Queueing Network Models of Packet Switching Networks Part 2: Networks with Population Size Constraints *

S.S. Lam
Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, U.S.A.

J.W. Wong
Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada

Received 2 February 1981
Revised 2 April 1982

An overview of the application of product-form queueing networks to the performance analysis of store-and-forward packet communication networks is presented. Queueing network models with closed chains and other forms of population size constraints are considered. The modeling and analysis of networks with window flow controlled virtual channels are described first. Models for the performance analysis of strategies for nodal buffer management and permit-oriented network congestion control are then presented.

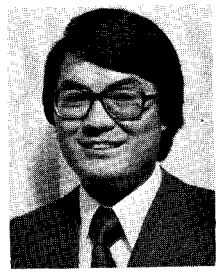
Keywords: Queueing Network Models, Packet Switching Networks, Flow Control, Congestion Control, Buffer Management, Resource Sharing, Computational Algorithms.

* The work of S.S. Lam was supported by National Science Foundation Grant No. ECS78-01803. The work of J.W. Wong was supported by the Natural Sciences and Engineering Research Council of Canada.

North-Holland Publishing Company
Performance Evaluation 2 (1982) 161–180

1. Introduction

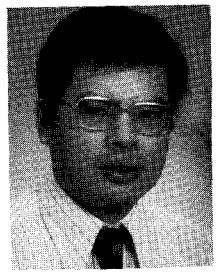
Recent efforts on the application of product-form queueing network models [1–4] to the performance analysis of store-and-forward packet switching networks are reviewed in this paper and a companion paper [5]. Open queueing network models are covered in the companion paper. The class of open network models is characterized by



Simon S. Lam received the B.S.E.E. degree (with Distinction) from Washington State University, Pullman, Washington, in 1969, and the M.S. and Ph.D. degrees in engineering from the University of California at Los Angeles, in 1970 and 1974, respectively.

From 1972 to 1974, he was with the ARPA Network project at UCLA and did research on satellite packet communication. From 1974 to 1977, he was a research staff member with the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, where he worked on performance analysis problems of packet switching networks, SNA and satellie networks. Since September 1977, he has been with the University of Texas at Austin where he is now Associate Professor of Computer Sciences. His current research interests include modeling and analysis of computer systems, networks and communication protocols.

At the University of California at Los Angeles, he held a Phi Kappa Phi Fellowship from 1969 to 1970, and a Chancellor's Teaching Fellowship from 1969 to 1973. In 1975, he received the IEEE Leonard G. Abraham Award for the best paper in the field of Communications Systems. Simon is a member of Tau Beta Pi, Sigma Tau, Phi Kappa Phi, Pi Mu Epsilon and the Association for Computing Machinery, and a senior member of IEEE.



J.W. Wong received the B.S. in engineering, the M.S. and Ph.D. degrees in computer science from the University of California at Los Angeles in 1970, 1971, and 1975, respectively. He is currently an Associate Professor of Computer Science at the University of Waterloo, Waterloo, Ontario, Canada.

explicit formulas for the performance measures of mean delay and throughput that are of interest. Queueing networks with closed chains and other forms of chain population size constraints are useful as models for the analysis of various network flow and congestion control mechanisms and buffer management strategies. This class of queueing network models is characterized by algorithmic solutions for the mean delay and throughput performance measures. These models constitute the subject of this paper.

The key modeling assumptions, notation and definitions needed in this paper are presented next. The reader is referred to [5] for a more detailed treatment.

The basic unit of data transfer in the communication networks being modeled is a packet; it corresponds to a customer in the queueing network models. A store-and-forward packet switching network consists of a set of switching nodes interconnected by full-duplex communication channels (see Fig. 1). The function of the communication network is to transport packets from their sources to their sinks. Each packet traverses from its source node to its destination node through a series of intermediate nodes and communication channels along a selected path (or route). Queues of packets are formed inside the switching nodes. The key assumption necessary for the application of queueing network models was first stated by Kleinrock [6].

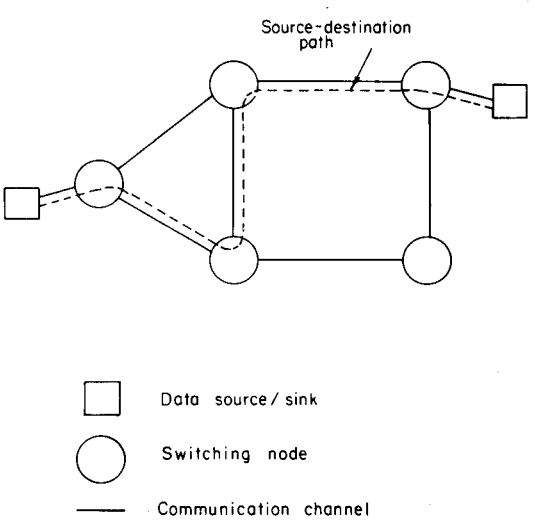


Fig. 1. A store-and-forward packet switching network.

The Independence Assumption. Each time a packet joins a queue in the network, its length is determined afresh from the probability density function

$$b(x) = \mu \exp\{-\mu x\}, \quad x \geq 0$$

where $1/\mu$ is the mean packet length (in number of bits).

In the queueing network models to be considered, first-come first-served (FCFS) servers are used to model communication channels and IS servers are used to model random delays associated with acknowledgements and timeouts. (An IS server is a service center with infinitely many servers in parallel [2].) Processing delays within switching nodes are often ignored since they are typically much smaller than channel delays. If included, nodal processors are also modeled as FCFS servers [7]. The servers in a network model are indexed by $i = 1, 2, \ldots, M$. Customers (i.e., packets) belong to different (routing) chains, indexed by $k = 1, 2, \ldots, K$. Chains are characterized by different routing behaviors. The routing behavior of customers in chain k is specified by a first-order Markov chain with transition probabilities

$$p_{ij}^{(k)} = \mathrm{P}(\text{to server } j \mid \text{currently at server } i),$$

$$i, j = 1, 2, \ldots, M.$$

Such a representation is adequate for modeling both the case of a single fixed route between a source and a sink and the case of multiple routes. When multiple routes exist, the selection of a route for individual packets is done probabilistically. Retransmissions and rerouting due to random transmission errors can also be modeled by an appropriate definition of the transition probabilities [8]. (A limitation of the class of product-form queueing networks is that adaptive routing cannot be modeled.)

Since the routing behaviors of packets travelling between different source–destination node pairs must be different, at least one routing chain must be specified for each source–destination node pair for which there is nonzero traffic. Sometimes, multiple chains may be needed between the same source–destination node pair to correspond to different virtual channels connecting data sources in the same source node and data sinks in the same destination node.

It is assumed that the source of chain k generates new packets according to a Poisson process

with rate γ_k packets per second where $k = 1, 2, \ldots, K$. Define

$$\gamma = \gamma_1 + \gamma_2 + \cdots + \gamma_K.$$

The corresponding throughput rates are denoted by γ^* and γ_k^* for $k = 1, 2, \ldots, K$. We must have

$$\gamma_k^* \leq \gamma_k, \quad k = 1, 2, \ldots, K,$$

since external arrivals to a virtual channel may not always be accepted due to buffer, flow or congestion control constraints.

Let the state of the queueing network be denoted by

$$S = (\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_M)$$

where

$$\boldsymbol{n}_i = (n_{i1}, n_{i2}, \ldots, n_{iK})$$

where n_{ik} is the number of chain k packets at server i. Define

$$n_i = n_{i1} + n_{i2} + \cdots + n_{iK}$$

and

$$\boldsymbol{n} = (n_1, n_2, \ldots, n_M).$$

A chain in the queueing network model is said to be *open* if both external arrivals to the chain and departures from the chain can occur freely. A routing chain is said to be *closed* if the number of customers (circulating) in the chain is fixed. (The concept of closed chains will be explained further below.)

In Section 2 below we shall describe queueing network models with closed chains for the performance analysis of packet switching networks with flow controlled virtual channels. A communication network typically has many virtual channels. The main source of difficulty in the modeling of such networks is that the computational requirements of conventional solution techniques [3, 9, 10] increase exponentially with the number K of closed chains in a queueing network model. Some approximate models and approximate solution techniques have been proposed and found to be useful. They are described in Section 2. Also described is the tree convolution algorithm [11] which provides an exact solution and has been designed to exploit the sparseness property of chains that is typical of communication network models. The problem of optimally routing a small amount of incremental traffic is also considered.

In Section 3 queueing network models with various population size constraints for the analysis of congestion control and buffer management strategies are described. Approximate models and solution techniques for a network of finite-buffer nodes are also presented.

2. Queueing network models with closed chains

Most packet networks nowadays provide virtual channels that are end-to-end flow controlled. Flow controlled virtual channels may be maintained between data sources and sinks [12,13] or between pairs of source-destination nodes [14,15]. In some networks, both types of flow controlled virtual channels are maintained [14]. In the queueing network models to be considered, only one type of flow controlled virtual channel is assumed. Such virtual channels may be interpreted as being maintained either between packet sources and sinks or pairs of source–destination switching nodes.

An important function of end-to-end flow control protocols is the synchronization of the data source input rate to the data sink acceptance rate. All of them work by limiting the number of packets that a virtual channel can have in transit within the network. Hence, they also provide, to some extent, a congestion control capability for the network as a whole. (However, when the number of virtual channels supported by the network is very large, a separate congestion control mechanism for the network is often necessary. Some such congestion control mechanisms are considered in Section 3.)

In this section we consider the modeling of flow controlled virtual channels as closed chains. The chain population size corresponds to the maximum number of packets that can be in transit within the virtual channel. This number will be referred to as the *virtual channel window size.* The effect of virtual channel window sizes on the throughput-delay characteristics of the network can be studied using queueing network models with closed chains.

2.1. Modeling virtual channels with closed chains

Fig. 2 illustrates a queueing network model of a packet switching network with K virtual channels. Each virtual channel has a source and a sink both of which are also modeled as FIFO servers with

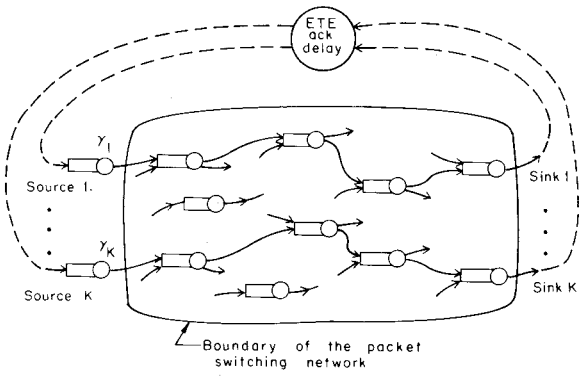


Fig. 2. Modeling flow controlled virtual channels in a packet switching network.

exponentially distributed service times. Packets in the same virtual channel follow a fixed route which may be chosen probabilistically from a finite set of routes between source and sink.

The delay for the return of an end-to-end (ETE) acknowledgement (ACK) from the sink to the source indicating receipt of a packet is modeled by an independent random variable, the distribution of which may be different for different virtual channels. This delay is modeled by an IS server that joins the sink to the source to yield a closed chain in the queueing network model. It is not really important to model the route of an ACK explicitly because ETE ACKs are typically either sent piggybacked in data packets, or, if sent separately, very short. Thus, they consume relatively small amounts of buffer and channel resources in the network, which may be acounted for separately. In [16] Reiser suggested that the ACK traffic may be accounted for by reducing the channel capacities by amounts equal to the throughputs of ACK packets.

The flow control window size of a virtual channel is the maximum number of packets that it can have in transit within the communication network at the same time. Let N_k denote the window size of virtual channel k for $k = 1,\ldots,K$. If the number of packets in transit within a virtual channel is equal to its window size, then the source server is 'blocked'. A blocked source server is later unblocked when an ETE ACK returns from the sink indicating receipt of a packet.

The blocking behavior is naturally modeled in a queueing network by a closed chain with a fixed number of circulating customers. Each customer corresponds to an 'access token'. Initially, N_k tokens are placed at the source server of virtual channel k. Each packet admitted into the network carries a token with it. When there is no more token at the source server, it is blocked. A packet arriving at the sink node of the virtual channel releases its token which is then carried back by the ETE ACK to the source server to be reused again. Thus, the N_k circulating tokens of a virtual channel correspond to the N_k circulating customers of a closed chain.

When the source server is not blocked, it generates a new packet for input into the network at the rate γ_k. The physical interpretation of the rate γ_k depends upon the loading on the virtual channels. For a lightly loaded network, γ_k may be interpreted as the external arrival rate of packets to the user of virtual channel k. (This corresponds to the assumption of packet generation according to a Poisson process.) For a heavily loaded network such that the queue of data packets at the source is nonempty most of the time, γ_k may be interpreted as the reaction speed of the user to a signal (or message) from the network interface of the virtual channel authorizing new input [17,18].

For Section 2 we shall make the assumption that the number of buffers at each switching node is very large (infinite). But, with flow control windows, a virtual channel does not always accept packets from its source. Thus the throughput γ_k^* of a virtual channel (modeled by chain k) is less than the rate γ_k; the latter is often referred to in the literature as the offered load of the virtual channel. With the above interpretation, the ratio γ_k^*/γ_k measures the fraction of time virtual channel k is not blocked. The above model assumptions enable us to focus our attention upon the network behavior only and ignore the behavior of queues external to the packet switching network.

2.2. *The solution of closed queueing networks*

A closed chain has a fixed number of circulating customers. However, it is sometimes physically meaningful to think of a closed chain as an open chain with the following two mechanisms in place at all times [1,4]:

(a) A loss mechanism whereby an external arrival to the chain is rejected and lost forever.

(b) A trigger mechanism whereby a departure from the network triggers an instantaneous injec-

tion of a new customer into the same chain as the departed customer (from an infinite source of customers).

For a closed chain (say k), its packet arrival rate to server i in the queueing network can only be determined to within a multiplicative constant, called the *scaling factor* of the chain, from

$$\lambda_{ik} = \sum_{j=1}^{M} \lambda_{jk} p_{ji}^{(k)}, \quad i = 1, 2, \ldots, M. \tag{2.1}$$

This is due to the fact that the matrix of transition probabilities of chain k is a stochastic matrix. The arrival rate of packets from all chains to server i is

$$\lambda_i = \sum_{k=1}^{K} \lambda_{ik}. \tag{2.2}$$

If server i is FCFS, it works at a constant rate of C_i bits per second. If server i is IS, it works at a rate of C_{ik} bits per second for chain k packets. The traffic intensity of chain k packets at server i is defined to be

$$\rho_{ik} = \begin{cases} \dfrac{\lambda_{ik}}{\mu C_i}, & \text{server } i \text{ is FCFS,} \\ \dfrac{\lambda_{ik}}{\mu C_{ik}}, & \text{server } i \text{ is IS,} \end{cases} \tag{2.3}$$

and the overall traffic intensity of server i is

$$\rho_i = \sum_{k=1}^{K} \rho_{ik}. \tag{2.4}$$

Consider a network consisting of closed chains only. Let N_k be the population size of chain k and define the *population vector*

$$\boldsymbol{N} = (N_1, N_2, \ldots, N_K).$$

The equilibrium network state probability has the following product form [2]:

$$\mathrm{P}(S) = \frac{1}{G(\boldsymbol{N})} \prod_{i=1}^{M} p_i(\boldsymbol{n}_i) \tag{2.5}$$

where

$$p_i(\boldsymbol{n}_i) = \begin{cases} n_i! \displaystyle\prod_{k=1}^{K} \frac{\rho_{ik}^{n_{ik}}}{n_{ik}!}, & \text{server } i \text{ is FCFS,} \\ \displaystyle\prod_{k=1}^{K} \frac{\rho_{ik}^{n_{ik}}}{n_{ik}!}, & \text{server } i \text{ is IS} \end{cases}$$

and $G(\boldsymbol{N})$ is the *normalization constant*. By definition,

$$G(\boldsymbol{N}) = \sum_{S \in \mathbb{S}(\boldsymbol{N})} \prod_{i=1}^{M} p_i(\boldsymbol{n}_i) \tag{2.6}$$

where the summation is done over the following set of feasible network states:

$$\mathbb{S}(\boldsymbol{N}) = \left\{ S: \sum_{i=1}^{M} \boldsymbol{n}_i = \boldsymbol{N} \right\}. \tag{2.7}$$

From (2.5) the following can be derived:

$$\mathrm{P}(\boldsymbol{n}) = \frac{1}{G(\boldsymbol{N})} \prod_{i=1}^{M} f_i(n_i) \tag{2.8}$$

where $f_i(n_i)$ is given by

$$f_i(n_i) = \begin{cases} \rho_i^{n_i}, & \text{server } i \text{ is FCFS,} \\ \dfrac{\rho_i^{n_i}}{n_i!}, & \text{server } i \text{ is IS.} \end{cases}$$

We mentioned earlier that the λ_{ik}'s can only be determined to within a multiplicative constant for each k. We note that, although $P(S)$ is independent of the scaling factors, they do affect the numerical value of the normalization constant [19]. Note that the normalization constant $G(\boldsymbol{N})$, defined by (2.6), is the sum of an extremely large number of product terms when K is large and also when the chain population sizes $\{N_k\}$ are large. There are two difficult problems in the evaluation of $G(\boldsymbol{N})$. First, depending on the scaling factors selected in the determination of $\{\lambda_{ik}\}$ in (2.1), $G(\boldsymbol{N})$ may become very large (causing a floating point overflow) or very small (causing a floating point underflow). This problem may be solved by a dynamic scaling technique [19]. The second problem is the extremely large computational time and space requirements to evaluate $G(\boldsymbol{N})$ for large values of K and $\{N_k\}$ using either the convolution algorithm [3,9], or the MVA algorithm [10]. The convolution algorithm is presented next. The MVA algorithm is introduced in Section 2.4.

Note that each of the functions p_i in (2.6) can be represented by a K-dimensional array indexed between $\mathbf{0}$ and $\boldsymbol{N}$, where $\mathbf{0}$ is a K-vector of all zeros. The convolution of two such functions, say p_1 and p_2, defines a real-valued function, say g_2, over the same domain, as follows [3,9]:

$$g_2(\boldsymbol{i}) = \sum_{j_1=0}^{i_1} \cdots \sum_{j_K=0}^{i_K} p_1(\boldsymbol{j})\, p_2(\boldsymbol{i}-\boldsymbol{j})$$

$$\text{for } \mathbf{0} \leqslant \boldsymbol{i} \leqslant \boldsymbol{N}. \tag{2.9}$$

In shorthand notation, (2.9) will be written as

$$g_2 = p_1 \otimes p_2 = p_2 \otimes p_1. \tag{2.10}$$

We note that $G(\boldsymbol{N})$ is simply the element indexed by $\boldsymbol{N}$ in the following array:

$$g_{\{1,2,\ldots,M\}} = p_1 \otimes p_2 \otimes \ldots \otimes p_M. \tag{2.11}$$

The convolution algorithm solves for $G(\boldsymbol{N})$ by performing the convolutions in (2.11) sequentially as follows:

$$g_{\{1,2,\ldots,m\}} = g_{\{1,2,\ldots,m-1\}} \otimes p_m \quad \text{for } m = 2, 3, \ldots, M. \tag{2.12}$$

The time requirement to compute the array $g_{\{1,2,\ldots,M\}}$ and hence $G(\boldsymbol{N})$ is of the order of $(M-1)\prod_{k=1}^{K}(\frac{1}{2}(N_k+2)(N_k+1))$ while the space requirement is of the order $2\prod_{k=1}^{K}(N_k+1)$. Note that both requirements grow exponentially with K. Two observations can be made to reduce the time requirement in queueing network models considered herein [3]. First, for the purpose of evaluting $G(\boldsymbol{N})$, all IS servers can be lumped together into an equivalent IS server with a traffic intensity equal to the combined traffic intensities of the individual IS servers. Second, if server m is an FCFS server with a constant service rate, the convolution of (2.12) can be accomplished with the following recursion:

$$g_{\{1,2,\ldots,m\}}(\boldsymbol{0}) = 1, \tag{2.13}$$

$$g_{\{1,2,\ldots,m\}}(\boldsymbol{i}) = g_{\{1,2,\ldots,m-1\}}(\boldsymbol{i}) + \sum_{\text{all } k,\, i_k>0} \rho_{mk}\, g_{\{1,2,\ldots,m\}}(\boldsymbol{i}-\mathbf{1}_k)$$

where i_k is the kth component of $\boldsymbol{i}$ and $\mathbf{1}_k$ is a K-vector with its kth component equal to one and all others equal to zero. The time requirement of the convolution algorithm based on (2.13) is of the order $MK\prod_{k=1}^{K}(N_k+1)$ which is still exponential in K.

The performance measures of throughputs and mean ETE delays of individual routing chains can be obtained as follows. The throughput of chain k at server m for a network of closed chains with population vector $\boldsymbol{N}$ is [3, 20]

$$\lambda^*_{mk}(\boldsymbol{N}) = \lambda_{mk}\frac{G(\boldsymbol{N}-\mathbf{1}_k)}{G(\boldsymbol{N})} \quad \text{for } \boldsymbol{N} \geq \mathbf{1}_k \tag{2.14}$$

where $G(\boldsymbol{N}-\mathbf{1}_k)$ is the normalization constant of the same network but with population vector $\boldsymbol{N}-\mathbf{1}_k$, and λ_{mk} is the relative arrival rate of chain k customers to server m given by (2.1). The chain throughput γ^*_k is then equal to the throughput of chain k customers at the source server of the chain.

The mean number of chain k customers at a FCFS server, say m, is given by [3]

$$q_{mk}(\boldsymbol{N}) = \rho_{mk}\frac{G_{m+}(\boldsymbol{N}-\mathbf{1}_k)}{G(\boldsymbol{N})} \quad \text{for } \boldsymbol{N} \geq \mathbf{1}_k \tag{2.15}$$

where G_{m+} is the result of convolving p_m (again) with $g_{\{1,2,\ldots,M\}}$. The mean delay incurred by chain k customers at a FCFS server (m) is given by Little's formula [21] to be $q_{mk}(\boldsymbol{N})/\lambda^*_{mk}(\boldsymbol{N})$.

We also know that the mean number of chain k customers at an IS server (m) is ρ_{mk}. Finally, the mean ETE delay of a virtual channel is given by

$$T_k = \frac{1}{\gamma^*_k}\sum_{m\in Q(k)} q_{mk} \tag{2.16}$$

where $Q(k)$ is the set of servers within the boundary of the packet switching network that are visited by chain k (see Fig. 2). Note that the argument $\boldsymbol{N}$ is omitted from T_k, γ^*_k and q_{mk} in (2.16). We shall do so as long as there is no ambiguity.

In the rest of Section 2 we shall first discuss two approximate analysis approaches that avoid the large space-time computational requirements encountered when the number of virtual channels is large. The first approach, in Section 2.3, is to focus upon a single closed chain and replace all other closed chains by open chains having the same throughputs [22]. The second approach, in Section 2.4, is an approximate solution technique [16] that is a natural extension of the mean value analysis (MVA) algorithm of Reiser and Lavenberg [10]. The tree convolution algorithm [11] for an exact analysis is next introduced in Section 2.5. The algorithm makes use of routing information and is very efficient when routing chains have sparseness and locality properties that are typical of communication network models. Next, we examine in Section 2.6 the traffic conditions under which an open-chain model accurately predicts the mean ETE delays of a closed-chain model having the same chain throughputs [17,18]. Finally, in Section 2.7, we discuss the problem of optimally routing a small amount of incremental traffic [18].

2.3. Analysis of a single virtual channel

One approach to avoid the large computational requirements of analyzing a closed network is to approximate some of the closed chains by open chains having the same throughputs. (We discuss the accuracy of this approximation in Section 2.6.) We thus have a queueing network model where some routing chains are open while others are closed. This is referred to as a mixed network model [2]. In this section we first present analytic results for mixed networks and then illustrate how these results can be applied to a model which focuses on a single virtual channel.

For an open chain (say k) its packet arrival rates to servers in the queueing network are determined uniquely by

$$\lambda_{ik} = \gamma_k q_i^{(k)} + \sum_{j=1}^{M} \lambda_{jk} p_{ji}^{(k)}, \quad i = 1, 2, \ldots, M \tag{2.17}$$

where $q_i^{(k)}$ is the probability for a customer freshly generated by the source of chain k to be routed next to server i. For a closed chain these packet arrival rates to servers are given by (2.1) to within a multiplicative constant. The traffic intensities ρ_{ik} and ρ_i for all i and k are defined as before by (2.3) and (2.4) for both open and closed chains in a mixed network model.

The equilibrium state probability of a mixed network also has the product-form solution [2], i.e.,

$$\mathrm{P}(S) = \frac{1}{G} \prod_{i=1}^{M} p_i(\boldsymbol{n}_i) \tag{2.18}$$

where $p_i(\boldsymbol{n}_i)$ is the same as defined in (2.5). The normalization constant G can be written as [3]

$$G = G^{\mathrm{o}} G^{\mathrm{c}}(\boldsymbol{N}) \tag{2.19}$$

where G^{o} and $G^{\mathrm{c}}(\boldsymbol{N})$ are defined as follows. Let

$$\rho_i^{\mathrm{o}} = \sum_{\substack{\text{all open} \\ \text{chains, } k}} \rho_{ik} \quad \text{and} \quad \rho_i^{\mathrm{c}} = \sum_{\substack{\text{all closed} \\ \text{chains, } k}} \rho_{ik}$$

for $i = 1, 2, \ldots, M$.

G^{o} is given by

$$G^{\mathrm{o}} = \prod_{i=1}^{M} G_i^{\mathrm{o}} \tag{2.20}$$

where

$$G_i^{\mathrm{o}} = \begin{cases} 1/(1-\rho_i^{\mathrm{o}}), & \text{server } i \text{ is FCFS,} \\ \exp\{\rho_i^{\mathrm{o}}\}, & \text{server } i \text{ is IS.} \end{cases}$$

$G^{\mathrm{c}}(\boldsymbol{N})$ is equal to the normalization constant of a network model with closed chains only and with the following modification to the traffic intensities of the closed chains:

$$\rho'_{ik} = \begin{cases} \rho_{ik}/(1-\rho_i^{\mathrm{o}}), & \text{server } i \text{ is FCFS,} \\ \rho_{ik}, & \text{server } i \text{ is IS.} \end{cases} \tag{2.21}$$

We now apply the above results to the single virtual channel model shown in Fig. 3. This model is based on the abstraction of all other network traffic into a single open chain, an approach used by Pennotti and Schwartz [22]. Thus, there are two routing chains: Chain 1 is closed and models the virtual channel, and chain 2 is the open chain mentioned above. For notational convenience, the ETE ACK delay is assumed to be zero. (All servers in Fig. 3 are of the FCFS type. However, an IS server representing an ETE ACK delay can be included without much difficulty.) The service time at the source server is assumed to be exponentially distributed with mean $1/\gamma_1$. Without loss of generality, packets belonging to the open chain (chain 2) are assumed to arrive from an external source to server m, for $m = 2, 3, \ldots, M$, at rate λ_{m2} and depart from the network after receiving service from server m.

For chain 1, a solution to (2.1) is $\lambda_{i1} = 1$ for all i. The equilibrium state probability is then given by

$$\mathrm{P}(S) = \frac{1}{G} \prod_{i=1}^{M} n_i! \prod_{k=1}^{2} \frac{\rho_{ik}^{n_{ik}}}{n_{ik}!}$$

where

$$\rho_{i1} = \begin{cases} 1/\gamma_1, & i = 1, \\ 1/(\mu C_i), & i = 2, 3, \ldots, M \end{cases}$$

and

$$\rho_{i2} = \begin{cases} 0, & i = 1, \\ \lambda_{i2}/(\mu C_i), & i = 2, 3, \ldots, M. \end{cases}$$

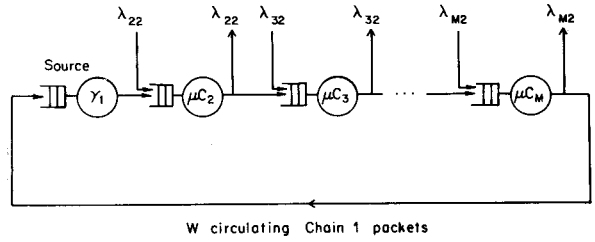


Fig. 3. A model of a single virtual channel.

Applying (2.19), G is given by

$$G = \left(\prod_{i=2}^{M} (1-\rho_{i2}) \right) G^{c}(W)$$

where $G^{c}(W)$ is the normalization constant for a network with a single closed chain of W circulating customers and with traffic intensity $\rho'_{i1} = \rho_{i1}/(1-\rho_{i2})$ for $i = 1, 2, \ldots, M$.

From the equilibrium state probability, the marginal queue length distribution of chain 1 (the virtual channel being considered) can be shown to be given by [3, 22]

$$\mathrm{P}(\{n_{11}, n_{21}, \ldots, n_{M1}\}) = \frac{1}{G^{c}(W)} \prod_{i=1}^{M} \left(\frac{\rho_{i1}}{1-\rho_{i2}} \right)^{n_{i1}}.$$

This is the same as the queue length distribution of a network model with only the closed chain but with the service rate of channel i reduced by the factor of $(1-\rho_{i2})$ for all i. The reduction factor accounts for the presence of open-chain packets in the network.

The mean number of chain 2 (open-chain) packets at channel i has the following simple form [22]:

$$\mathrm{E}[n_{i2}] = \frac{\rho_{i2}}{1-\rho_{i2}} (1 + \mathrm{E}[n_{i1}]). \tag{2.22}$$

Note that the term $\rho_{i2}/(1-\rho_{i2})$ is the mean queue length in a network model with the open chain only. The expansion factor $(1+\mathrm{E}[n_{i1}])$ accounts for the presence of closed-chain packets in the network. $\mathrm{E}[n_{i1}]$ may be computed by any of the computational algorithms for closed networks using the set of modified traffic intensities $\{\rho'_{i1}\}$.

2.4. A heuristic solution technique based upon the MVA algorithm

The time and space complexity of the MVA algorithm [10] are of the same order as the convolution algorithm. However, it has an intuitively appealing extension to an efficient heuristic solution technique that is shown to be asymptotically valid as the chain population sizes become infinite [10].

Let $Q(k)$ be the set of servers visited by chain k and $C(i)$ be the set of chains that visit server i. Also define the following notation for a closed queueing network with population vector $\boldsymbol{N}$:

$T_{ik}(\boldsymbol{N})$ – mean delay of a chain k packet at server i,

$q_{ik}(\boldsymbol{N})$ – mean number of chain k packets at server i,

$\gamma_k^*(\boldsymbol{N})$ – throughput of chain k in packets per second (this is equal to the throughput of the source server).

We first introduce the MVA algorithm which is based upon the following recursive equation,[1] for $i = 1, 2, \ldots, M$ and $k = 1, 2, \ldots, K$:

$$T_{ik}(\boldsymbol{N}) = \begin{cases} \tau_{ik}\left(1 + \sum_{c \in C(i)} q_{ic}(\boldsymbol{N} - \mathbf{1}_k)\right), \\ \qquad \text{server } i \text{ is FCFS}, \\ \tau_{ik}, \quad \text{server } i \text{ is IS} \end{cases} \tag{2.23}$$

where τ_{ik} is the mean service time of server i. It is equal to $1/\mu C_i$ for any k if server i is a communication channel (FCFS server). $q_{ic}(\boldsymbol{N} - \mathbf{1}_k)$ is zero if N_k in $\boldsymbol{N}$ is zero.

Using Little's formula first for chain k, and then for chain k at server i, the following is also derived for the MVA algorithm:

$$\gamma_k^*(\boldsymbol{N}) = \frac{N_k}{\sum_{i \in Q(k)} T_{ik}(\boldsymbol{N})} \quad \text{for } k = 1, 2, \ldots, K \tag{2.24}$$

and

$$q_{ik}(\boldsymbol{N}) = \gamma_k^*(\boldsymbol{N})\, T_{ik}(\boldsymbol{N}) \quad \text{for} \quad \begin{matrix} i = 1, 2, \ldots, M, \\ k = 1, 2, \ldots, K. \end{matrix} \tag{2.25}$$

Starting with $q_{ik}(\mathbf{0}) = 0$ for $i = 1, 2, \ldots, M$ and $k = 1, 2, \ldots, K$, $T_{ik}(\boldsymbol{N})$, $\gamma_k^*(\boldsymbol{N})$ and $q_{ik}(\boldsymbol{N})$ can be solved recursively using (2.23)–(2.25) for all i and k.

The computational time and space requirements of the above MVA algorithm grow exponentially with K. For communication network models characterized by a large K, a heuristic technique was proposed by Reiser [16] to solve the above set of equations iteratively. Suppose we can calculate the difference

$$\varepsilon_{ic}(k-) = q_{ic}(\boldsymbol{N}) - q_{ic}(\boldsymbol{N} - \mathbf{1}_k) \quad \text{for } i = 1, 2, \ldots, M, \quad c = 1, 2, \ldots, K. \tag{2.26}$$

[1] It is shown in [23] that the MVA recursion in (2.23) can be derived very simply from the convolution algorithm's recursion in (2.13). Also, for simplicity, each chain is assumed to have a single fixed route.

Then the MVA equations above can be written as

$$T_{ik} = \begin{cases} \tau_{ik}\left\{1 + \sum_{c \in C(i)} (q_{ic} - \varepsilon_{ic}(k-))\right\}, \\ \quad \text{server } i \text{ is FCFS}, \\ \tau_{ik}, \quad \text{server } i \text{ is IS}, \end{cases} \tag{2.27}$$

$$\gamma_k^* = \frac{N_k}{\sum_{i \in Q(k)} T_{ik}} \tag{2.28}$$

and

$$q_{ik} = \gamma_k^* T_{ik} \tag{2.29}$$

where the argument N has been omitted from T_{ik}, γ_k^* and q_{ik}.

Eqs. (2.26)–(2.29) form a set of nonlinear simultaneous equations. A simple method for solving them is by a successive substitution technique starting with an initial set of mean queue lengths $\{q_{ik}\}$ and iterating sequentially through the four equations until convergence is observed.

The space requirement is now substantially less since we only need to keep a single set of values for T_{ik}, γ_k^*, q_{ik} and $\epsilon_{ic}(k-)$. The time requirement is also significantly reduced with the following heuristic method [16] for evaluating $\epsilon_{ic}(k-)$, $i = 1, 2, \ldots, M$, $c = 1, 2, \ldots, K$:

(a) Assuming that the chain with one less packet is affected the most, use the estimate

$$\epsilon_{ic}(k-) = 0 \quad \text{for any } c \neq k. \tag{2.30}$$

(b) $\epsilon_{ik}(k-)$ is estimated by a single-chain network with suitably redefined parameters as follows. The mean service time for all FCFS servers in the single-chain network is

$$\tau_i = \frac{\tau_{ik}}{1 - \sum_{c \in C(i),\, c \neq k} \gamma_c^* \tau_{ic}} \tag{2.31}$$

where $\{\gamma_c^*\}$ is the set of chain throughputs at the current iteration step. Eq. (2.31) suggests that a chain k packet sees only a fraction of the channel capacity and it is consistent with the interference effect of the open-chain traffic on closed chains at a FCFS queue considered in the last section. Let $q_i(N_k)$ denote the mean queue size of server i in the single-chain network with service times given by (2.31). These mean queue lengths can be calculated quickly since $K = 1$. The following estimate is used in conjunction with those in (2.30) for the heuristic method,

$$\epsilon_{ik}(k-) = q_i(N_k) - q_i(N_k - 1). \tag{2.32}$$

The time complexity per iteration step in the heuristic solution is of the order $KM(N_1 + N_2 + \cdots + N_K)$ which is affordable even for large population sizes provided that convergence is achievable within a small number of iterations. It was observed empirically by Reiser [16] that the above iterative procedure converges rapidly from any set of initial values for $\{q_{ik}\}$ and $\{\gamma_k^*\}$. The only requirements to be satisfied by the initial condition are

$$q_{ik} \geq 0 \quad \text{for all } i, k$$

and

$$\sum_{i \in Q(k)} q_{ik} = N_k \quad \text{for all } k.$$

If was also argued that the iterative procedure is asymptotically valid as population sizes become infinite. This heuristic technique has also been generalized to queueing network models that do not have the product-form solution [16].

2.5. The tree convolution algorithm

A computational algorithm, called the tree convolution algorithm, was recently reported in [11]. It is intended for the solution of product-form queueing networks in which routing chains do not visit all servers (or service centers) in the network. In models of communication networks and distributed systems, it is often true that chains visit only a small faction of all queues in the network (*sparseness property*). Furthermore, chains are often clustered in certain parts of the network and their routes are constrained by the network topology (*locality property*). By making use of the routing information of chains, the time and space requirements of the tree convolution algorithm can be made substantially less than those of the (sequential) convolution and MVA algorithms presented earlier. The number of routing chains that can be handled varies depending upon the extent of sparseness and locality present in their routes. In [24], many numerical examples with 32–50 routing chains have been found to be solvable. In some cases, the solution of networks with up to 100 routing chains has been found to be possible.

The tree convolution algorithm provides an exact solution of normalization constants and per-

formance measures for product-form queueing networks. It is based upon two ideas. First, we note that the convolution in (2.11) can be performed in any order to obtain $g_{\{1,2,\ldots,M\}}$. Specifically, in the tree convolution algorithm, the arrays $\{p_m\}$ are placed at the leaf nodes of a tree (see Fig. 4). Each node in the tree corresponds to a subset of service centers that are descendants of that node. To compute the array $g_{\{1,2,\ldots,M\}}$, visit all nodes in the tree according to some order of tree traversal. The root node is visited last. A branch node may be visited only after all its sons have been visited. When a branch node is visited, an array g_{SUBNET} is computed for the node from the arrays g_{SUBNET1} and g_{SUBNET2} of its (two) sons by

$$g_{\text{SUBNET}} = g_{\text{SUBNET1}} \otimes g_{\text{SUBNET2}} \tag{2.33}$$

where SUBNET = SUBNET1 ∪ SUBNET2. If the node has more than two sons, then convolutions are performed sequentially one after the other. Finally, when the root node is visited, the array $g_{\{1,2,\ldots,M\}}$ is obtained.

Both the tree convolution algorithm and the sequential convolution algorithm require $M-1$ convolutions. In fact, it is easy to see that the sequential convolution algorithm is just a special case of the tree algorithm. However, substantial time and space savings can be achieved by the general tree algorithm by making use of the following additional observation.

Consider routing chain k. Let CENTERS(k) be the set of service centers visited by chain k. Let SUBNET denote a subset of the M service centers. With respect to SUBNET, chain k is said to be *fully covered* if CENTERS(k) ⊆ SUBNET; chain k is said to be *noncovered* if the intersection of CENTERS(k) and SUBNET is null; otherwise, chain k is said to be *partially covered*.

Partition the set of K chains into the following 3 sets with respect to SUBNET:

$\sigma_{\text{pc}} = \{k \mid$ chain k is partially covered by SUBNET$\}$,

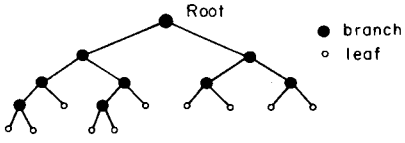


Fig. 4. A tree.

$\sigma_{\text{fc}} = \{k \mid$ chain k is fully covered by SUBNET$\}$,
$\sigma_{\text{nc}} = \{k \mid$ chain k is noncovered by SUBNET$\}$.

Now make the observation that only those elements of g_{SUBNET} corresponding to the following index values are needed for further convulsions to arrive at $G(\boldsymbol{N})$,

$$\{\boldsymbol{i} \mid i_k = 0, 1, 2, \ldots, N_k, k \in \sigma_{\text{pc}};\ i_k = N_k, k \in \sigma_{\text{fc}};$$
$$i_k = 0, k \in \sigma_{\text{nc}}\}.$$

Let $|\sigma|$ denote the cardinality of set σ. For the purpose of computing $G(\boldsymbol{N})$ it is sufficient to store g_{SUBNET} in an array with dimensionality $|\sigma_{\text{pc}}|$ indexed by $\boldsymbol{i}_{\text{pc}} = \{i_k,\ k \in \sigma_{\text{pc}}\}$. Such an array is termed a *partially covered array*. The amount of space needed for a partially covered array is $\prod_{k \in \sigma_{\text{pc}}}(N_k + 1)$ locations. (Additionally, a small amount of space is also needed to store the set σ_{pc}.) The time requirement of the convolution in (2.33) using partially covered arrays is shown in [11]. Very large time and space savings are possible if $|\sigma_{\text{pc}}| << K$ at every node of the tree.

Given a subset of centers in a network that has many centers and sparse routing chains, it is highly likely that only a few chains will be partially covered by the subset. Thus, for queueing networks with properties of sparseness and locality, the time and space savings from the use of partially covered arrays instead of K-dimensional arrays will be very substantial.

The actual time and space needed for the tree convolution algorithm also depend upon the following 'tree planting' decisions: the tree configuration, the order of tree traversal and the placement of service centers at leaf nodes. The objective of tree planting is to minimize the overall space and time needed by the algorithm by minimizing the numbers of partially covered chains in subnets associated with branch nodes in the tree. Tree planting algorithms are discussed in [11, 24].

In addition to substantial space and time savings, the tree convolution algorithm has several additional significant advantages over the other algorithms. First, the tree of partially covered arrays employed by the algorithm provides a very flexible data structure for tailoring time-space tradeoffs to individual queueing networks in the calculation of network performance measures. It will also facilitate the solution of very large queueing networks with the help of storage management techniques or by means of parallel computation on

a multiprocessor machine. Furthermore, the computation of the marginal distribution of queue lengths in a service center may be obtained with $(\log_2 M) - 1$ convolutions instead of $M - 1$ convolutions needed by a sequential convolution algorithm, where M is the total number of service centers and a balanced binary tree is assumed. Lastly, an analysis of the expected time and space requirements of the algorithm as a function of the sparseness of routing chains is presented in [25].

2.6. Network design using both closed-chain and open-chain models

With the tree convolution algorithm, communication network models with many queues and a relatively large number of sparse routing chains can be solved exactly. (Chains are said to be sparse if the average number of queues visited by a chain is much smaller than the number of queues in the network.) Both throughputs and mean transit delays for individual chains can be calculated exactly with time and space requirements within the limits of present computers. However, these time and space requirements are still fairly large. Hence, the tree algorithm is not very practical for use within network design procedures that require numerous applications of the algorithm.

Both Pennotti and Schwartz [22] and Gerla and Nilsson [26] suggested the use of open chains to approximately model flow controlled virtual channels. The difficulty encountered is that the throughputs of the flow controlled virtual channels needed as input parameters of an open-chain model are not known. Lam and Lien [17,18] proposed to use a combination of both open-chain and closed-chain models in design procedures for networks with flow controlled virtual channels. A closed-chain model is first employed and chain throughputs and mean delays are computed exactly using the tree convolution algorithm. An open-chain model with the same chain throughputs is then employed for a sequence of intermediate optimization steps in the network design procedure (e.g., the routing of incremental flows to be considered in Section 2.7). To avoid the accumulation of errors, the closed-chain model is employed at various checkpoints of the design procedure to recompute chain throughputs and mean delays. The question of interest then is: How accurate is such an approximation?

The above question was addressed by Lam and Lien [17,18] who considered a network model with 64 communication channels and 32 virtual channels. The throughputs and mean ETE delays of a closed-chain model are first computed using the tree convolution algorithm. An open-chain model having the same chain throughputs is then specified and its mean ETE delays calculated using the M/M/1 delay formula for individual queues as described in [5–7].

The service rate of each communication channel was assumed to be the same; $\mu C_m = 10$ packets/second for all m. The source input rate of each virtual channel was also assumed to be the same; $\gamma_k = \gamma$ for all k (see Fig. 2).

The effect of varying the relative source and communication channel speeds γ and μC was considered. γ was varied from 10 to 0.5 while keeping μC constant at 10 packets/second. The accuracy of the open-chain model was found to be very poor for $\gamma = 10$ (same value as μC) and it improved as γ was decreased. The average error in the mean ETE delays of the virtual channels was 40.3% for $\gamma = 10$, 2.02% for $\gamma = 1$ and 0.40% for $\gamma = 0.5$. There are two possible reasons for this behavior. First, when the utilization of a M/M/1 queue is high, its delay distribution has a long tail, which gives rise to a poor estimate of the delay in a closed network where the queue lengths are bounded. Note that when $\gamma = \mu C$, the bottleneck in a routing chain is at one of the communication channels within the packet switching network. On the other hand, when μC is much smaller than γ, the source server is the bottleneck in a routing chain and it behaves like a Poisson source at rate γ much of the time (i.e., like an open chain).

In general, the accuracy of the open-chain model suffers from the presence of bottlenecks within the network (either due to a large γ or due to poor routing).

The effect of varying the virtual channel window size was also investigated. Window sizes of 2, 3 and 4 were considered. It was found that as the window size increases, the accuracy of the open-chain model improves, despite increases in the channel utilizations.

Finally, another observation in [17,18] is that in almost all cases considered, the mean delay estimates are larger than the actual mean delays. There are two possible reasons for the open-chain model to overestimate mean delays. First, delay

estimates are obtained from M/M/1 queue delay distributions that have long tails. Second, the mean-value analysis shows that the mean delay encountered by a closed-chain customer is determined by the mean queue lengths of a network with that customer removed [10]. The open-chain model as described above does not account for this behavior.

A consequence of the overestimation of delays is that the impact of bottlenecks on chain delays in an open-chain model is exaggerated compared to that in a closed-chain model. This means that if an open-chain model is used for the routing of incremental flows (see the following section), bottlenecks will be avoided more 'rigorously' than if a closed-chain model is employed.

2.7. Optimal routing of incremental flows

We consider the problem of introducing a small amount of incremental flow from a source node to a destination node into a network with existing flows. Several optimal routing problems may be formulated depending upon the nature of the incremental flow and the optimization objective.

The objective of the ARPANET routing algorithm [27] is to minimize the (estimated) delay of an individual packet from its source node to its destination node. However, it has been observed by several authors [28,29] that routing algorithms with the objective of individual-optimization do not necessarily lead to network-optimization, i.e., minimizing the mean delay of all packets in the network. The flow deviation method [7,30,31] considers an incremental flow that is infinitesimal relative to existing flows in the network. The network-optimization objective is pursued; specifically, the route for the incremental flow is chosen to minimize the (infinitesimal) increase ΔT in the mean network transit delay T of all packets.

A similar problem for networks with flow-controlled virtual channels was posed by Lam and Lien in [18]. The incremental flow corresponds to the addition of a new virtual channel with a window size of one (not necessarily an infinitesimal amount of flow). In this case, one method to evaluate ΔT is to calculate T using the tree convolution algorithm for the network both with and without the additional virtual channel (given a specific route for it). However, to determine the optimal route with this approach would require numerous applications of the tree convolution algorithm and would be very expensive in terms of computation time.

The solution approach described in Section 2.6 was proposed [18]. A closed-chain model is initially used to calculate the throughputs of the existing flow controlled virtual channels. These are then modeled as open chains. The new virtual channel to be added is modeled as a closed chain.

Let the aggregate arrival rate of the existing traffic in the network to communication channel m be denoted by λ_m packets/second. The service rate of channel m is μC_m packets/second where $1/\mu$ is the average length of a packet in bits and C_m is the channel speed in bits/second. Define $\rho_m = \lambda_m/(\mu C_m)$. The total throughput rate at which open-chain packets leave (or enter) the network is γ_o packets/second. The mean ETE delay of open-chain packets is T_o. The closed chain representing the virtual channel being added has a population size of one (i.e., window size is one), a source server work-rate of γ packets/second and a mean ETE ACK delay of τ seconds. The source and sink nodes of the virtual channel are known but its route is to be determined.

Let Q denote the set of communication channels constituting a route chosen for the new virtual channel. The increase in the mean network delay due to the incremental traffic was found to be

$$\Delta T = \frac{\sum_{m\in Q} \dfrac{\mu C_m}{(\mu C_m - \lambda_m)^2} - T_o}{\gamma_o\left(\dfrac{1}{\gamma} + \tau + \sum_{m\in Q} \dfrac{1}{\mu C_m - \lambda_m}\right) + 1}. \tag{2.34}$$

Since the amount of incremental traffic is not infinitesimal, it makes sense to impose a maximum delay bound $\tau_{\max}$ on the mean delay of the new virtual channel. (Most likely, the user requesting for the new virtual channel will want his mean delay to be bounded.) The following constrained optimization problem results:

$$\min_{Q} \quad \Delta T$$

$$\text{s. t.} \quad \sum_{m\in Q} \frac{1}{\mu C_m - \lambda_m} < \tau_{\max}. \tag{2.35}$$

A dual of the above problem is

$$\min_{Q} \quad \sum_{m\in Q} \frac{1}{\mu C_m - \lambda_m}$$

$$\text{s. t.} \quad \Delta T < \Delta_{\max} \tag{2.36}$$

where Δ_{max} is a bound on ΔT. The above problems were formulated by Lam and Lien [18] who also provided an algorithm to solve the problem in (2.35). We can interpret each of the above constrained problems as a compromise between the objectives of individual-optimization and network-optimization. Note that the individual-optimization objective of ARPANET routing does not consider the impact of the incremental flow on the network. On the other hand, the network-optimization objective of flow deviation ignores the performance of the incremental flow (since it is assumed to be infinitesimal). The above constrained problems take into account both considerations.

3. Queueing network models with population size constraints

In this section we shall consider the modeling of congestion control and buffer management strategies in packet networks. To do so, we need to introduce the class of queueing networks with population size constraints [4].

3.1. The model

Routing chains in queueing network models considered prior to now in this paper and in [5] are either open or closed. Recall our discussion in Section 2.2 that a closed chain can be viewed as an open chain with the loss and trigger mechanisms in place at all times. Suppose the loss and trigger mechanisms are invoked or revoked as a function of the network's population vector $\boldsymbol{N} = (N_1, N_2, \ldots, N_K)$. Such queueing networks are said to have population size constraints. Given the loss and trigger mechanisms as functions of $\boldsymbol{N}$, let V be the set of feasible network population vectors. A sufficient condition for the equilibrium network state probability P(S) to have the product-form solution is stated in [4]:

> For any chain k and population vectors $\boldsymbol{N}$ and $\boldsymbol{N} + \mathbf{1}_k$ in V, the loss mechanism is invoked for a chain k external arrival in any network state with population vector $\boldsymbol{N}$ if and only if the trigger mechanism is invoked for a chain k external departure in any network state with population vector $\boldsymbol{N} + \mathbf{1}_k$.

This is equivalent to the condition that feasible transitions between population vectors in V are always paired.

By permitting V to be a singleton set as well as an infinite set, both networks of closed chains and networks of open chains are included here as special cases. The set of feasible network states is $\mathbb{S} = \cup_{\boldsymbol{N} \in V} \mathbb{S}(\boldsymbol{N})$ where $\mathbb{S}(\boldsymbol{N})$ is defined by (2.7). Thus, the (improper) equilibrium probability of the set of states $\mathbb{S}(\boldsymbol{N})$ is equal to the normalization constant $G(\boldsymbol{N})$ of an equivalent closed network with population vector $\boldsymbol{N}$; chain arrival rates to individual servers are given by (2.1) for closed chains and (2.17) for chains that permit external arrivals [4,19]. For constant external chain arrival rates, a queueing network model with population size constraints has the product-form solution given by (2.5) with the following expression for the normalization constant:

$$G = \sum_{\boldsymbol{N} \in V} G(\boldsymbol{N}). \tag{3.1}$$

In Sections 3.2–3.4 we discuss queueing network models with population size constraints for the analysis of strategies for buffer management and congestion control.

3.2. Finite-buffer single node model

When a packet is in transit in a packet switching network, it occupies buffer space in the intermediate store-and-forward nodes. When nodes with finite buffers are considered, the allocation of buffers to the various chains will affect the node's performance (and hence, the network's performance).

To model a packet switching network with finite buffer space, one must consider the situation when a packet forwarded to a node finds no available buffer in that node. The store-and-forward protocol requires the sending node to keep a copy of the packet until an ACK is returned from the receiving node. If an ACK is not received within a time-out interval, the packet is retransmitted. Buffer space in the sending node is therefore occupied by this packet until an ACK is received.

An exact analytic model of an entire network with finite buffers at each node is not presently available. Models of a single switching node, hower, have been successfully analyzed [32–35]. In this section, buffer allocation strategies studied by

Kamoun and Kleinrock [35] are considered. Their model does not include ACKs. As soon as a packet has been transmitted on an outgoing channel, its buffer space is assumed to be released immediately. A single node model with provisions for ACK and timeout delays [8,32] will be discussed in Section 3.4. Approximate solution techniques to analyze a network of finite-buffer nodes will also be described in that section.

The single node model analyzed by Kamoun and Kleinrock [35] is shown in Fig. 5. It is a specialized queueing network model with M servers in parallel, one for each outgoing channel. There are M routing chains: packets routed to the same outgoing channel are in the same chain. The arrival process of chain i packets to the node is assumed to be Poisson with rate λ_i, $i=1,2,\ldots,M$. The population size constraints for the various routing chains are determined by the buffer management scheme used.

Packets belonging to the various chains share a total of B buffers. When an arriving packet is rejected from entering the node, it is assumed to be lost. (In a network of finite-buffer nodes, rejected packets are not really lost, but are retransmitted later; hence, the arrival process to each node is not likely to be Poisson. The Poisson arrival assumption mentioned above is therefore only an approximation in this context.)

We next present a general buffer allocation scheme that includes as special cases the four different buffer-sharing strategies studied in [35]. The general scheme is specified in terms of minimum allocations and maximum limits for the different classes of packets. Define

(i) the number of buffers dedicated to chain i to be b_i ($b_i \geq 0$), and

(ii) the maximum number of buffers that chain i packets can occupy to be B_i ($B_i \leq B$) for $i=1,2,\ldots,M$.

We must have $b_i \leq B_i$ for all i, $\sum_{i=1}^{M} b_i \leq B$, and $\sum_{i=1}^{M} B_i \geq B$ to be physically meaningful.

It is often desirable to use an over-commitment strategy such that

$$\sum_{i=1}^{M} B_i > B.$$

Let a state of the single node model be

$$\boldsymbol{m} = (m_1, m_2, \ldots, m_M)$$

where m_i is the number of chain i packets in the node.[2]

The general buffer allocation scheme can be modeled by a queueing network model with population size constraints. The set of feasible states can be written as

$$V = \left\{ \boldsymbol{m}: m_i \leq B_i, \text{ for all } i, \text{ and } \sum_{i=1}^{M} \max\{0, m_i - b_i\} \leq B - \sum_{i=1}^{M} b_i \right\}. \quad (3.2)$$

Applying the product-form solution, the equilibrium network state probability is given by

$$P(\boldsymbol{m}) = \frac{1}{G} \prod_{i=1}^{M} \rho_i^{m_i} \quad (3.3)$$

where $\rho_i = \lambda_i/(\mu C_i)$ and

$$G = \sum_{\boldsymbol{m} \in V} \prod_{i=1}^{M} \rho_i^{m_i}. \quad (3.4)$$

Special cases of the above scheme that have been studied by Kamoun and Kleinrock [35] are the following:

Case 1. Complete partitioning ($b_i > 0$ and $\sum_{i=1}^{M} b_i = B$) – The B buffers are partitioned into M groups, the ith group (with size b_i) is allocated to chain i. There is no sharing of buffers among chains. Hence, B_i is equal to b_i, and we also have $\sum_{i=1}^{M} B_i = B$.

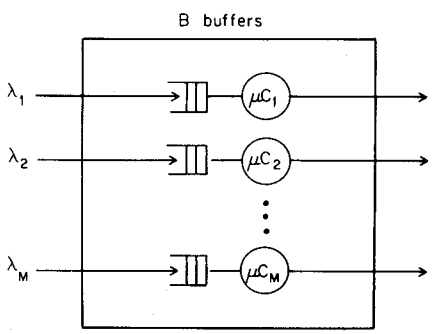


Fig. 5. A model of a switching node with a finite buffer pool.

[2] In a general network model, n_{ik} denotes the number of chain k packets at channel i. Since the current model has the simple behavior that chain i packets visit channel i only, n_{ii} is therefore the only non-zero element in the vector $\boldsymbol{n}_i$. We use m_i instead of $(0,\ldots,n_{ii},\ldots,0)$ for convenience. Furthermore, feasible states in this model correspond to feasible population vectors considered in Section 3.1.

Case 2. Complete sharing ($b_i = 0$ and $B_i = B$) – The B buffers are completely shared by the M chains. Buffers are allocated on a FCFS basis.

Case 3. Sharing with maximum allocations ($b_i = 0$ and $B_i < B$) – The B buffers are shared by all chains with the restriction that the number of buffers occupied by chain i cannot exceed B_i.

Case 4. Sharing with minimum allocations ($b_i > 0$ and $B_i = B$) – Chain i packets are guaranteed at least b_i buffers, and the remaining $B - \sum_{i=1}^{M} b_i$ buffers are shared by all chains.

We now illustrate how one can obtain analytic expressions for performance measures such as blocking probability, throughput and mean delay. Let $V_i \subset V$ be the set of states in which a chain i arrival is rejected. For the general buffer allocation scheme, V_i is given by

$$V_i = \left\{ \boldsymbol{m}: m_i = B_i \text{ or } \left(m_i \geq b_i \text{ and } \sum_{i=1}^{M} \max\{0, m_j - b_j\} = B - \sum_{j=1}^{M} b_j \right) \right\}. \quad (3.5)$$

Let β_i be the blocking probability of chain i packets,

$$\beta_i = \sum_{\boldsymbol{m} \in V_i} \mathrm{P}(\boldsymbol{m}). \quad (3.6)$$

The throughput of chain i packets is then given by

$$\gamma_i^* = \lambda_i (1 - \beta_i). \quad (3.7)$$

To get the mean delay experienced by chain i packets, we must first get the mean number of chain i packets in the node. This can be obtained from

$$\mathrm{E}[m_i] = \sum_{\boldsymbol{m} \in V} m_i \mathrm{P}(\boldsymbol{m}). \quad (3.8)$$

Little's formula [21] is then used to get the following expression for the mean delay of chain i:

$$T_i = \mathrm{E}[m_i]/\gamma_i^*. \quad (3.9)$$

It should be noted that the mean delay in (3.9) is for packets which are accepted into the node. It does not include those that are rejected.

Eqs. (3.6)–(3.9) are expressed in terms of a summation over a set of network states. In special cases, they can often be simplified and expressed in terms of the normalization constant G. The reader is referred to [35] for such simplifications. Numerical examples showing the relative merits of the four special cases are also provided in that reference. A general conclusion is that the best scheme and the best setting of parameters (b_i and B_i) depend upon the values of the ρ_i's.

3.3. Permit-oriented congestion control

A packet switching network can be viewed as a set of resources shared by a population of users. Such resources include channels and buffers. If the resources are not managed properly, an increased demand from a single user or a group of users may cause degradation in network performance. This degradation is usually in the form of substantially reduced throughput [36, 37]. When this happens, the network is said to be in a state of congestion. The objective of congestion control is then to prevent the network from going into the congestion state. Congestion control schemes usually involve some form of restriction on the amount of network resources allocated to each external user. The window flow control mechanism discussed in Section 2.1 provides a congestion control function because it places a limit on the number of packets belonging to a virtual channel in the network. However, if the number of virtual channels is large, the combined load on the network can still become excessive, and an additional network-wide congestion control scheme may be required.

A basic principle is to apply control at the point of entry to the network. An example of such a technique is the isarithmic control scheme suggested by Davies [38]. This scheme places a limit on the total number of packets in the network; no discrimination is made on the basis of routing chains. It can be implemented by circulating a number of 'permits' in the network, and requiring a packet to secure a permit before it can be admitted into the network. Another example is window flow control which specifies different types of permits for packets belonging to different virtual channels.

We observe that the principles of permit-oriented congestion control schemes for a network are basically the same as those of buffer allocation schemes for a switching node. Here, permits play the role of buffers considered earlier in Section 3.2. The general scheme, based upon the specification of maximum limits and minimum allocations, discussed there can be used to specify the allocation of permits for network congestion control as

well. In this section we present the work reported in [39] which considers the special case of maximum limits only.

The routing chains in the network are assumed to be partitioned into disjoint groups, and the number of permits used by each group cannot exceed some pre-specified maximum. The notion of group allows the flexibility of imposing controls on selected sets of routing chains. The resulting congestion control scheme provides a two-level isarithmic control [39, 40]. At level 1, a limit is placed on the total number of packets in the network; and at level 2, separate limits are placed on each group. Let L be the total number of permits, D be the number of groups, and L_u be the limit for group u, $u = 1, 2, \ldots, D$. The general scheme can be implemented by two types of permits. Type 1 consists of L permits shared by all packets. Type 2 permits are also distinguished by a group number; the number of permits for group u is L_u. A packet must acquire *both* a type 1 permit and a type 2 permit for its group before it can be admitted into the network.

The two types of permits are not always needed in special cases. For example, the complete sharing scheme corresponds to the method of isarithmic control [38], which can be implemented by the L type 1 permits only. In the complete partitioning scheme, type 2 permits are required while type 1 permits are not used.

The implementation of permit-oriented congestion control for a network is substantially more complicated than buffer allocation schemes for a node, due to the need for decentralized control in a network. A packet acquires one or more permits when it enters the network and releases its permit(s) when it reaches its destination node. The distribution of free permits is an important but difficult implementation problem. One would like to minimize the probability that when a permit is needed at a particular node, all the free permits are somewhere else in the network. Davies [38] suggested that each node may keep up to a maximum number of free permits, and extra permits are sent to randomly-selected neighbors. (Note that this is a maximum allocation strategy applied to individual nodes.)

We now consider the use of a queueing network model with population size constraints to study permit-oriented congestion control. For mathematical tractability we need to assume that the buffer space at each node is unlimited. It is also necessary to assume that free permits circulate through the network with no delay so that an external packet arrival to the network receives a permit right away if a free one is present in the network. In reality, a packet's entry to the network may be delayed because the available free permits may be in other parts of the network. The last assumption therefore results in optimistic estimates of network performance.

Recall that the state of the network model is given by $S = (\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_M)$ where $\boldsymbol{n}_i = (n_{i1}, n_{i2}, \ldots, n_{iK})$; n_{ik} is the number of chain k packets at server i. The K routing chains are partitioned into D groups, and a packet is said to belong to group u (denoted by Γ_u) if its routing chain is in group u. The equilibrium network state probability $P(S)$ is given by the product term in (2.5). The appropriate normalization constant is defined below.

It is convenient to define a less detailed state description

$$S' = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M)$$

where

$$\boldsymbol{y}_i = (m_{i1}, m_{i2}, \ldots, m_{iD})$$

where m_{iu} is the number of group u packets at server i. Let

$$N_u = \sum_{i=1}^{M} m_{iu}$$

be the population size of group u packets when the network is in state S'. The set of feasible network states is

$$\mathbb{S}' = \bigcup_{\boldsymbol{N} \in V} \mathbb{S}'(\boldsymbol{N})$$

where $\boldsymbol{N}$ denotes the population vector $(N_1, N_2, \ldots, N_D)$; the set of feasible population vectors is

$$V = \left\{ \boldsymbol{N}: \sum_{u=1}^{D} N_u \leq L \text{ and } N_u \leq L_u \text{ for all } u \right\} \quad (3.10)$$

and

$$\mathbb{S}'(\boldsymbol{N}) = \left\{ S': \sum_{i=1}^{M} \boldsymbol{y}_i = \boldsymbol{N} \right\}.$$

By adding all state probabilities $P(S)$ such that $\Sigma_{k \in \Gamma_u} n_{ik} = m_{iu}$ for $i = 1, 2, \ldots, M$ and $u =$

$1, 2, \ldots, D$, we get [39]

$$P(S') = \frac{1}{G} \prod_{i=1}^{M} m_i! \prod_{u=1}^{D} \frac{\phi_{iu}^{m_{iu}}}{m_{iu}!}, \quad S' \in \mathcal{S}' \tag{3.11}$$

where

$$m_i = \sum_{u=1}^{D} m_{iu}, \qquad \phi_{iu} = \sum_{k \in \Gamma_u} \rho_{ik}$$

and

$$G = \sum_{N \in V} G(N)$$

where $G(N)$ is the normalization constant of a closed network with the same set of traffic intensities $\{\phi_{iu}\}$ and N as its population vector [19]. $G(N)$ may be evaluated using any of the computational algorithms described earlier in Section 2.

To obtain the network throughput of each group, we follow the developments which lead to (3.7) in Section 3.2. The blocking probability of group u packets is

$$\beta_u = \sum_{N \in V_u} G(N) \tag{3.12}$$

where

$$V_u = \left\{ N: \sum_{i=1}^{D} N_i = L \text{ or } N_u = L_u \right\}. \tag{3.13}$$

The throughput of group u is then given by

$$\gamma_u^* = \sum_{k \in \Gamma_u} \gamma_k (1 - \beta_u). \tag{3.14}$$

3.4. Finite-buffer network models

The peak throughput of a network is attained when all its communication channels are transmitting packets (assuming that the mean packet length and the mean path length of packets in the network are constant). The network throughput over a period of time is usually less than the peak value because of (i) the lack of input traffic, or (ii) some constraints or interference effects that force communication channels to be 'nonproductive' part of the time. In Sections 3.2 and 3.3 we considered models which were used to study the throughput degradation behavior arising from interference between different streams of traffic [34, 35, 39]. In this section we shall consider network throughput degradation behavior due to insufficient buffers at switching nodes.

In what follows we shall first describe an approximate analysis method for dealing with the problem of blocking arising from the assumption of finite nodal buffer pools. Two special applications of the analysis method are discussed: A model for determining nodal buffer requirements needed to achieve small nodal blocking probabilities, and a model for studying the performance of the 'input buffer limit' strategy for network congestion control [8, 41].

A packet received by a switching node for forwarding may be accepted or discarded in accordance with some buffer management strategy depending upon some attribute of the packet (such as its priority level, class, destination, etc.) and the utilization level(s) of the node's buffer pool(s). A packet, discarded in this fashion, is said to be *blocked*. The sending node, however, has a copy of the packet and will retransmit it later when a timeout occurs and no positive ACK has been received.

To get around the difficulty of modeling blocking in a queueing network, the following approximate solution technique was first proposed in [8]. The overall problem (network of switching nodes) is first decomposed into a set of analytically tractable problems, i.e., a queueing network model for each switching node. The single-node results are then combined by requiring conservation of the various packet flows within the network as described below.

The queueing network model of a switching node shown in Fig. 6 was proposed by Lam and Schweitzer [8, 32] to study buffer requirements for networks with completely shared buffer pools. It

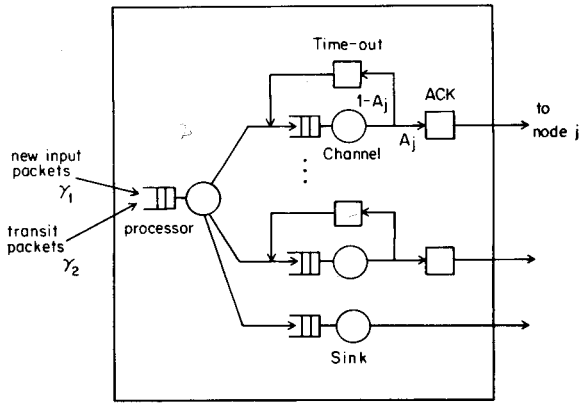


Fig. 6. A single node model with ACK and time-out delays.

was also employed later by Lam and Reiser [41] to study the input buffer limit strategy for network congestion control. FCFS servers are used to model the nodal processor, the communication channels and the sink for packets destined for this node. IS servers are used to model ACK delays and time-out delays.

Two types of packets are distinguished[3]: Transit packets forwarded by adjacent nodes and new input packets generated locally. They are represented by two routing chains with exernal arrival rates γ_2 and γ_1 respectively. The following buffer management strategy is considered. Suppose the node has N_T buffers. Transit packets are rejected only when all buffers are occupied. However, when there are N_I input packets in the node, any newly arrived input packet is rejected. We have $N_I \leq N_T$. The ratio N_I/N_T is said to be the *input buffer limit*.

For each routing chain, routing transition probabilities from the nodal processor to one of the communication channels or the sink are determined by the routing behavior of source-destination paths in the real network. The routing transition probability from a communication channel to either the ACK or the time-out IS server depends upon the rejection probability for transit packets of the neighboring node (say node j) at the other end of the communication channel. A packet is accepted by the neighboring node with probability A_j; it joins the ACK server and subsequently leaves the current node (when an ACK is received). A packet is rejected with probability $1 - A_j$ by the neighboring node; no ACK will be returned. Conceptually, the packet joins the time-out server and subsequently rejoins the channel queue. Note that this routing behavior is consistent with the probabilistic routing behavior defined for product-form queueing network models. The acceptance probability A_j is assumed to be the product $(1 - E)(1 - \beta_j)$ where E is the probability of a transmission error in the packet and β_j is the blocking probability of node j.

Suppose that the set of nodal blocking probabilities $\{\beta_j\}$ for transit packets is known for each communication channel in Fig. 6. Then the node can be modeled by a product-form queueing network with two routing chains and the population size constraints

$$0 \leq N_1 + N_2 \leq N_T$$

and

$$0 \leq N_1 \leq N_I.$$

Let $P_m(N_1, N_2)$ be the equilibrium probability of having N_1 input packets and N_2 transit packets in node m. The equilibrium blocking probability for transit packets at this node is

$$\beta_m = \sum_{N_1=0}^{N_I} P_m(N_1, N_T - N_1), \quad m = 1, 2, \ldots, M. \tag{3.15}$$

Since the set $\{P_m(N_1, N_2)\}$ in turn depends upon the set $\{\beta_j\}$, we thus have a set of M nonlinear equations for the M unknowns in $\{\beta_j\}$, which can be solved numerically.

Note that in the calculation of $P_m(N_1, N_2)$ for node m, the arrival rate γ_2 of its transit packets should be set equal to the throughput rate of such transit packets multiplied by $1/A_m$ to account for retransmissions following rejections. Also, if γ_1 is the arrival rate of new input packets (offered load), then the throughput rate of such packets should be reduced by a fraction equal to the blocking probability of input packets at node m, which is

$$\beta_m + \sum_{N_2=0}^{N_T - N_I - 1} P_m(N_I, N_2).$$

An iterative solution technique incorporating the Newton–Raphson method was developed [8] for the special case of $N_I = N_T$ at each node, i.e., no input buffer limit control and the buffer pool is completely shared. It was found that the model is accurate when switching nodes have adequate buffers (for given external input rates) so that $\{\beta_j\}$ takes on small values. The model is thus useful for predicting buffer requirements to achieve small nodal blocking probabilities. (Note that in a procedure to determine such buffer requirements, although the model results may not be very accurate at intermediate steps of the procedure, the results at the termination of the procedure would be accurate.) The model is also useful for comparing the performance of various buffer capacity assignment schemes.

The above model was also employed in [41] to

[3] The model can be used for many types of packets. Only two types of packets are considered to reduce the computational requirements of the model.

study the design of input buffer limits that can effectively prevent throughput degradation due to insufficient buffers when the network is under a heavy external load. Both the analytic results in [41] and a subsequent simulation study [42] showed that input buffer limits can be designed to provide a very effective congestion control mechanism for temporary network overloads. For a detailed treatment of input buffer limits as a congestion control mechanism, see [41, 42]. A slightly different input buffer limit strategy was later proposed and studied by Saad and Schwartz [43] and independently by Kamoun et al. [44].

In [41] the *homogeneous network model* was also proposed to reduce the computational complexity of solving numerically the set of nonlinear equations in (3.15). Instead of considering a specific network topology, it is assumed that the network is 'homogeneous' so that each node has identical behavior. Specifically, the blocking probability is the same at each node. As a result, the evaluation of a single queueing network model representing a single switching node is sufficient at each iteration step of the numerical solution. The homogeneous network model permits us to examine the influence of the input buffer limit strategy on the network's performance without the computational complexity that goes with an arbitrary network topology.

4. Conclusions

We have provided in this paper and in [5] a tutorial treatment of both exact and approximate models based upon product-form queueing networks, that have proved to be useful for analyzing the performance of packet communication networks. Some approximate solution techniques have also been described.

We have tried to provide a general framework for the various models presented. First we observed that the class of product-form queueing networks with population size constraints [4] include open networks, closed networks and mixed networks as special cases [2]. Open network models are considered in Part 1 [5]. The topics there include: channel capacity assignment, optimal routing, distributions of chain ETE delays and a study of fairness among routing chains. All these problems may be thought of as resource allocation problems related to the sharing of finite-capacity communication channels in a packet network. Problems related to the sharing of finite nodal buffer pools are considered herein (Part 2). Queueing networks with closed chains and other forms of population size constraints are employed to study the performance of window flow controlled virtual channels, and strategies for buffer management and permit-oriented network congestion control. We observed that all these strategies (including window flow control) fall within the *general resource allocation scheme of differentiating packets into classes, and providing each class with a minimum allocation and a maximum limit of the shared resource.*

We have restricted our discussions to the formulation and application of models based upon product-form queueing networks only. An earlier survey article on this subject appeared in [45]. For a comprehensive survey of network design problems using the basic model in Part 1, the reader is referred to [46]. For an in-depth treatment of the subject of network flow and congestion control strategies discussed in Part 2, the reader is referred to [47].

References

[1] J.R. Jackson, Jobshop-like queueing systems, Management Sci. 10 (1963) 131–142.

[2] F. Baskett, K.M. Chandy, R.R. Muntz and F. Palacios, Open, closed and mixed networks of queues with different classes of customers, J. Assoc. Comput. Mach. 22 (1975) 248–260.

[3] M. Reiser and H. Kobayashi, Queueing networks with multiple closed chains: Theory and computational algorithms, IBM J. Res. Develop. 19 (1975) 283–294.

[4] S.S. Lam, Queueing networks with population size constraints, IBM J. Res. Develop. 21 (1977) 370–378.

[5] J.W. Wong and S.S. Lam, Queueing network models of packet switching networks, Part 1: Open networks, Performance Evaluation 2(1) (1982) 8–20.

[6] L. Kleinrock, Communication Nets – Stochastic Message Flow and Delays (McGraw-Hill, New York, 1964).

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

[8] S.S. Lam, Store-and forward buffer requirements in a packet switching network, IEEE Trans. Comm. COM-24 (1976) 394–403.

[9] K.M. Chandy, U. Herzog and L.S. Woo, Parametric analysis of queueing networks, IBM J. Res. Develop. 19 (1975) 36–42.

[10] M. Reiser and S. Lavenberg, Mean value analysis of closed multichain queueing networks, J. Assoc. Comput. Mach. 27 (1980) 313–322.

[11] S.S. Lam and Y.L. Lien, A tree convolution algorithm for the solution of queueing networks, Tech. Rept. TR-165, Department of Computer Sciences, University of Texas at Austin, 1981; Comm. ACM, to appear.

[12] L. Pouzin, Presentation and major design aspects of the CYCLADES computer network, Proc. 3rd Data Communications Symposium, St. Petersburg, Florida, 1973.

[13] V. Cerf and R. Kahn, A protocol for packet network intercommunication, IEEE Trans. Comm. COM-22 (1974) 637–648.

[14] V. Ahuja, Routing and flow control in Systems Network Architecture, IBM Systems J. 18 (1979) 298–314.

[15] H. Opderbeck and L. Kleinrock, The influence of control procedures on the performance of packet-switched networks, Proc. Nat. Telecommunications Conference, San Diego, 1974.

[16] M. Reiser, A queueing network analysis of computer communication networks with window flow control. IEEE Trans. Comm. COM-27 (1979) 1199–1209.

[17] S.S. Lam and Y.L. Lien, Modeling and analysis of flow controlled packet switching networks, Proc. 7th Data Communications Symposium, Mexico City, 1981.

[18] S.S. Lam and Y.L. Lien, Optimal routing in networks with flow controlled virtual channels, Performance Evaluation Rev. 11 (1) (1982) 38–46.

[19] S.S. Lam, Dynamic scaling and growth behavior of queueing network normalization constants, Tech. Rept. TR-148, Department of Computer Science, University of Texas at Austin, 1980; J. Assoc. Comput. Mach. 29 (1982) 492–513.

[20] J.P. Buzen, Computational algorithms for closed queueing networks with exponential servers, Comm. ACM 16 (1973) 527–531.

[21] J.D.C. Little, A proof of the queueing formula: $L = \lambda W$, Oper. Res. 9 (1961) 383–387.

[22] M. Pennotti and M. Schwartz, Congestion control in store and forward tandem links, IEEE Trans. Comm. COM-23 (1975) 1434–1443.

[23] S.S. Lam, A simple derivation of the MVA and LBANC algorithms from the convolution algorithm, Tech. Rept. TR-184, Department of Computer Sciences, University of Texas at Austin, 1981.

[24] Y.L. Lien, Modeling and analysis of flow-controlled computer communication networks, Ph.D. Thesis, Department of Computer Sciences, University of Texas at Austin, 1981.

[25] S.S. Lam and Y.L. Lien, An analysis of the tree convolution algorithm for queueing networks, Tech. Rept. TR-166, Department of Computer Sciences, University of Texas at Austin, 1981.

[26] M. Gerla and P.O. Nilsson, Routing and flow control interplay in computer networks, Proc. 5th ICCC, Atlanta, 1980.

[27] J.M. McQuillan, G. Falk and I. Richer, A review of the development and performance of the ARPANET routing algorithm, IEEE Trans. Comm. COM-26 (1978) 1802–1811.

[28] C.E. Agnew, On quadrative adaptive routing algorithms, Comm. ACM 19 (1976) 18–22.

[29] R. Gallager, An optimal routing algorithm using distributed computation, IEEE Trans. Comm. COM-25 (1977) 73–85.

[30] L. Fratta, M. Gerla and L. Kleinrock, The flow deviation method: An approach to store-and-forward network design, Networks 3 (1973) 97–133.

[31] M. Gerla, The design of store-and-forward networks for computer communications, Ph.D. Dissertation, Department of Computer Sciences, UCLA, 1973.

[32] P.J. Schweitzer and S.S. Lam, Buffer overflow in a store-and-forward network node, IBM J. Res. Develop. 20 (1976) 542–550.

[33] M.A. Rich and M. Schwartz, Buffer sharing in computer-communication network nodes, IEEE Trans. Comm. COM-25 (1977) 958–970.

[34] M. Irland, Buffer management in a packet switch, IEEE Trans. Comm. COM-26 (1978) 328–337.

[35] F. Kamoun and L. Kleinrock, Analysis of shared finite storage in a computer network environment under general traffic conditions, IEEE Trans. Comm. COM-28 (1980) 992–1003.

[36] W.L. Price, Data network simulation experiments at the national physical laboratory, Comput. Networks 1 (1977).

[37] A. Giessler, J. Haenle, A. Koenig and E. Pade, Packet networks with deadlock-free buffer allocation – An investigation by simulation, Comput. Networks 2 (1978) 191–208.

[38] D. Davies, The control of congestion in packet switching networks, IEEE Trans. Comm. COM-23 (1975) 546–550.

[39] J.W. Wong and M.S. Unsoy, Analysis of flow control in switched data networks, proc. IFIP Congress 77 (1977) pp. 315–320.

[40] M.S. Unsoy, Credit-oriented congestion control in two-level hierarchical packet switching networks, Ph.D. Thesis, University of Waterloo, 1980.

[41] S.S. Lam and M. Reiser, Congestion control of store-and-forward networks by input buffer limits – An analysis, IEEE Trans. Comm. COM-27 (1979) 127–134.

[42] S.S. Lam and Y.L. Lien, Congestion control of packet communication networks by input buffer limits – A simulation study, IEEE Trans. Comput. C-30 (1981) 733–742.

[43] S. Saad and M. Schwartz, Input buffer limiting mechanisms for congestion control, ICC '80 Conf. Rec. (1980) 23.1.1–23.1.5.

[44] F. Kamoun, A. Belguith and J.L. Grange, Congestion control with a buffer management strategy based on traffic priorities, Proc. 5th Internat. Conf. Comp. Comm. (1980) pp. 845–850.

[45] J.W. Wong, Queueing network modeling of computer communication networks, ACM Comput. Surveys 10 (1978) 343–352.

[46] M. Gerla and L. Kleinrock, On the topological design of distributed computer networks, IEEE Trans. Comm. COM-25 (1977) 48–60.

[47] M. Gerla and L. Kleinrock, Flow control: A comparative survey, IEEE Trans. Comm. COM-28 (1980) 553–574.