Simon S. Lam, *Packet Switching in a Multi-Access Broadcast Channel with Application to Satellite Communication in a Computer Network*, Ph.D. Dissertation, Computer Science Department, University of California at Los Angeles, March 1974; published as Technical Report UCLA-ENG-7429 (ARPA), School of Engineering and Applied Science, UCLA, April 1974, 306 pages.

Available in eleven .pdf files:

- TRcovers.pdf
- Abstract+ToC.pdf
- Chapter1.pdf
- Chapter2.pdf
- Chapter3.pdf
- Chapter4.pdf
- Chapter5.pdf
- Chapter6.pdf
- Chapters7-8.pdf
- Bibliography.pdf
- Appendices.pdf

CHAPTER 6

DYNAMIC CHANNEL CONTROL

6.1 Introduction

Before we introduce channel control procedures, let us first examine the motivation for dynamic channel control.

In Chapter 1, we indicated that our interest in the multi-access broadcast channel stems from its capability to provide communication among a large population of users. In Chapter 3, equilibrium throughput-delay tradeoffs were given for the infinite population model (which approximates a large population of small users). The lower envelope of these tradeoffs characterizes the optimum channel performance. In Chapter 5, we showed that when the number of channel users M is sufficiently small, the channel is stable and the optimum channel performance envelope can actually be achieved over an infinite time horizon. However, for a large M, the channel is unstable. In this case, the optimum throughput-delay performance is achievable only for some finite time period before the channel goes into saturation.

In this chapter, we study dynamic channel control procedures which will enable an originally unstable slotted ALOHA channel not only to support a large number of users, but also to achieve a throughput-delay performance close to the optimum envelope with guaranteed channel stability.

The linear feedback model described in Section 5.1 is assumed throughout. In addition to this assumption, each channel user is assumed to know the exact current channel state (channel backlog size). This assumption is necessary in the mathematical model, but will be relaxed when we consider heuristic (but practical) control procedures based upon the insights gained from the analysis.

Here we summarize the contents of this chapter. In Section 6.2, we give a brief introduction of Markov decision theory for a finite-state Markov process (chain) and outline Howard's policy-iteration method. Several control procedures are considered in Section 6.3. The first, known as the input control procedure (ICP), allows the channel to either accept or "reject" new packets from their sources. The second, known as the retransmission control procedure (RCP), allows the channel transmitters to impose either large or small retransmission delays on previously collided packets. The third, known as the input-retransmission control procedure (IRCP), is a combination of the first two as its name suggests. Two cost (performance) measures are defined, namely, the stationary channel throughput rate S_{out} and the average packet delay D . It will be shown in Section 6.4 that for each of the above control procedures, an optimal policy exists (and can be found by the policy-iteration method) which will maximize S_{out} and minimize D at the same time. An efficient computational algorithm is given in Section 6.5, which enables the use of the policy-iteration method for a large state space with relatively small computational and storage demands on the computer. Both numerical and simulation results are then given in Section 6.6 for the throughput-delay performance of the controlled random access channel. In all cases considered, the optimal control policies were found to be of the control limit type. However, a rigorous proof of this result remains as an open problem.

In Section 6.7, we recognize the fact that the exact current channel state is not known to the individual channel users. A procedure is proposed which estimates the channel state and applies the above optimal control policies using this estimate. Another retransmission control procedure which circumvents the state estimation problem is also suggested. These control procedures are then tested through simulations and found to give not only a stable channel, but also achieve a throughput-delay performance close to the optimum performance envelope. Other channel control schemes proposed by Metcalfe [METC 73A] and Rettberg [RETT 73C] are then examined. Finally, we briefly discuss some channel design considerations.

6.2 Some Results from Markov Decision Theory

Most of the results in this section are taken from Howard [HOWA 60, HOWA 71] and Ross [ROSS 70]. Also, see Parzen [PARZ 62] for a general reference on Markov chains.

6.2.1 Markov Processes with Costs

We consider a finite Markov process (chain) N^t which is observed at time points $t = 0, 1, 2, \ldots$ to be in one of a finite number of possible states. The set of states S will be labelled by the nonnegative integers $\{0, 1, 2, \ldots, M\}$. The Markov process is assumed to have stationary state transition probabilities $\{p_{ij}\}$ (unless stated otherwise). The process incurs a cost c_{ij} when it makes a transition from state i to state j. Thus, the Markov process starting at some initial state generates a sequence of costs as it makes transitions from state to state. Each c_{ij} is assumed to be bounded (i.e., $c_{ij} < \infty$) and independent of time (unless indicated otherwise).

We define C_i to be the expected immediate cost for state i and $v_i(\tau)$ to be the expected total cost that the process N^t incurs in the next $\tau + 1$ time units starting in state i. Hence,

$$C_i = \sum_{j=0}^{M} p_{ij}\, c_{ij} \tag{6.1}$$

$$v_i(\tau) = E\left[\sum_{t=0}^{\tau} C_{N^t} \;\middle|\; N^0 = i \right] \tag{6.2}$$

The expected total costs $v_i(\tau)$ are given by the following recurrence relation [HOWA 60]

$$\begin{aligned} v_i(\tau) &= \sum_{j=0}^{M} p_{ij} \left[c_{ij} + v_j(\tau - 1) \right] \qquad \begin{matrix} i = 0, 1, 2, \ldots, M \\ \tau = 1, 2, 3, \ldots \end{matrix} \\ &= C_i + \sum_{j=0}^{M} p_{ij}\, v_j\,(\tau - 1) \end{aligned} \tag{6.3}$$

This set of equations can be solved recursively for the set of expected total costs $\{v_i(\tau)\}$ for any finite τ. However, when τ (called the time horizon of the process N^t) is very large, a more suitable cost measure is the cost rate (i.e., expected cost per unit time) of the process. Thus, we define

$$g_i = \lim_{\tau \to \infty} \frac{1}{\tau + 1} E\left[\sum_{t=0}^{\tau} C_{N^t} \middle| N^0 = i \right] \tag{6.4}$$

where the limit always exists since the c_{ij} are bounded.

Assuming that N^t is an irreducible Markov chain and since S is finite, N^t possesses a unique stationary probability distribution $\{\pi_i\}_{i=0}^{M}$ such that [PARZ 62]

$$\pi_j = \sum_{i=0}^{M} \pi_i \, p_{ij} \qquad j = 0, 1, \ldots, M$$

$$\pi_i \geq 0 \qquad i = 0, 1, \ldots, M \tag{6.5}$$

and

$$\sum_{i=0}^{M} \pi_i = 1$$

From the ergodic theorems in the theory of Markov chains [CHUN 67], we then have the following important result

$$g = g_i = \sum_{j=0}^{M} \pi_j \, C_j \qquad \forall i = 0, 1, \ldots, M \tag{6.6}$$

where g is defined to be the cost rate or expected average cost of the process N^t and will be used extensively as the cost (performance) measure under various definitions of the state transition costs $\{c_{ij}\}$.

When τ is large, the expected total costs of the process, $v_i(\tau)$, are then given [HOWA 60] asymptotically by

$$v_i(\tau) = g\,\tau + v_i \qquad i = 0, 1, 2, \ldots, M \tag{6.7}$$

where v_i is referred to as the asymptotic intercept of state i . For a large τ , g is the only significant variable. (However, it will be shown below that in a Markov decision process, relative values of the v_i will enable us to solve for an optimal control policy.)

6.2.2 Markov Decision Processes

We now introduce decision-making in the Markov process described above. Let A be a finite set of possible actions such that corresponding to each action $a \in A$, the set of state transition probabilities $\{p_{ij}(a)\}$ and costs $\{c_{ij}(a)\}$ (or equivalently the expected immediate costs $\{C_i(a)\}$) are uniquely specified. We define a policy f to be any rule for choosing actions and P to be the class of all policies. The action chosen by a policy at time t may, for instance, depend on the history of the process up to that point or it may be randomized in the sense that it chooses action a with some probability P_a, $a \in A$.

Suppose the action a^t is given by the policy f at time t , which in turn specifies the state transition probabilities and costs at that time. Thus, f determines both the evolution in time of the Markov process N^t and the sequence of costs it incurs. For a policy f which generates the following sequence of actions in time $\{a^0, a^1, a^2, \ldots, a^t, \ldots\}$, we define the expected average cost per unit time for N^t which was initially in state i as

$$\phi_i(f) = \lim_{\tau\to\infty} \frac{1}{\tau + 1} E_f\left[\sum_{t=0}^{\tau} C_{N^t}(a^t) \mid N^0 = i\right] \qquad (6.8)$$

where the limit always exists, since the costs are assumed to be bounded; the expectation is taken conditioning on the policy f. We say that the policy f^* is *average cost optimal* over all policies if $\phi_i(f^*) = \min_{f\varepsilon P} \phi_i(f)$ for all $i \varepsilon S$.

An important subclass of all policies is the *class of stationary policies* P_s. A *stationary policy* is defined to be one which is *nonrandomized* and the action it chooses at time t depends only on the state of the process at time t. Thus, a stationary policy f is a function $f(\cdot) : S \to A$. The Markov decision process employing a stationary policy f is in fact a Markov process with stationary transition probabilities and costs as described in the previous section. In this case, from Eq. (6.6)

$$\phi_i(f) = g(f) = \sum_{j=0}^{M} \pi_j(f)C_j(f) \qquad \forall i = 0, 1, \ldots, M \qquad (6.9)$$

We give the following important result concerning stationary policies.

Theorem 6.1 Given a finite state space, if every stationary policy gives rise to an irreducible Markov chain, then there exists a stationary policy f^* which is optimal over the class of all policies. Thus,

$$g(f^*) = \phi_i(f^*) = \min_{f \in P} \phi_i(f) \qquad \forall i = 0, 1, \ldots, M$$

Proof See [ROSS 70].

The conditions in Theorem 6.1 will always be satisfied in our optimization problems below. Thus, by the above theorem, we can and shall limit our attention only to the class of stationary policies in our search for an optimal policy.

In the following section, we outline a procedure which solves for the cost rate g of a Markov decision process given a stationary policy f. An iteration method is then described, which leads to an optimal stationary policy within a finite number of iterations.

6.2.3 The Policy-Iteration Method [HOWA 60, HOWA 71]

Given a stationary policy f, the cost rate g of the resulting Markov process can be determined as follows. Substituting Eqs. (6.7) into Eqs. (6.3), we obtain

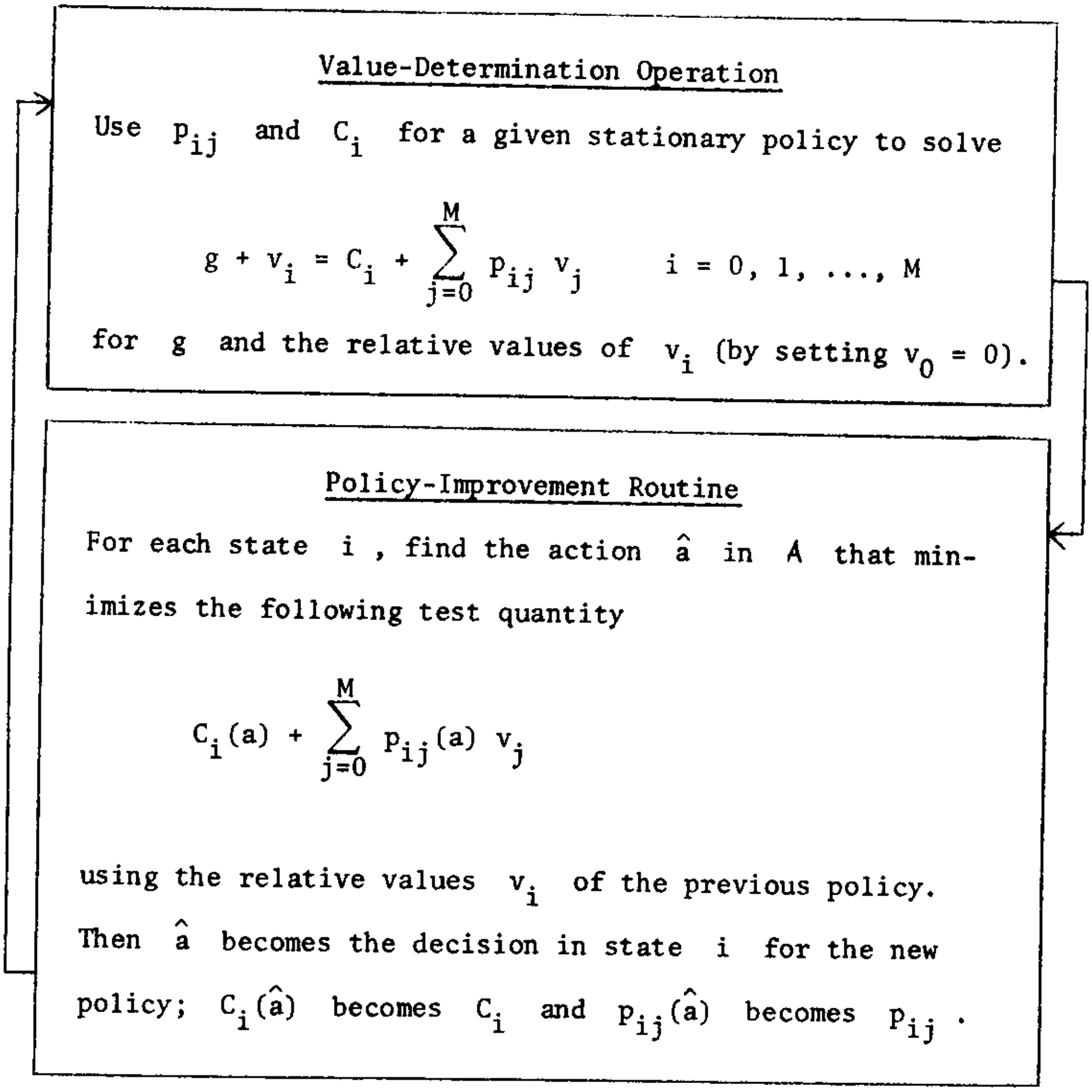
$$g + v_i = C_i + \sum_{j=0}^{M} p_{ij} v_j \qquad i = 0, 1, \ldots, M \qquad (6.10)$$

where the dependence of p_{ij} and C_i on the stationary policy f are suppressed. There are $(M + 2)$ unknown variables, namely, g

and $\{v_i\}$ in the $(M + 1)$ linear simultaneous equations. We see that Eqs. (6.10) are also satisfied if the v_i are replaced by $v_i + b$, where b is any arbitrary constant. Thus, although g can be determined uniquely, only relative values of the v_i can be obtained by solving Eqs. (6.10). Fortunately, g is the cost (performance) measure of the Markov process that we are interested in; a set of relative values of the v_i is sufficient for the purpose of the following iteration method in solving for an optimal policy.

The Policy-Iteration Method

The basic iteration cycle in the policy-iteration method is diagrammed below in Fig. 6-1.



Value-Determination Operation

Use p_{ij} and C_i for a given stationary policy to solve

$$g + v_i = C_i + \sum_{j=0}^{M} p_{ij}\, v_j \qquad i = 0, 1, \ldots, M$$

for g and the relative values of v_i (by setting $v_0 = 0$).

Policy-Improvement Routine

For each state i, find the action $\hat{a}$ in A that minimizes the following test quantity

$$C_i(a) + \sum_{j=0}^{M} p_{ij}(a)\, v_j$$

using the relative values v_i of the previous policy. Then $\hat{a}$ becomes the decision in state i for the new policy; $C_i(\hat{a})$ becomes C_i and $p_{ij}(\hat{a})$ becomes p_{ij} .

Figure 6-1 The policy-iteration cycle.

We may enter the iteration cycle in either box with an arbitrary initial policy or an arbitrary set of v_i . It is necessary to require that in the policy-improvement routine, if the decision f(i) for state i given by the old policy yields as small a value for the test quantity as any of the other actions in A , the decision is left unchanged. The stopping rule is as follows:

> The optimal policy has been reached (g is minimized) when the policies on two successive iterations are identical.

The following theorem on the policy-iteration method is due to Howard.

Theorem 6.2 (i) Suppose the policy-improvement routine has produced a policy f_2 that is different from the previous policy f_1 , then

$$g(f_2) < g(f_1)$$

(ii) An optimal policy is obtained within a finite number of iterations.

Proof See [HOWA 60].

6.3 The Controlled Random Access Channel Model

Consider the stable and unstable channels in Figs. 5-6(a) and (b). The channel operating point (n_o, S_o) gives the throughput-delay performance of a stable channel. However, for an unstable channel, the throughput-delay performance given by the channel operating point* (n_o, S_o) is what we strive to achieve over an infinite time horizon through the use of dynamic channel control.

In this section, channel control procedures are proposed and formulated under the assumption that all channel users have perfect

*We assume that the channel operating point has been optimized over K (such that n_o is minimized) and that the optimal K has been adopted as the operating value of K .

knowledge of the current channel state (channel backlog size). We shall refer to this assumption as perfect channel state information.

6.3.1 The Markov Process

Consider the linear feedback model in Section 5.1, which represents a slotted ALOHA channel supporting input from M small independent users. The channel backlog size N^t at time t is taken to be the state variable with the state space $S = \{0, 1, 2, \ldots, M\}$. As before, we assume that each channel user in the thinking state generates and transmits a new packet independently with probability σ in a time slot; each channel user in the blocked state independently retransmits his backlogged packet with probability p in each time slot.* Thus, with constant M, σ and p, N^t is a finite-state Markov process with stationary state transition probabilities given by Eqs. (5.1) which we rewrite below.

$$p_{ij} = \begin{cases} 0 & j \le i - 2 \\ ip(1-p)^{i-1}(1-\sigma)^{M-i} & j = i - 1 \\ (1-p)^{i}(M-i)\sigma(1-\sigma)^{M-i-1} & \\ \qquad + \left[1 - ip(1-p)^{i-1}\right](1-\sigma)^{M-i} & j = i \\ \left[1 - (1-p)^{i}\right](M-i)\sigma(1-\sigma)^{M-i-1} & j = i + 1 \\ \binom{M-i}{j-i}\sigma^{j-i}(1-\sigma)^{M-j} & j \ge i + 2 \end{cases}$$

$$0 \le i, j \le M \tag{6.11}$$

*We again assume $p = \frac{1}{R + (K+1)/2}$ in our numerical computations as in Chapter 5. Our numerical results will be presented in terms of K so that they can be compared with previous results.

Cost rates and performance measures

The performance measures of interst to us are the stationary channel throughput rate S_{out} and average packet delay D . We show here how we can define the expected immediate costs C_i such that either S_{out} or D can be obtained from the resulting cost rate g of the Markov process.

Given that the Markov process N^t is in state i at time t , the expected channel throughput in the time slot is given by Eq. (5.5), which we rewrite below as

$$S_{out}(i) = ip(1 - p)^{i-1}(1 - \sigma)^{M-i} + (1 - p)^{i}(M - i)\sigma(1 - \sigma)^{M-i-1} \tag{6.12}$$

Now define the expected immediate (throughput) cost for state i as

$$C_i = - S_{out}(i) \tag{6.13}$$

and define the resulting cost rate of N^t as g_s . It can easily be shown from Eqs. (6.11) that N^t is aperiodic and irreducible for $p , \sigma > 0$. Thus, N^t has a stationary probability distribution $\{\pi_i\}$. Using Eq. (6.6), the stationary channel throughput rate is given by

$$S_{out} = \sum_{i=0}^{M} S_{out}(i)\ \pi_i = -g_s \tag{6.14}$$

Note that S_{out} must be equal to the stationary channel input rate

$$S = \sum_{i=0}^{M} (M - i) \sigma \pi_i \tag{6.15}$$

To obtain the average packet delay D, we define the expected immediate (delay) cost for state i as

$$C_i = i$$

This accounts for the waiting cost of i packets incurred in the current time slot. In Markov decision theory terminology, this is sometimes referred to as the holding cost. Defining the resulting cost rate of the Markov process as g_d, we have from Eq. (6.6)

$$g_d = \sum_{i=0}^{M} i \pi_i$$

which is just the average channel backlog size $\bar{N}$ by definition. Applying Little's result [LITT 61], the average backlog time D_b of a packet is from Eq. (6.14)

$$D_b = \frac{\bar{N}}{S_{out}} = - \frac{g_d}{g_s}$$

and the average packet delay is from the above equation and Eq. (5.4)

$$D = - \frac{g_d}{g_s} + R + 1 \tag{6.16}$$

where $R + 1$ represent the packet transmission time and propagation delay incurred by every packet in its successful transmission.

We note that the cost rates g_s and g_d can be obtained using the value-determination operation in the previous section given the appropriate definitions of the expected immediate costs C_i . The performance measures of interest S_{out} and D can then be computed from g_s and g_d using Eqs. (6.14) and (6.16).

6.3.2 Channel Control Procedures

By channel control procedure we mean the set of available actions in the action space A . Given the above Markov process formulation of the channel, we propose the following control procedures for which there exist policies which convert an unstable channel into a stable channel:

(1) The input control procedure (ICP)

(2) The retransmission control procedure (RCP)

(3) The input-retransmission control procedure (IRCP)

In Appendix F, we consider a general dynamic channel control procedure which includes ICP, RCP and IRCP as special cases.

The input control procedure (ICP)

This control procedure corresponds to the action space of the Markov decision process, $A = \{\text{accept, reject}\} \overset{\Delta}{=} \{a,r\}$. Thus, in channel state i (i.e., given that $N^t = i$) , the actions are:

accept (action = a) or reject (action = r) all new packets that arrive* in the current time slot.

The retransmission control procedure (RCP)

Under this control procedure, the action space $A = \{p_o, p_c\} \overset{\Delta}{=} \{o, c\}$ where p_o and p_c are said to be the operating and control values of the retransmission probability p. (Through Eq. (5.3), p_o corresponds to K_o which gives the desired operating equilibrium contour and p_c corresponds to K_c which is large enough to render the channel stable.) Obviously, we must have $p_c < p_o$. Thus, in channel state i, the actions are: every backlogged packet is retransmitted in the current time slot with probability p_o (action = o) or with probability p_c (action = c).

In both control procedures, we see that channel stability is obtained through additional delays incurred by some or all packets in the system. However, they differ in their selection of such packets when the current channel state calls for "sacrifice" (i.e., choosing an action = r or c). In ICP, new packets are delayed ("rejected"); whereas in RCP, the backlogged packets are delayed for the social good.

The input-retransmission control procedure (IRCP)

This control procedure is a combination of ICP and RCP with the action space $A = \{(\text{accept}, p_o), (\text{accept}, p_c), (\text{reject}, p_o),$

*As discussed in Section 2.3.2, a new packet is said to arrive in the current time slot only after it has been generated by the channel user (or its external source), processed and ready for transmission over the channel in the current time slot. In the mathematical model, the rejected arrival is lost and the channel user generates a "new" packet in the next time slot with probability σ, etc. In a practical system, this new packet must actually be the previously rejected packet! We shall elaborate on this interpretation further below.

(reject, p_c)} $\overset{\Delta}{=}$ {ao, ac, ro, rc} . Thus, for example, when the action rc is taken, both new and backlogged packets are delayed.

By Theorem 6.2, an optimal stationary policy can always be found. By virtue of Theorem 6.1, the optimal stationary policy given by the policy-iteration method is optimal over the class of all policies P for the given control procedure (action space A) . However, we do not claim that the control procedures we consider here give optimal policies over the class of all possible control procedures (action spaces). There are two reasons why we do not consider more multi-action control procedures other than IRCP.* (For example, RCP may be generalized so that $A = \{p_o, p_1, p_2, p_3\}$.) First, we realize that the channel state is in reality not exactly known but must be estimated. When A has many actions, the partitioning of the state space S induced by the control policy f may be too "fine" compared to estimation errors. Second, as we show below, the control procedures proposed above will give channel throughput-delay tradeoffs very close to the optimum envelope of the infinite population model (for which we ignored stability considerations). Hence, more elaborate control procedures will only give minute incremental improvement in channel performance.

A stationary policy can be defined by a function $f : S \to A$. For ICP, any stationary policy is uniquely specified by the sets S_a and S_r such that $S = S_a \cup S_r$, $S_a \cap S_r = \phi$ (the null set) and

$$f(i) = \begin{cases} a & i \in S_a \\ r & i \in S_r \end{cases} \tag{6.17}$$

*A general dynamic control procedure is considered in Appendix F .

Similarly, a stationary policy of RCP is given by

$$f(i) = \begin{cases} o & i \in S_o \\ c & i \in S_c \end{cases} \tag{6.18}$$

where $S_o \cup S_c = S$ and $S_o \cap S_c = \phi$.

Within the class of stationary policies, a subclass of policies known as <u>control limit policies</u> can be described as follows for a two-action space A . Either the policy specifies the same action for all the states in S or there is a critical state $\hat{n}$ $(= 0, 1, 2, \ldots, M - 1)$ such that if the policy specifies one action for states 0 to $\hat{n}$, the other action is specified for states $\hat{n} + 1$ to M . $\hat{n}$ is said to be the <u>control limit.</u>

In Figs. 6-2 and 6-3, we show channel load lines corresponding to channels under ICP and RCP respectively. We find it easier to illustrate in both cases with control limit policies. In Fig. 6-2, $\hat{n}$ is the ICP control limit. When $N^t \leq \hat{n}$, the channel input rate $S^t = (M - N^t)\sigma$; when $N^t > \hat{n}$, $S^t = 0$. Similarly, suppose $\hat{n}$ is the RCP control limit in Fig. 6-3. When $N^t = n$, $K = K_o$, but as soon as N^t exceeds $\hat{n}$, $K = K_c$ is used. Note that both controlled channels are stable since the channel saturation point as shown in Fig. 5-6(b) no longer exists.

6.3.3 <u>The Input Control Procedure (ICP)</u>

Under this control procedure, recall that the action space $A = \{\text{accept, reject}\} = \{a, r\}$. We give below the state transition probabilities and costs of the Markov process N^t induced by each action in A .

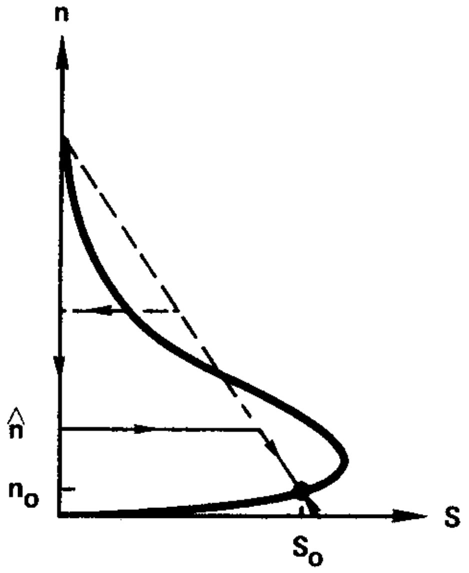


Figure 6-2. **An ICP Control Limit Policy Example.**

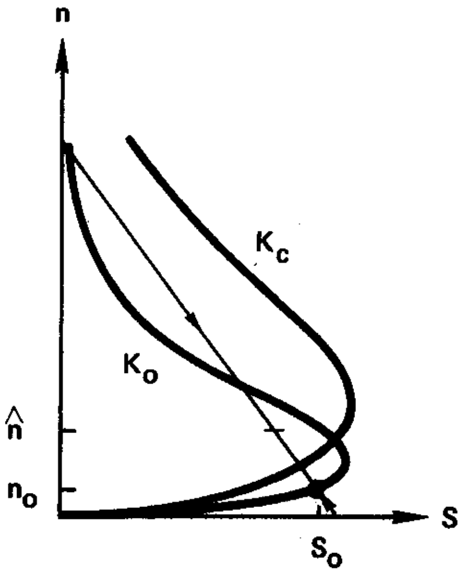


Figure 6-3. **A RCP Control Limit Policy Example.**

State transition probabilities

Suppose the channel is in state i (= 0, 1, ..., M) and the stationary control policy f(i) = a then $p_{ij}(a)$ is exactly as given in Eqs. (6.11), which we rewrite as

$$p_{ij}(a) = \begin{cases} 0 & j \le i - 2 \\ ip(1-p)^{i-1}(1-\sigma)^{M-i} & j = i - 1 \\ (1-p)^{i}(M-i)\sigma(1-\sigma)^{M-i-1} & \\ \qquad + \left[1 - ip(1-p)^{i-1}\right](1-\sigma)^{M-i} & j = i \\ \left[1-(1-p)^{i}\right](M-i)\sigma(1-\sigma)^{M-i-1} & j = i + 1 \\ \binom{M-i}{j-i}\sigma^{j-i}(1-\sigma)^{M-j} & j \ge i + 2 \\ & 0 \le i,\ j \le M \end{cases} \tag{6.19}$$

Suppose $f(i) = r$, then $p_{ij}(r)$ is given as follows.

$$p_{ij}(r) = \begin{cases} ip(1-p)^{i-1} & j = i-1 \\ 1 - ip(1-p)^{i-1} & j = i \\ 0 & \text{otherwise} \end{cases} \tag{6.20}$$

Except in the uninteresting cases when σ, $p = 0$ or $f(i) = r$ for all $i \varepsilon S$, the Markov process N^t under this control procedure is aperiodic and irreducible satisfying the conditions of Theorem 6.1.

Rejection costs

As in the Markov process formulation of an uncontrolled channel described in Section 6.2.1, expected immediate costs are incurred in every time slot. Depending on the performance measure (D or S_{out}), there is a holding cost which pertains to packet delays and there is a negative cost which is the expected channel throughput in that time slot. With ICP, we also introduce the rejection cost d_r which is the expected cost in units of delay per packet arrival rejected.

For an interpretation of this cost in terms of its effect on packet delays, we consider as an example the possible terminal access communications environment depicted in Fig. 6-4. A person sitting at a terminal generates a new packet with an average think time of $\frac{1}{\sigma}$ whenever his previous packet has been successfully transmitted. If, at the time of a packet arrival, the channel is in the reject state, this packet is lost in the sense that it is not transmitted

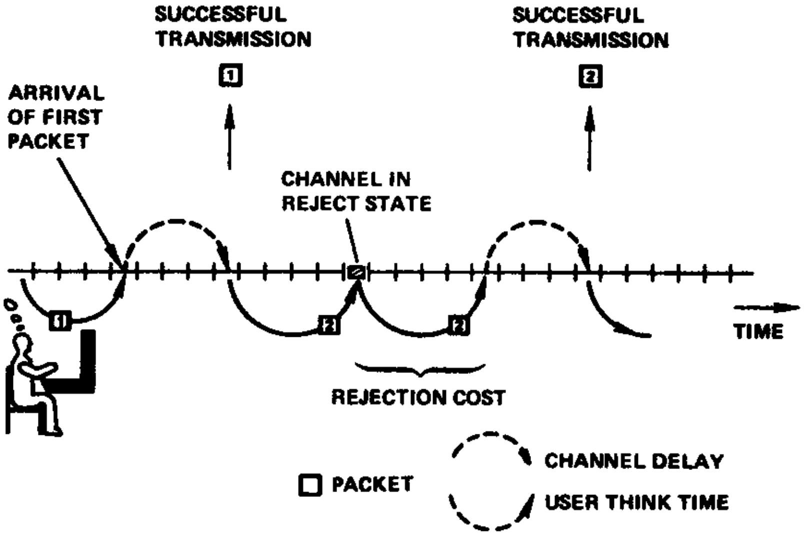


Figure 6-4. An Interpretation of the Rejection Cost.

over the broadcast channel at this time. In a practical situation, the user may be informed of the event and must enter some command character to "restart" the packet. Hence, the cost in terms of delay is probably in the order of an average think time $(= \frac{1}{\sigma})$. Let

$$d_r = \frac{\alpha}{\sigma} \tag{6.21}$$

<u>We shall assume $\alpha = 1$ throughout this chapter</u>. This assumption is actually necessitated by our Markov process model in Section 6.3.1, where each thinking user is assumed to transmit a new packet (which may be a previously rejected new packet) with probability σ in a time slot.

It is easy to think of situations in which α is not one. For example, we may want to insert additional delays to rejected packets or to account for some terminal processing time by using $\alpha > 1$. On the other hand, the human user may be very impatient and restarts his rejected packets very quickly such that $\alpha < 1$. (In this case, the terminal can always insert additional delays to make $\alpha = 1$.) In any case, if $\alpha \neq 1$, our channel state description will become more complex since we must distinguish blocked users who transmit in a time slot with probability p, thinking users with rejected packets who transmit with probability $\frac{\sigma}{\alpha}$ and the other thinking users who transmit with probability σ. Assuming $\alpha = 1$ in ICP (and also IRCP) simplifies the state description and consequently the amount of computation required in the policy-iteration method.

Average packet delay and channel throughput rate

Consider a stationary control policy $f : \mathcal{S} \to \mathcal{A}$ uniquely specified by the sets $\mathcal{S}_a$ and $\mathcal{S}_r$. The expected immediate (delay) costs for state i are (assuming $\alpha = 1$)

$$C_i(a) = i \tag{6.22}$$

$$\begin{aligned} C_i(r) &= i + (M - i)\, \sigma\, d_r \\ &= i + (M - i) \\ &= M \end{aligned} \tag{6.23}$$

From Eq. (6.9), the cost rate of the process N^t under policy f is given by

$$g_d(f) = \sum_{i=0}^{M} \pi_i(f)\ C_i(f)$$

$$= \sum_{i=0}^{M} i\ \pi_i(f) + d_r \sum_{i \varepsilon S_r} (M - i)\sigma\ \pi_i(f) \qquad (6.24)$$

where $\{\pi_i(f)\}$ are the stationary probabilities of the process N^t whose state transition probabilities $\{p_{ij}(f)\}$ are given by Eqs. (6.19) and (6.20). Define

$$\lambda_r = \sum_{i \varepsilon S_r} (M - i)\sigma\ \pi_i(f) \qquad (6.25)$$

to be the rate of packet rejection for all the channel users. Thus, Eq. (6.24) can be rewritten as

$$g_d(f) = \sum_{i=0}^{M} i\ \pi_i(f) + \lambda_r\ d_r$$

$$= \overline{N} + \overline{N_r} \qquad (6.26)$$

where by Little's result [LITT 61], $\overline{N}$ is the average channel backlog size and $\overline{N_r}$ is the average number of rejected packets in the system. Considering Fig. 6-5 and applying Little's result once more, the average packet delay (including rejection delays) is given by

$$D = \frac{g_d(f)}{S_{out}} + R + 1 \qquad (6.27)$$

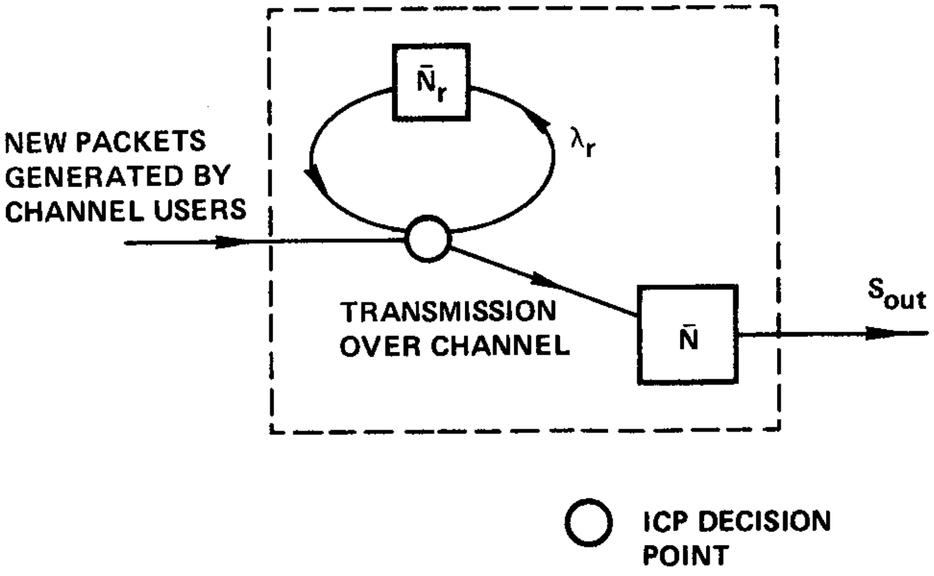

NEW PACKETS GENERATED BY CHANNEL USERS
$\bar{N}_r$
λ_r
TRANSMISSION OVER CHANNEL
$\bar{N}$
S_{out}
ICP DECISION POINT

Figure 6-5. Average Number of Packets in the System Under ICP.

where as before $R + 1$ account for the packet transmission time and propagation delay for the successful transmission and S_{out} is the stationary channel throughput rate to be obtained in the following manner.

Given policy f as above, we define the following expected immediate (throughput) costs

$$C_i(a) = -S_{out}(i, a)$$

$$= -[ip(1-p)^{i-1}(1-\sigma)^{M-i} + (1-p)^i(M-i)\sigma(1-\sigma)^{M-i-1}] \qquad (6.28)$$

$$C_i(r) = -S_{out}(i, r)$$

$$= -ip(1 - p)^{i-1} \qquad (6.29)$$

Using the above definitions, the cost rate of the Markov process N^t is by Eq. (6.9)

$$g_s(f) = - \sum_{i=0}^{M} \pi_i(f)\, S_{out}(i, f)$$

Thus, the stationary channel throughput rate is

$$S_{out} = - g_s(f) \tag{6.30}$$

The average packet delay is from Eq. (6.27)

$$D = - \frac{g_d(f)}{g_s(f)} + R + 1 \tag{6.31}$$

Given f, $g_d(f)$ and $g_s(f)$ can be calculated using the value-determination operation in the policy-iteration method assuming delay and throughput costs respectively.

6.3.4 The Retransmission Control Procedure (RCP)

Under this control procedure, the action space $A = \{p_o, p_c\} = \{o, c\}$. We give below the state transition probabilities and costs of the Markov process N^t induced by each action in A.

State transition probabilities

Suppose the channel is in state i (= 0, 1, ..., M) and action p_o is selected, then $p_{ij}(o)$ is given by

$$p_{ij}(o) = \begin{cases} 0 & j \le i - 2 \\ i\, p_o(1 - p_o)^{i-1}(1 - \sigma)^{M-i} & j = i - 1 \\ (1-p_o)^i(M-i)\sigma(1-\sigma)^{M-i-1} + [1-ip_o(1-p_o)^{i-1}](1-\sigma)^{M-i} & \\ & j = i \\ [1 - (1 - p_o)^i](M - i)\sigma(1 - \sigma)^{M-i-1} & j = i + 1 \\ \binom{M - i}{j - i} \sigma^{j-i} (1 - \sigma)^{M-j} & j \ge i + 2 \end{cases} \tag{6.32}$$

If action p_c is selected, then $p_{ij}(c)$ is given by

$$p_{ij}(c) = \begin{cases} 0 & j \le i - 2 \\ i\, p_c(1 - p_c)^{i-1}(1 - \sigma)^{M-i} & j = i - 1 \\ (1-p_c)^i(M-i)\sigma(1-\sigma)^{M-i-1} + [1-ip_c(1-p_c)^{i-1}](1-\sigma)^{M-i} & \\ & j = i \\ [1 - (1 - p_c)^i](M - i)\sigma(1 - \sigma)^{M-i-1} & j = i + 1 \\ \binom{M - i}{j - i}\sigma^{j-i}(1 - \sigma)^{M-j} & j \ge i + 2 \end{cases} \tag{6.33}$$

Except in the uninteresting cases when σ , p_o or $p_c = 0$, the Markov process N^t under this control procedure is aperiodic and irreducible satisfying the conditions of Theorem 6.1.

Average packet delay and channel throughput rate

Consider a stationary control policy $f : S \rightarrow A$ uniquely specified by the sets S_o and S_c . The expected immediate (delay) cost in state i is just the holding cost for both actions,

$$C_i(o) = C_i(c) = i \tag{6.34}$$

As before, the resulting cost rate of N^t is given by Eq. (6.9)

$$g_d(f) = \sum_{i=0}^{M} \pi_i(f)C_i(f) = \sum_{i=0}^{M} i\ \pi_i(f) = \overline{N} \tag{6.35}$$

Thus, the average packet delay is given by Eq. (6.27)

$$D = \frac{g_d(f)}{S_{out}} + R + 1 \tag{6.27}$$

where S_{out} is the stationary channel throughput rate.

The expected immediate (throughput) costs are given by

$$C_i(o) = - S_{out}(i, o)$$

$$= - [i\ p_o(1-p_o)^{i-1}(1-\sigma)^{M-i} + (1-p_o)^i(M-i)\sigma(1-\sigma)^{M-i-1}] \tag{6.36}$$

$$C_i(c) = - S_{out}(i, c)$$

$$= -[i\ p_c(1-p_c)^{i-1}(1-\sigma)^{M-i} + (1-p_c)^i(M-i)\sigma(1-\sigma)^{M-i-1}] \tag{6.37}$$

Using the above definitions, the cost rate of the Markov process N^t is given by

$$g_s(f) = - \sum_{i=0}^{M} \pi_i(f)\ S_{out}(i, f)$$

Thus, the stationary channel throughput rate is again

$$S_{out} = - g_s(f) \tag{6.30}$$

and the average packet delay is

$$D = - \frac{g_d(f)}{g_s(f)} + R + 1 \tag{6.31}$$

6.3.5 The Input-Retransmission Control Procedure (IRCP)

This control procedure is a combination of ICP and RCP. The action space $A = \{(\text{accept}, p_o), (\text{accept}, p_c), (\text{reject}, p_o), (\text{reject}, p_c)\} = \{ao, ac, ro, rc\}$. We give below the state transition probabilities and costs of the Markov process N^t induced by each action in A.

State transition probabilities

For $i = 0, 1, 2, \ldots, M$

$$p_{ij}(ao) = \begin{cases} 0 & j \le i - 2 \\ i\,p_o(1 - p_o)^{i-1}(1 - \sigma)^{M-i} & j = i - 1 \\ (1-p_o)^i(M-i)\sigma(1-\sigma)^{M-i-1} + [1-ip_o(1-p_o)^{i-1}](1-\sigma)^{M-i} & \\ & j = i \\ [1 - (1 - p_o)^i](M - i)\sigma(1 - \sigma)^{M-i-1} & j = i + 1 \\ \binom{M - i}{j - i} \sigma^{j-i} (1 - \sigma)^{M-j} & j \ge i + 2 \end{cases} \tag{6.38}$$

$$p_{ij}(ac) = \begin{cases} 0 & j \le i - 2 \\ i\,p_c(1 - p_c)^{i-1}(1 - \sigma)^{M-i} & j = i - 1 \\ (1-p_c)^i(M-i)\sigma(1-\sigma)^{M-i-1} + [1-ip_c(1-p_c)^{i-1}](1-\sigma)^{M-i} & \\ & j = i \\ [1 - (1 - p_c)^i](M - i)\sigma(1 - \sigma)^{M-i-1} & j = i + 1 \\ \binom{M - i}{j - i}\sigma^{j-i}(1 - \sigma)^{M-j} & j \ge i + 2 \end{cases} \tag{6.39}$$

$$p_{ij}(ro) = \begin{cases} i\ p_o(1 - p_o)^{i-1} & j = i - 1 \\ 1 - i\ p_o(1 - p_o)^{i-1} & j = i \\ 0 & \text{otherwise} \end{cases} \qquad (6.40)$$

$$p_{ij}(rc) = \begin{cases} i\ p_c(1 - p_c)^{i-1} & j = i - 1 \\ 1 - i\ p_c(1 - p_c)^{i-1} & j = i \\ 0 & \text{otherwise} \end{cases} \qquad (6.41)$$

As before, we neglect the uninteresting cases when σ , p_o or $p_c = 0$.

Average packet delay and channel throughput rate

Consider a stationary control policy $f : S \to A$ uniquely specified by the nonintersecting sets S_{ao}, S_{ac}, S_{ro} and S_{rc} such that

$$S = S_{ao} \cup S_{ac} \cup S_{ro} \cup S_{rc}$$

and

$$f(i) = \begin{cases} ao & i \in S_{ao} \\ ac & i \in S_{ac} \\ ro & i \in S_{ro} \\ rc & i \in S_{rc} \end{cases} \qquad (6.42)$$

Let

$$\begin{aligned} S_a &= S_{ao} \cup S_{ac} \\ S_r &= S_{ro} \cup S_{rc} \end{aligned} \tag{6.43}$$

Define the expected immediate (delay) costs to be

$$C_i(ao) = C_i(ac) = i \tag{6.44}$$

and

$$\begin{aligned} C_i(ro) = C_i(rc) &= i + (M - i)\sigma \cdot d_r \\ &= M \end{aligned} \tag{6.45}$$

The cost rate of the process N^t under policy f is given by Eq. (6.24)

$$g_d(f) = \sum_{i=0}^{M} i\ \pi_i(f) + d_r \sum_{i \in S_r} (M - i)\sigma\ \pi_i(f) \tag{6.24}$$

and the average packet delay (including rejection delay) is given by Eq. (6.27)

$$D = \frac{g_d(f)}{S_{out}} + R + 1 \tag{6.27}$$

To obtain S_{out}, the following expected immediate (throughput) costs are adopted.

$$C_i(ao) = - S_{out}(i,\ ao)$$

$$= -[i\ p_o(1-p_o)^{i-1}(1-\sigma)^{M-i} + (1-p_o)^i(M-i)\sigma(1-\sigma)^{M-i-1}] \tag{6.46}$$

$$C_i(ac) = - S_{out}(i,\ ac)$$

$$= -[i\ p_c(1-p_c)^{i-1}(1-\sigma)^{M-i} + (1-p_c)^i(M-i)\sigma(1-\sigma)^{M-i-1}] \tag{6.47}$$

$$C_i(ro) = - S_{out}(i,\ ro)$$

$$= -i\ p_o(1 - p_o)^{i-1} \tag{6.48}$$

$$C_i(rc) = - S_{out}(i,\ rc)$$

$$= -i\ p_c(1 - p_c)^{i-1} \tag{6.49}$$

The cost rate of the Markov process and the stationary channel throughput rate are again given by

$$g_s(f) = - \sum_{i=0}^{M} \pi_i(f)\ S_{out}(i,f)$$

and

$$S_{out} = -g_s(f) \tag{6.30}$$

Thus,

$$D = - \frac{g_d(f)}{g_s(f)} + R + 1 \tag{6.31}$$

6.4 A Theorem on the Equivalence of the Performance Measures

The channel throughput rate S_{out} and average packet delay D constitute the performance measures of interest for the controlled channel. Under any one of the previously described channel control procedures and given a stationary control policy f, either one of the performance measures can be evaluated by appropriate definitions of the state transition probabilities and expected immediate costs of the Markov decision process N^t. The value-determination operation yields the cost rate of N^t, from which the value of the performance measure can be computed. Given a single performance measure, the policy-iteration method will, in fact, lead to an optimal stationary policy with respect to the given performance measure in a finite number of steps.

Under any one of the control procedures, some obvious optimization problems seem to be:

(1) $\underset{f \in P_s}{\text{Min}}\ D$ given some (minimum) constraint on S_{out}

(2) $\underset{f \in P_s}{\text{Max}}\ S_{out}$ given some (maximum) constraint on D

(3) $\underset{f \in P_s}{\text{Min}}\ (D - \beta S_{out})$ for some $\beta > 0$

where P_s is the class of all stationary policies. Markov decision theory as introduced in Section 6.2 does not provide for the solution of the first two optimization problems with constraints. In the third problem, there is no natural candidate for the positive constant β

which determines the relative weights we put on the two performance measures (D and S_{out}) . Luckily, we have been able to establish the following lemma and theorem which enable us to get around this difficulty.

Lemma 6.3 Under each of the control procedures ICP, RCP or IRCP

$$g_d(f) = \frac{g_s(f)}{\sigma} + M \tag{6.50}$$

where f is any stationary control policy.

Proof The proof hinges on the observation that under a stationary control policy, N^t is a finite-state Markov process with stationary transition probabilities in which case the stationary channel throughput rate S_{out} must be equal to the stationary channel input rate.

We first consider the input control procedure (ICP). From Eqs. (6.21) and (6.24)

$$\begin{aligned} g_d(f) &= \sum_{i=0}^{M} i\ \pi_i(f) + \frac{1}{\sigma} \sum_{i\varepsilon S_r} (M - i)\sigma\ \pi_i(f) \\ &= \sum_{i=0}^{M} i\ \pi_i(f) + \frac{1}{\sigma} \sum_{i\varepsilon S_r} (M - i)\sigma\ \pi_i(f) \\ &\quad + \frac{1}{\sigma} \sum_{i\varepsilon S_a} (M - i)\sigma\ \pi_i(f) - \frac{1}{\sigma} \sum_{i\varepsilon S_a} (M - i)\sigma\ \pi_i(f) \end{aligned}$$

$$= \sum_{i=0}^{M} i\ \pi_i(f) + M - \sum_{i=0}^{M} i\ \pi_i(f) - \frac{1}{\sigma} \sum_{i \varepsilon S_a} (M-i)\sigma\ \pi_i(f)$$

$$= -\frac{1}{\sigma} \sum_{i \varepsilon S_a} (M - i)\sigma\ \pi_i(f) + M$$

Note that $\sum_{i \varepsilon S_a} (M - i)\sigma\ \pi_i(f)$ is just the stationary channel input rate and is thus equal to the stationary channel throughput rate $S_{out} = -g_s(f)$. Hence,

$$g_d(f) = \frac{g_s(f)}{\sigma} + M$$

and the proof is complete for ICP.

We next consider the retransmission control procedure (RCP). From Eq. (6.35),

$$g_d(f) = \sum_{i=0}^{M} i\ \pi_i(f)$$

$$= \sum_{i=0}^{M} (i - M)\ \pi_i(f) + M$$

$$= -\frac{1}{\sigma} \sum_{i=0}^{M} (M - i)\sigma\ \pi_i(f) + M$$

Again, $\sum_{i=0}^{M} (M - i)\sigma \pi_i(f)$ is just the stationary channel input rate and is thus equal to $S_{out} = -g_s(f)$. Thus,

$$g_d(f) = \frac{g_s(f)}{\sigma} + M$$

and the proof is complete for RCP.

The proof for IRCP is identical to that for ICP.

Q.E.D.

Theorem 6.4 Under each of the control procedures ICP, RCP or IRCP,

(i) there exists a stationary policy $\hat{f}$ such that

$$g_d(\hat{f}) = \min_{f \varepsilon P_s} g_d(f)$$

if and only if

$$g_s(\hat{f}) = \min_{f \varepsilon P_s} g_s(f)$$

(ii) if $\hat{f}$ is a stationary policy satisfying the preceding condition, then $\hat{f}$ minimizes D over the class P of all policies and at the same time, $\hat{f}$ maximizes S_{out} over the class P of all policies.

Proof (i) This is a direct consequence of Lemma 6.3 and the existence of $\hat{f}$ is guaranteed by Theorem 6.2. (ii) By Eqs. (6.30) and (6.31), $\hat{f}$ minimizes D and maximizes S_{out} over the class of all stationary policies. The generalization to the class $\mathcal{P}$ of all policies is a consequence of Theorem 6.1. Q.E.D.

Lemma 6.3 and Theorem 6.4 can be generalized to control procedures similar to ICP, RCP and IRCP, but with more alternatives in their action spaces. This is done in Appendix F.

Summarizing the results in Theorems 6.1, 6.2 and 6.4, we state that *under each of the control procedures ICP, RCP and IRCP, a stationary policy $f : \mathcal{S} \to \mathcal{A}$ always exists which minimizes the average packet delay D and maximizes the stationary channel throughput rate S_{out} over the class $\mathcal{P}$ of all policies. Such an optimal control policy and its channel performance measures D and S_{out} can be obtained by applying the policy-iteration method*. In the next section, we shall present an efficient computational algorithm which utilizes the policy-iteration method.

An interpretation of Theorem 6.4 and the optimization problem

The average packet delay D is given by Eq. (6.31) as

$$D = - \frac{g_d(f)}{g_s(f)} + R + 1 \tag{6.31}$$

where f is a stationary control policy in any of the above control procedures. Applying Eqs. (6.30) and (6.50) to substitute for $g_d(f)$ and $g_s(f)$ in the above equation, we have

$$D = R + 1 + \left(\frac{M}{S_{out}} - \frac{1}{\sigma} \right) \tag{6.51}$$

which relates D as a one-to-one function of S_{out} given fixed values of R , M and σ . (Note that the last two variables determine the channel load line.) Moreover, this function is monotonically decreasing.

Assuming a fixed R , we show in Fig. 6-6 a family of curves each of which depicts D as a function of S_{out} given by Eq. (6.51). The parameters M and σ , which determine the channel load line, also define a curve in the two-dimensional space of the performance measures D and S_{out} . We may consider each one of the control procedures in Section 6.3 as a mathematical operator which maps P_s (the space of all stationary policies) into the above curve. Each f in P_s is mapped into one point on the curve. The range space of the operator must be a proper subset of points on the curve. Otherwise, it is possible that $D = R + 1$ and $S_{out} = M\sigma$ (i.e., no congestion at all!). The optimization problem thus corresponds to finding the extreme points (maximum S_{out} and minimum D) of the range space. Since the curve under consideration is monotonically decreasing, these extreme points coincide. Thus, the same control policy f must maximize S_{out} and minimize D at the same time.

Given a family of channel load lines (e.g., M varying from 0 to ∞ at fixed σ or σ varying from 0 to 1 at fixed M) each channel control procedure gives rise to an infeasible region such as shown in Fig. 6-6. The boundary of this region represents the optimum throughput-delay tradeoff under the above constraints. The optimization problem here is to find the optimal control policies which

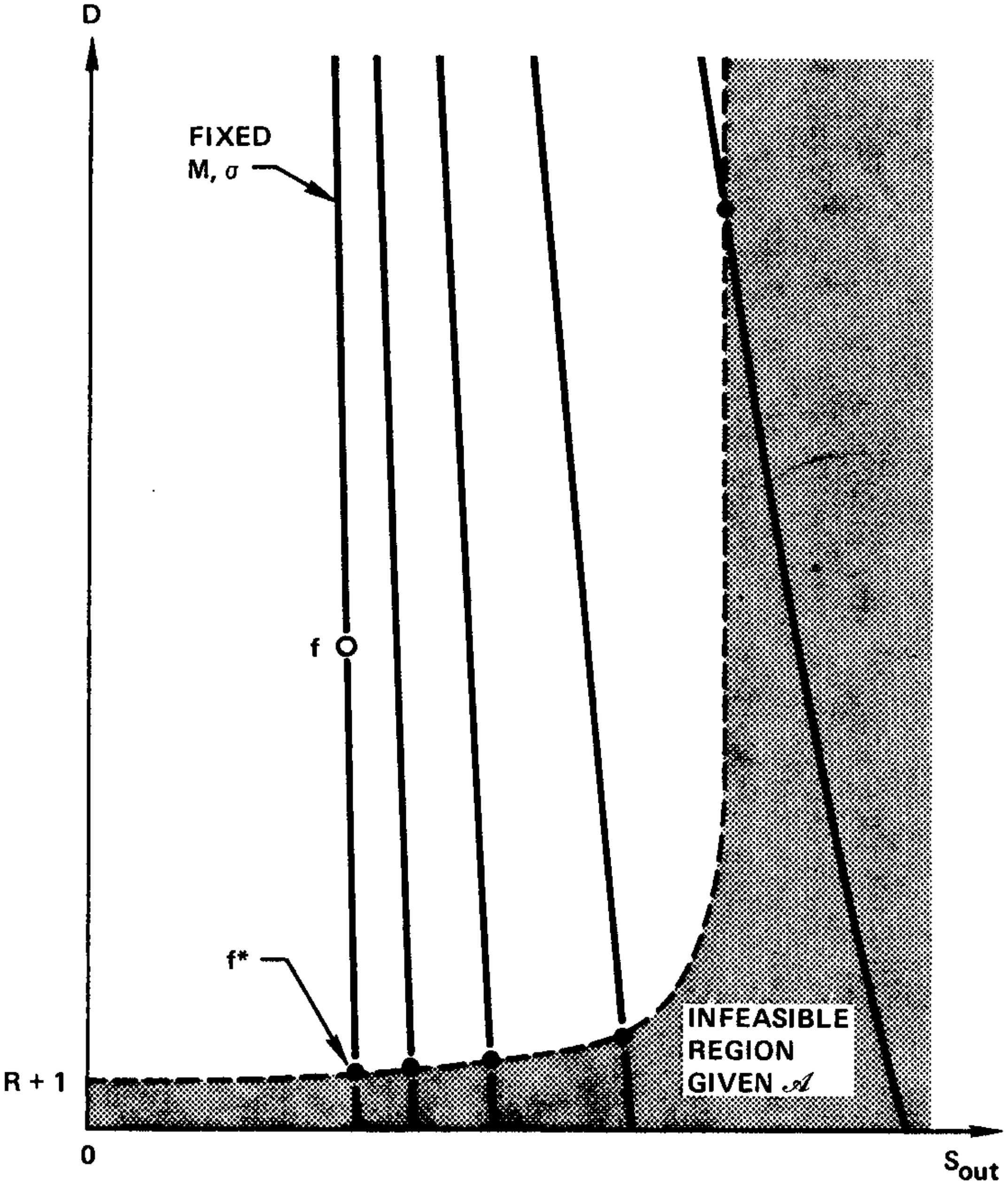


Figure 6-6. Optimum Performance of a Channel Control Procedure.

achieve this optimum channel performance. Below we give a computational algorithm to do so.

6.5 An Efficient Computational Algorithm (POLITE)

In any optimization problem, the optimum solution is readily available if we can enumerate all possible solutions. Thus, an optimization problem is solvable in the sense that the task of enumerating the set of possible solutions is within the limits of the computing capability of our machine(s). Even when a problem is solvable, we must look for ways to reduce the computational cost in terms of the time and space allocation of our machine(s) to the problem.

For the problem at hand, we have seen the tremendous savings in computational cost by reducing the set of possible solutions from the class of all policies to the class of stationary policies. Still, we have not altogether escaped from the "curse of dimensionality" since, for example, if S has 300 states and A has two actions, there are still 2^{300} (an astronomical number) stationary policies to consider. Howard's policy-iteration method described in Section 6.2.3 enables us to find an optimal policy usually in a small number of iterations. The method is composed of two parts as shown in Fig. 6-1, the value-determination operation and the policy-improvement routine. The difficulty now arises in the solution of the $(M + 1)$ linear simultaneous equations in Eqs. (6.10) for g and the relative values of v_i (setting $v_0 = 0$) when M is large (say, a few hundred, which is our range of interest).

$$g + v_i = C_i + \sum_{j=0}^{M} p_{ij}\ v_j \qquad i = 0, 1, 2, \dots M \tag{6.10}$$

For example, if $M = 400$, the task to solve Eqs. (6.10) is somewhat equivalent to inverting a 401 x 401 matrix with 160,801 entries! The fact that the state transition probabilities $p_{ij} = 0$ for $j \le i - 2$ in all our models enables us to decompose the $(M + 1)$ linear simultaneous equations in Eqs. (6.10) into two sets of M linear simultaneous equations, each of which can then be solved by applying Algorithm 5.1. We summarize the procedure in the following algorithm, which plays a crucial role in making possible the use of the policy-iteration method to solve optimization problems involving hundreds of channel users. Its derivation is given in Appendix E.

Algorithm 6.5

This algorithm solves for g and $\{v_i\}_{i=1}^{M}$ in the following set of $(M + 1)$ linear simultaneous equations,

$$\begin{aligned} g &= C_0 + \sum_{j=1}^{M} p_{0j}\ v_j \\ g + v_1 &= C_1 + \sum_{j=1}^{M} p_{1j}\ v_j \\ g + v_i &= C_i + \sum_{j=i-1}^{M} p_{ij}\ v_j \qquad i = 2, 3, \dots, M \end{aligned} \tag{6.52}$$

where

$$\sum_{j=0}^{M} p_{0j} = \sum_{j=i-1}^{M} p_{ij} = 1 \qquad i = 1, 2, \dots, M$$

(1) Define

$$b_{M-1} = \frac{1}{p_{M,M-1}}$$

$$d_{M-1} = - \frac{C_M}{p_{M,M-1}}$$

(2) For $i = M - 1, M - 2, \ldots, 2$ solve recursively

$$b_{i-1} = \frac{1}{p_{i,i-1}} \left[b_i + 1 - \sum_{j=i}^{M-1} p_{ij}\ b_j \right]$$

$$d_{i-1} = \frac{1}{p_{i,i-1}} \left[d_i - C_i - \sum_{j=i}^{M-1} p_{ij}\ d_j \right]$$

(3) Define

$$u_M = - \frac{1}{p_{10}} \left[b_1 + 1 - \sum_{j=1}^{M-1} p_{1j}\ b_j \right]$$

$$w_M = - \frac{1}{p_{10}} \left[d_1 - C_1 - \sum_{j=1}^{M-1} p_{1j}\ d_j \right]$$

$$u_i = u_M + b_i \qquad\qquad i = 1, 2, \ldots, M - 1$$

$$w_i = w_M + d_i$$

(4) Let

$$g = \frac{C_0 + \sum_{j=1}^{M} p_{0j}\ w_j}{1 - \sum_{j=1}^{M} p_{0j}\ u_j}$$

$$v_i = u_i\ g + w_i \qquad\qquad i = 1, 2, \ldots, M$$

Algorithm 6.5 has the same advantages as Algorithm 5.1 (which it utilizes) discussed in Section 5.3.1. Briefly, they are:

(1) The crucial variables b_i and d_i in the algorithm are computed recursively such that the state transition probabilities p_{ij} can be computed as needed. This eliminates the need for storing the $\frac{(M + 1)(M + 2)}{2} + M$ elements in the state transition matrix and virtually eliminates any machine storage constraint on the dimensionality of the optimization problem.

(2) The number of arithmetic operations required is also smaller than that of a standard solution method such as Gauss elimination [CRAI 64].

These considerations render the policy-iteration method a very efficient tool in the solution of our optimization problem.

We give below an algorithm (called POLITE) which combines the POLicy-ITEration method, Algorithm 6.5 and Theorem 6.4. Given a Markov decision process model of the channel, POLITE finds the optimal control policy and evaluates the optimum channel performance measures.

Algorithm 6.6 (POLITE)

Given the Markov decision process N^t with

state space $S = \{0, 1, 2, \dots, M\}$,

finite action space A (ICP, RCP or IRCP),

throughput or delay costs $\{C_i(a) \mid i \in S , a \in A \}$,

state transition probabilities $\{p_{ij}(a) \mid i, j \in S , a \in A ,$

$p_{ij}(a) = 0 \text{ if } j \le i - 2\}$,

and stationary policies $f : S \to A$.

To determine a stationary policy f^* such that the cost rate g of N^t is minimized.

Start at either step (1) or step (2).

(1) Given a policy f , apply Algorithm 6.5 to obtain g and $\{v_i\}_{i=1}^M$; $p_{ij}(f)$ and $C_i(f)$ are computed when need in Algorithm 6.5.

(2) Given a set of $\{v_i\}_{i=1}^M$, for state $i = 0, 1, \dots, M$ define the test quantity

$$\text{Cost}(i, a) = C_i(a) + \sum_{j=1}^{M} p_{ij}(a)v_j \tag{6.53}$$

Find $\hat{a}$ such that $\text{Cost}(i, \hat{a}) = \min_{a \in A} \text{Cost}(i, a)$.

If $\text{Cost}(i, f(i)) = \text{Cost}(i, \hat{a})$, then let $\hat{f}(i) = f(i)$; otherwise, let $\hat{f}(i) = \hat{a}$.

(3) If $\hat{f}$ and f are identical, go to step (5).

(4) Replace f by $\hat{f}$ and go to step (1).

(5) $f^* = f$ is an optimal control policy.

(6) $g = g_s(f^*)$ or $g_d(f^*)$ depending on the expected immediate costs $C_i(a)$.

Apply $g_d(f) = \frac{g_s(f)}{\sigma} + M$

(7) The optimum performance measures are,

$$S_{out}^{*} = - g_s(f^{*})$$

$$D^{*} = - \frac{g_d(f^{*})}{g_s(f^{*})} + R + 1$$

6.6 Evaluation of Control Procedures by POLITE

6.6.1 Computational Costs and Convergence

The POLITE algorithm is our tool for computing optimal control policies and evaluating performance measures of the controlled channel using ICP, RCP or IRCP. The algorithm has been coded in Fortran and runs on the IBM 360/91 of the UCLA Campus Computing Network (CCN). For the numerical examples we considered, which will be given in the following sections, the core memory requirement is less than 90K bytes and the job CPU time for each run is between 1 to 6 seconds. (Double precision is used, M is up to 508 and the number of algorithm iterations[*] is in all cases less than 5.) These numbers translate to less than one dollar per run on the average at the current CCN charge rate and are very reasonable considering the size of the problems involved. For comparison, consider the following example. If $M = 400$, the state transition matrix $[p_{ij}]$ alone has $\frac{(401)(402)}{2} + 400 = 81001$ nonzero entries and requires 649K bytes of memory to store it in double precision.

[*] By an iteration of the algorithm POLITE, we mean a complete cycle of steps (1) to (4) in the algorithm.

No conscious effort has been made to optimize the program code except for the following options. First, in step (2) of POLITE, when Cost(i, f(i)) and Cost(i, $\hat{a}$) are compared, they are assumed to be equal if Cost(i, $\hat{a}$) is within $1 \pm \varepsilon$ of Cost(i, f(i)) . In all our numerical computations, ε is taken to be 10^{-5} . Second, to prevent the occurrence of "underflows" during program execution, some threshold must be specified in the program so that whenever a number is less than the threshold it is put equal to zero. For our purposes, the threshold value is taken to be 10^{-30} (instead of the possible 10^{-75} in the IBM 360/91) to save some computations. Smaller threshold values have been used to recompute several cases. No discrepancy in the program output values is observed.

In applying POLITE to solve the ICP and RCP optimization problems, we adopt the following strategy. A control limit policy is always used as the initial control policy to start the algorithm at step (1). This control limit is chosen somewhere between the operating point n_o on the channel load line and the unstable equilibrium point n_c (see Fig. 5-6(b)). Under such an initial control policy, the algorithm requires in most cases between 2 to 4 iterations to arrive at the optimal control policy (algorithm termination).

Although our optimization problem can now be solved by POLITE with relatively small time-space demands on the computer, there exists another constraint which bounds the dimensionality of our problem--the precision of numbers in the computer. When M is large and/or A has many elements, we need to distinguish numbers which are so close together that they are no longer distinguishable given the precision of the computer. Furthermore, increases in the number of recursive steps within the algorithm produce bigger round-off errors, the effect of which is becoming more pronounced. We found that for a value of M larger than 500, the program may not converge* if the initial control policy is not close to the optimal policy. This is (probably) caused by the accumulation of round-off errors as the algorithm requires more iterations for an initial policy which is farther away from the optimal policy.

6.6.2 "Optimality" of the Control Limit Policy

Consider ICP and RCP. The action space A of both control procedures consists of two actions $\{a_o, a_c\}$. a_o is the operating action, designed to give good channel throughput-delay performance conditioning on equilibrium conditions. a_o corresponds to "accept" in ICP and p_o (or K_o) in RCP. a_c is the control action, designed to prevent the channel from going into saturation. a_c corresponds to "reject" in ICP and p_c (or K_c) in RCP.

Our intuition suggests that a good control policy (for either ICP or RCP) must be such that the control action should be applied whenever the channel backlog size N^t exceeds some threshold value to prevent it from drifting toward saturation. But as soon as N^t decreases below this threshold value, the control action should be

*In our computations, each application of POLITE is allowed a maximum of 5 iterations, after which the program stops. Remember that the algorithm is guaranteed to terminate by Theorem 5.2. The difficulty here stems from machine limitations rather than the algorithm itself.

replaced by the operating action, since its use costs the system much more in terms of both channel throughput and packet delay. This intuition has been confirmed in all our numerical computations for ICP and RCP. In each case, the optimal control policy given by POLITE is a control limit policy of the following form.

$$f\ (i) = \begin{cases} a_o & i \le \hat{n} \\ a_c & i > \hat{n} \end{cases} \tag{6.54}$$

where $\hat{n}$ is said to be the control limit (CL) of the control limit policy f .

A rigorous mathematical proof of the optimality of the control limit policy remains an open problem. In many problems characterized by optimal policies of the CL type, the usual method of attack in their proof is to demonstrate monotonicity for the sequences $\{v_i\}$ and $\{\mathrm{Cost}(i,\ a_o)\ -\ \mathrm{Cost}\ (i,\ a_c)\}$. The lack of monotonicy in most such sequences is clearly seen in Figs. 6-7 to 6-10. These figures also serve to illustrate some of the steps of the algorithm POLITE.

An ICP example is shown in Figs. 6-7 and 6-8 where the sequences $\{\mathrm{Cost}(i,\ a)\ -\ \mathrm{Cost}(i,\ r)\}$ and $\{v_i\}$ have been plotted as functions of i . Delay costs corresponding to Eqs. (6.22) and (6.23) are assumed. Each curve in these figures is obtained using the control policy generated during the previous iteration of the algorithm. Consider Fig. 6-7. The initial control policy is a control limit policy with $\hat{n} = 40$ (which interestingly corresponds to the joining point of

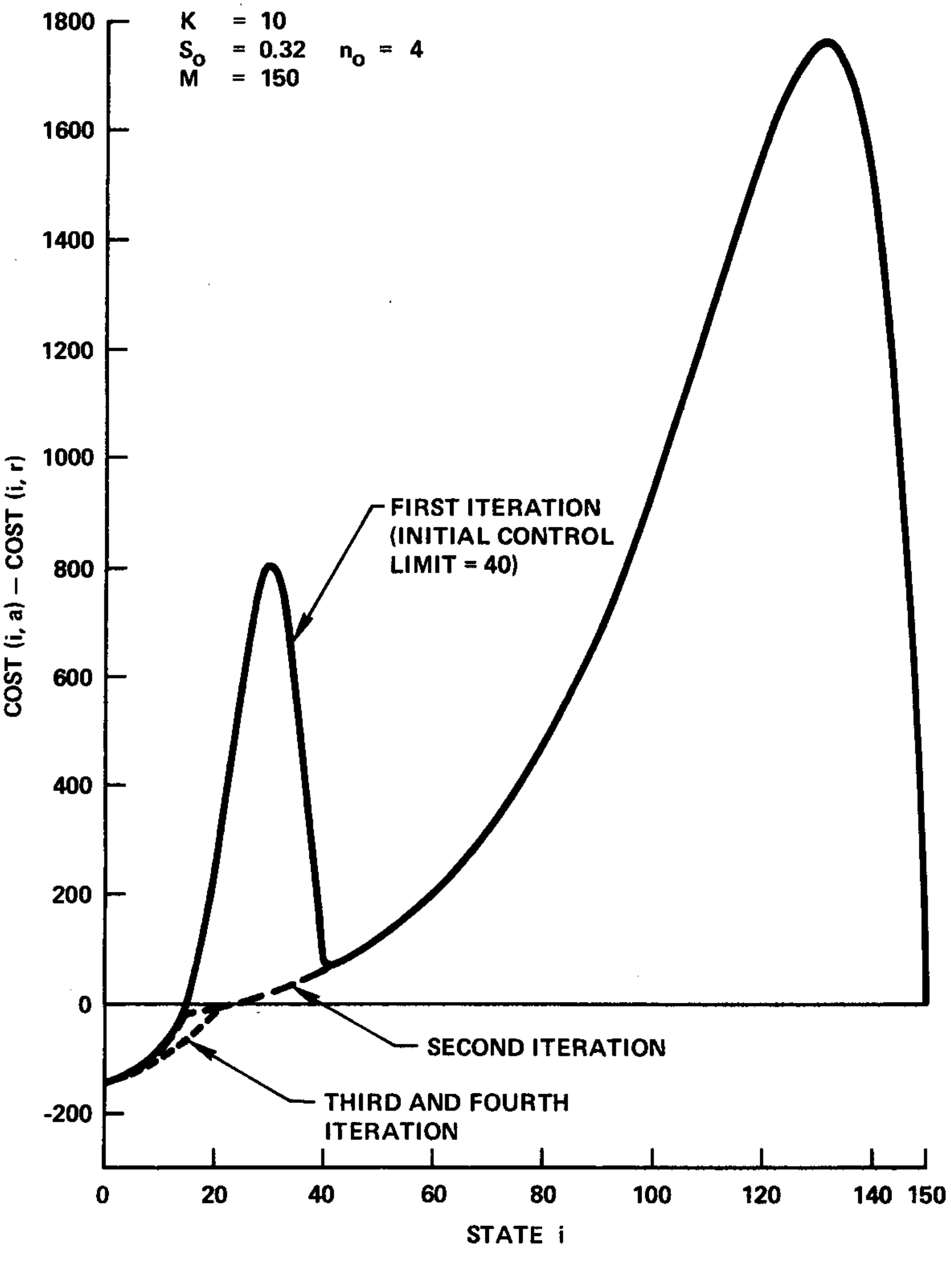


Figure 6-7. POLITE Iterations for ICP With Delay Costs — Control Limits.

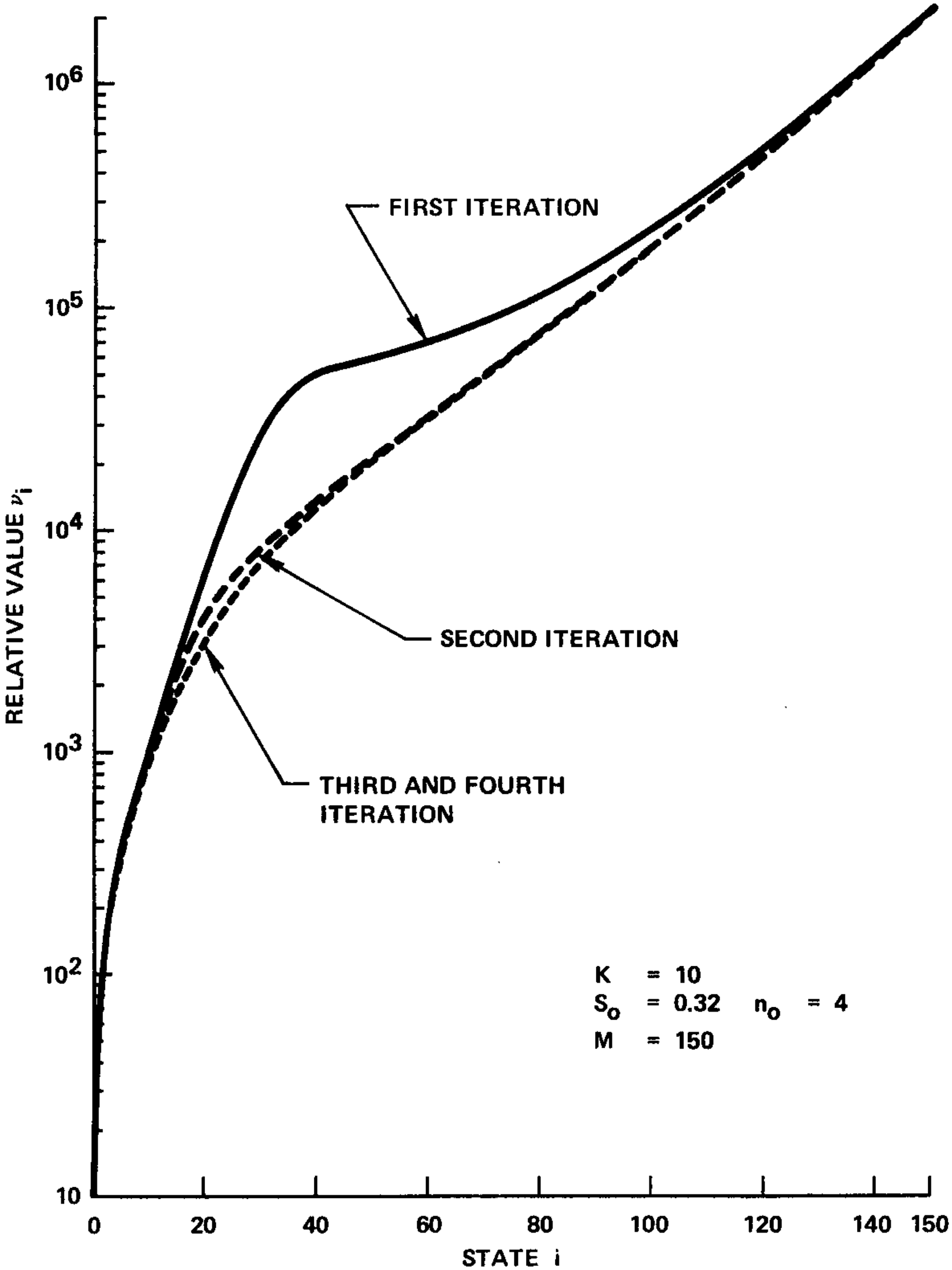


Figure 6-8. POLITE Iterations for ICP With Delay Costs – ν_i

the two humps of the first iteration curve). The first iteration curve crosses zero exactly once between $i = 14$ and $i = 15$. Thus, $\hat{n} = 14$ becomes the control policy for the second iteration. (Recall step (2) in POLITE.) The second iteration curve yields the control policy $\hat{n} = 23$. Finally, the optimal control policy $(\hat{n} = 22)$ is obtained in both the third and fourth iterations and the algorithm terminates. In Fig. 6-8, the relative values v_i in each iteration are shown. We see that v_i is monotonically increasing in i. This implies that the expected total cost in delay(over a finite time horizon) increases as a function of the channel state i at time zero (see Eq. (6.7)).

A RCP example is shown in Figs. 6-9 and 6-10. Throughput costs corresponding to Eqs. (6.36) and (6.37) are assumed (which explains the negative values in Fig. 6-10). Note that the algorithm terminates in only three iterations.

Observe in Figs. 6-7 and 6-9 that when the initial control policy for POLITE is a CL policy, not only is the final optimal policy a CL policy, but all intermediate control policies generated by POLITE are of the control limit type. To test if POLITE generates CL policies only when a CL policy is fed into the algorithm as the initial policy, we tried the following. Let $0 = m_1 < m_2 < \ldots < m_J = M$. Define the control policy

$$f'(i) = \begin{cases} a_o & i = 0 \text{ or } m_j < i \le m_{j+1}\text{ , } j \text{ is odd} \\ a_c & m_j < i \le m_{j+1}\text{ , } j \text{ is even} \end{cases}$$

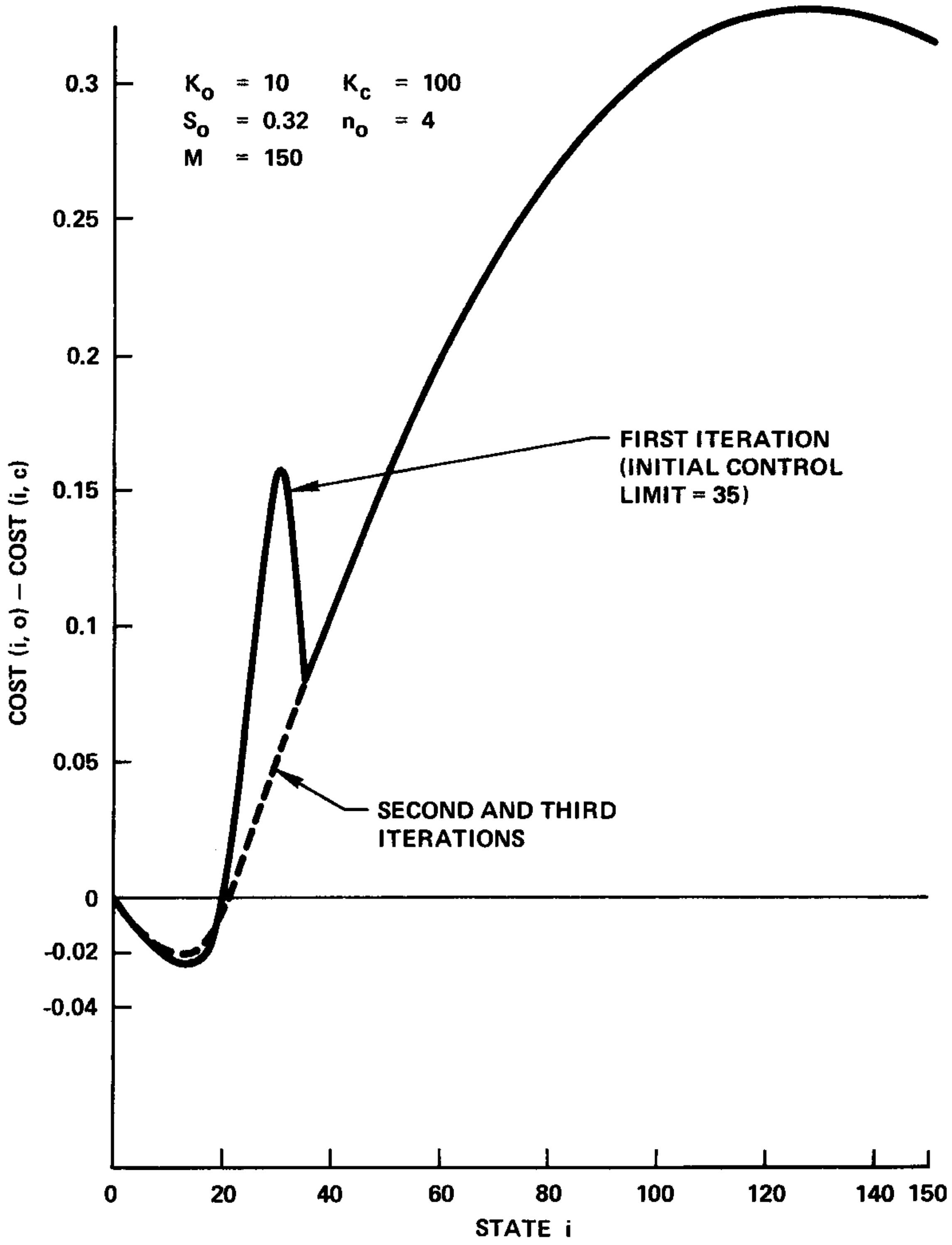


Figure 6-9. POLITE Iterations for RCP With Throughput Costs – Control Limits.

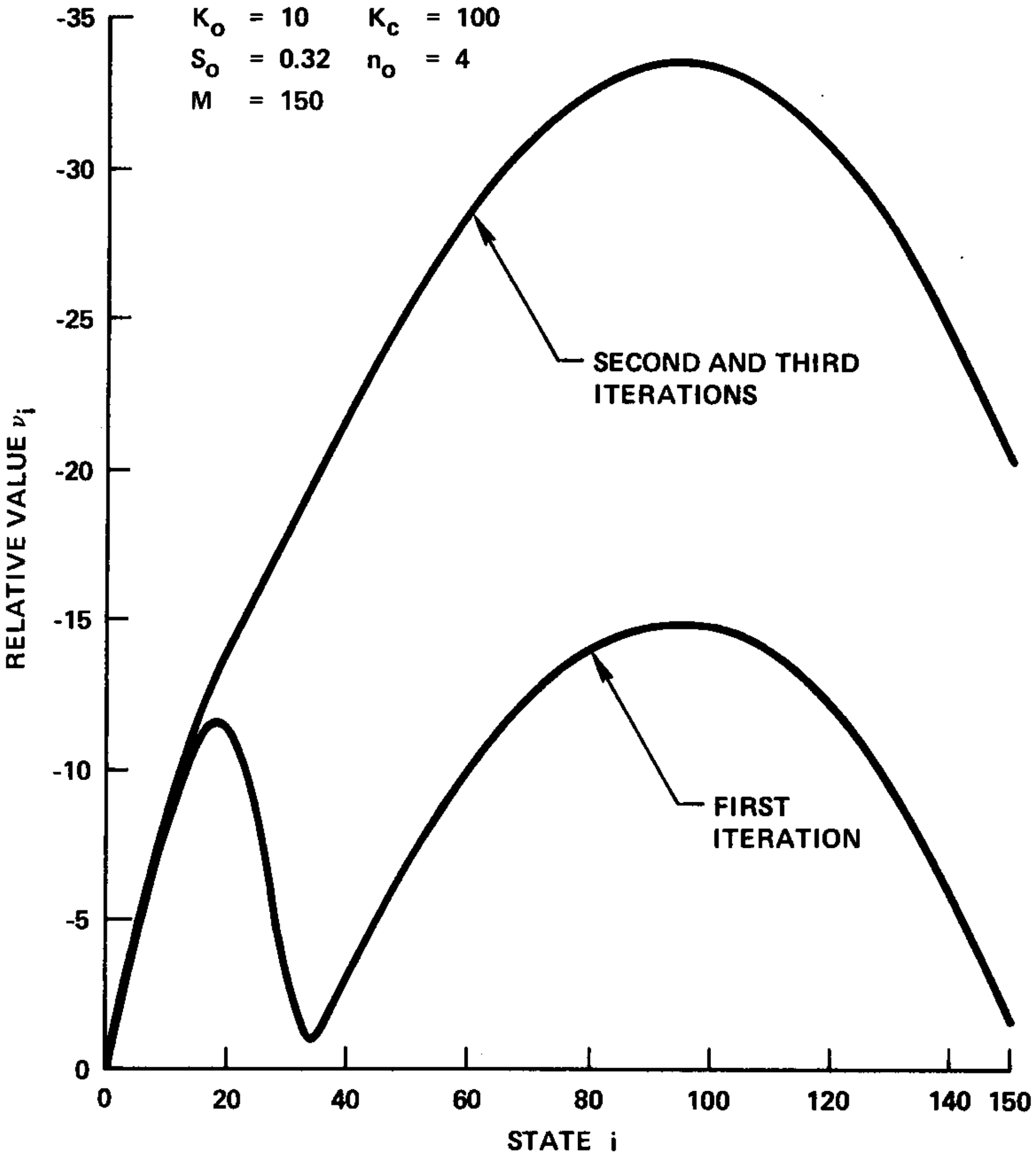


Figure 6-10. POLITE Iterations for RCP With Throughput Costs — ν_i

Such a control policy was used as the initial policy to drive the algorithm POLITE in several cases. In each case, the same CL policy as before was generated by POLITE to be the optimal policy.

6.6.3 Channel Performance

We show in this section throughput-delay performances of the controlled channel using ICP, RCP or IRCP.

Given an unstable channel load line, the throughput-delay performance at the operating point (n_o, S_o) is what we strive to achieve through dynamic channel control. Thus, it is essential that the operating value of K gives an operating point (n_o, S_o) close to the optimum. In Figs. 3-4 to 3-5, we see that $K = 10$ is an excellent choice and will be used throughout this chapter as the operating value of K. The channel load line is a straight line uniquely specified by its intercept on the vertical axis, M, and its slope $-\frac{1}{\sigma}$. However, often we would prefer to specify the load line by specifying M and the operating point (n_o, S_o) on the equilibrium contour (instead of σ). Thus, different load lines specified by the same channel operating point can be compared by showing how well they approach the throughput-delay performance at the operating point.

The equilibrium contour corresponding to $K = 10$ is shown in Fig. 5-3. Each channel load line to be used in our computations will be specified by M and one other point on the (n,S) plane. The points shown in Table 6.1 will be used.

n_o =	1	2	3	4	5	7	10
S_o =	0.2	0.25	0.3	0.32	0.34	0.36	0.374

Table 6.1 Points on the $K = 10$ contour.

(Note that these points only approximate points on the $K = 10$ contour. For example, given $S_o = 0.32$, n_o given by the $K = 10$ contour is actually between 3 and 4, but has been rounded off to 4 for convenience.) In particular, the points $(n_o, S_o) = (4, 0.32)$ and $(7, 0.36)$ will be used in most of our examples. Assuming a large M, these points correspond to a channel which is moderately to very heavily "loaded" when the problems of channel instability and channel control become significant.

From our discussion in the last section, all control policies considered below for ICP and RCP are of the CL type.

In ICP the control action is to reject all new packet arrivals. In RCP the control action is to use a large enough value of $K = K_c$ which renders the channel load line stable. We illustrate this last statement in Fig. 6-11. The average packet delay D given by an optimal RCP control policy is shown as a function of K_c. Note that K_c somewhat less than the necessary value of K to render the channel load line stable can be used. However, if K_c is too small, the channel performance "blows up" since now the controlled channel is still unstable. Observe that for a sufficiently large K_c, D is quite insensitive to its exact value except when $S_o = 0.36$, in which case D increases slowly with K_c. Note that for the same S_o, a much larger K_c is

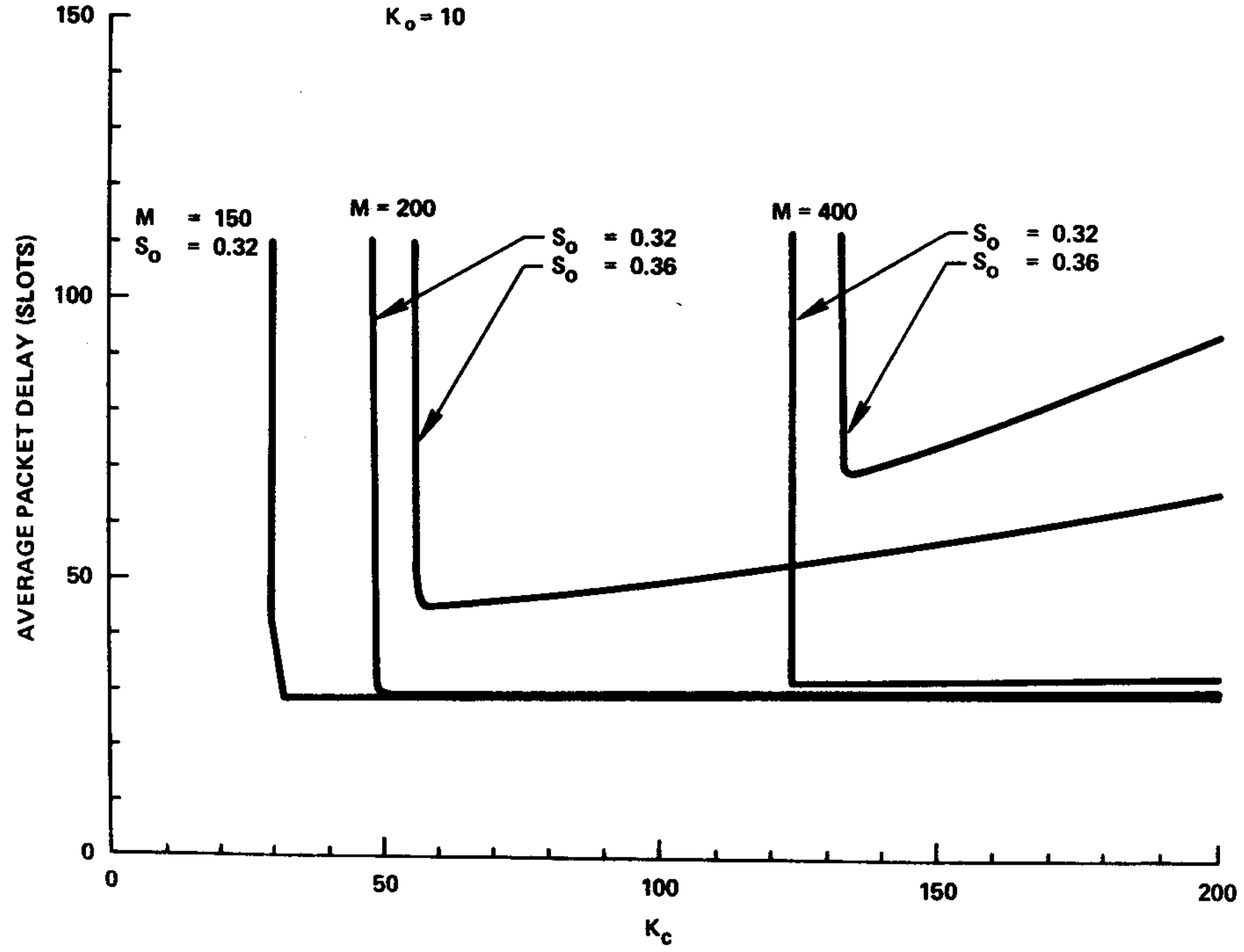


Figure 6.11. RCP Channel Performance Versus K_c.

required for a larger M. In the limit as $M \to \infty$, RCP becomes ineffective since no sufficiently large value of K can be used for K_c.

Knowing that the optimal control policy is a CL policy, we show in Figs. 6-12 to 6-15, the channel performance measures S_{out} and D (given by ICP and RCP for $M = 200{,}400$ and $S_o = 0.32, 0.36$) over a range of control limits. Observe that the same control limit minimizes D and maximizes S_{out} at the same time as predicted by Theorem 6.4. Note the amazing flatness of S_{out} and D near the optimum point, especially when $S_o = 0.32$ and $M = 200$ in Figs. 6-12 and 6-13. The consequence is that even if a nonoptimal control policy is used (due to, for example, not knowing the exact current backlog size such as in most practical systems), it is still possible to achieve a throughput-delay performance close to the optimum. However, such flatness of S_{out} and D is not as pronounced when S_o is 0.36. Comparing the four figures, we see that the optimum values of S_{out} and D given by ICP and RCP are approximately the same, but RCP gives less severe degradation in channel performance with control limits much smaller and much larger than the optimal. However, recall from Fig. 6-11 the potential disastrous channel behavior if K_c is not sufficiently large. This must be taken into consideration in any system design using RCP since in a practical system both the parameters M and σ may change with time. To provide the necessary design safety margin, a much bigger value of K_c than deemed necessary may have to be adopted. In Fig. 6-13, we show the degradation in channel performance when $K_c = 200$ is used instead of $K_c = 60$. (The use of $K_c = 200$ allows the channel to support more than 400 users instead of 200.) On the other hand, M

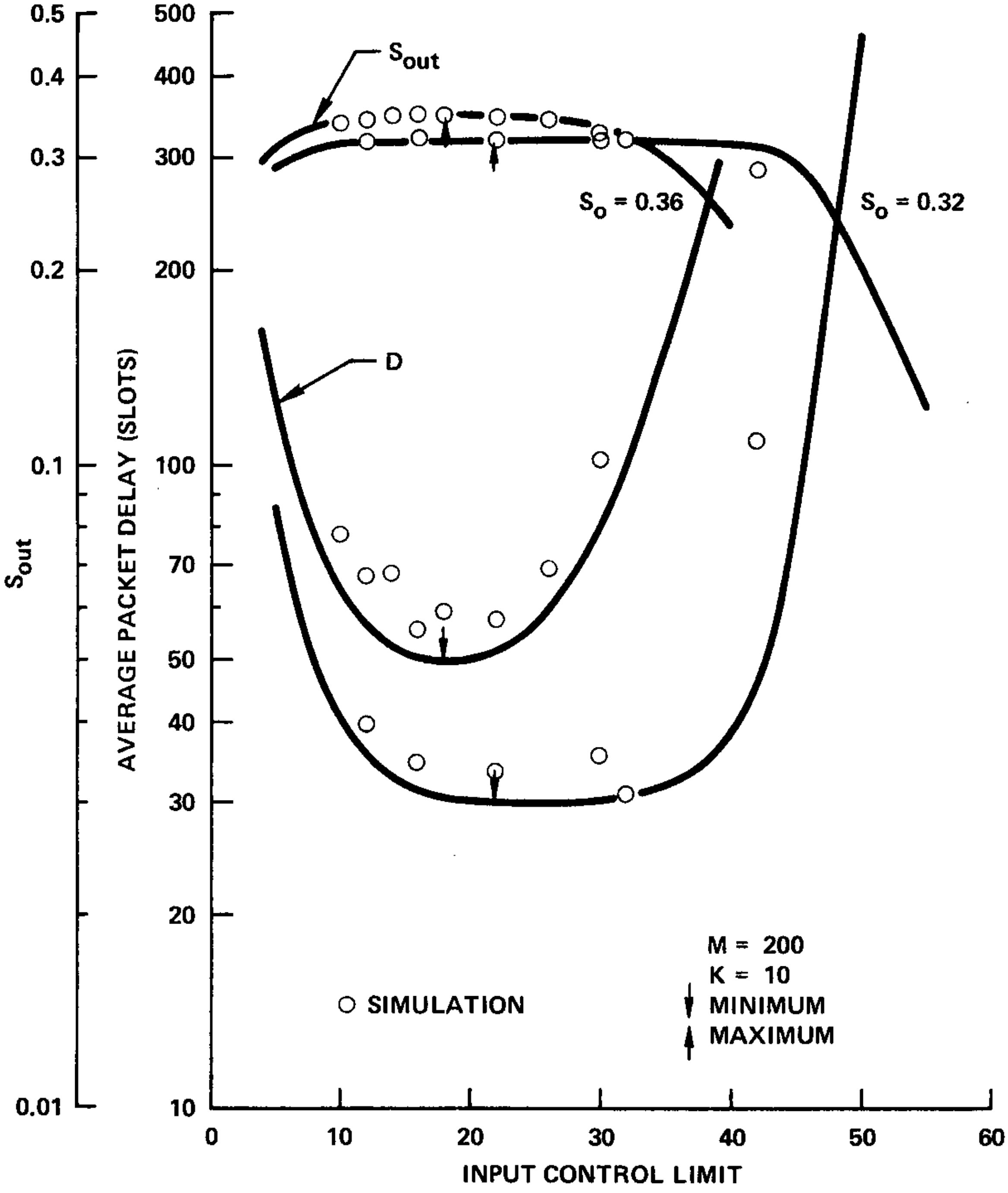


Figure 6-12. Channel Performance Versus ICP Control Limit for M = 200.

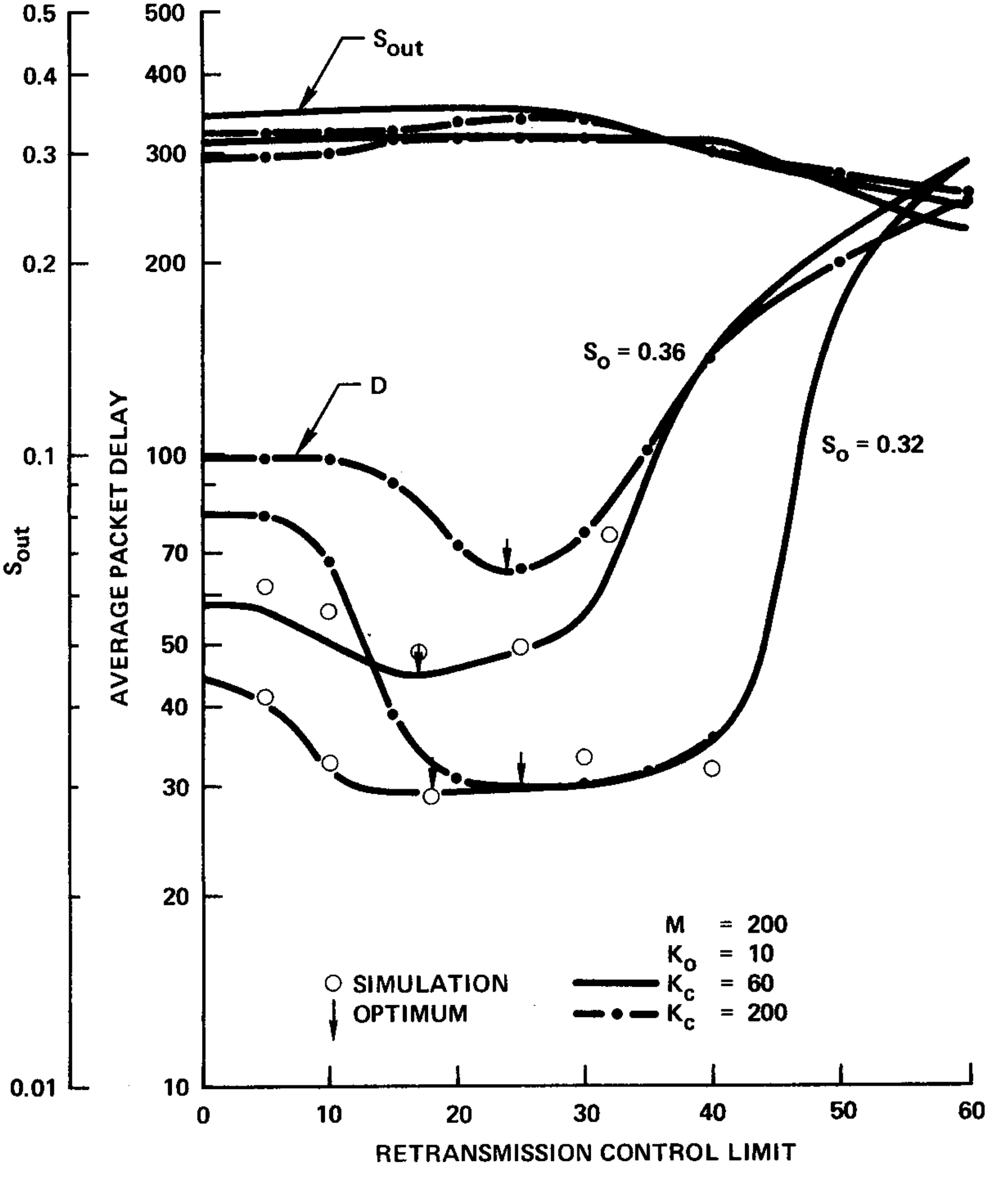


Figure 6-13. Channel Performance Versus RCP Control Limit for M = 200.

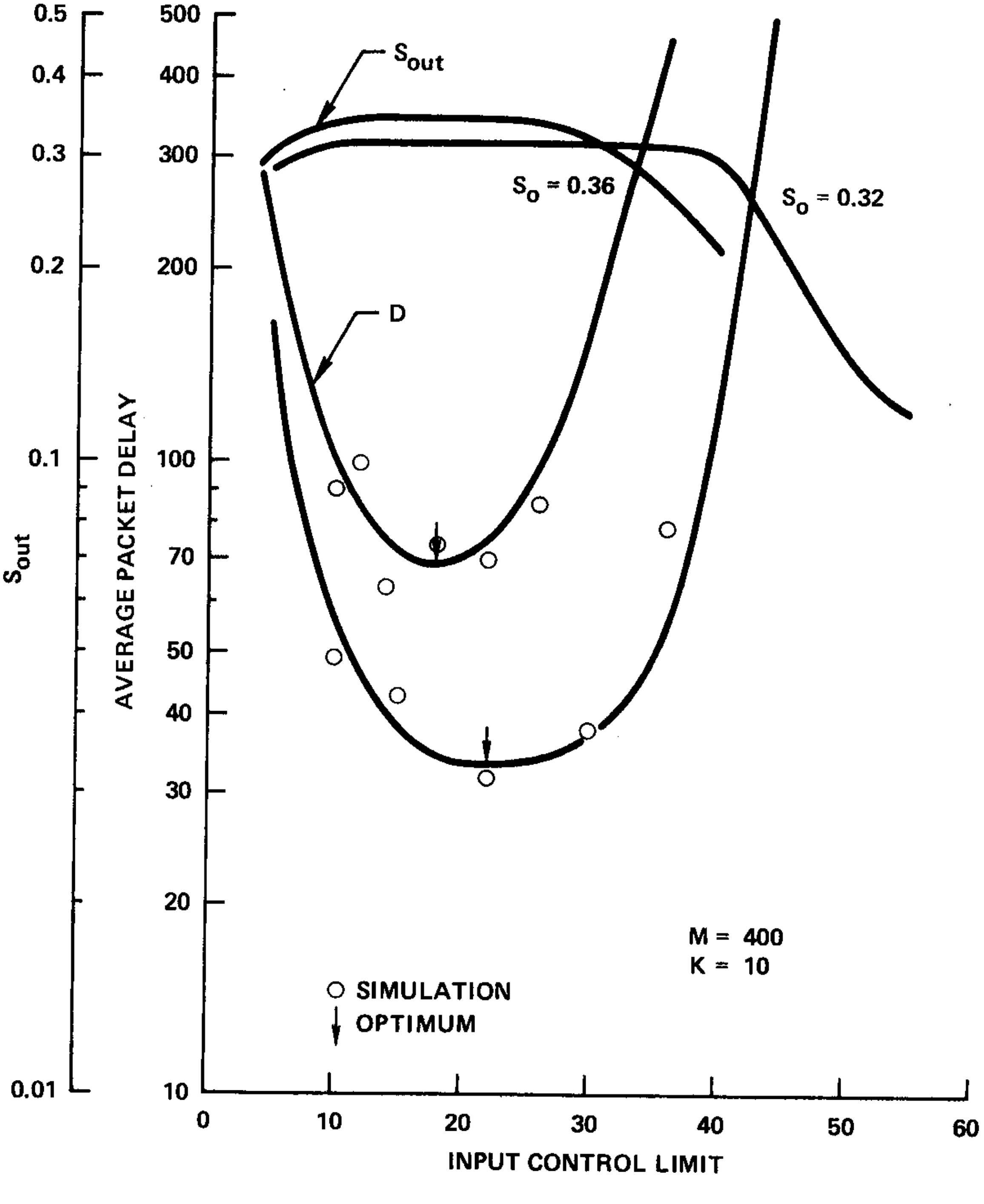


Figure 6-14. Channel Performance Versus ICP Control Limit for M = 400.

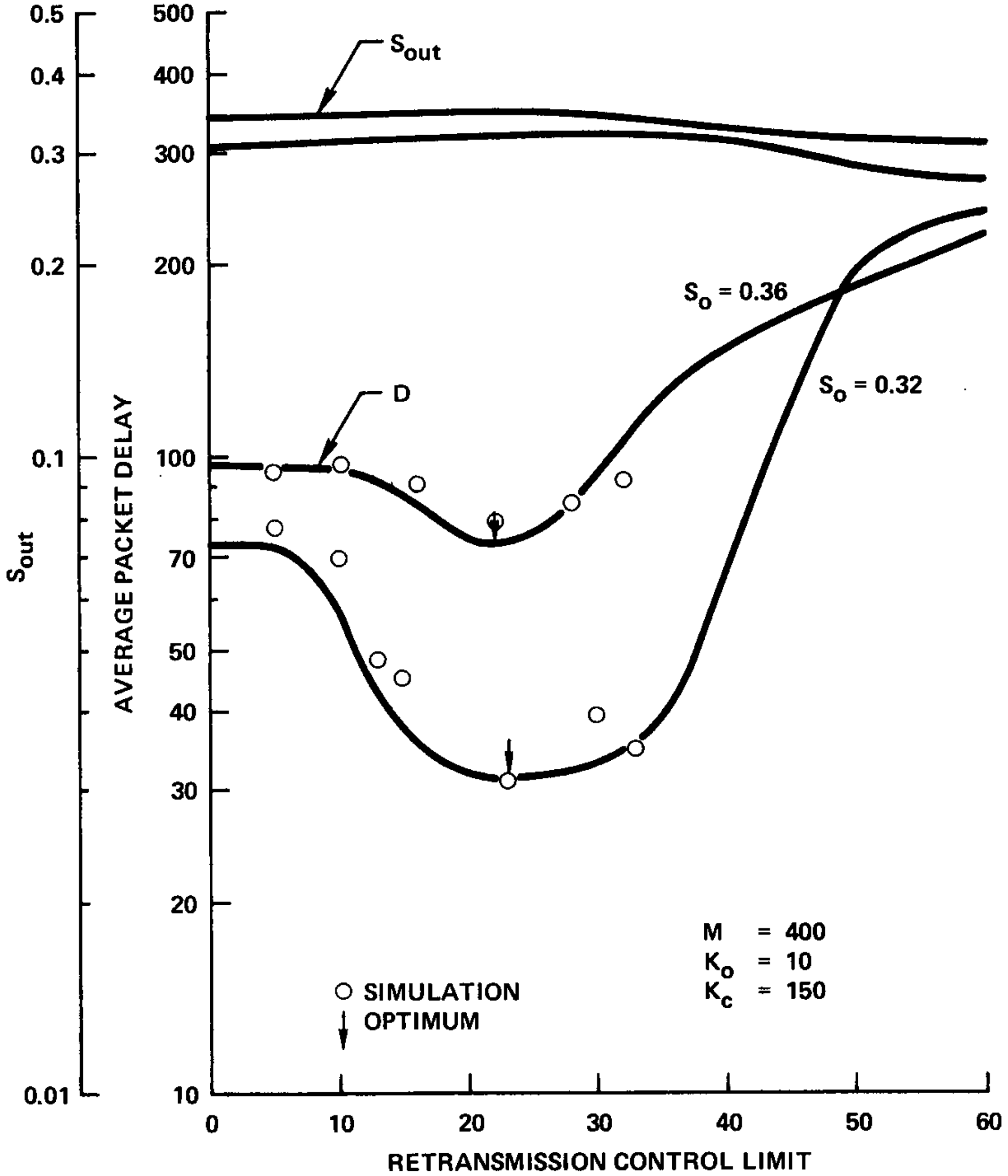


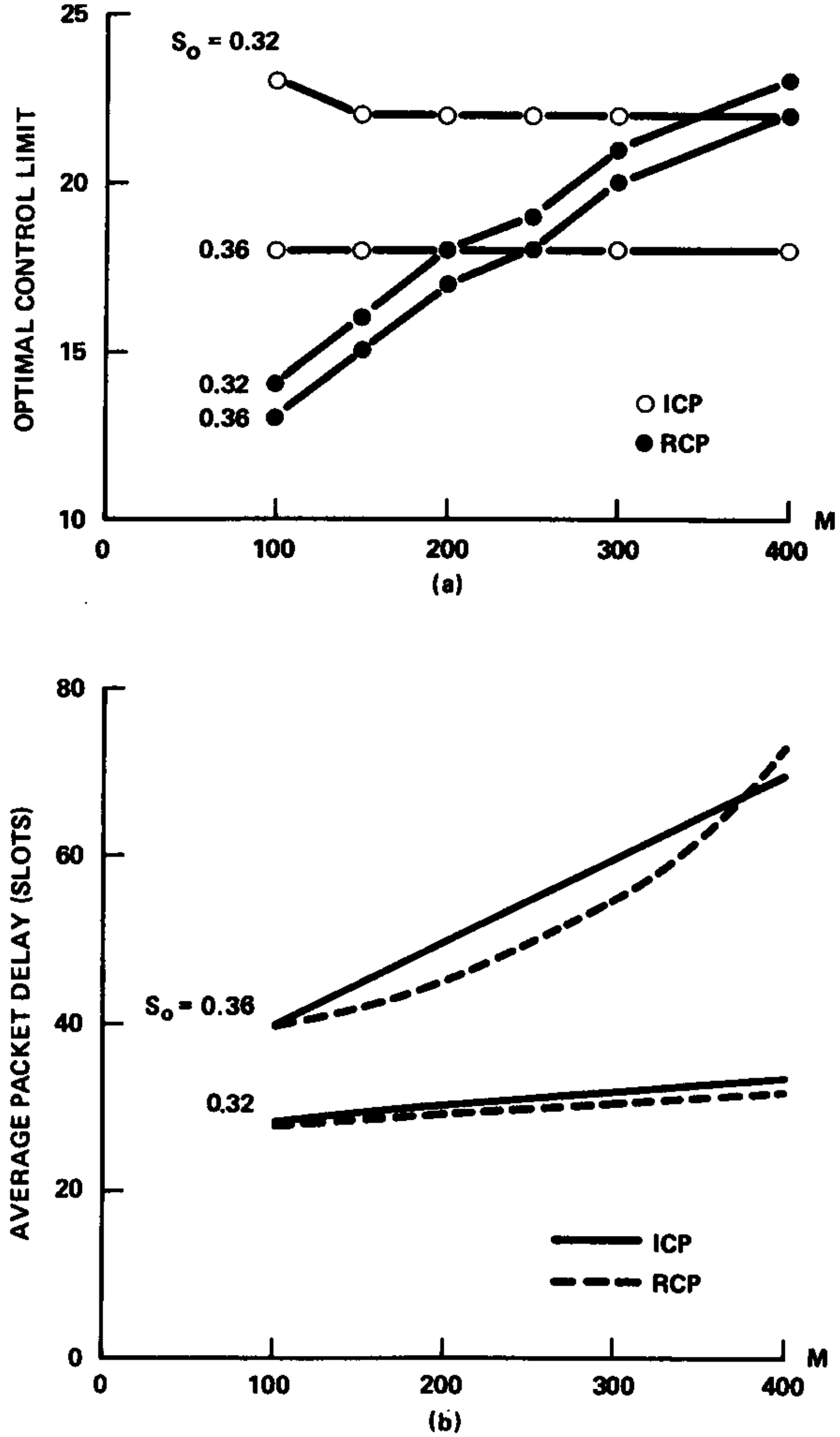
Figure 6-15. Channel Performance Versus RCP Control Limit for M = 400.

has relatively little effect on the optimal ICP control limit as shown in Fig. 6-16(a). Thus, even if M fluctuates in time in a real system, the same ICP control limit policy is still optimal. Of course, the optimum channel performance must deteriorate as M increases as shown in Fig. 6-16(b). We see also in Fig. 6-16(a) that in the case of RCP, as M (and hence, K_c)* increases, the optimal RCP control limit increases. In Fig. 6-16(b), the optimum D given by ICP and RCP are compared. RCP is found to be slightly better than ICP. However, as M becomes large, K_c must also be large, in which case the trend indicates that ICP is superior to RCP.

We mentioned earlier that for a value of M larger than 500, we run into difficulties with round-off errors such that using POLITE, the optimal control policy can be found only when it is close to the initial control policy. We see here that for a very large M, ICP is superior to RCP. The ICP optimal control limit is also insensitive to M and thus, the same control limit may be used even when M becomes very large.

In Figs. 6-12 to 6-15, we have also indicated simulation results for throughput and delay. Throughput results are shown in Fig. 6-12 only and omitted in the other three figures (but they agree as well with the analytic results as shown in Fig. 6-12). In these simulations, channel control policies are applied assuming that the exact channel backlog size N^t is known to all channel

*For both $S_o = 0.32, 0.36$ and corresponding to $M = 100, 150, 200, 250, 300$ and 400, we let $K_c = 20, 40, 60, 80, 100, 150$ respectively.



(a)

(b)

Figure 6-16. ICP and RCP Channel Performance Versus M.

users. However, contrary to the mathematical model, each collided packet is assumed to suffer a fixed delay R and its retransmission to be randomized uniformly over the next K slots. The mathematical model is idealized since R is assumed to be zero while each backlogged packet retransmits in a time slot with probability $p = \frac{1}{R+(K+1)/2}$. (In both cases, the average retransmission delay is the same.) This approximation was examined in Section 5.1 for an uncontrolled channel and found to be very good under the assumption of channel equilibrium. The excellent agreement between the simulation and analytic results presented here demonstrates that this approximation is good even for a dynamically controlled channel. The duration of each simulation run was taken to be 30,000 time slots. The reason for using such a long duration is that in those cases when the control limit $\hat{n}$ is large or when S_o is relatively small, such as 0.32, N^t may exceed $\hat{n}$ only once in a long time. If such time periods are large compared to the duration of a run, the simulation results will not be accurate since we are trying to determine the average value of a random quantity using only a small number of samples.

Optimum throughput-delay tradeoffs

Given a channel control procedure, we consider here the optimum throughput-delay tradeoff corresponding to the boundary of the infeasible region in Fig. 6-6. In Fig. 6-17, given $M = 400$ and a fixed σ , we see that S_{out} is maximized and D minimized by the optimal control limit $\hat{n} = 22$. With a fixed M , the optimum throughput-delay tradeoff curve is obtained by increasing σ

from zero and for each value of σ , finding the optimal CL and evaluating the optimum channel performance through application of POLITE. Such optimum throughput-delay tradeoffs at fixed values of M are shown in Figs. 6-17 and 6-18 for ICP and RCP respectively. Also shown in these figures is the optimum performance envelope of the infinite population model given in Chapter 3. Note how close the ICP and RCP throughput-delay tradeoff curves are to the optimum envelope. In fact, the $M = 50$ tradeoff curve lies a little below the optimum envelope. This is to be expected since $M = 50$ actually gives rise to a stable channel, in which case the channel performance at the operating point is achieved. Note that these two curves are obtained from two different analytic models based upon different approximations, namely, the first order approximation model in Chapter 3 and the linear feedback model in Chapter 5. It is comforting to see that the two different approximations lead to such close results.

In Figs. 6-19 and 6-20, we show optimum throughput-delay tradeoffs at fixed values of σ for ICP and RCP respectively. ($\frac{1}{\sigma}$ is the average think time of a channel user.) In this case, increasing S_{out} corresponds to increasing M , that is, admitting more channel users. We see that the channel performance improves as the packet generation probability σ increases, since this implies that for the same S_{out} , the number of channel users M is smaller. We considered average think times of 10-30 seconds (see Section 5.1.2). User populations with smaller average think times will probably give

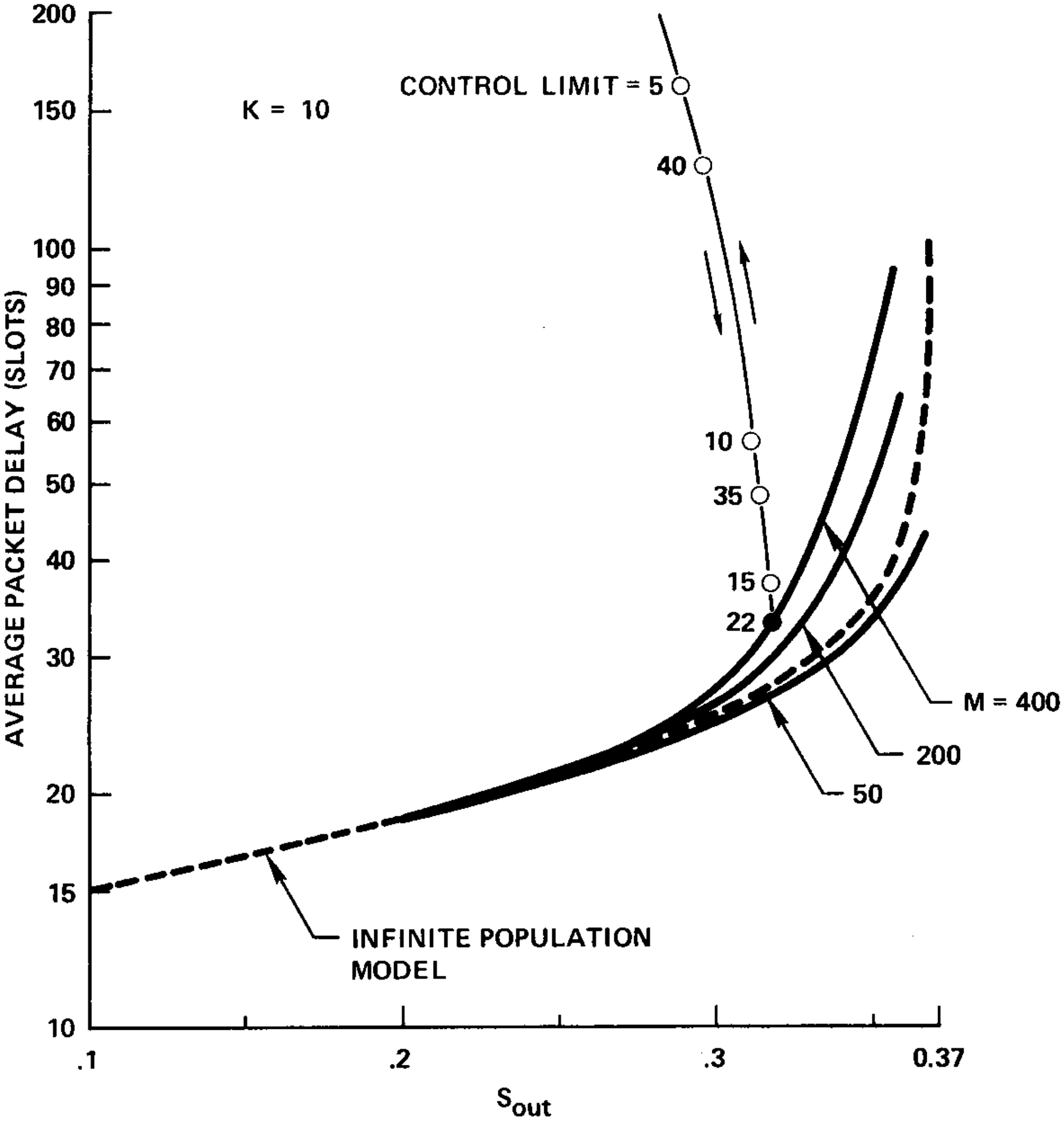


Figure 6-17. ICP Optimum Throughput-Delay Tradeoffs at Fixed M.

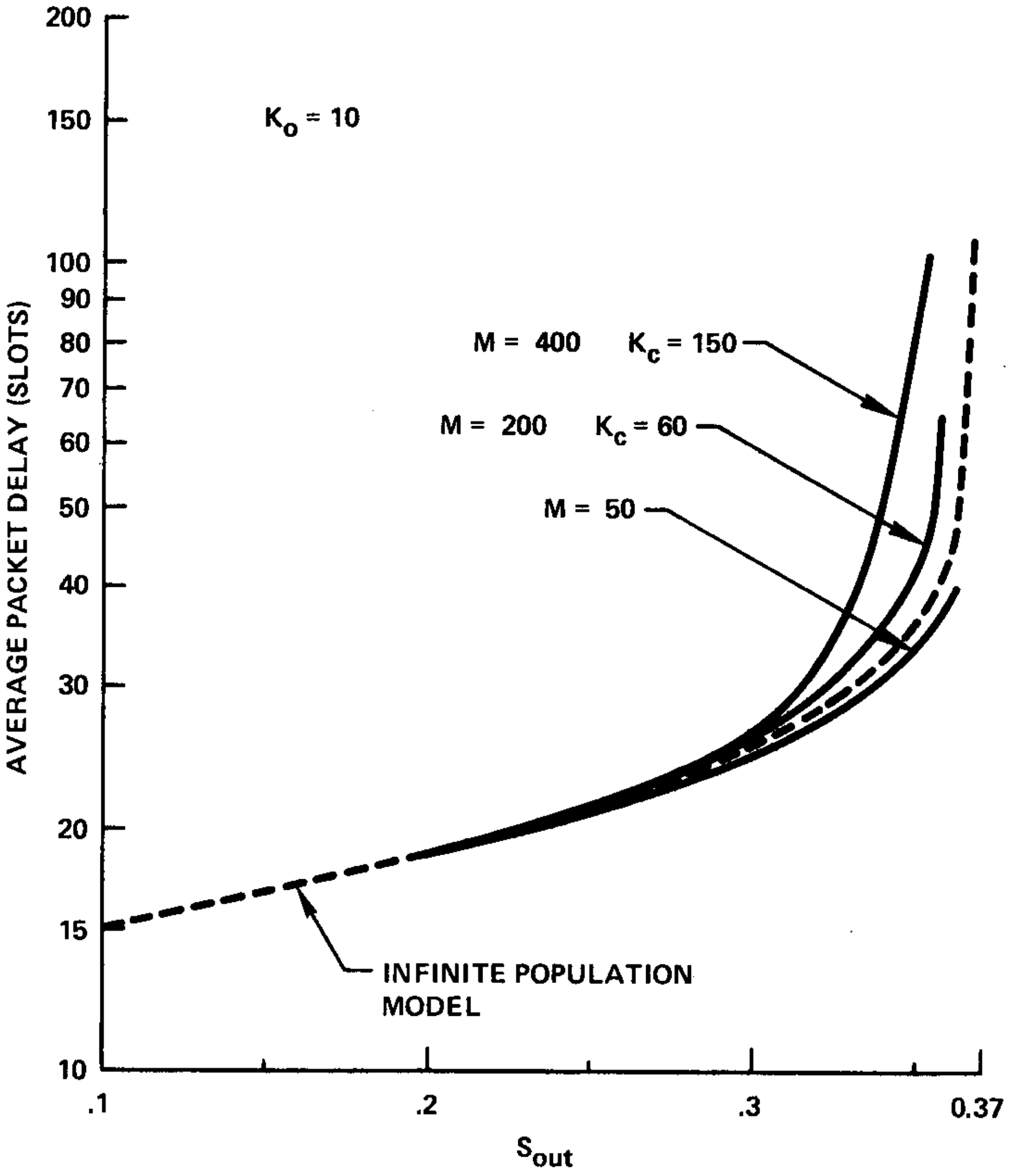


Figure 6-18. RCP Optimum Throughput-Delay Tradeoffs at Fixed M.

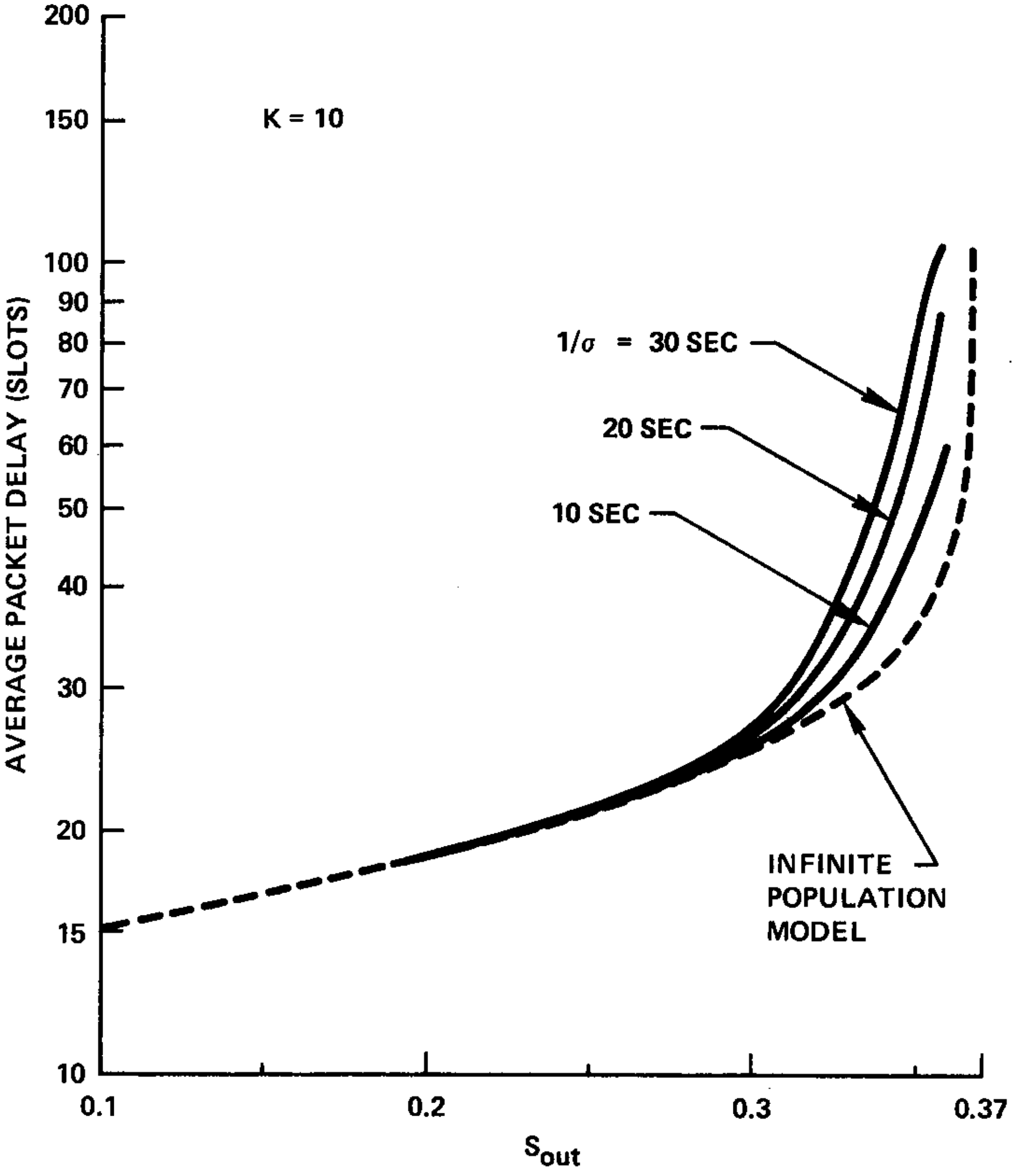


Figure 6-19. ICP Optimum Throughput-Delay Tradeoffs at Fixed σ.

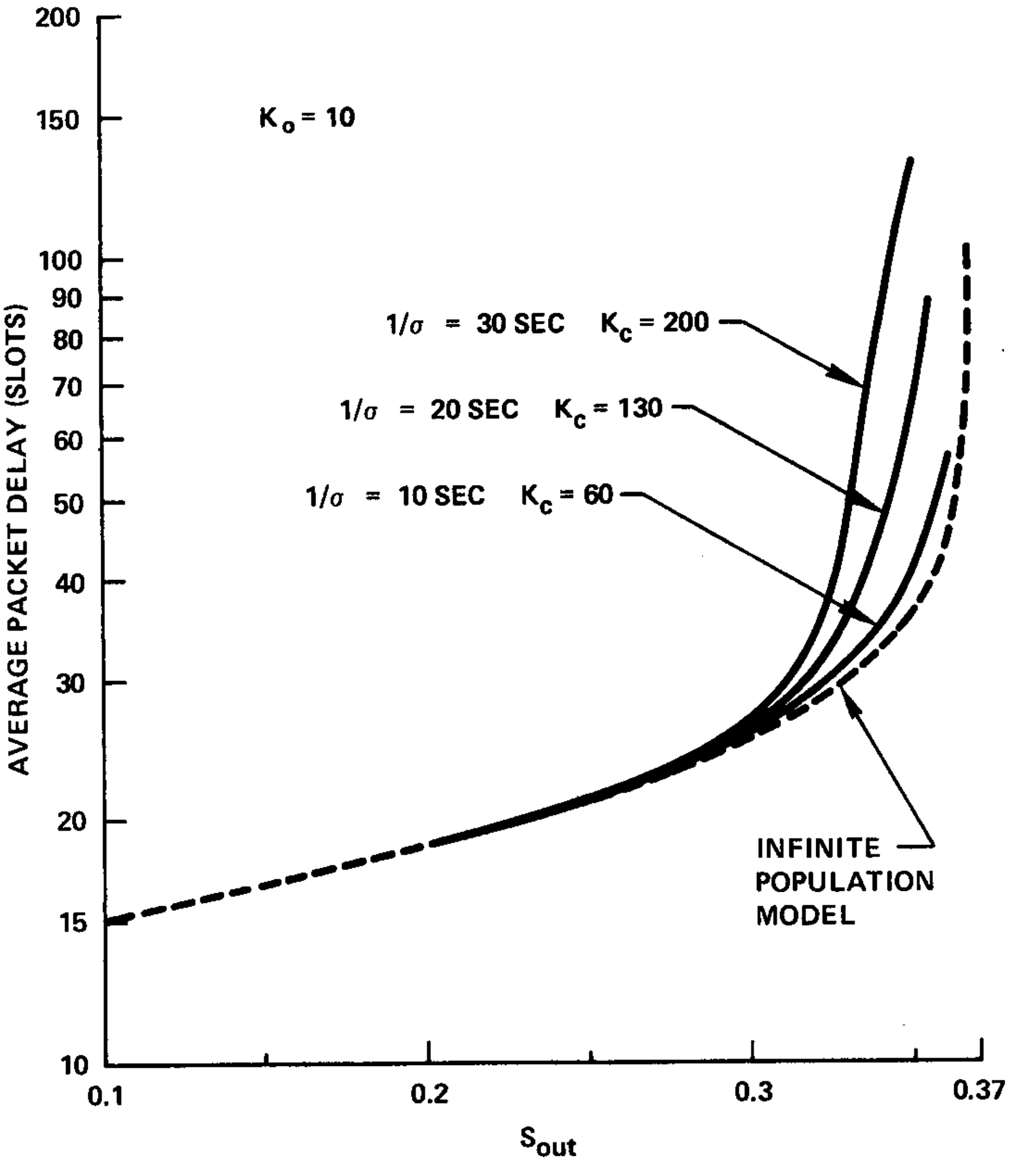


Figure 6-20. RCP Optimum Throughput-Delay Tradeoffs at Fixed σ.

rise to a stable channel, since M must be smaller in this case. Tradeoff curves for larger average think times can be generated by the algorithm POLITE if necessary.

Comparing ICP and RCP in the last four figures, we see that they give rise to almost the same throughput-delay tradeoffs. RCP is slightly better than ICP except when M or $\frac{1}{\sigma}$ is large (e.g., $M > 400$ or $\frac{1}{\sigma} = 30$ seconds).

IRCP channel performance

Recall that the ICP and RCP action spaces are both subspaces of the IRCP action space. Therefore, the channel performance given by IRCP must be better or at least as good as that given by ICP or RCP. This has been verified in all cases we considered. However, in each case, the differences in S_{out} and D among these three channel control procedures are small as shown in Table 6.2 for the four cases involving M = 200, 400 and $(n_o, s_o) = (4, 0.32), (7, 0.36)$. Observe that in every instance, IRCP gives the best performance, but only by a very slim margin. Note also that the optimal policy for IRCP is of the form

$$f(i) = \begin{cases} ao & 0 \le i \le \hat{n}_1 \\ ac & \hat{n}_1 < i \le \hat{n}_2 \\ rc & \hat{n}_2 < i \end{cases} \tag{6.55}$$

which is uniquely specified by $(\hat{n}_1, \hat{n}_2)$. This is similar to a "concatenation" of RCP and ICP control limits! In fact, $\hat{n}_1$ is either equal or very close to the optimal RCP control limit in each case and

		$M = 200$ $S_o = 0.32$ $(K_c = 60)$	$M = 200$ $S_o = 0.36$ $(K_c = 60)$	$M = 400$ $S_o = 0.32$ $(K_c = 150)$	$M = 400$ $S_o = 0.36$ $(K_c = 150)$
$\hat{n}$	ICP	22	18	22	18
$\hat{n}$	RCP	18	17	23	22
$(\hat{n}_1, \hat{n}_2)$	IRCP	(18, 56)	(17, 43)	(23, 116)	(23, 91)
S_{out}	ICP	0.31778	0.34925	0.31807	0.34846
	RCP	0.31817	0.35217	0.31844	0.34715
	IRCP	0.31817	0.35219	0.31844	0.34847
D	ICP	29.857	49.552	33.096	69.237
	RCP	29.085	44.802	31.608	73.588
	IRCP	29.085	44.772	31.608	69.215

Table 6.2 Comparison of ICP, RCP and IRCP.

the use of $\hat{n}_2$ brings about only minor improvement in the channel performance except in the case of $M = 400$ and $S_o = 0.36$. We shall also refer to $\hat{n}_1$ and $\hat{n}_2$ as control limits.

6.7 Practical Control Schemes

The optimal throughput-delay channel performance given in the last section is achievable over an infinite time horizon if the channel users have exact knowledge of the channel state at any time. In a practical system, the channel users often have no means of communication among themselves other than the multi-access broadcast channel itself. Each channel user must individually estimate the channel state by observing the outcome in each channel slot. Moreover, whatever channel state information available to the channel users is at least one round-trip propagation delay old and may introduce additional errors in the users' estimates if R is large (such as in a satellite channel). Thus, the control action applied based upon an estimate of the channel state may not necessarily be the optimal one at that time, which then will lead to some degradation in channel performance.

Below we first give a procedure for estimating the channel state assuming that the history (i.e., empty slots, successful transmissions or collisions) of the channel is available to all channel users. The optimal ICP, RCP and IRCP control policies will be applied based upon the above estimate. A heuristic control procedure is then proposed which circumvents the state estimation problem. These control procedures are examined through simulations and compared with the optimal throughput-delay results in the previous section. The ability of these control procedures to handle time-varying inputs (with pulses) is also examined. Two other control procedures will then be discussed and some channel design considerations given.

6.7.1 Channel Control-Estimation Algorithms (CONTEST)

Our heuristic procedure for estimating the channel state is based upon the observation that the channel traffic in a time slot is approximately Poisson distributed (see Chapter 4 and Appendix A). Below we present algorithms which implement channel control procedures studied in the previous sections using the estimated channel state. These channel CONTrol-ESTimation algorithms will be referred to as CONTEST algorithms.

CONTEST algorithms

We give here a procedure for implementing RCP. As before, we let K_o be the operating value and K_c be the control value of K. Suppose $\hat{n}$ is the RCP control limit such that the channel users switch their retransmission K value from K_o to K_c when the channel backlog size exceeds $\hat{n}$ and from K_c to K_o as soon as the channel backlog size drops below $\hat{n}$. We let

$$\hat{G}_o = \hat{n}\, p_o + (M - \hat{n})\sigma \tag{6.56}$$

where from Eq. (5.3)

$$p_o = \frac{1}{R + (K_o + 1)/2}$$

We also define

$$\hat{G}_c = \hat{n}\, p_c + (M - \hat{n})\sigma \tag{6.57}$$

where

$$p_c = \frac{1}{R + (K_c + 1)/2}$$

$\hat{G}_o$ and $\hat{G}_c$ are thus the average channel traffic rates given that the channel backlog size is $\hat{n}$ packets with K equal to K_o and K_c

respectively. Assuming that the channel traffic is approximately Poisson distributed, we define the following critical values (corresponding to the probability of zero channel traffic in a time slot),

$$\hat{f}_o = e^{-\hat{G}_o} \tag{6.58}$$

and

$$\hat{f}_c = e^{-\hat{G}_c} \tag{6.59}$$

Since $K_c > K_o$ we must have

$$\hat{f}_o < \hat{f}_c$$

Suppose each channel user keeps track of the channel history (one round-trip propagation delay ago) within a window frame of W slots as shown in Fig. 6-21. Let $\bar{f}^t$ be the fraction of empty slots in the W slots within the history window for the t^{th} time slot. $\bar{f}^t$ will closely approximate the probability of zero channel traffic

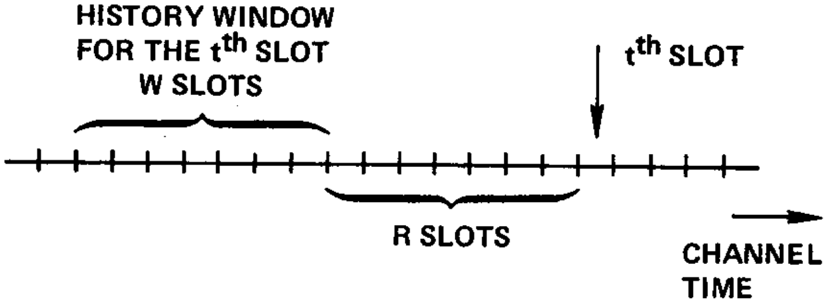


Figure 6-21. The Channel History Window at Time t.

in the t^{th} time slot provided that the channel traffic probability distribution does not change appreciably in $(W + R)$ time slots and the Poisson traffic assumption holds. We give the following CONTEST algorithm to be adopted by each channel user. Let d^t be the control decision at time t .

Algorithm 6.7 (RCP-CONTEST)

This algorithm generates the decision $d^t = K_o, K_c$ at each time point based upon the channel state estimate $\overline{f}^t$ and the RCP control limit $\hat{n}$. Start at step (1) or step (4).

```
(1)    $t \leftarrow t + 1$
       $d^t = K_o$
(2)    If $\overline{f}^t < \hat{f}_o$ , go to (4)
(3)    Go to (1)
(4)    $t \leftarrow t + 1$
       $d^t = K_c$
(5)    If $\overline{f}^t > \hat{f}_c$ , go to (1)
(6)    Go to (4)
```

Next we consider a similar implementation for ICP. In ICP, the control actions are {accept, reject} . Suppose $\hat{n}$ is the ICP control limit such that the channel always rejects new packet arrivals when the current backlog size exceeds $\hat{n}$ and always accepts new packets when the current backlog size is less than or equal to $\hat{n}$. We let

$$\hat{G}_a = \hat{n}\, p + (M - \hat{n})\sigma \qquad (6.60)$$

and

$$\hat{G}_r = \hat{n}\ p \tag{6.61}$$

where

$$p = \frac{1}{R + (K + 1)/2}$$

$\hat{G}_a$ and $\hat{G}_r$ are the average channel traffic rates given that the channel backlog size is $\hat{n}$ packets with the current decision = accept, reject respectively. Again assuming a Poisson channel traffic, we define the following critical values (corresponding to the probability of zero channel traffic in a time slot),

$$\hat{f}_a = e^{-\hat{G}_a} \tag{6.62}$$

$$\hat{f}_r = e^{-\hat{G}_r} \tag{6.63}$$

Since $\hat{G}_a > \hat{G}_r$, we must have

$$\hat{f}_a < \hat{f}_r$$

Algorithm 6.8 (ICP-CONTEST)

This algorithm generates the decision d^t = accept, reject at time t , based upon the channel state estimate $\bar{f}^t$ and ICP control limit $\hat{n}$. Start at step (1) or step (4).

```
(1)    $t \leftarrow t + 1$
       $d^t$ = accept
```

```
(2)   If $\overline{f}^t < \hat{f}_a$ go to (4)
(3)   Go to (1)
(4)   $t \leftarrow t + 1$
      $d^t$ = reject
(5)   If $\overline{f}^t > \hat{f}_r$ go to (1)
(6)   Go to (4)
```

To implement IRCP, we assume that the control policy is of the form given in Eq. (6.55) such that it is uniquely specified by the control limits $\hat{n}_1$ and $\hat{n}_2$. To be consistent with this assumption, we shall distinguish only three decision states: ao, ac and rc. We define $\hat{f}_o$ and $\hat{f}_c$ by using $\hat{n}_1$ in Eqs. (6.56)-(6.59), $\hat{f}_{ac}$ and $\hat{f}_{rc}$ by using $\hat{n}_2$ and p_c in Eqs. (6.60)-(6.63), and $\hat{f}_{ao}$ by using $\hat{n}_2$ and p_o in Eqs. (6.60) and (6.62). Since $p_o > p_c > \sigma$ and $\hat{n}_2 > \hat{n}_1$, we have $\hat{f}_{ao} < \hat{f}_o$ and $\hat{f}_{ac} < \hat{f}_c$.

Algorithm 6.9 (IRCP-CONTEST)

This algorithm generates the decision d^t = ao, ac, rc at time t based upon the channel state estimate $\overline{f}^t$ and IRCP control policy $(\hat{n}_1, \hat{n}_2)$. Start at step (1), (4), or (7).

```
(1)   $t \leftarrow t + 1$
      $d^t$ = ao
(2)   If $\overline{f}^t < \hat{f}_{ao}$ go to (7)
      otherwise, if $\overline{f}^t < \hat{f}_o$ go to (4)
(3)   go to (1)
(4)   $t \leftarrow t + 1$
      $d^t$ = ac
```

```
(5)    If $\overline{f}^t > \hat{f}_c$ go to (1)
       otherwise, if $\overline{f}^t < \hat{f}_{ac}$ go to (7)
(6)    go to (4)
(7)    $t \leftarrow t + 1$
       $d^t = rc$
(8)    If $\overline{f}^t > \hat{f}_{rc}$ go to (4)
(9)    go to (7)
```

The channel history window

The size W of the channel history window kept by each channel user is very important for successful channel state estimation. If W is too large, we may lose information on the dynamic behavior of the channel such that the necessary actions are taken belatedly. If W is too small, we may get large errors in approximating the probability of zero channel traffic by the fraction of empty slots in the history window. A good initial estimate is that W should be bigger than R and of the same order of magnitude. Below we compare simulation results on channel performance for different values of W.

To implement the channel state estimation procedure, each channel user needs to maintain the channel history for W slots. Since it is only necessary to record whether or not a slot is empty, W bits of information suffice. A possible implementation is depicted schematically in Fig. 6-22. The bit string stored in the shift register represents the channel history in a window of W slots. An empty channel slot is represented by '1' while a nonempty channel slot is represented by '0'. In the figure, the circle represents a summer, the triangle an attenuator and the square a unit delay of one slot.

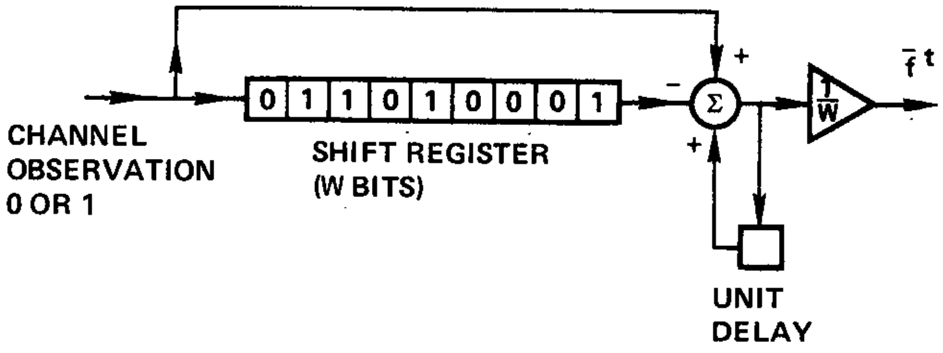


Figure 6-22. Determination of $\bar{f}^t$

Simulation results on the channel performance given by the CONTEST algorithms will be examined below in Section 6.7.3.

6.7.2 Another Retransmission Control Procedure

We describe in this section a simple heuristic control procedure which has the property that when the channel traffic increases the retransmission delays of backlogged packets will also increase. Hence, it will be referred to as the heuristic retransmission control procedure (Heuristic RCP). The advantage of such a control procedure is that it is simple and can be implemented easily without any need for monitoring the channel history and estimating the channel state. In the next section, this and the above CONTEST algorithms will be compared through simulations.

The Control Scheme

For a backlogged packet with m previous channel collisions, the uniform retransmission randomization[*] interval is taken to be $K = K_m$ where K_m is a monotone nondecreasing function in m.

When the channel traffic increases, the probability of channel collision increases. As a result, the "effective" value of K increases. If K_m is a steep enough function in m, we see that channel saturation will be prevented. An effective value of K can be defined only with respect to a specific performance measure (e.g., average packet delay). To illustrate the effect of the function K_m, we derive below the average value of K as a function of q (the probability of successful transmission). Let

$$\begin{aligned} r_i &= \text{Prob[a packet retransmits } i \text{ times before success]} \\ &= (1-q)^i\, q \qquad i \geq 1 \end{aligned}$$

Case 1 $\qquad K_m = K_2$ for $m \geq 2$ and $K_2 > K_1$

$$\begin{aligned} \overline{K} &= \text{average value of } K \\ &= \frac{1}{1-q} \sum_{i=1}^{\infty} r_i \sum_{m=1}^{i} \frac{K_m}{i} \end{aligned}$$

[*] Note that the same control scheme can be extended to geometric retransmission randomization by letting $p = p_m$ where p_m is a monotone nonincreasing function.

$$= \frac{1}{1 - q} \sum_{i=1}^{\infty} (1 - q)^i \, q \left(\frac{K_1}{i} + \frac{i - 1}{i} K_2 \right)$$

$$= K_2 + \frac{q \, \ell n \, q}{1 - q} (K_2 - K_1) \qquad (6.64)$$

which is equal to K_1 at $q = 1$ and increases to K_2 as q decreases to zero; ℓn is the natural logarithm function.

Case 2 $\qquad K_m = K_3$ for $m \geq 3$ and $K_3 > K_2 > K_1$

$$\overline{K} = \frac{1}{1 - q} \sum_{i=1}^{\infty} r_i \sum_{m=1}^{i} \frac{K_m}{i}$$

$$= \frac{1}{1 - q} \sum_{i=2}^{\infty} (1 - q)^i \, q \left(\frac{K_1}{i} + \frac{K_2}{i} + \frac{i - 2}{i} K_3 \right)$$

$$+ (1 - q) \, q \, K_1$$

$$= K_3 + (K_3 - K_2) \, q + \frac{q \, \ell n \, q}{1 - q} (2K_3 - K_1 - K_2) \qquad (6.65)$$

which is equal to K_1 at $q = 1$ and increases to K_3 as q decreases to zero.

Case 3 $\qquad K_m = m \, K \qquad m \geq 1$

$$\overline{K} = \frac{1}{1 - q} \sum_{i=1}^{\infty} r_i \sum_{m=1}^{i} \frac{K_m}{i}$$

$$= \frac{K}{1 - q} \sum_{i=1}^{\infty} \frac{(1 - q)^i q}{i} \sum_{m=1}^{i} m$$

$$= \frac{K}{2} (1 + \frac{1}{q}) \tag{6.66}$$

which is equal to K at $q = 1$ and increases to infinity as q decreases to zero.

The above results indicate that the average value of K behaves in the desired manner, namely, $\overline{K}$ increases as q decreases due to an increasing channel traffic. This behavior is similar to that of the retransmission control procedure. That is why the above procedure is called Heuristic RCP. Below we examine the CONTEST algorithms and Heuristic RCP through simulations.

6.7.3 Simulation Results

We summarize in Tables 6.3-6.6, throughput-delay results for the following channel load lines,

(1) $M = 200$, $(n_o, S_o) = (4, 0.32)$

(2) $M = 400$, $(n_o, S_o) = (4, 0.32)$

(3) $M = 200$, $(n_o, S_o) = (7, 0.36)$

(4) $M = 400$, $(n_o, S_o) = (7, 0.36)$

In all cases, K_o is equal to 10. K_c is taken to be 60 and 150 for M equal to 200 and 400 respectively. Included in these tables are (a) optimum POLITE results for ICP, RCP and IRCP, (b) simulation results for ICP and RCP using optimal control policies and under the assumption of perfect channel state information, (c) simulation results for the CONTEST algorithms using ICP and RCP optimal control policies, and (d) simulation results for Heuristic RCP. Each simulation run is identified by the seed supplied to the random number generator. The duration of each simulation run was taken to be 30,000 time slots. IRCP was not tested by simulation since the optimal value of $\hat{n}_2$ is in all cases so large that within the simulation duration, the channel state N^t (almost surely) will not exceed it; the control procedure becomes effectively RCP specified by $\hat{n}_1$.

The ICP-CONTEST algorithm was tested with channel history window sizes of 20, 40, 60 and 80 time slots. We see from Tables 6.3-6.6 that $W = 40$ appears to give the best throughput-delay results. Note that for $R = 12$ and $K = K_o = 10$, $W = 40$ is approximately twice $R + K$.

The RCP-CONTEST algorithm was also tested with various values of W . In this case, K takes on two values, K_o and K_c where $K_c = 60$ or 150 depending on M . There is no clear-cut optimal W . It appears that $W = 60$ is a good choice for $K_c = 60$ and $M = 200$ while $W = 80$ is a good choice for $K_c = 150$ and $M = 400$.

Results for $S_o = 0.32$ and $M = 200, 400$ are shown in Tables 6.3 and 6.4. We see that there is no significant degradation in channel performance (from the optimum) given by the CONTEST algorithms and Heuristic RCP. The CONTEST algorithms, however, seem to have an edge over Heuristic RCP. The excellent performance of the CONTEST algorithms can be attributed to the flatness of S_{out} and D near the optimum as a function of the control limit (see Figs.

CONTROL SCHEME	RANDOM NUMBER GENERATOR SEED IN SIMULATION	S_{out}	D
ICP	-----	0.31778	29.857
RCP	-----	0.31817	29.085
IRCP	-----	0.31817	29.085
ICP	39474	0.315	33.427
RCP	78453	0.318	28.824
ICP-CONTEST W = 20	73645	0.314	40.893
" " W = 40	39587	0.315	30.514
" " W = 60	59478	0.317	32.355
" " W = 80	54857	0.318	35.809
RCP-CONTEST W = 20	49784	0.315	33.052
" " W = 40	58474	0.322	33.335
" " W = 60	20494	0.319	32.138
" " W = 80	10398	0.317	32.501
Heuristic RCP $\left\{\begin{array}{l} K_1 = 10 \\ K_m = 60 \quad m\geq 2 \end{array}\right.$	18867	0.316	33.720
	61111	0.315	34.554
Heuristic RCP $\left\{\begin{array}{l} K_1 = 10 \\ K_2 = 60 \\ K_m = 120 \quad m\geq 3 \end{array}\right.$	63037	0.310	35.425
	07275	0.316	34.635

Table 6.3 Throughput-delay results of a controlled channel ($M = 200$, $S_o = 0.32$)

CONTROL SCHEME	RANDOM NUMBER GENERATOR SEED IN SIMULATION	S_{out}	D
ICP	-----	0.31807	33.096
RCP	-----	0.31844	31.608
IRCP	-----	0.31844	31.608
ICP	84023	0.315	31.427
RCP	40393	0.317	31.023
ICP-CONTEST W = 20	94875	0.315	43.262
" " W = 40	39848	0.314	34.723
" " W = 60	74945	0.312	53.240
" " W = 80	94875	0.316	39.112
RCP-CONTEST W = 20	49784	0.313	41.087
" " W = 40	58474	0.319	43.379
" " W = 60	20494	0.318	38.821
" " W = 80	10398	0.317	40.068
" " W = 100	64945	0.314	35.689
" " W = 120	18494	0.319	47.149
Heuristic RCP $\left\{\begin{array}{ll} K_1 = 10 & \\ & m\geq 2 \\ K_m = 150 & \end{array}\right.$	57298	0.316	45.150
	16489	0.316	44.750
Heuristic RCP $\left\{\begin{array}{ll} K_1 = 10 & \\ K_2 = 100 & \\ K_m = 200 & m\geq 3 \end{array}\right.$	38687	0.312	42.040
	46534	0.311	43.136

Table 6.4 Throughput-delay results of a controlled channel (M = 400, S_o = 0.32)

CONTROL SCHEME	RANDOM NUMBER GENERATOR SEED IN SIMULATION	S_{out}	D
ICP	-----	0.34925	49.552
RCP	-----	0.35217	44.802
IRCP	-----	0.35219	44.772
ICP	18654	0.346	59.111
RCP	95646	0.348	48.655
ICP-CONTEST W = 20	18947	0.331	83.664
" " W = 40	53857	0.339	77.357
" " W = 60	89574	0.330	87.614
" " W = 80	10394	0.332	73.310
RCP-CONTEST W = 20	03847	0.347	67.900
" " W = 40	39575	0.345	50.853
" " W = 60	60389	0.345	50.534
" " W = 80	10489	0.347	51.787
Heuristic RCP $\begin{cases} K_1 = 10 & \\ K_m = 60 & m\geq 2 \end{cases}$	94854	0.349	48.535
	37776	0.344	46.116
Heuristic RCP $\begin{cases} K_1 = 10 & \\ K_2 = 60 & \\ K_m = 120 & m\geq 3 \end{cases}$	94854	0.350	50.267
	18495	0.347	54.583

Table 6.5 Throughput-delay results of a controlled channel (M = 200, S_o = 0.36)

CONTROL SCHEME	RANDOM NUMBER GENERATOR SEED IN SIMULATION	S_{out}	D
ICP	-----	0.34846	69.237
RCP	-----	0.34715	73.588
IRCP	-----	0.34847	69.215
ICP	28879	0.343	73.524
RCP	44217	0.350	79.270
ICP-CONTEST W = 20	38457	0.334	128.460
" " W = 40	06348	0.330	98.994
" " W = 60	74948	0.336	126.143
" " W = 80	74394	0.332	119.628
RCP-CONTEST W = 20	38457	0.341	99.701
" " W = 40	06348	0.335	97.676
" " W = 60	74948	0.343	97.048
" " W = 80	74394	0.340	91.833
" " W = 100	38373	0.343	107.722
" " W = 120	93875	0.337	99.192
Heuristic RCP $\left\{\begin{array}{l} K_1 = 10 \\ K_m = 150 \quad m \geq 2 \end{array}\right.$	99581	0.344	66.327
	54857	0.352	70.590
Heuristic RCP $\left\{\begin{array}{l} K_1 = 10 \\ K_2 = 100 \\ K_m = 200 \quad m \geq 3 \end{array}\right.$	38378	0.345	81.324
	36949	0.348	62.662

Table 6.6 Throughput-delay results of a controlled channel ($M = 400$, $S_o = 0.36$)

6-12 to 6-15). This property is typical of channel load lines specified by small to moderate values of S_o . We see in the same figures that the flatness in S_{out} and D near the optimum is not as pronounced when $S_o = 0.36$. This explains the more significant degradation in channel performance given by the CONTEST algorithms shown in Tables 6.5-6.6. Note that for $S_o = 0.36$, Heuristic RCP gives much better throughput-delay results than the CONTEST algorithms.

In Figs. 4-2 and 4-4, it was shown that in an uncontrolled channel, a channel input rate of 0.8 packet/slot sustained for 100 time slots was enough to cripple the channel. In Figs. 6-23 and 6-24, we show by simulations that under similar but more severe circumstances both the IRCP-CONTEST algorithm and Heuristic RCP prevented the channel from going into saturation. In these simulations, the normal channel load line was given by $M = 400$ and $(n_o, S_o) = (4, 0.32)$ both before and after the pulse. During a period of 200 slots (namely, the time period 1000-1200), the packet generation probability σ was increased such that $M\sigma = 1.0$ packet/slot. Observe that both algorithms handled the sudden influx of new packets with ease. In both cases, the channel throughput, instead of vanishing to zero as in an uncontrolled channel, maintained at a high rate and within less than 3000 slots, the channel returned to almost normal operation.

6.7.4 Other Proposed Schemes

Other channel control procedures have been proposed by Metcalfe [METC 73A] and Rettberg [RETT 73C].

In Metcalfe's proposal, Q is defined to be the current number of channel users who have a packet ready to transmit over the channel.

INPUT PARAMETERS:

NUMBER OF TERMINALS M = 400 , PROPAGATION DELAY R = 12
FOR THE TIME PERIOD 1-1000, INPUT RATE $M\sigma$ = 0.3232
FOR THE TIME PERIOD 1001-1200, INPUT RATE $M\sigma$ = 1.0
FOR THE TIME PERIOD 1201-6000, INPUT RATE $M\sigma$ = 0.3232
RETRANSMISSION CONTROL LIMIT = 23, INPUT CONTROL LIMIT = 116
K_o = 10 , K_c = 150 , WINDOW SIZE W = 60

RANDOM NUMBER GENERATOR SEED = 81724

AVERAGE VALUES IN 200 TIME SLOT PERIODS:

```
  TIME PERIOD      THROUGHPUT    TRAFFIC      PACKET     FRACTION    AVERAGE   PACKETS
                    RATE-  S     RATE- G     DELAY- D     EMPTY      BACKLOG   REJECTED

     1 -  200        0.290        0.625        30.29       0.555        5.5        0
   201 -  400        0.325        0.700        34.06       0.505        6.9        0
   401 -  600        0.235        0.450        23.67       0.635        2.8        0
   601 -  800        0.295        0.625        31.76       0.555        5.9        0
   801 - 1000        0.325        0.850        42.57       0.455        9.3        0
  1001 - 1200        0.205        2.345       524.32       0.190       50.0       49
  1201 - 1400        0.345        1.330       389.68       0.325       75.2       13
  1401 - 1600        0.355        0.880       188.11       0.435       51.5        0
  1601 - 1800        0.375        0.735       179.37       0.460       34.9        0
  1801 - 2000        0.225        1.295       297.85       0.383       33.7        5
  2001 - 2200        0.325        1.005       530.77       0.415       35.3       21
  2201 - 2400        0.380        0.905       127.32       0.365       16.7        0
  2401 - 2600        0.305        0.485        27.64       0.605        3.1        0
  2601 - 2800        0.290        0.430        20.36       0.640        2.3        0
  2801 - 3000        0.345        0.745        35.38       0.485        7.2        0
  3001 - 3200        0.300        0.455        17.62       0.535        2.4        0
  3201 - 3400        0.280        0.615        28.48       0.575        5.6        0
  3401 - 3600        0.390        0.810        37.90       0.425        7.9        0
  3601 - 3800        0.330        0.655        30.65       0.520        5.6        0
  3801 - 4000        0.300        0.390        19.30       0.655        1.7        0
  4001 - 4200        0.315        0.615        29.24       0.560        5.1        0
  4201 - 4400        0.335        0.600        24.51       0.545        4.5        0
  4401 - 4600        0.300        0.450        24.32       0.630        2.6        0
  4601 - 4800        0.280        0.480        25.29       0.625        3.7        0
  4801 - 5000        0.285        0.585        32.07       0.580        5.3        0
  5001 - 5200        0.330        0.570        26.41       0.555        4.3        0
  5201 - 5400        0.335        0.550        23.67       0.560        3.7        0
  5401 - 5600        0.335        0.640        28.81       0.530        5.2        0
  5601 - 5800        0.275        0.410        21.56       0.660        2.4        0
  5801 - 6000        0.285        0.445        22.35       0.645        2.7        0
```

Fig. 6-23 Simulation run for IRCP-CONTEST subject to a channel input pulse.

INPUT PARAMETERS:

NUMBER OF TERMINALS M = 400 , PROPAGATION DELAY R = 12
FOR THE TIME PERIOD 1-1000, INPUT RATE $M\sigma$ = 0.3232
FOR THE TIME PERIOD 1001-1200, INPUT RATE $M\sigma$ = 1.0
FOR THE TIME PERIOD 1201-6000, INPUT RATE $M\sigma$ = 0.3232
$K_1 = 10$ $K_m = 150$ $(m \geq 2)$

RANDOM NUMBER GENERATOR SEED = 67289

AVERAGE VALUES IN 200 TIME SLOT PERIODS:

```
 TIME PERIOD     THROUGHPUT   TRAFFIC      PACKET     FRACTION    AVERAGE
                  RATE-  S    RATE-  G    DELAY- D     EMPTY      BACKLOG

    1 -  200       0.285       0.395       19.877      0.665        2.1
  201 -  400       0.320       0.390       16.328      0.650        1.2
  401 -  600       0.255       0.425       22.824      0.660        2.8
  601 -  800       0.290       0.475       26.172      0.630        4.0
  801 - 1000       0.325       0.570       28.554      0.570        5.7
 1001 - 1200       0.230       2.395       34.109      0.120       68.8
 1201 - 1400       0.285       1.695      141.333      0.215      112.6
 1401 - 1600       0.310       1.500      273.177      0.230       91.8
 1601 - 1800       0.375       1.415      288.693      0.190       68.5
 1801 - 2000       0.280       1.110      224.661      0.375       53.1
 2001 - 2200       0.360       1.240      257.333      0.300       48.8
 2201 - 2400       0.355       0.925      193.986      0.395       31.3
 2401 - 2600       0.385       0.655      122.818      0.490       15.2
 2601 - 2800       0.320       0.565       68.094      0.565        8.8
 2801 - 3000       0.280       0.420       39.357      0.660        5.6
 3001 - 3200       0.295       0.495       31.678      0.615        6.3
 3201 - 3400       0.265       0.630       45.000      0.545       11.7
 3401 - 3600       0.350       0.750       37.057      0.485       13.3
 3601 - 3800       0.310       0.465       65.274      0.625        8.2
 3801 - 4000       0.275       0.520       33.618      0.610        7.7
 4001 - 4200       0.330       0.480       34.652      0.595        5.2
 4201 - 4400       0.325       0.615       29.585      0.540        7.5
 4401 - 4600       0.370       0.525       38.608      0.560        7.6
 4601 - 4800       0.260       0.705       44.250      0.550       15.9
 4801 - 5000       0.375       0.720       63.520      0.460       11.1
 5001 - 5200       0.350       0.635       41.729      0.520        9.0
 5201 - 5400       0.285       0.475       29.368      0.625        6.6
 5401 - 5600       0.315       0.510       36.460      0.595        4.9
 5601 - 5800       0.290       0.425       24.190      0.650        4.1
 5801 - 6000       0.305       0.490       28.738      0.610        4.7
```

Fig. 6-24 Simulation run for Heuristic RCP subject to a channel input pulse.

(Q is different from our channel backlog size N^t since Q includes both backlogged and newly generated packets.) The control scheme suggested is that each of the Q channel users transmits in the next time slot with probability $\frac{1}{Q}$. This strategy maximizes the expected channel throughput in the next time slot (provided that Q is known exactly) and is referred to as throughput maximizing retransmission control. The channel performance given by this control scheme was studied through a steady-state analysis by Metcalfe [METC 73A]. However, the channel performance given by this control scheme in a dynamic environment (either through analysis or simulation) has not been studied.

Rettberg's proposal is concerned with satellite communication involving a small number (e.g., $M = 2$ to 10) of stations, each of which has buffering and scheduling capabilities. In Rettberg's scheme, newly generated packets attempt transmission over the channel without any delay. Previously collided packets form a queue at each station. Each station has a "gating" probability x of transmitting the packet at the head of its (backlog) queue in a time slot. Rettberg suggested that the gating probability may be chosen such that $Mx + S \leq 1$ where S is the channel input rate of new packets. Since in this case the channel traffic rate G is forced to be less than or equal to one,* no channel saturation will occur. Simulations [RETT 73C] supported this claim.

This scheme may be referred to as probability division multiplexing (PDM). Each channel user, instead of getting a fixed fraction

*$G = Mx\rho + S$, where ρ is the probability that a station's backlog queue is nonempty.

of the communication channel capacity such as in time division multiplexing (TDM) or frequency division multiplexing (FDM), now gets a random fraction of the channel capacity through the gating probability x . Thus, similar to TDM and FDM, this scheme will work quite well when M is small and each station has a relatively "smooth" input source. However, when M is large and each user has a bursty input source, PDM will suffer from the same pitfalls of FDM and TDM. That is, many channel users will often have an empty backlog queue (while others have very long queues). As a result, the actual channel traffic rate is very low, which gives rise to a small channel throughput rate. However, the average packet delay is high, since a small x (due to a large M) has been adopted. In this case, some scheme which allocates gating probabilities x_i to channel users dynamically as a function of their instantaneous transmission requirements may prove useful. (The constraint is now $\sum_{i=1}^{M} x_i + S \leq 1$.)

6.7.5 Channel Design Considerations

Consider the design of a slotted ALOHA channel characterized by the linear feedback model. Given M , σ and K , the channel load line and the equilibrium contour may intersect in three different ways depicted in Figs. 5-6 (a), (b) and (d).

In Fig. 5-6(d), the channel is overloaded in the sense that the globally stable equilibrium point corresponds to the channel saturation point. This situation should always be avoided (e.g., by reducing the number of channel users).

In Fig. 5-6(a), the channel operating point (n_o, S_o) is also the globally stable equilibrium point.

In this case, the assumption of channel equilibrium at (n_o, S_o) is valid. Hence, no channel control is necessary.

We have been mostly concerned with the dynamic control of an unstable channel such as shown in Fig. 5-6(b).

Consider the $K = 10$ equilibrium contour in Fig. 5-3. Given an average user think time $= \frac{1}{\sigma}$ (where $-\frac{1}{\sigma}$ is the slope of the channel load line), there is a maximum value of M such that the channel is stable. For example, if $\frac{1}{\sigma} = 615$ slots ($\cong$ 14 seconds), the maximum number of channel users is approximately 100 without rendering the channel unstable. At this value of M, the channel throughput rate $S_o \cong 0.162$ and the average packet delay $D \cong 17.5$ slots (0.394 second). If we want to increase the channel utilization (throughput) by increasing the number of channel users M, one of several things can be done:

(1) Do nothing.

(2) Increase K.

(3) Dynamic channel control.

Suppose M is 150 giving $S_o \cong 0.244$. The channel is now unstable, but from results in Chapter 5, has a channel FET of several days. If this is an acceptable channel failure rate, no external control is necessary except to restart the channel whenever it goes into saturation.

Increasing K from 10 to 60 allows the channel to support up to 200 users at $S_o \cong 0.32$. But now the throughput-delay tradeoff curve for $K = 60$ is much above the optimum performance envelope in Fig. 3-4. In Fig. 3-5, we see that for $S_o = 0.32$, $D = 45.5$ slots (1.02 second).

Dynamic channel control can give rise to a stable channel as well as providing a throughput-delay tradeoff close to the optimum envelope. For example, consider the results in Table 6.3 for $M = 200$ and $S_o = 0.32$. Under the assumption of perfect channel state information, a channel throughput-delay tradeoff very close to the optimum envelope is possible as shown in Figs. 6-17 to 6-20 for ICP and RCP. Without perfect channel state information, we have shown by simulations that throughput-delay results close to the optimum envelope can still be achieved using the CONTEST algorithms up to $S_o = 0.32$. (Recall that this is a consequence of the amazing flatness of S_{out} and D near the optimum except when S_o is large.) In any case, the channel operating point probably should not be designed with a value of $S_o > 0.32$. For $S_o > 0.32$, even if it is possible to achieve the optimum envelope, the incremental gain in channel throughput is at the expense of a sizable increase in delay.

In a real system, it is imaginable that the channel input may vary with time (say M fluctuating between say 100 to 200 in the above example). We must emphasize the fact that the control algorithms considered have been designed to control statistical channel fluctuations under the assumption of a stationary channel input.

Although we showed that they can temporarily handle very high channel input rates (see Figs. 6-23 to 6-24), other control mechanisms should be designed into the system to make sure that an overloaded channel such as depicted in Fig. 5-6(d) does not prevail for any long period of time (e.g., by limiting the maximum number of users who can "sign-on" and become active channel users).

We showed earlier that IRCP gives a channel performance at least as good as ICP and RCP. Furthermore, with two control limits $\hat{n}_1$ and $\hat{n}_2$, it acts like RCP (with $\hat{n}_1$) under normal channel conditions, but has a second "defense" in $\hat{n}_2$ whenever the channel traffic increases to a very high value. Comparing IRCP-CONTEST and Heuristic RCP, we see that the latter is easier to implement and exhibited in several simulations better channel performance for a heavily loaded channel $(S_o = 0.36)$. However, under a normal load (say $S_o \leq 0.32$), IRCP-CONTEST is superior to Heuristic RCP. This is because Heuristic RCP introduces longer delays to collided packets even when these packets are just unlucky in light channel traffic. On the other hand, in IRCP, control actions are not exerted until the channel traffic exceeds some critical values.