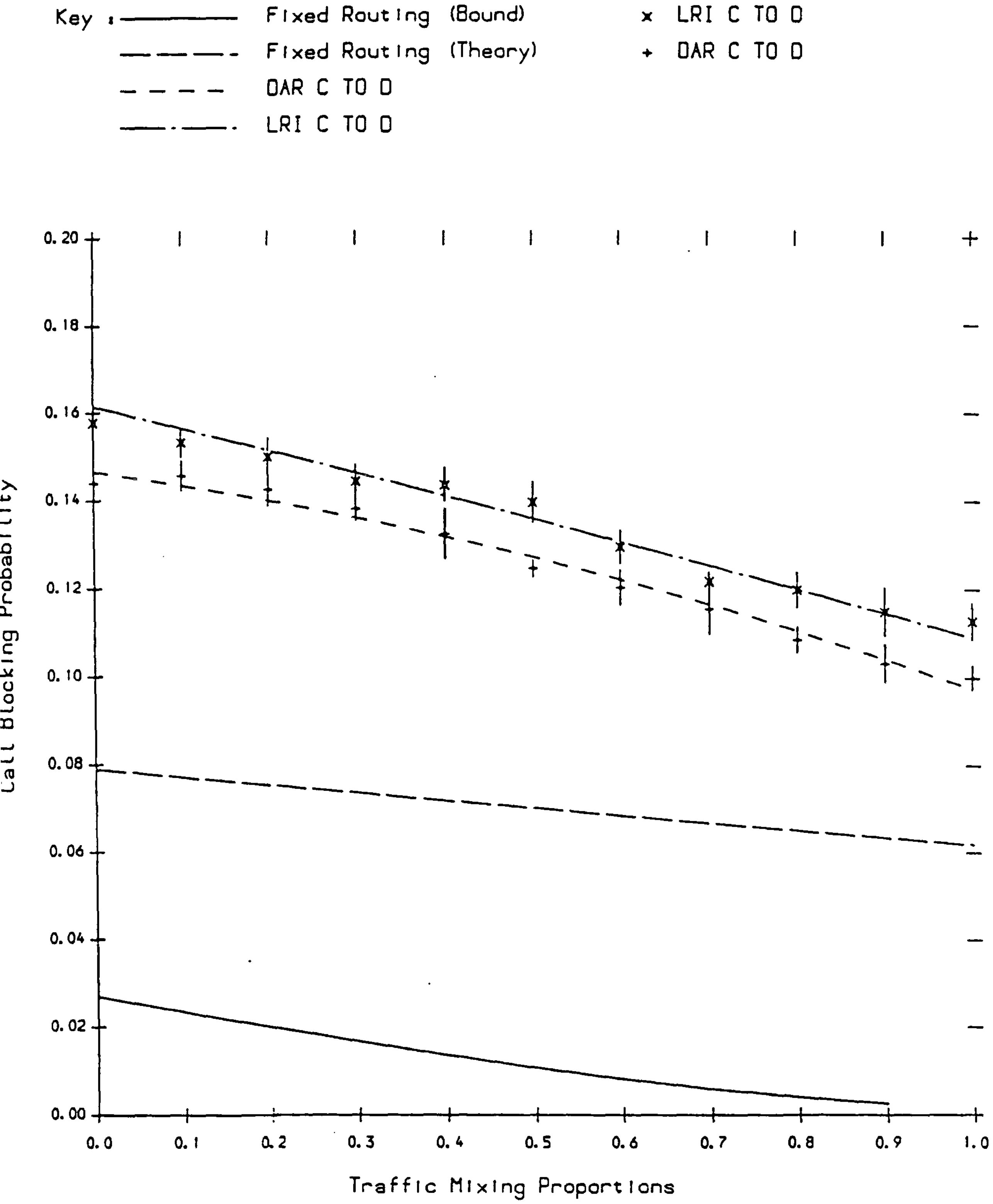


FIGURE 5. NETWORK UNDER STUDY



DYNAMIC ROUTING IN AN INCREASINGLY UNBALANCED NETWORK

FIGURE 7



DYNAMIC ROUTING WITH VARIABLE, UNBALANCED TRAFFIC

FIGURE 8

PRESENTED AT EUROMICRO 88 ZURICH

DESIGN OF A HIGH SPEED SIMULATION TOOL FOR WAN USING PARALLEL PROCESSING

S.J. Nichols, R.T. Clarke and P. Mars

University of Durham, School of Engineering and Applied Science,
Science Laboratories, South Road, Durham, DH1 3LE, U.K.

This paper presents the design strategy behind a high speed network simulation machine developed at Durham University. The simulator uses configured arrays of transputers to create a multiprocessor environment for the modelling of circuit and packet switched networks, taking advantage of the transputers unique features for parallel implementation of such models. Some interesting results gathered from a software prototype of a circuit switched model are presented and the proposed implementation of a packet switched model is outlined.

1. INTRODUCTION

The analysis of wide area communication networks to determine the performance of dynamic routing strategies rapidly becomes analytically intractable as the complexity of the problem increases. In addition behaviour under transient conditions such as traffic fluctuations or component failures are difficult to express mathematically. Under such conditions the use of simulation techniques to determine relevant network parameters becomes necessary. The conventional, sequential simulations, running on mainframe computer systems suffer from limitations imposed by the excessive use of CPU time required to achieve the required depth of information and is further compounded by the statistical nature of the results. These problems increase as functions of traffic intensity across the network and size of the network itself, leading to detailed simulations of large networks often becoming economically and even physically impossible to implement. In this paper the design of a network simulator is presented which seeks to avoid the mainframe implementation limitations by the introduction of concurrent processing into a ledicated simulation environment and liscusses the major features of the software lesign in terms of their implementation requirements over the loosely coupled ıultiprocessor system.

'revious attempts to develop high performance simulation environments discussed in the iterature [1-3] have introduced pipelined rocessors, distributed environments ontrolled from a central control unit or ommon bus structures connecting processors imulating network nodes. These structures uffer from bottlenecks at some point in heir division of the workload or us contention between processors requiring ommunication which limits their potential performance. The basis of this new design is to capitalise on the advantages of multiprocessor implementation highlighted by these previous attempts but avoid the degradation in performance caused by insufficient communication bandwidth or unequal division of processing requirements between independant hardware modules.

In the simulator, transputers are used to represent the nodes in a communications network and the links are used to mimic the connections between the nodes. By using the computational power provided by each transputer and its serial links, a powerful simulation tool has been constructed from an array of these devices. The models are written in Occam [4], a special purpose language written for the transputer. Based on CSP [5], the language allows code to be separated into independant groups, communicating with each other through point-to-point channels. This allows concurrent execution of these groups with synchronisation and data transfer using the channels between them.

2. SIMULATOR HARDWARE

The INMOS transputer is a high speed processor which has been designed to form the fundamental building block of parallel computers of tomorrow. Each device in the transputer family has a RISC based processor capable of processing up to ten million instructions in a single second and four high speed serial links which transfer data at 20 Mbit/s. These links are used to communicate with other transputers to produce parallel computing arrays. These devices form the heart of the high speed simulator, providing it with processing power to carry out detailed investigations into network behaviour under varied conditions.

e hardware of the network simulator is made from three basic units, a user front-end d file server, a graphical results unit and 'black box' transputer based simulator igure 1). These units are used to produce simulation tool which requires the end-user have little or no knowledge of the ftware or hardware on which the simulation executed.

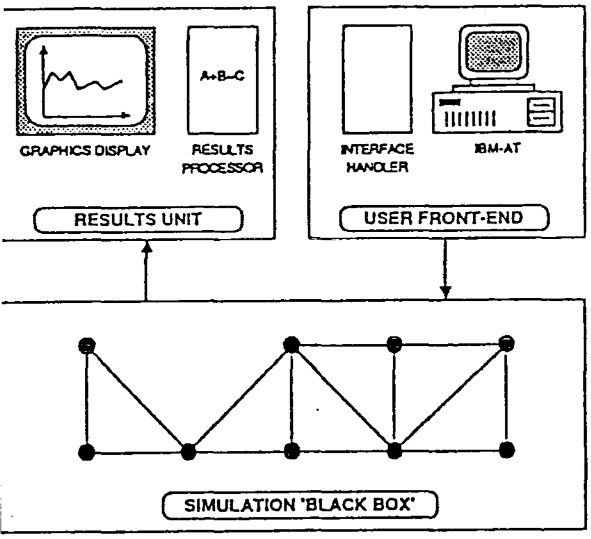


FIGURE 1. SCHEMATIC DIAGRAM OF THE HIGH-SPEED TRANSPUTER-BASED SIMULATOR

e user front-end and the file server is sed around an IBM-AT computer which ntains a transputer that interfaces onto e PC-bus. During a simulation run it is ed as a tool for drawing up the network pology graphically, entering the initial nditions for a simulation, and to introduce difications at run-time. A further nction of the IBM-AT is to act as a result ore for the data produced during a mulation run. A graphical results unit is ed to display results during the simulation n, making it possible to investigate the haviour of a network after introducing difications such as lost links, failed des and traffic fluctuations at run time. e graphical results unit both enhances the wer and increases the usability of the mulator as a network support tool. The lack box' simulator is constructed from a configurable array of transputers which are ed to model the nodes within a mmunication network by adopting the same pology as the network under investigation. e array consists of a number of 32 bit terger transputers (IMS T414-20) which are ter-connected via programmable cross point switches (IMS C004). These switches are used to reconfigure the array under software control to any given network topology using a master/slave arrangement for nodes with more than four links. Although the limit on the size of the network which can be simulated is dependent on the link switches and the number of transputers available, it is possible to model a network of any conceivable size using the arrangement adopted for the hardware presented in this paper.

3. SOFTWARE DESIGN

The conceptual design of the simulation environment can be separated into five major areas and mapped directly into Occam using the SEQ and PAR programming concepts to produce a three-stage pipeline. By their concurrency the processes allow on-line modification and network parameter calculation, enabling the operator to trace and direct the simulation during execution. The transformation is reproduced in Figure 2 identifying the parallel processes and the major points of data exchange between these processes. The central simulation process can then be further divided into a number of concurrent processes each corresponding to a nodal process and connected by channels recreating the network topology. This combination of pipelined and mesh topologies, implemented on dedicated hardware, in a

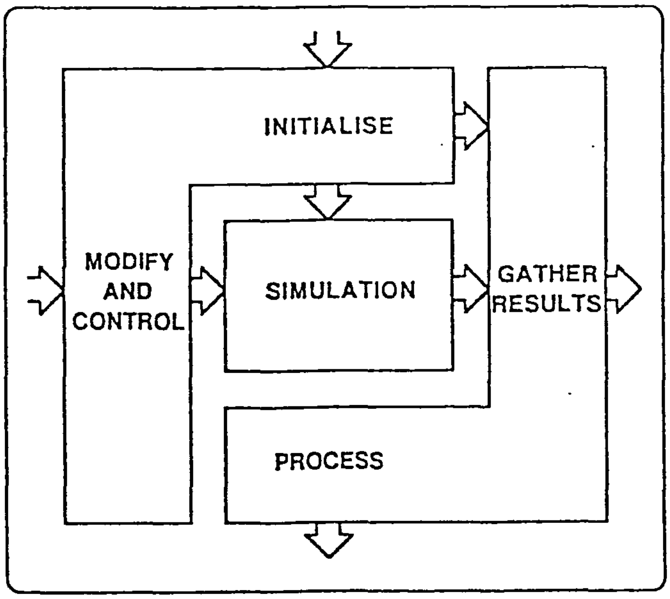


FIGURE 2. SOFTWARE ARCHITECTURE

.ltiprocessor environment, leads to a werful and flexible simulation structure ich can be expanded to encompass networks arbitrary topological size and complexity.

.e distributed nature of the network mulation, isolating nodal data structures om both each other and equally the iterface processes requires special software nstructs to enable the simulation to oceed effectively. Most important of these nstructs are those which enable nodal nchronization within the simulation process id allow access to the nodal processes from ie result and modification procedures.

nventional discrete - event simulators can divided into those which use synchronous id asynchronous clock mechanisms. The plementation of an asynchronous timing chanism within a distributed environment is ide difficult by the lack of a central data ructure in which to store the linked list forthcoming events. Interaction between cal queues, swapping next event times, uld require a large transmission overhead id reduces concurrency within the system to ssentially zero since only the event at the ead of all the combined queues would undergo cecution at any time. The synchronous pproach, while less flexible for the reasons entioned above, creates concurrency through vent epoch approximation and grouping and as used as the clock mechanism in the model.

addition the lack of a central data ructure also prevents the storage of imulated call information in an easily ccessible form and it must be retrieved from ie distributed network processors. This formation is periodically sampled and the odal position updated by the same structure. ie simulation process on each node is omentarily suspended and a second process ctivated. This second process allows the ow of information to and from the interface ocesses controlling the results and network ondition respectively, and acts as one link a linear chain between the interfaces made of all the nodal processes. The results e passed out as a summary of each nodes story up to the moment of simulation ispension and builds into a picture of the ie simulations progress the detail of which n be controlled by the operator.

ie individual nodal processes must perform ie normal functions of such units in a rcuit switched simulation with the added mplication that they must also support ncurrent communication with their ighbours without deadlocking. The actual ructure of the nodal process is reproduced Figure 3, in which three parallel

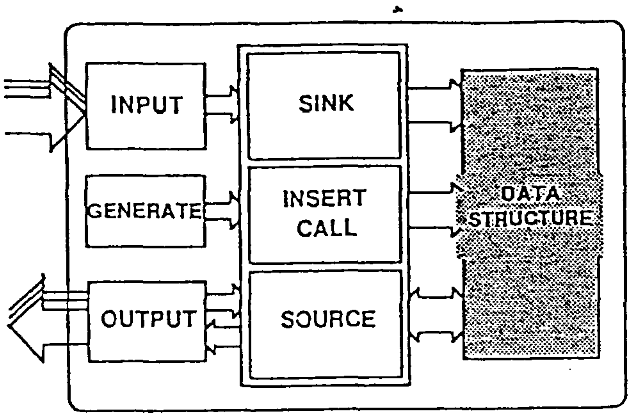


FIGURE 3. NODAL ARCHITECTURE

procedures are connected to a central process by channels while this central process accesses the data structure for each of them. The node operates on a two-level interrupt mechanism. The first layer multiplexes and demultiplexes the input and output channels to and from the network. The second layer feeds requests for process recognition to the central processing section. When a peripheral process is selected by the central process the channel communication is initiated and this is followed by execution of the relevant code segment in the central process for that particular interruption. The structure operates in a way similar to conventional interrupt mechanisms with the advantage of simultaneous data transfer over the links.

4. SIMULATION RESULTS

Initial work has concentrated on software development of a fully connected circuit circuit switched model, using a sophisticated call modelling algorithm which seeks to capture the effects of nodal processing ability as well as trunk availability. These models are especially suitable for the multiprocessor environment as the necessary processing for each call is distributed throughout the calls path. The results presented here compare the blocking probabilities of dynamic routing strategy DAR [6], chosen by British Telecom for implementation in system X, with another dynamic algorithm LRI [7], based on learning automata theory and on which work has been done by Bell [8], for the five node, fully

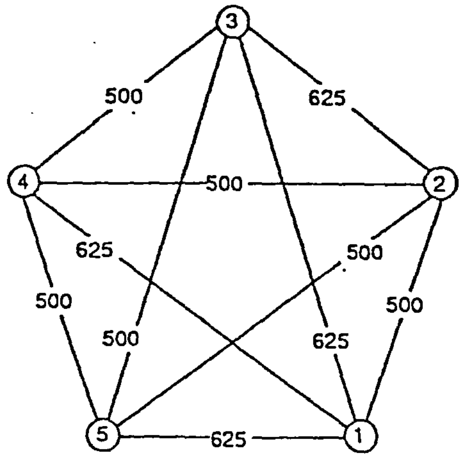
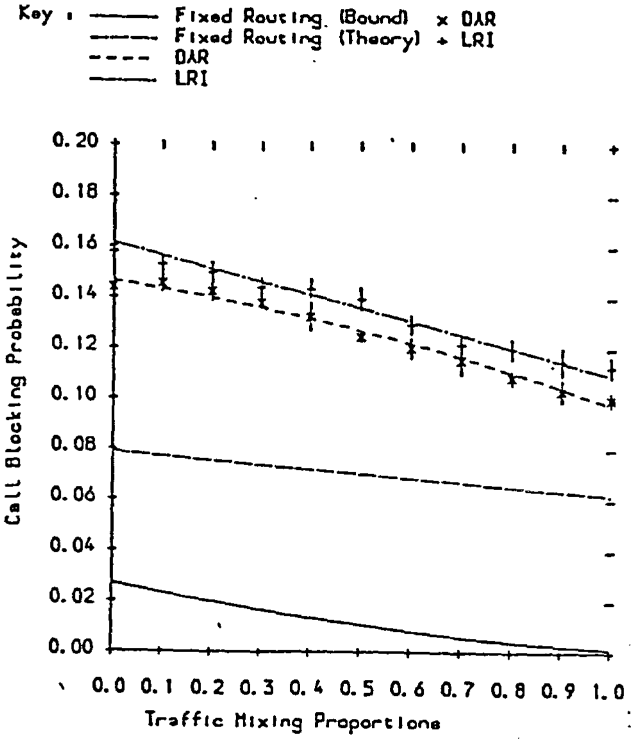


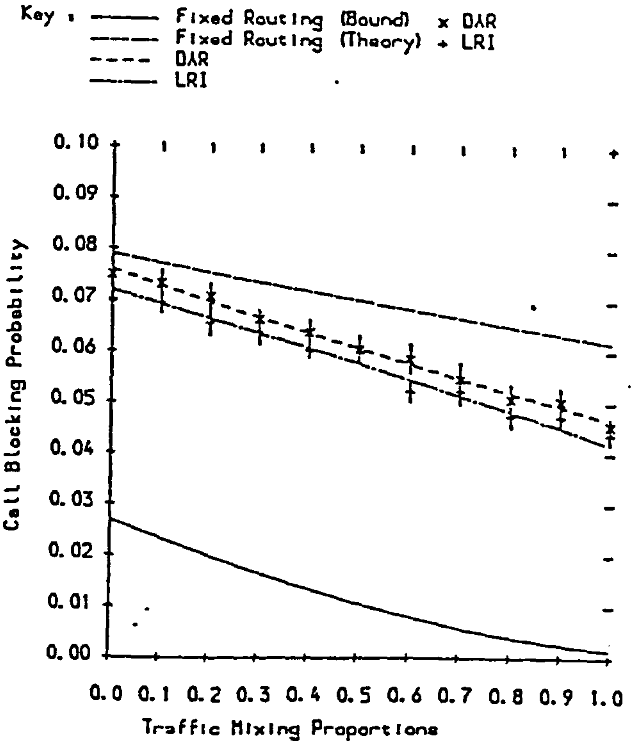
FIGURE 4.
NETWORK UNDER STUDY

terconnected, asymmetrical network
produced in Figure 4. Each point is the
sult of over 3/4 of a million calls and
ok just over two hours to produce using a
ngle transputer to execute the code. This
rformance is comparable with similar code,
itten in Fortran, if ran on a dedicated
dahl 470/V8 mainframe. When exported to an
ray of transputers it is projected that a
rformance benefit of O(N) will be achieved
r an N node network. The experiment
vestigated the performance of the two
gorithms as the offered traffic matrix
oothly progresses from an overloaded and
balanced situation to the underloaded but
ill unbalanced case. This is done by
rming composite traffic matrices between
e two extremes and plotting the result of
ese simulations. The aim is to compare the
rformance of the two algorithms as capacity
r alternative routing becomes increasingly
ailable. Figure 5(a) shows the results for
e case where each call is allowed to seek
alternative path if the direct path fails,
th no restrictions and is only blocked if
is attempt also fails. The second graph,
gure 5(b), shows the effect of including a
nk reservation factor (TRP) of 2% over
ch network link to prevent excess use of
ndem paths. Each graph includes two
eoretical curves, which calculate the
ocking probability using the direct path
y (Theory) and a lower bound on
rformance calculated using Erlangs formulae
a single trunk with capacity and traffic
ual to the sum of the individual components
the network (Bound). It is easy to show
s gives a lower bound on the performance
any routing strategy.



DYNAMIC ROUTING VITH NO OVERFLOV PROTECTION

FIGURE 5 (a)



DYNAMIC ROUTING VITH TRUNK RESERVATION

FIGURE 5 (b)

Two important results can be extracted from these results. First the inclusion of a TRP is important when considering dynamic routing algorithms as traffic approaches its engineered load to restrict overflow traffic to those paths where true spare capacity occurs. Secondly, in this important case, LRI consistently outperformed DAR in these simulation results. Further work is continuing to extend these simulation results and develop analytic models to study these and other algorithms.

5. CONCLUSIONS

The paper has described the hardware and software design strategy for a high-speed multiprocessor network simulator using transputers and when complete the simulator is expected to provide high-speed/low-cost simulations for realistic network modelling and convenient sensitivity analysis. In addition some interesting initial and thought provoking results on circuit switched networks have been presented.

Leading on from the work in circuit switched networks, and using exactly the same hardware system, a packet switched model is being developed for the analysis of flow control and congestion avoidance in these store-and-forward networks. These networks are even more taxing to simulate than their circuit switched counterparts as the time scales involved are orders of magnitude smaller and the transfer of information generally has to be reproduced more accurately as it involves the interlacing of data packets rather than merely the reservation of dedicated trunk capacity. The software currently being constructed implements a simplified form of the first four protocol layers of the OSI model developed by the ISO [9]. The model concentrates on the third layer controlling the routing of packets across the network and flow control within the network. It is hoped to carry out simulation studies into the interaction of dynamic routing strategies and network flow control mechanisms as tools to delay congestion within heavily loaded networks by traffic bifrucation.

ACKNOWLEDGEMENTS

The authors wish to thank the SERC for supporting the work described in this paper.

REFERENCES

[1] Lehnert, A., "A Special Processor for Fast Simulation of Queueing Networks" (ITC-9 Lehnert 1-7).

[2] Barel, M., "A Flexible High Performance Multiprocessor for Data Network Simulation", (ITC-190, Session 3.3 Paper No.9).

[3] Toda, A., et al. "A Parallel Processing Simulator for a Network System using Multimicroprocessors", (Microprocessors and Microsystems, Jan/Feb 1982) pp 15-20.

[4] Pountain, D., "Occam Reference Manual", (INMOS Ltd., March 1987).

[5] Hoare, C.A.R., "Communicating Sequential Processes", (Comms of the ACM, Vol 21, No 8, Aug.1978) pp 666-677.

[6] Gibbens, R., "Some Aspects of Dynamic Routing in Circuit Switched Telecommunication Networks", Rayleigh Prize Essay, (Statistical Laboratory of the University of Cambridge, Jan.1986).

[7] Narendra K.S., et al., "Learning Automata: A Survey", (IEEE Trans on Systems Man and Cybernetics, Vol SMC-24, No.4, July 1974).

[8] Ash G.R., et al., "Servicing and Real Time Control of Networks with Dynamic Routing", (Bell System Tech.Journal, Vol.60, No.8, Oct.1981).

[9] Zimmermann, H., "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection", (IEEE Trans on Comms., Vol.COM-28, No.4, April 1980) pp 425-432.

North-Holland
Microprocessing and Microprogramming 25 (1989) 327 – 332

Design of a High Speed Simulation Tool for WAN Using Parallel Processing

S.J. Nichols, R.T. Clarke and P. Mars

University of Durham, School of Engineering and Applied Science,
Science Laboratories, South Road, Durham, DH1 3LE, U.K.

This paper presents the design strategy behind a high speed network simulation machine developed at Durham University. The simulator uses configured arrays of transputers to create a multiprocessor environment for the modelling of circuit and packet switched networks, taking advantage of the transputers unique features for parallel implementation of such models. Some interesting results gathered from a software prototype of a circuit switched model are presented and the proposed implementation of a packet switched model is outlined.

1. INTRODUCTION

The analysis of wide area communication networks to determine the performance of dynamic routing strategies rapidly becomes analytically intractable as the complexity of the problem increases. In addition behaviour under transient conditions such as traffic fluctuations or component failures are difficult to express mathematically. Under such conditions the use of simulation techniques to determine relevant network parameters becomes necessary. The conventional, sequential simulations, running on mainframe computer systems suffer from limitations imposed by the excessive use of CPU time required to achieve the required depth of information and is further compounded by the statistical nature of the results. These problems increase as functions of traffic intensity across the network and size of the network itself, leading to detailed simulations of large networks often becoming economically and even physically impossible to implement. In this paper the design of a network simulator is presented which seeks to avoid the mainframe implementation limitations by the introduction of concurrent processing into a dedicated simulation environment and discusses the major features of the software design in terms of their implementation requirements over the loosely coupled multiprocessor system.

Previous attempts to develop high performance simulation environments discussed in the literature [1-3] have introduced pipelined processors, distributed environments controlled from a central control unit or common bus structures connecting processors simulating network nodes. These structures suffer from bottlenecks at some point in their division of the workload or introduce bus contention between processors requiring communication which limits their potential performance. The basis of this new design is to capitalise on the advantages of multiprocessor implementation highlighted by these previous attempts but avoid the degradation in performance caused by insufficient communication bandwidth or unequal division of processing requirements between independant hardware modules.

In the simulator, transputers are used to represent the nodes in a communications network and the links are used to mimic the connections between the nodes. By using the computational power provided by each transputer and its serial links, a powerful simulation tool has been constructed from an array of these devices. The models are written in Occam [4], a special purpose language written for the transputer. Based on CSP [5], the language allows code to be separated into independant groups, communicating with each other through point-to-point channels. This allows concurrent execution of these groups with synchronisation and data transfer using the channels between them.

2. SIMULATOR HARDWARE

The INMOS transputer is a high speed processor which has been designed to form the fundamental building block of parallel computers of tomorrow. Each device in the transputer family has a RISC based processor capable of processing up to ten million instructions in a single second and four high speed serial links which transfer data at 20 Mbit/s. These links are used to communicate with other transputers to produce parallel computing arrays. These devices form the heart of the high speed simulator, providing it with processing power to carry out detailed investigations into network behaviour under varied conditions.

The hardware of the network simulator is made up from three basic units, a user front-end and file server, a graphical results unit and a 'black box' transputer based simulator (Figure 1). These units are used to produce a simulation tool which requires the end-user to have little or no knowledge of the software or hardware on which the simulation is executed.

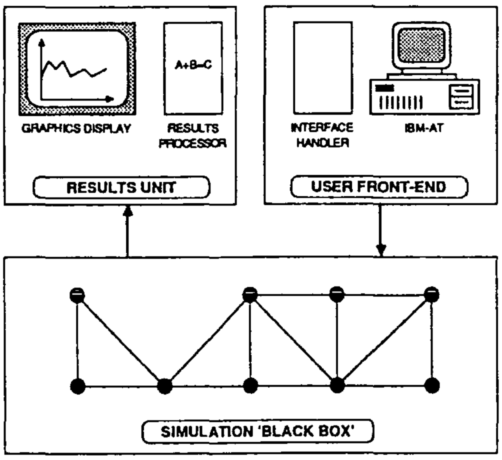


FIGURE 1. SCHEMATIC DIAGRAM OF THE HIGH-SPEED TRANSPUTER-BASED SIMULATOR

The user front-end and the file server is based around an IBM-AT computer which contains a transputer that interfaces onto the PC-bus. During a simulation run it is used as a tool for drawing up the network topology graphically, entering the initial conditions for a simulation, and to introduce modifications at run-time. A further function of the IBM-AT is to act as a result store for the data produced during a simulation run. A graphical results unit is used to display results during the simulation run, making it possible to investigate the behaviour of a network after introducing modifications such as lost links, failed nodes and traffic fluctuations at run time. The graphical results unit both enhances the power and increases the usability of the simulator as a network support tool. The 'black box' simulator is constructed from a reconfigurable array of transputers which are used to model the nodes within a communication network by adopting the same topology as the network under investigation. The array consists of a number of 32 bit interger transputers (IMS T414-20) which are inter-connected via programmable cross point switches (IMS C004). These switches are used to reconfigure the array under software control to any given network topology using a master/slave arrangement for nodes with more than four links. Although the limit on the size of the network which can be simulated is dependent on the link switches and the number of transputers available, it is possible to model a network of any conceivable size using the arrangement adopted for the hardware presented in this paper.

3. SOFTWARE DESIGN

The conceptual design of the simulation environment can be separated into five major areas and mapped directly into Occam using the SEQ and PAR programming concepts to produce a three-stage pipeline. By their concurrency the processes allow on-line modification and network parameter calculation, enabling the operator to trace and direct the simulation during execution. The transformation is reproduced in Figure 2 identifying the parallel processes and the major points of data exchange between these processes. The central simulation process can then be further divided into a number of concurrent processes each corresponding to a nodal process and connected by channels recreating the network topology. This combination of pipelined and mesh topologies, implemented on dedicated hardware, in a

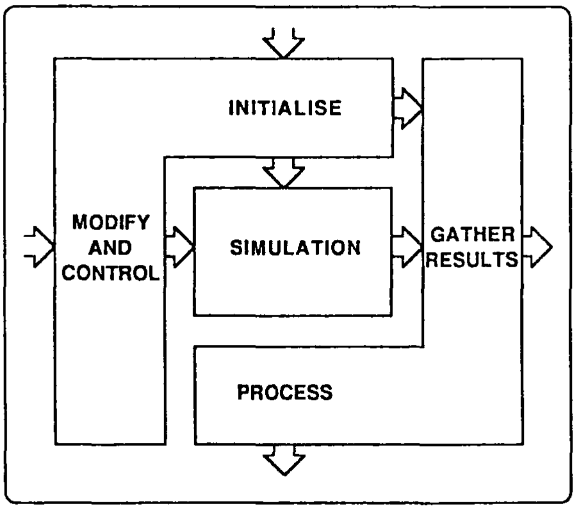


FIGURE 2. SOFTWARE ARCHITECTURE

multiprocessor environment, leads to a powerful and flexible simulation structure which can be expanded to encompass networks of arbitrary topological size and complexity.

The distributed nature of the network simulation, isolating nodal data structures from both each other and equally the interface processes requires special software constructs to enable the simulation to proceed effectively. Most important of these constructs are those which enable nodal synchronization within the simulation process and allow access to the nodal processes from the result and modification procedures.

Conventional discrete - event simulators can be divided into those which use synchronous and asynchronous clock mechanisms. The implementation of an asynchronous timing mechanism within a distributed environment is made difficult by the lack of a central data structure in which to store the linked list of forthcoming events. Interaction between local queues, swapping next event times, would require a large transmission overhead and reduces concurrency within the system to essentially zero since only the event at the head of all the combined queues would undergo execution at any time. The synchronous approach, while less flexible for the reasons mentioned above, creates concurrency through event epoch approximation and grouping and was used as the clock mechanism in the model.

In addition the lack of a central data structure also prevents the storage of simulated call information in an easily accessible form and it must be retrieved from the distributed network processors. This information is periodically sampled and the nodal position updated by the same structure. The simulation process on each node is momentarily suspended and a second process activated. This second process allows the flow of information to and from the interface processes controlling the results and network condition respectively, and acts as one link of a linear chain between the interfaces made up of all the nodal processes. The results are passed out as a summary of each nodes history up to the moment of simulation suspension and builds into a picture of the the simulations progress the detail of which can be controlled by the operator.

The individual nodal processes must perform the normal functions of such units in a circuit switched simulation with the added complication that they must also support concurrent communication with their neighbours without deadlocking. The actual structure of the nodal process is reproduced in Figure 3, in which three parallel procedures are connected to a central process by channels while this central process accesses the data structure for each of them. The node operates on a two-level interrupt mechanism. The first layer multiplexes and demultiplexes the input and output channels to and from the network. The second layer feeds requests for process recognition to the central processing section. When a peripheral process is selected by the central process the channel communication is initiated and this is followed by execution of the relevant code segment in the central process for that particular interruption. The structure operates in a way similar to conventional interrupt mechanisms with the advantage of simultaneous data transfer over the links.

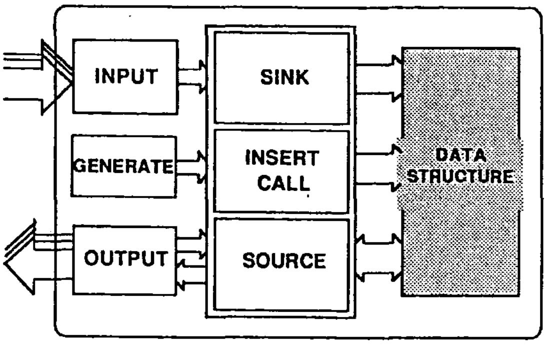


FIGURE 3. NODAL ARCHITECTURE

4. SIMULATION RESULTS

Initial work has concentrated on software development of a fully connected circuit circuit switched model, using a sophisticated call modelling algorithm which seeks to capture the effects of nodal processing ability as well as trunk availability. These models are especially suitable for the multiprocessor environment as the necessary processing for each call is distributed throughout the calls path. The results presented here compare the blocking probabilities of dynamic routing strategy DAR [6], chosen by British Telecom for implementation in system X, with another dynamic algorithm LRI [7], based on learning automata theory and on which work has been done by Bell [8], for the five node, fully

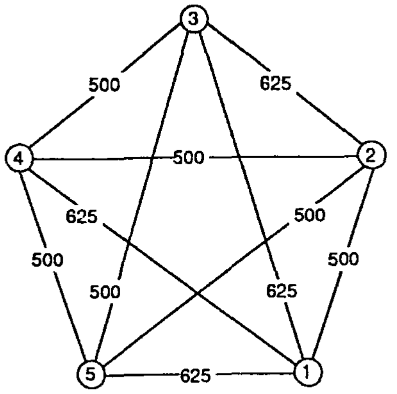
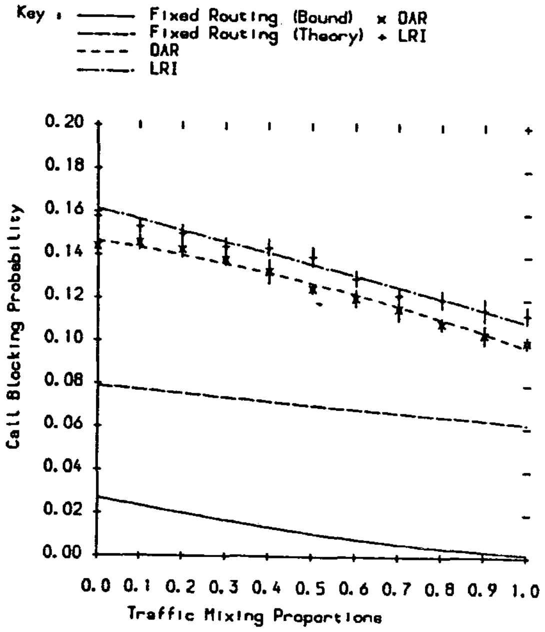


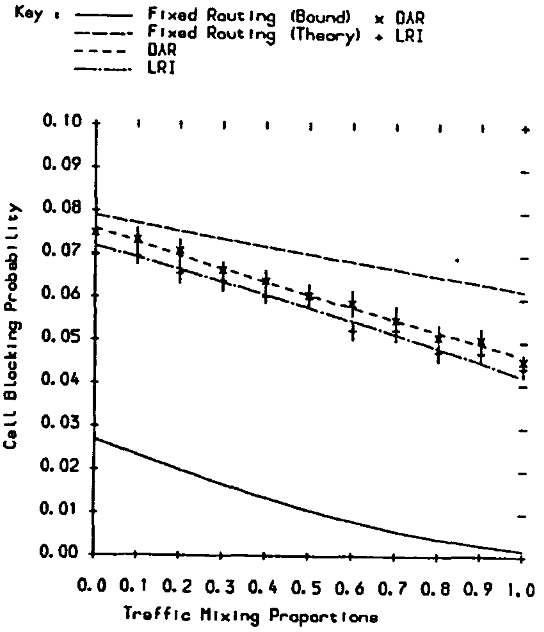
FIGURE 4.
NETWORK UNDER STUDY

interconnected, asymmetrical network reproduced in Figure 4. Each point is the result of over 3/4 of a million calls and took just over two hours to produce using a single transputer to execute the code. This performance is comparable with similar code, written in Fortran, if ran on a dedicated Amdahl 470/V8 mainframe. When exported to an array of transputers it is projected that a performance benefit of O(N) will be achieved for an N node network. The experiment investigated the performance of the two algorithms as the offered traffic matrix smoothly progresses from an overloaded and unbalanced situation to the underloaded but still unbalanced case. This is done by forming composite traffic matrices between the two extremes and plotting the result of these simulations. The aim is to compare the performance of the two algorithms as capacity for alternative routing becomes increasingly available. Figure 5(a) shows the results for the case where each call is allowed to seek an alternative path if the direct path fails, with no restrictions and is only blocked if this attempt also fails. The second graph, Figure 5(b), shows the effect of including a trunk reservation factor (TRP) of 2% over each network link to prevent excess use of tandem paths. Each graph includes two theoretical curves, which calculate the blocking probability using the direct path only (Theory) and a lower bound on performance calculated using Erlangs formulae for a single trunk with capacity and traffic equal to the sum of the individual components in the network (Bound). It is easy to show 'this gives a lower bound on the performance of any routing strategy.



DYNAMIC ROUTING WITH NO OVERFLOW PROTECTION

FIGURE 5 (a)



DYNAMIC ROUTING WITH TRUNK RESERVATION

FIGURE 5 (b)

Two important results can be extracted from these results. First the inclusion of a TRP is important when considering dynamic routing algorithms as traffic approaches its engineered load to restrict overflow traffic to those paths where true spare capacity occurs. Secondly, in this important case, LRI consistently outperformed DAR in these simulation results. Further work is continuing to extend these simulation results and develop analytic models to study these and other algorithms.

5. CONCLUSIONS

The paper has described the hardware and software design strategy for a high-speed multiprocessor network simulator using transputers and when complete the simulator is expected to provide high-speed/low-cost simulations for realistic network modelling and convenient sensitivity analysis. In addition some interesting initial and thought provoking results on circuit switched networks have been presented.

Leading on from the work in circuit switched networks, and using exactly the same hardware system, a packet switched model is being developed for the analysis of flow control and congestion avoidance in these store-and-forward networks. These networks are even more taxing to simulate than their circuit switched counterparts as the time scales involved are orders of magnitude smaller and the transfer of information generally has to be reproduced more accurately as it involves the interlacing of data packets rather than merely the reservation of dedicated trunk capacity. The software currently being constructed implements a simplified form of the first four protocol layers of the OSI model developed by the ISO [9]. The model concentrates on the third layer controlling the routing of packets across the network and flow control within the network. It is hoped to carry out simulation studies into the interaction of dynamic routing strategies and network flow control mechanisms as tools to delay congestion within heavily loaded networks by traffic bifrucation.

ACKNOWLEDGEMENTS

The authors wish to thank the SERC for supporting the work described in this paper.

REFERENCES

[1] Lehnert, A., "A Special Processor for Fast Simulation of Queueing Networks" (ITC-9 Lehnert 1-7).

[2] Barel, M., "A Flexible High Performance Multiprocessor for Data Network Simulation", (ITC-190, Session 3.3 Paper No.9).

[3] Toda, A., et al. "A Parallel Processing Simulator for a Network System using Multimicroprocessors", (Microprocessors and Microsystems, Jan/Feb 1982) pp 15-20.

[4] Pountain, D., "Occam Reference Manual", (INMOS Ltd., March 1987).

[5] Hoare, C.A.R., "Communicating Sequential Processes", (Comms of the ACM, Vol 21, No 8, Aug.1978) pp 666-677.

[6] Gibbens, R., "Some Aspects of Dynamic Routing in Circuit Switched Telecommunication Networks", Rayleigh Prize Essay, (Statistical Laboratory of the University of Cambridge, Jan.1986).

[7] Narendra K.S., et al., "Learning Automata: A Survey", (IEEE Trans on Systems Man and Cybernetics, Vol SMC-24, No.4, July 1974).

[8] Ash G.R., et al., "Servicing and Real Time Control of Networks with Dynamic Routing", (Bell System Tech.Journal, Vol.60, No.8, Oct.1981).

[9] Zimmermann, H., "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection", (IEEE Trans on Comms., Vol.COM-28, No.4, April 1980) pp 425-432.

Transputer-based simulation tool for performance evaluation of wide area telecommunications networks

Highly concurrent wide area networks are best simulated by parallel multiprocessor implementations. **R T Clarke, S J Nichols and P Mars** present a transputer-based simulator that improves performance over conventional sequential simulation techniques

The paper presents the design of a high-speed network simulation machine, developed at Durham University. The simulator uses configured arrays of transputers to create a multiprocessor environment for the modelling of circuit and packet switched networks, taking advantage of the unique features of the transputer for parallel implementation of such models. The design divides easily into hardware and software considerations and the paper concludes with some initial results on performance and projections for the fully populated system.

microsystems simulation telecommunications networks transputers

Analysis of wide area communication networks to determine performance parameters rapidly becomes mathematically insolvable as the detail and complexity of the problem increase. This is compounded when the effects of transient events are taken into account to investigate phenomena such as traffic fluctuation or component failure. The solution here lies in the use of simulation techniques to produce the detailed requirements of such models. Sequential simulations, running on mainframe or personal systems, are limited by the excessive use of CPU time required to achieve the required depth of information. This is because of the inherent mismatch between the conventional Von Neumann architecture on which they are constructed and the highly concurrent operation of the wide area networks themselves. These problems increase as the traffic intensity across the network rises and as the size of the network grows, so that detailed simulations of large networks often become economically and even physically impossible to implement. In this paper the design of a network simulator is presented for use in avoiding these implementation limitations by the introduction of concurrent processing into a dedicated simulation environment. The paper outlines the major features of both the hardware and software design, highlighting the most important considerations for the implementation of the simulation requirements over the loosely-coupled multiprocessor system.

In previous attempts to develop high performance simulation environments[1-3], concurrency has been introduced in the form of pipelined processors, distributed environments controlled from a central control unit and hierarchical bus structures connecting processors simulating network nodes. All these structures suffer from bottlenecks at some point caused by their division of the workload or by the introduction of bus contention between processors requiring communication, which limit their potential performance. The basis of this new design is to capitalize on the advantages of multiprocessor implementation highlighted by these previous attempts, at the same time avoiding the degradation in performance caused by insufficient communication bandwidth or unequal division of processing requirements between independent hardware modules.

In the simulator, transputers are used to represent the nodes in a communications network and the links are used to provide the connections between the nodes. By using the computational power provided by each trans-

School of Engineering and Applied Science, University of Durham, South Road, Durham DH1 3LE, UK

Paper received: 12 August 1988. Re\ised: 5 December 1988

puter and its serial links, a powerful simulation tool has been constructed from an array of these devices. Many applications exist for a low cost/high speed simulation tool within the telecommunications industry. Three areas which are of particular interest are network design, performance evaluation of network routing and flow control, and the use of a simulator as a realtime support tool for existing networks. The potential use of the simulator as a network support tool has greatly influenced the design of both the software and the hardware used to obtain a performance increase over conventional sequential simulation techniques. This has ultimately resulted in a machine which models networks in preference to carrying out *n* individual simulations on *n* transputers to obtain an *n*-fold speed increase. This paper describes some of the software and hardware aspects of a transputer application which has generated great interest within the telecommunications industry.

SIMULATOR HARDWARE

The hardware of the network simulator is made up from three basic units: a user front-end and file server, a graphical results unit, and a 'black box' transputer based simulator (see Figure 1). These units are used to produce a simulation tool which requires the end user to have little or no knowledge of the software or hardware on which the simulation is executed.

The user frontend and the file server are based around an IBM-AT personal computer which contains a transputer that interfaces onto the PC bus. During a simulation run it is used as a tool for drawing up the network topology graphically, entering the initial conditions for a simulation, and to introduce modifications at run time. A further function of the IBM-AT is to act as a result store for the data produced during a simulation run. These results files are stored in MS-DOS format text files which can be analysed outside the simulation environment on either the IBM-AT or on a larger computer system. The graphical results unit is used to display results during the simulation run, making it possible to investigate the behaviour of a network after introducing modifications such as lost links, failed nodes and traffic fluctuations at run time. The graphics unit is based around a commercially available product which has a display resolution of 512 X 512 pixel using a 24-bit colour data field to display any number of 16 M possible colours. The graphical results unit both enhances the power and increases the usability of the simulator as a network support tool.

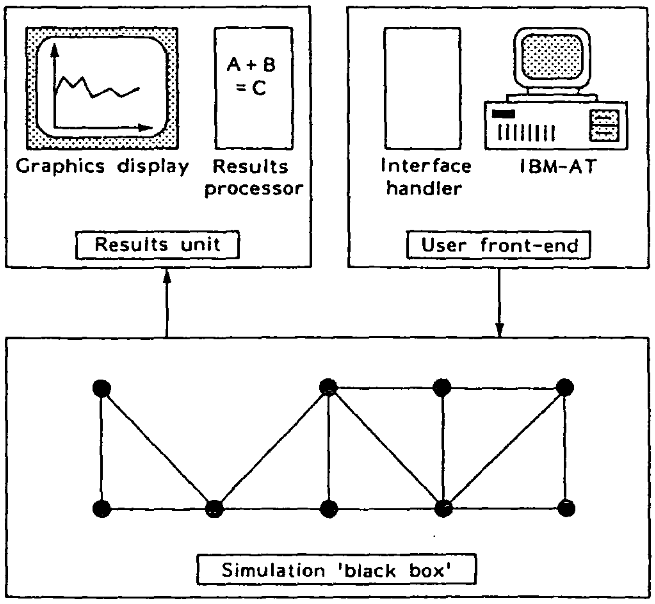


Figure 1. Schematic of the high-speed transputer-based simulator

The 'black box' simulator is constructed from a reconfigurable array of transputers which are used to model the nodes within a communication network by adopting the same topology as the network under investigation. The array consists of up to 32 integer T414-20 transputers[4] which are interconnected via five Inmos programmable cross point switches (IMS C004). These switches are used to reconfigure the array under software control to any given network topology using a master/slave arrangement for nodes with more than four links. Although the limit on the size of the network which can be simulated is dependent on the link switches and the number of transputers available, it is possible to model a network of any conceivable size using the arrangement adopted for the hardware. Two types of board have been specifically developed within the university for the network simulator: a transputer board and a switching card.

The same transputer board design is used for the user interface, the results unit and the simulation nodes. The board is based around the T414-20 transputer which allows it to be upgraded by simply replacing the T414 with a T800-20 floating-point transputer[5]. Each board can address up to 16 Mbyte of 5-cycle dynamic memory using 1 Mbyte memory modules or 4 Mbyte of dynamic memory using 256 kbyte memory modules. In addition to the memory, an 8-bit I/O expansion bus has been included within the design to allow each board to access various types of peripheral devices. The serial transputer links are buffered to allow error-free transmission over short distances and their transmission rates are switch selectable. To reduce the number of ICs on the board, extensive use has been made of programmable logic arrays to provide all the required signals for the memory devices and the Inmos subsystem of daisy-chained reset, error and analyse. The cost of these boards is relatively low compared with similar sized boards on offer in the commercial sector and, because of the design techniques used in their construction, they have a wide number of applications in other fields. The design of the board is tracked onto a double sided printed circuit board of 6U Eurocard size and has a neat and professional appearance.

The switching cards which are used to reconfigure the array of transputers which form the simulator are based around the Inmos C004 cross point switches[6]. Five of these devices are arranged so that an array of 32 transputers can be completely interconnected. Four of the switches are used to connect the transputers while the fifth acts as a master controller for the other switches (see Figure 2). The design of the switching card allows two C004s to be placed with their link buffers on a single card of 6U Eurocard size, once again using double-sided PCB technology. The simulator requires three of these cards, the arrangement of which allows the user interface and the results unit access to the array for the collection of simulation results. The control of the switches is undertaken by the user interface.

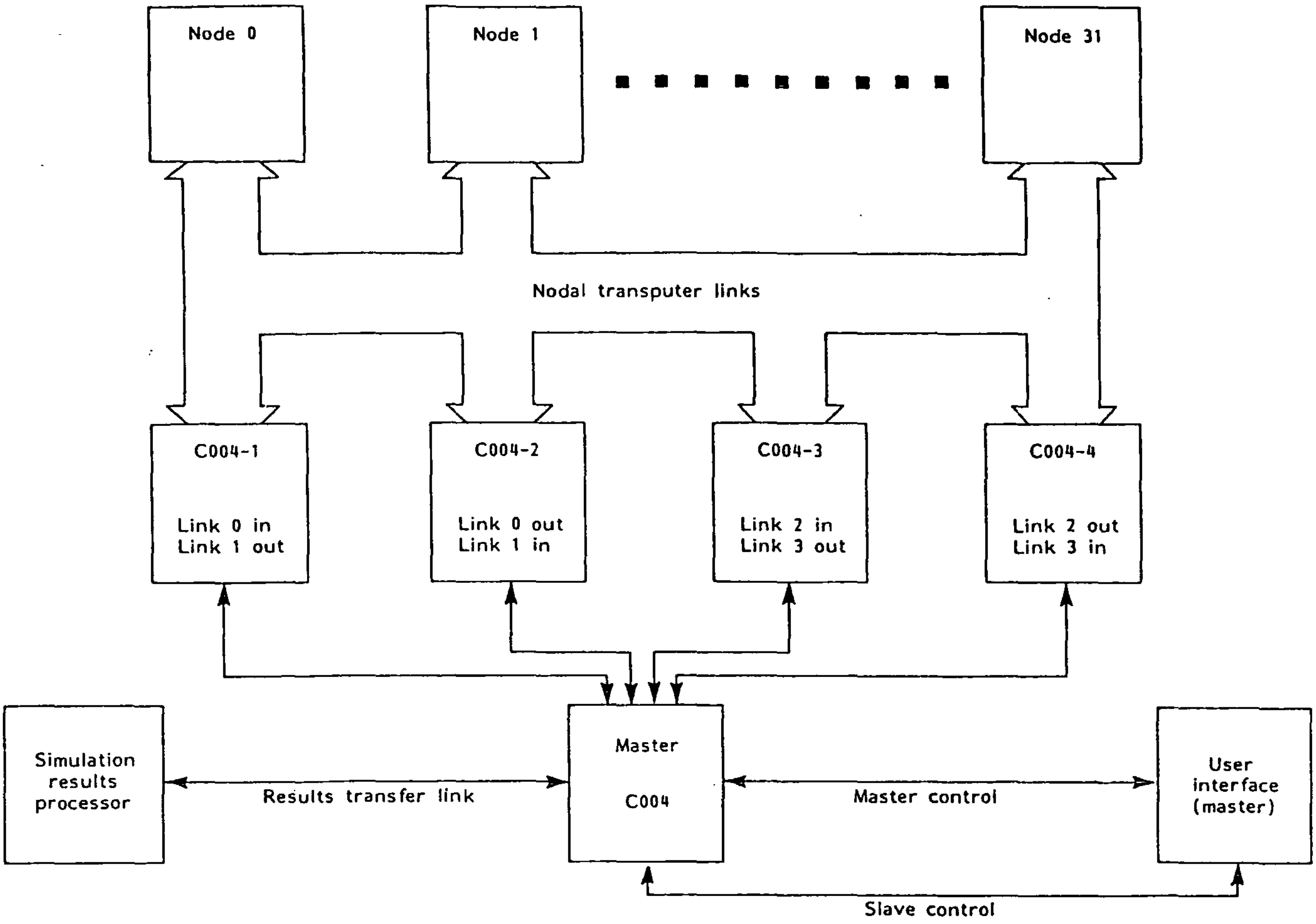


Figure 2. C004 switching arrangement

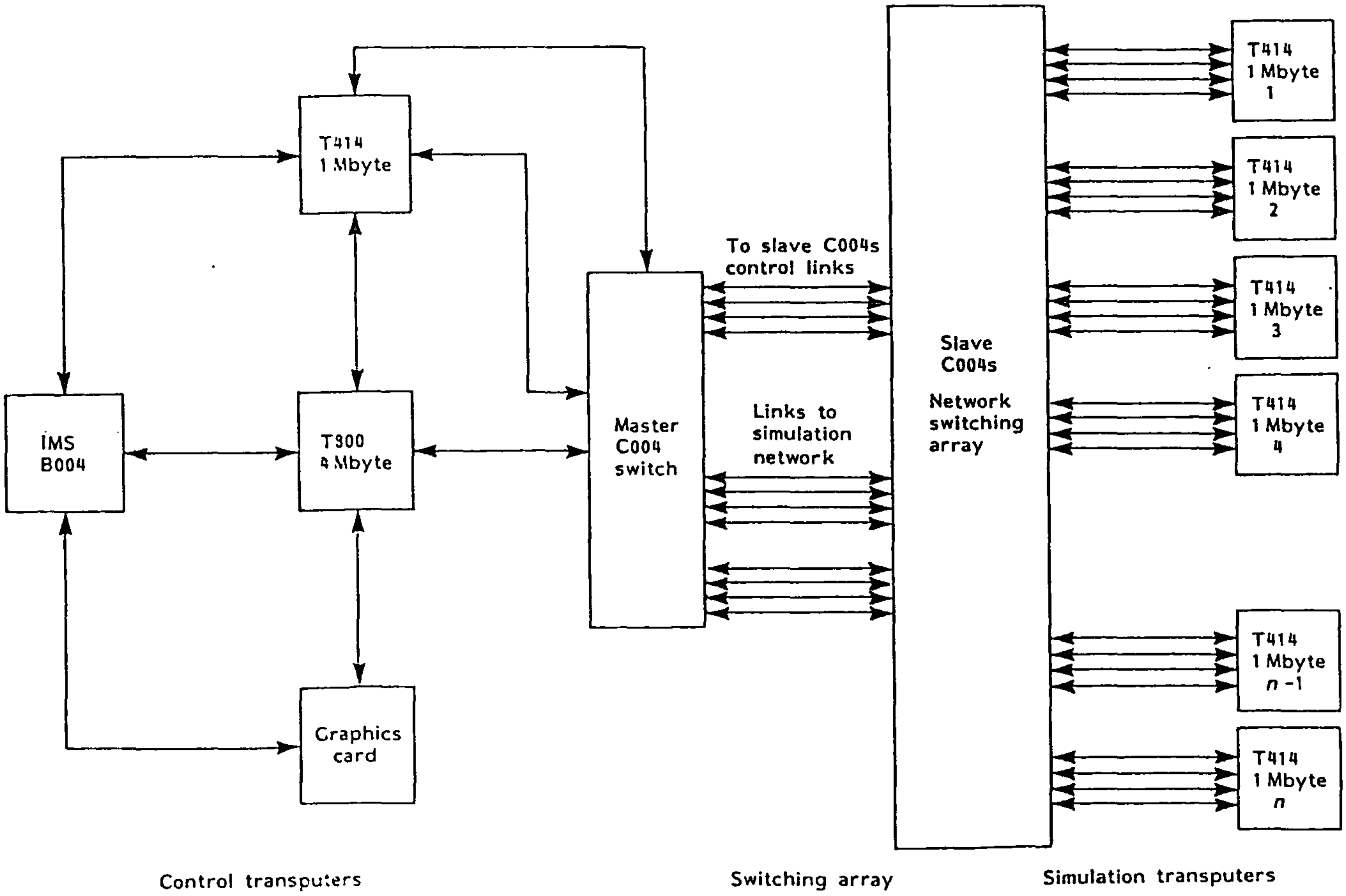


Figure 3. High-speed transputer-based network simulator — hardware configuration

The transputers are interconnected, as shown in Figure 3, to form the network simulator. The transputers to the left on this diagram are the masters, or control, devices and those to the right the simulation nodes, while switching array is in the centre. At present there are ten simulation nodes and four masters, these being the filing unit, the graphical unit, the results unit and the user interface. Using the present arrangement of the switching array, the number of simulation nodes can be expanded up to 32. The hardware is cased in a box about the size of a standard filing cabinet which can accommodate up to 16 simulation nodes, two of the master devices and three switching cards in three 6U Eurocard size racks. The remaining two master devices are resident within the IBM-AT. Each rack has its own 60 A power supply and three cooling fans are provided to dissipate the heat generated by the transputer boards. An additional box will be required to house the two racks which allow the expansion of the prototype simulator up to 32 transputers. However, as mentioned above, the addition of further switching cards results in the simulator being infinitely expandable. It is onto this transputer hardware that the simulation packages discussed below directly map.

SOFTWARE DESIGN

The conceptual design of the simulation environment can be separated into six major areas and mapped directly into OCCAM to produce a three-stage pipeline. By introducing concurrency at this point the processes allow online modification and network parameter calculation, enabling the operator to trace and direct the simulation during execution. The transformation is reproduced in Figure 4 identifying the parallel processes and the major points of data exchange between these processes. The central simulation process can then be further divided into a number of concurrent processes each corresponding to a nodal process and connected by channels recreating the network topology. This combination of pipelined and mesh topologies, implemented on dedicated hardware, in a multiprocessor environment, leads to a powerful and flexible simulation structure which can be expanded to encompass networks of arbitrary topological size and complexity.

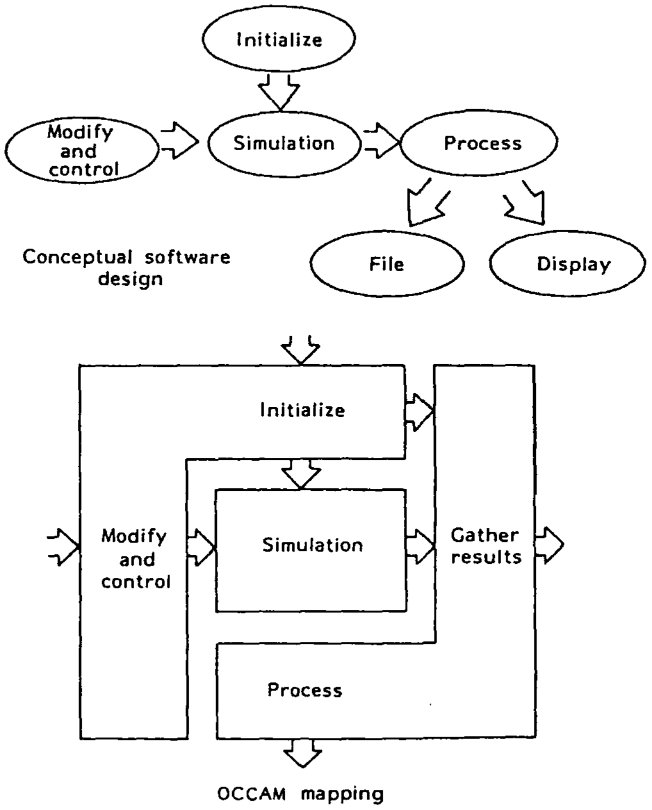


Figure 4. System design strategy

The distributed nature of the network simulation, isolating nodal data structures both from each other and also from the interface processes, requires special software constructs to enable the simulation to proceed effectively. Most important of these constructs are those which enable nodal synchronization within the simulation process and allow access to the nodal processes from the result and modification procedures.

Node synchronization

Conventional discrete event simulators can be divided into those which use synchronous and those using asynchronous clock mechanisms, with asynchronous techniques being favoured in many cases because of their superior handling of event ordering and clock updates.

The implementation of an asynchronous timing mechanism within a distributed environment is made difficult by the lack of a central data structure in which to store the linked list of forthcoming events. Interaction between local queues, swapping next event times, would require a large transmission overhead and reduce the efficiency of the code. In addition, an asynchronous timing mechanism would reduce concurrency within the system to almost zero, since, even if global information could be made available, only the event at the head of all the combined queues would undergo execution at any one time.

The synchronous approach, while less flexible for the reasons mentioned above, creates concurrency through event epoch approximation and grouping and was used as the clock mechanism in the model. Each nodal process executes the events occurring within a time interval and signals the completion of this task to its neighbours. When a nodal process has received final confirmation from its surrounding processes of the completion of their respective processing within a time interval, the local clock can be incremented. This function is complicated by the possibility of events at one node generating further events within the same time interval at a second node. If this second event involves a communication destined for a third node which has already incremented its local clock, this communication cannot be delivered on time. To minimize the probability of this occurring, redundancy is added to the synchronization overhead to 'second chance' communications on each of the channels, effectively double checking that a node is idle before moving on the next time period.

Result access

The lack of a central data structure also prevents the storage of simulated call information in an easily accessible form, and this must be retrieved from the distributed

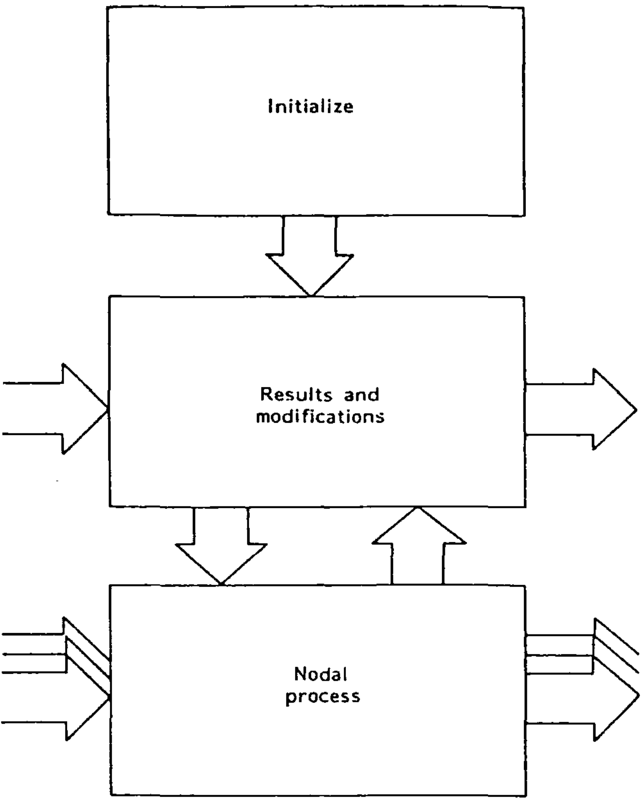


Figure 5. Process interleaving to access system state

network processors if online processing is to be possible. The information is retrieved by periodically sampling the nodal structures. The simulation process on each node is momentarily suspended and a second process activated. This second process allows the flow of information from the interface process through the simulation nodes to the result gathering process. This structure is also used for the implementation of simulation modifications to investigate traffic fluctuations and component failure. The interleaving of processes is shown in Figure 5. In this arrangement each node acts as one link of a linear chain between the interfaces. The results are passed out as a summary of the history of each node up to the moment of simulation suspension and builds up a picture of the progress of the simulations, the detail of which can be controlled by the operator.

Node structure

The individual nodal processes must perform the normal functions of such units in a circuit switched simulation with the added complication that they must also support concurrent communication with their neighbours without deadlocking. The basic structure of the nodal process is reproduced in Figure 6, showing the main subroutines and the channel structure between them. Three peripheral parallel subroutines each communicate with a section of a central subroutine which in turn accesses the data structure. The node operates a two level interrupt mechanism. The first layer multiplexes and demultiplexes the I/O channels to and from the rest of the network using ALT arrays. In addition a third peripheral routine generates new calls for injection to the network.

In the second layer the peripheral routines request recognition from the central subroutine, again through an ALT structure. When a peripheral routine is selected by the central subroutine, the portion of code within the central subroutine is initiated. With this implementation either the input or output routines can be engaged in external channel communication independently without interfering with the access to the central subroutine by the other peripheral processes.

Hardware limitations

One additional problem arises at the last stage — the transfer of the software model to a suitably configured array of transputers. This arises because of the limited number of hardware links on the present range of transputer chips, restricting the connectivity of the network. Using the nodal model described earlier on its own, it is impossible to map the software onto a network in which any member has more than four nearest neighbours. This can be overcome by the definition of a 'slave process', which runs on a separate transputer and acts as a multiplexer/demultiplexer for the I/O stages of the 'master nodal process' and which can be duplicated to provide whatever link availability is required.

RESULTS AND FURTHER WORK

The initial work has been concerned mainly with the development of a simulation package to a model circuit switched telecommunications networks, and it is this which the software section describes above. This software has been extensively tested by comparing theoretical results for various traffic patterns on a four node, fully interconnected network with those produced by simulations carried out on a single transputer model of the simulator. The results from the simulator showed a close correlation to those produced from the theoretical calculations giving a high degree of confidence in the software package which has been developed for the transputer. Work then progressed to comparing learning algorithms[7] with dynamic alternative routing[8] which is used within British Telecom's trunk network.

Both these algorithms simply attempt to redirect blocked telephone calls by chosing an alternative route for the call. To achieve this, a five node, fully interconnected network was devised which was given various traffic patterns which either exceed the capacity of the network or spread the traffic in an uneven distribution.

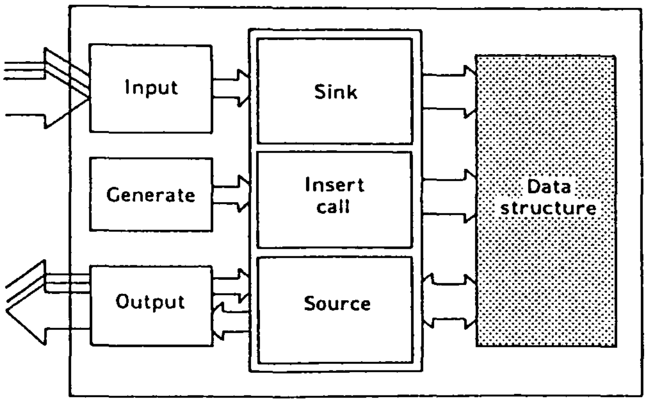


Figure 6. Nodal process structure

The results from these simulations were as expected. Both algorithms performed well by reducing the number of blocked calls on the network, with the learning algorithms producing a slightly lower number of blocked calls.

The performance of any simulation package and the hardware on which it is executed is difficult to measure since much depends on the complexity of the model which is used. The model used for the circuit switched package is definitely complex, and models such things as the time taken to set up and break down a call through a telephone exchange. However, in an attempt to give a performance figure, it can be said that a single transputer will process half a million telephone calls in one hour of CPU time. Thus ten transputers can process 5 M calls in one hour.

At the time of writing, work is under way to develop a software package for the simulation of packet switched networks. This package uses the first four layers of the open systems interconnection (OSI) model developed by by the International Standards Organization (ISO) to allow analysis of flow control and congestion avoidance in this type of network. The problems involved in modelling this type of network are far more complex than for circuit switched networks since the timescales involed are several orders of magnitude lower. However, the required development time for this package will be reduced by reusing some of the parts of the circuit switched package. The packet switched software will use the same hardware configuration, which allows the user to draw simply a network, simulate it, and receive the results at run time via an extensive results processing package, while still retaining the ability to carry out post-simulation analysis of the results. The packet switched package will be used within the university environment to investigate dynamic routing strategies and flow control mechanisms in heavily loaded packet switched networks.

CONCLUSIONS

This paper has described the hardware and software designs behind a high-speed multiprocesor telecommunications network simulator based around the transputer. The applications of such a machine have been described and the initial results from the circuit switched simulation package have been mentioned together with the forthcoming work on packet switched networks. The simulator is expected to provide a high-speed simulation of realistic network models and convenient sensitivity analysis of communication networks at low cost.

ACKNOWLEDGEMENTS

The authors wish to thank SERC for supporting the work described in this paper.

REFERENCES

1 **Lehnert, R** 'A special processor for fast simulation of queueing networks' *ITC 9* (1985) 1-7

2 **Barel, M** 'A flexible high performance multiprocessor for data network simulation' *ITC 10 session 3.3 paper No 9* (1986)

3 **Toda, A *et al.*** 'A parallel processing simulator for a network system using multimicroprocessors' *Microprocessors Microsyst.* Vol 6 No 1 (Jan./Feb. 1982) pp 15-20

4 *IMS T414 transputer* Inmos, Bristol, UK (February 1987)

5 *IMS T800 transputer* Inmos, Bristol, UK (February 1987)

6 *IMS C004 programmable link switch* Inmos, Bristol, UK (April 1987)

7 **Narendra, K S and Mars, P** 'The use of learning algorithms in telephone traffic routing — a methodology' *Automatica* Vol 19 No 5 (1983) pp 495-502

8 **Gibbens, R J, Kelly, F P and Key, P B** 'Dynamic altemative routing — modelling and behaviour' *ITC12, Turin, Italy* (June 1988)

Roy Clarke *graduated from Brighton Polytechnic, UK, in 1986 with an MSc in microprocessor, control and application. He is currently a research assistant at Durham University. He is studying for a PhD in high speed simulation of telecommunication networks.*

Simon Nichols *graduated in 1986 with a first class honours degree in applied physics and electronics from Durham University. He is currently studying for a PhD. His areas of interest include routing in circuit and packet switched networks, learning automata and distributed systems.*

Phil Mars *is chairman of the School of Engineering and Applied Science and Professor of electronics at the University of Durham. He is the author of a research monograph and over 80 research papers in stochastic systems, learning algorithms and communications networks.*

Presented at 7th U.K. Teletraffic Symposium, University of Durham, 4th April 1990

ANALYTIC MODELLING OF ASYMMETRICAL CIRCUIT SWITCHED NETWORKS USING DYNAMIC ROUTING

S. J. Nichols and P. Mars

In this paper the classical model of blocking in a symmetrical, fully connected circuit switched network which has been used extensively by such authors as Yum, Krupp and others (1) (2) is extended to cover asymmetrical networks. The steady state convergence characteristics of a number of different dynamic routing schemes are incorporated into the model as sets of non-linear equations over the routing space to enable these schemes to be modelled under asymmetrical traffic and trunk distributions. Two major areas are studied, the performance of networks under overload and the phenomena of network instability, both with and without the inclusion of a trunk reservation parameter over the network trunk groups (3) (4).

The presentation considers the following routing disciplines; random routing (as a lower bound on performance), a form of proportional routing based on a global strategy used by Bell-Northern, Dynamic Alternate Routing (DAR) installed in British Telecom's System X and the automata based algorithm L_{RI} (5) (6). In each case the algorithm is used to select a single alternative path via a tandem node if the call is blocked on the direct path. Analytic results for a fully connected four node network, subject to structured asymmetry, are presented and shown to be equally valid for larger network configurations subject to a similar structure. The results suggest that proportional routing and DAR adopt a very similar approach to the allocation of overflow traffic to alternative paths. However L_{RI} displays a more complex behaviour which leads to an interesting deviation in its behaviour in comparison to the other strategies, resulting in the equalisation of the GOS seen by each of the traffic sources, or the minimisation of the deviation from this state.

To explain the way in which each of the routing strategies behaves in an unconstrained environment we return to the conditions under which each is governed. Random routing directs traffic equally over all the alternative routes at each node. Each node selection is isolated and does not react to trunk or traffic imbalances. In this work it represents an upper bound, or least intelligent option.

Proportional routing assigns traffic according to trunk availability. DAR uses not trunk availability but trunk blocking rate as criteria for traffic splitting, defined as the blocking probability over a link multiplied by the probability of using that link. Both strategies involve very different ways of selecting routing probabilities but produce very similar solutions under asymmetrical conditions. In both cases the routing policy at each node makes identical routing decisions as its network neighbours faced with the same environment. From single source models blocking rate can be seen to be a convex function of selection probability. Similarly the plot of normalised spare capacity vs selection probability is also found to be convex. In the multiple source model this produces an array of constraints based on convex functions over the multidimensional routing space. This suggests that any solution will be unique and therefore

The authors are with the School of Engineering and Applied Science, University of Durham.

by definition requires identically situated sources to behave in an identical manner to prevent multiple solutions by simply swapping indicies in the network.

LRI routing involves the equalisation of blocking over each of the alternative paths to a destination from each node. The blocking function is a strictly increasing but non-convex function of selection probability. Therefore it will form a non-convex set of constraint equations and the argument outlined for DAR and proportional routing cannot be applied. Multiple solutions can and do exist in the solutions to the network models used. In each case a new and equally valid solution can be formed by relabelling the model in a suitable fashion to maintain identical traffic over each link while rearranging the selection probabilities of each of the sources over their alternative paths. However a full explanation as to why the equalisation of the individual blocking probabilities for each source leads to an equalisation of the GOS of each source remains an open problem. Despite this the result remains one of the most important conclusions of this presentation.

The behaviour of the strategies when a TRP is applied is also quite different. Both proportional routing and DAR rearrange themselves according to their new environment, splitting the overflow traffic over the available alternate pathways in the same way as in the case of the unconstrained network. In each case the function used to select the traffic distribution over the alternative paths for each source, i.e. spare capacity or blocking rate, covers a range of values for the variable of path selection probability sufficient to allow an exact solution. In contrast the blocking probability over a link is relatively unaffected by the selection probability of a single traffic source using that link as an alternative route. In consequence LRI is often unable to equalise blocking over all its paths for all its sources, especially those with very different alternative paths to a destination and this gives rise to the *LRI Routing* solutions. The imposition of a TRP further inhibits this ability, by inhibiting the traffic and therefore the influence any one source may have on a link. Again this drives the LRI strategy to deterministic routing for traffic sources in which the maximum allowed traffic overflow is insufficient to equalise blocking, but in this instance the asymmetry required to initiate such action is very small.

The work on instability has two important results. First, it is clear from both the analytic and simulation models that the addition of even a small trunk reservation results dramatically improves the performance of the network. However, without such a device instability is predicted by the analytic models of all the dynamic strategies to varying degrees over a range of traffic and trunk distributions. In contrast the simulation models for a four node network over th affected regions suggest only a gradual increase in blocking producing a significantly better performance. This and other work at higher traffic intensities suggest that the analytic assumptions are not valid over these regions for small networks.

Acknowledgements

The authors wish to acknowledge the support of the SERC for this work. Simon Nichols also wishes to thank Reuters and the European RACE programme for financial support.

References

1. KRUPP, R.S.: "Stabilisation of alternate routing networks", IEEE Int.Conf. on Comm. Philadelphia, June 1982, pp.31.2.1, 31.2.5.

2. YUM, T.G. and SCHWARTZ, M.: "Comparison of routing procedures for circuit-switched traffic in non-hierarchical networks", IEEE Trans.on Comms, COM-35, May 1987, pp.535-544.

3. AKINPELU, J.M.: "The overload performance of engineered networks with non-hierarchical and hierarchical routing", AT & T Bell Lab Tech.J., 63, 7, Sept.1984, pp.1261-1281.

4. ACKERLEY, R.G.: "Hysteresis-type behaviour in networks with extensive overflow", Br.Telecom J., 5,4, Oct.1987, pp.42-50.

5. NARENDRA, K.S. and MARS, P.: "The use of learning algorithms in Telephone Traffic Routing - A methodology", Automatica, 19, 5, pp.495-502, 1983.

6. GIBBONS, R.J, KELLY, F.P. and KEY, P.B.: "Dynamic alternative routing - modelling and behaviour", ITC 12 Torino, Italy, June 1988, Paper 3.4 A3.