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-

0090-6778/80/0200-0271$00.75

tatively study the former should start with the development of an analytic model for the latter. Due to the complex multivariate environment of these distributed dynamic control procedures, to date little work has been done in evaluating the performance of flow control procedures in terms of their efficiency and freedom from deadlock and degradation. In view of this, the recent work in the field has been extremely welcome [8]-[15].

The purpose of this paper is to develop an analytic model for end-to-end communication protocols and to study the token mechanism for flow control in store-and-forward computer based communication networks (we will be mainly concerned with message- and packet-switching systems). We first develop a static flow control model in which the parameters of the system are not dynamically adjusted to the stochastic fluctuations of the system load. We then present numerical results to show the effect of different parameters on the overall performance of the system. We use the static flow control results in a subsequent paper to develop a dynamic flow control system.

The analysis carried out in this paper is based on many simplifying assumptions. We consider the calculations as an initial step towards a better understanding of more complex and more elaborate systems. It is believed that such a study is necessary even if it is based on idealized assumptions.

II. THE MODEL AND THE ASSUMPTIONS

In this section we develop an analytic model for a network in which messages are flow controlled with a token mechanism. We begin by describing the structure of the network and its components. We then elaborate on the assumptions involved in our analysis, and finally we present the analysis.

A. Structure of The Network

With a credit mechanism using tokens, the total number of messages in the network between a source-destination pair is restricted to a maximum value. We model the network 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 (we occasionally refer to w as the *token limit*).

The structure of the TC is further elaborated in Fig. 2(a). We may think of a circulating pool of w "tokens," each of which permits exactly one message to be accepted to the network and to be sent toward the destination node; a copy of each message is retained in the buffer space (labeled timeout box in Fig. 2(a)) provided by the TC. If an ACK for this message is received within τ seconds (the timeout period). The TC discards its stored copy of the message and the token is returned back to the pool. This token may then be used to accept a new message from the source. Thus, we allow up to w outstanding messages between a source-destination pair. If, however, the ACK is not received within τ seconds, the message is retransmitted and no token is released. In our anaylsis we required that the ACK for the last (re)transmitted message be received so that the TC can accept a new message to network. This means, once an ACK is not received within the timeout period, the TC assumes the transmission has failed and any ACK received after the timeout period is ignored. The reason for this simplifying assumption is that modeling of the exact operation is extremely difficult and the analysis is unnecessarily complicated (for a discussion regarding the implications encountered in the exact analysis the interested reader is referred to [13]).

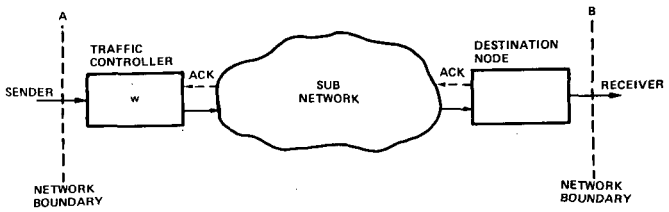

Fig. 1. Structure of a network.

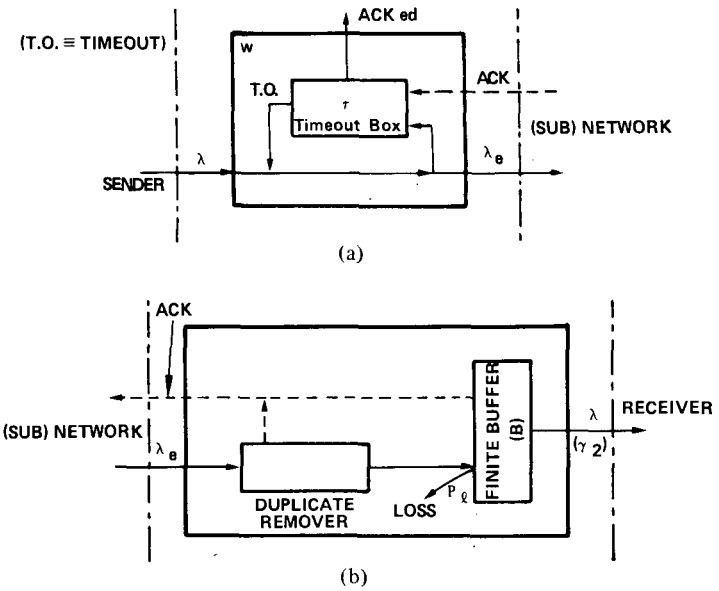

Fig. 2. Structure of network components.

A detailed specification of the structure of the subnetwork is not the object of our study; we view the subnetwork as a source of random delay which we choose to represent by a proper distribution function (which will be discussed in Section II-B below).

Fig. 2(b) shows the structure of the destination node. Because of our requirement that the TC must receive an ACK for the last (re)transmitted message, the destination node may receive multiple copies of the same message. Clearly, only the first received copy should be delivered to the receiving user and the later copies must be discarded. The "duplicate remover" in the destination node (Fig. 2(b)) checks all the received messages and discards all the previously accepted ones, but sends an ACK for these duplicate messages back to the sending TC; only the first copy of a message is passed to the buffer storage. If there is space available at the buffer, the message is admitted and is put in the queue to be delivered to the receiving user and an ACK is sent back to the TC. If, however, there is no buffer to store the message, it is rejected, no ACK is sent back, and later copies will be accepted.

B. Assumptions

Measurements on existing networks, in particular on the ARPANET [16], show that the average number of packets per message is very close to one (1.2 packets per message in the ARPANET); hence, we make the simplifying assumption that

messages consist of single packets and we do not differentiate between packet- and message-switching.

We study the system shown in Fig. 1 under a "heavy traffic" condition, i.e., we assume that the sender is fast compared to the network so that whenever the TC can accept a new message, the sender has one ready for transmission. With this assumption, the throughput is mainly determined by the network and throughout our study we will be concerned with the maximum-throughput versus delay behavior of the network within the boundaries shown in Fig. 1. Maximum throughput is the maximum rate at which the network can accept new messages (hence it is the upper bound of the accepted input rate to the network). For simplicity, when there is no ambiguity, we use "throughput" to indicate "maximum throughput." Corresponding to this maximum throughput there is an end-to-end (or a total) delay which is the delay that a message experiences from the time it is submitted to the TC until it is delivered to the receiver. The total delay consists of the network delay (i.e., the delay incurred in the TC and the (sub)network) plus the destination node delay. (Note that the end-to-end delay does not include the admittance delay to the system.) Another delay measure important in our study is the acknowledgment delay which is the period from the time a message is delivered to the (sub)network until its ACK is received by the TC.

In our analysis we will accept the following assumptions:

1) Exponential message length distribution (with mean $1/\mu$).

2) Poisson arrival of messages to the (sub)network.

3) The independence assumption [17] regarding the operation of the (sub)network.

4) Round-trip delays (hence ACK delays) are independently and identically distributed. Each round-trip delay has an Erlangian distribution of degree two consisting of the concatenation of two system delay distributions of an $M/M/1$ queue, that is,

$$F_{\tilde{t}_r}(t) = 1 - e^{-(\gamma_1-\lambda_e)} - (\gamma_1 - \lambda_e)te^{-(\gamma_1-\lambda_e)t}$$

where $\tilde{t}_r$ is a random variable representing the round-trip-delay, λ_e is the (sub)network traffic rate and $\gamma_1 = \mu C_1$, C_1 being the network channel capacity.

5) The destination node has a finite number of buffers; each buffer can store one (and only one) message.

6) Negligible transmission error rate.

7) At the destination node only the first successful copy of a message is accepted; however, ACK's for later copies are sent back. After the removal of the duplicate messages, the counting process of message arrivals to the destination buffer is also assumed to be Poisson.

8) No message sequencing.

Some Remarks: 1) The first three assumptions are made to substantiate assumption number 4. With these three assumptions the (sub)network can essentialy be modeled as a Markovian network of queues. Simulation and measurement studies of existing networks [18], [16] and [19] show that a multistage Erlangian distribution is a good approximation for the round-trip delay distribution. Ideally, an Erlangian distribution of a degree larger than 10 is the best choice; however, any degree larger than 2 makes the derivations and the integrations extremely complicated. Considering the fact that the protocol performance is relatively insensitive to the degree of the distribution ([11], also our analysis with degrees = 1 and 2), we choose to use an Erlangian distribution of the second degree for the round-trip delay. We further assume that the one way delay is similar to the system delay of a simple $M/M/1$ queueing system. Because messages which contain errors, or which cannot be accepted at the destination buffer are not acknowledged, the acknowledgment delay may be infinite. We present this fact in our model by including an impulse (of size P_l, the probability of loss) in the density function of the ACK delay at time equal to infinity. Therefore, the distribution function fo the ACK delay approaches $(1 - P_l)$ (instead of approaching to 1) as time goes to infinity. This reflects the fact that there is a chance that the ACK delay becomes infinite.[1]

2) With assumption 5, one might think that the assumption of exponentially distributed message length (Assumption 1) should be replaced with a truncated exponential distribution. However, measurements on existing networks, in particular on the ARPANET [16], shows that the average message length is relatively small (250 bits in the ARPANET). A reasonably large buffer size, and considering the fact that the average message length is much smaller than the buffer size, and that large messages occur very infrequently, justifies Assumption 5 made above.

3) In Section II-A we pointed out that a message is not acknowledged only if the buffer storage is full. This means we are implicityly assuming that all messages are received error free. We take the liberty of ignoring the channel error rate (Assumption 6) as the quantity is often very small (on the order of 10^{-5}).

4) Assumption 7 is made to simplify the calculation of the blocking probability at the destination node buffer.

5) The important problem of message sequencing is also ignored (assumption 8), as it is beyond the scope of this paper.

For further consideration regarding the implications raised by the above assumptions, the interested reader is addressed to [13].

C. The Analysis

We start our analysis by deriving the ACK delay distribution. Based on assumption 4 above, the distribution of the acknowledgment delay is

$$F_{\tilde{t}_a}(t) = (1 - P_l)[1 - e^{-\Lambda t} - \Lambda te^{-\Lambda t}] \tag{1}$$

and its density function will be

$$f_{\tilde{t}_a}(t) = (1 - P_l)\Lambda^2 te^{-\Lambda t} \tag{2}$$

where

$$\Lambda = \gamma_1 - \lambda_e. \tag{3}$$

In the above equations $\tilde{t}_a$ is a random variable representing the acknowledgment delay and P_l is the probability that an ACK is not received. We note that $\tilde{t}_a$ is a "degenerate" random variable.

[1] It should be emphasized that the first 4 assumptions (including assumption 7) lose their consistency when w is small and/or the (sub)network is lightly loaded. In fact it can be easily shown that for $w = 1$ and when $P_l = 0$ the arrival process to the (sub)network becomes Erlangian of the second degree. Only when a high traffic is offered from TC to (sub)network the above assumptions become more consistent. The fact that we study the system under heavy traffic justifies these assumptions to some degree.

To calculate the retransmission probability we note that a message must be retransmitted when either the message is lost at the destination buffer, or the transmission is successful but the acknowledgment delay is greater than the timeout (τ); therefore, we have

$$P_r = \Pr[\text{retransmission}]$$
$$= P_l + (1 - P_l) \Pr[\tilde{t}_a > \tau \mid \text{no loss}]$$

or

$$P_r = P_l + (1 - P_l)(1 + \Lambda)e^{-\Lambda\tau} \tag{4}$$

The effective network traffic, λ_e, is related to the input rate of messages to the network by the following:

$$\lambda_e = \frac{1}{1 - P_r}\lambda. \tag{5}$$

We now consider the length of the time a message occupies a buffer in the TC. Using a renewal theory argument [20] we have [13]:

$$T_{oc} = T_1 + \frac{P_r}{1 - P_r}\tau$$

where

$$T_{oc} = E[\text{time of occupancy}]$$

and

$$T_1 = E[\tilde{t}_a \mid \tilde{t}_a \leqslant \tau].$$

Calculation of T_1 is fairly easy. It can be shown that [13]

$$T_1 = \frac{2}{\Lambda} - \frac{\Lambda\tau^2 e^{-\Lambda\tau}}{1 - [1 + \Lambda\tau]e^{-\Lambda\tau}}.$$

Finally, we find the value of T_{oc} as follows:

$$T_{oc} = \frac{2}{\Lambda} + \frac{P_l + (1 - P_l)e^{-\Lambda\tau}}{(1 - P_l)\{1 - [1 + \Lambda\tau]e^{-\Lambda\tau}\}}. \tag{6}$$

When the flow is controlled by the token mechanism described in Section II-A, w messages are accepted to the network on the average every T_{oc} seconds; therefore, the (maximum) input rate will be

$$\lambda^* = \frac{w}{T_{oc}^*} \tag{7}$$

where T_{oc}^* is the occupancy time when the flow controlled throughput is maximum (with w tokens) and is given by (6) (when $\lambda = \lambda^*$). From (5) we have

$$\lambda_e^* = \frac{1}{1 - P_r}\frac{w}{T_{oc}^*}. \tag{8}$$

We can use the values of P_r and T_{oc}^* from (4) and (6) in (8) to get

$$\lambda_e^* = \frac{w\Lambda}{2(1 - P_l)[1 - (1 + \Lambda\tau)e^{-\Lambda\tau}] + [P_l + (1 - P_l)e^{-\Lambda\tau}]\Lambda\tau}$$

In general we have

$$\lambda_e^* = G_1(\tau, w, \gamma_1, P_l, \lambda_e). \tag{9}$$

The unknowns in the above equation are λ_e^* and P_l, the probability that an ACK is not returned. By virtue of assumption 7, the destination buffer behaves like an $M/M/1/B$ queueing system where B is the buffer size at the destination node. To calculate the loss probability P_l, we note that at equilibrium the input rate of messages to the network (λ^*) is equal to the output rate of messages from it. Hence λ_d^*, the total arrival rate of messages to the buffer (the accepted messages plus the rejected ones), will be

$$\lambda_d^* = \frac{\lambda^*}{1 - P_l}$$

and by (5) we have

$$\lambda_d^* = \frac{1 - P_r}{1 - P_l}\lambda_e^*$$

and so we have [21]

$$P_l = \frac{\gamma_2 - \lambda_d^*}{\gamma_2^{B+1} - \lambda_d^{*B+1}}\lambda_d^{*B} \tag{10}$$

where $\gamma_2 = \mu C_2$ and C_2 is the capacity of the output channel. The above equation can be written as

$$P_l = G_2(\gamma_2, B, P_l, \lambda_e). \tag{11}$$

The analytic results presented so far depend upon the solution of the two equations (9) and (11). For a given value of the parameters (τ, w, γ_1, γ_2, and B) the above system of equations can be solved through an iterative procedure [22].

Having found the network traffic λ_e^*, we can use (5) to find the maximum input rate (or the maximum throughput) of the network. The total end-to-end delay consists of two components: the network delay and the destination buffer delay. In calculating the network delay we should notice that only the delay of the first successful message is of importance to us; further retransmissions only affect the network traffic (hence have an indirect effect on the delay). Using a renewal theory argument [20], the (maximum) average network delay can be found to be [13]

$$T^* = \frac{1}{\gamma_1 - \lambda_e^*} + \frac{P_l}{1 - P_l}\tau. \tag{12}$$

The other component of the total delay is the destination buffer delay. This is equivalent to the system delay of the corresponding $M/M/1/B$ queue [21]

$$T_{dst}^* = \frac{(1/\mu C_2)}{1 - \rho_d^*}\frac{1 - (1 + B)(\rho_d^*)^B + B\rho_d^{*B+1}}{1 - (\rho_d^*)^B} \tag{13}$$

where $\rho_d^* = (\lambda_d^*/\mu C_2)$ is the utilization factor of the destination buffer. Finally we have

$$T_{ee}^* = T^* + T_{dst}^* \tag{14}$$

where T_{ee}^* is the maximum end-to-end delay.

This completes our analysis of the flow control mechanism. As noted, the analysis does not provide us with a closed-form solution for the network throughput and delay; we can only solve the model through numerical techniques. In [13], based on some simpler and less realistic assumptions, other models for flow control have been developed for which closed-form solutions have been derived. These models, though very simplistic, have been used to develop a network algebra for a systematic study of series and parallel interconnections of networks and also to study the problem of the optimal allocation of tokens and channel capacity among a number of parallel connected networks between a source and destination. The interested reader is referred to [13], [15].

The analysis of this section has been based on the equilibrium of the system for a selected set of parameters (τ, w, μ, C_1, C_2, B) which are not dynamically adjusted to the stochastic fluctuations in the load of the system. We refer to the mechanism analyzed above as "static flow control."

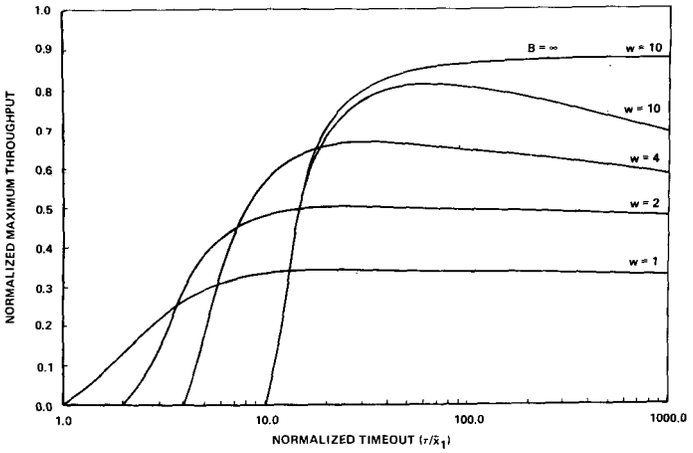

Fig. 3. Dependency of network throughput on timeout (B = 10, C_1/C_2 = 1.0).

III. NUMERICAL RESULTS

In this section we present numerical examples for static flow control, and study the effect of different parameters (w, τ, and B) on the throughput-delay performance of a network with μ, C_1, and C_2 given. For a given set of parameters we can find the maximum throughput (λ^*) and the corresponding (therefore maximum) total average delay (T_{ee}^*). Whenever there is no ambiguity, we will refer to these quantities simply by throughput and delay. Throughout our presentation we consider the normalized values of certain quantities; for measures of time, the normalization constant is $\bar{X}_1 = (1/\mu C_1)$, the average transmission time of a message on a network channel. It also turns out that only the ratio C_1/C_2 is of importance to us, and it is that which we use in this section.

We begin by studying the effect of the timeout period on the network throughput. Fig. 3 shows the normalized throughput ($\lambda^*/\mu C_1$) as a function of the normalized timeout ($\tau/\bar{X}_1$) for different w's when the destination buffer size $B = 10$ and $C_1/C_2 = 1$. It can be seen that when the timeout is very small the throughput decreases to zero. The reason for this degradation is the following: when the timeout is small, the TC retransmits messages very often; hence the (sub)network channel utilization ($\lambda_e/\mu C_1$) approaches 1 and the network delay becomes unbounded; because the number of unacknowledged messages is limited, no new message is admitted to the network and the throughput degrades to zero ((7)). It is interesting to note that in order for the system to remain stable, the normalized timeout should be larger than w. This fact is intuitively clear in the deterministic case: the transmission of w messages takes wX_1 seconds (X_1 being the transmission time of one message), and so a timeout of $\tau < wX_1$ causes retransmission of a message to start before transmission of the w messages can possibly be completed. As a result, the network becomes saturated. This fact can be established analytically; the interested reader is referred to [13]. We should point out that this behavior is a consequence of the requirement that the ACK for the last (re)transmitted message should be received. In real life, only when the timeout approaches 0 does the network becomes saturated.

Continuing our discussion, Fig. 3 also shows that when the timeout increases, the network throughput first grows; however, after a certain value of τ, the throughput starts to decrease. In fact, when the destination buffer size (B) is finite $\lim_{\tau \to \infty} \lambda^* = 0$. This fact is easily shown by using (7) and (8). Fig. 3 shows that this throughput degradation is more severe for larger w. The reason for this degradation is that for a large timeout the TC waits a long time for an ACK to come back; however, because of the finite destination buffer size, there is a nonzero probability that an ACK never comes back, hence the buffer of the traffic controller becomes full of unacknowledged messages, and no further external traffic is accepted. Clearly, this degradation is not present if the destination buffer size is infinite; the throughput curve in Fig. 3 for $w = 10$ and the destination buffer size of infinity explicitly shows this fact. Fig. 3 shows that for a fixed w, there is a value of timeout that optimizes (maximizes) the throughput. We will designate this optimal value by τ_t^* (as opposed to τ_d^*, which is the optimal value of timeout with respect to the end-to-end delay; see below). This figure shows that for a small w, the throughput curve, after an initial increase, remains quite flat, and only for a large w is the maximal throughput sensitive to timeout.

The effect of the timeout period on delay is shown in Fig. 4, where the normalized (maximum) end-to-end delay ($T_{ee}^*/\bar{X}_1$) is shown as a function of the normalized timeout for different w's. As in Fig. 3, this figure also shows that for a fixed w, there is a value of timeout which optimizes (minimizes) the end-to-end delay; this optimal timeout (τ_d^*, d for delay) is usually different from τ_t^*. When the timeout is too small, like the throughput, the end-to-end delay becomes unbounded (the minimum allowable timeout is, of course, $w\bar{X}_1$); as the value of the timeout increases, the end-to-end delay first reaches a minimum and then becomes unbounded.

We now study the effect of the number of tokens on the network performance. In Fig. 5 the normalized throughput is shown as a function of w. On the curve designated "optimal throughput," for each w the optimal value of timeout (τ_t^*) is chosen such that the throughput becomes maximal; on the curve designated "optimal delay," the optimal value of timeout (τ_d^*) is chosen so that the end-to-end delay becomes minimal. This figure also shows the throughput curves for several constant (normalized) timeouts. For a fixed timeout the throughput curve is concave and the throughput reduces to zero for $w \geq \tau/\bar{X}_1$; this agrees with our previous result. The throughput curves for constant timeout display the catastrophic behavior of a contention system (described in

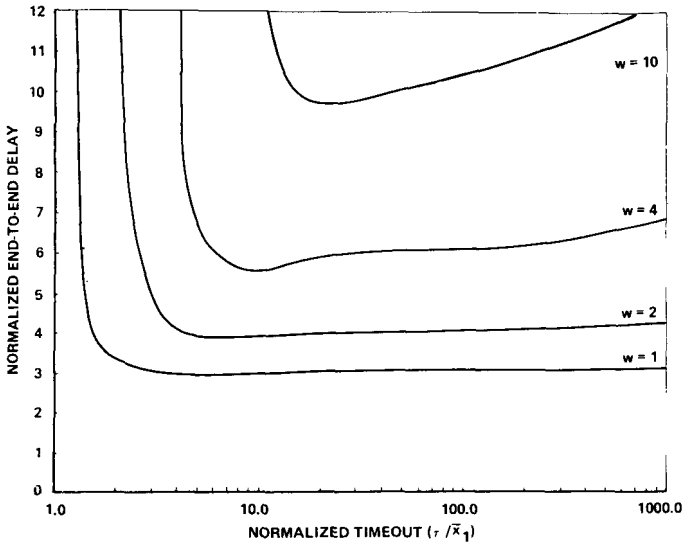

Fig. 4. End-to-end delay as a function of timeout ($B = 10$, $C_1/C_2 = 1.0$).

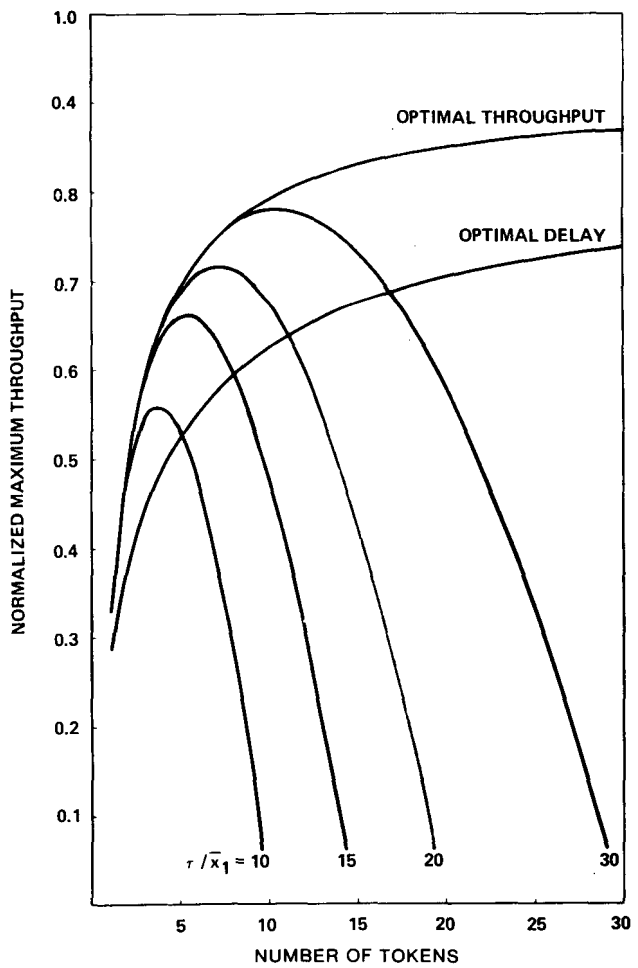

Fig. 5. Network throughput versus number of tokens ($B = 10$, $C_1/C_2 = 1.0$).

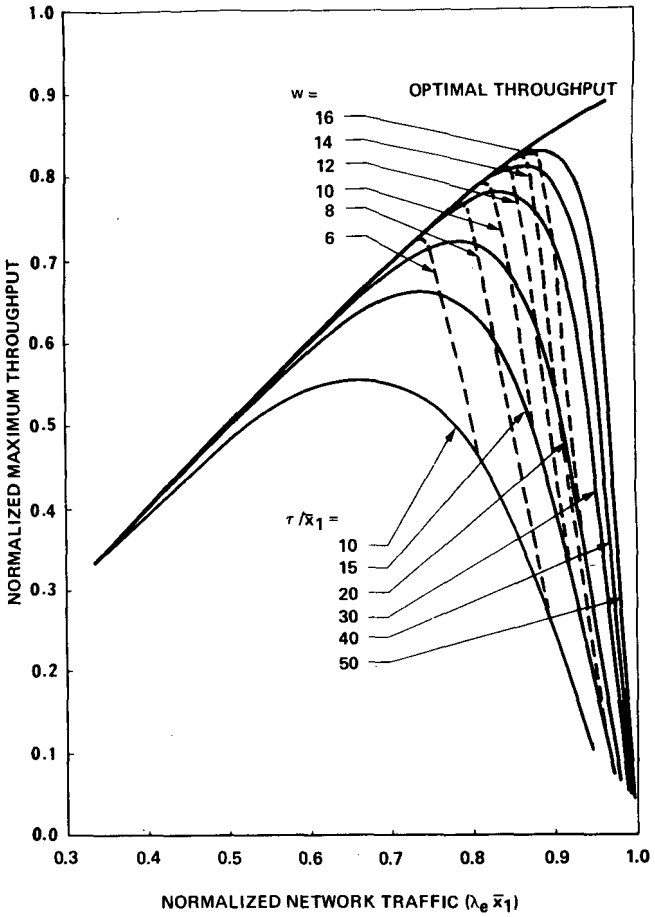

Fig. 6. Throughput versus network traffic ($B = 10$, $C_1/C_2 = 1.0$).

the opening remarks of this paper); here w represents the load of the system. Note that the optimal throughput curve is the maximum envelope of these contours, and defines the *tight upper bound* on the throughput-token performance.

To show the effect of the contention more explicitly, we have plotted the network traffic versus the network throughput for (constant) normalized timeouts in Fig. 6. For a fixed τ, an increase in w results in an increase in the network traffic. Initially the throughput increases as the network traffic grows; however, after a certain value of λ_e^*, the throughput starts declining, and when the network saturates (i.e., $(\lambda_e^*/\mu C_1) = 1$), the throughput decreases to zero. In the same figure we have also shown the contours of constant w. Note that for a fixed w, when the timeout increases, throughput first grows and then levels off; however, the network traffic decreases continuously. Note also that all the w-contours peel off from a common upper bound curve, which is the maximal throughput curve (see Fig. 5). If any one of parameters w and/or τ is held fixed and the other one is increased, the throughput decreases to zero; only when both w and τ are increased simultaneously (such that for each w the optimal τ_t^* is chosen) will the throughput not degrade (see the optimal throughput curve in Fig. 6). This figure shows that the (normalized) throughput never reaches 1; even when w and τ are chosen optimally; the throughput is limited by the destination node buffer size.

The price for having more throughput is, of course, the delay which is shown in Fig. 7. This figure shows the throughput-delay equilibrium curves for fixed values of (normalized)

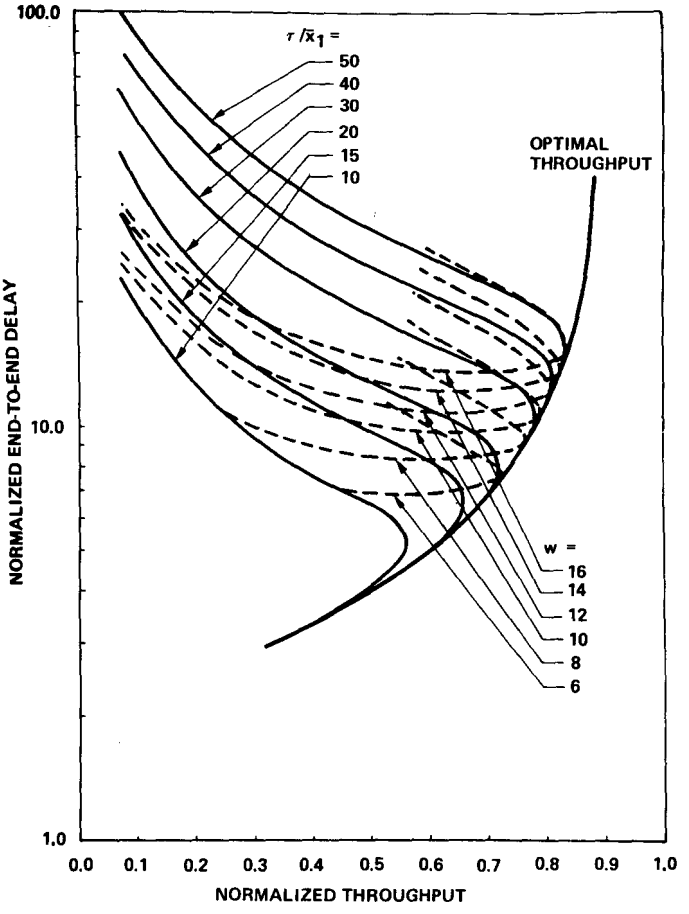

Fig. 7. Throughput-delay tradeoff (B = 10, C_1/C_2 = 1.0).

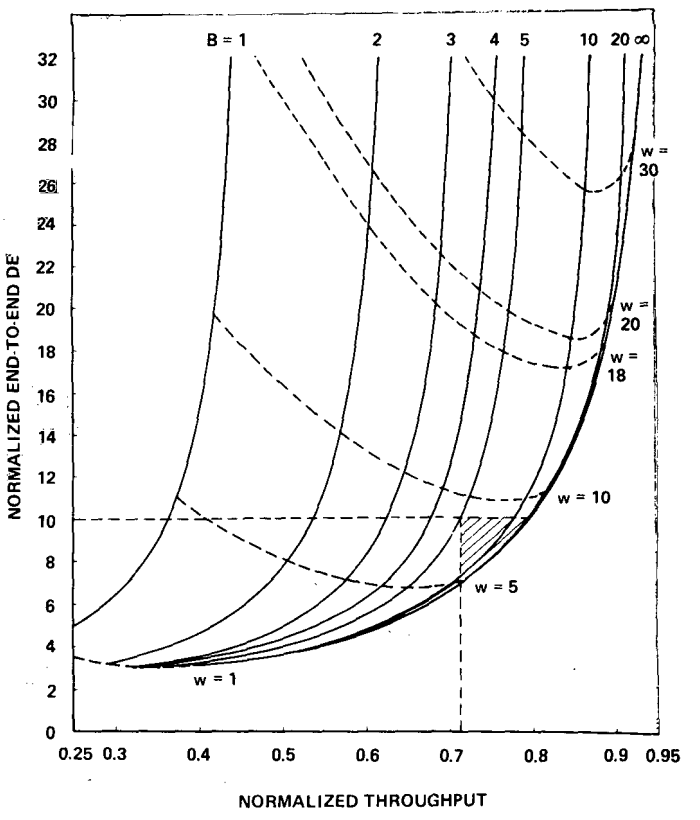

Fig. 8. Throughput-delay for different destination buffer sizes (C_1/C_2 = 1.0).

timeouts and number of tokens. The minimum envelope of these contours defines a tight lower bound on the throughput-delay performance and thus, can be viewed as the *optimal network performance* for our token mechanism model. As we pointed out earlier, the (normalized) throughput never reaches 1; the maximum throughput (which is achieved when $w \to \infty$ and $\tau \to \infty$ such that τ is optimal) is determined by the destination buffer size and the destination channel capacity. For example, in Fig. 7 this maximum is achieved when the network traffic approaches its maximum ($\lambda_e{}^* = \mu C_1$), and for this network traffic the (normalized) throughput is approximately 0.909 (and the end-to-end delay is, of course, unbounded).

In Fig. 8 we have shown the optimal throughput-delay curves for different destination buffer sizes and a fixed destination channel capacity ($C_1/C_2 = 1$). Along each curve the timeout and w vary, and we have also shown the contours of constant w. This figure shows that for a fixed end-to-end delay the network throughput increases rapidly as the destination buffer size increases. (Note that for $B = 20$ the maximum throughput is almost achieved.) When the channel capacities (network channels as well as the destination node channel) are known, this figure can be used for guidance in determining a proper destination buffer size. For example, if we set a minimum level of throughput, say $(\lambda^*/\mu C_1) \geq 0.72$, and a maximum end-to-end delay, say $T_{ee}{}^*/\bar{X}_1 < 10$, then this figure shows that the destination buffer size should be at least 5 (the point specified by the intersection of the light dashed lines in Fig. 8). Note that a destination buffer size of $B > 5$ would also satisfy our requirements (the shaded area); however, we have chosen the minimum buffer size in order to minimize the overall network cost.

When the destination buffer size and the capacities of channels are known, then it is important to know what (w, τ) combination to use in order to provide a certain grade of service. In Fig. 9 we have plotted contours of constant (normalized) throughput (the heavy solid curves), and (normalized) end-to-end delay (the heavy dashed curves); we will refer to these curves as λ^*-contours and T^*-contours, respectively. This figure displays most of the information contained in the previous figures; in particular we observe the following characteristics:

1) The locus of minimum allowable (normalized) timeouts is the $\lambda^*(=0)$-contour (which coincides with the $T^*(=\infty)$-contour, designated by "$w = \tau/\bar{X}_1$" in Fig. 9. Along this curve the throughput is zero and the delay is unbounded (infinite). For small values of τ, the system is unstable.

2) A line of constant w (a vertical line) may intersect a specified λ^*-contour (say $\lambda^* = \lambda_1{}^*$) at (a) two points; (b) one point (tangent to the $\lambda^*(= \lambda_1{}^*)$-contour); and (c) no point. In case (a) the two values of timeout give the same throughput (but the delays at these two points are different from each other). In case (b) the value of timeout at the tangent point is $\tau_t{}^*$, the optimal timeout for maximal throughput (compare with Fig. 3). We have indicated the loci of these points by curve A in this figure (which we will refer to as the $\tau_t{}^*$-curve). If the line of constant w does not intersect the $\lambda^*(= \lambda_1{}^*)$-contour [case (c)], that indicates that a throughput of $\lambda_1{}^*$ cannot be realized by this w.

3) The above discussion can be carried out for the T^*-contours; the loci of points where the constant w lines are tangent to T^*-contours (the $\tau_d{}^*$-curve) is designated by curve B.

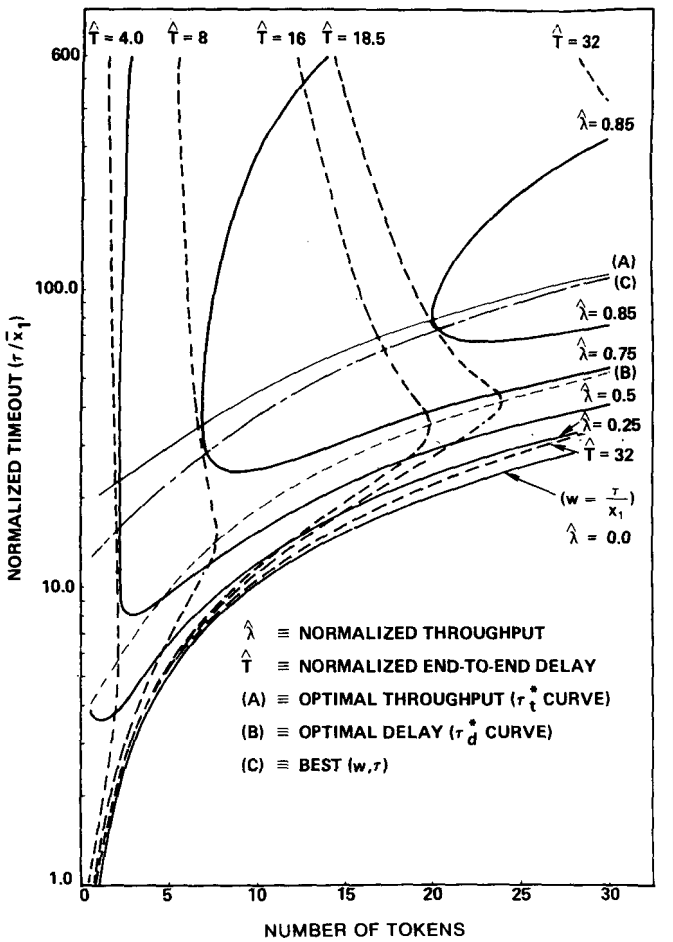

Fig. 9. Contours of throughput and end-to-end delay (B = 10, C_1/C_2 = 1.0).

4) We can use Fig. 9 to find the maximum achievable throughput (and the corresponding w) for a fixed timeout. The λ^*-contour which is tangent to the constant τ line (a horizontal line with an intercept equal to the specified timeout) determines the maximum throughput, and the value of w at the tangent point is the w which provides this throughput. It is interesting to note that this value of the throughput is the point where the τ-constant line intersects the envelope of the throughput curves in Fig. 3 (we have not drawn this curve).

5) Fig. 9 can be used to find the best (w, τ) combination which guarantees a certain grade of service. Assume that for a given network (for which the parameters μ, C_1, C_2, and B are known) we require at least a minimum throughput (say $\lambda^*/C_1 > 0.85$). In order to find the (w, τ) which results in the minimum end-to-end delay, we trace the $\lambda^*(= 0.85)$-contour and find the T^*-contour that is tangent to it (in Fig. 9 the $T^*(= 18.5)$-contour is such a contour, which is tangent to the $\lambda^*(= 0.85)$-contour at $w = 20$ and $\tau/\bar{X}_1 = 73$). Any (w, τ) other than 20,73) results in a higher delay; hence this pair is the best choice. The loci of these tangent points (or the best (w, τ) combinations) is shown in Fig. 9 as curve C. Note that we can use a smaller w and still have the same throughput (e.g., intersection of $\lambda^*(= 0.85)$-contour and curve A); however, the delay at this point is larger than 18.5. Considering the fact that w represents the buffer size of the traffic controller, this latter operating point (with a smaller w) may be a better choice, as it results in a lower network (buffer) cost.

Fig. 9 can be used in an analogous way in situations where the end-to-end delay is required to be no more than a given maximum value. Similar to the previous case, the best operating (w, τ) value (which results in a maximum throughput) is found by locating the intersection of the T^*-contour and curve C.

IV. CONCLUSION

Based on a token mechanism, we developed a model for static flow control in store-and-forward computer communication networks.

It was shown that for a w, if the timeout period is too large and/or too small, the throughput degrades to zero. In fact, for a given w, there is a value for timeout which optimizes the throughput. The model explicitly shows the effect of contention in that, when the timeout period is fixed, as w (or the network traffic) increases, the throughput first increases and then degrades to zero. By studying the throughput-delay performance of networks (when traffic is controlled by a token mechanism), we found the *optimal network performance curve,* namely, the minimum envelope of the throughput-delay equilibrium curves for fixed values of timeouts and number of tokens. A number of design problems were proposed and, through a very useful graphical presentation, some guidelines regarding the optimal selection of system parameters were given.

The development and the analysis of the model presented in this paper was based on a number of simplifying assumptions regarding the acknowledgment and timeout strategy, the subnetwork delay distribution, and the destination node structure. We also assumed that the system is operating under heavy traffic condition. These simplifying assumptions, though to some extent idealistic, were necessary to lend the model amenable to analysis. In view of these, the results presented in this paper should be regarded as an approximation to the behavior of the system in real life. Analysis of flow control mechanisms is still in its infancy. Much work need to be done in this area to quantitatively understand implications of these mechanisms; hopefully, this paper has added to this understanding.

In a forthcoming paper, by using the results of this static flow control analysis, we develop and study a dynamic flow control mechanism in which the number of tokens and timout are dynamically adjusted to the stochastic fluctuations in the load on the system.

REFERENCES

[1] R. Kahn and W. Crowther, "Flow control in resource sharing computer networks," *IEEE Trans. Commun.*, vol. COM-20, pp. 546–550, June 1972.

[2] C. E. Agnew, "Dynamic modeling and control of congestion-prone systems," Dep. Eng.–Econ. Syst., Stanford Univ., Stanford, CA, Tech. Rep. 10, Jan. 1974.

[3] L. Pouzin, "Flow control in data networks-methods and tools," in *Conf. Proc. Int. Conf. Comput. Comm.*, Toronto, Canada, Aug. 1976, pp. 467–474.

[4] D. W. Davies, "The control of congestion in packet switching networks," in *Proc. 2nd ACM-IEEE Symp. Problems of the Optimization of Data Communication*, Palo Alto, CA., Oct. 1971, pp. 46–49.

[5] L. Kleinrock, *Queueing Systems, Vol. II: Computer Applications*. New York: Wiley-Interscience, 1976.

[6] J. M. McQuillan, W. R. Crowther, B. P. Cossell, D. C. Walden, and F. E. Heart, "Improvements in the design and performance of the ARPA network," in *1972 Fall Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 41, Anaheim, CA, Dec. 1972, pp. 741–754.

[7] V. G. Cerf and R. E. Kahn. "A protocol for packet network interconnection," *IEEE Trans. Commun.*, vol. COM-22, pp. 637–648, May 1974.
[8] H. Rudin, "Chairman's remarks: An introduction to flow control," in *Proc. Int. Conf. Commun.*, Toronto, Canada, Aug. 1976, pp. 463–466.
[9] S. S. Lam and M. Reiser. "Congestion control of store-and-forward networks by input buffer limits," in *Proc. Nat. Telecommun., Conf.*, Los Angeles, CA, Dec. 1977, pp. 12:1-1–12:1-6.
[10] M. C. Pennotti and M. Schwartz. "Congestion control in store and forward tandem links," *IEEE Trans. Commun.*, vol. COM-23, pp. 1434–1443, Dec. 1975.
[11] C. A. Sunshine, "Interprocess communication protocols for computer networks," Stanford Electronics Lab., Stanford Univ., Stanford, CA, Tech. Rep. 105, Dec. 1975.
[12] 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.
[13] P. Kermani, "Switching and flow control techniques in computer communication networks," Ph.D. dissertation, Comput. Sci. Dep., Univ. Calif., Los Angeles, CA, Dec. 1977.
[14] L. Kleinrock, "On flow control in computer networks," in *Proc. Int. Conf. Commun.*, Toronto, Ont., Canada, June 1978, pp. 27.2.1–27.2.5.
[15] L. Kleinrock and P. Kermani. "A network algebra for the performance evaluation of interconnected computer networks," in *Proc. Nat. Telecommun. Conf.*, Birmingham, AL, Dec 1978.
[16] L. Kleinrock and W. E. Naylor, "On measured behavior of the ARPA network," in *1974 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 43, Chicago, IL, May 1974, pp. 767–780.
[17] L. Kleinrock, *Communication Nets: Stochastic Message Flow And Delay*. New York: McGraw-Hill, 1964.
[18] G. D. Cole, "Computer network measurements: Techniques and experiments," Comput. Sci. Dep., Univ. Calif., Los Angeles, CA, UCLA-ENG-7165, Oct. 1971.
[19] J. W. Forgie, "Speech transmission in packet-switched store-and-forward networks," in *1975 Spring Joint Comput. Conf. AFIPS Conf. Proc.*, Anaheim, CA, May 1975, pp. 137–142.
[20] D. R. Cox, *Renewal Theory*. London: Butler, Tonner, 1967.
[21] L. Kleinrock, *Queueing Systems, Vol. I: Theory*. New York: Wiley-Interscience, 1975.
[22] J. M. Ortega and W. C. Rheinbold. *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic, 1970.

Packet Radio Performance Over Slow Rayleigh Fading Channels

JAMES A. ROBERTS, SENIOR MEMBER, IEEE, AND
TIMOTHY J. HEALY, MEMBER, IEEE

Abstract—**Expressions for the throughput and average packet delay for a pure-ALOHA single-hop packet radio system operating in slow Rayleigh fading are derived. For noncoherent frequency-shift-keying (NCFSK), an exact closed form expression is presented. For coherent phase-shift-keying (CPSK) an excellent approximation for large packet sizes is derived. This approximation technique is valid in general for other modulation schemes and for other fading channel statistical characterizations. The packet length which maximizes the useful data throughput in slow Rayleigh fading is found. The results of this investigation indicate that a packet radio system can be designed with a modest link margin for fading and achieve identical throughput performance over a nonfading channel and a fading channel with only a small increase in average packet delay for the fading channel.**

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication without oral presentation. Manuscript received January 19, 1979; revised August 16, 1979.

J. A. Roberts is with ESL Incorporated, Sunnyvale, CA 94086.
T. J. Healy is with the Department of Electrical Engineering and Computer Science, University of Santa Clara, Santa Clara, CA 95053.

I. INTRODUCTION

Packet switching, a new way to establish communications between computer and computer and between user and computer, began with the development of the ARPANET [1]–[4] in 1969. The ALOHA system at the University of Hawaii [5], [6] was the first network to employ radio links for packet communication. In the ALOHA system, user terminals are connected to a central computer via a single shared UHF radio channel. The first analyses of packet radio performance assumed that packet collisions were the major cause for loss of a packet and subsequent retransmission. More recently, efforts to design packet radio systems to operate over degraded channels have been undertaken [7]. This paper examines the performance of the pure-ALOHA single-hop packet radio technique over a slow Rayleigh fading channel. There are several applications including ionospheric scintillations experienced in satellite communications [8], [9], radio communications in an urban environment [10], and HF radio [11].

Previous efforts have addressed blocked data transmission over random channels. Cavers [12] considers buffer control using variable-rate blocked-data transmission over slow Rayleigh fading channels. The variable-rate techniques, as well as other variable communications parameter methods for mitigating the effects of fading have received considerable attention [13]–[18], but, with the exception of the paper by Cavers, none consider blocked data. Eaves and Levesque [19] analyze the probability of block error in slow Rayleigh fading. They develop a series solution for noncoherent frequency-shift-keying which is also derived here. Their final results are obtained by numerical integration.

The work described here differs from previous efforts in a number of significant ways. First, we emphasize the performance of a multiaccess technique, specifically pure-ALOHA, in a slow Rayleigh fading environment. The goal is to characterize the performance of systems in fading which are not designed specifically to operate over fading channels. An example would be a satellite system for mobile users operating at UHF which is subject to ionospheric scintillation. Also, as opposed to the numerical results presented by Eaves and Levesque [19], this paper derives closed-form expressions which are either exact results or excellent approximations for the probability of block error in slow Rayleigh fading. The approximation technique is applicable in general to slow fading channel statistics and to arbitrary modulation formats.

II. PACKET RADIO SYSTEM MODELING FOR FADING CHANNELS

The channel throughput and packet delay of packet radio channels have been determined for basic system concepts such as pure-ALOHA, slotted-ALOHA, and carrier sense multiple access (CSMA) [20]–[23]. However, none of these fundamental calculations includes the effect of link errors due to noise and fading. In the absence of fading the noiseless assumption is quite good, but on a fading channel the signal-to-noise ratio becomes a critical parameter. The approach in this section is to derive pure-ALOHA throughput and delay as a function of the probability of a packet error and the probability of an acknowledgment error. We will assume that individual user transmissions are independent of

0090-6778/80/0200-0279$00.75