NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

DIMENSIONING OF MESSAGE-SWITCHED COMPUTER-COMMUNICATIONS NETWORKS WITH END-TO-END WINDOW FLOW CONTROL

by

Jackson Y. K. Chan

Submitted to the School of Graduate Studies
in partial fulfillment of the requirements for
the degree of Master of Applied Science.

Department of Electrical Engineering
Faculty of Science and Engineering
University of Ottawa
Ottawa, Ontario

September 1979

TABLE OF CONTENTS

ABSTRACT

This thesis considers message-switched computer communications networks with end-to-end window flow control. Because of the excessive computational requirements in the exact analysis of the queueing models of such networks, the selection of good end-to-end window settings has been hitherto intractable. A new computationally-efficient heuristic algorithm, WINDIM, for the dimensioning of end-to-end windows, is presented. The criterion of performance is the ratio of the network throughput to the average network delay. Results obtained from the network examples considered may be readily extended to provide insights into the dimensioning problem for larger networks.

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to his supervisor, Dr. N.D. Georganas, for his patience, continued support and guidance throughout the preparation of this thesis.

Indebtedness is also due to Mr. R. LeHénaff and the other graduate students for their assistance and fruitful discussions.

Acknowledgement is expressed to the School of Graduate Studies and the Department of Electrical Engineering, University of Ottawa, for their financial support in the forms of Postgraduate Scholarships and assistantships.

Last but by no means least, the author is in debt to Annie for her kind assistance and moral support.

LIST OF FIGURES AND TABLES

CHAPTER 1

INTRODUCTION

Computer facilities are valuable processing tools. In a resource-sharing environment measures need to be taken to prevent any user from monopolizing these resources. Batch mode processing provides a solution, but here the user suffers from an irritatingly long response time. Moreover, should a given job's processing require only a part of the available system capacity, the excess could not be used for some other jobs. One alternative is to allow the user to access the computer(s) directly in a time-sharing mode. Modern technology has brought in larger and cheaper core memories, interrupt capability, memory protection and supervisory system programs in resource allocation and sharing. This unquestionably enables the multiaccess time-shared system to promise the many users it accommodates reasonable response-time characteristics, while maintaining high utilization for the available computer resources.

Computer networks find applications in practically

all walks of life, ranging from sophisticated military systems (Fig. 1.1) to the everyday use in airline reservations (Fig. 1.2) [1], banking, travel and stock market quotation systems.

In its more evolved form, a computer system may comprise many geographically distributed time-shared computer facilities interconnected in a computer-communication network so that resources of each could be shared by all [2]. The U.S. Advanced Research Projects Agency (ARPA) Network [3], provides an excellent example.

The typical computer communication network may assume a multilevel hierarchical structure as shown in Fig. 1.3. A set of inhomogeneous, multiaccess and time-shared computers called HOSTS provide local access for terminals which are directly available to users. Messages in the form of commands, inquiries and transactions, etc. flow between the hosts via the so-called communications subnet. This subnet essentially comprises a set of switching computers interconnected by communication channels.

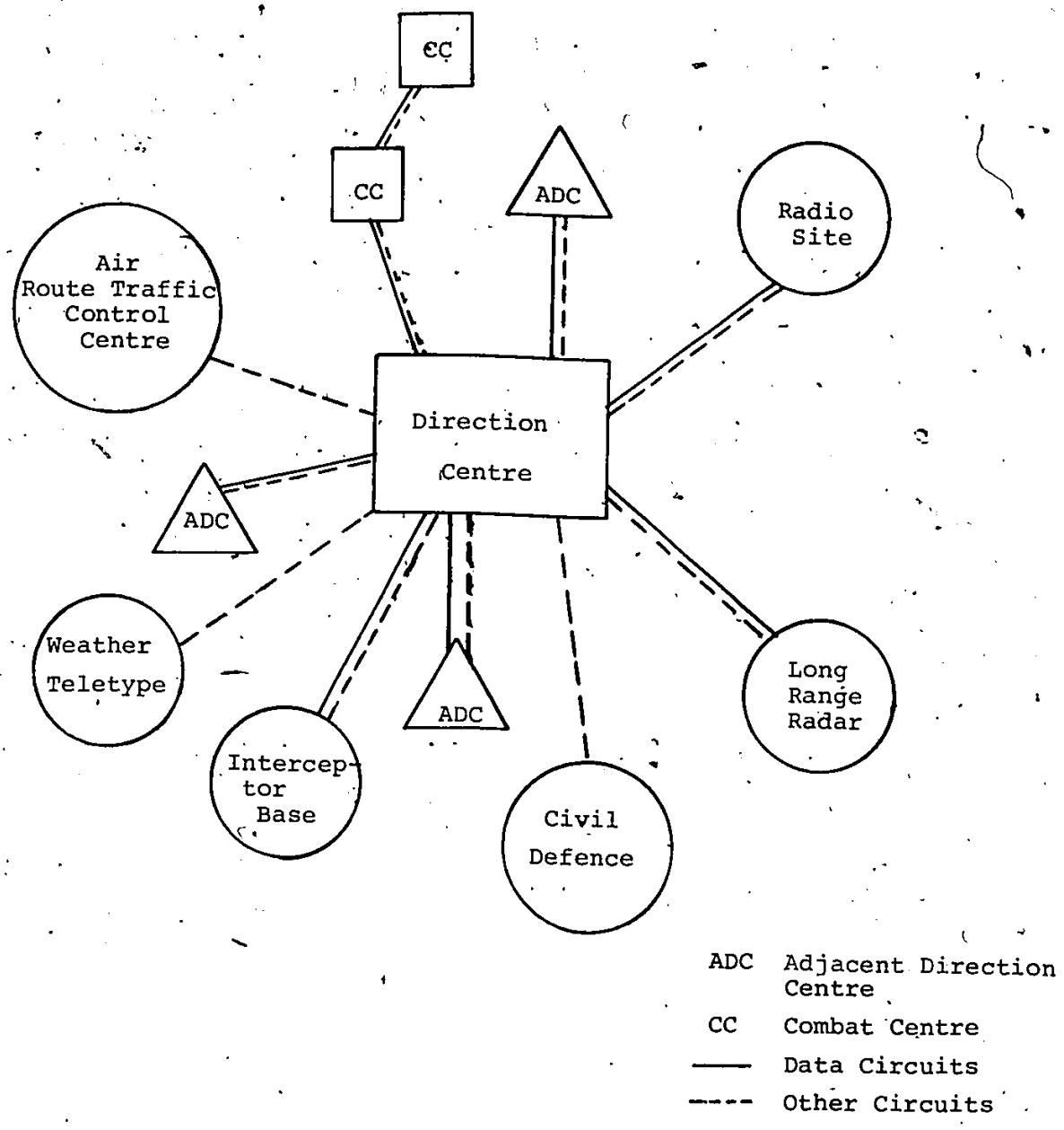


Fig. 1.1 Tactical Military System

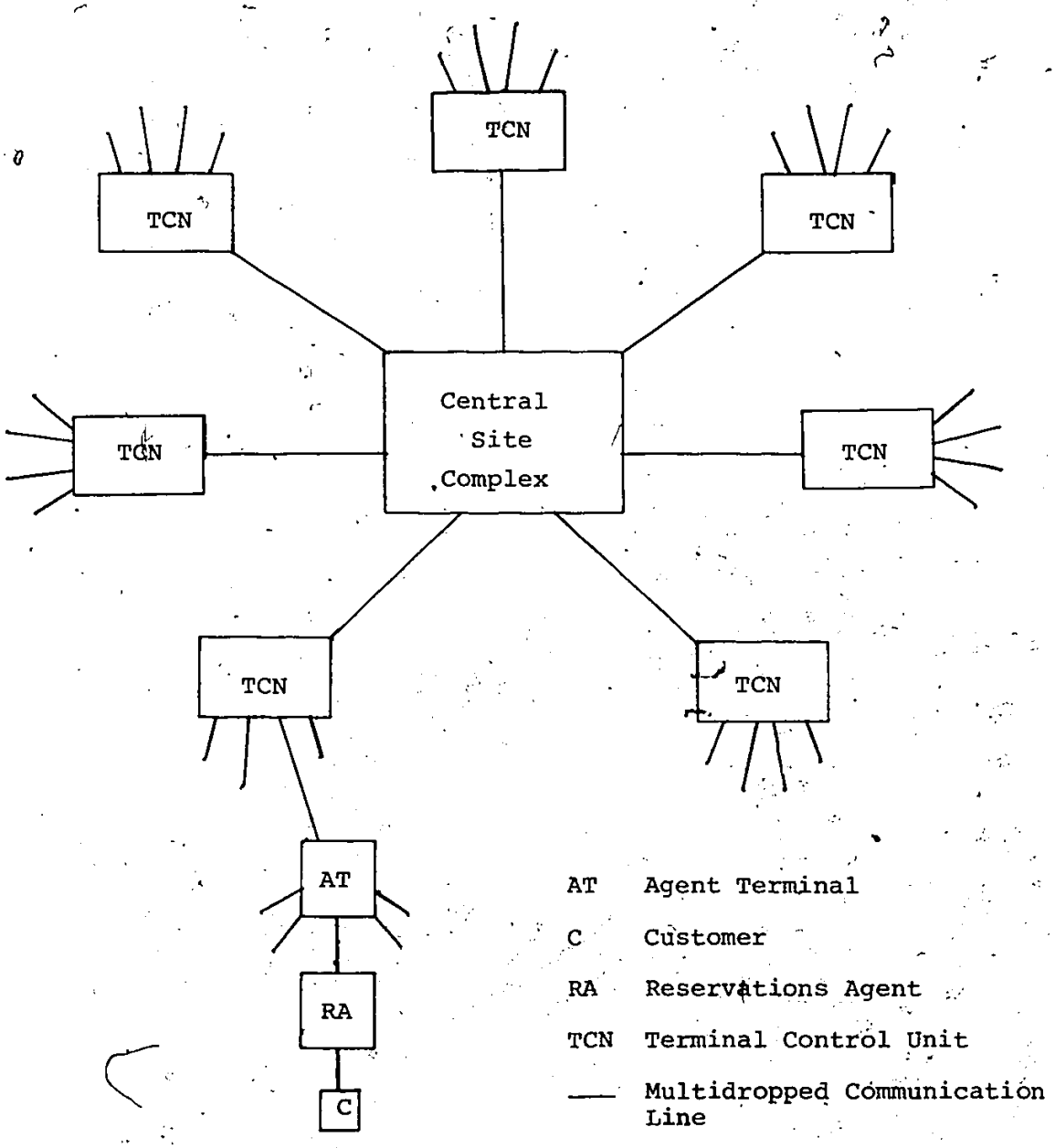


Fig. 1.2 Airlines Reservations System

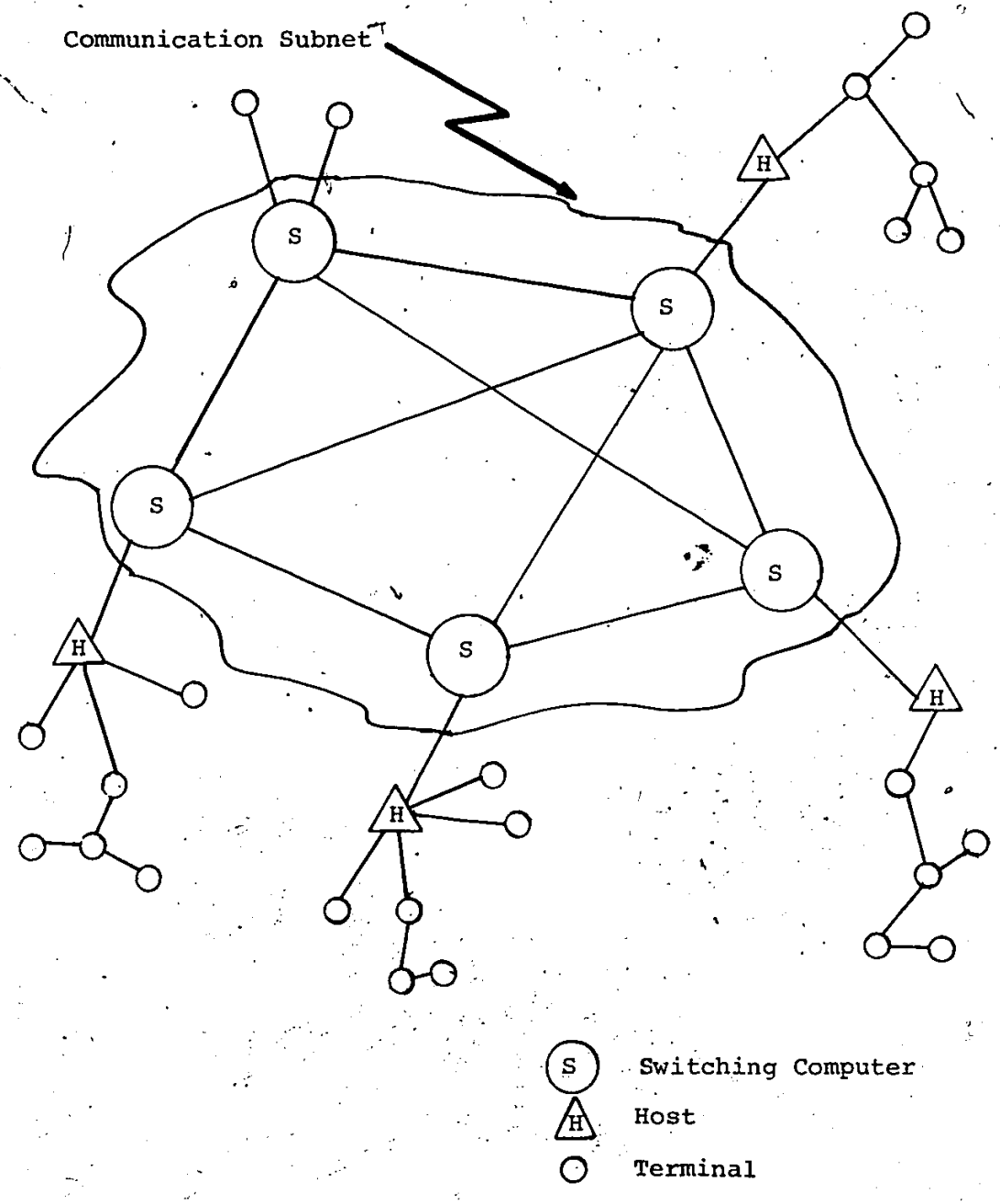


Fig. 1.3 Typical Computer Communication Network

With the wider use of digital transmission technology in the switched telephone network, one is tempted to think that circuit-switching can be employed in the communications subnet. However, circuit-switching requires that a complete physical path be established from the source to the destination that remains in effect for the duration of transmission. While a reasonable time frame can be equipped for continuous transmission, as in speech, the setup is inappropriate for computer traffic. This is because computer traffic is inherently bursty, that is the actual transmission capacity is required only intermittently with a small duty cycle. The statistical nature of computer traffic calls for lower setup time.

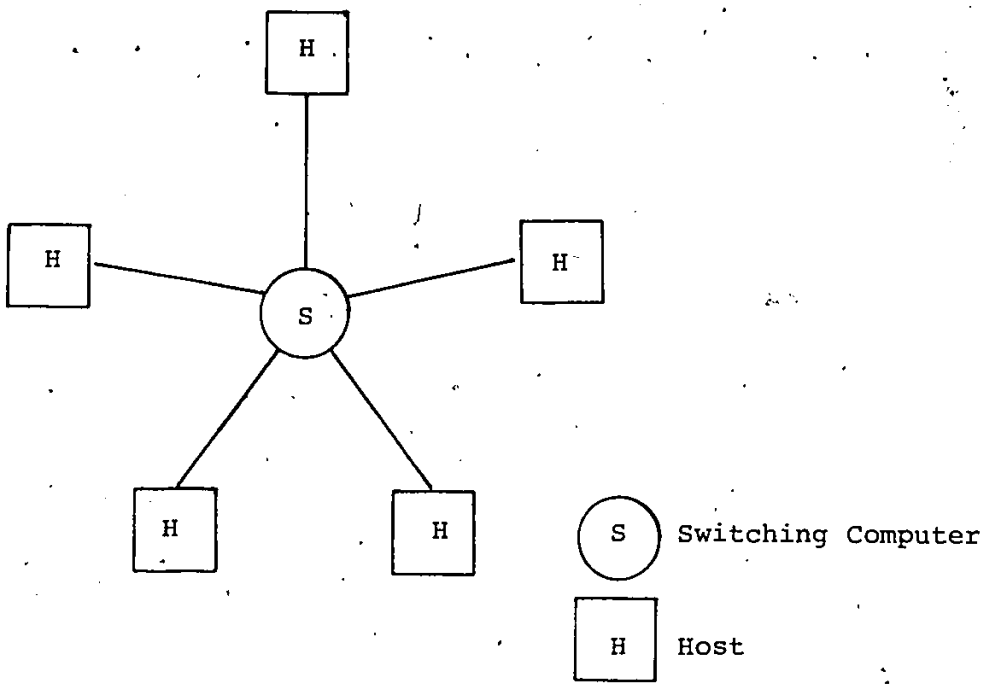
Define a link to be a unidirectional logical connexion between two nodes. Since computer traffic may be viewed as a string of statistical events consisting of bursts separated by idle time, input channels can dynamically share a link on a demand basis. This constitutes the so-called statistical multiplexing technique. Statistical multiplexing is superior to the conventionally used frequency-division and time-division multiplexing techniques in that resources can be more widely shared.

The tradeoff is that occasional transmission delays may be incurred. Message switching, as practised in the telegraph network, is a generalized form of such multiplexing in a network environment.

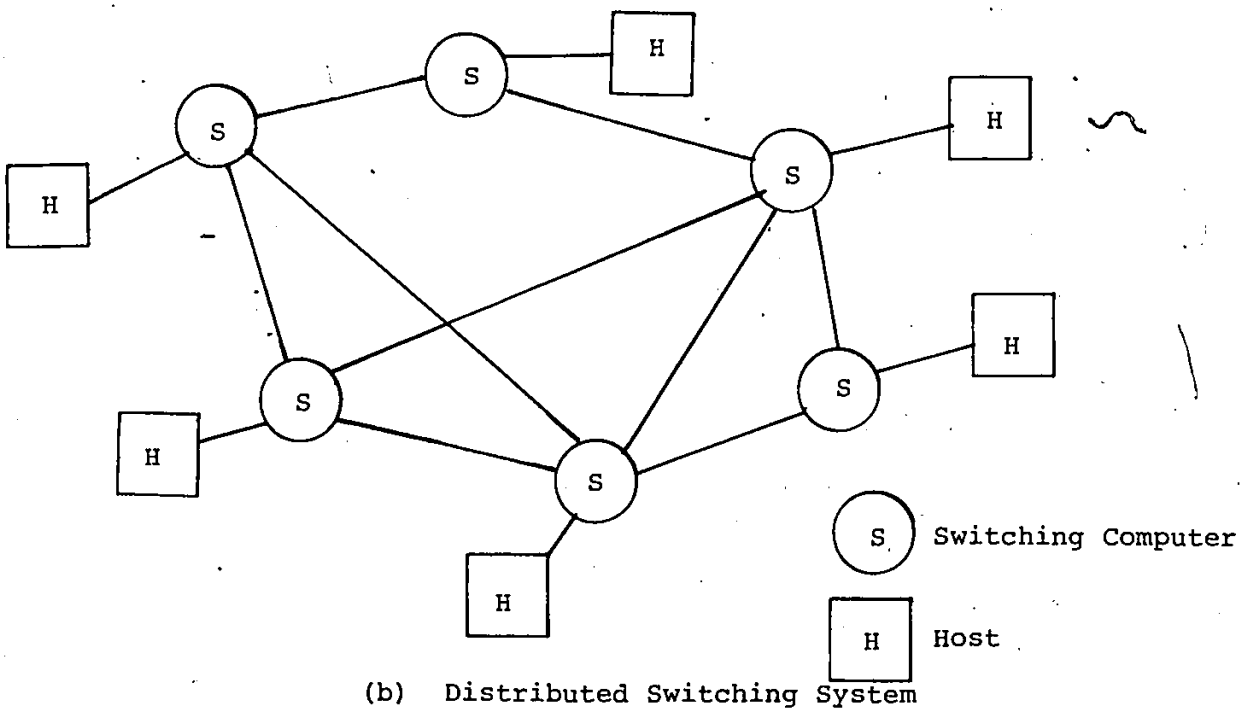
In message-switching, each message is furnished with a destination address by the source. This address is examined by each node through which the message passes. The node then guides the message to the best output line towards the destination, based on the current status of the network. Hence at any time, only one line is used for the transmission of a message. If the selected output line is busy, the message waits in a queue until the line is free and then transmitted. Messages are therefore successively stored and forwarded through the network, hopping from node to node. Messages may also be sent piecemeal. Such messages are decomposed into smaller units called packets, each of which has a maximum length. The packets are numbered and addressed and sent in a store-and-forward fashion as described before. The messages are then said to be packet-switched. Message reassembly is done at the destination node.

We now concentrate on the performance of the communications subnet in Fig. 1.3. In its most degenerate form, the communications subnet might simply consist of a single switch, as shown in Fig. 1.4(a), constituting a so-called centralized switching system. But such a system is highly unreliable because should the central switch fail to function, all communications between the hosts will be disrupted. Moreover, the communications cost is excessively high due to the large circuit distance to be traversed. In practice, the distributed switching configuration depicted in Fig. 1.4(b) is more commonly seen. Here, the switching computers perform the error and flow controls, routing, and some required speed and code conversion functions to ensure efficient transmission of messages. The communications lines thereby statistically multiplex messages from a large number of source-destination pairs of nodes.

The fact that the traffic rate is bursty creates a flow control problem which is very similar to the traffic control problem in a road network. Flow control ensures smooth traffic flow throughout the communications subnet by throttling the rate of traffic flow from the local access network (comprising the hosts and terminals) into



(a) Centralized Switching System



(b) Distributed Switching System

Fig. 1.4 Store-And-Forward Switching System

the communications subnet. If we define window size as the upper bound of unacknowledged messages (or packets) [4], then three classes of window flow control mechanisms can be identified according to the regions in which they are applied:

(a) End-to-end: for each source-destination pair.

(b) Local: for each switching node.

(c) Global: for the entire communications subnet.

This thesis addresses the problem of setting the end-to-end window sizes for a given distributed message-switched network. The performance criterion used is the ratio of network throughput to average network delay,called the network "power" [5].

In Chapter Two existing flow control mechanisms are reviewed with pertinent existing network examples.

Chapter Three deals with the exact analysis of the queueing model for computer communication networks with end-to-end flow control.

A new heuristic algorithm for the optimal setting of end-to-end flow control windows is developed in Chapter Four.

Chapter Five presents conclusions and recommendations for further study.

CHAPTER 2

FLOW CONTROL IN STORE-AND-FORWARD NETWORKS

2.1 INTRODUCTION

It is recalled that in a circuit-switched telephone network, each communication channel has constant bandwidth, and is open throughout the duration of transmission, so that flow control has little meaning.

The computer communication network, on the other hand, consists of heterogeneous computer resources which are widely dispersed. Access to these resources is largely asynchronous and in a highly bursty manner. The fundamental goal is of course the smooth flow of traffic with attendant efficient sharing of the resources.

In a store-and-forward (packet-switched or message-switched) communication network, a virtual connection is said to exist between two switching nodes if traffic flows from one node to the other. The traffic in each node is usually contributed by many users, each of whom has

relatively small demands (i.e. the ratio of their peak demands to the average demands is very high). Hence through the smoothing effect of a large population [2], the users collectively present a total demand profile which is relatively smooth and of medium to high utilization [6]. The network link capacity assignments are based on the nodes' total demand profiles in order that transmission delays can be acceptable. However, at times a certain user may have demands exceeding those assumed in designing the network, leading possibly to the contention of resources. If such contention is not properly controlled, the network throughput will be degraded drastically with each user suffering from an inordinately long time delay whereby congestion is manifested. Eventually a deadlock results in which communication becomes impossible. Physically, when the network is congested, some processes may be blocked, and traffic may be lost or held back owing to a lack of resources. Work being not conserved in either case, and so network throughput degrades [11]. The situation can be expressed diagrammatically as in Fig. 2.1. A network is said to be congested when it operates in the region of negative slope [7].

The congestion of store-and-forward networks has essentially brought two new problems into the foreground, those of routing and flow control.

Good control of the routing of traffic can increase the load the network will take. But when the limit is eventually reached, several links or nodes will tend to be blocked simultaneously [8]. Flow control sets out to optimize the throughput-delay performance, protecting resources from each other, and the network from being overloaded. Flow control effectively shifts the congestion from the interior of the network to the point of traffic admittance [4]. It anticipates and prevents congestion by regulating the entry of traffic into the network.

Flow control procedures fall under three categories:

(a) End-to-end

(b) Local

(c) Global

These procedures are considered separately in the following section.

2.2 FLOW CONTROL PROCEDURES

2.2.1 End-to-end Flow Control

Consider the flow of traffic from a switching node A, via nodes C and D, to switching node B, as shown in Fig. 2.2. A and B are respectively called the source and the destination (or sink) nodes. The unidirectional logical connexion between A and B is called a virtual channel. Traffic flows in a store-and-forward fashion, from node to node, following the virtual channel. And, unlike the case of circuit switching, transmission on any link along the virtual channel is shared with a number of other users [9].

To avoid overloading the device at B, and hence network congestion, the source input rate at A has to be synchronized to the destination acceptance rate at B. This is achieved by imposing a limit on the number of messages (in the case of message-switching networks) or packets (in the case of packet-switching networks) allowed in the virtual channel. The traffic rate on the virtual channel is therefore gradually decreased when the network becomes congested and the end-to-end transmission delay climbs up [12]. This control scheme is termed end-to-end

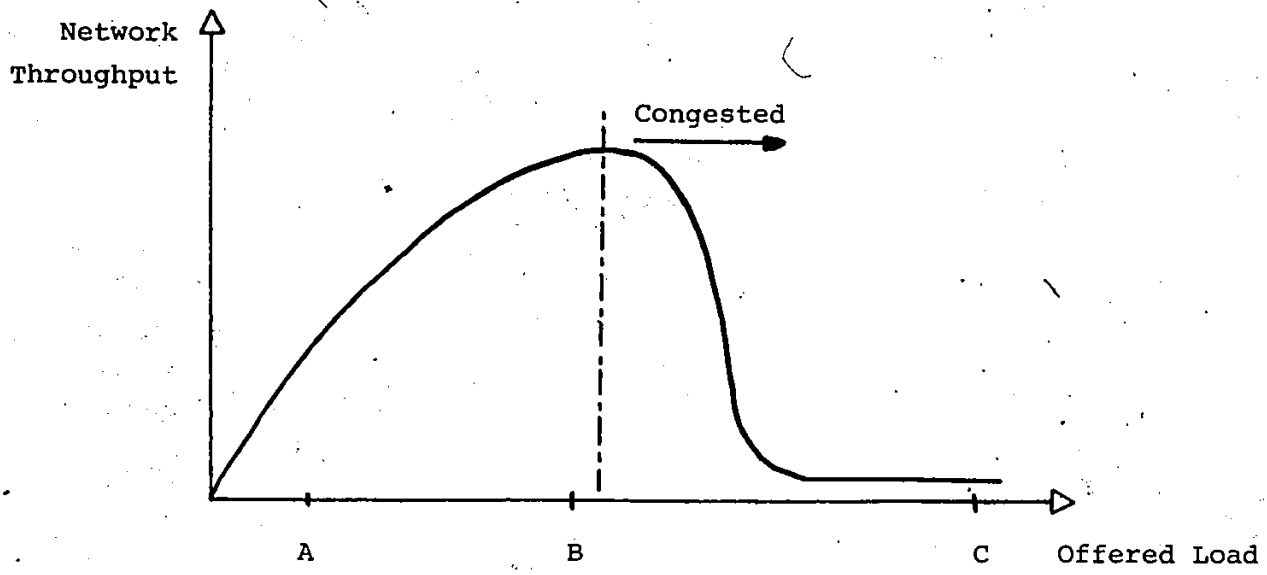


Fig. 2.1 Network Throughput Versus Offered Load

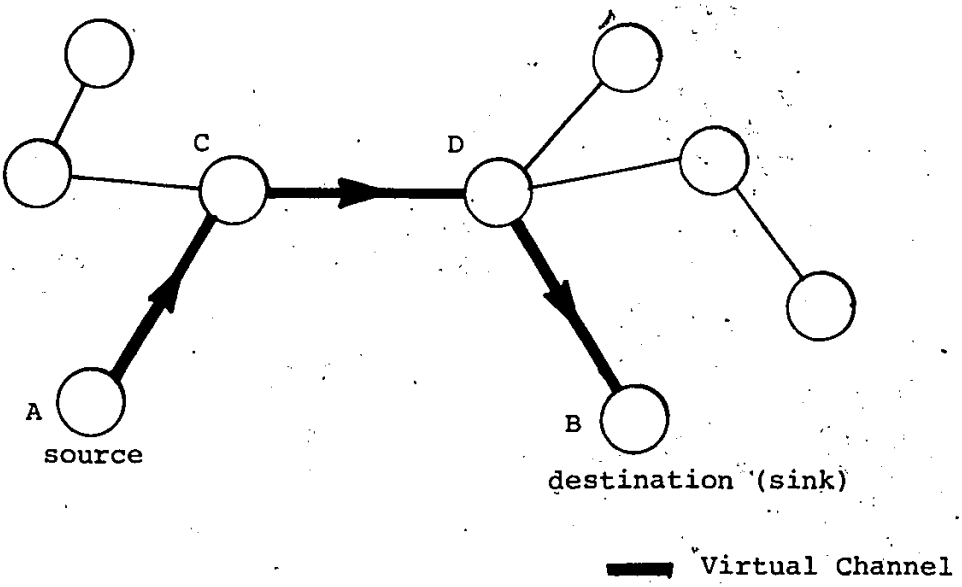


Fig. 2.2 A Virtual Channel

flow control.

An example is the source-to-destination flow control scheme of the ARPA network shown in Fig. 2.3. The communications function is carried out by nodal minicomputer communication processors called Interface Message Processors (IMPs). Terminals are connected to their HOST computers which, in turn, are connected to the IMP's. Terminals may also be connected directly to the network via nodal concentrators called Terminal Interface Message Processors (TIPs). The IMPs and TIPs comprising the communications subnet are interconnected with wideband communication lines or links primarily of 50 kbps capacity.

The ARPA Network accepts both single-packet and multipacket messages. A message is considered as a multipacket message if the HOST-IMP interface has not received an end-of-message after the input of the first packet is completed. A full multipacket message comprises eight packets.

(a) Transmission of Multipacket Message [3,10]

As soon as the source IMP receives the first packet

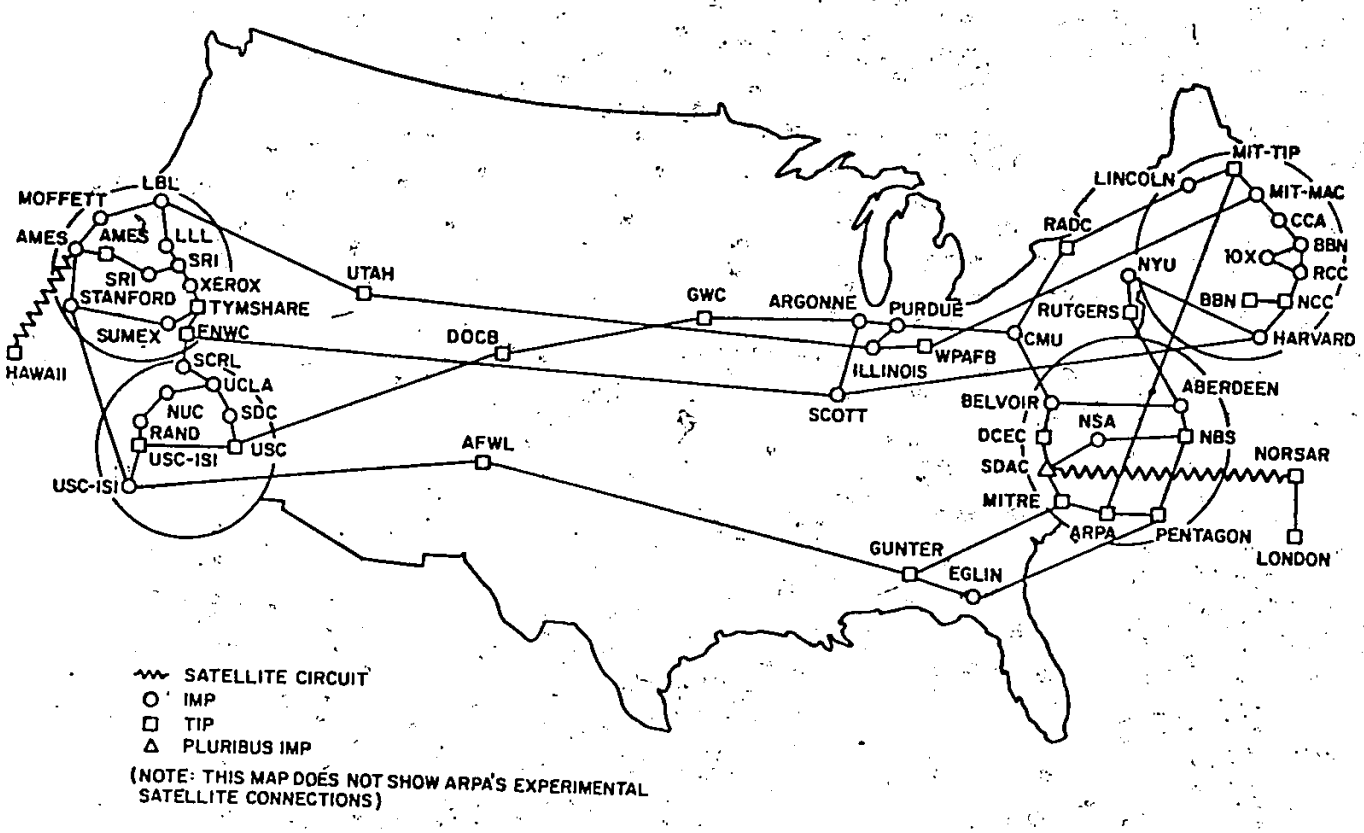


Fig. 2.3 The ARPA Network, Feb. 1976

of a multipacket message from the source HOST, it sends a small control message to the destination IMP requesting that eight reassembly buffers be reserved. If no buffer is available, the request is repeated until successful reservation can be made. Meanwhile, it does not take in further packets from the HOST. If buffers are available, the destination IMP confirms the reservation and notifies the source IMP. The source IMP then starts message transmission. After all the packets in the message have been fully assembled, the destination IMP delivers the message to the destination HOST. To reduce the request/allocate overhead for long sequences of messages, only the first message in a sequence of multipacket messages go through the reservation procedure. From that point on, an acknowledgement called RFNM (Request For Next Message) is sent back to the source IMP after each successful delivery and reassembly of message. The reservation for the eight reassembly buffers is held for the message sequence as long as more data is delivered to the source IMP within 250 ms after the RFNM is received.

(b) Transmission of Single-Packet Messages [3,10]

Here no buffer reservation at the destination IMP

is needed. Single-packet messages are transmitted immediately after a source IMP receives them while a copy is kept at the source. If there is space at the destination, the packet is accepted and passed on to the HOST and a RFNM is received by the source IMP. If no buffer is available at the destination IMP, the message is dropped and a request for buffer allocation is queued. When space becomes available the source IMP is so informed and the stored message may then be retransmitted.

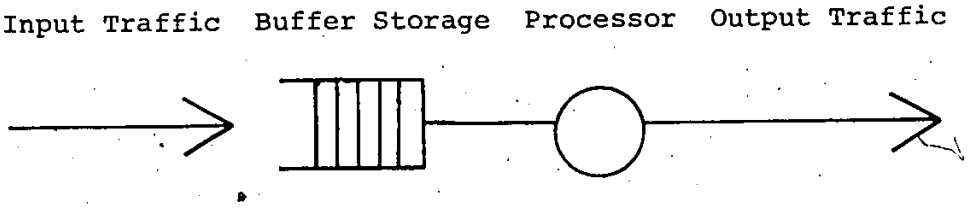
The ARPA Network flow control is such that there can only be a maximum number of four messages outstanding between any pair of source-destination IMPs. Hence, whenever four RFNMs are outstanding, further input from any of the source HOSTs is stopped.

2.2.2. Local Flow Control

Local flow control prevents store-and-forward buffer congestion at a switching node by regulating (and possibly discarding) inputs from adjacent nodes [12]. In contrast to the end-to-end flow control, restriction is placed on the messages (or packets) once they are inside the network.

In a store-and-forward network of N switching nodes, node i, i=1,2,...,N may be modelled as shown in Fig. 2.4. Because of the limited buffer storage space, no more than, say K_i messages (or packets) are permitted to be at node i at one time. As soon as the limit K_i is exceeded, the node blocks inputs from adjacent nodes. The blocked messages (or packets) remain at their current nodes, until node i is again free to admit them. This blocking could reach all the way back to the source node(s) of one or more virtual channels, whereby incoming traffic to the network is blocked.

The TYMNET network affords an example of local flow control [13]. This network, shown in Fig. 2.5, has been designed to accommodate input-output devices, e.g. teletypewriters, push-button control units, and their associated display devices, where they exist. Here flow control is provided in the form of a computer-to-user direction shut-off feature that prevents traffic from building up. The computer output rate may well be 1000 characters per second, yet the terminal to which it is directed may only be able to print ten characters per second. Hence it is imperative that the output flow from



K_i = maximum number of messages (packets) allowed.

Fig. 2.4 Model of Switching Node i

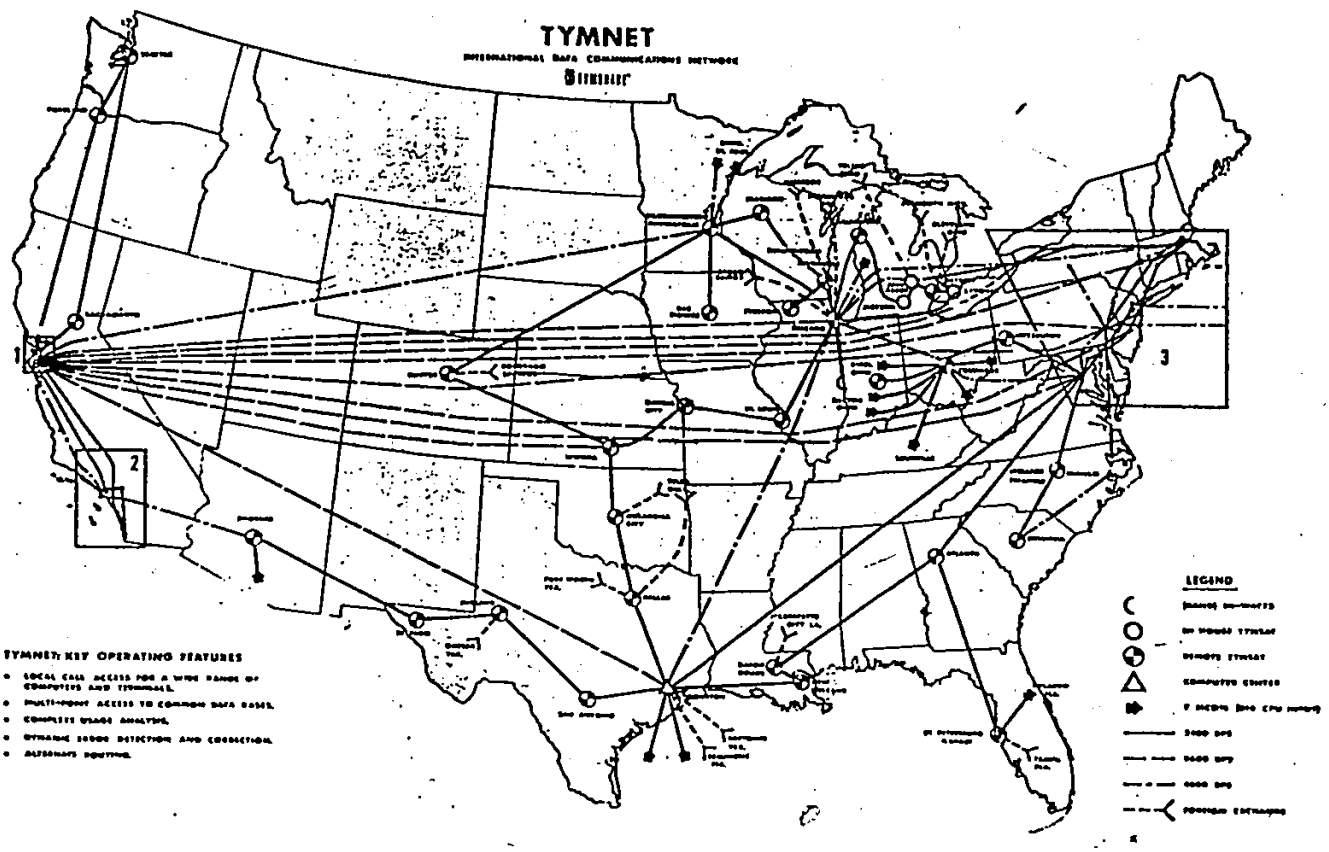


Fig. 2.5 The TYMNET Network

computers be buffered. It is also noted that intermediate node buffers are also limited. Hence, to prevent data from piling up on any channel, a given sending node subtracts the number of characters sent out over the channel from a counter associated with the channel. When the counter reaches zero no more characters are sent over that channel. The receiving node sends back one bit twice a second to the sending node. The bit indicates whether the receiving node has less than or more than 32 characters in the receiving buffer for the channel. If there are less than 32 characters in the receiving buffer the sending node resets its counter to 32 and starts transmitting again.

2.2.3. Global Flow Control

In the two preceding flow control procedures, messages (or packets) are distinguished either by the virtual channels on which they flow or by the switching nodes to which they are inputted.

Control may equally well be placed on all messages (or packets) independent of their identity, constituting what is called the global flow control. This control

procedure imposes a limit on the total number of messages (or packets) that can circulate throughout a store-and-forward network. If this limit is reached, all traffic seeking admission to the network will be blocked by the source nodes. The network resumes admission of incoming traffic only when the circulating traffic level falls below the imposed limit.

There are at least two techniques for implementing the global flow control:

(a) Loop Method: This has been used on some proposed loop systems. A fixed-length frame circulates around a loop of control points. When the frame reaches a control point, messages waiting in buffers at the control point is read out onto the line constituting the loop, provided that the frame is not fully occupied [14,15].

(b) Isarithmic Method: This method can be used for more complex distributed networks. A fixed number of permits is circulated in the network. A packet enters the network only after it has acquired one of the permits. The permits may circulate randomly over the network, to

be picked up by the first node it reaches at which a packet is waiting. Alternatively, each node may maintain a small store of permits, to be readily accessed by arriving packets, thus reducing the corresponding admission delay [8]. A combination of the two permit allocation schemes is also possible. In fact, in one of the simulation studies done by the National Physical Laboratory of Great Britain [8], it has been shown that, for the 18-node network under study, the admission delay was negligible if about half the permits were free to circulate, and the other half assigned to pools at specific nodes.

2.3 Comments

Consider a store-and-forward network with R virtual channels. Define $E_{max} = \sum_{r=1}^{R} E_r$, where E_r is the end-to-end flow control limit imposed on the r^{th} virtual channel.

Because of the bursty nature of the incoming traffic, the average network utilization is very low. In other words, it is quite unlikely that the total network traffic can ever reach E_{max}. This seems to suggest that the network can operate on the principle of overcommitment [7]. That is, E_{max} may be made as large as desired, and a global network flow control constraint I, $I < E_{max}$ may be imposed

to guard against network congestion due to statistical fluctuations.

However, the finite buffer storage capacity of the switching nodes must not be overlooked. In fact, if E_r were allowed to become so large that it exceeds the storage capacity K_i of node i along the r^{th} virtual channel, a large amount of traffic may at times converge on one place. This overloads the node without violating the end-to-end flow control rules, rendering the control totally ineffective.

Nevertheless, it is inadequate to have only local flow control. This is because if the users who chiefly contribute to an overload are distant from the point of congestion, local control methods require congestion control measures to spread a long way before effective measures are taken [8]. Nor can local flow control be dispensed with by increasing indefinitely the nodal storage capacity, for the store-and-forward delays will then be increased proportionally [16].

Hence apparently all three of the flow control

schemes have their respective significance in preventing network congestion. Judicious choice of the flow control limits is an expedience. Unfortunately, this choice depends very much on the network topology and operating conditions, so that each network case needs to be separately scrutinized.

CHAPTER 3

COMPUTER-COMMUNICATION NETWORK QUEUEING MODELS

EXACT METHODS OF SOLUTION

3.1 INTRODUCTION

For the systems designer, it is almost impossible to overemphasize the importance of monitoring the performance of a system. The system may be characterized by some given resource parameters and subject to certain system workloads. This is particularly true in the cost-effective design of computer-communication systems, which are ever growing in complexity and sophistication.

In a computer-communication network, the backbone switching computers, constituting the highest level of the network, support efficient and flexible sharing of geographically separated host computers. Critical resources in the design and operation of the network therefore include the storage capacities of the individual switches and the communication links connected to them.

Because of the stochastic nature of work demand by the messages or processes, queues may be formed and delays incurred at the critical resources. Performance prediction includes such important measures as response time, throughput and resource utilization. Noteworthy is the accomplishment of Kleinrock [17] and others in formulating the problem of link capacity assignment as an optimization or mathematical programming problem based on the average delay formula derived from queueing analysis. Effective scheduling and allocation of resources among competing requests is certainly the realm of congestion or queueing theory in a broader sense [18]. Hence queueing network models provide a basic framework and the mathematical tools for the quantitative assessment of system performance.

Taking into consideration the substantial amount of modelling efforts in and high cost of simulation runs, analytic solutions should be sought, whenever possible. Even though one may have to resort to simulation techniques ultimately, an analytic model, however crude it may be, can help alleviate the modelling efforts by narrowing down the range of system configurations and parameters

under which a simulation runs. Moreover, it may also detect possible errors introduced in the design and implementation phases of a simulator [18].

It is therefore the objective of this chapter to explore the analytic models for a general class of queueing networks. The methods of solution of such models will be seen to lend themselves to the analysis of the store-and-forward communications subnet in any computer-communication network with end-to-end flow control.

3.2 A CLASS OF STOCHASTIC QUEUEING MODELS: CONCEPTS AND DEFINITIONS

3.2.1 Introduction

Stochastic models [19] may be of overwhelming complexity. For example, a general stochastic model driven by dependent processes and controlled by a centralized algorithm which has access to the full system state at all times. The ensuing discussion will be confined to a less complex class of stochastic models, Q, in which

each model is structured as a network of locally scheduled queues and driven by renewal processes. The flow of jobs is governed by a sequence of routing decisions made upon service completion at one of the queues. It will be seen that the class of separable queueing networks is a subset of Q.

Consider a store-and-forward network of message switches. This may be, for example, the communications subnet of a typical computer-communication network. Define the unit of network traffic flow, which in this case is a message, as customer.

A queueing model of this network may comprise:

(a) Service stations providing service for customers in queue.

(b) A set of flow control protocols regulating the population of customers.

(c) A population of customers with concomitant work demands placed upon the service stations.

(d) A routing algorithm governing the flow of customers through the network.

(e) Source nodes through which customers enter the network.

(f) Sink nodes through which customers leave the network.

3.2.2 Customer Arrivals

A source generates customers according to a renewal process specified by the interarrival time distribution [19]. Successive work demands associated with individual customers are independent and identically distributed random variables.

Define a queueing network to be open if a customer is drawn from an infinite population or input service and eventually leaves the network after being served. In this network the number of potential customers remaining in the input source is independent of the number of customers already in the queueing system [18]. Fig. 3.1 shows an example of an open queueing network. In case the source input arrival rate is significantly affected by the number of customers in the queueing network, a finite population source may result. This is conveniently

modelled as a closed queueing network, as exemplified by Fig. 3.2.

An exogenous arrival of customers is described by a Poisson process. This description is adequate as long as the number of the component processes, assumed to be statistically independent, is large and the contribution of each to the aggregate arrival rate is small. The bursty traffic in a large computer-communication network naturally fits into this delineation. The customer interarrival times are exponentially distributed. Indeed the Poisson process is simply the counting process constructed from the renewal process of exponentially distributed lifetimes [18].

3.2.3 Routing

The impact of routing on the class Q of queueing network is lucidly depicted by a directed graph, as shown in Fig. 3.1 or Fig. 3.2. Two nodes are connected by an edge should a routing transition be feasible between them.

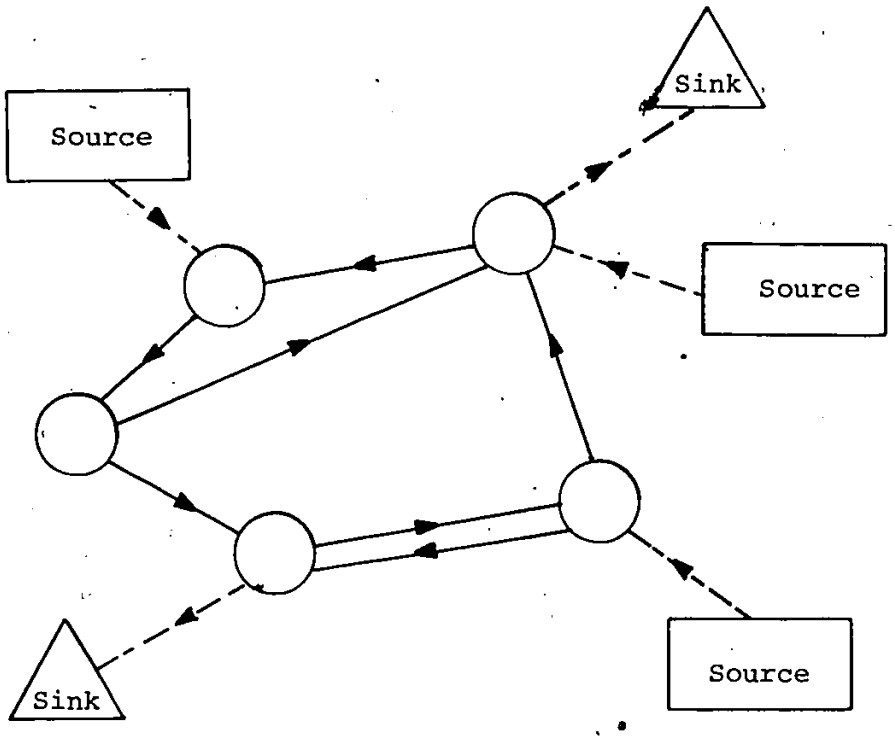


Fig. 3.1 Example of an Open Queueing Network

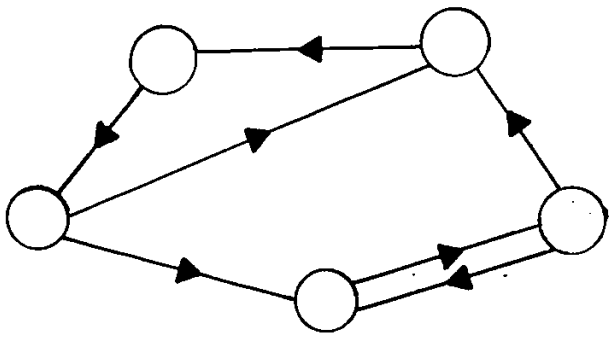


Fig. 3.2 Example of a Closed Queueing Network

The routing may define equivalence classes of nodes. Each class is characterized by a source node and a destination node which may be connected by a sequence of routing transitions. Such equivalence classes are also called routing chains. An example of a multiclass queueing network is shown in Fig. 3.3.

3.2.4 Service Mechanism

Each service station (fig. 3.4) is composed of a buffer storage space to accommodate customers waiting for service, a set of servers and a queue discipline to govern the manner in which customers are to be served.

Consider a subset of Q which comprises networks with rational service time distributions (those with rational Laplace transforms). Such distributions can be represented by a set of exponential servers (or stages) combined in series and parallel manner [20], as shown in Fig. 3.5. Customers enter at point A. A customer will enter the i^{th} stage with probability a_i or depart from the service station with probability b_i. Evidently it

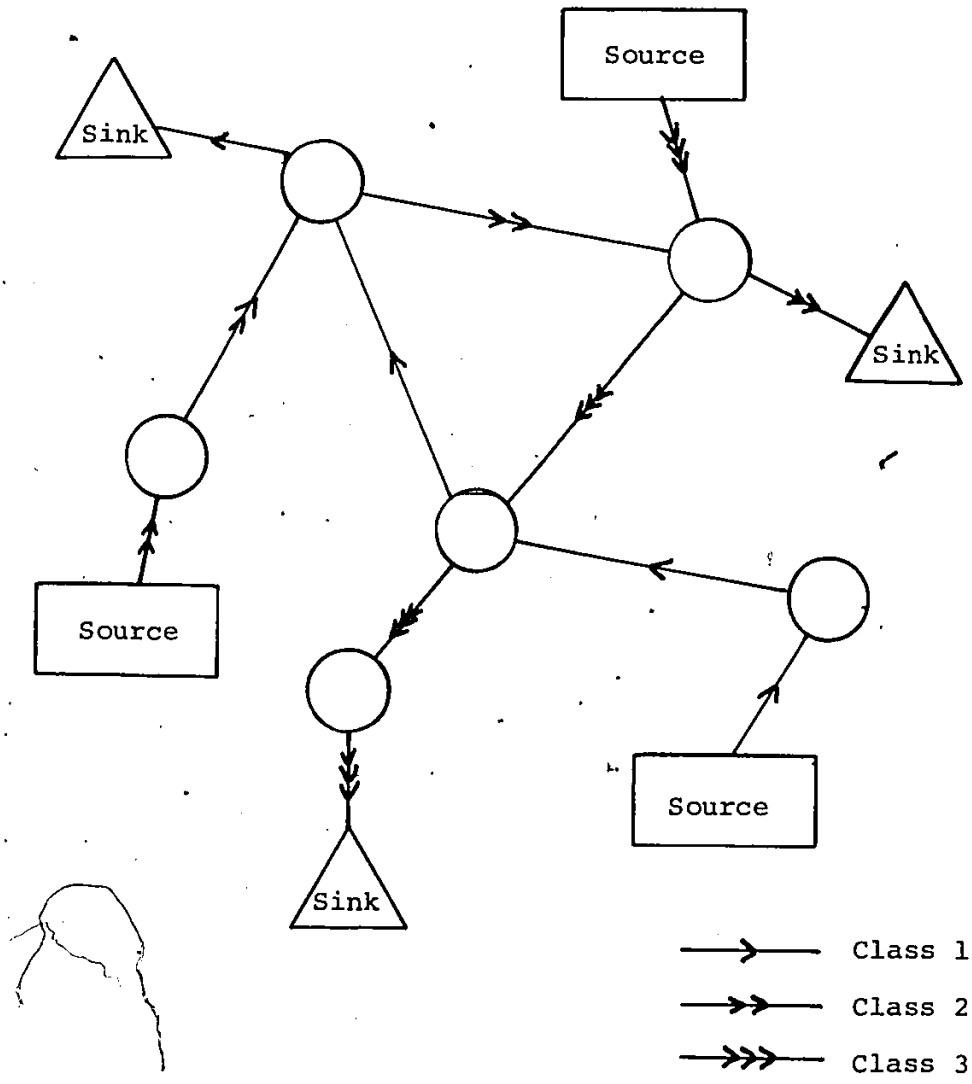


Fig. 3.3 A 3-Class Queueing Network

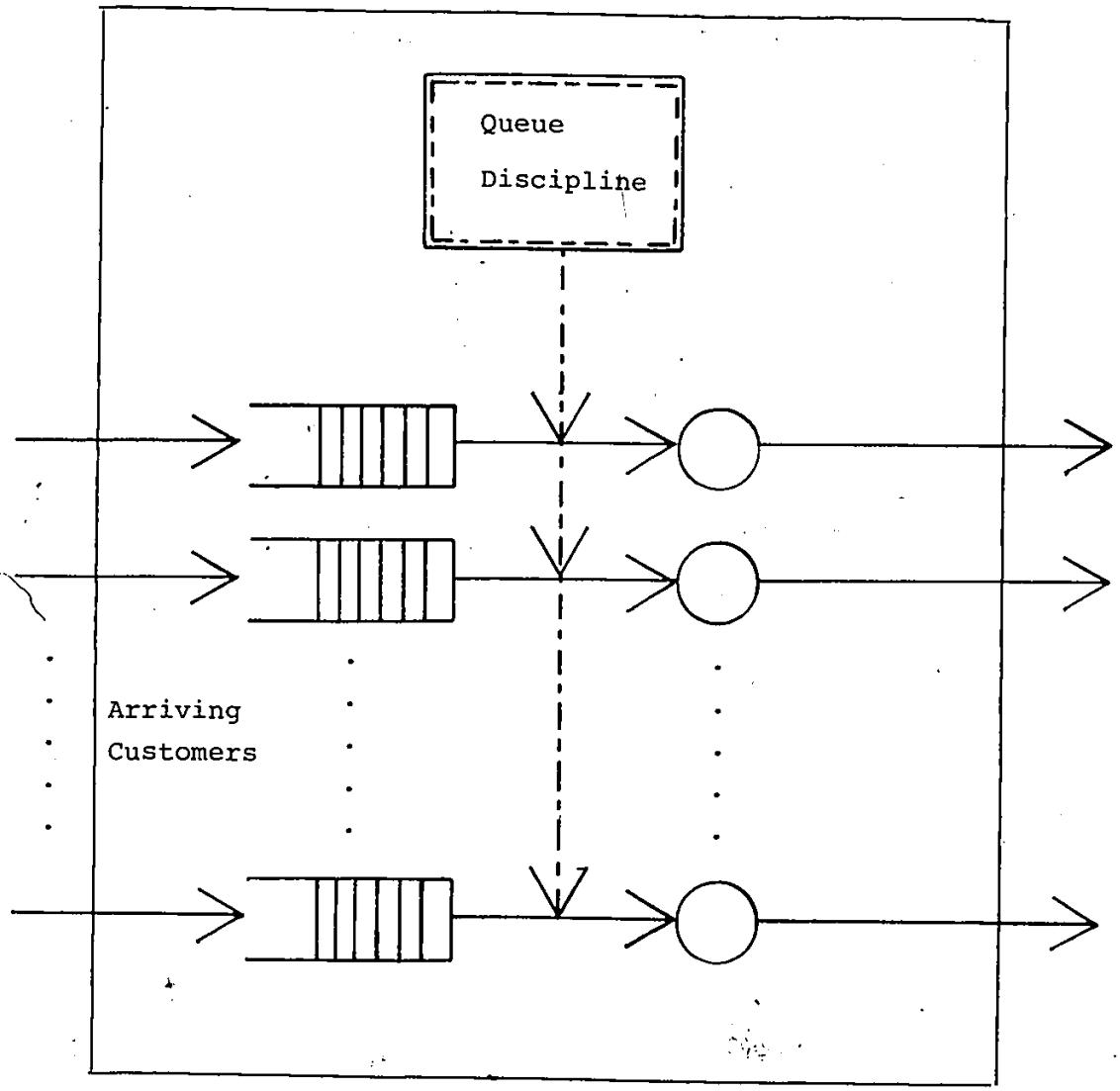


Fig. 3.4 Schematic Representation of a Service Station

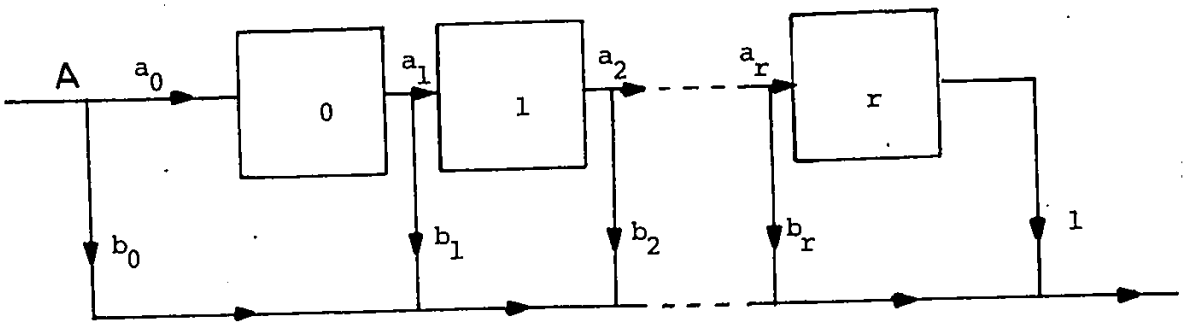


Fig. 3.5 Stage-type Representation of Service Station

is required that $a_i + b_i = 1$ for $i=0,1,2,\ldots,r$. The customer leaves the service station after completing the r^{th} stage with probability 1. It should be emphasized that for such a stage-type server, only one stage can accommodate a customer at a given time, the entrance stage is blocked as long as a job exists in some stage [20].

A queue discipline is said to be work-conserving [21], if:

(a) The work demand of each customer is independent of the queue discipline.

(b) The queue discipline does not take advantage of knowledge about work demands (or times) and/or arrival times of the individual customers.

(c) The service station is not idle when there exist customers awaiting service.

Examples of work-conserving queue disciplines include FCFS (First-Come-First-Served), PS (Processor-Sharing), LCFSPR (Last-Come-First-Served Preemptive Resume), and IS (Infinite Server) queues.

In an FCFS service station, all customers, irrespective of classes, have the same exponential service time distribution. The service rate $\mu(j)$ has been stipulated to depend on the number j of customers at the service station. This is to ensure that blocking, which drastically increases the complexity of the analysis, needs not be considered [24]. In PS, IS or LCFS service stations, each class of customers may have its own general service time distribution which has a rational Laplace transform [27].

3.2.5 Network Stability

A queueing network is said to be stable if a unique steady state solution exists. A necessary condition for stability is $\sum_{j=1}^{M} p_{ij} = 1$, for all i=1,2,...,M where p_{ij} is the probability for a customer to join node j after service completion at node i, and M is the total number of nodes that spans the network. The network is unstable if there exists at least one queue in it with external arrivals and with work intensity exceeding unity [19].

3.3 SEPARABLE QUEUEING NETWORKS

3.3.1 Introduction

If the work demand distributions and inter-arrival time distributions are restricted to those whose Laplace transforms are rational functions, then the distributions can be represented by a set of exponential stages combined in series and parallel manner. The class of networks having these attributes is called Markovian queueing networks, also known as memoryless queueing networks.

By introducing an appropriate state space, the system behaviour can be treated as a birth-and-death process [24]. Define a balance equation as a difference equation that relates the steady-state probability of a given state with the probabilities of the adjacent states. The equilibrium (or steady-state) joint queue size distribution is then determined by solving a set of balance equations for the system. That is,

$$\forall \text{ states } S_i ,$$

$$\sum_{\substack{\text{all states} \\ S_j}} P(S_j) \text{ [rate of flow from } S_j \text{ to } S_i\text{]} = P(S_i) \text{ [rate of flow out of } S_i\text{]}$$

However, a numerical solution of the balance equations is impossible for all but the most simple models [18]. Some works on the assemblage of the balance equations and on their solution by iterative methods are found in [28-30].

The queueing network problem can alternatively be solved by approximating the original problem by solutions which are in some subset Q* of the class Q of queueing networks, possibly through the numerical evaluation of certain reduced systems of balance equations [31]. Approximation methods such as diffusion approximation [32-33] or forced decomposition approach [31] can be used. However, for both methods general theoretical foundations are not fully established. Moreover, the diffusion techniques seem to work well only for high traffic intensities.

A more practical approach is to consider some subset Q* of Q, say, such that in Q* the need for a numerical solution of the balance equations can be dispensed with.

The most general class of queueing networks in Q* [34] is described as follows:

(a) There are multiple classes of customers.

(b) The routing is a Markov chain of arbitrary order [18] over the set of classes. This Markov chain is decomposable into subchains.

(c) Subchains may be open or closed. A network may comprise open and closed chains.

(d) Open chains are driven by independent Poisson streams which may have population dependent rates.

(e) Queues have work-conserving queue disciplines (as described earlier) [35].

(f) All jobs have the same mean service time with respect to a given queue.

(g) The service time distribution may be exponential, as is in the case of FCFS queues, whereby all customers share a common distribution with queue-dependent service rates. Alternatively the queue discipline may allow class-dependent service times drawn from phase-type, general service time distributions. Examples of this are LCFS, PS and IS queues.

For any queueing network in this class Q*, the balance equations can be solved by considering merely locally balanced networks [36] through separation of variable techniques. The network solution is of product form. This class Q* of network is now formally termed separable queueing networks.

Substantial progress has been made in exploring the solution of separable queueing networks. Jackson [37] pioneered in showing that the equilibrium network state probability distribution had a product form for open queueing networks with exponential servers. Gordon and Newell [38] later showed that closed networks with queue-dependent service rates and exponential service times also had product-form solution. Baskett [39] showed that central-server models with a PS CPU and exponential inputs and outputs had product form provided the CPU service time distribution had a rational Laplace transform [20]. He also showed that the steady-state behaviour of such models only depend on mean CPU service time and not on higher order moments.

Baskett's results were extended by Chandy [31] to arbitrary networks by defining and using local balance for queueing networks. Chandy further showed that networks with PS and LCFSPR queue disciplines had product form and satisfied local balance. Muntz [40] investigated the "M→M" property, that is "Poisson arrival implies Poisson departure" property. He showed that a queueing network with this property had product form. Moore [41] introduced the use of generating functions for the treatment of network queueing models. Buzen [25] developed computationally efficient procedures for determining the normalization constant, steady-state probabilities, throughputs and queue length distributions for single class locally balanced networks.

Baskett, Chandy, Muntz and Palacios [36] extended the class of queueing networks with product form solution to include different classes of customers and general service time distributions for certain service disciplines. In this model, open, closed and mixed networks were considered and customers traversing the network might change class membership while making a transition from one service station to another.

Reiser and Kobayashi [24] developed computationally efficient procedures for local balance networks with several classes of customers where some classes may be open and others closed. Chandy, Howard and Towsley [48] introduced a property of queueing disciplines, called "station balance". This explained why queues with FCFS cannot have product form if service times are nonexponential for all classes. It also explained why some disciplines result in local balance for nonexponential service distributions while other disciplines do not.

Lam [27] enlarged the known class of queueing networks with a product-form solution to include mechanisms of state-dependent lost arrivals and triggered arrivals. He found a sufficient condition to guarantee product form solution for networks with such arrival mechanisms. Recently, a number of efficient computational algorithms [24,42,49,44,45] have been developed for evaluating the normalization constant and marginal queue length statistics for queueing networks with product form solution.

3.3.2 Open Queueing Networks

Consider an M/M/1 queueing system, then a Poisson process driving the exponential server generates a Poisson process for departure. In fact, Burke [46] has established that

(i) the steady-state output of a stable M/M/m queue with input parameter λ and service time parameter μ for each of the m channels is a Poisson process at the same rate λ.

(ii) the output process is independent of the order processes in the system.

These results are generally known as Burke's theorem. Indeed, it follows from Burke's theorem that many multiple-server nodes (each server with exponential pdf) can be connected together in a feedforward (i.e. customers may not return to previously visited nodes) network fashion and the node-by-node decomposition is still preserved. The conclusion remains valid for work-conserving queue disciplines. It should be mentioned that the input process need not be of Poisson distribution, but any arbitrary initial distribution, and the output process will

converge in law as $t \to \infty$ to the Poisson distribution [34].

Jackson generalized the treatment to any arbitrary open Markovian network of queues [47]. Consider an N-node network, the i^{th} node of which contains m_i exponential servers each with a mean service time $\frac{1}{\mu_i}$. Upon leaving the i^{th} node a customer then proceeds to the j^{th} node with probability r_{ij}, or leave the network (never to return again) with probability $1 - \sum_{j=1}^{N} r_{ij}$. Denoting the total average arrival rate and the exogenous arrival rate to node i by λ_i and γ_i, respectively, then λ_i must satisfy

$$\lambda_i = \gamma_i + \sum_{j=1}^{N} \lambda_j r_{ji} \qquad (3.1)$$

for all $i = 1,2,\ldots,N$

Jackson showed that each node (say the j^{th}) in the network behaves as if it were an independent M/M/m system with a Poisson input rate λ_j, disregarding the fact that the total input might not be a Poisson process. Defining $\vec{k} = (k_1,k_2,\ldots,k_N)$ to be the state variable for the N-node system, where k_i is the number of customers in the i^{th} node, including the customer(s) in service,

the state probability $p(\vec{k})$ is then given by

$$p(\vec{k}) = \prod_{i=1}^{N} p_i(k_i) \qquad (3.2)$$

$$\text{for } \frac{\lambda_i}{m_i \mu_i} < 1 \quad \text{for all } i$$

where $p_i(k_i)$ is the marginal distribution of finding k_i customers in the i^{th} node.

Hence the joint distribution for all nodes factors into the product of each of the marginal distributions.

The class of open queueing networks is further generalized to comprise networks with N work-conserving queues and R different classes of customers. Exogenous customer arrivals are drawn from R independent Poisson sources (one for each class) with fixed rates, λ_i, $i=1,2,\ldots,R$. Customers are routed according to a Markov chain of arbitrary order. There is one chain per class. The routing is assumed to be independent of the system state. Service and arrival processes are mutually

independent. Without loss of generality, customers are not allowed to change class membership [34].

The state of the network is uniquely determined by the population vector $s = (\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N)$ where $\vec{h}_n$, the state of queue n, n=1,2,...,N, is given by

$$\vec{h}_n = (h_{n1}, h_{n2}, \ldots, h_{nr}, \ldots, h_{nR})$$

with h_{nr} being the number of class r jobs at queue n.

The stationary queue-size distribution, if it exists, is determined by a system of linear equations, similar to the ones discussed earlier. The solution is obtained as follows [34]:

STEP 1: Determine the throughput λ_{nr} of class r jobs at queue n for all n=1,2,...,N and r=1,2,...,R.

STEP 2: Determine the steady-state queue size distribution $p_n(\vec{h}_n)$ (marginal distribution) of queue n, n=1,2,...,N isolated from the network and subjected to Poisson arrivals of rate λ_{n1}, λ_{n2}, ..., λ_{nR}.

STEP 3: Obtain the joint distribution as a product of

the N marginal distributions of STEP 2, i.e.

$$p(s) = p(\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N) = \prod_{n=1}^{N} p_n(\vec{h}_n) \qquad (3.3)$$

To state the queue size distribution $p_n(\vec{h}_n)$ of queue n, n=1,2,...,N, define the capacity function [19] of queue n as follows:

$$C_n(x) = \sum_{i=0}^{\infty} a_n(i) x^i \qquad (3.4)$$

where

$$a_n(i) = \frac{(\mu_n{}^0)^i}{\prod_{j=1}^{i} \mu_n(j)} \quad \text{for } i=1,2,\ldots \qquad (3.5)$$

$$a_n(0) = 1$$

$\mu_n{}^0$ = unit work rate

$\mu_n(j)$ = work rate as a function of queue size j.

Define also the work intensity of queue n as follows

$$\vec{\rho}_n \triangleq (\rho_{n1}, \rho_{n2}, \ldots, \rho_{nR}) \qquad (3.6)$$

where

$$\rho_{nr} = \frac{\lambda_{nr}\bar{V}_{nr}}{\mu_n^0} \qquad , r=1,2,\ldots,R \qquad (3.7)$$

with

$\bar{V}_{nr}$ being the mean work demand of class r at queue n.

Define an improper queue size distribution [34] of queue n as

$$p_n^*(\vec{h}_n) \triangleq a_n(|\vec{h}_n|)\ |\vec{h}_n|! \prod_{r=1}^{R} \left(\frac{\rho_{nr}^{h_{nr}}}{h_{nr}!} \right) \qquad (3.8)$$

If the network is stable, then $p_n(\vec{h})$, not being identically zero, is obtained by normalizing $p_n^*(\vec{h})$.

The generating function (g.f.) of $p_n^*(\vec{h})$ [34] is defined as follows:

$$p_n^*(\vec{z}_n) = \sum_{\vec{h}_n \geq 0} p_n^*(\vec{h}_n) \prod_{r=1}^{R} z_{nr}^{h_{nr}} \qquad (3.9)$$

where

$\vec{z} \triangleq (z_1, z_2, \ldots, z_R)$ is a vector of transform variables associated with $\vec{h}$.

From (3.4),(3.8) and (3.9), it follows that

$$P_n^* (\vec{z}_n) = C_n (\vec{\rho}_n \cdot \vec{z}_n) \tag{3.10}$$

where

$$\vec{\rho}_n \cdot \vec{z}_n \triangleq \sum_{r=1}^{R} \rho_{nr} z_{nr} \tag{3.11}$$

Equation (3.10) simply states that $P_n^* (\vec{z}_n)$ is the capacity function of queue n evaluated at a linear combination of the workloads [19].

The probability generating function (p.g.f.) of $p_n(\vec{h}_n)$ is procured by normalizing (3.10), i.e.

$$P_n (\vec{z}_n) = \frac{C_n(\vec{\rho}_n \cdot \vec{z}_n)}{C_n(|\vec{\rho}_n|)} \tag{3.12}$$

Table 3.6 summarizes some practically important capacity functions.

From Table 3.6, some well-known probability distribution functions can readily obtained as shown in Table 3.7.

Table 3.6

Some Practically Important Capacity Functions

Queue	$\mu_n(h)$	$C_n(x)$
(1) Single server, fixed rates. (M/M/1)	μ_n^0 (constant)	$\frac{1}{1-x}$
(2) Limited queue dependent servers.	$\mu_n(h) = \begin{cases} \text{arbitrary for } 1 \leq j < d \\ \\ \mu_n^0 \text{ , otherwise} \end{cases}$	$\frac{\Theta_n(x)}{1-x}$ where $\Theta_n(x) = \sum_{i=0}^{d-1} a_n(i)[1-\mu_n(i)]x^i$
(3) M/G/∞ (IS)	$h\mu_n^0$ (linear)	$\exp(x)$

Table 3.7

Some Well-Known Probability Distribution Functions

Queue	P(z)	p(h)
Single class, fixed rate (M/M/1)	$\frac{1-\rho}{1-\rho z}$	$(1-\rho)\ \rho^h$ (geometric)
$M/G/\infty$ (IS)	$e^{-\rho_e \rho z}$	$\frac{e^{-\rho_e} \rho_e^h}{h!}$ (Poisson)

Table 3.8

Obtaining P.G.F.s of Some Marginal Distributions

p.g.f. for	Requisite modification of (3.12)
$p(h)$	set $z_{n1}=..=z_{nr}=..=z_n$, $\forall$ n=1,2,...,N
$p(h_n)$	set $z_{n1}=..=z_{nr}=..=z_n$ and all other z variables to unity
$p(H)$	substitute z_r for all z_{nr} such that (nr) ε class r, and for all r=1,2,...,R

The p.g.f. of marginal distributions are obtained by equating certain transform variables and setting others to unity (see Table 3.8).

Define $\vec{H} = (H_1, H_2, \ldots, H_R)$ = chain population vector where H_r is the total number of customers in class r = $\sum_{(nr)\ \varepsilon\ \text{class } r} h_{nr}$

Define also $h_n = \sum_{r=1}^{R} h_{nr}$ = total number of jobs at centre n.

It is clearly seen that the computational solution of queueing networks presents little difficulty. For known values of throughput λ_{nr}, $n = 1,2,\ldots,N$ and $r = 1,2,\ldots,R$, the numerical effort involved to evaluate (3.12) is negligible.

3.3.3 Mixed Networks With Multiple Closed Chains

Gordon and Newell [38] considered a variant of the Jackson model in which the total number of customers

is fixed. In this arbitrary closed Markovian network model the state space can be described by the set:

$$S = \{(h_1,h_2,\ldots,h_N) : h_i > 0, \sum_{i=1}^{N} h_i = E\}$$

where E is the number of customers allowed in the system.

Define the N-dimensional vector-value random process $X = \{X_t = (X_{t,1},\ldots,X_{t,N})\}$ for the Gordon-Newell model. Since X has a finite state space, the general theory of Markov chains assumes that it has a stationary distribution provided that the process is irreducible [18].

Gordon and Newell proved that, like Jackson's result, the stationary distribution is of product form

$$\begin{aligned} p(h_1,h_2,\ldots,h_N) &= \lim_{t\to\infty} p(h_1(t),\ldots,h_N(t)) \\ &= \lim_{t\to\infty} \text{Prob}\{X_t = (h_1,h_2,\ldots,h_N)\} \\ &= J \prod_{i=1}^{N} p_{i,h_i} \end{aligned} \qquad (3.13)$$

where

$$p_{i,h_i} = \rho_i^{h_i} (1-\rho_i^{h_i}) \qquad (3.14)$$

The normalisation constant J is selected so that

$$J^{-1} = \sum_{\text{all } X_t \in X} \prod_{i=1}^{N} p_{i,h_i} \qquad (3.15)$$

That is the probabilities in (3.13) when summed over the state space, equal unity.

Baskett et al. [36] extended the Gordon-Newell model to open, closed, and mixed queuing networks with different classes of customers. This more general model also allows general service time distributions for certain service disciplines.

Consider the network or aggregate system state S

$$\underline{S} = (h_1, h_2, \ldots, h_N)$$

where $h_i = (h_{i1}, h_{i2}, \ldots, h_{iR})$, $i = 1,2,\ldots,N$

Define the following set of linear equations for each routing chain r:

$$e_{ir} = \sum_{j=1}^{N} e_{jr} \, p_{ji,r} + q_{ir} \qquad (3.15a)$$

where

q_{ir} is the probability that the r^{th} Poisson stream enters station i (for open networks), with

$$\sum_{i=1}^{N} q_{ir} = 1 \ , \ \forall r = 1,2,\ldots,R \qquad (3.15b)$$

For closed networks,

$$q_{ir} = 0 \ , \ \forall \begin{matrix} i = 1,2,\ldots,N \\ r = 1,2,\ldots,R \end{matrix}$$

e_{ir} is the aggregate arrival rate of class r customers to station i.

For closed networks, e_{ir} are determined to within a multiplicative constant for each chain.

$p_{ij,r}$ is the probability that a customer of class r completing service at station i will next require service at station j.

The equilibrium network state probability distribution is given [27,36,45] from the following product form

$$P(S) = Cd(S) \prod_{i=1}^{N} f_i(\vec{h}_i) \qquad (3.15c)$$

where

(i) C is a normalization constant

(ii) f_i is a function that depends on the type of station i:

$$f_i(\vec{h}_i) = |\vec{h}_i|! \prod_{r=1}^{R} \frac{e_{ir}^{h_{ir}}}{h_{ir}!} \prod_{j=1}^{|\vec{h}_i|} \left(\frac{1}{\mu_i(j)}\right) \text{ for FCFS,}$$

$$f_i(\vec{h}_i) = |\vec{h}_i| \prod_{r=1}^{R} \left(\frac{1}{h_{ir}!}\right) \left(\frac{e_{ir}}{\mu_{ir}}\right)^{h_{ir}}$$

for LCFS or PS,

$$f_i(\vec{h}_i) = \prod_{r=1}^{R} \left(\frac{1}{h_{ir}!}\right) \left(\frac{e_{ir}}{\mu_{ir}}\right)^{h_{ir}} \text{ for IS.}$$

Where $\mu_i(j)$ is the exponential service rate (identical for all classes) dependent on the number

j of customers at FCFS station i.

μ_{ij} is the service rate for class j customers at station i.

$$\text{(iii)} \quad d(S) = \begin{cases} \prod_{r=1}^{R} \lambda_r^{\sum_{i=1}^{R} h_{ir}} & \text{for open network} \\ 1 & \text{for closed network} \end{cases}$$

$$C^{-1} = \sum_{S_0 \in F_s} d(S) \prod_{i=1}^{N} f_i(\vec{h}_i) \qquad (3.15d)$$

where F_s is the set of feasible states.

Georganas [45] has recently extended the consideration further to include semiclosed chains. A chain r is semiclosed if $H_r^- \leq h_r \leq H_r^+$, $r=1,2,\ldots,R$

If $h_r = H_r^-$, a departing customer of class r is immediately replaced.

If $H_r^- \leq h_r \leq H_r^+$ class-r customers arrive as a Poisson process with rate λ_r, r=1,2,...,R.

If $h_r = H_r^+$, the class-r arrivals stop.

The whole network is semiclosed with parameters H^- and H^+, if $H^- \leq \sum_{r=1}^{R} h_r \leq H^+$.

The feasible state space F_s for (3.20c) becomes

$$F_s = \left\{ S \mid h_{ir} \geq 0;\ H_r^- \leq \sum_{i=1}^{R} h_{ir} \leq H_r^+ ,\ \text{for } r=1,2,\ldots,R \right.$$

and

$$\left. H^- \leq \sum_{i=1}^{N} \sum_{r=1}^{R} h_{ir} \leq H^+ \right\}$$

In general the normalization constant cannot be straightforwardly determined. The following discussion deleineates the numerical approach taken by Reiser[24].

Consider now a mixed queueing network having W closed chains and (R-W) open chains. Let the first W ($w \leq R$) chains be conditioned [34] to contain H_w ($w=1,2,\ldots,W$) jobs.

The conditional probability

$$p(S,\vec{H}) = \text{Prob}\{\text{state}=S \mid \text{chain } w \text{ contains } H_w \text{ jobs}, \quad w=1,2,\ldots,W\} \tag{3.16}$$

where

$\vec{H}$ = the chain population vector

$= (H_1, H_2, \ldots, H_W)$

From the law of conditional probability,

$$p(S,\vec{H}) = \begin{cases} p(S)/g(\vec{H}) & \text{if } S \text{ is feasible} \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

where

$$g(\vec{H}) = \text{Prob}\{\text{chain } w \text{ contains } H_w \text{ jobs}, \quad w=1,2,\ldots,W\} \tag{3.18}$$

Evidently [19],

$$g(\vec{H}) = \sum_{S \in F} p(S) \qquad (3.19)$$

where

$$F \triangleq \{S : S \geq 0 \text{ and } \sum_{n=1}^{N} h_{nw} = H_w \text{ for } w=1,2,\ldots,W\}$$

= set of feasible states

A conditioned chain is equivalent to a closed chain up to renaming of customers [34]. The conditioned solution is independent of the arrival rates to the underlying open network [19].

The normalization constant $g(\vec{H})$ is the convolution of the N individual queue size distributions, taken at the point of chain populations [19].

Define the g.f. of the improper joint queue-size distribution as

$$P^*(\phi,\vec{Z}) \triangleq \sum_{S \geq 0} \left(\prod_{n=1}^{N} p_n^*(\vec{h}_n) \right) \left(\prod_{n=1}^{N} \prod_{r=1}^{R} z_{nr} \right) \left(\prod_{w=1}^{W} z_w^{h_w} \right) \qquad (3.21)$$

where

$\vec{Z} \triangleq (Z_1, Z_2, \ldots, Z_W)$ is a transform vector associated with $\vec{H}$.

Introduce the notation

$$r_n^0 \triangleq \sum_{r=W+1}^{R} \rho_{nr} \quad , \text{ and} \tag{3.22a}$$

$$\vec{r}_n \triangleq (\rho_{n1}, \rho_{n2}, \ldots, \rho_{nW}) \tag{3.22b}$$

For closed chain w, the throughput at queue n is given by

$$\lambda_n(\vec{H}) \triangleq \sum_{S \in F} p(S, \vec{H}) \sum_{r=1}^{R} \frac{\mu_n(|\vec{h}_n|)}{\bar{V}_{nr}} \tag{3.23}$$

also define

$$G^*_{(n-)}(\vec{Z}) \triangleq \prod_{\substack{i=1 \\ i \neq n}}^{N} C_i \, (r_i^0 + \vec{r}_i \cdot \vec{Z})$$

$$G^*_{(n+)}(Z) \triangleq C_n(r_n^0 + \vec{r}_n \cdot \vec{Z}) \prod_{i=1}^{N} C_i(r_i^0 + \vec{r}_i \cdot \vec{Z}) \tag{3.24a}$$

Some important p.g.f.s [34] are summarized in Table 3.9.

The derivations of the above p.g.f.s are found in [24].

It should be noted that the joint queue-size probability distribution $p(S,\vec{H})$ is independent of λ_w, $w=1,2,\ldots,W$, the arrival rates of conditioned chains [18]. Moreover, a mixed network is said to be stable if it is stable with respect to the open chains (i.e. be stable if $\lambda_w=0$, for $w=1,2,\ldots,W$) [34].

As seen from the expression for p.g.f. of $g^{\bullet}(H)$ in Table 3.9, the open (nonconditioned) chains shift the argument of the capacity function in the solution. This shift does not enhance the computational complexity for infinite, fixed rate or limited queue-dependent server system [24]. Hence, save for the evaluation of marginal queue size distribution, open chains in general do not increase the computational complexity. Hence in the analysis to follow, we

Table 3.9 Some Important Probability Generating Functions

Distribution	Probability Generating Functions
Improper Joint Queue-size	$P^*(\phi,\vec{Z}) = \prod_{n=1}^{N} C_n\left(\sum_{r=W+1}^{R} \rho_{nr} z_{nr} + \sum_{w=1}^{W} \rho_{nw} z_{nw} Z_w \right)$
Improper Marginal Queue-Size	$P_n^*(z,\vec{Z}) = C_n(z\,[r_n^0 + \vec{r}_n \cdot \vec{Z}])\; G^*_{(n-)}\;(\vec{Z})$
Improper Throughput of Queue n	$\Lambda_n^*\;(\vec{Z}) = (\lambda_n^0 + \vec{\lambda}_n \cdot \vec{Z})\; G^*(\vec{Z})$ where $\lambda_n^0 = \sum_{r=W+1}^{R} \lambda_{nr}$, $\vec{\lambda}_n \triangleq (\lambda_{n1}, \lambda_{n2}, \ldots, \lambda_{nW})$
Improper Mean Queue Size of Queue n	$Q_n^*(\vec{Z}) = (r_n^0 + \vec{r}_n \cdot \vec{Z})\; C_n'\;(r_n^0 + \vec{r}_n \cdot \vec{Z})\; G^*_{(n-)}\;(\vec{Z})$ where $C_n'\;(x) = \frac{dC_n(x)}{dx}$
Improper Normalization Constant	$G^*(\vec{Z}) = \prod_{n=1}^{N} C_n\;(r_n^0 + \vec{r}_n \cdot \vec{Z})$

exclude the open chains completely.

The main objective, as said earlier, is the computation of the normalization constant. Now

$$G(\vec{Z}) = \prod_{n=1}^{N} C_n (\vec{r}_n \cdot \vec{Z}) \qquad (3.25)$$

There exist two categories of practical methods [18]:

(a) Inversion of generating functions

(b) Convolution of the N known inversion C_n of the terms $C_n(r_n \cdot Z)$ in (3.25)

Well-behaved algorithms do not appear to exist for the first approach. In the case when there is only one closed chain and all queues have fixed rate servers, even as the r_i may not be all distinct, Lam [35] has shown that $g(\vec{K})$ can be obtained by a partial fraction expansion of $G(\vec{Z})$, a rational function in Z. However, computations based on partial fraction expressions involve alternating sign sums [34], so that

they are only conditionally stable numerically. Moreover they are of the same computational complexity and much less versatile than the convolution methods [18]. Hence the convolution methods reign supreme.

Convolution methods for simple closed networks were first introduced by Buzen [25] and independently by Reiser [34]. These methods were extended to multichain systems by Reiser and Kobayashi [24,50].

Consider the evaluation of $G(\vec{Z})$ in (3.25). Now

$$C_n(\vec{r}_n \cdot \vec{Z}) = \sum_{\vec{i} \geq 0} c_n(\vec{i}) \prod_{w=1}^{W} Z_w^{i_w} \qquad (3.26)$$

where

$\vec{i}$ = W - dimensional index vector = $(i_1, i_2, \ldots, i_W)$

$c_n(\vec{i})$ = inverse of $C_n(\vec{r}_n \cdot \vec{Z})$

By definition of capacity function:

$$c_n(\vec{i}) = a_n(|\vec{i}|)\ |\vec{i}|! \prod_{w=1}^{W} \frac{r_{n,w}^{i_w}}{i_w!} \qquad (\vec{i} \geq 0) \qquad (3.27)$$

Hence g can be obtained by inverting $G(\vec{Z})$ by N convolutions as follow

$$g(\vec{H}) = (c_1 * c_2 * \ldots * c_N) \quad (\text{at } \vec{i}=\vec{H}) \qquad (3.28)$$

Recursive computation of the convolution in (3.28) is explored for queues with fixed rate, limited queue-dependent, and infinite servers.

(i) Queues With Fixed Rate Servers

Here

$$C(x) = \frac{1}{1-x}$$

Hence

$$G_{(n)}(\vec{Z}) = \frac{1}{1-\vec{r}_n \cdot \vec{Z}} \; G_{(n-1)}(\vec{Z}) \qquad (3.29)$$

Inverting,

$$g_{(n)}(\vec{i}) = g_{n-1}(\vec{i}) + \sum_{w=1}^{W} r_{n,w} \, g_{(n-1)}(\vec{i}-\vec{e}_w) \qquad (3.30)$$

where

$$e_w = (\varepsilon_{w1}, \varepsilon_{w2}, \ldots, \varepsilon_{wW})$$

and

$$s_{ij} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$$

(ii) Limited Queue-Dependent Servers

Recall that in this case

$$C(x) = \frac{\theta(x)}{1-x}$$

Therefore

$$G^{(n)}(\vec{Z}) = \frac{\theta_n(\vec{r}_n \cdot \vec{Z})}{1-\vec{r}_n \cdot \vec{Z}} \quad G^{(n-1)}(\vec{Z})$$

Where $\theta_n(x)$ is as defined in Table 3.6.

(iii) Infinite Server (M/G/∞) Queues)

The capacity coefficient in this case is

$$a_n(i) = \frac{1}{i!} \quad , \quad i > 0$$

so that

$$C_n(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!} = e^x$$

$$G(\vec{Z}) = \prod_{n=1}^{N} C_n (\vec{r}_n \cdot \vec{Z})$$

$$= \prod_{n=1}^{N} \exp(\vec{r}_n \cdot \vec{Z})$$

$$= \exp(\vec{Z} \cdot \sum_{n=1}^{N} \vec{r}_n)$$

Much has been said on the computation of the normalization constant $g(\vec{H})$. However, it is just as important to compute queue statistics such as throughput, mean queue-size and utilization factors. It will be seen that the g.f. of the queue statistics are related to $G(\vec{Z})$, the g.f. of $g(\vec{H})$ [19,34].

(i) Throughput of Queue n

From Table 3.8, excluding the open chains, the improper throughput $\Lambda_n^*(\vec{Z})$ of queue n is given by

$$\Lambda_n^*(\vec{Z}) = (\vec{\lambda}_n \cdot \vec{Z}) \cdot G^*(\vec{Z}) \qquad (3.33)$$

where

$$\vec{\lambda}_n \triangleq (\lambda_{n1}, \lambda_{n2}, \ldots, \lambda_{nW}$$

Hence, inverting (3.33) and normalizing

$$\lambda_n(\vec{H}) = \frac{\sum_{w=1}^{W} \lambda_{nw}\, g(\vec{H}-\vec{u}_w)}{g(\vec{H})} \qquad (3.34)$$

(ii) Mean Queue Size

(a) Queue n With Fixed Rate Server

From Table 3.9 again excluding the open chains, for fixed rate servers

$$Q_n^*(\vec{Z}) = (\vec{r}_n \cdot \vec{Z})\; G^*_{(n+)}(\vec{Z}) \qquad (3.35)$$

where

$$G^*_{(n+)}(\vec{Z}) \triangleq C_n(\vec{r}_n \cdot \vec{Z}) \cdot \prod_{i=1}^{N} C_i(\vec{r}_i \cdot \vec{Z})$$

Inverting (3.35) and normalizing

$$q_n(\vec{H}) = \frac{\sum_{w=1}^{W} r_{nw} g_{(n+)}(\vec{H}-\vec{u}_w)}{g(\vec{H})} \qquad (3.36)$$

(b) Infinite Server Queue

Again, from Table 3.9 and excluding the open chains, for infinite server,

$$Q_n^*(\vec{Z}) = (\vec{r}_n \cdot \vec{Z})\ G^*(\vec{Z}) ,$$

Inverting (3.37) and normalizing

$$q_n(\vec{H}) = \frac{\sum_{w=1}^{W} r_{nw} g(\vec{H}-\vec{u}_w)}{g(\vec{H})} \qquad (3.37)$$

(iii) Other Quantities

These include the mean queue-size of queues with queue-dependent servers, utilization factors and marginal queue size distributions.

The computation of these quantities require $g_{(n-)}$, the inverse of $G_{(n-)}(\vec{Z})$ [19], where

$$G^*_{(n-)}(\vec{Z}) = \prod_{\substack{i=1 \\ i \neq n}}^{N} C_i(r_i^0 + \vec{r}_i \cdot \vec{Z})$$

An efficient algorithm for computing $g_{(m-)}$ is portrayed in [34].

3.4 COMMENTS

It is recalled that in the end-to-end flow control scheme, a limit is placed on the number of customers allowed in each open routing chain. The situation can be viewed as follows:

Customers flow from the source node, via a sequence of forward route link queues, to the sink node. After the customer is absorbed at the sink, an acknowledgment traverses from the sink to the source. The source throttles the incoming traffic as the number of overdue acknowledgments builds up. And as soon as the latter reaches a certain fixed limit E, quiescence of incoming traffic sets in.

It is then not difficult to see that the return route of the acknowledgments from sink to source closes an open routing chain. That is the end-to-end flow

control scheme transforms an open queueing network into a closed one with multiple cyclic routing chains.

The convolution algorithms for closed networks just portrayed are then conjectured to be conducive to the analysis of queueing networks with end-to-end flow control, as is confirmed by the work of [44]. However, the computational limitations of these algorithm [4] do not favour recursive applications in practical design problems like selection of window settings. A computationally-effective algorithm is discussed in the following chapter as a circumvention.

CHAPTER 4

THE WINDIM ALGORITHM

4.1. INTRODUCTION

We now propose the WINDIM algorithm for resolving the window setting problem in computer-communication networks with end-to-end flow control. For the class of multichain queueing networks considered in this work, WINDIM determines "good" window settings $\vec{E}_{op}$ that optimize a certain performance criterion. This criterion is simply the ratio of network throughput to average network delay, called the network "power" [5].

WINDIM essentially comprises the well-known multi-dimensional search technique called Pattern Search [22]. More advanced search techniques can of course be used. However, since we are interested only in integral window settings, we have an integer programming problem and the Pattern Search suffices. The Pattern Search has the added advantage of being an easily programmed accelerated climbing technique with ridge-following properties, as will

be seen later.

Because of the inherent prodigious computational requirements, the exact analytical model, described in the previous chapter, cannot be used to perform the required function evaluations. These evaluations are afforded instead by the heuristics of the Mean Value analytical model recently developed by Reiser [4,23].

4.2 MEAN VALUE ANALYSIS OF MULTICHAIN NETWORKS

The motivation towards a heuristic approach to the analysis of multichain networks is twofold:

(a) The exact analysis, even with the use of the convolution algorithm [24,25] for closed networks, has a considerable programme complexity.

(b) The joint probability distribution obtained from the exact analysis contains too much detail. The computation of simpler and more practical quantities such as mean queue sizes, mean waiting times, and throughputs from the product form solution

proves to be as complex as computing the normalization constant for the distribution [23].

Reiser and Lavenberg [23] have proposed a recursive analysis which is based on a simple mean-value relation between the waiting time and queue size of a system with one less customer. This efficient heuristic analysis successfully avoids the difficult calculations of normalization constants and product terms. Hence the computational limitations that beset the exact analysis no longer exist in this case.

In the following discussion, the basic network model described in the third chapter will be assumed. The class of message switching networks considered will therefore comprise N switching nodes, L half-duplex links and R classes of messages. We further stipulate that:

(a) the links be modelled by simple FIFO infinite queues
(b) there be a FCFS single unit rate server to each queue
(c) the message lengths be exponentially distributed with the same mean length for all classes
(d) an end-to-end flow-control protocol be imposed on each class.

It was observed [4] that by imposing end-to-end flow control on an open queueing network, a closed one with multiple cyclic closed chains results.

The analysis starts by considering a simple cyclic closed chain with population D and throughput $\lambda(D)$ as shown in Fig. 4.1.

For any queue i, i=1,2,...,M , define:

Γ_i = mean service time.

$N_i(D)$ = mean queue length (including message in service).

$t_i(D)$ = mean queue time (including message in service).

It is then clear [4] that:

$$t_i(D) = \Gamma_i + \Gamma_i \cdot [\text{mean queue length of queue i upon arrival of a message}] \qquad (4.1)$$

$$\lambda(D) = \frac{D}{\sum_{i=1}^{M} t_i(D)} \qquad (4.2)$$

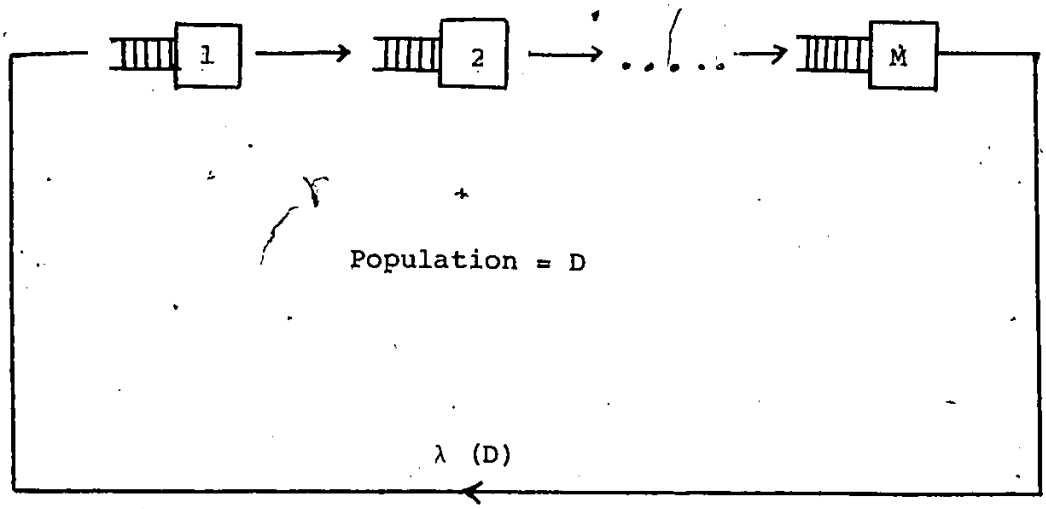


Fig. 4.1 A Simple Cyclic Closed Chain

$$N_i(D) = \lambda(D)\ t_i(D) \tag{4.3}$$

Equations (4.2) and (4.3) are simply Little's equations for the entire chain and for queue i, respectively.

For the class of closed multichain queueing networks under consideration, Reiser and Lavenberg [23] have shown that an arriving customer "sees" the system with himself removed in equilibrium. In other words,

$$t_i(D) = \Gamma_i[1+N_i(D-1)] \tag{4.4}$$

Extending the single chain result to the multichain case [23], we have, for r=1,2,...,R

$$\lambda_r = \frac{D_r}{\sum_{i\in Q(r)} t_{ir}} \tag{4.5}$$

$$N_{ir}(D) = \lambda_r\ t_{ir} \tag{4.6}$$

$$t_{ir} = \Gamma_{ir}[1+ \sum_{j\in R(i)} N_{ij}\ (\vec{D}-\vec{u}_r)] \tag{4.7}$$

where

Q(r) = set of queues in chain r

$$\vec{D} = (D_1, D_2, \ldots, D_R)$$

$$\vec{D} - \vec{u}_r = (D_1, D_2, \ldots, D_{r-1}, D_r - 1, \ldots, D_R)$$

R(i) = set of chains visiting queue i

However, although the mean value analysis is simple, the R-dimensional recursive solution of equations (4.5) - (4.7) is computationally as complex as the original recursion which uses the traditional convolution algorithm approach [24]. The operations count is in the order of $\prod_{r=1}^{R} E_r$.

A heuristic was next developed to circumvent this problem [23].

Define

$$\sigma_{ij}(r-) = N_{ij}(\vec{D}) - N_{ij}(r-)$$

$$= N_{ij}(\vec{D}) - N_{ij}(\vec{D} - \vec{u}_r) \qquad (4.8)$$

Here $\sigma_{ij}(r-)$ denotes the increment in mean queue size for chain j at queue i when the population of chain r is augmented by one.

We must have

$$\sum_{i \varepsilon Q(r)} \sigma_{ij}(r-) = \begin{cases} 1 & \text{if } j = r \\ 0 & \text{otherwise} \end{cases} \qquad (4.9)$$

It is intuitively appealing to assume [4] that

$$0 \leq \sigma_{ij}(r-) \leq 1 \qquad (4.10)$$

As a first heuristic approximation, it is ruled that [23] the chain with one customer removed is affected mostly. In other words, for i=1,2,...,L , $j \neq r$, and j, $r \varepsilon R(i)$,

$$\sigma_{ir}(r-) >> \sigma_{ij}(r-) \qquad (4.11)$$

Hence, we will assume that $\sigma_{ij}(r-)$ equals zero, for $j \neq r$.

It is further reasoned [4] that, with suitably redefined parameters, the affected chain can be isolated and analysed in a single closed chain problem:

$$\sigma_{ir}(r-) = \tilde{N}_i(D_r) - \tilde{N}_i(D_r-1) \qquad (4.12)$$

where,

$\tilde{N}_i$ is the mean queue size in the single chain problem.

Of course we still have

$$t_{ir} = \Gamma_{ir}\left[1+ \sum_{j\varepsilon R(i)} (N_{ij} - \sigma_{ij})\right] \qquad (4.13)$$

$$\lambda_r = \frac{D_r}{\sum_{i\varepsilon Q(r)} t_{ir}} \qquad (4.14)$$

$$N_{ir} = \lambda_r t_{ir} \qquad (4.15)$$

The system (Equations (4.12) - (4.15)) can therefore be solved iteratively starting with $\tilde{N}_i(0) = 0$

For a given network, the operations count in this case is of the order $\sum_{r=1}^{R} E_r$. This is a considerable improvement over the exact solution procedure.

We recall that

$Q(r)$ = set of queues in chain r

$R(i)$ = set of chains visiting queue i

The following procedure now describes the iterative heuristic:

STEP 1: Initialize the mean queue size N_{ir} and class throughput λ_r for all r=1,2,...,R and all $i \epsilon Q(r)$.

STEP 2: Estimate $\sigma_{ir}(r-)$ from the heuristic of the single chain system summarized by Equation (4.12) for all $i \epsilon Q(j)$ and all $r \epsilon R(i)$.
Set $\sigma_{ij}(r-)$ to zero for j=1,2,...,R and $j \neq r$ and all $i \epsilon Q(j)$.

STEP 3: Calculate the mean queueing times t_{ir} for

$r=1,2,\ldots,R$ and all $i \varepsilon Q(r)$

$$t_{ir} = \Gamma_{ir}[1+ \sum_{j \varepsilon R(i)} N_{ij} (\vec{D} - \vec{u}_r)] \quad , i \varepsilon Q(r)$$

STEP 4: Calculate the class throughputs using Little's equation for chains:

$$\lambda_r = \frac{D_r}{\sum_{i \varepsilon Q(r)} t_{ir}} \qquad \text{for } r=1,2,\ldots,R$$

STEP 5: Calculate the mean queue lengths using Little's equation for queues:

$$N_{ir} = \lambda_r t_{ir} \qquad \text{for } r=1,2,\ldots,R \text{ and all } i \varepsilon Q(r)$$

STEP 6: GO TO STEP 2 if the stopping condition (e.g. convergence criterion) is not met. Otherwise, terminate the procedure.

It has been shown [26] that, at least for networks with product-form solution, the above procedure is

asymptotically valid as the population D becomes large. In other words, the heuristic algorithm yields better results as the chain population and/or the number of chains increases.

There are at least two reasonable ways in which the mean queue size can be initialized in STEP 1:

(1) Static Location of Bottleneck Queue

A bottleneck queue has the largest relative utilization $\frac{\lambda}{u}$ among all the queues in a chain. As the chain population D_r increases without bounds, the bottleneck queue becomes infinite while the other queues remain finite.

Here at the initialization step D_r messages are put at the queue whose mean service time for class r is largest. That is,

$$N_{ir} = \begin{cases} D_r & \text{for } 1 \le i^* \le N \text{ such that } \Gamma_{i^*r} \\ & \text{is largest.} \\ 0 & \text{for all other } i. \end{cases} \qquad (4.16)$$

(2) Static Assumption of Totally Balanced Chain

A chain is said to be totally balanced if the

chain population D_r is evenly distributed over the queues in the chain.

Hence, denoting the cardinality of Q(r) by q_r, one can initialize N_{ir} by setting

$$N_{ir} = \begin{cases} \frac{D_r}{q_r} & \text{for all } i \varepsilon Q(r) \\ 0 & \text{for all } i \notin Q(r) \end{cases} \qquad (4.17)$$

N_{ir} can of course be initialized otherwise as long as the mean queue sizes in each chain of the network sum up to the chain population. That is,

$$\sum_{i \varepsilon Q(r)} N_{ir} = D_r , \quad r=1,2,\ldots,R \qquad (4.18)$$

4.3 PATTERN SEARCH

Pattern search [22] is a direct search routine for minimizing a certain objective function with search moves

pointing in the approximate gradient directions. The requisite function evaluations are furnished by the efficient heuristic procedure just described.

A performance ratio P called "power", is now defined [5] where:

$$P \triangleq \frac{\lambda}{T} \tag{4.19}$$

and

λ = network throughput = $\sum_{r=1}^{R} \lambda_r$

T = average network delay = $\frac{1}{\lambda} \sum_{r=1}^{R} \sum_{i \in V(r)} t_{ir}$

$V(r)$ = $Q(r)$ - (reentrant queue from sink to source)

$Q(r)$ = set of queues in chain r

let

$\vec{e}$ = initial window sizes = $(e_1, e_2, \ldots, e_R)$

$\vec{y}$ = initial parameter increments (step sizes) = $(y_1, y_2, \ldots, y_R)$

F = objective function to be minimized = $\frac{1}{p}$

It is clear that F is a function of the window sizes, i.e.:

$$F = F(\vec{e}) \qquad (4.20)$$

Each step of the pattern search consists of two kinds of moves, namely, exploratory and pattern. These are described as follows:

(a) Exploratory Move

Here the local behaviour of F about a point Q (designated by the position vector $\vec{e}$) is investigated. A single variable (i.e. a window) of the point is perturbed by either increasing or decreasing its value (i.e. window size) by the prescribed increment. If this move successfully minimizes F, the altered value is retained, otherwise, the original value is restored. Such perturbation is made for each variable, and the final point reached becomes a new base point, i.e. a new point from which a pattern move is made. Fig. 4.2 provides an illustration of local exploration in the two dimensional case.

(b) Pattern Move

This move is determined from the results of the exploratory move. It is reasoned that if a similar exploration is made from the present base point, the results will likely be the same. The move simply duplicates the combined moves from the previous base point. In other words, all the window sizes constituting the present base point are changed by an amount equal to the difference between the present base point and the previous base point. An exemplary pattern move is shown in Fig. 4.3.

At any stage, moves subsequent to the local exploration depend on the sequence of moves already made as part of the current pattern. Each pattern comprises a series of successful moves, in each of which F is reduced. The step sizes are augmented if the search successfully reduces F. Otherwise the pattern is terminated, and a new pattern is started by local exploration with reduced step sizes about the last base point determined. In order to demonstrate various modes of action, the progress of a single two-dimensional pattern be followed with reference to Fig. 4.4.

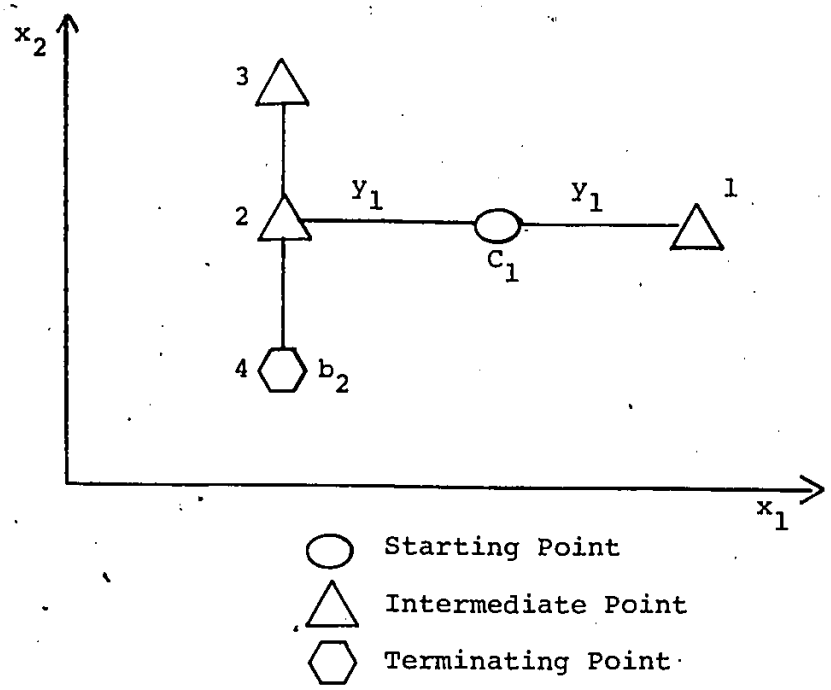


Fig. 4.2 Local Exploration in two dimensions

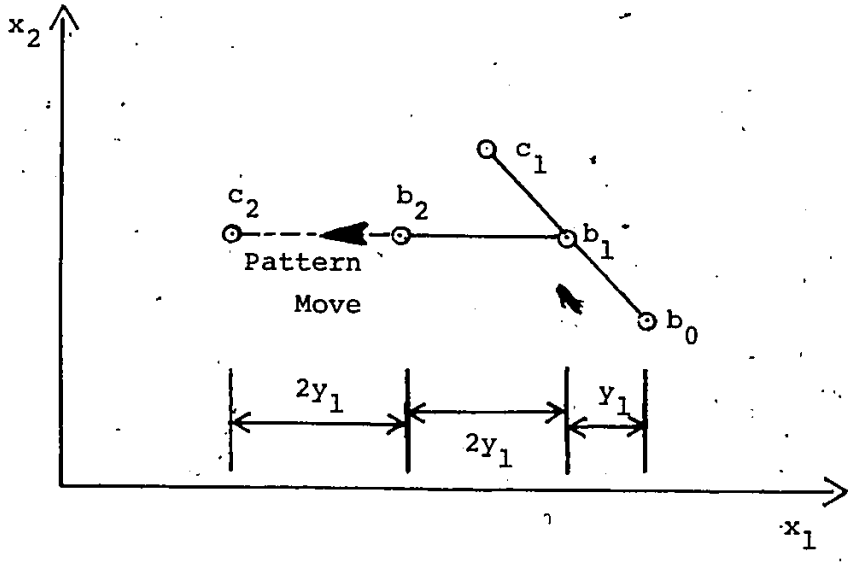


Fig. 4.3 Pattern Move in two dimensions

Local exploration about the first base point b_0 produces point b_1 at which F has value less than that at b_0. b_1 is now the new base point. The pattern move is made from b_0 to c_1, by extending the line from b_0 to b_1 to twice its length. At c_1, local exploration establishes a new minimum at b_2. b_2 becomes the new base point, and the corresponding pattern move ensues.

It is noted that pattern moves are always made in the direction determined by a line form the previous base point to the current base point, and by extending this line to twice its length. Hence the size of pattern movement is increased as a successful direction is established, as elucidated in Fig. 4.2.

The current pattern reveals itself by alternate exploratory and pattern moves from the first base point b_0 up to b_4. c_4 is the point reached by a pattern move from b_4. Local exploration at c_4 fails to produce a point at which F has value less than that at b_4. Hence the current pattern is terminated. A new pattern is started by local exploration about b_4 with increments reduced by a factor of half. A new base point b_5 is hence established, and the search process continues with the new pattern.

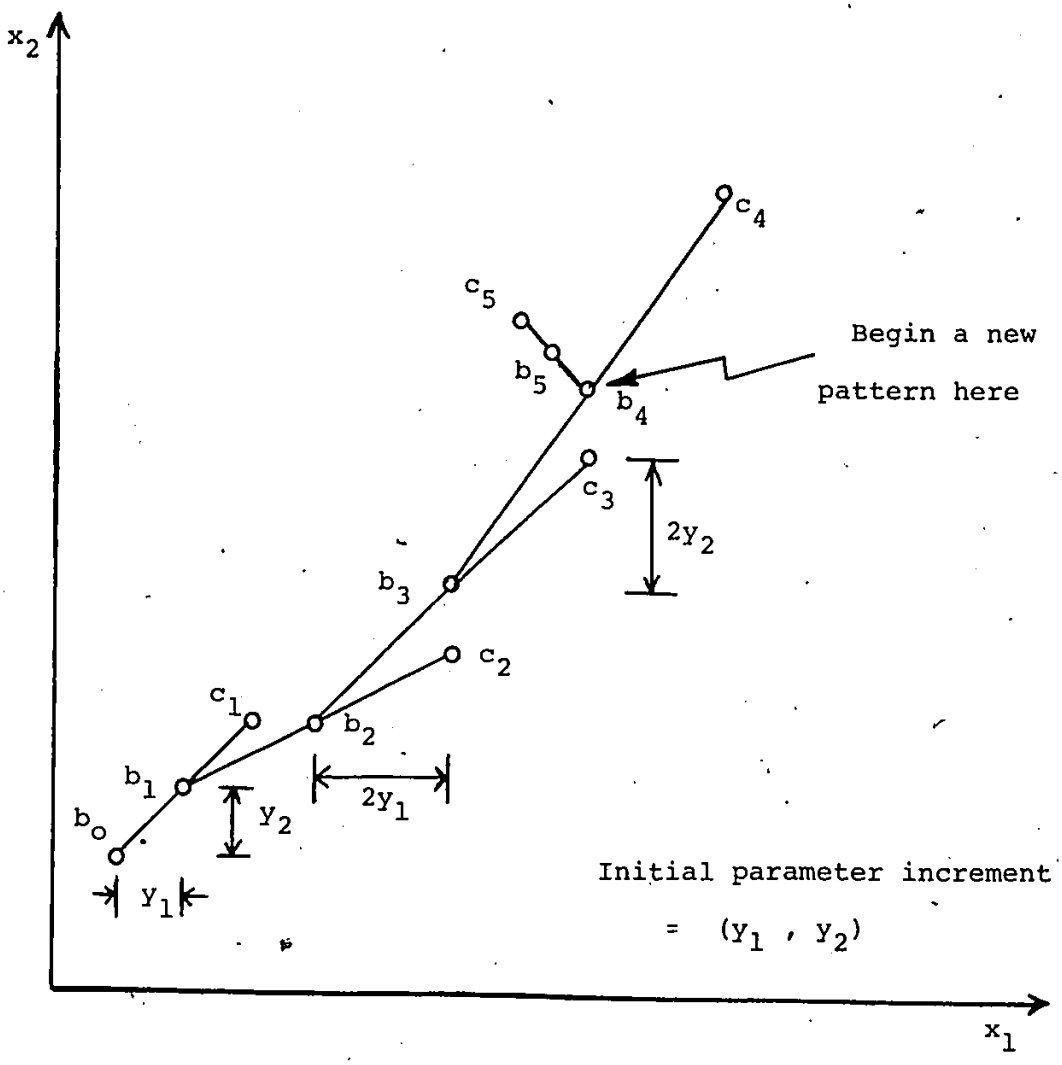


Fig. 4.4 The Two Dimensional Pattern Search

4.4 WINDIM ALGORITHM [51]

The WINDIM algorithm sets out to dimension the end-to-end flow control windows such that a performance criterion, called the network "power" [5], is optimized. This criterion , as defined earlier, is the ratio of the network throughput to the average network delay.

The objective function F, being defined as the reciprocal of the network power, is evaluated by invoking the Mean Value Algorithm.

The WINDIM algorithm can be described by the following procedure:

STEP 1: Choose the initial window-setting vector $\vec{e}$ where $\vec{e} = (e_1, e_2, \ldots, e_R)_{initial}$ for all classes.

STEP 2: $\vec{e}$ designates the current base point window settings. Evaluate $F(\vec{e})$, set t=0.

STEP 3: Make exploratory moves about $\vec{e}$ to arrive at

settings $\vec{e}_{e(t)}$.

STEP 4: If $F(\vec{e}_{e(t)}) < F(\vec{e})$, $\vec{e}_{e(t)}$ becomes the new base point window-setting vector.
Else GO TO STEP 8.

STEP 5: Make a pattern move from $\vec{e}_{e(t)}$ to $\vec{e}_{p(t)}$; set $t = t + 1$.

STEP 6: Make exploratory moves from $\vec{e}_{p(t-1)}$ to $\vec{e}_{e(t)}$.

STEP 7: If $F(\vec{e}_{e(t)}) < F(\vec{e}_{e(t-1)})$, $\vec{e}_{e(t)}$ becomes the new base point window-setting vector. GO TO STEP 5. Else $\vec{e} = \vec{e}_{e(t-1)}$, GO TO STEP 2.

STEP 8: If step size is not small enough, reduce step size and GO TO STEP 2. Else record $\vec{e}$ as the optimal window-setting vector and terminate the procedure.

For the initialization of the window sizes, the single chain problem is considered. If each link is modelled by a single M/M/1 queue, and if there are β hops in the chain, then P is maximized (and hence F minimized) when the window size is β [52]. Thus a good

approximation to the initial window-setting vector is $(\beta_1, \beta_2, \ldots, \beta_R)$ where β_r is the number of hops in chain r.

4.5 NUMERICAL RESULTS

The WINDIM algorithm has been implemented in APL and run on an IBM 360/65 to study some simple networks.

The first network is shown in Fig. 4.5. This network has six switching nodes and seven half-duplex channels. Channels 1 through 5 have a capacity of 50 kbits/sec each. Channels 6 and 7 each has capacity 25 kbits/sec. The queueing discipline is FIFO.

Two message classes are defined in the network. Class 1 messages originate, with Poisson rate S_1, at Edmonton and are routed to Ottawa via Winnipeg, Toronto, and Montréal. Class 2 messages originate, with Poisson rate S_2, at Montréal and are routed to Vancouver via Winnipeg, Toronto, and Edmonton. The messages are exponentially distributed with mean length 1000 bits for both classes.

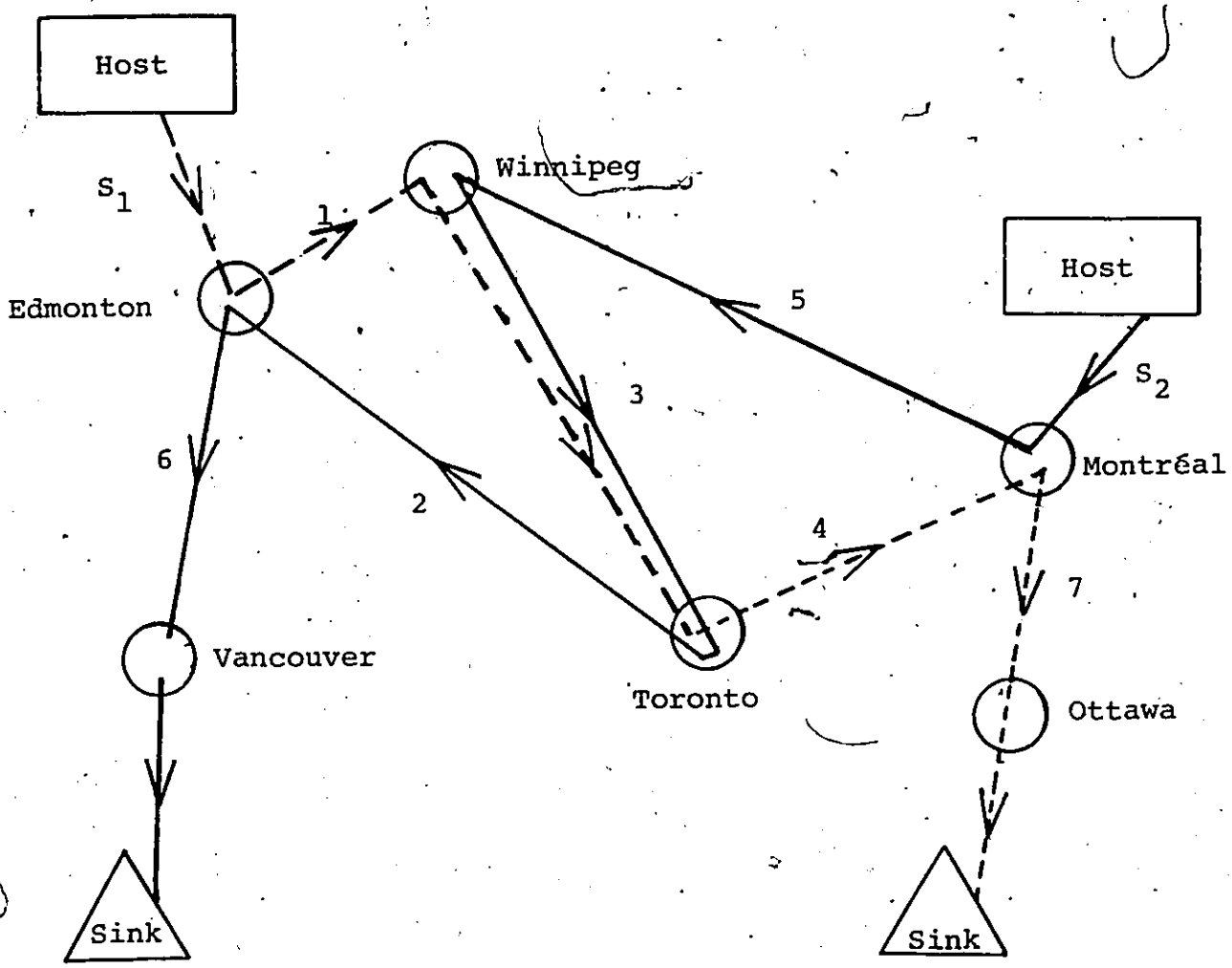


Fig. 4.5 A Network Example in Two Classes

The queueing model of the network is shown in Fig. 4.6. This model comprises two chains and nine queues. Queues 1 to 7 represent the 7 channels interconnecting the nodes. Queues 8 and 9 model the sources of the two chains.

The WINDIM algorithm has been applied to investigate the effect of varying the class arrival rates S_1 and S_2 on the optimal window settings, i.e. that set of windows which maximizes the power P as defined in [5]. The results are summarized in Table 4.7 and 4.8.

From Table 4.7 it can seen that, with symmetrical class loadings, the optimal window sizes are also symmetrical. This is readily explained by the symmetry in the routings and network parameters for the two classes. If the traffic arrival rates are increased, the maximum power also increases. Moreover, smaller window settings are needed to attain this maximum power. This is true since the heavier the traffic, the greater is the tendency for the network to be congested, and so a more stringent end-to-end control is in order.

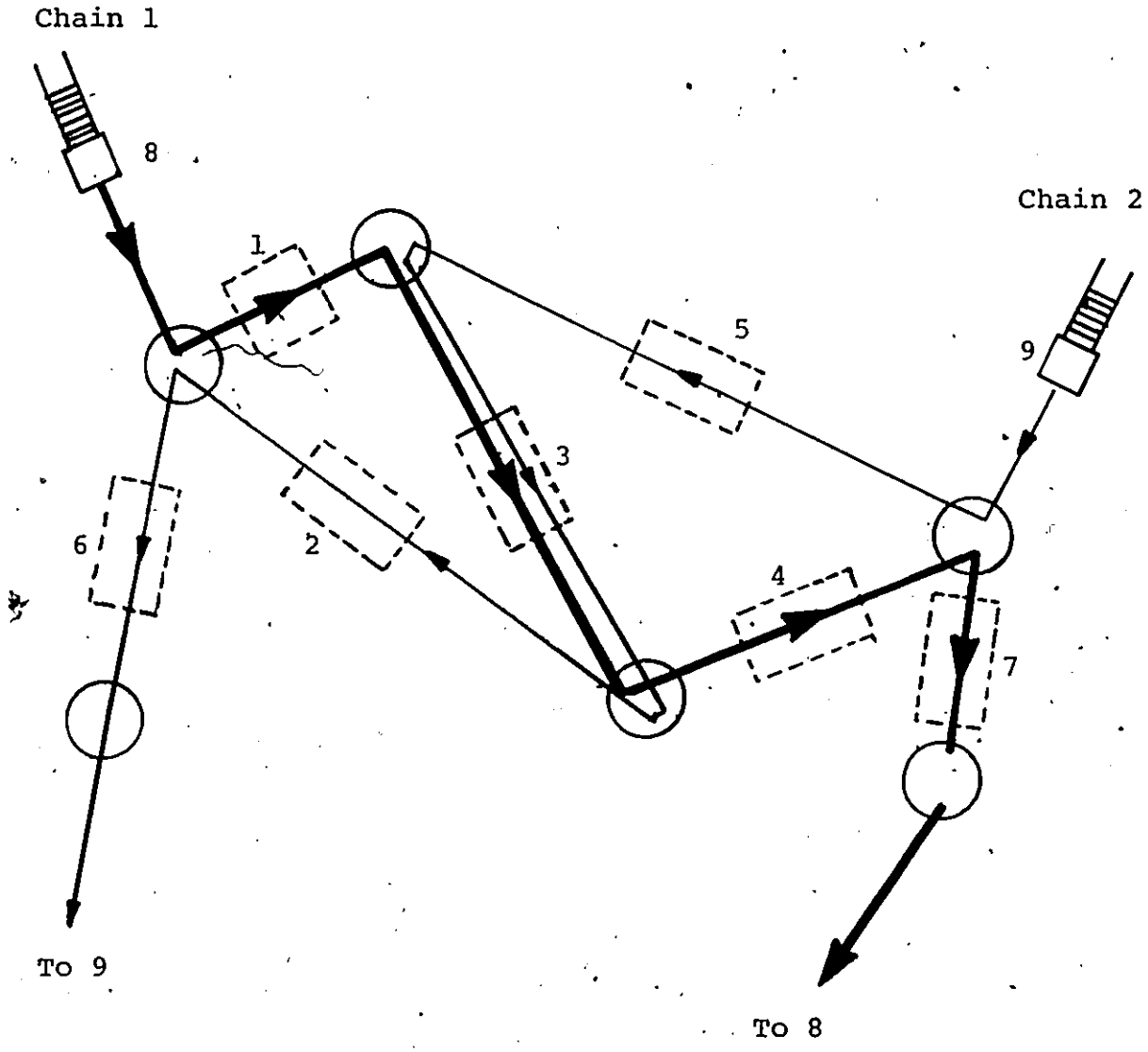


Fig. 4.6 Closed Chain Model of Example Network in Two Classes

Table 4.7

Effect of Symmetrical Class Loadings on Optimal Window Settings for a 2-Class Network Example

Class 1 Arrival Rate S_1 (msg/sec)	Class 2 Arrival Rate S_2 (msg/sec)	Total Network Arrival Rate $S_1 + S_2$ (msg/sec)	Optimal Window Settings	Network Power
12	13	25	5 5	159
15.5	15.5	31	5 5	173
18	18	36	4 4	179
20	20	40	4 4	182
22.5	22.5	45	4 4	183
25	25	50	3 3	184
37.5	37.5	75	3 3	190
50	50	100	3 3	192
62.5	62.5	125	2 2	194
75	75	150	2 2	196

The effect of dissimilar class arrival rates on the optimal window settings is shown in Table 4.8. It is noted that even for class arrival rates differing by a factor of up to 3 or 4, the optimal window sizes remain close to those for symmetrical loading. This is especially appealing as instantaneous window sizing is virtually impractical, and so the window settings should be as insensitive to traffic fluctuations as possible. Table 4.8 also shows that the degradation in the maximum power increases as the class arrivals differ more and more. It is therefore advantageous to operate the network with similar loading for the classes.

In probing the global optimality of the window sizes selected, the power P has been obtained for the full range of arrival rates with different window sizes. The results are shown in Fig. 4.9. It is observed that for window sizes greater than or equal to (5,5), and with increasing applied traffic, the power initially builds up rapidly to a maximum value, but then degrades just as fast to some steady-state value, to remain there unaffected by further increase in loading. It is evident

Table 4.8

Effect of Dissimilar Class Loadings on Optimal Window Settings for a 2-Class Network Example

Class 1 Arrival Rate S_1 (msg/sec)	Class 2 Arrival Rate S_2 (msg/sec)	Total Network Arrival Rate $S_1 + S_2$ (msg/sec)	$\frac{S_2}{S_1}$	Optimal Window Settings		Network Power
12	13	25	1.08	5	5	159
10	15	25	1.50	5	5	157
8.4	16.6	25	1.98	5	4	153
7	18	25	2.57	5	4	147
5	20	25	4.00	5	4	138
18	18	36	1.00	4	4	179
15	21	36	1.40	5	4	177
12	24	36	2.00	5	3	172
9	27	36	3.00	5	3	161

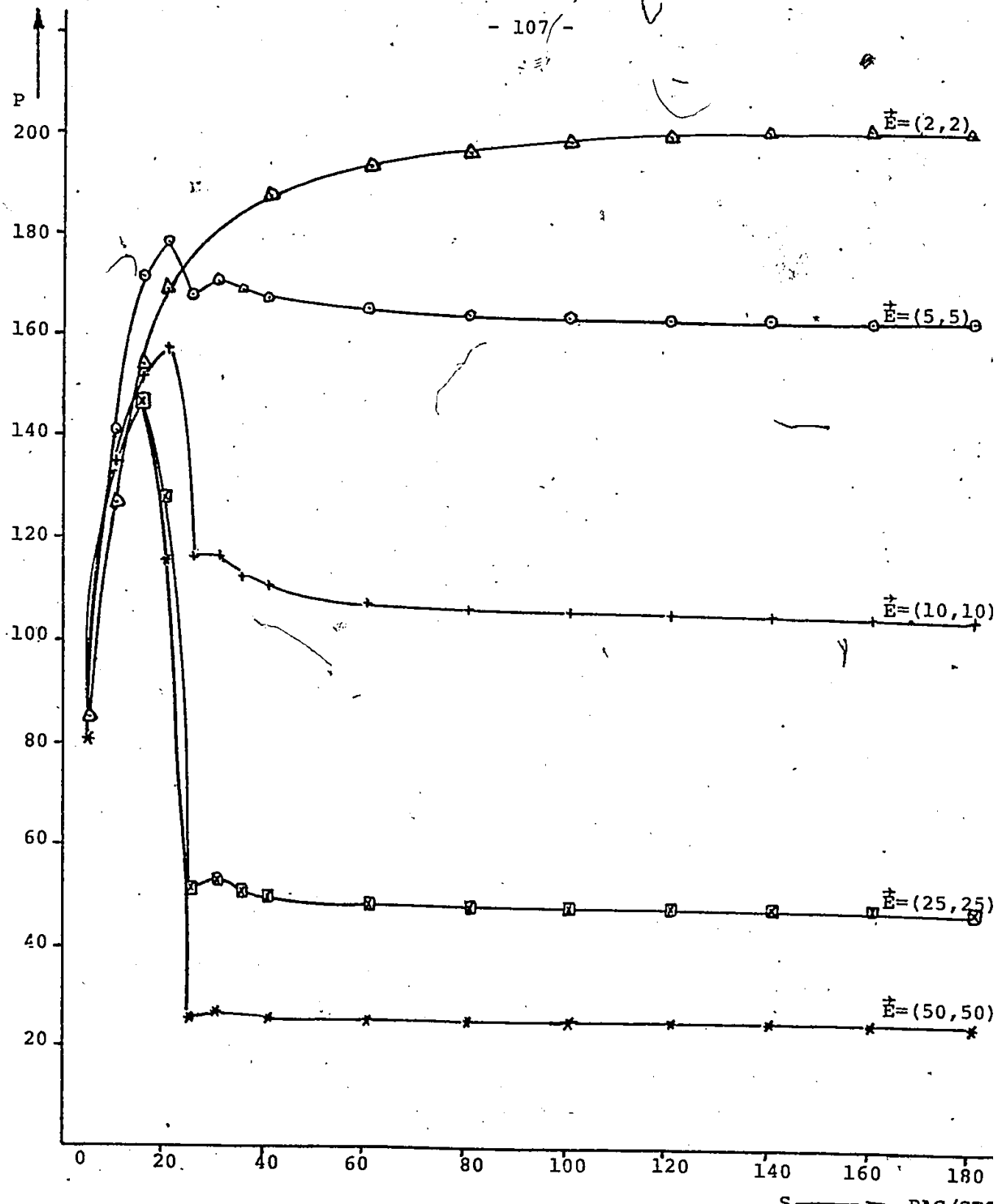


Fig. 4.9 Network Power Versus Class Traffic Arrival Rate

that window sizes that exceed 5 are inferior as they always give smaller power than for E = (5,5) at almost any traffic loading. If the throughput and delay requirements are not too stringent, such that a window size less than 5 can be used, the network will be operating with a power that is monotonically increasing to a steady-state value as the arrival rates build up.

The second network example is shown in Fig. 4.10. This network is quite similar to the one just considered. Four message classes are defined in this case, the routing chains of classes 1 and 2 are identical to those of the previous network. Class 3 messages originate, with Poisson rate S_3, at Vancouver and are routed to Montréal via Edmonton and Winnipeg. Class 4 messages originate, with Poisson rate S_4 at Toronto and is routed directly to Winnipeg. The specifications for channel capacity and queue discipline are the same as for the previous network. The messages are exponentially distributed with mean length 1000 bits for both classes.

Fig. 4.11 shows the queueing model of the network. The model consists of 4 chains and 11 queues.

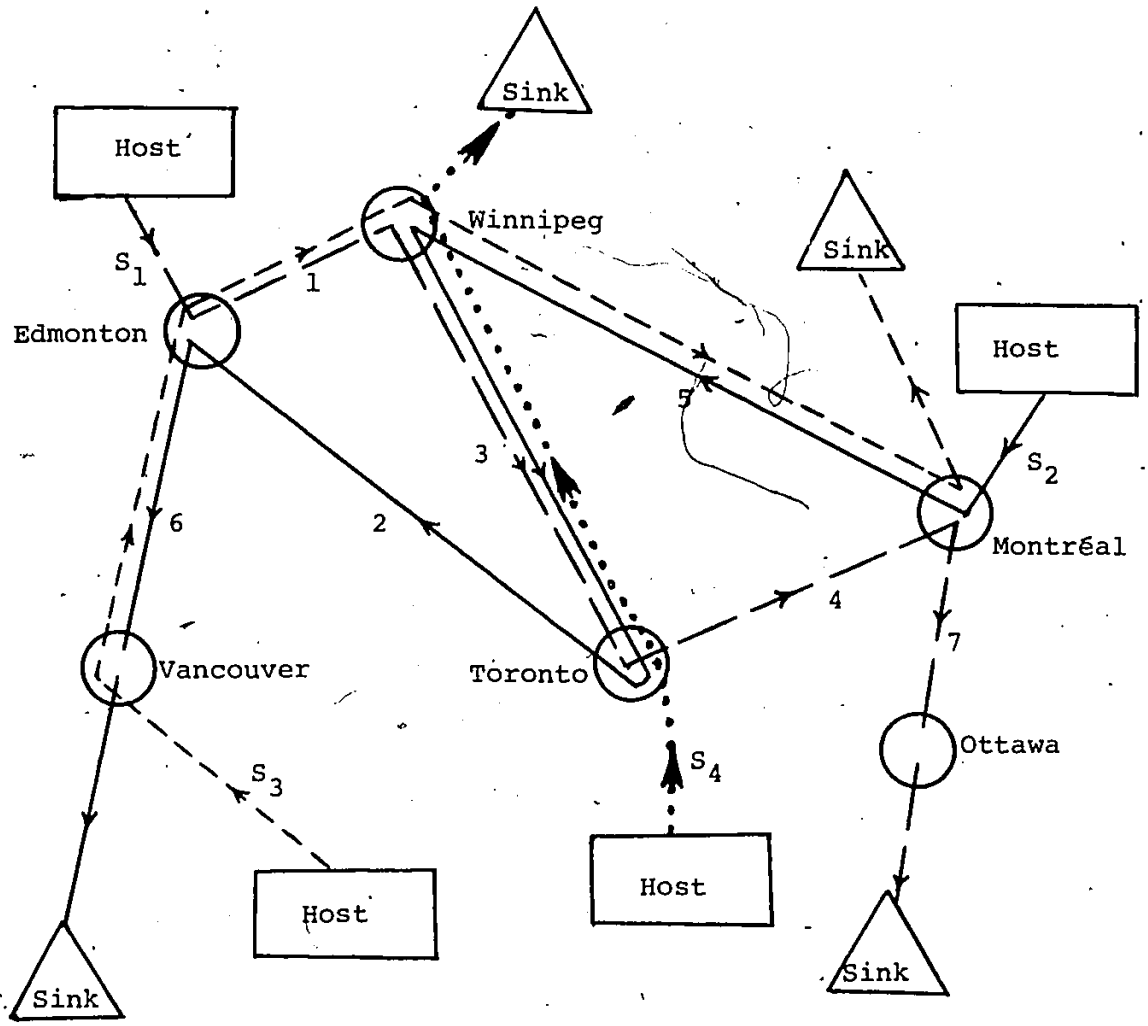


Fig. 4.10 A Network Example in Four Classes

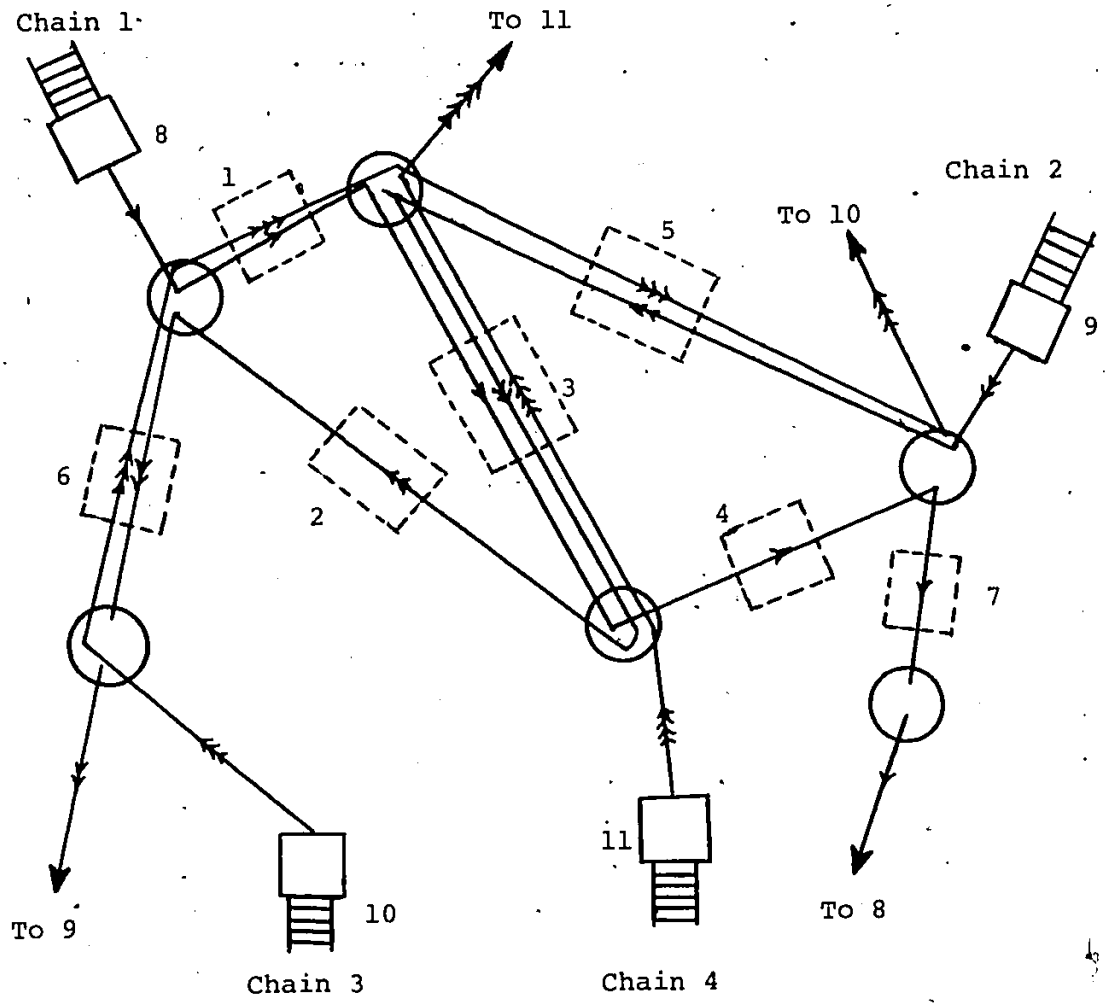


Fig. 4.11 Closed Chain Model of Example Network in Four Classes

Queues 1 to 7 represent the 7 channels interconnecting the nodes. Queues 8 through 11 model the sources of the four chains.

The routing and network parameters are quite asymmetrical for this network example. Moreover there is greater interaction among different class traffic than for the two-class network just studied. Table 4.12 summarizes the influence of the class arrival rates on the optimal window sizes. Randomly chosen customer arrival rates have been considered for the four sources. It is observed that for a given total network traffic arrival rate, the network power is apparently maximized if the traffic arrival rates are in the ratio of the virtual channel capacities. It also appears that the magnitude of optimal network power increases as the total network traffic arrival rate increases. This has already been predicted by the two-class example network just described.

The increased interaction among different class traffic renders Kleinrock's prediction [52] of window settings (4,4,3,1), based on the number of hops in each

Table 4.12

Effect of Variations in Traffic Arrival Rates on Optimal Window Settings for a 4-Class Network Example

S_1	S_2	S_3	S_4	$\sum_{i=1}^{4} S_i$	E_{op}	P_{op}	P_{4431}
6	6	6	12	30	1 1 1 4	352	279
9.957	4.419	7.656	7.968	30	2 1 2 5	286	253
17.61	3.56	3	5.83	30	3 3 3 2	225	210
12.50	12.50	12.50	25	62.5	1 1 1 4	543	320
21.24	9.86	18.85	12.55	62.5	1 1 1 4	383	271
33.59	1.70	24.15	3.06	62.5	2 1 3 14	253	228
20	20	20	40	100	1 1 1 2	599	277
28.18	38.02	2.87	30.93	100	1 1 2 3	520	250

where,

S_i = Class i arrival rate in msg/sec, i= 1,2,3,4.

E_{op} = Optimal window sizes.

P_{op} = Optimal network power.

P_{4431} = Network power corresponding to window settings (4 4 3 1).

class, a very poor estimate of the true optimal window settings. This is clearly seen when P_{op} is compared to P_{4431} for given arrival rates in Table 4.12.

4.5 COMMENTS

It is recalled that in the initialization of the window sizes for the WINDIM algorithm, the window-setting vector $(\beta_1, \beta_2, \ldots, \beta_R)$, where β_r is the number of hops in chain r, has been used.

For the 2-class network example, there is only little interaction between the class traffic in the two routing chains. Tables 4.7 and 4.8 indicate that although (β_1, β_2) is not the optimal choice, the corresponding network power is comparable to that for optimal window sizes. However, for the 4-class network example, where there is substantial interaction among different class traffic, the said initial window setting vector does not even crudely approximate to the designation for optimal settings. It is therefore apparent that the initial window settings approach the optimal choice asymtotically as the interaction among different class traffic is

gradually decreased. This conclusion is confirmed by consideration of Kleinrock's original network model [52].

The fundamental assumption in Kleinrock's model is that the average network delay be modelled as a β-hop network with each hop modelled by an M/M/1 queue and with an instantaneous end-to-end acknowledgement. Accordingly, the total average network delay is given by

$$T(\lambda) = \frac{\beta}{\mu - \lambda(\xi)} \qquad (4.21)$$

where μ = network capacity in msg/sec.

$\lambda(\xi)$ = network throughput in msg/sec.

ξ = input traffic rate applied to the network in msg/sec.

(4.21) assumes that the β hops contribute identically to the total average network delay. The equation may apply to individual routing chains as long as there is no interaction among the different class traffic, in which case the optimal window settings are precisely $(\beta_1, \beta_2, \ldots, \beta_R)$ for an R-class network

$$\bar{\omega} = \frac{\beta \lambda(\xi)}{\mu - \lambda(\xi)} \quad , \quad \text{with} \quad \begin{array}{l} \bar{\omega} = \text{window size} \\ = \lambda(\xi) T(\lambda) \end{array}$$

$$\text{or} \quad \lambda(\xi) = \frac{\omega}{\beta + \omega} \mu$$

For optimality, $\lambda(\xi) = \mu/2$ [52] , implying that $\omega = \beta$.

CHAPTER 5

CONCLUSIONS

The exact analytical model of separable queueing networks, and its attendant overwhelming computational requirements present a formidable problem rather than an expedience to the selection of good window settings for computer-communication networks in an end-to-end flow control environment. The new heuristic algorithm, WINDIM, presented in this thesis provides an efficient and a reliable method for resolving the aforementioned window dimensioning problem.

WINDIM essentially comprises the well-known multidimensional search technique called Pattern Searching. The required function evaluations are afforded by the heuristics of the mean-value analytical model recently developed by M. Reiser. The computational and storage requirements of WINDIM are modest compared with a similar search built around the exact analytical model (e.g. using the traditional convolution algorithm approach).

The algorithm has been implemented in APL and used in dimensioning the end-to-end windows of some simple network examples. Results obtained from these examples are undoubtedly conducive to the good selection of end-to-end windows for larger networks. Although the network examples considered comprise only FCFS queues with a single exponential server for each queue, WINDIM can be readily extended to analyse networks with LCFSPR, PS, IS or other work-conserving queue disciplines.

It should, however, be noted that in a practical analytical model, the flow control protocols necessarily reflect the natural attribute and limitation of a finite buffer storage in any store-and-forward switching node. The exact modelling of the local flow control scheme is hitherto unsuccessful. Nevertheless, the recent developments [44] in the analysis of networks with end-to-end, local and isarithmic flow controls is noteworthy. It is therefore imperative to continue the heuristic development effort, delineated in this thesis, to expedite the dimensioning of end-to-end, local, and possibly, the isarithmic flow control windows.

REFERENCES

[1] I.T. Frisch and H. Frank, "Computer Communications ----- How We Got Where We Are", Proc. AFIPS Nat. Comput. Conf., Vol. 44, May 19-22, 1975, pp. 109-117.

[2] L. Kleinrock, Queueing Systems, Vol. 2: Computer Applications , Wiley Interscience, New York, 1976.

[3] J.M. McQuillan, W.R. Crowther, B.P. Cossell, D.C. Walden and F.E. Heart, "Improvements in the Design and Performance of the ARPA Network", AFIPS Conference Proceedings, 1972 Fall Joint Computer Conference, Vol. 41, pp. 741-754.

[4] M. Reiser, "A Queueing Network Analysis of Computer Communication Networks with Window Flow Control", IEEE Trans. on Comm., Vol. COM-27, August 1979, pp. 1199-1209.

[5] A. Giessler, J. Hanle, A. Konig and E. Pade, "Free Buffer Allocation --- An Investigation by Simulation", Computer Networks,Vol. 2, 1978, pp. 191-208.

[6] L. Kleinrock, "Principles and Lessons in Packet Communications", Proceedings of the IEEE, Vol. 66, No. 11, Nov. 1978, pp. 1320-1329.

[7] S.S. Lam and M. Reiser, "Congestion Control of Store-and-Forward Networks by Input Buffer Limits", Proceedings of NTC 1977, Los Angeles, 1977.

[8] D.W. Davis, "The Control of Congestion in Packet Switching Networks", IEEE Trans. on Comm., Vol. COM-20, June 1972, pp. 546-550.

[9] M. Schwartz, Computer-Communication Network Design and Analysis, Prentice-Hall Inc., Englewood Cliffs, N.J., 1977.

[10] R. Kahn and W. Crowther, "Flow Control in a Resource-Sharing Computer Network", IEEE Trans. on Comm., Vol. COM-20, June 1972, pp. 539-546.

[11] Opderbeck and L. Kleinrock, "The Influence of Control Procedures on the Performance of Packet-Switched Networks", Proc. National Telecomm. Conf., San Diego, California, Dec. 1974.

[12] W. Chou and M. Gerla, "A Unified Flow and Congestion Control Model for Packet Networks", Proc. 3rd Intern. Conf. on Comp. Comm., Toronto, pp. 475-482.

[13] L.R. Tymes, "TYMNET --- A Terminal Oriented Communication Network", IFIPS Conf. Proc., Vol. 38, 1971, pp. 211-216.

[14] J.R. Pierce, "How Far Can Data Loops Go?", IEEE Trans. Comm., Vol. COM-20, pp. 527-530, June 1972.

[15] J.F. Hayes and D.N. Sherman, "Traffic Analysis of a Ring-Switched Data Transmission Service", Bell Systems Technical Journal, Vol. 50, Nov. 1971, pp. 2947-78.

[16] D.W. Davies, "A Review of Computer Communications Technology", Computer Communication Networks, Noordhoff International Publishing, 1975, pp. 1-17.

[17] L. Kleinrock, Communication Nets, New York, McGraw-Hill, 1964.

[18] H. Kobayashi and A.G. Konheim, "Queueing Models for Computer Communications System Analysis", IEEE Trans. Comm., Vol. COM-25, January 1977, pp. 2-29.

[19] M. Reiser and C.H. Sauer, "Queueing Network Models: Methods of Solution and Their Program Implementation", IBM T.J. Watson Research Center, Yorktown Heights, New York, Research Report RC6109, July 1976.

[20] D.R. Cox, "A Use of Complex Probabilities in the Theory of Stochastic Processes", Proc. Cambridge Phil. Sc., Vol. 51, 1955, pp. 313-319.

[21] H. Kobayashi, "System Design and Performance Analysis Using Analytic Models", IBM Research Report RA 75, Yorktown Heights, New York, December 1975.

[22] R. Hooke and T.A. Jeeves, "'Direct Search' Solution of Numerical and Statistical Problems", J. Assn. Comp., Vol. 8, April 1961, pp. 212-29.

[23] M. Reiser and S.S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks", IBM Research Report RC 7023, Yorktown Heights, N.Y., 1978.

[24] M. Reiser and H. Kobayashi, "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms", IBM Journal of Research and Development, Vol. 19, 1975, pp. 285-294.

[25] J.P. Buzen, "Computational Algorithms for Closed Queueing Networks with Exponential Servers", CACM, Vol. 16, September 1973, pp. 527-531.

[26] B. Pittel, "Closed Exponential Networks of Queues with Blocking, the Jackson Type Stationary Distribution and its Asymptotic Analysis", IBM Research Report RC 6174, Yorktown Heights, U.S.A., 1976.

[27] S.S. Lam, "Queueing Networks with Population Size Constraints", IBM J. Res. and Devel., Vol. 21, #4, July 1977, pp. 370-378.

[28] V.L. Wallace, "Toward an Algebraic Theory of Markovian Networks", Proc. Symp. Computer Communications Networks and Teletraffic, Polytechnic Press of the Polytechnic Inst. of Brooklyn, April 1972, pp. 397-407.

[29] V.L. Wallace and R.S. Rosenberg, "RQA-1, The Recursive Queue Analyser", Tech. Rep. 2, Systems Eng. Lab., Univ. Michigan, Ann Arbor, Michigan, Feb. 1966.

[30] U. Herzog, L. Woo and K.M. Chandy, "Solution of Queueing Problems by a Recursive Technique", IBM J. Res. Develop., Vol. 19, May 1975, pp. 209-232.

[31] K.M. Chandy, U. Herzog and L. Woo, "Approximate Analysis of General Queuing Networks", IBM J. Res. Develop., Vol. 19, Jan. 1975, pp. 43-49.

[32] H. Kobayashi, "Application of the Diffusion Approximation to Queueing Networks", J. ACM, Vol. 21, April 1974, pp. 316-318 (Part I); and J. ACM, Vol. 21, July 1974, pp. 459-469 (Part II).

[33] M. Reiser and H. Kobayashi, "On the Accuracy of the Diffusion Approximation for Some Queuing Systems", IBM J. Res. Develop., Vol. 18, March 1974, pp. 110-124.

[34] M. Reiser, "Numerical Methods in Separable Queueing Networks", Studies in the Management Sciences, Vol. 7, 1977, pp. 113-142.

[35] S.S. Lam, "On an Extension of Moore's Results for Closed Queueing Networks", IBM Research Report, April 9, 1975; also IBM J. Res, Develop., Vol. 21, #4, July 1977, pp. 384-387.

[36] F. Baskett, K.M. Chandy, R.R. Muntz and F.G. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", J. ACM, Vol. 22, 1975, pp. 248-260.

[37] J.R. Jackson, "Jobshop-like Queueing Systems", Management Science, Vol. 10, October 1963, pp. 131-142.

[38] W.T. Gordon and G.F. Newell, "Closed Queueing Systems with Exponential Servers", Operations Research, Vol. 15, April 1967, pp. 252-265.

[39] F. Baskett, "Mathematical Models of Multiprogrammed Systems", Ph. D. Diss., Comptr. Ctr. Rep. TSN-17, Comptr. Sci. Dep., U. of Texas at Austin, Austin, Tex., 1970.

[40] R.R. Muntz, "Poisson Departure Processes and Queueing Networks", IBM Research Report RC 4145, Dec., 1972.

[41] F.R. Moore, "Computational Model of a Closed Queueing Network with Exponential Servers", IBM J. Res. Dev.,Vol. 16, November 1972, pp. 567-572.

[42] M. Reiser and H. Kobayashi, "Numerical Solution of Semiclosed Exponential Server Queueing Networks", Proc. 7th Asilomar Conf. on Circuits, Systems and Computers, Monterey, Cal., Nov. 1973, pp. 308-312.

[43] M.C. Pennotti, M. Schwartz, "Congestion Control in Store and Forward Tandem Links", IEEE Trans. Comm., Vol. COM-23, December 1975, pp. 1434-1443.

[44] N.D. Georganas, "Analysis of Packet-Switched Computer Communication Networks with Multilevel Flow-Control", IBM Technical Report TR 0498, C.E.R. La Gaude, France, Dec. 1977; also, Proceedings, COMPCON 78 (Fall), Washington, D.C., Sept. 5-8, 1978, pp. 4-11.

[45] N.D. Georganas, "Numerical Solution of Queueing Networks with Multiple Semi-Closed Chains", Proc. IEE, Vol. 126, #3, March 1979, pp. 229-231.

[46] P.J. Burke, "The Output of a Queueing System", Operations Research. Vol. 4, pp.699-704, 1956.

[47] J.R. Jackson, "Networks of Waiting Lines", Operations Research, Vol.5, pp. 518-521, 1957.

[48] K.M. Chandy, J.H. Howard Jr., D.F. Towsley, "Product Form and Local Balance in Queueing Networks", Journal of the Association for Computing Machinery, Vol. 24, No. 2, April 1977, pp. 250-263.

[49] R.R. Muntz and J. Wong, "Efficient Computational Procedures for Closed Queueing Network Models", Proceedings of the 7th Hawaii International Conference on System Sciences, Honolulu, Hawaii, pp. 33-36, January 8-10, 1974.

[50] M. Reiser and H. Kobayashi, "Numerical Methods in Queueing Networks", Proc. Camp. Sci. and Statistics, 8th Annual Symp. on the interface, Univ. of California, Los Angeles, Feb. 1975.

[51] J.Y.K. Chan and N.D. Georganas, "Dimensioning of Message-Switched Computer Communications Networks with End-to-End Window Flow Control", Proc. 6th ACM/IEEE Data Communication Symposium, Pacific Grove, California, Nov. 27-29, 1979; also,Computer Communications, IPC Science and Technology Press Limited, Surrey, England (to appear).

[52] L. Kleinrock, "On Flow Control in Computer Networks", Proc. of International Conf. on Comm., Toronto, June 1978, pp. 27.2.1 - 27.2.5.

APPENDIX : APL PROGRAMS

```
▽ WINDIM
[1] ⍝  THIS MAIN PROGRAM EMBODIES THE WELL-KNOWN PATTERN
[2] ⍝  SEARCH TECHNIQUE. THE REQUISITE FUNCTION EVALUATIONS
[3] ⍝  ARE FURNISHED BY FUNCTION SUBPROGRAM 'FCT'.
[4] ⍝
[5] ⍝  PATTERN SEARCH ESSENTIALLY COMPRISES EXPLORATIVE MOVES TO
[6] ⍝  ESTABLISH BASE POINTS, FOLLOWED BY PATTERN MOVES.
[7] ⍝  A NEW BASE POINT IS ESTABLISHED IF THE VALUE OF THE OBJECTIVE
[8] ⍝  FUNCTION AT THAT POINT IS LOWER THAN THAT AT THE PREVIOUSLY
[9] ⍝  OBTAINED BASE POINT, OTHERWISE THE NEW BASE POINT IS REJECTED.
[10] ⍝  IF A NEW ROINT CANNOT BE LOCATED, THE STEP-SIZE IS
[11] ⍝  REDUCED. A NEW PATTERN IS DEVELOPED UNLESS THE STEP-SIZE
[12] ⍝  FALLS BELOW SOME PRESCRIBED VALUE. IN THAT CASE THE
[13] ⍝  OPTIMAL WINDOW-SETTINGS ARE GIVEN BY THE COORDINATES OF
[14] ⍝  THE LATEST BASE POINT OBTAINED.
[15] ⍝
[16] ⍝  INPUT PARAMETERS:
[17] ⍝    X0      = INITIAL WINDOW SIZES FOR END-TO-END CONTROL (AN R-VECT
[18] ⍝    Y       = INITIAL STEP SIZE FOR SEARCH ADVANCEMENT (AN R-VECTOR)
[19] ⍝    KMAX    = DESIRED MAXIMUM NUMBER OF HALVINGS OF STEP SIZE
[20] ⍝
[21] ⍝  ASSOCIATE INPUT PARAMETERS FOR FUNCTION SUBPROGRAM
[22] ⍝ 'FCT':
[23] ⍝
[24] ⍝    R      = NUMBER OF CLOSED, CYCLIC ROUTING CHAINS.
[25] ⍝    L      = TOTAL NUMBER OF QUEUES IN THE NETWORK.
[26] ⍝    C      = AN R×L LINK CAPACITY MATRIX.  C[I;J] IS THE LINK
[27] ⍝             CAPACITY AT QUEUE J OF THE ITH CHAIN IN KBITS/SEC.
[28] ⍝   MU      = AN R×L MATRIX. (1÷MU[I;J]) IS THE MEAN PACKET LENGTH
[29] ⍝             IN KBITS FOR A CLASS I PACKET IN QUEUE J.
[30] ⍝    Q      = AN R×L LOGICAL MATRIX.  Q[I;J] EQUALS 1 IF CHAIN I
[31] ⍝             USES QUEUE J, AND ZERO OTHERWISE.
[32] ⍝    E      = AN R-VECTOR OF WINDOW-SIZES FOR END-TO-END CONTROL.
[33] ⍝
[34] ⍝ N.B.: FOR REENTRANT QUEUE Z (FROM SINK TO SOURCE) OF CLASS I,
[35] ⍝       C[I;Z]×MU[I;Z]=S[I],       WHERE S[I] IS THE EXTERNAL
[36] ⍝      TRAFFIC RATE IN PACKETS/SEC FOR CLASS I.
[37] ⍝      C[I;Z] HAS BEEN SET TO UNITY FOR CONVENIENCE.
[38] ⍝
[39] TSTART←⌷21
[40] XDIM←300,R
[41] FXA←FXCMP←300⍴0.
[42] DIM←R
[43] XCMP←XDIM⍴0
[44] XPT←1,
[45] XCMP[XPT;]←X0
[46] K←0
```

```
[47] FXCMP[XPT]←FXA0R←FCT X0
[48] BR11: N←R⍴0
[49] ⍝
[50] ⍝ TERMINATE THE PATTERN SEARCH IF K>KMAX
[51] ⍝
[52] →(K>KMAX)/BR133
[53] ⍝
[54] ⍝ LOCAL EXPLORATION ABOUT INITIAL BASE POINT X0
[55] ⍝
[56] FXA0←FXA0R
[57] I←1
[58] LP11: XP←XM←X0
[59] XP[I]←XP[I]+Y[I]
[60] FXP←FLOC XP
[61] →(FXP≥0)/BR16
[62] FXCMP[XPT]←FXP←FCT XP
[63] BR16: →(FXP≥FXA0)/BR22
[64] N[I]←N[I]+1
[65] X0←XP
[66] FXA0←FXP
[67] →BR33
[68] BR22: XM[I]←XM[I]-Y[I]
[69] FXM←FLOC XM
[70] →(FXM≥0)/BR27
[71] FXCMP[XPT]←FXM←FCT XM
[72] BR27: →(FXM≥FXA0)/BR33
[73] N[I]←N[I]-1
[74] X0←XM
[75] FXA0←FXM
[76] BR33: I←I+1
[77] →(I≤DIM)/LP11
[78] ⍝
[79] ⍝ IF LOCAL EXPLORATION FAILS TO LOCATE NEW BASE POINT,
[80] ⍝  REDUCE STEP SIZE AND CHECK IF SEARCH NEEDS BE CONTINUED.
[81] ⍝
[82] ⍝ ELSE SET THE NEW BASE POINT FROM LOCAL EXPLORATION
[83] ⍝
[84] →((+/N)≠0)/BR44
[85] K←K+1
[86] Y←.5×Y
[87] →BR11
[88] BR44: XA0←X0
[89] ⍝
[90] ⍝ MAKE THE FIRST PATTERN MOVE FROM THE NEW BASE POINT
[91] ⍝  TO POINT 'XT'
[92] ⍝
[93] XT←XA0+N×Y
[94] T←1
[95] ⍝
[96] ⍝ LOCAL EXPLORATION ABOUT 'XT' TO LOCATE NEW BASE POINT
[97] ⍝
```

```
[98] →(FXT≥0)/BR49
[99] FXCMP[XPT]←FXT←FCT XT
[100] BR49: FXA[T]←FXT
[101] I←1
[102] LP33: XP←XM←XT
[103] XP[I]←XP[I]+Y[I]
[104] FXP←FLOC XP
[105] →(FXP≥0)/BR52
[106] FXCMP[XPT]←FXP←FCT XP
[107] BR52: →(FXP≥FXA[T])/BR55
[108] N[I]←N[I]+1
[109] XT←XP
[110] FXA[T]←FXP
[111] →BR66
[112] BR55: XM[I]←XM[I]-Y[I]
[113] FXM←FLOC XM
[114] →(FXM≥0)/BR60
[115] FXCMP[XPT]←FXM←FCT XM
[116] BR60: →(FXM≥FXA[T])/BR66
[117] N[I]←N[I]-1
[118] XT←XM
[119] FXA[T]←FXM
[120] BR66: I←I+1
[121] →(I≤DIM)/LP33
[122] →(T≠1)/BR88
[123] →(FXA[1]<FXA0)/BR111
[124] →BR99
[125] ⍝
[126] ⍝ IF VALUE OF OBJECTIVE FUNCTION IS NOT REDUCED AT
[127] ⍝  NEW BASE POINT, REJECT THE NEW BASE POINT
[128] ⍝
[129] BR99: X0←XT-N×Y
[130] FXA0R←FLOC X0
[131] →(FXA0R≥0)/BR105
[132] FXCMP[XPT]←FXA0R←FCT X0
[133] BR105: →BR11
[134] ⍝ IF NEW BASE POINT CANNOT BE LOCATED, REDUCE STEP-SIZE AND
[135] ⍝  CHECK IF SEARCH NEEDS BE CONTINUED
[136] ⍝
[137] ⍝ ELSE MAKE A PATTERN MOVE FOLLOWED BY ANOTHER LOCAL
[138] ⍝  EXPLORATION FOR NEW BASE POINT
[139] ⍝
[140] BR111: →((+/N)≠0)/BR122
[141] Y←.5×Y
[142] K←K+1
[143] →BR11
[144] BR122: XT←XT+N×Y
[145] T←T+1
[146] →LP22
[147] BR133: 'THE OPTIMAL WINDOW-SIZES = ';X0
[148] TEND←I21
```

```
[149] 'SECS CPU TIME = ';(TEND-TSTART)÷60
```

```
▽ F←FCT E;I;K;TSTART;TEND
[1]  ⍝  THIS FUNCTION SUBPROGRAM IS CALLED BY MAIN PROGRAM
[2]  ⍝   'WINDIM'. IT ANALYSES QUEUEING NETWORKS WITH WINDOW FLOW
[3]  ⍝  CONTROL,  THE NETWORK TO BE STUDIED MUST BELONG TO ONE OF THE
[4]  ⍝  FOLLOWING CLASSES:
[5]  ⍝   (1) FIFO SCHEDULING AND ALL CHAINS AT QUEUE I EXPERIENCE THE
[6]  ⍝       SAME EXPONENTIAL SERVICE RATE.
[7]  ⍝   (2) PS OR LCFS PREEMPTIVE-RESUME SCHEDULING WITH A PHASE-TYPE
[8]  ⍝       SERVICE-TIME DISTRIBUTION.
[9]  ⍝
[10] ⍝  INPUT PARAMETERS:
[11] ⍝    R      = NUMBER OF CLOSED, CYCLIC ROUTING CHAINS.
[12] ⍝    L      = TOTAL NUMBER OF QUEUES IN THE NETWORK.
[13] ⍝    C      = AN R×L LINK CAPACITY MATRIX.  C[I;J] IS THE LINK
[14] ⍝             CAPACITY AT QUEUE J OF THE ITH CHAIN KBITS/SEC.
[15] ⍝    MU     = AN R×L MATRIX. (1÷MU[I;J]) IS THE MEAN PACKET LENGTH
[16] ⍝             IN KBITS FOR A CLASS I PACKET IN QUEUE J.
[17] ⍝    Q      = AN R×L LOGICAL MATRIX.  Q[I;J] EQUALS 1 IF CHAIN I
[18] ⍝             USES QUEUE J, AND ZERO OTHERWISE.
[19] ⍝    E      = AN R-VECTOR OF WINDOW-SIZES FOR END-TO-END CONTROL.
[20] ⍝    TOLER = PRESCRIBED STOPPING CRITERION FOR THE PROGRAM.
[21] ⍝
[22] ⍝  N.B.:  FOR REENTRANT QUEUE Z (FROM SINK TO SOURCE) OF CLASS I,
[23] ⍝          C[I;Z]×MU[I;Z]=S[I],      WHERE S[I] IS THE EXTERNAL
[24] ⍝          TRAFFIC RATE IN PACKETS/SEC FOR CLASS I.
[25] ⍝        C[I;Z] HAS BEEN SET TO UNITY FOR CONVENIENCE.
[26] ⍝
[27] TSTART←⎕TS
[28] NMCLS←LMBDA←DLAY←R⍴0
[29] TN←LTAU←L⍴0
[30] DD←R,R,L
[31] DN←R,L
[32] TQS←TSVSC←TSV←NRNEG←NM←DN⍴0
[33] NRMNS←DD⍴0
[34] CLSR←⍳Q
[35] 'FOR WHAT FOLLOWS, E = ';E
[36] ITER←0
[37] ⍝
[38] ⍝ INITIALIZE MEAN QUEUE LENGTHS AND THROUGHPUTS
[39] ⍝
[40] K←1
[41] LP11:TSV[K;]←Q[K;]÷MU[K;]×C[K;]
[42] TMAX←⌈/TSV[K;]
[43] NM[K;TSV[K;]⍳TMAX]←E[K]
[44] LMBDA[K]←E[K]÷(+⌿E)×TMAX
[45] K←K+1
[46] →(K≤R)/LP11
[47] ⍝
```

```
[48] ⍝ COMPUTE MEAN QUEUE LENGTHS WITH ONE LESS CUSTOMER
[49] ⍝
[50] BR11: K←1
[51] LP22: I←1
[52] LP33: TERM1←+/CLSR[I;]×LMBDA×TSV[;I]
[53] TERM2←CLSR[I;K]×LMBDA[K]×TSV[K;I]
[54] LTAU[I]←TERM1-TERM2
[55] I←I+1
[56] →(I≤L)/LP33
[57] TSVSC[K;]←TSV[K;]÷1+(TSV[K;]×LMBDA[K])-LTAU
[58] NRMNS[K;;]←NM
[59] K←K+1
[60] →(K≤R)/LP22
[61] K←1
[62] LP44: W←1
[63] NSC←L⍴0
[64] LP55: TSC←TSVSC[K;]×1+NSC
[65] NSCM1←NSC
[66] NSC←TSC×W÷+/Q[K;]×TSC
[67] W←W+1
[68] →(W≤E[K])/LP55
[69] NRMNS[K;K;]←NM[K;]-NSC-NSCM1
[70] K←K+1
[71] →(K≤R)/LP44
[72] ⍝
[73] ⍝ COMPUTE MEAN QUEUEING TIMES
[74] ⍝
[75] K←1
[76] LP66: I←1
[77] LP77: TN[I]←+/CLSR[I;]×TSV[;I]×NRMNS[K;;I]
[78] I←I+1
[79] →(I≤L)/LP77
[80] TQS[K;]←Q[K;]×TSV[K;]+TN
[81] K←K+1
[82] →(K≤R)/LP66
[83] LPREV←LMBDA
[84] ITER←ITER+1
[85] ⍝
[86] ⍝ COMPUTE CLASS AND NETWORK THROUGHPUTS
[87] ⍝
[88] K←1
[89] LP88: LMBDA[K]←E[K]÷+/TQS[K;]
[90] NM[K;]←LMBDA[K]×TQS[K;]
[91] K←K+1
[92] →(K≤R)/LP88
[93] TH←+/LMBDA
[94] CRIT←(+/(LMBDA-LPREV)*2)*0.5
[95] 'CRIT = ';CRIT
[96] →(CRIT>TOLER)/BR11
[97] ⍝
[98] ⍝ COMPUTE MEAN QUEUE LENGTHS AND MEAN CLASS AND NETWORK DELAYS
```

```
[99] ∆
[100] K←1
[101] LP99: DLAY[K]←+/(TQS[K;]×(C[K;]≠1))
[102] NMCLS[K]←+/(NM[K;]×(C[K;]≠1))
[103] K←K+1
[104] →(K≤R)/LP99
[105] D←(+/NMCLS)÷+/LMBDA
[106] F←D÷TH
[107] 'FOR WINDOW-SIZES = ';E
[108] 'CLASS THROUGHPUTS = ';LMBDA
[109] 'CLASS DELAYS = ';DLAY
[110] 'NETWORK THROUGHPUT = ';TH
[111] 'AVG. NETWORK DELAY = ';D
[112] 'POWER = ';F
[113] TEND←□TS
[114] CPU←(TEND-TSTART)÷60
[115] 'EXECUTION TIME = ';CPU;' SEC.'
```

```
▽ FSTR←FLOC X;I
[1] ⍝  THIS FUNCTION SUBPROGRAM, IF CALLED, WILL STORE EVALUATED
[2] ⍝  FUNCTION VALUES IN THE MATRIX XCMP.
[3] ⍝
[4] FSTR←¯10
[5] I←1
[6] LP11: →(∨/XCMP[I;]=X)/BR11
[7] I←I+1
[8] →(I≤XPT)/LP11
[9] XPT←XPT+1
[10] XCMP[XPT;]←X
[11] →0
[12] BR11: 'FOR WHAT FOLLOWS, E = ';X
[13] FSTR←FXCMP[I]
[14] 'POWER = ';FSTR
[15] ' (THE NECESSARY COMPUTATIONS WERE DONE PREVIOUSLY)'
▽
```