ADAPTIVE CONTROL OF COMMUNICATION NETWORKS USING LEARNING AUTOMATA

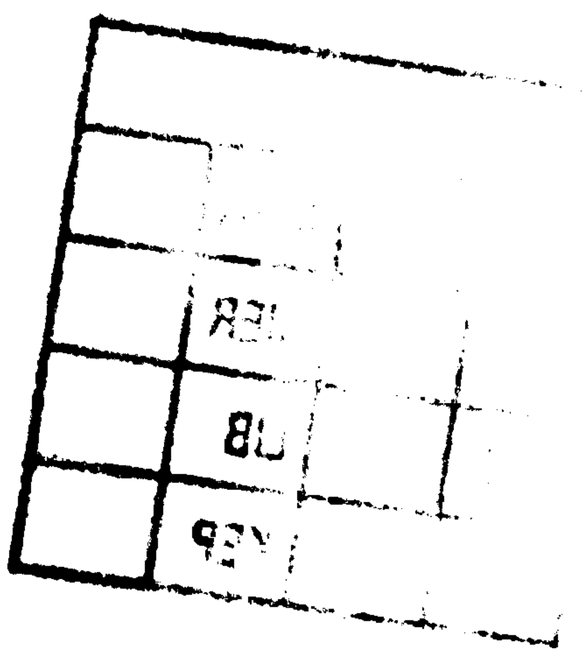
by

MICHAEL S CHRYSTALL BSc

A thesis submitted in partial fulfilment of the requirements of the Council for National Academic Awards for the Degree of Doctor of Philosophy (Ph D)

School of Electronic and Electrical Engineering
Robert Gordon's Institute of Technology
Schoolhill
Aberdeen

March 1982



ACKNOWLEDGEMENT

I wish to express my gratitude to the many people who have supported and encouraged my work over the last three years. In particular, thanks to Dr J Eades and Mr Angus Brown of RGIT for their programming advice and knowledge and to Mr B McK Davidson and Mrs A Hobbs for their invaluable efforts in preparing this thesis. Also, I must acknowledge the assistance of Dr Wyn Price of NPL, who acted as my external advisor and Professor K S Narendra of Yale University, U S A for his continued interest in the automata routing project and for allowing me to visit Yale.

Finally, I am truly indebted to my supervisor, Professor P Mars, who not only enthusiastically guided my study, but also advised me on numerous other topics.

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 referenced.

In the course of the research, the following were included in an approved programme of advanced studies.

1 SRC Vacation School on "Instrumentation and Measurement for Control Systems" held at the City University, London in July 1979

2 Visiting Pre-Doctoral Fellow with the Systems and Information Science Group, Department of Engineering and Applied Science, Yale University, Connecticut, U S A, from August to December 1979

3 SRC Vacation School on "Stochastic Processes in Control Systems", held at the University of Warwick in March 1981

Michael J Angstadt

ABSTRACT

This research investigates communications network routing procedures, based on distributed learning automata concepts for circuit and packet switched networks. For this application, the learning automaton is shown to be an ideal adaptive control mechanism, with simple feedback and updating strategies which allow extremely practical implementations and perform very close to the desired optimum.

In this thesis, the nature of learning automata routing schemes are explored by analytical and computer simulation techniques, primarily developing an elementary understanding of the automata routing and adaption process. Using simple circuit and message switched networks the conditions for minimum blocking probability and average delay are established and compared with the equilibrium behaviour of learning automata operating under alternative reinforcement algorithms.

Later, large scale simulations of real networks are used to demonstrate and relate the learning automata scheme to existing routing techniques. These experiments, which are performed on sophisticated simulation packages produced for this study, take as examples hierarchical and general structured telephone networks and packet switched communications networks configured with both virtual call and datagram protocols. In addition, studies under failure mode conditions, including link, node and focussed overloads, conclusively demonstrate the superior performance afforded by the learning automata routing approach.

CONTENTS

CHAPTER 1

REVIEW OF LEARNING AUTOMATA

1.1 Introduction

Learning as defined in[1], is described as any permanent change in behaviour resulting from past experience, a learning system therefore exhibits the ability to improve on previous operations with regard to some well defined performance criterion. The automaton approach to learning follows logically from this theme, whereby the learning automaton interacts with a random environment, improving on past performance by suitably varying the output strategy in relation to the environmental reaction.

Typically a learning automaton operates in such a manner as to choose an optimal action from an allowable set and to apply the selected action to a random environment. In turn, the environment responds with a feedback signal which initiates an up-dating of the automaton state and consequently the future action selection process.

Consider the automaton/environment configuration as shown in Figure 1.1. This shows an automaton capable of performing r-actions, operating on a single random environment which in general is a media of which little or no apriori information is available. Further, the operation of the combination is at discrete intervals of time, the action and subsequent response at stage n leading to the automaton state of stage n+1.

1.2 The Learning Automaton

The learning automaton (Figure 1.2a) as considered in this study may be regarded as a device with specified input and output ports, supplemented by an internal mechanism /

mechanism responsible for modifying and recording an internal state.

To study the operation of the automaton in more detail, the quintuple $A(n)$ is defined

$$A(n) = \left\{\underline{\beta}, \underline{\alpha}, \varphi, F(\beta,\varphi), G(\varphi)\right\}$$

The constituent components of $A(n)$ are as follows:-

$\underline{\beta}$: Input Set

The input to the automaton $\beta(n)$ at stage n is an element of the set $\underline{\beta}$ and is normally termed the response

$$\beta(n) \epsilon \left\{\beta_1, \beta_2 \ldots\right\}$$

The exact form of this response will be dependent on the nature of the environment, i.e., discrete or continuous, as described more fully in a later section.

$\underline{\alpha}$: Output Set

The output or action $\alpha(n)$ of the automaton at any instant n is a member of the finite set $\underline{\alpha}$.

$$\alpha(n) \epsilon \left\{\alpha_1, \alpha_2 \ldots \alpha_r\right\}$$

$\underline{\varphi}$: State Set

The state of the automaton at n, denoted by $\varphi(n)$, is also a member of a finite set $\underline{\varphi}$.

$$\varphi(n) \epsilon \left\{\varphi_1, \varphi_2 \ldots \varphi_s\right\} \qquad s \geq r$$

In effect, the state represents the memory of the automaton and records the past experience.

$F\left(\beta(n),\varphi(n)\right)$: Transition Function

The transition function F provides a means of determining the next state of the automaton $\varphi(n+1)$, given the response $\beta(n)$ and the current state $\varphi(n)$.

$$\varphi(n+1) = F\left\{\beta(n),\varphi(n)\right\}$$

The /

The role of the transition function is that of an up-dating operation such that the automaton state can be modified to reflect the experience gained through the received response.

G $(\varphi(n))$: Output Function

The output function G provides the necessary mapping between the state of the automaton $\varphi(n)$ and the performed action $\alpha(n)$.

$$\alpha(n) = G\left\{\varphi(n)\right\}$$

The above description provides an extremely general introduction to the learning automaton which can exist in several forms. Further classification and development of the concept will be present later along with suggested application areas.

1.3 The Environment

The term environment is used in learning automaton studies to describe some random media on which the automaton will operate. Normally, insufficient or no apriori information is available to describe the dynamics of this media, and it is under these circumstances that the learning automaton is an attractive alternative to classical techniques. For this study, the environment may be defined mathematically by the triple E(n), Figure 1.2b.

$$E(n) = \left\{\underline{\alpha}, \underline{\beta}, \underline{c}\right\}$$

where $\underline{\alpha}$ represents the input set, $\underline{\beta}$ the output or response set and c the penalty set.

A further classification of environmental types leads to the definition of three response models:

1 P-model

2 Q-model

3 S-model

In /

In the P-model the response to an action is of a binary nature, i.e., 1 or 0, penalty or reward.. If, however, the response is continuous in the region (0, 1) then the environment is referred to as an S-model. Similarly, the Q-model is suited to the intermediate case when the response can be one of a finite set of discrete values in the range (0, 1).

$\underline{\alpha}$: Input Set

The input to the environment $\alpha(n)$ corresponds to the performed action of the automaton which is a member of the set $\underline{\alpha}$.

$$\alpha(n) \in \left\{ \alpha_1, \alpha_2, \ldots \alpha_r \right\}$$

$\underline{\beta}$: Output Set

In the conventional automaton/environment configuration, the output of the environment $\beta(n)$ is the received response of the automaton. As described previously the exact nature of this is dependent on the environment but in general $\beta(n)$ is an element of the set $\underline{\beta}$.

$$\beta(n) \in \left\{ \beta_1, \beta_2 \ldots \right\}$$

$\underline{c}$: Penalty Set

In the operation of the described learning system, the environment reacts to the performed action with a response which is characterised by the penalty set of the media. The penalty set therefore indicates the desirability of an action, the media responding with a penalty signal or penalty weight depending on the environmental model.

For the P-model, the penalty set dictates the probability that a given action will receive a penalty response $\beta(n) = 1$. The penalty set $\underline{c}$ therefore consists of r-penalty /

penalty probabilities.

$$\underline{c} = \left\{ c_1, c_2, \ldots c_r \right\}$$

where

$$c_i = \text{prob} \left[\beta(n) = 1 \middle| \alpha(n) = \alpha_i \right]$$

which is the probability that given action α_i was performed, the response will be a penalty.

The penalty set for the S and Q-models are best described by a family of distribution functions, one for each action. The response to an action will therefore be a random variable limited to the range (0, 1). For the S and Q-models the penalty sets are denoted by

$$\underline{c} = \left\{ s_1, s_2 \ldots s_r \right\}$$
$$\underline{c} = \left\{ q_1, q_2 \ldots q_r \right\}$$

where s_i is a continuous distribution function of the penalty weight S_i and q_i is a discrete function for the penalty weight Q_i.

In this study of communication network routing, all three response models play a significant role, the P-model in circuit switched networks and both Q and S-models in the store and forward type of network. A complete discussion of these aspects is presented in later chapters.

1.4 Classification of Learning Automata

Learning automata are classified according to the nature of the functions F and G which can best be considered as matrix operators, transforming $\varphi(n) \rightarrow \varphi(n+1)$ and $\varphi(n) \rightarrow \alpha(n)$. This will be demonstrated and used to illustrate the main types of automata in the following examples.

If the elements of both matrices are purely deterministic, /

deterministic, i.e., the operation of the automaton can be uniquely described by a logical operator then the device is defined as fixed structure deterministic. Figure 1.3a gives an example of such an element including the desired form of the F and G operators. Alternatively the transition probabilities (valued at 0 or 1 in the deterministic automaton) can be allowed to take on any value (subject to row and column sum constraints), thus describing an automaton which is random in nature and consequently classified as a fixed structure stochastic automaton. A general example of this type is shown on Figure 1.3b, the matrix element p_{ij} representing the probability of a transition from state i to state j. Fixed structure learning automata were first considered in an original paper[2] and developed further as reported in[3][4] using the technique of Markov chains.

In comparison variable structure stochastic automata have the potential of greater flexibility and application, the concept being initially proposed in[5]. This type of automaton is noted by the ability to modify the elements of the transition matrix by means of an up-dating or reinforcement algorithm. Such an algorithm normally exists in one of two forms

(a) state transitional form

or

(b) total state form,

the latter being used in conjunction with the state probability vector $\underline{p}(n)$, governing the choice of state at stage n. In most cases the state set relates directly to the action set (r = s) which allows the state probability vector to be equated with the action probability vector. Variable structured automata have been extensively investigated with the benefit of /

of mathematical tools such as Markov process theory and in addition the results from extensive computer simulation experiments[6].

1.5 Stochastic Automata with Variable Structure

Stochastic Automata with Variable Structure (SAVS) as detailed in the previous section are best described in terms of the state probability vector $\underline{p}(n)$ which can also be the action probability vector. This vector relates to the probability that at instant n a certain action will be performed.

$$\underline{p}(n) = \left\{ p_1(n), p_2(n) \ . \ . \ . \ p_r(n) \right\}$$

where

$$p_i(n) = \text{prob}\left[\alpha(n) = \alpha_i\right]$$

The reinforcement scheme is utilised to continually revise this vector as part of the learning process. By convention, the reinforcement scheme is represented by the function T such that

$$\underline{p}(n+1) = T(\underline{p}, \alpha, \beta).$$

As shown, the up-dated vector is a function of the current action probability set, the performed action and received response.

For further consideration of the SAVS the re-introduction of A(n) as

$$A(n) = \left\{ \underline{\alpha}, \underline{\beta}, \underline{p}, T(p, \alpha, \beta) \right\}$$

is a more suitable and indeed more clear form (Figure 1.2c). Again $\underline{\alpha}$ and $\underline{\beta}$ represent the action and response sets with $\underline{p}$ and T describing the action probability vector and reinforcement algorithm respectively.

Clearly, the reinforcement algorithm plays a crucial role in the operation and behaviour of a variable structure automaton /

automaton and as such must be given a great deal of careful and detailed consideration. Classification of such algorithms are normally carried out on the basis of their performance (to be discussed later) or on the nature of the function used in the scheme. If, for example, $\underline{p}(n+1)$ is a linear function of $\underline{p}(n)$, then the scheme is obviously termed linear . In contrast, schemes involving higher orders of $\underline{p}(n)$ are described as non-linear[7] with a further class of algorithm, the hybrid scheme[8] combining linear and non-linear components. In the majority of applications, the linear schemes are of the greatest interest, the non-linear and hybrid algorithms only giving slightly better convergence rates under special circumstances.

Several variations of the linear algorithm have been considered including

L_{R-I} linear reward inaction

L_{R-r} linear Reward reward (different degrees of reward)

L_{R-P} linear reward penalty

L_{p-P} linear penalty Penalty (different degrees of penalty)

L_{I-P} linear inaction penalty

In this thesis, emphasis is placed on the L_{R-I} and L_{R-P} schemes as they have in the past been shown to be superior to the alternatives. Two specific reinforcement algorithms are now considered, the P-model L_{R-P}[8] and the S-model L_{R-I}[9].

P-model L_{R-P} for r-Actions

Reward on $\alpha(n) = \alpha_i$ $\beta(n) = 0$

$$p_{j\neq i}(n+1) = (1-a)\,p_j(n) \qquad 0 < a, b < 1$$

$$p_i(n+1) = 1 - \sum_{j\neq i} p_j(n+1)$$

Penalty /

Penalty on $\alpha(n) = \alpha_i \quad \beta(n) = 1$ . . . 1.1

$$p_i(n+1) = (1-b)p_i(n)$$
$$p_{j\neq i}(n+1) = p_j(n) + \left\{\frac{b}{r-1}\right\} p_i(n)$$

To implement the L_{R-I} algorithm, the punishment coefficient b is set to zero, effectively rendering the penalty sections of the algorithm in-operative.

S-model L_{R-I} Algorithm for r-Actions

$\alpha(n) = \alpha_i \quad \beta(n) = S_i \qquad 0 \leqslant S_i \leqslant 1$

$$p_i(n+1) = p_i(n) + a(1-S_i)\left[1-p_i(n)\right] \qquad \text{. . . 1.2}$$
$$p_{j\neq i}(n+1) = p_j(n) - a(1-S_i)p_j(n) \qquad 0 < a < 1$$

1.6 Performance Criteria

In order that the behaviour of various learning schemes can be compared and assessed, it is necessary to set quantative norms of behaviour. One convenient and frequently used measure is the average received penalty M(n) which is defined as

$$M(n) = E_r\left[\beta(n) \,\middle|\, p(n)\right] = \sum_{i=1}^{r} p_i(n)c_i$$

For most learning situations the initial action probabilities will be un-biased, i.e., equal and therefore the initial expected penalty M_o is given by

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

In studies of learning automata, the asymptotic behaviour of M(n) relates to the convergence properties of the different reinforcement algorithms which can be classified according to the following behavioural norms.

Expedient /

Expedient

A learning automaton is described as expedient if convergence is such that

$$\lim\ n\to\infty \qquad E\left[M(n)\right] < M_o$$

This corresponds to the automaton evolving such that it does better than a purely random structure, attempting each action with equal probability. One example of this behaviour is the L_{R-P} algorithm which theory and simulation has shown to be expedient when operating on a statistically stationary environment.

Optimal

If the behaviour of the automaton is such that

$$\lim\ n\to\infty \qquad E\left[M(n)\right] = c_m \text{ where } c_m = \text{Min}\left[c_i\right]$$

then the device is described as optimal. This is the case when the automaton converges to consistently select the action which corresponds to the minimum penalty probability. In practice however, an optimal automaton is not truly realisable, the L_{R-I} scheme for instance being best described as ϵ optimal.

$$\lim\ n\to\infty \qquad E\left[M(n)\right] < c_m+\epsilon \qquad \text{where } \epsilon \text{ is small positive value}$$

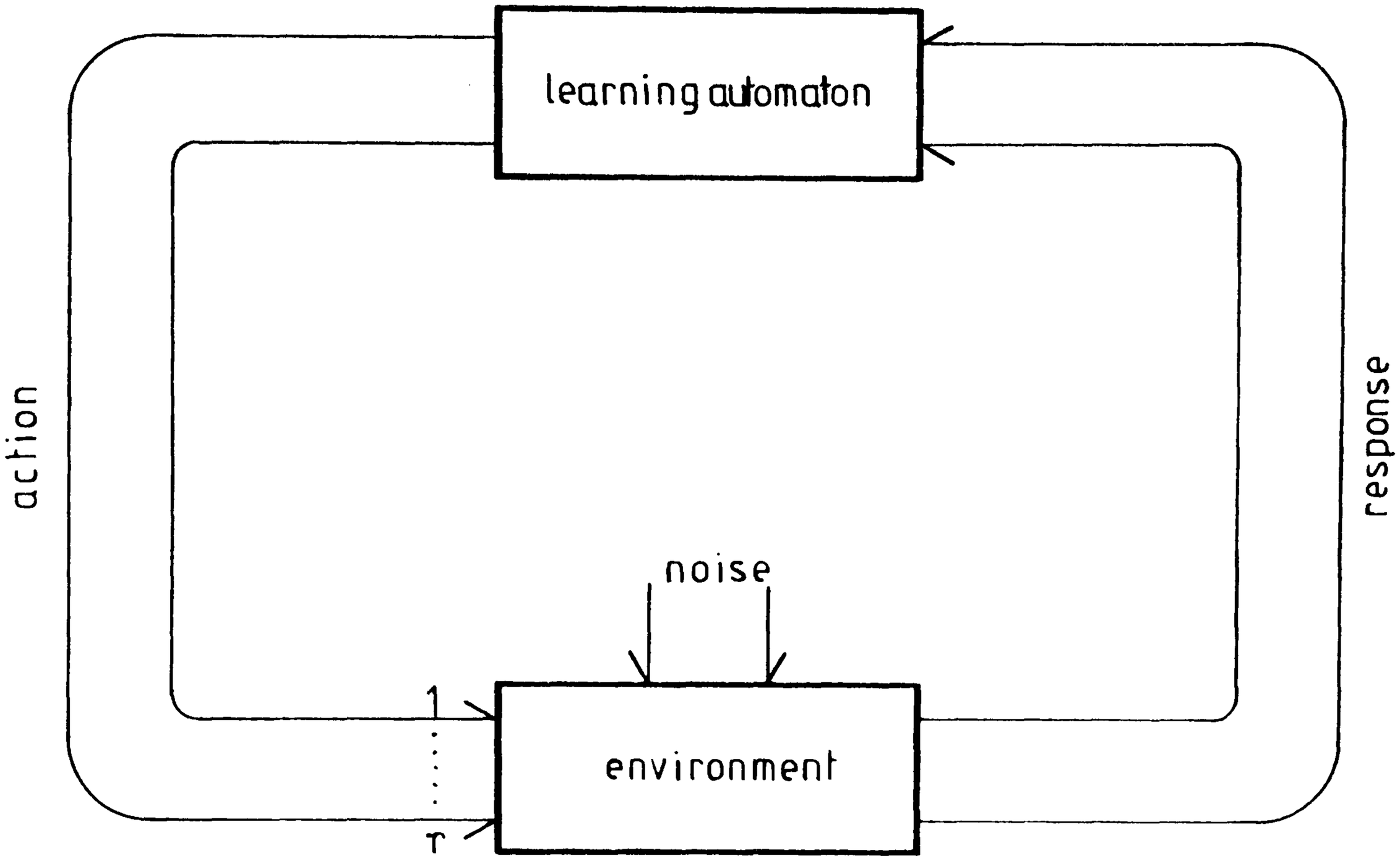
1.7 Summary and Applications

The aim of this chapter has been to provide a general review of the form and behaviour of learning automata, considering both fixed and variable structure types. Particular emphasis has been placed on the variable structure stochastic automaton with details provided on two of the most important reinforcement schemes, the P-model L_{R-P} and the S-model L_{R-I}. Additional sections have discussed /

discussed the nature of the random environment and considered behavioural norms for the various schemes.

Several application areas are seen as suitable for the stochastic learning automaton, generally involving control and allocation strategies. One of the most important and best suited is the adaptive routing problem for circuit and packet switched communication networks, both of which may be regarded as distributed stochastic allocation problems. Typically, independent automata at intermediate nodes (Figure 1.4a) select an outgoing link for transmission (action set) and dependent on the outcome of the operation (response) the automata are up-dated. A further allocation problem concerns the optimal scheduling[10] of jobs in a multi-processor computer system (Figure 1.4b), the goal of the automaton being to maximise the job throughout by placing appropriate jobs on a suitable processor.

In adaptive control, the use of automata as optimising components for multi-modal performance indices has been studied. Experiments using multi-modal functions[11] and plant[12] have shown the technique to be clearly suitable for locating global stationary points, even when severe noise masks the function which could possibly consist of several maximum and minima (Figure 1.5a). The use of hardware learning automata also shows great promise particularly in adaptive control techniques such as model reference schemes (Figure 1.5b) where a need exists for a rapid global searching component. In this area the use of stochastic processing elements[13] has proved an extremely efficient design tehcnique and using suitably developed components, multi-state automata have been constructed[14].



P model : $\beta \in \{0, 1\}$

$\beta = 0$ reward

$\beta = 1$ penalty

Q model : $\beta \in \{\beta_1, \beta_2 \cdots \beta_k\}$

$0 \leq \beta_i \leq 1$

S model : $0 \leq \beta \leq 1$

$\beta \in R$

Figure 1.1 Learning automaton / environment configuration

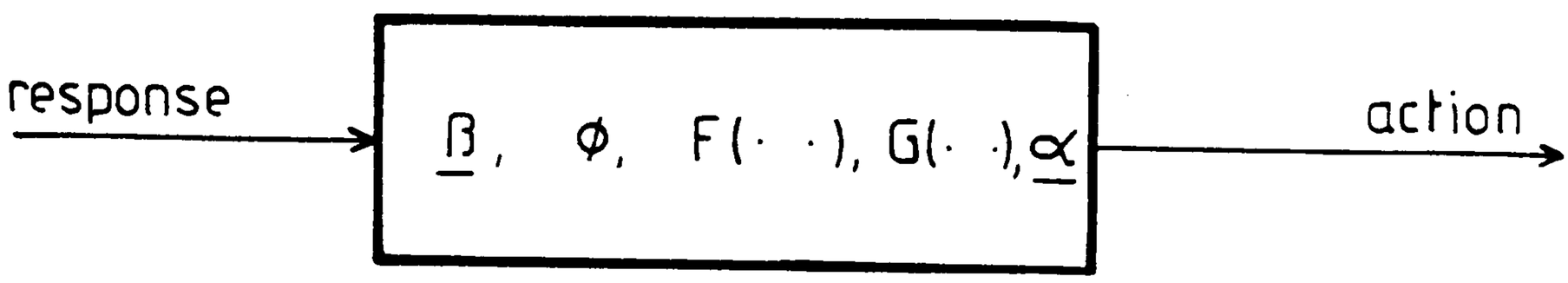


Figure 1.2a Learning automaton

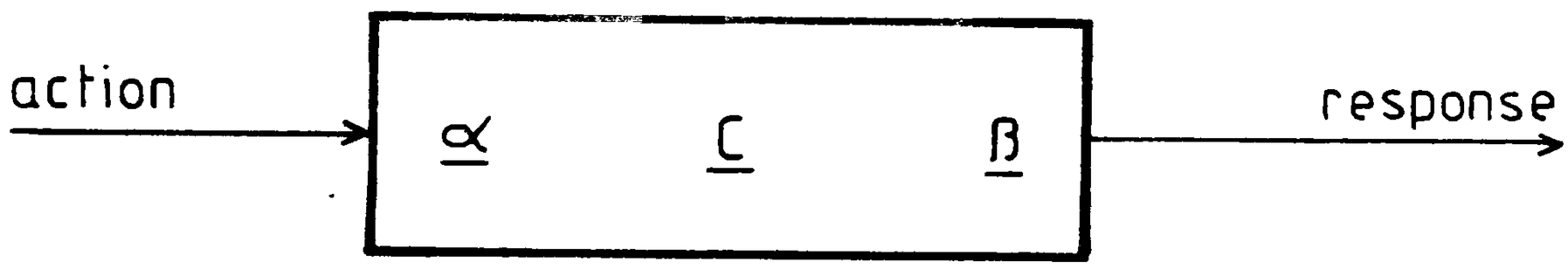


Figure 1.2b Environment

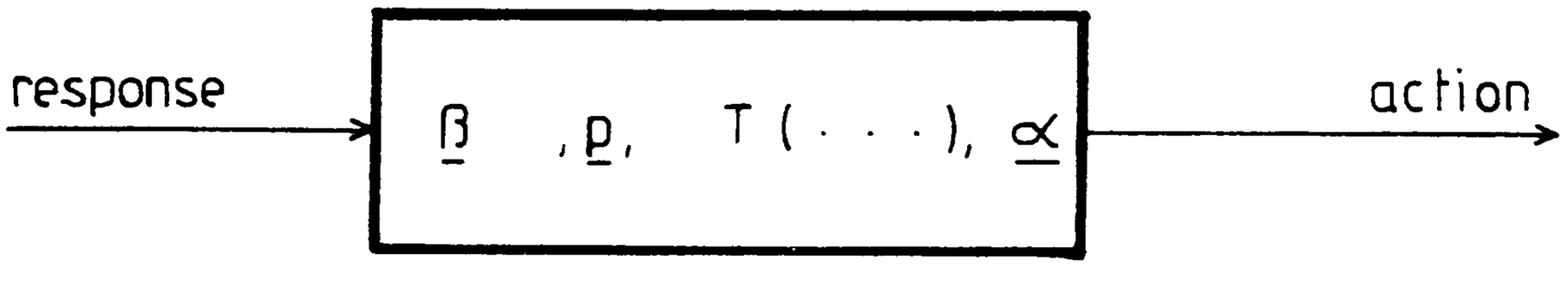


Figure1.2c S.A.V.S.

$\phi_1 n$ \ $\phi_1 n+1$					
ϕ_1	0	1	0	·	·
·	0	0	1	·	·
·	0	0	0	1	·
·	0	0	0	0	1
·					

F(0)

$\phi_1 n$ \ $\phi_1 n+1$					
ϕ_1	1	0	·	·	·
·	1	0	·	·	·
·	0	1	0	·	·
·					

F(1)

ϕ_1 \ α_1					
ϕ_1	1	0	·	·	·
·	1	0	·	·	·
·	0	1	·	·	·
·					

G

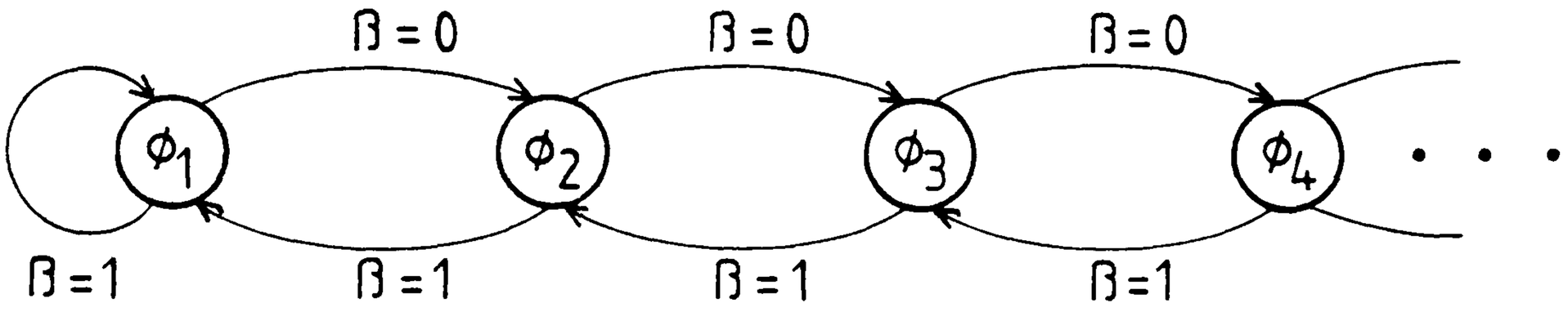


Figure 1.3a Fixed structure deterministic automaton

$\phi_1 n$ \ $\phi_1 n+1$					
ϕ_1	p_{11}	p_{12}	p_{13}	·	·
·	p_{21}	p_{22}	p_{23}	p_{24}	·
·					
·					
·					

F(0)

$\phi_1 n$ \ $\phi_1 n+1$					
ϕ_1	q_{11}	q_{12}	q	·	·
·	q_{21}	q_{22}	q	·	·
·	q_{31}	q_{32}	·	·	·
·					
·					

F(1)

ϕ_1 \ α_1					
ϕ_1	k_{11}	·	·	·	·
·	k_{21}	k_{22}	·	·	·
·	k_{31}	k_{32}	·	·	·
·					
·					

G

for clarity only transitions from state 3 are shown

penalty transition

reward transition

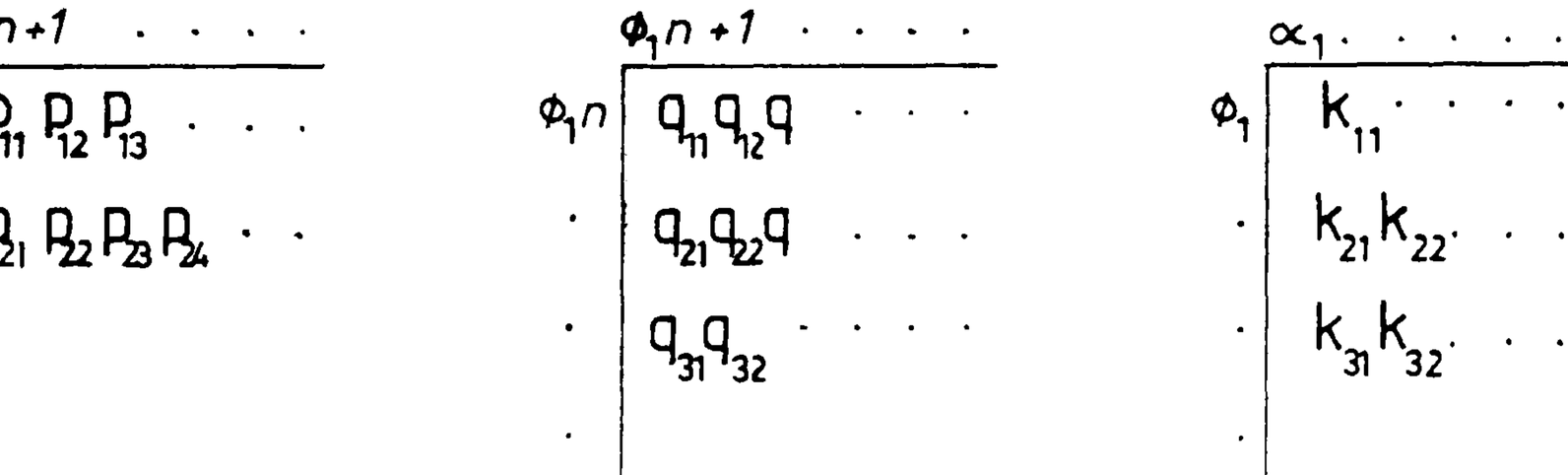


Figure 1.3b Fixed structure stochastic automaton

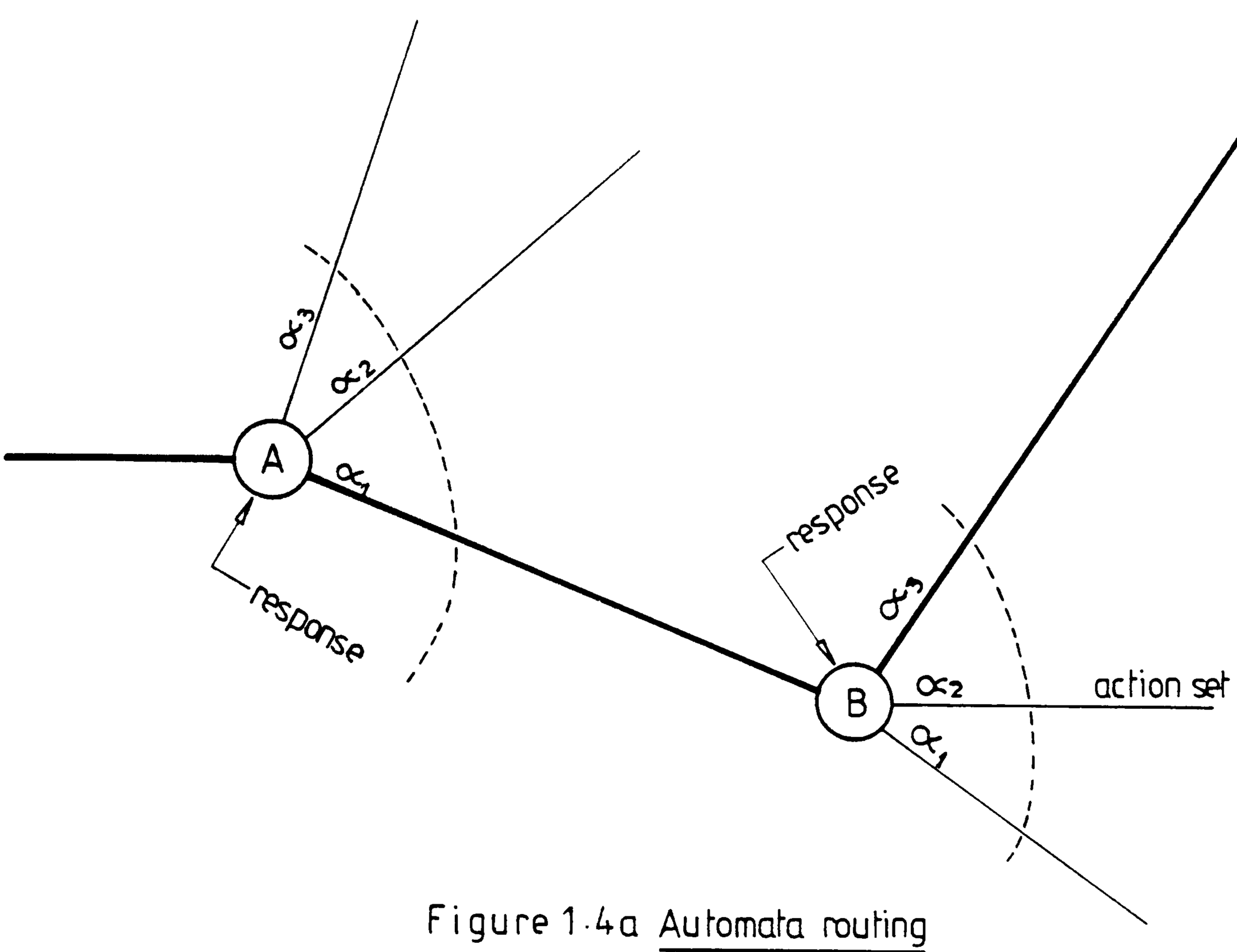


Figure 1.4a Automata routing

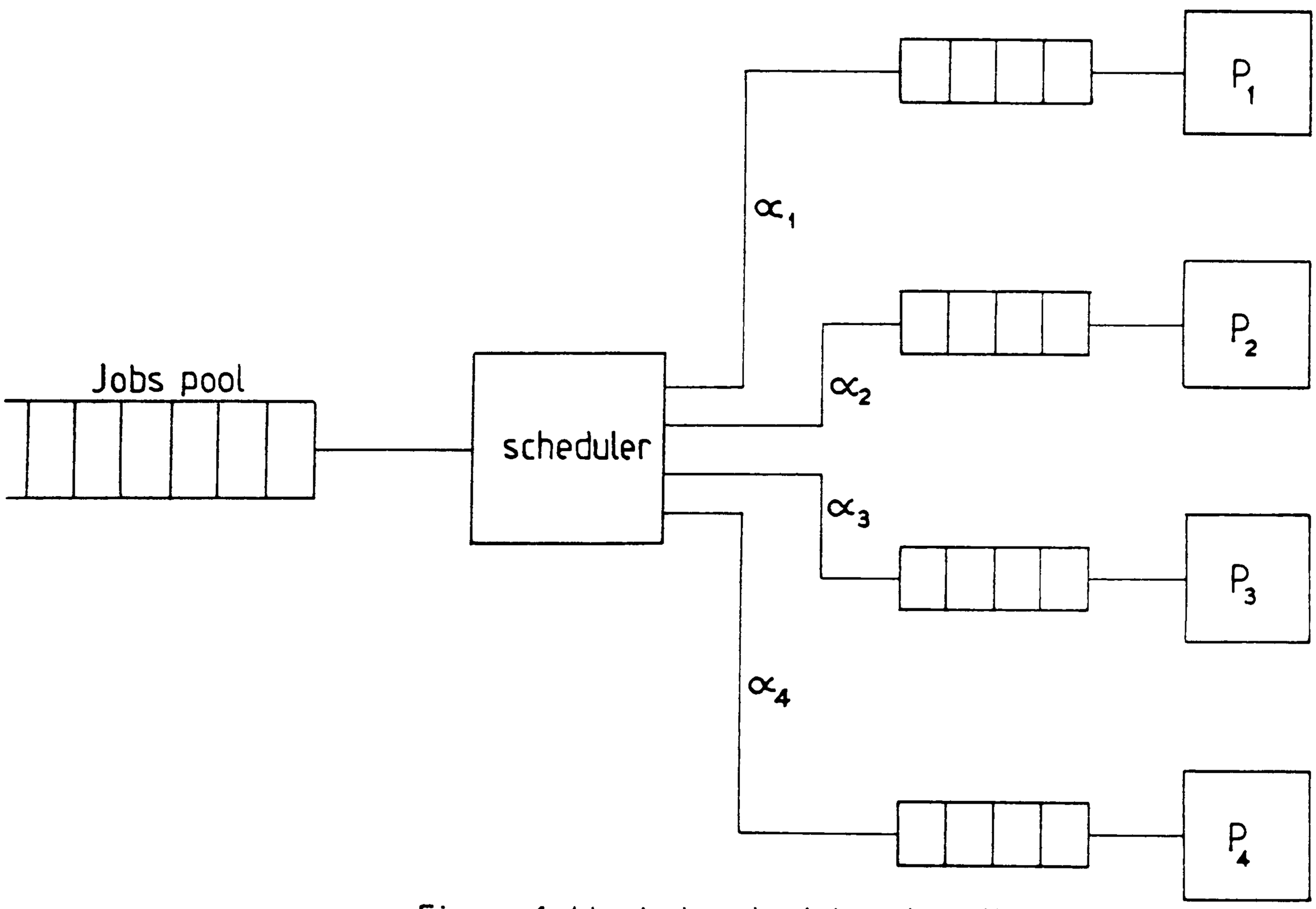


Figure 1.4b Automata job scheduling

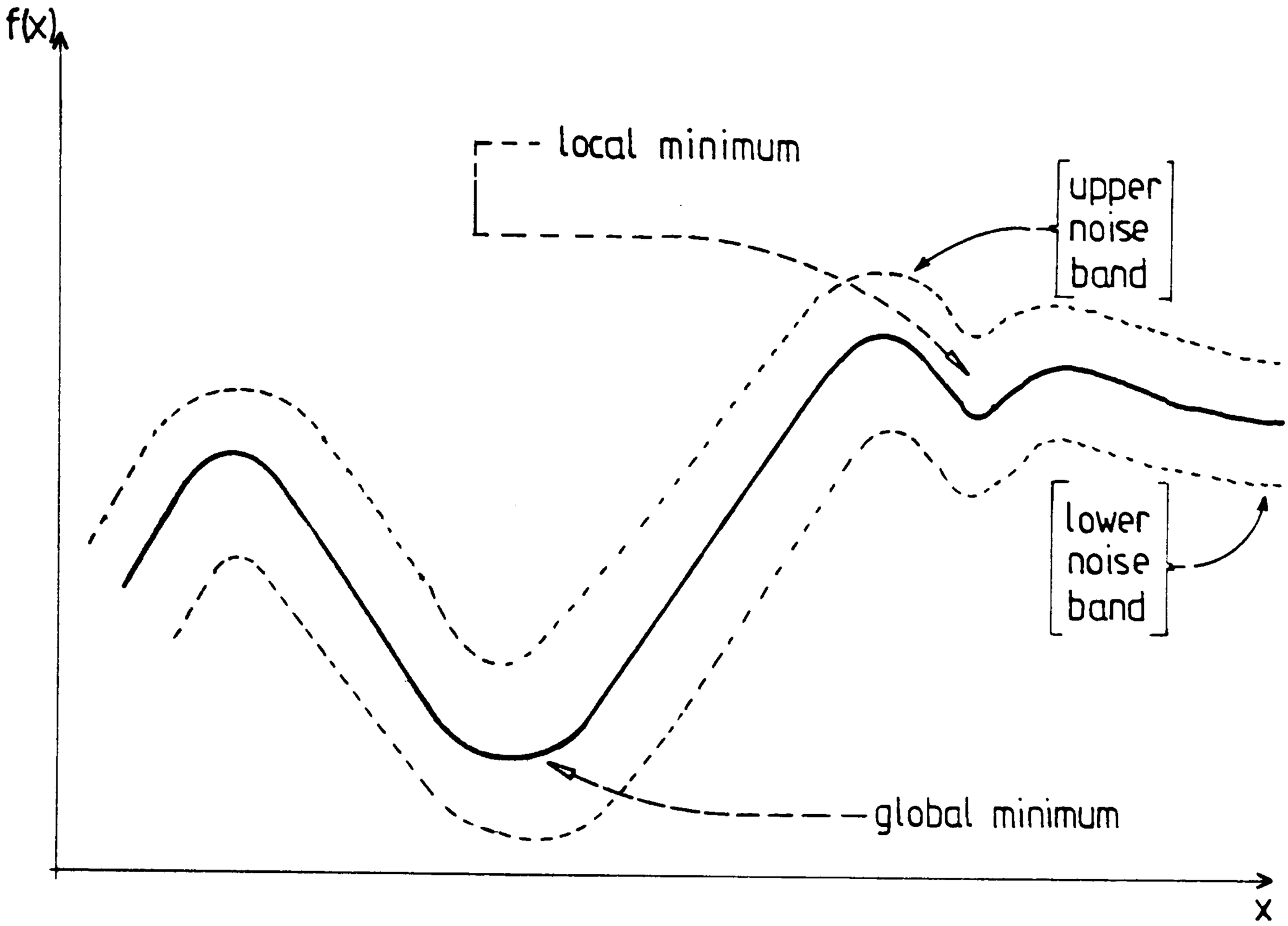


Figure 1.5a Multi-modal performance index

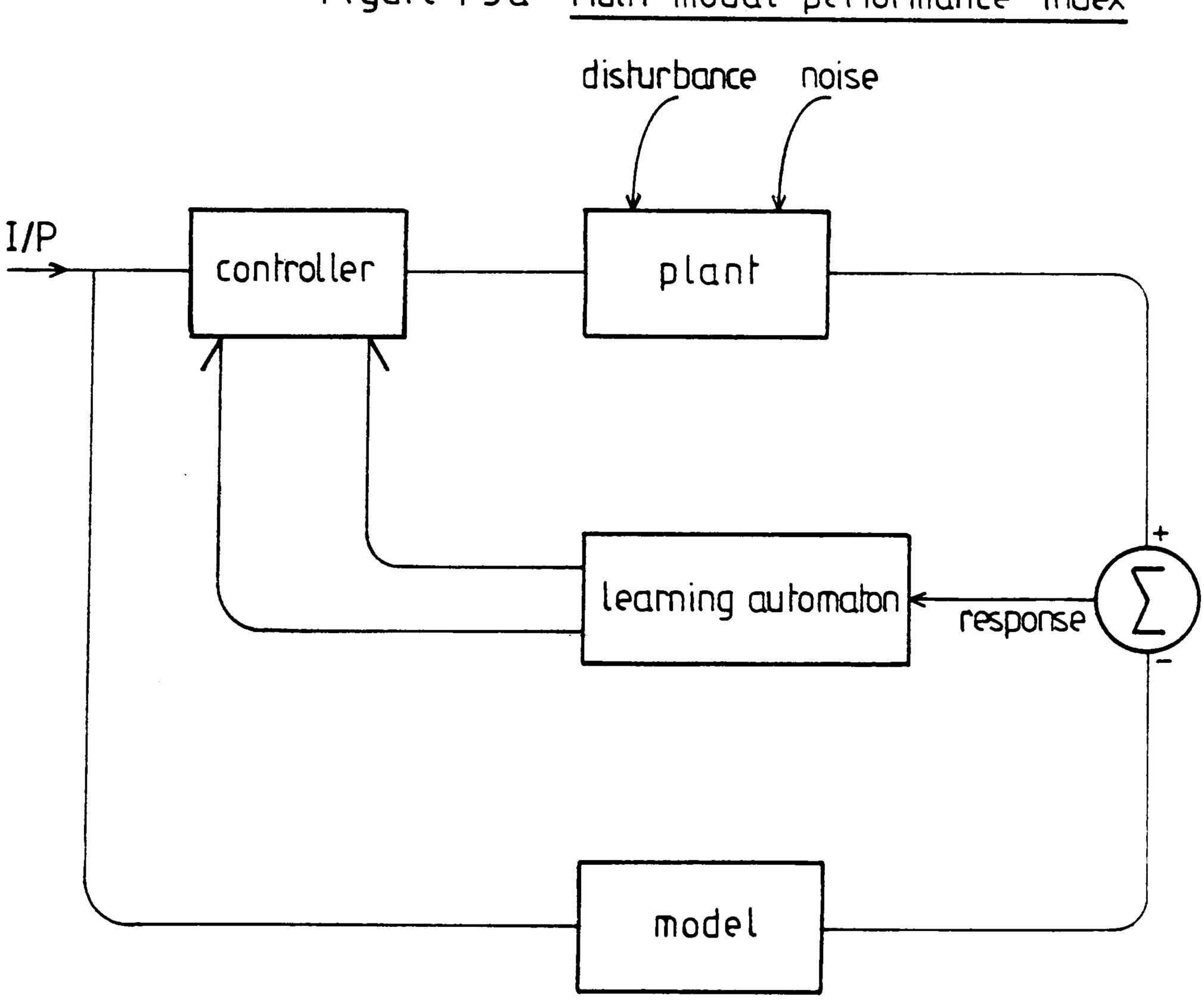


Figure 1.5b Model reference adaptive control using a learning automaton

CHAPTER 2

NON-AUTONOMOUS ENVIRONMENTS

2.1 Introduction

In Chapter 1, the concept of a random environment was introduced merely in sufficient depth to explain the form and operation of a learning automaton. As presented, the environment was defined to represent some random media on which the automaton operates, typically reacting to the performed actions with a random response. The environment was therefore characterised by the triple E(n) which includes $\underline{\alpha}$ the action set, $\underline{\beta}$ the response set and $\underline{c}$ the penalty set.

$$E(n) = \{\underline{\alpha}, \underline{\beta}, \underline{c}\}$$

In this chapter, a more detailed study of the environment is pursued, concerning the nature of the penalty set $\underline{c}$ and leading to the definition of environments which help in the understanding of practical automata applications.

The penalty set described the statistical properties of the random response issued by the environment for each member of the action set. As shown previously, the exact nature of the response is dependent on the type of environment and consequently the penalty set is similarly connected. For the P-model, a set of penalty probabilities was sufficient, whereas the Q and S-models required distribution functions to fully describe the response behaviour.

As of yet, the statistics of the penalty set have been thought of as stationary, the response characteristics of the media remaining constant for the duration of the operation. These environments are suitably termed stationary environments. However, in most learning automata applications, the environmental characteristics will vary with /

with time and hence the term non-stationary environment is adopted.

Non-Stationary Environments

Non-stationary environments are characterised by a penalty set $\underline{c}$ which varies with time.

Hence

$$\underline{c} = c(n)$$

One simple example of this is the switched non-stationary environment[15] where the penalty probabilities of a two action automaton are switched as shown in Figure 2.1a. Under these conditions the point of interest is the mean adjustment time, i.e., the average time required by the automaton to reconverge, given the new conditions.

A more elaborate non-stationary environment can be represented by a sequence of switches (Figure 2.1b) in which a range of penalty sets are interswitched according to some profile. This might typically be the case in a telephone network with the traffic load varying according to the time of day, e.g., peak and off-peak periods.

Non-Autonomous Environments

A further and very important class of non-stationary environment, is the non-autonomous environment where the penalty set is influenced by the actions of the automaton.

$$\underline{c} = c(\alpha)$$

The nature of a non-autonomous environment can best be described with reference to the following simple example. Consider a 2-action automaton selecting between two people to carry out an undisclosed job. The automaton allocates a person, who carries out the task, incurring a penalty response in relation to how well the job was performed. This /

This response will obviously depend on the skill of the individual, and since the two workers are different, the automaton should converge to a suitable allocation strategy. However, in a dynamic problem such as this, the ability of a person to carry out some function is also dependent on the rate at which the individual is required to work. As a person is selected to do the task more and more, fatigue sets in which decreases his ability to satisfactorily carry out the work, with a resulting increase in penalty responses. In contrast, the alternative worker who is chosen less, is better rested when required to do the job and consequently receives more favourable feedback.

In the routing problem, the network represents a non-autonomous environment when an automaton routing controller is used to select an outgoing link. By choosing a certain link with a high probability, that channel obviously becomes less attractive due to increased delay (or call blocking), caused by the higher traffic rate. On the otherhand, the alternative channel which is not used for an interval of time, is handling less traffic and therefore produces a better response.

For the remainder of this chapter, the nature of the non-autonomous environment and the interaction with the L_{R-I} and L_{R-P} automata are studied. This is further supplemented by the consideration of two simple non-autonomous environments which have been abstracted to gain an understanding of the process. Of the two proposed models, Model A is an elementary environment in which the penalty probabilities $c_1(n)$ and $c_2(n)$ are incremented and decremented according to which action is performed. In Model B, the penalty probabilities are assumed to be monotonically increasing functions of the automaton action probabilities, /

probabilities, which gives the desired behaviour of a high action probability resulting in a correspondingly high penalty probability and vice-versa. Finally, a series of simulation experiments verify the results shown for the 2-state P-model automaton for the various configurations of environmental model and learning algorithm.

2.2 Behaviour of a Variable Structure Automaton in a Non-Autonomous Environment

The behaviour of the variable structure automaton operating in a non-autonomous environment is considered with reference to the steady state behaviour of the L_{R-I} and L_{R-P} reinforcement algorithms. For convenience, a 2-state automaton interacting with a P-model environment is investigated, although the results are generally applicable to the r-state case and for the alternative response models.

2-Action L_{R-I} Algorithm

For 2 actions, the L_{R-I} reinforcement algorithm may be expressed as

$$\begin{aligned} p_1(n+1) &= p_1(n)+ap_2(n) && \alpha(n) = \alpha_1 \text{ Reward} \\ p_2(n+1) &= p_2(n)-ap_2(n) && \ldots 2.1 \\ p_1(n+1) &= p_1(n) && \alpha(n) = \alpha_1 \text{ Penalty} \\ p_2(n+1) &= p_2(n) && 0 < a < 1 \end{aligned}$$

For a non-autonomous environment[16], consider the expected equilibrium condition of the L_{R-I} automaton.

Consider the sequence

$$p_1(0), p_1(1) \ldots p_1(n) \triangleq \tilde{p}_1(n)$$

and defining

$$\Delta p_1(n) \triangleq E\left[p_1(n+1)-p_1(n) \middle| \tilde{p}_1(n)\right]$$

Using /

Using Equation 2.1 we write the expected $p(n+1)$

$$E\left[p_1(n+1)\right] = p_1(n)+ap_2(n)p_1(n)E\left[d_1(n)\right]-ap_1(n)p_2(n)E\left[d_2(n)\right]$$

$$\therefore \quad \Delta p_1(n) = ap_1(n)p_2(n)E\left[d_1(n)-d_2(n)\,\middle|\,\tilde{p}_1(n)\right]$$

$$d_i = 1-c_i$$

$$= ap_1(n)p_2(n)E\left[c_2(n)-c_1(n)\,\middle|\,\tilde{p}_1(n)\right] \quad \ldots 2.2$$

If the process $p_1(n)$ converges in such a way that $\lim\ n\to\infty\ \ \Delta p_1(n)\to 0$, then Equation 2.2 yields some very interesting convergence properties. Considering the zeros of the equation we find

(i) $p_1(n)\to 0 \qquad p_2(n)\to 1$

(ii) $p_2(n)\to 0 \qquad p_1(n)\to 1$

(iii) $E\left[c_2(n)-c_1(n)\,\middle|\,\tilde{p}_1(n)\right]\to 0$

The first two conditions arise due to the conventional absorbing states of the L_{R-I} algorithm which as stated previously is ϵ optimal in a stationary environment. The third possibility exhibits a completely different behaviour since convergence is related to the evolution of the penalty probabilities. In the limit, condition 3 is such that an equalisation of the expected penalty probabilities is obtained.

$$E\left[c_1(n)\,\middle|\,\tilde{p}_1\right] = E\left[c_2(n)\,\middle|\,\tilde{p}_1\right] \quad \ldots 2.3$$

2-Action L_{R-P} Algorithm

Turning to the L_{R-P} algorithm a similar result is given[17]. With the 2-state L_{R-P} reinforcement scheme.

$$p_1(n+1) = p_1(n)+ap_2(n) \qquad \alpha(n) = \alpha_1 \text{ Reward}$$

$$p_2(n+1) = p_2(n)-ap_2(n) \quad \ldots 2.4$$

$$p_1(n+1) = p_1(n)-bp_1(n) \qquad \alpha(n) = \alpha_1 \text{ Penalty}$$

$$p_2(n+1) = p_2(n)+bp_1(n) \qquad 0<a,b<1$$

Therefore /

Therefore

$$E\left[p_1(n+1)\right] = p_1(n)+ap_2p_1E\left[d_1\right] -ap_1p_2E\left[d_2\right] +bp_2^2E\left[c_2\right] -bp_1^2E\left[c_1\right]$$

for $a = b$

$$= p_1(n)+aE\left[p_2(n)c_2(n)-p_1(n)c_1(n)\,\middle|\,\tilde{p}_1(n)\right]$$

$$\therefore \quad \Delta p_1(n) = aE\left[p_2(n)c_2(n)-p_1(n)c_1(n)\,\middle|\,\tilde{p}_1(n)\right] \qquad \ldots 2.5$$

Again considering the limit as $n\to\infty$, $\Delta p_1(n)\to 0$ we find that

$$E\left[p_1(n)\ c_1(n)\,\middle|\,\tilde{p}_1(n)\right] = E\left[p_2(n)c_2(n)\,\middle|\,\tilde{p}_1(n)\right] \qquad \ldots 2.6$$

Whereas the L_{R-I} automaton converged such as to equalise the penalty probabilities, the L_{R-P} algorithm is seen to balance the penalty rates, $p_i c_i$.

r-Action L_{R-I} Algorithm

The result for the 2-state L_{R-I} automaton may be generalised for the r-action case. From the r-action L_{R-P} algorithm Equation 1.1, we can write for r-actions

$$\Delta p_i(n) = a\sum_{j\neq i} p_j p_i d_i - ap_i \sum_{j\neq i} p_j d_j$$

$$= ap_i\Big[\sum_{j\neq i} p_j c_j - c_i \sum_{j\neq i} p_j\Big] \qquad \ldots 2.7$$

As $n\to\infty$, $\Delta p_i(n)\to 0$

$$p_j c_j = c_i \sum_{j\neq i} p_j \qquad \ldots 2.8$$

This indicates that for the r-action case each penalty probability is equalised.

$$E\left[c_a\right] = E\left[c_b\right] \qquad a, b = 1, 2, \ldots r$$

r-Action L_{R-P} Algorithm

The operation of a multi-action L_{R-P} algorithm in a non-autonomous environment is shown in [17], giving

$$\Delta p_i(n) = a\Big[\frac{1}{r-1}\sum_{j\neq i} p_j c_j - p_i c_i\Big] \qquad \ldots 2.9$$

Again this confirms the behaviour of the L_{R-P} algorithm for r-actions giving

$$E\left[p_a c_a\right] = \quad /$$

$$E\left[p_a c_a\right] = E\left[p_b c_b\right] \qquad a, b = 1, 2, \ldots r$$

SL_{R-I} Algorithm

For the 2-action SL_{R-I} algorithm, a similar condition to the P-model case is noted

$$\Delta p_1(n) = a p_1(n)\left[1-p_1(n)\right] E\left[S_2(n)-S_1(n)\,\middle|\,\tilde{p}_1(n)\right] \quad \ldots 2.10$$

taking lim $n\to\infty$, $\Delta p_1(n)\longrightarrow 0$ we obtain

$$E\left[S_1(n)\,\middle|\,\tilde{p}_1(n)\right] = E\left[S_2(n)\,\middle|\,\tilde{p}_1(n)\right] \quad \ldots 2.11$$

Physically this corresponds to the equalisation of the expected penalty weights.

In[17] a mathematically rigorous treatment of the L_{R-P} automaton operating in a non-autonomous environment is presented. The significant contribution of this work is to show that the process $p_i(n)$ converges to a normal distribution with mean p^*. Furthermore, it shows how the variance is influenced by the learning coefficients a and b, high values giving large variance. The matter of learning coefficients however, is an extremely complex one, the values representing a trade off between steady state accuracy (variance) and convergence rate.

2.3 Environment Model-A

This model represents an initial attempt[16] to abstract a non-autonomous environment suitable for modelling the behaviour of a telephone network under dynamic routing schemes.

For 2-states the environment is as follows:-

Given $\alpha(n) = \alpha_1$

$$\begin{aligned} c_1(n+1) &= c_1(n)+\theta_1 \\ c_2(n+1) &= c_2(n)-\varphi_2 \qquad 0<\varphi_i, \theta_i<1 \end{aligned} \quad \ldots 2.12$$

Given $\alpha(n) = \alpha_2$ /

Given $\alpha(n) = \alpha_2$

$$c_1(n+1) = c_1(n) - \varphi_1 \qquad 0 \leqslant c_i(n+1) \leqslant 1$$
$$c_2(n+1) = c_2(n) + \theta_2$$

The above expressions show the selection of an action to cause an increase in the related penalty probability and a subsequent decrease in the alternative. In general, θ_i and φ_i may be functions of p_i and c_i, but for simplicity, they are assumed to be constants.

It is shown[16] that for the described environment, the following condition exists.

$$E\left[c_2(n)-c_1(n)\,\middle|\,p_1(n)\right] = c_2(0)-c_1(0)+(\varphi_1+\theta_2)n-(\theta_1+\theta_2+\varphi_1+\varphi_2)\sum_{n=1}^{N-1} p_1(n)$$

Again considering $E\left[c_2-c_1\right] \rightarrow 0$ as lim $n \rightarrow \infty$

then

$$\frac{1}{N}\sum_{n=1}^{N-1} p_1(n) \rightarrow \frac{\theta_2+\varphi_1}{\theta_1+\theta_2+\varphi_1+\varphi_2}$$

$$E\left[p_1(n)\right] = \frac{\theta_2+\varphi_1}{\theta_1+\theta_2+\varphi_1+\varphi_2} \qquad \ldots 2.13$$

The sample average of $p_1(n)$ is therefore seen to converge to a value dictated purely by the characteristics of the abstract environment. Additionally, expressions for the conditional expectancy of the penalty probabilities are given by

$$E\left[c_1(n)\,\middle|\,\tilde{p}_1(n)\right] = c_1(0)+\frac{\theta_1\theta_2-\varphi_1\varphi_2}{\theta_1+\theta_2+\varphi_1+\varphi_2}\cdot n$$

$$E\left[c_2(n)\,\middle|\,\tilde{p}_1(n)\right] = c_2(0)+\frac{\theta_1\theta_2-\varphi_1\varphi_2}{\theta_1+\theta_2+\varphi_1+\varphi_2}\cdot n \qquad \ldots 2.14$$

The above equations show the penalty probabilities to assymptotically converge towards a probability of 1 or 0, dependent on the relative values of $\theta_1\theta_2$ and $\varphi_1\varphi_2$. For example, with $\theta_1\theta_2 < \varphi_1\varphi_2$ both probabilities decrease towards /

towards 0 and in the other extreme, $\theta_1\theta_2 > \varphi_1\varphi_2$ convergence is towards 1.

2.4 Environment Model B

In an alternative approach to abstract a non-autonomous environment, a model is proposed in which the penalty probabilities are assumed to be functions of the action probability set.

$$c_i(n) = F\{\underline{p}(n)\}$$

Again considering the 2-state case, the penalty probabilities are chosen to be only a function of the corresponding action probability.

$$\begin{aligned} c_1(n) &= k_1 p_1^m(n) \\ c_2(n) &= k_2 p_2^m(n) \end{aligned} \qquad \ldots 2.15$$

As shown previously, convergence of the L_{R-I} automaton is such that

$$E\left[c_1(n)\middle|\tilde{p}_1(n)\right] = E\left[c_2(n)\middle|\tilde{p}_1(n)\right]$$

giving

$$k_1 p_1^m = k_2 p_2^m \qquad p_2 = 1-p_1$$

$$p_1^* = \frac{\sqrt[m]{k_2}}{\sqrt[m]{k_1}+\sqrt[m]{k_2}} \qquad \ldots 2.16$$

Similarly for the L_{R-P} algorithm we expect convergence such that

$$E\left[p_1 c_1\middle|\tilde{p}_1\right] = E\left[p_2 c_2\middle|\tilde{p}_1\right]$$

giving

$$k_1 p_1^{m+1} = k_2 p_2^{m+1}$$

$$p_1^* = \frac{\sqrt[m+1]{k_2}}{\sqrt[m+1]{k_1}+\sqrt[m+1]{k_2}} \qquad \ldots 2.17$$

Equations /

Equations 2.16 and 2.17 give the expected steady state action probabilities for the L_{R-I} and L_{R-P} automata operating on the specified environment. Further, this behaviour is illustrated on Figures 2.2a and 2.2b which show convergence diagrams for both schemes in an environment with $m = 1$.

Clearly, the behaviour of both schemes can no longer be classified according to the conventional norms of behaviour listed in the previous chapter. In the non-autonomous environment the optimal automaton is one which converges to give the minimum average penalty. Returning to the linear environment we write the average penalty M(n).

$$M(n) = \sum_{i=1}^{2} p_i c_i$$
$$= k_1 p_1^{m+1} + k_2 p_2^{m+1} \qquad \ldots 2.18$$

The true minimum average penalty is obtained by

$$\frac{dM(n)}{dp_1} = 0$$

this yielding

$$p_1^{opt} = \frac{\sqrt[m]{k_2}}{\sqrt[m]{k_1} + \sqrt[m]{k_2}} \qquad \ldots 2.19$$

This equation gives the action probability for action 1 which results in the minimum average penalty and thus corresponds to the optimum output strategy of an automaton operating in the specified environment. Comparison with Equation 2.16 indicates the L_{R-I} automaton to be truly optimal, the L_{R-P} tending to be minimally sub-optimal.

2.5 Experiment Results

Using a computer model[18] of a 2-state automaton, the various combinations of reinforcement algorithms and non-autonomous /

non-autonomous environments were explored. These included L_{R-I}/Model A, L_{R-I}/Model B and L_{R-P}/Model B, the results providing plots of the action and penalty probabilities against number of iterations.

Experiment 1(a) L_{R-I}/Model A

In this experiment, the L_{R-I}/Model A configuration was investigated, the run producing results as shown on Figure 2.3.

a	θ_1	θ_2	φ_1	φ_2	Sample Average	$\theta_1\theta_2-\varphi_1\varphi_2$
0.02	0.04	0.02	0.01	0.05	0.25	3×10^{-4}

As predicted by the theory, the action probability sample average tends to a value dictated by the environmental parameters θ_i and φ_i. In this case, a sample average of 0.246 is obtained, this giving good correlation with the theoretical value of 0.25. From Equation 2.14 it was shown that both c_is should tend to 0 or 1, depending on the relative values of θ_i and φ_i. Again this is supported by the results, with both penalty probabilities tending to 1 in 700 iterations.

Experiment 1(b) L_{R-I}/Model A

As in the previous run, the L_{R-I}/Model A configuration is considered, this experiment using an alternative set of environmental parameters.

a	θ_1	θ_2	φ_1	φ_2	Sample Average	$\theta_1\theta_2-\varphi_1\varphi_2$
0.02	0.01	0.05	0.04	0.02	0.75	-3×10^{-4}

The results from this run are shown in Figure 2.4 and present the alternative action to Experiment 1(a). This particular /

particular experiment shows extreme variance on the penalty probability graph which can be explained as follows. As the penalty probabilities converge towards zero, the learning automaton receives a considerable number of rewards and because the up-dating scheme is Reward Inaction, there is continual modification of the action probabilities. In the previous example, since convergence is to 1, the high number of penalty signals are ignored by the L_{R-I} reinforcement algorithm.

Experiment 2 L_{R-I}/Model B

Experiments 2 and 3 are used to show the characteristics of environmental Model B with m = 1. In this example, the L_{R-I} automaton is used, the run producing results as shown in Figure 2.5.

a	k_1	k_2	p_1^*
0.02	0.2	0.875	0.814

In accordance with the theory, the penalty probabilities are seen to converge to equal values (given variance) with the action probability p_1 converging to a value dictated by the environment coefficients k_1 and k_2.

Experiment 3 L_{R-P}/Model B

This experiment presents the L_{R-P}/Model B combination, the results being displayed on Figure 2.6.

a	b	k_1	k_2	p_1^*
0.02	0.02	0.02	0.875	0.677

In /

In the previous experiment, the equalisation of the penalty probabilities was demonstrated by the L_{R-I}/Model B pair. This experiment confirms that in the same environment, the L_{R-P} algorithm operates to equalise the penalty rates.

$$p_1 c_1 = p_2 c_2$$

2.6 Conclusions

In an attempt to describe a practical automaton/environment combination, the concept of an non-autonomous environment has been introduced. The fundamental property of such an environment being the relationship between the penalty and action sets, the action selection process of the automaton bearing direct influence on the environmental response, as is the case in many practical applications.

This chapter has investigated the behaviour of the L_{R-I} and L_{R-P} reinforcement algorithms, showing the convergence conditions in a general non-autonomous environment. Further studies have resulted in the abstraction of two simple non-autonomous environments to demonstrate the operation of the learning schemes and also as an initial attempt at modelling the routing operation of the automaton. A first elementary non-autonomous environment was implemented in the form of Model A. Although this particular model proved unrealistic convergence of the penalty probabilities being to either 1 or 0, the expected convergence of the L_{R-I} automaton was clearly demonstrated. Model B on the otherhand, represents a more realistic approach to a non-autonomous environment, the penalty probabilities taking on values in the range (0, 1), dictated entirely by the environmental parameters.

From /

From this chapter, several conclusions can be drawn regarding the non-autonomous environment and the convergence behaviour of the L_{R-I} and L_{R-P} reinforcement algorithms.

1 In practical applications such as network routing, the non-autonomous environment provides a suitable framework to model the behaviour of the adaption process.

2 The operation of the L_{R-I} automaton in a non-autonomous environment is such that the penalty probabilities c_i are equalised. The L_{R-P} automaton also converges to a balance, which in this case is the equalisation of the penalty rates $p_i c_i$.

3 An optimal automaton operating in a non-autonomous environment is one which operates to minimise the average expected penalty. In general, this need not result in the automaton converging to a specific action.

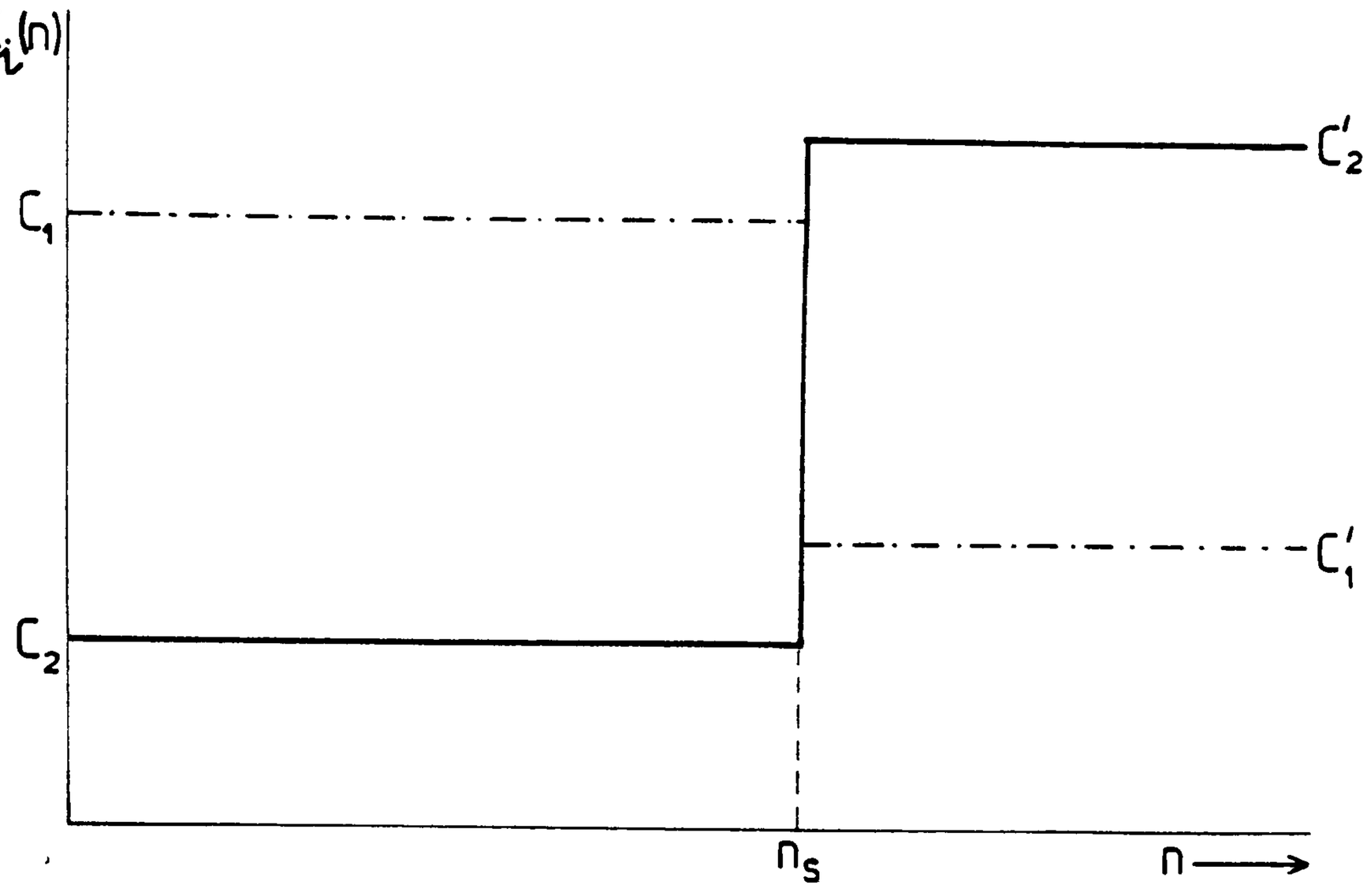


Figure 2.1a Switched non-stationary environment

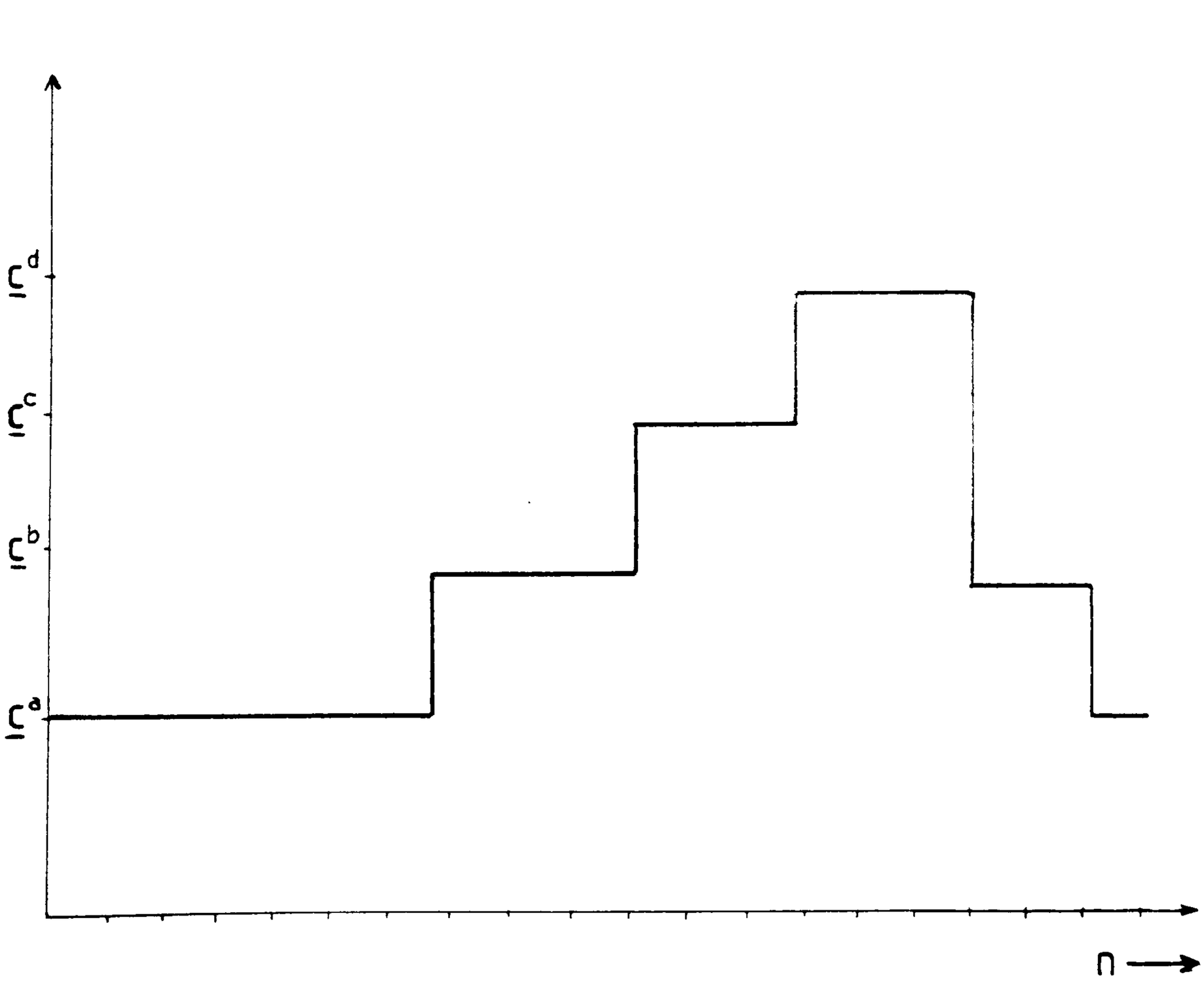


Figure 2.1b Stepped non-stationary environment

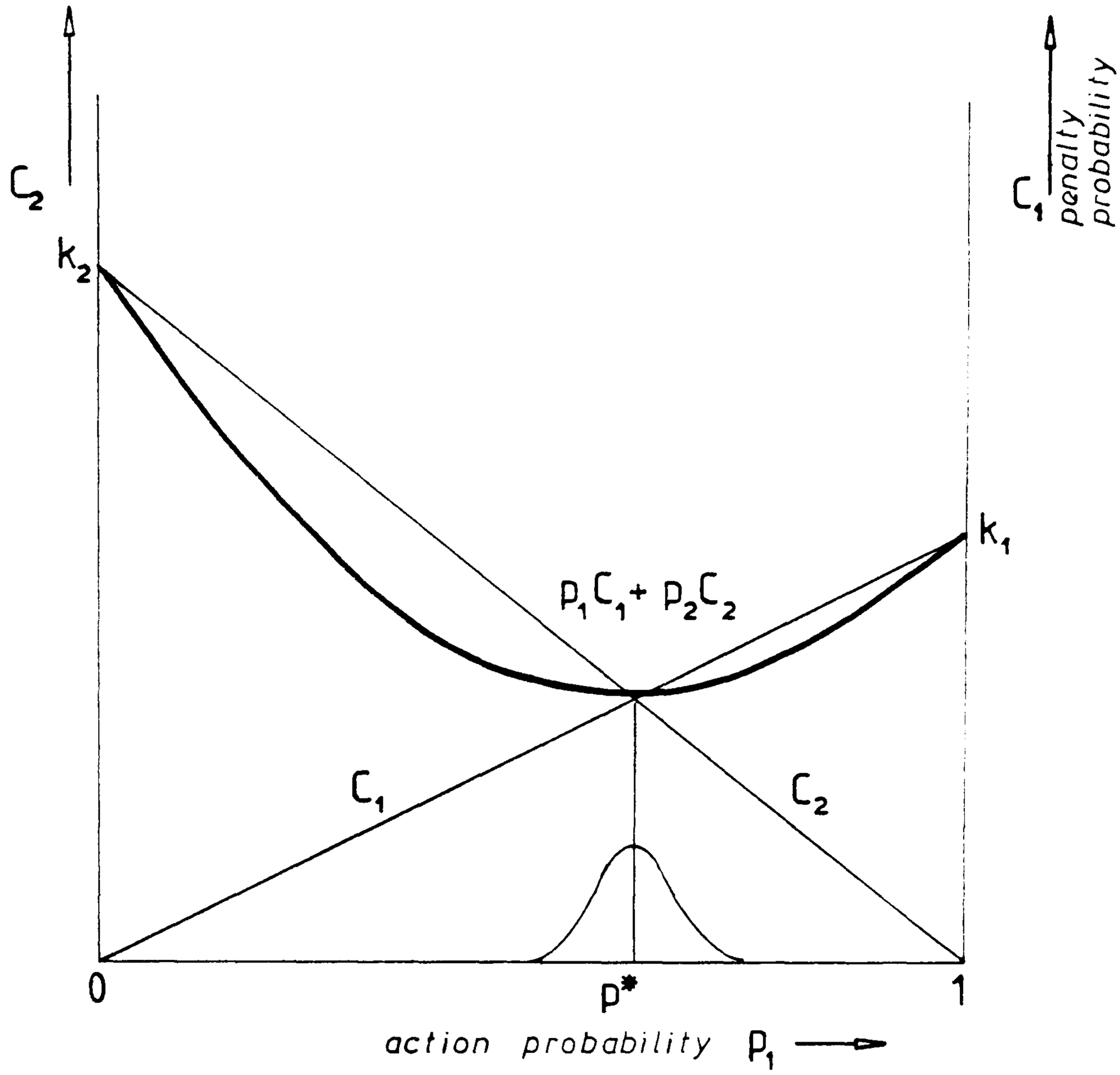


Figure 2.2a LRI in a linear non-autonomous environment

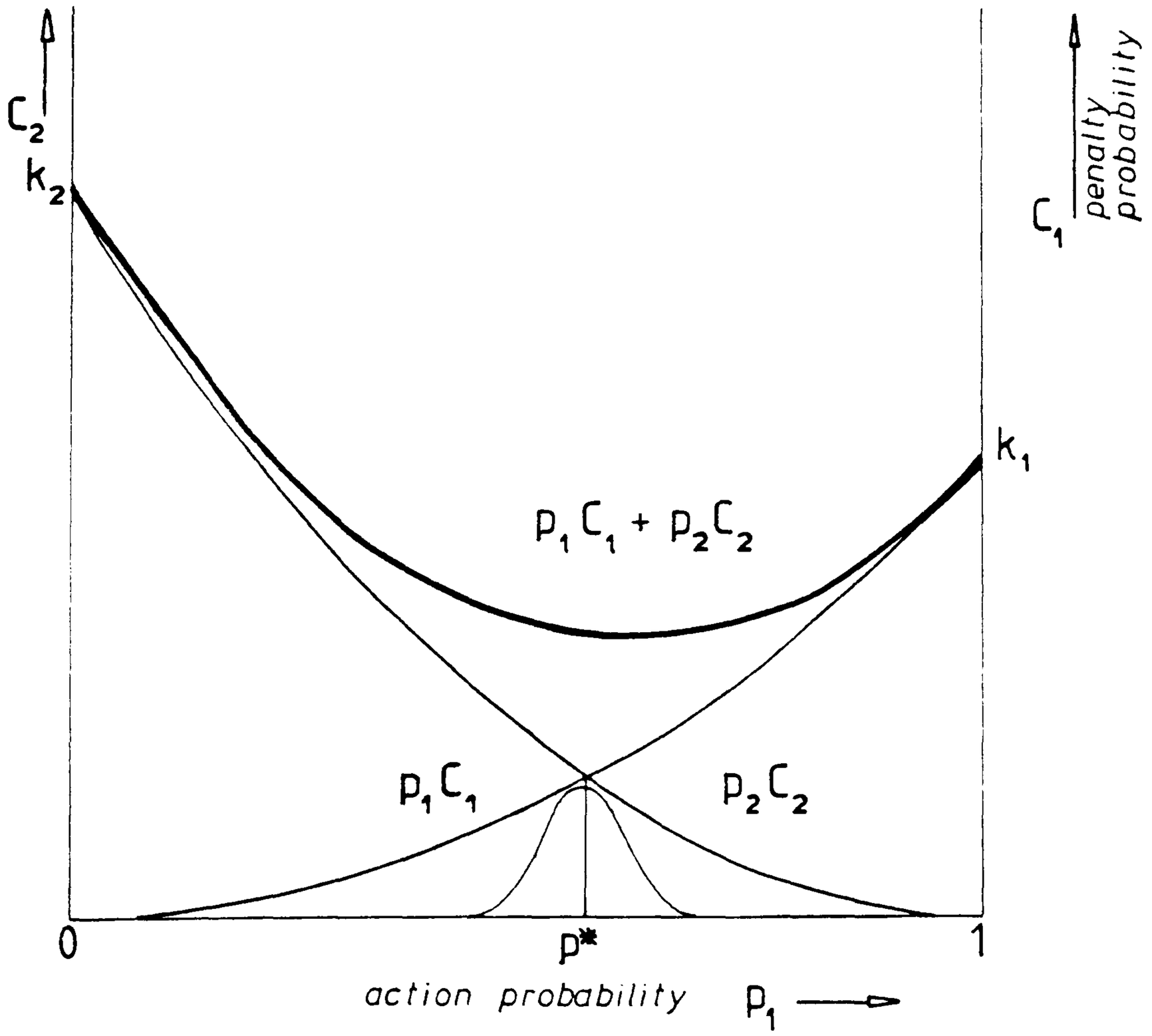


Figure 2.2b LRP in a linear non-autonomous environment

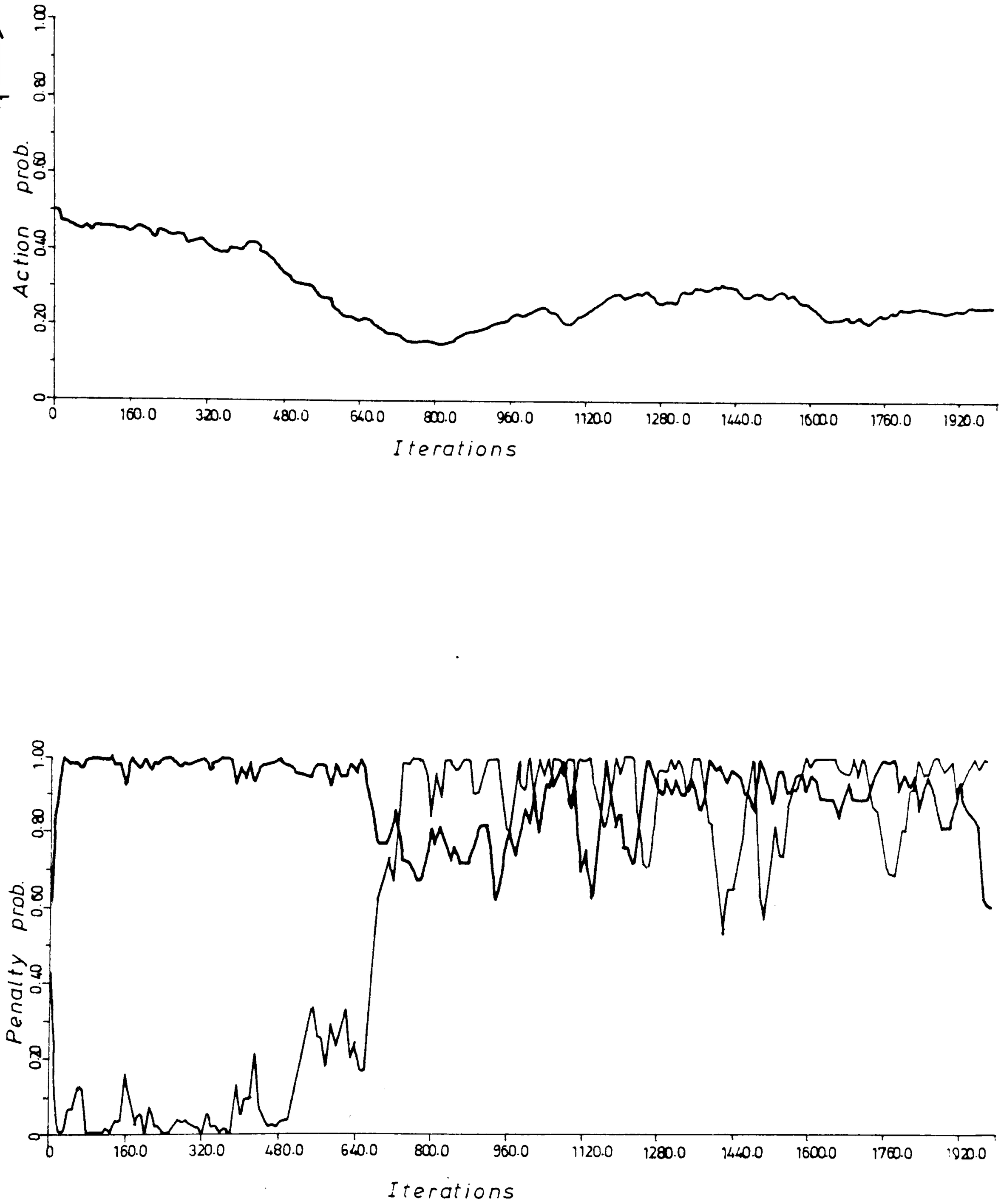


Figure 2.3 LRI / Model A

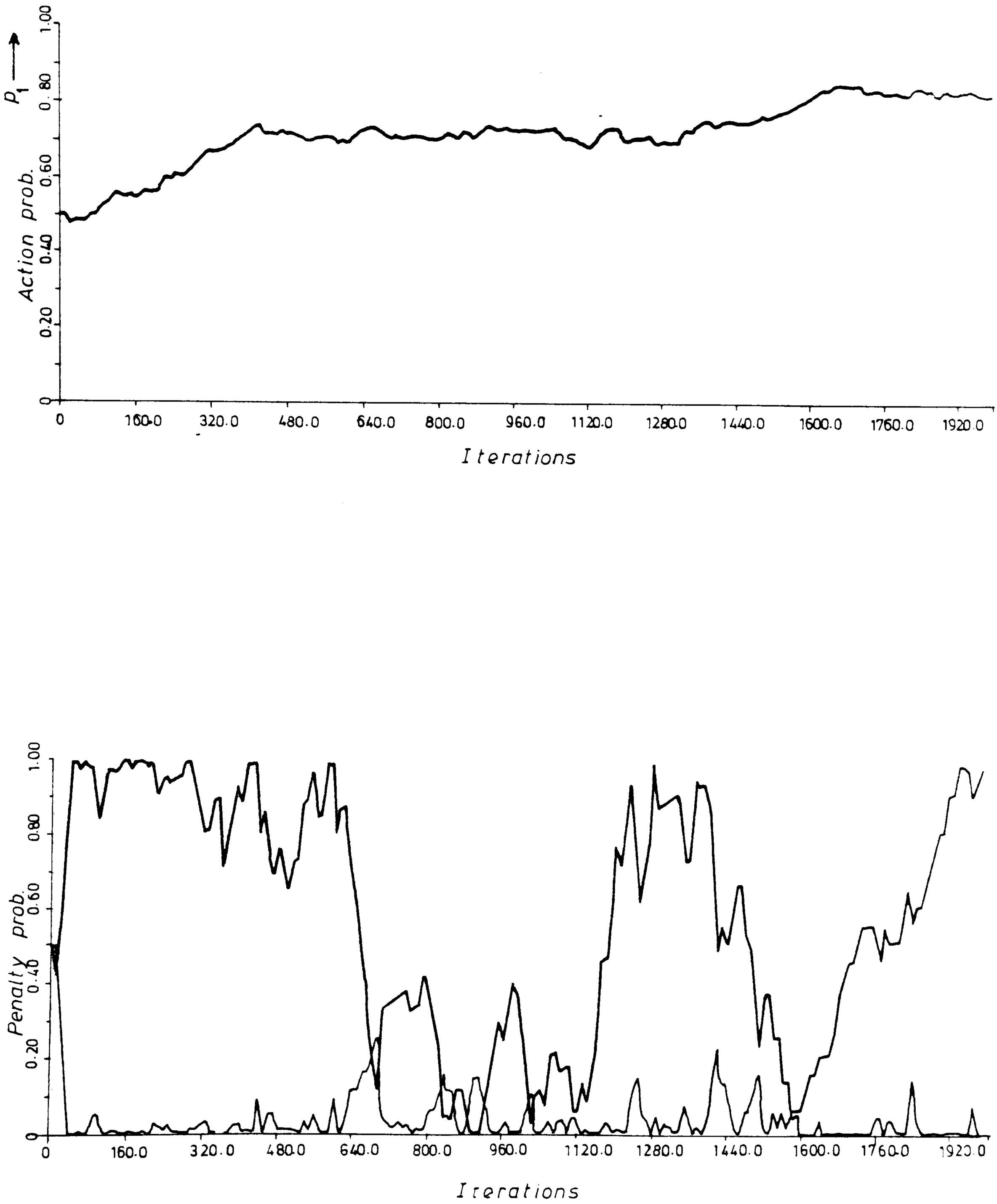


Figure 2.4 LRI/Model A

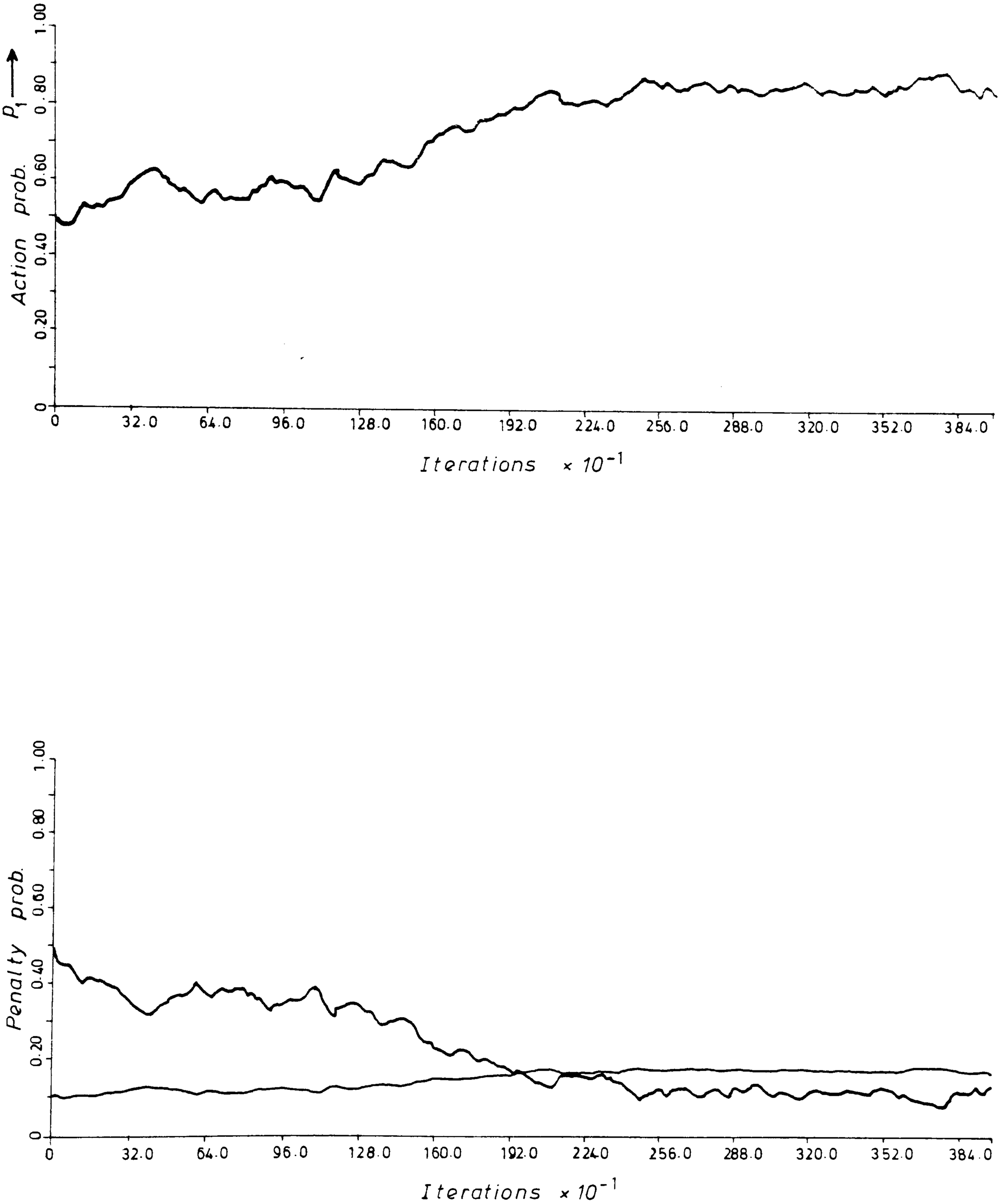


Figure 2.5 LRI/Model B

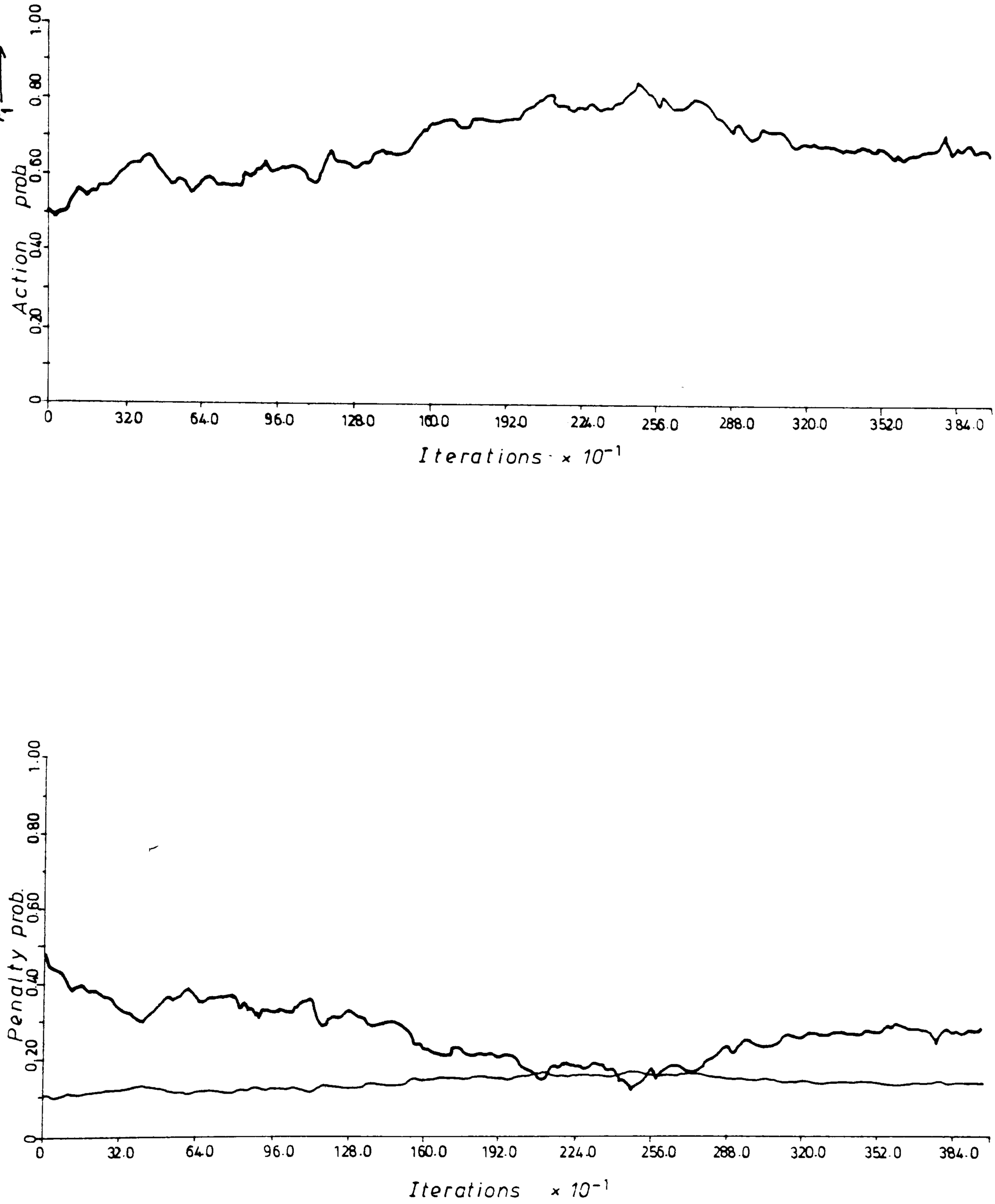


Figure 2.6 LRP/Model B

CHAPTER 3

CIRCUIT SWITCHED NETWORKS

3.1 Introduction

The purpose of this chapter is twofold; to provide a general introduction and review of circuit switched networks and also to consider the requirements and form of a simulation package, suitable for carrying out experimental studies of circuit switched networks and routing schemes. In the first sections the network structure and components are described, progressing to an overview of the operating principles and a discussion of existing routing techniques. Later sections are concerned with computer modelling of circuit switched networks and describe two custom prepared packages, one designed for a small minicomputer and a further, more sophisticated version to run on a mainframe. In discussing both simulators, emphasis is placed on the program and data structures which utilise some novel approaches to the problem of network simulation and in addition offer a complete range of services for detailed experiments.

A telecommunications network is a means for providing distributed users with complete connectibility, thus permitting the transmission and reception of information between any end pair. In a telephone network, this information is conveyed by a voltage analogue along an electrical path between two subscribers with transducers at both ends carrying out necessary conversions between sound and electrical energy. The telephone network may be described as a circuit switched network since individual lines in the network are switched to form a path suitable for communication, the network providing full connectivity without direct links between all stations.

In /

In general, a circuit switched network provides service through a collection of exchanges or switching centres connected by trunk groups, each group containing a number of lines which are capable of providing a communication path. Operating with these resources, the network forms connections by electrically switching lines to set up a complete circuit between the source user (the caller) and the destination user. Normally this operation involves a special signalling message which traverses the network, holding open lines through which it has passed and on formation of a complete circuit allocates it for communication. The call then has the sole use of this path for the duration, the lines only becoming available for further calls when the caller releases the circuit.

Routing in a circuit switched network is the decision mechanism which determines how the signalling message proceeds through the many possible path combinations in the network. For maximum network performance an optimal routing scheme is required, i.e., one which operates to minimise the chance of a call attempt being rejected when the routing signal becomes stranded at an intermediate node. To quantify this problem, the call blocking probability is defined as the ratio of calls rejected to the total number of calls attempted.

Already in this chapter the telephone network has been referred to as an example of a major circuited switched network. In addition, several other networks operate on a circuit switching principle including the Telex network, the DATRAN[19] computer network and military communications networks such as Ptarmigan[20]. This final type of network is a prime contender for adaptive routing, since the tactical communications environment offers /

offers conditions which are highly non-stationary in terms of network topology and traffic.

3.2 Network Components and Operation

The fundamental function of a communications network is the provision of a switched path facility for the transmission of information. In very general terms the constituent components of a network should include:-

Switching Centres or Nodes

A network node may be considered as an intersection of several trunk groups with an internal mechanism for performing a switching function to connect call circuits. For a general network with N nodes, the set $\mathcal{N}$ is the collection of nodes, each uniquely identified by a numerical address.

$$\mathcal{N} = \{1, 2, \ldots\}$$

Trunk Group

The trunk group T_{ij}, connecting nodes i and j, consists of l lines. A set $\mathcal{T}$ defines the groups in the network. For this network, call attempts from both nodes have equal claim on the resource, the available lines forming a common pool for i and j.

$$\mathcal{T} = \{T_{12}, T_{13} \ldots T_{nn-1}\} \quad |\mathcal{T}| \leq N \times N$$

Terminal Equipment

The terminal equipment in a network serves a dual role, acting as a transducer between the user and network and also operating to generate and receive network control signals. In the simple network shown on Figure 3.1, the terminal equipment attached to a node forms a local network effectively feeding traffic to the higher level trunk network. The routing problem is in the domain of the trunk /

trunk network operations and consequently the local network and terminal equipment can be regarded as a collection of traffic sources and sinks.

Call

A call of class i, j is a session of communication between the calling node i and some node j. All calls handled by the system constitutes the network traffic.

Route

A route R_{ij}, from node i to j, is an ordered sequence of lines $\{T_{ia}, T_{ab}, \cdots T_{fj}\}$ connected to provide a path for a call . The network is defined as connected if R_{ij} exists for $ij \epsilon \mathcal{N}$. Furthermore, the network is considered strategic if $|R_{ij}| > 1$ for $ij \epsilon \mathcal{N}$. For this study, strategic networks are of primary concern since a good adaptive routing scheme should have the ability to utilise all possible routes in the network this leading to greater operating reliability.

As considered initially, the underlying concepts in a circuit switched network involves the establishment and allocation of a direct communication path between two end stations. This is illustrated in Figure 3.2 which shows a circuit switched connection between S and D. This same diagram also serves to show the need for a routing mechanism, the network obviously containing several possible paths for all source/destination pairs.

The typical operating protocol for establishing a connection is shown in Figure 3.3. Initially, the local network has made a connection between the caller and the trunk node S, which commands the following set-up procedure. A routing signal is sent on a line to a suitable next node I (selected by the routing scheme at S) and a line /

line in T_{si} is allocated. A similar operation is repeated at node I with a line in T_{sd} providing the final connection in the trunk network. Communication between both users is possible when the required path in the local network is made and the entire path allocated for transmission. Finally, on completion of the call, a release signal backtracks along the connection, dropping lines for use by other requests.

3.3 Network Traffic

The traffic demands from network terminals are stochastic processes, influenced by the behaviour and circumstances of the calling population. Normally, these are characterised by the calling or arrival rate and the call duration or holding time which signifies the generation rate of call requests at the source node and the length of the communication session respectively.

Arrival Statistics

For this study, the behaviour of calls ariving at a source node is assumed to be a Poisson process with an average point to point arrival rate given by λ_{ij} calls/unit time. The probability of K arrivals at node i, destination j in time T is given by

$$\text{Prob}\left[K\ i,j \text{ arrivals in } T\right] = \frac{(\lambda_{ij}\ T)\ e^{-\lambda_{ij}\ T}}{K!}$$

Hold Time Statistics

The most commonly used distribution[21] for the call duration is the exponential distribution in which the probability of a call lasting at least t seconds is given by

$$p(t) = e^{-\mu t} \qquad \text{where } \frac{1}{\mu} = \text{mean call hold time.}$$

Although two specific distributions have been used to describe the nature of the call processes, realistically a more involved situation exists. Primarily this is due to the non-stationary /

non-stationary behaviour of the calls, since average calling rates and hold times vary from period to period. In addition, other factors such as the behaviour of users who encounter call blocking etc., also have a limited effect on the true statistics.

Traffic Flow

Studies of switching networks are normally best performed on the basis of total occupancy rather than the arrival rate and hold time and in this context it is convenient to introduce the notion of traffic flow. Considering a trunk group where calls with average hold time $\frac{1}{\mu}$ arrive with rate λ the traffic flow U to the trunk is given by

$$U = \lambda . \frac{1}{\mu} \text{ Erlangs.}$$

Normally the traffic flow is specified in the dimensionless unit of Erlangs, one Erlang of traffic flow to one circuit suggesting a continuous occupancy of that facility. Similarly, a trunk with 10 lines carrying a traffic flow of 5 Erlangs would expect on average to have half of the lines busy.

3.4 Hierarchical Networks

The international telephone network now extends to virtually every country in the world, permitting 300 million users a means of voice communication and in certain circumstances, data communication. Networks of this scale however, introduce several major design and operating problems. Large volumes of traffic between any two points are generally routed most economically over direct trunks. When the volume of network users is very large and hence the effective point to point traffic relatively low, the use of direct trunks is no longer a viable proposition. Careful /

Careful network structuring is essential under these conditions to promote efficient network operations, the most widely adopted structure formed on hierarchical principles.

The hierarchical network configuration provides for the collection and distribution of traffic from a wide range of users while permitting total interconnectability for all points. Figure 3.4 displays the geographic layout of a typical hierarchical network showing how local areas are serviced by switching centres which feed into higher level units covering a wider area. Details of the functions performed and the terminology differ from country to country but as an example the AT and T network of the United States is discussed[22].

Conceptually, the AT and T hierarchical network may be represented as drawn on Figure 3.5. In this system the basic network structure consists of a five level hierarchy, each level communicating with adjacent levels. At the lowest level the end office performs a concentration and distribution service for the local network. On the next level further concentration and distribution is carried out by toll centres concerning end office traffic. Calls routed by the toll centres can be progressed further by the primary centres which carry out switching on the third level of the network. The final levels of the network hierarchy are controlled by the major switching centres, the secondary centre and the regional centre.

In addition to the backbone structure of the hierarchy, several high usage trunks between strategic centres in the network are included as shown by the broken lines in Figure 3.5. When high usage routes are established, traffic between the centres involved will first be offered to the shortest high usage route and overflow would take place through the alternatives to a last attempt on the hierarchical backbone /

backbone path.

3.5 Network Routing

The routing function in a circuit switched network decides how a call should proceed through the network, selecting one route from many possible paths. There are generally two methods[22] of achieving this:-

(1) right through routing

and

(2) own exchange routing.

In right through routing the entire route from source to destination is determined entirely at the source node. Own exchange routing on the otherhand allows for independent routing decisions at intermediate nodes as the call is moved across the network and is therefore a fully distributed routing technique.

For this study, we are concerned specifically with distributed adaptive routing and consequently consider the performance of existing own exchange routing schemes. In this section, the nature of a circuit switched adaptive routing scheme is also discussed with a brief review of previous work in this area.

Directory Routing

In a directory routing scheme, each node maintains a table indicating the next suitable node for a call with a certain destination. This table or directory is used on a "look up" basis, the call routing mechanism selecting the next node in conjunction with a destination "tag" from the call. Problems, however, result from this scheme since no protocol exists to re-route a call which fails to seize a line in the alloted trunk group due to heavy traffic or /

or component failure. The next scheme offers a more suitable approach.

Alternate Path Routing

In alternate path routing, the routing scheme is allocated a sequence of next nodes to attempt, for calls of each possible destination. The operation of such a scheme is shown with reference to Figure 3.6 in which calls arrive at Node A, destined for Node E. If the direct route AE is busy a route may be attempted via Nodes B, C and D. It is normally a design constraint that the choice of routes should be dependent on the destination in order that exceedingly long and cyclic routes are not attempted.

A straightforward technique to avoid the above problems is applied in many networks and involves a "far to near strategy". The first choice is the centre nearest to the destination. Later choices are ordered according to the available intermediate nodes which although further than the best choice are nevertheless nearer than the centre making the selection.

Applying the above strategy to Figure 3.6 and assuming calls from A to E, the first choice for switching centre A is the direct route AE. If this route is busy, the first alternative is via the nearest switch to E, which in this case is D. Switching centre D has a direct route to E; if this route is busy the call will be blocked since the possible routes via C or B would not be permitted. Again if route AD is saturated the next alternative route from A would be AC. At C the direct route CE can be attempted or as an alternative CD. In this manner, the valid routes from A to E are seen to be AE, ADE, ACE, ACDE, ABE, ABDE, ABCE and ABCDE.

The /

The above principles can be extended to cover the routing function in the hierarchical network as shown on Figure 3.7. Consider the problem of providing a route between the two primary switching centres P1 and P2. Each primary centre is parented to a different sectional centre. Similarly, the secondary centres S1 and S2 are linked to regional centres R1 and R2 respectively. With the previous routing constraints in operation, the first choice at P1 is the direct route P1-P2. The alternative choices would be via the secondary centres S2 and S1. As shown on the diagram centre S1 has the options P2, S2, R1. Regional centre R1 is permitted the choices P2, S2, R2 and R1 with centre R2 taking the direct route to P2 as first choice, followed by S2. Finally, S2 has the direct route to P as its only possibility. It should be observed that the above rules prohibit "ring around" and enables the routing decisions to be made at each switching centre independent of the call origin or past history.

Adaptive Routing

Even the alternate path routing scheme can fail under certain circumstances due to a routing doctrine which only implements routing decisions on a local basis, having no facility to act on feedback from the calls progress in later sections of the network. With adaptive routing the possibility of reacting to feedback is introduced, allowing the alteration of the routing mechanism to suit the given conditions, re-routing calls from areas of heavy local traffic and component failures to under utilised sections of the network.

Unlike the adaptive routing problem in store and forward relatively few dynamic routing schemes have been proposed for circuit switched networks. Early attempts such as reported in[24] have suggested the use of learning algorithms /

algorithms in military circuit switched networks. Later studies[25] formalising the problem through the notion of the learning automaton and carrying out an initial investigation to apply the technique to a telephone network.

In a learning routing scheme independent automata at each node select a suitable trunk group from an allowable set. A complete routing operation is therefore a chain of decisions, the route taking one of several paths. Feedback in such schemes is elegantly simple, a blocked call producing a penalty response to all automata in the chain, the successful call giving a corresponding sequence of rewards. This type of operation suggests a completely different policy to the previously described alternate path scheme where individual trunks were offered traffic until overflow occurred. The learning approach may be seen as proportioning the traffic flow over the various paths open for calls of different source/destinations, each automaton striving to minimise the blocking between itself and the destination.

The question of the optimal operation of the learning schemes may be viewed in conjunction with the following network flow assignment problem.

In general this concerns -

Given	Network Topology and Trunk capacities T_{ab}
Minimise	b the overall network blocking probability
With respect to	U^t_{ab} the offered traffic flow to each trunk group.

To achieve this, the routing scheme has to optimally proportion the traffic to each trunk group to minimise the blocking probability. The traffic offered in each trunk group is controlled by the routing scheme and may be considered /

considered by

$$U^t_{ab} = R(U_{ij})$$

where U_{ij} is the offered point to point traffic.

3.6 Simulation of Circuit Switched Networks

The use of computer simulation software is now recognised as an indispensable aid in investigations of many systems and physical phenomena. The simulation of circuit switched networks is no exception, the complex network topology and multiplicity of events, call arrivals from many sources, call completion and routing operations, preventing analytical solution in all but a few very simple networks. The remaining sections of this chapter are ultimately concerned with such a simulation and the details of two packages are discussed. Both simulators have been designed to handle general topology networks along with user defined call statistics and routing schemes and have been shown by extensive simulation runs to provide a valuable experimental service.

Computer simulation of communications networks requires a Monte-Carlo approach involving the generation of random arrivals and call hold times. The calls can then be fed to a network model which assumes the role of the switching operations and in addition maintains a network status with regards to the calls in progress, trunk and queue occupancy and also call durations. Moreover, the software is required to record the progress of call attempts, signifying the routes taken and displaying call blocking at intermediate nodes.

Conceptually the simulation of the network may be regarded as the sequencing and handling of a series of events in time. To do this two approaches may be considered

(1) /

(1) An event to event or asynchronous time scale

and

(2) An epoch to epoch or synchronous scale.

In the event to event technique, the software ascertains when various events will occur and moves the system clock to the point of the nearest event. An epoch to epoch simulator steps through time in constant jumps, checking if any component wishes to generate an event and if so facilitates the event handling. For circuit switched networks of the complexity envisaged in this study the decision to an adopt epoch to epoch simulation was taken and has since proven to be correct. Such a technique is particularly suited to the operation of a circuit switched network where typical switching events are sufficiently close to warrant a synchronous time scale.

Nodal Model

For this study, a simple switching centre has been formulated to model the operation of a typical trunk node. As shown in Figure 3.8(b) the basic device consists of a control and routing mechanism R, trunk senders S_1, . . S_m and a sender queue. Fundamentally, the role of the routing mechanism is to react to incoming routing and release signals, to select an outgoing trunk group for the call and to activate a sending mechanism to transmit a routing signal to the next node. Since the sender operation takes a finite period of time to function, a sender queue provides a facility for orderly waiting by calls. The model for the network trunks, Figure 3.8(a), consist of a pool of lines, which are common to and equally accessable to both ends.

Traffic Model

As described previously, the use of Poisson and exponential /

exponential statistics are used to describe the call arrival and holding behaviour. To generate exponentially distributed hold times[26] the exponential probability distribution function $y = F(x)$ is transformed to an inverse form $x = F^{-1}(y)$. A random variable of the correct distribution is then generated by warping a uniformly distributed deviate by the inverse function

$$X = F^{-1}(U)$$

For the exponential distribution

$$F(x) = 1 - e^{-\mu x} \qquad \text{where } \frac{1}{\mu} \text{ is the mean}$$

$$\therefore \quad F^{\cdot}(u) = -\frac{1}{\mu} \log_e(1-u) \qquad u \neq 1, \quad 0<u<1$$

Therefore by generating uniform deviates from the conventional computer random number source the above equation provides an exponential distributed random number with the desired mean.

Generating random arrivals in an epoch to epoch simulation can be handled in a extremely simple fashion. From the probability that the next arrival occurs with the next Δt sec[27].

$$\text{prob}\left[\tilde{t} \leqslant t + \Delta t\right] = 1 - e^{-\lambda \Delta t} \qquad \tilde{t} \text{ is the time of the arrival}$$

which expanded gives

$$= 1 - \left(1 - \lambda\Delta t + \frac{\lambda \Delta t^2}{2!} \quad \cdots\right)$$

$$= \lambda \Delta t \qquad \text{if } \lambda\Delta t \gg 1$$

$$\text{prob}\left[1 \text{ arrival in next } \Delta t \text{ seconds}\right] = \lambda \Delta t$$

A series of Poisson distributed arrivals can therefore be generated by comparing $\lambda\Delta t$ every Δt time units with a uniformly distributed random number.

$u \leqslant \lambda\Delta t$	arrival
$u > \lambda\Delta t$	no arrival

General Package Structure /

General Package Structure

In essence, the proposed simulator consists of a three program package and associated data structures. The main elements of the package are:-

1 A program to construct a topological data structure representative of the network and routing scheme under investigation with facilities to edit and check the data file.

2 An actual simulation program for carrying out experimental runs, taking the topological data structure as input and producing a log file of the simulation.

3 Finally, an analysis program for detailed study of the data produced during the simulation run.

The package programs and data files interact as shown in Figure 3.9. In the remaining sections, two implementations of the described simulation model are described. Initially a simple version was produced to run on a LSI 11/03 micro-computer and was successfully utilised in early simulator experiments. Later, an enhanced version to run on a DEC 20 mainframe was implemented, the greater processing power and storage of the computer permitting studies of much larger networks involving many call sources.

3.7 PDP 11 Simulation Package

In this simulator design, the network and routing scheme are modelled on a LSI 11/03 micro-computer with traffic generated /

generated by an external SDK-85 micro-computer. Time is referenced by an external clock which synchronously interrupts both the network and call source processors which are configured as shown in Figure 3.10. The SDK-85 call source also acts as an interrupt source, randomly interrupting the network to inform of a new arrival.

Prior to simulating a network, the desired topology is formed into a data structure which is used as a framework for the simulation. The program to do this is operated off-line[28] and during execution the user is prompted to input data from the keyboard, specifying the nodal data (number of senders etc.,) trunk capacities and details of the routing scheme. After formulation this file can be used repeatedly for the simulation of the specified network.

During the simulation phase the data structure is used as a basis for a node by node simulation. Calls enter the network software from the random call source and are allowed to filter through the data structure, moving from node to node until the connection is made. Details of this are recorded in the holding data structure which forms a store for all calls holding in the network, maintaining details of the lines in use and the call durations.. When a call is successfully completed, or blocked, the relevent block of data is written to the log file, indicating the history of the call. Upon completion of the simulation, a complete record of all calls is stored.

With the completed log file stored on disk the user can utilise the analysis software to extract and display the information required from the experiment. Typically this would involve the calculation of blocking probabilities, number of calls blocked at a particular node and details of the /

the learning automata conditions.

Data Structures

As indicated previously, the topological data structure is used to carry the network topology to the simulation program. Figure 3.11 displays the overall structure format, which has the general form of a ring data structure, with nodal beads logically connected with a continuous loop of pointers. Each bead contains information and a memory allocation to perform the function of the senders, sender queue and a process queue. Note that although the senders and sender queues are actual network elements, the process queue is a functional requirement of the simulator and is described later. Supplementary to the nodal data, information on the link capacities of the various trunk groups is stored in a further data block, which as the simulation proceeds details the availability of lines between the nodes.

The holding data structure is responsible for providing a record of the calls currently holding in the network. This is illustrated in Figure 3.12 which shows the overall structure along with details of the individual call beads. Basically, each bead contains a call number, the time left in the network and a complete list of nodes connected by a call. As calls finish in the network (successful completion or blocked) the holding data structure call bead is written to the log file on floppy disk. On completion of the simulation, the log file contains a complete record of all calls, and is ordered on the basis of call termination.

The use of the topological data structure provides the simulation with the flexibility to handle networks of differing topology and size, without any changes in the simulation software. In addition, the ring structure offers a natural format /

format for the processing of the network on a node by node basis. Also, the existence of the holding data structure permits an efficient means of storing the notion of a call, the final log data file being ideal for detailed analysis of the experiment after the actual simulation run.

Simulation Program

For this simulator, a real-time program consisting of three pseudo-parallel processes has been adopted. These take the form of a main process, a real-time clock process and a call arrival process; the main process dealing with the general running of all nodes, the real-time clock process handling network timing and the call arrival process allowing for the insertion of new calls. Further, since the program cannot be simulating all nodes simultaneously, the concept of a process queue is introduced. Calls arriving at a node therefore being stored in this queue until the program reaches the node, at which point they are removed and processed.

The main process of the program is responsible for the system set up and more significantly the switching operation of the network nodes. This involves sequentially moving from node to node removing calls from the process queues and allocating them to free senders. If none are available, calls must be inserted into the first in/first out stack where they wait for senders becoming available. Details of this is displayed in Figure 3.13, the flow chart showing the operation of a typical node.

At any time during the execution of the main process, the real time clock or call source may interrupt the network computer. When this occurs the appropriate interrupt routine is selected and performed.

Upon /

Upon execution of the real-time clock routine (Figure 3.14) all network functions of time are processed. For instance, calls current in the holding data structure are timed down and if found to be complete, are removed from the network, freeing facilities for subsequent calls. Other conditions affected by the real-time clock are the timing mechanisms of the senders and sender queue. When a call is timed out of a sender the line to be used is seized and the call progressed to the next node. As calls in the sender queues are timed out a check is made on the availability of senders, which if are still not free, the calls are blocked.

Calls arrive in the network when the call generator interrupts the network processor. On this occurrence, the call source and destination numbers are read from the hardware and an entry made in the process queue of the source node. Additionally, space in the holding data structure is allocated for the new call. Figure 3.15 highlights the flowchart for this process.

Call generator

The call generator provides a source of random call interrupts for the network computer each interrupt representing the birth of a new call. When the generator decides that a call has arrived, two binary numbers are presented to the network, indicating the source and destination of the call. Such an arrangement permits the use of a single call source, this having the advantage of a reduction in cost and complexity.

Generation of the Poisson distributed interrupts is implemented as described previously. From the call arrival rate λ_{ij} the probability that a call will arrive in the next second is found. Comparing this value with a random number (derived from a puesdo random binary sequence) permits /

permits a decision on interrupt pulse generation. This algorithm has proved adequate for the purpose and provided the maximum value of λ_{ij} is restricted no problems are encountered.

Operation of the complete call generator is as follows (Figure 3.16). After initialisation, the software waits for a pulse from the real-time clock. When this occurs, the arrival probability for the first call source is read from a programmable call stack and used in the manner described previously to reach an arrival decision. If the outcome dictates a new call the call source and destination are obtained from the stack and the processor interrupt line set. This function is continued until all entries in the stack have been offered the opportunity to generate an arrival. At the end of this sequence, the program marks time for the next clock pulse and the entire operation repeats.

The described call generator has been implemented on an Intel SDK-85 micro-computer system using a small but versatile program. Subsequent modification to the software allows for dynamic switches between alternative sets of data, thus enabling the simulation of non-stationary arrival statistics.

3.8 DEC 20 Simulation Package

The previously described simulation package has been used extensively for experiments on small networks, including topologies of up to 12 nodes. However, several limitations have restricted the effective use of this system as a large scale, general use simulation package. These include -

1 The power and 16-bit architecture of the microprocessor has proved a severe /

severe limitation on the processing speed of the system.

2 Large experiments were restricted by the limited storage and data transfer capacity of the floppy disks.

3 The simplicity of the call generator prevented the simulation of networks with more than 10 arrival sources.

4 The software lacked facilities for the dynamic injection of faults, i.e., link and node failures and also the ability to provide realistic non-stationary traffic statistics.

5 Finally, further experiments would require provisions to make additional on-line logging of important parameters.

With the above points noted a second simulation package was produced to run on a DEC 20 mainframe computer, with the obvious benefits of processing power and storage capacity. In addition, it has been possible to produce a more user oriented design with the inclusion of interactive graphics facilities to specify the topology and to aid with the analysis of the experiments. Although several restrictions were found with the previous implementation it can be said that in general the simulation principles were in order and subsequently these form the basis of the new design.

New Design

As in the previous design, a three program package forms the main elements of the simulator although in this version several extra files are formed during the simulation (Figure 3.9). In the pre-simulation software the user can graphically /

graphically specify (via a Tektronix 4014 Graphics Terminal) a network of any topology, setting up the trunk capacities and routing rules along with the arrival statistics of the traffic. Facilities also exist to edit the network, i.e., reposition, delete and add components, these normally used in conjunction with utility services to redraw and window the picture to different scales; both extremely useful for work on large networks. Details of this program and other elements of the package are to be found in[29].

Additional facilities offered by this software gives the option of setting up node and link failures, these being triggered at a pre-specified point in the simulation. Arrival rate switches can be programmed in much the same way.

The bulk of the actual simulation program is essentially the same as that described for the LSI 11 package. One major difference however being the integration of the node processing, time handling and call generation elements into one program to run on a conventional multi access computer system. The use of ring data structure techniques has been maintained for the topological file and extended to cope with the log file as well. In this simulator access to the data structures is through a virtual memory system, the software transparently swapping pages of both files between main memory and disk.

Figure 3.17 shows the basic structure of the topological data structure with Figure 3.18 detailing the individual component beads. Clearly the ring and direct pointers assist greatly in moving logically through the structure, manipulating calls from node to node and seizing the appropriate lines. The log data structure (Figure 3.19 and Figure 3.20) is also ideally suited for efficient call processing /

processing. Basically, this structure consists of two main rings with call beads connected to either the holding or finished rings, new calls first joining the holding ring and on completion transferring to the finished ring.

3.9 Summary

A very general introduction to circuit switched communication networks, high-lighting the aspects relevant to this study has been given. Also, a simple computer model has been proposed and software developed to carry out a range of experiments on networks operating under conventional and adaptive routing schemes.

In summarising the decisions made, the following points are emphasised:-

1. The simulator is designed to handle a general topology trunk network, the local network components acting as sources and sinks of traffic.
2. A simple nodal and traffic model has been proposed, although due to the modular nature of the software these could be altered to include greater detail.
3. The use of ring data structures offers a novel and extremely effective method of handling the complexity of a network simulation.

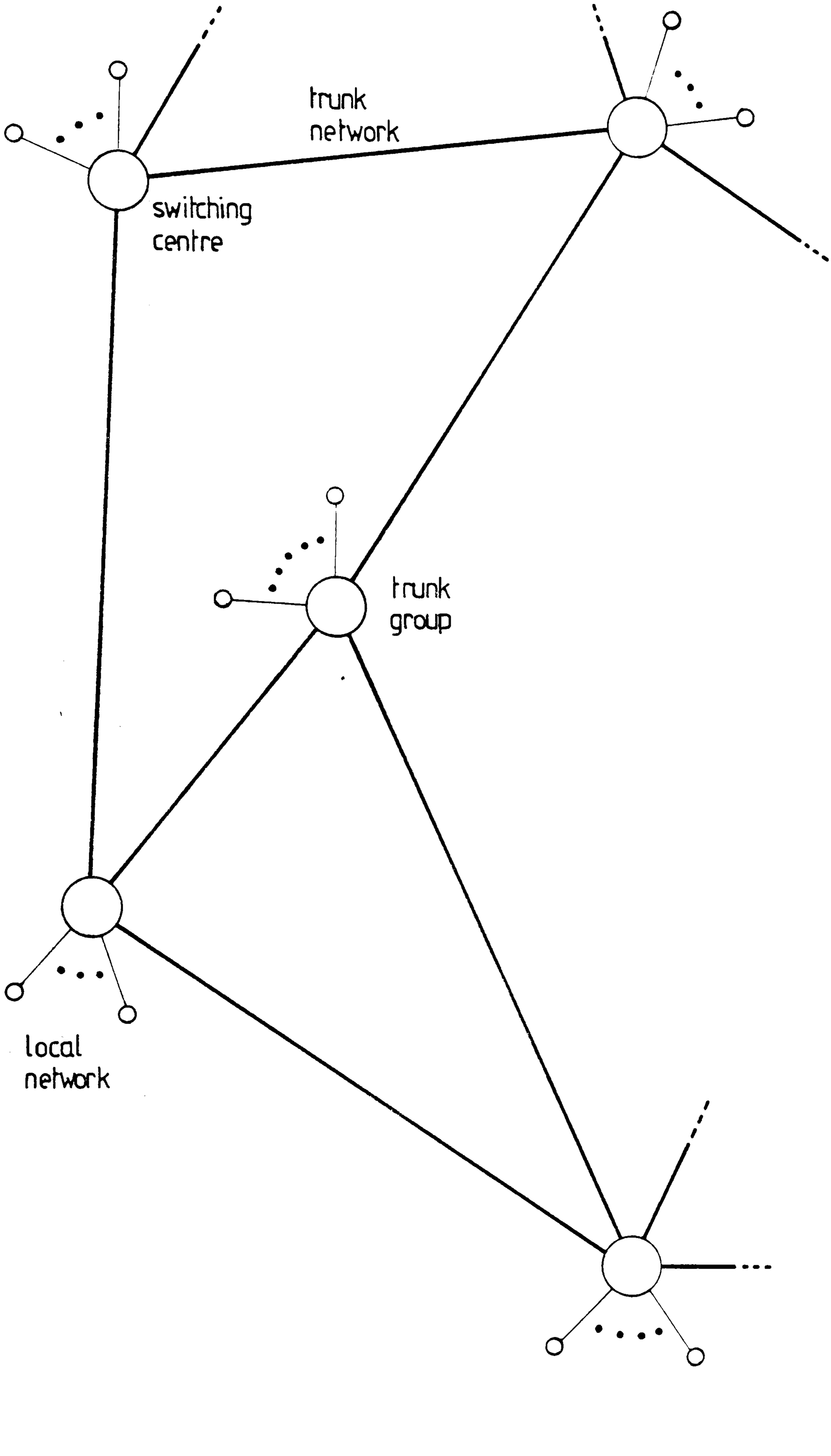


Figure 3.1 Components of a circuit switched network

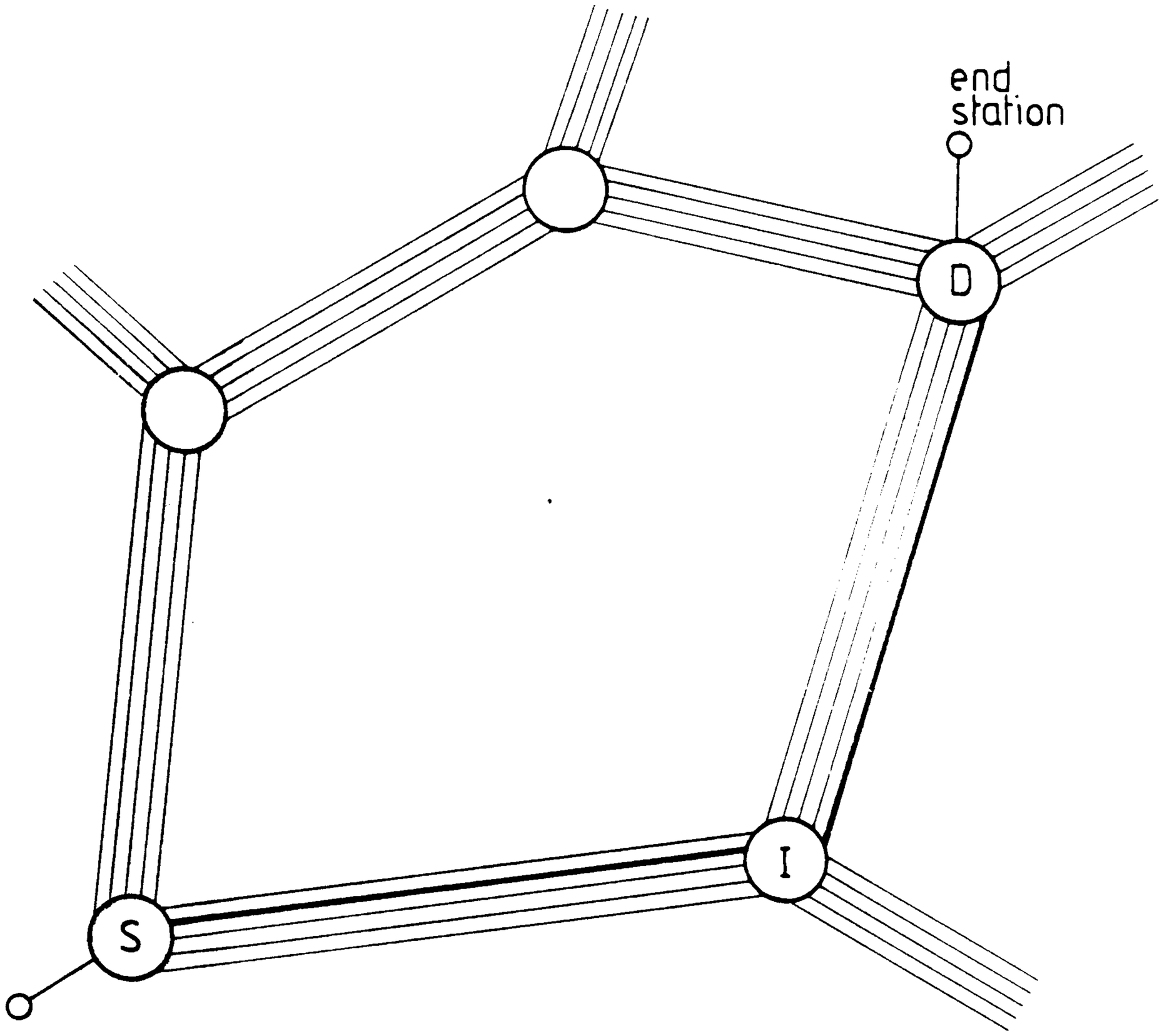


Figure 3.2 Circuit switched connection

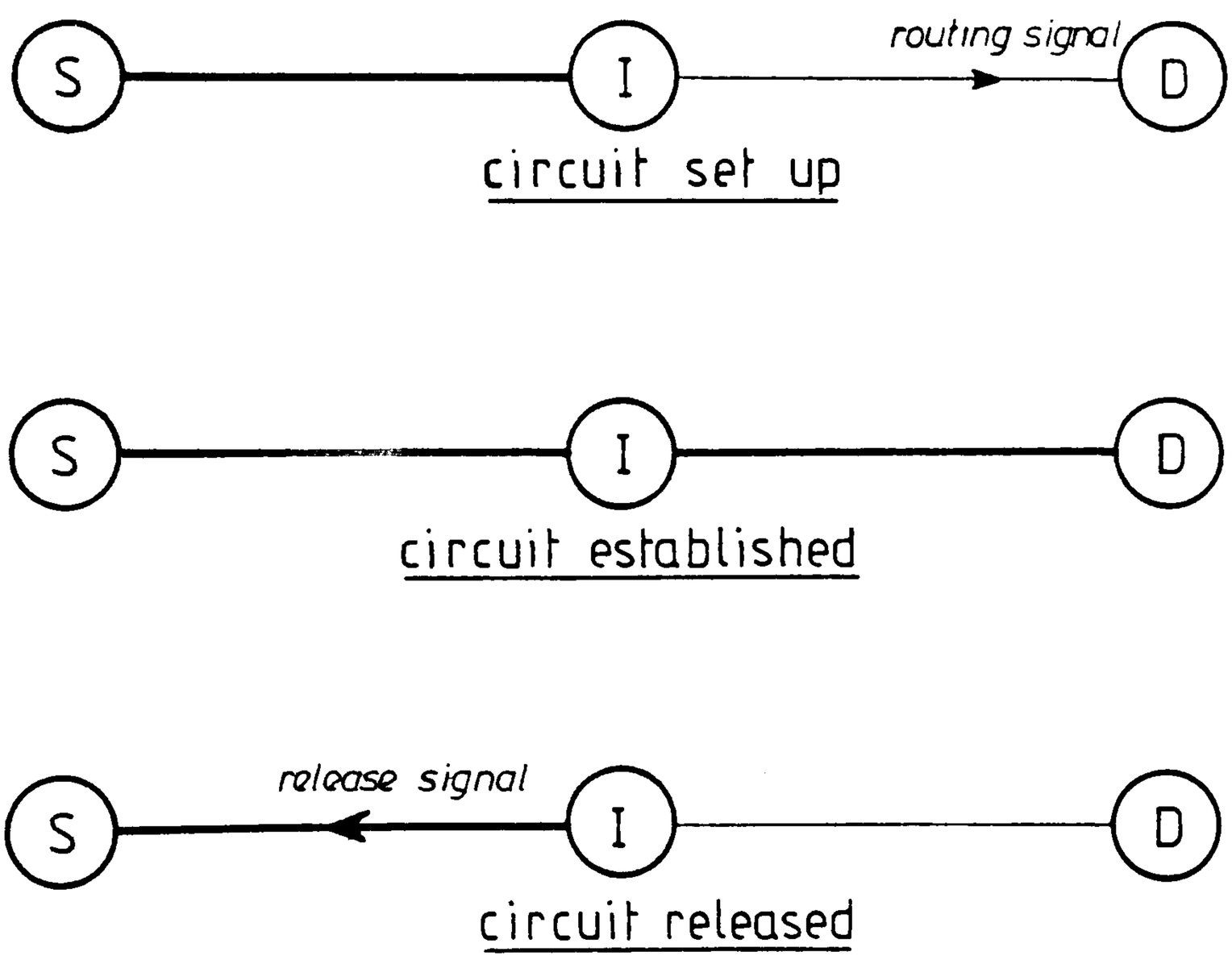


Figure 3.3 Signalling in a circuit switched network

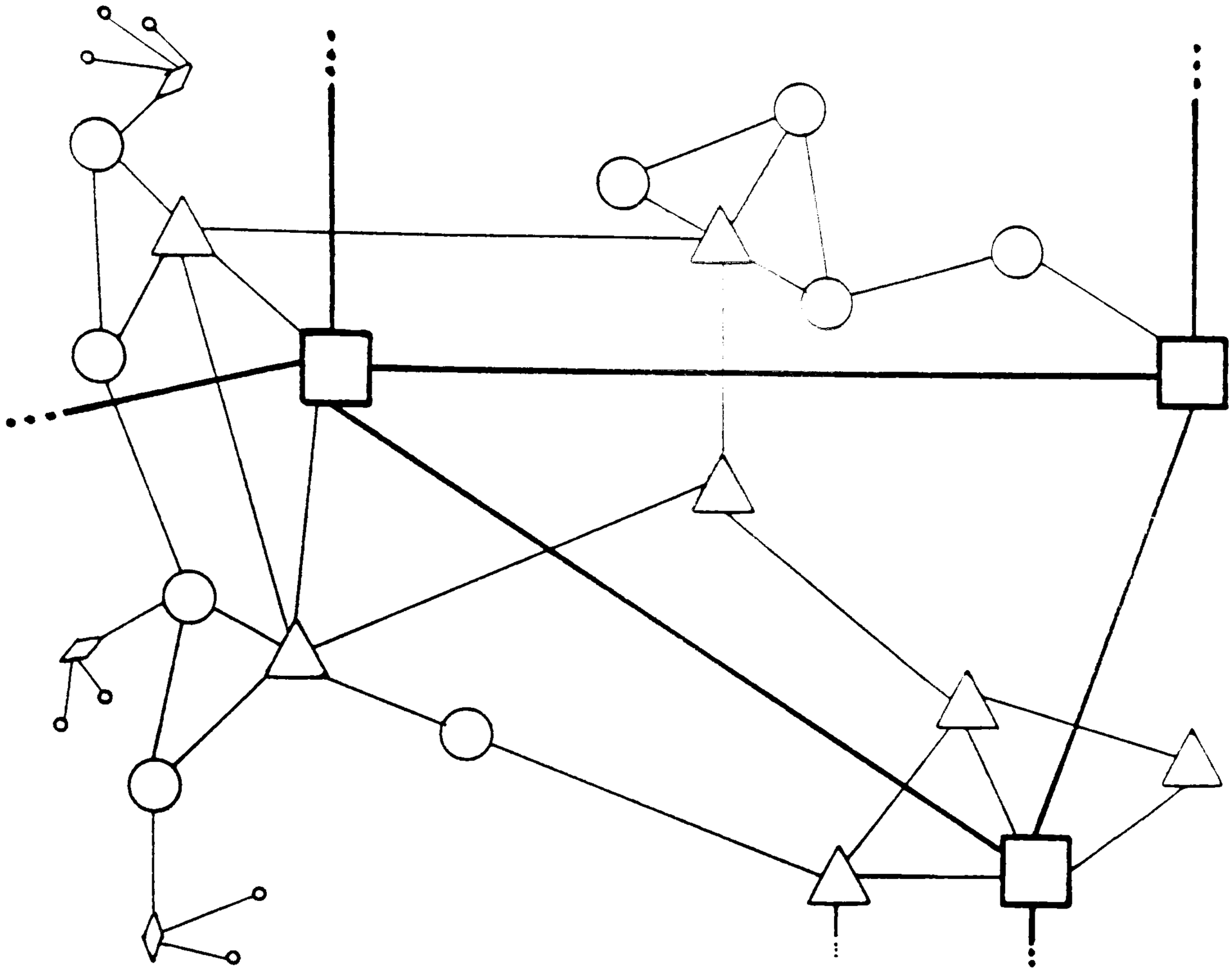


Figure 3.4 Geographic layout of a hierarchical network

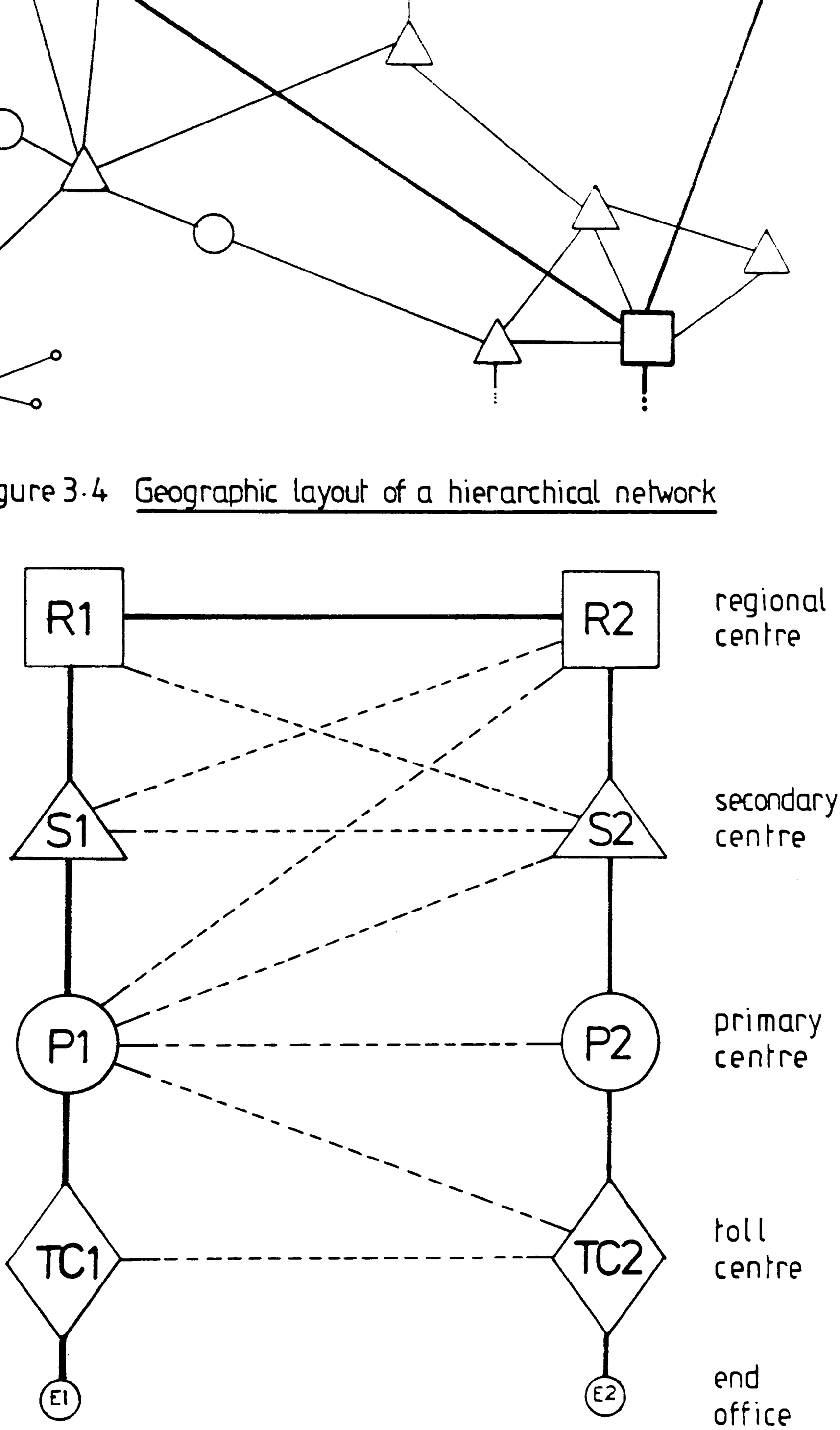


Figure 3.5 AT&T network hierarchy

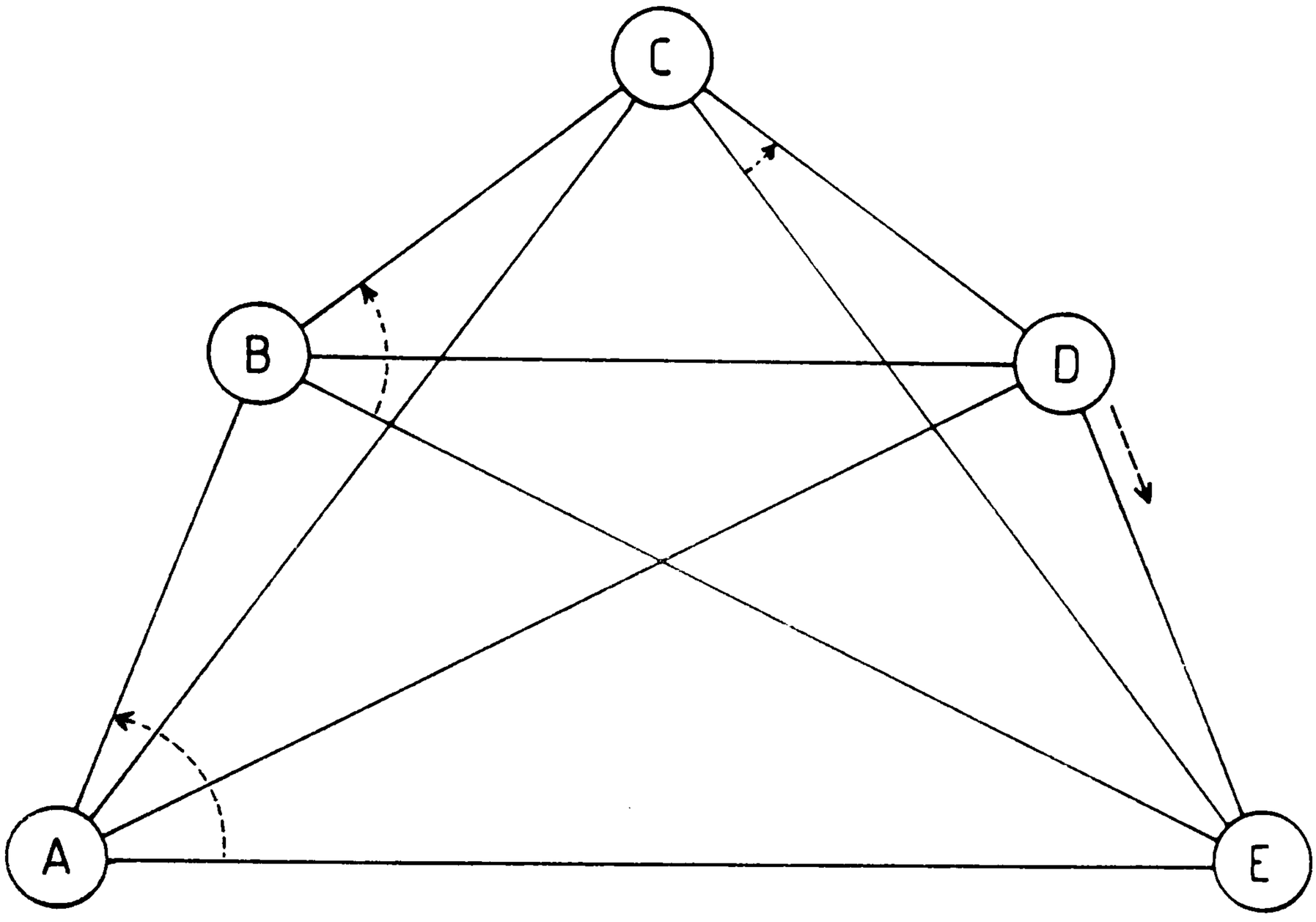


Figure 3.6 Alternate routing in a simple network

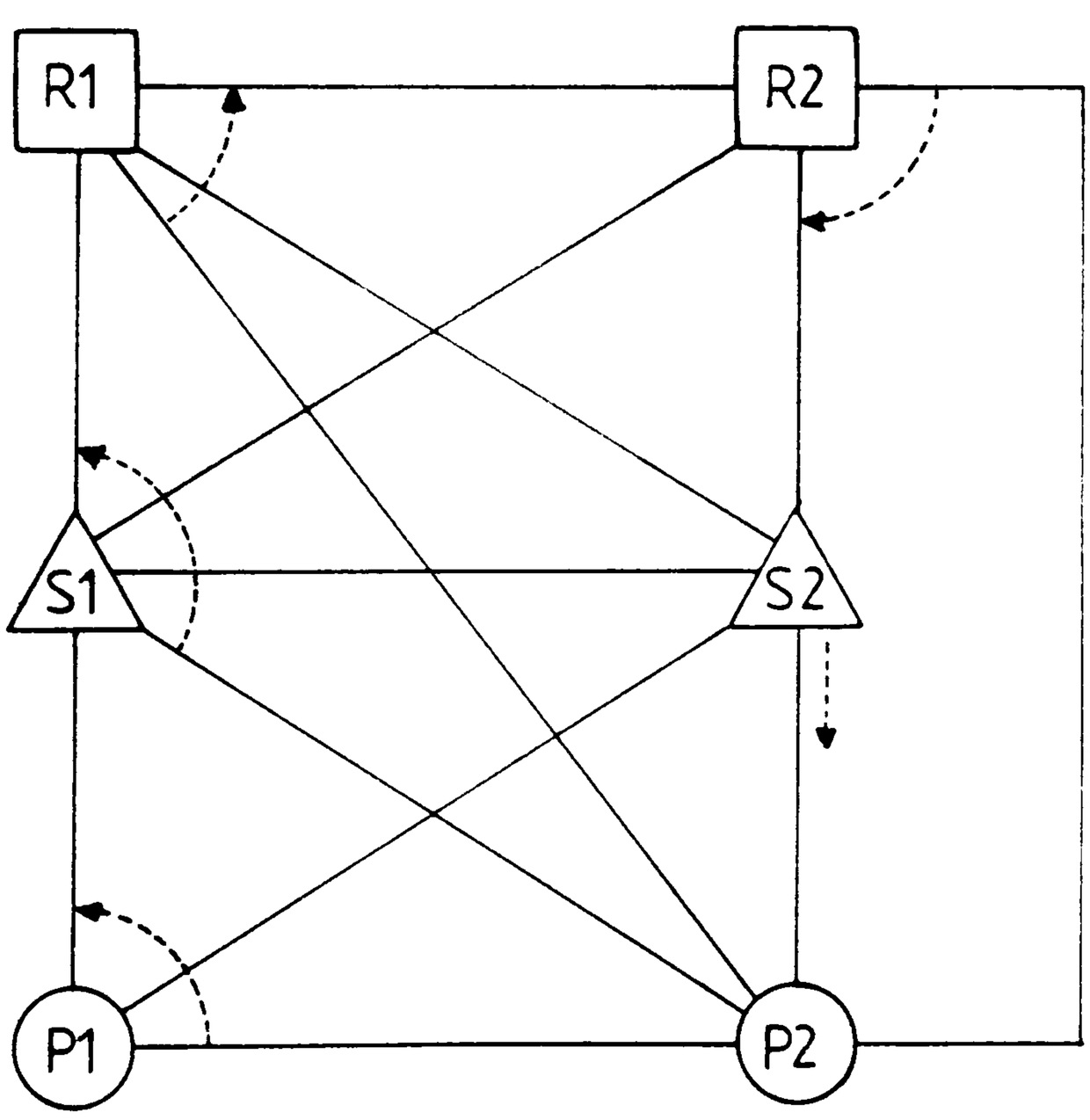


Figure 3.7 Alternate routing in a hierarchial network

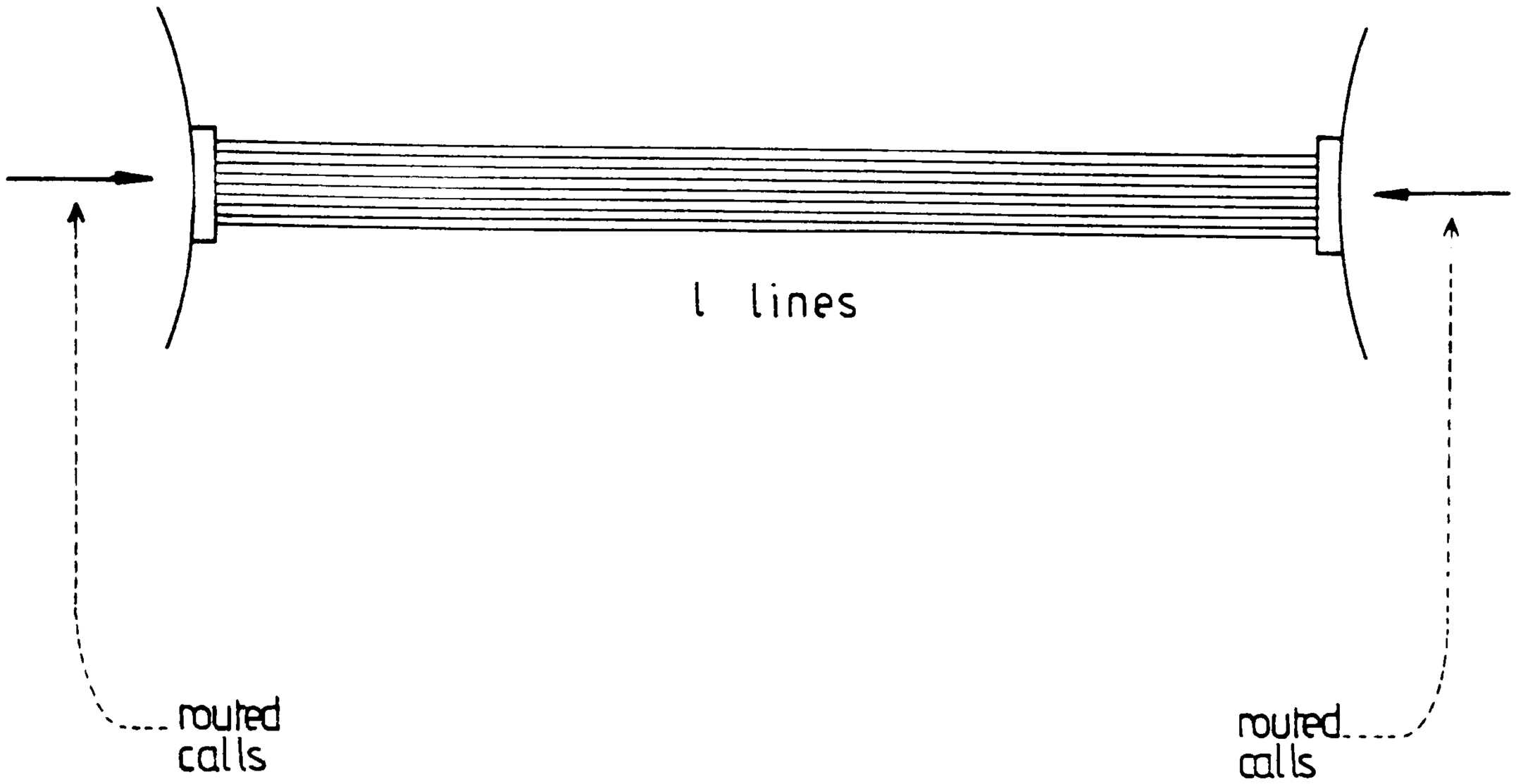


Figure 3.8a Network trunk

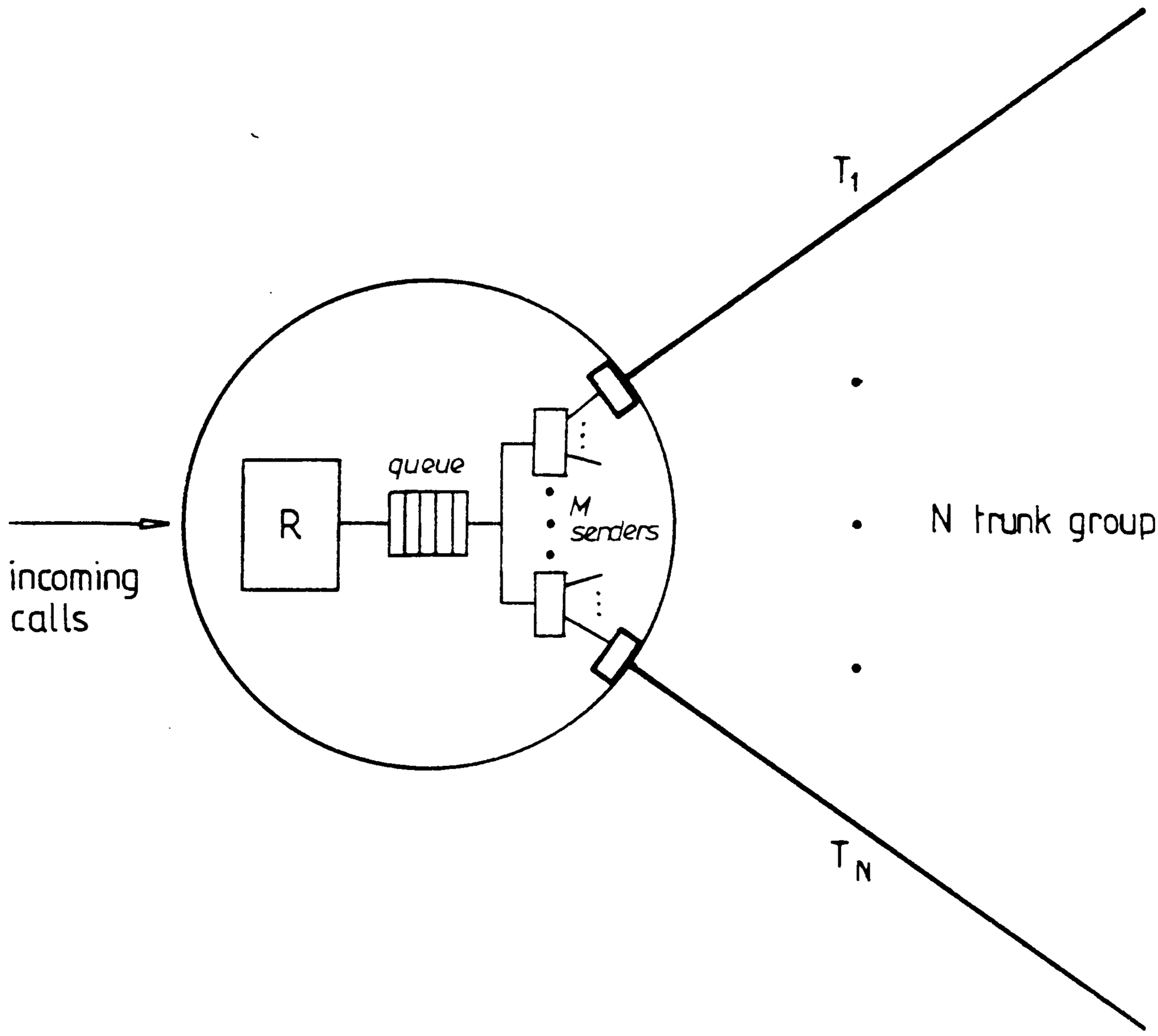


Figure 3.8b Nodal model

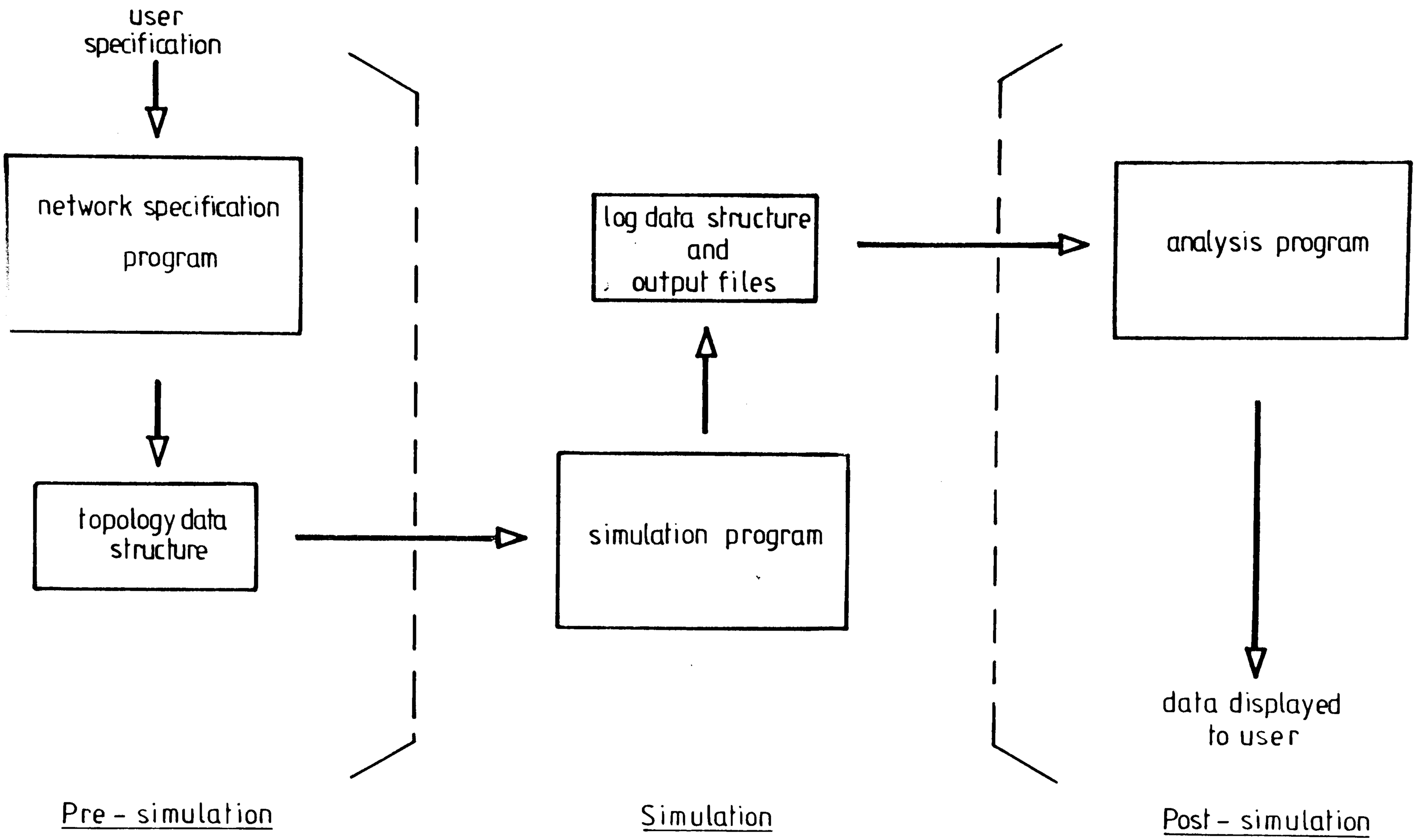


Figure 3.9 Simulation package structure

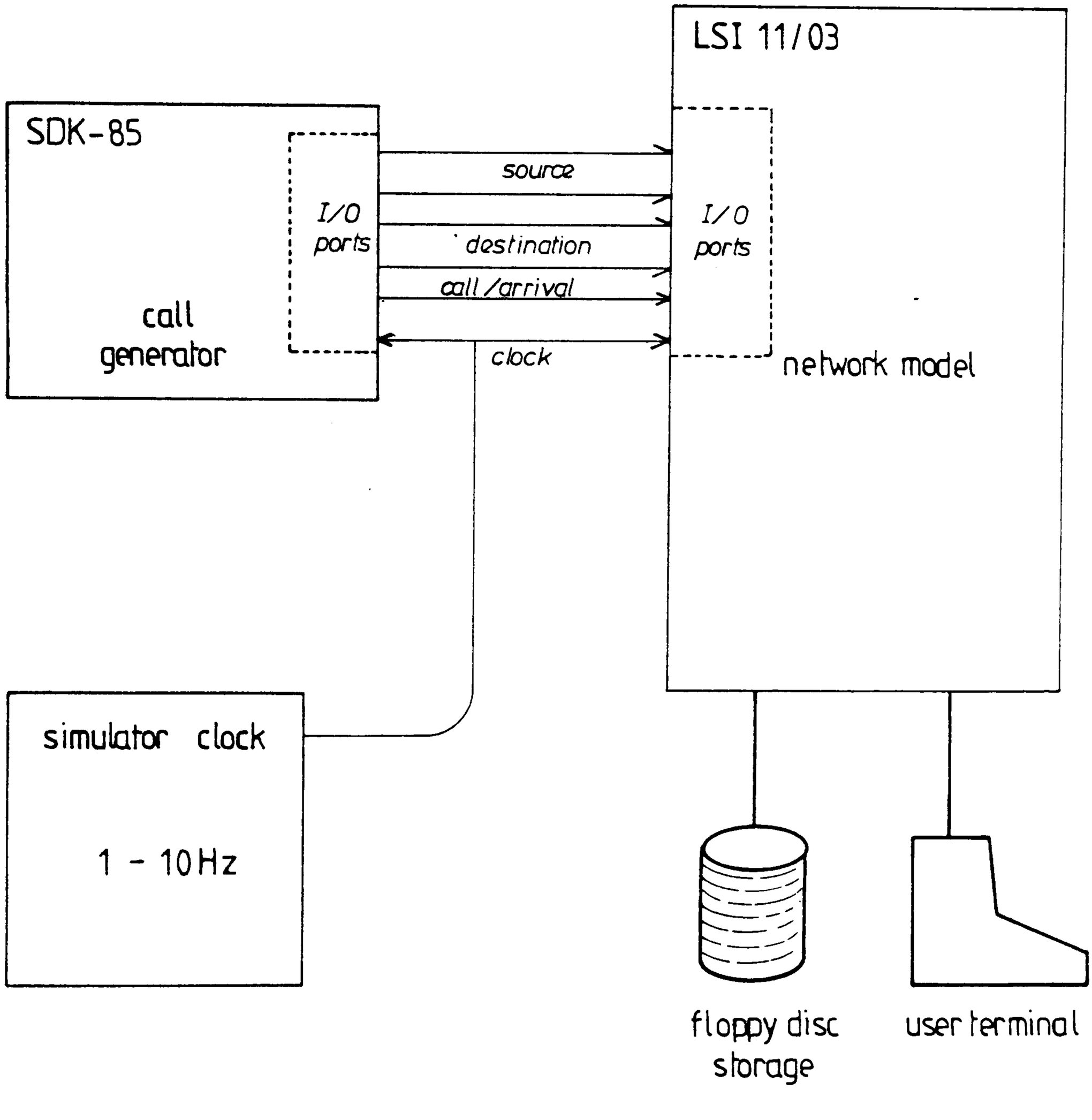


Figure 3.10 Simulator hardware configuration

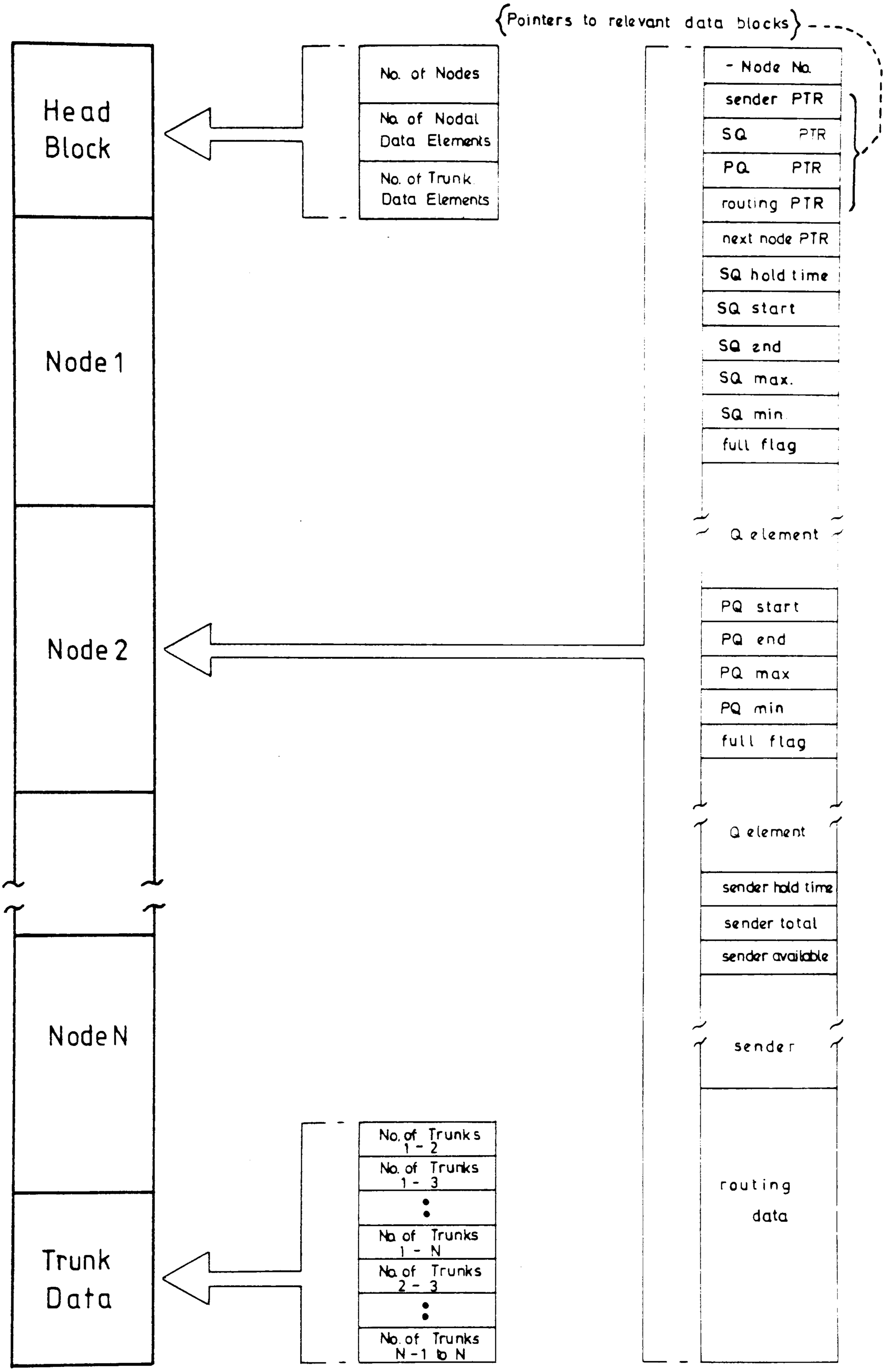


Figure 3.11 Topological data structure

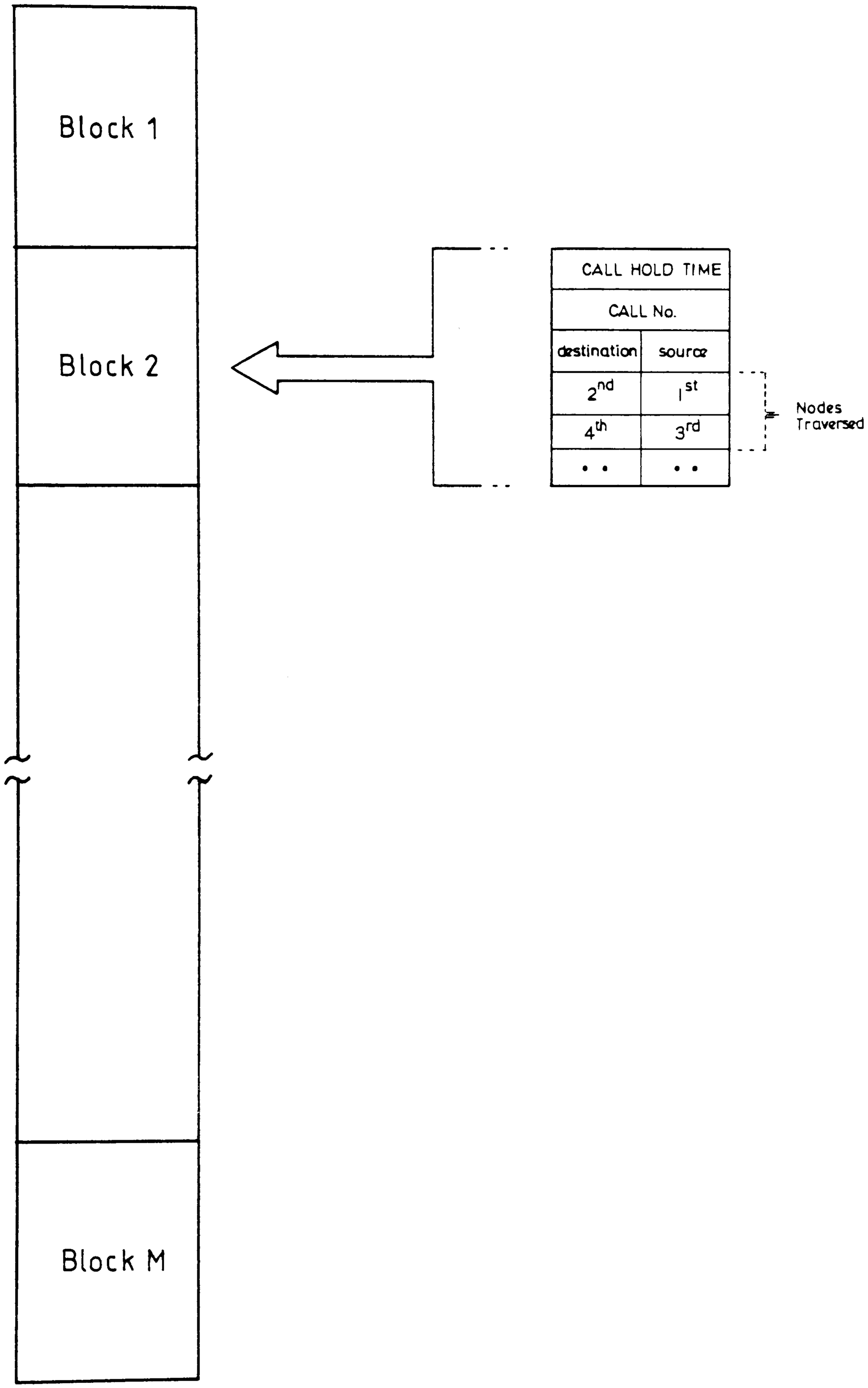


Figure 3·12 Holding data structure

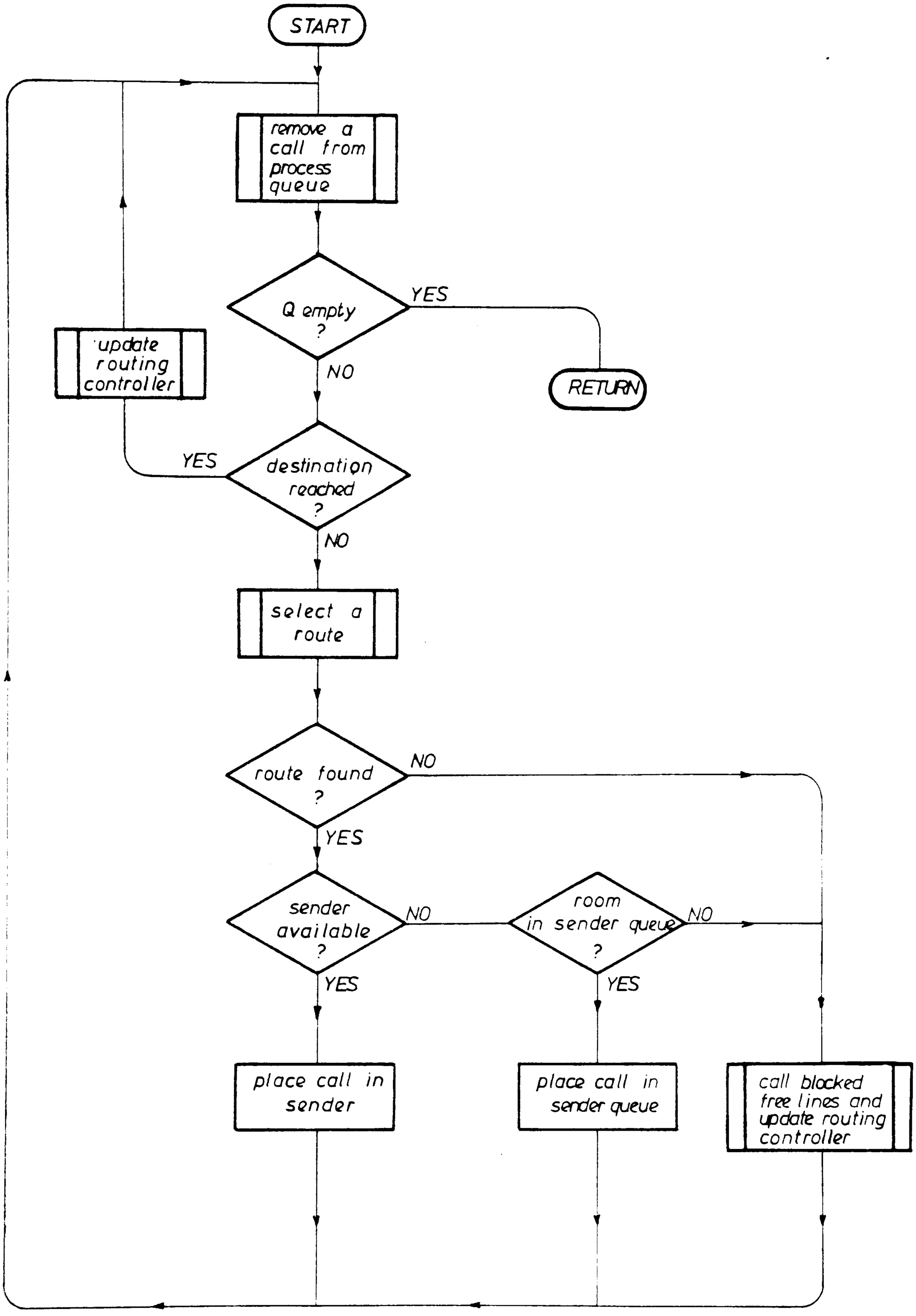


Figure 3·13 Nodal process

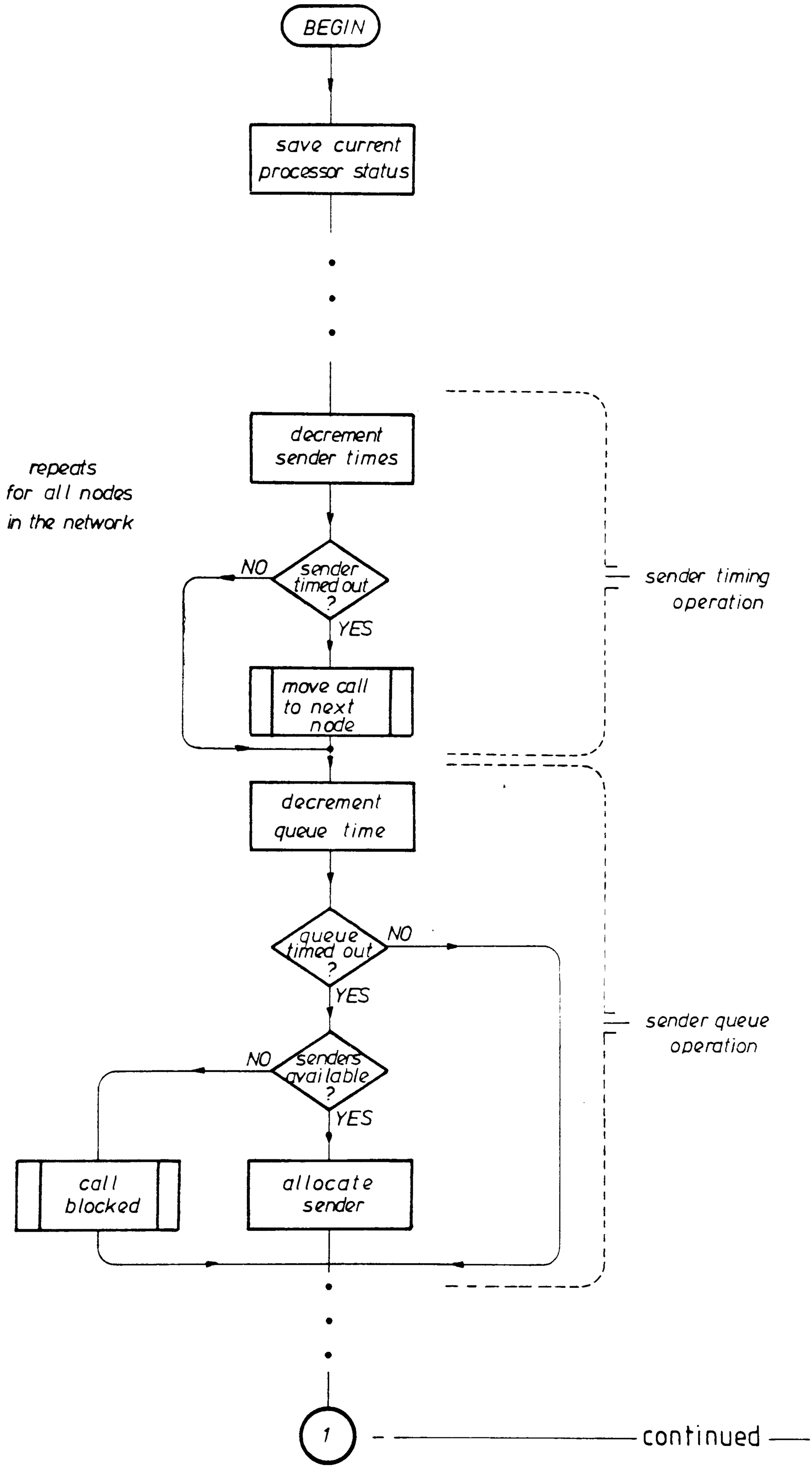


Figure 3.14 Real time clock process

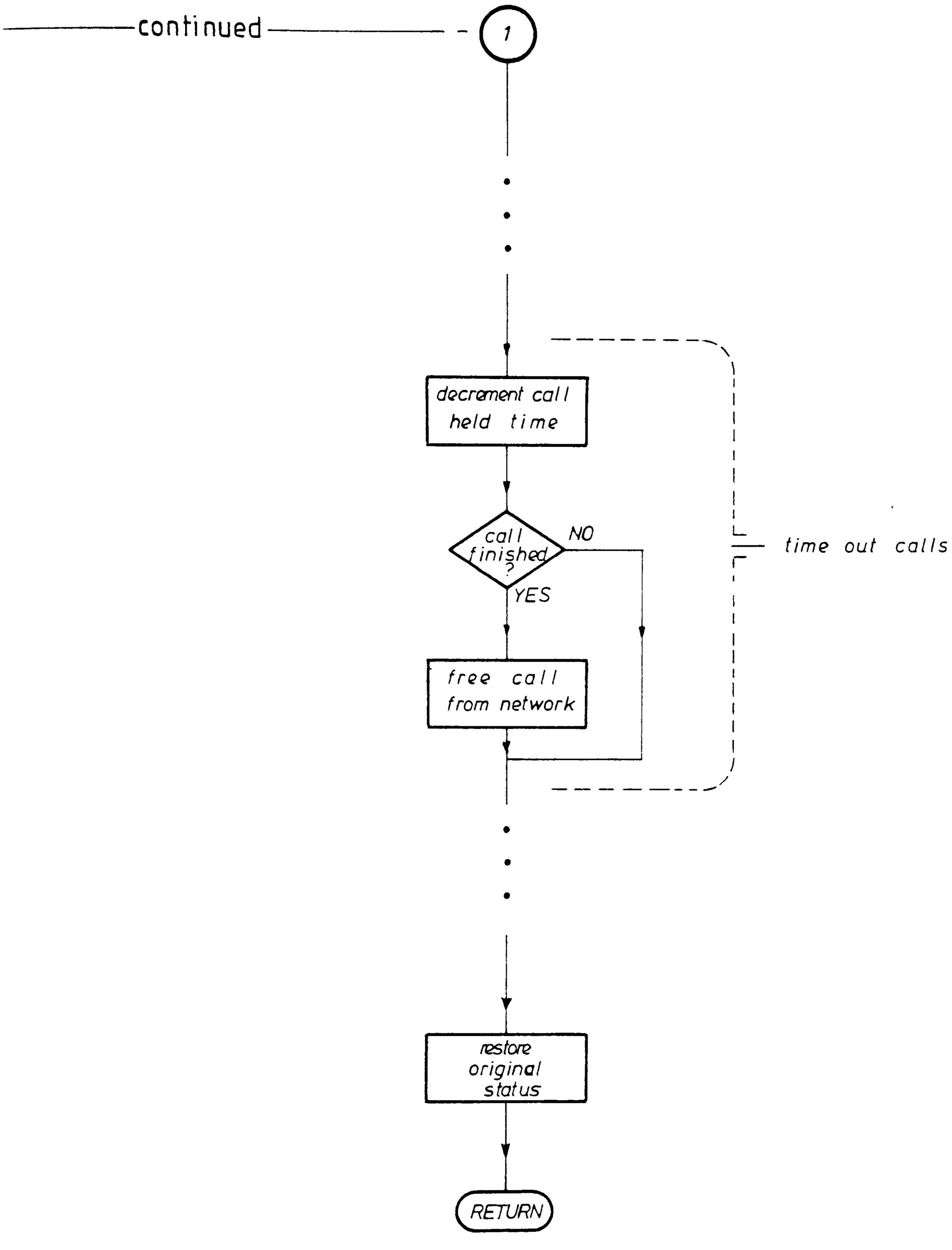


Figure 3·14 (cont.) Real time clock process

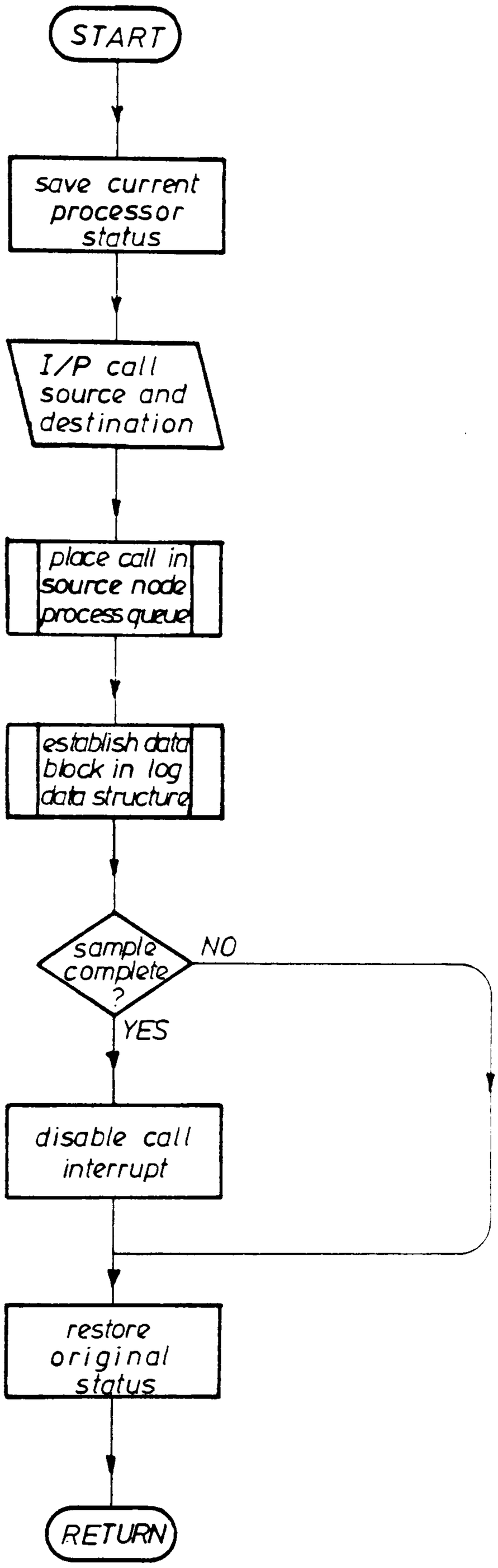


Figure 3.15 Call arrival process

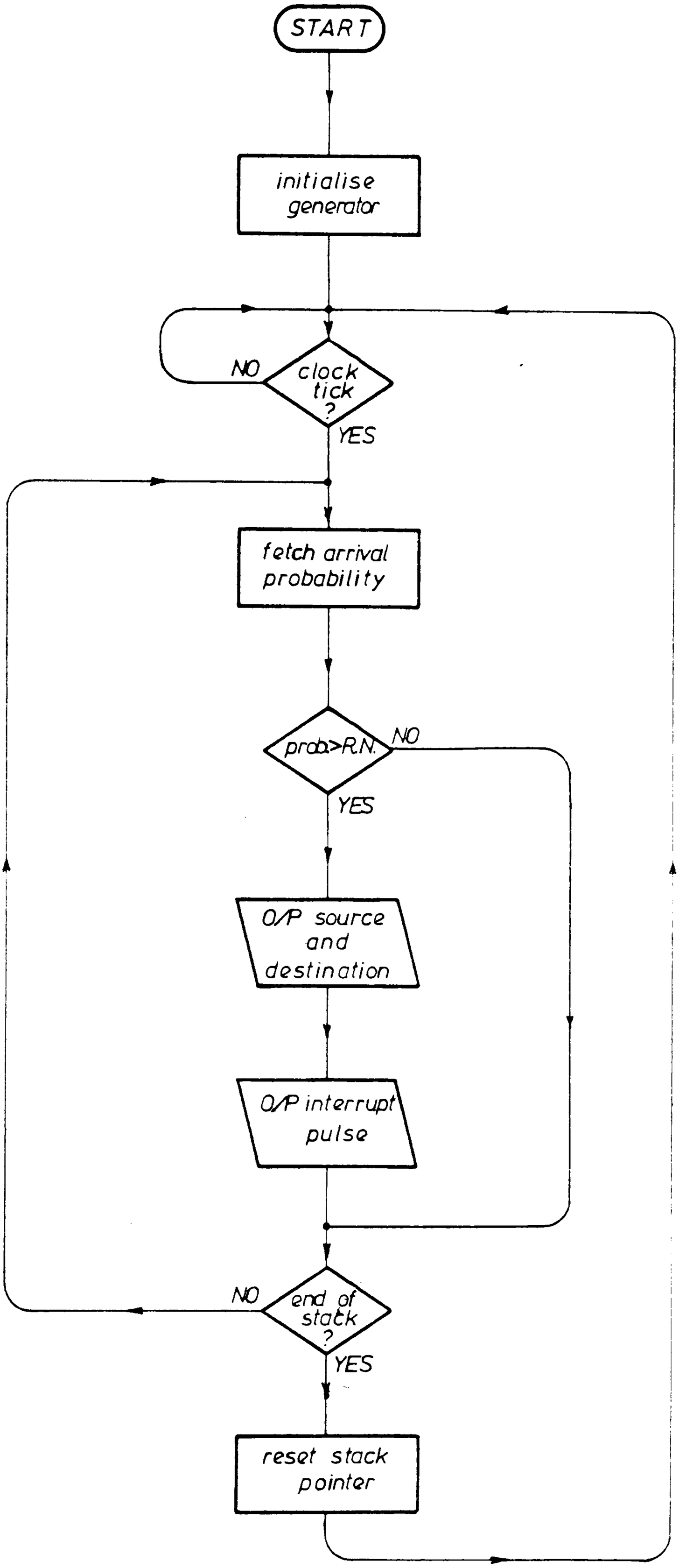


Figure 3.16 Call generator flow chart

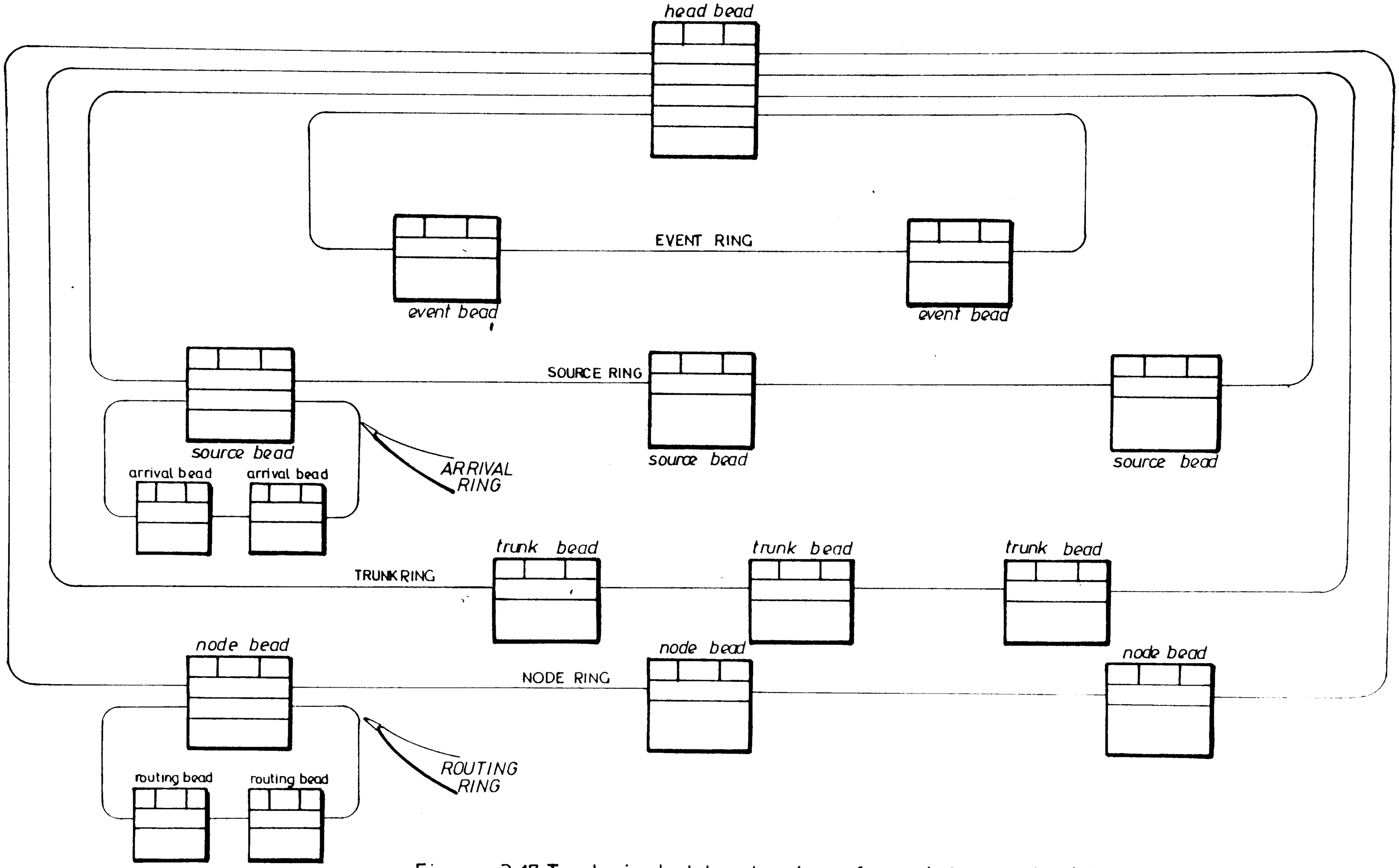


Figure 3.17 Topological data structure for mainframe simulator

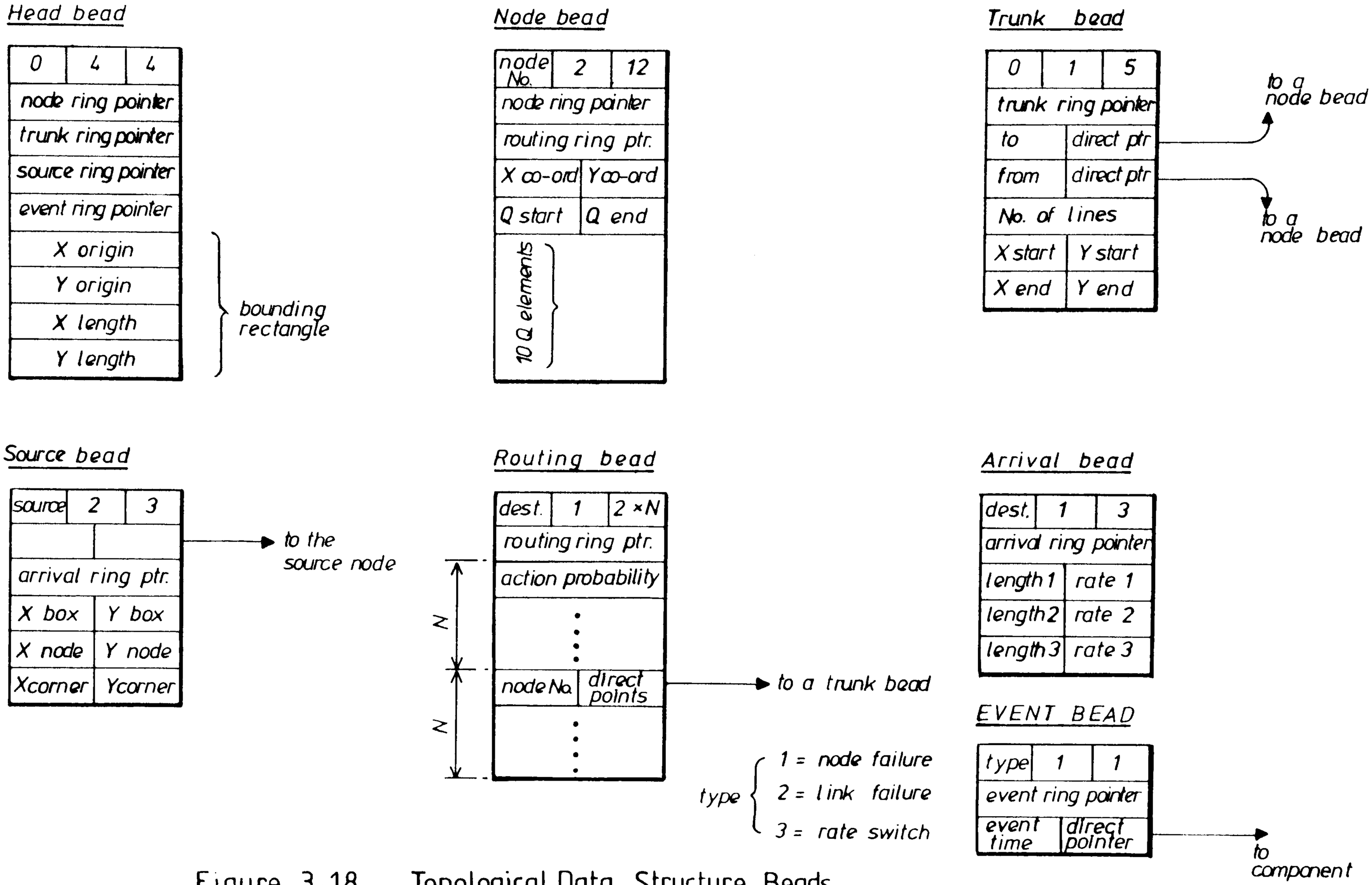


Figure 3·18 Topological Data Structure Beads

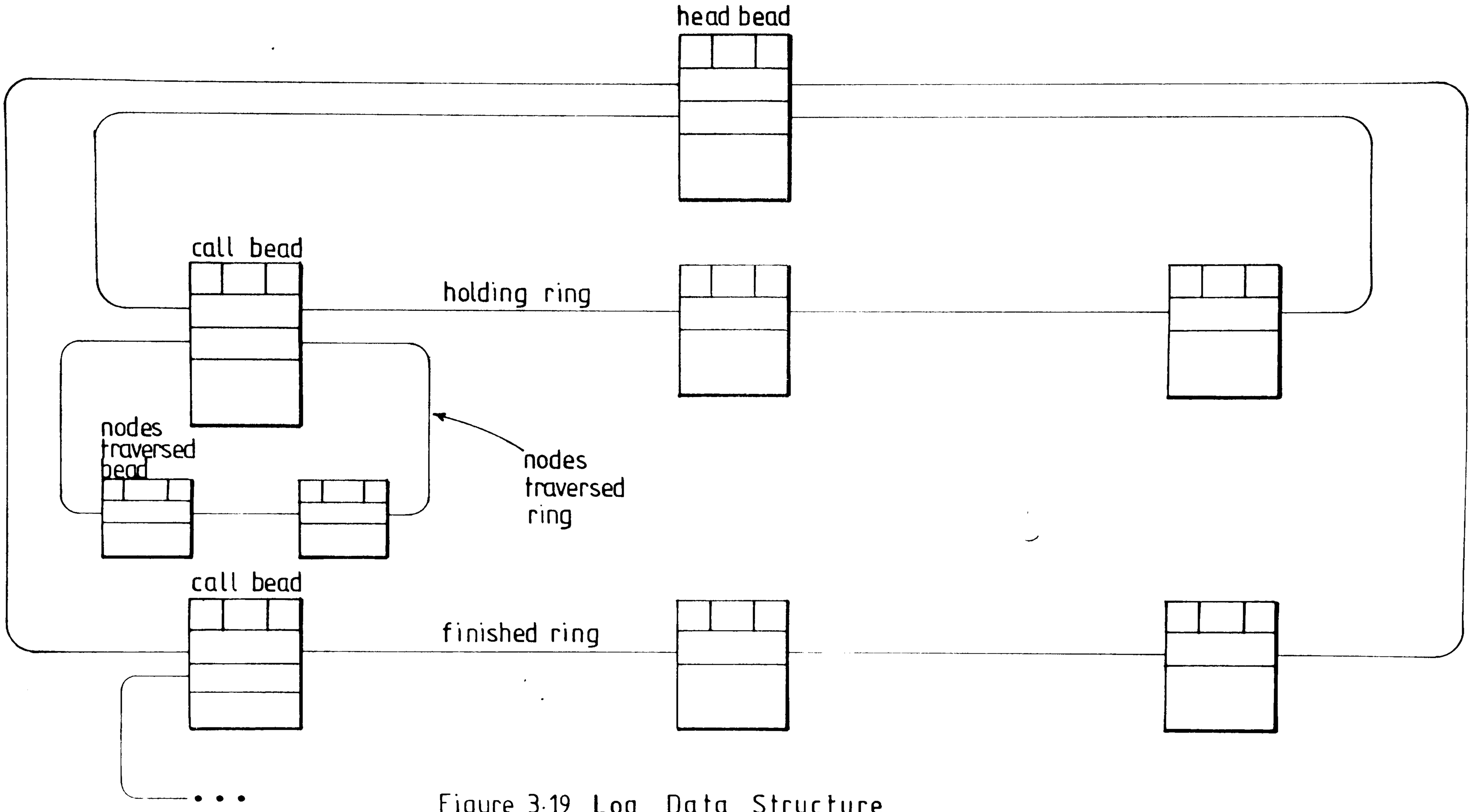
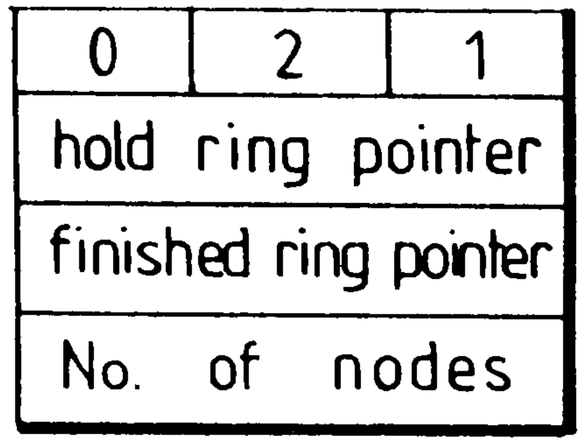
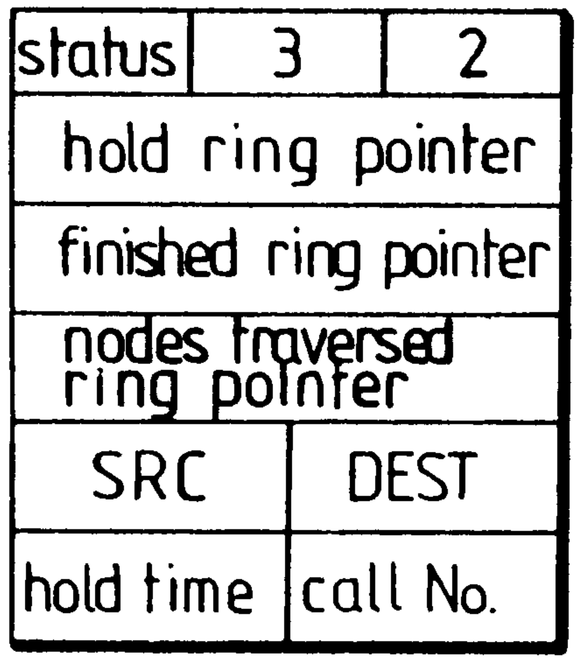


Figure 3·19 Log Data Structure

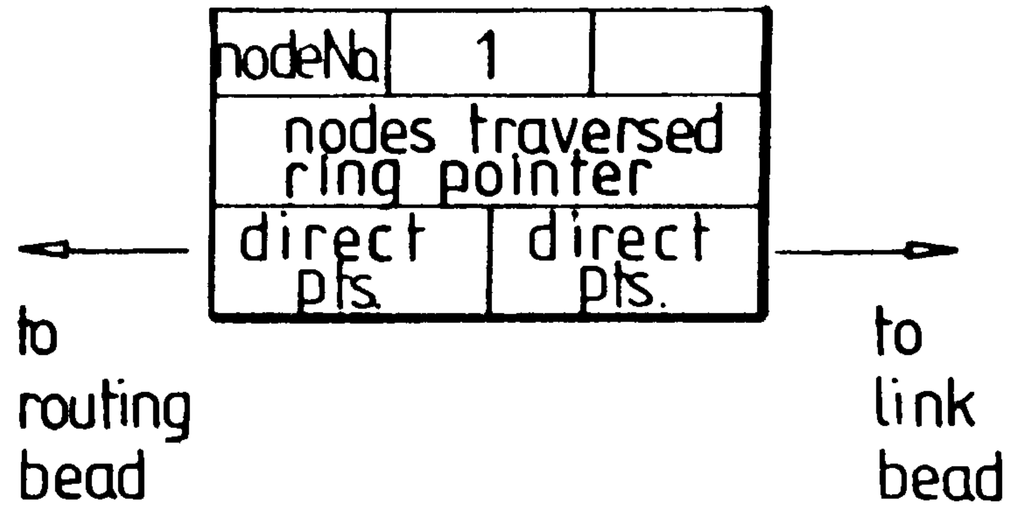
Head bead



Call bead



Node traversed bead



STATUS	0	call O.K.
	-1	blocked No. lines
	-2	blocked No. nodes
	-3	ring - around
	-4	lost due to failure

Figure 3.20 Log Data Structure Beads

CHAPTER 4

LEARNING AUTOMATA ROUTING IN ELEMENTARY CIRCUIT SWITCHED NETWORKS

4.1 Introduction

In this chapter, the learning automaton is introduced as a prospective distributed routing controller through theoretical and simulation studies of an elementary 4 node network. By using this simple network, the operation of the automaton can be studied in isolation thus permitting a detailed understanding of the learning process and the interaction between the controller and network. A further advantage in starting with a small network is the opportunity for gaining confidence with the simulation software which in later experiments is used to study networks with more complex topologies.

Initially, experiments using one automaton are observed, the routing controller selecting one of two possible paths for a single call source. Later experiments introduce two call sources and consequently two automata operate together in the same network. Also, the 4 node network is extended to a 6 node topology and the influence of the initial conditions on the L_{R-P} and L_{R-I} reinforcement algorithms investigated.

In all, three routing schemes are considered for comparison, including

1 Alternate Path Fixed Rule.

2 Automaton routing controller with the L_{R-I} reinforcement algorithm.

3 Automaton routing controller with the L_{R-P} reinforcement algorithm.

For the network proposed, the aim of the routing scheme is the /

the efficient allocation of network facilities, which if used optimally gives a greater call throughput and also a lower blocking probability. Measurement of the network performance under all routing schemes is carried out in the experiments and in addition, the steady state action and penalty probabilities noted.

The simulation results are shown to provide further confirmation of the theoretical predictions concerning the behaviour of the L_{R-I} and L_{R-P} automata in non-autonomous environments. Furthermore, through a series of identification experiments an understanding of the network response characteristics are obtained, this leading to theoretical predictions on the equilibrium conditions of the automata and further predictions on the truly optimum routing strategy.

4.2 Theory

Central to this chapter is the four node network shown in Figure 4.1, the traffic consisting of calls arriving at node 1, going to node 3 and also at node 3 going to 1 with arrival rates λ_{13} and λ_{31} respectively. The capacities of the four trunk groups are as labelled on the diagram.

For fixed routing, the alternate path scheme is defined by stating the sequence for attempting the allowable trunk groups;

$$R_{ij} = \left[T_{ia}, T_{ib} \cdots\right]$$

where R_{ij} is the routing scheme at node i, destination j, the sequence dictating T_{ia} as the first choice etc.

With the learning schemes, automata at nodes 1 and 3 select on action which corresponds to one of two possible sequences. For example, the automaton at node 1 has an action α_1 representing the sequence $\left[T_{12}, T_{14}\right]$, this being performed with probability p_1. Subsequently α_2 corresponds /

corresponds to the sequence $[T_{14}, T_{12}]$, this occurring with probability p_2. A similar arrangement exists for the automaton at node 3 with α_1 denoting the sequence $[T_{32}, T_{34}]$ with α_2 for $[T_{34}, T_{32}]$.

Feedback to an automaton occurs when the routed call successfully reaches the desired destination or all options in the chosen sequence are attempted without finding a line. The penalty set of the automaton is therefore the probability that a sequence does not find a line for any option, i.e.,

$$c_i = \text{prob}\left[\text{call is blocked on sequence } i\right]$$

By letting N represent the total number of calls processed by an automaton and N_1 N_2 the number of times each action is performed, it follows:-

$$p_1 = \frac{N_1}{N} \quad \text{and} \quad p_2 = \frac{N_2}{N} \qquad \lim\ N \to \infty$$

and

$$c_1 = \frac{N_{b1}}{N_1} \qquad c_2 = \frac{N_{b2}}{N_2}$$

where N_{bj} is the number of blocked calls due to action i

For the particular case $\lambda_{31} = 0$, the overall blocking probability of the network is by definition (again $N \to \infty$)

$$c_T = \frac{N_{b1}+N_{b2}}{N} = p_1c_1+p_2c_2 \qquad \ldots 4.1$$

$$= c_2+p_1\left[c_1-c_2\right]$$

Before proceeding further, several assumptions must be noted, which in a real network can be justified.

1 Calls rejected from the network may not be resubmitted.

2 Calls in progress can not be pre-emptied, in other words, all calls have equal priority.

3 /

3 The call set up time is negligible in relation to the average call hold time.

4 No mechanism exists for retrieving calls stranded at an intermediate node.

Traffic Flow in the Network

By considering the traffic flow in the network ($\lambda_{31} = 0$) the difference between the alternate path and the learning routing schemes can be shown. Considering first the flow patterns produced by an alternate path scheme, (Figure 4.2) the flow associated with second attempts Ub_1, resulting from blocking on T_{12}, is clearly dependent on the routing sequence and the blocking probabilities on the individual trunks. In contrast, the flow produced by the random routing policy of the learning automata (Figure 4.3) is also influenced by the action probabilities of the automaton and through variation of these a means of optimising the carried traffic is provided.

Blocking Probability

Again with $\lambda_{31} = 0$, the probability that a call from source λ_{13} being routed successfully by the alternate path scheme is

$$d_T = b_1(1-b_4)(1-b_3)+(1-b_1)(1-b_2) \quad \ldots 4.2$$

Similarly for the learning automata schemes $\ldots 4.3$

$$d_T = p_2\left[(1-b_4)(1-b_3)+b_4(1-b_1)(1-b_2)\right] + p_1\left[(1-b_1)(1-b_2)+b_1(1-b_4)(1-b_3)\right]$$

In the present form very little can be done with the expressions for for the overall blocking probability, although techniques[30] can be applied to solve for the individual trunk blocking probabilities of the alternate path scheme. However, by artificially generating certain conditions several extremely interesting characteristics of both schemes can be displayed. Let /

Let us consider two conditions, Case 1 with $l_2 > l_1$ and $l_3 > l_4$ and Case 2 with $l_4 > l_3$ and $l_1 > l_2$.

Case 1

Since $l_2 > l_1$ and $l_3 > l_4$ all blocking takes place on T_{12} and T_{14} and consequently $b_2 = 0$ and $b_3 = 0$. The overall blocking probability for the alternate path scheme is therefore given by

$$\begin{aligned} b_T &= 1-d_T \\ &= 1-\left[b_1(1-b_4)+(1-b_1)\right] \\ &= b_1 b_4 \end{aligned} \qquad \ldots 4.4$$

Giving for the learning scheme

$$\begin{aligned} b_T &= 1-p_2\left[(1-b_4)+b_4(1-b_1)\right]+p_1\left[(1-b_1)+b_1(1-b_4)\right] \\ &= p_1 b_1 b_4+p_2 b_1 b_4 \\ &= b_1 b_4 \end{aligned}$$

This interesting result revealing that under the conditions given both schemes should perform equally well, each producing the same blocking probability.

Case 2

Taking the second condition of $l_1 > l_2$ and $l_4 > l_3$ causes b_1 and $b_4 = 0$ since calls blocked at nodes 2 and 4 immediately drop lines from 1, thus maintaining a surplus to prevent blocking on T_{12} and T_{14}.

Again for the alternate path scheme

$$\begin{aligned} b_T &= 1-(1-b_2) \\ &= b_2 \end{aligned} \qquad \ldots 4.5$$

and for the learning scheme

$$\begin{aligned} b_T &= 1-\left[p_2(1-b_3)+p_1(1-b_2)\right] \\ &= p_1 b_2+p_2 b_3 \end{aligned} \qquad \ldots 4.6$$

This /

This current configuration shows a situation where the alternate path scheme provides a blocking probability of b_2, i.e., call blocking occurs on T_{23} whereas the learning scheme with a random routing strategy, attempts both T_{43} and T_{23}.

Automaton Penalty Probabilities

For the same network configuration ($\lambda_{31} = 0$) the penalty probabilities for each action may be obtained

$$c_1 = b_1 b_4 + b_2(1-b_1) + b_1 b_3(1-b_4)$$
$$c_2 = b_1 b_4 + b_3(1-b_4) + b_2 b_4(1-b_1) \quad \ldots 4.7$$

For case 1 c_1, c_2 may be reduced to give

$$c_1 = b_1 b_4$$
$$c_2 = b_1 b_4$$

This showing the penalty probabilities to be equal for both actions.

The conditions of case 2 produce a different result,

$$c_1 = b_2$$
$$c_2 = b_3$$

which are also the path blocking probabilities for the paths via node 2 and node 4 respectively. For this case the sequence action probabilities are transformed into specific trunk selection probabilities, p_1 representing the probability of selecting T_{23} with p_2 the probability of selecting T_{43}. The expected penalty probability for the network is therefore

$$c_T = p_1 b_2 + p_2 b_3$$

Reinforcement Algorithms

In Chapter 2, the non-autonomous environment was described and the operation of the L_{R-I} and L_{R-P} reinforcement algorithms in such environments explored. For /

For the L_{R-I} scheme convergence was shown to result in

$$E\left[c_1\right] = E\left[c_2\right]$$

and the L_{R-P} producing an equilibrium behaviour such that

$$E\left[p_1 c_1\right] = E\left[p_2 c_2\right]$$

Considering the conditions of case 2 and a L_{R-I} reinforcement algorithm, the operation of the automaton is to equalise the blocking probabilities on the two paths between the source and destination, giving

$$b_3 = b_2$$

$$\frac{N_{b1}}{N_1} = \frac{N_{b2}}{N_2} \qquad \lim\ N \rightarrow \infty$$

The L_{R-P} also converges to give a balance in the network, such that

$$p_1 b_2 = p_2 b_3$$

simply

$$\frac{N_1}{N} \cdot \frac{N_{b1}}{N_1} = \frac{N_2}{N} \cdot \frac{N_{b2}}{N_2} \qquad \lim\ N \rightarrow \infty$$

$$N_{b1} = N_{b2}$$

Whereas the L_{R-I} scheme equalises the blocking probability the L_{R-P} automaton acts to equalise the number of blocked calls or on a timed basis, the blocking rates.

Erlangs Loss Formula

A major contribution in switching theory was made by the mathematician Erlang who was first to quantify the probability of blocking on a trunk group. In the Erlang Loss Formula[31] the probability that a call requesting use of a line is blocked is given by

b = /

$$b = \frac{\rho^{l}/\,l!}{\sum_{k=0}^{l}\rho^{k}/\,k!} \qquad \ldots 4.8$$

where $\rho = \lambda/\mu$, λ = call arrival rate, $1/\mu$ = mean call time and l the number lines in the trunk group.

For the 4 node network (case 2) the effective input rate for paths 123 and 143 is $p_1\lambda_{13}$ and $p_2\lambda_{13}$ respectively. The path blocking and penalty probabilities may be written:

$$c_1 = \frac{(p_1\rho)^{l_2}/\,l_2!}{\sum_{k=0}^{l_2}(p_1\rho)^{k}/\,k!}$$

$$c_2 = \frac{(p_2\rho)^{l_3}/\,l_3!}{\sum_{k=0}^{l_3}(p_1\rho)^{k}/\,k!}$$

The overall blocking probability is therefore

$$c_T = \frac{(p_1\rho)^{l_2+1}/\,l_2!}{\rho\sum_{k=0}^{l_2}(p_1\rho)^{k}/\,k!} + \frac{(p_2\rho)^{l_3+1}/\,l_3!}{\rho\sum_{k=0}^{l_3}(p_2\rho)^{k}/\,k!} \qquad \ldots 4.9$$

The condition for minimum blocking probability can now be found through differentiation of c and equating to zero. This gives the following relationship which must be satisfied for optimal network performance.

$$c_1\left[l_2+1+\rho p_1(1-c_1)\right] = c_2\left[l_3+1+p_2\rho(1-c_2)\right]$$

4.3 Identification Experiments

In order to verify the mathematical model of the simple network/automaton system a sequence of controlled experiments was performed on the network of Figure 4.1, with $\lambda_{31} = 0$, $l_1 = 30$, $l_2 = 20$, $l_4 = 40$ and $l_3 = 10$.

Using /

Using a simple random routing scheme the penalty probabilities (in this case equal to the path blocking probabilities) c_1 and c_2 were measured for varying values of action probability and for a range of call rates λ. Typical experimental results for $\lambda_{13} = 10$ calls/min are shown on Figure 4.4 along with the theoretical predictions from Erlangs formula for a mean holding time $\frac{1}{\mu} = 6$ mins. This good correlation between experimental and theoretical results was found to hold for λ_{13} ranging from 5 to 30 calls/min.

Extending from the results of the above mentioned experiment something can be shown of the operating nature of the learning schemes. On Figure 4.5 we plot graphs of c_1, c_2, p_1c_1 and p_2c_2 as a function of the action probability p_1, for $\lambda_{13} = 10$ calls/min and $\frac{1}{\mu} = 6$ mins/call. Also, with the aid of Equation 4.9 the total blocking probability c_T is plotted as a function of p_1.

On this diagram the interesting areas are the cross-over points of the c_1, c_2 and p_1c_1, p_2c_2 graphs. From the intercepts of the c_1 and c_2 curves the steady state action probability p_1^* for the L_{R-I} scheme is obtained, which for Figure 4.5 gives a value of $p_1^* = 0.67$. Similarly for the L_{R-P} automaton the intercept of the p_1c_1 and p_2c_2 plots provide the equilibrium action probability $p_1^* = 0.58$.

Clearly the c_T curve is very flat within the range of the L_{R-P} and L_{R-I} steady state action probabilities which tends to support later experimental evidence of both schemes giving virtually identical blocking probabilities. Figure 4.6 shows the variation of the optimal action probability for various values of λ_{13} using the same network. As λ_{13} is decreased from 30 to 5 calls/min a change in the optimal action probability from 0.58 to 0.66 is noted. A further diagram (Figure 4.7) shows a family of convergence curves for /

for the same varying traffic. As illustrated, the p_1^* for the L_{R-P} scheme moves from 0.525 to 0.65 as λ is decreased, the L_{R-I} p_1^* tending to show little variation over the same range.

4.4 4 Node Network Simulation

Experiment 1 4 Nodes 1 Source

In this experiment the 4 node network is operated with the following network parameters

λ_{13} = 10 calls/min $l_1 = 30$, $l_2 = 20$, $l_4 = 40$, $l_3 = 10$

Routing Rules

Fixed rule at Node 1 : $[T_{12}, T_{14}]$

Learning Automata at Node 1 : $\alpha_1 = [T_{12}, T_{14}]$ $\alpha = [T_{14}, T_{12}]$

Fixed Rule	500 Calls	1 000 Calls	1 500 Calls
B P	.628	.651	.673
Calls Blocked Node 2	314	651	1 007
L_{R-I} Scheme (a = 0.02)			
B P	.464	.49	.501
Calls Blocked Node 2	144	328	508
Calls Blocked Node 4	88	162	243
L_{R-P} Scheme (a = b = 0.02)			
B P	.458	.498	.504
Calls Blocked Node 2	119	257	382
Calls Blocked Node 4	110	241	374

Automaton Steady State Conditions

	p_1^*	c_1	c_2	$p_1 c_1$	$p_2 c_2$
L_{R-I}	.667	.507	.487	-	-
L_{R-P}	.58	.44	.59	.255	.248

In /

In this experiment, we see the learning schemes accessing all useable network capacity in a topology which always gives sub-optimal results for a fixed rule. The configuration detailed, corresponds to case 2 in Section 4.2, where the alternate path fixed rule was shown only to attempt one path, which in this experiment consists of only 20 lines, consequently leading to a blocking probability of 0.673. The learning schemes on the otherhand, manage to utilise all 30 lines and as confirmed by tables of Erlangs formula[32] blocking probabilities in the region of 0.5 are obtained.

The steady state automaton behaviour matches well with previous theoretical predictions. In the L_{R-I} experiment we see the penalty probabilities, in this case also the blocking probabilities, being equalised (given slight experimental variance). The action probabilities are seen to converge to the values predicted in the previous identification experiments, this being in accordance with the trunk capacity ratio. This can also be shown using a simple linearisation of the blocking probability equation.

Let

$$c_1 = p_1 k/l_2 \qquad c_2 = p_2 k/l_3 \qquad p_1 = 1-p_2$$

For the L_{R-I} algorithm

$$c_1 = c_2$$

$$\therefore \quad p_1^* = \frac{l_2}{l_2 + l_3} = .667$$

and for the L_{R-P}

$$p_1^2 k/l_2 = p_2^2 k/l_3$$

$$\therefore \quad p^* = \frac{\sqrt{l_2}}{\sqrt{l_2}+\sqrt{l_3}} = .58$$

The experiment with the L_{R-P} automaton also behaved in the manner /

manner predicted by the simple theory. In the table above the equalisation of the condition $p_1 c_1 = p_2 c_2$ is met, with the action probabilities converging to the value found from Figure 4.5. Further evidence of the L_{R-P} convergence can be noted in the number of calls blocked at Node 2 and 4, these showing the expected equalisation of the blocking rates.

Experiment 2 4 Nodes 1 Source

In the following experiments the last network configuration is maintained but now with $\lambda_{13} = 0$ and $\lambda_{31} = 10$ calls/min.

Routing Rules

Fixed rule at Node 3 : $[T_{32}, T_{34}]$

Learning automata at Node 3 : $\alpha_1 = [T_{32}, T_{34}], \alpha_2 = [T_{34}, T_{32}]$

Fixed Rule	500 Calls	1 000 Calls	1 500 Calls
B P	.426	.53	.55
Calls Blocked Node 3	213	530	825
L_{R-I} Scheme (a = 0.02)			
B P	.468	.472	.478
Calls Blocked Node 3	234	472	717
L_{R-P} Scheme (a = b = 0.02)			
B P	.482	.501	.508
Calls Blocked Node 3	241	501	762

Automata Steady State Conditions

	p_1^*	c_1	c_2	$p_1 c_1$	$p_2 c_2$
L_{R-I}	.618	.465	.498	-	-
L_{R-P}	.58	.486	.539	.284	.227

Due to the network topology in this experiment, the automaton actions correspond to routing sequences and hence the penalty probabilities /

probabilities no longer equate to the path blocking probabilities. However, it is possible to transform the sequence probabilities to path probabilities and penalty probabilities to path blocking probabilities[33]. This is done in the following way

$$\pi^2 \triangleq \text{prob}\,[\text{path } 321 \text{ is attempted}]$$

$$\pi^4 \triangleq \text{prob}\,[\text{path } 341 \text{ is attempted}]$$

$$\pi^2 = \frac{\text{total number of } \alpha_1 + \text{ number of } \alpha_2 \text{ second attempts}}{\text{total attempts on both paths}}$$

$$\pi^4 = \frac{\text{total number of } \alpha_2 + \text{ number of } \alpha_1 \text{ second attempts}}{\text{total attempts on both paths}}$$

$$b^2 \triangleq \text{prob}\,[\text{call is rejected on path } 321]$$

$$b^4 \triangleq \text{prob}\,[\text{call is rejected on path } 341]$$

$$b^2 = \frac{\text{number blocked on path } 321}{\text{total attempts on path } 321}$$

$$b^4 = \frac{\text{number blocked on path } 341}{\text{total attempts on path } 341}$$

From the above relationships, the transformed probabilities give

	π^2	b^2	b^4	πb^2	πb^4
L_{R-I}	.552	.578	.714	.319	.32
L_{R-P}	.54	.618	.724	.334	.284

In this experiment all three schemes are shown to provide a comparable routing service, each attaining an overall blocking probability in the region of 0.5. In comparison to the last experiment, the learning schemes have no advantage over the alternate path scheme in this network configuration but would in the event of a failure assume a better routing strategy as shown in experiment 1.

Experiment 3 4 Nodes 2 Call Sources /

Experiment 3 4 Nodes 2 Call Sources

Again using the same network topology but now with two call sources $\lambda_{13} = 5$ calls/min, $\lambda_{31} = 5$ calls/min, the following simulation runs were performed with the stipulated routing schemes.

Routing Rules

Fixed Rule Node 1 : $[T_{12}, T_{14}]$ Node 3 : $[T_{32}, T_{34}]$

Learning Automata Node 1 : $\alpha_1 = [T_{12}, T_{14}]$ $\alpha_2 : [T_{14}, T_{12}]$

Node 3 : $\alpha_1 = [T_{32}, T_{34}]$ $\alpha_2 : [T_{34}, T_{32}]$

Fixed Rule	500 Calls	1 000 Calls	1 500 Calls
Overall B P	.494	.511	.528
B P λ_{13}	.637	.636	.643
B P λ_{31}	.336	.378	.411
L_{R-I} Scheme (a = 0.02)			
Overall B P	.436	.48	.498
B P λ_{13}	.554	.591	.618
B P λ_{31}	.327	.376	.39
L_{R-P} Scheme (a = b = 0.02)			
Overall B P	.448	.496	.51
B P λ_{13}	.537	.586	.61
B P λ_{31}	.364	.41	.41

Automata Steady State Conditions

	p_1^1	p_1^3	c_1^1	c_2^1	c_1^3	c_2^3
L_{R-I}	.675	.787	.598	.641	.378	.433
L_{R-P}	.59	.54	.543	.707	.402	.42

p_i^j action i at node j

This experiment has been carried out to show an automata routing scheme optimising the network performance under multi-source conditions. As demonstrated, both L A schemes provide /

provide a similar grade of service, outperforming the fixed rule by approximately 0.025.

4.5 6 Node Network Simulation

The objective of the following experiments is the study of the steady state automaton action probabilities in relation to the initial conditions. As is shown in the results the steady state conditions of the L_{R-I} are influenced by the initial action probability vector, the ergodic L_{R-P} scheme however converging to a unique equilibrium irrespective of the initial action probabilities.

In order to carry out further experiments with the 2 path arrangement detailed in previous experiments, the 4 node network was modified to a 6 node topology as shown in Figure 4.8. This has the desired effect of transforming the actions of both automata from routing sequences to specific path selections and equating the penalty probabilities to path blocking probabilities.

Experiment 1 Un-biased Initial State

For the network of Figure 4.8 with $\lambda_{14} = \lambda_{41} = 5$ calls/min, $l = 40$, $l_1 = 20$, $l_2 = 10$ and all automata with equal action probabilities $p_i^j = 0.5$.

L_{R-I} Scheme (a = 0.02)	1000 Calls	2000 Calls	3000 Calls
Overall B P	.502	.519	.529
λ_{14} B P	.505	.531	.543
λ_{41} B P	.499	.507	.515
L_{R-P} Scheme (a = b = 0.02)			
Overall B P	.508	.52	.515
λ_{14} B P	.511	.521	.517
λ_{41} B P	.505	.519	.513

Automata Steady State Conditions /

Automata Steady State Conditions

	p_1^1	p_1^4	c_1^1	c_2^1	c_1^4	c_2^4
L_{R-I}	.304	.935	.532	.582	.494	.552
L_{R-P}	.57	.6	.473	.632	.446	.661

This experiment yields a very interesting result. As shown by the blocking probability, results both schemes give a virtually identical performance to the previous measurements. The same applies to the L_{R-P} steady state conditions, demonstrating the nature of the ergodic reinforcement algorithm to converge to a unique action distribution. In contrast, the convergence of the L_{R-I} scheme has obviously been affected by the initial conditions, with a clearly noticeable change in the action probabilities. The penalty probabilities on the otherhand remain unchanged (given acceptable variance), the scheme continuing to equalise the path blocking probabilities.

Assuming a linear relationship between the blocking and action probabilities $b_i = kp_i/l_i = k_i p_i$ the L_{R-I} automaton converges under all conditions such that

$$\text{prob}\left[\text{blocking on path } 1\right] = \text{prob}\left[\text{blocking on path } 2\right]$$

$$k_1(\lambda_{14}p_1^1 + \lambda_{41}p_1^4) = k_2(\lambda_{14}p_2^1 + \lambda_{41}p_2^4)$$

$$\lambda_{14}p_1^1 + \lambda_{41}p_1^4 = (\lambda_{14}+\lambda_{41})\frac{k_2}{k_2+k_1}$$

$$= (\lambda_{14}+\lambda_{41})\frac{l_1}{l_1+l_2} \quad . \; . \; 4.10$$

Equation 4.10 reveals the condition to be met for steady state operation of the automata at both ends of the network. Clearly no unique value of $\underline{p}^1$ or $\underline{p}^4$ exists, steady state existing for all values of p_1^1 and p_1^4 which satisfies the condition. /

condition.

4.6 Summary and Conclusions

Working on a thesis of understanding the detailed operation of individual routing controllers a series of analytic and simulation studies on simple 2 path networks have been reported. These have shown the operation of the learning schemes, demonstrated the influence of the reinforcement algorithms and in addition defined short-comings in the alternate path routing scheme under certain conditions. Further experiments with the L_{R-I} and L_{R-P} routing schemes have investigated the effect of the initial conditions on the steady state behaviour of the automata. As expected, the ergodic L_{R-P} scheme always converged to a unique distribution.

From this chapter several important conclusions can be summarised.

1 Even in a simple network, circumstances can prevent the alternate path routing scheme from making efficient use of the available network facilities.

2 In the case when an optimal alternate path scheme is possible, the adaptive schemes can only match the alternate path performance.

3 However, when only a sub-optimal fixed scheme is implemented the learning schemes show an improvement by utilising the unused network capacity.

4 Although neither the L_{R-I} nor L_{R-P} schemes converge to the true optimum, the nature of the performance curve is such /

such that they both give very near to optimal results.

As expected, the ergodic L_{R-P} algorithm gives convergence to steady state action probabilities independent of the initial conditions. In contrast, the L_{R-I} scheme converges to values influenced by the starting values, although still operating to equalise the penalty probabilities.

In the next two chapters the learning automata routing schemes are extended to operate in larger, more practical networks. As will be shown, the results obtained endorse the conclusions drawn from these simple experiments and in addition, show the learning schemes to be vastly superior to fixed schemes under network fault conditions.

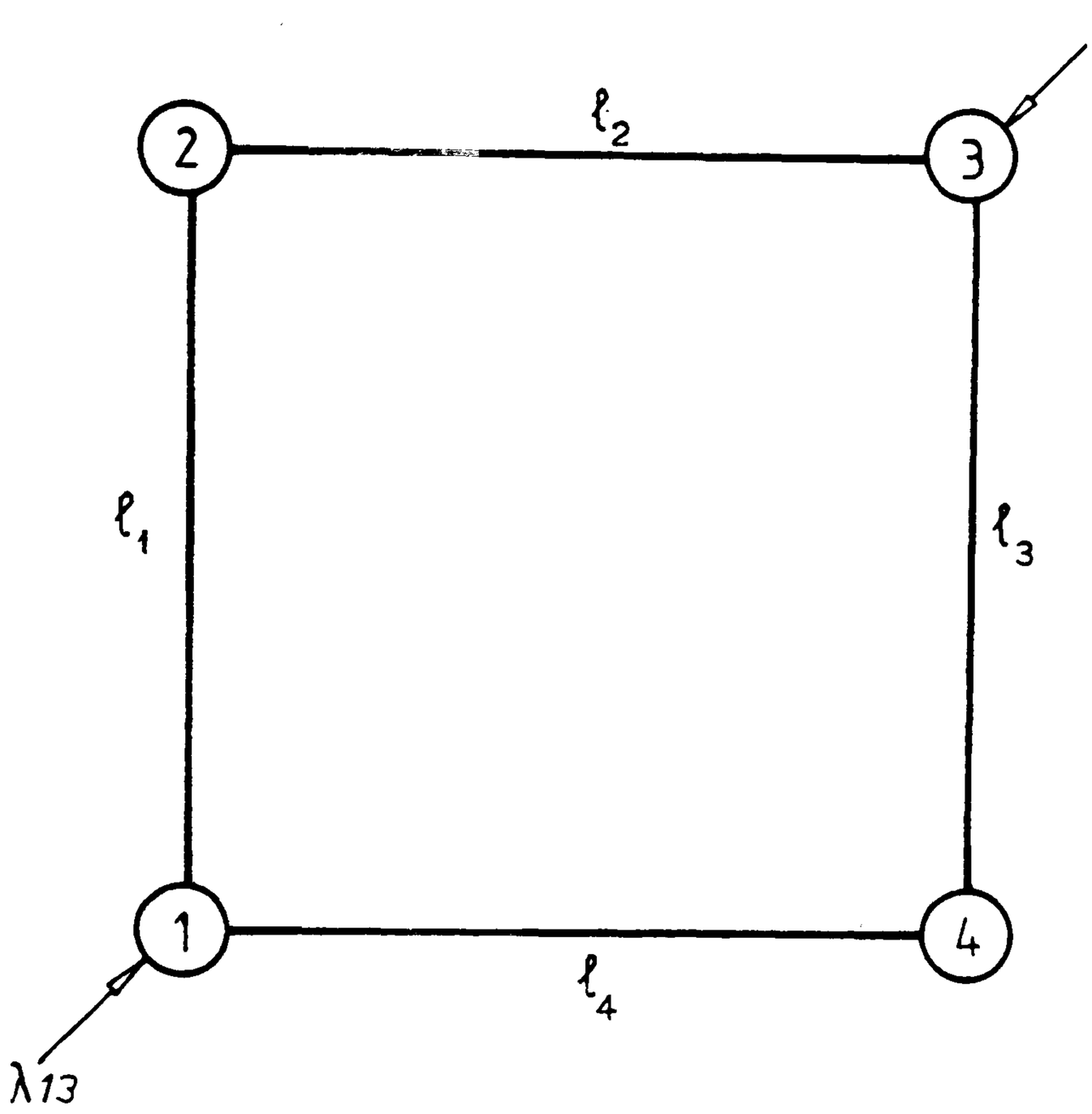
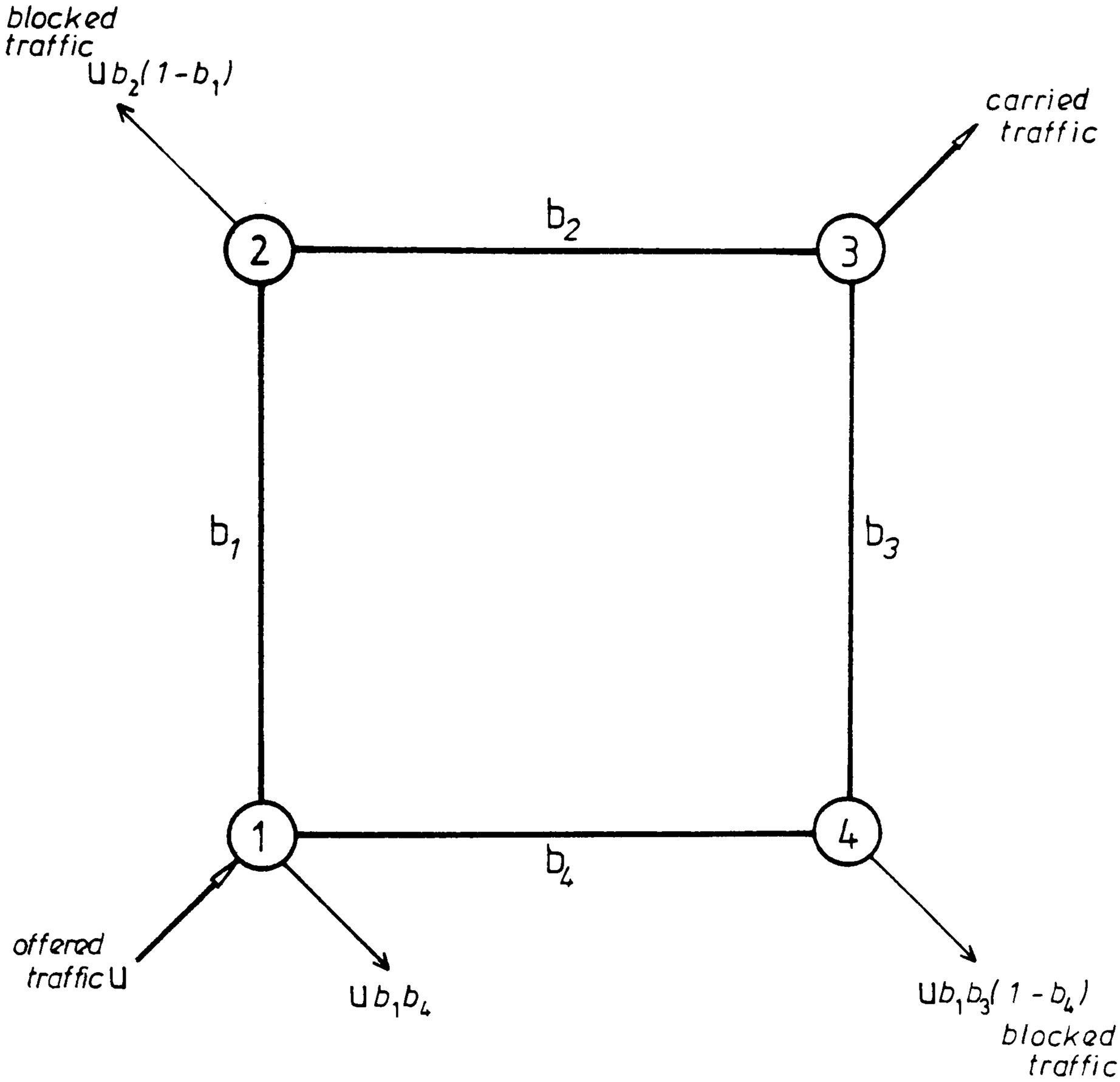


Figure 4.1 Four node network

$$R_{13} = [T_{12}, T_{14}]$$

$b_i \triangleq$ *blocking probability on* l_i



$$\text{offered traffic} = u$$

$$\text{blocked traffic} = u\left[b_1b_4 + b_1b_3(1-b_4) + b_2(1-b_1)\right]$$

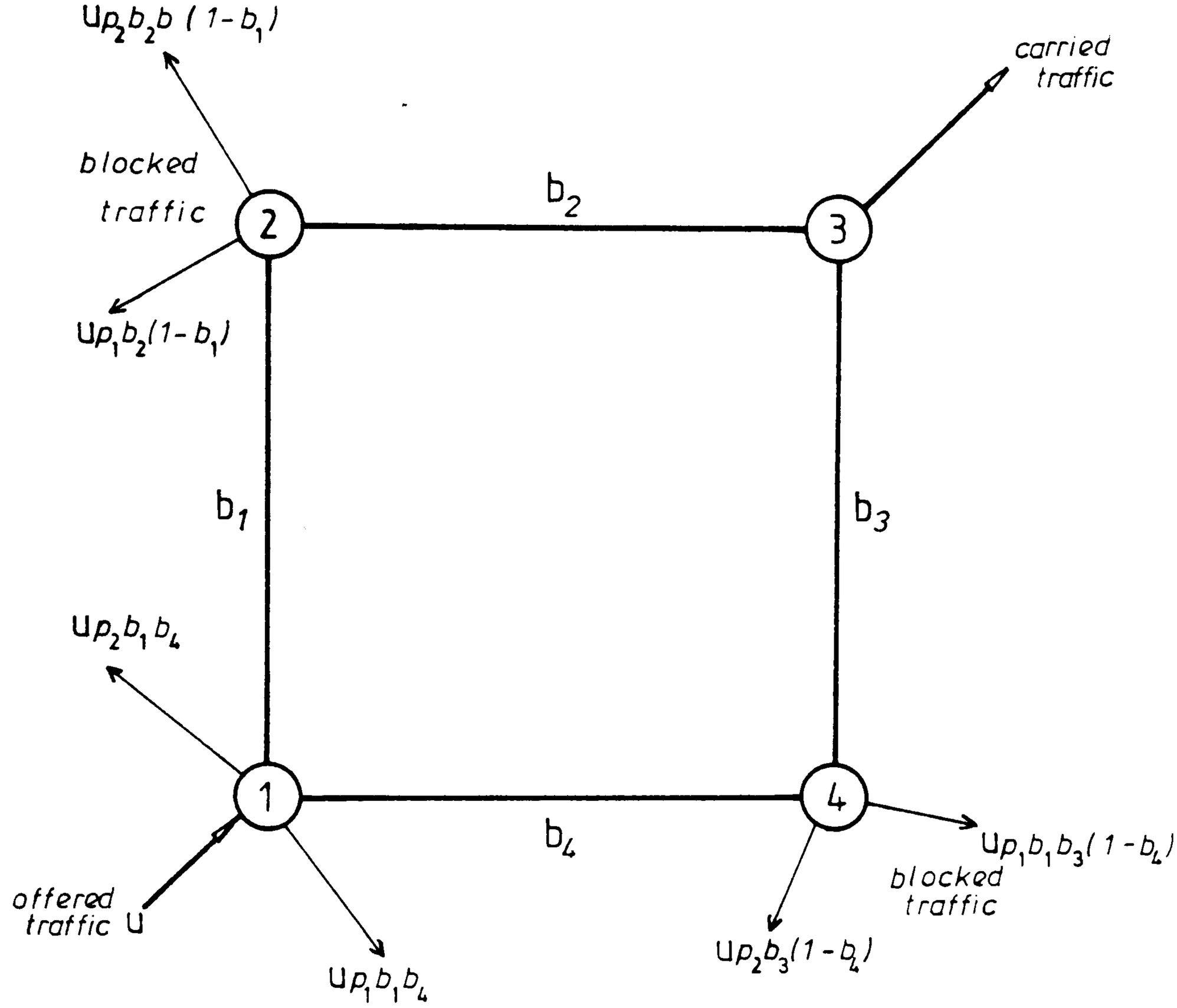
$$\text{carried traffic} = u\left[b_1(1-b_4)(1-b_3) + (1-b_1)(1-b_2)\right]$$

Figure 4.2 Traffic flow with alternate path routing

$$\alpha_1 = [T_{12}, T_{14}]$$

$$\alpha_2 = [T_{14}, T_{12}]$$

$b_i \triangleq$ *blocking probability on* l_i



$$\textit{offered traffic} = U$$

$$\textit{blocked traffic} = U\left[b_1b_4 + p_2b_3(1-b_4) + p_2b_2b_4(1-b_1) + p_1b_2(1-b_1) + p_1b_1b_3(1-b_4)\right]$$

$$\textit{carried traffic} = U\left[p_2(1-b_4)(1-b_3) + p_2b_4(1-b_1)(1-b_2) + p_1(1-b_1)(1-b_2) + p_1b_1(1-b_4)(1-b_3)\right]$$

Figure 4.3 Traffic flow with learning automata routing

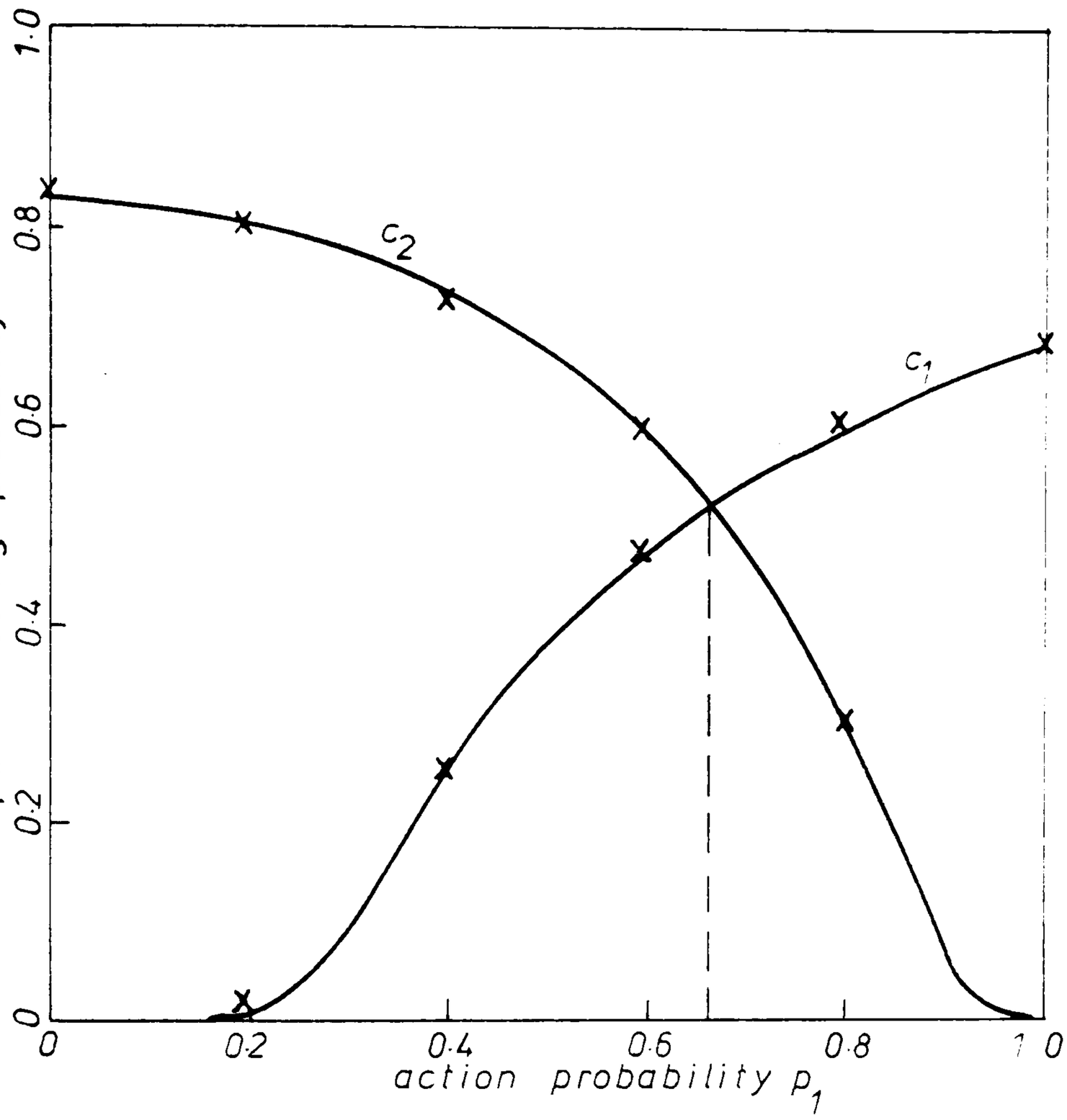


Figure 4.4 System identification results

[λ_{13} = 10 calls/min.]

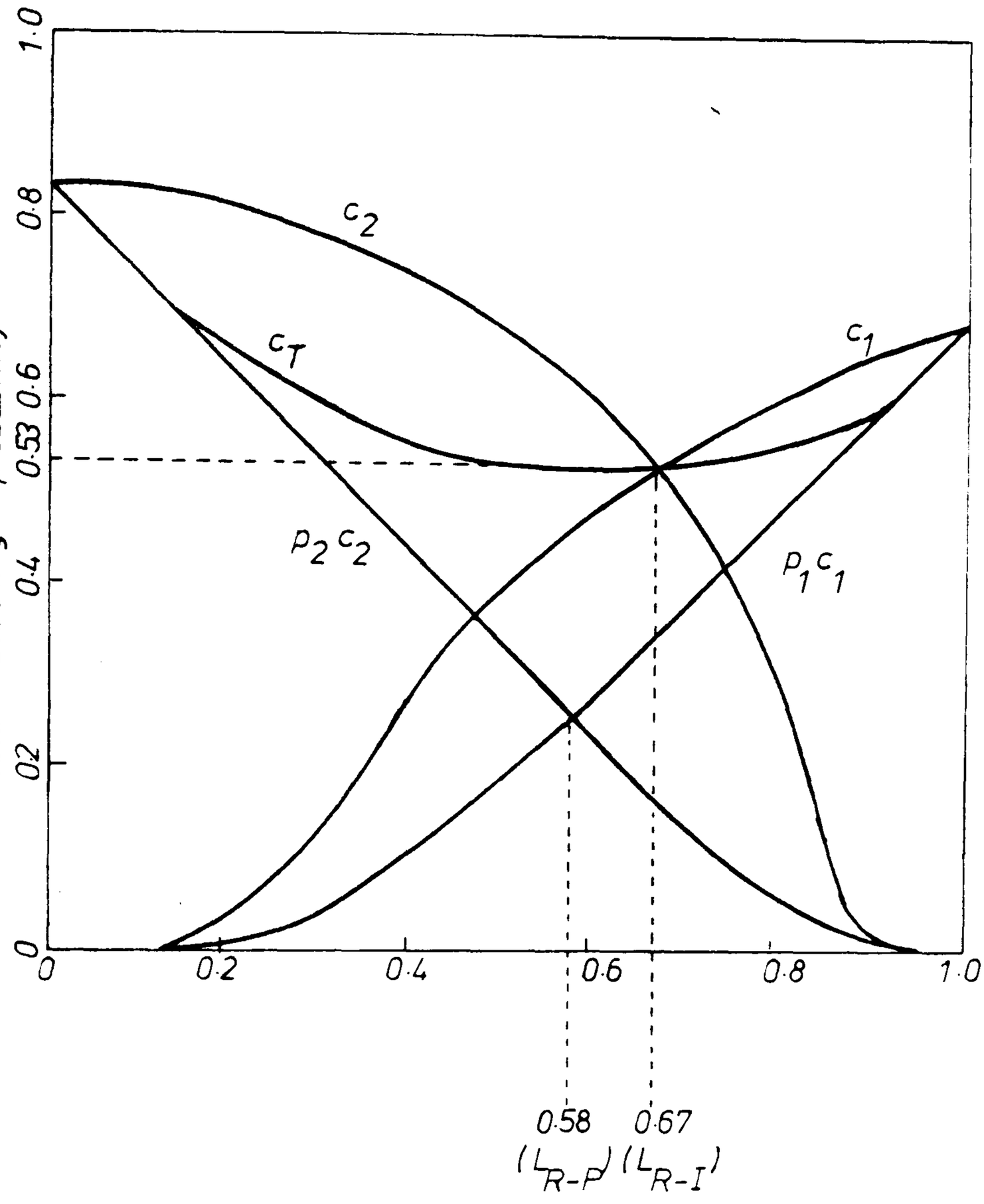


Figure 4.5 Convergence diagram for L_{RI} and L_{RP} schemes

[λ_{13} = 10 calls/min., p_1 (optimum) = 0.61]

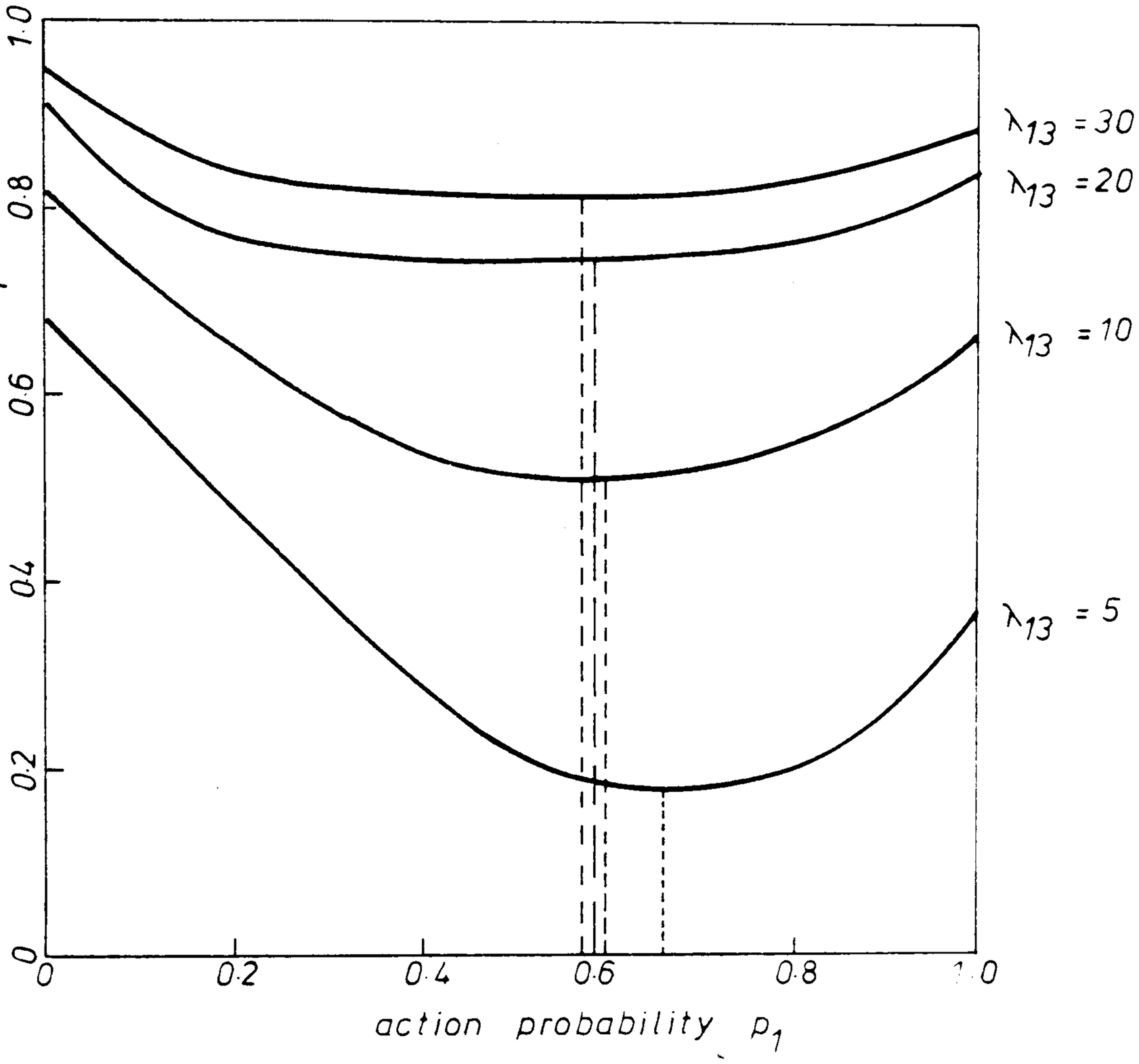
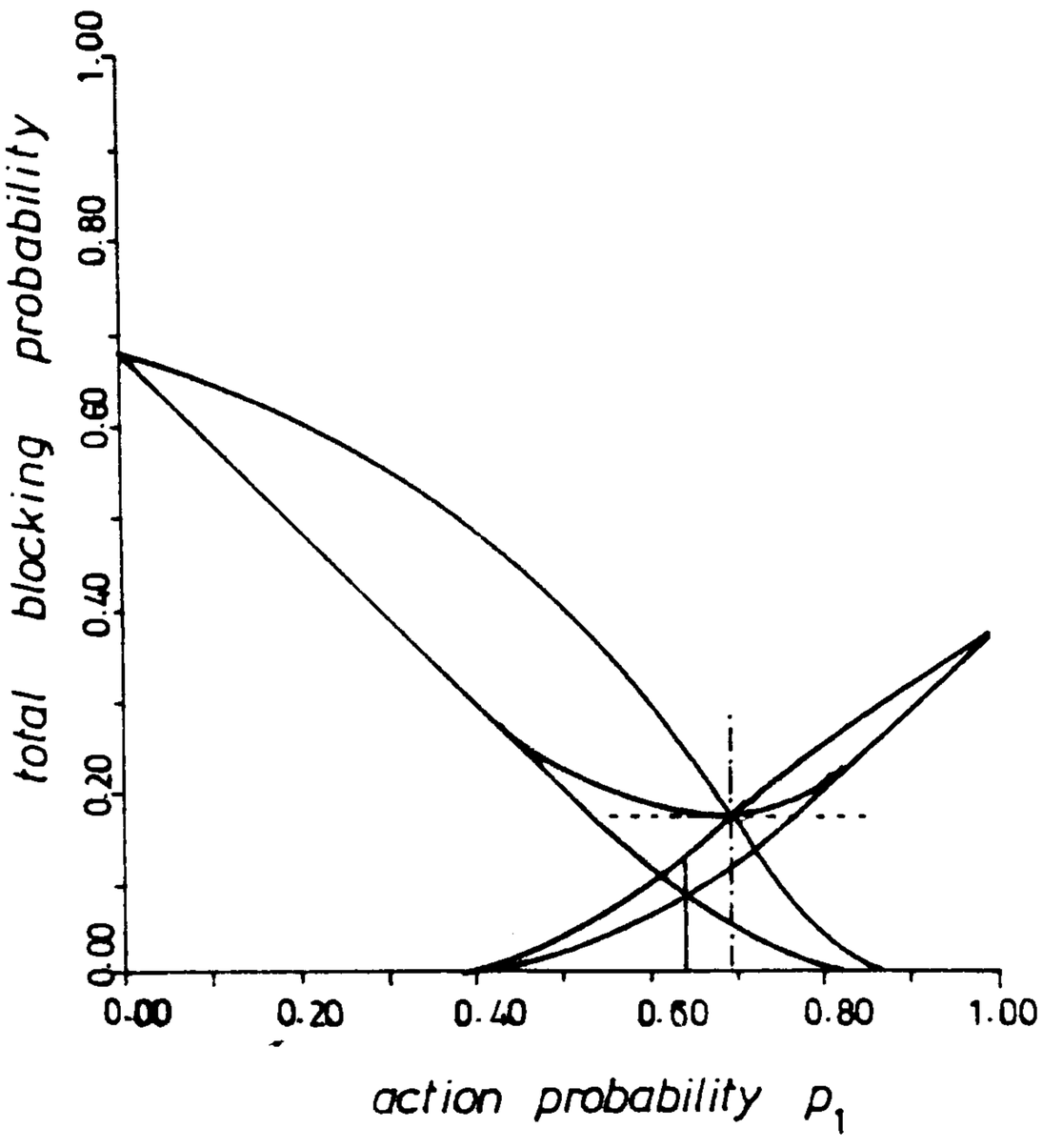
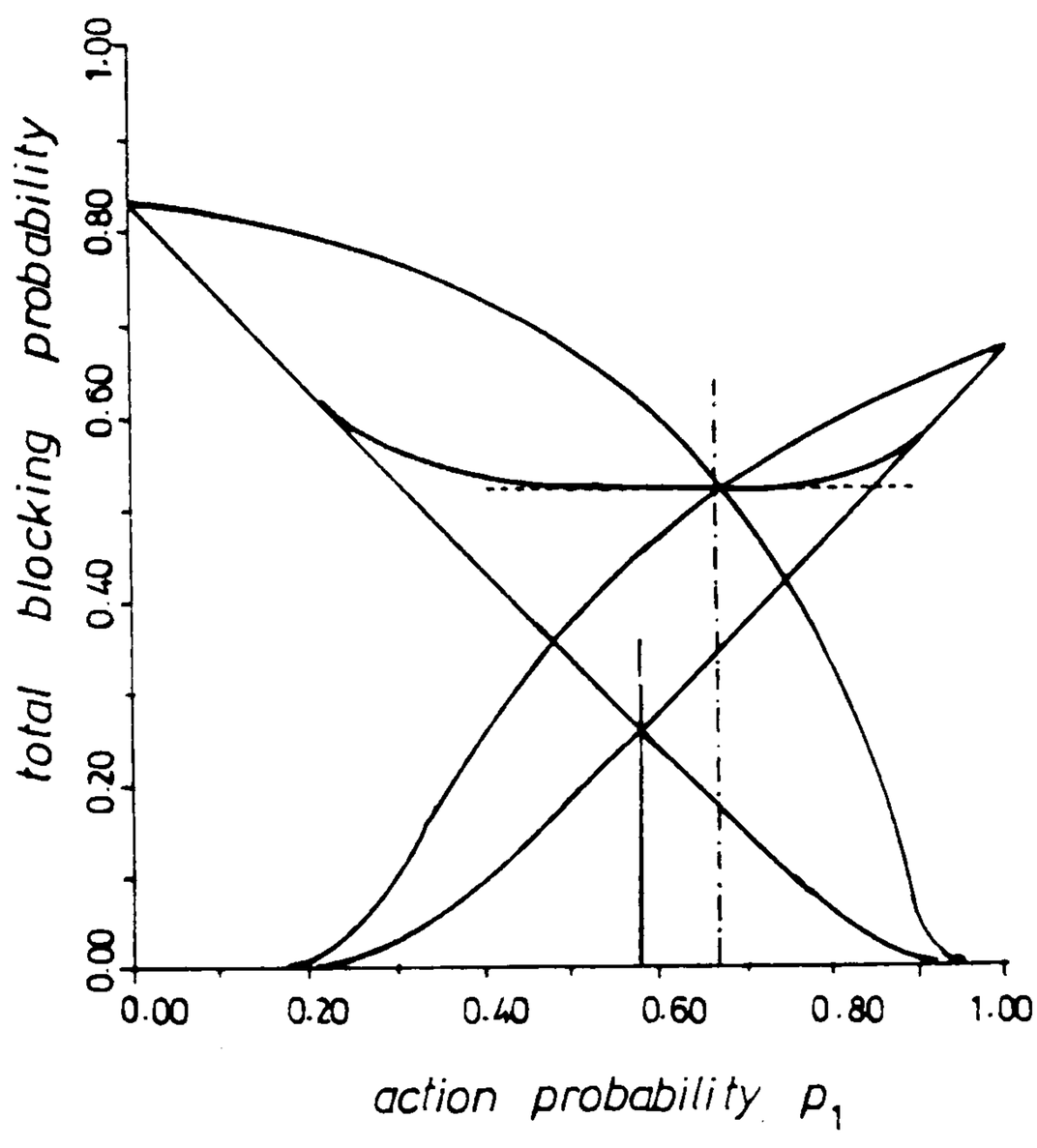


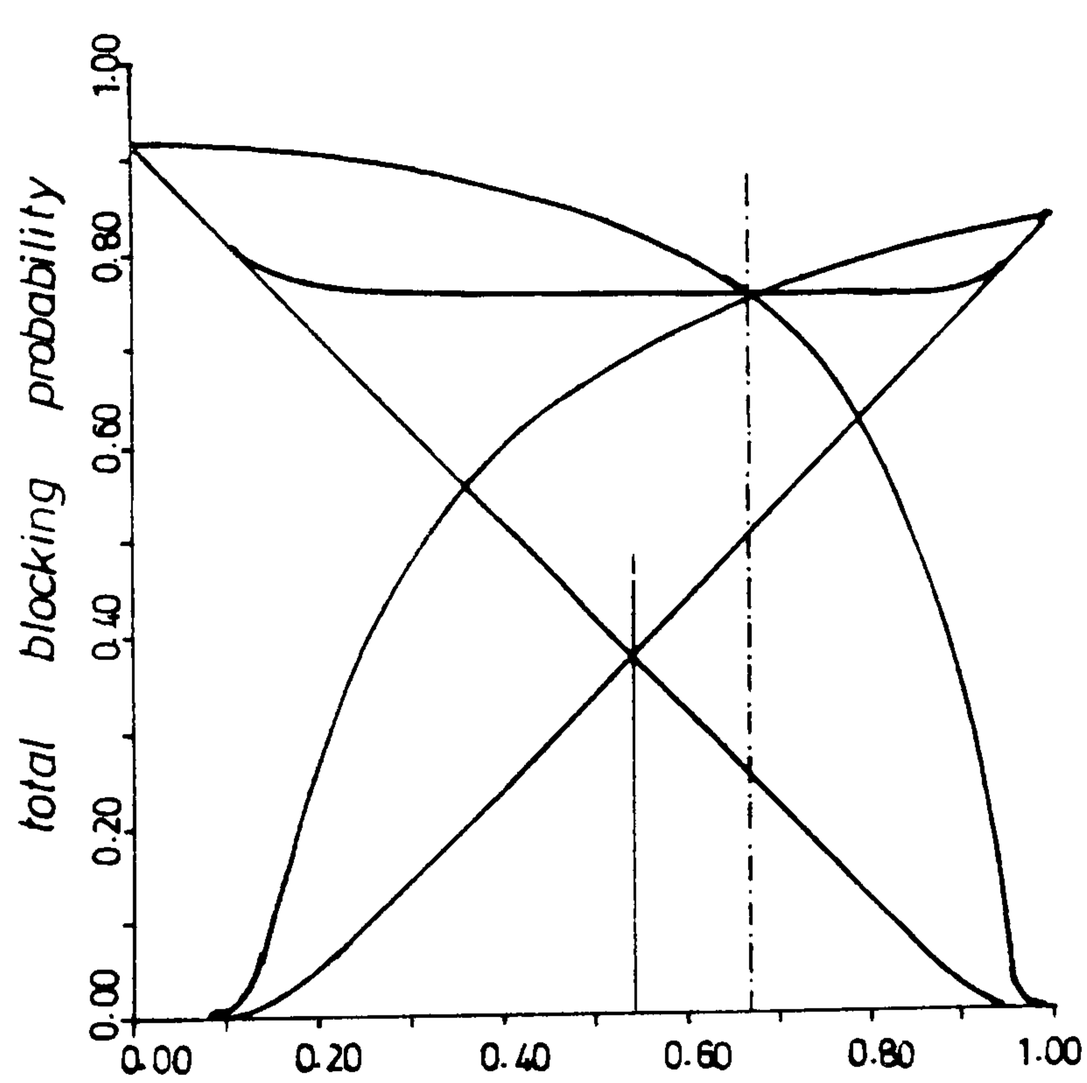
Figure 4.6 Total blocking probability graphs for varying λ



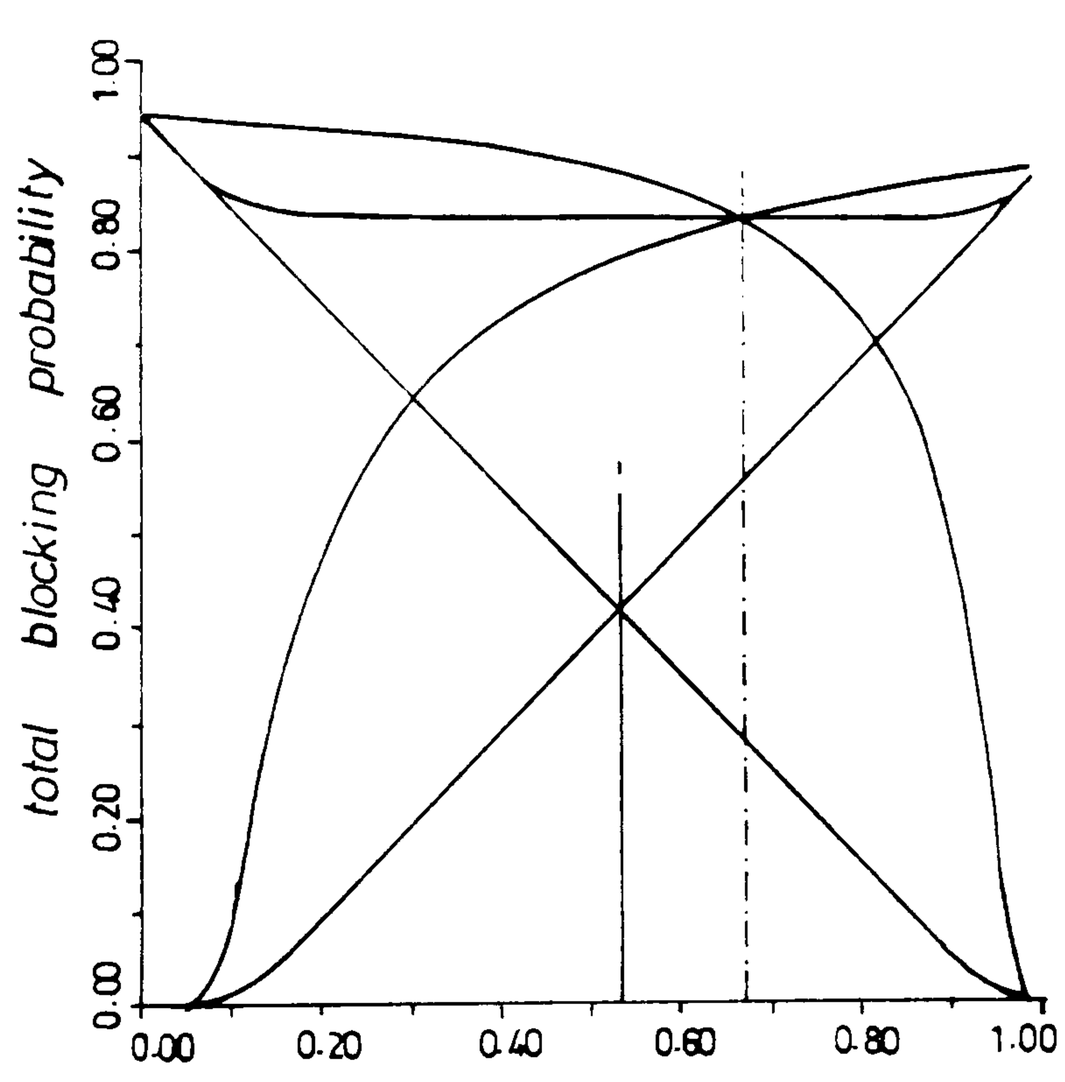
$\lambda = 5$ *calls/sec.*



$\lambda = 10$ *calls/sec.*



$\lambda = 20$ *calls/sec.*



$\lambda = 30$ *calls/sec.*

Figure 4.7 Influence of arrival rate

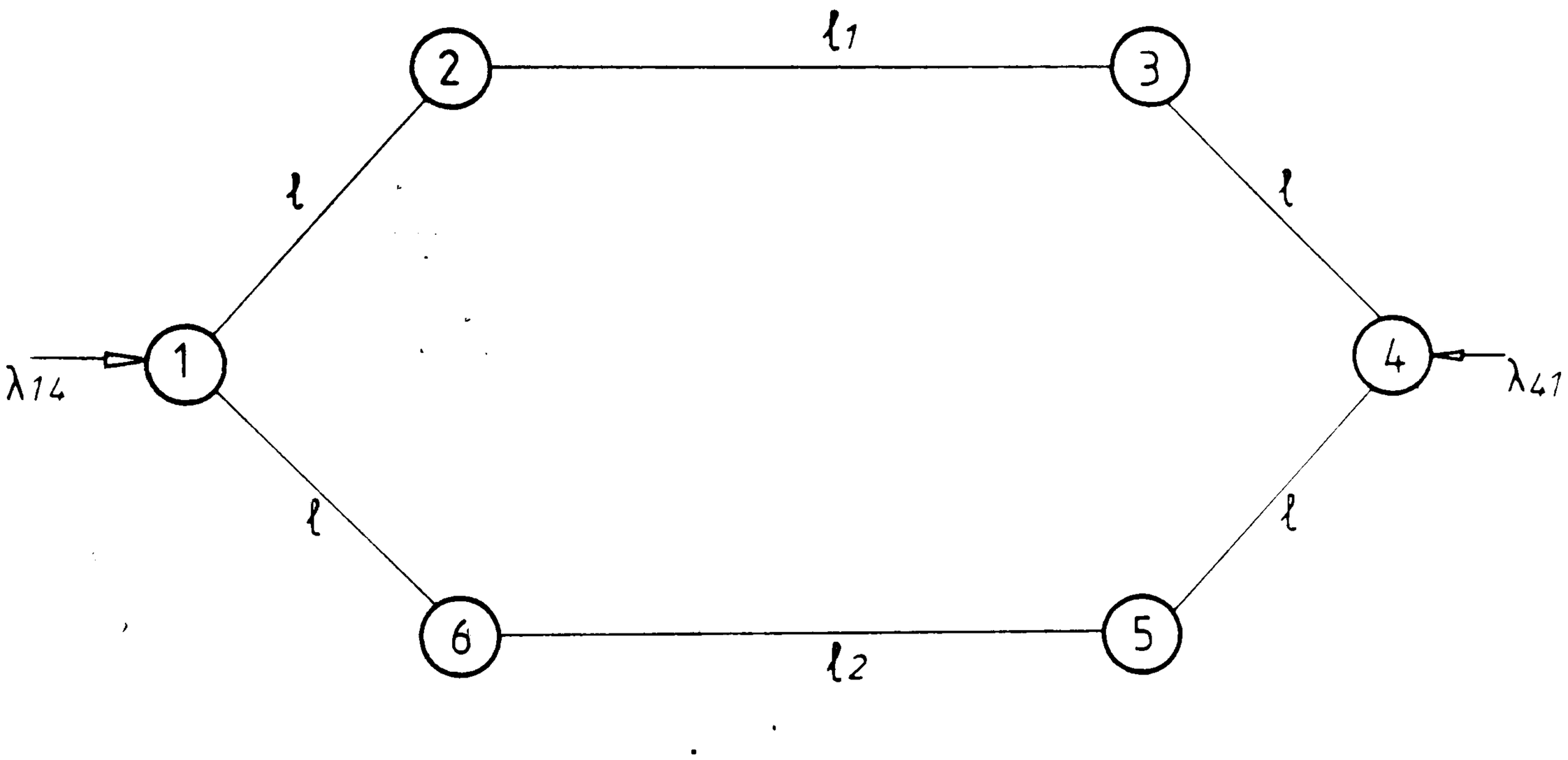


Figure 4.8 Six node network

CHAPTER 5

LEARNING AUTOMATA ROUTING IN HIERARCHICAL AND MESH STRUCTURED NETWORKS

5.1 Introduction

Previously we have considered the problem of learning automata routing in abstract models and simple networks. Specifically for the class of networks considered it was demonstrated that both the Linear Reward Inaction and Linear Reward Penalty schemes gave extremely close to optimal results. In this chapter, traffic routing in more complex hierarchical and generalised networks is considered using both alternate path and L_{R-I} automata routing schemes. The objective is to compare and demonstrate the operation of the adaptive routing in a more involved environment.

For these simulations, three networks have been utilised, including two hierarchical networks (8 and 12 nodes) based on the conventional telephone network and a general mesh structured network corresponding to the higher levels of the network hierarchy. In addition, experiments have been carried out to model the networks under fault conditions such as link and node failure by removing lines and injecting additional call sources at selected points in the network. These studies provide conclusive evidence to support the adaptive action of the learning scheme and categorically demonstrate the advantage over conventional fixed rules.

In the course of the experiments several desirable properties of learning automata routing are noted. As mentioned previously the ability to cope with adverse network conditions such as component failures and traffic overloads is observed. Also shown is the load balancing effect whereby the individual source blocking probabilities tend to equalise, suggesting a fairer allocation of network facilities. Finally, /

Finally, since the automata scheme employs such a simple feedback strategy (call routed successfully or blocked) the technique could be readily applied to existing networks with the minimum inconvenience.

5.2 Implementation of Adaptive Routing Schemes using Learning Automata

In the simple networks relatively few routing decisions were required for the successful operation of the network, since only two paths were open to calls. In larger networks this is no longer the rule, with many paths possible for a wide range of traffic sources. The implementation of learning automata schemes for such networks is now discussed with reference to both the call set up and feedback operations.

Routing Mechanism

The notion of a stochastic learning automaton forms an ideal building block for the implementation of de-centralised adaptive routing schemes for large networks. Under such circumstances, the establishment of a route R_{ij} for a call of class (i, j) (source node i, destination j) requires a chain of distributed routing decisions $\{A^i, A^a, \ldots\}$, each carried out by an independent automaton A^k (at node k) whose action set corresponds to the outgoing trunk groups of the current node $\{T_{ka}, T_{kb}, \ldots\}$. The resulting path therefore consists of an ordered collection of lines

$$R = \{l_{ia}, l_{ab} \ldots l_{gj}\}$$

In a typical network (Figure 5.1), the routing mechanism at at node k, A^k, is required to process calls in transit for the remaining N-1 nodes and as such must make a routing decision based on the proposed destination. For implementation /

implementation of this A^k normally would consist of N-1 automata structures with module A_j^k acting as a controller at node k for calls of destination j. The general form of the routing automaton may be summarised by an expanded description of the variable structure automaton.

$$A_j^k = \left\{ \alpha_j^k , \beta_j^k , p_j^k , T_j^k (\alpha, \beta, p) \right\}$$

$\underline{\alpha_j^k}$: The action set of the automaton as mentioned previously corresponds to the outgoing trunk groups of the current node. This set might specify any trunk leading to a neighbour of k (excluding the group which the call arrived on), or if a Markovian constraint is applied, any from a limited range. The Markovian constraint is applied to prevent the formation of routes which are cyclic, i.e., pass through a node more than once.

$\underline{\beta_j^k}$: The response to A_j^k results from the outcome of the routing operation, the P-model ideally suited to this application with a penalty signal indicating a blocked call and reward for a successful route establishment.

$\underline{p_j^k}$: The action probability vector of A_j^k contains the probabilities of selecting specific trunk groups.

$$p_j^{kr} = \left[\text{prob } T_{kr} \text{ is attempted} \right]$$

$T_j^k (\alpha, \beta, p)$: The reinforcement algorithm at A_j^k is responsible for modifying the vector $\underline{p_j^k}$ in accordance with the trunk selection and the received response.

In general, each automaton might operate under different algorithms, using a range of learning coefficients.

Flow Splitting

As a consequence of the distributed nature of this approach, many alternative paths are possible for each source /

source/destination pair. This ability to load split (Figure 5.2a) is a desirable property and the function of this scheme to proportion various degrees of traffic is highlighted by the probability P_{ij}^{s} that some path s is used by i, j calls is

$$P_{ij}^{s} = \prod_{k \epsilon S} p_j^{kr}$$

where S is the set of nodes on path s.

This expression further demonstrates the ability of the distributed automata to influence the global routing strategy of the scheme, the overall probability of selecting the various paths evolving as individual automata receive feedback from the network.

Feedback Mechanism

When the call propagation is halted, the call either reaching the destination or becoming stranded at an intermediate node due to no available lines, all automata linked in the chain are up-dated. As in the simple 4 node network, a binary response is employed, a successful routing operation resulting in a reward and a blocked route producing a penalty.

R_{ij} completed $\quad \beta_j^k = 0 \quad k \epsilon S$

R_{ij} blocked $\quad \beta_j^k = 1$

Figure 5.1 displays typical routing operations in a network, showing attempts to route calls of class (i, j) and (i, g). In the first example, a call at i, requiring a path to j, is progressed to node m which in turn moves the call to n and finally to j, thus completing a successful operation and consequently A_j^i, A_j^n and A_j^m receive reward signals ($\beta_j^i = \beta_j^n = \beta_j^m = 0$). The call of class (i, g) on the otherhand, although successfully routed to node f, cannot complete the desired path due to no available lines from f to the destination. The /

The up-dating mechanism therefore punishes all automata in the chain, i.e., A_g^e and A_g^i.

In this global feedback arrangement automata experience penalty/reward responses in relation to the blocking probability over the paths available from the present node to the destination. For the automaton A_j^k, the probability of receiving a penalty for action r is given by

$$c_j^{kr} = \sum_{s \in Q} P_{kj}^s \, B_{kj}^s$$

given that Q is the set of feasible paths resulting from α_j^{kr} and B_{kj}^s is the blocking probability on path s between k and j.

$B_{kj}^s = b_{k\alpha} + (1-b_{k\alpha}) b_{\alpha b} + (1-b_{kb}) b_{bc} + \ldots$ b_{fg} is the blocking probability on T_{fg}

Global Behaviour of the L_{R-I} Scheme

With the L_{R-I} reinforcement algorithm implemented for all automata, the equilibrium behaviour at node k may be described by

$$E\left[c_j^{k\alpha}\right] = E\left[c_j^{kb}\right] \qquad a, b = 1, 2 \ldots r$$

This suggests that each automaton in the network attempts to equalise the blocking probabilities resulting from the alternative actions at the current node. Furthermore, since every automaton is concerned with equalising the blocking probability between the destination and the present stage, the global effect is an equalisation of the total path blocking probabilities (Figure 5.2b).

$$E\left[B_{ij}^a\right] = E\left[B_{ij}^b\right]$$

$a, b \in V$, V is the set of feasible paths between i and j.

Global Behaviour of the L_{R-P} Scheme /

Global Behaviour of the L_{R-P} Scheme

Under similar circumstances the L_{R-P} scheme operates to balance the penalty rates at each stage (Figure 5.2c) in the network. For A_j^k convergence is such that

$$E\left[p_j^{ka} c_j^{ka}\right] = E\left[p_j^{kb} c_j^{kb}\right] \qquad a, b = 1, 2 \ldots r$$

Self-tuning Adaptive Routing

As mentioned previously, two possible operating strategies may be considered for learning automata routing. By invoking a Markovian constraint on the allowable range of actions at each automaton only routes which are loop-free are formed. Therefore all routing will be carried out on a choice of pre-determined routes, the individual automata acting to optimally proportion traffic to the various options. Since the automata can only perform within the constraints of the allowable paths the scheme may be best regarded as a self-tuning routing scheme.

Self-Organising Adaptive Routing

An alternative strategy is to relax any restrictions on the action sets of individual automata, permitting the selection of any trunk group, but excluding the trunk used to arrive on. Operation of such a scheme can be described as self-organising, the routing mechanism having complete freedom to seek out the optimal paths in the network, eventually converging to exclude actions which result in loops and exceedingly long paths. In routing schemes of this type the feedback of a path length parameter is a desirable and necessary feature, the response to the automata resulting from the outcome of the call and the number of lines accululated. In the experiments that follow, the self-tuning technique is demonstrated although in the next chapter the self-organising scheme /

scheme is shown in operation under normal and adverse network conditions.

Practical Considerations

Prior to implementing the learning automata scheme in a real network certain aspects concerning the nature of the learning algorithms and mechanisms for performing alternative actions must be considered.

Learning Coefficients

The choice of learning algorithm parameters a and b in general represents a trade off in speed and accuracy, coarse values producing fast convergence at the expense of high action probability variance and consequently inaccurate steady state behaviour. In constrast, automata with small values give good equilibrium results but lack the response to adapt with fast changes in the network. Moreover, to achieve a truly optimum performance, it might be necessary to allocate different parameters to the various automata in the network, although as of yet this problem has not been studied.

Maximum Path Length

In order to prevent excessive call lengths, a maximum path length constraint is a helpful addition to a practical routing scheme. This may be a necessary physical constraint which is imposed by signal losses in the network, for instance to maintain a specified signal to noise ratio.

Repeated Routing Attempts

In the experiments with the small networks an action of the automata was designated to the sequence which the allowable paths were attempted. In the case of two paths this proved an entirely satisfactory arrangement, a 2 action automaton /

automaton providing the necessary number of actions. However for N paths a N! action automaton is required to cover all combinations of trunk selection, which unfortunately proves unsuitable for networks where several options exist at a node.

A suitable modification on the original scheme is to allow individual actions to correspond to a specific trunk group. If the first choice has failed due to no lines then the remaining trunks are again considered, after temporarily removing the failed entry and re-normalising the action probabilities. This operation repeats until either the call is successfully routed or dropped from the network.

5.3 8 Node Hierarchical Network

The network used in the following sequence of experiments is shown along with the associated call sources on Figure 5.3. The design is based on the logical paths available between two end users in different regions of a conventional hierarchical structured telephone network. Further, the implementation of both routing schemes, alternate path and learning automata, are governed by the previously described routing limitations and as such promote the realism of the experiments. The results presented in this section also include a group of experiments modelling the network under fault conditions and as is shown, the adaptive property proves a valuable asset under these circumstances.

For the adaptive schemes the L_{R-I} reinforcement algorithm provides the up-dating function, using learning parameters a = 0.02, b = 0.0. Also, the experiments were run using the PDP 11 simulation package[34], since at this stage in the project, the mainframe simulation software had not been fully prepared. Duplicate experiments with the /

the 8 node network were later used to confirm the correct operation of the second version.

Experiment 1 Normal Network Conditions

In this experiment, the network is allowed to operate under engineered loads and with no induced faults.
By carrying out a max-flow analysis on the network it was found that for both the fixed and adaptive schemes the effective number of lines from 1 to 8 is 110. Consultation with the tables in[32] shows that for this capacity blocking probabilities in the region of 0.11 to 0.138 might be obtained.

In all, two traffic conditions are attempted, a single source λ_{18} = 20 calls/min and a dual traffic set $\lambda_{18} = \lambda_{81}$ = 10 calls/min. For these simulations, the following results were obtained.

λ_{18} = 20 calls/min				
Fixed Rule (L_{R-I} Scheme)		500 Calls	1000 Calls	1500 Calls
Overall B P		.06(.088)	.09(.142)	.115(.125)
Calls Blocked	Node 1	30(40)	91(104)	173(147)
	7	-(-)	-(38)	-(41)
$\lambda_{18} = \lambda_{81}$ = 10 calls/min				
Overall B P		.088(.054)	.094(.079)	.111(.086)
B P λ_{18}		.083(.052)	.108(.069)	.116(.073)
B P λ_{81}		.093(.056)	.081(.088)	.106(.099)
Calls Blocked	Node 1	18(-)	40(7)	66(11)
	2	5(7)	9(37)	20(68)
	7	3(13)	13(28)	21(43)
	8	18(7)	32(7)	60(7)

With the results obtained from this series of simulations it can clearly be noted that the learning schemes converge to give comparable results with the optimal fixed rule.
Although the blocking probabilities have not been altered in any appreciable manner, it is interesting to note the shift in congestion /

congestion patterns, i.e., the various schemes tending to have different numbers of blocked calls at nodes 1, 2, 7 and 8. Furthermore, the blocking probability obtained compares favourably with the predictions for 110 trunks.

Although only a small cross section of a large and complex network has been simulated, the experiments described provides a valuable insight into the operation of the learning automata routing in a large network.

Experiment 2 Fault on Trunk T_{78}

By failing trunk group T_{78} i.e., $l_{78} = 0$, it can be shown that the fixed rule encounters severe problems while routing calls from 1 to 8. This behaviour is a direct consequence of the local routing policy of the alternate path scheme, since once passed on from a node the call cannot attempt any previous options when blocking is met at a later stage. Further analysis reveals that the optimal fixed rule for λ_{18} can only access 30 lines whereas the adaptive scheme, after converging, can possibly utilise 80. Again consulting with the blocking probability tables shows the expected values for blocking to be 0.75 for the fixed rule and 0.35 for the automata. For this experiment 3 simulation runs are described, with $\lambda_{18} = 20$ calls/min, $\lambda_{81} = 20$ calls/min and $\lambda_{18} = \lambda_{81} = 10$ calls/min.

$\lambda_{18} = 20$ calls/min

Fixed Rule (L_{R-I} Scheme)	500 Calls	1 000 Calls	1 500 Calls
Overall B P	.702(.344)	.733(.353)	.738(.355)
Calls Blocked Node 7	351(172)	733(353)	1 107(553)

$\lambda_{81} = 20$ calls/min

Overall B P	.458(.43)	.483(.476)	.471(.48)
Calls Blocked Node 8	229(215)	483(476)	706(720)

$\lambda_{18} = \lambda_{81} = 10$ calls/min /

$\lambda_{18} = \lambda_{81} = 10$ calls/min		500 Calls	1 000 Calls	1 500 Calls
Overall B P		.46(.386)	.494(.377)	.51(.39)
B P λ_{18}		.685(.428)	.693(.39)	.709(.353)
B P λ_{81}		.238(.284)	.284(.363)	.307(.426)
Calls Blocked Node	2	- (-)	- (-)	- (7)
	7	170(110)	356(201)	537(263)
	8	60(69)	138(176)	228(315)

From the significant change in the blocking probabilities, it can be seen that T_{78} is an extremely important part of the network. As predicted, the fixed rule results show a large increase in the blocking probability for λ_{18} calls, this due to the schemes inability to access more than 30 lines. In contrast, the learning scheme, with the power to attempt call routing over extra paths makes full use of the 80 lines. The results for $\lambda_{81} = 20$ calls/min illustrates the influence of T_{78} on λ_{81} calls which for both routing schemes has a similar effect and consequently both schemes give similar results. Finally, it is interesting to note the results obtained from the dual source simulation and to study the individual source blocking characteristics. Prior to the automata converging the difference in blocking $(\lambda_{18} - \lambda_{81})$ is noted as $\approx$ 0.14. However, on convergence the automata show load balancing properties which reduce this to $\approx$ 0.07, the automata thus giving a fairer allocation strategy.

Experiment 3 Fault on Node 7

In this experiment, node 7 is "eliminated" by removing trunk groups T_{17}, T_{37}, T_{67} and T_{78}. Again carrying out the max-flow analysis we find that for λ_{18} calls the fixed rule can utilise 70 trunks with the learning scheme converging on 80. Under these conditions once more the learning scheme will out-perform the fixed rule although the difference will be substantially reduced as compared with /

with the previous experiment.

λ_{18} = 20 calls/min				
Fixed Rule (L_{R-I} Scheme)		500 Calls	1000 Calls	1500 Calls
Overall B P		.38(.322)	.395(.324)	.401(.339)
Calls Blocked Node	1	- (77)	- (198)	- (369)
	2	190(84)	395(136)	602(140)
λ_{81} = 20 calls/min				
Overall B P		.428(.43)	.458(.48)	.493(.467)
Calls Blocked Node	8	214(215)	458(480)	739(706)
$\lambda_{18} = \lambda_{81}$ = 10 calls/min				
Overall B P		.36(.3)	.417(.29)	.434(.307)
B P λ_{18}		.245(.239)	.321(.222)	.349(.225)
B P λ_{81}		.467(.36)	.505(.359)	.519(.394)
Calls Blocked Node	1	- (4)	- (20)	- (63)
	2	- (1)	- (5)	- (17)
	6	59(55)	156(96)	261(111)
	8	121(90)	121(167)	390(269)

Although node 7 was completely removed from the network, the system blocking probabilities were lower than those obtained in the previous experiment. This can be explained as follows. With node 7 completely out, calls are pushed further up the hierarchy where they can tap the capacity of the direct links from 5 and 8. With T_{78} alone in a fault condition calls get trapped at node 7 and because of the lack of feedback no other paths are attempted. As in the previous experiment the results for the fixed rule and learning automata are comparable for λ_{81} = 20 calls/min. This condition allows both schemes access to 60 lines, thus leading to blocking probabilities in the region of 0.46.

Experiment 4 Focused Overload Conditions

In order to highlight the problems of assymmetric traffic and focused overloads, an additional call source is added to node 7 (λ_{78}). As expected, and confirmed by the results, the /

the introduction of such conditions are a major influence on the performance of the routing scheme. In this particular example the result of the overload is a reduction in the effective number of lines available to calls of λ_{18} and λ_{81} with the subsequent increase in blocking probability.

		λ_{78} = 10 calls/min		
λ_{18} = 20 calls/min				
Fixed Rule (L_{R-I} Scheme)		500 Calls	1 000 Calls	1500 Calls
Overall B P		.314(.242)	.345(.278)	.401(.291)
B P λ_{18}		.251(.177)	.27(.186)	.312(.195)
B P λ_{78}		.434(.371)	.491(.456)	.477(.472)
Calls Blocked Node	7	157(121)	345(278)	602(436)
λ_{81} = 20 calls/min				
Overall B P		.256(.296)	.344(.314)	.379(.356)
B P λ_{81}		.192(.182)	.276(.197)	.308(.256)
B P λ_{78}		.378(.572)	.471(.54)	.516(.542)
Calls Blocked Node	7	65(87	164(184)	266(285)
	8	63(61)	180(136)	303(249)
λ_{81} = λ_{18} = 10 calls/min				
Overall B P		.302(.26)	.367(.302)	.381(.305)
B P λ_{18}		.301(.257)	.377(.251)	.378(.24)
B P λ_{81}		.072(.086)	.196(.162)	.192(.167)
B P λ_{78}		.484(.426)	.52(.492)	.558(.499)
Calls Blocked Node	2	- (-)	- (3)	- (16)
	7	140(117)	303(250)	480(377)
	8	11(13)	64(49)	92(64)

Again the adaptive routing scheme proves a worthwhile service in the network which although in a failure-free condition suffers performance degradation through a traffic asymmetry in the form of a focused overload. For instance, the λ_{18} = 20 calls/min case, when routed by a fixed rule gave an overall blocking probability of 0.401 compared to the 0.291 of the learning scheme.

To summarise the results from this last series of experiments it is useful to construct a table showing the overall /

overall blocking probabilities for the different network conditions.

	$\lambda_{18} = 20$	$\lambda_{81} = 20$	$\lambda_{18} = \lambda_{81} = 10$	
Normal Conditions	.115 .125		.111 .086	Fixed Learning
Trunk T_{78} Failed	.738 .355	.471 .48	.51 .39	
Node 7 Failed	.401 .339	.493 .467	.434 .307	
Focused Overload λ_{78}	.401 .291	.379 .356	.381 .305	

Several salient features can be observed from this table regarding the comparative studies between the fixed and learning routing schemes. In the first row the results for the network operating under normal conditions is presented, indicating the equal ability of the learning scheme and the optimal fixed rule. Next the influence of a failed T_{78} on calls from λ_{18} is shown, in this case the learning scheme clearly providing a better routing service. Similar conditions are again noted for the adverse conditions caused by the failure of node 7 and the injection of additional calls at node 7, the learning scheme operating in such a way as to tap network capacity unused by the alternate path scheme with the distinctive improvement in service.

5.4 12 Node Hierarchical Network

A network similar to that used in the previous experiments is expanded to 12 nodes and is as shown in Figure 5.4. Once more experiments using the conventional fixed /

fixed rule and a L_{R-I} learning scheme are presented, the network operating under both normal and fault conditions. Further analysis of the network leads to the conclusion that T_{78} is still the most sensitive component for failure and as a result gives the worst case conditions.

Experiment 1 Normal Network Conditions

In this experiment the network is operated under normal conditions using engineered loads and fault-free components. For all call sources set to 10 calls/min the following results were obtained.

Calls	500	1 000	1 500	2 000
B P Overall	.162(.052)	.206(.159)	.245(.189)	.25(.21)
B P λ_{18}	.235(.079)	.280(.212)	.321(.265)	.345(.28)
B P λ_{81}	.088(.017)	.140(.122)	.15(.13)	.152(.167)
B P $\lambda_{8\,12}$	.084(0)	.095(.083)	.131(.099)	.133(.142)
B P $\lambda_{12\,8}$	.235(.109)	.304(.216)	.368(.259)	.368(.25)
Blocked Calls				
Node 2	-(2)	-(11)	-(20)	-(37)
7	60(24)	149(109)	265(198)	359(266)
8	21(-)	57(37)	103(59)	141(101)
11	-(-)	-(2)	-(6)	-(16)

As in previous experiments for networks operating under normal conditions the learning scheme is seen to produce comparable results with the optimal fixed rule. It should be noted that although both routing schemes produced similar overall blocking probabilities, the L_{R-I} scheme manages to provide a better service distribution for the individual sources, again suggesting a fairer allocation strategy by the automata. This fact is also brought out in the congestion distribution in the network with the fixed scheme causing blocking at only two nodes and the learning scheme spreading the blocked calls over four.

Experiment 2 Fault on Trunk T_{78} /

Experiment 2 Fault on Trunk T_{78}

As in a previous experiment a network fault is introduced by open circuiting trunk group T_{78}, effectively rendering the fixed rule sub-optimal for two of the sources (λ_{128} and λ_{18}). The following results were obtained for both fixed and learning routing with all sources generating 10 calls/min.

Calls	500	1 000	1 500	2 000
B P Overall	.436(.33)	.472(.358)	.479(.39)	.496(.393)
B P λ_{18}	.646(.487)	.707(.455)	.715(.458)	.723(.417)
B P λ_{81}	.029(.263)	.196(.337)	.231(.359)	.247(.357)
B P $\lambda_{8\,12}$	.206(.098)	.223(.173)	.192(.247)	.243(.295)
B P $\lambda_{12\,8}$	.765(.485)	.761(.463)	.774(.496)	.67(.501)
Blocked Calls				
Node 2	188(117)	367(226)	560(354)	747(456)
7	30(48)	105(132)	158(231)	245(329)

As expected the learning automata scheme is able to make the most of the unused network capacity to produce lower blocking probabilities than the fixed rule. Once more the influence on the service distribution is noted, the adaptive scheme clearly attempting to balance the grade of service to individual users.

From the results of both experiments it is possible to plot graphs showing the blocking probability against the number of calls handled. On Figure 5.5 four overall blocking probability graphs have been drawn.

1 Fixed rule under normal conditions

2 Learning scheme under normal conditions

3 Fixed rule under fault conditions ($T_{78} = 0$)

4 Learning scheme under fault conditions

In accordance with prior results, the fixed rule and learning schemes operating under normal conditions show virtually /

virtually identical steady state behaviours, proving the ability of the adaptive routing technique to match the performance of an optimal alternate path scheme. With group T_{78} removed, the learning scheme is shown to outperform the fixed rule which as suggested previously is no longer optimal. It should however be noted that initially (first 200 calls) the learning scheme gives higher blocking probabilities than the fixed rule. This can be attributed to the transient learning phase of the automata, primarly attempting all routing alternatives to finally converge to the optimal routing strategy.

5.5 10 Node Mesh Network

In the upper levels of the network hierarchy (regional centres) and in small private networks the overall structure of the net may be regarded as a mesh. To ensure that networks of this variety are covered an additional set of experiments have been included to study the use of learning automata routing schemes in this case.

Experiment 1 Normal Operating Conditions

Following previous experimental procedure, a lower bound to the network performance is found by staging a simulation with fault-free components under engineered loads. Considering one call source $\lambda_{17} = 10$ calls/min the following results were obtained.

Fixed Rule (L_{R-I} Scheme)	500	1 000	1500	2 000
Overall BP	.314(.178)	.315(.148)	.291(.137)	.298(.131)
Blocked Calls				
Node 2	- (88)	- (147)	- (204)	- (238)
3	157(-)	315(-)	436(-)	596(-)
8	- (1)	- (1)	- (1)	- (1)

By /

By intention, no attempt in preparing this experiment was made to produce an optimal fixed rule to purposefully show the automata converging to the desired optimum strategy. Further, an analysis of the network reveals that for calls from λ_{17} 45 lines are made available by the fixed rule with the learning automata converging on 60. The results obtained confirm this with the overall blocking probability measured as 0.298 for the fixed rule as opposed to 0.131 for the learning scheme.

Experiment 2 Fault on Trunk Group T_{67}

For this experiment the previous mesh structured network and traffic conditions was used, but in order to simulate a failure T_{67} was set to contain no lines. Again a max-flow analysis of the network indicates that the fixed rule is sub-optimal, accessing only 20 lines in comparison to the adaptive scheme converging on 55.

Calls	500	1 000	1 500	2 000
Overall B P	.7(.264)	.682(.272)	.698(.275)	.68(.269)
Blocked Calls				
Node 2	- (82)	- (160)	- (249)	- (313)
6	350(28)	682(72)	1 047(115)	1 360(127)
8	- (22)	- (40)	- (48)	- (98)

With T_{67} in a failed condition calls destined for node 7 tend to be blocked at node 6 when routed by the fixed rule. The learning scheme on the otherhand utilises the additional capacity offered by nodes 4, 5, 9 and 10 and as a result produces a significantly lower blocking probability.

Experiment 3 Focused Overload by λ_{67}

A traffic asymmetry is caused by injecting an additional source of calls at node 6, effectively eroding the available lines for the prime source. For the experiment both sources /

sources provide a rate of 10 calls/min.

Calls	500	1 000	1 500	2 000
Overall B P	.786(.582)	.808(.602)	.819(.581)	.817(.577)
B P λ_{17}	.762(.449)	.778(.455)	.801(.414)	.792(.413)
B P λ_{67}	.81(.73)	.837(.755)	.838(.748)	.842(.741)
Calls Blocked				
Node 2	- (38)	- (121)	- (175)	- (262)
6	393(216)	808(444)	1 229(659)	1 634(856)
8	- (37)	- (37)	- (37)	- (37)

As suggested by previous simulation experiments, the learning scheme produces a better performance by utilising the extra capacity in the network. In comparison, the alternate path scheme operating on a purely local routing policy cannot attempt the additional routes, this reflected in the higher overall blocking probabilities.

To supplement the numerical data, graphs of the blocking probabilities were drawn as a function of calls in, attempting to show the transient behaviour of the learning schemes. The fixed rule, as expected, allows the network to fill up in an orderly manner, heavy traffic overflowing into the secondary routes as desired. The learning scheme on the otherhand exhibits a "learning phase" with the blocking probability initially rising above the steady state value to drop to the final condition. Figure 5.7 displays the overall blocking probability graphs for normal and T_{67} failed conditions with Figure 5.8 showing the individual blocking probabilities for λ_{17} and λ_{67} for the various routing schemes.

5.6 Conclusions

By carrying out a set of experiments on both hierarchical and mesh structured networks, this study has endeavoured /

has endeavoured to cover all topographical aspects of the modern telephone system, proving the learning automata technique to be a suitable contender for adaptive routing. Even in the hierarchy where routing operations are suitably constrained by a set of governing rules the problems of focused overloads and component failures can cause significant performance degredation. As demonstrated by the experiments the use of an adaptive scheme provides a more efficient means of allocating network facilities which as shown must be performed using global knowledge of the network.

From this chapter, several major conclusions can be drawn on the nature and performance of the learning automata schemes. In addition further evidence has been produced to support conclusions brought forward in the previous chapter.

1 Under normal conditions the learning scheme, operating in steady state, can match the routing performance of the optimal fixed rule.

2 When adverse conditions cause the fixed rule to produce non-optimal routing decisions, the learning scheme operating under similar conditions has the ability to adapt from previous routing strategy, re-converging to utilise the unused network capacity.

3 The ability of the learning scheme to perform load sharing is a desirable attribute. This is shown in operation with the tendency for the adaptive scheme to equalise the individual source blocking probabilities and to modify the congestion distribution for the nodes in the network.

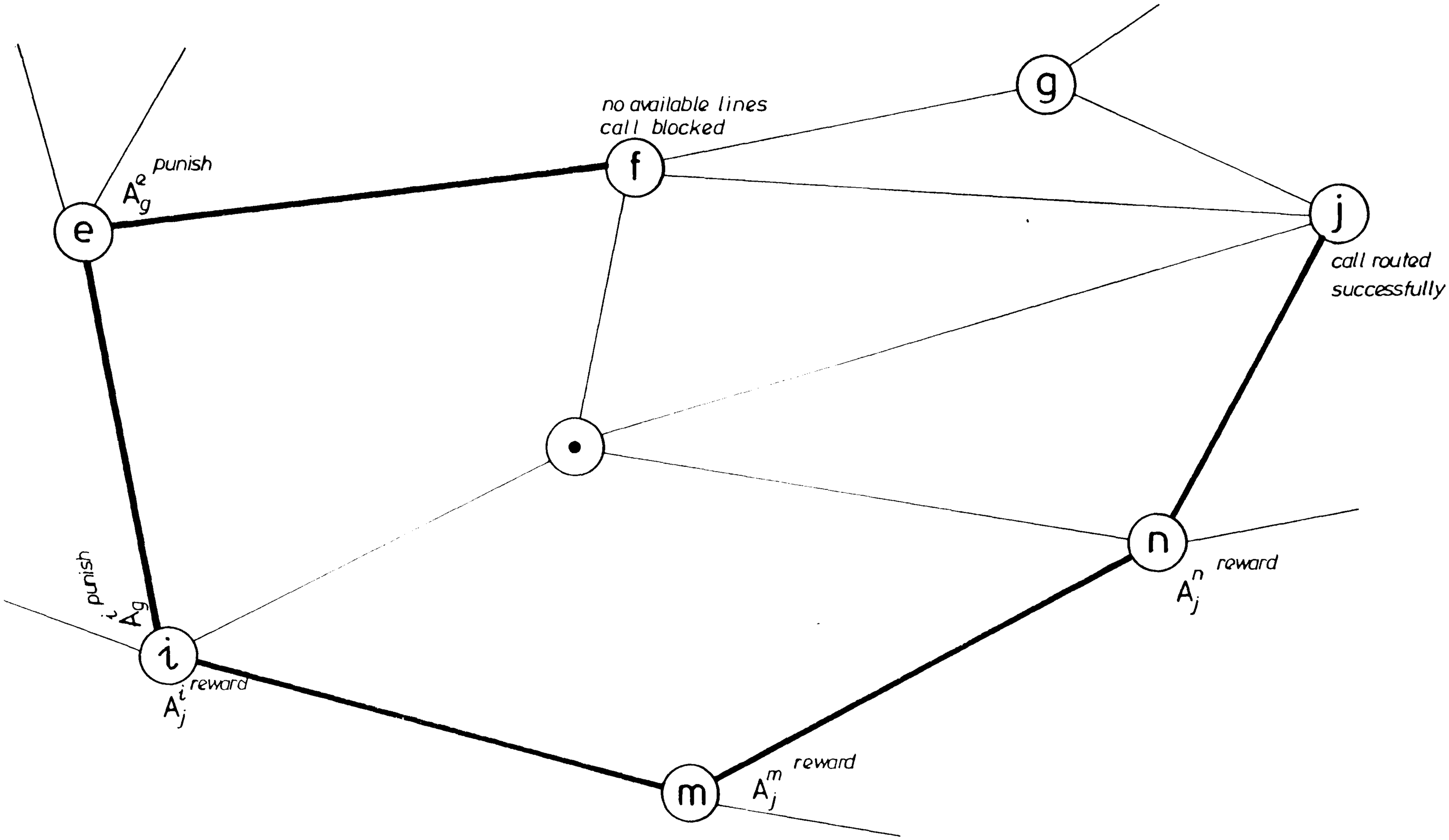


Figure 5.1 Learning automata routing in a large network

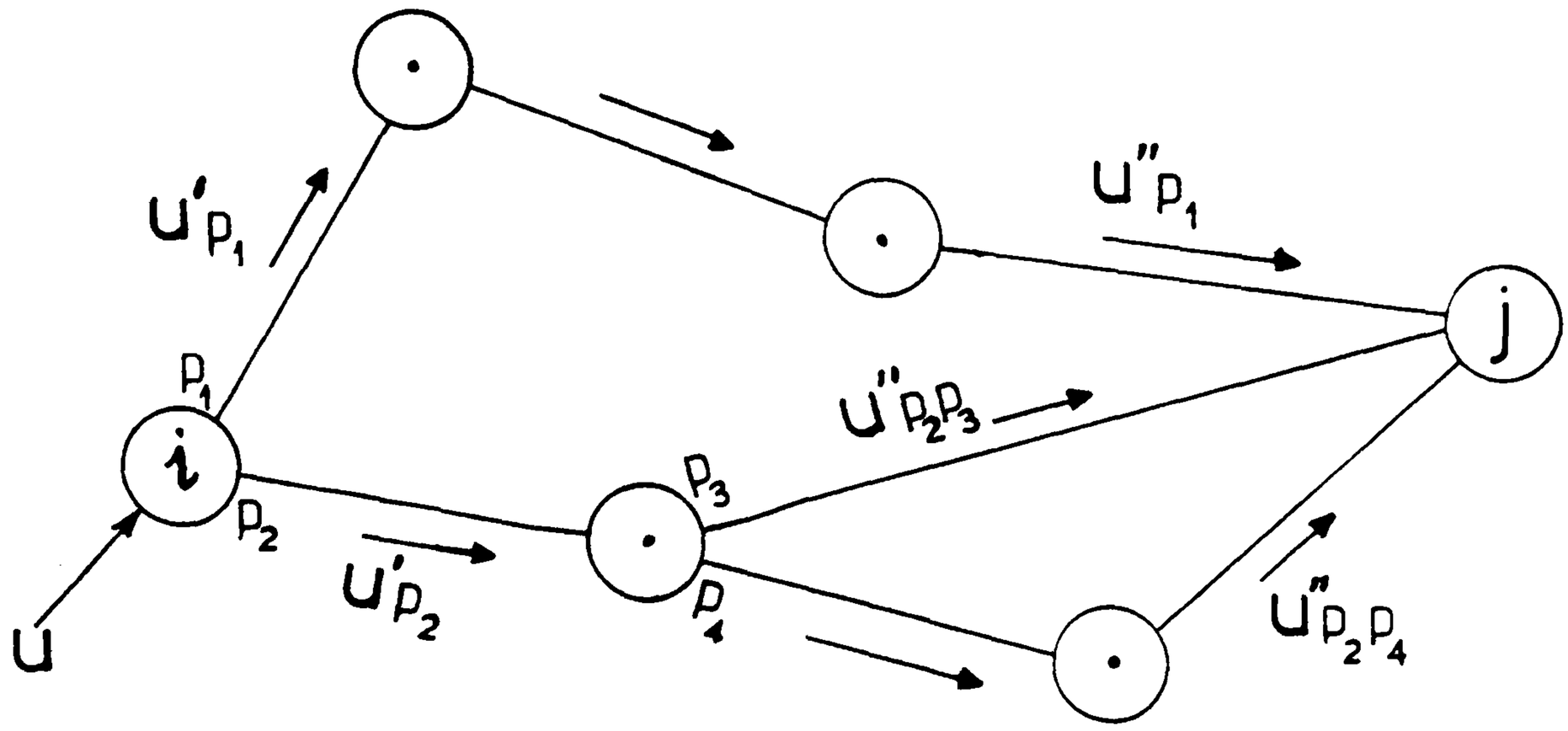


Figure 5.2a Flow splitting

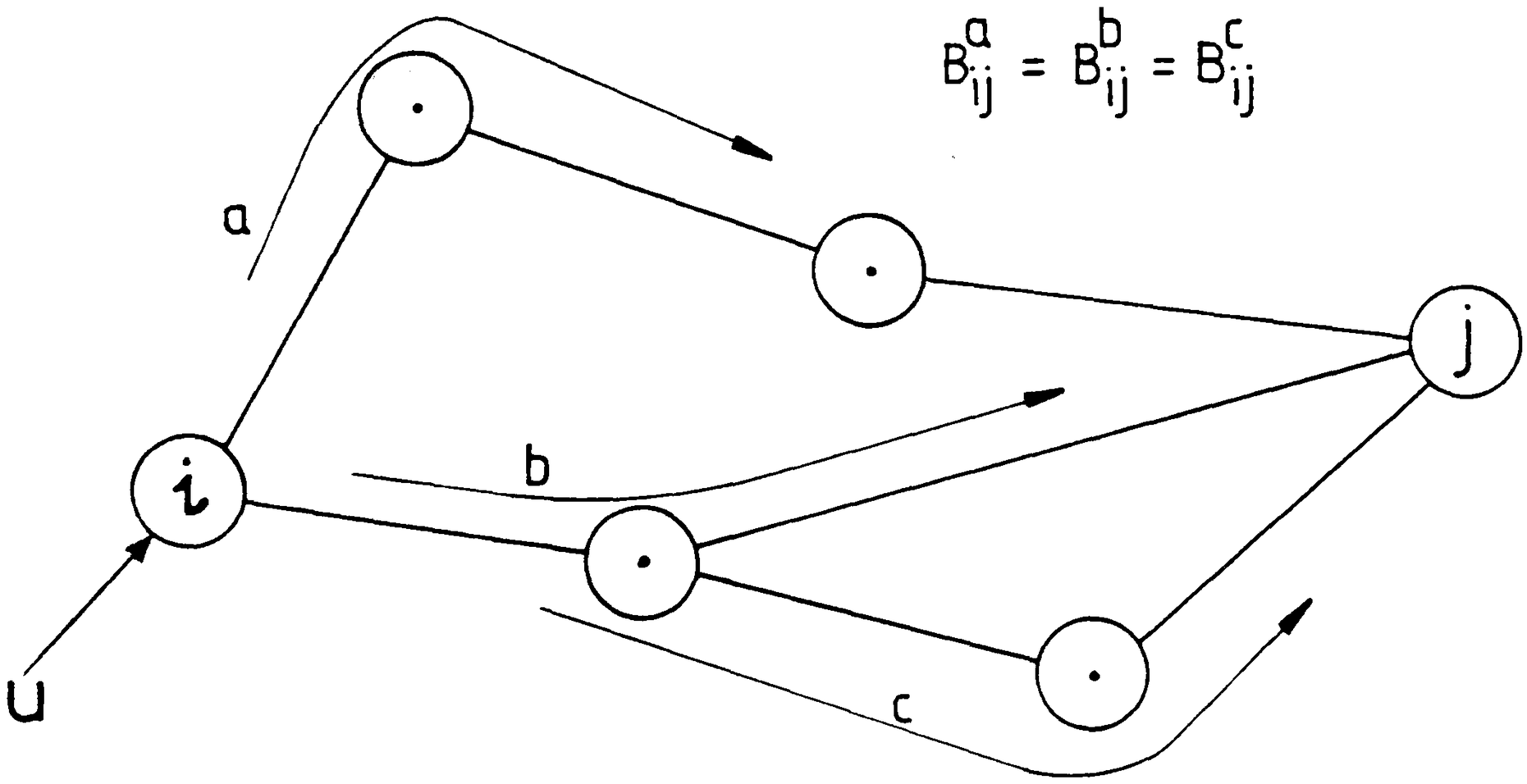


Figure 5.2b Path blocking probability equalisation

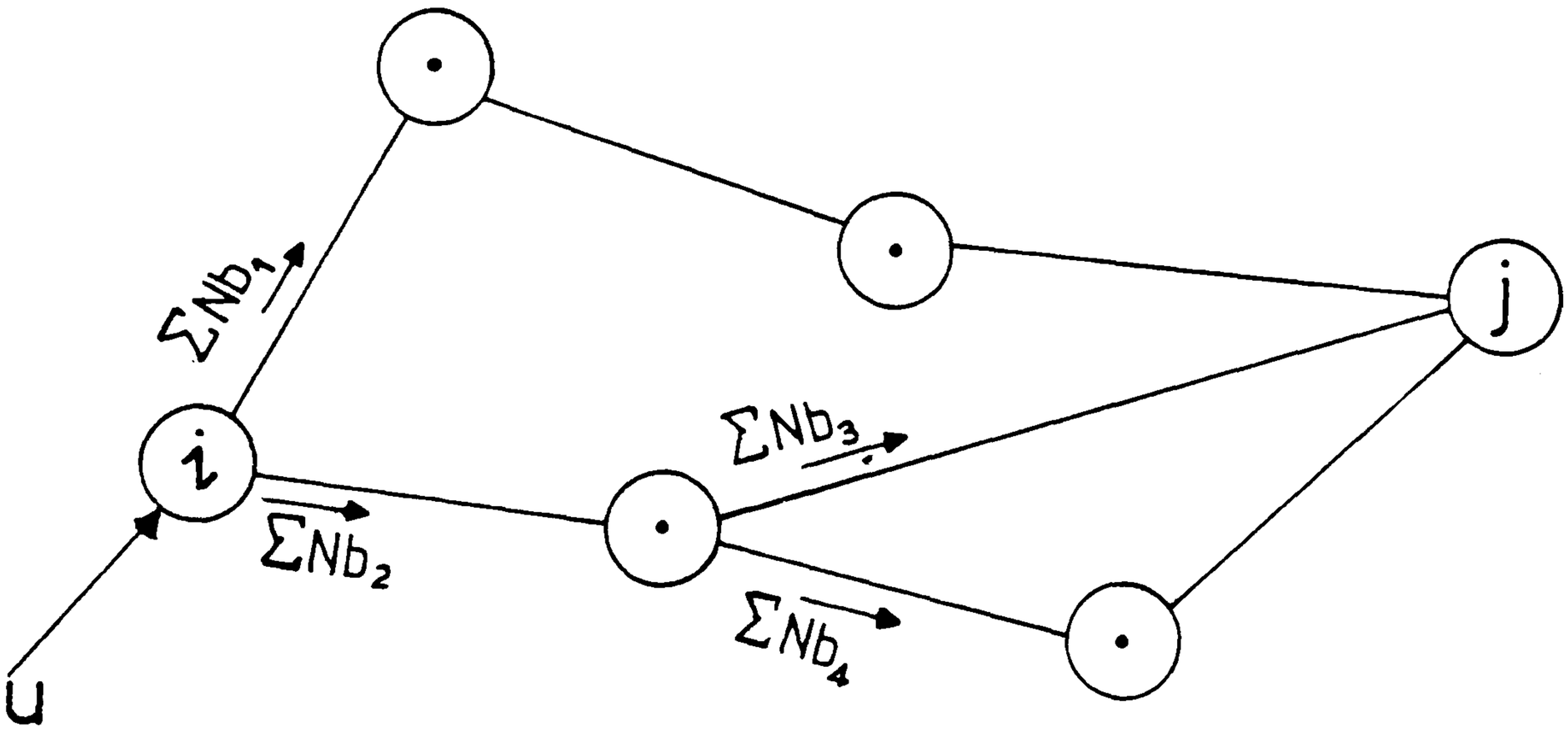


Figure 5.2c Blocking rate equalisation

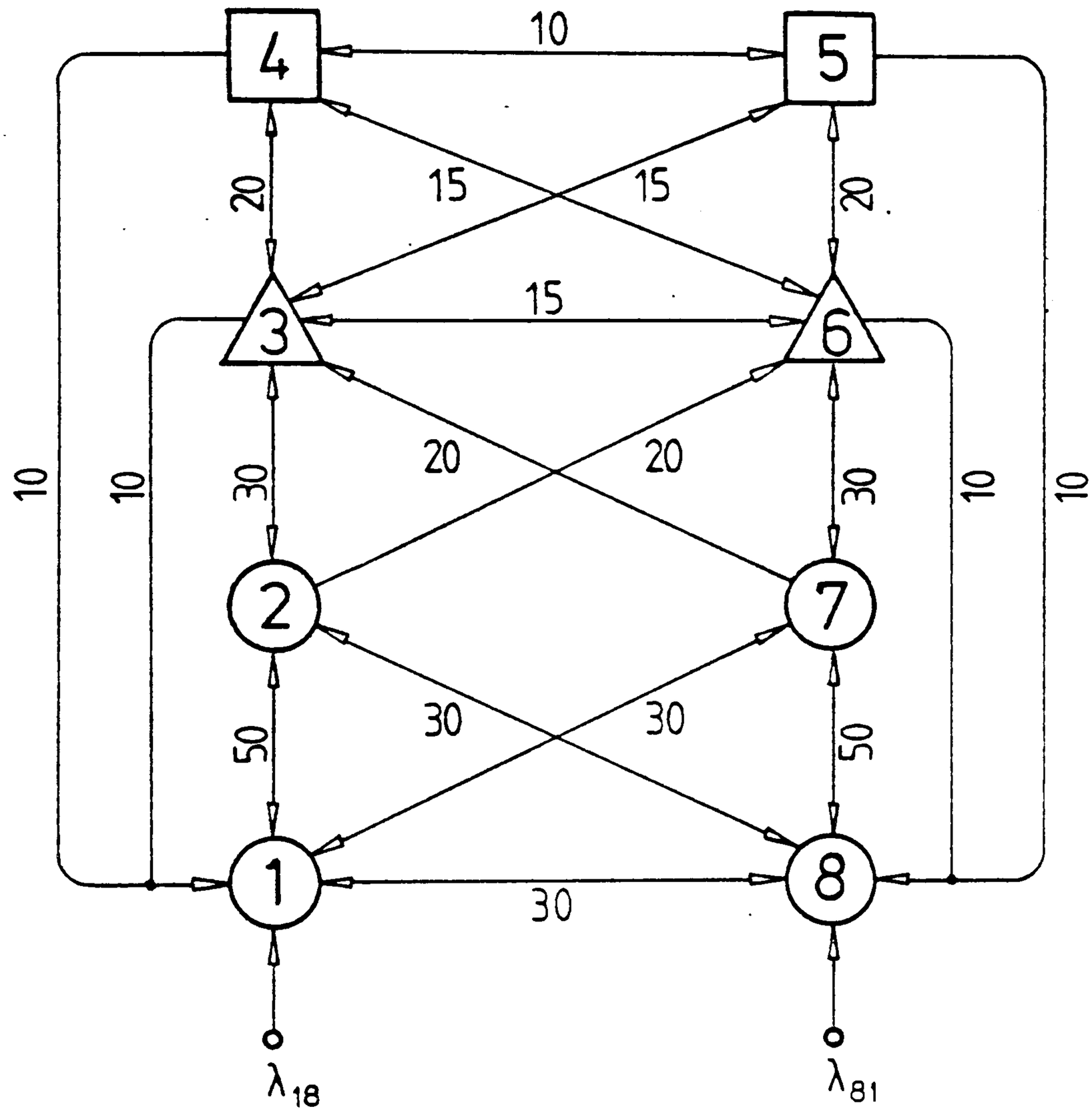


Figure 5.3 8 Node hierarchical network

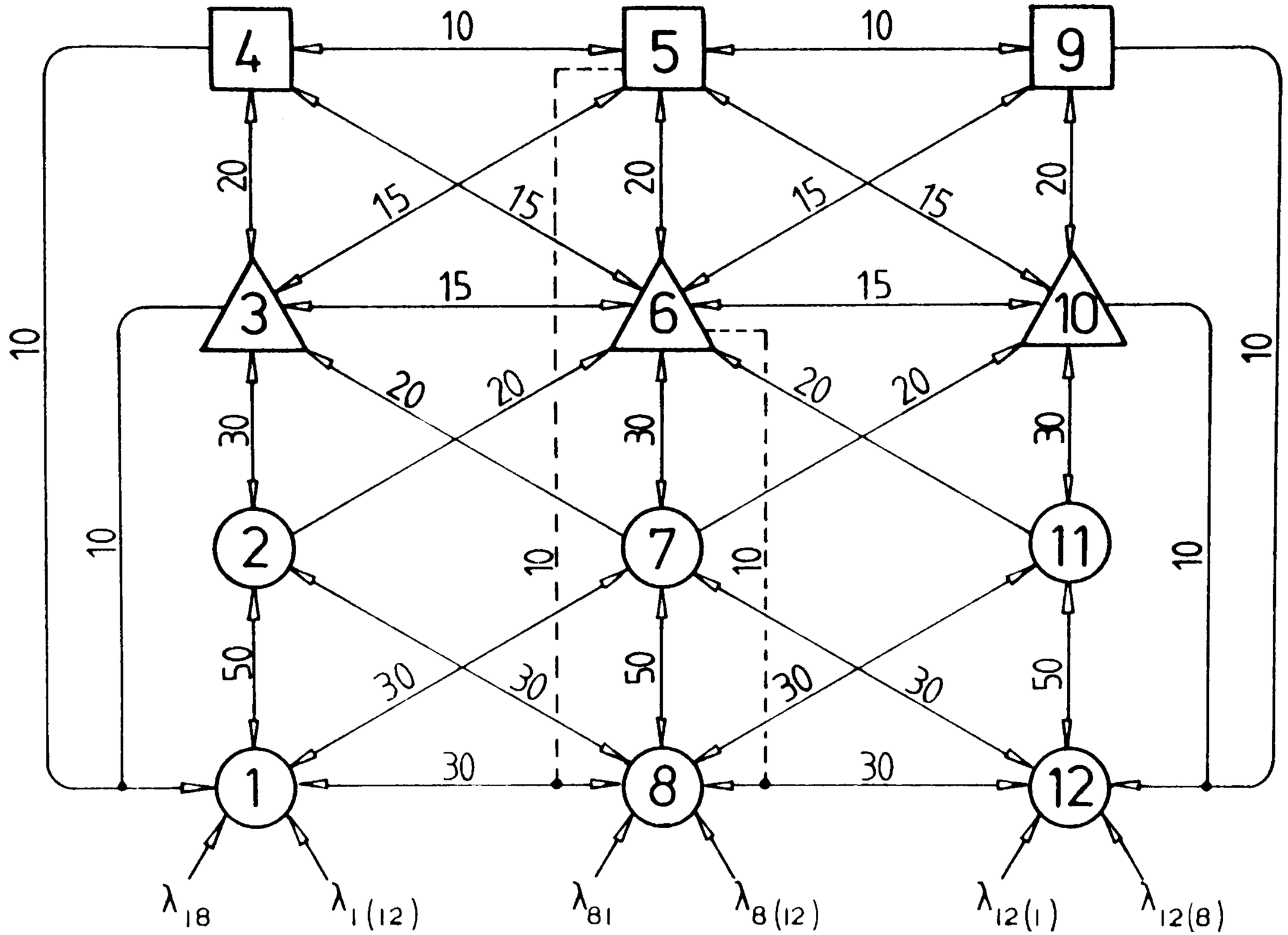
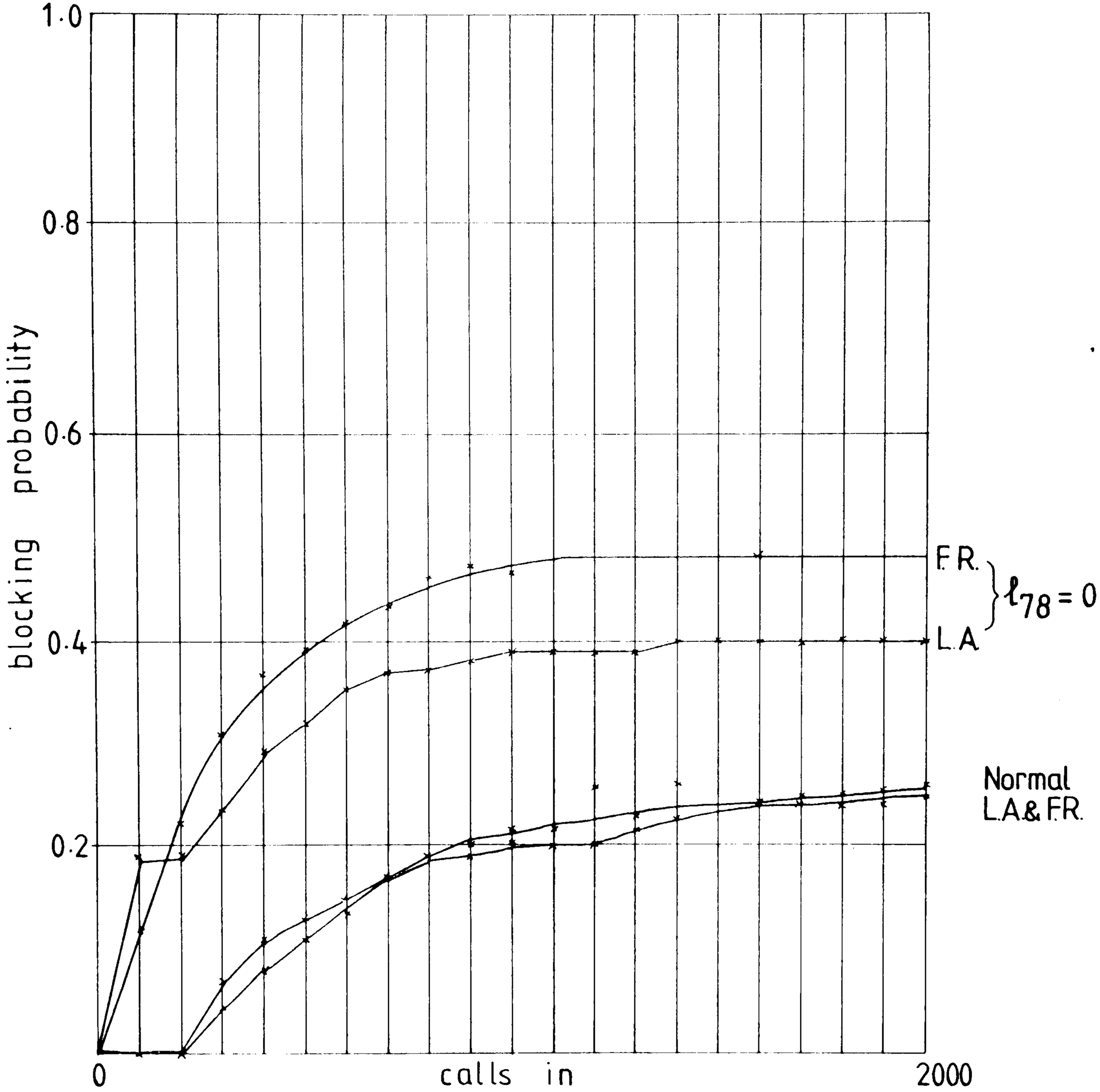


Figure 5.4 12 Node hierarchical network



Overall blocking probability graph for 12 node hierarchy

Figure 5.5

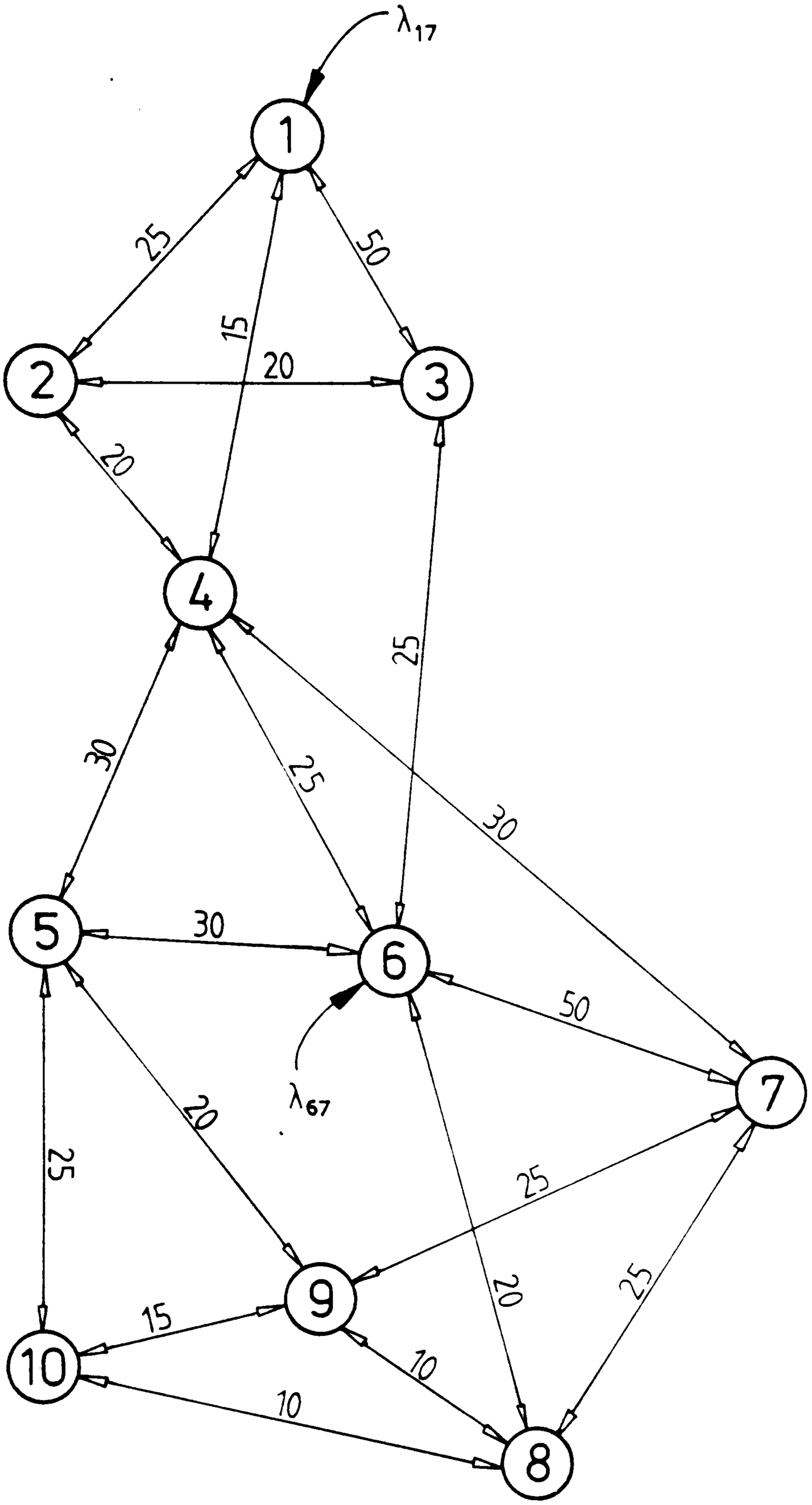
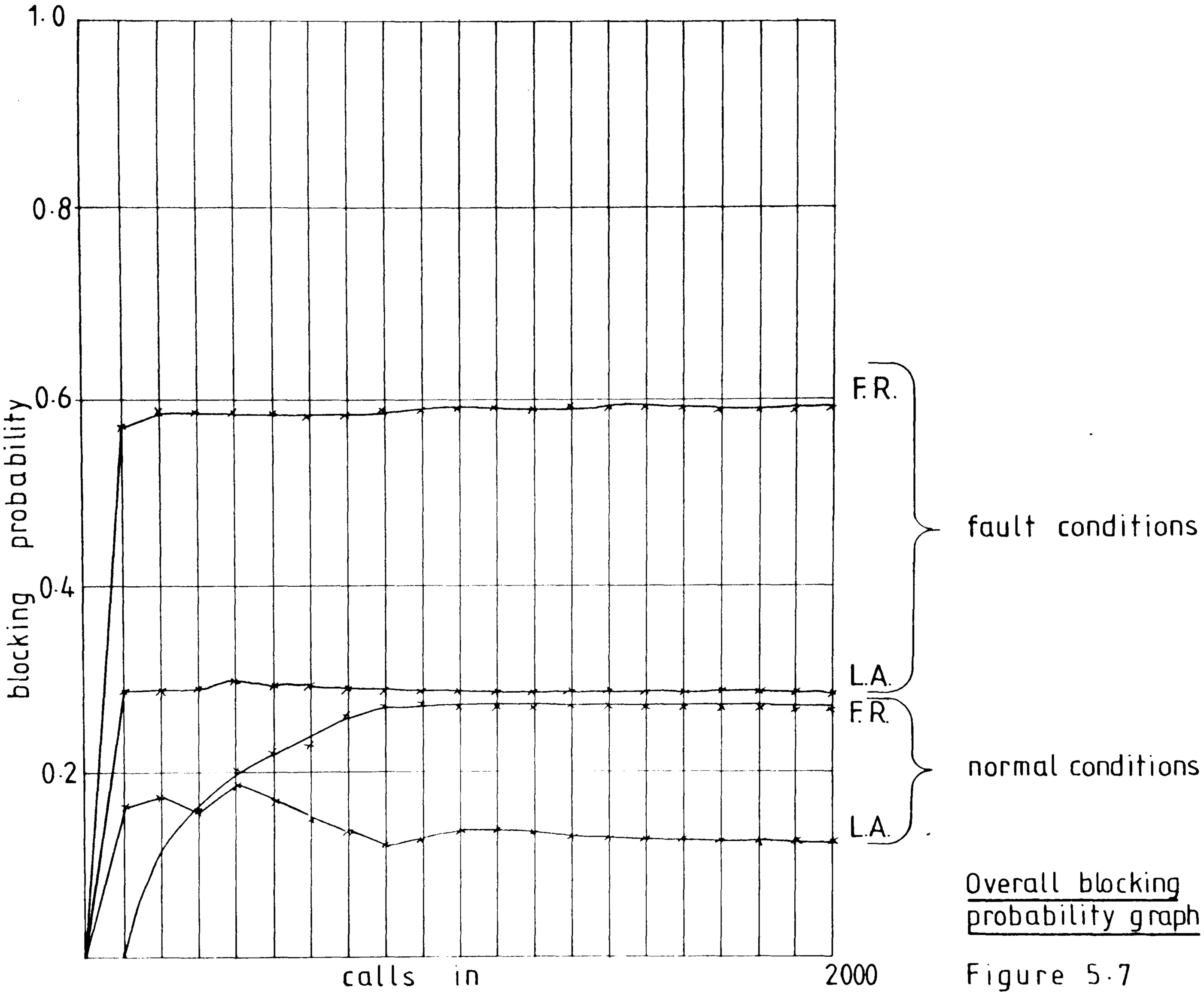
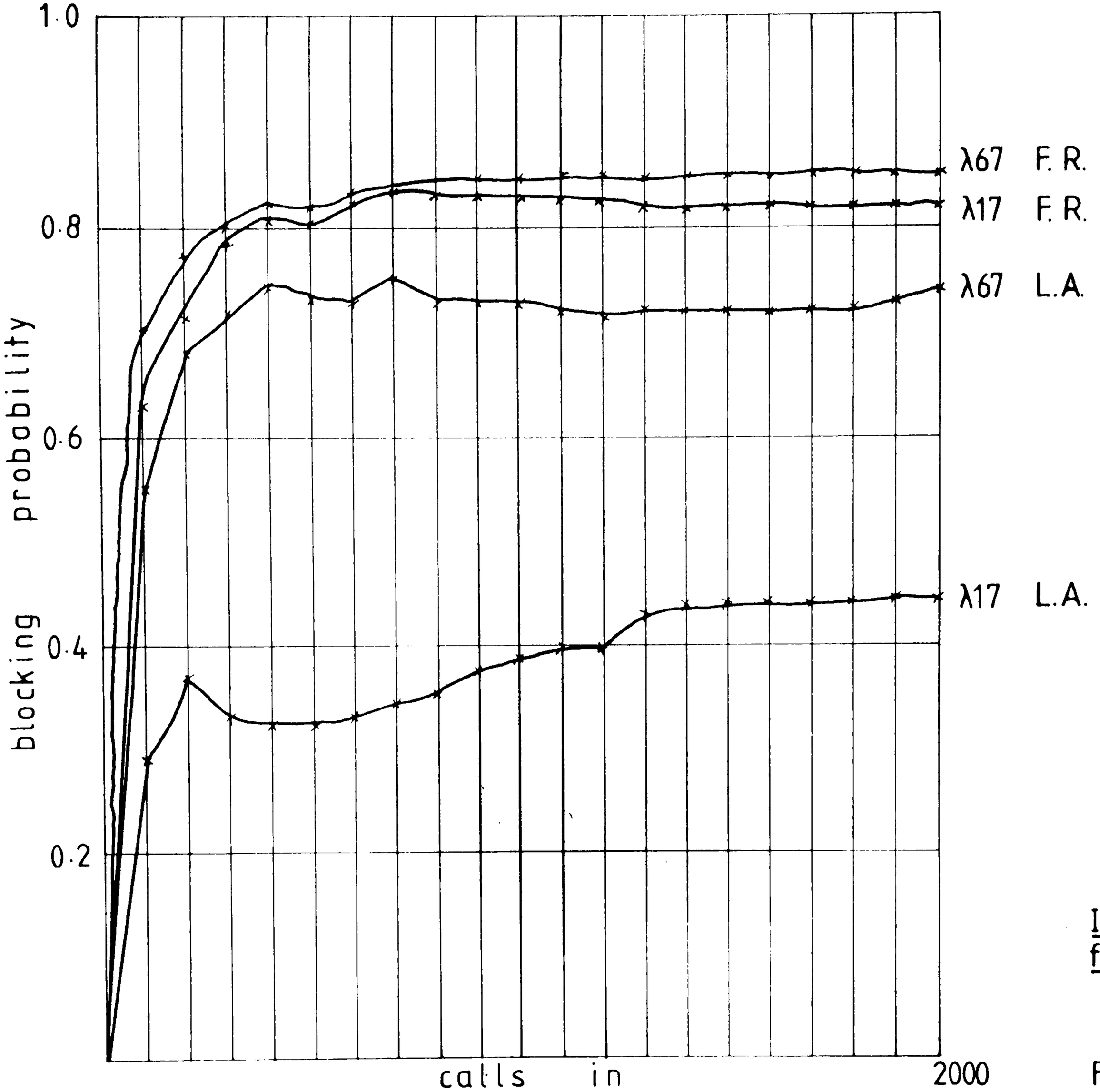


Figure 5.6 10 Node mesh network



Overall blocking probability graph

Figure 5·7



Individual call sources for mesh network

Figure 5·8

CHAPTER 6

APPLICATION OF LEARNING AUTOMATA ROUTING IN A PRACTICAL COMPLEX NETWORK

6.1 Introduction

In previous chapters, theoretical and simulation studies of learning automata routing have been presented, initially considering simple network configurations to permit investigations of the convergence behaviour of individual automata and the relationship with optimum performance. Later experiments, based on cross sections of the hierarchical telephone network consisting of up to 12 nodes, but with limited traffic sources allowed further evaluation of the adaptive routing technique including the impositions of network faults and asymmetric traffic conditions. A need therefore exists to demonstrate the L A routing scheme in a realistic network with a complete traffic set and to further compare the technique with the optimal fixed rule under static and non-stationary network conditions. In addition, a study of the realistic network provides the opportunity to investigate the self-organising learning automata scheme.

Bell Network

For the following sequence of experiments, formal specifications of a 10 node network topology, optimal alternate path routing scheme and a set of traffic statistics were obtained from Bell Laboratories[35] and are presented fully in Appendix 1. The essential data for this network was derived using off-line optimisation techniques, to produce a compatible network/routing rule design for the given constraints of the network topology and estimated traffic. It therefore offers an ideal test-bed for comparing and /

and proving the adaptive scheme, since the operation of the optimal fixed rule may be considered as a performance datum, dictating a lower bound on the expected behaviour.

Self-Organising Routing

As suggested in an earlier chapter, the self-organising routing scheme requires no prior knowledge of the network, allowing each automaton to select any of the outgoing trunk groups at a node. This arrangement may be regarded as a true adaptive routing scheme and presents an ideal routing technique for several communication networks, particularly military implementations, where both the topology and traffic behaviour are highly non-stationary. Furthermore, the optimal fixed scheme proposed for the simulations also consists of all possible options at a node and makes no attempt to invoke the Markovian constraint to exclude cyclic paths.

Since both routing schemes introduce the possiblity of ring-around, a maximum path length condition is established to limit the maximum lines a single call can collect. Calls which do not successfully reach the destination within the allotted number of lines are dropped from the network, to be re-attempted by the source user. For this study, the maximum call length is set to 6 lines.

6.2 Performance Measurement

The size and complexity of this new network has posed interesting problems concerning the measurement and evaluation of the network/routing scheme performance and as a result consideration has been given to suitable techniques which indicate the behaviour of the network and routing scheme. For studying the operation of the network, the overall blocking probability and traffic carrying properties are proposed /

proposed as convenient performance criteria. Supplementary information regarding the degree and rate of adaption are provided by the abstract measures of entropy and average path length, these projecting changes in the routing scheme structure.

Blocking Probability

The blocking probability is defined as the ratio of the calls unable to be serviced (rejected from the network) to the total number generated by all sources.

$$B\,P \triangleq \frac{\text{Number of calls rejected}}{\text{Number of calls attempted}}$$

In the results that follow the total rejected calls consists of calls blocked by no lines and also calls which are still unconnected after the maximum path length.

Network Traffic

In public utilities such as the telephone network, revenue is earned on the basis of total information carried and for this reason studies of switching networks are normally made according to the total network occupancy. In this study, indications of the network occupancy are based on the following criteria.

Total Traffic Carried

The total traffic carried, c may be regarded as the sum of hold times h_i for all calls serviced by the network over the simulation period. For a simple tariff scale this figure of merit can be directly related to the revenue earned.

$$c = \sum h_i$$

Calls Currently in Network

The measurement of calls currently in the network gives /

gives a dynamic indication of network performance, specifying the total calls in progress at an instant of time. This is conveniently displayed in a graphical form which ideally shows the variation of the network traffic condition as the simulation progresses.

Average Path Length

The average path length $\bar{n}$ is a measure of the mean number of lines required by a call and is consequently a criterion for the efficient usage of network facilities.

$$\bar{n} = \frac{\text{Number of lines in use}}{\text{Number of calls in network}}$$

Again this is drawn in a graphical form and is a useful measure to observe the adaption of the routing scheme.

Routing Scheme Entropy

The use of entropy[10] as a measure of dis-organisation is used in many areas of engineering and physics and is mathematically stated

$$H \overset{\Delta}{=} \sum_{i=1}^{n} p_i \log p_i \quad \text{bits}$$

(where p_i is the probability of performing the i^{th} action)

In this form, the entropy is a practical method for studying the order or structure of the routing scheme, and in a general network is given by

$$H_{rs}(n) = \sum_{i \epsilon N} \sum_{j \epsilon D} \sum_{k \epsilon R} p_j^{ik} \log p_j^{ik} \quad \text{bits}$$

where N is the set of nodes, D the set of destinations and R the allowable actions at each automaton in the network.

From the entropy, the redundancy, $R_{rs}(n)$ can also be stated

$$R_{rs}(n) = 1 - H'_{rs}(n)$$

where H' is the normalised entropy given by

$$H'_{rs} = /$$

$$H'_{rs} = \frac{H_{rs}}{H_m}$$, where H_m is the maximum entropy

As the adaptive scheme converges to a suitable routing structure the redundancy of the routing scheme might be expected to increase as certain routes are given preference by the evolution of individual automata.

Therefore

$$\frac{dR_{rs}}{dn} > 0$$

It follows

$$\frac{d}{dn}\left[1 - H_{rs}/H_m\right] > 0$$

which gives

$$H_{rs}\frac{dH_m}{dn} > H_m\frac{dH_{rs}}{dn}$$

For the schemes suggested in this thesis the maximum entropy H_m remains constant for the duration of each experiment since the number of automata involved is fixed, with no automata joining or leaving the network. The above inequality can thus be modified to suit

$$\frac{dH_{rs}}{dn} < 0$$

This implies that for adaption or learning, the routing schemes must operate in such a way as to reduce the entropy. A decrease in the routing scheme entropy may be regarded as a reduction in the disorganisation of the scheme or an increase in redundancy as individual controllers learn of the network conditions.

6.3 Alternate Path Scheme /

6.3 Alternate Path Scheme

The results for the optimal alternate path routing scheme are described. For this experiment the topology and traffic statistics documented in Appendix 1 are used, the optimal sequences for the fixed routing scheme also as listed. As shown by the profile of Figure 6.1, the traffic statistics are switched to produce puesdo non-stationary arrival rates, further adding to realism of the simulation.

The fundamental reason for running a simulation experiment with an optimal fixed rule is to provide a bench mark result from which all other experiments can be compared. The data obtained from this experiment is subsequently superimposed as a datum point for the experiments that follow.

Results

Calls	2 000-4 000	4 000-7000	7 000-10 000
Overall B P	0.053	0.021	0.027

Calls	0-10 000
Traffic	810.17 hours

The above table provides a numerical measure of the time averaged blocking probability for the three pertinent regions of the simulation (Figure 6.1). Also, a graph of the blocking probability as a function of calls handled is provided (superimposed on each graph of blocking probability for the L A schemes). Clearly, these both indicate a satisfactory performance by the fixed rule under the engineered conditions and typically demonstrates a performance expected from a real network.

In addition to the results of the blocking probability, (Figure6.2) the traffic carrying capability of the network is illustrated by the /

the Total Traffic Carried and graphs showing the variation of calls in progress against calls handled. The line utilisation of the network/routing scheme is demonstrated with a graph of the average path length. Initially, with the network empty, calls tend to use the direct path between source and destination but as the network reaches steady state calls must also be routed over secondary routes. This is reflected in the average path length increasing from the minimum 1 line/call to a value of around 1.2 lines/call. These low values suggest an efficient usage of network facilities and in conjunction with the other performance criteria show the operation of an optimal routing scheme.

This experiment provides an excellent result for comparison of the adaptive schemes to be presented later. Furthermore, the performance obtained gives good correlation with the original specifications of the network and confirms the optimality of the design.

6.4 Self-Organising Learning Automata Schemes

In the following simulation experiments, learning automata routing in a self-organising scheme is demonstrated using the predescribed network (same topology and traffic conditions), aiming to minimise call blocking. The routing scheme utilises a L_{R-I} reinforcement algorithm in each automaton and in an attempt to enhance the performance, variations of the basic scheme are proposed and simulated.

Basic Learning Automata Scheme

For this simulation the learning automata scheme is operated using the simple feedback strategy described in previous chapters where

$$\beta_j^{kr} = 1 \quad /$$

β_j^{kr} = 1 call blocked

β_j^{kr} = 0 call routed successfully

In other terms, this strategy is only concerned with whether a call is routed successfully or is blocked, irrespective of how many lines are used in setting up the call. Although this arrangement might be expected to produce sub-optimal results for a self-organising scheme it nevertheless proves an interesting technique since an extremely simple and practical feedback policy is employed.

Results

Calls	2 000-4 000	4 000-7000	7 000-10 000
Overall B P	0.121	0.089	0.101

Calls	0-10 000
Traffic	749.62 hours

The blocking probability results (table above and Figure 6.2a) show a marginal increase in the overall blocking probability when compared to the results from the previous experiment. It is interesting to note the form of the blocking graph; the early performance of the routing scheme giving high values of blocking probability which decrease as the learning scheme adopts a suitable routing structure for the given conditions. The calls in progress graph (Figure 6.2b) also reflects the degradation in the network performance with less traffic carried by the network for the same sample size.

Since this experiment considers an adaptive routing scheme, the entropy and average path length graphs are of considerable interest. As suggested, the entropy curve (Figure 6.3b) displays the required reduction in entropy ($dH/dn<0$), /

($dH/dn < 0$), this falling from a maximum of 252 bits to a value around 110. This graph also indicates convergence to be complete after approximately 7 500 calls.

The average path length graph as shown on Figure 6. 3a gives a completely different behaviour from the fixed rule result. Whereas the fixed rule produced a minimum path length initially, increasing to a higher steady state, the adaptive scheme adopts a high average path length in the early stages and converges to a reduced steady state value. This can be attributed to the learning schemes initial lack of structure, the adaption of individual automata bringing the measure down as the optimum paths are found.

Although the results obtained from this experiment show a poorer performance than the optimal fixed rule, the self-organising ability of the learning automata scheme has nevertheless been satisfactorily demonstrated. From completely unbiased initial conditions the routing rule has converged in such a manner as to attempt the minimisation of the blocking probability. It must be emphasised that in this experiment comparison has been drawn with an optimally designed fixed rule, which under the design conditions and load, gives the best attainable performance.

However, in some circumstances, formal design of an optimal routing scheme is not possible, due primarily to insufficient data such as traffic estimates. Moreover, such a design can be of limited use, particularly when the network topology and traffic are highly non-stationary. Tactical defence networks such as Ptarmigan fall into this category, differing from civil systems not only in equipment engineering but also in traffic and operating aspects. Unlike the conventional telephone system military traffic predictions are entirely a matter of guesswork with the peak /

peak and off peak periods being strongly related to tactical events, thus making reliable traffic estimates for routing rule design a difficult task.

Reduced Option Learning Automata Scheme

In an attempt to improve on the performance of the basic learning scheme a routing scheme is proposed with less allowable actions per automaton, which for this example is the first three elements of the optimal fixed rule;

Results

Calls	2 000-4 000	4 000-7 000	7 000-10 000
Overall B P	0.161	0.082	0.047

Calls	0-10 000
Traffic	756.44 hours

A study of the above table and Figure 6.4a shows a good improvement in the blocking characteristics of the network/ routing scheme combination, particularly in the latter stages of the simulation when convergence is complete. A similar enhancement of the routing scheme performance is indicated by the traffic measurements, the graph of calls in progress showing on average, a greater call carrying ability with respect to the simple scheme suggested previously.

As in the earlier adaptive routing experiment, the average path length and entropy graphs provides a means to study the dynamics of the routing scheme. In this case they show the consequences of reducing the action sets of the automata with the maximum entropy H_m taking a lower value than previously noted. Likewise the plot (Figure 6.5a) of the average /

average path length against calls handled shows convergence to a lower steady state of 1.75.

Despite the obvious improvements in performance, this scheme has several restrictions which limit effective usage. Prior to establishing the routing scheme, knowledge of the network topology is required, and if the best three options are to be chosen, some off-line optimisation procedure must be implemented. Furthermore, the limited range of actions severly constrains the operation of the routing scheme under failure conditions. The results from the experiment do however promote the ability of the L A scheme as a means of producing a favourable performance.

Path Length Feedback Scheme

Whereas the previous scheme tended to improve the network/routing scheme behaviour by limiting the choice of the automata, it is suggested[36] that utilising the path length of calls as an additional feedback parameter is a further and more desirable means for improvement. This scheme has the desirable property of minimising ring-around conditions and long path lengths and hence supports a more efficient strategy.

β_j^{kr}

Path Length	Call Status: Blocked	Call Status: Routed
$\leq M$	Penalty	Reward
$>M$	Penalty	Penalty

For the scheme implemented, M was set to two lines with the previously mentioned path maximum maintained at 6.

This policy has the desired effect of encouraging the automata to route calls along the routes which utilise the minimum of network facilities, effectively permitting other source/destination pairs access to the unused capacity.

Results /

Results

Calls	2 000-4 000	4 000-7000	7 000-10 000
Overall B P	0.107	0 074	0.058

Calls	0-10 000
Traffic	765.15 hours

The table and graph (Figure 6.6a) shows the influence of the path length feedback on the blocking behaviour of the learning automata routing scheme. In comparison with the measured performance of the basic L A scheme, this modified algorithm converges to a lower blocking probability in a shorter period of time. This tends to confirm proposals that the availability of additional information (path length) improves the ability of the scheme to converge on an optimal strategy. The traffic measurements (Figure 6 6b) displays a similar improvement, the lower blocking probability resulting in a greater number of calls in progress.

The average path length graph also helps present this current scheme as the best alternative to the optimal fixed rule under the design conditions. In the steady state region of the graph (Figure 6.7a), the average path length is noted as approximately 1.6 lines/call, an improvement over all other automata experiments.

The path length feedback scheme offers the best approach to self-organising routing of all the learning schemes demonstrated, being truly adaptive with no apriori information assumed by the scheme. Furthermore, implementation in a real network is a practical proposition, the scheme utilising the straightforward P-model algorithm /

algorithm and a simple measure for the path length, both of which are compatible with existing network operations.

In retrospect, a more suitable technique of employing the path length parameter can be adopted. This new feedback policy is demonstrated on Figure 6.8b alongside the original variant Figure 6.8a, which as discussed earlier only rewards when a call is successfully routed within the allotted number of lines. In the modified strategy, a call routed on a path greater than the up-dating length constraint, rewards the automata nearest to the destination (a maximum of M up-dates) and penalises the remaining automata in the chain. This arrangement is seen as a superior up-dating mechanism as it gives the opportunity to reward automata which find the proposed destination but also helps restrict the formation of long and cyclic paths, which are seen to be wasteful of network resources.

Start-up Bias

In the schemes presented so far, the automata convergence has been from a completely unbiased condition, i.e., all actions equally likely. However, in most practical applications, the geography of the network is likely to be known and it should be possible to bias the initial action probabilities in favour of the most direct paths.

In the unbiased scheme, the initial state of an automaton was given by

$$p_i(0) = \frac{1}{r}$$ where the automaton has r possible actions

For the biased scheme, the initial action probabilities are structured such as to encourage the automata to take the better paths in the early stages of operation, without imposing rigid constraints should a need exist to adopt an alternative /

alternative routing strategy

$$p_1(0) = .5, \quad p_2(0) = .25, \quad p_3(0) = .125, \ldots p_{n-1}(0) = p_n(0)$$

Where action i corresponds to the i element in the descending order of shortest paths.

Results

Calls	2 000-4 000	4000-7 000	7 000-10 000
Overall B P	0.13	0.011	0.054

Calls	0-10000
Traffic	792.78 hours

The blocking probability results (Figure 6.9a) for this experiment indicate the initial bias of the action probabilities to be effective in improving the early behaviour of the routing scheme. As shown on Figure 6.10a the average path length graph also shows the consequence of the technique, restricting this to less than 1.8 lines/call throughout the experiment. Furthermore, the latter stages of the simulation again illustrates the advantages of the path length constraint scheme, in this case giving good results for the mid and final regions of the run.

The global convergence of the scheme, displayed by the entropy graph (Figure 6.10b) gives the effect of the initial bias on learning automata. Primarily, a new H_m is noted, reduced in value due to the initial structure of the action probabilities. Also noted, is the improved convergence of the scheme, a lower entropy being reached in a shorter period of time.

The idea of establishing a start-up bias for the action probabilities has been shown to be an effective method of encouraging /

encouraging a rapid convergence by the automata and improving the global convergence of the routing scheme. It still however leaves the scheme completely flexible to handle unforseen events which require re-convergence and also allows automata to operate in the desired independent manner.

6.5 Dynamic Injection of Abnormal Traffic

As indicated in previous chapters, the benefits of a learning automata routing scheme are truly realised when failure or abnormal conditions prevail, a formally designed fixed rule no longer able to provide an optimal routing service. In this section, the effect of dynamically imposing adverse traffic conditions on the network is investigated, showing the operation of the optimal fixed rule under non-engineered loads and the capability of the adaptive scheme to reconverge so as to provide a better routing structure.

The generation of adverse traffic conditions is seen as a most effective means of disrupting the normal operation of the network without escaping from realistic bounds in order that a significant change in operating conditions can be noted. In this particular study, the switching between the normal traffic matrix (Appendix 1, Table 1) to an alternative traffic set with the same overall load (Appendix 1, Table 4) is performed according to the profile shown on Figure 6.11. For the two experiments that follow, the routing schemes implemented are the optimal alternate path scheme and the path length constrained learning automata scheme, set with the parameters suggested in previous experiments.

Optimal Alternate Path Scheme

The alternate path scheme specified in Appendix 1 is utilised as the routing algorithm in a 12 000 call sample, no /

no attempt being made to modify the routing sequences despite the change in traffic. Again the results from the optimal fixed scheme are used to provide a datum for comparing the adaptive scheme.

Results

Calls	4 000-6 000	6 000-9 000	9 000-12 000
Overall B P	0.063	0.35	0.371

Calls	0 - 6000	6 000 - 12 000
Traffic	481.82 hours	325.0 hours

From the blocking probability results shown it is evident that the fixed rule is no longer optimal after the switch to the second traffic group. Although still numerically equal in total, the traffic situation is now distributed in a different manner and thus causes problems to the rigid routing policy of the alternate path routing scheme. Similarly, the effects of the non-optimal performance by the routing scheme is shown in the traffic measurements, these indicating a significant decrease after the change to the adverse traffic conditions.

The average path length gives further proof of the inability of the alternate path scheme to service the new traffic conditions. Prior to the switch the average path length is noted as approximately 1.2 lines/call. The new condition however causes a significant increase, in this experiment to about 2.3 lines/call, suggesting a non-optimal performance from the scheme.

The results from this experiment clearly indicate the dependence of optimal routing on the offered traffic conditions, showing how the distribution of individual traffic sources influence /

influence the behaviour of the fixed scheme. It further demonstrates the inflexible nature of the alternate path scheme in dealing with abnormal conditions and shows the need for a routing scheme with a variable structure capable of providing satisfactory service whatever conditions are met.

Learning Automata Scheme

In this experiment the path length feedback scheme is implemented as the routing algorithm in the network and subjected to the traffic statistics introduced previously.

Results

Calls	4 000-6 000	6000-9 000	9 000-12 000
Overall B P	0.099	0.363	0.332

Calls	0-6 000	6 000-12 000
Traffic	424.32 hours	333.34 hours

From the blocking probability graph (Figure 6.12a) it is well illustrated that the optimal fixed rule gives a good routing service when design conditions prevail, allowing the scheme to operate as originally designed. The learning automata scheme initially lacking structure also converges to provide a reasonable result in the same period, although inferior to the optimal design. However, the introduction of the abnormal traffic set highlights the limitations of a fixed rule, with a noticeable increase in the blocking probability. On the otherhand, the versatility of the learning scheme is such that after experiencing (learning) the change in traffic, the restructuring of the routing probabilities helps provide a better blocking behaviour.

The average path length (Figure 6.13a) provides a convenient medium for displaying the learning ability of the adaptive /

adaptive routing scheme. At the outset of the experiment convergence of the automata is shown by a decrease in the average path length. After the traffic switch a similar effect is noted with the average length showing an increase followed by a reduction, reflecting the re-convergence of the routing scheme to suit the experienced conditions.

The results of this experiment provides further evidence to support the adaptive powers of the L A routing scheme which as suggested can cope with conditions poorly serviced by a fixed scheme. Although the behaviour of the scheme has tended to be marginally sub-optimal in relation to the alternate path scheme under design loads and conditions, the learning scheme has the flexibility to out-perform the fixed rule when conditions become other than those originally specified.

6.6 Summary and Conclusions

The operation of a self-organising automata routing scheme has been demonstrated and compared with an optimal fixed rule in a complex network. Variations of the basic learning scheme have been suggested, including the use of the path length as an additional feedback parameter and the biasing of the initial action probabilities to speed the convergence process. From the results obtained several conclusions can be drawn regarding the nature of learning automata routing in large networks and their relationship with optimal fixed rules.

1 Under design conditions, the optimal alternate path routing scheme provides a service which gives excellent blocking and traffic carrying performance. Moreover, in the same circumstances, the operating protocol of the alternate path /

path scheme, first attempting the most direct paths, gives an efficient and effective routing operation.

2 The self-organising ability of the learning automata routing technique has been suitably demonstrated, the scheme converging to give a marginally sub-optimal performance with respect to the formally designed fixed rule. As suggested, such a service is adequate in certain types of network where it is usually impossible and impractical to produce an optimal design for the alternate path scheme.

3 The added constaint of path length helps improve the performance of the basic learning automata scheme by encouraging call routes to follow the most direct paths, without restricting the scheme from adopting alternative routing strategies if required.

4 As shown in the final experiments the occurrance of other than design conditions cause the fixed path scheme to produce sub-optimal results. Likewise, the learning scheme, although in the early stages is also disrupted, re-converges to produce a better performance.

General Comments

In this series of experiments the use of an externally specified network has been necessary to further promote the technique of learning automata routing. The topology of /

of this network however tends to support the alternate path routing technique, since many direct paths are possible between source/destination pairs, favouring the "far to near" policy of the fixed scheme. In a less highly connected network it could be construed that the learning automata scheme might be of greater advantage, particularly when the average path length is in excess of 2 lines/call, the global feedback policy permitting individual routing controllers to react to down chain restrictions, otherwise impossible to the isolated operation of an alternate path scheme.

The learning automata approach to adaptive routing has much to be commended. From an engineering standpoint the simple feedback strategy, which is fundamentally inherent in the network operation, is compatible with existing network operating protocols. Therefore, to fully implement such a scheme, little alteration of existing hardware is required, the only significant changes being to the routing procedures of the switching centre software. Also, since the scheme operates in a completely de-centralised manner, using simple up-dating algorithms, little additional load is imposed on the processing facilities of individual exchanges.

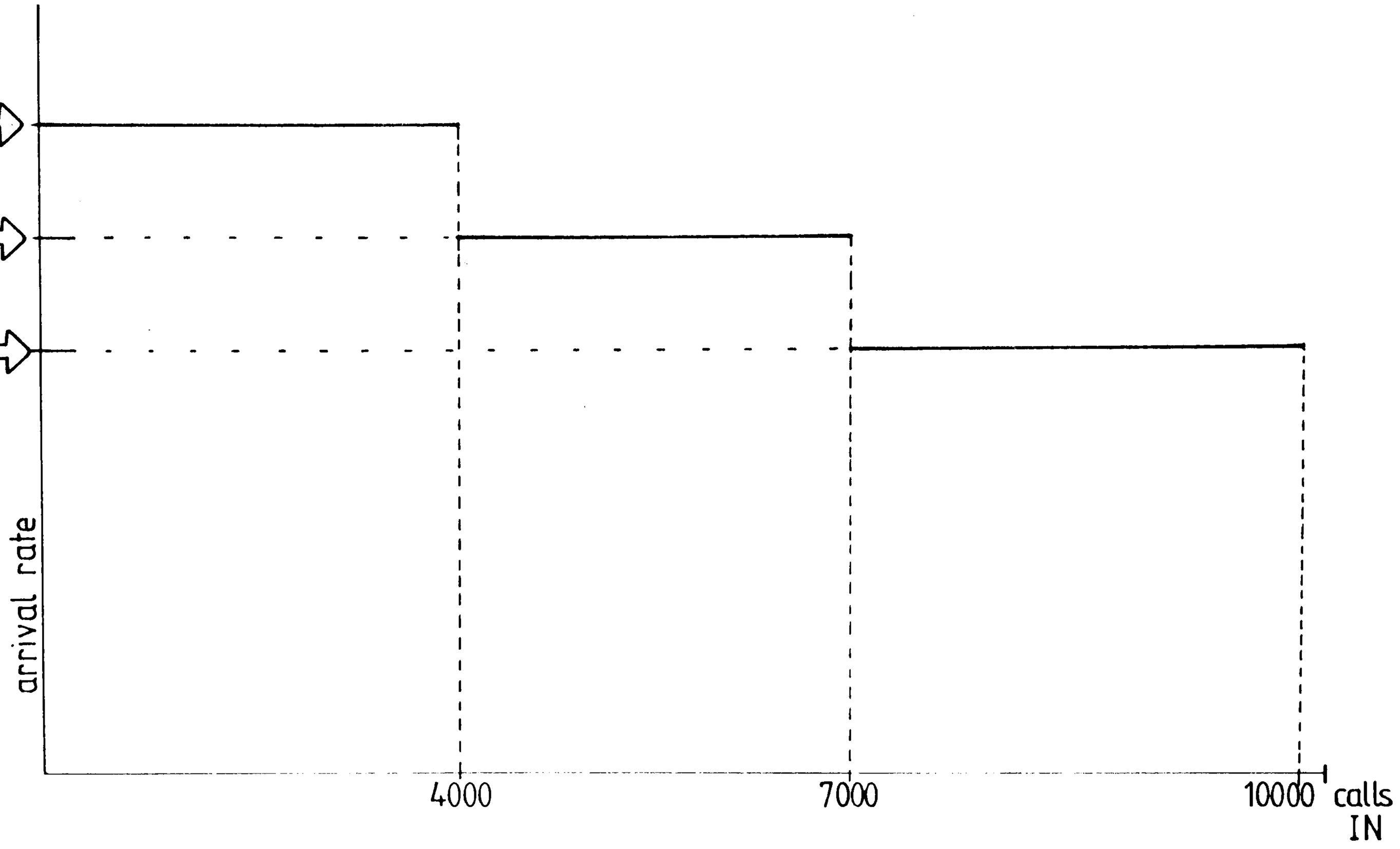
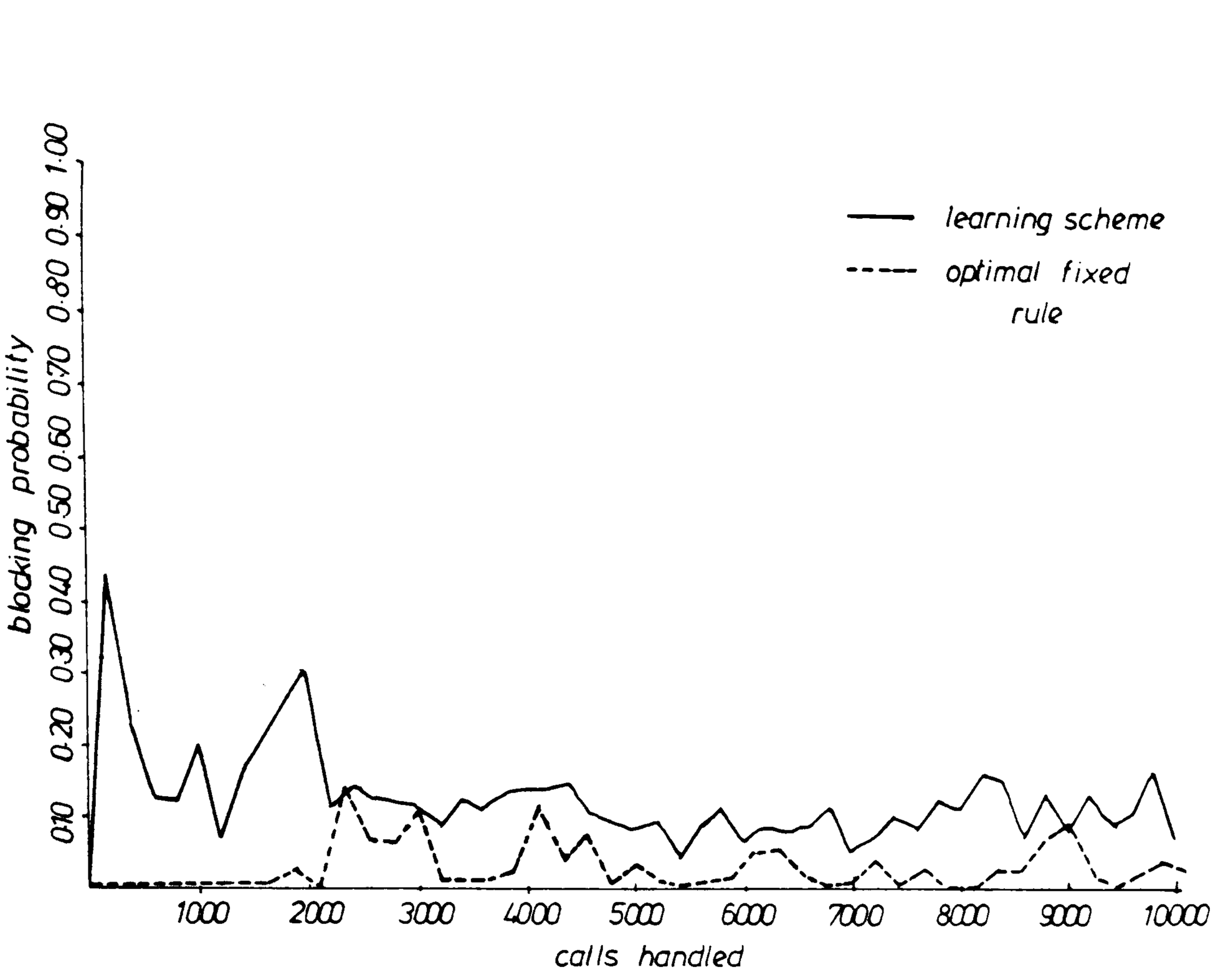
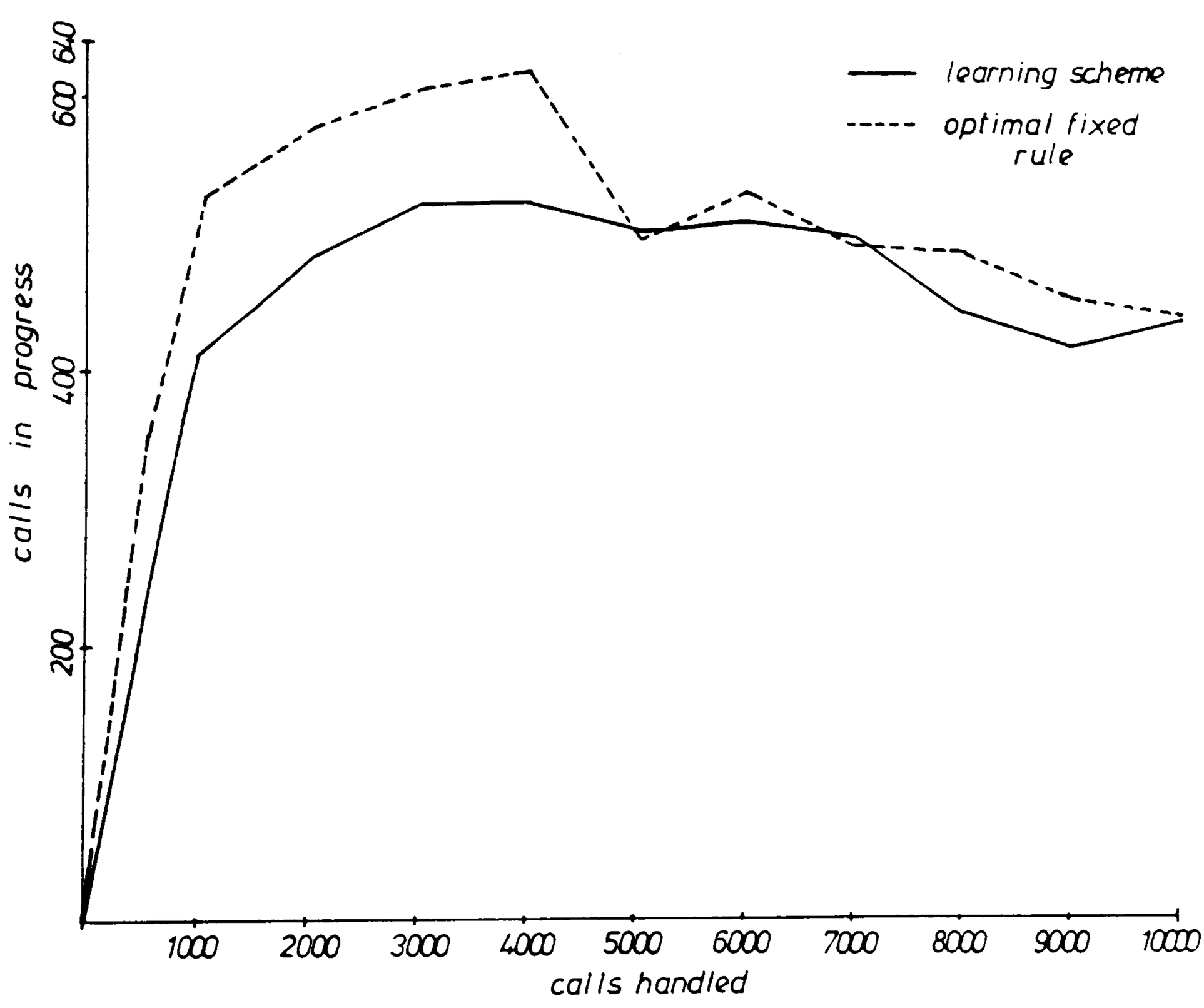


Figure 6.1 Traffic profile for normal condition experiments

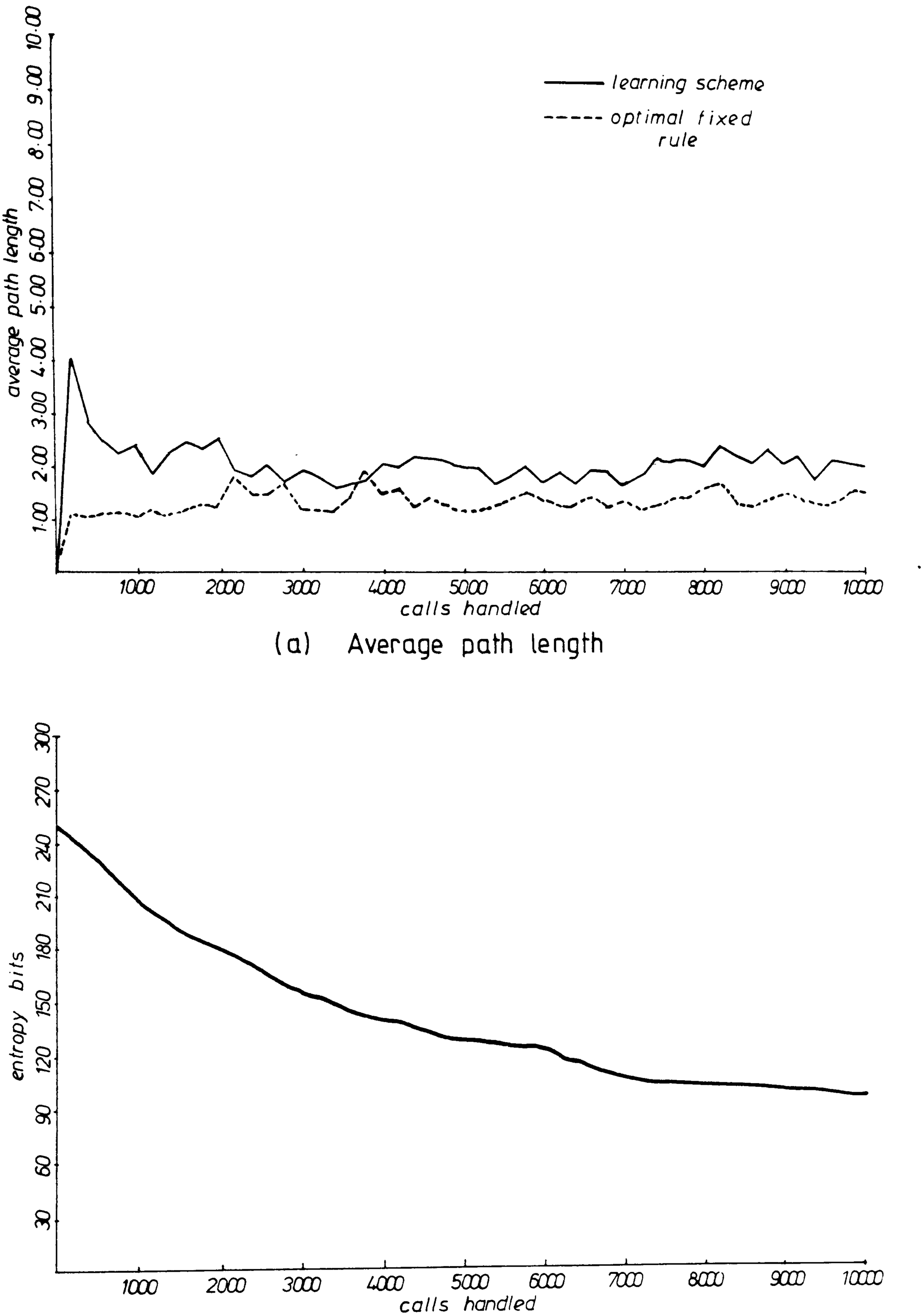


(a) Overall blocking probability



(b) Calls carried

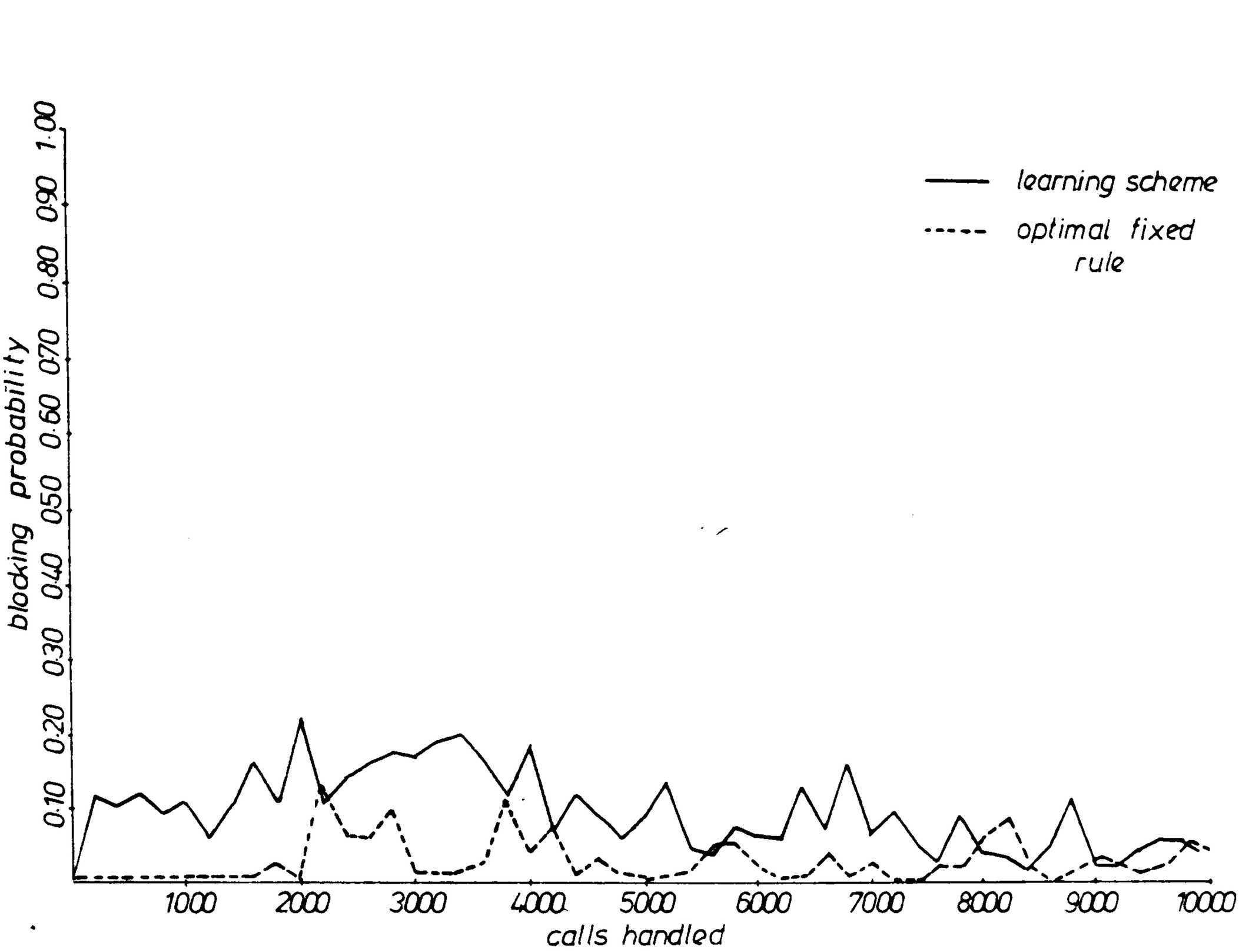
Figure 6·2 Results for basic learning scheme



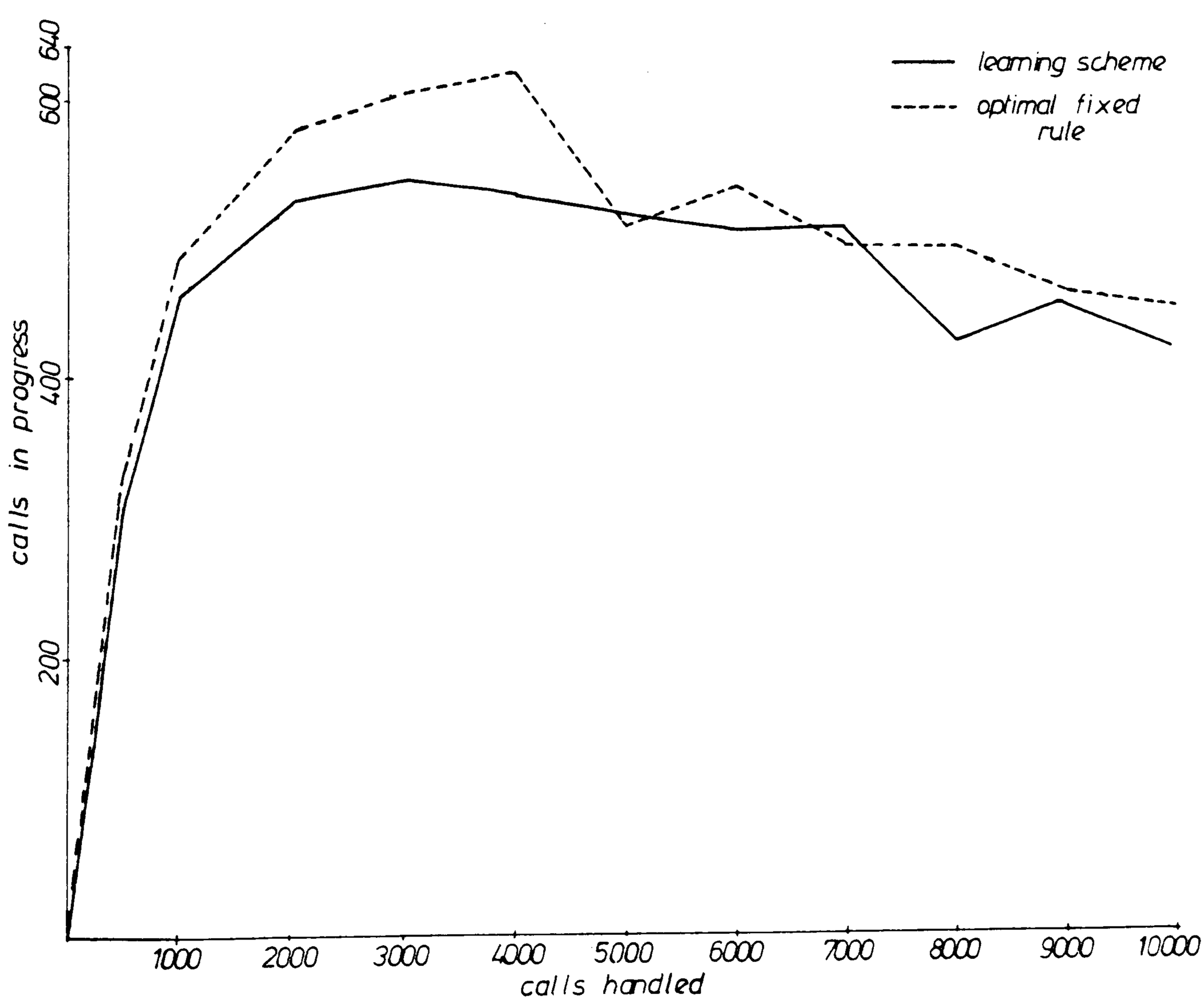
(a) Average path length

(b) Routing scheme entropy

Figure 6·3 Results for basic learning scheme

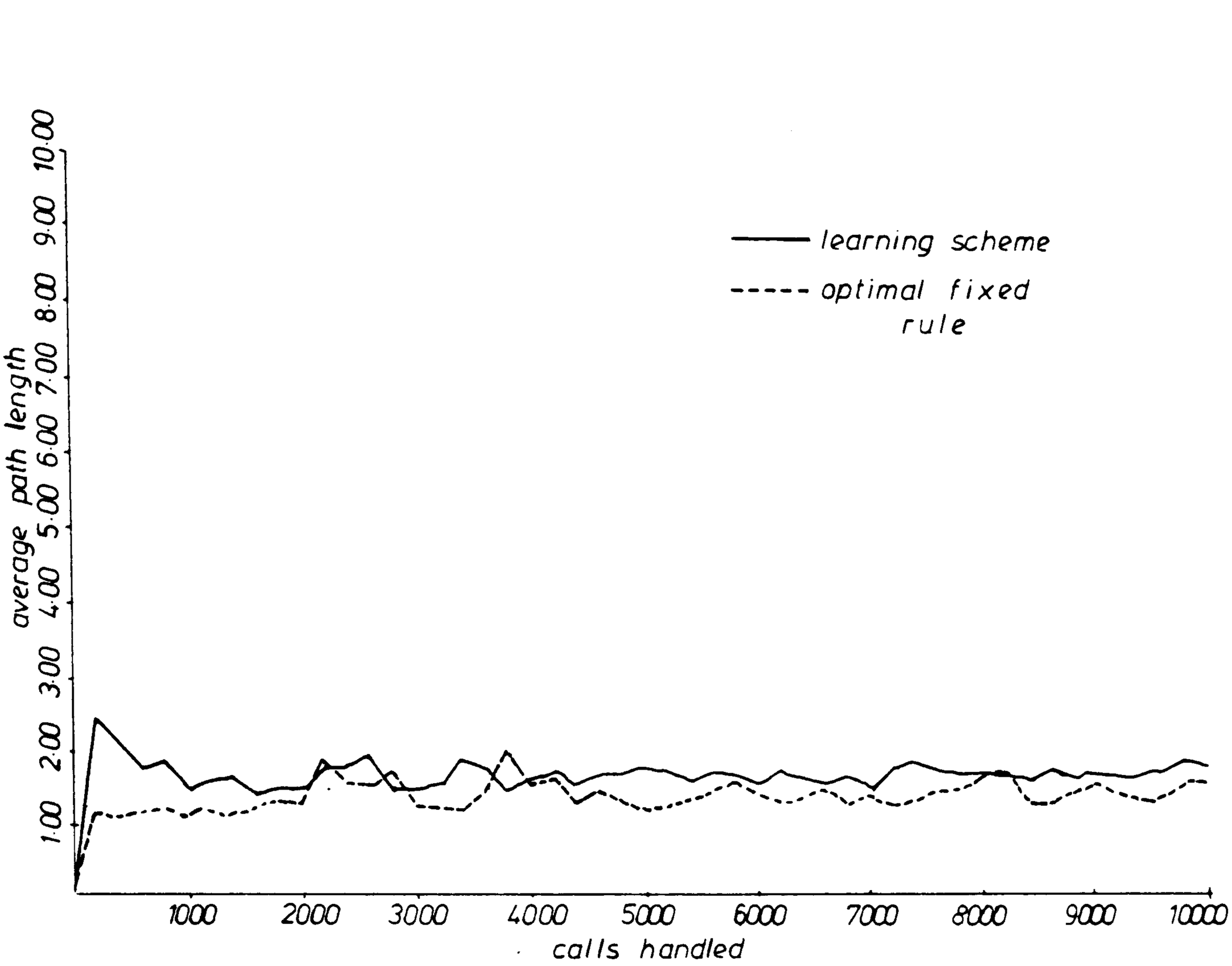


(a) Overall blocking probability

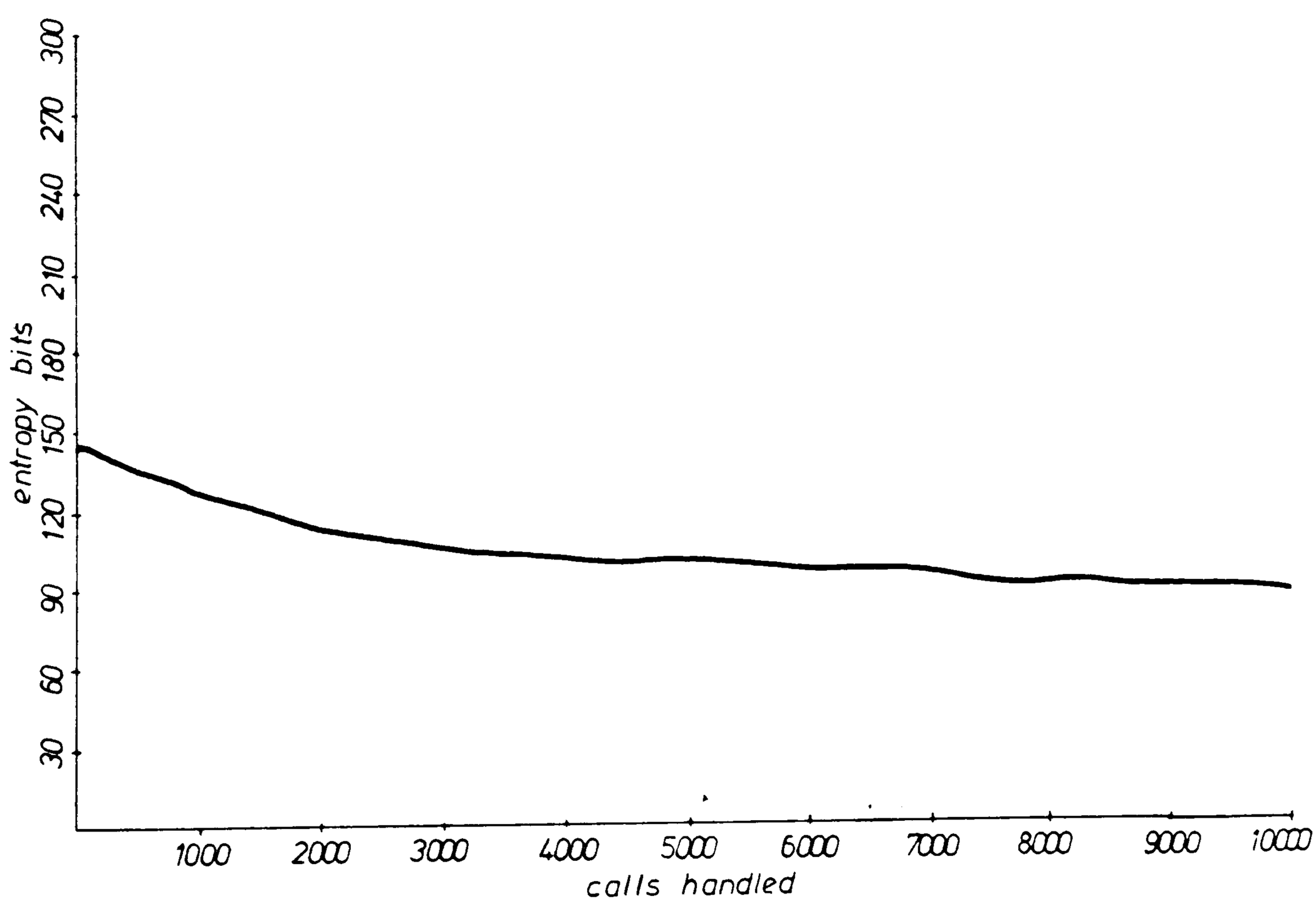


(b) Calls carried

Figure 6·4 Results for reduced options scheme

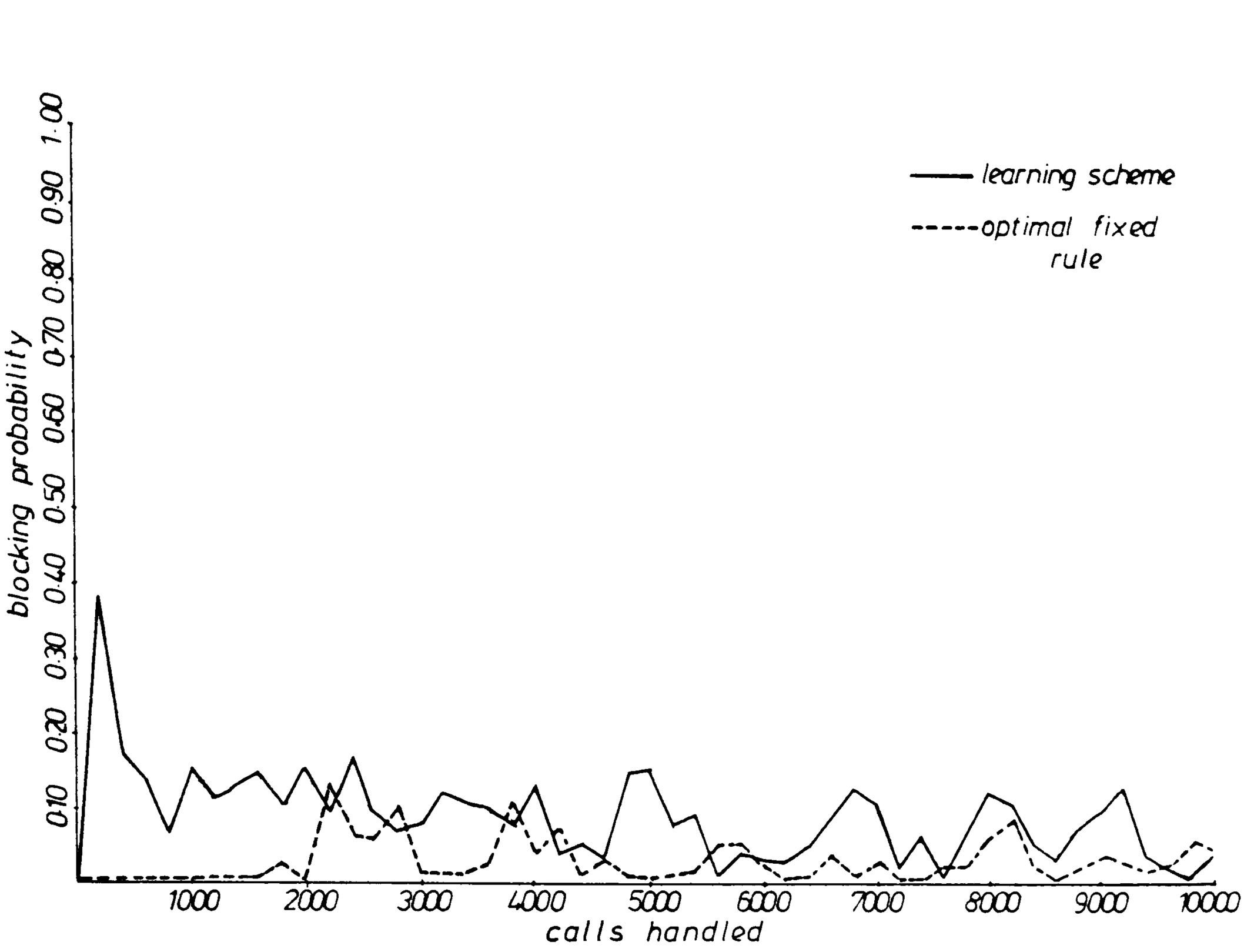


(a) Average path length

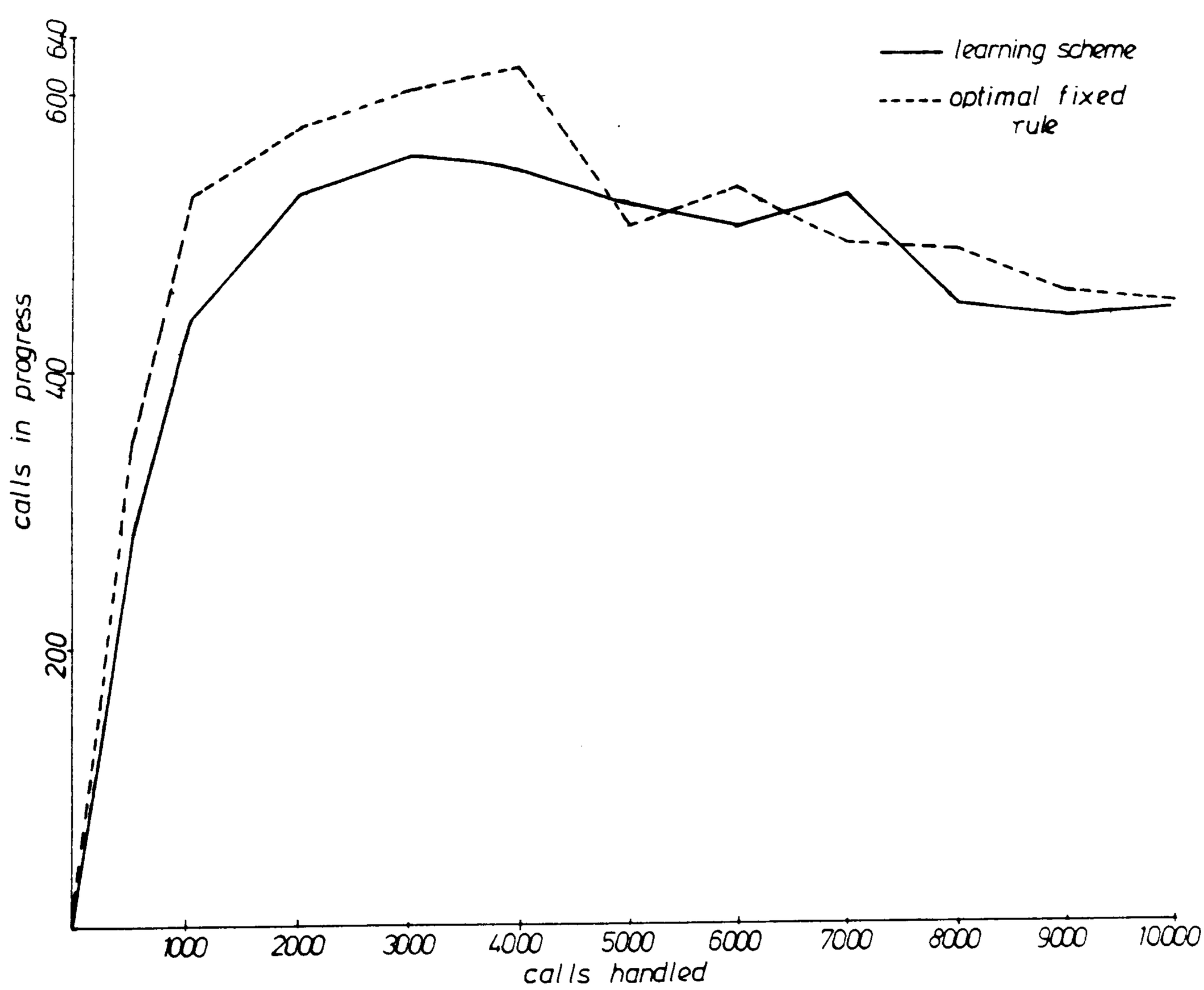


(b) Routing scheme entropy

Figure 6·5 Results for reduced options scheme

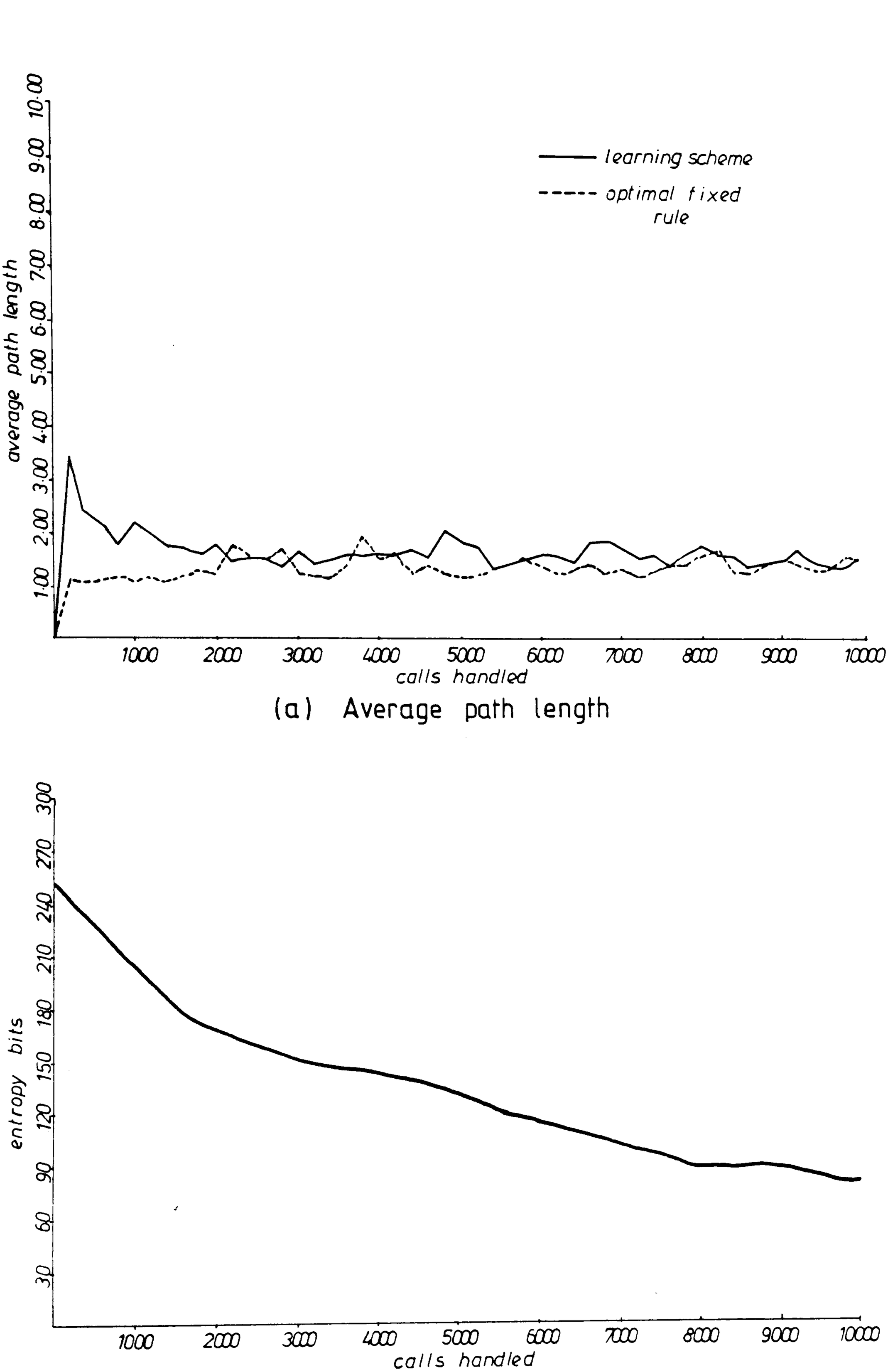


(a) Overall blocking probability



(b) Calls carried

Figure 6·6 Results for path length feedback scheme



(a) Average path length

(b) Routing scheme entropy

Figure 6.7 Results for path length feedback scheme

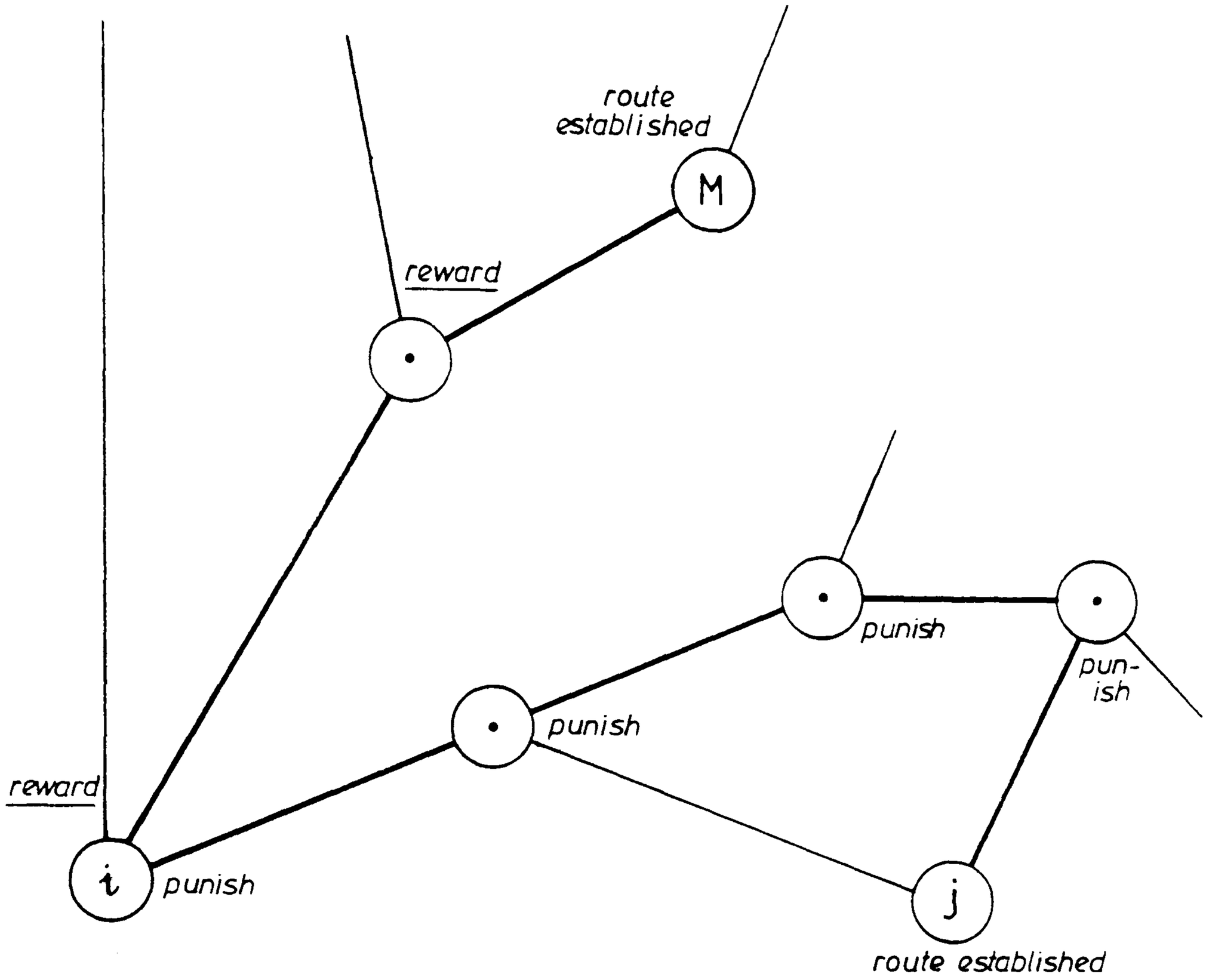


Figure 6·8a Original path length feedback policy

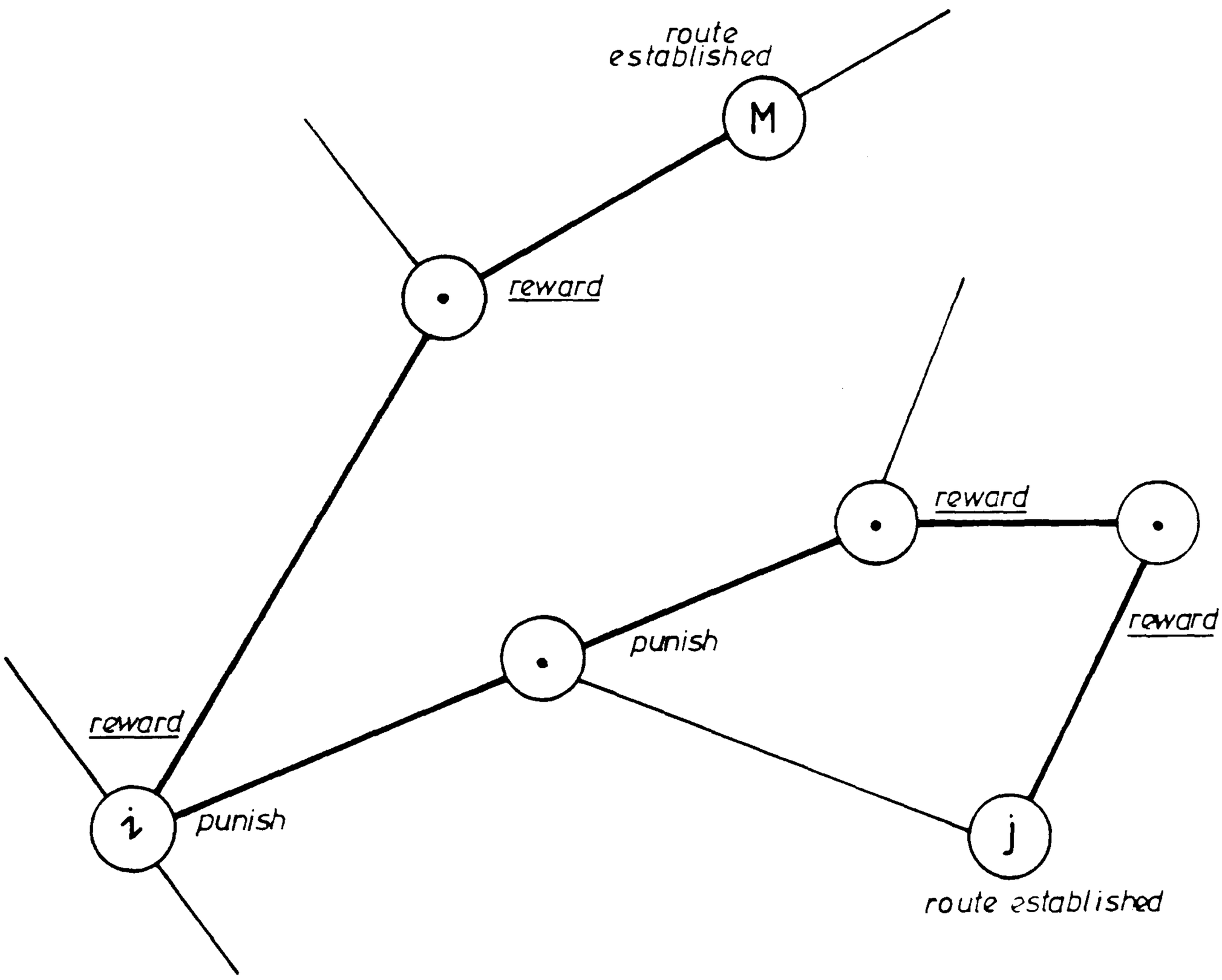
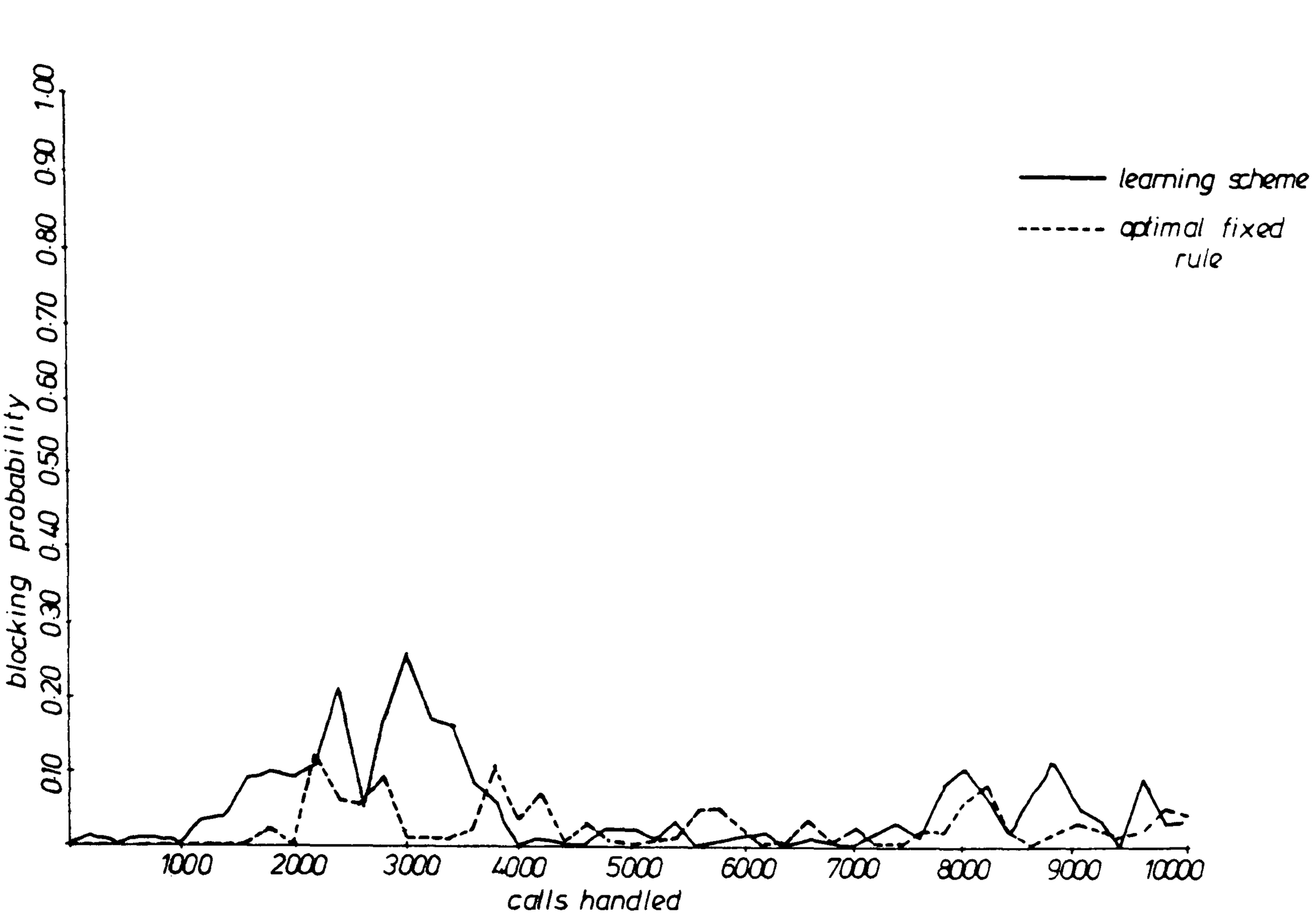
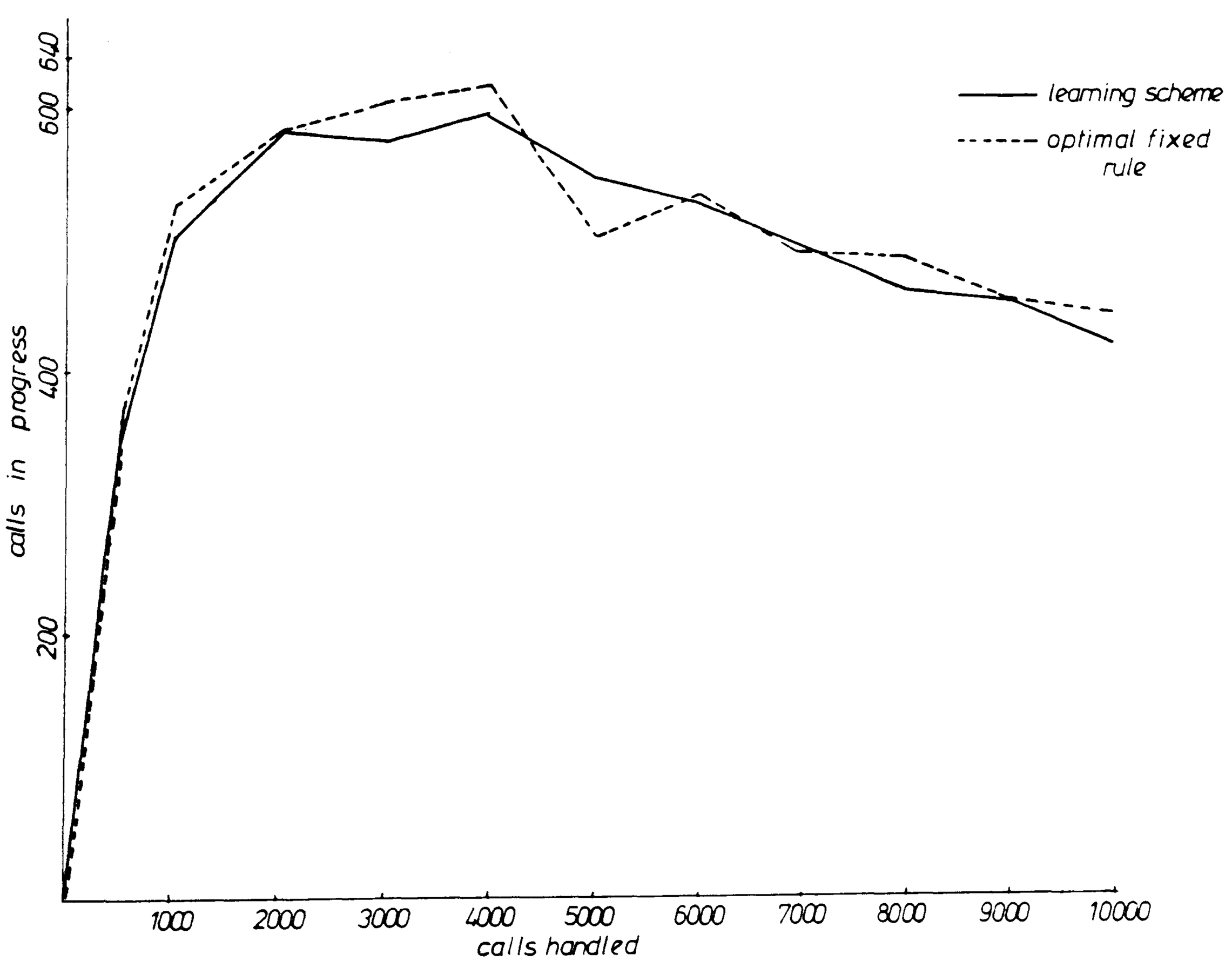


Figure 6·8b Improved policy

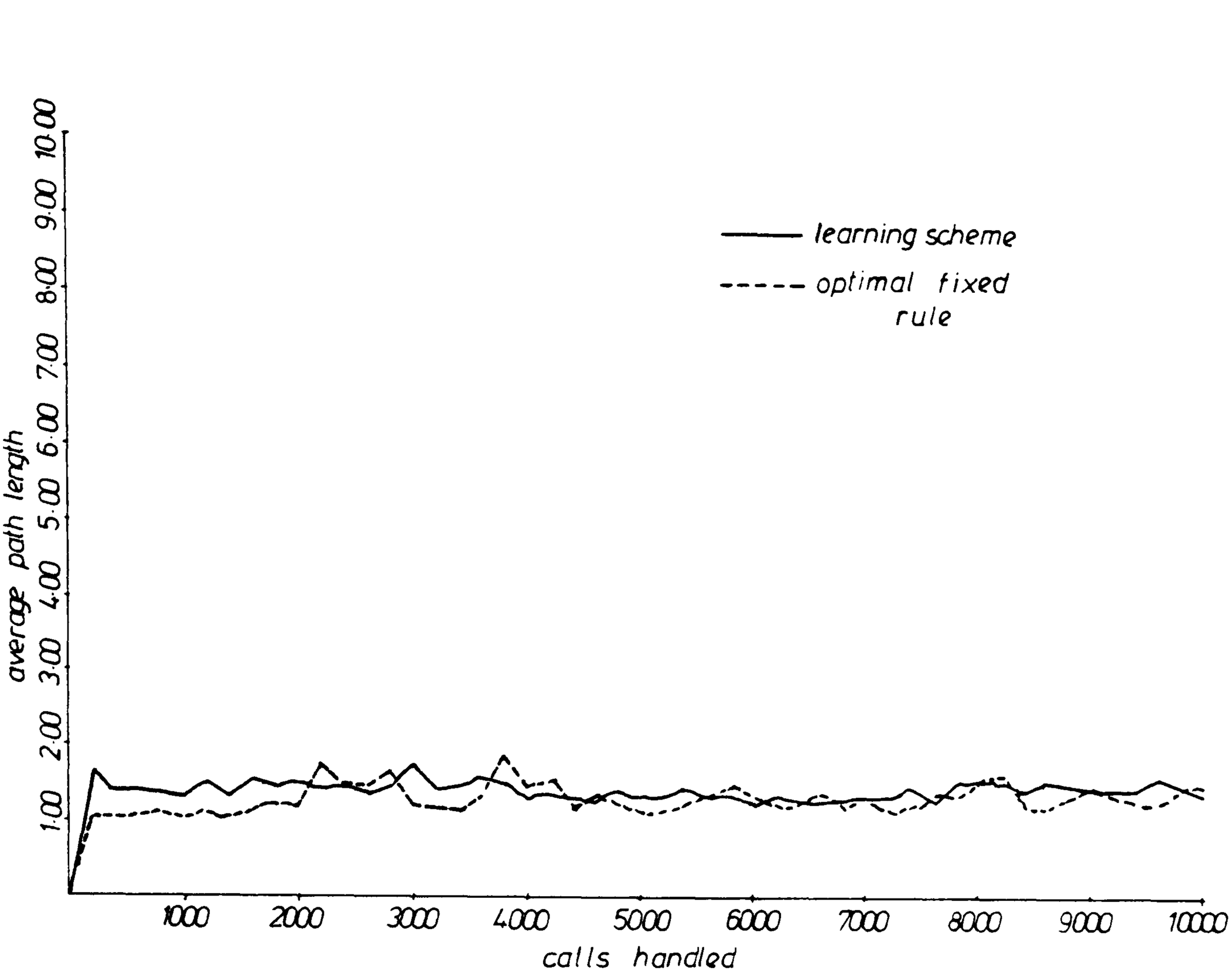


(a) Overall blocking probability

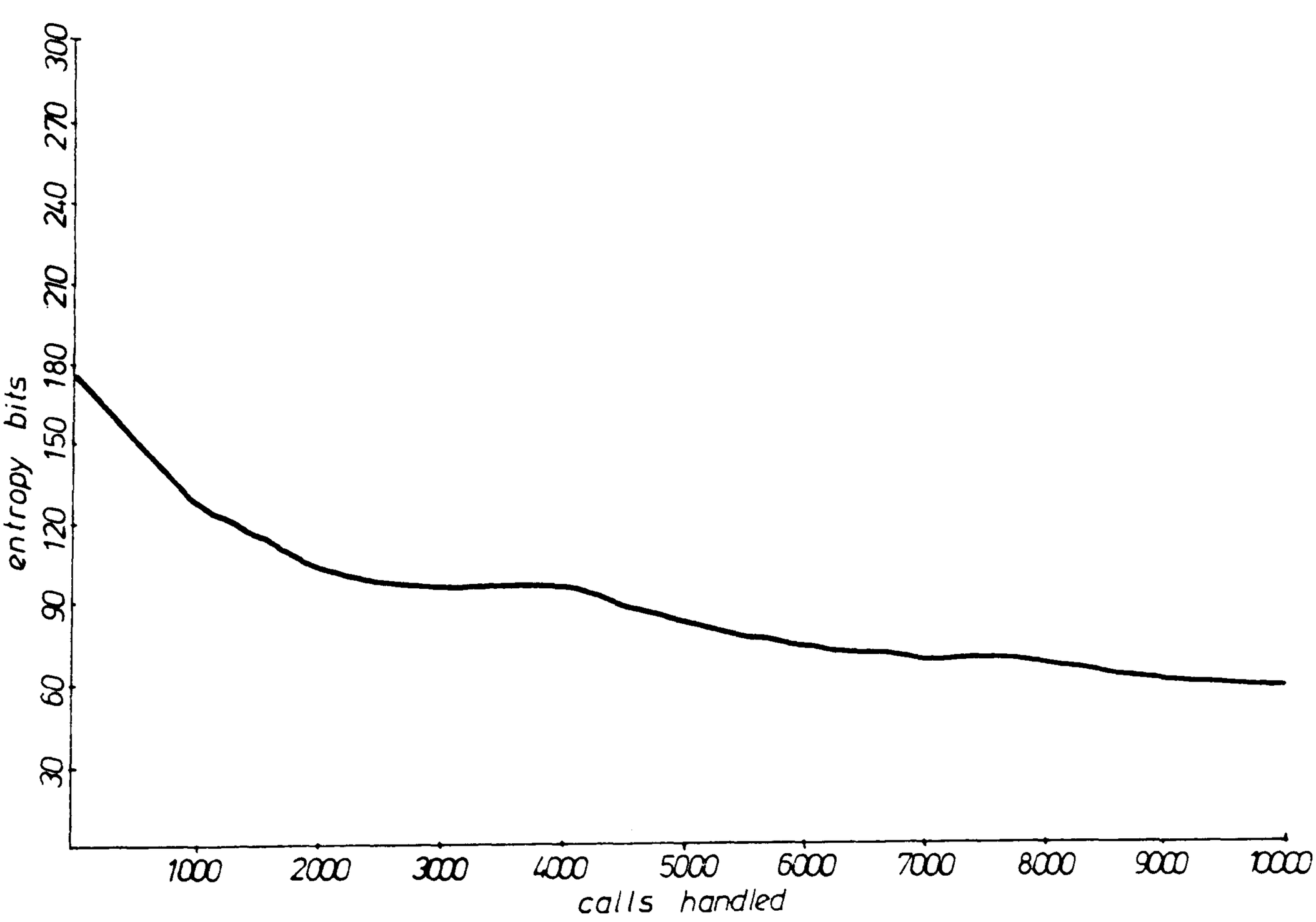


(b) Calls carried

Figure 6·9 Results for start-up bias



(a) Average path length



(b) Routing scheme entropy

Figure 6·10 Results for start-up bias

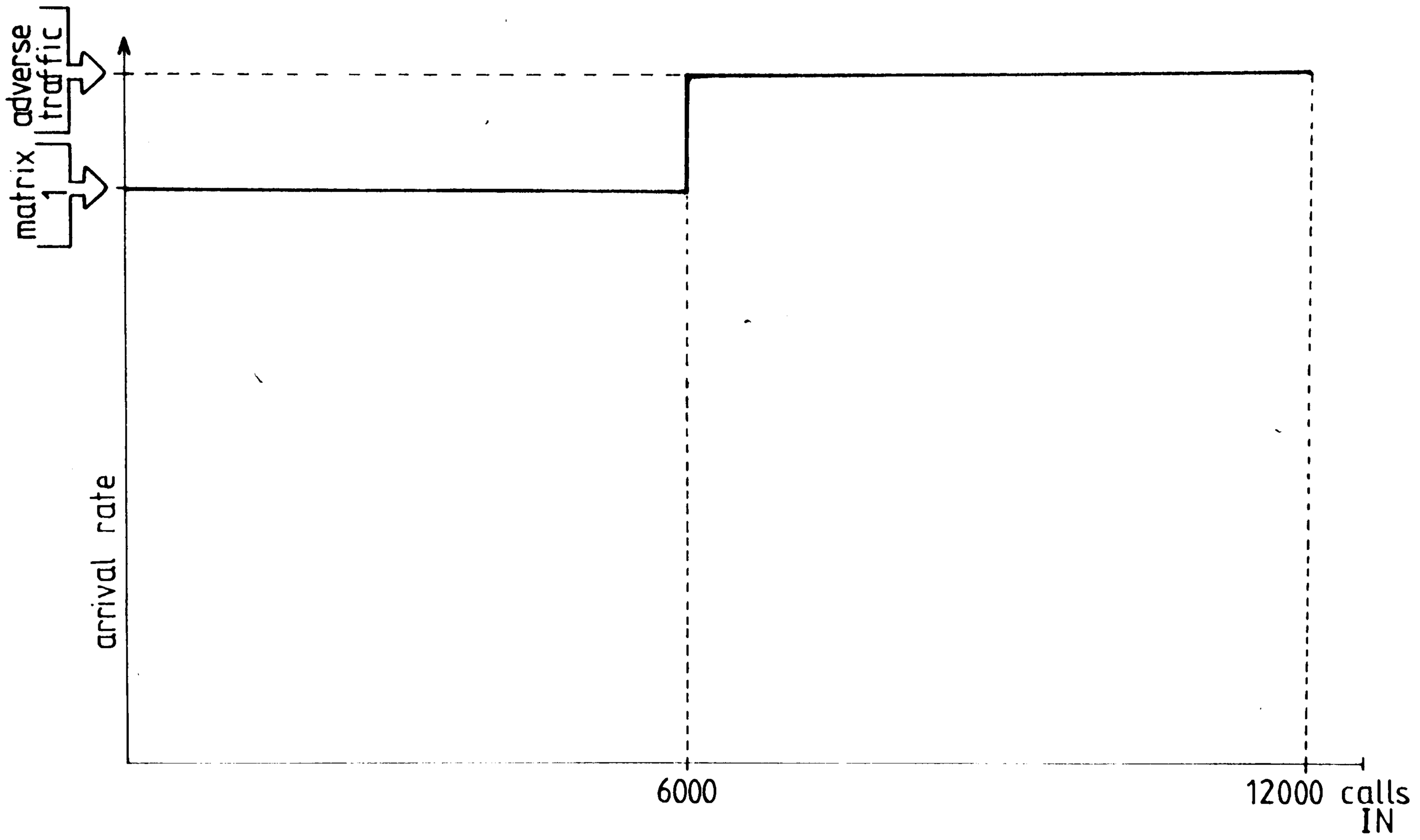
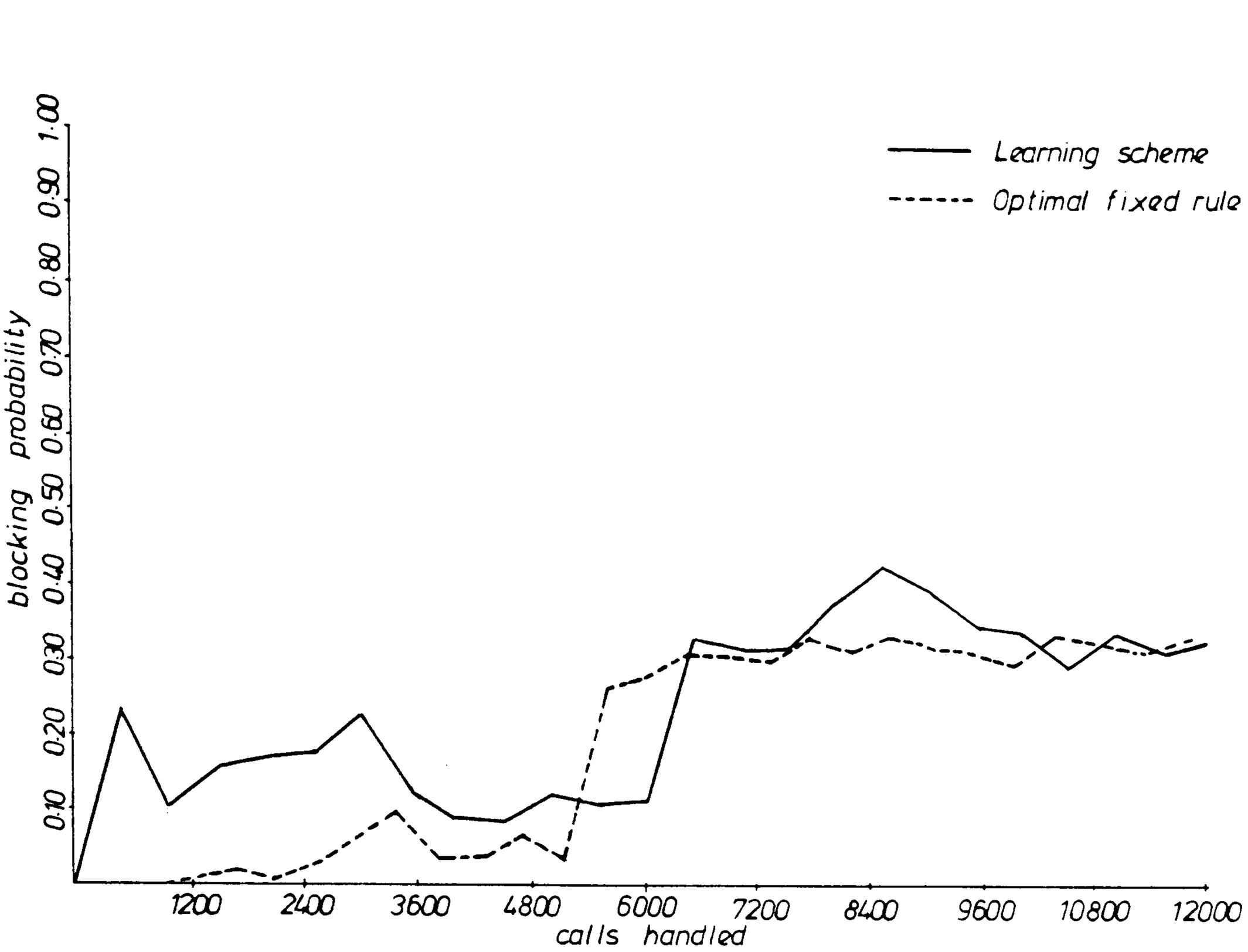
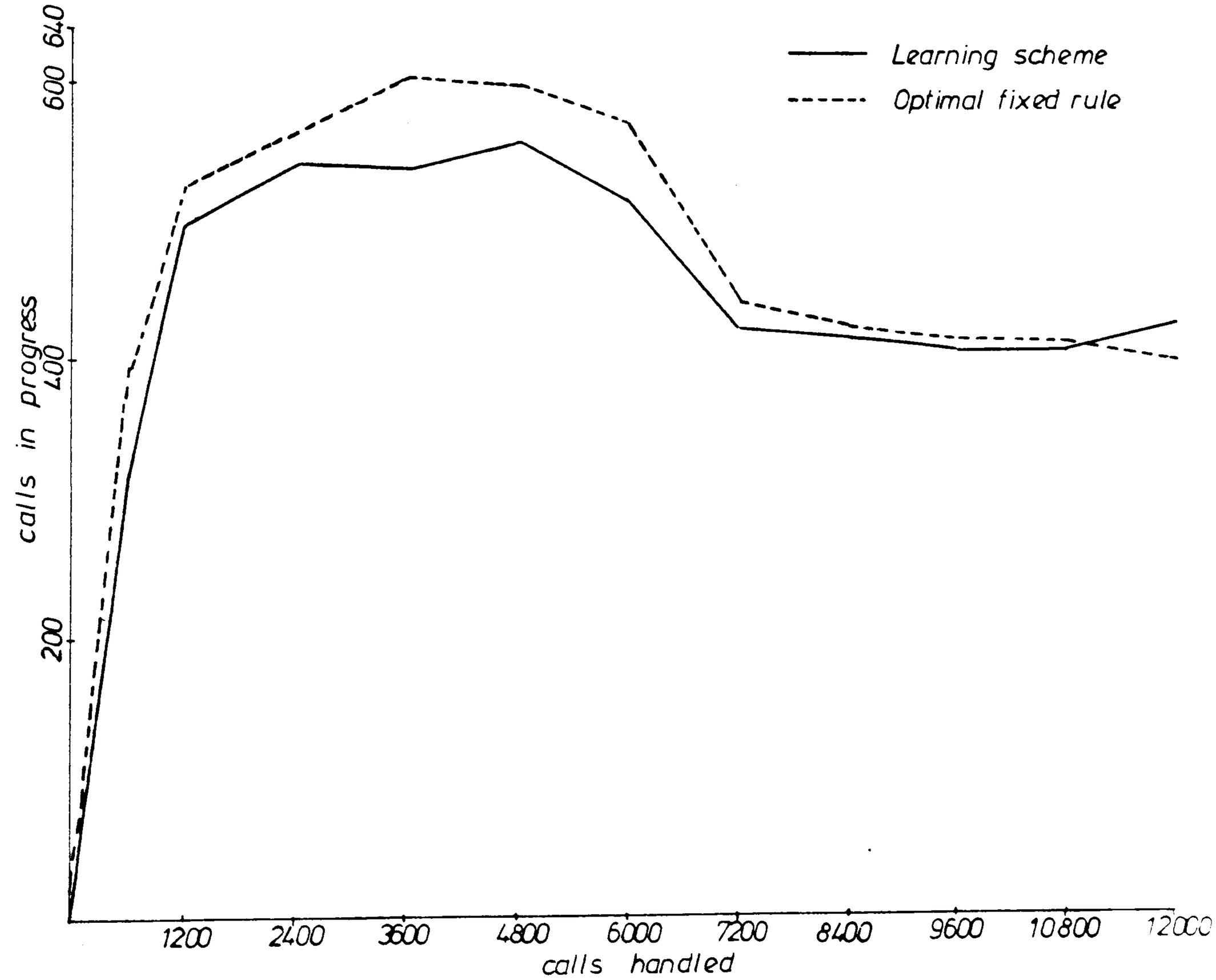


Figure 6.11 Traffic profile for adverse condition experiments

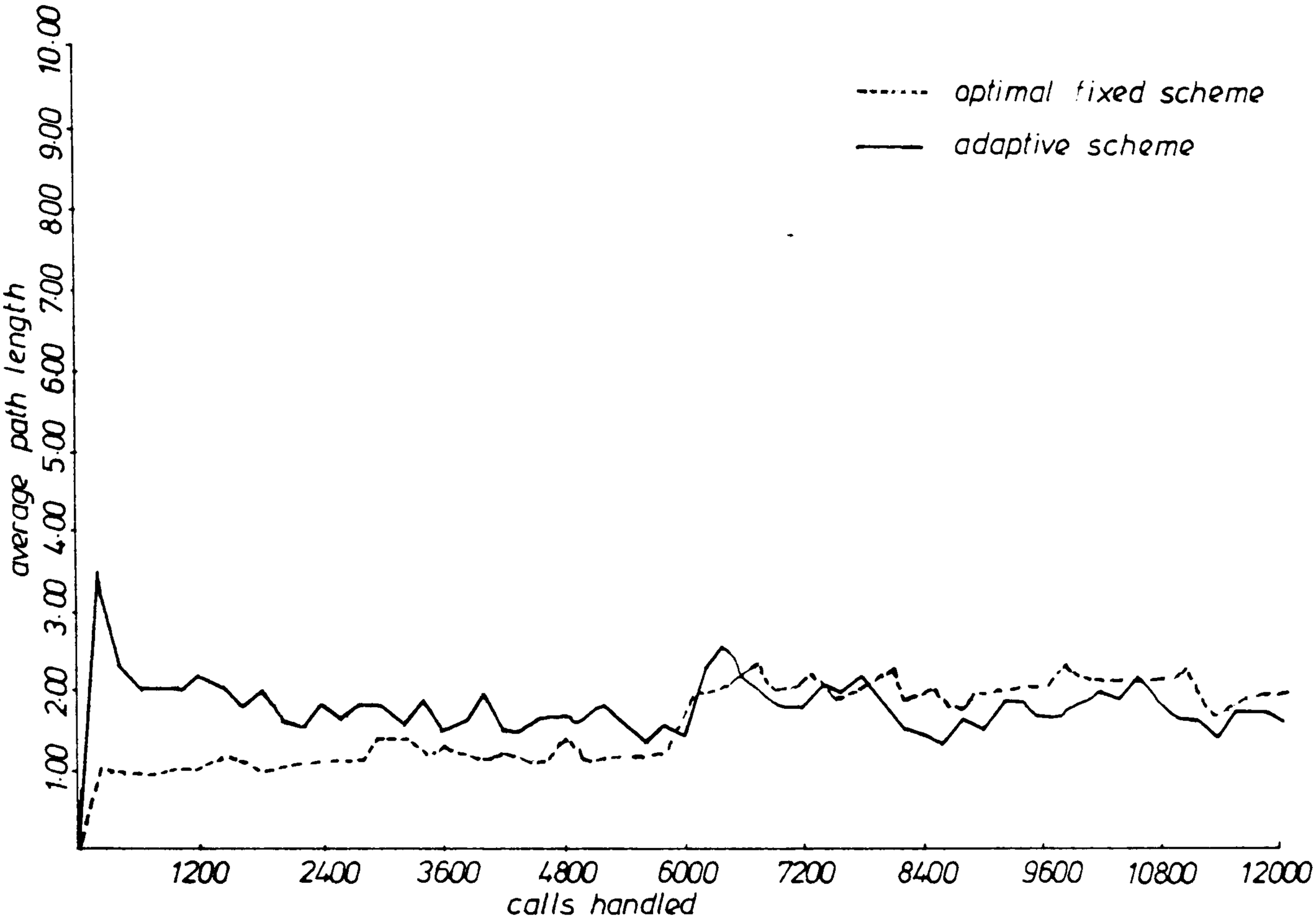


(a) Overall blocking probability

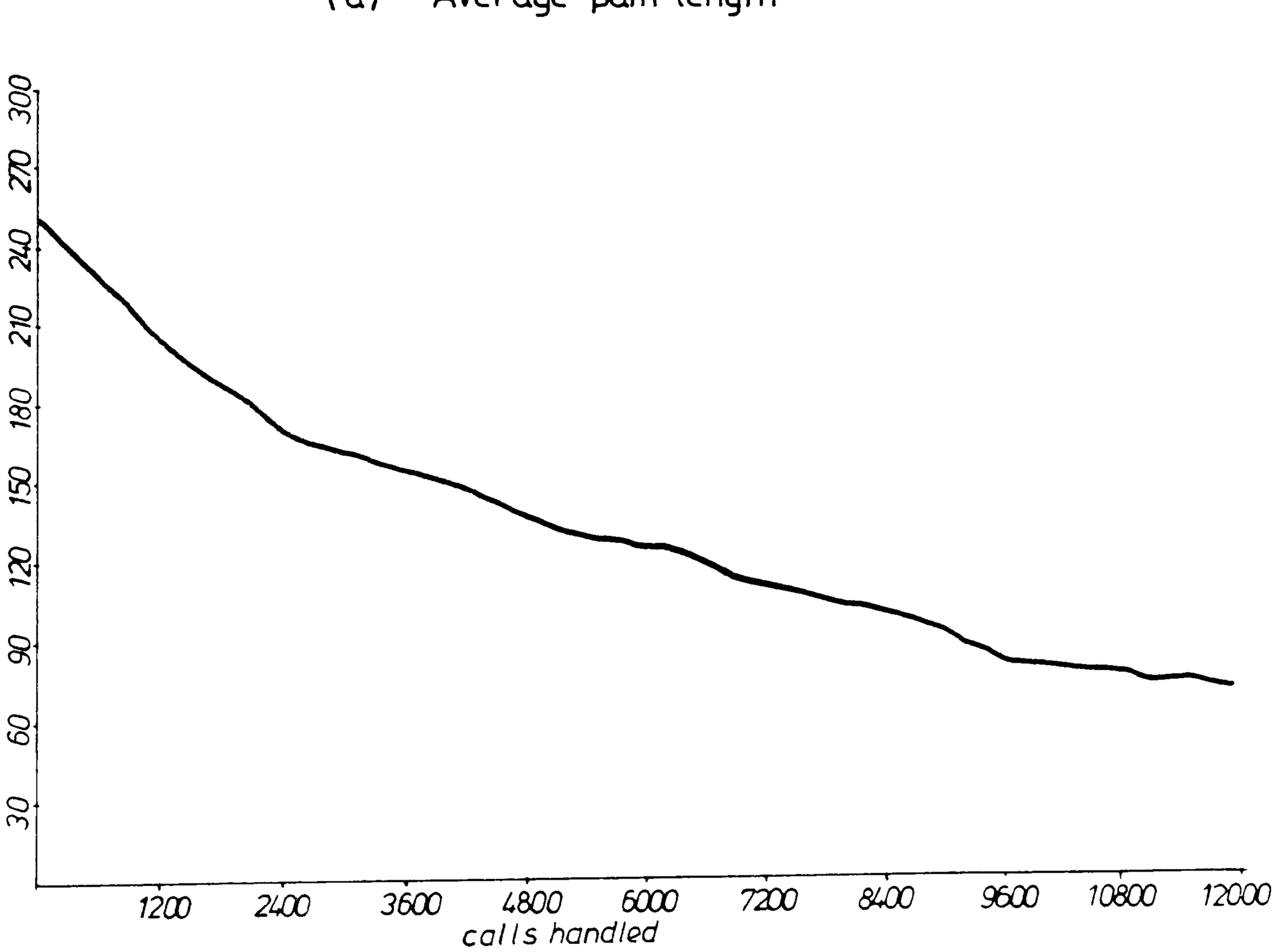


(b) Calls carried

Figure 6·12 Dynamic failure injection results



(a) Average path length



(b) Routing scheme entropy

Figure 6·13 Dynamic failure injection results

CHAPTER 7

STORE AND FORWARD SWITCHED NETWORKS

7.1 Introduction

The transmission of digitally coded information has shown tremendous growth over the last decade and will undoubtedly continue to expand in the future. Primarily, this can be attributed to the growing awareness and rapidly developing use of computer networks which serve to provide distributed computing facilities to a wide range of geographically dispersed users. Also relevant is the growth of digital electronic switching techniques and through these the implementation of more complex network protocols and routing schemes.

A typical computer network (Figure 7.1) is a structured collection of data processing and communications orientated computers, interlinked by high bandwidth links which provide a full duplex transmission capability. Within the complete network, there exists a communication network or subnet, whose function is to facilitate long haul communications for the data processing tasks of the host computers which make up the host subnet. In this study, we are specifically interested in the internal operations of the communications subnet, and can regard the host computers as merely a series of message sources and sinks.

Within the communications subnet, data transmission is handled by the IMPs (Interface Message Processors), which switch messages between links, allowing for a progressive flow between the message source and destination. Also shown on Figure 7.1 is a TIP (Terminal Interface Processor) which in addition to performing the normal functions of an IMP, gives users without a local host, access to the network. One factor which /

which distinguishes the data transmission mechanism of a computer network, from other types, is the unique switching technique. Whereas the telephone network uses line or circuit switching, the computer network operates on a store and forward principle, which experience has shown to be far more effective to handle the high arrival rates and short message lengths, found in digital communications. Under these circumstances, the call set up procedure and resulting delay of the circuit switching proves prohibitive in all but a few computer network applications.

Store and Forward Switching, as the name implies, involves the queueing of messages in network nodes until a suitable outgoing link is free for transmission. When the link does become available the message, or part of it, is transmitted at a rate governed by the capacity of the link, to be received and buffered at the next node. A message traversing the network is therefore seen to hop from node to node (Figure 7.2), the data being buffered at each node, until finally the destination is reached. These networks therefore introduce buffering or queueing delay and thus the message delay or response time plays a critical role in the system design.

In studies of store and forward networks, the performance criterion of fundamental concern in most cases is the mean end to end delay, with respect to the movement of a message from the source node to the destination node. This provides a measure of the average transit time for a message and allows for an assessment of the network response time. Ideally, minimal delay is required and in addition, delay characteristics which show minor variation with increasing traffic. An efficient routing doctrine has influence on this aspect and in networks where traffic and topology /

topology are non-stationary, the provision of a suitable adaptive routing scheme should help achieve the desired response times by making better use of the available capacity.

In the sections that follow, three distinct topics are discussed. First, the nature of store and forward networks is developed and details of the network components, operating protocols and design issues are reviewed. Secondly, an extensive summary of existing adaptive routing techniques is presented, relating the performance and general operating principles of these schemes and suggesting suitable areas for the application of the learning automata. Finally, the technique of modelling store and forward networks by digital computer is discussed, with details of three software simulators, produced for this study.

7.2 Network Components and Protocols

Similar to the circuit switching counterpart, the store and forward network is an interconnected set of switching centres or nodes and communications links, fed by a series of message sources. There are however, significant differences in the nature of the individual components and also in the operating protocols which govern their use.

Store and Forward Node

Conceptually, the store and forward node is a series of input and output queues which can store digital messages. In general an input and output queue per link is provided, which under normal operating procedures function on a first in/first out basis. Switching between queues is performed by a stored program processing element which also carries out additional duties such as routing, link control and error recovery operations. As with the circuit switched node, the element is uniquely identified by a numerical address.

Communication Link /

Communication Link

Unlike the trunk group connecting two circuit switched nodes, the link in a store and forward network consists of only two lines, the link L_{ij} providing a line for transmission between i and j and also a line for transmission between j and i. The operation of L_{ij} is described as full duplex, since both nodes can simultaneously communicate with each other. The capacity of L_{ij}, C_{ij}, specifies the maximum data transmission rate for a message on L_{ij} (in both directions) and is normally dimensioned in units of bits/sec. Neglecting the phase delay of the line, the time required to send a message of k bits is therefore k/C_{ij} seconds.

Traffic Sources

The host computers and network terminals constitute the sources and sinks of message traffic in a network, which are of course random processes. Similar to the circuit switched network, this traffic is characterised by the point to point average message arrival rates λ_{ij} (source i, sink j) and the mean message length $1/\mu_{ij}$ bits/message. In analytic studies the message arrival statistics are normally assumed to be Poisson distributed, with the exponential distribution used to describe the message length process.

Operating Protocols for Store and Forward Networks

The operating protocols in the network are the formal rules by which the communication is governed, the two main types including message switching and an improved variant, packet switching. Essentially, the first networks operated on a message switching protocol, later networks using the more efficient technique of packet switching.

Message Switching

In message switching, the entire message irrespective of /

of length, is transmitted as a complete data block, from node to node in the network. This is demonstrated conceptually on Figure 7.3a, showing a message transmission over three links, the end to end delay for the message related to the sum of the three individual link delays. Assuming Poisson/Exponential statistics, it is possible to analytically specify the average delay experienced by a message on a link, using the M/M/1 queueing arrangement[27]. For a link with capacity C, handling messages arriving at a rate λ with mean length $1/\mu$ the average delay is given by

$$T = \frac{1}{\mu C - \lambda}$$

Studying this in more detail reveals two components, a mean transmission delay and a queueing delay

$$T = \frac{1}{\mu C} + \frac{\lambda/\mu C}{\mu C - \lambda}$$

Packet Switching

The major difference between message and packet switching is the procedure in the latter which decomposes messages into fixed length segments or packets, which can be transmitted as separate units, the original message being reconstituted at the destination. The advantages of such an operation are clearly shown by Figure 7.3b, the pipe-lining of packets reducing the overall message delay. Analysis of the packet delay is unfortunately more involved than the message switching example since the queue experiences Poisson arrival statistics with a constant service rate. The average delay is therefore provided by the M/G/1 queueing theory[27] and gives

T = /

$$T = \frac{2-\lambda/\mu C}{2\mu C-2\lambda}$$

$$= \frac{1}{\mu C} + \frac{1}{2}\,\frac{\lambda/\mu C}{\mu C-\lambda}$$

Further classification of packet switched networks introduces the Datagram and Virtual Call operating protocols, both of which offer a distinctive communication service using packet switching principles.

Datagram Packet Switched Network

In the Datagram network, Figure 7.4a, each packet is treated as an isolated unit and routed accordingly, with the constituent packets of a common message perhaps taking separate paths between the source and destination. This has the desirable effect of spreading traffic throughout the network, utilising the available capacity. It does however, result in packets arriving out of sequence and requires a facility to implement re-ordering.

Virtual Call Packet Switched Network

The operation of a Virtual Call Network, (Figure 7.4b), is analogous with circuit switched networks, where before communication can take place a set-up procedure establishes a direct connection between the information source and sink. Similarly, the virtual call packet switched network operates by forming a logical channel over which a data packet sequence is sent, the switching continuing on a store and forward basis.

In modern computer networks both the datagram and virtual call protocols will find ever increasing use, the datagram type of operation is ideally suited to networks which carry significant proportions of short, interactive traffic and the virtual call network for situations when large file transfers are /

are required. In this study of learning automata routing methods, it is proposed to investigate schemes which are suitable to both types of network.

Design Issues

With regards to the correct and optimal operation of a store and forward network, three important design issues must be faced, including the problems of routing, flow and congestion control.

Flow Control

The flow control scheme is the mechanism by which a network throttles and regulates the rate that data is transmitted in the network. At a link level, simple flow control techniques such as stop-and-wait and the sliding window[37], prevent nodes transmitting packets at a rate which cannot be handled by the receiving node, which perhaps is inundated by packets from many other nodes. Flow control is also necessary at a higher level to control a source host sending messages to a slower destination which does not have the capability to process the data at the same rate. In these circumstances, the flow control scheme reduces the possibility of a packet back-log developing in the network.

Congestion Control

Congestion control is concerned with the avoidance and recovery from a network state in which the network performance is degraded in some way because of an excessive number of packets in transit within the network. In some circumstances, the congestion may be local, that is confined to a limited region of the network, or more seriously, it may exist throughout the network. Proposed congestion control schemes have included the Isarithmic scheme[38], which operates using a constant stock of permits, which packets must first /

first secure before entering the network. Congestion is thus controlled by restricting the maximum number of packets within the subnet.

It is generally agreed that the flow control mechanisms in a network are not sufficient to prevent congestion, although, poor flow control might be at the root of congestion problems in some networks. Local congestion, as mentioned previously, may be alleviated by an efficient adaptive routing scheme, which on detecting heavy saturation in a particular section, rapidly reacts to utilise alternative paths.

Routing Scheme

The reliability of a store and forward communication network is critically dependent on the behaviour and performance of the routing scheme. In circuit switched networks, the routing scheme operates at the outset of a call, establishing a connection of lines over which a direct communication can take place. Routing in store and forward networks however, represent a different problem, the objective being to ensure the systematic progression of each message from source to destination, aiming to minimise some performance criterion such as the mean end to end delay.

The design of an efficient routing procedure for a network is typically a stochastic resource allocation problem, distributed in time and space. In general, the task is to find a routing structure $R(t)$ which offers the best allocation strategy in a network with facilities $N(t)$ and offered traffic $T(t)$, such as to minimise the overall average delay.

$$R(t) = G\left\{N(t),\ T(t)\right\} \qquad \min \text{ (overall average delay)}$$

Problems are introduced however, by the time variant nature of $N(t)$ and $T(t)$, the network suffering from topological changes due to faults, modifications, etc., and the /

the traffic set exhibiting statistical non-stationarity as a consequence of user trends, e.g., peak and off-peak loads. It is therefore obvious that the optimal routing structure for a realistic network is not fixed, and in order that a satisfactory network performance can be achieved at all times, it is essential that an adaptive routing capable or re-organisation be employed.

Properties of an Ideal Adaptive Routing Scheme

Prior to looking at a range of existing and proposed routing techniques, it may be beneficial to note the nature of an ideal adaptive routing scheme such that comparisons can be drawn. The following properties are considered as desirable in an adaptive scheme.

1 Optimality

2 Stability

3 Simplicity

4 Robustness

An optimal routing scheme is a necessary requirement from the viewpoint of achieving the best performance with the available network facilities given the traffic offered by the message sources. Minimising the mean message delay is an obvious candidate, but so is maximising the total throughput and minimising the worst case delay. Stability considerations are also very important in routing algorithms, since under any conditions, an adaptive scheme must be capable of converging to a steady state and reliable routing structure. It is noted that this has been a serious problem with some algorithms, the unstable behaviour resulting, in certain circumstances, to a complete disruption of the network service.

From an implementation standpoint, the simplicity of a /

a routing scheme is a crucial issue. This is particularly so when the processing power at the nodes is at a premium, and consequently it is undesirable to produce schemes involving lengthy computations and large storage requirements. Also, since adaptive routing requires some form of feedback from the network, a good implementation should attempt to utilise existing control traffic or, at least, introduce as little as possible additional traffic for this operation. A robust routing doctrine suggests a scheme which can satisfactorily contend with instantaneous changes in the network topology, coping with nodes dropping out and coming into service.

7.3 Review of Existing Routing Techniques

Since the origin of the first store and forward networks, many suggested routing methods have been presented, some following a practical solution to the problem, others looking theoretically at the mathematics of finding the true optimum. This section aims to classify the general types of routing scheme, using examples of many previously proposed and implemented techniques. It is also intended to show in this classification, the nature of the learning automata scheme in relation to the others, pointing out the advantages of this new approach.

In a general classification of routing schemes for store and forward networks, Figure 7.5 forms a useful framework for discussion, primarily suggesting the option of invariant or adaptive routing. As implied, invariant routing schemes possess no ability to react according to network conditions and to re-organise the routing structure. This arrangement is not entirely staisfactory in a practical network operating under non-stationary conditions, but the simplicity and underlying concepts warrant the description of a few schemes.

Of /

Of the simple, invariant schemes several consistently appear in the literature[39][40], including fixed routing, random routing and flooding.

Invariant Routing Schemes

Fixed Routing

In fixed routing schemes, (Figure 7.6a), tables or directories are furnished at each node, containing information on one particular line to be used for each possible destination. To successfully design such a scheme involves an off-line optimisation technique, which given the topological parameters and estimates of the traffic, produces the required directory entries. Normally, these form the shortest path routes i.e., the single path which has the minimum end to end delay for a particular source/destination pair.

Random Routing

As the name suggests, random routing, (Figure 7.6b), selects an outgoing link on a probabilistic basis. In some schemes any of the outgoing links can be utilised, each being chosen with equal probability. Other implementations of this technique might consider directories of routing probabilities at a particular node, (organised in a similar structure to the fixed rule) where the probabilities are not equal, the distribution biased in favour of the better routes. Operating in this manner, the random routing approach offers a flow splitting or traffic bifurcation scheme[41] and improves the transit of messages by utilising multiple paths which spreads the traffic throughout the network.

Flooding

Flooding (Figure 7.6c) is a routing technique employing the transmission of duplicate copies of a message on all available links (discounting the link on which the message arrived) /

arrived) and provided the destination is reachable, a copy is always delivered. Unfortunately, this is only achieved at the expense of heavy traffic overheads, caused by the generation of redundant traffic which can impose a considerable load on the network.

Invariant routing schemes have been implemented in several networks, notably the SITA network[19] where an enhanced form of fixed routing allows for network failures by specifying a secondary link in the routing tables. Also, the load splitting behaviour of the random routing scheme is important, since this forms the foundation for the learning automata schemes to be described later.

Adaptive Routing Schemes

In contrast to the invariant routing techniques, adaptive routing schemes have the ability to suitably modify the routing structure, to satisfy the network and traffic conditions, normally attempting to minimise the overall average delay. In the type of network under investigation, the adaptive routing strategies can be broadly classified (Figure 7.5) into four main doctrines, which are as follows:

1 Isolated
2 Centralised
3 Distributed
4 Hybrid

Isolated Adaptive Routing

From a design and implementation point of view, the simplest class of adaptive routing involves an isolated doctrine where routing decisions are taken only on the basis of information available locally at each node. Clearly, a technique of this nature cannot be globally optimal since the limited feedback policy pays no regard to the global status of the /

the network. Any feedback however is readily accessible and constitutes the most recent state network. To illustrate the principles of isolated routing, two simple schemes are described; Hot Potato Routing[40] and Shortest Queue + Bias[42].

Hot Potato Routing

The term Hot Potato admirably describes the operation of an isolated routing technique where each node attempts to dispose of a packet at the earliest opportunity, by first checking for an empty queue and initiating a transmission on the associated link. Such a scheme, although desirably simple, suffers from "leap frog" problems near to the proposed destination, where packets tend to be shunted backwards and forwards without reaching the desired destination node (Figure 7.7a).

Shortest Queue + Bias

An improved variation of hot potato routing is the SQ+B scheme, where in addition to the queue length, a bias parameter also influences the choice of link for sending a packet. This is illustrated on Figure 7.7b by a specimen node with three possible outgoing links, these designated the primary, secondary, etc., and allocated as appropriate bias coefficient. Used in conjunction with the available output queue space, the coefficients determine the proportion of traffic to be routed on each link, e.g., 5+4: 5+3: 3+1. In the situation of a queue being full, the next best alternative is selected, and if all are saturated, the message is deleted, to be retransmitted by the error recovery mechanism of the previous node.

Obviously, the hot potato and SQ+B routing techniques are severly limited by the myopic feedback policy, which reacts to a down stream disturbance, only after a chain of full /

full queues have formed between the disruption and a node where an alternative routing decision can be made. One suggestion to improve on this lack of global feedback, is the SQ+B+PU (Shortest Queue + Bias + Periodic Update), where in addition to the fixed bias, a central routing facility provides a variable parameter to modify the nodal routing decisions, reflecting the overall condition of the network.

Centralised Adaptive Routing

Centralised routing methods (Figure 7.8a) rely on individual nodes reporting the local network conditions in their own immediate area to some central processing facility, commonly referred to as the Network Routing Centre (NRC). On assembly of the global state, optimal routing tables are formulated from the solution of multi-commodity flow algorithms and transmitted for use by the individual nodes, these remaining the sole routing policy until the next update is received.

With the opportunity to utilise formal optimisation techniques, the centralised doctrine should in principle offer an efficient routing strategy. However, the inherent delay between nodes reporting the local status, the NRC computing the optimal routing strategy and the subsequent distribution of the routing tables curtails the advantages of such a scheme, this delay even introducing the possibility of unstable behaviour. A further drawback with the centralised routing discipline, is the reliance on a single computing facility, which from a reliability viewpoint is highly undesirable. Also, the need for additional control traffic to transport status reports and fresh routing information is inconvenient, the updating cycles causing periodical bursts of heavy traffic, particularly in the immediate area of the NRC.

Despite the problems of centralisation, one suggested scheme /

scheme deserves mention since it adopts a simple and practical approach to the problem.

Proportional Routing

Proportional routing has been suggested as a centralised routing technique, which although never implemented, has been throughly investigated[45]. With proportional routing, all links to directly neighbouring nodes receive a given percentage of the traffic, (as in random routing), based in proportion to the delays experienced on the various routes. This is shown with the assistance of the simple network on Figure 7.9a, and even consisting of only two paths, demonstrates the underlying principles.

In this example, the random routing mechanism at node S periodically receives the probabilities p_1 and p_2, these determining the proportion of traffic to paths 1 and 2 respectively. The NRC (shown conceptually on the diagram) periodically collects and evaluates the average path delays T_1 and T_2 and subsequently produces modified routing probabilities with the following algorithm.

$$T_r = T_2 / T_1 \qquad 0<\theta_s, \theta_l<1$$

if

$T_r > 2$	$p_1 = p_1 + \theta_l$, $p_2 = p_2 - \theta_l$
$1 < T_r < 2$	$p_1 = p_1 + \theta_s$, $p_2 = p_2 - \theta_s$
$T_r = 1$	$p_1 = p_1$, $p_2 = p_2$
$\frac{1}{2} < T_r < 1$	$p_1 = p_1 - \theta_s$, $p_2 = p_2 + \theta_s$
$T_r < \frac{1}{2}$	$p_1 = p_1 - \theta_l$, $p_2 = p_2 + \theta_l$

In the more involved environment of a practical network (Figure 7.9b), each node is responsible for dispatching status reports concerning the link delays in the local area. From this information the NRC collates the path delays for all source destination pairs and calculates the routing probabilities with /

with the following simple algorithm.

$$p_j^i = \frac{1}{T_i \sum_{k=1}^{n} 1/T_k}$$

Despite the simple approach of proportional routing, it still however incurs the overheads of centralised routing, which in comparison with the distributed routing techniques to be described, are unacceptable in most cases.

Hybrid Adaptive Routing

Hybrid routing schemes attempt to combine the virtues of the Isolated, Centralised and Distributed routing doctrines, whilst avoiding their faults. As of yet, only one significant scheme of this doctrine has been proposed, delta routing, although the fundemental principles of hybrid routing deserve further study.

Delta Routing

Delta routing[46] is a hybrid scheme, integrating the functions of isolated and centralised routing, to provide an algorithm which offers global optimisation while still maintaining the ability to react to minor local disturbances. Centralised adaptive routing, with a network wide overview of the network state, can prepare a global strategy with conventional optimisation techniques. Unfortunately this incurs a delay penalty which reduces the responsiveness of the scheme. On the otherhand, the isolated form of routing allows a node to react rapidly to changes in the local environment, but is lacking in global awareness of the network, a vital necessity for true optimum routing. On this basis, delta routing implements global and local decisions in such a manner as to achieve a consonant global strategy, but also giving individual nodes freedom to respond to instantaneous changes in their local environment.

In /

In delta routing, the centralised component of the overall scheme keeps track of the global situation, while on a local level, the isolated portion allows nodes to carry out updating from the immediate delays experienced by packets. Furthermore, the scheme provides a facility to delegate varying degrees of authority to either the centralised or isolated components, through a single parameter δ. With this arrangement, a completely centralised scheme results from a low value of δ, a high δ allowing the isolated doctrine to dominate and for the most effective routing, an intermediate value, to produce hybrid routing which can out-perform either of the two extremes.

Although Delta routing has experimentally been shown to provide a worthwhile routing service, it does however rely on a centralised routing, which as discussed before is not entirely satisfactory. Despite this, the general approach of hybrid routing is promising, and if exploited using perhaps distributed and isolated routing, could produce an agreeable scheme.

Distributed Adaptive Routing

The nature of a distributed routing scheme is more suitable for service in a dynamic environment such as a computer network and consequently several algorithms have been installed in networks, including the extensively reported ARPANET. In general, distributed schemes, Figure 7.8b form estimates of the complete path delays, through nodes exchanging partially complete delay information, constructing estimates for the network global delays and with locally implemented algorithms, produce optimal routing tables. Unlike the centralised technique, no node has a master role in the system and therefore any node can discontinue operations without causing a catastrophic effect on the rest of the network.

In /

In this review of distributed routing schemes, a distinction (Figure 7.5) is drawn between algorithms which provide an optimal single path and load splitting schemes which operate to proportion the traffic optimally over several paths. For the first category, the operation of Backward Learning and the ARPANET schemes are detailed as examples, the second group containing a method by Gallager which is a distributed implementation similar to the Flow Deviation method[48] and modifies the flow assignments to find the optimum. The learning automata routing also may be included as a load splitting scheme, although unlike Gallagers technique, updating occurs asynchronously, on a packet by packet basis.

Backward Learning

An early distributed routing technique is Backward Learning[43], where a node attempts to gather global network information from messages arriving from the remaining nodes. One simple method of implementing this, is to include the identity of the source node in each packet, together with a counter that is incremented on each hop of the message. If, for example, a node j finds a packet arriving on link k, from a source node i with hop count n, node j can record that node i cannot be more than n hops, via link k. Also, if n is less than the previously measured minimum distance, link k may be marked as the choice for traffic at j, destined to i. After a while, every node in the network discovers the shortest path to every other node.

One major problem exists with this scheme, caused by the nature of the scheme to react only to "good news", i.e., information suggesting a quicker route to some point in the network, unfortunately leaving problems such as link and node failures unrecorded. This can partially be overcome, by /

by nodes periodically neglecting the converged state, to re-initiate the learning operation, again introducing a transient which results in a spell of sub-optimal routing.

ARPANET Routing Schemes

An excellent paper[41] has described the first ARPANET routing algorithm design philosophy and implementation, from original conception to subsequent modifications. The algorithm, which operates under a synchronous updating policy, requires nodes to transmit, every $\frac{2}{3}$ of a second, minimum delay tables to their immediate neighbours, relaying information on the experienced delay to all destinations. As these tables percolate through the network, individual nodes add their local contribution and form routing tables which direct packets along paths for which the total estimated transit time is smallest.

Take for example Figure 7.10, where node N periodically receives delay tables from nodes A, B and C with estimates of the global delay to all destinations. From local measurements, node N also has knowledge of the link delays between N and the neighbouring nodes and consequently the global delay between N and all destinations can be computed for each outgoing link. Using this data, N forms two data sets, the minimum delays to all other nodes (for transmission to A, B and C) and a routing table containing the best next nodes for each destination.

Since the initial implementation of this scheme, several modifications have been proposed for performance improvements. Early stability problems, associated with the algorithm converging on erroneous paths, have been overcome through the use of'hold down". In this strategy, the algorithm is tuned to react quickly to "good news", but more cautiously to "bad news" such as nodes reporting link failures. This modification /

modification appears to have corrected any unstable behaviour but unfortunately degrades the speed of reaction to network disturbances. Although never implemented, a promising approach of asynchronously updating the nodes has been suggested[50], where information is only exchanged when significant changes in network conditions are experienced.

Within the last three years, an entirely new routing algorithm has replaced the original and is regarded as an improvement since it uses less network resources and is not prone to unstable behaviour. In this new scheme[51], each node maintains a data base describing the network topology and link delays, typically in a sparse matrix form. From this information, the desired routing tables are independently computed at each node using a shortest path algorithm. Updating operations also follow a different approach to the original algorithm. Whereas in the previous scheme, each node reported, the estimated delay between itself and all destinations, only to immediate neighbours, the new scheme requires nodes to transmit updates which contain local link delays to all nodes.

Figure 7.11 attempts to briefly demonstrate the technique, the shortest path algorithm at node N, evaluating the data in [N] to form a routing table. The diagram also indicates how [N] is periodically revised, with all other nodes transmitting link delays and in addition illustrates node N contributing to the other nodes with the delays on L_{nq}, L_{nb} and L_{nc}.

Gallagers Algorithm

Both ARPANET algorithms have been implemented, and operated in a major computer network for a number of years. Neither however, produce a truly optimal performance, /

performance, since only a single path is formed for each source/destination pair. An optimal routing service can only be achieved with a load splitting algorithm which proportions the traffic over several paths.

A distributed load splitting algorithm has been suggested in[52], which although only described and studied theoretically, proposes a highly desirable form of adaptive routing. The scheme breaks into two parts; a protocol between nodes to calculate and exchange marginal delays and an algorithm for modifying the routing probabilities. The marginal delays are the first derivatives of the path delays with respect to the offered packet flow and can be physically regarded as a "resistance" to further flow. The algorithm for updating is as follows.

$$\varphi_{ik}(j) = \varphi_{ik}(j) - \eta \Delta_{ik}(j) \qquad k \neq k_{min}$$

$$\varphi_{ik}(j) = \varphi_{ik}(j) + \eta \sum_{k \neq k_{min}} \Delta_{ik}(j) \qquad k = k_{min}$$

where

$$\Delta_{ik}(j) = D'_{ik} + \delta D_t / \delta r_k(j) - \min\left[D'_{ik} + \delta D_t / \delta r_k \right]$$

where D'_{ik} and $\delta D_t / \delta r_k$ are the marginal delays due to the traffic on link i,k and the external traffic at node k, $\varphi_{ik}(j)$ is the probability that k is selected as a next node from i and k is the next node on the path with the minimum marginal delay. The scale factor η alters the rate of convergence, small η guaranteeing convergence, but rather slowly. Increased η improves the speed of convergence, however introducing the danger of no convergence.

Learning Automata Routing

The learning automata approach to adaptive routing may also be described as a distributed load splitting technique, but unlike the previous algorithm a simpler and more practical updating /

updating process is involved. Moreover, updating occurs asynchronously, as individual packets are routed and hence dispenses with a need for periodical exchanges of information between all nodes. To further detail the advantages of learning automata routing, the following points are noted:-

1 Updating is implemented on the basis of measured delay, as opposed to the marginal delay d' of the previous scheme.

2 The multi-modal search capability of the automata will find the global optimum, even when several local optima exist.

3 The range of reinforcement algorithms offers alternative convergence strategies, usually by simple modification of the learning parameters "a" and "b".

4 No assumptions need be made concerning the statistics of the traffic.

7.4 Simulation of Store and Forward Networks

This section describes the general simulation technique and program structure of a detailed store and forward simulator which has been produced to carry out a range of experiments involving alternative routing techniques. In all, the package permits the simulation of the three important network types, including message switching, datagram packet switching and a virtual call variant with full end to end control.

Prior to describing the software implementation, the basic network and nodal models are presented, leading to a description of the overall simulation package, constituent programs /

programs and the data structures around which the simulation is formed. As with the earlier circuit switched simulators, extensive use has been made of ring data structure concepts, which are particularly suitable to support the operations required by network simulators. Also, the package has been prepared to run on a mainframe computer using an interactive graphics terminal for efficient input and output of data.

Network Model

In a store and forward network, digital data (messages) are transmitted from node to node, with messages queueing for service by the links which interconnect the various regions of the network. Conceptually a network of this type may be regarded as a collection of interactive queues, structured according to the topology of the network and fed by traffic sources which generate randomly arriving messages of varying length. The nodes typically consist of a memory allocation to act as message queues and a processing element for control, details of which are provided below. The network links (Figure 7.12a) are full duplex in operation, consisting of two lines each capable of supporting transmissions from both ends of the link. For the purposes of traffic generation, host computers assumed to act as message sources, randomly producing arrivals at each node, destined to all remaining nodes.

Nodal Model

For this study, a simple node is proposed (Figure 7.12b), consisting essentially of a common input queue, output queues for each outgoing link and a rudimentary data processing element responsible for link protocol and routing. All queues operate on a FIFO (First In/First Out) basis and it /

it is assumed that no queue jumping or message pre-emption can take place. The operation of this node is as follows. Messages arriving at the node are buffered in the input queue until service can be provided by the processor. Typically, this consists of a message check, effectively checking if the desired destination has been reached and the execution of the routing algorithm to select an appropriate outgoing link. If required, any updating of the adaptive routing scheme can be implemented, the operation being initiated on reception of the delay feedback. Routed messages are queued for transmission in the output buffer of the selected link, to be dispatched when the link is free.

Simulation Package Overview

For the task of simulating general store and forward networks, the standard three program package has been developed to implement the proposed model. This includes a network specification program, a simulation program (three versions) and an analysis program to calculate and display results from the basic data assembled during the experiment.

The network specification program for the store and forward simulation package is effectively a modified copy of the original circuit switched program which used the storage interactive graphics terminal as an input/output device. There are however major differences to be found in the topological data structure produced, but full facilities are again provided to handle a variety of network sizes of any topology.

Topological Data Structure

The data structure produced by the specification program represents the network under investigation and provides a framework for the simulation using the topological information and /

and storage allocations compiled during the construction phase. For this application, the data is stored in blocks or beads, logically connected through a pointer arrangement to form continuous rings and paths, representative of the network structure.

Figure 7.13 shows a schematic diagram of the entire data structure, illustrating the various data beads and connecting pointers. Essentially, three main rings connect the network components, IMP ring, Link ring, and the Host ring. In a hierarchical fashion each IMP bead forms a head for the routing and output queue rings, these connecting the routing data and output queue elements of the node. Further direct pointers provide a means for the simulation software to move messages through the data structure, mimicking the motion of messages in a real network.

Simulation Program

All three versions of the simulation program operate on an event to event principle, which basically functions by establishing the order in which events will occur, processing the nearest event and advancing time asynchronously. In the actual implementation, the majority of software is common, the different operating protocols required by the message and packet switching, detailing individual changes to the program sections which emulate the operation of the network nodes.

The overall structure of a simulation program operates on a simple cycle. After program initialisation (open files, set up variables etc.,) a stock of first arrival times is generated for all message sources (hosts). This primes the network and allows the normal processing to commence, searching for the closest event, handling of the event, carrying out any outstanding operations at a node and finally /

finally checking if the required termination conditions have been met.

Event Handling

In a discrete event simulation, an event is an incident or occurrence of some critical aspect of the system operation. In our network simulation, three events are defined.

1 Message Birth

2 Message transmission from a host to an IMP

3 Message transmission from an IMP to an IMP

On the birth of a new message, the program is required to generate a random message length and to initiate the transmission of the data from the host to the local IMP. Also at this stage, it is necessary to produce a random number, representative of the next message arrival from that source. The host to IMP event occurs when a message enters the network from a host. If, however, the transmission has not resulted in a successful transfer, perhaps the data being corrupted or blocked by a full input queue, a re-transmission can be performed by the host. Likewise the third event caters for message transmission between IMPs with a similar recovery mechanism for the re-transmission of messages.

Nodal Algorithm

The nodal algorithm describes the behaviour of the network IMP and generally consists of an input queue process and an output queue process. In the sequential environment of a general purpose computer this operates on a round robin basis, executing the input process followed by the output queue process, effectively removing newly arrived messages from the input queue, selecting a suitable outgoing link (if required) and handling the transmission of any messages which find /

find a free link. Details of the input queue process vary according to the operating protocols implemented, although the message switched version is used as an example. The alternatives are described in[53].

The input queue process for the message switched protocol (Figure 7.14a) primarily checks if anything has arrived at the node since the last operation and continues to check if the message has reached the desired destination. If not, the routing procedure selects an outgoing link and locates the appropriate output queue where the message is inserted in the first free location. If the destination has been reached, the delay experienced by the message is noted and fedback to update the appropriate routing controllers. The output queue process (Figure 7.14b) is common to all three simulation programs and simply tests if a link is free for transmission and if so, removes a queued message and places it on the link.

Log Data Structure

The log data structure is produced as a simulation progresses and maintains details of individual messages as they traverse the network, providing a permanent record of all messages. For the message switched protocol, (Figure 7.15a), this consists of a single pointer ring connecting message beads, each containing details of a single message. Messages entering the network join the ring in sequence and on completion, the ring structure can be used to determine the simulation order. As shown by Figure 7.15b, the packet switching simulators also consists of a ring of message beads, but in addition each message further consists of a packet ring, logically connecting the constituent packets of the message.

7.5 Summary /

7.5 Summary

The store and forward network now plays a prominent role in data communication systems and is very much an active area for research. In particular, the design and analysis of adaptive routing schemes has proved to be a challenging and important subject, and has in the past decade generated considerable interest.

This chapter has served to provide a general introduction to store and forward networks, discussing in basic terms the switching operation, network components and operating protocols. An additional section has reviewed existing work on routing schemes, classifying algorithms into suitable categories and demonstrating their general operation. Distributed adaptive routing schemes have been suggested as being particularly suited to the computer network, where the topology can change several times in the course of a working period, with significant shifts in the traffic patterns. In discussing distributed routing a distinction has been drawn between schemes which route packets over single paths and others which split traffic over several, adaptively proportioning the flows to achieve a global optimum.

A comprehensive simulation package for carrying out experimental studies on store and forward networks has been implemented. This package includes specific programs to follow the operation of message switched, datagram and virtual call packet switched networks. Since initial design, these have been exercised and tested on a wide range of networks and have been shown to be clearly adequate for the task.

Typical Computer Network

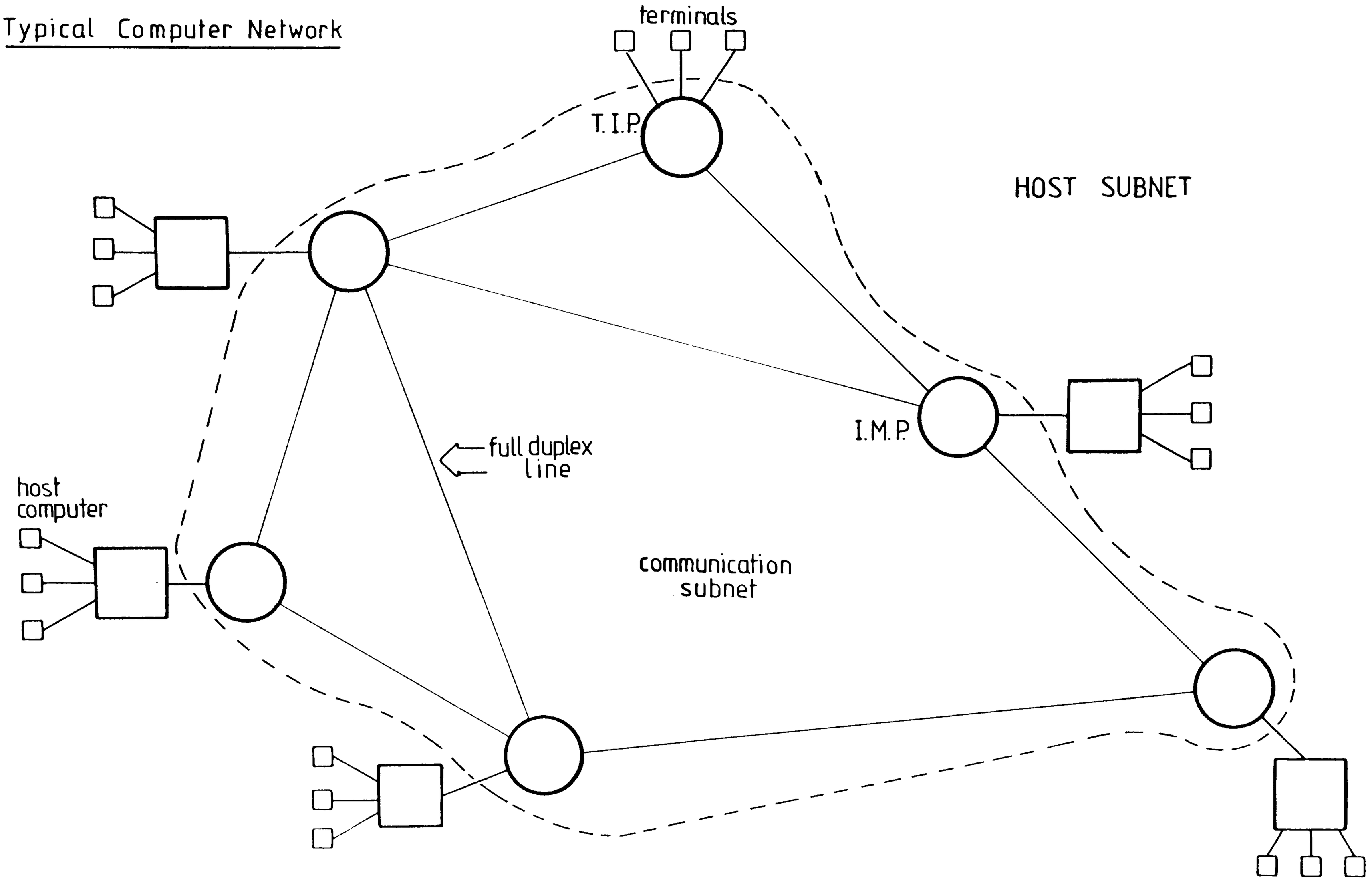


Figure 7.1

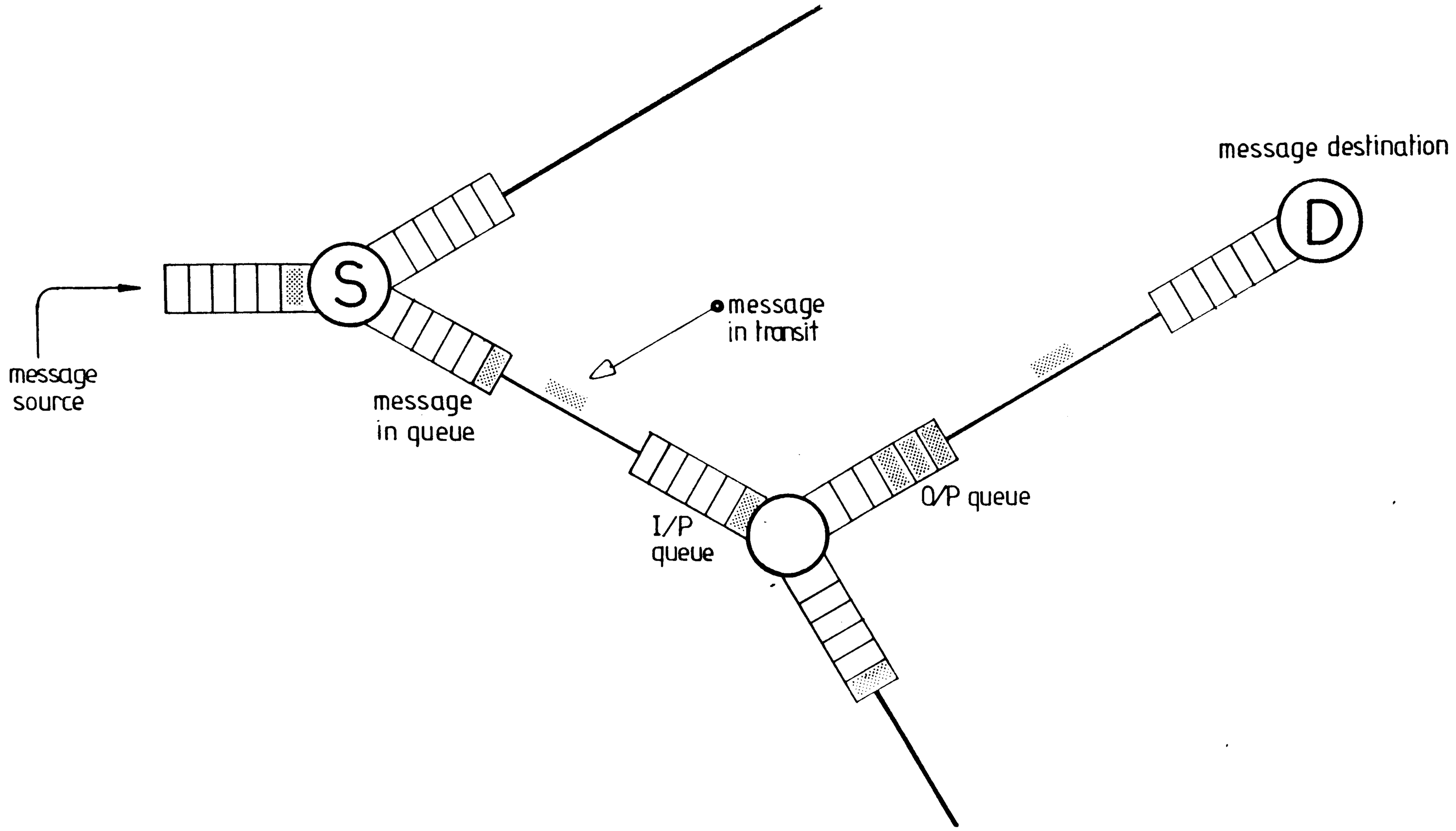


Figure 7.2 Store and forward switching

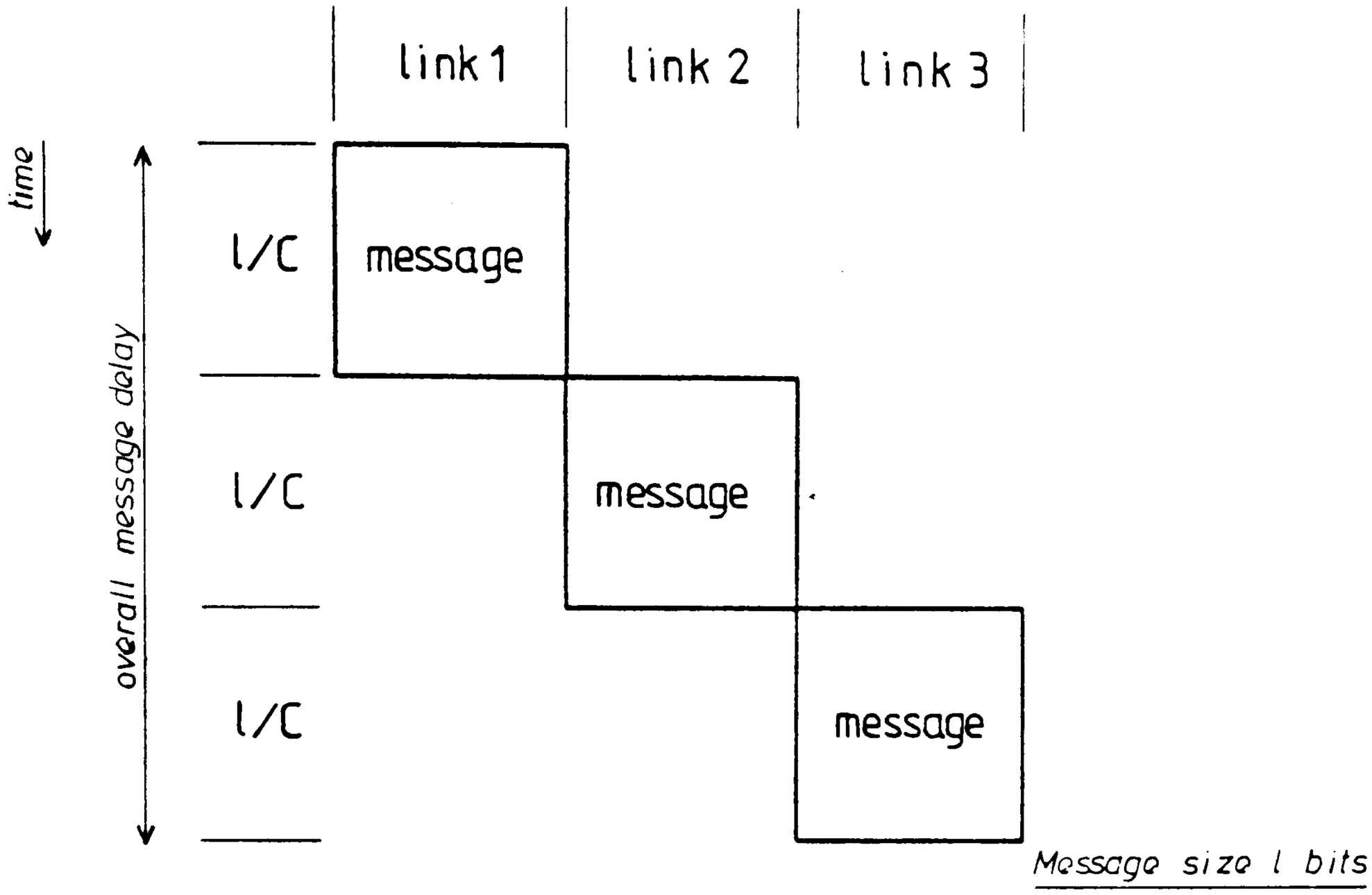


Figure 7.3(a) Message switching

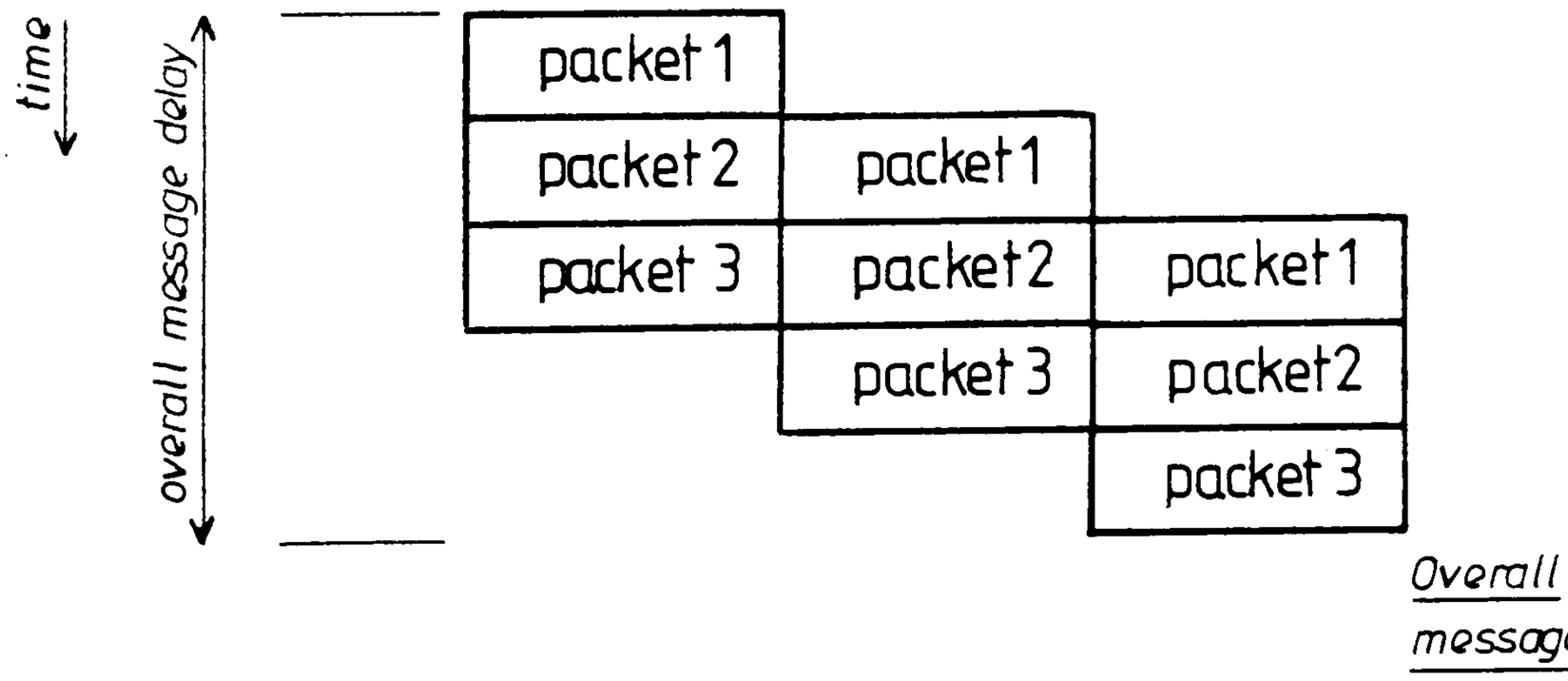


Figure 7.3(b) Packet switching

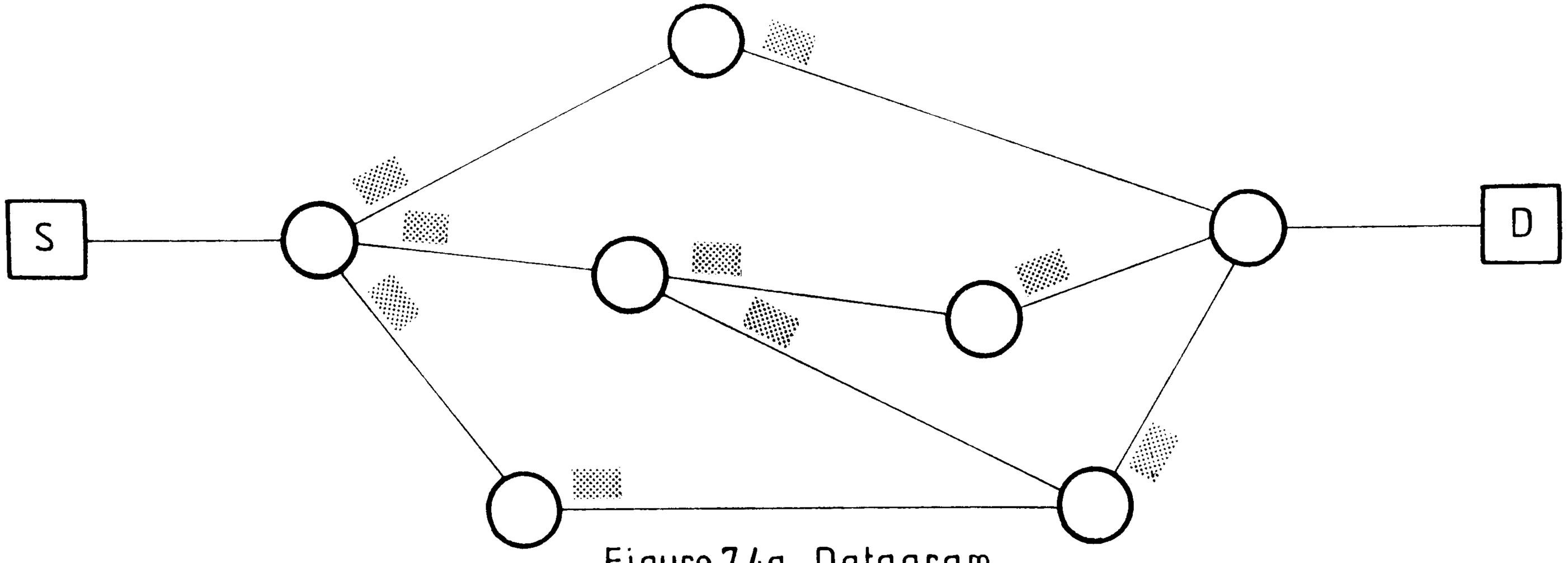


Figure 7.4a Datagram

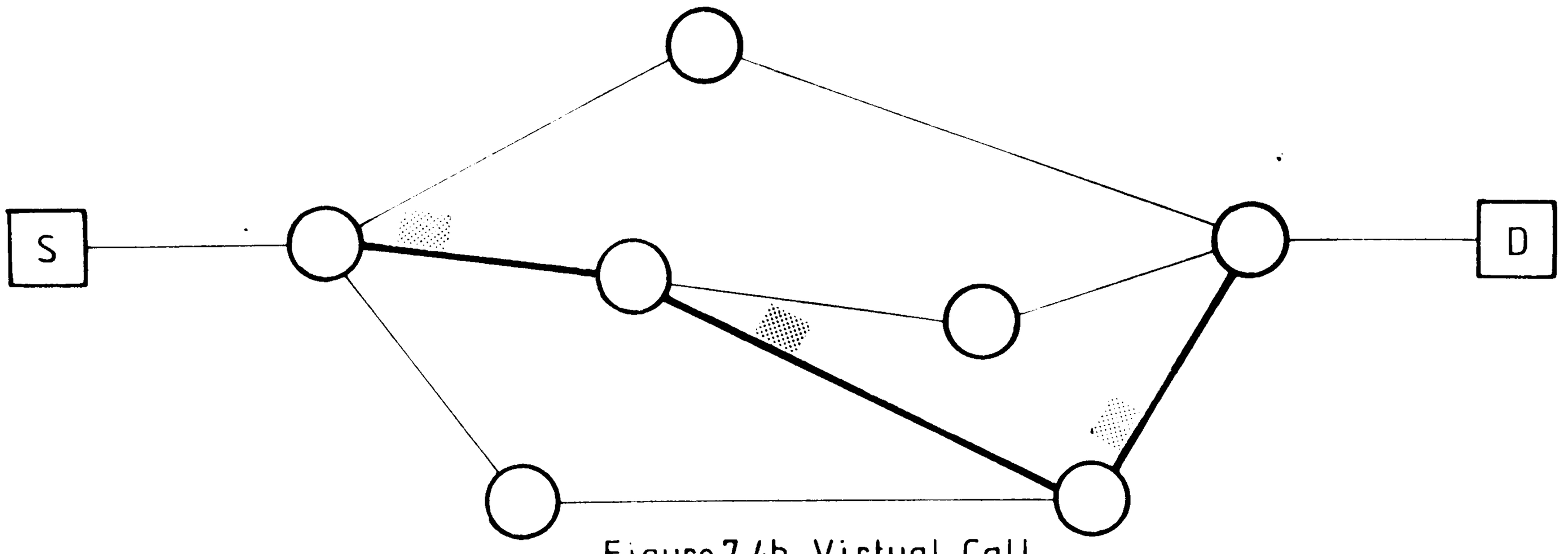


Figure 7.4b Virtual Call

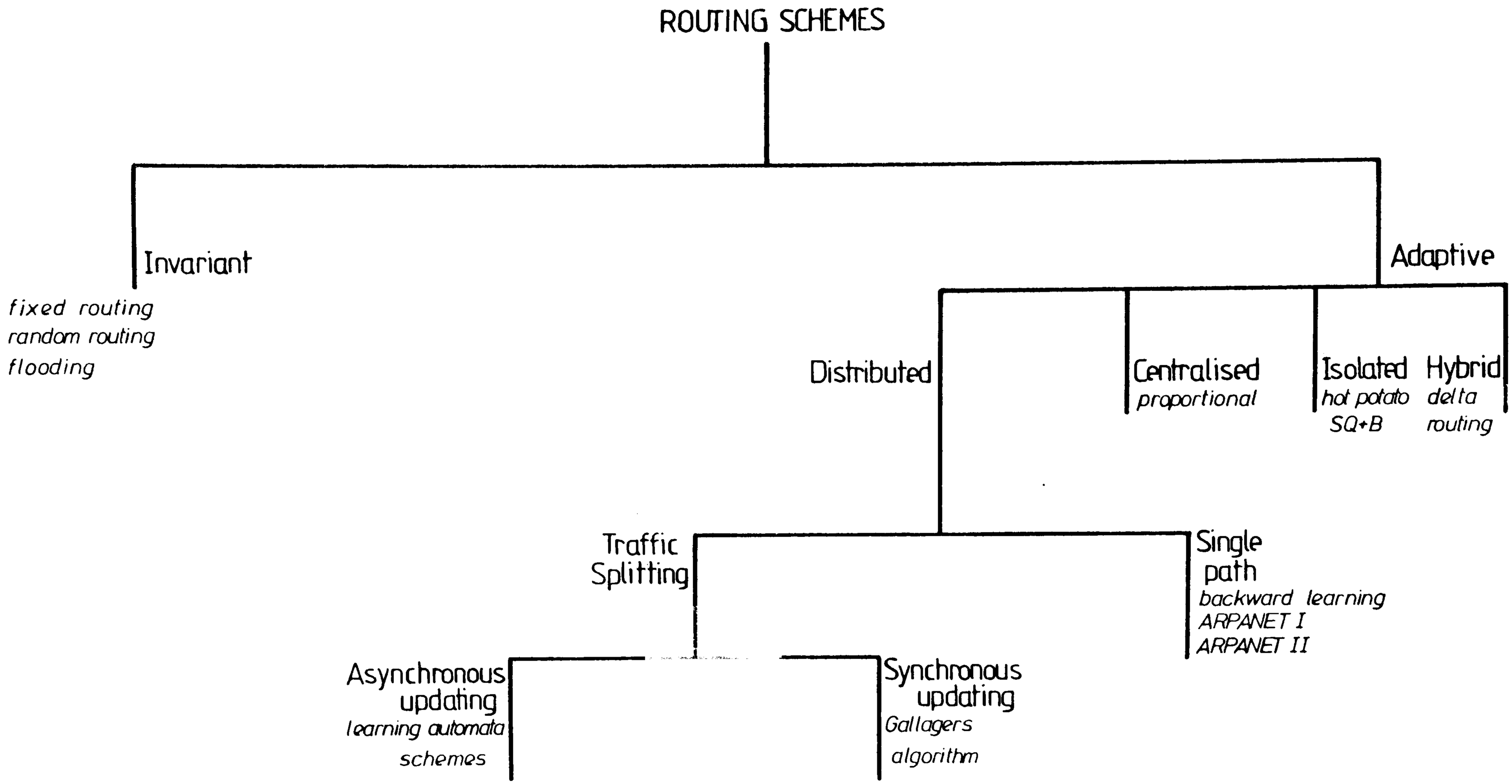
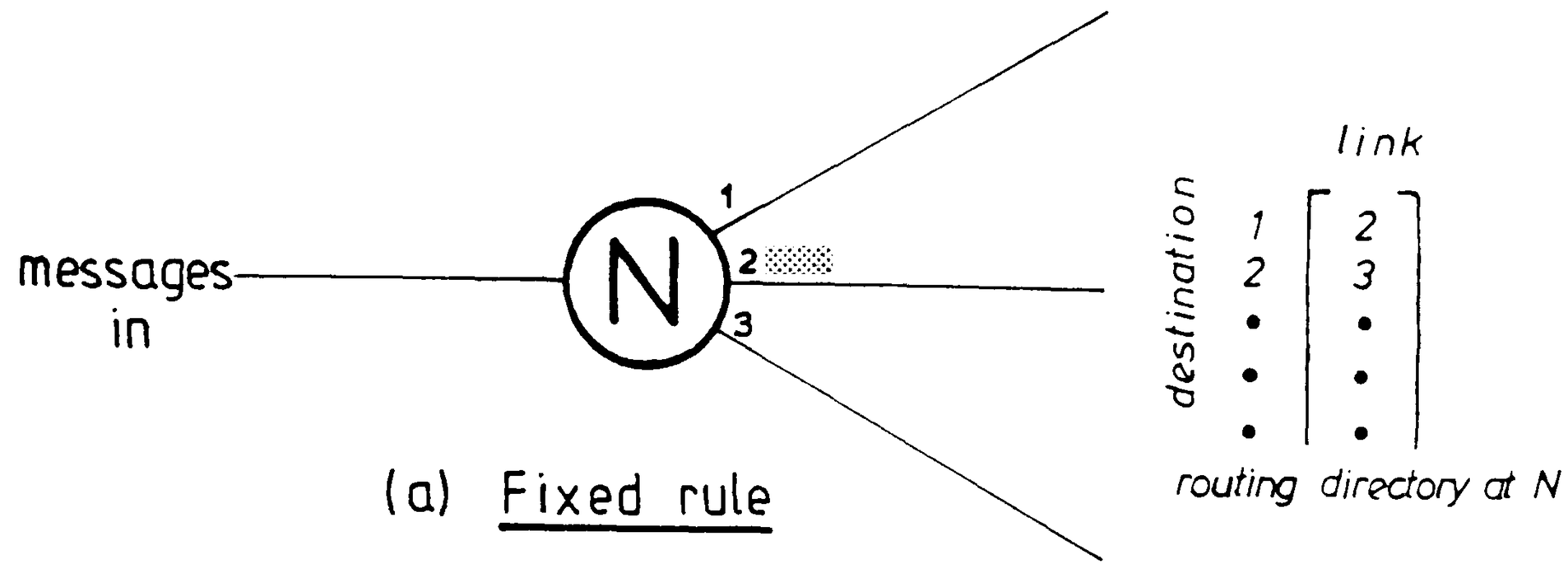
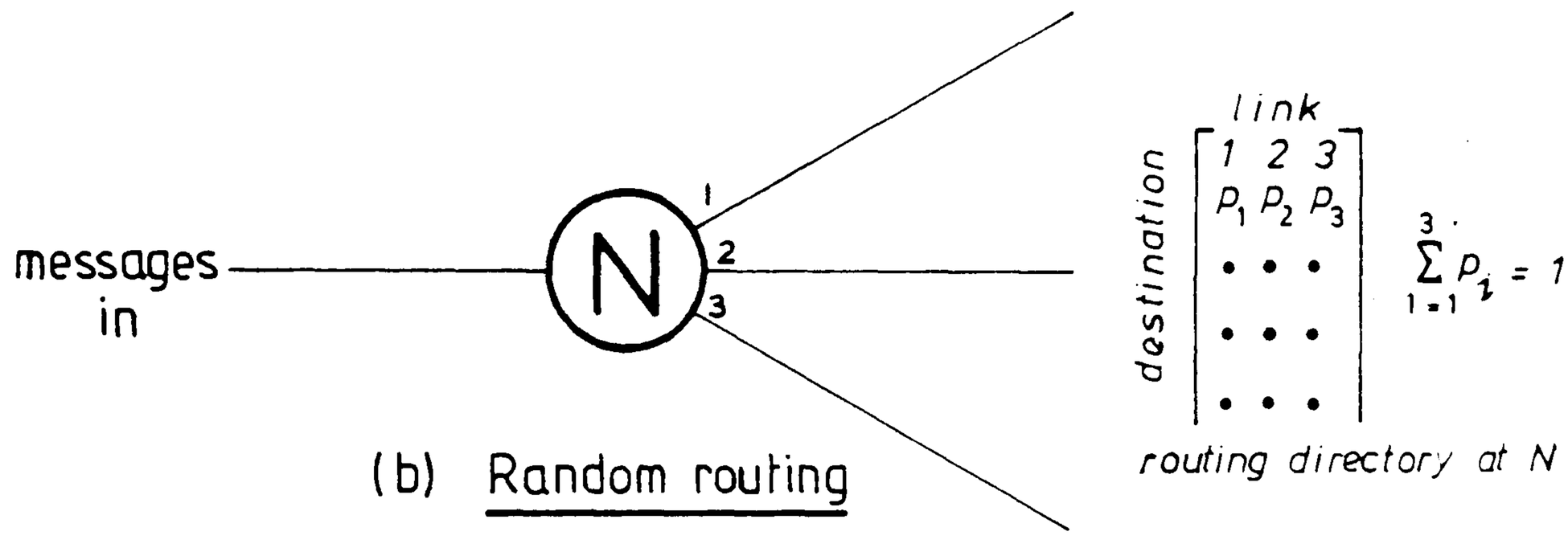


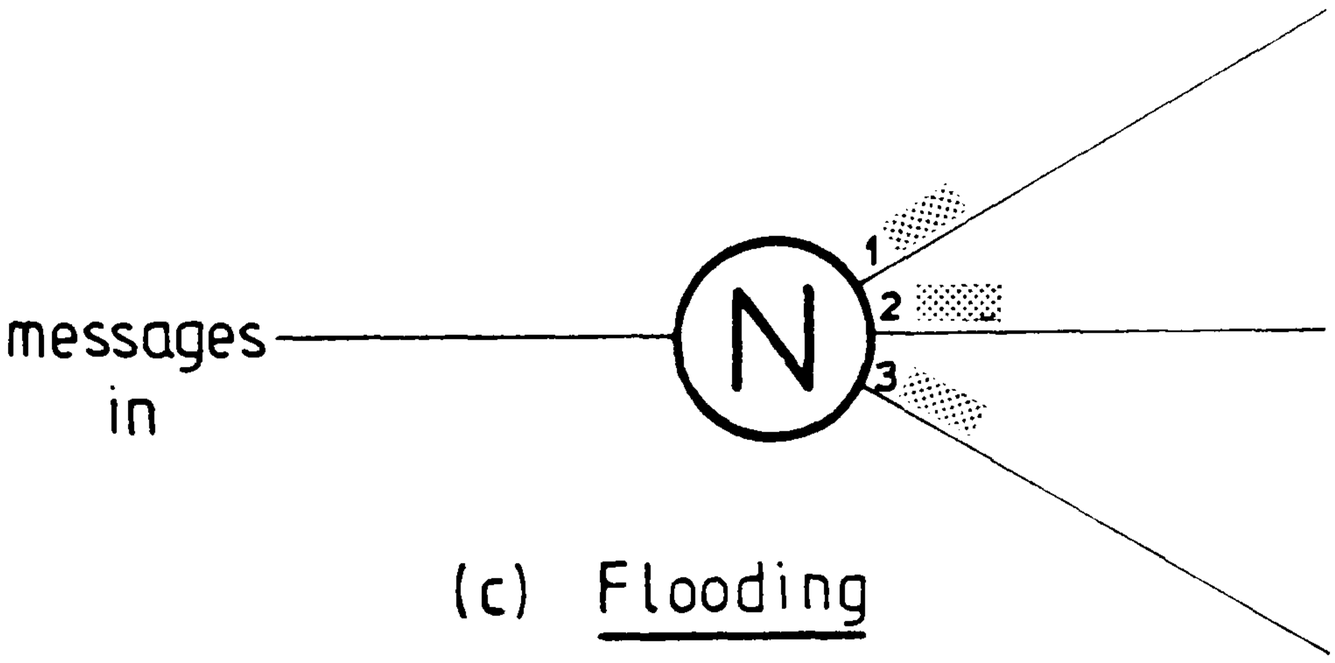
Figure 7.5 Classification of routing schemes



(a) Fixed rule

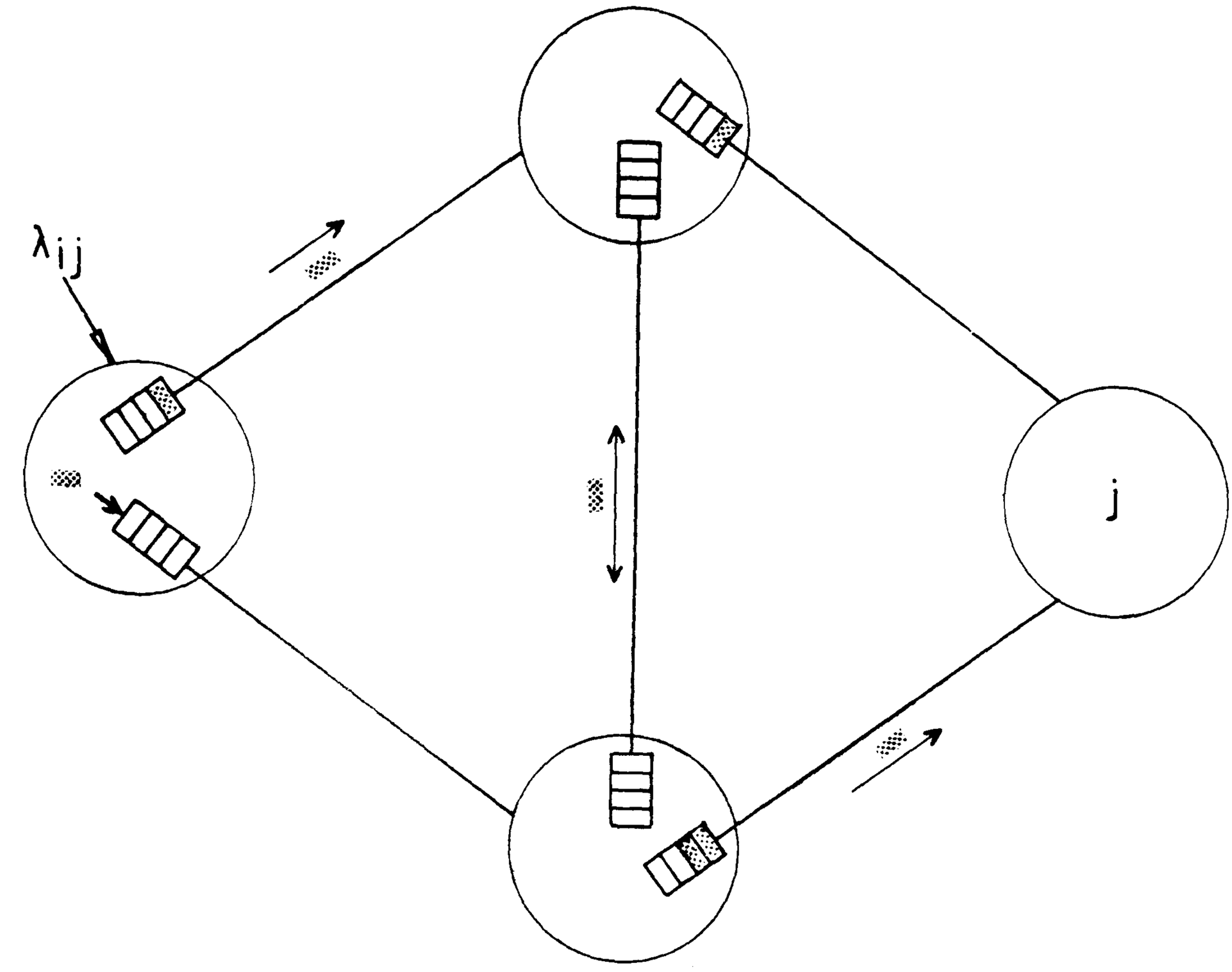


(b) Random routing

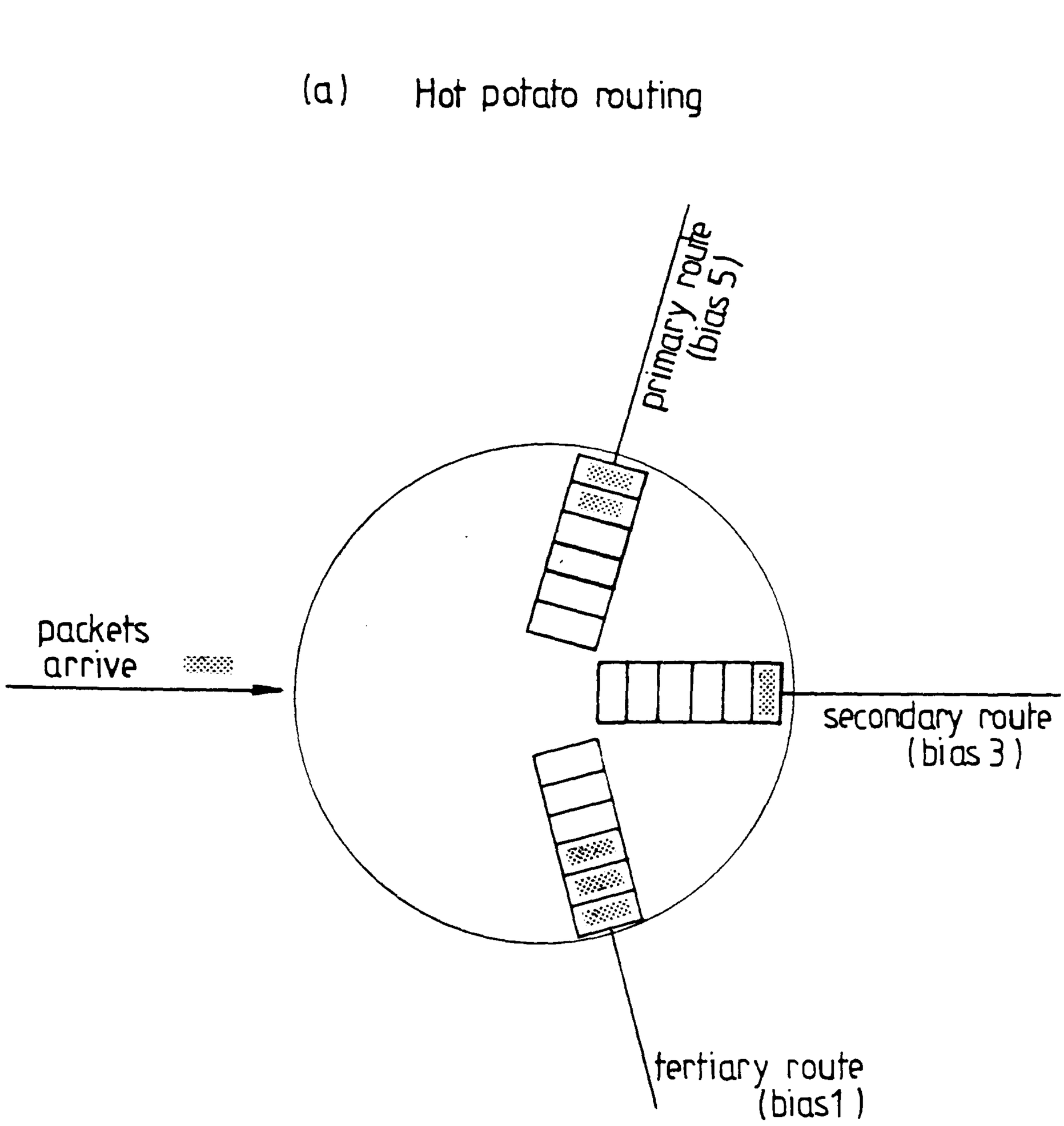


(c) Flooding

Figure 7.6 Simple routing doctrines



(a) Hot potato routing



(b) Shortest queue + bias

Figure 7.7 Isolated routing schemes

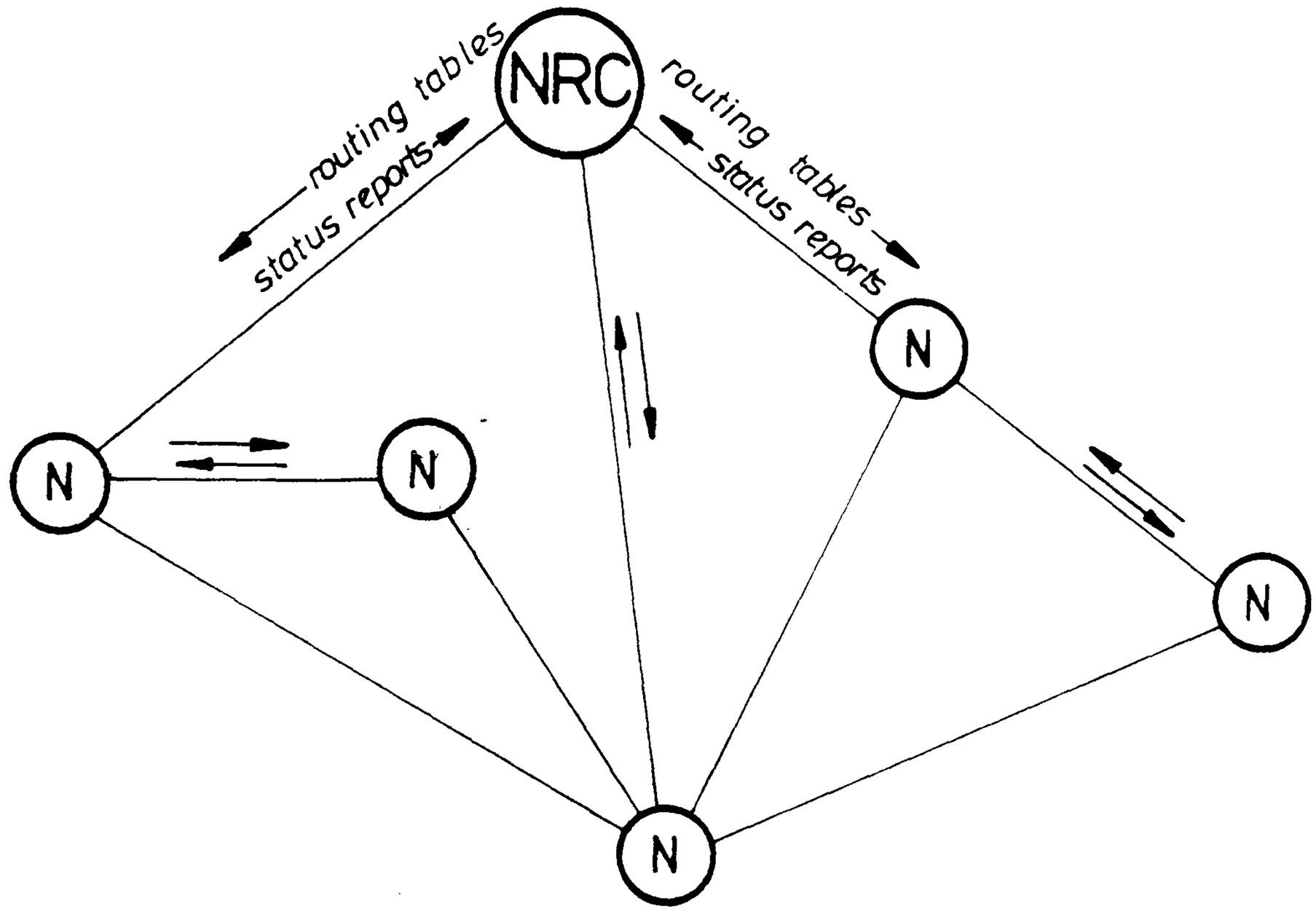


Figure 7·8a Centralised routing

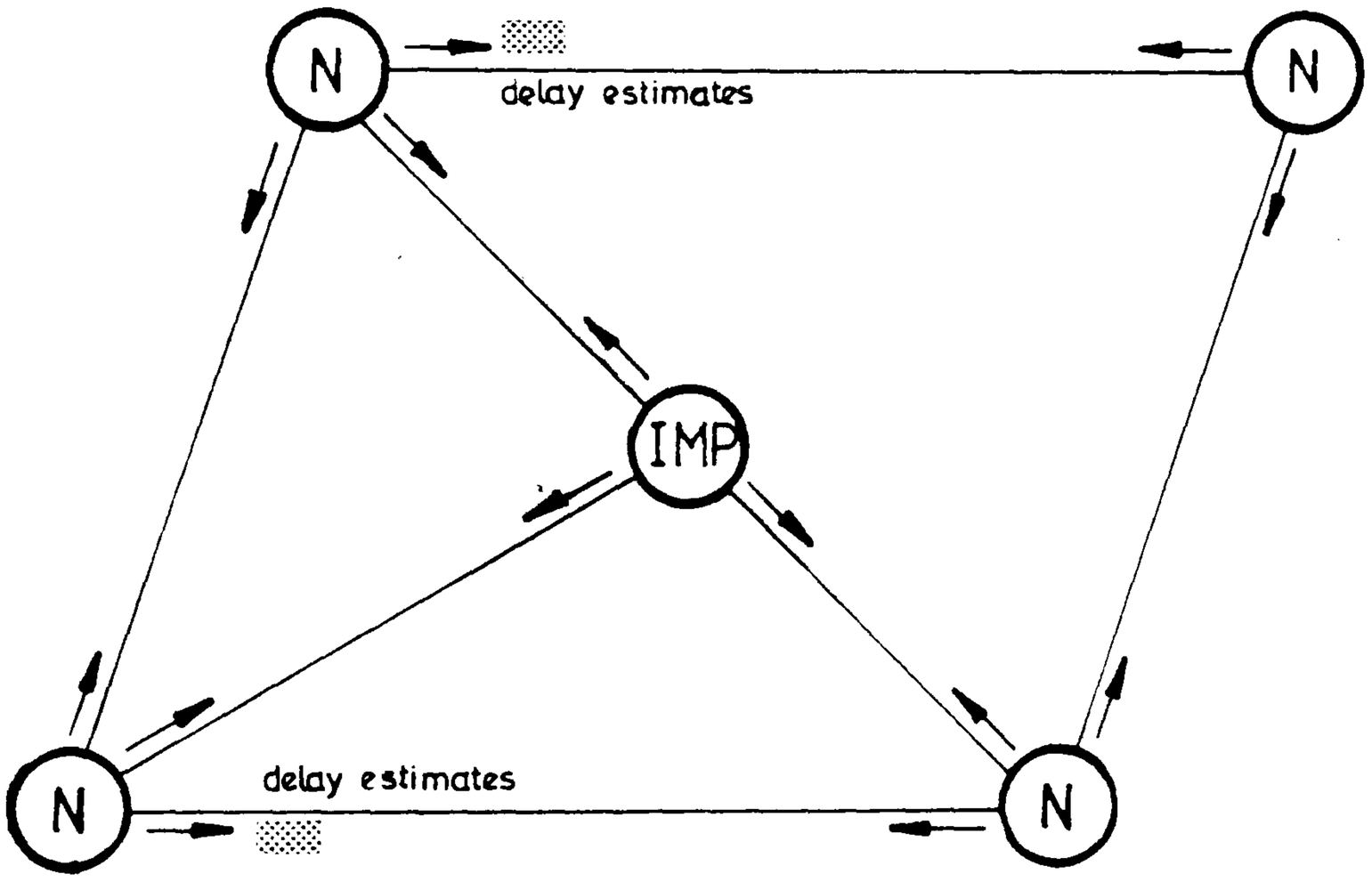
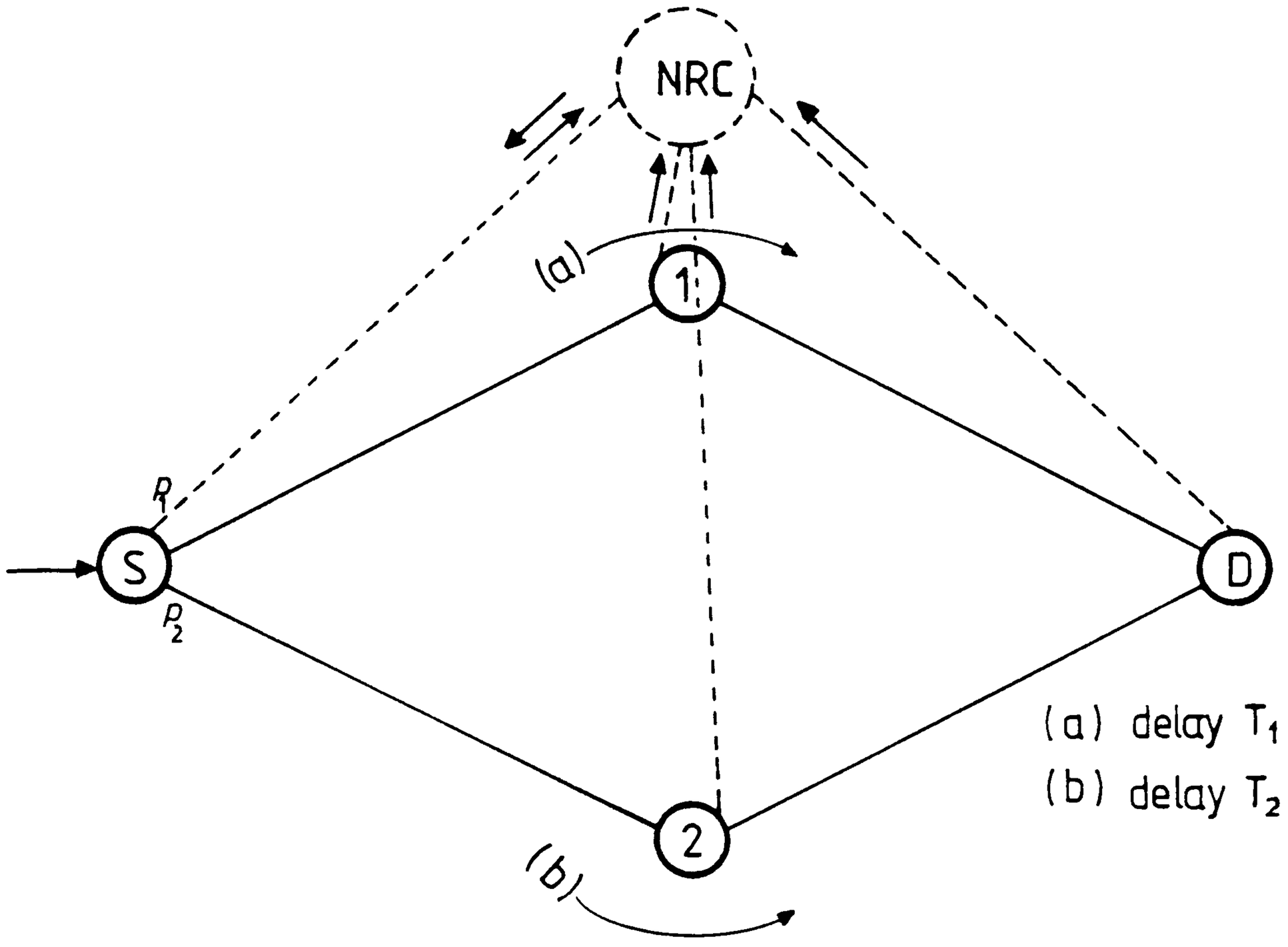
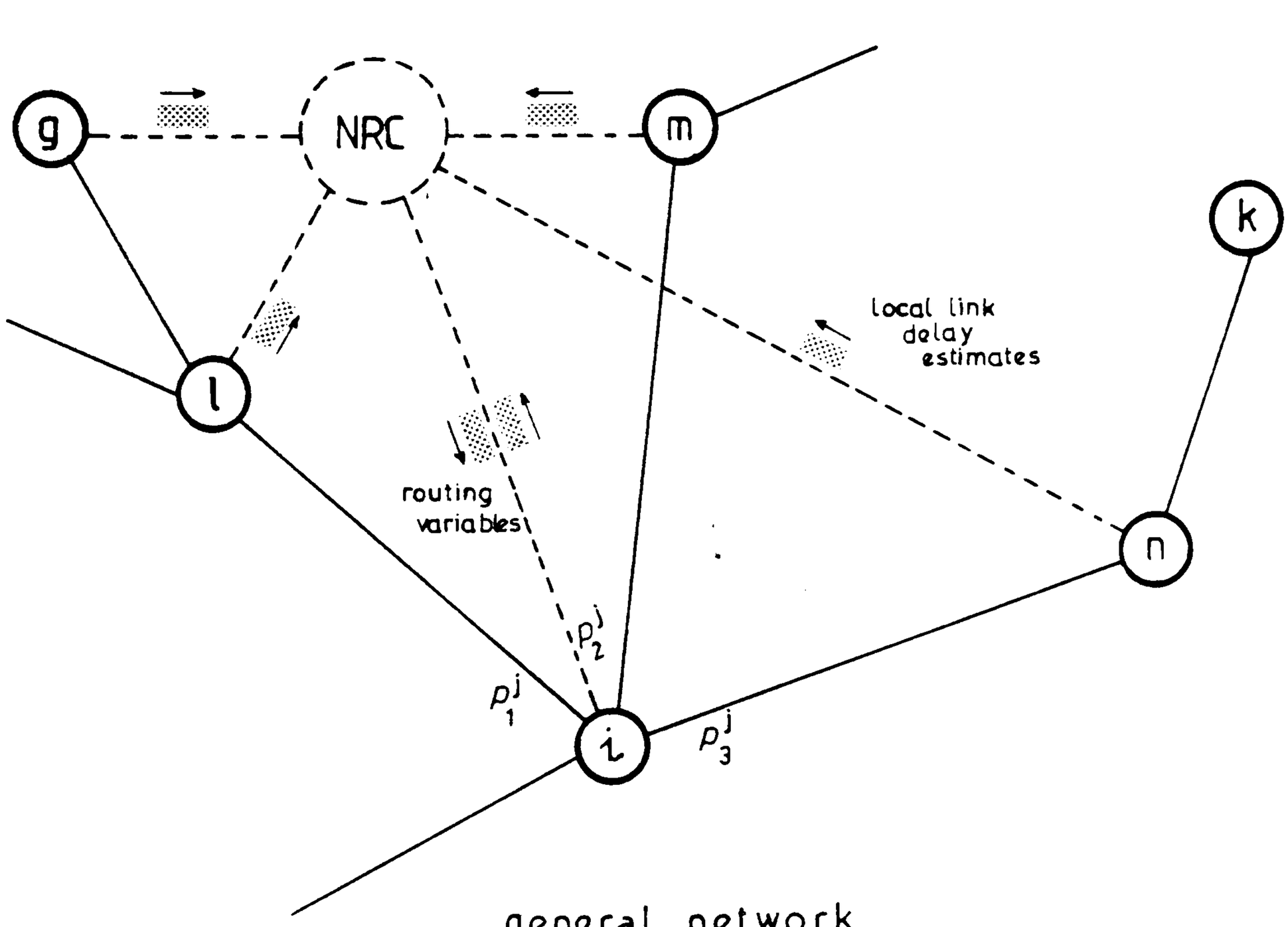


Figure 7·8b Distributed routing



simple network



general network

Figure 7.9 Proportional routing

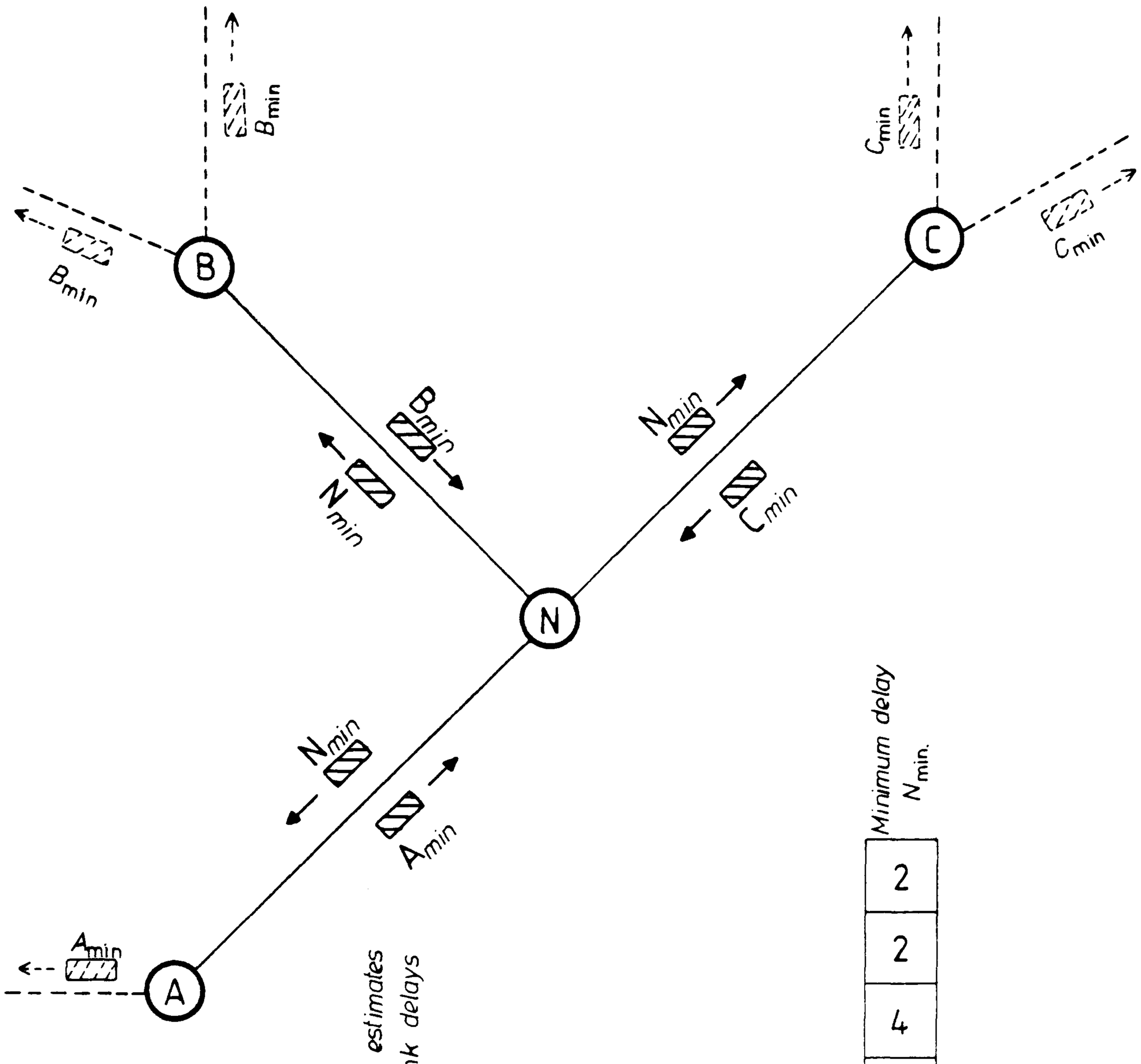
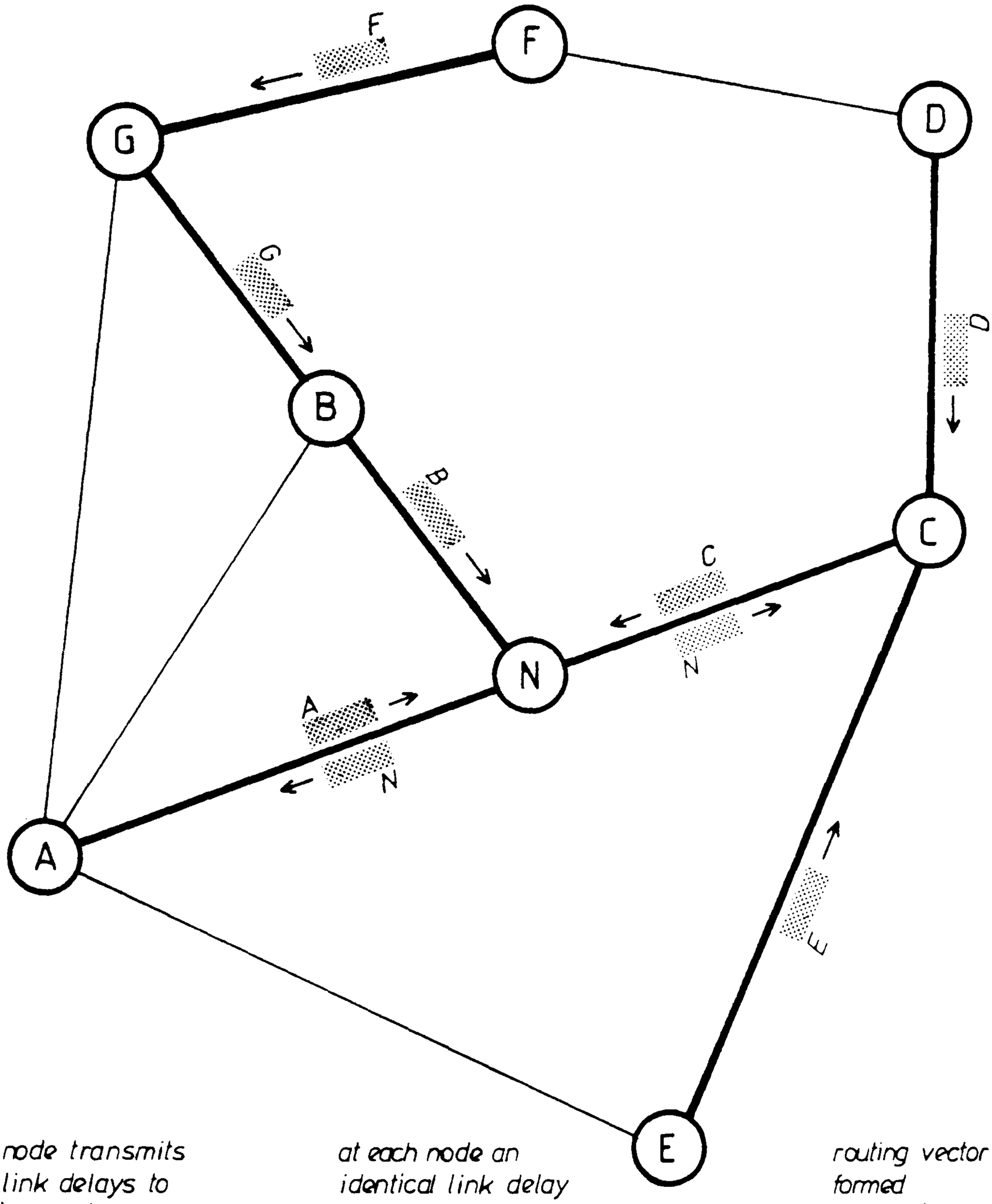


Figure 7.10 Original ARPANET routing algorithm



each node transmits local link delays to all other nodes e.g. node G broadcasts

GA	3
GB	2
GF	4

at each node an identical link delay matrix is constructed [N]

	A	B	C	D	E	F	G	. . .
A	-	2	-	-	1	-	3	
B	1	-	-	-	-	-	2	
C	-	-	-	5	3	-	4	
D	2	-	3	-	-	-	-	
E	6	-	2	-	-	-	-	
F	-	-	-	1	-	-	4	
G	4	1	-	-	-	2	-	

. . .

routing vector formed e.g. node N

B
A
C
⋮

Figure 7.11 New ARPANET routing algorithm

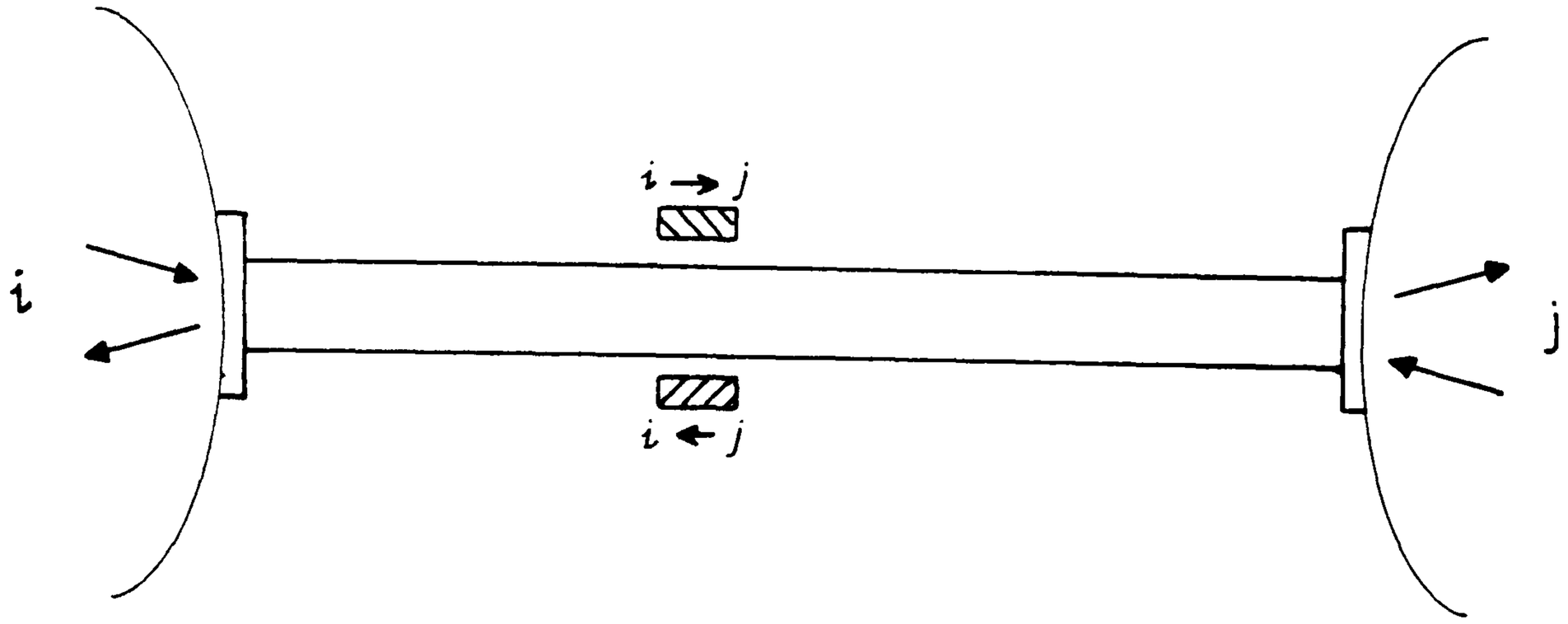


Figure 7·12a Network link (full duplex)

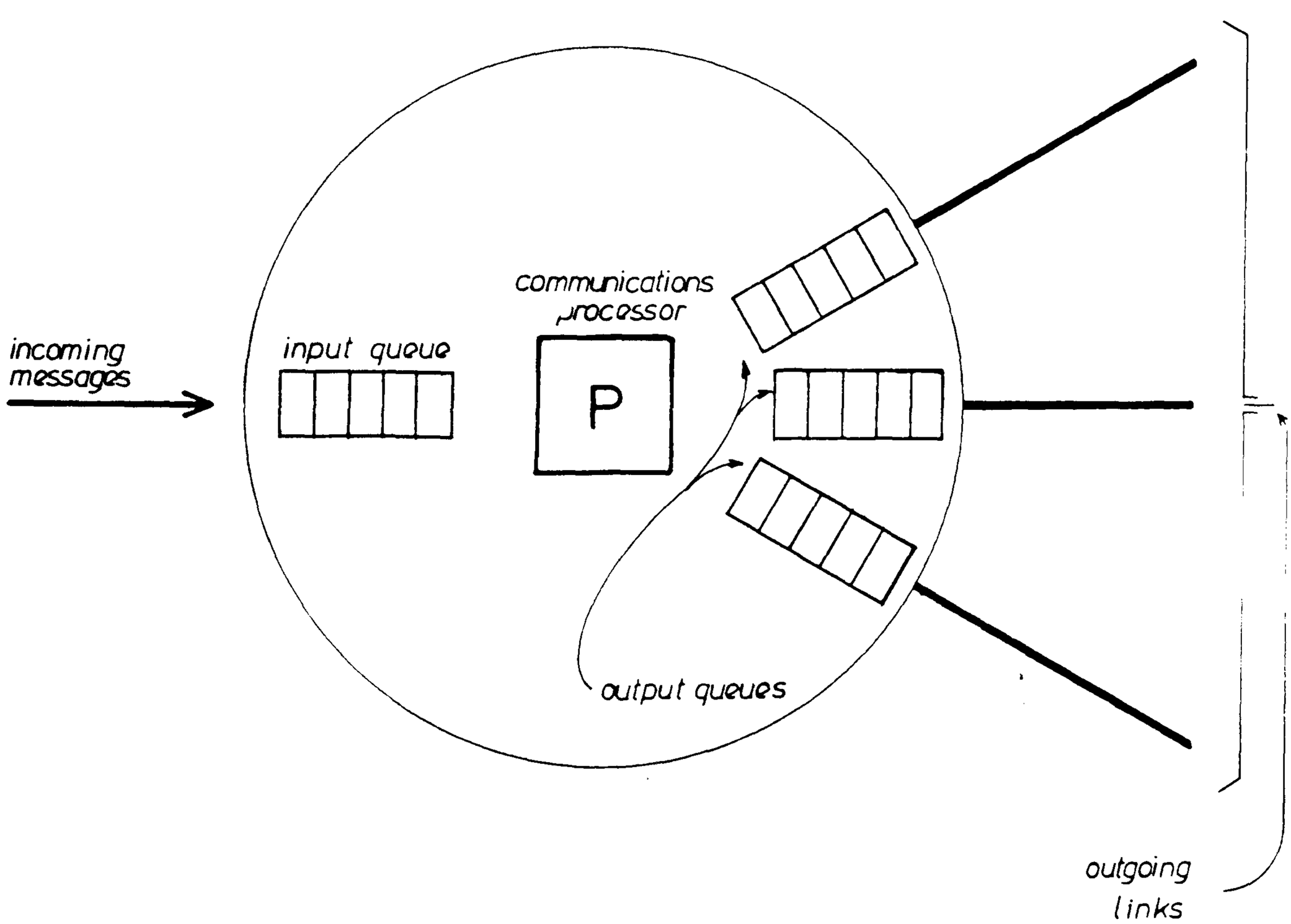


Figure 7·12 b Network node

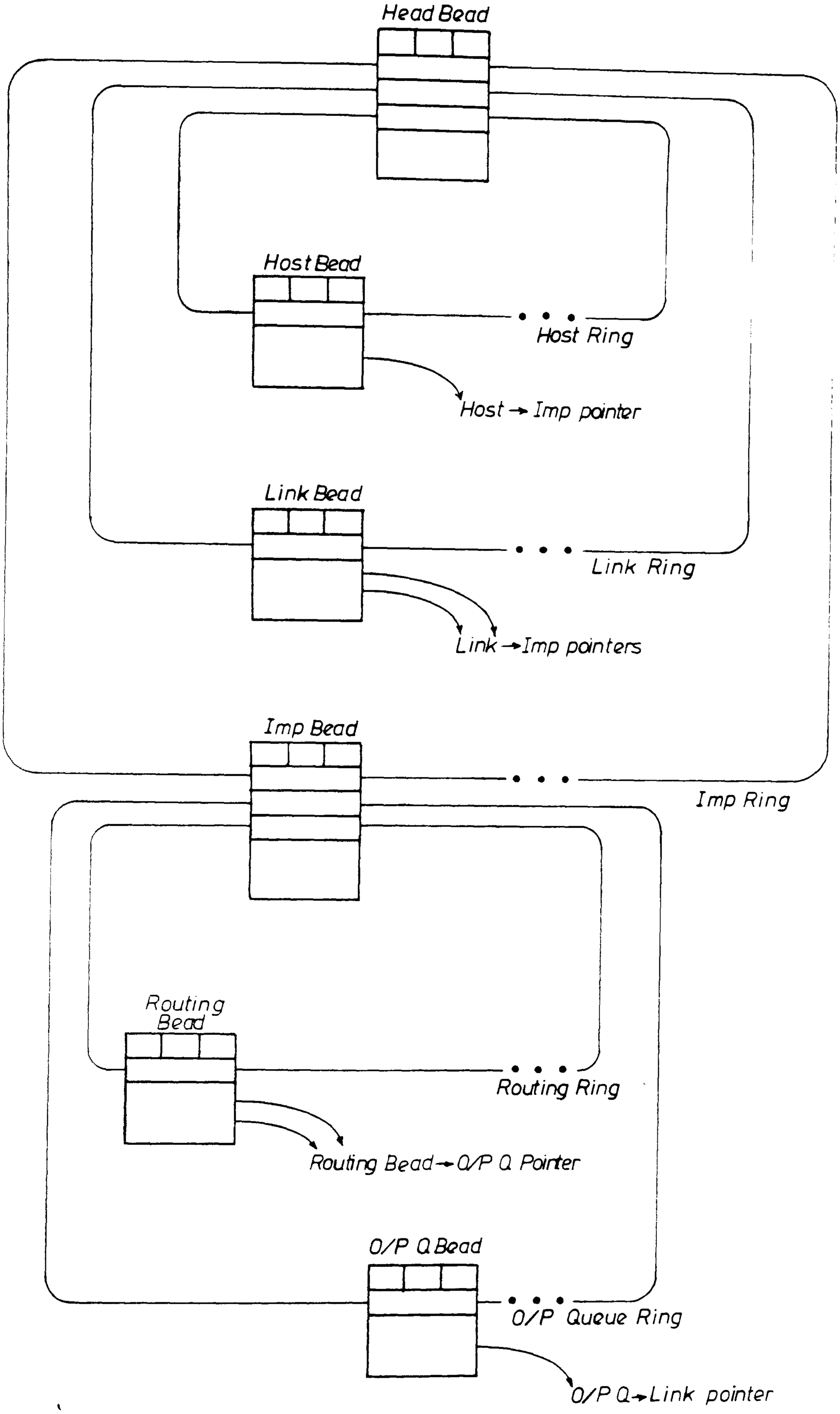


Figure 7·13 Topological data structure

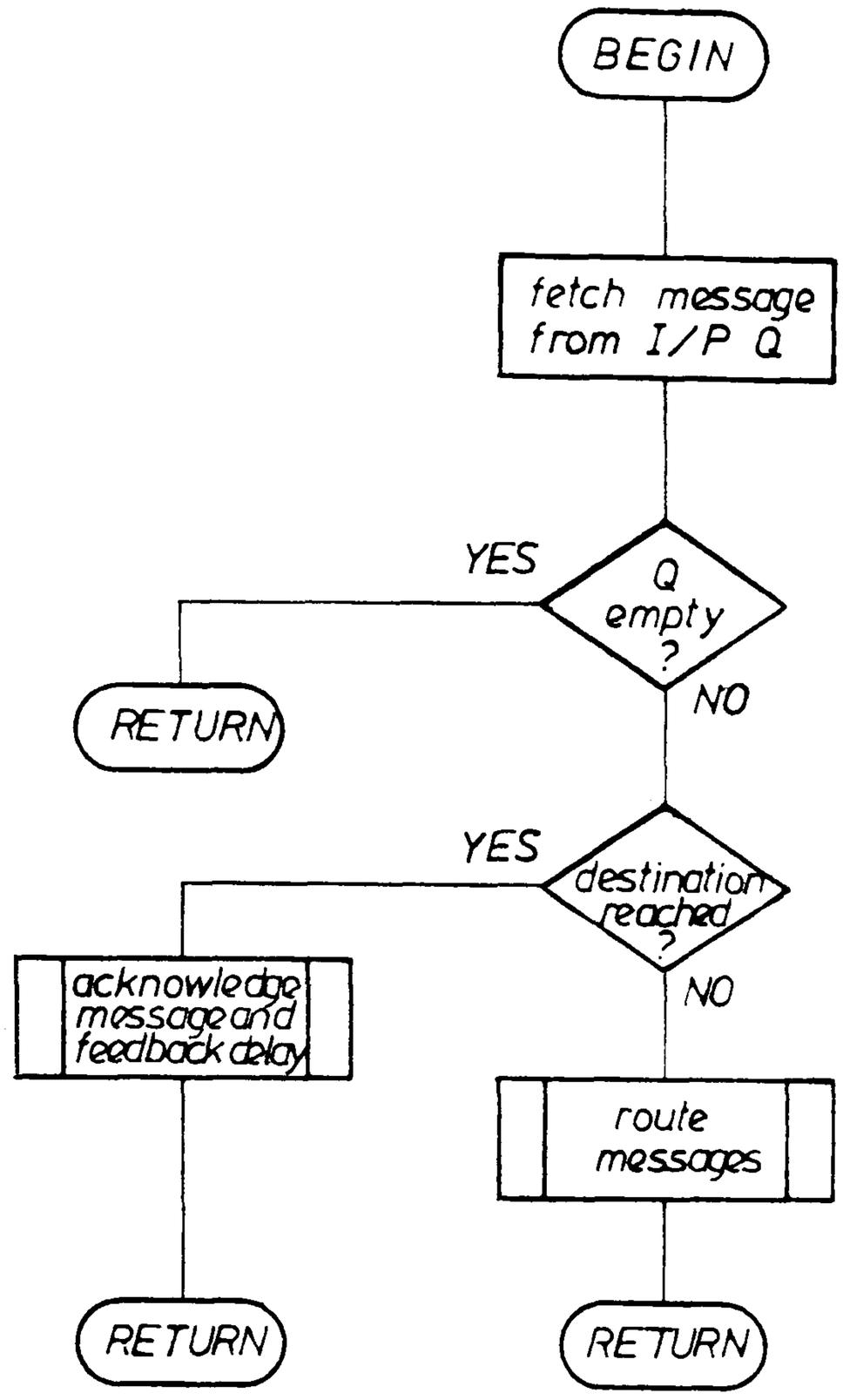


Figure 7.14a Input queue process

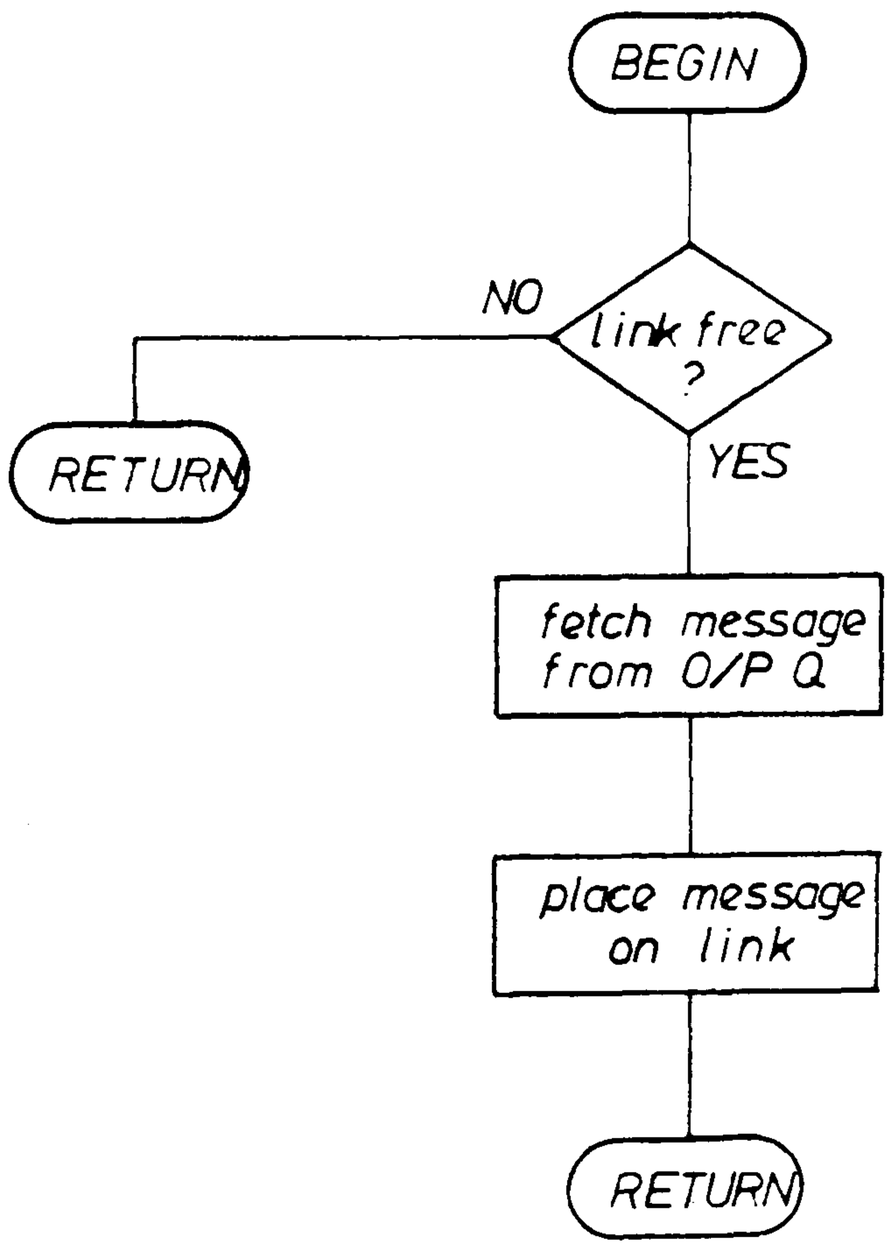


Figure 7.14b Output queue process

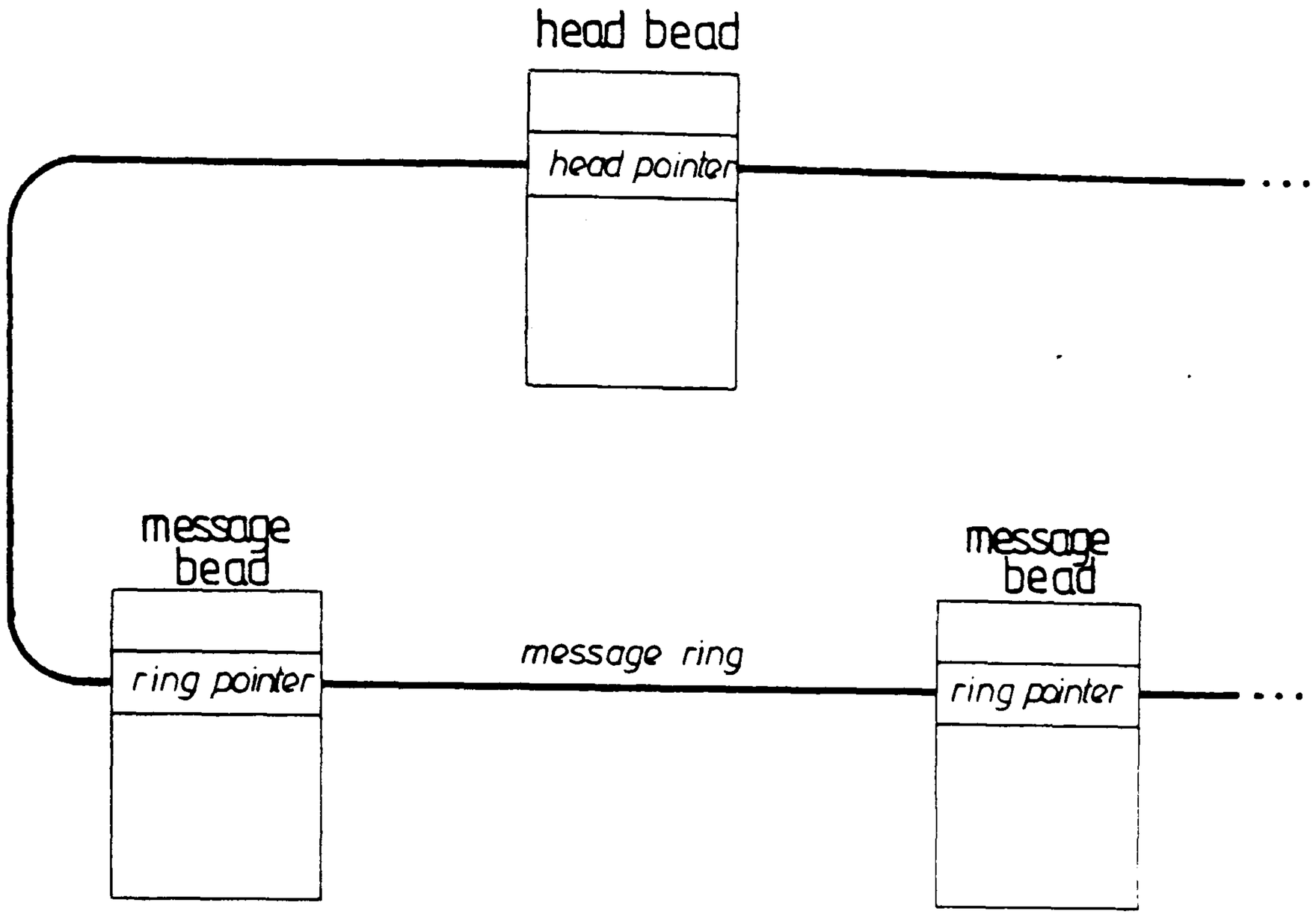


Figure 7.15a Log data structure for message switched simulator

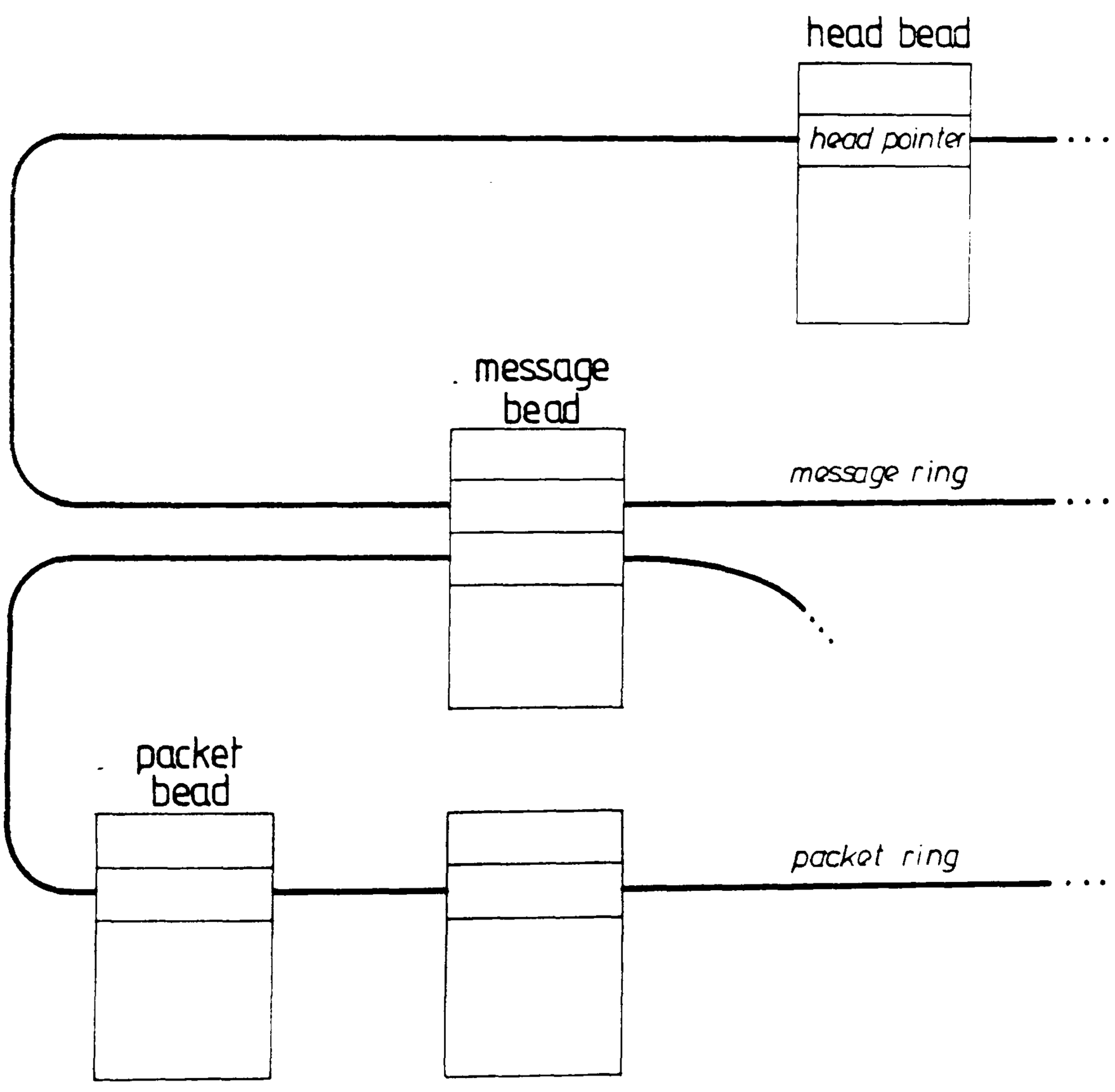


Figure 7.15b Log data structure for packet switched simulator

CHAPTER 8

LEARNING AUTOMATA ROUTING IN MESSAGE SWITCHED NETWORKS

8.1 Introduction

As a continuation of the policy to carry out initial, detailed studies on small networks, followed by realistic network topologies a simple network/automaton combination is proposed with a view to establishing a basic understanding of the learning scheme routing behaviour in a message switched environment. In particular, the influence of the L_{R-I} and L_{R-P} reinforcement algorithms on the network performance are investigated and the steady state behaviour of both automata related to the true optimal condition.

As in the circuit switching study, a four node topology is configured and used with alternative network models where imbalance is produced through differential link capacities and down link interfering traffic. In both these situations a condition for minimum average delay is obtained and compared with the corresponding equilibrium conditions for the specified learning algorithms.

Experimental confirmation of the analysis is provided by simulation studies using the pre-described message switched simulator, and includes experiments with the P-model L_{R-I} and L_{R-P} automata, along with a S-model L_{R-I} variant. In these experiments, results are recorded for the automata action probabilities, the average delays as associated with each automata action and an overall average delay for the entire network. Further simulations evaluate the performance of the fixed, random and proportional routing schemes and permit a comparison with the learning schemes.

8.2 Automaton Routing Controller for Message Switched Networks/

8.2 Automaton Routing Controller for Message Switched Networks

As suggested throughout this study, the learning automaton forms an ideal adaptive control mechanism for implementing distributed routing schemes in communication networks. In the case of circuit switched networks, the automaton operated to select a suitable trunk group, from which a line could be seized and carried out updating from the response generated by the call outcome, i.e., call blocked/call established. With this binary response set, the P-model reinforcement algorithm was therefore a natural choice. For the message switched network, the automaton is required to choose a link on which the message should be transmitted, the feedback in this case being the delay experienced on the selected path.

The basic routing element for the message switched network is shown on Figure 8.1a which illustrates the differences between this application and that for circuit switching. Clearly, a need exists to pre-process the raw delay feedback into a form suitable for the response set of the implemented reinforcement algorithm. The requirements of the three response models and reinforcement algorithms are now discussed.

S-model: The response to the S-model algorithm must be a value in the range (0, 1), which can be obtained by transforming the message delay with a normalisation procedure, mapping the delay T_i to a value t_i.

P-model: The P-model algorithm can also be utilised, a stochastic comparator arrangement, as shown on Figure 8.1b, producing a penalty/response sequence in relation to the feedback delay. This has the advantage /

advantage of allowing the implementation of the L_{R-I} and L_{R-P} algorithms, both of which produce unique convergence characteristics.

Q-model: In the Q-model, the response can be one of a finite number of discrete values, e.g., 0.1, 0.2 etc., and can be suitably produced by normalising and rounding the message delay. Alternatively, the Q-model response set can be made to represent the buffer states for a link, e.g., a queue length of two messages causing a response 0.2 etc.

8.3 Optimal Routing in a Simple Message Switched Network

In this section a simple 4 node network is analysed to find the true optimum routing strategy under asymetrical conditions produced through differential link capacities and also interfering traffic. In addition, the steady state behaviour for the L_{R-I} and L_{R-P} learning schemes are obtained and an interpretation made of the physical behaviour of both automata, concerning the routing strategy and the relationship with the true optimum.

Case 1 Differential Link Capacities

For this case, the network displayed on Figure 8.2 is employed, the traffic consisting of messages originating at node 1, destined for node 3. To simplify the analysis the messages are assumed to exhibit Poisson/Exponential statistics with an average arrival rate λ and mean message length $1/\mu$. Routing the messages along either of the two possible routes is an automaton with action set

$$\underline{\alpha}(n) = \{ \text{path 1}, \text{ path 2} \}$$

the /

the actions performed with probabilities p_1 and p_2. Feedback to the automaton is the transit delay experienced by the message between nodes 1 and 3, this being suitably transformed to suit the reinforcement algorithm.

Primarily we wish to show the condition required for p_1, to minimise the overall average delay in the network. The average delay in message switched networks has been extensively analysed[27] and an independence assumption proposed, which permits the analysis of tandem networks. This assumption, although clearly not corresponding to reality, allows messages arriving at an intermediate node to assume a new length and hence relieves the problems of dependency between the interarrival times and the lengths of adjacent messages, as they travel within the net.

An equation describing the mean delay in a network with k links is

$$\bar{T} = \sum_{i=1}^{k} \frac{\lambda_i}{\Lambda} \bar{d}_i \qquad \ldots 8.1$$

where $\bar{d}_i$ is the average delay on link, λ_i the arrival rate at link i and Λ is the total arrival rate from all message sources.

Specifically for the four node network (Figure 8.2) the average delay is given by

$$\bar{T} = \frac{p_1}{\mu C_1 - p_1\lambda_{13}} + \frac{p_1}{\mu C_3 - p_1\lambda_{13}} + \frac{p_2}{\mu C_2 - p_2\lambda_{13}} + \frac{p_2}{\mu C_4 - p_2\lambda_{13}} \qquad \ldots 8.2$$

To find the optimality conditions, with respect to the routing variables set

$$\frac{d\bar{T}}{dp_1} = 0 \qquad p_2 = 1 - p_1$$

which reveals

$$\frac{\mu C_1}{(\mu C - p_1\lambda_{13})^2} + \frac{\mu C_3}{(\mu C - p_1\lambda_{13})^2} = \frac{\mu C_2}{(\mu C - p_2\lambda_{13})^2} + \frac{\mu C_4}{(\mu C - p_2\lambda_{13})^2} \qquad \ldots 8.3$$

Steady /

Steady State Conditions for the L_{R-I} Algorithms

For the S-model L_{R-I} reinforcement scheme, the operation of an automaton was shown to result in an equalisation of the penalty weights

$$E\left[S_1(n)\,\middle|\,\tilde{p}_1\right] = E\left[S_2(n)\,\middle|\,\tilde{p}_1\right] \qquad n \to \infty$$

Since the penalty weights are the values of normalised delay, the S-model algorithm has the effect of equalising the delays produced on the alternative routing actions, in this case the path delays between nodes 1 and 3.

$$E\left[T_1\right] = E\left[T_2\right] \qquad \ldots 8.4$$

Likewise, for the P-model, the steady state conditions reveal an equalisation of the penalty probabilities

$$E\left[c_1(n)\,\middle|\,\tilde{p}_1\right] = E\left[c_2(n)\,\middle|\,\tilde{p}_1\right] \qquad n \to \infty$$

In this case the penalty probability **c** is the probability that the stochastic comparator (Figure 8.1b) will produce a 1, on being fed a delay value t_i (normalised).

$$c_i = \text{prob}\left[t_i > u\right]$$
$$E\left[c_i\right] = \bar{t}_i$$

Consequently, in the steady state the P-model algorithm behaves as the S-model variant, by also equalising the path delays.

For the simple network, the conditions established by the L_{R-I} automata is therefore

$$\frac{1}{\mu C_1 - p_1 \lambda_{13}} + \frac{1}{\mu C_3 - p_1 \lambda_{13}} = \frac{1}{\mu C_2 - p_2 \lambda_{13}} + \frac{1}{\mu C_4 - p_2 \lambda_{13}} \qquad \ldots 8.5$$

Steady State Conditions for the L_{R-P} Algorithm

In a general non-autonomous environment, the P-model L_{R-P} algorithm will balance the expected penalty rates

$$E\left[p_1 c_1\,\middle|\,\tilde{p}_1\right] = \;/$$

$$E\left[p_1 c_1 \middle| \tilde{p}_1\right] = E\left[p_2 c_2 \middle| \tilde{p}_1\right] \qquad n \to \infty$$

Again with the penalty responses provided by the stochastic comparator we might expect

$$p_1 E\left[T_1\right] = p_2 \; E\left[T_2\right]$$

which can be simply interpreted

$$\frac{N_1}{N} \cdot \frac{\Sigma T_1}{N_1} = \frac{N_2}{N} \cdot \frac{\Sigma T_2}{N_2} \qquad N \to \infty$$

where N_i is total number of action α_i and $N = N_1 + N_2$

$$\Sigma T_1 = \Sigma T_2 \qquad \ldots 8.6$$

This suggests that the L_{R-P} algorithm operates to equalise the accumulated or total delay of all traffic associated with paths 1 and 2. This may also be regarded as an attempt to balance the link utilisation on the various paths, the accumulated delay specifying the total time a link was carrying traffic.

For the simple network under consideration, the following condition would result.

$$\frac{p_1}{\mu C_1 - p_1 \lambda_{13}} + \frac{p_1}{\mu C_3 - p_1 \lambda_{13}} = \frac{p_2}{\mu C_2 - p_2 \lambda_{13}} + \frac{p_2}{\mu C_4 - p_2 \lambda_{13}} \qquad \ldots 8.7$$

Equations 8.5, 8.7 and 8.3 state the equilibrium conditions for the L_{R-I} and L_{R-p} automata and also the necessary condition for optimality, in the four node network. These indicate neither of the learning schemes to be truly optimal, but as will be shown in a later diagram, the conditions are sufficiently close for acceptable performance.

In general, no elegant solution exists for the optimum action probabilities etc., but assuming $C_1 = C_3$ and $C_2 = C_4$, the following are easily obtained.

$$p_1 = \quad /$$

$$p_{opt} = \frac{\mu C_1\sqrt{C_2} - \mu C_2\sqrt{C_1} + \lambda_{13}\sqrt{C_1}}{\lambda_{13}(\sqrt{C_1} + \sqrt{C_2})}$$

$$p^*_{Lri} = \frac{\mu(C_1 - C_2) + \lambda_{13}}{2\lambda_{13}} \qquad \ldots 8.8$$

$$p^*_{Lrp} = \frac{C_1}{C_1 + C_2}$$

Case 2 Interfering Traffic

In the previous case, an imbalance in the delays resulted due to the different link capacities on the alternative paths. This case shows how by introducing additional message sources (Figure 8.3), downstream of the main traffic, a similar asymmetry is formed and illustrates the conditions adopted by the learning schemes.

The average delay in this case is given by

$$\overline{T} = \frac{p_1\lambda_{13}}{\bigwedge(\mu C - p_1\lambda_{13})} + \frac{p_2\lambda_{13}}{\bigwedge(\mu C - p_2\lambda_{13})} + \frac{p_1\lambda_{13}+\lambda_{23}}{\bigwedge(\mu C - p_1\lambda_{13}-\lambda_{23})} + \frac{p_2\lambda_{13}+\lambda_{43}}{\bigwedge(\mu C - p_2\lambda_{13}-\lambda_{43})}$$

where

$$\bigwedge = \lambda_{13}+\lambda_{23}+\lambda_{43}$$

In this particular network configuration the following condition must be achieved to satisfy optimum routing

$$\frac{1}{(\mu C - p_1\lambda_{13})^2} + \frac{1}{(\mu C - p_1\lambda_{13}-\lambda_{23})^2} = \frac{1}{(\mu C - p_2\lambda_{13})^2} + \frac{1}{(\mu C - p_2\lambda_{13}-\lambda_{43})^2} \qquad \ldots 8.9$$

For the automata equilibrium conditions, the L_{R-I} establishes

$$\frac{1}{(\mu C - p_1\lambda_{13})} + \frac{1}{(\mu C - p_1\lambda_{13}-\lambda_{23})} = \frac{1}{(\mu C - p_2\lambda_{13})} + \frac{1}{(\mu C - p_2\lambda_{13}-\lambda_{43})} \qquad \ldots 8.10$$

and for the L_{R-P} /

$$\frac{p_1}{(\mu C - p_1\lambda_{13})} + \frac{p_1}{(\mu C - p_1\lambda_{13} - \lambda_{23})} = \frac{p_2}{(\mu C - p_2\lambda_{13})} + \frac{p_2}{(\mu C - p_2\lambda_{13} - \lambda_{23})} \quad \ldots 8.11$$

8.4 Optimal Routing in a General Network

In the previous section, conditions were derived for the minimum average delay in a simple network and compared with the steady state behaviour of the alternative learning schemes. In contrast to this system optimisation approach, we now study the user optimisation problem in a general network, in which the goal is to find the optimum routing strategy at some node in order to minimise the average delay between the node and a particular destination.

Consider a r action automaton at an arbitrary node k (Figure 8.1a), routing all messages with destination j. Let $\bar{T}_i$ be the average delay experienced by a message routed by action i of the automaton, with the average delay $\bar{T}$ of all messages routed at k given by

$$\bar{T} = \sum_{i=1}^{r} p_i \bar{T}_i$$

Again we wish to find the necessary conditions to satisfy the minimum average delay

$$\frac{d\bar{T}}{d\underline{p}} = 0 \qquad \text{given } \underline{p} = \{p_1, \ldots p_r\}$$

$$\text{and } \sum_{i=1}^{r} p_i = 1$$

By the method of Lagrange multipliers

$$F(p, \epsilon) = \sum_{i=1}^{r} p_i \bar{T}_i(\underline{p}) + \epsilon\left(\sum_{i=1}^{r} p_i - 1\right)$$

$$F_{p_i} = \bar{T}_i(\underline{p}) + \sum_{n=1}^{r} p_n \frac{d\bar{T}_n}{dp_i}(\underline{p}) + \epsilon = 0$$

$$F_{\epsilon} = \sum_{i=1}^{r} p_i - 1 = 0$$

The required conditions for minimum message delay are therefore given by

$$T_a + \quad /$$

$$\bar{T}_a + \sum_{n=1}^{r} p_n \frac{d\bar{T}_n}{dp_a} = \bar{T}_b + \sum_{n=1}^{r} p_n \frac{d\bar{T}_n}{dp_b} \qquad a, b = 1, \ldots r$$

In physical terms, this condition suggests that for true optimality, the routing strategy must attempt to equalise the combination of average delay and average marginal delay associated with each routing alternative. If the delay corresponding to a certain action is not influenced by the other routing actions and consequently $d\bar{T}_i / dp_j = 0 \quad i \neq j$, the optimum condition may be specified as

$$\bar{T}_a + p_a \frac{d\bar{T}_a}{dp_a} = \bar{T}_b + p_b \frac{d\bar{T}_b}{dp_b} \qquad a, b = 1, \ldots r$$

Turning to the operating conditions of the r action reinforcement algorithms, it follows from the non-autonomous environment and simple network studies, that the L_{R-I} scheme attempts to equalise the average delay on all routing alternatives

$$\bar{T}_a = \bar{T}_b \qquad a, b = 1, \ldots r$$

Likewise for the r action L_{R-P} the equilibrium conditions are described by

$$p_a \bar{T}_a = p_b \bar{T}_b \qquad a, b = 1, \ldots r$$

Although neither of the learning schemes are truly optimal, under light traffic conditions the marginal delay $d\bar{T}_i / dp_i$ becomes small and hence the operation of the L_{R-I} scheme approaches the desired routing strategy. Also, because no assumptions have been made regarding the nature of the traffic statistics, the above conditions hold for a range of networks.

8.5 Simulation Studies

In order to verify the theoretical predictions of section 8.3, a range of simulations have been performed using the message switched simulator described previously. These experiments /

experiments include system identification studies carried out to check the simulator and confirm initial assumptions regarding message length independence. Further results are recorded showing the performance of the learning schemes (L_{R-I}, L_{R-P} and SL_{R-I} algorithms), noting the convergence of the routing probabilities and the overall average delay. For comparative studies, experiments have been performed with a fixed rule, random and proportional routing schemes.

Network Topology

The network used in the following experiments is the 4 node topology as shown on Figure 8.2, with link capacities C_1, C_3 = 400 Kbits/sec and C_2, C_4 = 600 Kbits/sec. The nodal input and output buffers have been assigned sufficient queueing elements to exclude the possibility of message blocking. Finally, the network traffic, source node 1, is assumed to have an average message length of 350 bits and is operated at arrival rates from 1000 to 1800 messages/sec.

Identification Studies

In the following series of experiments, a random routing mechanism at node 1 routes messages along either of the two paths, with probabilities p_1 and p_2. For each experiment a given proportion of traffic is assigned to the paths, by setting the appropriate probabilities e.g., $p_1 = 0.1$, $p_2 = 0.9$. A complete set of results were then obtained for the average delay on both paths for p_1 ranging from 0 to 1.

Typical experimental results for λ_{13} = 1100 messages/sec are plotted on Figure 8.4, along with theoretical plots of the average delays. These indicate a fair degree of correlation /

correlation between theory and practice, justifying the message independence assumption introduced to simplify the analysis. From Figure 8.4, the expected convergence characteristics of the L_{R-I} and L_{R-P} reinforcement schemes can be studied, the interception of the average delay curves $\bar{T}_i$ specifying the steady state condition for the L_{R-I} algorithm and the crossover of $p_i \bar{T}_i$ showing the nature of the L_{R-P}. In addition, it is possible by inspection to note the true optimal condition, which reveals the optimum routing probability and minimum expected delay.

Routing Experiments

The experiments are now concerned with comparative studies of several routing schemes, including fixed, random, proportional and learning automata routing using the L_{R-I} (P and S-models) and L_{R-P} algorithms. Again the network was maintained in the topology used previously and measurements of the message delay and steady state routing probabilities taken for a range of traffic rates. Furthermore, the results displayed were measured over the last 2000 messages of a 3000 message sample, ensuring that the queues and routing schemes were in steady state.

Fixed Rule Routing

In the simple network, the optimum fixed rule is the path 1, 4, 3, since this route offers a greater link capacity. With this arrangement, the results shown in Table 1 and Figure 8.5 were obtained, the higher traffic intensities however, causing heavy congestion with results in queue blocking.

Random Routing

The implemented random routing scheme was set for $p_1 = p_2$, and produced the results as shown in Table 2 and Figure 8.5, /

Figure 8.5, the traffic rate ranging from 1 000 to 1 800 messages/sec.

Proportional Routing

A simple proportional routing scheme, similar to that described in Chapter 7, is implemented at node 1, with θ_s= 0.02 and θ_l = 0.05. The results for these experiments are shown on Table 3.

L_{R-P} Learning Automata Routing

For a first attempt with a learning algorithm, the P-model L_{R-P} scheme was established, using the learning coefficients a = b = 0.02. The measured results are shown on Table 4 and Figure 8.5.

L_{R-I} Learning Automata Routing

The L_{R-I} reinforcement algorithm has been studied in two forms, the P and S-model, each using an "a" parameter of 0.02. The results for the P-model are shown on Table 5 and Figure 8.5, with the S-model results on Table 6.

The results clearly indicate the superior performance of the adaptive schemes in comparison to the fixed and random routing schemes, in the conditions offered by the four node network. The family of theoretical delay curves (Figure 8.6) which shows the overall delay characteristics for varying λ provide a further insight as to why this should be so, the figure also indicating the routing policy of the fixed, random and Learning schemes.

A close study of the results from the adaptive schemes reveals the different optimisation operations of the L_{R-I} and L_{R-P} reinforcement schemes, confirming earlier theoretical predictions. For both the P and S-model L_{R-I} algorithms, the operation leads to an equalisation of the /

the path average delays, the automaton converging to a routing strategy which results in a balancing of the average penalty weight for each action. The L_{R-P} on the otherhand provides a means to equalise the accumulated delay for each path, effectively balancing the utilisation of the path links. This behaviour illustrates an interesting dual with the learning automata adaptive routing for circuit switched networks, where the L_{R-I} algorithm was shown to equalise the path blocking probabilities with the L_{R-P} providing a similar equalisation of the blocking rates (or the accumulated number of blocked calls).

The results for the proportional routing also tend to suggest a routing strategy which attempts to equalise the average path delays, which might be expected from the simple implementation, since it resembles the operation of a learning automaton. In the more involved environment of a practical network, the proportional routing scheme would, however, be far more involved to implement, in relation to the simplicity of the learning schemes.

Figure 8.7 displays the effects of the varying traffic on the optimum routing probabilities and the asymptotic convergence conditions of the L_{R-I} and L_{R-P} automata. It is interesting to note the variation of the true optimum under different traffic conditions and the ability of the L_{R-I} algorithm to track this movement. Equally interesting is the behaviour of the L_{R-P} scheme which, both theoretically and experimentally, indicates an independence of traffic, although in general this need not be the case.

8.6 Conclusions

An elementary study into the application of learning algorithms for adaptive routing in message switched networks has /

has been presented. Through a simple network, a model of the routing process has been formulated and used to investigate the properties of the L_{R-I} and L_{R-P} reinforcement algorithms. In turn, this has been validated by detailed computer simulations of the network, with additional experiments demonstrating the performance of the fixed, random and proportional routing schemes under identical conditions.

From this initial study, it is possible to draw conclusions concerning the learning approach to message switched adaptive routing.

1 The learning schemes, with the ability to adapt to the experienced network conditions, out-performs the invariant routing schemes such as fixed and random routing. Despite the rigid routing policy of the random scheme a comparable performance is achieved when the network loading is light. Under heavier traffic conditions however, the learning routing schemes permit a better utilisation of the available facilities and thus attain a superior delay performance.

2 The L_{R-I} algorithm operates to equalise the average delay for messages routed by the allowable actions of the automaton. Under a slightly different policy, the L_{R-P} produces a balance in the accumulated delay for each action. Both these effects are desirable, as they offer a load balancing service in the event of any network asymmetry.

3 /

3 As suggested, the option exists to exploit the range of automata, P, Q and S-models, the P-model allowing the implementation of both the L_{R-I} and L_{R-P} schemes, while the S-models provide a simpler, and more suitable approach for operation with the delay feedback from the network. The Q-model could also be utilised in an arrangement where any response to an action is based on the buffer occupancy of the down stream nodes.

The learning automaton forms a useful adaptive control component for the implementation of practical distributed routing schemes in store and forward networks. In the following chapter schemes are proposed which employ learning automata in de-centralised routing techniques for virtual call and datagram packet switched networks.

Fixed Routing

Table 1

λ_{13}	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	ΣT_1	ΣT_2	ΣT
1000	-	2.81	2.81	-	5.62	5.62
1200	-	3.89	3.89	-	7.78	7.78
1400	-	6.25	6.25	-	13.5	13.5

$\bar{T}$ Average Delay (mSecs)

ΣT Accumulated Delay (Sec)

Random Routing

Table 2

λ_{13}	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	ΣT_1	ΣT_2	ΣT
1000	3.06	1.79	2.43	3.83	2.24	6.07
1200	4.14	1.82	3.0	5.25	2.24	7.49
1400	4.87	1.89	3.38	6.09	2.36	8.45
1600	4.86	2.31	3.57	6.01	2.92	8.93
1800	6.36	2.42	4.37	7.87	3.06	10.93

Proportional Routing

Table 3

λ_{13}	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	ΣT_1	ΣT_2	ΣT	p_1^*
1 000	2.43	2.41	2.42	0.57	4.27	4.83	0.177
1 200	2.76	2.60	2.63	1.19	4.07	5.26	0.217
1 400	2.96	2.63	2.74	1.91	3.56	5.48	0.324
1 600	3.65	3.59	3.61	2.50	4.73	7.22	0.342
1 800	3.79	3.82	3.81	2.60	5.02	7.63	0.343

Linear Reward Penalty P-model

Table 4

λ_{13}	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	ΣT_1	ΣT_2	ΣT	p_1^*
1 000	2.04	3.02	2.44	2.41	2.47	4.88	0.408
1 200	3.31	2.09	2.59	2.71	2.47	5.18	0.409
1 400	3.18	2.51	2.80	2.72	2.88	5.60	0.427
1 600	3.98	2.77	3.25	3.16	3.34	6.5	0.397
1 800	4.24	2.87	3.44	3.49	3.38	6.88	0.411

Linear Reward Inaction P-model

Table 5

λ_{13}	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	ΣT_1	ΣT_2	ΣT	p_1^*
1 000	2. 11	2. 18	2. 17	0. 71	3. 64	4. 35	0. 167
1 200	2. 46	2. 60	2. 57	1. 21	3. 92	5. 14	0. 247
1 400	2. 95	2. 99	2. 98	1. 60	4. 37	5 96	0 27
1 600	3. 21	3. 29	3. 27	1. 82	4. 72	6. 53	0. 283
1 800	3. 73	3. 69	3. 70	2. 37	5. 04	7. 41	0. 368

Linear Reward Inaction S-model

Table 6

λ_{13}	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	ΣT_1	ΣT_2	ΣT	p_1^*
1 000	2. 5	2. 58	2. 56	0. 67	4. 46	5. 13	0. 135
1 200	2. 58	2. 59	2. 59	1. 64	3. 54	5. 18	0. 217
1 400	3. 11	2. 96	3. 00	1 73	4. 28	6. 01	0. 277
1 600	3. 55	3. 51	3. 53	2. 53	4. 53	7. 06	0. 356
1 800	3 85	4. 04	3. 98	2. 37	5. 61	7. 98	0. 307

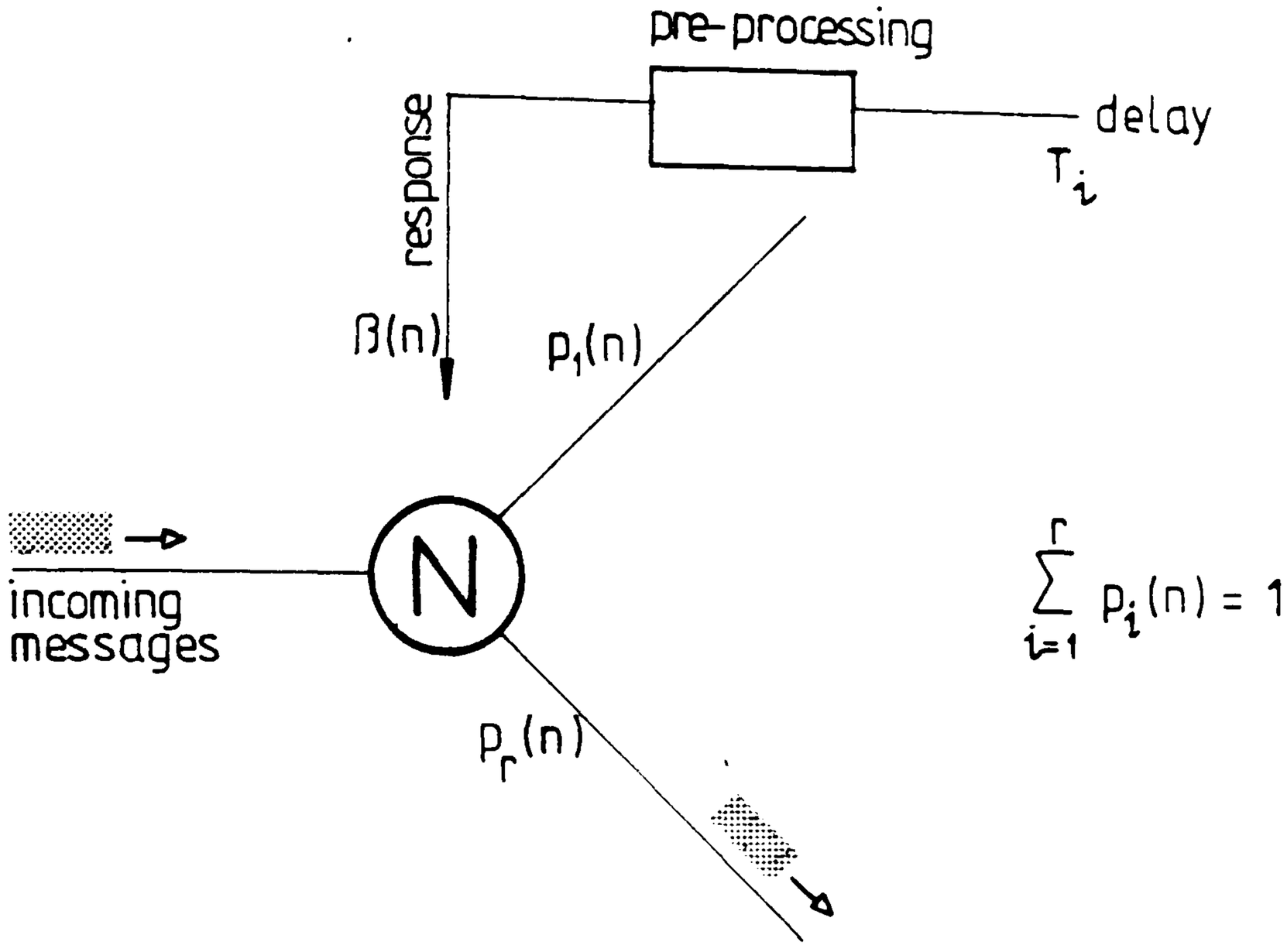


Figure 8·1a Automata routing elements for message switching

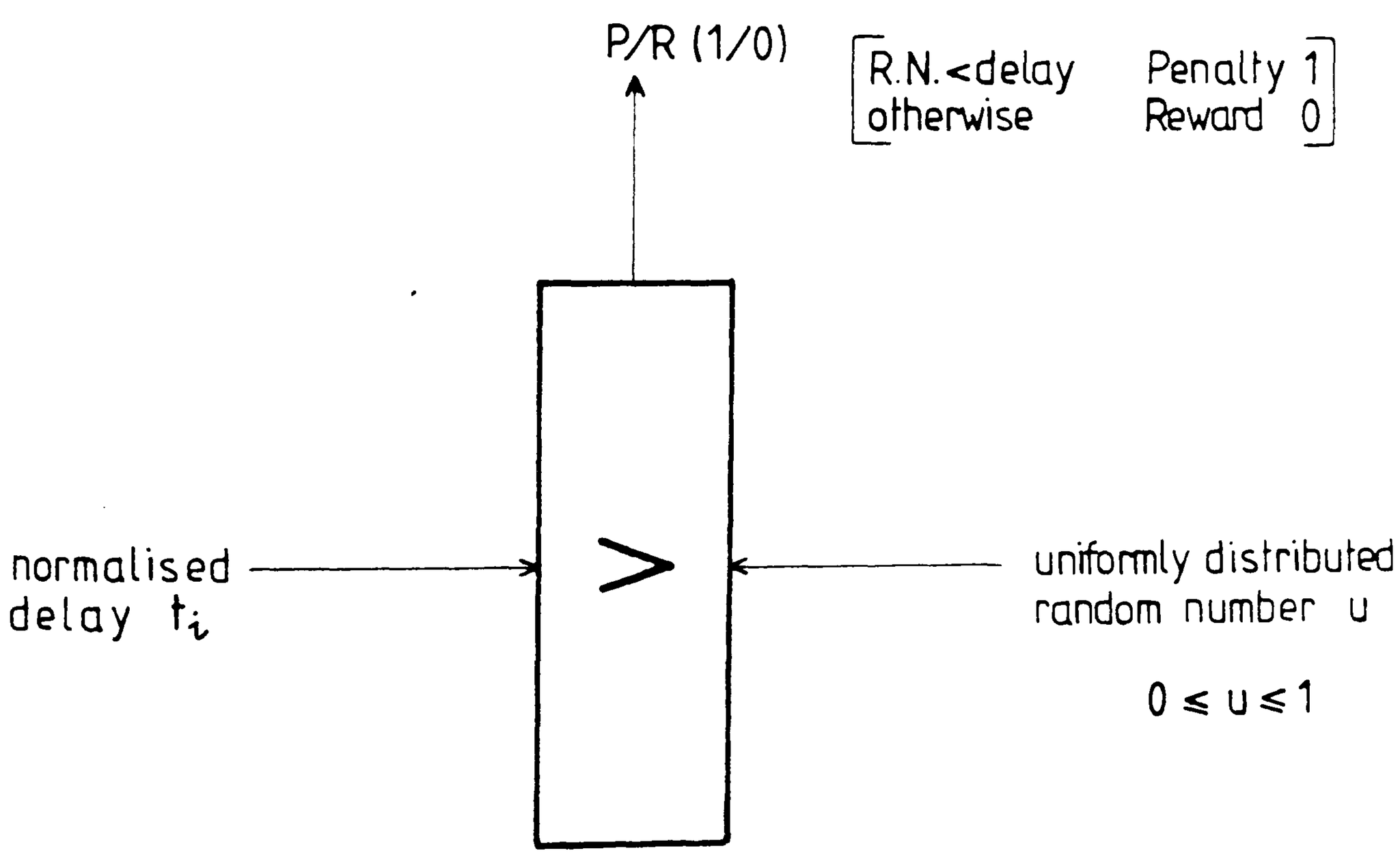


Figure 8·1b Stochastic Comparator

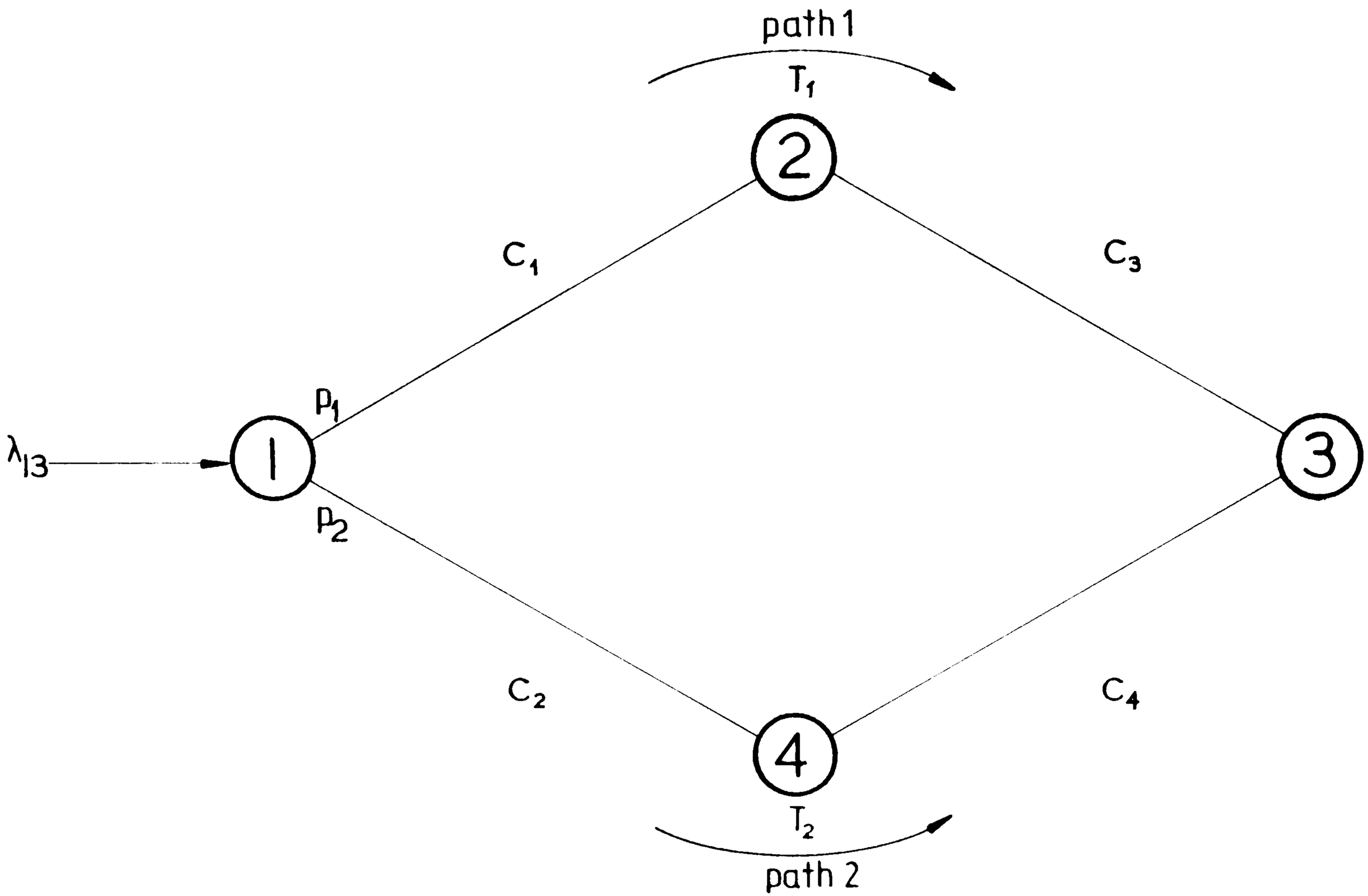


Figure 8.2 4 Node Network

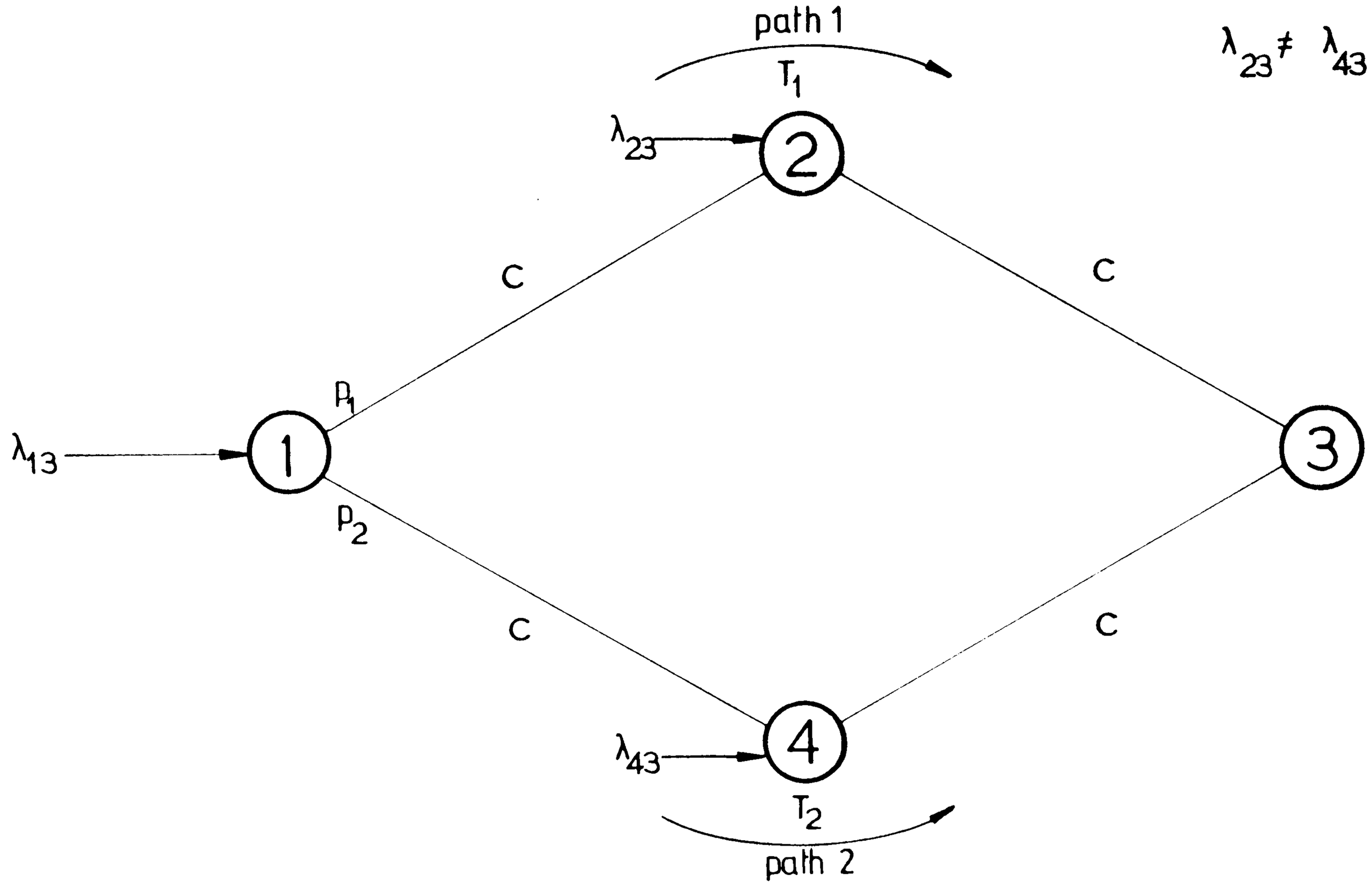


Figure 8.3 4 Node Network with Interfering Traffic

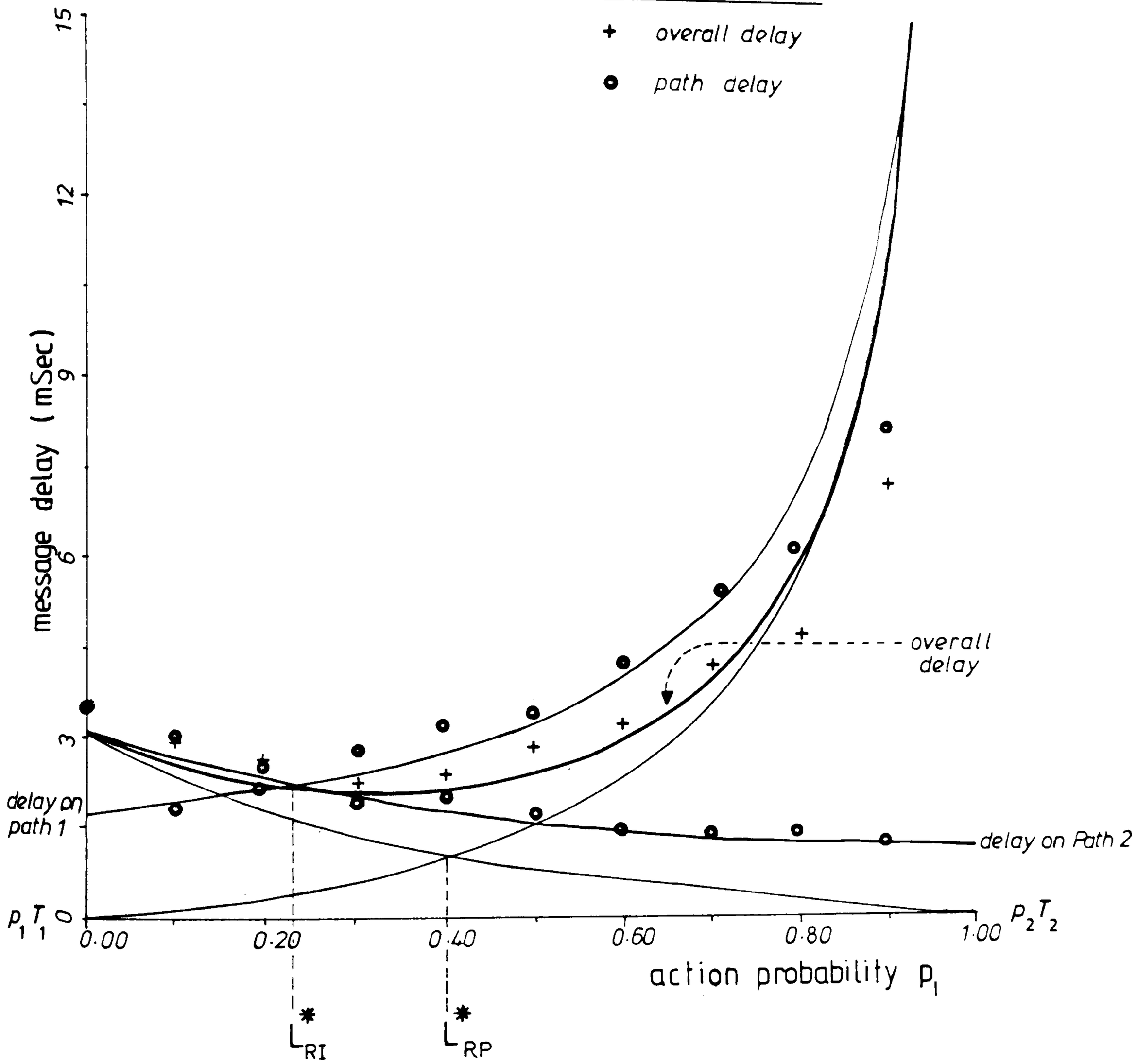


Figure 8.4 Convergence conditions in a message switched network

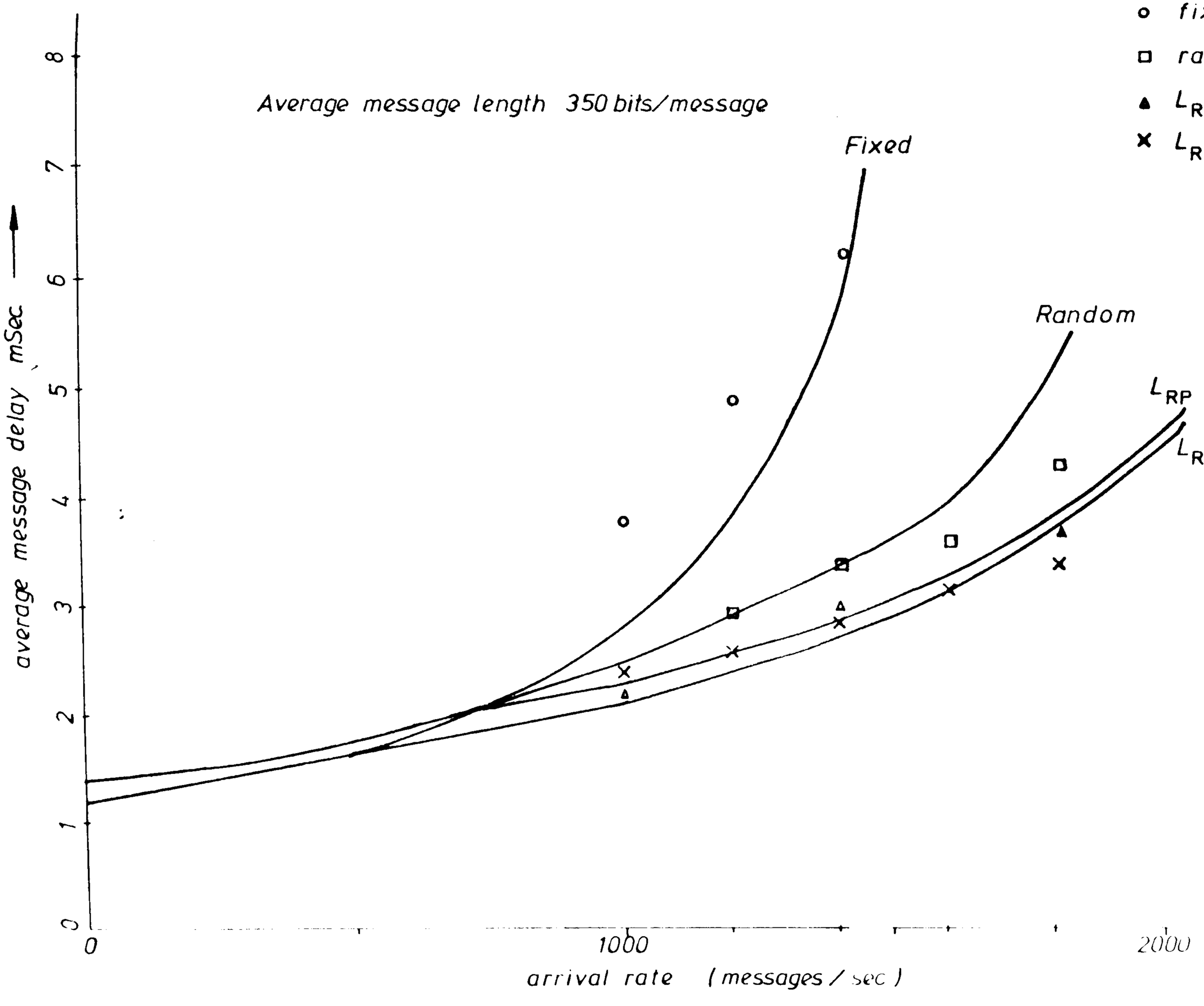


Figure 8.5 Average message delay / arrival rate

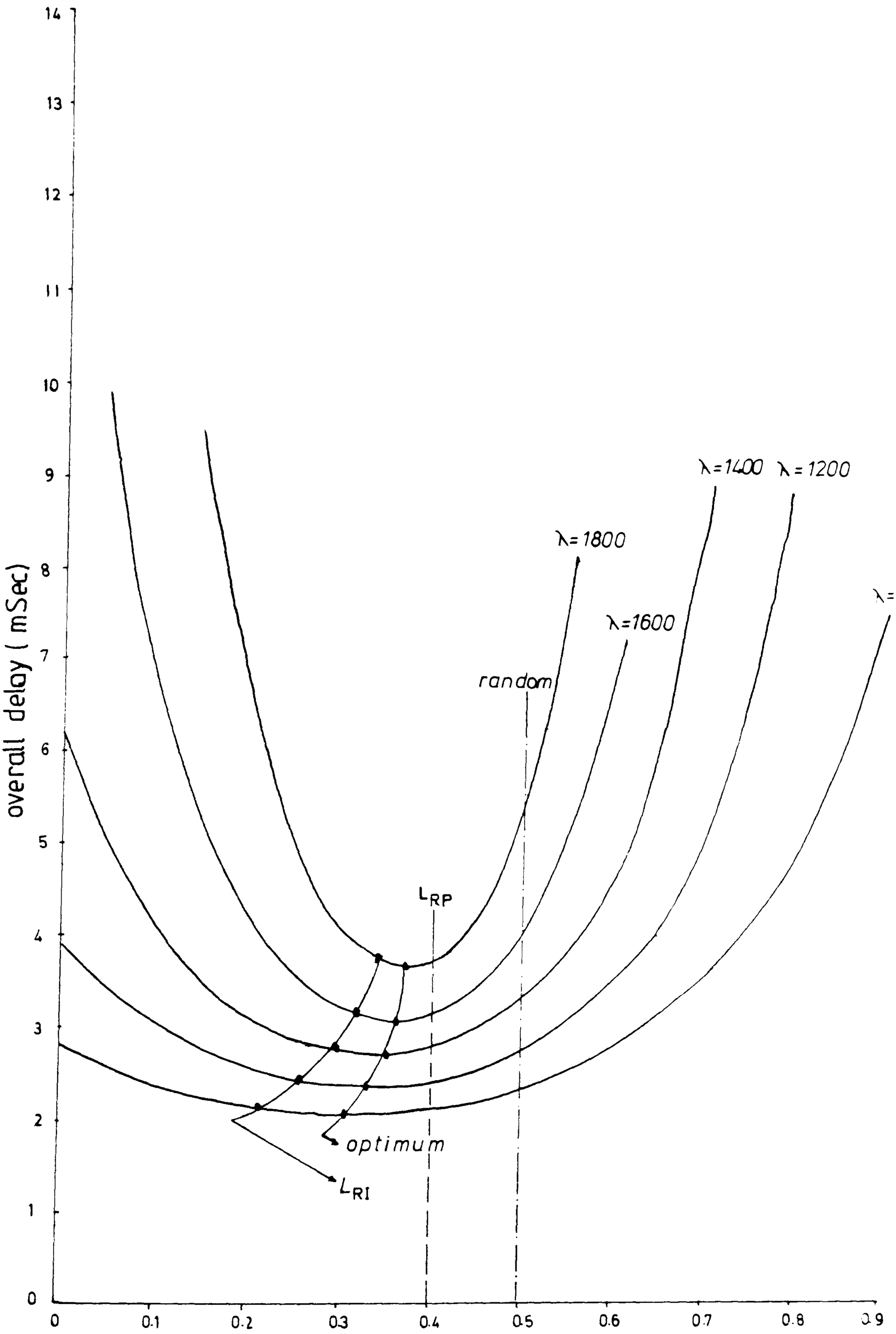


Figure 8.6 Comparison of routing techniques

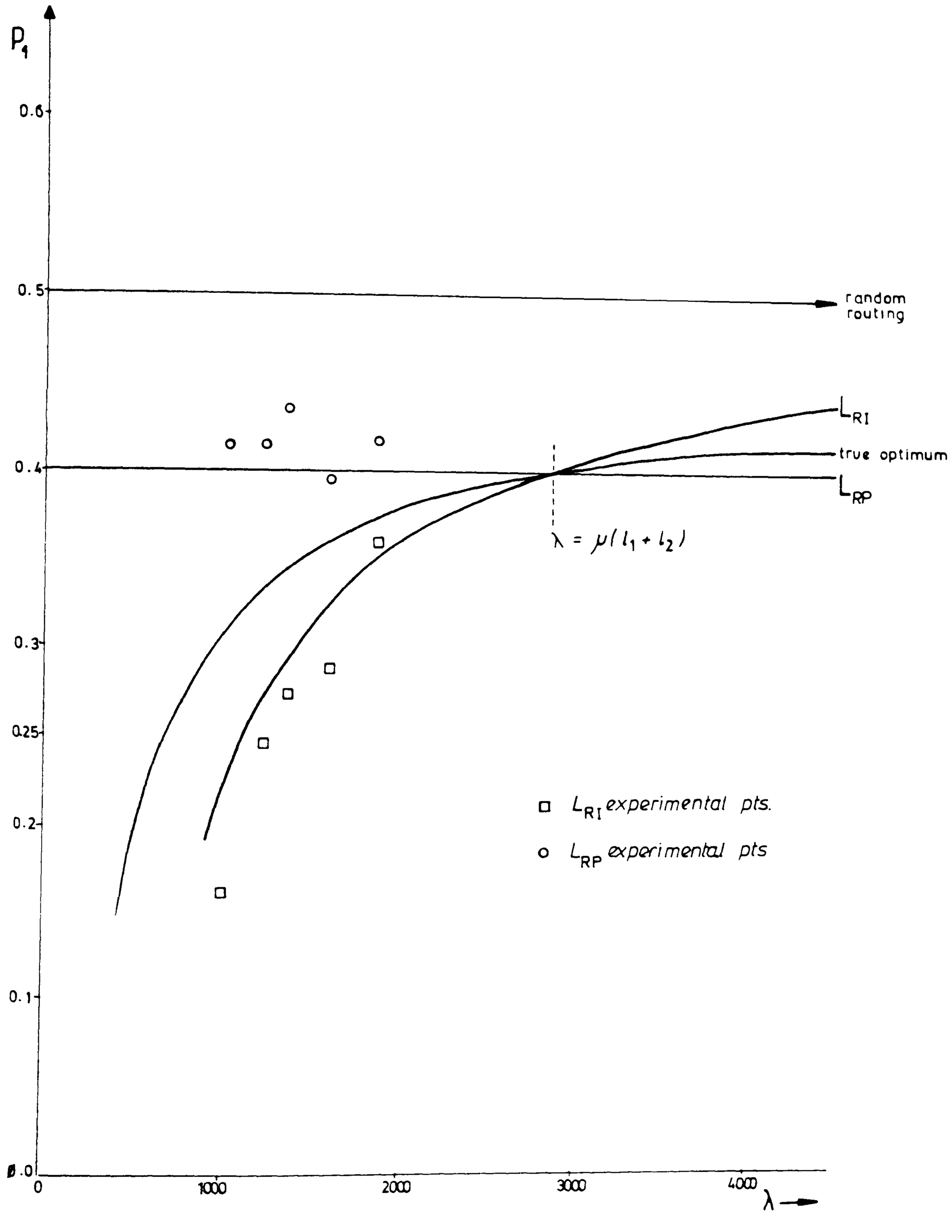


Figure 8.7 Variation of p_1 with λ

CHAPTER 9

LEARNING AUTOMATA ROUTING SCHEMES FOR PACKET SWITCHED NETWORKS

9.1 Introduction

Progressing from the detailed investigation of small networks, a study is now made of learning automata schemes for generalised datagram and virtual call packet switched networks. Initially, this chapter describes both implementations, outlining the routing and feedback operations, and suggests how the required global feedback can be transported efficiently, without increasing the levels of control traffic. Later, through extensive simulation studies, the proposed schemes are demonstrated and evaluated using realistic network examples, with further experiments comparing their performance with optimal fixed rule and cyclic random routing schemes.

The learning routing scheme for a virtual call network is a natural extension from the technique suggested for circuit switched networks, with the automata operating to establish a conceptual circuit between the source and destination nodes. The feedback in this case however, is a measure of the delay experienced on the path, this ideally carried to update the automata by means of the existing end to end control packets. Likewise in the datagram network, the measured delay forms the basis for updating the individual automata, although in contrast to the previous example, the routing operation is required to progress individual data packets.

For the simulation studies[55], a series of experiments have been performed on realistic ten node datagram and virtual call networks, using the previously described simulation packages. The experiments, which subject the networks to a range of traffic intensities, compares the learning /

learning schemes performance with optimal fixed and random schemes, presenting measurements of criteria such as the message and packet delay, average hop count and link utilisation. In additional experiments, the datagram scheme is displayed under non-stationary traffic conditions, illustrating by example, the important ability to reconfigure. The datagram scheme is also demonstrated in a self-organising role, where no apriori knowledge is assumed of the network topology.

The results to be presented in this chapter conclusively confirm the capability of learning automata routing schemes in realistic packet switched networks, the adaptive load splitting behaviour providing a near to minimum achievable delay. Moreover, the desirably practical approach to the feedback and updating operations has significant advantages over existing schemes of a similar nature.

9.2 Automata Routing Implementations for Virtual Call and Datagram Networks

This section aims to describe how the learning automaton may be used in the formation of routing schemes suitable for generalised virtual call and datagram networks. Emphasis is placed on producing designs which are compatible with existing protocols and efficient in terms of control overheads, while attempting to provide a comprehensive global feedback policy. The organisation and placement of the automata in these packet switched networks is along similar lines to the telephone networks, with the automaton A_j^k carrying out routing at node k, for any packet of destination j.

Virtual Call Scheme

In many ways, the virtual call application is similar to /

to the circuit switched network and typically requires the automata to establish a virtual connection from the source to destination, which can be used to transport a sequence of data packets.

Generally, the basic protocol of a virtual call network may be described with reference to Figure 9.1. A new message at node S, requesting transmission to D, initiates a path set-up procedure by sending a trace packet into the network. As this packet traverses the network, the routing scheme directs it to the destination and at the same time takes note of the path under formation. Confirmation of a successful operation is provided by a path acknowledge packet which returns to the source node along the newly established virtual path. At this point, the contents of the message can be sent as a group of packets, to the destination, where the message is reformed (the original order is maintained) and passed to the necessary end user. Finally, a third control packet, the message acknowledge packet, backtracks the connection, confirming the safe arrival of the message and releases the virtual path.

In a learning automata routing scheme, the virtual path is formed by a chain of routing decisions, individual nodal automata selecting an outgoing link for the trace packet. During the entire set-up operation, the delay experienced by the packet on the various sections of the network is recorded. This information is in turn "piggy-backed" by an acknowledge packet, which transports the relevant delay feedback to the automata involved in the initial routing operation. With suitable pre-processing, the delay is transformed into a response for each automaton, and individual updating operations initiated.

Both the call set-up and feedback operations are illustrated /

illustrated on Figure 9.2. From the source node i, the trace packet is progressed by the routing controllers A_j^i, A_j^m and A_j^n, to the destination node j. In turn, the acknowledge packet is dispatched with values for the delay experienced between each automa ton and the destination, e.g., A_j^m receives d_{mj}, the delay noted between nodes m and j. Using this arrangement, a sensible and practical adaptive routing technique can be installed in a virtual call network, meeting the constraints concerning the implementation and control overheads.

Datagram Network

In the datagram network, each data packet is routed as a separate entity, with individual packets of a common message travelling on completely disjoint paths to the destination. This unfortunately presents a formidable problem to provide an efficient global feedback policy, required to support the updating operations at all nodes. However, a particularly suitable method is proposed which overcomes this problem, giving all automata a global network awareness, without increased levels of control traffic.

Prior to describing the routing scheme operation, it is beneficial to basically outline the essential operations in a datagram network. When a message requesting transmission is presented to the communication system, a packet assembler/dis-assembler (PAD) procedure decomposes the message into fixed length data segments or packets, as shown on Figure 9.3a. Individual packets are then numbered and inserted into the network, to traverse independently to the destination (Figure 9.3b), utilising several paths. On reaching the proposed end node, the packets are buffered, to be finally reconstituted into the original message (Figure 9.3c). In this type of network, the /

the ability of the learning automata to adaptively proportion traffic on the various paths has therefore distinct advantages.

Similar to other L A schemes, routing decisions in the datagram network are carried out by independent automata at each node. In order to support complete global feedback, each node is also required to maintain delay estimate vectors $[\hat{d}]^k$, which contain averaged values representative of the delay between the present node k and all destinations j. Since a different route to j might exist for each routing option at k, a separate vector is made available for each outgoing link, giving $\hat{d}_j^{ka}$ as the delay between nodes k and j, assuming option a at k. During the updating operation, the elements of the delay vectors are revised in conjunction with the learning automata.

As shown by Figure 9.4, data packets routed by a node are acknowledged to be correctly received by means of a short control packet. In the learning automata routing scheme, further use is made of these packets to transport the delay feedback to the previous node, this being used to update the action probabilities of the automaton and the necessary delay vector. This information consists of two components, a local delay measured between the two nodes and a path delay value obtained from the vector at the node nearest the destination. With this arrangement, the global delay percolates through the network as packets are transmitted between nodes, the delay vectors gathering estimates of the average delay to the destinations, with the learning automata action probabilities converging to a suitable routing strategy.

In a typical routing example, the automaton at node m selects a next node for the packet destined to j. As illustrated on Figure 9.4, node n is chosen, this being reached /

reached after a local delay d_{mn} (queueing plus link transmission delay). At n a further routing decision is performed, in this case selecting link a and allowing the collection of a path delay estimate. The global delay used for updating at node m, d_{mj} is then found as follows and transferred to node m by the acknowledge packet.

$$d_{mj} = d_{mn} + \hat{d}_j^{na}$$

The actual response β_j^m to the automaton, as in the virtual call scheme, is the normalised delay d'_{mj}.

$$\beta_j^m = d'_{mj}$$

In addition to modifying the action probabilities of A_j^m, the fedback delay is also required for updating the delay estimates at node m, $\hat{d}_j^{ma}$. This is implemented by means of the widely used exponential smoothing technique.

$$\hat{d}_j^{ma} = \epsilon \hat{d}_j^{ma} + (1-\epsilon) d_{mj} \qquad 0<\epsilon<1$$

Delay Pre-Processing

In both schemes, pre-processing of the delay feedback is necessary to produce a response which suits the automata reinforcement algorithms. Typically, for the algorithms available, any delay value must be normalised to either the range (0, 1) or (-1, 1) to accommodate the S-model or transformed into a binary signal for the P-model.

One possible approach is to simply divide the delay by a normalising factor

$$d'_{mj} = \frac{d_{mj}}{d_{max}}$$

This arrangement however, is not entirely satisfactory, since a prior knowledge of the network conditions is required to select the normalising factor d_{max}.

A more useful technique for transforming the delay is given /

given by the function

$$d'_{mj} = 1 - \sqrt[n]{\frac{1}{d_{mj}/d_{min}}}$$

With this function, delay values are mapped into the region (0, 1) as shown by Figure 9.5, only requiring the absolute minimum recorded delay d_{min}. This technique has the added attraction of effectively separating low values of delay which are closely spaced and compressing the range of longer delays. For the following simulation studies, a square root law has been applied (n = 2).

Steady State Behaviour

Previously, the influence of the various reinforcement schemes has been investigated, and showed for the L_{R-I} algorithm

$$E\left[d_a\right] = E\left[d_b\right] \qquad a, b = 1, \ldots r$$

Under steady state each individual automaton attempts to equalise the average delay between the particular node and destination, for all allowable options. With all automata operating under this policy, the global effect is an equalisation of the delay on all paths open to a particular source/ destination pair.

Likewise for the L_{R-P} algorithm, the following condition was shown to hold

$$p_a E\left[d_a\right] = p_b E\left[d_b\right] \qquad a, b = 1, \ldots r$$

This produces an equalisation of the accumulated delays for the various options and has the effect of balancing the link utilisation on the paths.

9.3 Network Measurements /

9.3 Network Measurements

Several network measurements are now described, which provide convenient performance conditions for the network/routing scheme combinations to be studied. In addition to the previously discussed routing scheme entropy the important measures in this application include, the message and packet delay, average hop count and link utilisation.

Delay

The end to end delay is justifiably an important parameter in a network which involves packet queueing and transmission. In this study, the average packet and message delays are specified, the packet delay defined as the time for a packet to cross from source to destination, the message delay being the delivery time for a complete message (group of packets).

Further, since the delay is a random variable, conventional statistical parameters such as the variance and higher order moments provide additional information on the delay characteristics of the network. These are suitably displayed by means of delay distribution plots which show the relative frequency of a packet delay occurring within a certain time band.

Link Utilisation

The object of the link utilisation measurement is to record the levels of traffic handled by each link in the network, highlighting the relative utilisation and showing sections which are under/over utilised. Graphically, this is best displayed in a histogram form.

Average Hop Count

The average hop count $\bar{n}$, quantifies the mean number of hops required by packets on the path between source and destination. This measurement helps in the understanding of /

of how the network resources are used, a minimum hop count showing efficient operation. This may be calculated by

$$\bar{n} = \frac{\text{Number of Packet Transmissions}}{\text{Number of Packets Entering the Network}}$$

9.4 Simulation Studies

The following experiments are carried out to demonstrate the learning automata routing schemes and to compare them with some existing techniques. Testbed networks for this study, covering the virtual call and datagram protocols, are detailed in Appendix 2, the initial designs originating from external sources[46][56]. Like the Bell circuit switched network, these networks provide a realistic environment for the new routing scheme, and also allow experiments with optimal shortest path and random routing schemes.

In this simulation study, a wide range of experiments have been performed, initially concerning single traffic sources and the complete network. These permit the study of individual automata, displaying typical results for the steady state action probabilities and the average response to each action. Subsequent simulation experiments have been carried out using complete traffic sets (90 sources), studying the performance of both network models, for a range of traffic intensities.

Learning routing schemes are a particular benefit under non-stationary conditions and accordingly simulations are presented which involve dynamic traffic switches, aiming to introduce interfering traffic on arbitrary links. In a final example, the operation of a self-organising scheme is demonstrated, illustrating the capability of the learning routing /

routing technique in networks which are globally unknown.

Single Source Traffic

Datagram Network

In this case, a single source of messages was established at node 1, generating traffic for node 9, this being maintained at a rate of 10 messages/sec for all experiments.

The shortest path fixed rule for messages between nodes 1 and 9, dictates the route 1, 3, 4, 7, 9, a length of 4 links. After running an experiment for a 1 000 message sample an average message delay of 212 mSec was obtained. Alternatively, the random routing scheme for λ_{19} traffic is shown on Figure 9.6, this also displaying the measured packet delay between each node and the destination, on the allowable routing options. In this particular example, the overall message delay was found to be 193 mSec. This lower delay might be expected, since in comparison to the single route policy of the shortest path technique, the random routing scheme possesses the ability to spread traffic over several paths, tapping a greater effective transmission capacity.

The learning automata scheme may be regarded as an adaptive variant of the random routing scheme, with the same options open to the automata placed at the various nodes in the network. Figure 9.6 also indicates the results obtained from an experiment with this scheme, showing the steady state average delays and action probabilities. For this experiment, a significantly improved performance is achieved, the average message delay recorded at 141 mSec.

A close study of the combined automata conditions reveals the interesting convergence strategy, with automata proportioning traffic over certain routes and on others, dropping allowable options which do not contribute to a satisfactory /

satisfactory performance (nodes 2, 7 and 8). For the remaining options, the automata, which have L_{R-I} reinforcement algorithms, attempt to equalise the average delay associated with each action.

Virtual Call Network

Along similar lines to the single source datagram (Figure 9.7) experiments, a further series have been completed using the virtual call network and routing schemes. In this example, the source of traffic is placed at node 10, generating traffic for node 1 at a rate of 7.5 messages/sec.

For the optimal fixed rule (10, 8, 6, 4, 1) the average message delay of a 1000 message sample was measured at 401 mSec. Likewise for the random routing scheme Figure 9.7 shows the allowable options and the measured trace packet delays between the nodes and the destination. This experiment yielded an average message delay of 359 mSec.

The best message delay results are obtained from the learning automata experiment, with a value of 340 mSec. Additionally, Figure 9.7 illustrates the steady state conditions for the entire scheme, which reveal a similar convergence strategy to the datagram example, with the mixed policy of dropped options and traffic splitting.

Complete Traffic Set

Now, both networks are fed from the complete traffic sets of 90 sources, each node generating traffic for every other. The experiments yield the network performance over a range of traffic rates, the overall arrival rate varying from 10 to 100 messages/sec.

Datagram Network /

Datagram Network

Average Delay

The graphs for the average message delay (Figure 9 8a) and packet delay (Figure 9. 8b) plots the delay/arrival rate characteristics for the shortest path, random and learning automata routing schemes.

Comparing first the fixed and random routing. At low message rates, the shortest path scheme outperforms the random scheme since the delay is predominently due to the link transmission times and consequently the best policy is to follow the minimum hop routes. However, as the levels of traffic increase, significant queueing develops in the network, which causes the single path strategy of the fixed rule to result in higher delays. Alternatively, the random routing scheme tends to spread the traffic throughout several paths in the network, which ultimately helps control the increase in delay.

An interesting result is achieved by the learning scheme, which at low traffic intensities produces identical delay characteristics to the shortest path rule. With increasing traffic the learning automata provide a significantly better routing service, with a clearly lower message and packet delay. This behaviour can be explained along similar lines to the previous argument, although in this case the ability to control the traffic proportions on the various routes results in a superior performance.

Delay Distribution

To provide further information on the delay characteristics, the data from selected experiments is displayed in a discrete distribution form, showing the relative frequency of delay values within certain time bands. From these plots additional /

additional parameters can be noted, including the variance and maximum delay.

Distribution plots are provided for each routing scheme, at low (10 mess/sec) and high (90 mess/sec) traffic intensities. At the low end, the fixed rule (Figure 9.9a), random (Figure 9.9b) and the learning routing schemes (Figure 9.9c) give virtually identical results, the only notable difference being the slightly higher average delay of the random routing. By inspection, it may be observed that the maximum packet delay never exceeds 350 mSec, for any of the routing schemes.

With an overall arrival rate of 90 messages/sec, substantial changes may be noted in the distribution plots. In comparison to the fixed rule (Figure 9.10a) and the random routing (Figure 9.10b), the learning scheme (Figure 9.10c) produces a delay distribution which displays significant reductions in the average and maximum delays. A lower variance is also shown, this reflected in the shorter "tail" of the plot, with less components in the higher delay regions of the graph.

Average Hop Count

The average hop count (Figure 9.11a) is an ideal measure for studying the performance of the routing scheme in relation to the network parameters. Predictably, the optimal fixed rule establishes a lower bound for this, since the original design criterion for this scheme was to route packets on the shortest path to the destination. The random routing scheme on the otherhand produces a noticeably increased hop count, although like the fixed rule, this remains constant for the entire range of traffic rates.

The hop count for the learning scheme reveals an interesting result, the graph showing a shallow rise with increased /

increased traffic rate. It is suggested that this is a consequence of the convergence behaviour of the scheme, which at low arrival rates tends to converge to similar conditions as the shortest path rule, hence only using the minimum hop routes. At the other extreme, the automata find the best strategy is to spread the traffic over several routes, which need not be the shortest.

Routing Scheme Entropy

The transient behaviour of the adaptive routing scheme is suitably shown by the graph of entropy against messages handled (Figure 9.11b), this producing the entropy drop as the learning automata converge. In a typical experiment (λ = 50 mess/sec) the initial entropy is noted at 66.34 bits and reduces to approximately 30 bits as the routing structure evolves.

Link Utilisation

A further useful technique to assess the performance of the various routing schemes is to study the utilisation or total packets carried by individual links in the network, drawn in a histogram form.

As a direct consequence of the invariant routing policy of the fixed and random routing schemes, the corresponding histograms show no alteration for different traffic conditions. A distinctive feature of the fixed rule utilisation plot (Figure 9.12a) is the apparent spread in the individual link traffic, some links being under utilised, while others operate with significantly higher traffic. A contrasting result is given by the random routing (Figure 9.12b), which shows a "smoother" histogram, although on average each link carries a greater load.

Figure 9.13 illustrates the results for the link utilisation offered /

offered by the learning scheme, graph a for an initial arrival rate of 50 mess/sec and graph b for 90 mess/sec. These show a clearly reduced average utilisation, tending to that of the shortest path scheme, without the same spread in the individual link loads. This effect is even more discernible at the higher arrival rate, where as suggested by the delay characteristics and supported by the average hop count results, the learning scheme adopts a more pronounced load splitting strategy, spreading the traffic over any available network capacity.

Virtual Call Network

Using the 10 node virtual call network and associated traffic set, a series of experiments were run for a range of arrival rates. These provide an opportunity to study the learning automata scheme as part of a virtual call operation and to compare it with an optimal fixed rule and random routing scheme.

The average message (Figure 9.14a) and packet delay (Figure 9.14b) characteristics indicate the operation of the learning scheme to be similar, but less effective in the virtual call network than in the previous example. Again the learning routing approach is shown to be a clear advantage at the high traffic intensities, although in the lower regions a slightly sub-optimal service is detected. It is suggested that this results from the reduced ability to spread traffic, a virtual call routing operation only influencing sequential groups of packets.

Results are also presented for the average hop count (Figure 9.15a) and routing scheme entropy (Figure 9.15b) for the experiment at λ = 50 mess/sec. The hop count plot is similar in nature to the datagram experiments, the fixed /

fixed rule showing the minimum value, closely approached by the learning scheme, with the random scheme consistently higher. Likewise, the entropy result also produces comparable behaviour, although, this tends to illustrate a reduced convergence rate, due to the less frequent updating operations.

Non-Stationary Traffic Experiments

In the following experiments, the ability of the learning scheme to reconfigure in the light of new traffic conditions is tested, using the datagram network. Again, in order to study the operation of individual automata, the traffic set is reduced, with message sources generating main-stream traffic ($\lambda_{19} = 8$, $\lambda_{59} = 5$) and interfering traffic ($\lambda_{57} = 8$, $\lambda_{34} = 3$). The following delay results were obtained from an experiment where the interfering traffic is dynamically injected after 1 500 messages.

		Before	After	
Fixed Rule	λ_{19}	322	377	mSec
	λ_{59}	252	411	
Random	λ_{19}	187	250	
	λ_{59}	120	188	
Learning	λ_{19}	170	174	
	λ_{59}	112	138	

The above table gives the average message delay for the individual (main-stream) sources, as produced by the various routing schemes. The measurements are recorded before and after the switch and show the effect of the alteration in traffic conditions. Noticeably the invariant routing schemes cannot deal adequately with the change, their results showing significantly increased delay. The learning scheme on the otherhand, by virtue of the ability to adaptively proportion the traffic, settles down with only slightly increased delay. /

delay.

The dynamics of the traffic switch are illustrated on Figure 9.16, which displays learning curves for the automata at nodes 1 and 5, both responsible for traffic to 9. Prior to the switch A_9^1 assumes a relatively unbiased strategy, splitting traffic equally ($p_1 \approx 0.5$) on the two allowable paths, while the automaton at node 5 tends to send more packets via node 7 ($p_1 \approx 0.35$). Through the introduction of additional traffic on links 3, 4 and 5, 7, A_9^1 experiences a high delay on action 2 (node 3) and consequently p_1 increases to approximately 0.6. Similarly, with interfering traffic adding to the delay of packets routed by action 2 (node 7) of A_9^5, p_1 increases to 0.55, attempting to equalise the delay met on the 3 link path with the increased delay on the 2 link alternative.

Self-Orgainising Routing

Self-organising routing schemes assume no apriori knowledge of the global network topology, although each node is fully aware of any directly connected neighbours. In this experiment a self-organising datagram scheme is demonstrated, in which any link leading from a node is considered an allowable routing option. However, to prevent unnecessary local cycling of packets and long term undelivered packets, constraints are imposed which restrict packets being routed to the previous node and drop packets with hop counts which exceed 10. The following results were obtained from a simulation of the datagram network, with traffic at an overall rate of 50 messages/sec.

Delay Results

A graph of the average packet delay against messages handled (Figure 9.17a) suitably demonstrates the operation of /

of the self-organising scheme. Initially, when the routing probabilities are in an unbiased state, a high average delay is detected, due to packets taking completely unsuitable options at a node. After convergence, a steady state condition is reached which shows a significantly reduced delay performance. Clearly, the transient phase exhibits a desirably short learning period, with steady state reached after approximately 3000 messages.

Average Hop Count

The graph for the average hop count (Figure 9.17b) tends to support the points noted in the delay characteristics, with the similarly placed transient and steady state operating regions. With this particular measure the routing schemes initial lack of structure is shown by the high average hop count, packets tending to follow any path to the destination. After convergence is completed, the packets are routed along the most direct paths to a destination, hence the reduced hop count.

Routing Scheme Entropy

In contrast to the learning scheme with the constrained routing options, the initial entropy (Figure 9.18a) of the self-organising scheme is greater, reflecting the change in routing structure introduced by the expanded routing options. As the scheme converges this is shown to decrease significantly, approaching the conditions recorded in the earlier example.

Dropped Packets

Figure 9.18b provides a graph for the accumulated number of dropped packets as the simulation proceeds. This plot presents further evidence to support the self-organising ability of the routing scheme, showing in the early regions of the experiment, a high percentage of packets being dropped /

dropped due to excessive hop counts. In the steady state region, once the routing scheme is organised, no more calls are dropped, suggesting a more appropriate routing structure, for the particular network topology.

9.5 Conclusions

The adaptive load splitting strategy offered by learning automata routing has been demonstrated in virtual call and datagram packet switched networks. Although marginally sub-optimal with respect to the true minimum achievable delay, this approach has been shown to have definite advantages in the simplicity and practicality, and could if required, be readily implemented in existing networks with minimal inconvenience and overheads. In addition, the flexibility afforded by the modular reinforcement algorithm, will in the future, allow the installation of schemes which provide minimum delay routing.

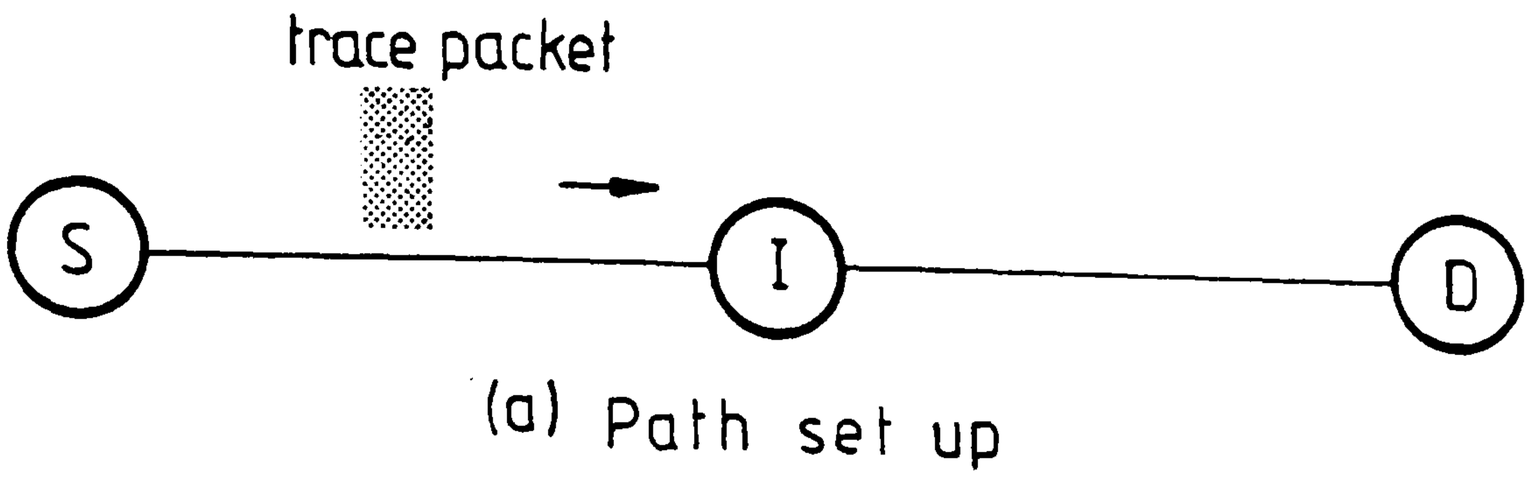
On the basis of the experimental studies, the following conclusions are noted.

1 In both the datagram and virtual call networks, the learning automata scheme operates using a mixed strategy of shortest path and traffic division routing. Typically, when several paths of approximately the same length exist, the automata adopt a load splitting behaviour, adaptively proportioning traffic over the allowable paths. In circumstances where a mixture of short and long paths are possible the automata tend to converge optimally to a particular option, /

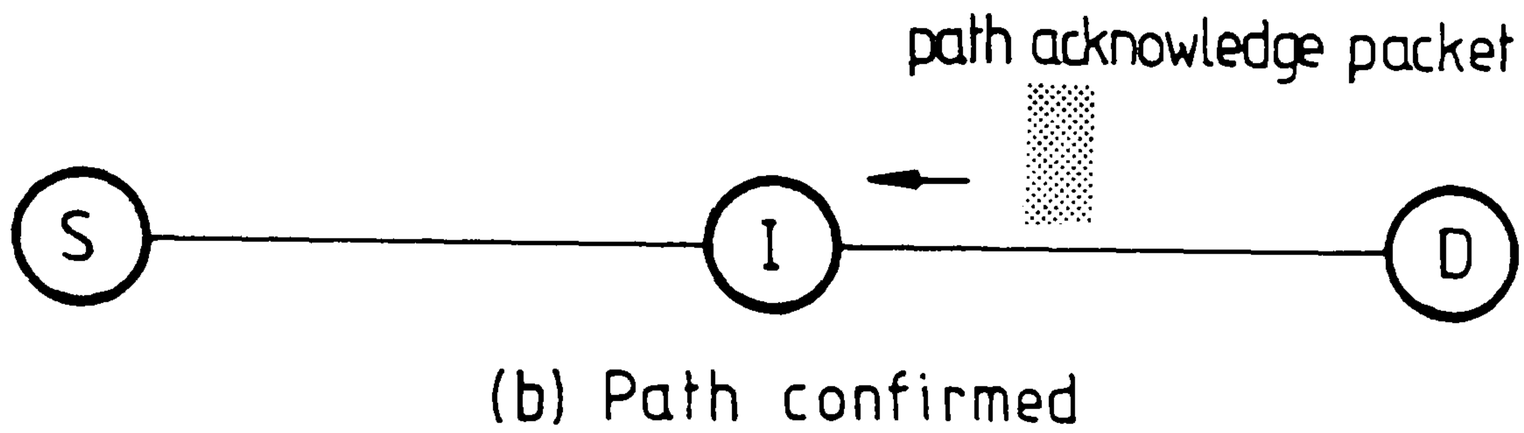
option, effectively dropping the less attractive alternatives.

2 The results from the datagram experiments reveal a satisfactory service from the learning scheme, this showing a clearly lower average delay, particularly at high levels of traffic. When the traffic conditions are light, the automata converge to produce an identical performance to the optimal shortest path strategy. Significantly at the increased traffic conditions, the adaptive load splitting service permits a more effective utilisation of the network capacity, the spreading of packets helping to reduce the queueing delay.

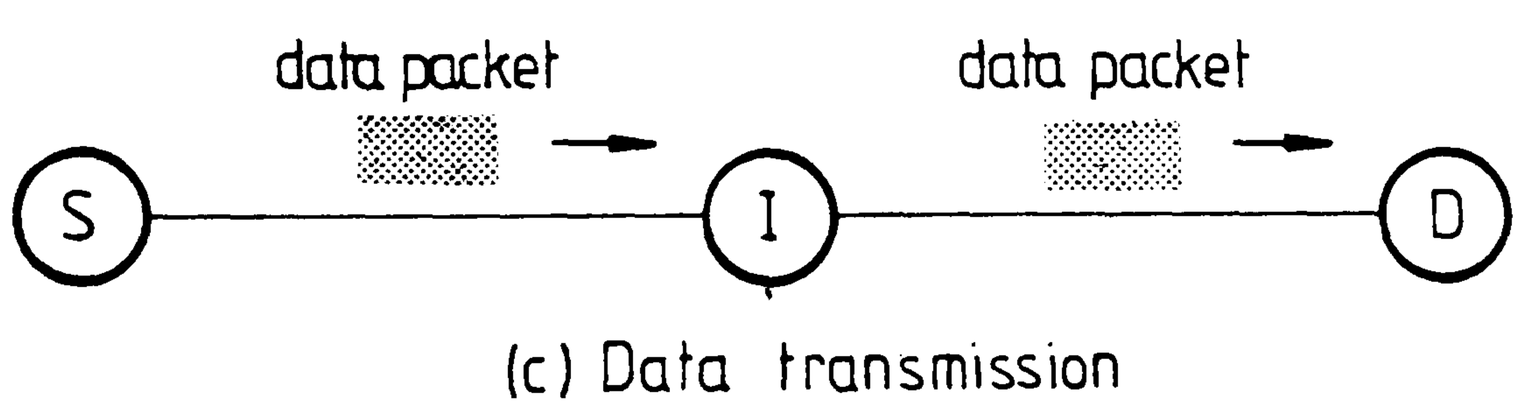
3 In comparison to other load splitting routing algorithms, the learning automata technique offers a desirably practical approach to the problem. Furthermore, the experiments suggest the technique to be certainly adequate in terms of convergence rates and stability, although this is an area where additional studies will be essential.



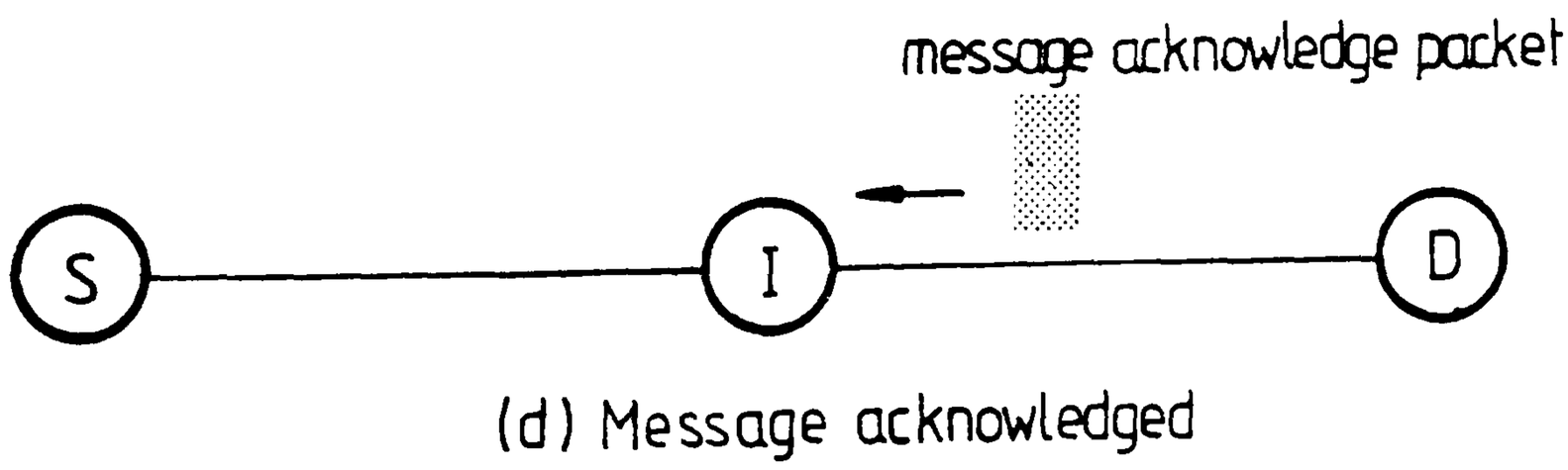
(a) Path set up



(b) Path confirmed



(c) Data transmission



(d) Message acknowledged

Figure 9.1 Virtual call protocol

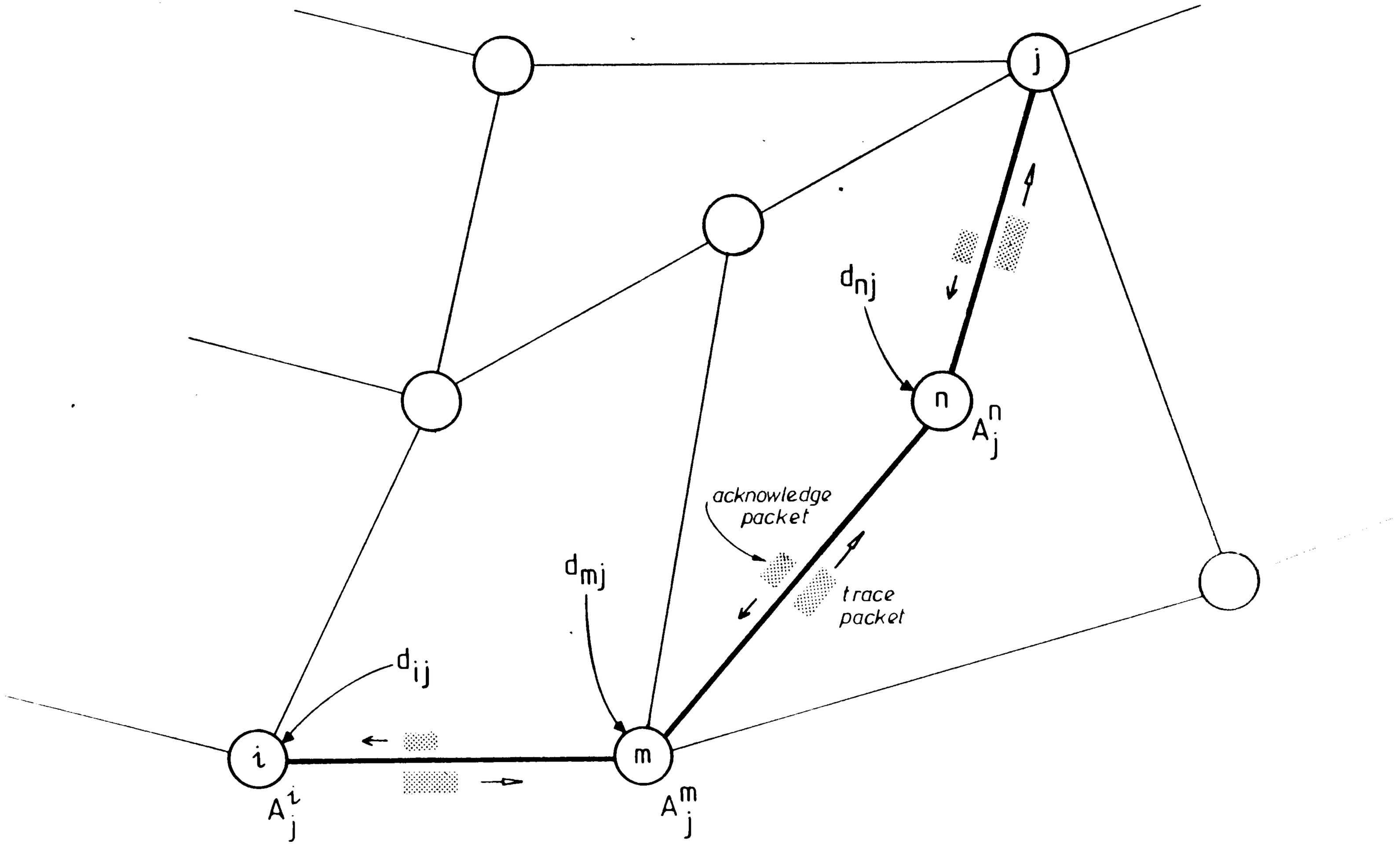
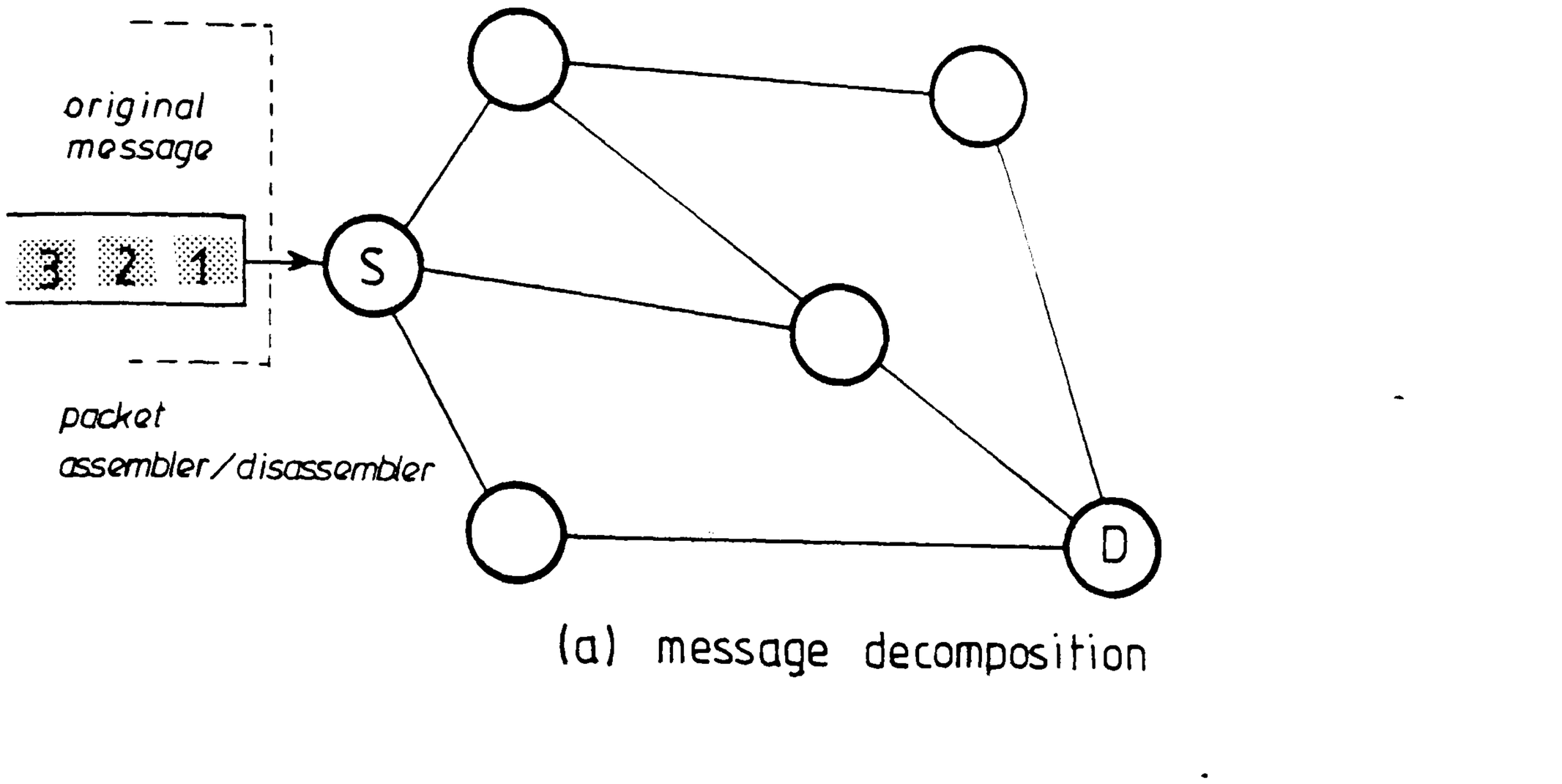
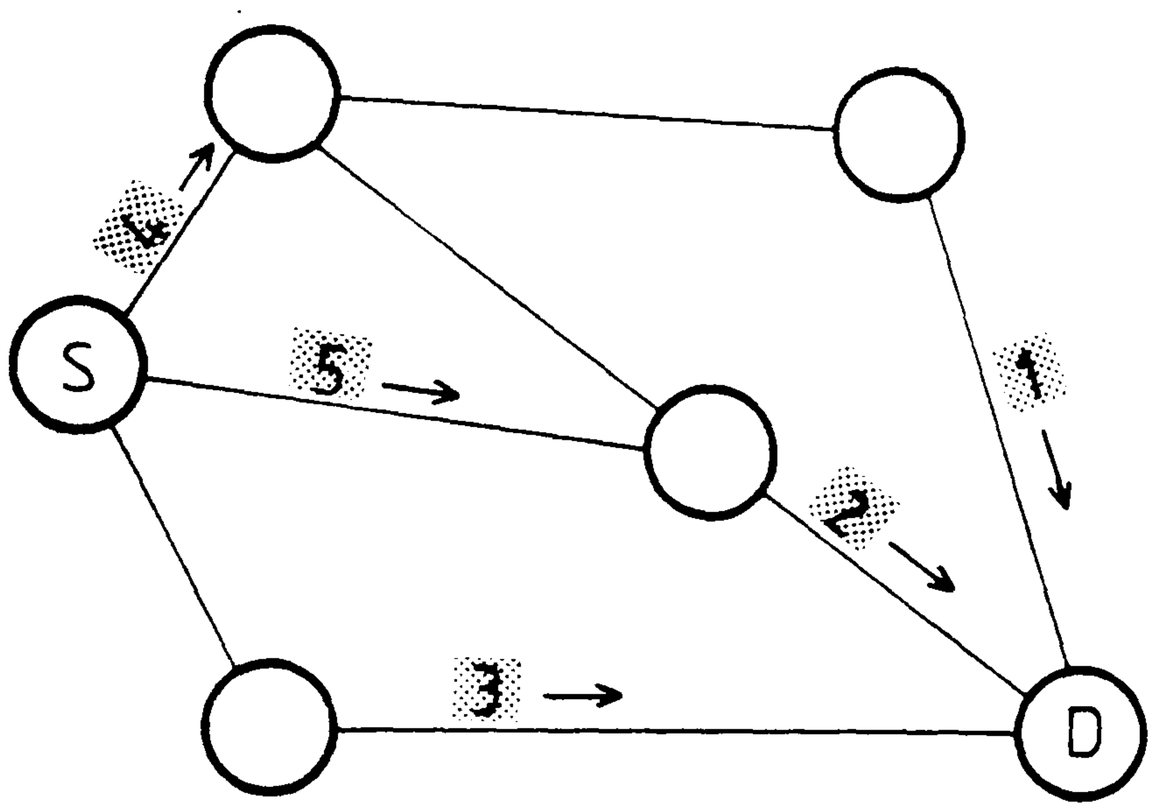


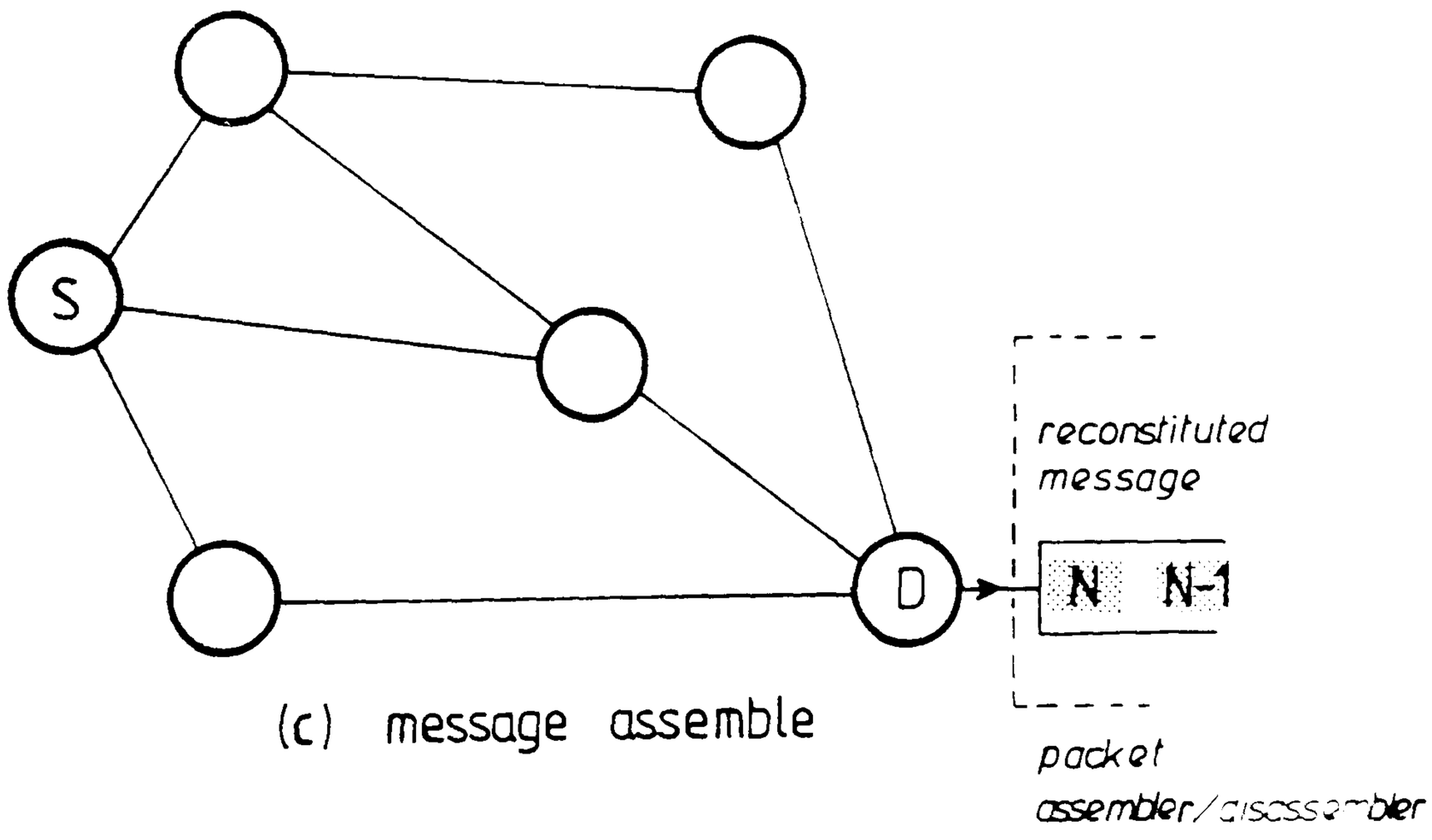
Figure 9.2 Virtual call routing technique



(a) message decomposition



(b) message transmission



(c) message assemble

Figure 9.3 Datagram protocol

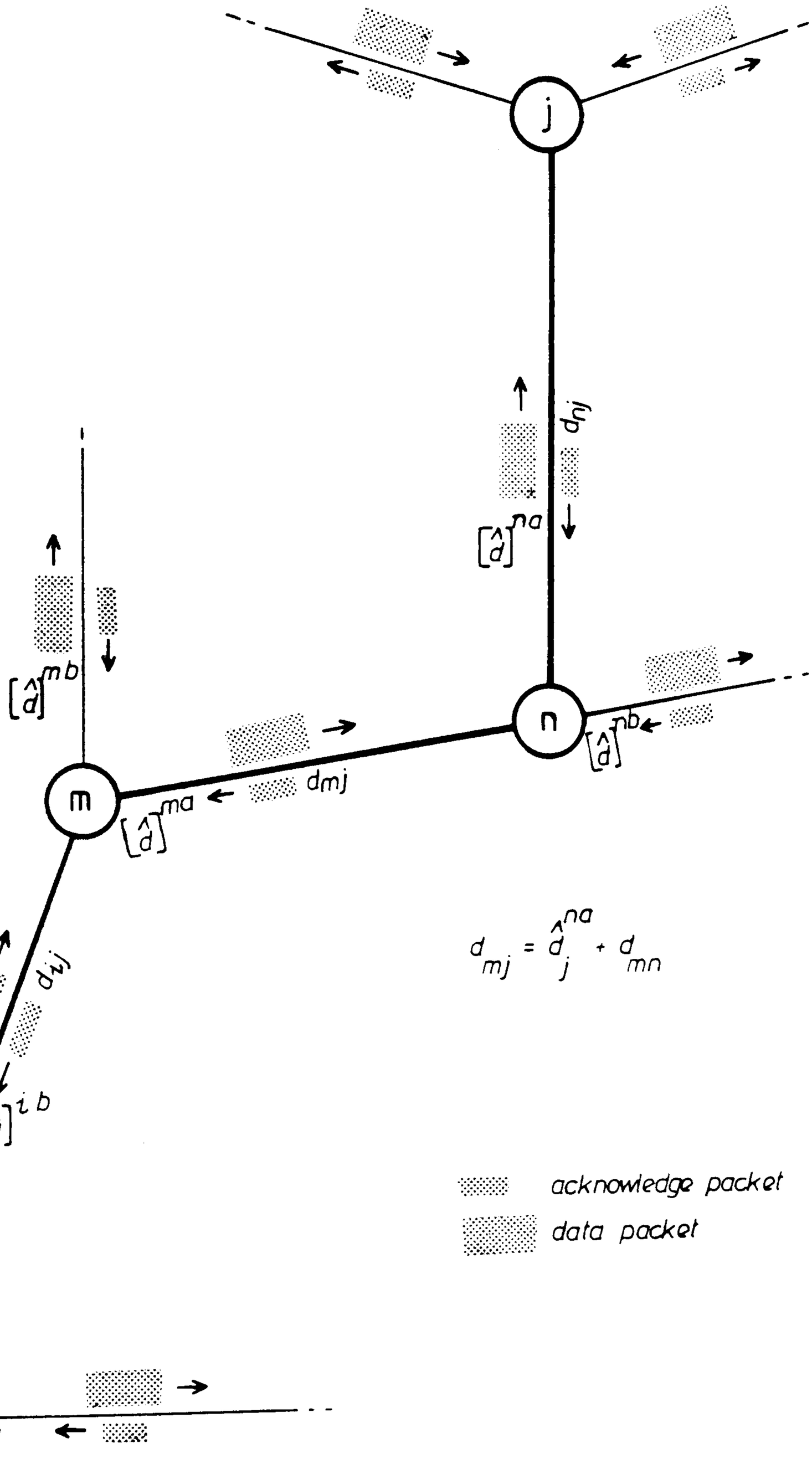
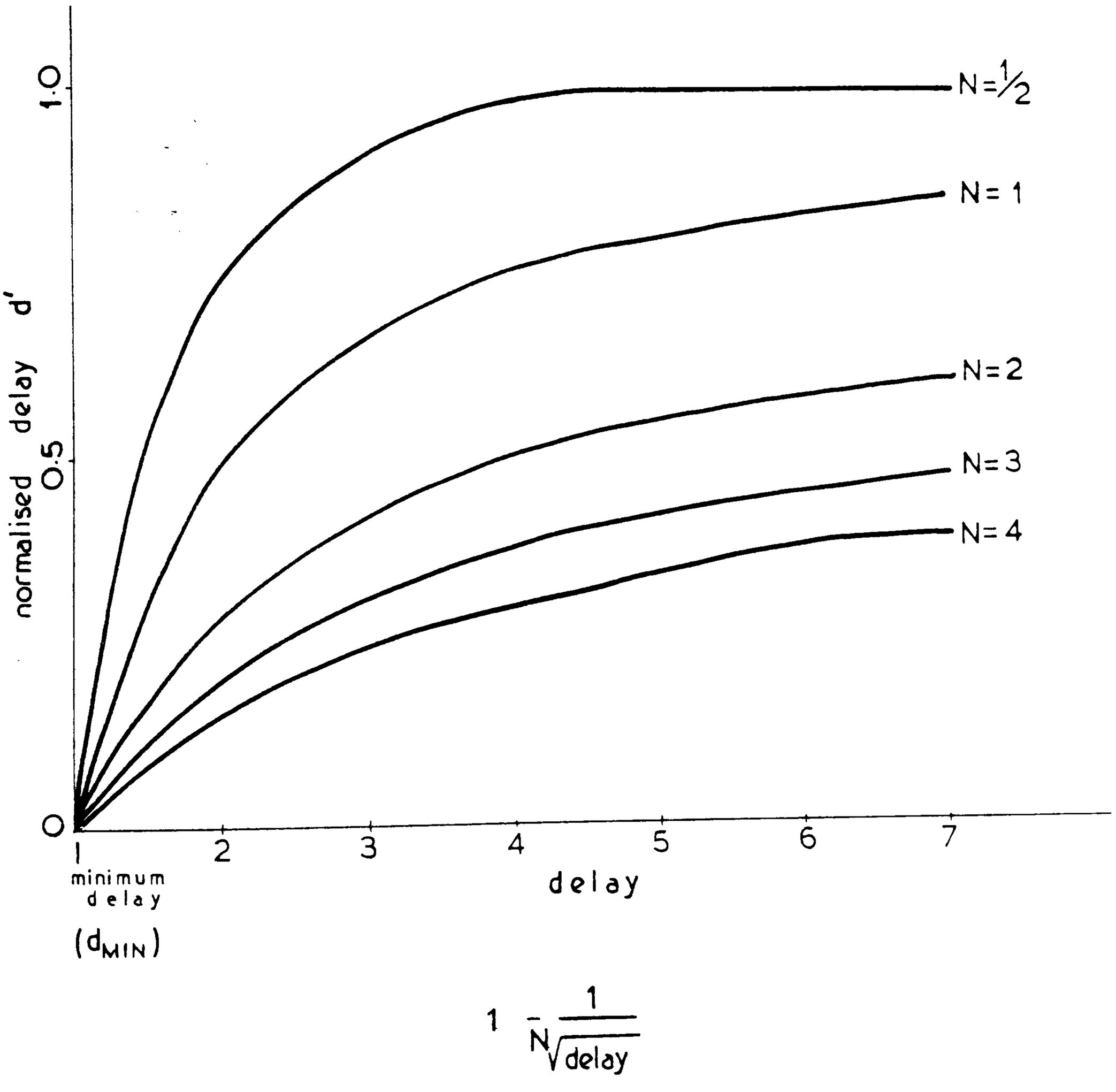


Figure 9.4 Datagram routing technique



$$1 - \frac{1}{N\sqrt{\text{delay}}}$$

Figure 9.5 Delay pre-processing function

Average message delay

Fixed = 211·8 mSec.

Random = 193 mSec.

Learning automata = 147·3 mSec.

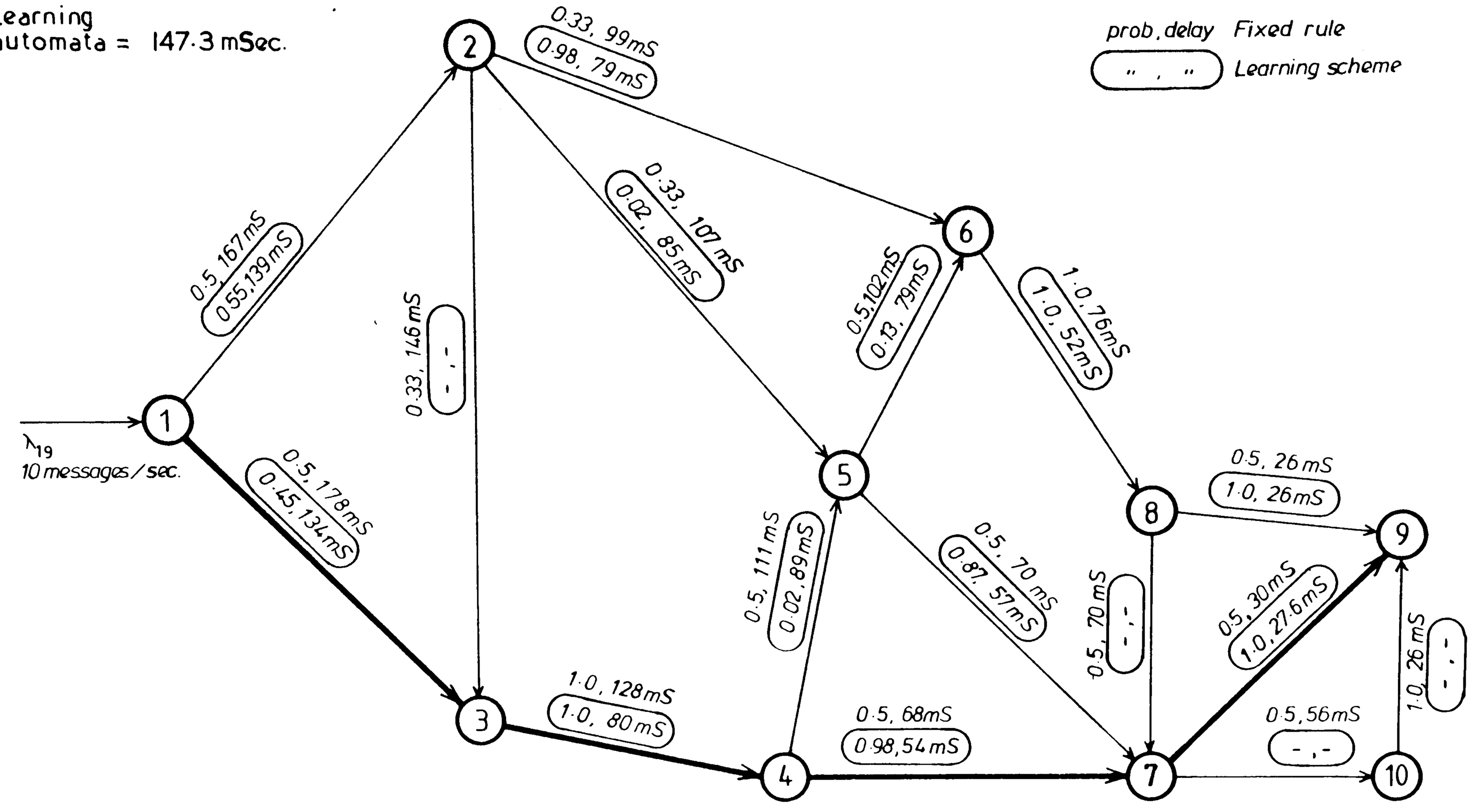


Figure 9·6 Random and L.A. routing strategies for λ_{19} (Datagram network)

average message delay

Fixed = 401 mSec.
Random = 359 mSec.
Learning automata = 340 mSec.

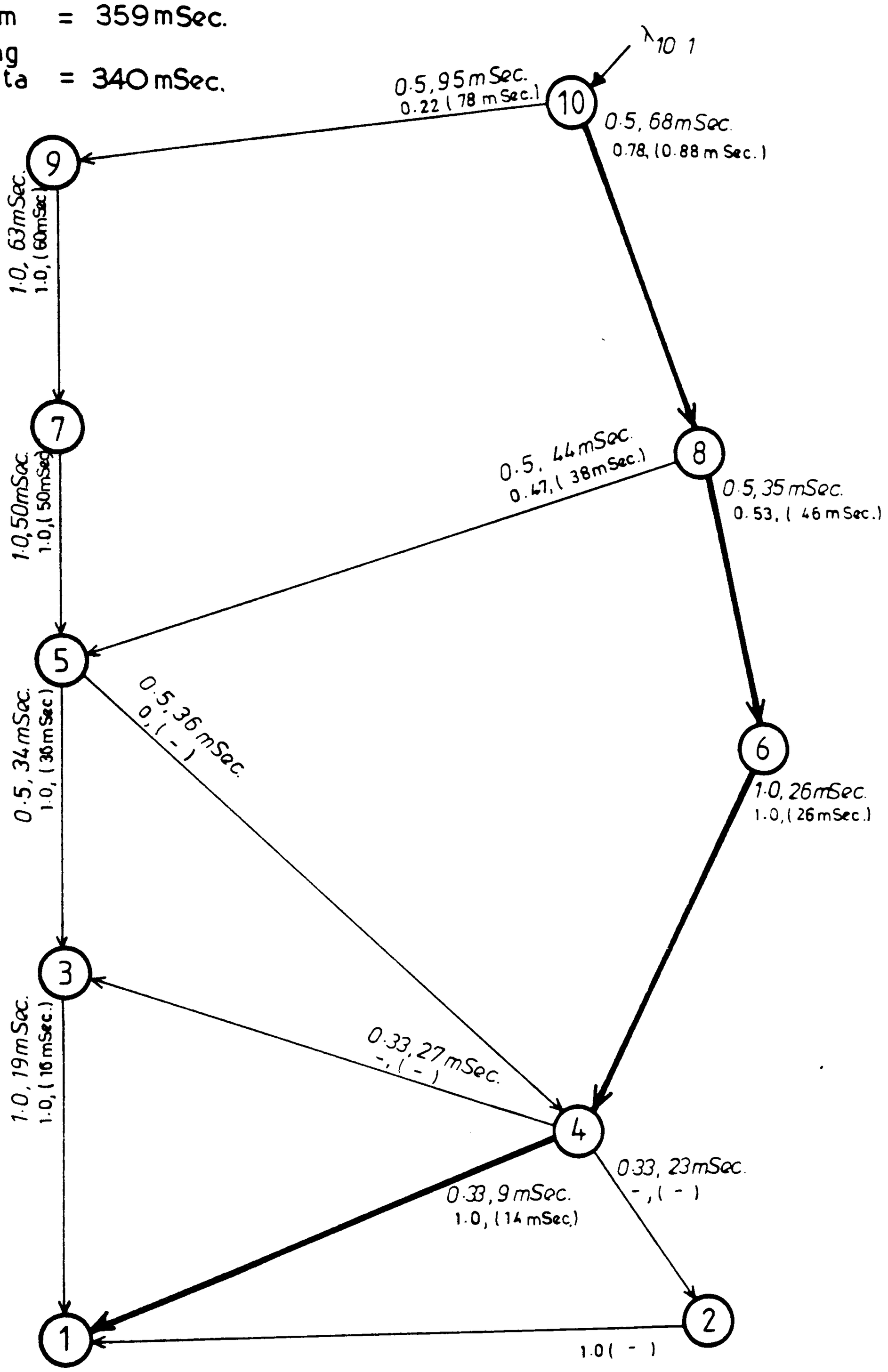


Figure 9.7 Random and L.A. routing strategies for $\lambda_{10\,1}$ [Virtual call]

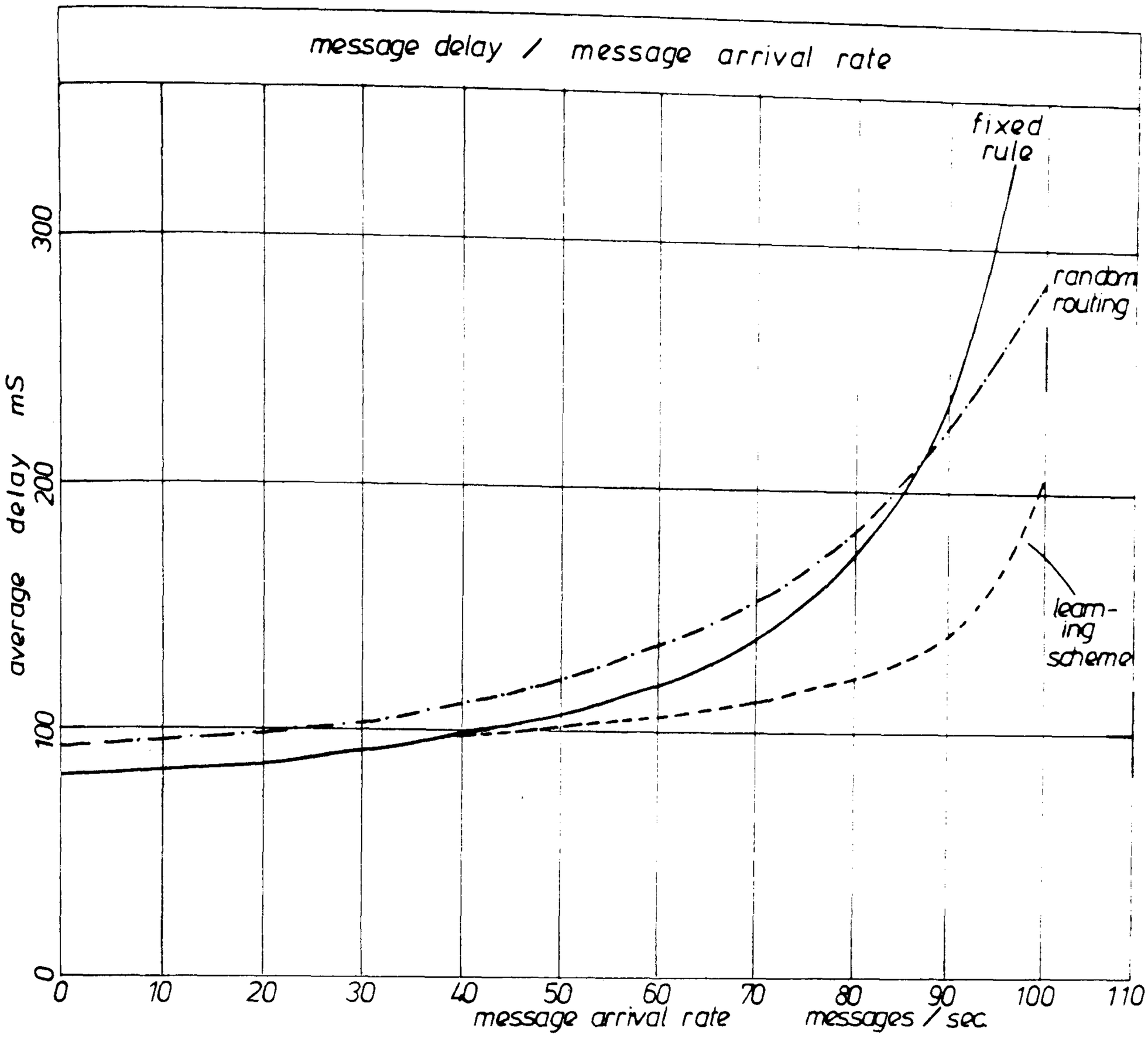


Figure 9·8a Average message delay characteristics for datagram network

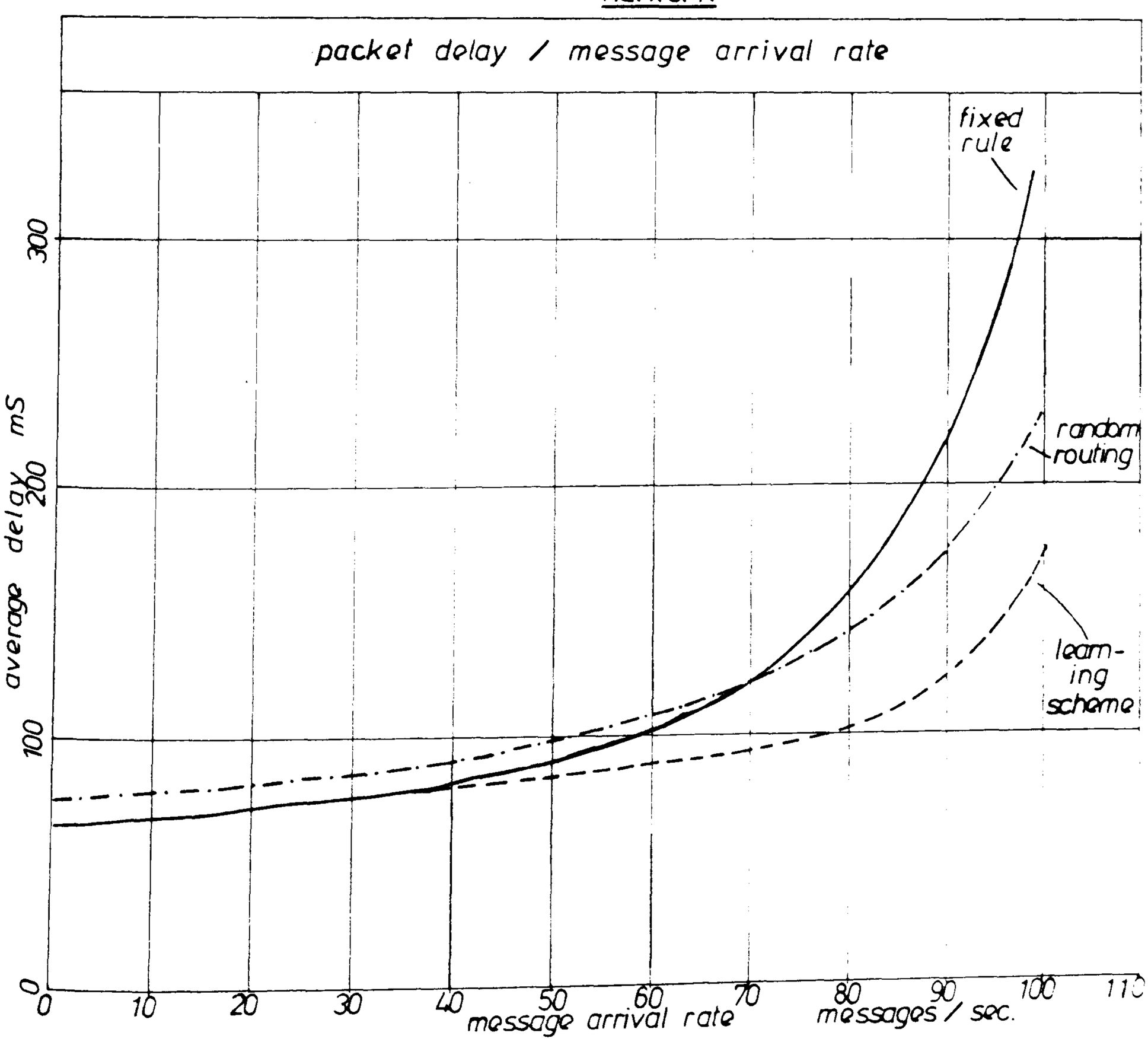


Figure 9·8b Average packet delay characteristics for datagram network

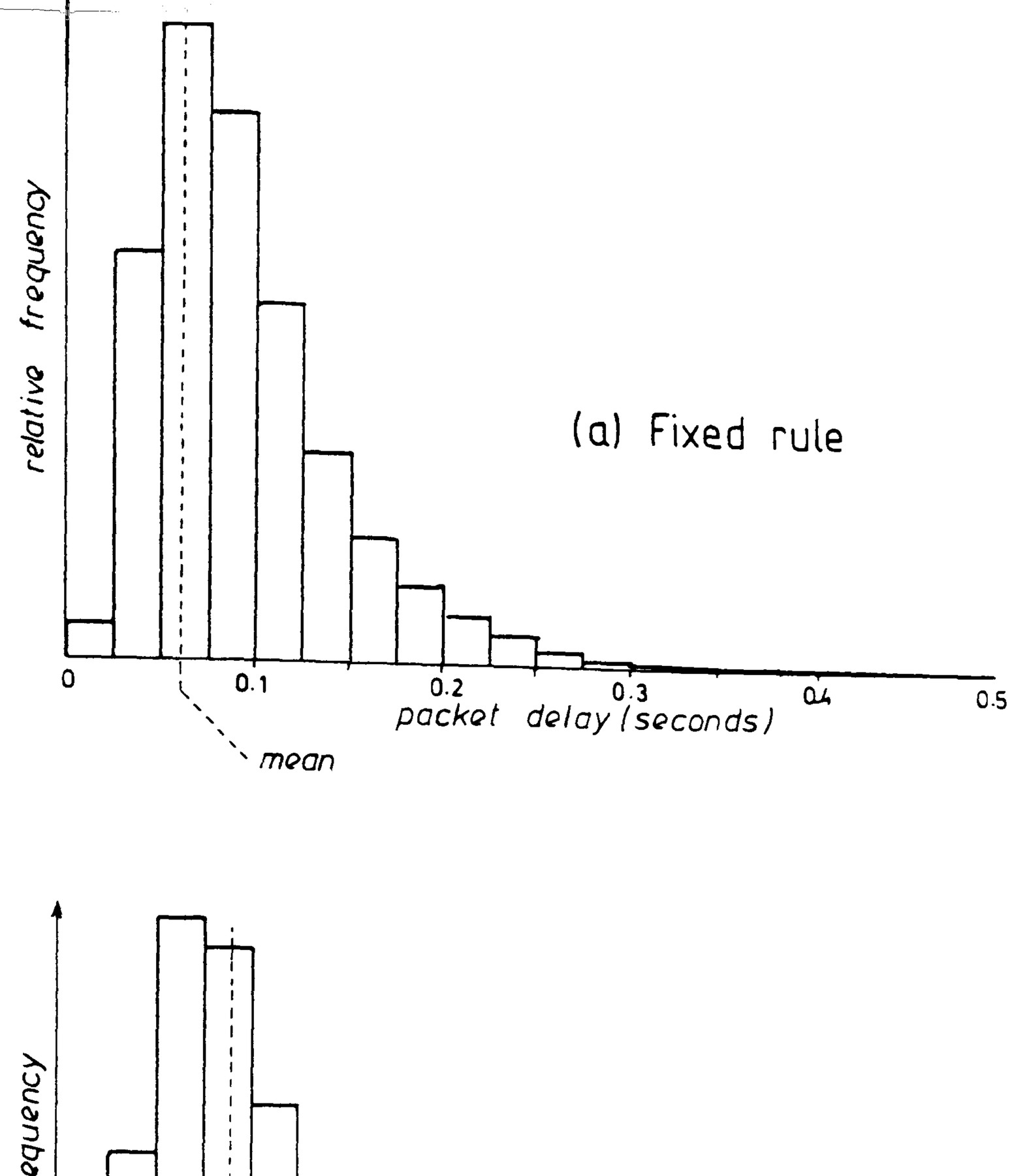
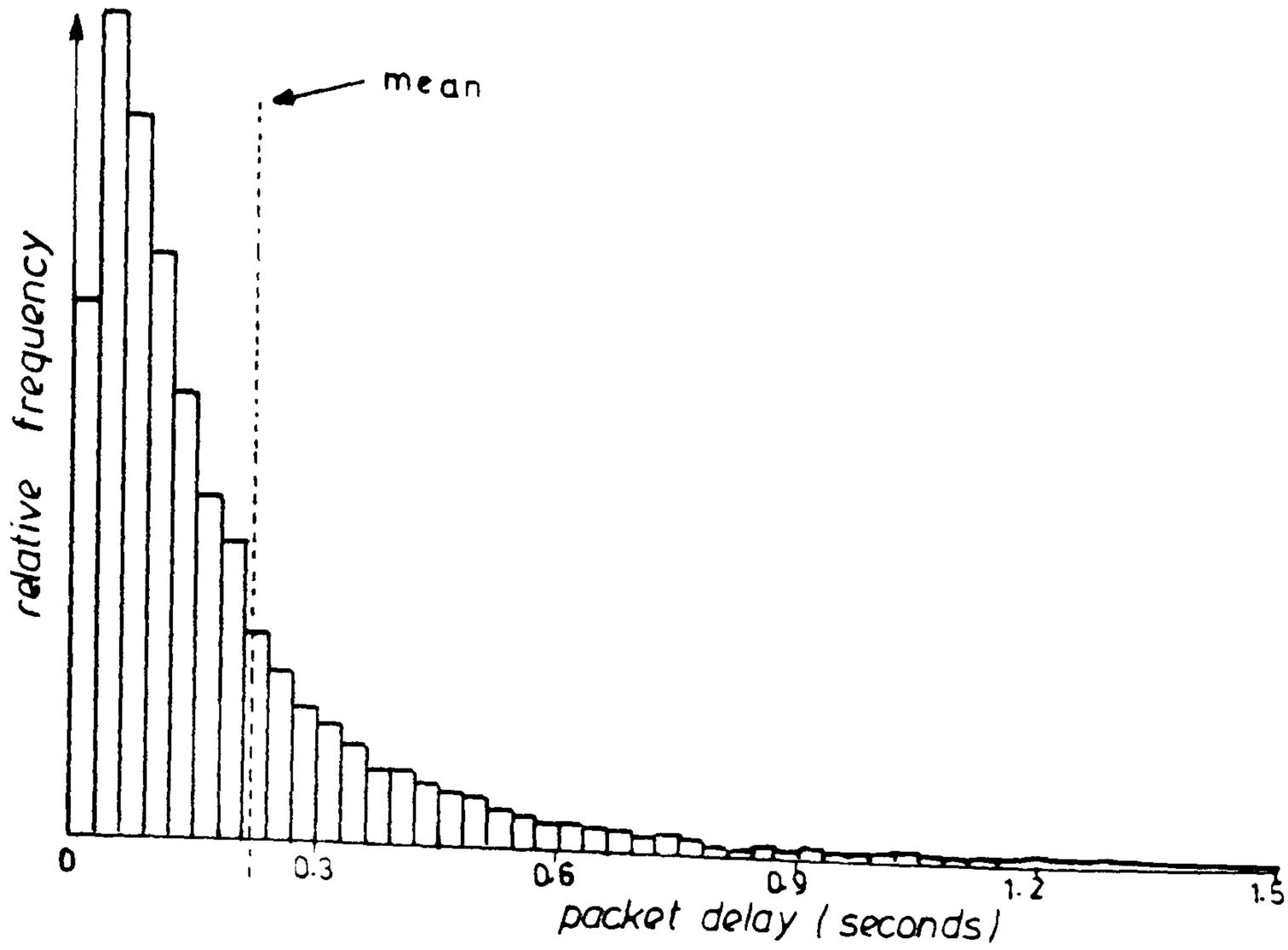
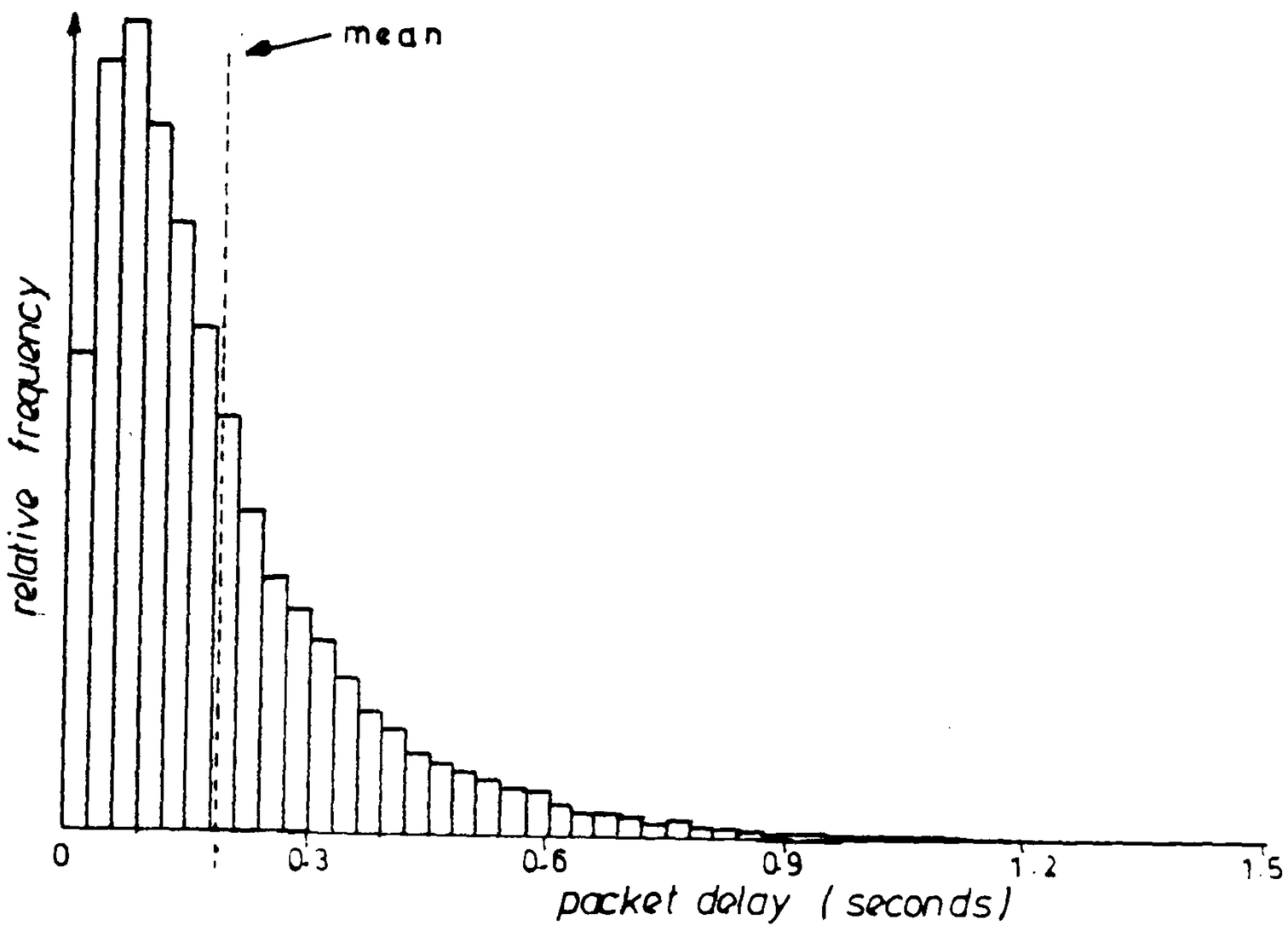


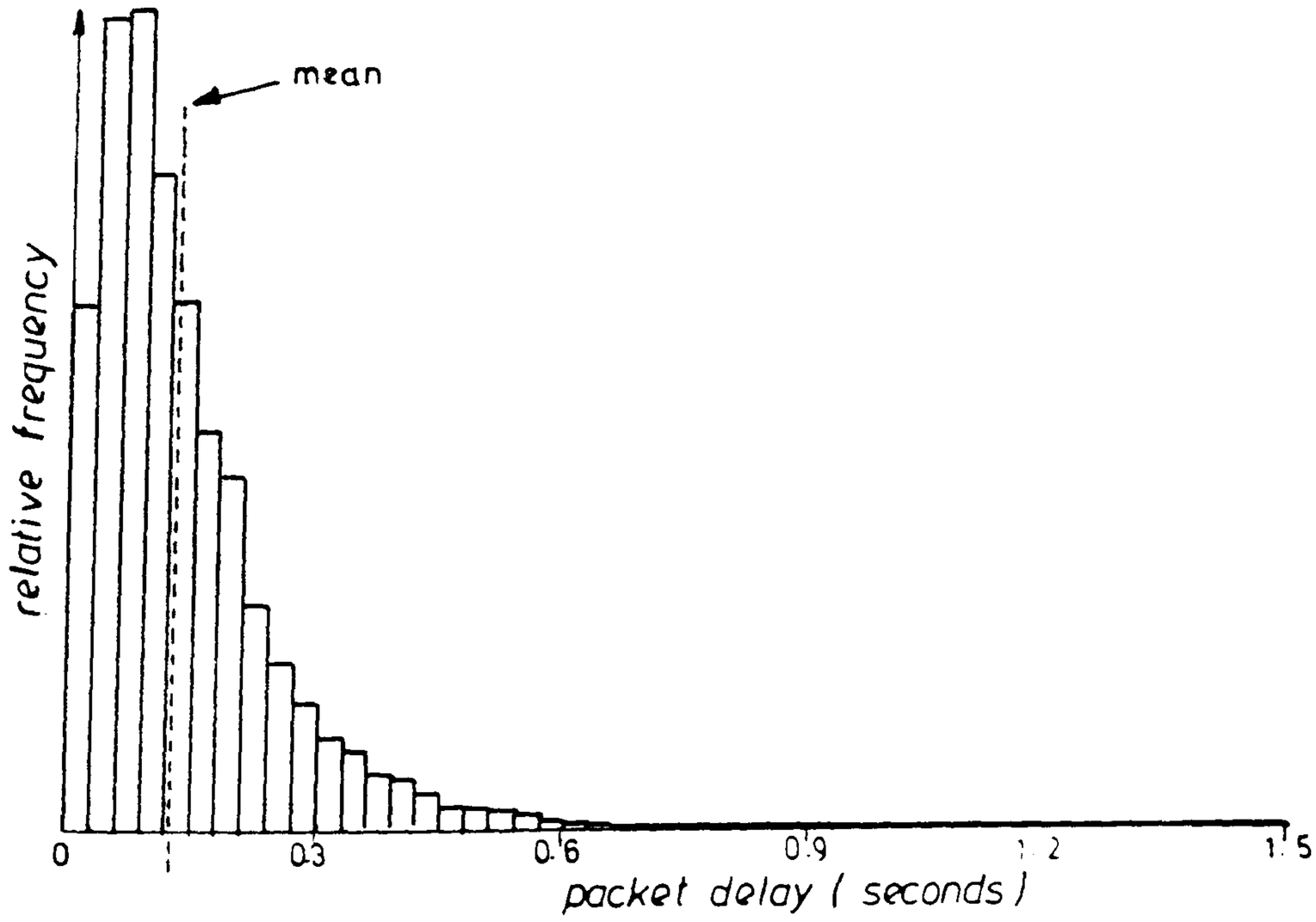
Figure 9.9 Packet delay distribution λ = 10 messages/sec.



(a) Fixed rule



(b) Random routing



(c) Learning automata routing

Figure 9.10 Packet delay distribution λ = 90 messages/sec.

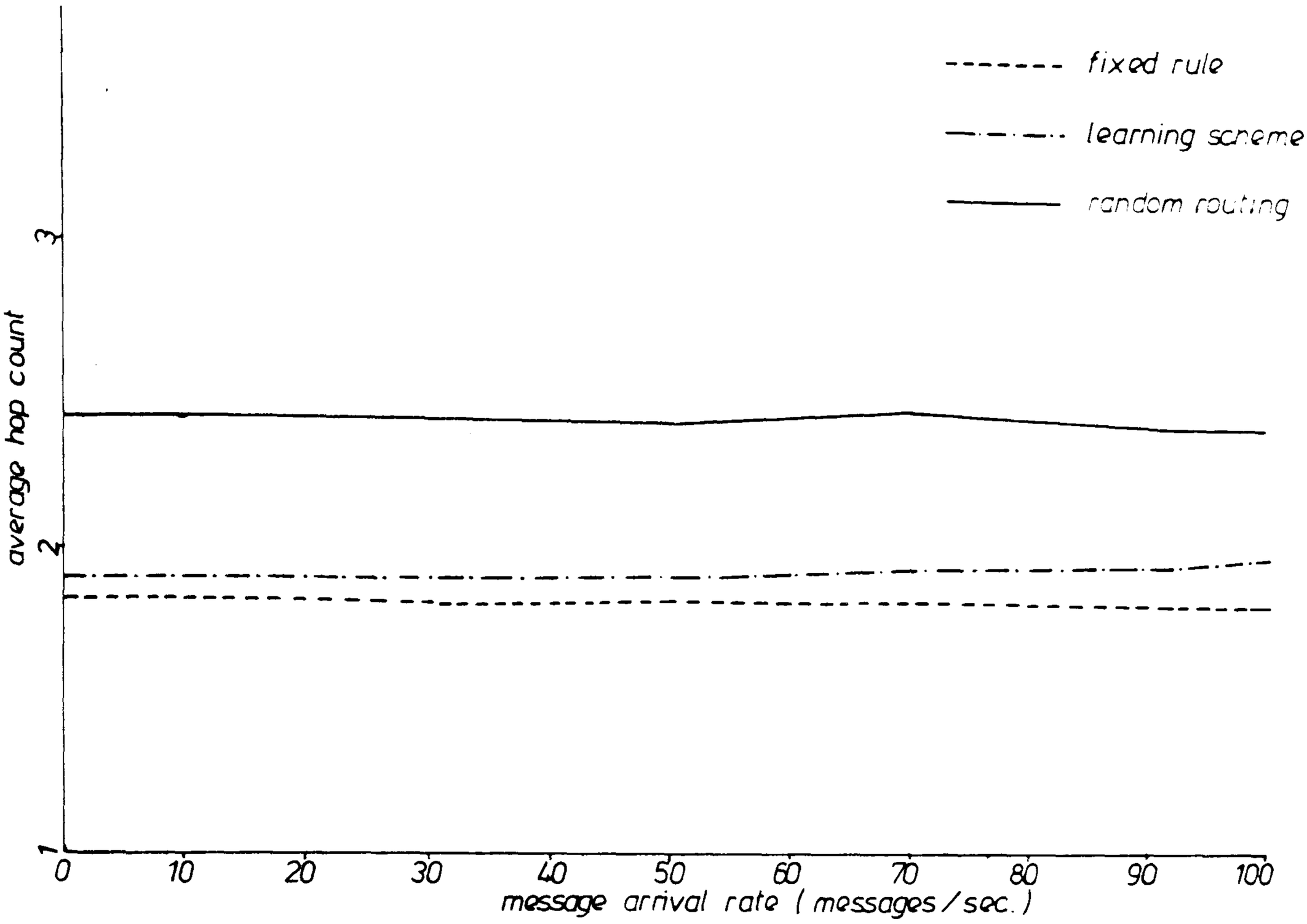


Figure 9.11a Average hop count

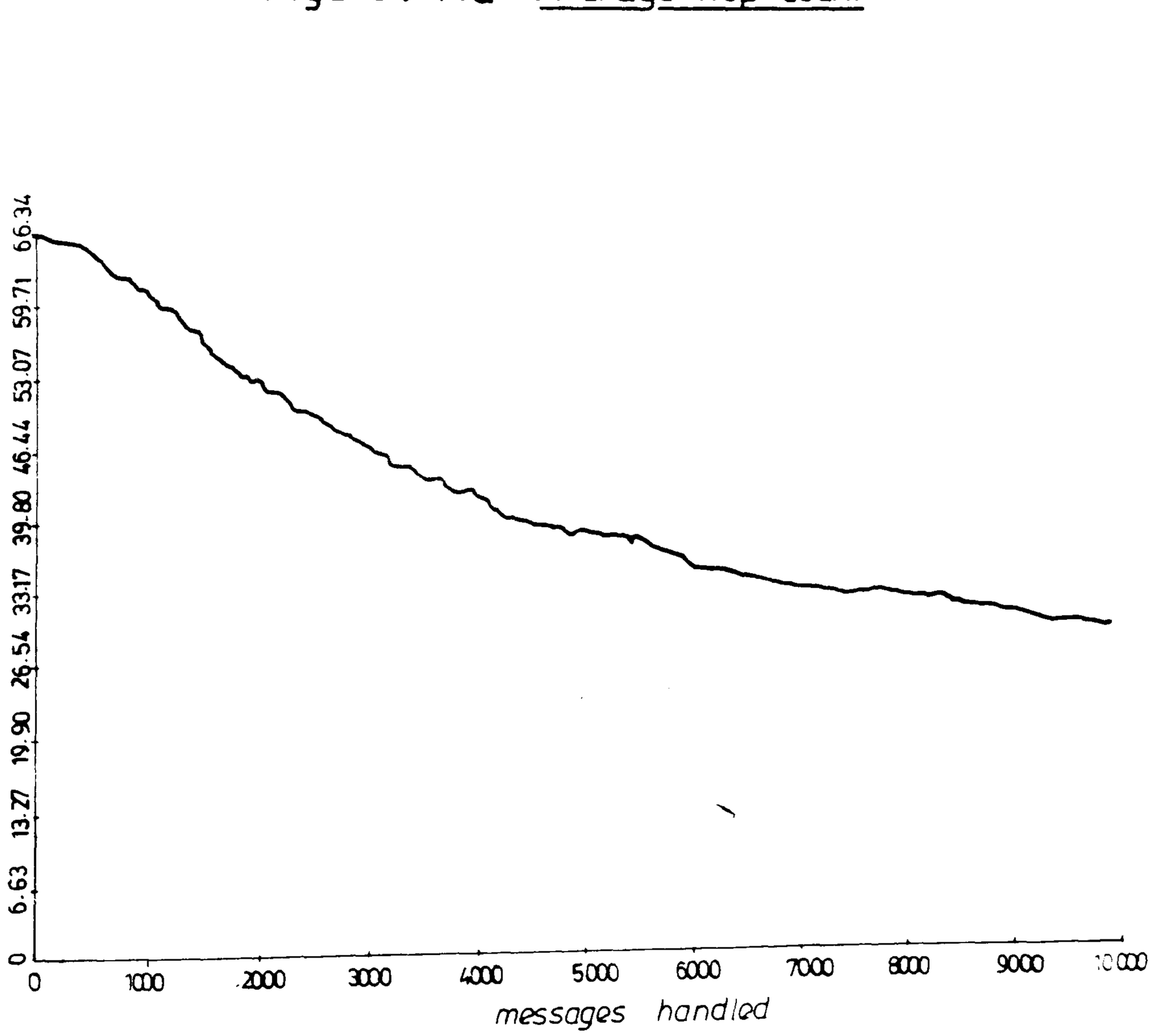
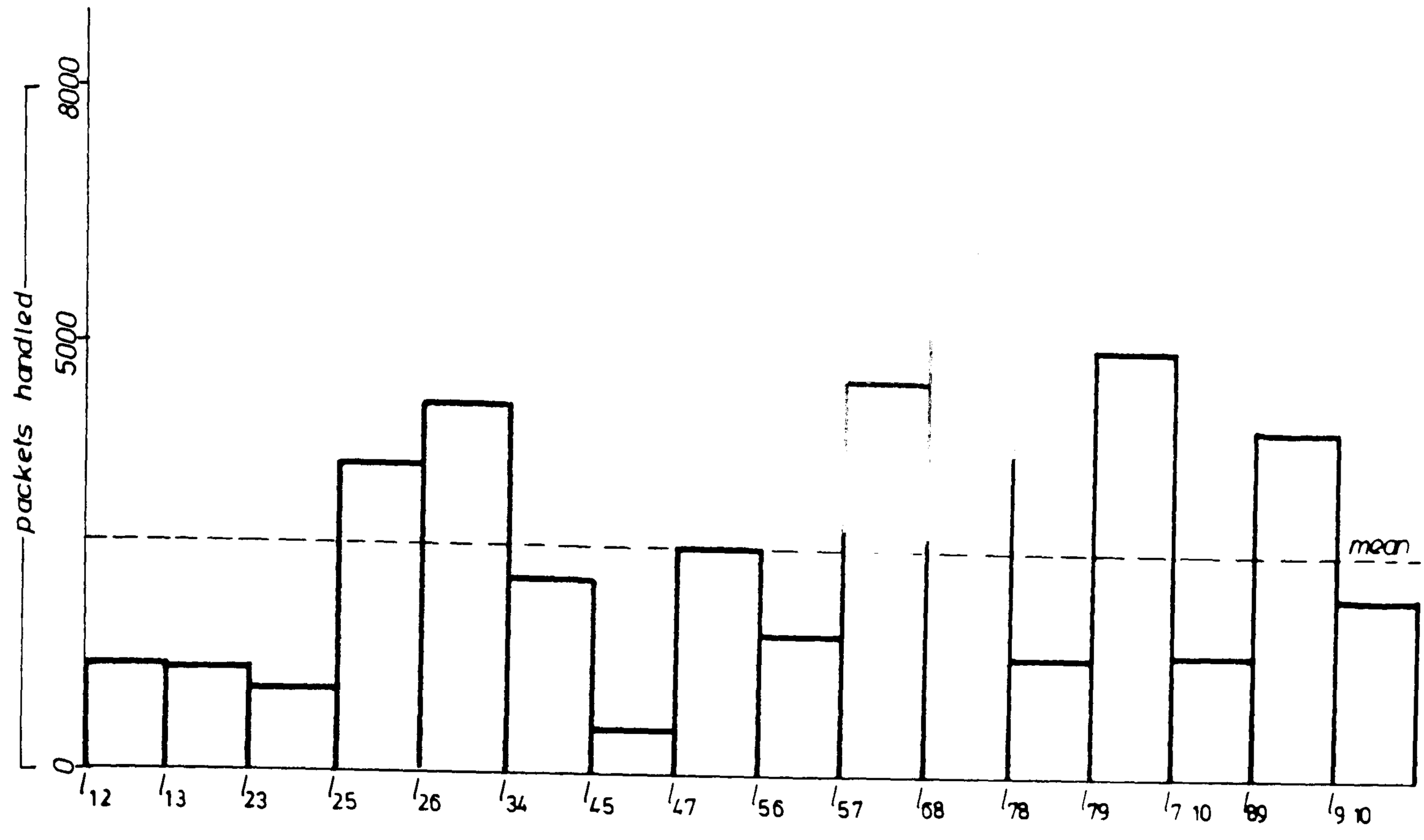
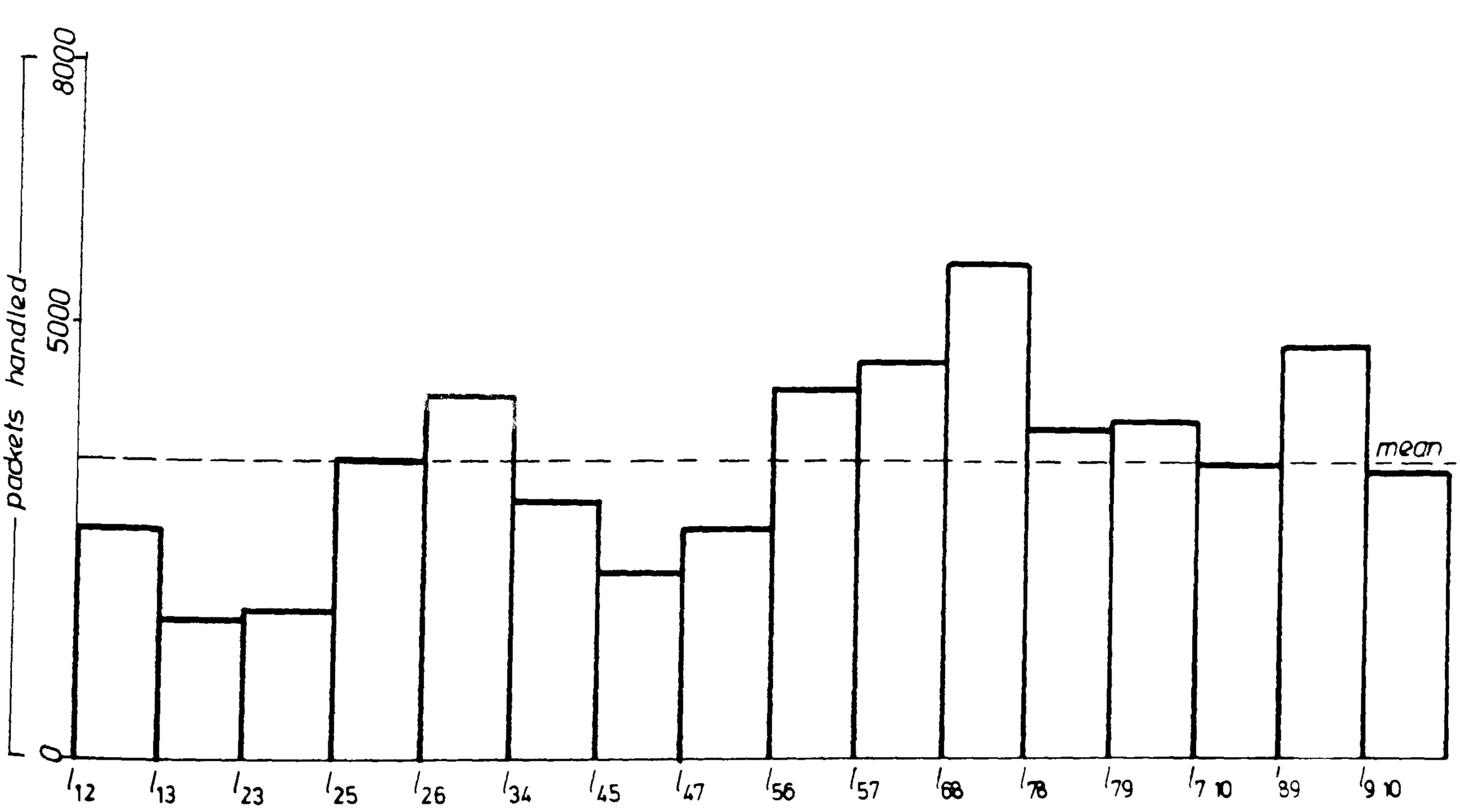


Figure 9.11b Routing scheme entropy (λ = 50 messages/sec.)

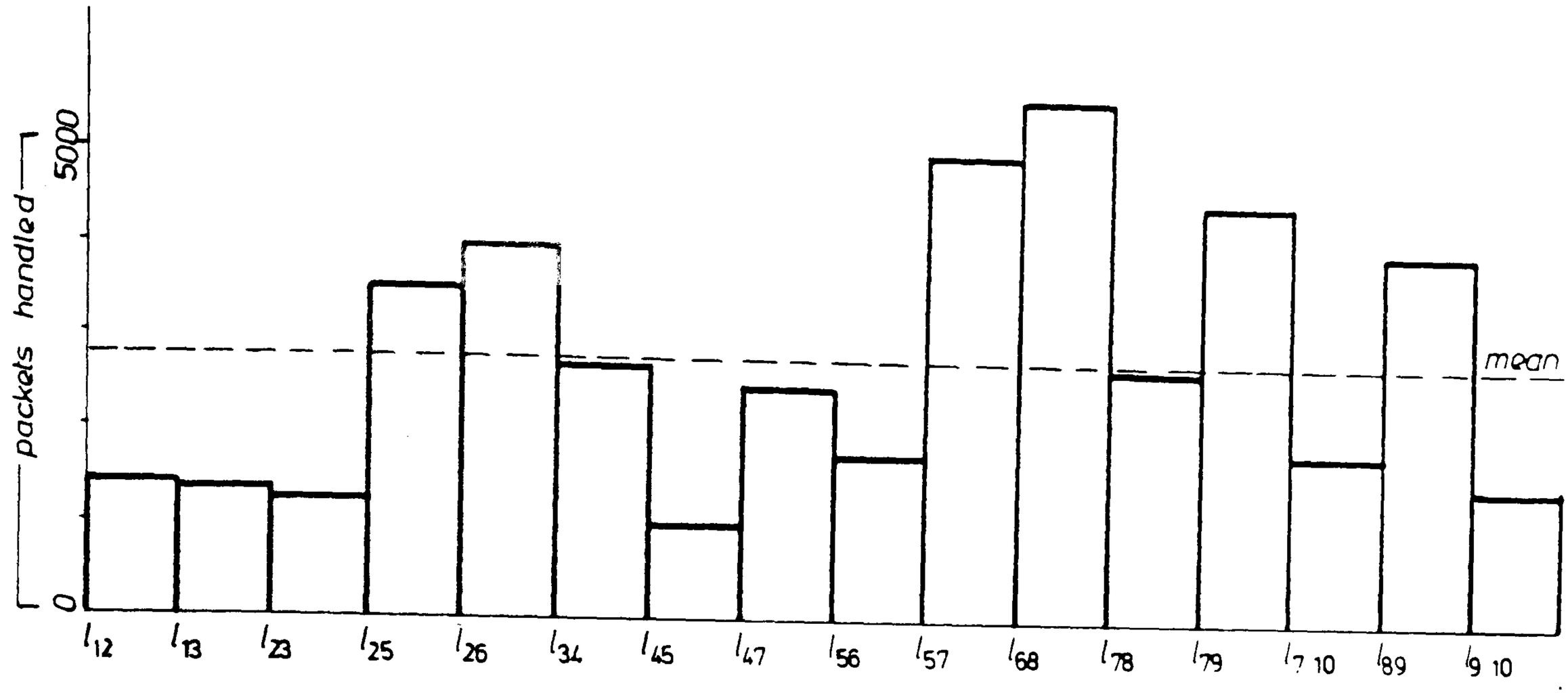


(a) Fixed rule 50 messages/sec

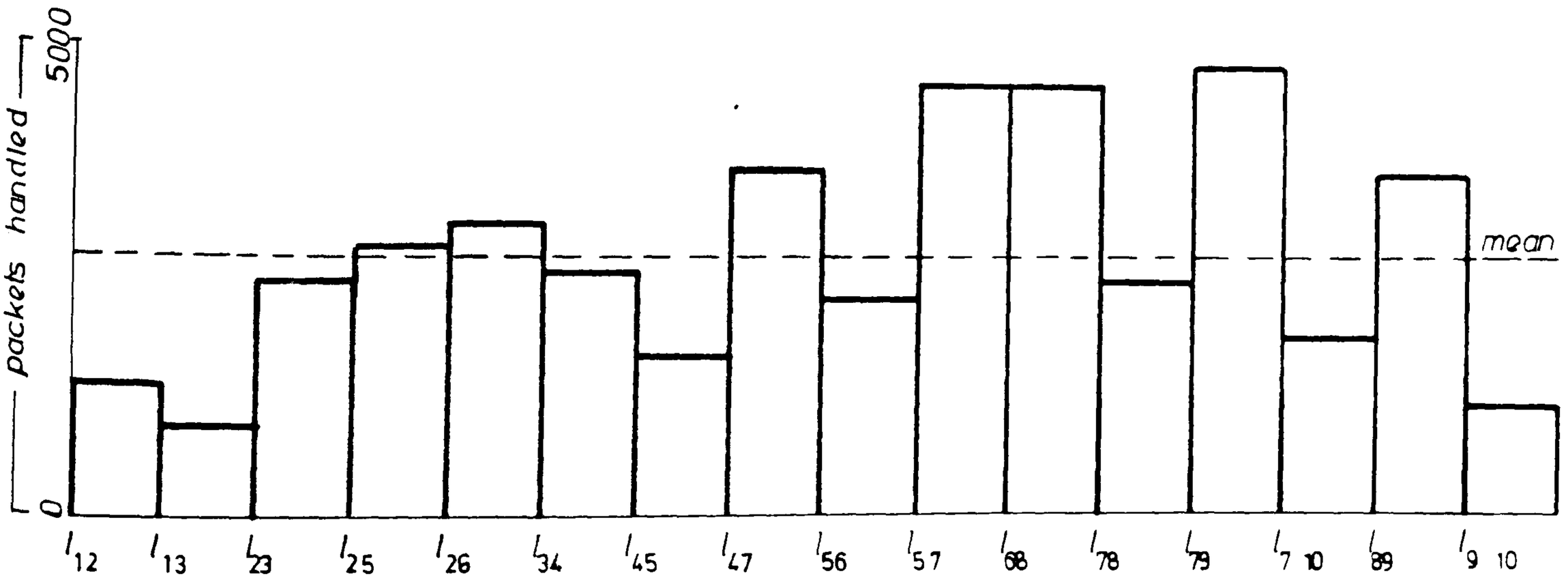


(b) Random routing 50 messages/sec.

Figure 9·12 Link utilisation



(a) Learning automata 50 messages/sec.



(b) Learning automata 90 messages/sec.

Figure 9.13 Link utilisation

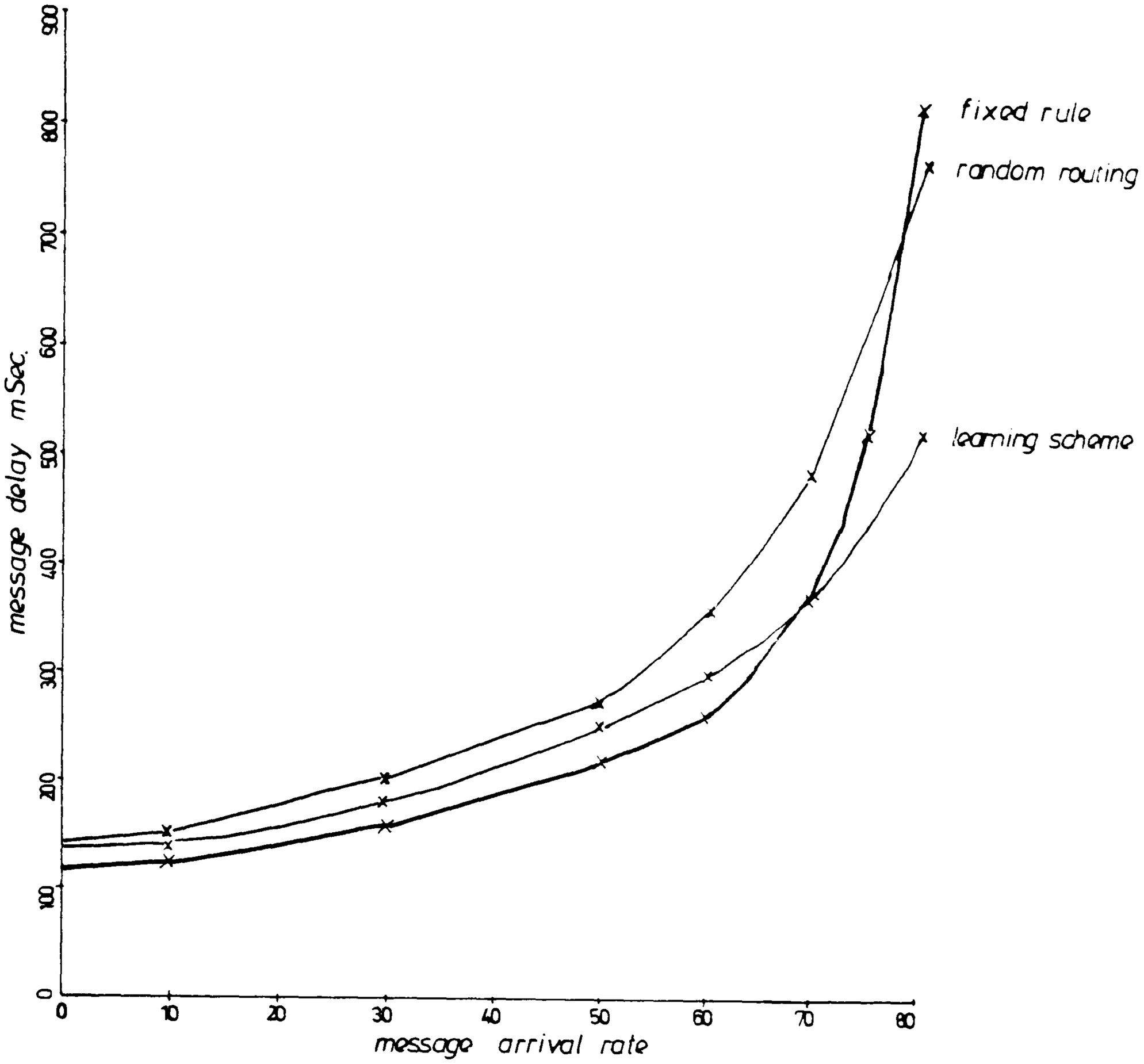


Figure 9·14a Virtual call message delay

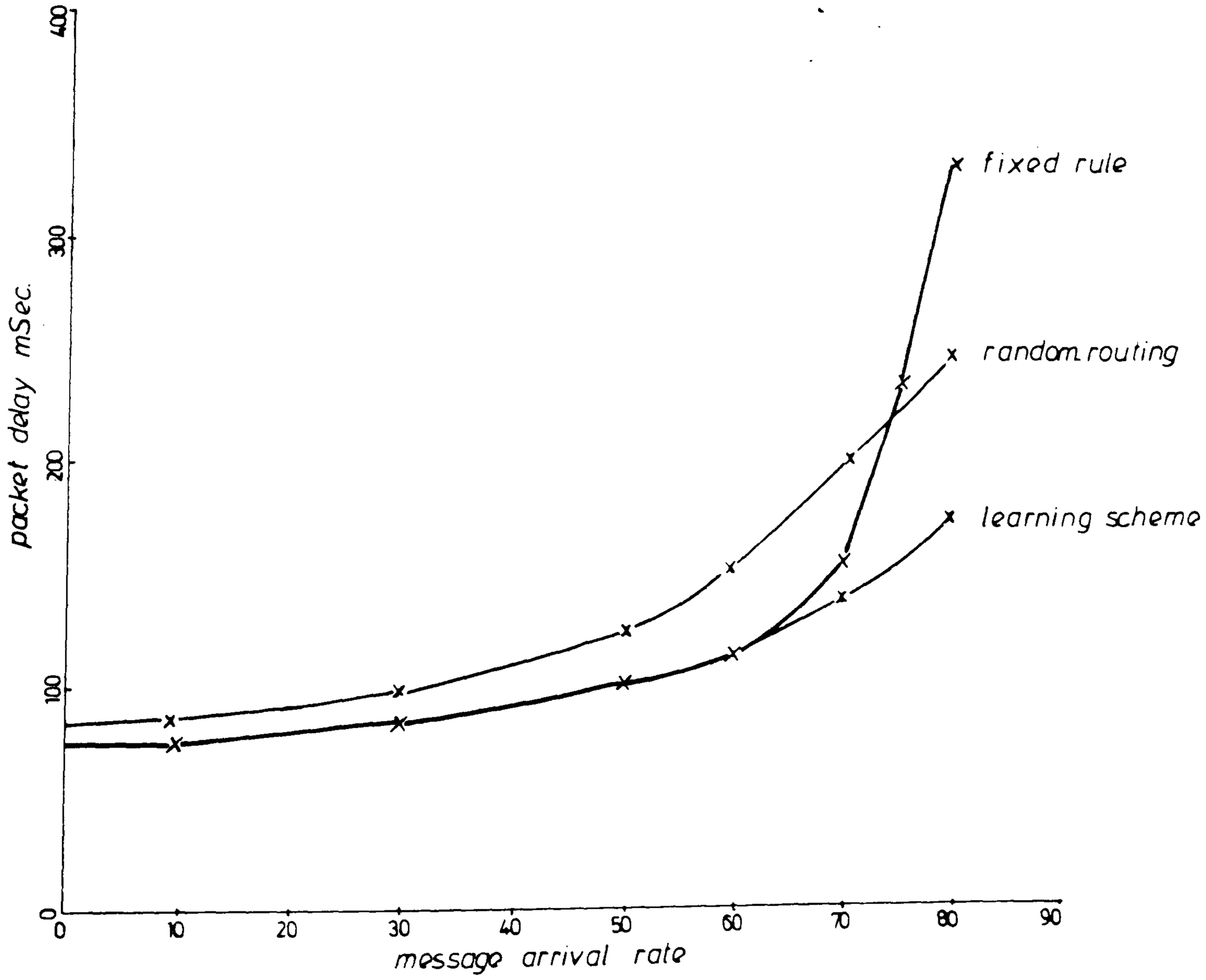


Figure 9·14b Virtual call packet delay

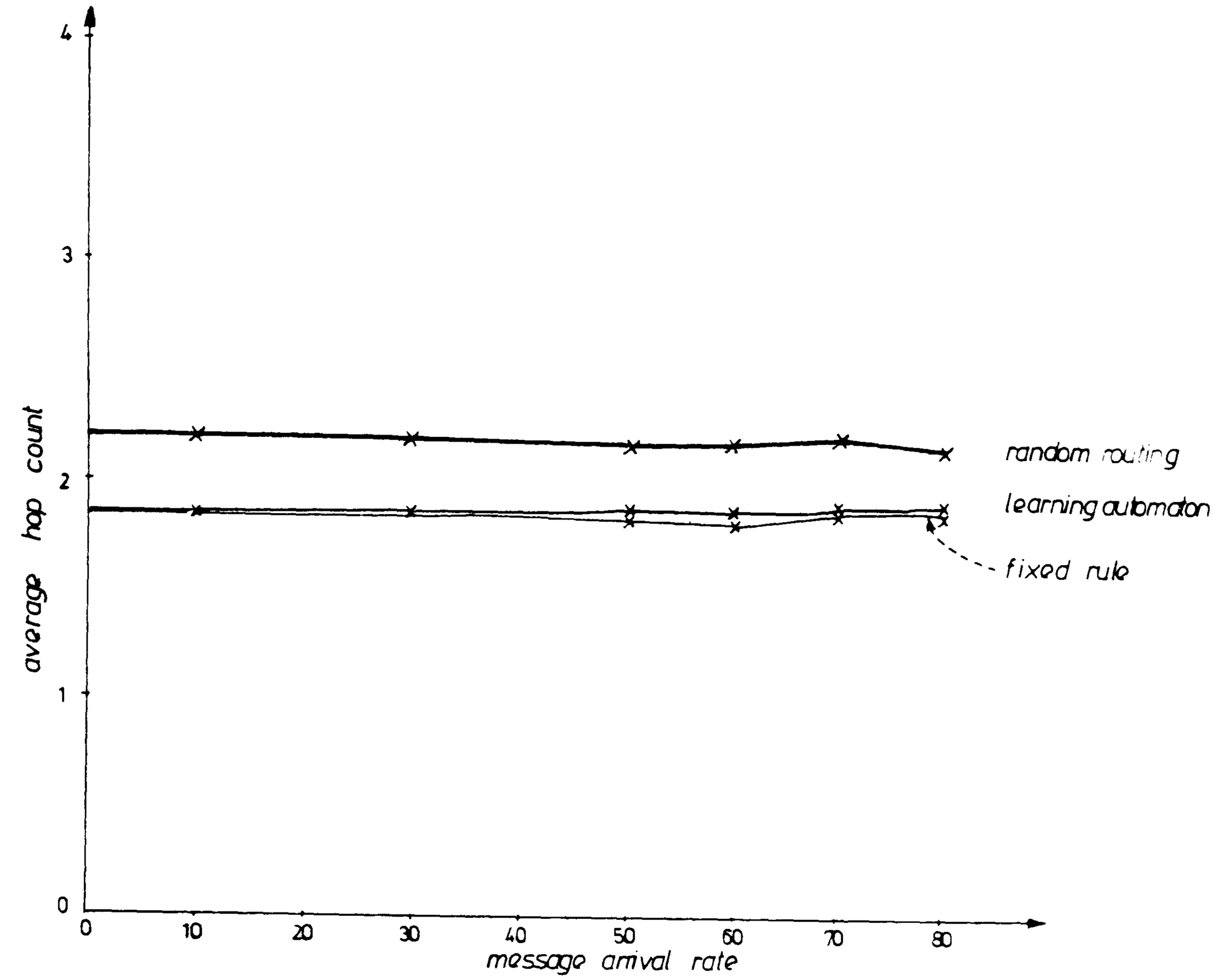


Figure 9.15a Average hop count (virtual call network)

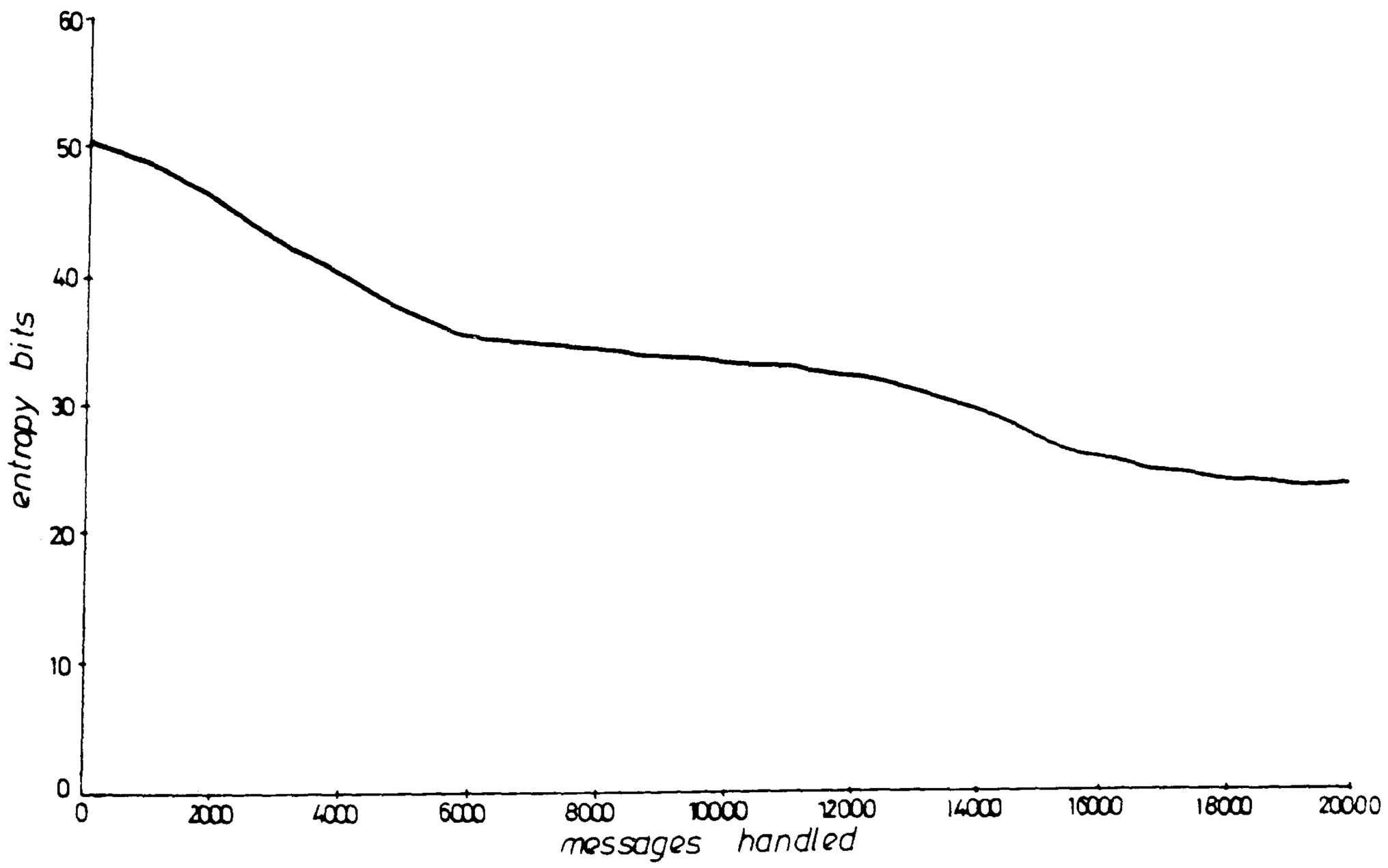


Figure 9.15b Routing scheme entropy λ = 50 messages/sec.
(virtual call network)

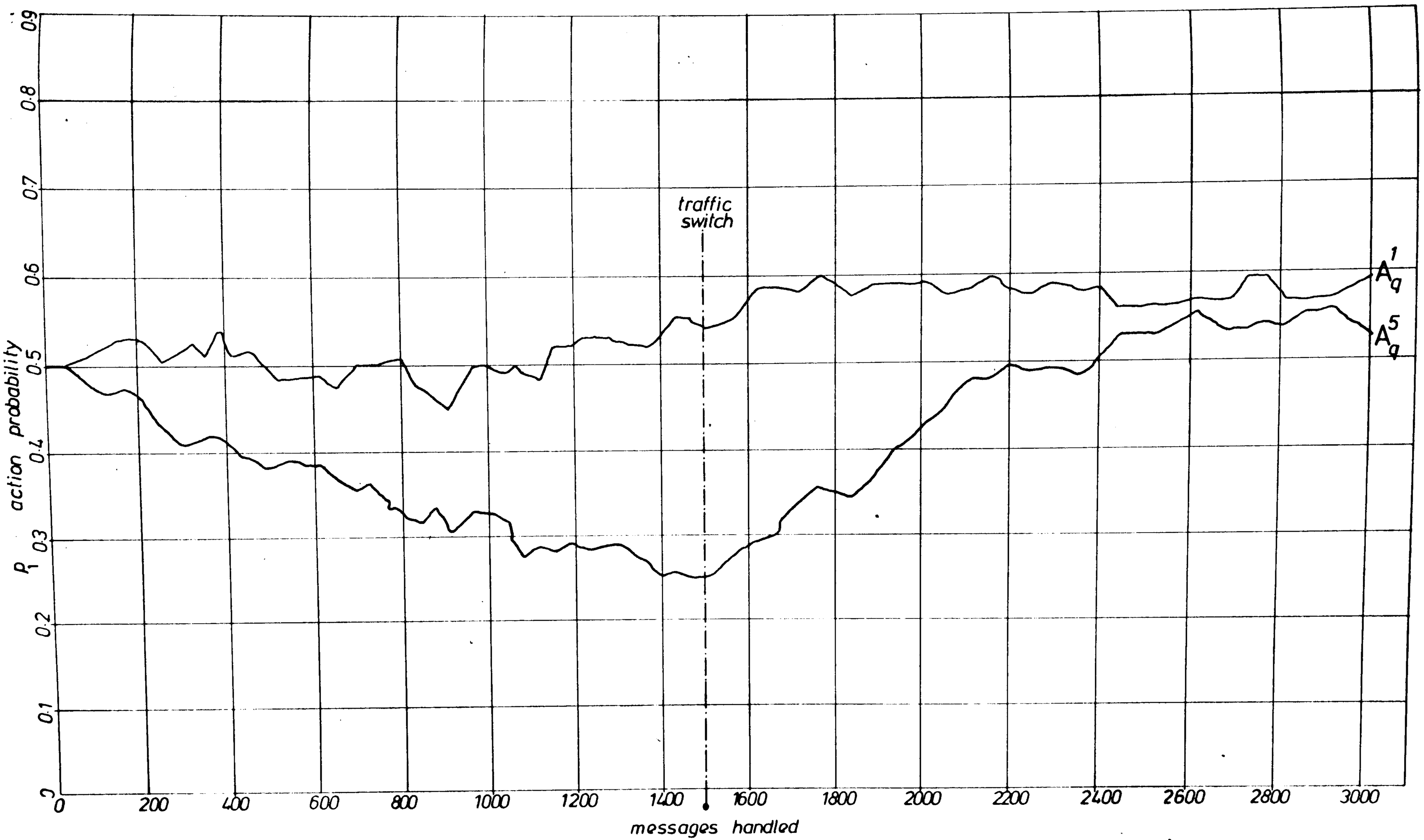
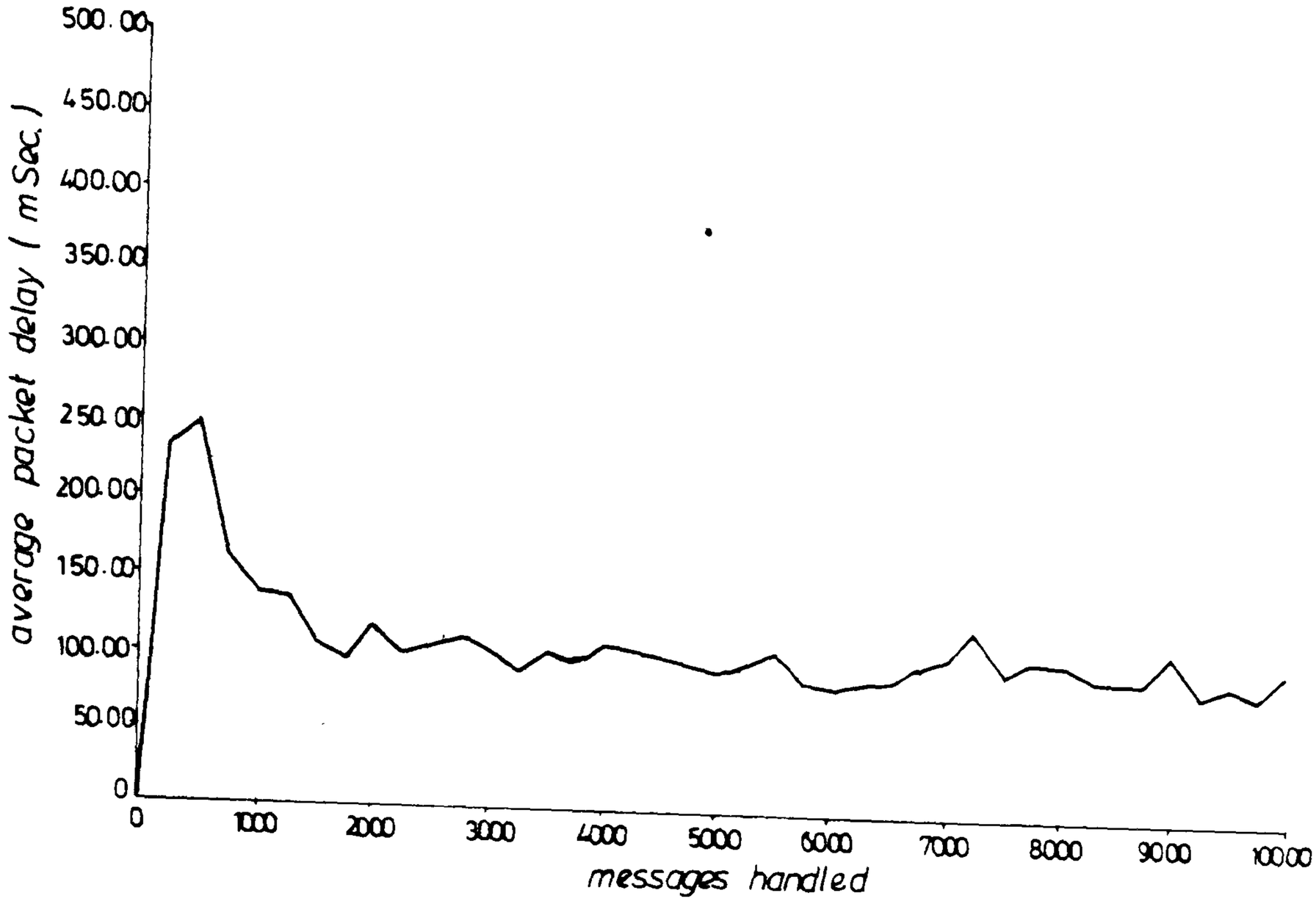
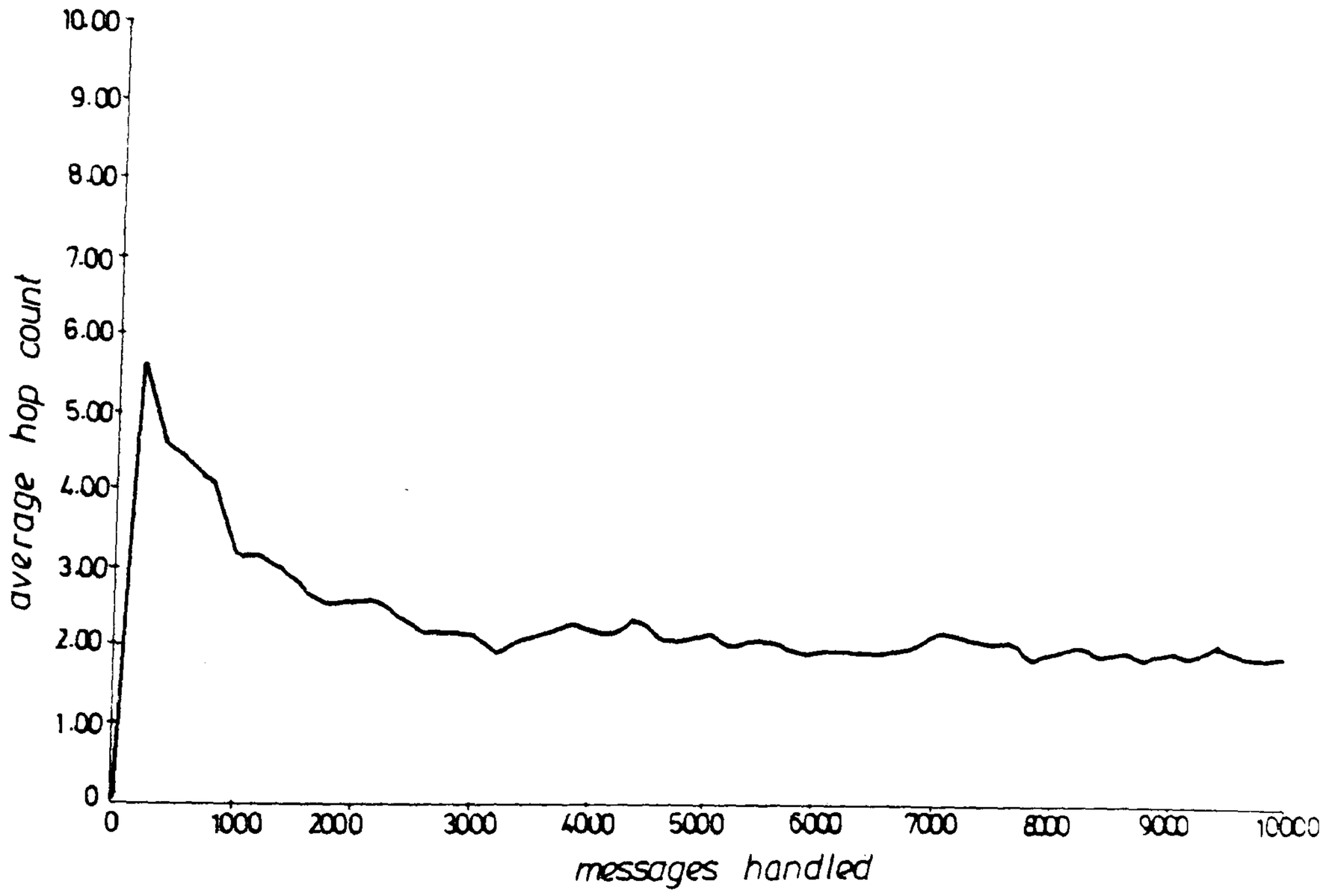


Figure 9.16 Learning curves for A_g^1 and A_g^5 under non-stationary traffic

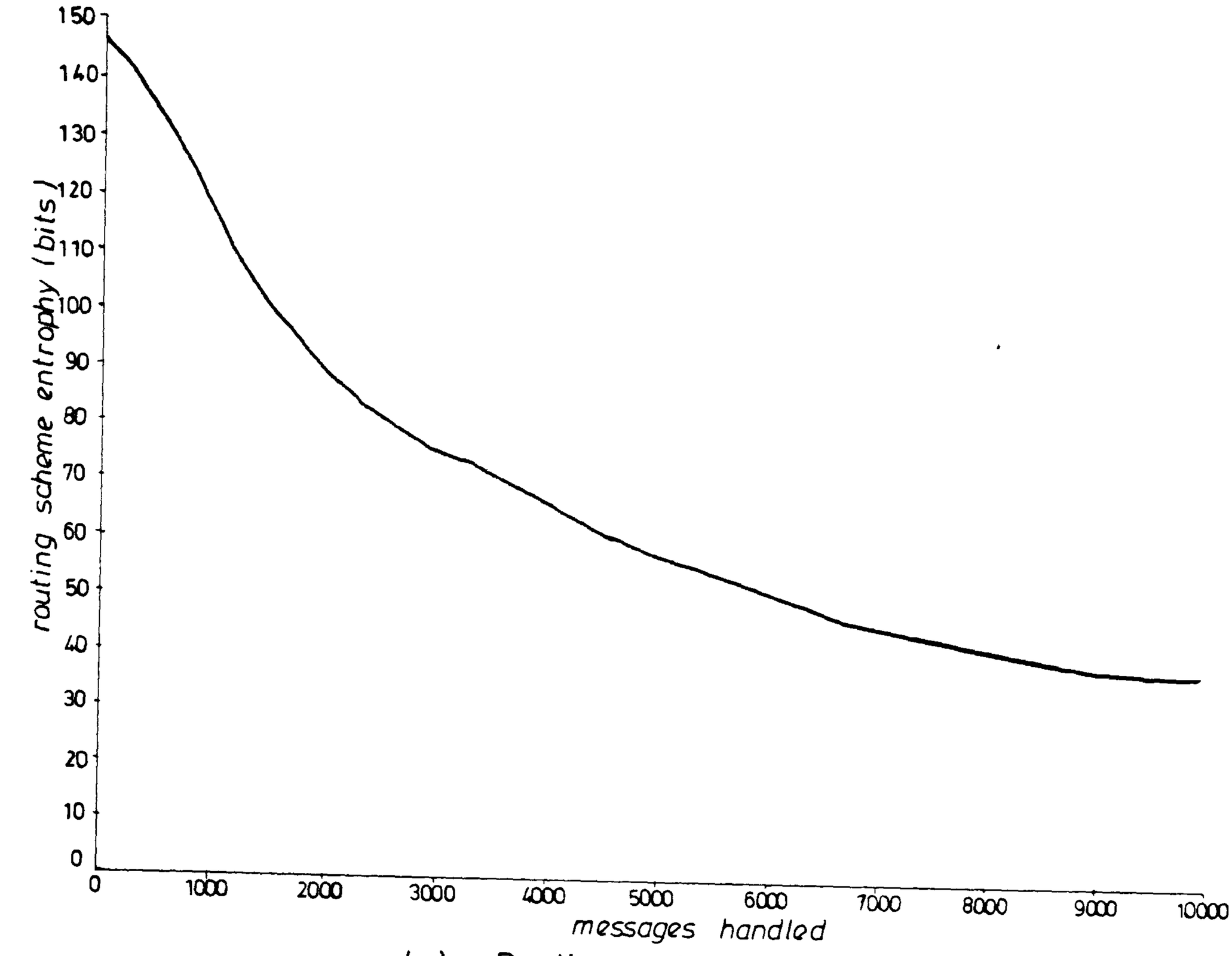


(a) Delay

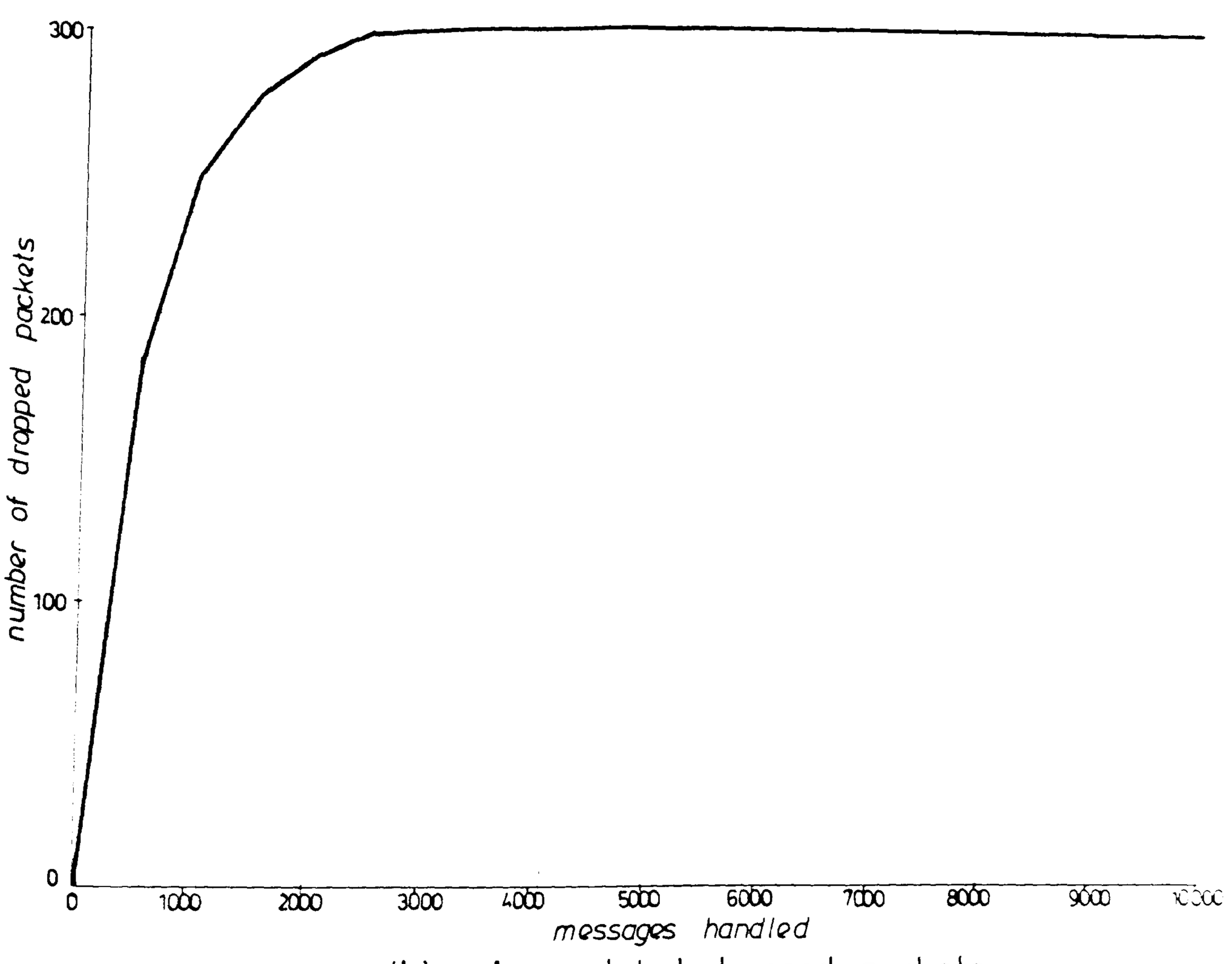


(b) Average hop count

Figure 9.17 Transient performance of self-organising scheme



(a) Routing scheme entropy



(b) Accumulated dropped packets

Figure 9.18 Transient performance of self-organising scheme

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

10.1 Project Review

This project has endeavoured to advance the practical application of stochastic learning automata by a detailed study of adaptive routing schemes for communications networks, based on learning concepts. In the past, a wide body of knowledge has been gathered on the theoretical aspects of learning automata, principally regarding issues such as reinforcement algorithm synthesis and convergence characteristics. Only recently has significant advances been made in applying these principles in areas such as adaptive control, parameter identification, dynamic allocation activities and adaptive routing. Building on established theory, these applications show promise as practical solutions to many complex problems and in addition provides incentive for further exploration of learning techniques.

A recent development in learning automata theory has introduced the notion of the non-autonomous environment, which as suggested, occurs in many practical situations. This has greatly contributed to the development of the learning routing schemes, and will in the future, provide an invaluable framework for further application studies. Particularly relevent to this study has been the interaction between the non-autonomous environment and the variable structure automaton, operating with the L_{R-I} and L_{R-P} reinforcement algorithms. In conjunction with this, attempts have been made to abstract simple non-autonomous environments with a view to extending the understanding of the basic routing operation.

Operating on the principle that it is better to understand in /

in reasonable detail what happens in small networks, than vaguely the routing operation in larger examples, simple automata/network combinations have been utilised for initial study. From these elementary circuit and message switched configurations, the routing behaviour of the learning automaton has been investigated and the influence of the alternative reinforcement schemes noted. Also, using derived conditions for optimum routing, the L_{R-I} and L_{R-P} algorithms have been shown to be marginally sub-optimal in both network types, although as later illustrated by simulation studies of the small networks, the steady state results are very close to the minimum achievable delay and blocking probability.

An important aspect of this present study has been the application and demonstration of the learning routing technique in realistic network environments, using as an experimental vehicle, the detailed simulation packages developed for the project. In the circuit switched application, testbed telephone networks, including hierarchical and mesh structures, have been modelled. These experiments have been instrumental in demonstrating the learning scheme as a series alternative to the conventional fixed rules, and have illustrated the significant advantages to be gained, particularly under adverse conditions caused by failures and traffic overheads. Moreover, the simulation exercise has been extended to include a network specified by Bell Laboratories, which has provided a complete circuit switched environment to test the learning automata in a self-organising role.

Equally important has been the extension of the simple message switched routing implementations to full scale packet switched networks, with routing schemes designed to /

to cover both the virtual and datagram operating protocols. In the proposed routing schemes the basic learning concepts have been applied in a manner which is practical and efficient in terms of control overheads. Again extensive simulation experiments have confirmed the performance of the new approach in relation to commonly implemented routing techniques, using realistic network and traffic models.

In conclusion, this study has attempted to analyse and implement learning automata routing applications on a practical basis, particularly by the use of detailed simulation models of large networks. Significantly, the operation of learning automata routing is now better understood, from the convergence conditions of the individual routing automata to the behaviour of full scale implementations in circuit and packet switched networks.

10.2 Conclusions

The power and potential of learning automata routing has been clearly demonstrated by the analysis and simulation studies presented in this thesis. In order that the important points may be summarised, this section gathers conclusions from the complete study, and classifies these under the main headings of operation, implementation and performance.

Operation

The operation of the learning automata routing schemes, in both circuit and packet switched networks, has revealed interesting theoretical behaviour which has significant practical implications. Furthermore, the operation of these schemes present a fresh approach to dynamic routing, which as shown /

shown, has distinct advantages over existing techniques.

Circuit Switched Networks

In comparison to the commonly implemented alternate path technique, the learning routing approach justifiably represents a marked improvement in routing service and network reliability. Under normal operating conditions, the optimal alternate path and learning automata schemes have been shown to produce a comparable performance, both tending to the minimum achievable blocking probability. However, operations outside the engineered conditions have highlighted the need for an adaptive routing policy, the isolated strategy of the alternate path scheme proving unable to exploit unused network capacity. The learning scheme on the otherhand, by virtue of the simple feedback arrangement, has demonstrated the ability to alter the routing strategy to provide a more efficient allocation of resources and consequently results in a superior performance.

A major contribution of this present study has been to relate the nature of various reinforcement algorithms to specific routing strategies. In particular, the L_{R-I} algorithm has been shown to equalise the blocking probabilities associated with the allowable routing options at a node, while for the same situation the L_{R-P} scheme equalised the blocking rates. Equally significant has been the ergodic nature of the L_{R-P} algorithm, which irrespective of any initial conditions, consistently converged to a unique statistical distribution.

Packet Switched Networks

The learning automata routing scheme for packet switched networks has been classified as a distributed load splitting technique, with the ability to adaptively proportion traffic /

traffic over several paths, in such a manner as to provide a close to optimal performance. Undoubtedly this technique has significant advantages over other load splitting schemes, such as Gallagers[52], in that it entails a far more practical form of feedback and updating operation.

Again the influence of the various reinforcement algorithms has been shown to result in alternative routing strategies, the L_{R-I} leading to an equalisation of the average delays on the outgoing paths at a node. The operation of the L_{R-P} on the otherhand results in a balancing of the accummulated delays on the various options, with the effect of levelling the link utilisation. Furthermore, although neither of the schemes produce a truly optimum service, conditions have been established which show that such a performance requires the equalisation of the marginal and average delays. The logical progression in this area is therefore the design of a suitable reinforcement scheme on the basis of these conditions.

Implementation

The implementation aspects of routing schemes cannot be over stressed, since for reliable and efficient service any installation must follow a practical approach. In this context, the learning automata schemes have significant advantages, particularly due to the sensible feedback and updating strategies which are founded on existing control mechanisms and protocols. In more detail, the important points involved with implementation includes:

Simplicity: An outstanding feature of the proposed technique lies in the simplicity of routing and updating operations, being completely viable in modern circuit and packet switched networks.

Feedback Strategy: The simplicity of the learning routing schemes /

schemes are a direct consequence of the feedback strategies employed to respond to an automaton routing action. The circuit switching application shows a particularly elegant solution, the P-model type response (Penalty/reward) simply produced by the normal control signals responsible for establishing circuits. An equally straightforward approach has been exercised in the store and forward networks, with the existing network control traffic used to transport the network delay to the relevant automata. After appropriate pre-processing, an S-model response can be produced, although if required, the P and Q-models are possible alternatives.

Reinforcement Algorithms: The nature of the reinforcement algorithms also contribute to the overall scheme practicality, these in general based on arithmetically simple and robust equations. Moreover, the ability to control the characteristics of the algorithms through simple learning coefficients, permits a trade-off between the convergence rate and the steady state accuracy.

Self-tuning/Self-Organising Routing: The option of self-tuning or self-organising routing schemes provide additional flexibility to suit the requirements of a particular network application. In circumstances where the overall topology is known, and is fairly stationary, the self-tuning policy is advantageous, the automata operating to proportion traffic on pre-specified paths which are cyclic. Alternatively, when the network is globally unknown, with perhaps nodes and links frequently going down, coming up and moving position, then the self-organising scheme is an ideal choice. Under these conditions the routing scheme has complete freedom to seek out any possible paths, although in some attempts local loops may be formed.

Performance

Performance

The performance of the learning schemes has received continual attention throughout this study. In general, the important performance issues, particularly from the standpoint of real applications, concerns the optimality and convergence characteristics.

Optimality: In the circuit and packet switched networks, the equilibrium conditions of the L_{R-I} and L_{R-P} reinforcement algorithms result in sub-optimal routing. However, from the experiments carried out, both appear to produce a performance which is very close to the optimal. Despite the present inability to apply a truly optimal routing strategy, the future provision of a suitable reinforcement algorithm should permit routing schemes which converge to give the minimum achievable delay and blocking probability.

Convergence: The topic of convergence embodies several issues, including the convergence rate, steady state accuracy and stability. Theoretically, these points are still new ground, and will in the future require substantial investigation to establish conditions and criteria for satisfactory designs. From an experimental position, the simulations carried out in this present study indicate the possible convergence rates to be clearly adequate for the application. Furthermore, the stability and accuracy aspects also appear to be sufficient, although further experiments are required to conclusively establish that the proposed schemes are intrinsically stable under general conditions.

10.3 Future Work

As the current project has progressed, questions have been raised which are as yet unanswered and in addition, new study areas proposed. This future work can be placed in four /

four main areas; learning automata studies, circuit switched automata routing, packet switched automata routing and network simulation.

Learning Automata Studies

In order to support further application studies, several theoretical aspects of learning automata principles are specified for extended investigation.

1 The subject of convergence stability in multi-automata configurations is of paramount importance in any network or distributed control application. Future theoretical studies in this area are required to look into possible stability criteria, implications of instability and methods of enhancing stable convergence.

2 A study of additional reinforcement algorithms operating on non-autonomous environments, aimed at presenting alternative convergence conditions for the routing problem and other applications projects.

Circuit Switched Automata Routing

Additional study of the telephone network automata routing are proposed for further study.

1 A more detailed and generalised analysis of the optimal routing solution in a circuit switched network, directed at producing the necessary equilibrium behaviour for the automata.

2 In conjunction with point 1, an attempt to formulate an optimal reinforcement algorithm, based on the derived conditions for optimality. Ideally any future routing scheme developments will maintain the existing feedback strategy and require any new algorithm to continue operating from a P-model response.

3 Further simulation studies to test and verify new algorithms, /

algorithms, and also to investigate fully the questions raised on stability and accuracy. If possible, additional trials may be undertaken on a pilot network, which if successful would warrant the application of the technique in a full scale network.

Packet Switched Automata Routing

Future developments of the packet switched automata routing scheme are likely to consider the following:

1 The implementation of a complete adaptive routing scheme which would comprise of two components, (a), the existing learning automata scheme to carry out optimising within the constraints of the specified allowable actions, and, (b) a distributed algorithm responsible for allocating allowable actions to each automaton, suitably established to prevent cyclic paths.

2 Design of a hybrid routing scheme, involving the distributed learning automata and a mechanism which operates with the instantaneous local delay, typically based on the output buffer states. This might be organised along similar lines to delta routing[46] with a single variable δ, used to alter the policy of the scheme, by shifting the bias between the automaton action probabilities and the local queue conditions.

$$\varphi_i = \delta p_i + (1 - \delta) S_i$$

With this arrangement the routing variables φ_i, which influence the link selection, are obtained from a combination of the automata action probabilities p_i and the instantaneous state of the output queues S.

3 A continuing study to find, or synthesise, a reinforcement algorithm which produces the required conditions /

conditions for optimum routing. This would follow directly from the conditions produced earlier, which indicate a need to equalise the average and marginal delays.

4 Additional simulations are required to compare the learning approach with alternative distributed routing schemes, such as the ARPANET algorithm and Gallagers load splitting technique. Moreover, simulation experiments need to be continued on the learning schemes, in order that the proposed modifications may be related to the performance of the original implementations.

5 The application of learning automata routing and optimisation techniques in mixed media and local area networks appears to be a promising area. It is also suggested that learning principles may be applied to congestion and flow control schemes, and be used to intelligently regulate traffic in the network.

Network Simulation

The present study has actively involved software simulation techniques for experiments on circuit and packet switched networks, which in general has proved adequate, albeit slow for the complexity of network model under investigation. It is suggested that an extremely practical alternative to this, would be to construct a hardware simulator which could in effect be a miniature version of a real network and allow detailed implementations of protocols and network services. With the advantages afforded by parallel processing, significant improvements in simulation speeds might be achieved and consequently permit larger experimental runs on more complex models.

Typically, /

Typically, a hardware model of a packet or circuit switched network might consist of an array of interconnectable microprocessors, each required to concurrently emulate the operation of individual nodes. Further processing elements could be configured to act as traffic sources, and model the behaviour of host computers and local communication circuits. Central to the system, a control computer would act as a user interface and allow experiments to be set-up, measured and recorded by means of an ergonomically efficient command language.

Implementation of a hardware simulator would initially be a more complex and costly exercise than a conventional software model, due to the design and construction of non-standard hardware. However, from the rapidly developing field of distributed microprocessor architecture[57] methods of physically and logically connecting many processors into an integrated form are becoming common practice. Therefore, in the long term, the hardware simulator would constitute a very usable simulation tool, and result in substantial economies of time and effort.

REFERENCES

1 Narendra K S and Thathachar M A L
'Learning Automata - A Survey'
I E E E Transactions on Systems Man and Cybernetics
Vol SMC-4 No 4 July 1974 pp 323-334

2 'Tsetlin M L
'On the Behaviour of Finite Automata in Random Media'
Automatika i Telekhanida Vol 22 No 10
October 1961 pp 1345-1354

3 Krylov V U
'On One Stochastic Automaton which is Asymptotically
Optimal in a Random Media'
Automation and Remote Control 24 1963 pp 1114-1116

4 Krinskii V I
'Asymptotically Optimal Automaton with Exponential Speed
of Convergence'
Biofizika No 4 pp 484-489 1964

5 Varshavskii V and Vorontsava I
'On the Behaviour of Stochastic Automata with Variable
Structure'
Automata i Telemekh 24 1963 pp 353-360

6 Viswanathan R and Narendra K S
'Simulation Studies of Stochastic Automata Models Part 1:
Reinforcement Schemes'
Tech Report CT-45 Becton Center Yale University
December 1971

7 Lakshmivarahan S and Thathachar M A L
'Optimal Non-Linear Reinforcement Schemes for Stochastic
Automata'
Information Science 1973 pp 78-103

8 /

8 Viswanathan R and Narendra K S
'A Note on the Linear Reinforcement Scheme for Variable-Structure Stochastic Automata'
I E E E Transactions on Systems Man and Cybernetics
Vol SMC-2 No 2 April 1972 pp 292-294

9 Viswanathan R and Narendra K S
'Stochastic Automata Models with Applications to Learning Systems'
I E E E Transactions on Systems Man and Cybernetics
Vol SMC-3 No 1 pp 107-111

10 Glorioso R M and Colon Osorio F C
'Engineering and Intelligent Systems'
Digital Press 1980

11 Neville R G, Chrystall M S and Mars P
'Application of a Hierarchical Structure Stochastic Learning Automaton'
S and I S Report No 7906 Becton Center Yale University September 1979

12 Neville R G and Mars P
'Adaptive Control of Multi-Modal Stochastic Systems using Learning Automata'
I E E Control Conference Warwick University March 1981

13 Mars P and Poppelbaum
'Stochastic and Deterministic Averaging Processors'
Peter Peregrinus I E E 1981

14 Neville R G and Mars P
'Hardware Design for a Hierarchical Structure Stochastic Learning'
Journal of Cybernetics and Information Science Vol 2
No 1 Spring 1979

15 Loui M C and Narendra K S
'Comparison of Learning Automata Operating in Non-Stationary /

Non-Stationary Environments'
Tech Report CT-65 Becton Center Yale University
May 1975

16 Narendra K S and Thathachar M A L
'On the Behaviour of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing'
I E E E Transactions on Systems Man and Cybernetics
vol SMC-10 No 5 May 1980 pp 262-269

17 Srikanta Kumar P R and Narendra K S
'Learning Model for Telephone Traffic Routing'
to appear in SIAM Journal of Control and Optimisation

18 Mackie N J, Chrystall M S and Mars P
'Some Aspects of Stochastic Learning Automata in Non-Autonomous Environments'
Tech Report School of Electronic and Electrical Engineering RGIT May 1979

19 Schwartz M
'Computer Communication Network Design and Analysis'
Prentice Hall 1977

20 Akass J K
'The Ptarmigan System'
Communications 78 I E E Conference Proceedings
pp 298-300

21 Hills M T
'Telecommunications Switching Principles'
George Allan and Unwin 1978

22 American Telephone and Telegraph Company, Engineering and Network Services Department, Systems Planning System
'Notes on Distance Dialing'
Internal Notes 1975

23 Freeman R L
'Telecommunication System Engineering'
Wiley-Interscience 1980

24 /

24 Glorioso R M, Grueneich G R and McElroy D
'Adaptive Routing in a Large Communications Network'
Proceedings I E E E Symposium on Adaptive Processes
1970 pp 5.1-5.4

25 Narendra K S, Wright E A and Mason L G
'Application of Learning Automata to Telephone Traffic Routing Problems'
I E E E Transactions on Systems Man and Cybernetics
November 1977 pp 785-792

26 Knuth D E
'The Art of Computer Programming: Semi-Numerical Algorithms'
Addison-Wesley 1969

27 Kleinrock L
'Queueing Systems Volume One Theory'
Viley-Interscience 1975

28 Chrystall M S, Coutts M J and Brown A W
'Telephone Network Simulation Package: User Manual'
Tech Report School of Electronic and Electrical
Engineering RGIT January 1980

29 Chrystall M S and Brown A W
'DEC-20 Circuit Switched Simulation Package: Program Details'
Tech Report School of Electronic and Electrical
Engineering RGIT October 1980

30 Masterman P W
'A Method for the Analysis of Circuit-Switched Communications Networks'
Report No 74009 Signals Research and Development
Establishment June 1974

31 Benes V
'Mathematical Theory of Connecting Networks and Telephone Traffic'
Academic /

Academic Press New York 1965

32 Narendra K S and McKenna D M
'Simulation Study of Telephone Traffic Routing using Learning Algorithms Part 1'
S and I S Report No 7806 Department of Engineering and Applied Science Yale University December 1979

33 Narendra K S, Mars P and Chrystall M S
'Simulation Study of Telephone Traffic Routing using Learning Algorithms Part II'
S and I S Report No 7907 Department of Engineering and Applied Science Yale University October 1979

34 Mars P and Chrystall M S
'Real-Time Telephone Traffic Simulation using Learning Automata Routing'
S and I S Report No 7909 Department of Engineering and Applied Science Yale University November 1979

35 Private Communication with J Ash
Bell Laboratories September 1979

36 Chrystall M S and Mars P
'Learning Automata Routing in a Realistic Network'
Tech Report School of Electronic and Electrical Engineering RGIT October 1980

37 Tanenbaum A S
'Computer Networks'
Prentice Hall 1980

38 Davies, Barber, Price and Solomonides
'Computer Networks and their Protocols'
Wiley 1979

39 Prosser R T
'Routing Procedures in Communication Networks Part 1 Random Procedures (also Part II)'
I R E Transactions on Communications Systems Vol CS-10 pp 322-329 (and pp 329-335) December 1962

40 /

40 Baran P
'On Distributed Communication Networks'
I E E E Transactions on Communications Systems Vol CS-12
pp 1-9 March 1964

41 Fultz G L
'Adaptive Routing Techniques for Message Switching
Computer Communication Networks'
University of California Los Angeles Report UCLA-Eng-7352
July 1972

42 Fultz G L and Kleinrock L
'Adaptive Routing Techniques for Store and Forward
Computer Communication Networks'
Proc of the Int Conference on Comms 39-1 39-8
June 1971

43 Cantor D G and Gerla M
'Optimal Routing in a Packet Switched Computer Network'
I E E E Transactions on Computers Vol C-23 October
1974 pp 1062-1968

44 Chou W and Frank H
'Routing Strategies for Computer Network Design'
Proceedings of Symposium on Computer Communications
Networks and Teletraffic Polytechnic Institute of
Brooklyn Brooklyn N Y pp 301-309 April 1972

45 Mueller H
'A SIMPL/1 Simulation Model for Packet Switched Networks'
I B M Zurich Research Report January 1977

46 Rudin H
'On Routing and 'Delta Routing': A Taxonomy and
Performance Comparison of Techniques for Packet-Switched
Networks'
I E E E Transactions on Communications Vol C-24
No 1 January 1976 pp 43-59

47 /

47 Boehm B W and Mabley R L
'Adaptive Routing Techniques for Distributed Communication Systems'
I E E E Transactions on Communication Technology
Vol COM-17 No 3 June 1969 pp 340-349

48 Fratta L, Gerla M and Kleinrock L
'The Flow Deviation Method: an Approach to Store and Forward Communication Network Design'
Networks 3 97 1973

49 McQuillan J M, Falk G and Richer I
'A Review of the Development and Performance of the ARPANET Routing Algorithm'
I E E E Transactions on Communications Vol COM-26
No 12 December 1978 pp 1802-1811

50 Kleinrock L
'Queueing Systems - Vol 2 Computer Applications'
Wiley-Interscience 1976

51 McQuillan J M, Richer I and Rosen E C
'The New Routing Algorithm for the ARPANET'
I E E E Transactions for Communications Vol COM-28
pp 711-719 May 1980

52 Gallager R G
'A Minimum Delay Routing Algorithm using Distributed Computation'
I E E E Transactions on Communications Vol COM-25
No 1 January 1977 pp 73-84

53 Chrystall M S and Brown A W
'User Data for the Store and Forward Network Simulation Package'
Tech Report 8102 School of Electronic and Electrical Engineering RGIT March 1981

54 /

54 Chrystall M S and Mars P
'Learning Automata Routing in Message Switched Communication Networks'
Tech Report 8101 School of Electronic and Electrical Engineering RGIT February 1981

55 Chrystall M S and Mars P
'Learning Automata Adaptive Routing for Datagram and Virtual Call Packet Switched Networks'
Tech Report 8103 School of Electronic and Electrical Engineering RGIT September 1981

56 Price W L
'Data Network Simulation Experiments at the National Physical Laboratory 1968-1976'
Computer Networks 1 1977 pp 199-210

57 Weitzman C
'Distributed Micro/Minicomputer Systems'
Prentice Hall 1980

APPENDIX 1

Bell Circuit Switched Network Specifications

The data presented in this appendix originates from network specifications furnished by Bell Laboratories, New Jersey[35]. This network has been designed using an off-line optimisation technique to generate a topology and alternate path routing rule which gives an optimal performance for the given traffic conditions. Details of the network topology, traffic statistics and routing data are now listed.

Network Topology

The network topology, as displayed on Figure A1.1, is a highly connected mesh structure, consisting of 10 nodes and 35 trunk groups, providing a total capacity of 959 links. The capacity of individual trunks are indicated on the diagram.

Traffic Statistics

For the simulation model used in this study, the necessary traffic statistics include the point to point arrival rates and an overall mean call hold time. In order to permit non-stationary traffic studies, three matrices are specified with a fourth used to simulate the adverse conditions imposed by selective traffic overloads.

Mean Call Hold Time $1/\mu$ = 5 Minutes/call

Total Traffic Rate λ :	Matrix 1	128 calls/min
	Matrix 2	114 calls/min
	Matrix 3	95 calls/min
	Matrix 4	128 calls/min

Routing Data

Precise details of the optimal alternate path routing schemes are now presented. The operation is self-explanatory, each table providing routing information for a particular node, covering the sequence for attempting to route a call to all possible destinations. The /

The same data constitutes the action sets for the automata in the learning routing scheme.

Routing Table at Node 1

2: 2, 4, 3, 6, 5, 9, 7
3: 3, 2, 4, 5, 7, 6, 9
4: 4, 2, 3, 6, 5, 7, 9
5: 5, 3, 6, 2, 4, 7, 9
6: 6, 2, 4, 5, 3, 7, 9
7: 7, 3, 5, 4, 2, 6, 9
8: 9, 2, 4, 5, 6, 3, 7
9: 9, 2, 5, 4, 3, 7, 6
10: 2, 9, 5, 4, 6, 3, 7

Routing Table at Node 2

1: 1, 4, 3, 6, 5, 9, 8, 10
3: 3, 1, 4, 6, 5, 9, 8, 10
4: 4, 1, 3, 6, 8, 5, 9, 10
5: 1, 6, 3, 4, 5, 9, 8, 10
6: 6, 4, 1, 3, 5, 8, 10, 9
7: 1, 3, 4, 6, 5, 8, 9, 10
8: 8, 9, 4, 10, 1, 6, 3, 5
9: 9, 8, 1, 20, 4, 6, 3, 5
10: 10, 8, 9, 1, 4, 6, 3, 5

Routing Table at Node 3

1: 1, 2, 4, 5, 7, 6
2: 2, 1, 4, 5, 7, 6
4: 4, 1, 2, 5, 7, 6
5: 5, 1, 7, 4, 2, 6
6: 5, 4, 2, 1, 7, 6
7: 7, 5, 1, 4, 2, 6
8: 5, 2, 4, 7, 1, 6
9: 5, 1, 2, 7, 4, 6
10: 5, 2, 1, 7, 4, 6

Routing Table at Node 4

1: 1, 2, 3, 6, 7, 8
2: 2, 1, 3, 6, 8, 7
3: 3, 1, 2, 6, 7, 8
5: 6, 3, 1, 2, 7, 8
6: 6, 2, 1, 3, 7, 8
7: 7, 3, 1, 6, 2, 8
8: 8, 2, 1, 6, 3, 7
9: 8, 1, 2, 6, 3, 7
10: 8, 2, 1, 6, 3, 7

Routing Table at Node 5 /

Routing Table at Node 5

1: 1, 3, 6, 7, 2, 9, 8, 10

2: 1, 6, 3, 2, 7, 9, 8, 10

3: 3, 1, 6, 7, 2, 9, 8, 10

4: 6, 3, 1, 7, 2, 8, 9, 10

6: 6, 7, 8, 9, 10, 3, 1, 2

7: 7, 6, 8, 9, 10, 3, 1, 2

8: 8, 9, 10, 6, 7, 3, 1, 2

9: 9, 8, 10, 6, 7, 1, 3, 2

10: 10, 9, 8, 6, 7, 3, 1, 2

Routing Table at Node 6

1: 1, 2, 4, 5, 7, 3, 8, 10

2: 2, 4, 1, 5, 7, 8, 10, 3

3: 5, 4, 2, 1, 7, 3, 8, 10

4: 4, 2, 1, 5, 7, 8, 3, 10

5: 5, 7, 8, 10, 1, 4, 2, 3

7: 7, 5, 8, 10, 4, 1, 2, 3

8: 8, 5, 10, 7, 2, 4, 1, 3

9: 5, 8, 10, 7, 2, 1, 4, 3

10: 10, 8, 5, 7, 2, 4, 1, 3

Routing Table at Node 7

1: 1, 3, 5, 4, 6, 9, 8, 10

2: 1, 3, 4, 6, 5, 8, 9, 10

3: 3, 5, 1, 4, 6, 8, 9, 10

4: 4, 3, 1, 6, 5, 8, 9, 10

5: 5, 6, 8, 9, 10, 3, 1, 4

6: 6, 5, 8, 9, 10, 4, 3, 1

8: 8, 9, 5, 10, 6, 4, 3, 1

9: 9, 8, 5, 10, 6, 1, 3, 4

10: 10, 8, 9, 5, 6, 3, 1, 4

Routing Table at Node 8

1: 9, 2, 4, 5, 10, 6, 7

2: 2, 9, 4, 10, 6, 5, 7

3: 5, 2, 4, 9, 7, 10, 6

4: 4, 2, 9, 6, 10, 5, 7

5: 5, 9, 10, 6, 7, 2, 4

6: 6, 5, 10, 9, 7, 2, 4

7: 7, 9, 5, 10, 6, 4, 2

9: 9, 10, 5, 7, 6, 2, 4

10: 10, 9, 5, 6, 7, 2, 4

Routing Table at Node 9

1: 1, 2, 8, 5, 10, 7

2: 2, 8, 1, 10, 5, 7

3: 5, 1, 2, 8, 7, 10

4: 8, 1, 2, 10, 5, 7

5: 5, 8, 10, 7, 1, 2

6: 5, 8, 10, 7, 2, 1

7: 7, 8, 5, 10, 1, 2

8: 8, 10, 5, 7, 2, 1

10: 10, 8, 5, 7, 2, 1

Routing Table at Node 10

1: 2, 9, 8, 5, 6, 7

2: 2, 8, 9, 6, 5, 7

3: 5, 2, 9, 8, 7, 6

4: 8, 2, 9, 6, 5, 7

5: 5, 9, 8, 6, 7, 2

6: 6, 8, 5, 9, 7, 2

7: 7, 8, 9, 5, 6, 2

8: 8, 9, 5, 6, 7, 2

9: 9, 8, 5, 7, 6, 2

Matrix 1

	1	2	3	4	5	6	7	8	9	10
1	-	.706	.001	.001	1.066	.42	1.319	.642	1.966	.534
2		-	.165	.001	.848	.186	.696	.958	1.241	.473
3			-	.001	.588	.489	1.057	.406	.001	.001
4				-	.425	.84	.763	.588	.225	.219
5					-	16.93	12.407	2.366	2.289	.634
6						-	.346	1.173	.001	.567
7							-	1.577	.471	.609
8								-	3.762	1.274
9									-	3.399
10										-

Matrix 2

	1	2	3	4	5	6	7	8	9	10
1	-	.001	2.074	.001	1.019	.53	1.092	.001	1.179	.61
2		-	3.526	.001	1.399	.353	.074	.00	2.77	.669
3			-	.687	.48	1.007	.001	.227	.308	.158
4				-	.472	1.09	1.242	.743	.168	.222
5					-	9.372	11.313	1.061	2.447	.691
6						-	.362	1.287	.001	.57
7							-	.78	.378	.338
8								-	1.684	1.186
9									-	2.754
10										-

λ calls/min

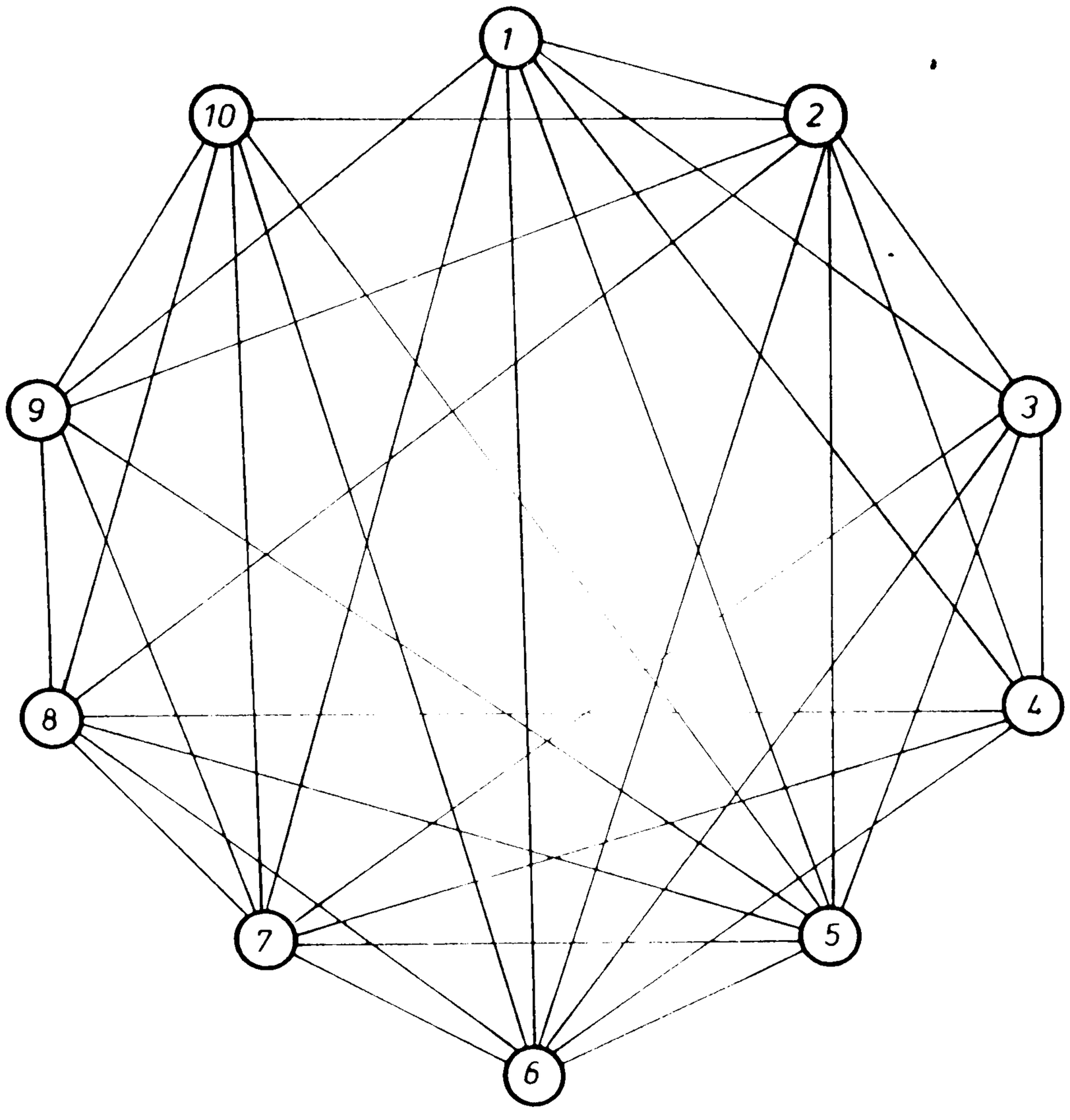
Matrix 3

	1	2	3	4	5	6	7	8	9	10
1	-	.825	3.715	.001	1.919	.001	1.323	.631	1.922	1.166
2		-	1.406	.001	.48	.001	.001	.407	1.02	.455
3			-	.001	.568	1.271	.001	.669	.15	.001
4				-	.001	1.761	1.26	1.091	.011	.416
5					-	6.425	6.79	.484	.477	.468
6						-	.363	1.235	.001	.921
7							-	2.483	.001	1.171
8								-	1.71	.555
9									-	1.943
10										-

Matrix 4

	1	2	3	4	5	6	7	8	9	10
1	-	3.399	1.274	.609	.567	.634	.219	.001	.473	.534
2		-	3.762	.471	.001	2.289	.225	.001	1.211	1.966
3			-	1.577	1.173	2.366	.588	.406	.958	.642
4				-	.846	12.407	.763	1.057	.699	1.319
5					-	16.93	.84	.489	.186	.42
6						-	.425	.588	.848	1.066
7							-	.001	.001	.001
8								-	.165	.001
9									-	.706
10										-

λ calls/min



trunk size matrix

	nodes									
	1	2	3	4	5	6	7	8	9	10
1	-	43	52	12	10	3	15	-	19	-
2	-	-	40	17	1	11	-	20	17	8
3	-	-	-	13	26	1	12	-	-	-
4	-	-	-	-	-	18	8	11	-	-
5	-	-	-	-	-	172	148	28	28	8
6	-	-	-	-	-	-	13	11	-	5
7	-	-	-	-	-	-	-	18	8	6
8	-	-	-	-	-	-	-	-	72	36
9	-	-	-	-	-	-	-	-	-	49
10	-	-	-	-	-	-	-	-	-	-

A1.1 Bell network

APPENDIX 2

NPL and Rudin Packet Switched Networks

This appendix provides details of the two 10 node packet switched networks used in the simulation experiments, including the network topologies, traffic statistics and routing data. The NPL network constitutes a hypothetical virtual call computer network for the U K and was first proposed at the National Physical Laboratory, Teddington, for simulation studies of store and forward networks. The alternative network operates under a datagram protocol and was first introduced by Rudin at IBM Research Laboratories, Zurich as a testbed for adaptive routing studies.

Network Topology

The NPL network (Figure A2.1) consists of 10 nodes and 15 full duplex links, each with a capacity of 9.6 Kbits/sec, giving a total network capacity of 288 Kbits/sec. Figure A2.2 displays the 10 node datagram network which contains 16 links rated at 9.6 Kbits/sec. Both networks differ slightly from the original designs, each having two additional links to increase the number of paths within the network, thus permitting a more interesting routing problem.

Traffic Statistics

For both networks the data traffic is characterised by a normalised point to point message arrival rate matrix and an overall mean message length.

Mean Message Length	: 550	bits/message
Data Packet Size	: 256	bits/packet
Control Packet	: 64	bits/packet

Routing Data

This section presents data for fixed rule and load splitting routing schemes in both networks. The load splitting tables form the basis for the random and learning automata schemes, listing the allowable options at each node, for all possible destinations. The fixed rule data specifies the minimum hop routes between all source/destination pairs.

Fixed Rule

NPL Network

1: -, 2, 3, 4, 3, 4, 3, 3, 3, 3

2: 1, -, 4, 4, 4, 4, 4, 4, 4, 4

3: 1, 1, - 4, 5, 4, 5, 5, 5, 5

4: 1, 2, 3, -, 5, 6, 5, 6, 5, 6

5: 3, 4, 3, 4, -, 8, 7, 8 7, 8

6: 4, 4, 4, 4, 8, -, 8, 8, 8, 8

7: 5, 5, 5, 5, 5, 8, -, 8, 9, 9

8: 6, 6, 5, 6, 5, 6, 7, -, 10, 10

9: 7, 7, 7, 7, 7, 10, 7, 10, -, 10

10: 8, 8, 8, 8, 8, 8, 9, 8, 9, -

Rudin Network

1: -, 2, 3, 3, 2 2, 3, 2, 3, 3

2: 1, -, 3, 3, 5, 6, 5, 6, 5, 5

3: 1, 2, -, 4, 4, 2, 4, 2, 4, 4

4: 3, 5, 3, -, 5, 5, 7, 7, 7, 7

5: 2, 2, 4, 4, -, 6, 7, 6, 7, 7

6: 2, 2, 2, 5, 5, -, 8, 8, 8, 5

7: 4, 5, 4, 4, 5, 8, -, 8, 9, 10

8: 6, 6, 6, 7, 6, 6, 7, -, 9, 9

9: 7, 8, 7, 7, 7, 8, 7, 8, -, 10

10: 7, 7, 7, 7, 7, 9. 7, 9, 9, -

Load Splitting /

Load Splitting

NPL Network

Node 1

2	2
3	3
4	4, 3, 2
5	3, 4
6	2, 4, 3
7	3, 4
8	3, 4, 2
9	3, 4
10	3, 4, 2

Node 2

1	1
3	1, 4
4	4
5	1, 4
6	4
7	1, 4
8	4
9	4
10	4

Node 3

1	1
2	1, 4
4	4
5	5
6	4, 5
7	4, 5
8	4, 5
9	4, 5
10	4, 5

Node 4

1	1, 2, 3
2	2
3	3
5	3, 5
6	6
7	5, 6
8	5, 6
9	5, 6
10	5, 6

Node 5

1	3, 4
2	3, 4
3	3
4	3, 4
6	4, 8
7	7
8	7, 8
9	7, 8
10	7, 8

Node 6

1	4
2	4
3	4
4	4
5	4
7	8
8	8
9	8
10	8

Node 7

1	5
2	5, 8
3	5
4	5, 8
5	5, 8
6	8
8	8
9	9
10	8, 9

Node 8

1	5, 6
2	5, 6
3	5, 6, 7
4	5, 6
5	5, 6
6	6
7	7
9	7, 10
10	10

Node 9

1	7
2	7, 10
3	7
4	7, 10
5	7
6	7, 10
7	7
8	7, 10
10	10

Node 10

1	8, 9
2	8
3	8, 9
4	8
5	8, 9
6	8
7	8, 9
8	8
9	9

Rudin Network /

Rudin Network

Node 1

2	2
3	3
4	2, 3
5	2, 3
6	2, 3
7	2, 3
8	2, 3
9	2, 3
10	2, 3

Node 2

1	1
3	1, 3
4	3, 5
5	5
6	5, 6
7	3, 5, 6
8	5, 6
9	3, 5, 6
10	3, 5, 6

Node 3

1	1
2	1, 2
4	4
5	2, 4
6	2, 4
7	4
8	2, 4
9	4
10	4

Node 4

1	3, 5
2	3
3	3
5	5
6	5, 7
7	5, 7
8	5, 7
9	5, 7
10	5, 7

Node 5

1	2, 6
2	2, 6
3	2, 4
4	4
6	6
7	7
8	6, 7
9	6, 7
10	7

Node 6

1	2
2	2
3	2, 5
4	2, 5
5	2, 5
7	5, 8
8	8
9	8
10	5, 8

Node 7

1	4, 5
2	4, 5
3	4, 5
4	4, 5
5	4, 5
6	5, 8
8	8, 9
9	9, 10
10	10

Node 8

1	6, 7
2	6, 7
3	6, 7
4	6, 7
5	6, 7
6	6
7	7, 9
9	7, 9
10	7, 9

Node 9

1	7, 8
2	7, 8
3	7, 8
4	7, 8
5	7, 8
6	7, 8
7	7, 8, 10
8	8
10	10

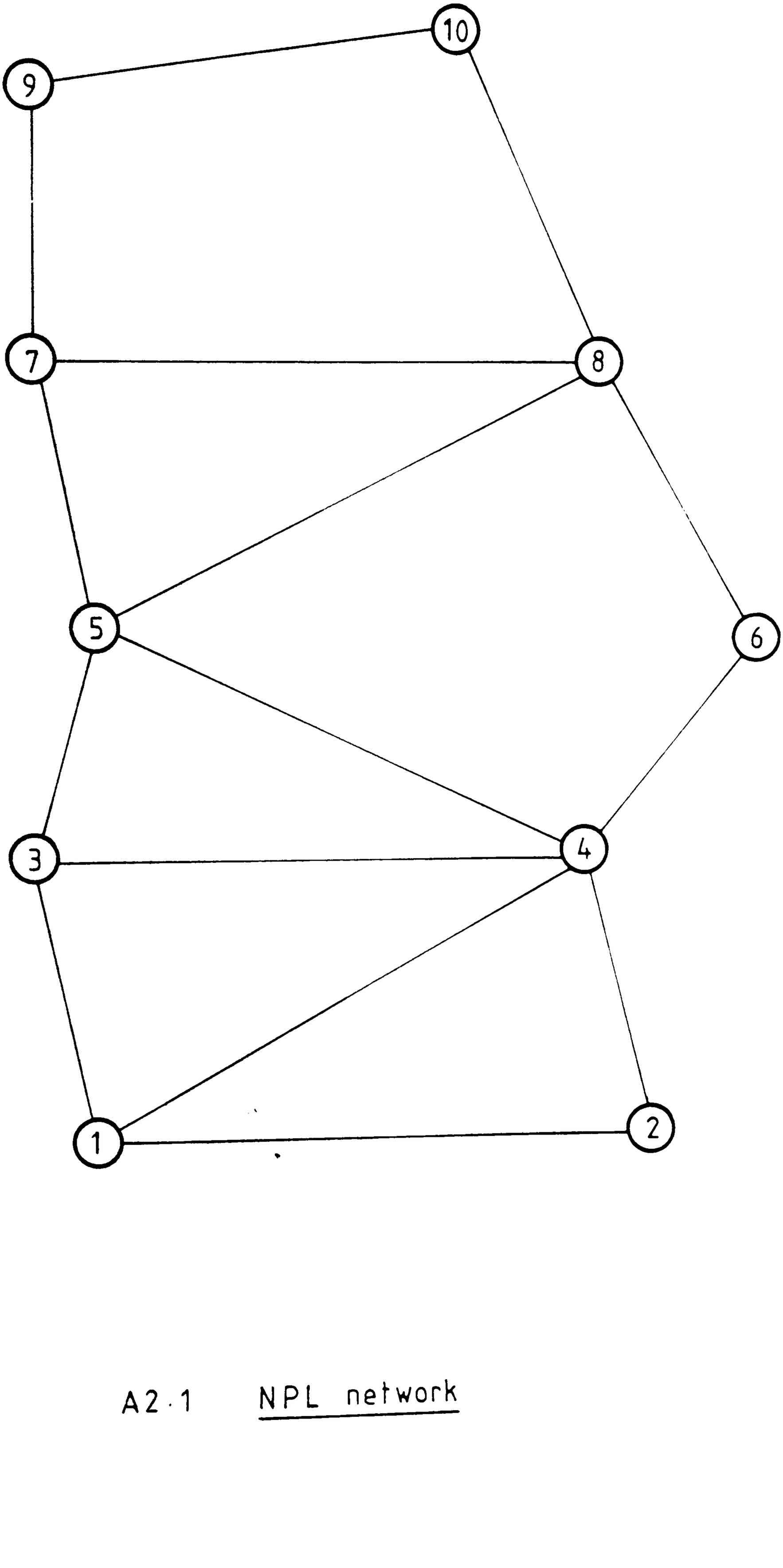
Node 10

1	7, 9
2	7, 9
3	7, 9
4	7, 9
5	7, 9
6	7, 9
7	7
8	7, 9
9	9

NPL Network

	1	2	3	4	5	6	7	8	9	10
1	-	3.314	6.621	9.912	8.261	4.969	6.621	9.912	4.969	3.314
2		-	6.621	9.912	8.261	4.969	6.621	9.912	4.969	3.314
3			-	19.82	16.52	9.937	13.24	19.82	9.937	6.628
4				-	24.74	14.88	19.83	29.68	14.88	9.912
5					-	12.40	16.52	24.74	12.40	8.261
6						-	9.937	14.88	7.457	4.969
7							-	19.82	9.937	6.621
8								-	14.88	9.912
9									-	4.969
10										-

Normalised Arrival rates λ messages/min x 10



A2.1 NPL network

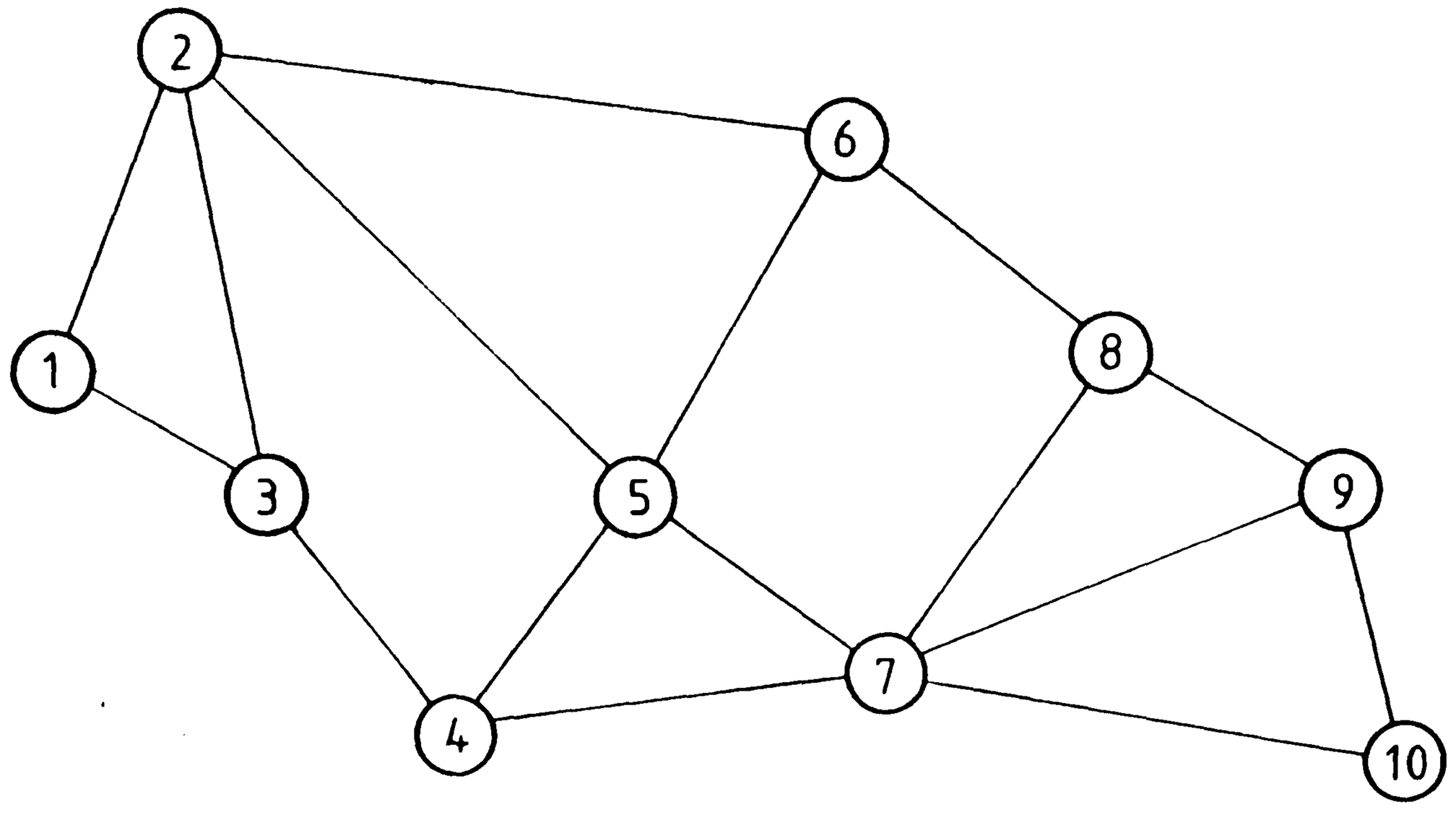


Figure A2.2 Rudin network

ADAPTIVE ROUTING IN COMPUTER COMMUNICATION NETWORKS USING LEARNING AUTOMATA

M S Chrystall and P Mars

School of Electronic and Electrical Engineering
R G I T, Aberdeen AB9 1FR, Scotland, U K

A de-centralised adaptive routing technique based on learning automata concepts is proposed for message-switched computer communication networks. Messages are routed by the automata selecting suitable outgoing links, with the delay experienced by a message used as a feedback response for updating future selection strategy. This simple feedback policy which is a realistic representation of the actual feedback received in communication networks is shown to give a routing performance close to the minimum achievable delay.

1 Introduction

The transmission of digital information has shown tremendous growth over the last decade and will undoubtedly continue to expand in the future. Primarily, this can be attributed to the growing awareness and rapidly developing use of computer networks, which serve to provide distributed computing facilities to geographically dispersed users. Also relevant is the growth of digital electronic switching techniques and through these the implementation of more complex network protocols and routing schemes.

In this paper some initial work is presented on automata routing in small networks including a set of simulation experiments carried out using a recently developed software package. From an analysis of the end to end delay, conditions for optimality are stated and compared with the equilibrium behaviour of various learning algorithms. The experimental studies confirm the theoretical predictions and demonstrate the learning schemes performance in comparison with alternative routing techniques.

The use of learning automata concepts in communication network routing has already been successfully studied in circuit-switched networks, including the telephone network (1)(2)(3). Through formally designed algorithms (with origins in mathematical psychology) the learning automata update their routing behaviour on the basis of call connection or rejection and consequently produce an adaptive scheme whereby the routing structure can alter to suit the given conditions. The possible application of learning automata routing to message-switched networks appears to be a very promising approach.

In such schemes messages are routed by the automata selecting suitable outgoing links, with the delay experienced by a message used as a feedback response for updating future selection strategy. This simple feedback policy can be very conveniently applied to real communication networks and is shown to give a routing performance close to the minimum achievable delay.

2 Routing in Message-Switched Networks

The operation of a communication network is critically dependent on the behaviour and performance of the routing algorithm. In a store and forward network, this algorithm operates to ensure the systematic progression of each message from source to destination, normally attempting to minimise the average message delay. The design of an optimal scheme to implement and maintain this behaviour is however difficult, with the network suffering from topological changes and the traffic exhibiting statistical non-stationarily due to user trends, e.g., peak and off-peak loads. The optimal routing scheme must therefore be capable of adapting to any conditions experienced.

Adaptive routing schemes provide a mechanism which can carry out routing decisions on the basis of feedback from the network, reacting to congestion and facility failures with intelligently modified routing actions. Past studies have classified adaptive

schemes in four distinct doctrines; Centralised, Isolated, Hybrid and Distributed. For an excellent review of routing schemes, the reader is referred to (4).

Centralised adaptive routing schemes rely on a central routing facility to assemble the global network state in order to formulate a routing strategy and broadcast the necessary routing data to the individual nodes. Schemes of this nature have therefore the potential for optimal routing, but unfortunately this is only achieved at the expense of increased control overheads and an undesirable reliance on a common control element. In isolated routing schemes, decisions are made on a purely local basis, from information readily available at the node. This policy is however myopic, since no attempt is made to gain a global network awareness, which consequently results in a non-optimal performance. The hybrid form of routing (5) may be regarded as a mixed strategy, attempting to combine the best features of the other doctrines.

Distributed schemes offer the most practical form of adaptive routing. In general, this approach allows nodes to implement local routing decisions, but unlike the isolated technique, these are supported by global feedback, gathered in co-operation with other nodes. A distinction must be drawn between distributed routing schemes which specify a single path between end users and others which operate to spread the traffic over several paths. In the first category the ARPANET (6) routing algorithm is included, this scheme attempting to find the shortest (minimum delay) paths in the network, from data acquired through all nodes exchanging details of the link delays in their own vicinity.

Only the load splitting schemes have the capability to provide optimum routing, by adaptively proportioning traffic over the allowable paths. To provide a scheme of this nature, Gallager (7) has proposed a technique which attempts to equalise the differential delays, using feedback which is measured and periodically exchanged between nodes. In the learning automata schemes an asynchronous updating policy exists, the automaton revising the routing strategy when a response to a previously routed message is received. Also, the algorithms utilised by the automata provide a simple and extremely practical solution to the routing problem, which although do not produce the required conditions for true optimum routing, result in a very close to optimum performance.

3 Learning Automata

Typically, a learning automaton operates in such a manner as to choose an optimal action from an allowable set and to apply the selected action to a random environment. In turn, the environment responds with a feedback signal, which initiates an updating of the internal state vector responsible for the future action selection process. Consider the learning automaton/environment configuration as shown in Figure 1. This illustrates an automaton capable of performing r-actions operating on a single random environment, which in general is a media of which little or no apriori information is available.

A wide body of knowledge has now been established on the theory and application of learning automata and as a result, different forms of automata have been proposed, including deterministic, stochastic, fixed and variable structured, all of which are surveyed in (8). For this study, we concentrate on variable structure stochastic automata, and in particular, those where the environmental response is of a binary nature, i.e. Penalty/Reward.

Variable Structure Stochastic Automata

To further consider the form and operation of the learning automaton, consider A(n)

$$A(n) = \{\underline{\alpha}, \underline{\beta}, \underline{p}, T(\alpha, \beta, \underline{p})\}$$

where

$\underline{\alpha}$: The Action Set

The action set of the automaton is the range of allowable outputs. The performed action at stage n is $\alpha(n)$ where

$$\alpha(n) \in \{\alpha_1, \ldots \alpha_r\}$$

$\underline{\beta}$: The Response Set

The input to the automaton $\beta(n)$ is the response of the environment at stage n, where

$$\beta(n) \in \{0, 1\}$$

0 - Reward
1 - Penalty

$\underline{p}$: The Action Probability Set

The internal state of a variable

structure atomaton is specified by the action probability at $\underline{p}$

$$\underline{p}(n) = \{p_1 \ldots p_r\}$$

where p_i = prob $[\alpha(n) = \alpha_i]$

T (α, β , $\underline{p}$) : The Reinforcement Algorithm

The reinforcement algorithm provides the necessary means to modify the action probability vector in relation to the performed action and received response.

$$\underline{p}(n+1) = T\ (\alpha\ ,\ \beta\ ,\ \underline{p}(n))$$

A range of algorithms may be synthesised, although the most popular is the Linear Reward Penalty L_{RP} and from this the Linear Reward Inaction L_{RI}, where only a reward response produces a modification of the action probabilities.

L_{RP} Algorithm for r-action

Reward on α_i

$$p_{j\neq i}(n+1) = (1-a)p_j(n)$$

$$p_i(n+1) = 1 - \sum_{j\neq i} p_j(n+1)$$

$$0 < a, b < 1$$

Penalty on α_i

$$p_{j\neq i}(n+1) = p_j(n) + \{\frac{b}{r-1}\}p_i(n)$$

$$p_i(n+1) = (1-b)p_i(n)$$

The Environment

The environment is described by the triple E(n)

$$E(n) = \{\underline{\alpha}\ ,\ \underline{\beta}\ ,\ \underline{c}\}$$

where $\underline{\alpha}$ represents the input set of the environment (output of the automaton), $\underline{\beta}$, the response set and $\underline{c}$ the penalty set.

$\underline{c}$: The Penalty Set

The penalty set dictates the probability that a given action will receive a penalty response $\beta(n) = 1$. This set therefore consists of r probabilities

$$\underline{c} = \{c_1, \ldots c_r\}$$

where

$$c_i = \text{prob}\ [\beta(n)=1 \mid \alpha(n)=\alpha_i]$$

Non-Autonomous Environments

The penalty set of an environment can take several forms. If the individual elements c_i are constants for all time, the environment is classified as stationary. However in most learning automata applications, the environmental characteristics will vary with time and the term non-stationary is adopted. A further type of non-stationary environment is the non-autonomous environment,(9) where the response behaviour is influenced as a result of the actions performed by the automata.

In the dynamic routing problem, the network represents a non-autonomous environment, when an automata routing controller is used to select suitable outgoing links. By selecting a certain link with a high probability, the path obviously becomes less attractive due to increased delay caused by the higher traffic rate. On the other hand the alternative links are required to handle less traffic and consequently a more favourable response is obtained when a message is routed on any of these.

In this study of learning routing schemes the convergence of the automata, operating with the different reinforcement algorithms, is an important aspect. Consider the condition

$$\Delta p_i(n) \triangleq p_i(n+1) - p_i(n)$$

for the r action L_{RI} automaton, in a non-autonomous environment

$$\Delta p_i = a\ p_i [\sum_{j\neq i} p_j c_j - c_i \sum_{j\neq i} p_j]$$

if $\Delta p_i(n) \to 0$ as $n \to \infty$ then the above condition reveals that $p_i \to 0$ or

$$\sum_{j\neq i} p_j c_j = c_i \sum_{j\neq i} p_j$$

which corresponds to an equalisation of the penalty probabilities c_i.

Similarly for the r action L_{RP} algorithm, the condition

$$\Delta p_i = a\ [\frac{1}{r-1} \sum_{j\neq i} p_j c_j - p_i c_i] \qquad a = b$$

shows that the equilibrium condition in a non-autonomous environment results in an equalisation of the penalty rates $p_i c_i$

$$p_i c_i = \frac{1}{r-1} \sum_{j\neq i} p_j c_j$$

4 Learning Automata Routing

The notion of a stochastic learning automaton forms an ideal building block for the implementation of distributed adaptive routing schemes in message-switched networks. In a typical application (Figure 2), the allowable set of outgoing links correspond to the automaton action set, the routing operation selecting a link on the basis of the action probabilities. The response to the performed action is obtained by passing the delay experienced by a message to the simple

stochastic comparator arrangement, which consequently produces the necessary feedback to the reinforcement scheme.

A simple network/learning automaton combination is now proposed with a view to establishing an understanding of the basic routing behaviour in a message-switched network. Consider the 4 node arrangement shown on Figure 3, with a single source of traffic (rate λ, length $1/\mu$) at node 1, generating messages destined for node 3. An automaton at node 1 is used to select a suitable path for a message, the path options being performed with probabilities p_1 and p_2.

Assuming Poisson/Exponential statistics and the Independence assumption, the average delay of all messages in the network is given by

$$\bar{T} = \frac{p_1}{(\mu l_1 - p_1\lambda)} + \frac{p_1}{(\mu l_3 - p_1\lambda)} + \frac{p_2}{(\mu l_2 - p_2\lambda)} + \frac{p_2}{(\mu l_4 - p_2\lambda)}$$

Optimal Conditions

To find the conditions for minimum delay equate

$$\frac{d\bar{T}}{dp_1} = 0 \qquad p_2 = 1-p$$

which produces the following conditions for optimality

$$\frac{\mu l_1}{(\mu l_1 - p_1\lambda)^2} + \frac{\mu l_3}{(\mu l_3 - p_1\lambda)^2} = \frac{\mu l_2}{(\mu l_2 - p_2\lambda)^2} + \frac{\mu l_4}{(\mu l_4 - p_2\lambda)^2}$$

As shown by Gallager (7) the conditions for minimum delay are produced by a routing algorithm which equalises the differential delay over the alternative routing options.

Automaton Steady State Conditions

For the L_{RI} reinforcement algorithm, we expect an attempt to equalise the penalty probabilities

$$E[c_1(n)] = E[c_2(n)] \quad \text{as } n\to\infty$$

In this particular application, the penalty responses result from the operation of the stochastic comparator, hence

$E[c_i(n)] = E[T_i > u]$ where u is a uniform deviate,

$= \bar{T}_i$ T_i the delay experienced on path i, and $\bar{T}_i$ is the average delay

The steady state operation of the L_{RI} routing controller is therefore noted by an equalisation of the average path delays. For the simple network

$$\frac{1}{\mu l_1 - p_1\lambda} + \frac{1}{\mu l_3 - p_1\lambda} = \frac{1}{\mu l_2 - p_2\lambda} + \frac{1}{\mu l_4 - p_2\lambda}$$

Under similar circumstances, the L_{RP} automaton is expected to equalise the penalty rates, with

$$E[p_1c_1] = E[p_2c_2] \quad n\to\infty$$

In the message-switched network, this leads to

$$p_1E[T_1] = p_2E[T_2]$$

which simply corresponds to an equalisation of the accumulated path delays

$$\frac{N_1}{N}\frac{\Sigma T_1}{N_1} = \frac{N_2}{N}\frac{\Sigma T_2}{N_2}$$ where N_i is the number of messages routed on path i, $N = N_1 + N_2$

$$\Sigma T_1 = \Sigma T_2$$ as $N\to\infty$

which produces

$$\frac{p_1}{\mu l_1 - p_1\lambda} + \frac{p_1}{\mu l_3 - p_1\lambda} = \frac{p_2}{\mu l_2 - p_2\lambda} + \frac{p_2}{\mu l_4 - p_2\lambda}$$

To further illustrate the alternative routing strategies of the L_{RI} and L_{RP} reinforced algorithms and their relationship with optimality, Figure 4 displays the convergence diagram for the simple 4-node network (l_1, l_3=400 K bits/sec and l_2, l_4 = 600 K bits/sec with traffic λ = 1100 messages/sec and $1/\mu$ = 350 bits/message). On this diagram, the steady state condition of the L_{RI} automaton is located at the interception of the average delay graphs which are drawn as functions of the routing probability p_1. Similarly for the L_{RP} scheme, the crossover point of $p_i T_i$ produces the equilibrium condition, which as shown previously results in an equalisation of the accumulated path delays. Furthermore, by inspection the true optimum may be noted, which for the conditions specified, shows the learning schemes to be marginally sub-optimal.

The effects of the traffic conditions are displayed on Figure 5, this showing a family of overall delays $\bar{T}$, for a range of arrival rates λ. From this diagram, the expected behaviour of various routing schemes may be noted, the L_{RI} and L_{RP} schemes established from the previous conditions, the random routing scheme for $p_1 = p_2 = 0.5$ and the true optimal condition from the equalisation of the differential delays.

5 Simulation Studies

In order to verify the theoretical

predictions of the previous section, a range of simulation experiments have been performed on the simple network using a computer model of a message-switched network. The experiments include studies of the learning schemes and alternative routing techniques such as random and proportional routing. The above network parameters are maintained.

Random Routing

With this scheme, an outgoing link is selected on a random basis ($p_1=p_2=0.5$). The results presented (Table 1) were averaged over the last 2500 messages of a 3000 message sample.

Proportional Routing

The simple proportional routing as suggested in (5), has been implemented and results (Table 2) obtained.

Learning Automata Scheme

The results for the learning automata are presented in Table 3 (L_{RP}) and Table 4 (L_{RI}).

The results confirm the expected convergence conditions for the learning schemes and provide further information on the alternative techniques. The L_{RI} algorithm has been shown to equalise the average delays on each path, with the L_{RP} scheme providing a means to balance the accumulated delays for the paths. The performance by the two linear schemes in this study provides an interesting dual with the previously studied behaviour of the same algorithms in circuit-switched networks, where the L_{RI} was shown to equalise the path blocking probabilities, with the L_{RP} providing an equalisation of the blocking rates (or the accumulated number of blocked calls).

The operation of the proportional routing scheme is also shown to equalise the average delays, which might be expected since the simple implementation resembles an automaton. As shown by the results, the random scheme provides a comparable service when the network traffic is light, although when heavier conditions exist this shows a significant degradation with respect to the adaptive schemes.

λ	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	$\sum T_1$	$\sum T_2$	$\sum T$
1000	3.06	1.79	2.43	3.83	2.24	6.07
1200	4.14	1.82	3.0	5.25	2.24	7.49
1400	4.87	1.89	3.38	6.09	2.36	8.45
1600	4.86	2.31	3.57	6.01	2.92	8.93
1800	6.36	2.42	4.37	7.87	3.06	10.93

Table 1

λ	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	$\sum T_1$	$\sum T_2$	$\sum T$	p_1
1000	2.43	2.41	2.42	0.57	4.27	4.83	0.177
1200	2.76	2.60	2.63	1.19	4.07	5.26	0.217
1400	2.96	2.63	2.74	1.91	3.56	5.48	0.324
1600	3.65	3.59	3.61	2.50	4.73	7.22	0.342
1800	3.79	3.82	3.81	2.60	5.02	7.63	0.343

Table 2

λ	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	$\sum T_1$	$\sum T_2$	$\sum T$	p_1
1000	2.04	3.02	2.44	2.41	2.47	4.88	0.508
1200	3.31	2.09	2.59	2.71	2.47	5.18	0.409
1400	3.18	2.51	2.80	2.72	2.88	5.60	0.427
1600	3.98	2.77	3.25	3.16	3.34	6.5	0.397
1800	4.24	2.87	3.44	3.49	3.38	6.88	0.411

Table 3

λ	$\bar{T}_1$	$\bar{T}_2$	$\bar{T}$	$\sum T_1$	$\sum T_2$	$\sum T$	p_1
1000	2.11	2.18	2.17	0.71	3.64	4.35	0.167
1200	2.46	2.60	2.57	1.21	3.92	5.14	0.247
1400	2.95	2.99	2.98	1.60	4.37	5.96	0.27
1600	3.21	3.29	3.27	1.82	4.72	6.53	0.283
1800	3.73	3.69	3.70	2.37	5.04	7.41	0.368

Table 4

$\bar{T}$ average delay mSec.

$\sum T$ accumulated delay Sec.

Conclusions

The following conclusions may be made concerning the behaviour of learning automata schemes in simple message-switched networks. Although true optimal performance requires the equalisation of differential delays, for the simple networks considered equalising average delays produces results close to the minimum achievable delay. It should be stressed that the feedback and updating strategy used in the learning automata schemes are very practical and simple.

Future work will involve the application of learning automata schemes to both virtual circuit and datagram complex networks, and attempts will be made to synthesise learning automata reinforcement schemes which result in equalisation of differential delays.

Acknowledgements

The authors would like to thank Professor K S Narendra of Yale University for his continued support on this project. Thanks also to Mr A W Brown and Dr J D Eades for their programming advice. This work was supported by a grant from the U K Science and Engineering Research Council.

References

1 Narendra K S, Mars P and Chrystall M S, "Simulation Study of Telephone Traffic Routing using Learning Algorithms", Tech Report 7907, Dept of Engineering and Applied Science, Yale University, October 1979

2 Mars P and Chrystall M S, "Real-Time Telephone Traffic Simulation using Learning Automata Routing", Tech Report 7909, Dept of Engineering and Applied Science, Yale University, November 1979

3 Chrystall M S and Mars P, "Learning Automata Routing in a Realistic Network", Tech Report, School of Electronic and Electrical Engineering, R G I T, Aberdeen, October 1980

4 Schwartz M and Stern T E, "Routing Techniques used in Computer Communication Networks", IEEE Trans on Communications, Vol COM-28, pp539-552, April 1980

5 Rudin H, "On Routing and 'Delta Routing' : A Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks", IEEE Trans on Communications, Vol COM-24, No 1, pp43-59, January 1976

6 McQuillan J M, Richer I and Rosen E C, "The New Routing Algorithm for the ARPANET" IEEE Trans on Communications, Vol 1 COM-28, pp711-719, May 1980

7 Gallager R G, "A Minimum Delay Routing Algorithm using Distributed Computation", IEEE Trans on Communications, Vol COM-25, No 1, pp73-84, January 1977

8 Narendra K S and Thathachar M A L, "Learning Automata - A Survey", IEEE Trans on Systems, Man and Cybernetics, Vol SMC-4, pp323-334, July 1974

9 Narendra K S and Thathachar M A L, "On the Behaviour of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing", IEEE Trans on Systems, Man and Cybernetics, Vol SMC-10, No 5, pp262-269, May 1980

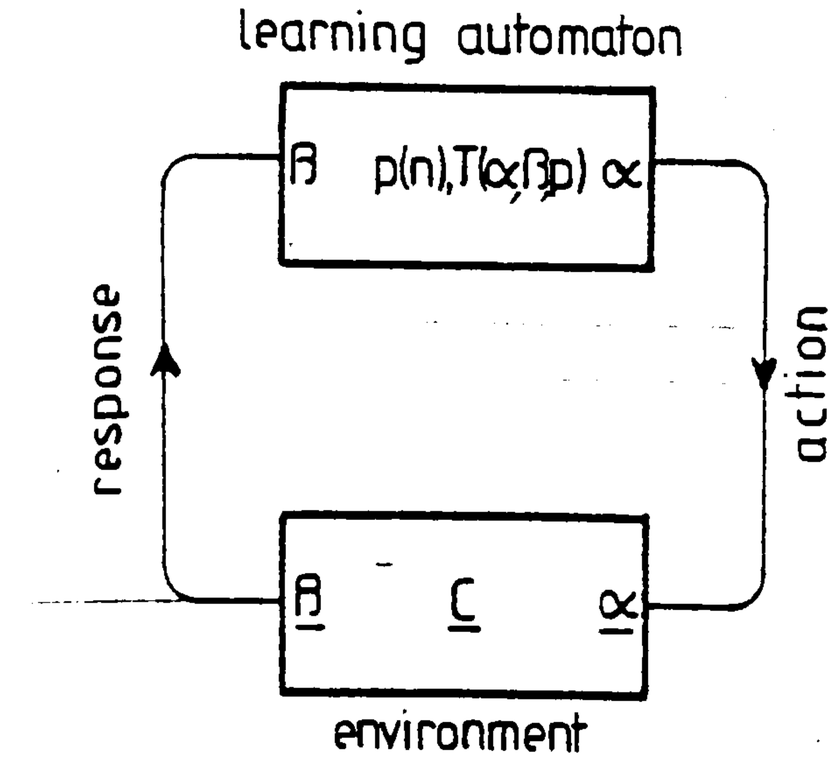


Figure 1 Learning automaton / environment combinations

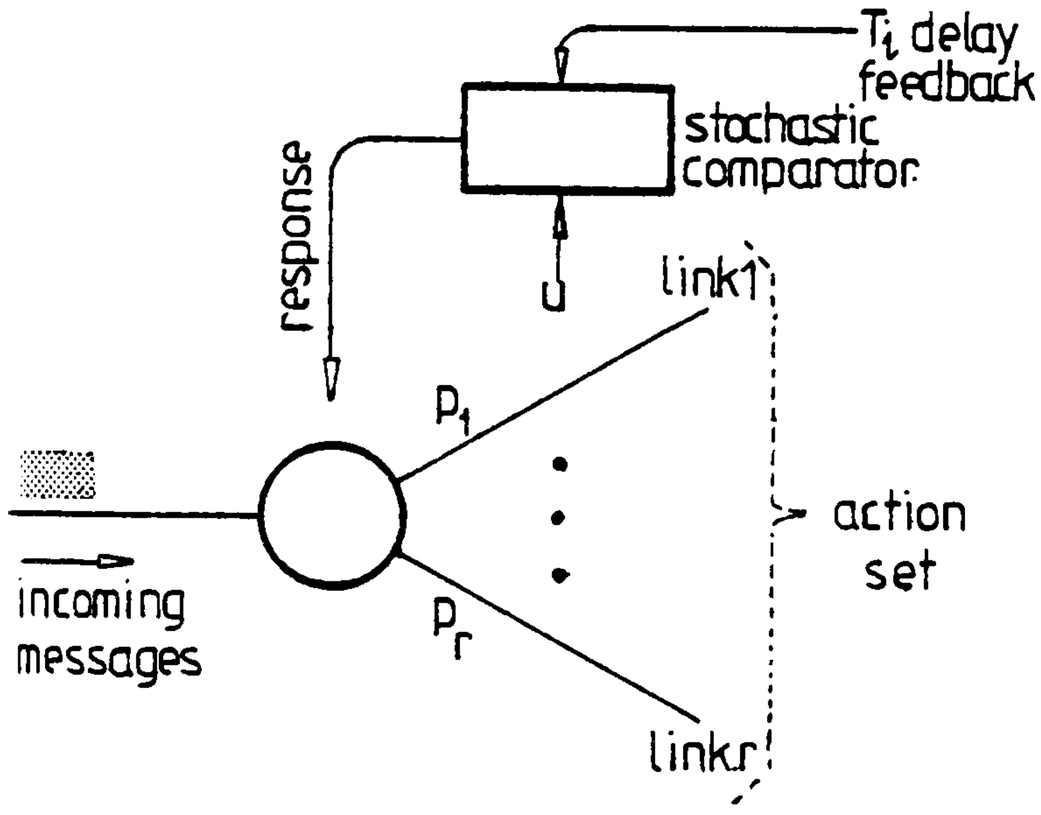


Figure 2 Automaton routing controller

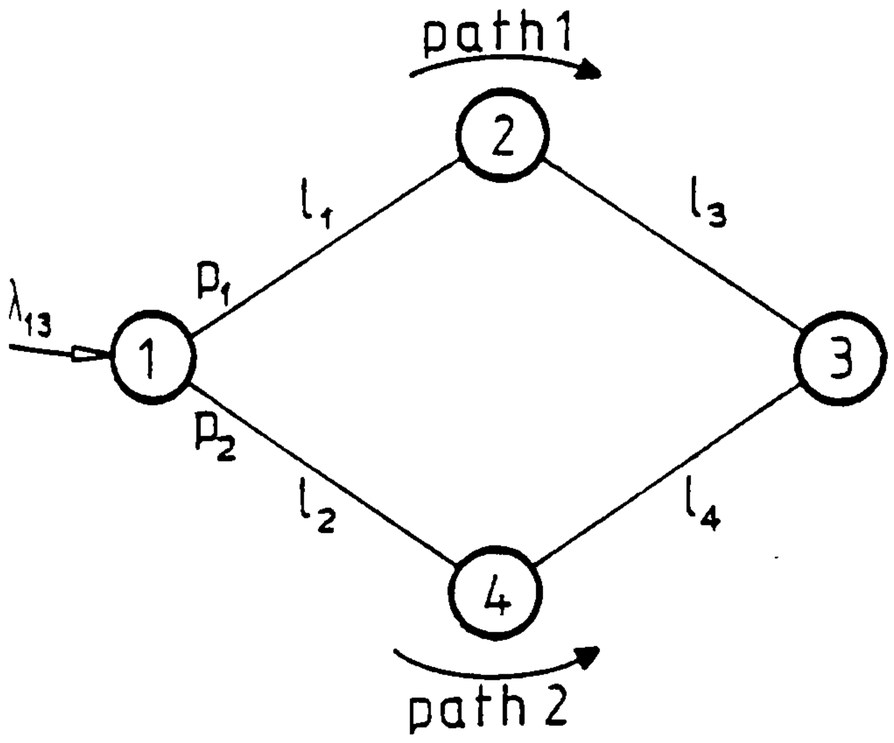


Figure 3 4 node network

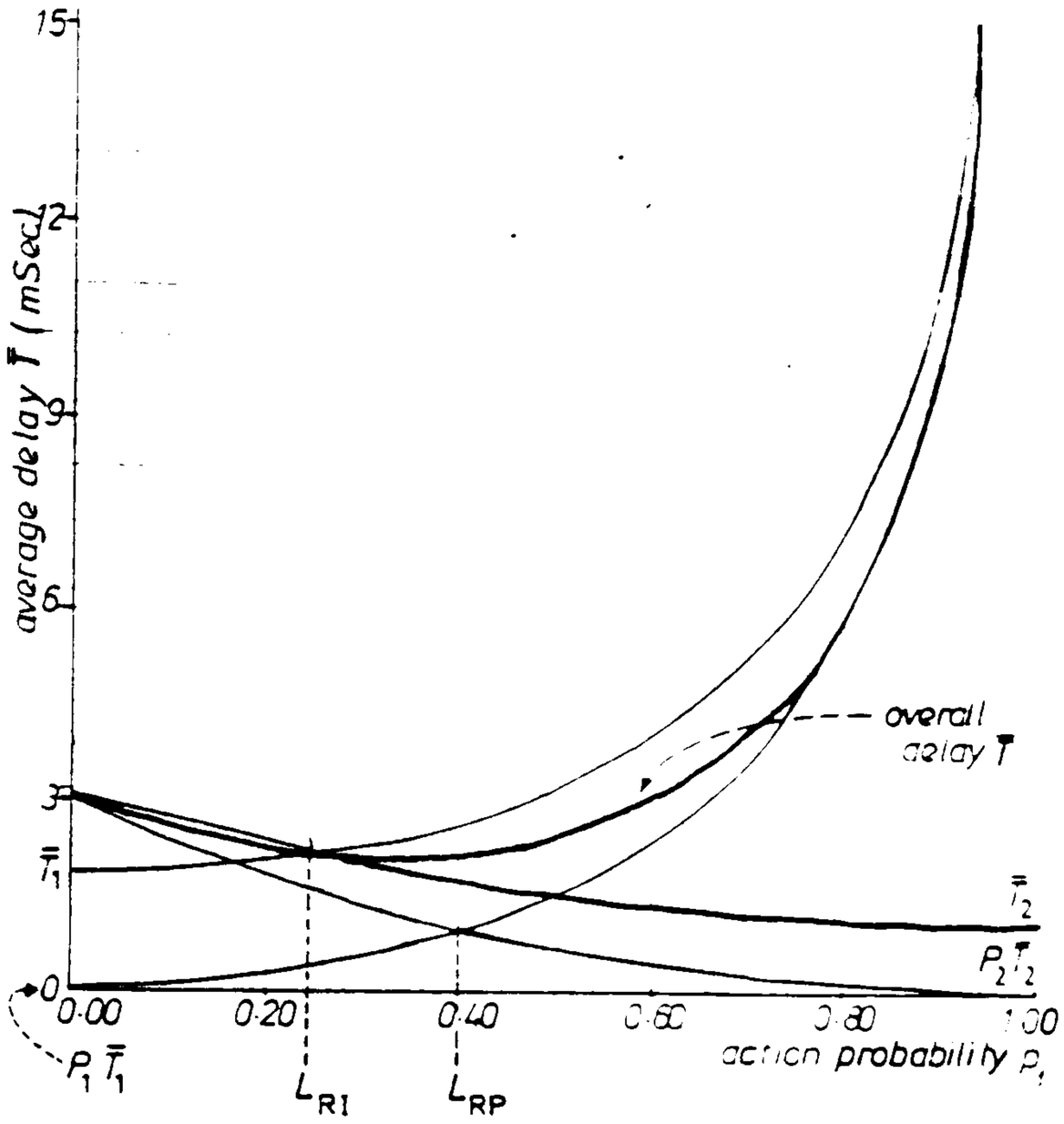


Figure 4 Convergence conditions

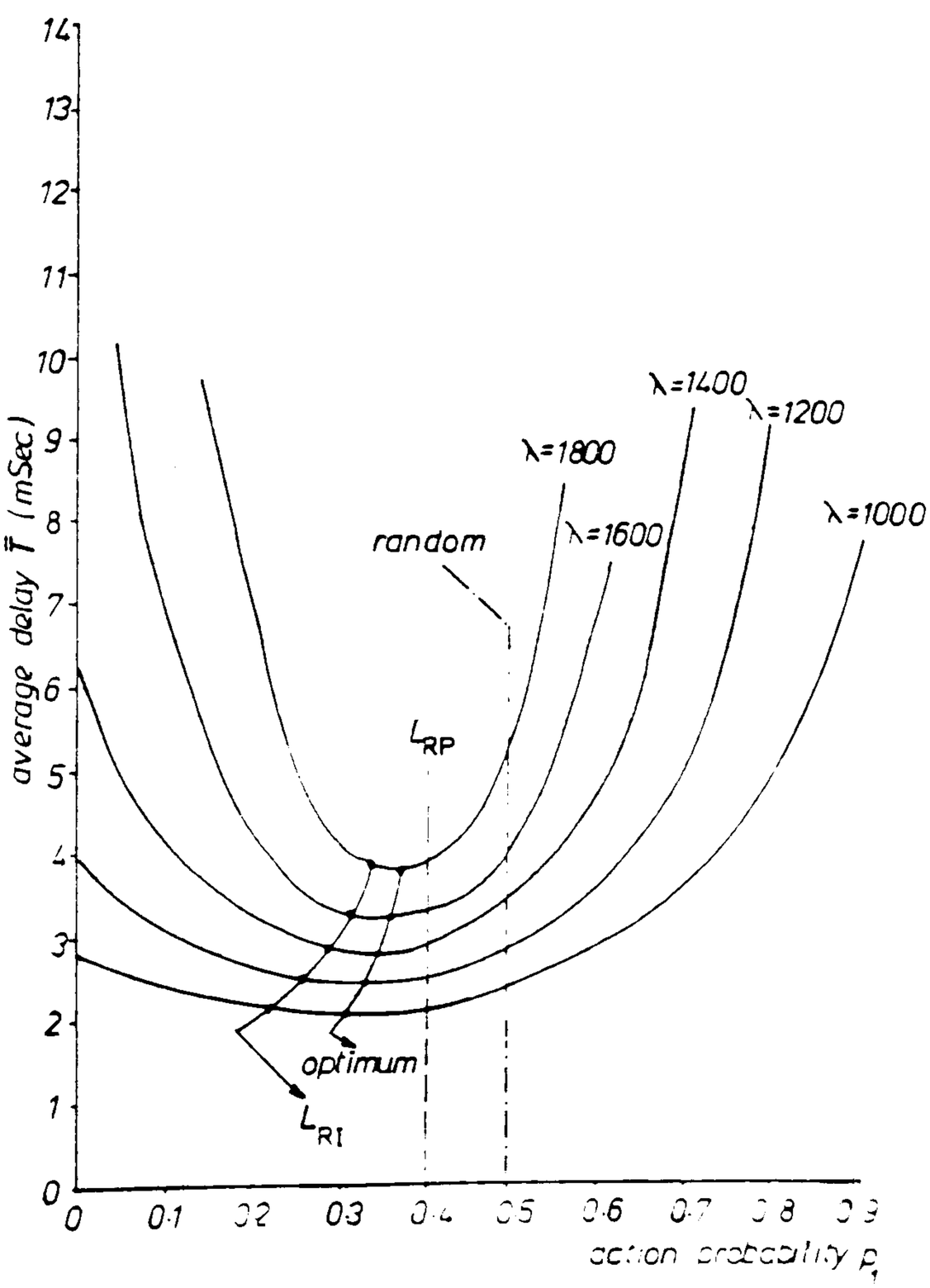


Figure 5 Conditions for range of traffic