REFERENCES

[1] H. Kobayashi and A. G. Konheim, "Queueing models for computer communication system analysis," *IEEE Trans. Commun.*, vol. COM-25, pp. 2–29, Jan. 1977.
[2] J. Hsu and P. J. Burke, "Behavior of tandem buffers with geometric input and Markovian output," *IEEE Trans. Commun.*, vol. COM-24, pp. 358–360, Mar. 1976.
[3] J. A. Morrison, "Two discrete-time queues in tandem," *IEEE Trans. Commun.*, vol. COM-27, pp. 563–573 Mar. 1979.
[4] D. Towsley and J. K. Wolf, "On statistical analysis of queue lengths and waiting times for statistical multiplexers with ARQ retransmission schemes," *IEEE Trans. Commun.*, vol. COM-27, pp. 693–702, Apr. 1979.
[5] B. Kim, D. Towsley, and J. K. Wolf, "On discrete time queueing systems," in *Conf. Rec. Int. Conf. Commun.*, June 1979, pp. 43.4.1–43.4.5.
[6] A. G. Konheim and M. Reiser, "Delay analysis for tandem networks," in *Conf. Rec. Int. Conf. Commun.*, Chicago, IL, pp. 12.2, 265–269.
[7] L. Kleinrock, *Queueing systems*, vol. 2. New York: Wiley, 1975.
[8] J. R. Jackson, "Network of waiting lines," *J. Oper. Res.*, vol. 5, pp. 518–521, 1957.
[9] S. Stidham, Jr., "A last word on $L = \lambda\omega$," *Oper. Res.*, vol. 22, pp. 417-421, 1974.
[10] L. Kleinrock, *Queueing Systems*, vol. 1. New York: Wiley, 1975.
[11] 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. Ass. Comput. Mach.*, vol. 22, pp. 248–260, Apr. 1975.
[12] K. Bharath-Kumar, "Discrete time queueing networks: Modeling, analysis and design," Ph.D. dissertation, Dep. Elec. Eng., University of Hawaii, Honolulu, HI, May 1979.

Dynamic Flow Control in Store-and-Forward Computer Networks

PARVIZ KERMANI AND LEONARD KLEINROCK, FELLOW, IEEE

Abstract—**In a recent paper we presented an analysis of flow control in store-and-forward computer communication networks using a token mechanism. The analysis assumed equilibrium conditions for a selected set of system parameters which were not dynamically adjusted to stochastic fluctuations in the system load; this mechanism was referred to as "static flow control." In this paper we study a "dynamic flow control" in which parameters of the system are dynamically adjusted to match the availability of resources in the network. Based on Markov decision theory, an optimal policy to dynamically select the number of tokens is formulated. Because an exact solution to the problem is extremely difficult, an effective heuristic solution to the problem is presented. Numerical results are given and it is shown that the throughput-delay performance of a network is better with dynamic control than with static control.**

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication without oral presentation. Manuscript received June 13, 1979; revised June 27, 1979. This work was supported by the Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-77-C-0272.

P. Kermani was with the Department of Computer Science, University of California, Los Angeles, CA 90024. He is now with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

L. Kleinrock is with the Department of Computer Science, University of California, Los Angeles, CA 90024.

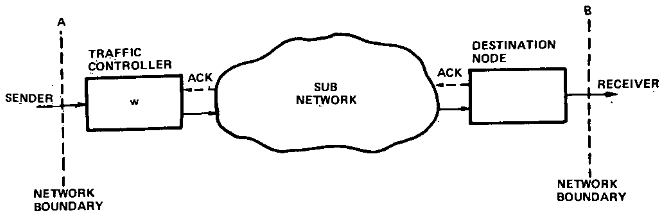

Fig. 1. Structure of a network.

I. INTRODUCTION

In a recent paper we presented an analysis of flow control in store-and-forward computer communication networks, using a token mechanism [1]. With a token mechanism, the total number of unacknowledged messages in a network between a source-destination pair is kept limited. The network was modeled as in Fig. 1, in which the traffic controller (TC) is responsible for keeping the number of unacknowledged messages below w, the limit on the number of tokens (for simplicity, we equivalently refer to this number as the *token limit*). The destination node has a finite number of buffers; when the buffer is full, any messages arriving at the destination are rejected and no acknowledgments for them are sent back to the TC. Messages which are not acknowledged within a timeout period (τ) are retransmitted from the TC. A "heavy traffic" assumption was made in [1] and the study was concerned with the maximum throughput of the network within the boundaries shown in Fig. 1. It was shown that for each token limit, there is an optimal timeout which results in a maximum throughput. The analysis assumed equilibrium conditions for a selected set of parameters which were not dynamically adjusted to stochastic fluctuations in the system load. These parameters are: the token limit, w: the timeout period, τ; the average message length, $1/\mu$; the network and the destination-node channel capacities, C_1 and C_2, respectively; and the destination-node buffer size, B. This mechanism was referred to as "static flow control."

In static flow control, parameters of the control mechanism are not dynamically adjusted to the network load. Due to the finite buffer size at the destination node, messages may be rejected and will require retransmission. The extra network traffic due to retransmissions results in an increased round-trip delay causing frequent timeouts, and consequently more retransmissions; thus we see the potential for a dangerous positive feedback situation. We note that if there were a further control to "slow down" the input of new messages to the network by reducing the token limit when the destination buffer begins to fill, then fewer messages would be rejected (or lost) at the destination node and thus, fewer retransmissions would be necessary. This is the idea behind the dynamic flow control mechanism studied in this paper.

We consider a mechanism located at the destination node which supervises buffer occupancy there; let us call this mechanism a traffic director (TD). At the disposal of the TD there are a number of different token limites which it can choose. When the occupancy of the buffer increases, the TD signals the TC at the source node to change its token limit to a smaller one; when the occupancy decreases, the TD notifies the TC to change the token limit to a larger one. This notification is carried out by control packets which are sent from the TD to the source node TC. The token limit used by the TC determines the input rate to the network; therefore, the above mechanism dynamically controls the input flow to the network. To prevent excessive overhead (as with acknowledgments), control packets are piggybacked on messages transmitted from the destination node to the source node; only if there is no such message is a control packet sent by itself.

The control mechanism operates as follows. A *decision table* is stored at the destination node which specifies the *best* token limit to be used for different *states* of the network. Knowing the state of the network, the TD can then use the decision table to decide upon the best token limit it should send to the TC. The questions which arise at this point are:

1) How is a network state specified?

2) What is meant by the best token limit?

3) How is the decision table set up?

To answer these questions we develop a mathematical model of the system based on Markov decision theory. We model the destination node as a finite waiting-room queueing system with K possible input rates (K is also the number of different token limits; see below), for which we wish to find an optimal selection policy for input rates such that the long-run average throughput of the buffer is maximal, and the average delay is minimal. (As we will see, delay is not due only to the destination node.) After defining a reward criteria, we derive a maximal reward-rate stationary operating policy for the finite queue; therefore, the problem will be reduced to the optimal control of a finite queue with variable input rates. As we will see below, due to special characteristics of the system an exact solution to the mathematical model is very laborious; therefore, we develop a heuristic solution to the problem.

In Section II the model is formulated as a discrete-time multistate-variable Markov decision process, which we refer to as a $\hat{T}_r$-cycle-delay decision process ($\hat{T}_r$ is the round-trip delay-see Section II-A). A heuristic solution to this Markov process is presented in Section III. In Section IV we discuss implementation of the heuristic optimal policy, and in Section V we further elaborate on 1-cycle-delay and continuous-time decision processes which we use to generate the decision table. Finally, in Section VI we present some numerical results.

II. THE MODEL

We consider the destination node of a network (Fig. 1) as a buffer storage of size B feeding an output transmission line with capacity C_2 (bits/s). Messages which arrive at the storage after passing through the network are buffered (if necessary) and are then transmitted to the receiving user. For the arrival process we make the following assumption:

Assumption 1: The counting process of message arrivals to the buffer is assumed to be Poisson with arrival rate $\lambda \in \Lambda$ (msg/s), where $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_K\}$ is the set of possible arrival rates.

We also assume that message lengths are exponentially distributed with an average message length of $1/\mu$ (bits). With this assumption the transmission times of messages out of the buffer are exponentially distributed with rate $\gamma_1 = \mu C_2$.

For reasons which will be clear shortly, we study a discrete-time model with very small time increments δ (e.g., $\delta = 4$ ms), and observe the system at these time intervals. This discrete approximation to the continuous case causes some inconveniences when we describe the state transition probabilities. We assume that at most one departure and one arrival may occur at each time slice (of δ s). This means that at the beginning of each time interval a new message may arrive in the buffer with probability $\lambda\delta$ (the result of a Bernoulli trial); thus the average number of arrivals per second is λ. The transmission times of messages are chosen independently from a geometric distribution such that, for $\hat{\gamma}_2 = \gamma_2\delta < 1$

$$s_n = \hat{\gamma}_2(1 - \hat{\gamma}_2)^{n-1} \tag{1}$$

where s_n is the probability that a message transmission time is exactly n time units long (i.e., that its service time is $n\delta$ s). The average transmission time is therefore

$$\hat{X}_2 = \frac{1}{\hat{\gamma}_2} \text{ time units} = \frac{\delta}{\hat{\gamma}_2} \text{ s} = \frac{1}{\mu C_2} \text{ s}. \tag{2}$$

Regarding the order in which events take place at the end of a time slice, we assume that departures take place before arrivals. This means that a message which has completed its transmission leaves the system and then a message which has arrived is delivered into the buffer. (If the buffer is full and there is no departure, the arriving message is lost and must be retransmitted.)

For notational purposes with the discrete-time version we use variables under a circumflex ($\hat{\ }$); in particular, we let $\hat{\lambda}_k = \lambda_k\delta$, $\hat{\lambda} = \lambda\delta$, and $\hat{\gamma}_2 = \gamma_2\delta$. The round-trip delay is designated by the random variable $\tilde{t}_r$ whose average is T_r and we let $\hat{T}_r = T_r/\delta$. For the one-way delay the corresponding notations will be $\tilde{t}$, T, and $\hat{T}$ ($= T/\delta$).

A. The State Space

The selection of the appropriate number of tokens by the TD (hence of the input rate to the buffer storage—see below) should be based on the occupancy of the buffer. Having decided on the new number of tokens, the TD notifies the TC of this decision through the control signaling process described earlier. When the TC receives this notification (after the one-way return delay of $\tilde{t}$ s), it adjusts the token limit, and the input rate to the network is changed corresponding to the new token limit. We assume that this change of input rate is instantaneous.

After the input rate is changed, messages which are admitted with the new rate reach the destination node after $\tilde{t}$ s. Therefore, there is a (random) time gap from the time the TD decides upon a new input rate until the result of this input rate becomes effective at the destination node. When this time gap is random, the resulting model becomes extremely difficult to analyze and in fact, no such analysis has been reported in the literature. To render the model amenable to analysis, we further assume that the time gap between the moment that a decision is made until the time it becomes effective is a *constant* interval of duration T_r s, the average round-trip delay. Since the input rate to the network varies, the average round-trip delay fluctuates as well; however, we take T_r as the average of the round-trip delay due to different possible choices of token limits (or input rate).

With the above considerations, we note that the decision at time t becomes effective at the destination node at time $t + T_r$. Considering the fact that a decision on the number of tokens is based on the buffer occupancy, the decision at time t should be based on the occupancy of the buffer at time $t + T_r$. Thus the state description at time t should contain information, not only regarding the buffer occupancy at time t, but also the input rates to the buffer in the interval $[t, t + T_r]$. (These input rates have been decided upon n the interval $[t - T_r, t]$.) In order to include this information in the state variable we consider a discrete-time model in which the system is observed every δ s; when a decision is made it becomes effective after $\hat{T}_r(=T_r/\delta)$ time units. With the above considerations, the state of the system at time $\hat{t}(=t/\delta)$ is represented by $X(\hat{t})$ given by

$$X(\hat{t}) = \{n(\hat{t}), \hat{\lambda}(\hat{t}, 1), \hat{\lambda}(\hat{t}, 2), \cdots, \hat{\lambda}(\hat{t}, \hat{T}_r - 1)\} \tag{3}$$

where $0 \leqslant n(\hat{t}) \leqslant B$ is the buffer occupancy at time t and $\hat{\lambda}(\hat{t}, k)$ (measured in msg/δ s) is the input rate to the buffer at time $(\hat{t} + k)$, $1 \leqslant k \leqslant \hat{T}_r$. Note that $\hat{\lambda}(\hat{t}, k)$ is the result of the decision which was made $\hat{T}_r - k$ time unites earlier than $\hat{t}$. We may also label the states by integers $0, 1, \cdots, N$, where $(N + 1)$ is the size of the state space; in equilibrium state, i is defined as

$$i = \{n_i, \hat{\lambda}_i^1, \hat{\lambda}_i^2, \cdots, \hat{\lambda}_i^{\hat{T}_r - 1}\} \tag{4}$$

(n_i and $\hat{\lambda}_i^k$, $1 \leqslant k < \hat{T}_r$ are defined in (5) below). The set of all states in the state space will be referred to by S. The cardinality of the set S, $|S|$, is

$$|S| = N + 1 = (B + 1)K^{(\hat{T}_r - 1)}$$

where K is the number of different token limits. For convenience we use the symbol "$\langle = \rangle$" to associate a state, say $X(\hat{t})$, to its label, i.e.

$$X(\hat{t}) \langle = \rangle i$$

if (5)

$$n(\hat{t}) = n_i \qquad \text{and } \hat{\lambda}(\hat{t}, k) = \hat{\lambda}_i^k \; 1 \leqslant k < \hat{T}_r.$$

B. The Action Space

We consider a finite set of token limits $W = \{w_1, w_2, \cdots, w_K\}$ at the disposal of the TD. It is shown in [1 and 2] that for a given network and for a token limit w, a timeout τ can be chosen such that the throughput of the network becomes maximal. Hence, when the TD decides upon a token limit, it also determines the appropriate timeout $\tau \in \{\tau_1, \tau_2, \cdots, \tau_K\}$ and notifies the TC in the source node (in the manner described above). The set W is the set of actions in our model; however, because each (w, τ) uniquely determines an input rate [1], we might as well assume that the corresponding set of input rates $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_K\}$ (or $\{\hat{\lambda}_1, \hat{\lambda}_2, \cdots, \hat{\lambda}_K\}$) is the set of actions. In the discussion below, we will refer to either of these two sets (W or A) as the action space, i.e.

$$A = \Lambda \qquad \text{or } A = W. \tag{6}$$

Corresponding to the above sets of input rates, timeouts, and token limits, there is a set of network delays $\{T_1, T_2, \cdots, T_K\}$ where T_k is the network delay when w_k is being used.

C. The Policy Space

A policy specifies a token limit (or equivalently an input rate) which should be sent to the TC. More precisely, by a policy f we mean a decision rule which says, given that the system is a state i, we should use $\hat{\lambda}_k$ (or w_k) as the input rate to the buffer which will be effective $\hat{T}_r$ time units later, that is

$$f(i) = k. \tag{7}$$

We are implicitly using policies that are functions of the present state of the system; policies in this class are known as "stationary policies," and will be denoted by F.

D. The State Transition Probabilities

Let the states of the system at times $\hat{t}$ and $\hat{t} + 1$ be $X(\hat{t}) \langle = \rangle i$, as defined in (3), and $X(t + 1) \langle = \rangle j$, where

$$X(t + 1) = \{n(\hat{t} + 1), \hat{\lambda}(\hat{t} + 1, 1)\, \hat{\lambda}(\hat{t} + 1, 2), \cdots, \hat{\lambda}(\hat{t} + 1, \hat{T}_r - 2), \hat{\lambda}(\hat{t} + 1, \hat{T}_r - 1)\}. \tag{8}$$

The input rate that the TD selects at time $\hat{t}$ to be effective at time $\hat{t} + \hat{T}$, $\hat{\lambda}_{f(X(\hat{t}))}$, becomes part of the state vector at time $\hat{t} + 1$; therefore the components of state vectors $X(\hat{t})$ and $X(\hat{t} + 1)$ are related to each other as follows:

$$\hat{\lambda}(\hat{t} + 1, k) = \begin{cases} \hat{\lambda}(\hat{t}, k + 1) & 1 \leqslant k \leqslant \hat{T}_r - 2. \\ \hat{\lambda}_{f(X(t))} & k = \hat{T}_r - 1 \end{cases} \tag{9}$$

The last compenent of state vector $X(\hat{t} + 1)$, $\hat{\lambda}(\hat{t} + 1, \hat{T}_r - 1)$, is the decision which is made at time $\hat{t}$. The number of messages in the buffer at time $\hat{t} + 1$ is different from $n(\hat{t})$; this is the result of the possible arrival of a message, with probability $\hat{\lambda}(\hat{t}, 1)$, or the possible departure of a message, with probability $\hat{\gamma}_2$. With the above considerations, the equilibrium transition probabilities under policy f between states i and j, $p_{ij}(f)$, defined as

$$p_{ij}(f) \equiv \Pr[X(\hat{t} + 1) \langle = \rangle j \mid X(\hat{t}) \langle = \rangle i \text{ and policy } f \text{ is used}]$$

can be easily derived (see [2] for such a derivation). Note that if $\hat{\lambda}_j^k \neq \hat{\lambda}_i^{k+1}$ or $\hat{\lambda}_j^{\hat{T}_r - 1} \neq \lambda_{f(i)}$, then $P_{ij}(f) = 0$. We refer to the matrix of transition probabilities under policy f by $P(f)$, i.e., $P(f) = \{p_{ij}(f)\}$.

E. The Performance Criteria and the Reward Function

The objective of our analysis is to find an optimal policy which, in the long run, maximizes the average throughput and minimizes the average delay of the system; this objective is reflected in our formulation through the reward criteria. We are faced with incorporating the opposing effects of throughput and delay in a single objective function. To this end, we choose to use the following performance measure in our decision problem: Let λ and T be the throughput and the corresponding delay of a system; then the reward of this system is defined as

$$R = \alpha\lambda - T \qquad \alpha \geqslant 0. \tag{10}$$

Note that a high throughput and a low delay result in a large value for R, which reflects the efficiency of the network.[1] The coefficient α reflects our relative preference for large throughput with respect to low delay. $\alpha = 0$ corresponds to a situation where we cannot accept any delay (we will see shortly that this value of α results in zero throughput, and hence, zero delay). The other extreme is when $\alpha \to \infty$, which reflects the case that throughput is the only measure of performance.

Having defined the objective function, we must now find the *expected immediate reward* (the reward to be expected in the next transition [5]) for each state. We use $R_i(f)$ for the expected immediate reward for state i under policy f. In [2] it is shown that for a state i given by (4), the expected immediate reward rate is

$$R_i(f) = \begin{cases} \alpha\hat{\lambda}_i^1 - (n_i + \hat{\lambda}_i^1\hat{T}_i^1) & n_i < B \\ \alpha\hat{\lambda}_i^1\hat{\gamma}_2 - [n_i + \hat{\lambda}_i^1\hat{\gamma}_2\hat{T}_i^1 + \hat{\lambda}_i^1(1-\hat{\gamma}_2)\hat{\tau}_i^1] & \\ & n_i = B. \end{cases} \quad (11)$$

The argument behind (11) is the following: In a transition from state i to any succeeding state, each one of the n_i messages in the buffer suffers one unit of delay (the term $-n_i$). If a message arrives (with probability $\hat{\lambda}_i^1$), and is accepted, it is guaranteed delivery to the receiver, hence the gain for the throughput is $\alpha\hat{\lambda}_i^1$. On the other hand each arriving message brings with it an amount of delay equal to $\hat{T}_i^1$ (which corresponds to the input rate $\hat{\lambda}_i^1$) which it has suffered in passing through the network; therefore, the loss due to delay is $\hat{\lambda}_i^1\hat{T}_i^1$. When the buffer is full (the second line of (11)), an incoming message can be accepted only if the message which is being sent out of the buffer completes transmission. If not, the incoming message is rejected and has to be retransmitted after $\hat{\tau}_i^1$ seconds. (For a detailed derivation of (11) see [2].) We designate the vector of expected immediate reward rates under policy f by $R(F)$, i.e., $R(f) = \{R_i(f)\}$.

F. Statement of the Problem

The above formulation specifies a Markov decision process (chain) that represents a system in which there is a constant time gap between the instant a decision is made until the time that decision becomes effective; we refer to this process as a *$\hat{T}_r$–cycle–delay* Markov decision process. Processes similar to this are encountered in inventory problems, where there is a *leadtime* for an order to arrive [6], [7].

The set of stationary policies in our decision process is actually a subset of the set of all policies. Under stationary policy f the average reward per unit time is designated by the vector $g(f) = \{g_i(f)\}$, where $g_i(f)$ is the long-run expected reward per unit time given that the process initially starts in state i. $g(f)$ is given by

$$g(f) = \lim_{n\to\infty} \left\{ \frac{1}{n+1} \sum_{t=0}^{n} [P(f)]^t R(f) \right\} \quad (12)$$

where $[P(f)]^t$, the tth power of $P(f)$, is the t-step probability transistion matrix under stationary policy f. If f is a policy such that the Markov chain defined by $P(f)$ is completely ergodic, i.e., if there only one recurrent chain in the system, then all components of $g(f)$ are equal [5]. Our objective is to find a (stationary) policy f^* which maximizes each component of the expected average reward per unit time, i.e.,

$$g(f^*) = \max_{f\in F} \{g(f)\}. \quad (13)$$

It can be shown that for a bounded reward function, when the action space and state space are finite, such a stationary policy always exists [8], [9]. These conditions are met in our problem.

III. SOLUTION TO THE PROBLEM–THE LOOK-AHEAD POLICY

The $\hat{T}_r$-cycle-delay Markov decision process formulated in the last section can, in principle, be solved for an optimal stationary policy by using any numerical technique, e.g., the policy-iteration method [5]. Unfortunately, for any realistic set of parameters, the size of the problem (especially the state space) becomes too large; it is extremely laborious, if not impossible, to solve such a large problem by any conventional means. As an example, consider a system in which the round-trip delay $T_r = 0.2$ s, $B = 10$, and $K = 5$. If we use $\delta = 4$ ms as our time unit, then $\hat{T}_r = 50$, and each state vector has 50 components. The size of the state space for this system is

$$|S| = 11 \times 5^{49} \approx 10^{35}.$$

Considering the fact that each cycle of the policy-iteration method requires solution of a set of $|S|$ linear simultaneous equations, a direct solution to this problem is out of the question; hence we must resort to a heuristic solution. To this end we consider the following:

1) The major difficulty in computation arises from the long round-trip delay. For a round-trip of 2 units ($\hat{T}_r = 2$), the size of the state space for the above example becomes $|S| = 11 \times 5 = 55$ and when $\hat{T}_r = 1$, it is only 11. ($\hat{T}_r = 1$ results in a 1-cycle-delay Markov decision process in which a decision becomes effective in the next transition; this is the case with an ordinary Markov decision process and we make extensive use of this case in our heuristic solution.)

2) Consider a 1-cycle-delay decision process ($\hat{T}_r = 1$). For this system the only information necessary to select the token limit is the buffer occupancy; in fact in this case the complete state of the system is represented simply by the buffer occupancy. When there is a leadtime of $\hat{T}_r > 1$ for a decision to become effective, the process makes use of the input rate components of a state vector to estimate the future occupancy of the buffer at $(\hat{T}_r - 1)$ time units later. Based on this prediction, it then decides on a token limit (or an input rate) to be effective at $\hat{T}_r$ time units later. Note also that if at time $\hat{t}$ we provide the decision process with the future buffer occupancy at, say, k time units later ($k < \hat{T}_r$), then from the decision process point of view, the leadtime for a decision to become effective reduces to $\hat{T}_r - k$.

The above considerations lead us to develop the following heuristic solution for the decision process. Consider state i (4) of a $\hat{T}_r$-cycle-delay Markov decision process. The component $\hat{\lambda}_i^k$ in the state vector is the input rate to the buffer k time units after the system is in state i. Knowing the future

[1] Another possible choice for the objective function may be "power," defined as the ratio of throughput and delay (λ/T) [3], [4]; however, because (10) combines λ and T linearly, the resulting analysis is more tractable.

input rates to the buffer, we can find the *expected* occupancy of the buffer at $\hat{T}_l$ time units later ($0 \leqslant \hat{T}_l < \hat{T}_r$); we refer to $\hat{T}_l$ as the *look-ahead* time and designate the expected buffer occupancy after $\hat{T}_l$ transitions from state i by $\bar{n}_i(+\hat{T}_l)$. When the set of policies and the state transition probabilities are properly defined, the vector

$$i' = \{\bar{n}_i(+\hat{T}_l), \hat{\lambda}_i^{\hat{T}_l+1}, \hat{\lambda}_i^{\hat{T}_l+2}, \cdots, \hat{\lambda}_i^{\hat{T}_r-1}\} \tag{14}$$

specifies a state of a $(\hat{T}_r - \hat{T}_l)$-cycle-delay decision process. We designate the set of states of this process by S'. For this process, a (stationary) policy f' is defined as a function from the state space S' to the action space A defined by (6) i.e., f': $S' \to A$. If $|S'|$ is not too large, we can find an optimal stationary policy f'^* for this process, and use this policy to find a (sub)optimal stationary policy $f_s{}^*$ for the original $\hat{T}_r$-cycle-delay decision process, so that

$$f_s{}^*(i) = f'^*(i') \tag{15}$$

where i' is given by (14). The expected reward rate (and the long-run average throughput) for policy $f_s{}^*$ is found to be very close to the optimal expected reward rate for the original process. (The numerical results in Section VI support this claim.) We refer to policy $f_s{}^*$ as the optimal $\hat{T}_l$-look-ahead policy, where $\hat{T}_l$ is the look-ahead time. We summarize the above procedure in the algorithm given below.

Algorithm 1–The Look-Ahead policy:

1) For the $\hat{T}_r$-cycle-delay Markov decision process, find a look-ahead interval $0 \leqslant \hat{T}_l < \hat{T}_r$, such that the resulting $(\hat{T}_r - \hat{T}_l)$-cycle-delay Markov decision process is solvable (i.e., the computational cost is acceptable).

2) Use any appropriate technique (e.g., the policy-iteration method [5]) to find an optimal policy f'^* for the $(\hat{T}_r - \hat{T}_l)$-cycle delay decision process.

3) For each state i of the $\hat{T}_r$-cycle-delay decision process find $\bar{n}_i(+\hat{T}_l($ and the corresponding state i (14) for the $(\hat{T}_r - \hat{T}_l)$-cycle-delay decision process.[2] ($\bar{n}_i(+ T_l)$ should be rounded to its nearest integer.)

4) The (sub)optimal decision at state i is given by $f_s{}^*(i)$, where $f_s{}^*(i) = f'^*(i')$.

The above heuristic solution plays a crucial role in making possible the use of the policy-iteration method to solve optimization problems involving long round-trip delays and large numbers of possible token limits.

IV. IMPLEMENTATION OF THE LOOK-AHEAD POLICY–THE DECISION TABLE

We have so far answered two of the three questions we posed in the opening remarks of this paper. In this section we address the last question and describe the structure of the decision table, how it is set up, and how it is used by the TD.

In Section III we described how a (sub)optimal stationary policy for the $\hat{T}_r$-cycle-delay Markov decision process could be found heuristically by using the optimal policy for the corresponding $(\hat{T}_r - \hat{T}_l)$-cycle-delay decision process. Once an optimal policy for this smaller size decision process is found, we can set up a decision table which has one entry for each state of the $(\hat{T}_r - \hat{T}_l)$-cycle-delay decision process. For each state, the corresponding (optimal) token limit is listed in a decision table; therefore the size of the decision table, N_{tbl}, is

$$N_{tbl} = (B + 1)K^{(\hat{T}_r - \hat{T}_l - 1)}. \tag{16}$$

The TD always stores the current state vector of the system; at time $\hat{t}$ this vector looks like $X(\hat{t})$ given in (3). Each time the TD must decide on a token limit, it calculates the expected occupancy of the buffer at time $\hat{t} + \hat{T}_l$, $\bar{n}(\hat{t} + \hat{T}_l)$, and sets up the following vector:

$$X'(\hat{t}) = \{\bar{n}(\hat{t} + \hat{T}_l), \hat{\lambda}(\hat{t}, \hat{T}_l + 1), \hat{\lambda}(\hat{t}, \hat{T}_l + 2), \cdots, \hat{\lambda}(\hat{t}, \hat{T}_r - 1)\}. \tag{17}$$

(In fact, the expected buffer occupancy should be rounded to its nearest integer.) $X'(\hat{t})$ is a state vector for the $(\hat{T}_r - \hat{T}_l)$-cycle-delay decision process, and the token limit for $X'(\hat{t})$ in the decision table is the one that the TD should use. In Table I we show a hypothetical decision table when $\hat{T}_l = \hat{T}_r - 1$, and the destination buffer size is $B = 6$. In this case the decision table specifies an optimal policy for a 1-cycle-delay decision process in which a decision becomes effective on the next transition (which is the case for an ordinary Markov decision process); therefore, the state of the process is simply the buffer occupancy. A decision is an optimal token limit for a state.

TABLE I
EXAMPLE OF A DECISION TABLE

State (buffer occupancy)	Decision (token limit chosen)
0	4
1	3
2	2
3	1
4	1
5	0
6	0

Note that when the buffer occupancy is 5 or 6, the optimal token limit is 0, i.e., for the particular value of α which is used for this table, the system is closed to further input when the occupancy is 5 or 6. (See Table II below for other examples of decision tables.) For Table I, $X'(\hat{t})$ given by (17) reduces to a single number, i.e.,

$$X'(\hat{t}) = \bar{n}(\hat{t} + \hat{T}_l) = \bar{n}(\hat{t} + \hat{T}_r - 1).$$

The choice of the look-ahead time, $\hat{T}_l$, should be based on two considerations: optimality of the decisions made by the TD and feasibility of solving the reduced decision process. For a small $\hat{T}_l$, the decisions made by the TD are closer to the optimal (note that for $\hat{T}_l = 0$, the TD's decisions are the optimal); however, the resulting $(\hat{T}_r - \hat{T}_l)$-cycle-delay decision process is large to solve, and also the size of the decision table becomes large, see (16). On the other hand, when $\hat{T}_l$ is large, the result of the decisions made by the TD may be far from the optimal; nevertheless, a large $\hat{T}_l$ also results in a small decision process to solve and a small decision table. The largest value of $\hat{T}_l$ is $\hat{T}_r - 1$, for which the state of the reduced decision process is simply the buffer occupancy. For $\hat{T}_l = \hat{T}_r - 2$, $X'(\hat{t})$ has two components which corresponds to the state of a 2-cycle-delay decision process. Although this system is not too large to solve (the

[2] For a derivation of $\bar{n}_i(+ T_l)$ the interested reader is referred to [2].

TABLE II

$B = 10$ $C_1 = C_2 = 50$ kbps
$W = \{0, 1, 2, \ldots, 20\}$ $1/\mu = 1000$ bits

$\alpha = 0.4$	State (buffer occupancy)	Prob.	Decision (token chosen)
	0	0.310	6
	1	0.232	5
	2	0.176	5
	3	0.128	4
	4	0.078	3
	5	0.047	3
	6	0.028	2
	7	0.014	1
	8	0.001	0
	9	0.000	0
	10	0.000	0
	Average token size		4.77
	Average buffer occupancy		1.78
	Average throughput		34.51
	Average buffer delay		0.05
	Average network delay		0.07
	Average total delay		0.12

$\alpha = 0.6$	State (buffer occupancy)	Prob.	Decision (token chosen)
	0	0.248	8
	1	0.199	8
	2	0.159	7
	3	0.123	6
	4	0.092	5
	5	0.066	5
	6	0.047	4
	7	0.031	3
	8	0.019	2
	9	0.009	2
	10	0.004	0
	Average token size		6.66
	Average buffer occupancy		2.38
	Average throughput		37.59
	Average buffer delay		0.06
	Average network delay		0.09
	Average total delay		0.15

size of the state space for $B = 10$ and $K = 5$ is 55), a problem which arises is that the resulting state transition probability matrix may have more than one chain; therefore, the solution of the decision process becomes very complex and time consuming. For values of $\hat{T}_l$ less than $\hat{T}_r - 2$, not only does the state space become large, but also we face the problem of multichain systems. For these reasons $\hat{T}_l = \hat{T}_r - 1$ is a convenient choice, and we have used this value of $\hat{T}_l$ in our numerical example. To support this choice of $\hat{T}_l$, we also used $\hat{T}_l = \hat{T}_r - 2$ and found that the performance of the network (in terms of throughput and delay) was almost identical with the case when $\hat{T}_l = \hat{T}_r - 1$.

V. 1-CYCLE-DELAY AND CONTINUOUS-TIME DECISION PROCESSES

When the look-ahead time $\hat{T}_l$ is equal to $\hat{T}_r - 1$, the decision table becomes an optimal (stationary) policy of a 1-cycle-delay (discrete-time) Markov decision process. The general formulation of $\hat{T}_r$-cycle-delay processes we presented in Section II also applies to discrete-time Markov decision processes.

When the time increment δ is small, decisions become effective very quickly; in the limit as $\delta \to 0$, the above discrete-time Markov decision process becomes a continuous-time process. In the continuous-time mode the rate of transition (rather than the probability of transition) out of a state is a function of the action taken in that state. Because in this process decisions effectively become instantaneous, the optimal reward rate for the continuous-time process is the best performance we can expect from the system, and the throughput for the optimal policy f^* is the highest that can be expected. Therefore, we will compare the performance of our look-ahead policy with the performance of this continuous-time model.

So far we have assumed that for a set of token limits W, we can use the results of the static flow control [1] to find the optimal input rates (or throughput) of the network $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_K\}$. In [1] it is shown that if the token limit w, channel capacity C_1, and the loss probability P_l of a network are known, then an optimal timeout (τ_t^*) can be found such that the throughput of the network becomes maximal. The problem which we face here is that the loss probability, P_l, is itself a function of the (optimal) policy that we use to select token limits. On the other hand the optimal policy is determined by using the set of input rates Λ; therefore, evaluation of the input rates $\{\lambda_k\}$ depends on the loss probability P_l, which is itself a function of the input rates. In [2] an iterative procedure has been developed to remove this recursion; the interested reader is referred to this reference.

VI. NUMERICAL RESULTS

In this section we present numerical examples for our dynamic-flow control, study optimality of the look-ahead policy, and investigate the effect of the parameter α on the throughput-delay performance of a network. Throughout our presentation we consider the normalized value of certain parameters; the normalization constant is $\bar{X}_1 = (1/\mu C_1)$, the average transmission time of a message on a network channel.

We start by studying the throughput-delay performance of a continuous-time model in which the effect of a decision is instantaneous. In this hypothetical system there is no time gap from the moment that a decision is made until it becomes effective. Hence, performance of the continuous-time model is the best that can be achieved.

As we pointed out earlier, the value of α reflects our relative preference for high throughput over low delay. Table II shows the effect of parameter α on the throughput-delay performance of the system. In this table the destination buffer size is $B = 10$, the channel capacities of the network and the destination node (C_1 and C_2) are both 50 kbits/s, the average message length is 1000 bits, and the set of allowable token limits is $W = \{0, 1, 2, \cdots, 20\}$. This table shows the optimal policies and their corresponding statistics for $\alpha = 0.4$ and $\alpha = 0.6$. It is seen that for $\alpha = 0.6$, larger token limits are used (and a higher throughput results); for example, when the buffer is empty (state 0) for $\alpha = 0.4$, $w = 6$ is the optimal decision, whereas when $\alpha = 0.6$, $w = 8$ becomes the best decision. Although the delay (and throughput) due to $w = 8$ is larger than $w = 6$, a token limit of 8 becomes the optimal decision because $\alpha = 0.6$ puts more weight on high throughput than $\alpha = 0.4$. It is seen that not all of the allowable token limits are used in either of the above cases; therefore, not only can the parameter α be used to control the throughput, but the largest token limit can be controlled also by a proper selection of α. Considering the fact that the largest token limit which is used reflects the number of buffers needed at the source node, the importance of this parameter becomes clear. Furthermore, the number of buffers required at the destination node is also determined by α (for example for $\alpha = 0.4$ a buffer size of

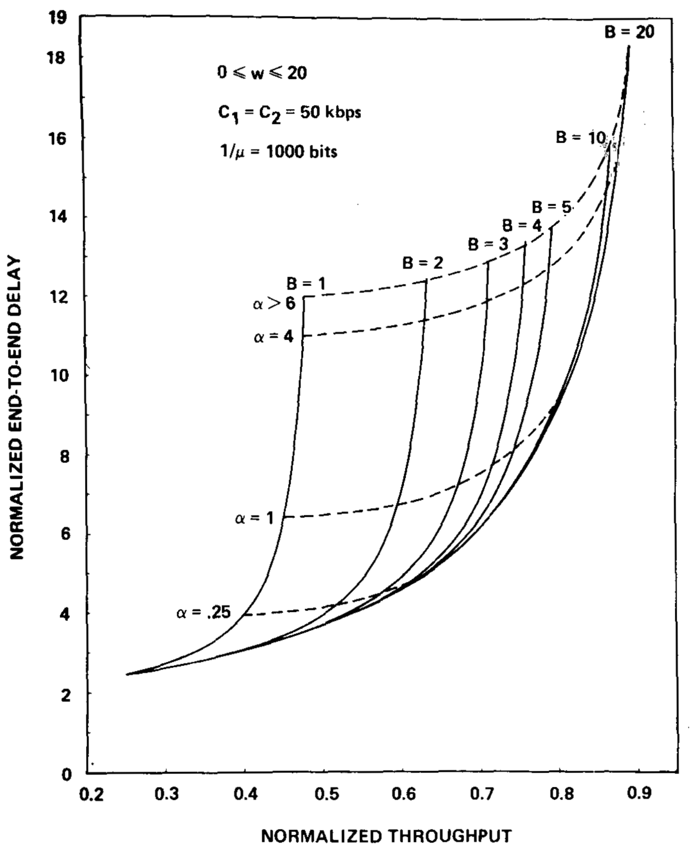

Fig. 2. Throughput-delay performance of continuous-time model for different buffer sizes.

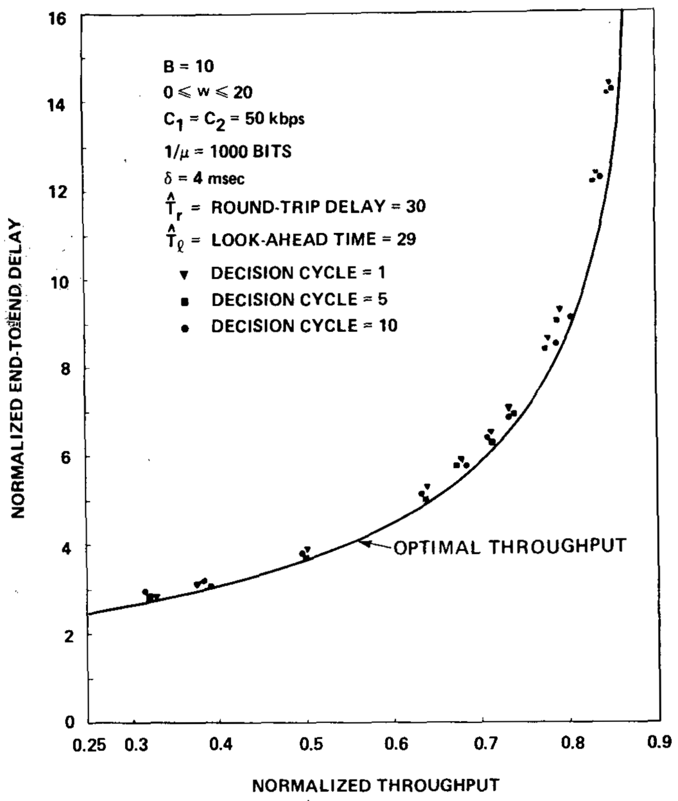

Fig. 3. Throughput-delay performance of look-ahead policy.

$B = 8$ is sufficient); therefore, this parameter can also be used to control the utilization of the destination node buffer.

Fig. 2 shows the throughput-delay (the end-to-end or the total delay) performance of the continuous-time model for different buffer sizes for the destination node. The set of allowable token limits and other parameters of the system are the same as in Table II. For a fixed B, the average throughput and delay increase as α increases. For a given value of α, an optimal decision is determined such that the expected (long-run) reward rate becomes maximal. From (10) we can relate the expected reward rate to the (long-run) average input rate (or throughput) and the average delay of the system,

$$\bar{R} = \alpha\bar{\lambda} - \bar{T}. \tag{18}$$

Equation (18) indicates that under an optimal policy, for each $\bar{\lambda}$ the corresponding $\bar{T}$ is the lowest that can be achieved (equivalently for each $\bar{T}$, $\bar{\lambda}$ is the maximal). If there were another policy that could generate lower delay for the same $\bar{\lambda}$, then $\bar{R}$ would not be maximal. Therefore, the throughput-delay curve for a fixed B in this figure defines a tight lower bound on delay and so can be viewed as the *optimal network performance* for our class of dynamic flow control policies.

When we include a leadtime for decisions to become effective, the throughput-delay performance of a network with a look-ahead policy will be inferior to the instantaneous feedback case. Fig. 3 shows the performance of the network under the look-ahead policy. In this figure the time interval δ is 4 ms (i.e., the state of the system is observed every 4 ms), and the destination node has 10 buffers ($B = 10$). The round-trip delay ($\hat{T}_r$) is assumed to be 30 time units for the entire range of throughput, and the look-ahead time is 29 (i.e., $\hat{T}_l = \hat{T}_r - 1$). The other parameters of the system are the same as in Fig. 2. The solid curve in Fig. 3 corresponds to a 1-cycle-delay (discrete-time) decision process (which is the counterpart of the continuous-time model); therefore, this curve represents the best performance that can be achieved in the discrete-time model from the look-ahead policy. In order to investigate its performance, we tested the look-ahead policy by simple simulation experiments.

In the simulation experiments we used $\hat{T}_r = 30$ and $\hat{T}_l = \hat{T}_r - 1 = 29$. The other parameters ($C_1$, C_2, $\cdots$, etc.) were the same as in Fig. 3. The decision table (Table I) was set up by finding an optimal policy for the corresponding 1-cycle-delay decision process (Section IV). Initial occupancy of the buffer, $n(0)$, and the input rates for the first 29 time units ($\hat{\lambda}(1), \hat{\lambda}(2), \cdots, \hat{\lambda}(29)$) were also initialized. (When the simulation run is long enough, say 20,000 time units, the initial values have no effect on the final statistics.) At each time unit, say $\hat{t}$, an input to the buffer and an output from the buffer (with probabilities $\lambda(\hat{t})$ and $\hat{\gamma}_2$, respectively) were generated by the Monte Carlo technique, and the buffer occupancy was updated. After collecting the statistics (the input rate and delay), an input rate for time $\hat{t} + \hat{T}_r$ was decided by using the decision table, and the simulation time was advanced. (Actually for decisions cycles greater than 1, the decisions were made periodically; see below.)

The points indicated by "▼" in Fig. 3 show the performance of the look-ahead policy when decisions are made at each time unit (decision cycle = 1). It is seen that the throughput-delay performance in this case is extremely close to the optimal. Considering that for $\hat{T}_l = \hat{T}_r - 1 = 29$

the performance of the look-ahead policy should be the worst, the excellent match between the optimal performance and the performance due to the look-ahead policy is most encouraging.

We pointed out earlier that control packets to carry decisions to the source node are piggybacked on messages going from the destination node to the source node, and if there are no such messages, control packets are transmitted by themselves. Therefore, signaling of control packets generates overhead traffic. For this reason it is not a good policy to decide on a new token limit at each time unit; it is better to make decisions periodically. In Fig. 3 we have also shown the performance of the look-ahead policy when decisions are made every 5 time units ("■"), or every 10 time units ("●"). (When the decision cycle is, say 5, the decision made at time $\hat{t}$ is used as the decision for times $\hat{t} + 1$ to $\hat{t} + 4$; at time $\hat{t} + 5$ a new decision is made.) For decision cycles larger than 10 the throughput-delay curves differed significantly from the optimal. Therefore, as long as the decision cycle is below a certain limit, the throughput-delay performance of the periodic look-ahead policy is not sensitive to the exact value of the decision cycle. This property is important, as a periodic decision reduces the overhead due to signaling of control packets.

So far we have assumed that the round trip delay, $\hat{T}_r$, is fixed for the whole range of throughput; in reality $\hat{T}_r$ is larger for higher throughputs. To investigate sensitivity of the throughput-delay performance to the round-trip delay, we used the simulation program described earlier, but changed the round-trip delay as the throughput increased (the look-ahead time was also changed such that $\hat{T}_l = \hat{T}_r - 1$). The points indicated by "●" in Fig. 4 correspond to the performance of the system when $\hat{T}_r$ varies. The solid curve in this figure is the throughput-delay curve for the 1-cycle delay decision process for which the round-trip delay is fixed at $\hat{T}_r = 30$. Fig. 4 indicates that the performance of the look-ahead policy is not sensitive to a variation in $\hat{T}_r$, and if a proper average round-trip delay is used, the performance is close to the optimal.

In order to investigate the sensitivity of the throughput-delay performance on the look-ahead time ($\hat{T}_l$), we simulated a network with parameters shown in Fig. 3, but we let $\hat{T}_l = \hat{T}_r - 2 = 28$. The performance in this case was almost identical to the case when $\hat{T}_l = 29$. In order to use a look-ahead time smaller than $\hat{T}_r - 2$, we must find an optimal policy for the corresponding n-cycle-delay Markov decision process, where $n > 2$. As pointed out earlier, the solution is fairly complicated and time consuming because the corresponding Markov process may have more than one irreducible chain. Considering the fact that a maximum look-ahead time (i.e., $\hat{T}_l = \hat{T}_r - 1$) results in a performance very close to the optimal (Fig. 3), $\hat{T}_r - 1$ is a convenient and good choice for the look-ahead time.

In Fig. 5 we compare the optimal throughput-delay curves for static and dynamic flow control. This figure shows that the throughput-delay profile of a network using dynamically adjusted token limits is somewhat better than the performance under a static control. Therefore, dynamic flow control can further enhance the traffic handling capability of computer networks.

VII. CONCLUSION

In this paper we developed and analyzed a dynamic decision policy to select the number of tokens to control the flow in store-and-forward computer networks. The dynamic flow control developed here is essentially an extension and (slight) improvement of the static flow control studied in [1].

Using the theory of Markov decision processes, the decision problem was formulated, and a heuristic solution to an optimal policy was presented as an alternate to the laborious exact solution. The formulation and the heuristic solution is general in the sense that it can be applied to any decision process in which there is a leadtime for a decision to become effective, a common situation in inventory systems. We introduced a parameter α which reflects our

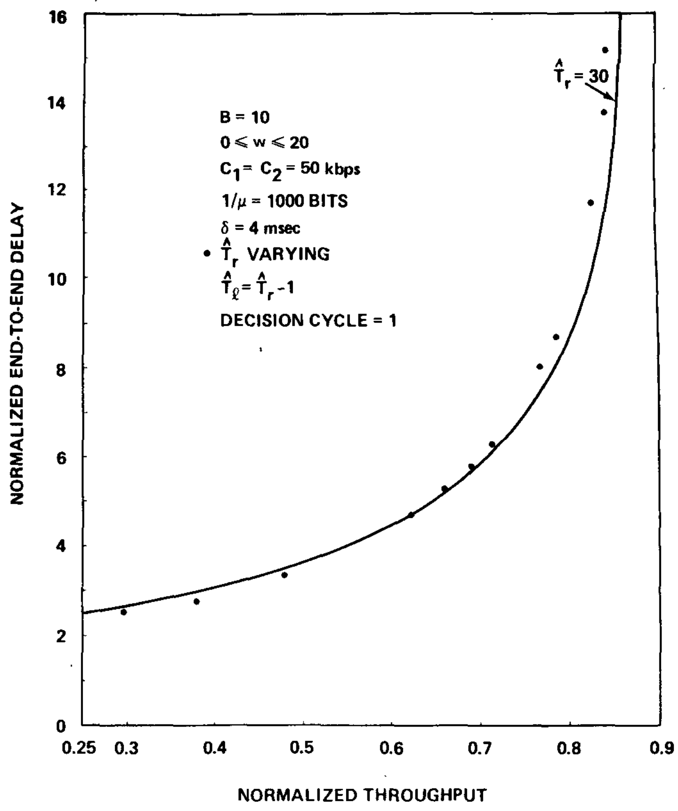

Fig. 4. Throughput-delay performance of look-ahead policy for varying round-trip delays.

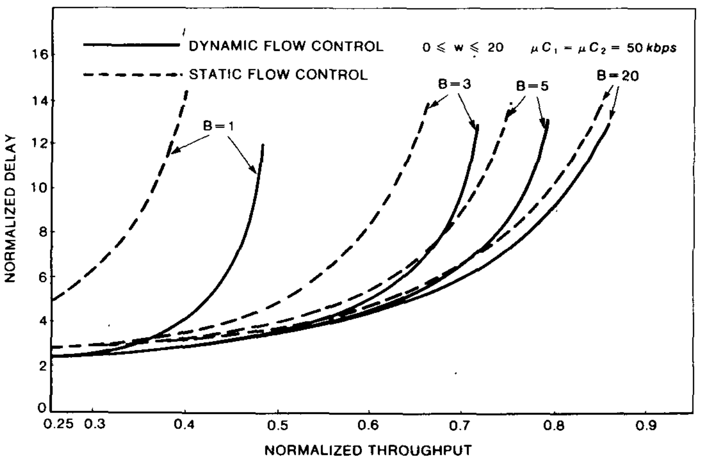

Fig. 5. Comparison of throughput-delay performance of optimal static and dynamic flow control.

preference for throughput over delay. This parameter can be considered as a further control tool to limit the throughput, the delay, the average and/or the maximum number of tokens, and to limit the destination buffer utilization (Table II).

The numerical examples indicate that the performance of a network when token limits are selected according to the suboptimal policy determined by our heuristic solution is very close to the performance of an optimal policy (Fig. 3). This figure further indicates that the performance of a periodic decision, is very close to the optimal as long as the decision period is chosen properly. This property is of significant importance in practical application, as a periodic decision reduces the overhead due to the signaling of control packets.

REFERENCES

[1] L. Kleinrock and P. Kermani, "Static flow control in store-and-forward computer networks," *IEEE Trans. Commun.*, this issue, pp. 271–279.
[2] P. Kermani, "Switching and flow control techniques in computer communication networks," Comput. Sci. Dep., Univ. Calif., Los Angeles, UCLA-ENG-7802, Feb. 1978 (also published as Ph.D. dissertation, Dec. 1977).
[3] A. Giessler, J. Haenle, A. Koenig, and E. Pade, "Packet networks with deadlock-free buffer allocation, an investigation by simulation," *Comput. Networks*, vol. 2, pp. 191–208, July 1978.
[4] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *Proc. Int. Conf. on Commun.*, Boston, MA, June 1979, pp. 43.1.1–43.1.10.
[5] R. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT, 1960.
[6] B. L. Miller, "Dispatching from depot repair in a recoverable item inventory system: On the optimality of a heuristic rule," *Management Sci.*, vol. 21, pp. 316–325, Nov. 1974.
[7] A. F. Veinott, Jr., "The status of mathematical inventory theory," *Management Sci.*, vol. 2, pp. 745–777, July 1966.
[8] D. Blackwell, "Discrete dynamic programming," *Ann. Math. Statis.*, vol. 33, pp. 719–726, 1962.
[9] M. J. Sobel, "Optimal operation of queues," in *Mathematical Methods in Queueing*. Berlin: Springer-Verlag, 1974, pp. 231–261.

Static Flow Control in Store-and-Forward Computer Networks

LEONARD KLEINROCK, FELLOW, IEEE, AND PARVIZ KERMANI

***Abstract*—In this paper we develop an analytic model for end-to-end communication protocols and study the window mechanism for flow control in store-and-forward (in particular message-switching) computer-based communication networks. We develop a static flow control model in which the parameters of the system are not dynamically adjusted to the stochastic fluctuation of the system load. Numerical results are presented and it is shown that the throughput-delay performance of a network can be improved by proper selection of the design parameters, such as the window size, the timeout period, etc.**

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication without oral presentation. Manuscript received June 13, 1978; revised June 27, 1979. This work was supported by the Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-77-C0272.

P. Kermani was with the Department of Computer Science, University of California, Los Angeles, CA 90024. He is now with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

L. Kleinrock is with the Department of Computer Science, University of California, Los Angeles, CA 90024.

INTRODUCTION

A computer network may be thought of as a collection of *resources* to be used by a competing population of *users.* Network resources include buffers, transmission bandwidth, processor time, name space, table entries, logical channels, etc.; the user population includes any source of data which requires transmission through the network. The collection of resources has a limited capacity which causes conflicts to occur among the users of the system. These conflicts may result in a degradation of system performance to the point that the system becomes clogged and the throughput goes to zero [1]. This behavior is typical of "contention" systems in which the throughput will increase with the applied load up to some optimum value, beyond which, due to unpredictable behavior by users and servers and additional user–user and user–server interaction and overhead, more load causes a reduction in throughput [2]. Networks cannot afford to accept all the traffic that is offered without control; there must be rules which govern the acceptance of traffic from outside and coordinate the flow inside the network. These rules are commonly known as *flow control* procedures. More precisely flow control is the set of mechanisms whereby a flow of data can be maintained within limits compatible with the amount of available resources [3].

In order to keep the network traffic within desirable limits, flow control procedures, among other things, must be equipped quipped with throttling mechanisms. These mechanisms include: *credit* (or *tokens*), which give permission for message flow; a *rate* at which a given flow may proceed; a *stop-and-go* procedure which turns a flow on and off according to some criteria; the introduction of *delay,* so as to slow down the flow, etc. [3]. A window mechanism is an example of the credit scheme.

Existing control methods in store-and-forward communication nets can be classified as either local control or global control. Local control is applied by a communication processor within the subnet on the basis of its own as well as its immediate neighbors' traffic data and resource utilization. Due to some limitations [4], [5], local control is not, by itself, sufficient to prevent congestion, and global control is necessary in order to further stop the input to the network well before the network is loaded to saturation. This control can be accomplished by limiting the number of packets simultaneously contained in the network. Examples of the existing methods of global flow control are: isarithmic flow control [4] studied for the NPL network; and end-to-end flow control ([6] and the references therein]) used in the ARPANET, where, basically, the total number of credits between two users are limited.

Most end-to-end flow control mechanisms use a variant of the credit throttling technique and are usually described in terms of a window mechanism [7], where the unacknowledged messages (or packets) are limited to lie within a sliding window.

End-to-end flow control is accomplished through interprocess communication protocols and any attempt to quanti-