January, 1978 Report ESL-R-798

SOURCE CODING FOR COMMUNICATION CONCENTRATORS

by

Pierre Amedee Humblet

This report is based on the slightly altered thesis of Pierre Amedee Humblet submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology in December, 1977. This research was conducted at the Decision and Control Sciences Group of the M.I.T. Electronic Systems Laboratory, with partial support extended by the Advanced Research Projects Agency under Grant ONR-N00014-75-C-1183.

Electronic Systems Laboratory
Department of Electrical Engineering
and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

SOURCE CODING FOR COMMUNICATION CONCENTRATORS

by

PIERRE AMEDEE HUMBLET

Ing.Civ.Elec., Université Catholique de Louvain
(1973)

S.M., Massachusetts Institute of Technology
(1975)

E.E., Massachusetts Institute of Technology
(1975)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

(JANUARY 1978)

Signature of the Author... Pierre A. Humblet .....
Department of Electrical Engineering and Computer Science
December 13, 1977

Certified by..... Robert G. Gallager .....
Thesis Supervisor

Accepted by.....
Chairman, Department Committee

SOURCE CODING FOR COMMUNICATION CONCENTRATORS

by

PIERRE AMEDEE HUMBLET

Submitted to the Department of Electrical Engineering
and Computer Science

on January 4, 1978 in partial fulfillment of the requirements

for the Degree of Doctor of Philosophy

ABSTRACT

Communication concentrators perform the basic network function of merging many input flows into a single output flow. This requires formating the data and encoding side information about when messages start, what their lengths are and what their origins and destinations are.

This thesis examines efficient ways of performing these functions, the objective being to minimize the average message delay, or some other queueing theoretic quantity, like the probability of buffer overflow.

The work is divided in four parts:

- encoding of the data;
- encoding of message lengths;
- encoding of message starting times;
- encoding of message origins and destinations.

With respect to data encoding, an algorithm is given to construct a prefix condition code that minimizes the probability of buffer overflow.

Next a theory of variable length flags is developed and applied to

the encoding of message lengths.

For concentrators with synchronous output streams, it is shown that the concept of average number of protocol bits per message is meaningless. Thus, in order to analyze the encoding of message starting times, a class of flag strategies is considered in which there is a tradeoff between delay and low priority traffic.

The problem of encoding message origins and destinations is attacked from two different points of view. Some strategies (variations of the polling scheme) are analyzed and shown to be much more efficient in heavy traffic than just using a header, as is usually done. A simplified model is also developed. Its analysis suggests that there exist strategies to encode message origins and destinations that are much more efficient than everything considered until now.

Name and Title of Thesis Supervisor:

Robert G. Gallager

Professor of Electrical Engineering

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Professor Robert Gallager, my thesis supervisor, for invaluable counsel and support. He stimulated my interest in protocols for computer networks, suggested the topic of this thesis and generously contributed his time during this research.

I offer my thanks to Professor John Wozencraft for his comments in the early stages of this work, and to Professors Al Drake and James Massey for serving as thesis readers and sharing their expertise.

Finally I am grateful to my officemates Messrs. Eberhard Wunderlich, Thomas Marzetta and Joseph Defenderfer for the exchanges of ideas that made this research experience pleasant and productive.

This work was supported by a research assistantship from the Electronic Systems Laboratory with funds provided by the Advanced Research Projects Agency, by a Frank A. Cowan scholarship and by a Vinton Hayes Fellowship.

Data for a numerical example were kindly provided by Codex Corporation, of Newton, MA.

Chapter I

Preliminaries

1. Introduction

The last decade has seen a tremendous development of computer networks. Numerous books and papers describing and analyzing systems have appeared (see Section 2).

From the operations research point of view, the most studied problems are those of modelling the queueing phenomena in the networks, of routing the messages so as to minimize some cost, usually the average message delay, and of laying out the network in some optimal fashion.

Computer scientists have been concerned with the architecture of the computers in the nodes, and with the protocol, i.e. control messages exchanged between subsystems of the network. This is related to the problems associated with distributed computation.

Presently the most important consideration in the design of protocols is to get a working system where no deadlock can occur. Little attention has usually been paid to the effects of the overhead produced by the protocol on the performance of the network. However, taking a queueing theorist view of the problem, [Kleinrock et al., 1976] pointed out that the effect was significant in the ARPANET. [Gallager, 1976] showed that information theory can be used to produce basic lowerbounds on some of the information that is carried in the protocol messages.

Our goal is to obtain results similar to those of Gallager, but under less restrictive hypotheses. In particular, we will not assume an infinite number of sources and links of infinite capacity. Thus we will take into account queueing effects and interactions between sources. One will find in this work concepts and methods from the fields of queueing theory on one hand, and information and coding theories on the other.

We do not plan to solve at once all the protocol problems in a complete network. Instead, we pay attention only to the nodes, i.e. the points in the network where different links join each other. From our point of view a node can be decomposed in a "router" followed by "concentrators" (see Figure 1.1).

The role of the router is to determine the destination of each input bit and to send it, together with some associated information to be described later, to the corresponding concentrator. The concentrators merge the many input flows into one output flow.

We will not consider the structure or the optimization of the router, instead we will regard it as a source, with known statistics, to the concentrators.

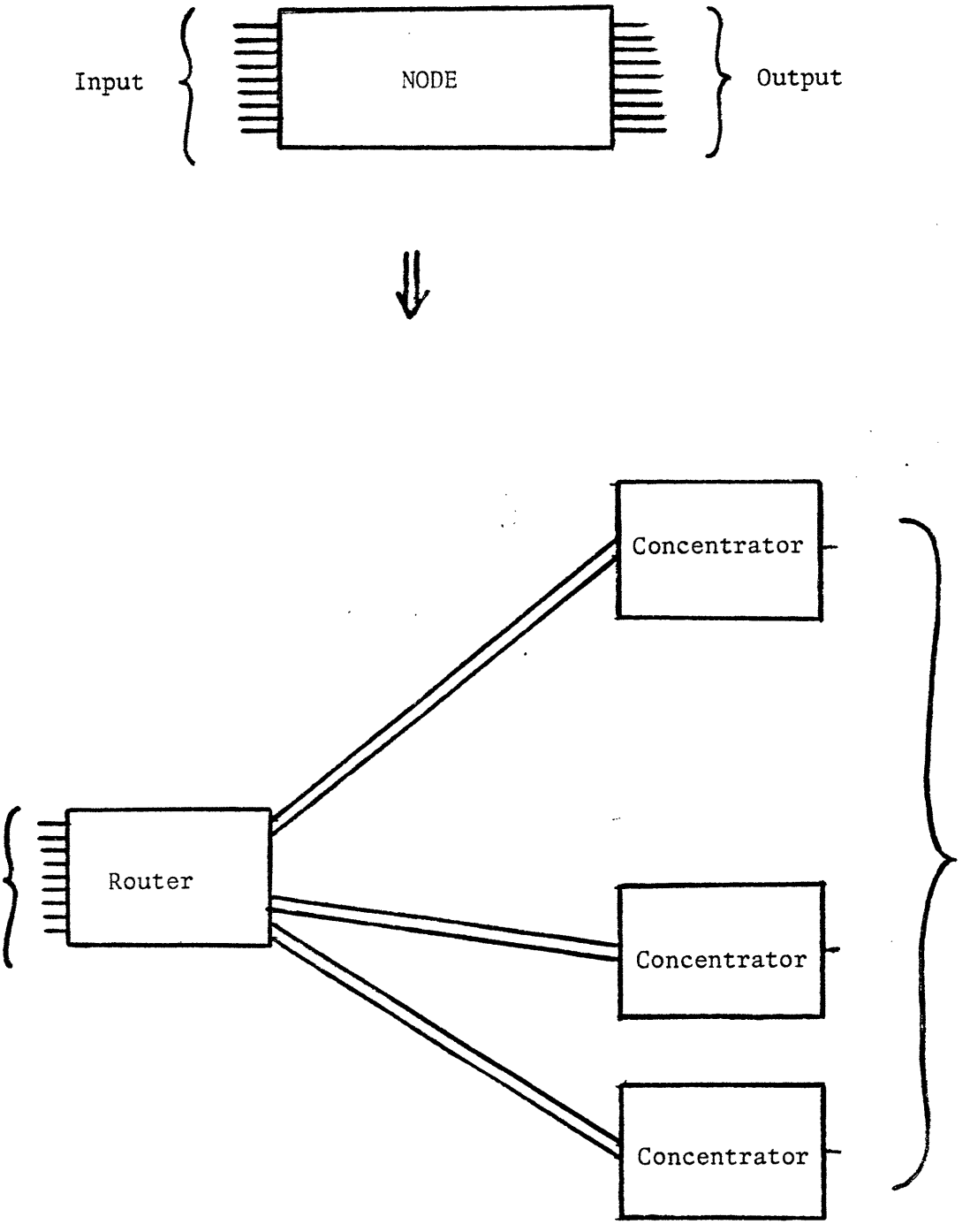
Because their input is generally stochastic, concentrators contain a buffer in which queueing phenomena occur. In addition to transmitting the data they received, concentrators usually perform other duties:

1° they reformat" the data. This may involve translating characters from one code to another, merging packets into messages or dividing messages into packets.

Figure 1.1

Decomposition of a Node into

a Router and Concentrators



2° they transmit service information to the downstream nodes:

- information about the line being idle or not;
- information about the origin and destination of the data.

3° they perform some kind of error control, typically implementing an error detection and retransmission system in conjunction with sending error detecting parity bits to the downstream node.

4° they send flow control information to the upstream nodes and/or the router indicating that they are unable in some way to handle the flow of data.

We will consider in this work only the first two functions; they are related to what information theorists call "source coding," whereas the third one is more like "channel coding." The fourth function should be studied with the routing question and is not touched here.

Note that classical source coding theory is interested in transmitting as little redundancy as possible. In computer networks the goal is usually to minimize the average message delay. These two objectives are not always compatible, as we shall see.

Note at this point that we consider all higher level protocol messages, like "end to end" messages to set-up a "session," and like flow control and routing messages, as regular data that must be transmitted by a concentrator to another node, together with information about its origin, destination and some error check.

The plan of this thesis is the following: in Section 2 of this chapter, we will review previous works of interest while we present in Section 3 an outline of the original contributions of this work. The next four chapters describe in detail the actual results. They are organized as follows:

In Chapter 2, we examine how the concentrator should encode the data so as to minimize in some sense the message delays.

In practical systems the data are often transmitted in batches, called "packets" or "messages." We analyse in Chapter 3 a very efficient way of encoding the length of these batches. This will introduce an encoding technique, using flags, which will be used extensively in the next two chapters.

In Chapter 4, we study efficient ways of solving a seemingly trivial problem: how should a concentrator indicate to the downstream node when it transmits idle bits. This simple problem will introduce some conceptual difficulties that appear more strongly in Chapter 5.

Chapter 5 treats the problem of encoding message origins and destinations. It has two distinct parts: in the first part we use a simplified model to see what issues are involved. In the second part we examine and compare different practical strategies for encoding the origins and destinations while not degrading too much the average message waiting time.

2. Review of Previous Works

We rapidly review previous works of interest, considering mainly works that give general ideas rather than technical details. These last references are mentioned in the text as they are needed.

Should a reader need general information about computer networks, the books of [Davies and Barber, 1973], [Abramson and Kuo, 1973] and [Schwartz, 1977] are valuable.

[Kleinrock, 1976] is an excellent reference on queueing models for computer systems, while [Gerla and Kleinrock, 1977] present an overview of the problems of optimal static routing and network layout and give a number of references. The subject of adaptive routing and numerous references on related subjects appear in [Segall, 1977] while [Gallager, 1977] offers an actual adaptive decentralized loopfree algorithm.

Many of the ideas used in high level protocols today were born during the development of the ARPANET; suitable references are [Crocker, 1972], [Cerf, 1977], [Kleinrock, 1976] and [Kleinrock and Opderbeck, 1977].

Of course the ARPANET is well known for sending data in packets. Another network that functions in a similar way is the CYCLADES,[Pouzin, 1973]. Some networks do not use this idea, but transmit the data character by character, e.g. see [Tymes, 1971] and [Vander Mey, 1976].

The references just mentioned describe the background of this thesis, but have no direct impact on it. We now review some works

that have a stronger relation to it.

The motivating paper behind this thesis is the one by [Gallager, 1976] which showed that there is a trade off between the delay incurred by a message and the amount of information necessary to indicate its origin or destination. However, the delay there is a "voluntary" delay in the sense that the concentrator sometimes chooses not to send a message although the line is available. We will examine how "involuntary" queueing delays can be exploited to minimize the amount of protocol.

Another paper along these lines is [Rubin, 1976]. Rubin notes that if some rate-distortion function exists for the output of a source, and if the output of the source encoder is sent over a link for which a relation exists between rate and average delay, one can obtain a delay-distortion relation. This approach is not very useful, because it neglects the delays added by the coding process and it assumes that the average delay on the link is only a function of the rate, and not of other parameters of the coder output statistics. It is an unfortunate fact that information theory is concerned only with rate.

A work that has a strong relation with this thesis is the one by [Jelinek, 1968] and [Jelinek and Schneider, 1972]. They were the first to show that a code with minimal redundancy is not necessarily optimal as far as buffering problems are concerned. We will use some of their ideas and extend their results in Chapter 2.

3. Outline of Original Contributions.

The goal of this thesis is to find efficient ways for a concentrator to perform the source coding functions described in Section 1, and divided in four main sections:

- encoding of the data;
- encoding of the message lengths;
- encoding of the idle times;
- encoding of the message origins and destinations.

The objective is to minimize the average message delay, or some other queueing theoretic quantity, like the probability of buffer overflow. We review briefly our contributions in these fields.

In Chapter 2, we present an algorithm to construct a prefix condition code minimizing the probability of buffer overflow. It is a generalization of Huffman's procedure.

Variable length flag strategies are studied exhaustively in Chapter 3. We give coding and decoding algorithms using flags, analyze their performance and sensitivity, and identify the classes of flags that have some desirable properties. The main result is that if well chosen flags are utilized to encode the length of a message, the expected number of bits used is upperbounded by the entropy of the distribution of the message length + .56 .

We study in Chapter 4 how to encode the message starting times to minimize the average message delay. Unfortunately the best way of doing this is still unknown. We were only able to show that the concept of average number of protocol bits per message is useless when the line is synchronous. We also analyzed a practical strategy, using flags, to

encode the starting times. This is a variation on the theme of the M/G/1 queue.

Our main contributions are in Chapter 5, where we study the encoding of the message origins. We first introduce a simplified model where the objective is to minimize the entropy of the sequence of the origins of the messages being transmitted. We also show that, at least for this model, the traditional methods (e.g. forming packets or polling) are far from being optimal. We give a lowerbound on the best achievable performance and show how dynamic programming can be used to find the optimal strategy.

We also analyze four practical strategies to encode the origins. They are based on well-known queueing strategies. Our main contributions are a closed form expression for the waiting time in cyclic queues with symmetric inputs, and a fast algorithm to compute the waiting times in the asymetric case. We also solved the problem of optimal source coding for an integer alphabet with Poisson distribution.

Chapter 2

Source Coding to Minimize Delay

1. Introduction

We devote this chapter to the problem of source coding to minimize delay. After presenting our model in Section 2, we consider briefly in Section 3 how to find a code minimizing the average delay. The problem of minimizing the probability of large delays or of buffer overflows is treated in Section 4. Finally, we review and generalize in Section 5 the work of [Jelinek and Schneider, 1972], which is stronly related to the topic of this chapter.

2. The Model

We propose the following model: an asynchronous memoryless source emits symbols drawn from the alphabet $\{1,2,3,\ldots,c\}$; symbol i has probability p_i . The time intervals between two source emissions are independent random variables with distribution function A . An encoder maps the source symbols into codewords which are placed in an output buffer of size M from which one letter is removed every unit of time (first in, first out). The output codewords are formed by letters from an alphabet of size d and the codeword corresponding to source symbol i has length m_i . Without loss of generality we can assume that $c = d + k(d-1)$ for some integer k and that $p_i \geq p_{i+1} \geq 0$.

In the following sections we consider the waiting time and delay of symbols that do not cause buffer overflows. The waiting

time is defined as the time difference between the moment a symbol arrives at the encoder and the moment the corresponding codeword starts leaving the buffer. The delay is the waiting time, plus the length of the codeword. We do not consider what to do when the buffer is empty or overflows; this is treated in Chapter 4.

3. Minimizing the Average Delay

Unfortunately, for most interemission processes, it is not possible to compute the average delay. Sometimes, though, it is feasible, e.g. if the buffer is infinite and if $A(t) = 1 - e^{-\lambda t}$ $t \geq 0$. In this case the average delay is equal to (this is a M/G/1 queue)

$$E[D] = \frac{\lambda(\Sigma\, p_i m_i^2 - (\Sigma\, p_i m_i)^2) + \Sigma\, p_i m_i}{1 - \lambda\, \Sigma\, p_i m_i}$$

for all codes such that $\lambda\, \Sigma\, p_i m_i < 1$. However, even in this simple case we are unable to find an algorithm yielding a code that minimizes this expression. We can only make three general observations valid for all problems.

First, Huffman codes, which minimize the average codeword length, are robust for this application. They are optimal when the load is light, because then the waiting time is negligable compared to the average codeword length. When the load is heavy, it is of primary importance to keep the system stable by minimizing the average codeword length, i.e. utilizing a Huffman code.

Next, by a simple exchange argument, one sees that in an optimum code $m_i \geq m_{i+1}$ (because $p_i \geq p_{i+1}$).

Finally, as in Huffman codes, the codewords of the d least likely symbols have the same length.

4. Minimizing the Probabilities of Buffer Overflow and of Long Delays

A. Introduction

[Kingman, 1970] showed that for infinite G/G/1 queues with interarrival time distribution A and service time distribution B, the stationary probability $W^c(x)$ that a symbol waits more than x units of time is upperbounded by

$$W^c(x) \leq e^{-s^o x}$$

where s^o is the supremum of the values of s such that

$$A^*(s)\, B^*(-s) \leq 1$$

Kingman's method yields the same result for finite queues.

From this, it is easy to upperbound the probability of buffer overflow: denoting by w and b the waiting time and length of a codeword we have

$$\begin{aligned} \text{probability of buffer overflow} &= P(w+b > M) \\ &= P(w > M-b) \\ &\leq E(e^{-s(M-b)}) \quad 0 \leq s \leq s^o \\ &= B^*(-s)\, e^{-sM} \quad 0 \leq s \leq s^o \end{aligned}$$

By more complicated arguments, [Wyner, 1974] established that there

exists a lowerbound decreasing like $K e^{-s^o M}$.

Applying these results to our model, we see that for every code C , the probability of buffer overflow is of the order of $e^{-s^o B}$, where $s^o(C)$ is the supremum of the values of s such that $F(C,s) := A^*(s) \sum_{i=1}^{c} p_i e^{sm_i} \leq 1$. Therefore it is desirable to find a uniquely decodable code with the largest s^o . Before doing this, we will bound this largest s^o .

B. Bounds on the Largest s^o

This section can be considered as an extension to asynchronous sources of results obtained by [Jelinek, 1968] and outlined in Section 5 . For any uniquely decodable code $\sum_{i=1}^{c} d^{-m_i} \leq 1$ [Gallager, 1968, p. 47], and by Hölder's inequality for all $s \geq 0$

$$\left(\sum_{i=1}^{c} p_i \, e^{sm_i} \right)^{\frac{\ln d}{\ln d + s}} \left(\sum_{i=1}^{c} d^{-m_i} \right)^{\frac{s}{\ln d + s}}$$

$$\geq \sum_{i=1}^{c} p_i^{\frac{\ln d}{\ln d + s}}$$

Thus for all uniquely decodable codes,

$$A^*(s) \sum_{i=1}^{c} p_i \, e^{sm_i} \geq A^*(s) \left(\sum_{i=1}^{c} p_i^{\frac{\ln d}{\ln d + s}} \right)^{\frac{\ln d + s}{\ln d}}$$

with equality for a given s iff

$$m_i = m_i^*(s) := - \log_d \left(p_i^{\frac{\ln d}{\ln d + s}} \Big/ \sum_{j=1}^{c} p_i^{\frac{\ln d}{\ln d + s}} \right)$$

which is rarely possible, because m_i must be integer. However, for

every s, there is a uniquely decodable code with

$$m_i^*(s) + 1 > m_i \geq m_i^*(s)$$

Thus we can conclude that the largest s^o is upperbounded by s_u, defined as the supremum of the values of s such that

$$A^*(s)\left(\sum_{i=1}^{c} p_i^{\frac{\ln d}{\ln d + s}}\right)^{\frac{\ln d + s}{\ln d}} \leq 1$$

and lowerbounded by the supremum of the values of s such that

$$e^s A^*(s)\left(\sum_{i=1}^{c} p_i^{\frac{\ln d}{\ln d + s}}\right)^{\frac{\ln d + s}{\ln d}} \leq 1$$

Further, s_u is achievable if $m_i^*(s_u)$ is an integer for all i.

Finally, we note that if we were encoding blocks of n input symbols, the largest s^o would still be upperbounded by s_u, and lowerbounded by the supremum of the values of s such that

$$e^s\left(A^*(s)\left(\sum_{i=1}^{c} p_i^{\frac{\ln d}{\ln d + s}}\right)^{\frac{\ln d + s}{\ln d}}\right)^n \leq 1$$

This supremum increases to s_u as n grows.

C. An Algorithm to Construct an Optimal Prefix Condition Code

In this section we present an algorithm to construct a prefix condition code with the largest achievable s^o. It is well known [Gallager, 1968, p. 49] that no gain would be achieved by considering non prefix condition, uniquely decodable codes. The algorithm has two

main steps that we describe first.

Step I finds a prefix condition code minimizing $\sum_{i=1}^{c} p_i e^{sm_i}$ for a given $s \geq 0$ by the following method. As [Huffman, 1952] noticed a quarter of century ago, there is an optimal prefix condition code where the codewords corresponding to symbols $c - d + 1$ to c are the longest and differ only in the last character. If $c = d$, this specifies an optimal code; if $c > d$, this reduces the problem to finding a prefix condition code of size $c - d + 1$ minimizing $\sum_{i=1}^{c-d} p_i e^{sm_i} + (e^{s} \sum_{i=c-d+1}^{c} p_i) e^{sm_{c-d+1}}$. Again we can make the same observation and continuing we will eventually reach the point where the code is completely specified.

One sees that for $s = 0$ this algorithm yields a Huffman code, whereas for s large enough, it assigns codewords of length $\lceil \log_d c \rceil - 1$ to the

$$\frac{d^{\lceil \log_d c \rceil} - c}{d - 1}$$

most likely symbols, and codewords of length $\lceil \log_d c \rceil$ to the others. By definition we will say that such a code is generated for $s = \infty$.

Note that, depending on the actual implementation of the algorithm, many different codes may be generated for a given s. They all minimize $\sum_{i=1}^{c} p_i e^{sm_i}$ but it may happen that all of them do not have the same s^o.

Step II computes the s^o corresponding to a particular code. Except in special cases this must be done by numerical methods, e.g. the Newton-Raphson algorithm [Klerer and Korn, 1967, p. 2-59]. There are

no special problems because the function $f(C,s)$, defined at the end of Section A, is convex in s for all codes C.

The main part of the algorithm is as follows: (see Fig. 2.1)

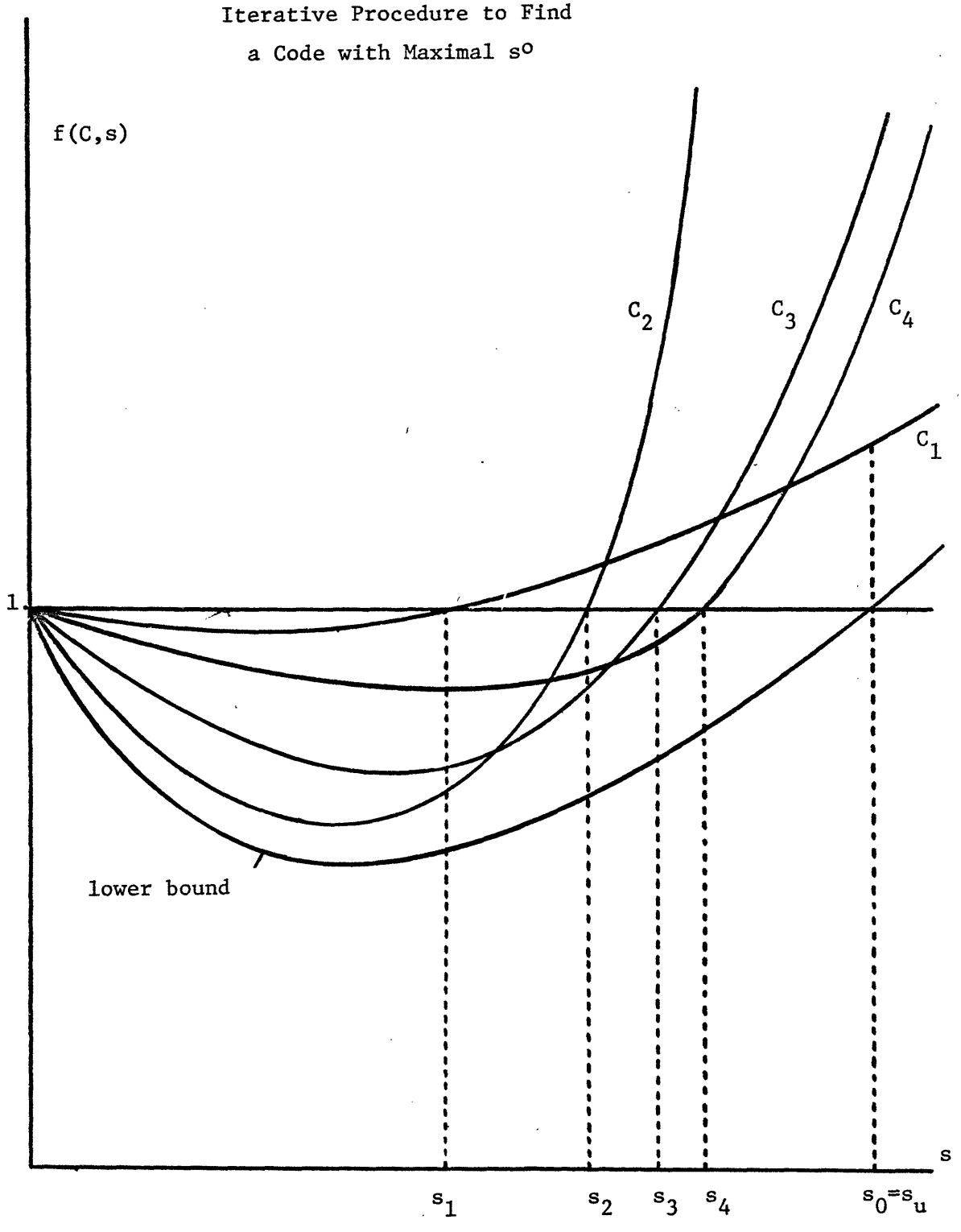
1		compute s^u
2		$s_o := s^u$
3		$j := 1$
4	Loop	use Step I to find a code minimizing $\sum_{i=1}^{c} p_i e^{s_{j-1} m_i}$
		denote this code by C_j
5		use Step II to find the s^o corresponding to C_j
		denote this s^o by s_j
6		if $s_j = s_{j-1}$ then stop
7		else $j := j+1$
8		go to Loop

Of course, we must show that this algorithm will terminate after a finite time, and that the last code generated is optimum. The proof is simple. First we note that $s_{j+1} \geq s_j \quad j \geq 1$ because $f(C_j, s_j) \leq 1$ (line 5), thus (line 4) $f(C_{j+1}, s_j) \leq 1$ so $s_{j+1} := \sup \{s : f(C_{j+1}, s) \leq 1\} \geq s_j$. Secondly, we observe that the maximum codeword length of any code generated by Step I is less than c, so the number of codes that can possibly be generated by Step I is finite. These two remarks insure that the algorithm will terminate in a finite time.

Let C_* and s_* be the last generated code and its s^o. We must show that C_* is optimal. If it is not, there will be a prefix

FIGURE 2.1

Iterative Procedure to Find
a Code with Maximal s^o



condition code C_*' and a corresponding s_*' with $s_*' > s_*$. Thus $0 \leq s_* < \infty$, so $f(C_*',s_*) = 1$. Also, by convexity of $f(C_*',s)$, $f(C_*',s_*) \leq 1$.

If $f(C_*',s_*) < 1$, C_* may not be the last code generated by the algorithm (lines 4,5,6).

If $f(C_*',s_*) = 1$ and $s_* > 0$, by invoking the facts that $f(C_*',0) = 1$, $f(C_*',s_*') \leq 1$ and the convexity of $f(C_*',s)$ we can conclude that $f(C_*',s) = 1$ $s \in [0,s_*']$. By analyticity of $f(C_*',s)$ $s > 0$ (Laplace-Stieltjes transform of a probability distribution), $f^*(C_*',s) = 1$ $s \geq 0$, so $s_*' = \infty$ and a fortiori $s_u = \infty$. From the algorithm, C_1 is the code described earlier that is generated by Step I for $s = \infty$. If for C_*' the waiting time is 0 with probability one (i.e. $\bar{s_*'} = \infty$), it is clear that the same will be true for C_1, because the length of the longest codeword in C_1 is no longer that the length of the longest codeword in any other code. Thus $\infty = s_*' = s_1 \leq s_*$, a contradiction.

If $f(C_*',s_*) = 1$ and $s_* = 0$, then, as noted earlier, C_* is an Huffman code, and as such minimizes $\frac{d}{ds} f(C,s)\Big|_{s=0}$ over all codes. The fact that $s_* = 0$ implies that $\frac{d}{ds} f(C_*,s)\Big|_{s=0} \geq 0$ so $\frac{d}{ds} f(C_*',s)\Big|_{s=0} \geq 0$ and by convexity either $s_*' = 0 = s_*$, which is a contradiction, or $s_*' > 0$, $\frac{d}{ds} f(C_*',s)\Big|_{s=0} = 0$. As in the previous paragraph this leads to the conclusion that $f(C_*',s) = 1$ $s \geq 0$ and to a contradiction.

We have exhausted all possibilities and may conclude that C^* is optimal.

Before leaving this section, we show that if one desires to find a prefix condition code minimizing

$$\sum_{i=1}^{c} p_i \, g(m_i)$$

then the algorithm of Step I can be used only if g is linear or exponential.

The following conditions on g must be met for the algorithm to work:

-- g is non-decreasing,

so that if $p_i > p_j$, $m_i \leq m_j$ in an optimal code;

-- $g(m+1) = a\, g(m) + b$

so that at every step the size of the problem can be reduced by 1, while the form of the problem does not change.

These conditions imply that f must have one of the forms

$$g(m) = \alpha^m + \beta \qquad \alpha \geq 1$$

or $$g(m) = \alpha m + \beta \qquad \alpha \geq 0$$

D. Numerical Results

A listing of a Fortran IV program implementing the two main steps of the previous algorithm appear in Appendix C. This program was used to compute the optimal code for a 128 symbol alphabet. The symbol probabilities are equal to the relative frequencies measured in an airline reservation system, and are listed in Table 2.1. We are grateful to Codex Corporation for furnishing these numbers.

We used two kinds of interarrival time distributions: deterministic and exponential. This last one is a realistic model of what happens in practice, see [Fuchs and Jackson, 1969], or [Lewis and Hue,

Table 2.1

Symbol Probabilities Used in the Example

1	0.208593E 00	44	0.543153E-02	87	0.434284E-03
2	0.413809E-01	45	0.532954E-02	88	0.344279E-03
3	0.359989E-01	46	0.519072E-02	89	0.301999E-03
4	0.344146E-01	47	0.510923E-02	90	0.282097E-03
5	0.341741E-01	48	0.495080E-02	91	0.281404E-03
6	0.310807E-01	49	0.495080E-02	92	0.240114E-03
7	0.297105E-01	50	0.431145E-02	93	0.227836E-03
8	0.252622E-01	51	0.410917E-02	94	0.125453E-03
9	0.250547E-01	52	0.410461E-02	95	0.123671E-03
10	0.239848E-01	53	0.381984E-02	96	0.800050E-04
11	0.214987E-01	54	0.373736E-02	97	0.207934E-05
12	0.205013E-01	55	0.371647E-02	98	0.376261E-05
13	0.204832E-01	56	0.335328E-02	99	0.100006E-04
14	0.204295E-01	57	0.334189E-02	100	0.514884E-05
15	0.203151E-01	58	0.323951E-02	101	0.237639E-05
16	0.185034E-01	59	0.321822E-02	102	0.237639E-05
17	0.170439E-01	60	0.289216E-02	103	0.891145E-05
18	0.141916E-01	61	0.279186E-02	104	0.376261E-05
19	0.134732E-01	62	0.271047E-02	105	0.257442E-04
20	0.126853E-01	63	0.261284E-02	106	0.360418E-04
21	0.126820E-01	64	0.252630E-02	107	0.192091E-04
22	0.126658E-01	65	0.219340E-02	108	0.514884E-05
23	0.126555E-01	66	0.213528E-02	109	0.207934E-05
24	0.120663E-01	67	0.181754E-02	110	0.308930E-04
25	0.115880E-01	68	0.171922E-02	111	0.171298E-04
26	0.114259E-01	69	0.168040E-02	112	0.960456E-05
27	0.114121E-01	70	0.155020E-02	113	0.514884E-05
28	0.110366E-01	71	0.143781E-02	114	0.297048E-06
29	0.104807E-01	72	0.143712E-02	115	0.178229E-04
30	0.969496E-02	73	0.142068E-02	116	0.236648E-04
31	0.957297E-02	74	0.136910E-02	117	0.306950E-05
32	0.944445E-02	75	0.131790E-02	118	0.554490E-05
33	0.932216E-02	76	0.123206E-02	119	0.138622E-05
34	0.881332E-02	77	0.116750E-02	120	0.378241E-04
35	0.844231E-02	78	0.942039E-03	121	0.653506E-05
36	0.831517E-02	79	0.912136E-03	122	0.306950E-05
37	0.826121E-02	80	0.865797E-03	123	0.930751E-05
38	0.809219E-02	81	0.767177E-03	124	0.116839E-04
39	0.753829E-02	82	0.719054E-03	125	0.196052E-04
40	0.737234E-02	83	0.639347E-03	126	0.207934E-05
41	0.648664E-02	84	0.630138E-03	127	0.116839E-04
42	0.645882E-02	85	0.594690E-03	128	0.415867E-05
43	0.602760E-02	86	0.583007E-03		

1972].

The results appear in Fig. 2.2 and 2.3. We give some additional information here:

the binary entropy of the alphabet is equal to 5.32 ;

the average codeword length of a Huffman code is equal to 5.35 ;

the number of iterations to reach the optimal code was generally small (1 or 2) for Poisson arrivals, but larger (3 to 10) for deterministic arrivals;

the difference between the upperbound on s^o , and the performance of the optimal code is extremely small (of the order of 1%) in the Poisson arrival case. This is the reason why the upperbound does not appear in Fig. 2.3.

The average codeword length of the optimal code behaves in the expected fashion; being largest in light traffic, but close to the average codeword length of the Huffman code in heavy traffic.

Figure 2.2: Code Performances: Deterministic Arrivals

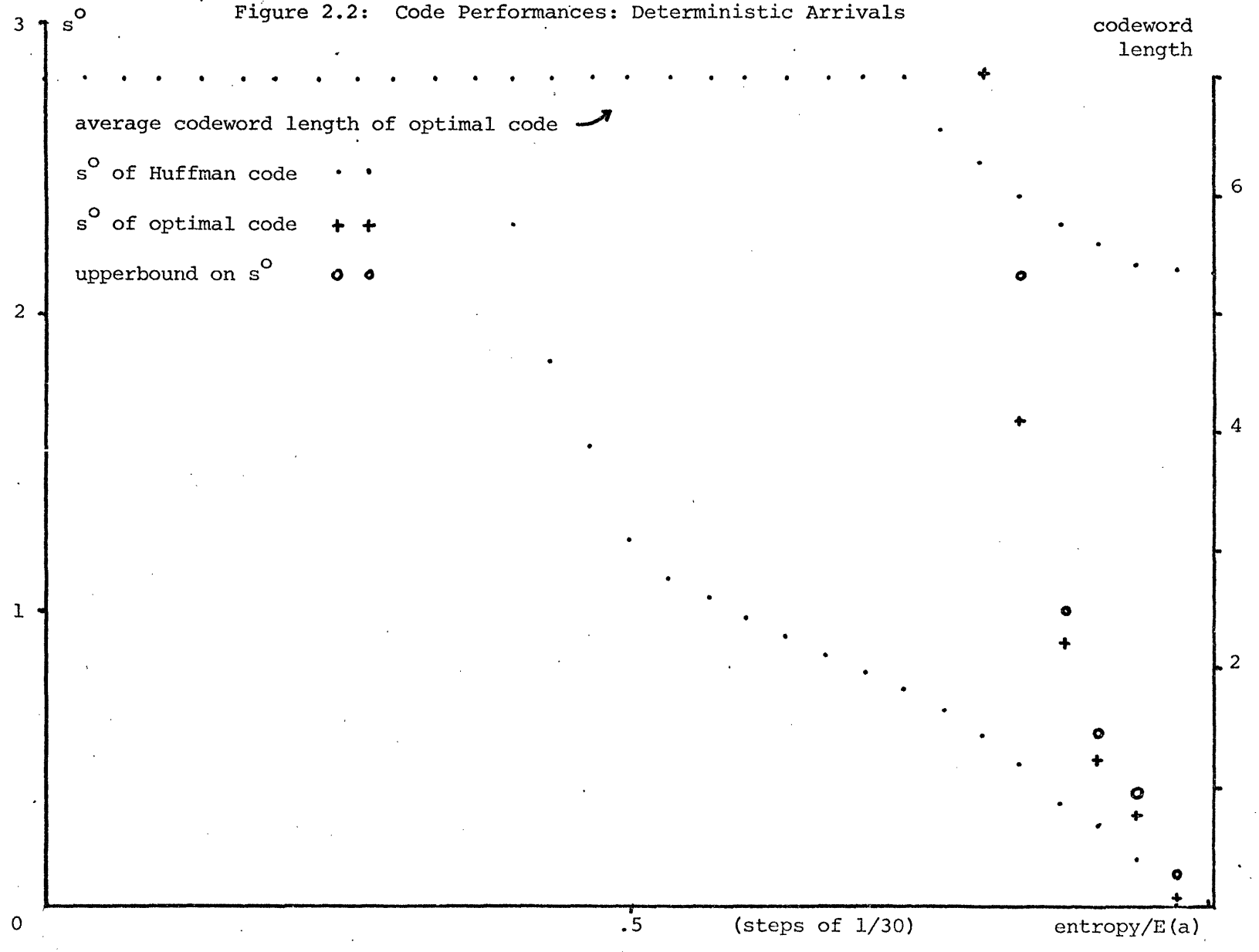
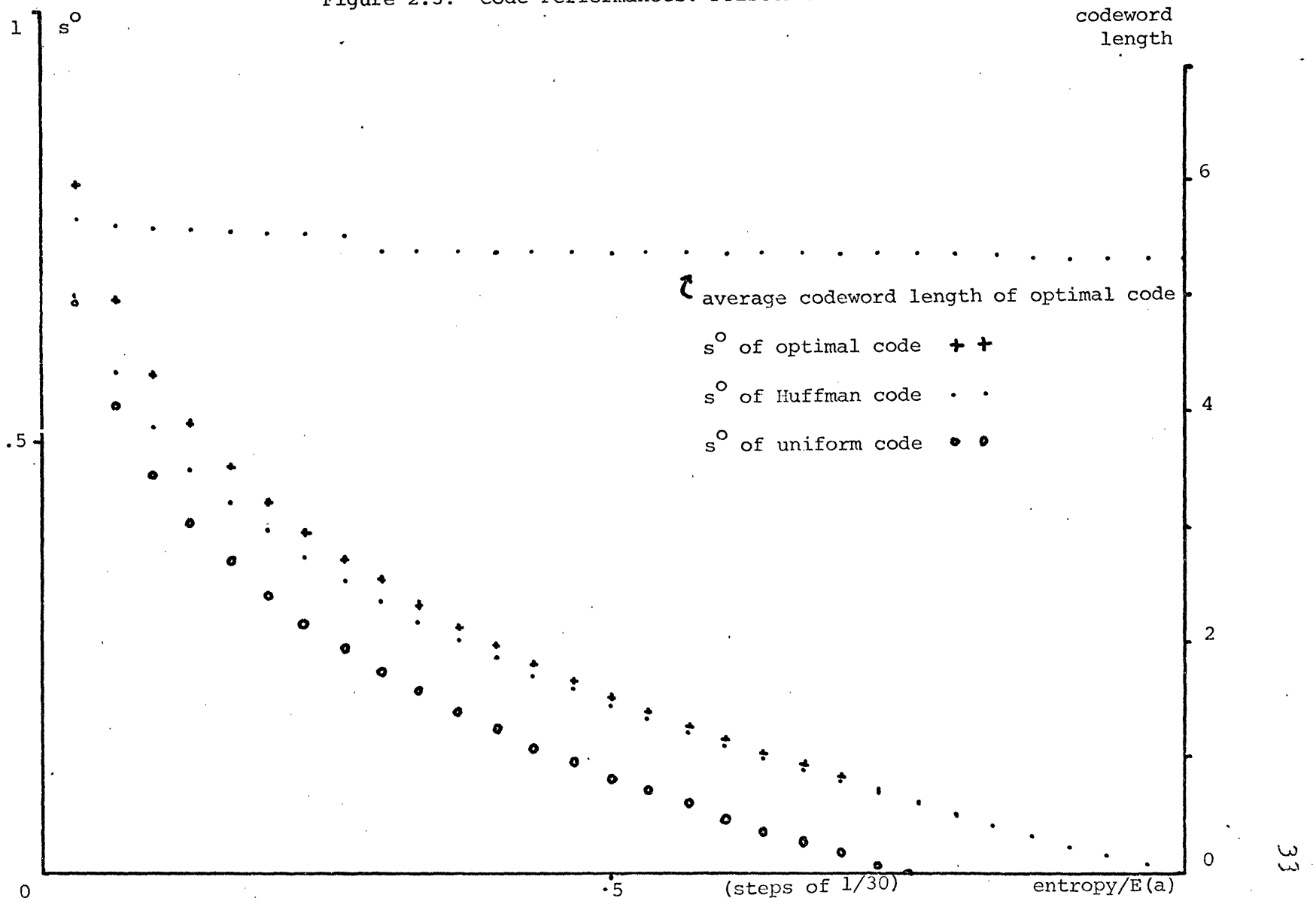


Figure 2.3: Code Performances: Poisson Arrivals



5. Review and Generalization of Jelinek and Schneider's Work

Jelinek and Schneider considered the following problem: once per time unit a memoryless source emits a letter from the alphabet $A := \{1,2,\ldots,c\}$. Letter i in this alphabet has probability $p_i > 0$. An encoder maps these source letters into codewords formed by letters from the alphabet $B := \{1,2,\ldots,d\}$. The mapping is as follows: a complete and proper set of N prefixes c_j is defined for the alphabet A (i.e. every sequence of letters from A starts with one and only one c_j). Prefix c_j has length r_j and probability q_j, induced by the p_i's. Every c_j is mapped into one codeword d_j formed by letters from B. Codeword d_j has length m_j and the d_j's are uniquely decodable, so that $\sum_{j=1}^{N} d^{-m_j} \leq 1$ (this is the Kraft inequality see [Gallager, 1968, p. 47]). Each time the prefix c_j is recognized by the encoder, codeword d_j is placed in a buffer of size B from which one letter is removed every time unit. Jelinek and Schneider address in detail the problem of what should be done when the buffer is empty or overflows.

Their main result is the following: for every block to variable length code (r_j constant), or variable length to block code (m_j constant), there exists $K_1, K_2 > 0$ and s^o such that in the stationary state, for all $B \geq 1$,

$$K_1 d^{-s^o B} \leq \text{Probability of buffer overflow} \leq K_2 d^{-s^o B} \quad (1)$$

where s^o is less than or equal to the supremum s_u of the values of s such that

$$d^s \geq \left(\sum_{i=1}^{c} p_i^{\frac{1}{1+s}} \right)^{1+s} \tag{2}$$

s_u is positive if the entropy (base d) of the source is less than one (this ensures stability) and is finite if $c > d$ (otherwise there need be no queueing effect). In the sequel, we always assume that s_u is positive and finite so that s_u can be dfined as the largest root of the equation

$$d^s = \left(\sum_{i=1}^{c} p_i^{\frac{1}{1+s}} \right)^{1+s}$$

They give algorithms yielding codes with exponent s^o arbitrarily close to s_u, and conjecture that the same result would hold in variable length to variable length coding. We show now that this conjecture holds.

To show that the theoretical limit on the exponent s^o is the same for the variable length to variable length codes as for the codes considered by Jelinek and Schneider, it is enough to show that for every code there exists a $K_1 > 0$ such that for every $B \geq 1$

$$\Pr(\text{Buffer overflow}) \geq K_1 d^{-s_u B}$$

Because we consider only the lowerbound, we can ignore the overhead

associated with the recovery procedures that have to be used when the buffer overflows or becomes empty.

Denote by

m^k the length of the k^{th} codeword placed in the buffer $(k = 1,2,3,\ldots)$

r^k the length of the prefix corresponding to the k^{th} codeword

n^k the number of letters in the buffer after the k^{th} codeword has been placed in it.

Note that m^k and r^k are strongly dependent, but are independent of the m^j's and r^j's $j \neq k$.

We have the relation

$$n^k = \text{Min}\,[B,\ m^k + \text{Max}[0,\ n^{k-1} - r^k]] \qquad k = 1,2,\ldots$$

$$= \text{Min}\,[B,\ \text{Max}\,[m^k,\ n^{k-1} + m^k - r^k]]$$

and we assume $n^o = 0$.

Now defining

$$w^o = 0$$

$$w^k = \text{Min}\,[B,\ \text{Max}[0,\ w^{k-1} + m^k - r^k]] \qquad k = 1,2,\ldots$$

we see that w^k obeys the standard relation for the waiting time in a queue and that surely $n^k \geq w^k \quad k = 0,1,2,\ldots$

Thus the probability of an overflow for the process n^k is greater than or equal to the probability of an overflow for the process w^k.

The results of [Wyner, 1974] can be applied to this last process, thus for every code, there are $K_1 > 0$ and s^o such that Pr [Buffer overflow] $\geq K_1\, d^{-s^o B}$ where s^o is the largest root of

$$\sum_{j=1}^{N} q_j \, d^{s(m_j - r_j)} = 1$$

($\sum_{j=1}^{N} q_j \, d^{s(m_j - r_j)}$ is the Laplace-Stieltjes transform (base d) of the distribution of $r^n - m^n$, $n = 1,2,\ldots$)

[Jelinek and Schneider, 1972] give a proof of the following Lemma, attributed to Forney:

If s_u is defined as before, then for all complete and proper set of prefixes, $\sum_{j=1}^{N} q_j \, d^{\frac{1}{1+s_u} - r_j \frac{s_u}{1+s_u}} = 1$

Now, Hölder's inequality yields

$$\left(\sum_{j=1}^{N} q_j d^{s_u(m_j - r_j)} \right)^{\frac{1}{1+s_u}} \left(\sum_{j=1}^{N} d^{-m_j} \right)^{\frac{s_u}{1+s_u}} \geq \sum_{j=1}^{N} q_j \, d^{\frac{1}{1+s_u} - r_j \frac{s_u}{1+s_u}}$$

thus by the Lemma and the fact that $\sum_{j=1}^{N} d^{-m_j} \leq 1$,

$$\sum_{j=1}^{N} q_j \, d^{s_u(m_j - r_j)} \geq 1$$

with equality if and only if

$$\sum_{j=1}^{N} d^{-m_j} = 1$$

and

$$d^{-m_j} = \left(q_j \, d^{-s_u r_j} \right)^{\frac{1}{1+s_u}}$$

Now, the function $\sum_{j=1}^{N} q_j \, d^{s(m_j - r_j)}$ is a Laplace-Stieltjes transform of a probability distribution, thus it is strictly convex (except in a trivial case), and its value at 0 is 1. We have seen that its value is greater than or equal to 1 at $s_u > 0$, thus it is greater than 1

for all $s > s_u$, so $s^o \leq s_u$ for all variable length to variable length codes.

Chapter 3

Flag Encoding Schemes

1. Introduction

Consider the problem of finding a binary Huffman code to jointly encode a binary random variable, which is equal to 1 with probability .15, and another random variable which takes the values $(0,1,2,\ldots,7)$ with equal probability. One readily finds that the following code is a solution:

(0,0)	000	(1,0)	111000
(0,1)	001	(1,1)	111001
(0,2)	010	(1,2)	111010
(0,3)	011	(1,3)	111011
(0,4)	100	(1,4)	111100
(0,5)	101	(1,5)	111101
(0,6)	1100	(1,6)	111110
(0,7)	1101	(1,7)	111111

This code has an interesting structure: all codewords corresponding to $(1,i)$ start with 111, followed by the binary representation of i. $(0,i)$ is encoded into the binary representation of i, except that a 0 is inserted in third position if the first two digits are 11. The same pattern reappears in the joint Huffman encoding of a binary random variable and a random variable taking with equal probability anyone of 2^n values.

This structure offers the possibility of doing the coding in two steps: first encoding the messages, then modifying the codewords, either by using a prefix, called a flag, or inserting an extra symbol to avoid confusion, to encode the binary random variable. The receiver will recognize if a flag is present, possibly recognize and delete the extra

character, then decode the message.

Often in computer communication networks and elsewhere, one needs to jointly encode messages, furnished by an outside source, and binary information generated locally, like an "acknowledgement" or "end of transmission." This can be done easily by eventually introducing a flag, known to the receiver, at the beginning of a message, or at some other point decided in advance, and inserting extra symbols to avoid confusion, if necessary.

This strategy is attractive for many reasons: it is simple, does not cause much delay, nor require much buffering because the message is not truly reencoded and does not need to be known in its entirety. It is optimal in some cases, as we have just seen, and can be made adaptive, as we shall see later.

In this chapter, we will study this strategy in detail . We will first give a very general algorithm that permits the use of any flag at any point in a message. Next we will study the performances of this strategy and see how it can be optimized. In the following section we examine the use of adaptive flags to encode messages and batch lengths. Finally we will see how reducing the class of allowable flags can improve performances.

Before doing this, we introduce some definitions. By <u>flag</u> we mean any finite sequence $(\alpha_1 \ldots \alpha_\nu)$ of symbols from the alphabet $\{0,1,..,d\}$; ν is called the <u>length</u> of the flag $(\nu \geq 1)$ while $(\alpha_1 \ldots \alpha_{\nu-1})$ is called the <u>root</u> ρ (ρ is possibly empty). We denote by β the symbol, different from α_ν , that is inserted when necessary

to avoid ambiguities, and we will call the sequence $(\alpha_1, \alpha_2, \ldots, \alpha_{\nu-1}, \beta)$ the antiflag.

Fixed-length flags are actually used in IBM Synchronous Data Link Control to encode message lengths [Donnan and Kersey, 1974]. They are analyzed in [Camrass and Gallager, 1976]. [Schalkwijk and Post, 1973] used flags to encode data for transmission on the Binary Symmetric Channel with Noiseless Feedback.

2. General Flag Coding Algorithm

We consider the following situation: a semi-infinite sequence (the data) $(u^1, u^2, \ldots)$ of d-ary symbols is furnished to an encoder, together with a sequence $(v^1, v^2, \ldots)$ of binary symbols. We give an algorithm to jointly encode these two streams using flags, i.e. the output $(x^1, x^2, \ldots)$ will consist of the sequence $(..u^t..)$ plus some flags or inserted symbols used to indicate the values of the v^t's.

We denote by $(\alpha_1^t, \ldots, \alpha_{\nu_t}^t)$ the flag to be used after u^t if $v^t = 1$, by ρ^t the root of this flag, and by β^t the symbol that is to be inserted in case of possible confusion. We place no restriction on the composition of the flags, except that of course $\beta^t \neq \alpha_{\nu_t}^t$.

Before giving the algorithms for coding and decoding we note that they need the following features to be efficient:

a) we want to either use the flag corresponding to a v^t, or to make at most one insertion;

b) if $d > 2$ we want to make an insertion only when it is necessary, i.e. when the next symbol is the same as the last flag symbol or the insertion.

We will illustrate these two points. Throughout examples 1 to 3 we use $d=3$ and

$$\nu_1 = 4 \qquad (\alpha_1^1\ \alpha_2^1\ \alpha_3^1\ \alpha_4^1) = (0\ ,\ 0\ ,\ 0\ ,\ 0) \qquad \beta^1 = 2$$

$$\nu_2 = 2 \qquad (\alpha_1^2\ \alpha_2^2) = (0\ ,\ 0) \qquad \beta^2 = 2$$

Example 1: Violation of requirement a)

$u^1 = 1 \quad u^2 = 1$

$v^1 = 1 \quad v^2 = 1$

x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}
1	0	0	2	0	0	1	0	0	.
u^1	α_1^1	α_2^1	β^2	α_3^1	α_4^1	u^2	α_1^2	α_2^2	

There we insert β^2 in the middle of the first flag to indicate that we are not transmitting the second flag. We transmit the second flag in x_8 and x_9. We have thus used both the flag and the insertion.

The correct way of proceeding is illustrated below.

Example 2:

x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^0
1	0	0	0	0	0	0	1	...
u^1	α_1^1	α_2^1	α_2^2	α_2^1	α_3^1	α_4^1	u^2	
		$\downarrow 2$						
		α_1						

We realize that if x_4 is 0, x_3 and x_4 will be interpreted as the second flag. We then repeat α_2^1 in x_5 and continue the transmission of the first flag, which will be decoded after the second.

If we had to transmit $u^1 = 1 \quad u^2 = 1$

$v^1 = 0 \quad v^2 = 1$

the output would be

Example 3:

$$x^1 \quad x^2 \quad x^3 \quad x^4$$
$$1 \quad 1 \quad 0 \quad 0$$
$$u^1 \quad u^2 \quad \alpha_1^2 \quad \alpha_2^2$$

We see that here the second flag appears after u_2 . To insure that the encoder does not repeat the second flag after u_2 in example 2 we introduce in the algorithm below the indicator variable w^t which is initially set to 1 , then to 0 as soon as an insertion or a flag corresponding to v^t are transmitted. Once $w^t = 0$ no more flag or insertion corresponding to v^t can be sent.

Let us look now at the peculiarities introduced by requirement b). Here we use $d=3$ and

$$\nu_1 = 3 \qquad (\alpha_1^1 \ \alpha_2^1 \ \alpha_3^1) = (0 \ , \ 0 \ , \ 2) \qquad \beta^1 = 0$$
$$\nu_2 = 2 \qquad (\alpha_1^2 \ \alpha_2^2) \quad = (0 \ , \ 0) \qquad \beta^2 = 2$$

Example 4:

$$u^1 = 1 \quad u^2 = 0 \quad u^3 = 0 \quad u^4 = 1$$
$$v^1 = 0 \quad v^2 = 0$$
$$x^1 \quad x^2 \quad x^3 \quad x^4 \quad x^5$$
$$1 \quad 0 \quad 0 \quad 1 \quad .$$
$$u^1 \quad u^2 \quad u^3 \quad u^4$$

No insertion is needed, neither for v^1 , nor for v^2 .

Example 5:

$u^1 = 1 \quad u^2 = 0 \quad u^3 = 0 \quad u^4 = 2$

$v^1 = 0 \quad v^2 = 0$

x^1	x^2	x^3	x^4	x^5	x^6
1	0	0	2	0	2
u^1	u^2	u^3	β^2	β^1	u^4

One sees that the change of value of u^4 from 1 to 2 provokes the appearance of two insertions. The point is that the decision to insert β^2 depends on the value of the next symbol, which itself depends on the value of the next symbol!

The algorithm given below solves this problem by establishing a first in first out stack of row vectors $s = (s_1, s_2, s_3)$. Normal flag or data characters occupy only the first element of the vector. An inserted character associated with v^t is represented by the triple $(?, \beta^t, \alpha^t_{\nu_t})$.

In the previous two examples, the stack would be

$s(1) = (?, \beta^2, \alpha^2_2) = (?, 2, 0)$

$s(2) = (?, \beta^1_1, \alpha^1_3) = (?, 0, 2)$

$s(3) = (u_4, -, -) = (1, -, -)$ Example 4

$= (2, -, -)$ Example 5

As soon as a normal / character enters the stack, the subroutine "cleanstack" is called. Starting from the end it compares $s(j)$ with $s(j-1)$. If $s_1(j-1) = ?$ and ($s_1(j) = s_2(j-1)$ or $s_3(j-1)$), $s_1(j-1)$ is replaced by $s_2(j-1)$; if $s_1(j-1) = ?$ but

$s_1(j) \neq (s_2(j-1)$ and $s_3(j-1))$, $s(j-1)$ is deleted and the stack collapsed.

Thus in Example 4 the following transformation occurs

$$\begin{matrix}(?,2,0) & (?,2,0) & (1,-,-) \\ (?,0,2) & \rightarrow (1,-,-) & \rightarrow \\ (1,-,-) & & \end{matrix}$$

whereas in Example 5

$$\begin{matrix}(?,2,0) & (?,2,0) & (2,-,-) \\ (?,0,2) & \rightarrow (0,-,-) & \rightarrow (0,-,-) \\ (2,-,-) & (2,-,-) & (2,-,-) \end{matrix}$$

The stack is then emptied to yield part of the output sequence.

Before giving the algorithms we make precise 2 syntactic points:

- -) $(\hat{u}^i,.., \hat{u}^j)$ means the empty set if $i > j$
- -) In a "do loop" of the form "For i := a step b until c do.." no statement is executed if (sign b) a > (sign b) c .

Most of the notation has been explained above or is self evident, except $(\hat{u}^1,...\hat{u}^{t'})$. It represents the output of the decoder. It is mimicked by the encoder. At every instant before $t'' \rightarrow t'' + 1$, these sequences are equal in both encoder and receiver. This, together with the fact that $\hat{u}^1,...,\hat{u}^{t-1}$ is equal to $u^1,...,u^{t-1}$ guarantees unique decodability of the (u^t) sequence. Unique decodability of the (v^t) sequence is guaranteed because the flag to be used after u^t appears if and only if $v^t = 1$.

Coding Algorithm

c1 Set the binary variables w^i , $i > 0$, to 1
v^0 and w^0 to 0

c2 Set the integer variables t , t' , t", stacksize to 0

c3 For j := 0 Step 1 until t' - 1 do

c4 begin

c5 if $(\hat{u}^{t'-j+1},\ldots,\hat{u}^{t'}) = \rho^{t'-j}$ and $w^{t'-j} = 1$

c6 then

c7 begin

c8 $w^{t'-j}$:= 0

c9 stacksize := stacksize + 1

c10 if $v^{t'-j} = 0$

c11 then s(stacksize) := $(?,\beta^{t'-j}, \alpha_{j+1}^{t'-j})$

c12 else

c13 begin

c14 s_1(stacksize) := $\alpha_{j+1}^{t'-j}$

c15 t' := t' - j

c16 cleanstack

c17 end

c18 end

c19 else continue

c20 end

```
c21   t' := t' + 1
c22   if  v^t = 1  and  w^t = 1
c23       then  û^t' := α^t_t'-t
c24       else
c25           begin
c26           t := t + 1
c27           û^t' := u^t
c28           end
c29   stacksize := stacksize + 1
c30   s_1(stacksize) = û^t'
c31   cleanstack
c32   go to  c3
```

Clean Stack

```
cs1   For  i := stacksize  Step -1  until  2  do
cs2        begin
cs3        if  s1(i-1) = ?
cs4            then
cs5                begin
cs6                if  s1(i) = s2(i-1)  or  s1(i) = s3(i-1)
cs7                  then  s1(i-1) := s2(i-1)
cs8                  else
cs9                      begin
cs10                     stacksize := stacksize - 1
cs11                     for  j := i-1  Step 1  until stacksize do
                                                    s(j) = s(j+1)
cs13                     end
cs14               end
cs15           else  continue
cs16       end
cs17  For  i := 1  until stacksize do  x^(t"+i) = s1(i)
cs18  t" := t" + stacksize
cs19  stacksize := 0
```

Decoding Algorithm

d1 Set the binary variables $\hat{v}^1$, $\hat{v}^2$, ... to 0

w^1 , w^2 , ... to 1

d2 Set the integer variables t' , t" to 0

d3 t" := t" + 1

d4 For j := 0 Step 1 until t'-1 do

d5 begin

d6 if $(\hat{u}^{t'-j+1}, \ldots, \hat{u}^{t'}) = \rho^{t'-j}$ and $w^{t'-j} = 1$

d7 then

d8 begin

d9 $w^{t'-j} := 0$

d10 if $x^{t"} = \alpha_{j+1}^{t'-j}$

d11 then

d12 begin

d13 $\hat{v}^{t'-j} := 1$

d14 t' := t' - j

d15 go to d3

d16 end

d17 else

d18 begin

d19 if $x^{t"} = \beta^{t'-j}$

d20 then t" := t" + 1

d21 else continue

d22 end

d23 end

d24 else continue

d25 end

d26 $t' := t' + 1$

d27 $\hat{u}^{t'} := x^{t''}$

d28 go to d3

A program implementing these algorithms has been written in Basic. Data and flag compositions were randomly chosen in a ternary alphabet, for $t=1$ to 100. The output of the coding program was fed into the decoding program which decoded it correctly.

As final remark, we note that there is no reason for all flags to be known in advance. All that is needed is that if the flag corresponding to v^t has length ν_t, the flag corresponding to v^{t+i}, must either be known at time t, or it must be known that its length is greater than $\nu_t - i$, this for $i=1,2,\ldots,\nu_t-1$. This guarantees that the transmission of the flag corresponding to v^t will not be interrupted because of the flag used to signal v^{t+1}.

3. Performance Analysis and Optimization

A. Method

We will investigate in this section the performances of the previous algorithm, and see how they can be optimized; more precisely, we will study how to minimize the total average number of symbols used (flag and inserted characters) because of the possible presence of a flag at time t. We denote by ν the length of this flag, and by p the probability that it will be used.

We have immediately that the average number of symbols used is equal to $p\nu + (1-p)$ Pr (insertion is needed).

We note that $\nu \geq 1$ whereas Pr (insertion is needed) ≤ 1, so that a flag should never be used to indicate an event of probability greater than .5 ; rather a flag should be used to indicate the complement of this event. From now on we will assume $p \leq .5$.

In general, Pr (insertion is needed) is a complicated function of the data process statistics, of the flag composition and of the compositions and probabilities of insertion of the neighboring flags. To avoid this difficulty we use a trick dear to information theorists, i.e. we will average Pr (insertion is needed) over the ensemble of flag compositions and insertion symbols. If the flag is not used after time t, an insertion due to this flag will occur if the symbols $(x^{t''+1}, x^{t''+\nu-1}, x^{t''+\nu})$ are equal to the flag or the antiflag. If their compositions are chosen randomly, the probability of an insertion is $2\,d^{-\nu}$. We will therefore minimize on ν the function $f(p,\nu)$ defined by $f(p,\nu) := p\nu + (1-p)\,2d^{-\nu}$. We will denote by $\nu^{o}(p)$ a value of ν that minimizes $f(p,\nu)$.

We stress that the value of $f(p,\nu)$ is an ensemble average over

the composition of the flag, and that there is no guarantee that a particular flag will perform as well. However, we are sure that for every u and v processes there will be at least a flag composition that will achieve this or a better result. Consequently, we do not claim that $\nu^o(p)$ is the length of the flag which causes the use of the minimum average number of symbols, but only that there is a flag of length $\nu^o(p)$ which will use no more than an average of $f(p,\nu^o(p))$ symbols for each given u and v process.

B. Optimization and Performance Analysis

If we allow ν to take real values, one checks that for p fixed $f(p,\nu)$ is convex in ν, and takes its minimum value $p(\log_d \frac{1-p}{p} + \log_d (2 \log_e d) + \log_d e)$ at $\nu = \log_d \frac{1-p}{p} + \log_d(2 \log_e d)$. Of course, $\nu^o(p)$ must be integer, and by convexity of $f(p,\nu)$ one sees that it must be equal to $\lceil \nu'(p) \rceil$ or $\lfloor \nu'(p)+1 \rfloor$ where $\nu'(p)$ is such that

$$f(p,\nu'(p)) = f(p,\nu'(p) + 1)$$

This equation yields

$$\nu'(p) = \log_d \frac{1-p}{p} + \log_d \frac{2(d-1)}{d}$$

so

$$\nu^o(p) = \left\lceil \log_d \frac{1-p}{p} + \log_d \frac{2(d-1)}{d} \right\rceil$$

$$\text{or} \quad \left\lfloor \log_d \frac{1-p}{p} + \log_d 2(d-1) \right\rfloor$$

Moreover, for every p the value of $f(p,\nu^o(p))$ (which is a piecewise linear function of p (see fig.3.1)) will be lowerbounded by the minimum value on ν of $f(p,\nu)$ and upperbounded by $f(p,\nu'(p))$ thus

$$p(\log_d \frac{1-p}{p} + \log_d (2 \log_e d) + \log_d e) \leq f(p,\nu^o(p))$$

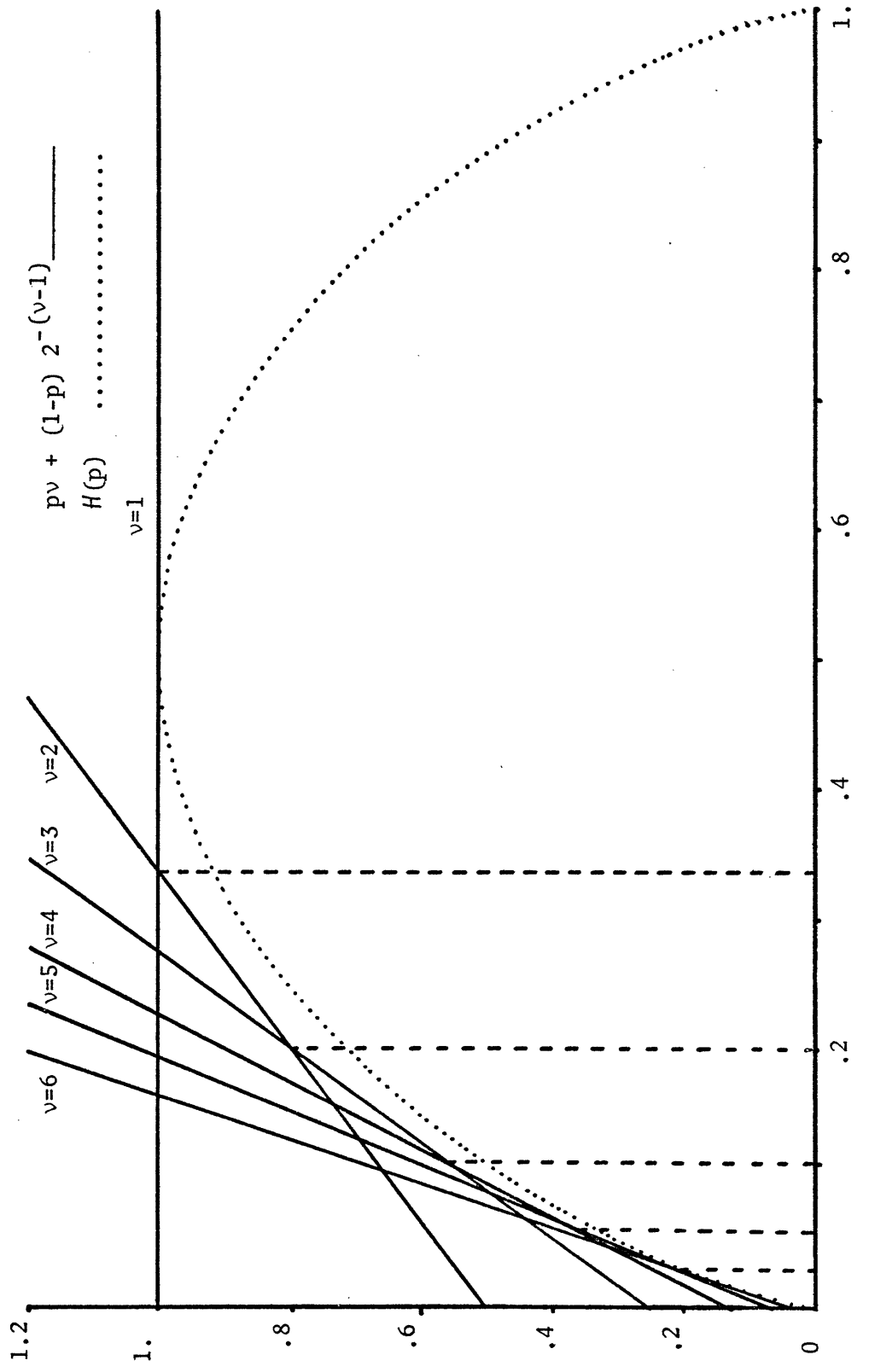


Figure 3.1: Performance of Flag Strategies

$$\leq p\ (\log_d \frac{1-p}{p} + \log_d 2(\frac{d-1}{d}) + \frac{d}{d-1})$$

Specializing these results to the case $d=2$, we see that $\nu^o(p) = \lceil \log_2 \frac{1-p}{p} \rceil$ or $\lfloor \log_2 \frac{1-p}{p} + 1 \rfloor$ (figure 3.2) or equivalently $\nu^o(p)$ is such that

$$\frac{1}{2^{\nu^o(p)} + 1} \leqslant p \leqslant \frac{1}{2^{\nu^o(p)-1} + 1}$$

and the value of $f(p,\nu^o(p))$ is lowerbounded by $p(\log_2 \frac{1-p}{p} + 1.91393)$ and upperbounded by $p(\log_2 \frac{1-p}{p} + 2)$.

It is interesting to compare the average number of bits (counting a symbol as $\log_2 d$ bits) used by this scheme to the binary entropy $H(p) := -p \log_2 p - (1-p) \log_2 (1-p)$ for the following reason: in general $H(p)$ is not a lowerbound to the average number of bits used when a particular flag is utilized, because we are jointly encoding the data and the fact that an event occurs. However if the entropy of the data is $\log_2 d$ bits per symbol, and if

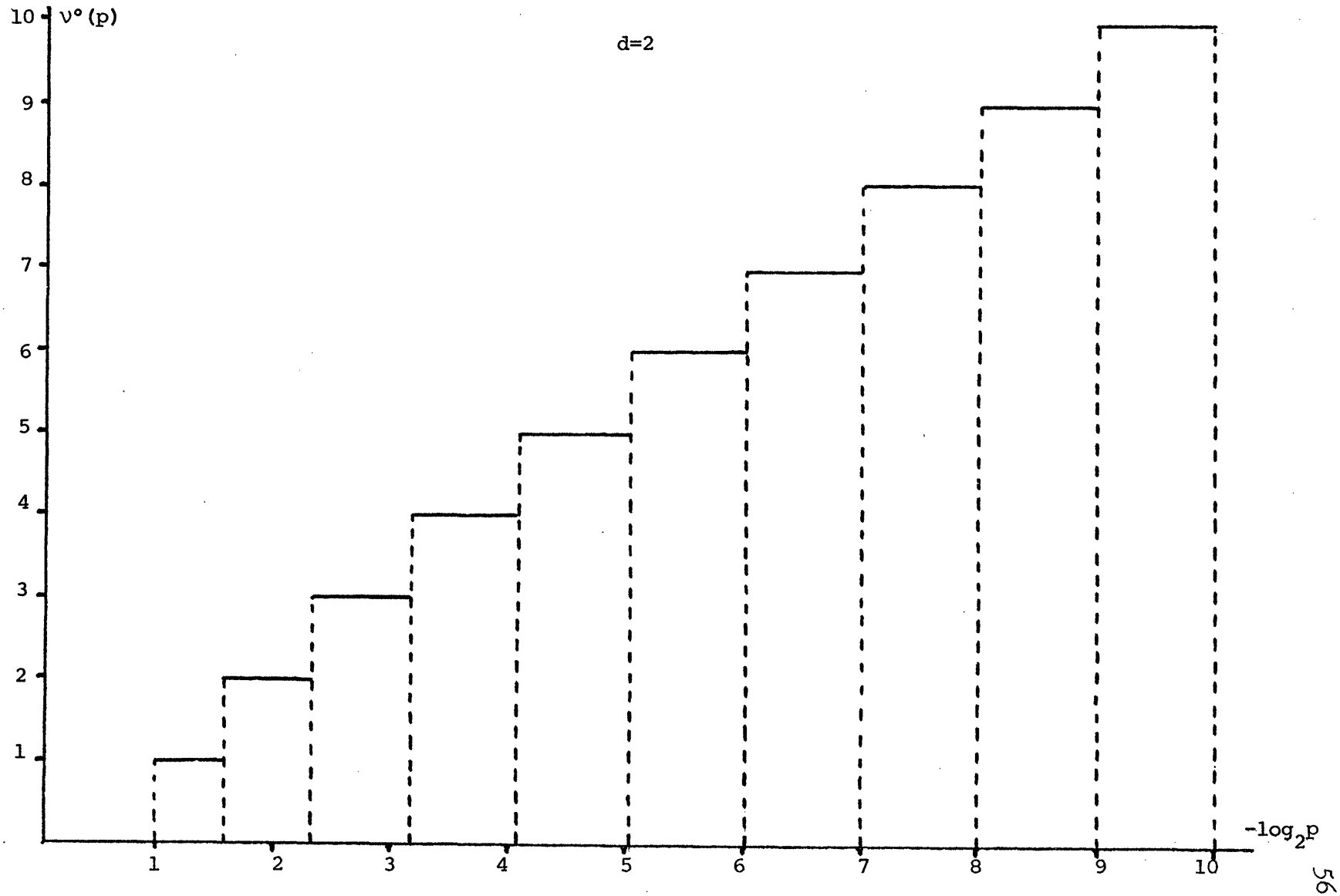
the only event to be signalled is the one we are considering at time t, $H(p)$ is a lowerbound to the average number of bits used by any scheme to indicate the possible occurrence of the event. Because $f(p,\nu)$ does not depend on any hypothesis about the data on the other flags, $H(p)$ is a lowerbound to $(\log_2 d)\ f(p,\nu)$.

From this remark and the bounds developed earlier, one finds immediately:

$$\text{Max}(0,\ p(\log_2(2 \log_e d) + (\log_2 e)) + \log_2 (1-p))$$

$$\leq (\log_2 d)\ f(p, \nu^o(p)) - H(p)$$

FIGURE 3.2 : OPTIMAL FLAG LENGTH AS A FUNCTION OF p



$$\leq p(\log_2 (2\,\tfrac{d-1}{d}) + (\log_2 d)(\tfrac{d}{d-1})) + \log_2 (1-p)$$

$$\leq p(\log_2 (\tfrac{2}{e}\,\tfrac{d-1}{d}) + (\log_2 d)(\tfrac{d}{d-1}))$$

The last inequality uses the fact that $\log_2 (1-p) \leq -p \log_2 e$.

In particular, for $d=2$ we obtain

$$\text{Max}(0,\ p(1.91393) + \log_2 (1-p)) \leq f(p,\ \nu^o(p)) - H(p)$$

$$\leq 2p + \log_2 (1-p)$$

$$\leq .55730\ p$$

For small p , for which $\log_2 (1-p) \simeq -p \log_2 e$,

$$.47123\ p \leq f(p,\ \nu^o(p)) - H(p) \leq .55730\ p\ . \qquad (1)$$

As p goes to 0 , $\dfrac{f(p,\ \nu^o(p)) - H(p)}{p}$ oscillates between $.47123$ and $.55730$. These facts will be used later.

For $d=2$, then, flag schemes are quite efficient, but they deteriorate as d increases: the lowerbound on $f(p,\nu) - H(p)$ increases like $\log_2 (\log_e d)$ while the upperbound increases like $\log_2 d$.

C. Sensitivity Analysis

We will investigate here the sensitivity of the performance of the flag schemes. Two issues are at hand: First, how does a wrong choice of ν degrade $f(p,\nu)$ for a given p ? Second, if p is imperfectly known, how does an error in the estimate of p affect the choice of the flag length? We will treat these problems for $d=2$ only.

The first point is easy to treat. If one uses a flag of length $\nu^o(p) + k$ in place of $\nu^o(p)$ the penalty is equal to $f(p,\ \nu^o(p) + k) - f(p,\ \nu^o(p))$

$$= (k + 2^{-(\nu^o(p)-1)} - 2^{-(\nu^o(p)+k-1)})(p - \frac{1}{2^{\nu^o(p)+k} + 1})\ ;$$

$$k > 0$$

$$(2^{-(\nu^o(p)+k-1)} - 2^{-(\nu^o(p)-1)} - k)\left(\frac{1}{2^{\nu^o(p)+k_{+1}}} - p\right); k < 0$$

These are saw-toothed functions of p , and are plotted in figure 3.3 for k=1 and k=-1 . These expressions are exact but do not give much insight, so we will derive simple upperbounds. We recall that $f(p,\nu)$ is a convex function of ν , and that $\nu'(p) \leq \nu^o(p) \leq \nu'(p) + 1$. Thus, by convexity, for $k \geq 0$

$$f(p,\nu^o(p) + k) \leq \frac{k}{k+\nu'(p)+1-\nu^o(p)} f(p,\nu'(p)+1+k)$$

$$+ \frac{\nu'(p)+1-\nu^o(p)}{k+\nu'(p)+1-\nu^o(p)} f(p,\nu^o(p))$$

and

$$f(p,\nu'(p)+1) \leq \frac{\nu'(p)+1-\nu^o(p)}{k+\nu'(p)+1-\nu^o(p)} f(p,\nu'(p)+1+k)$$

$$+ \frac{k}{(k+\nu'(p)+1-\nu^o(p)} f(p,\nu^o(p))$$

Adding these inequalities, one obtains

$$f(p,\nu^o(p) + k) + f(p,\nu'(p) + 1) \leq f(p,\nu'(p) + 1 + k)$$

$$+ f(p,\nu^o(p))$$

or

$$f(p,\nu^o(p) + k) - f(p,\nu^o(p)) \leq f(p,\nu'(p) + 1 + k)$$

$$-f(p,\nu'(p) + 1)$$

Computing the right hand side member, one gets

$$f(p,\nu^o(p) + k) - f(p,\nu^o(p)) \leq (k + 2^{-k} - 1)\, p$$

Similarly, for $k < 0$, one has

$$f(p,\nu^o(p)+k) \leq \frac{-k}{\nu^o(p)-\nu'(p)-k} f(p,\nu'(k)+k) + \frac{\nu^o(p)-\nu'(p)}{\nu^o(p)-\nu'(p)-k} \cdot$$

$$f(p,\nu^o(p))$$

$$f(p,\nu'(p)) \leq \frac{\nu^o(p)-\nu'(p)}{\nu^o(p)-\nu'(p)-k} f(p,\nu'(p)+k) + \frac{-k}{\nu^o(p)-\nu'(p)-k} \cdot$$

$$f(p,\nu^o(p))$$

Adding these inequalities, one obtains

$$f(p,\nu^o(p)+k) + f(p,\nu'(p)) \leq f(p,\nu'(k)+k) + f(p,\nu^o(p))$$

and thus

$$f(p,\nu^o(p)+k) - f(p,\nu^o(p)) \leq (k + 2^{-(k-1)} - 2)\, p$$

These upperbounds are plotted in figure 3.3 for $k=1$ and $k=-1$. The penalty is always less then $.5p$ if one uses flag length too large by one symbol, whereas it is less than p if the length is too small by one symbol. The same pattern appears for larger $|k|$, the penalty increasing roughly like kp for $k > 0$, but like $2^{-k}p$ for $k < 0$.

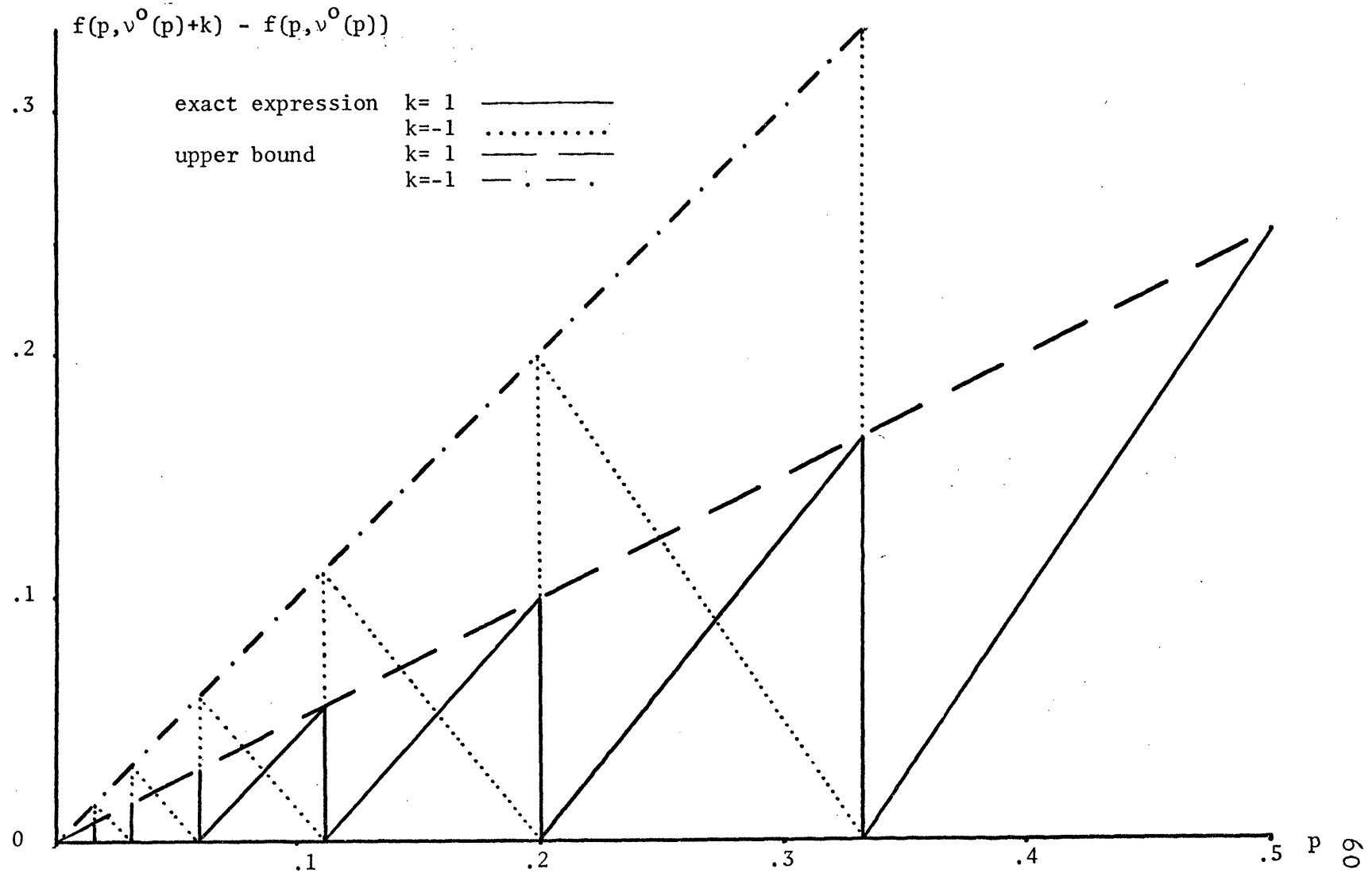
It will be important later to have an upperbound on $f(p,2) - H(p)$ for p between $1/3$ and $1/2$, i.e. in the region where $\nu^o(p) = 1$, because flags of length 1 have some awkward properties, and we will wish to use flags of length 2 instead. We want an upperbound of the form $\alpha p \geq f(p,2) - H(p)$. Because this function is convex, the tightest upperbound of this form will equal it at $p = 1/3$ or $p = 1/2$, so

$$\alpha = \text{Max}\,(3\,(f(1/3,2) - H(1/3)),\ 2\,(f(1/2,2) - H(1/2))$$

$$= .5 \qquad (2)$$

The second point, the sensitivity of the optimal length to an error in the estimate of p is more difficult to assess, due to the discontinuities in $\nu^o(p)$ (figure 3.2). A good rule of thumb is that when p is overestimated or underestimated by about a factor of 2, the resulting flag length is too small or too large by one symbol.

Figure 3.3: Penalty for not Using the Optimal Flag Length



4. Adaptive Flag Stategies to Encode Batch and Message Lengths

We consider the following problem: a batch of messages must be transmitted on a noiseless binary link. We denote by m the random number of messages in a batch, and by $b_1, b_2, \ldots$ the lengths (number of bits) of these messages. Being motivated by the case where a batch would be the set of messages in a busy period of a G/G/1 queue, we model the b_i's as independent identically distributed random variables, but we let the probability of having m messages in a batch depend on the lengths of these messages as follows.

Let $(\Omega, \mathcal{S}, P)$ be a probability space.

$b_1, b_2, \ldots$ be a sequence of measurable functions $b_i : \Omega \to \mathbb{N}^{++}$

m be a measurable function $m : \Omega \to \mathbb{N}^{++}$ $\quad (\mathbb{N}^{++} := \{1,2,\ldots\})$

$\mathcal{B}_i$ be the smallest σ - algebra making b_i measurable

We require the b_i's and m to have the following properties:

the b_i's are independent and have a probability mass function B^m

$E(I_{m(\omega)<i}(\omega) \mid \mathcal{B}_i) = E(I_{m(\omega)<i}(\omega))$

b_1 and m have finite means

In words, the second property says that the knowledge of b_i does not give any information as to whether or not m is smaller than i.

Our problem is that not only must we transmit the messages, but we must also indicate the number of messages in the batch and their lengths. We assume the starting time of the transmission to be known

to the receiver. We will examine different schemes to furnish this information, and we will evaluate their performances. Before doing this, we characterize precisely what we mean, and compute the entropy of the information we are sending.

We want to specify to the receiver which event from the countable set A of disjoint events, $A := \{\{\omega: m(\omega) = k, b_1(\omega) = x_1, \ldots, b_k(\omega) = x_k\}: k, x_1, \ldots, x_k \in \mathbb{N}^{++}\}$, occurred. Note that $\cup A = \Omega$. To obtain a simple expression when computing the entropy of A, it is handy to define the functions R_k, $k \in \mathbb{N}^{++}$, by $R_k: (\mathbb{N}^{++})^k \to \mathbb{R}$.

$$R_k(x_1,..,x_k) = \begin{cases} \dfrac{P(\{\omega: m(\omega)=k,\ b_1(\omega)=x_1,\ldots b_k(\omega)=x_k\})}{\prod_{i=1}^{k} B^m(x_i)} & \text{if } \prod_{i=1}^{k} B^m(x_i) > 0 \\ 13 & \text{otherwise} \end{cases}$$

In words, $R_k(b_1,\ldots,b_k)$ is the conditional probability that the batch contains k messages, given the lengths of the first k messages. We often denote $R_k(b_1,\ldots,b_k)$ by $R_k(\underline{b})$.

It is now easy to write the entropy of A as

$$H(A) = E(-\log_2(R_{m(\omega)}(\underline{b}(\omega)) \prod_{i=1}^{m(\omega)} B^m(b_i(\omega))))$$

$$= E(-\sum_{i=1}^{m(\omega)} \log_2 B^m(b_i(\omega)) - \log_2 R_{m(\omega)}(\underline{b}(\omega)))$$

$$= E(m)\ H(B) - E(\log_2 R_{m(\omega)}(\underline{b}(\omega)))$$

by the theorem proved in Appendix A, which holds because of the conditions imposed earlier on the b_i's and m. $H(B)$ denotes

$$-\sum_{i=1}^{\infty} B^m(i)\ \log_2 B^m(i)$$

This can be rewritten

$$H(A) = E(m)\ H(B) - E\sum_{i=1}^{\infty} R_i(\underline{b}(\omega))\ \log_2 R_i(\underline{b}(\omega))$$

and can be put under the form

$$H(A) = E(m)\ H(B) + E\sum_{i=1}^{\infty} R^c_{i-1}(\underline{b}(\omega))\, H\left(\frac{R_i(\underline{b}(\omega))}{R^c_{i-1}(\underline{b}(\omega))}\right) \qquad (3)$$

with $R^c_0 := 1$

$$R^c_i := 1 - \sum_{j=1}^{i} R_i \qquad i \geq 1$$

This form will be useful later.

We will refer to the second term in (3) as the conditional entropy of the number of messages given their lengths. It is smaller than the entropy of the number of messages which itself is bounded by $E(m)\ H(1/E(m))$, [Gallager, 1968, pp. 25 and 507]. This upperbound is achieved if m is geometrically distributed and independent of the message lengths. Because $E(m)\ H(1/E(m))$ is approximately equal to $\log_2(eE(m))$ the second term in (3) is generally smaller than the first.

We go on to the analysis of some coding schemes to transmit the information in A. From the point of view of minimizing the expected

codeword length, the optimum would be to jointly encode the number and lengths of the messages. This method uses at most one more bit than the theoretical minimum, but is generally infeasible, and can lead to large delays because all messages must be known to the transmitter before the appropriate codeword can be found.

It would be easier to encode separately the number of messages and their lengths, in such a way that a message could be transmitted and decoded correctly before all messages in the batch have been processed by the transmitter. We will examine three strategies in this class, using flags.

The first two strategies have in common that they transmit sequentially each message together with a codeword indicating its length. If the codewords are well chosen, this will require an average number of bits between $E(m)\ H(B)$ and $E(m)(H(B)+1)$.

To indicate the end of a batch, the first strategy transmits a flag of length ν after the last message, and makes appropriate insertions in the other messages. By the usual random coding argument, this will use an average of $\nu+(E(m)-1)2^{-(\nu-1)} = E(m)(\frac{\nu}{E(m)} + (1 - \frac{1}{E(m)})2^{-(\nu-1)}$ bits, so that, as we have seen earlier, the optimum $\nu = \nu^{o}(1/E(m))$ if $E(m) \geq 2$. If $E(m) < 2$, the flag should be used after a message if it is not the last in the batch. We do not consider this case any further. From previous studies this choice of flag length will use at most an average of $\log_2(E(m)-1) + 2$ bits, which lies between $E(m)\ H(1/E(m))$ and $E(m)\ H(1/E(m)) + .55730$. Thus this strategy is efficient if the conditional entropy of the number of

messages given their lengths is close to its maximum.

The second strategy, using variable flag lengths, is efficient under all circumstances. The idea is that at the end of the transmission of the i^{th} message, both transmitter and receiver know $b_1, b_2, \ldots, b_i$, and can compute $R_i(\underline{b})$ and $R_i^c(\underline{b})$. The cost of using a flag of length ν to indicate that message i is the last one in the batch, given there are more than $i-1$ messages in the batch, is $\frac{R_i(\underline{b})}{R_{i-1}^c(\underline{b})}\nu + \frac{R_i^c(\underline{b})}{R_{i-1}^c(\underline{b})} 2 \cdot 2^{-\nu}$. Thus ν should be a function of $b_1, b_2, \ldots, b_i$: $\nu = \nu^o\left(\frac{R_i(\underline{b})}{R_{i-1}^c(\underline{b})}\right)$ if $R_i(\underline{b}) \leq \frac{1}{2} R_{i-1}^c(\underline{b})$, and the strategy should be changed as indicated earlier if $R_i(\underline{b}) > \frac{1}{2} R_{i-1}^c(\underline{b})$. Given $\underline{b}$ this scheme uses less than

$$H\left(\frac{R_i(\underline{b})}{R_{i-1}^c(\underline{b})}\right) + \frac{R_i(\underline{b})}{R_{i-1}^c(\underline{b})} .55730 \quad \text{bits}$$

We will incur this cost if the number of messages in the batch is greater than $i-1$, so the average total number of bits used is less than

$$E \sum_{i=1}^{\infty} \left(R_{i-1}(\underline{b})\, H\left(\frac{R_i(\underline{b})}{R_{i-1}^c(\underline{b})}\right) + R_i(\underline{b})\ .55730\right)$$

$$= E \sum_{i=1}^{\infty} R_{i-1}(\underline{b})\, H\left(\frac{R_i(\underline{b})}{R_{i-1}^c(\underline{b})}\right) + .55730$$

(by comparison with (3))

which is very efficient. Note that if $R_i(\underline{b}) / R_{i-1}^c(\underline{b}) << 1$ for all i, we have from formula (1) that the average number of bits used is larger than

$$E \sum_{i=1}^{\infty} R_{i-1}(\underline{b}) \left(H \frac{R_i(\underline{b})}{R^c_{i-1}(\underline{b})} \right) + .47123 \text{ , approximately.}$$

The only problem with this strategy is that in general it does not meet the requirement of the general flag coding algorithm of Section 2 that if a flag of length ν may be inserted at time t, flags starting at time $t+i$ must either be of length greater than $\nu-i$, or be known to both transmitter and receiver at time t. There are two remedies to this: one is to assume that the message lengths are larger than the longest flag, which often makes sense; the other is to use a special class of flags developped in the next section. They do not have this requirement, but two new problems arise then. The averaging on the flag composition to get $f(p,\nu)$ does not work anymore, and this special class does not contain flags of length one. These difficulties can be overcome: on one hand, if for all j all messages of length j are equally likely, $f(p,\nu)$ will still be an upperbound on the average number of bits used by a flag of length ν from this class; on the other hand we have shown in (2) that the upperbound $f(p,\nu(p)) - H(p) \leq p$ .55730 still holds if one uses flags of length 2 instead of flags of unit length, thus the penalty for not using the optimal length is not unbearable.

To conclude the analysis of this variable flag length algorithm, we note that it can also be used to encode the length of a message. It is sufficient to replace the word message by the word symbol in the previous description, and to use flags from the special class mentioned above. If for all j all messages of length j are equally likely, the conclusion that the average number of bits used will be less than

the entropy of the message length + .55730 still holds.

The third strategy works only in the case where the messages have a variable length. It is based on the observation by [Gallager, 1978] that any Huffman code can be modified so that a 2 symbol prefix, say 00 , is not used, and so that the resulting redundancy is between .41503 and 1. The strategy is as follows: transmit sequentially each message together with a modified Huffman codeword indicating its length. After the last message in the batch, send 00. The number of bits used by this strategy lies between $E(m)\ (H(B) + .41503) + 2$ and $E(m)\ (H(B) + 1) + 2$. This strategy is indeed a flag strategy, so it must be less efficient then the previous optimal algorithm, but it is extremely easy to implement.

5. Desirable Flag Compositions

As we have noted earlier, the algorithm given in Section 2 suffers from the fact that insertions and flags may appear in the middle of other flags, and consequently that the v^t's are not necessarily received in order, and that the flag corresponding to v^t may have to be specified before time t. This complicates the algorithm and removes some freedom in using adaptive flags.

The problem of flags appearing in flags could be solved at the expense of making more insertions, but this can lead to more than one insertion per possible flag use and the analysis of Section 3 breaks down. We will not pursue this approach.

Instead we look at this in the context of Sections 3 and 4, where the important parameter from the user's point of view is the flag length, not the flag composition. We assume that we have a class of flags containing at most one flag of each length and we use only flags from this class in the following algorithms. The main difference between these algorithms and those of Section 2 is that flags are inserted at once (c'19, c'20, c'21) whereas in Section 2 a check was made between flag symbols to see if insertions were needed. Thus here no flags or insertions will appear in flags. Of course these algorithms will not work with all classes; we say that a class is allowable if the compositions of the flags in the class are such that the decoding algorithm yields the correct output for all associations of flags in the class with v^t.

Coding Algorithm

c'1 Set the integer variables t and t" to 0 .

c'2 Set the integer variable i to -1

c'3 For j := 0 Step 1 until i do

c'4 begin

c'5 if $(u^{t-j+1},\ldots,u^{t}) = \rho^{t-j}$ and $u^{t+1} = \beta^{t-j}$ or $\alpha_{\nu_{t-j}}^{t-j}$

c'6 then

c'7 begin

c'8 t" := t" + 1

c'9 $x^{t"} := \beta^{t-j}$

c'10 i := j - 1

c'11 end

c'12 else continue

c'13 end

c'14 t := t + 1

c'15 t" := t" + 1

c'16 $x^{t"} := u^{t}$

c'17 i := i + 1

c'18 if $v^{t} = 1$

c'19 then

c'20 begin

c'21 for j=1 Step 1 until i do

c'22 begin

c'23 if $(u^{t-j+1},\ldots,u^{t}) = \rho^{t-j}$ and $\alpha_1^t = \beta^{t-j}$ or $\alpha^{t-j}_{\nu_{t-j}}$

c'24 then

c'25 begin

c'26 t" := t" + 1

c'27 $x^{t"} := \beta^{t-j}$

c'28 end

c'29 else continue

c'30 end

c'31 for j=1 Step 1 until ν_t do

c'32 begin

c'33 t" := t" + 1

c'34 $x^{t"} := \alpha_j^t$

c'35 i := -1

c'36 end

c'37 end

c'38 else continue

c'39 go to c'3

Decoding Algorithm

d'1 Set the integer variables t and t" to 0

d'2 Set the integer variable i to -1

d'3 Set the binary variables $\hat{v}^i$, $i \geq 1$ to 0

d'4 t" := t" + 1

d'5 For j := 0 Step 1 until i do

d'6 begin

d'7 if $(\hat{u}^{t-j+1},\ldots,\hat{u}^{t}) = \rho^{t-j}$

d'8 then

d'9 begin

d'10 if $x^{t''} = \beta^{t-j}$

d'11 then

d'12 begin

d'13 i := j-1

d'14 t" := t"+1

d'15 end

d'16 else

d'17 begin

d'18 if $x^{t''} = \alpha^{t-j}_{\nu_{t-j}}$

d'19 then

d'20 begin

d'21 $\hat{v}^{t-j}$:= 1

d'22 t : t-j

d'23 i := -1

d'24 t" := t"+1

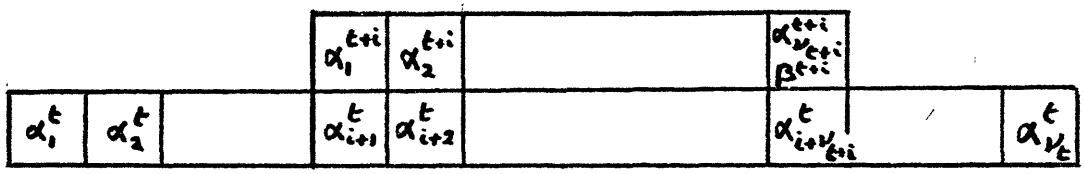
d'25 end

```
d'26                                    else  continue
d'27                                end
d'28                            end
d'29                        else continue
d'30  t := t+1
d'31  û^t := x^t''
d'32  i := i+1
d'33  go  to  d'4
```

Note that these algorithms are simpler than those given in Section 2, and the flags are inserted and received in order. This explains the role of i : if the last flag or insertion were sent at time $t-i$, it is useless to search for a root in lines c'4, c'15 and d'6 past $t-i+1$. Thus the presence of i limits the scope of the search, and makes sure that at most one insertion occurs for each flag.

We will now look at conditions for classes to be allowable. Without precautions, problems can arise in two cases because part of a flag may be misinterpreted as another flag or an insertion

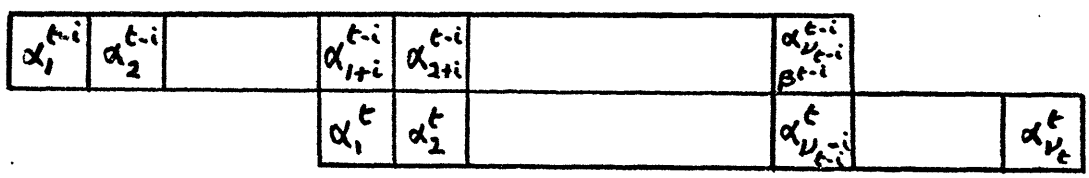
Case a)



if $(\alpha^t_{i+1} \ldots \alpha^t_{i+\nu_{t+i}}) = (\alpha_1^{t+i}, \ldots, \alpha^{t+i}_{\nu_{t+i}} \text{ or } \beta^{t+i})$, $i \geq 1$, $i+\nu_{t+i} \leq \nu_t$

the receiver may detect that a flag has been used at time $t+i$, or β^{t+i}, when the flag is used at time t . The same problem occurs in

Case b)



$(\alpha^t_1 \ldots \alpha^t_{\nu_{t-i}-i}) = (\alpha^{t-i}_{1+i}, \ldots, \alpha^{t-i}_{\nu_{t-i}} \text{ or } \beta^{t-i})$, $i \geq 1$, $1 < \nu_{t-i}-i < \nu_t$

If the flag compositions are such that these cases never arise, all flags

and insertions will be correctly recognized, and thus also the data. So a class is allowable if and only if cases a) and b) cannot occur. From this we proceed to prove three results:

1) We give explicit conditions for a class containing only one flag to be allowable, and we determine the number of such classes.
2) We show that no class containing a flag of length 1 and another flag is allowable if $d=2$, but many exist if $d > 2$.
3) We prove that if $d=2$ there are only two kinds of allowable classes containing a flag of length 2.

To derive the first result, we note that if only one flag, (say $(\alpha_1 \ldots \alpha_j)$ with β being the possible insertion) is allowed in a class, situation a) never occurs while situation b) will not occur for $j \leq 2$ or if the following $j-2$ inequalities are verified

$$\begin{aligned} (\alpha_1 , \alpha_2 , \ldots , \alpha_{j-1}) &\neq (\alpha_2 , \ldots , \alpha_j \text{ or } \beta) \\ (\alpha_1 , \alpha_2 , .. , \alpha_{j-2}) &\neq (\alpha_3 , .. , \alpha_j \text{ or } \beta) \\ (\alpha_1 , \alpha_2) &\neq (\alpha_{j-1} , \alpha_j \text{ or } \beta) \end{aligned} \tag{4}$$

The i^{th} condition may be interpreted as "the flag does not have period i", because if it is not true, $\alpha_1 = \alpha_{1+i}$

$$\alpha_2 = \alpha_{2+i}$$

$$\alpha_{j-i} = \alpha_j \text{ or } \beta$$

A flag satisfying all these conditions will be called <u>strong</u>; another flag will be called <u>weak</u>. To check if a flag is strong, it is enough to check the last $\lfloor \frac{j-1}{2} \rfloor$ conditions, for if a flag has period i,

$1 \le i \le \lfloor \frac{j}{2} \rfloor - 1$, it has also period mi for some mi $\in \{\lfloor \frac{j}{2} \rfloor, \ldots, j-3, j-2\}$.

It is of academic interest to know the number of strong flags of length j . We will compute how many among the d^{j-1} possible roots satisfy (4) once α_j and β have been chosen.

For pedagogical reasons we start by determining the number of roots such that

$$(\alpha_1 , \ldots , \alpha_i) \neq (\alpha_{j-1} , \ldots , \alpha_{j-1}) , \quad i=1,2,\ldots j-2 \qquad (5)$$

These roots will be called <u>strong</u> and their number denoted by $d^{j-1}\gamma(j-1)$; the other roots will be called <u>weak</u>. If a root is weak, let i_o be the least i for which (5) does not hold. Then as we have seen, $i_o \le \lceil \frac{j-2}{2} \rceil$. Then $(\alpha_1,\ldots,\alpha_{i_o}) = (\alpha_{j-i_o},\ldots,\alpha_{j-1})$ and is a strong root of a flag of length i_o+1 . For every such root there will be d^{j-1-2i_o} distinct weak roots of length $j-1$ ($\alpha_{i_o+1},\ldots,\alpha_{j-1-i_o}$ can take all possible values). Thus the number of weak roots of length $j-1$ is equal to

$$d^{j-1}(1 - \gamma(j-1)) = \sum_{i=1}^{\lfloor \frac{j-1}{2} \rfloor} d^i \gamma(i) d^{j-1-2i}$$

or

$$\gamma(k) = 1 - \sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} \frac{\gamma(i)}{d^i} \qquad (6)$$

We see that $\gamma(2k) = \gamma(2k+1) = 1 - \sum_{i=1}^{k} \frac{\gamma(i)}{d^i}$ and that γ is a decreasing non negative function of k , thus it has a limit, $\gamma(\infty)$ say, as k increases. We will bound $\gamma(\infty)$, and show that it is positive.

From (6) one finds

$$\gamma(4k+2) = 2 - (1 + \frac{1}{d}) \sum_{i=0}^{k} \frac{\gamma(2i)}{d^{2i}}$$

with $\gamma(0) := 1$

so $$\gamma(\infty) = \gamma(4k+2) - (1 + \frac{1}{d}) \sum_{i=k+1}^{\infty} \frac{\gamma(2i)}{d^{2i}}$$

and because γ is a decreasing function

$$\gamma(\infty) < \gamma(4k+2) - (1 + \frac{1}{d})\, \gamma(\infty) \cdot \sum_{i=k+1}^{\infty} \frac{1}{d^{2i}}$$

$$> \quad " \qquad " \quad \gamma(2k+2) \quad "$$

Thus

$$\gamma(4k+2) - \frac{1}{(d-1)\, d^{2k+1}}\, \gamma(2k+2) < \gamma(\infty) < \gamma(4k+2) - \frac{1}{(d-1)\, d^{2k-1+1}}\, \gamma(4k+2) \tag{7}$$

In particular, for $k=0$, using the fact that $\gamma(2) = 1 - \frac{1}{d}$

$$0 < 1 - \frac{1}{d} - \frac{1}{d^2} < \gamma(\infty) < 1 - \frac{1}{d} - \frac{1}{d^2}\left(1 - \frac{1}{d^2-d+1}\right)$$

These bounds are extremely tight for $d >> 1$.

We are grateful to Prof. Massey for pointing out that [Nielsen, 1973] obtained by a similar method but in a different context the same expression for $\gamma(i)$, and the same lowerbound for $\gamma(\infty)$. A strong root is called there bifix-free. Tables of numerical values are also given; in particular for $d=2$,

$\gamma(0) = 1$

$\gamma(2) = .5$

$\gamma(4) = .375$

$\gamma(6) = .3125$

$\gamma(8) = .2881$

$\gamma(\infty) = .2678$ whereas from (7) with $k=2$

$.2675 < \gamma(\infty) < .2690$

Now that we have seen the mechanics of the proof, we attack the problem of finding the number of strong flags of length j, terminating with a given α_j and using a given β as possible insertion. We define this number as $d^{j-1}\delta(j)$.

If a flag is weak, there is a smallest $i \in \{2,3,\ldots,1 + \frac{j-1}{2}\}$ such that

$$(\alpha_1,\ldots,\alpha_i) = (\alpha_{j+1-i},\ldots,\alpha_j \text{ or } \beta)$$

On the other hand from every strong flag of length i one can build $2\ d^{j-2i}$ distinct weak flags of length $j \geq 2i$, say $(\alpha_1',\ldots,\alpha_j')$ by choosing $\alpha_1' = \alpha_1,\ldots,\alpha_i' = \alpha_i$ or β, $\alpha_{j-i+1}' = \alpha_i,\ldots,\alpha_j' = \alpha_i$, and choosing $\alpha_{i+1},\ldots,\alpha_{j-i}$ arbitrarily. From every strong flag of length i such that $\alpha_1 = \alpha_i$, one can obtain the weak flag $(\alpha_1,\ldots,\alpha_{i-1},\alpha_i,\alpha_2,\ldots,\alpha_i)$ of length $2i-1$. Noting, by induction on i, that the fraction of strong flags of length i that have $\alpha_1 = \alpha_i$ is $2/d$, we can write in general

$$d^{j-1}(1 - \delta(j)) = \sum_{i=2}^{1+\left\lfloor\frac{j-1}{2}\right\rfloor} 2\, d^{j-2i}\, d^{i-1}\delta(i)$$

thus

$$\delta(j) = 1 - 2 \sum_{i=2}^{\left\lceil\frac{j}{2}\right\rceil} d^{-i}\delta(i)$$

As was the case for γ, δ is a non increasing function of i; $\delta(2j) = \delta(2j-1) = 1 + \frac{2}{d} - 2 \sum_{i=1}^{j} d^{-i}\delta(i)$ where $\delta(1) := 1$.

We can thus write

$$\delta(4k+3) = 1 + \frac{2}{d} - 2(1 + \frac{1}{d}) \sum_{i=0}^{k} d^{-(2i+1)}\delta(2i+1)$$

and

$$\delta(\infty) := \lim_{i\to\infty} \delta(i) = \delta(4k+3) - 2(1+\frac{1}{d}) \sum_{i=k+1}^{\infty} d^{-(2i+1)} \delta(2j+3) \quad \forall k \geq 0$$

As before this permits bounding $\delta(\infty)$:

$$\delta(4k+3) - \frac{1}{d^{2k+2}(d-1)/2}\,\delta(2k+3) < \delta(\infty) < \delta(4k+3) - \frac{1}{1 + d^{2k+2}(d-1)/2} \cdot \delta(4k+3) \quad k \geq 0$$

Using the fact that $\delta(3) = 1 - 2/d^2$:

$$(1 - \frac{2}{d^2})(1 - \frac{1}{d^2(d-1)/2}) < \delta(\infty) < (1 - \frac{2}{d^2})(1 - \frac{1}{1 + d^2(d-1)/2})$$

Of course for the binary case $\delta(i+1) = \gamma(i)$.

This concludes the analysis of classes containing only one flag. To show the second result mentioned above, note that situation a) cannot be avoided if $d=2$ and if the flag used at time t has length $j > 1$ while the flag used at time $t+1$ has unit length. On the other hand, if $d=3$ and the flags are 0, 02, 022 etc. with "1" being eventually inserted, situations a) and b) never occur.

We prove now the third result: suppose that $d=2$ and that a class contains a flag of length 2 and other flags. If the root of the flag of length 2 is "0" , situation a) is avoided only by having all symbols in the other flags, except the first and the last, be equal to "1" . Because the flags must be strong, the first symbol must be different from the penultimate. Thus we conclude that the root of a flag must have the form $(0,1,1,\ldots,1)$. If the root of the flag of length 2 is "1" , the same conclusion arises, with all "0"s replaced by "1" , and conversely. In both cases, the last

symbol can be chosen freely independently in all flags. One checks that these classes are allowable.

Chapter 4

Encoding Message Starting Times

1. Introduction

We consider in this chapter a seemingly trivial problem, which was mentioned briefly in Section 2 of Chapter 2 . We will not be able to solve it completely, but we will gain some insight into the peculiarities of making information theoretic and coding theoretic statements in a queuing environment.

The model is the following: an asynchronous memoryless stationary source emits messages which are stored in an infinite buffer and transmitted over a noiseless synchronous binary link with a capacity of 1 bit per unit of time. We assume that the interemission times and message lengths are mutually independent random variables and that each message contains a "codeword" indicating its length. By this we mean that if the receiver knows when a message starts it will be able to detect the end of the message from the information provided by the message itself. This can be done by prefixing a message with a codeword indicating its length, or by using flags as explained in Chapter 3 , or simply by using messages that are codewords from a prefix condition code, as in Chapter 2 . We denote an interarrival (service) time by a (b) and by A (B) its probability distribution function, and assume $Ea > Eb > 0$.

2. Discussion of the Problem

Because the arrivals and lengths are random, it may happen that the buffer becomes empty. The line being synchronous, something must still be sent out, and the receiver must be able to distinguish these idle bits from the data bits. From another point of view, this is equivalent to recognizing when the line becomes busy, i.e. detecting the message starting times. There are many possible strategies to do this, the most obvious one being to transmit "0" 's when the line is idle, and prefix every message with a "1" . Naturally one asks which is the "best" strategy. We should first agree on the meaning of "best."

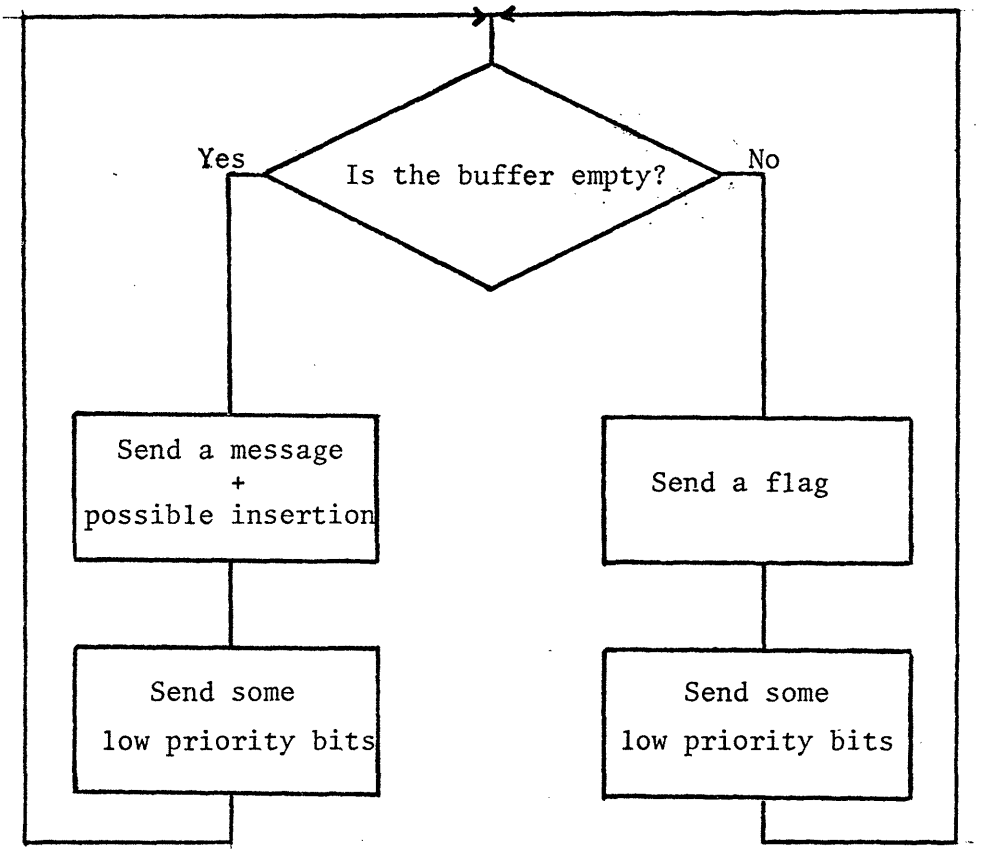
If we define as protocol bit a bit which is not a message bit (in the previous example, the idle bits "0" and the prefix bits "1" would be the protocol bits), it seems reasonable to find the strategy which minimizes the average number of protocol bits per message, i.e. the limit (if it exists and is constant with probability one) as the time goes to infinity of the number of protocol bits sent to the number of message arrivals. Unfortunately this criterion is most useless, for all strategies resulting in a stable system have the same average number of protocol bits per message, and this number is equal to Ea - Eb .This is so because if the system is stable, the law of large numbers says that the average total number of bits per message is Ea , and Eb of these are message bits.

This result is thus trivial, although surprising at first sight. Its information theoretic meaning is that although the amount of information carried during an idle period may be small, it cannot be encoded

efficiently. To give more sense to the concept of protocol bit, we can do the following: suppose that we have at our disposition an infinite reserve of "low priority" bits (this could represent some kind of service information) that we can transmit when we wish. Thus there is no reason for the line to be idle, but we may still need protocol bits (defined as bits that are not data nor low priority bits) to differentiate between the two other kinds of bits. Note that, as before, for a stable system, the expected number of protocol bits per message plus the expected number of low priority bits per message equals $Ea - Eb$. We can now ask the question: what is the infimum of the average number of protocol bits per message? The answer is 0, and this can be approached by the following strategy: send ξ (meant to be large) low priority bits, then a codeword indicating the number n of message arrivals since the last such codeword has been sent, then the n messages. Repeat the process. The average number of protocol bits per message will be equal to the expected codeword length divided by En. If the codewords are well chosen, the expected codeword length will be smaller than $(En+1)H(1/(En+1))+1$ [Gallager, 1968, p. 507], thus the average number of protocol bits per message is smaller than $(1+1/En)H(1/(En+1))+1/En$. Clearly, as ξ goes to ∞ so does En, thus the average number of protocol bits per message goes to zero. The drawback of this strategy is that the average message waiting time goes to infinity as ξ increases.

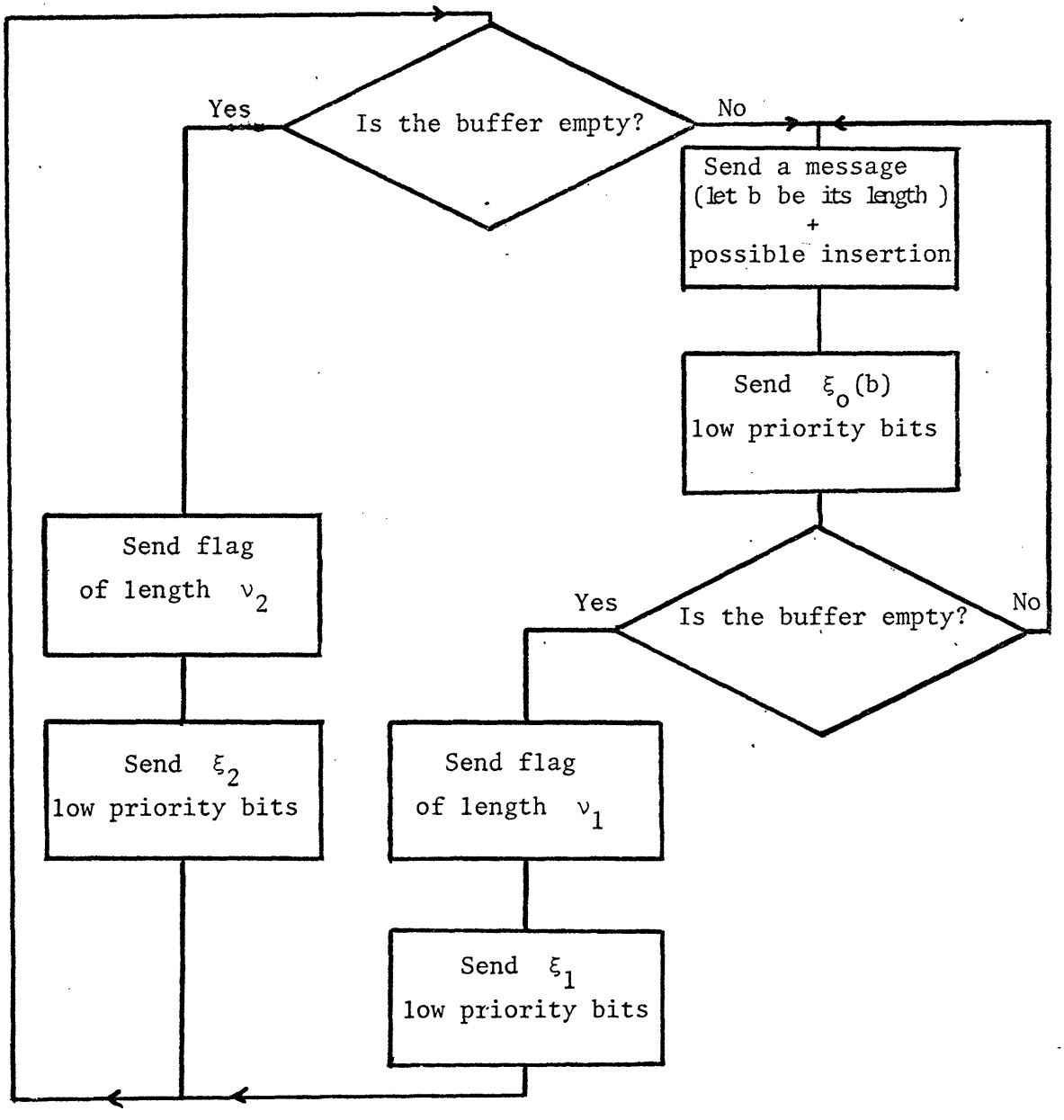
A meaningful problem would thus be to find a coding scheme minimizing the average message waiting time for a given average number of protocol bits per message. We are unable to solve this pro-

blem, or even to lowerbound the expected waiting time. We will be content to study the following class of flag strategies:



Ideally we should let this scheme be adaptive, i.e. we should allow flag and low priority bit sequence lengths to be functions of the times of reception and lengths of the previous messages, flags and low priority bit sequences. This is known to the receiver. In light of the results of Chapter 3 and of the fact that this scheme sends flags when the buffer is empty, which has a favorable influence on the message

waiting time, one expects that the best scheme from this class must be nearly optimal. Unfortunately this is still too complex to analyze, and here we restrict ourself to the following scheme:



We have thus removed much of the variability of the flag and low priority bit sequence lengths, allowing only the length $\xi_o(b)$ of the low

priority bit sequence immediately following a message to be a function of that message length and allowing the first flag and low priority bit sequence in an idle period to be different from the others in the idle period. We assume that with probability one a message is longer than Max (ν_1,ν_2); otherwise some messages cannot be considered as being received when they are fully transmitted!

To be able to obtain analytical results we will also model the arrival process as Poisson. The analysis will proceed in steps: in Section 3 we will study a general queueing model whose parameters will be identified in Section 4 so that it represents the flag strategy we want to examine. The main results will be given in Section 5, while the optimal function $\xi_o(b)$ will be looked at in Section 6. We will give numerical results in Section 7.

3. M/G/1 Queues with Overhead

A. Introduction

We analyze here the following problem: arrivals in a queue follow a Poisson process with rate $\lambda := 1/Ea$ and the service times have distribution B'. The first customer in a busy period suffers an extra delay with distribution F_1, while the services of the other customers are increased by a random amount with distribution F_2. We assume that the interarrival times, service times and extra delays are all independent. We will study the stationary distribution of the number of customers in the queue, the mean waiting time, and the joint distribution of the busy period length and of the number of customers in the busy period.

B. Stationary Distribution of the Number of Customers in the Queue

Let x_n be the number of customers in the queue right after the n^{th} customer has left the system and let Π_n be the probability mass function of x_n. We have the following recursive relation between the x_n's: $x_n = x_{n-1}$ + (number of arrivals during n^{th} service) - $I_{x_{n-1}>0}$ It is well known that the number of arrivals during the n^{th} service has a generating function equal to $F_1^*(\lambda-\lambda z)B'^*(\lambda-\lambda z)$ or $F_2^*(\lambda-\lambda z)B'^*(\lambda-\lambda z)$, depending on whether or not $x_{n-1} = 0$. Denoting by Π_n^* the z-transform of Π_n, we have immediately

$$\Pi_n^*(z) = \Pi_{n-1}^*(0)\ F_1^*(\lambda-\lambda z)\ B'^*(\lambda-\lambda z) + (\Pi_{n-1}^*(z) - \Pi_{n-1}^*(0))\ .$$
$$F_2^*(\lambda-\lambda z)\ B'^*(\lambda-\lambda z)\ \frac{1}{z}$$

By classical methods [Karlin, 1975, pp. 96-102] one sees that the system is ergodic if and only if $\lambda(Eb + Ef_2) < 1$; in this case the z transform Π^* of the stationary distribution Π must be equal to

$$\Pi^*(z) = \frac{\Pi^*(0)\ B'^*(\lambda-\lambda z)(z\ F_1^*(\lambda-\lambda z) - F_2^*(\lambda-\lambda z))}{z - F_2^*(\lambda-\lambda z)\ B'^*(\lambda-\lambda z)}$$

and $\Pi^*(1)$ must equal 1 , so, using L'Hopital's rule as $z \to 1$,

$$\Pi^*(0) = \frac{1 - \lambda E\ (b'+f_2)}{1 + \lambda E(f_1-f_2)}$$

thus

$$\Pi^*(z) = \left((1 - \lambda E(b'+f_2))\ \frac{B'^*(\lambda-\lambda z)\ F_2^*(\lambda-\lambda z)(z-1)}{z - F_2^*(\lambda-\lambda z)\ B'^*(\lambda-\lambda z)}\right) \cdot$$

$$\left(\frac{1}{1 + \lambda E(f_1-f_2)}\ \frac{z\ F_1^*(\lambda-\lambda z) - F_2^*(\lambda-\lambda z)}{F_2^*(\lambda-\lambda z)(z-1)}\right)$$

If $f_1^* = F_2^*$, the second factor in brackets ie equal to one, and we obtain the Pollaczek formula for M/G/1 queues with service distribution $F_2^*\ B'$.

Π is also the stationary distribution of the number of customers in the queue at an arrival time [Kleinrock, 1975, p. 176], and, because the arrival process is Poisson, also at a random time.

C. Average Delay

Combining the remark at the end of the last section with Little's formula [Little, 1961], one obtains by differentiating $\Pi^*(z)$ the following formula for the average message delay, where the delay is defined as the difference between the times of service completion and message arrival:

$$E(d) = Eb' + Ef_2 + \tfrac{1}{2}\frac{\lambda(Eb'^2 + 2Ef_2\, Eb' + Ef_2^2)}{1 - \lambda(Eb' + Ef_2)} + \frac{1}{1 + \lambda(Ef_1 - Ef_2)}$$

$$((Ef_1 - Ef_2)(1 - \lambda Ef_2) + \tfrac{1}{2}\lambda(Ef_1^2 - Ef_2^2))$$

$$= Eb' + \tfrac{1}{2}\frac{\lambda(Eb'^2 + 2Ef_2\, Eb' + Ef_2^2)}{1 - \lambda(Eb' + Ef_2)} + \frac{1}{1 + \lambda(Ef_1 - Ef_2)}$$

$$(Ef_1 + \tfrac{1}{2}\lambda(Ef_1^2 - Ef_2^2)) \qquad (1)$$

D. Busy Periods

Denote by g and m respectively the length of and the number of customers served in a busy period. We will characterize the function $GM^*(s,z) := E[z^m e^{-sg}]$.

It is well known [Kleinrock, 1975] that if $F_1 = F_2$, $GM^*(s,z)$, there denoted $GM_a^*(s,z)$, satisfies the relation

$$GM_a^*(s,z) = zF_2^*(s + \lambda - \lambda GM_a^*(s,z))\, B'^*(s + \lambda - \lambda GM_a^*(s,z)) .$$

We will express $GM^*(s,z)$ in terms of $GM_a^*(s,z)$ as follows: let b_1 and f_1 be the lengths of the first service and extra delay, and n_1 be the number of arrivals during b_1 and f_1 . We then have

$$E[z^m e^{-sg} | b_1, f_1, n_1] = ze^{-s(f_1+b_1)}(MG_a^*(s,z))^{n_1}$$

because the n_1 arrivals will generate n_1 independent busy periods characterized by GM_a . Averaging on n_1, b_1, and f_1, one obtains

$$GM^*(s,z) = z\, F_1^*(s + \lambda - \lambda GM_a^*(s,z))\, B'^*(s + \lambda - \lambda GM_a^*(s,z))$$

$$= \frac{F_1^*(s + \lambda - \lambda GM_a^*(s,z))}{F_2^*(s + \lambda - \lambda GM_a^*(s,z))}\, GM_a^*(s,z)$$

One obtains easily the moments

$$E(g) = \frac{Ef_1 + Eb'}{1 - \lambda(Ef_2+Eb')}$$

$$E(m) = \frac{1 + \lambda(Ef_1 - Ef_2)}{1 - \lambda(Ef_2+Eb')} \qquad (2)$$

4. Identification of B', F_1 and F_2

In the analysis of the queueing system of the previous section we have obtained expressions that involve the Laplace-Stieltjes transform B'^*, F_1^* and F_2^*. We will identify them from the previous description of the coding scheme.

B' will be the probability distribution function of $b' := b + \xi_o(b)$, i.e. b' is the sum of the lengths of a message and of the low priority bit sequence that immediately follows it.

f_2 will correspond to the extra delay for a message in the middle of the busy period. In our scheme, f_2 will be equal to 0 or 1, depending on whether or not an insertion is needed. So $F_2^* = 1 + 2^{-(\nu_1-1)}(e^{-s} - 1)$ if the first ν_1-1 bits of all messages are equally likely and

$$Ef_2 = Ef_2^2 = 2^{-(\nu_1-1)} . \qquad (3)$$

It is harder to compute F_1^*. We start by solving the following problem./ Let the times $0,t_1,t_2,\ldots$ form a renewal process, the probability distribution function of t_1 being C_1 ($C_1(0^-)=0$), and the distribution of $t_i - t_{i-1}$ being C_2 ($C_2(0^-)=0$), $i=2,3,\ldots$ At a random time t, independent of the renewal process and with distribution function $1 - e^{-\lambda t}$, $t \geq 0$, a "supervent" occurs. We wish to find the Laplace-

Stieltjes transform of the distribution F_1 of the random variable f_1 defined as follows:

$$f_1 = \min_{\substack{n \\ t_n \geq t}} (t_n - t) + d_1 I_{(t_1 \leq t)} + d_2 I_{(t_1 > t)}$$

where d_1 and d_2 are random variables independent of the renewal process and of t, with distributions D_1 and D_2 respectively. In other words, f_1 is equal to the time between the occurrences of the superevent and the following event, plus a random variable whose distribution is D_1 if the superevent occurs before the first event, and D_2 otherwise.

We have immediately:

$$F_1^*(s) = \Pr(t \leq t_1)\, E[e^{-s(t_1 - t)} \mid t \leq t_1]\, D_1^*(s) + \Pr(t > t_1)\, E[e^{-s \min_{\substack{n \\ t_n \geq t}} (t_n - t)} \mid t > t_1]\, D_2^*(s) \qquad (4)$$

We compute now:

$$\Pr(t > t_1) = \int_0^\infty e^{-\lambda t_1}\, d\, C_1(t_1) = C_1^*(\lambda) \qquad (5)$$

$$E[e^{-s(t_1 - t)} \mid t \leq t_1] = \frac{1}{1 - C_1^*(\lambda)} \int_0^\infty dC_1(t_1) \int_0^{t_1} dt\, \lambda e^{-\lambda t} e^{-s(t_1 - t)}$$

$$= \frac{\lambda}{\lambda - s} \frac{C_1^*(s) - C_1^*(\lambda)}{1 - C_1^*(\lambda)}$$

Similarly, because t is "memoryless,"

$$E[e^{-s(t_n - t)} \mid t_{n-1} < t \leq t_n\ ,\ n > 1] = \frac{\lambda}{\lambda - s} \frac{C_2^*(s) - C_2^*(\lambda)}{1 - C_2^*(\lambda)}$$

The right hand side member is independent of n, given $n > 1$; thus

$$E[e^{-s \min_{t_n \geq t} (t_n - t)} \mid t > t_1] = \frac{\lambda}{\lambda - s} \frac{C_2^*(s) - C_2^*(\lambda)}{1 - C_2^*(\lambda)}$$

Plugging these results into (4), we obtain,

$$F_1^*(s) = \frac{\lambda}{\lambda - s} \left[(C_1^*(s) - C_1^*(\lambda)) D_1^*(s) + \frac{C_1^*(\lambda)}{1 - C_2^*(\lambda)} (C_2^*(s) - C_2^*(\lambda)) D_2^*(s) \right]$$

and by differentiation,

$$Ef_1 = -\frac{1}{\lambda} + Ec_1 + \frac{C_1^*(\lambda)}{1 - C_2^*(\lambda)} Ec_2 + (1 - C_1^*(\lambda)) Ed_1 + C_1^*(\lambda) Ed_2$$

$$Ef_1^2 = \frac{2}{\lambda^2} - \frac{2}{\lambda} Ef_1 + Ec_1^2 + 2Ec_1 Ed_1 + \frac{C_1^*(\lambda)}{1 - C_2^*(\lambda)} (Ec_2^2 + 2Ec_2 Ed_2) + (1 - C_1^*(\lambda)) Ed_1^2 + C_1^*(\lambda) Ed_2^2$$

We will use the fact that

$$Ef_1 + \tfrac{1}{2}\lambda Ef_1^2 = \tfrac{1}{2} \lambda [Ec_1^2 + 2Ec_1 Ed_1 + \frac{C_1^*(\lambda)}{1 - C_2^*(\lambda)} (Ec_2^2 + 2Ec_2 Ed_2) + (1 - C_1^*(\lambda)) Ed_1^2 + C_1^*(\lambda) Ed_2^2]$$

We will need later the fact that

$$\Pr(t > t_n) = C_1^*(\lambda) (C_2^*(\lambda))^{n-1}, \quad n=1,2,... \qquad (6)$$

for the same reason as (5).

In our coding scheme, the "superevent" will be the arrival of a message. C_i will correspond to the distribution of the flag of length ν_i plus the low priority bit sequence of length ξ_i. Thus $C_i^* = e^{-s\nu_i} e^{-s\xi_i}$ $i=1,2$. d_1 and d_2 will be equal to zero, except if an insertion

is needed in the first message of a busy period, so $D_1^* = D_2^* = 1 + 2^{-\nu_2+1}(e^{-s}-1)$, and $Ed_1 = Ed_2 = Ed_1^2 = 2^{-(\nu_2-1)}$, under the usual assumptions.

Thus,

$$Ef_1 = -1/\lambda + \xi_1 + \nu_1 + \frac{e^{-\lambda(\xi_1+\nu_1)}}{1 - e^{-\lambda(\xi_2+\nu_2)}}(\nu_2 + \xi_2) + 2^{-(\nu_2-1)} \qquad (7)$$

and

$$Ef_1 + \frac{\lambda}{2}Ef_1^2 = \tfrac{1}{2}\lambda((\nu_1 + \xi_1)^2 + 2(\nu_1 + \xi_1)\,2^{-(\nu_2-1)} + \frac{e^{-\lambda(\nu_1+\xi_1)}}{1 - e^{-\lambda(\nu_1+\xi_2)}}((\nu_2+\xi_2)^2 + 2(\nu_2+\xi_2)\,2^{-(\nu_2-1)} + 2^{-(\nu_2-1)}) \qquad (8)$$

5. Main Result

Putting together all the results of the previous sections, we obtain a formula for the average message waiting time as a function of $\xi_o(b)$, ξ_1, ξ_2, ν_1 and ν_2: from (1), (3), (7) and (8)

$$Ew = \tfrac{1}{2}\frac{\lambda(2^{-(\nu_1-1)} + 2\,2^{-(\nu_1-1)}(Eb + E\xi_o) + E\,b'^2)}{1 - \lambda(E\,b + E\,\xi_o + 2^{-(\nu_1-1)})} + \tfrac{1}{2}\frac{(\nu_1+\xi_1)^2 + 2(\nu_1+\xi_1)\,2^{-(\nu_2-1)} + \left(\frac{e^{-\lambda(\nu_1+\xi_1)}}{1 - e^{-\lambda(\nu_2+\xi_2)}}\right)\cdot}{\xi_i + \nu_i + \frac{e^{-\lambda(\nu_1+\xi_1)}}{1 - e^{-\lambda(\nu_2+\xi_2)}}(\nu_2+\xi_2) +} \cdots$$

$$\frac{((\nu_2+\xi_2)^2 + 2(\nu_2+\xi_2)\, 2^{-(\nu_2-1)}) + 2^{-(\nu_2-1)} - 2^{-(\nu_1-1)}}{2^{-(\nu_2-1)} - 2^{-(\nu_1-1)}} \quad (9)$$

We also obtain the average number of low priority bits per message, which, after a little moment of reflection, must be equal to

$$E\xi_o + \frac{E(\text{number of low priority bits in an idle period})}{Em}$$

where m has been defined as the number of messages in a busy period. By substituting (3) and (7) in (2) one obtains

$$Em = \frac{\lambda(\xi_1 + \nu_1 + \frac{e^{-\lambda(\nu_1+\xi_1)}}{1 - e^{-\lambda(\nu_2+\xi_2)}}(\xi_2+\nu_2) + 2^{-(\nu_2-1)} - 2^{-(\nu_1-1)})}{1 - \lambda(E\,b + E\,\xi_o + 2^{-(\nu_1-1)})}$$

In the parlance of Section 4 , the number of low priority bits in an idle period is equal to $\xi_1 + i\,\xi_2$ if the superevent occurs between t_i and t_{i+1} . Thus its expected value is equal to

$$\xi_1 + \frac{e^{-\lambda(\nu_1+\xi_1)}}{1 - e^{-\lambda(\nu_2+\xi_2)}}\,\xi_2$$

as can be seen by using (6).

The expected number of low priority bits per message is thus equal to

$$E\,\xi_o + \frac{\left(\xi_1 + \frac{e^{-\lambda(\nu_1+\xi_1)}\,\xi_2}{1 - e^{-\lambda(\nu_2+\xi_2)}}\right)(\frac{1}{\lambda} - Eb - E\xi_o - 2^{-(\nu_1-1)})}{\xi_I + \nu_I + \frac{e^{-\lambda(\nu_1+\xi_1)}\,(\xi_2+\nu_2)}{1 - e^{-\lambda(\nu_2+\xi_2)}} + 2^{-(\nu_2-1)} - 2^{-(\nu_1-1)}} \quad (10)$$

What is left to do is to try to minimize $E(w)$ on $\xi_o(b)$, ξ_1 , ξ_2, ν_1 and ν_2 while keeping the expected number of low priority bits per message fixed. In the next section we will gain some insight into the problem of optimizing on $\xi_o(b)$ for $E\xi_o$ fixed. This will reduce the problem to optimizing on $E\xi_o$, ξ_1 , ξ_2 , ν_1 and ν_2 , which will require numerical computations.

6. Optimization of $\xi_o(b)$

We decided to let the length ξ_o of the low priority bit sequence following a message be a function of the length b of this message. Denoting $b + \xi_0(b)$ by b' , we see from formulas (9) and (10) that the average message delay depends on $E\, b'$ and $E\, b'^2$ while the expected number of low priority bits per message depends on $E\, b'$. The question then arises of how $\xi_o(b)$ should be defined so as to minimize $E\, b'^2$ for given $E\, b'$ and B .

We will solve this problem for the case where ξ_o may take non integer values. This will give some insight and a lower bound for the interesting case when ξ_o takes only integer values, which is an infinite dimensional non linear integer programming problem.

We must find

$$\min_{\xi_o(b)} \int_o^\infty (b + \xi_o(b))^2 \, dB(b)$$

subject to the constraints:

$$\xi_o(b) \geq 0$$

$$\int_o^\infty \xi_o(b) \, dB(b) \geq E(\xi_o) > 0 \qquad (11)$$

We start by defining α as the only root of the equation

$$\int_o^x (x-b)\, dB(b) = E(\xi_o)$$

This root exists and is unique and positive because the function $\int_o^x (x-b)dB(b)$ is continuous (left and right derivatives exist everywhere), is equal to 0 at $x=0$, is increasing if it is not equal to 0, and goes to ∞ as x increases.

We have

$$\int_o^\infty (b+\xi_o(b))^2\, dB(b) = \int_o^\alpha (b+\xi_o(b))^2\, dB(b) + \int_\alpha^\infty (b+\xi_o(b))^2\, dB(b)$$

$$\geq \int_o^\alpha (b+\xi_o(b))^2\, dB(b) + \int_\alpha^\infty b^2\, dB(b) + 2\alpha \int_\alpha^\infty \xi_o(b)\, dB(b)$$

by non negativity of ξ_o

$$\geq \frac{\left(\int_o^\alpha b + \xi_o(b)\, dB(b)\right)^2}{\int_o^\alpha dB(b)} + \int_\alpha^\infty b^2\, dB(b) + 2\alpha \int_\alpha^\infty \xi_o(b) dB(b)$$

by the Schwarz inequality

$$= \frac{\left(\alpha \int_o^\alpha dB(b) - E(\xi_o) + \int_o^\alpha \xi_o(b)\, dB(b)\right)^2}{\int_o^\alpha dB(b)} + \int_\alpha^\infty b^2\, dB(b)$$

$$+ 2\alpha \int_\alpha^\infty \xi_o(b)\, dB(b)$$

by the definition of α

$$= \frac{\left(\alpha \int_o^\alpha dB(b) - \int_\alpha^\infty \xi_o(b)\, dB(b)\right)^2}{\int_o^\alpha dB(b)} + \int_\alpha^\infty b^2\, dB(b)$$

$$+ 2\alpha \int_\alpha^\infty \xi_o(b)\, dB(b)$$

$$\geq \alpha^2 \int_o^\alpha dB(b) - 2\alpha \int_\alpha^\infty \xi_o(b)\, dB(b) + \int_\alpha^\infty b^2\, dB(b)$$

$$+ 2\alpha \int_{\alpha}^{\infty} \xi_o(b)\, dB(b)$$

$$= \alpha^2 \int_{o}^{\alpha} dB(b) + \int_{\alpha}^{\infty} b^2\, dB(b)$$

This lower bound is achieved if

$$b + \xi_o(b) = \begin{cases} \alpha & b \leq \alpha \\ b & b > \alpha \end{cases}$$

i.e. if $$\xi_o(b) = \begin{cases} \alpha - b & b \leq \alpha \\ 0 & b > \alpha \end{cases}$$

This ξ_o satisfies (11) with equality because of the definition of α and is thus optimal. This result is intuitively pleasing.

As noted above, the constraint that ξ_o must have integer values makes the problem much more difficult, except if α happens to be an integer. In general, constraint (11) will not be satisfied with equality by an integer solution if $\xi_o(b)$ is a deterministic function of b .

7. Numerical Results

Many of the results of this chapter have limited practical interest. This is due to the fact that generally there are no low priority bits to be sent. However, the analysis of the previous section is relevant as far as the use of flags is concerned. We will briefly consider how the flag lengths should be chosen to minimize the average waiting time when no low priority bits are sent. (Formula (9) with $\xi_0 = \xi_1 = \xi_2 = 0$.)

We recall that we use a flag of length ν_1 to indicate the end of a busy period, while flags of length ν_2 are sent during the residue of the idle periods.

From numerical computations it appears that the choice $\nu_2=2$ is never worse than $\nu_2=1$, and is in fact optimal in light traffic. In heavy traffic the second flag is rarely used, so its optimal length increases somewhat to reduce the probability of an insertion in the first message of a busy period. The effect on Ew is relatively negligeable, as illustrated in Table 4.1.

		$\nu_2=1$	$\nu_2=2$	$\nu_2=3$
$\lambda Eb = .5$	$\nu_1 = 3$			
	Ew =	1.739	1.733	1.975
$\lambda Eb = .95$	$\nu_1 = 9$			
	Ew =	80.971	80.724	80.646

Table 4.1

Influence of ν_2 on Ew

Eb=8 Eb^2=64

The situation is much more complicated as far as ν_1 is concerned. The presence of the expressions $2^{-(\nu_1-1)}$ Eb and $\lambda 2^{-(\nu_1-1)}$ respectively in the numerator and denominator of the first term of the right hand side member of (9) makes the optimal ν_1 an increasing function of Eb and λ. Contrary to the case of ν_2 Ew is quite sensitive to the value of ν_1, especially when the load is heavy:

if Eb = 8 $\quad Eb^2 = 64$

$\nu_2 = 2$

$\lambda Eb = .95$

then Ew = 92.89 for $\nu_1=5$

= 80.72 for $\nu_1=9$

We illustrate in Tables 4.2 and 4.3 the behavior of the optimal value of ν_1 as the load increases for two different message length statistics.

a) Eb = 8 $\quad Eb^2 = 64$

b) Eb = 5 $\quad Eb^2 = 30$

The first case represents the transmission of single characters without special source encoding, whereas the second is representative of the message length statistics when some source coding (see Chapter II) is performed. Note that we did not take into account the effects that occur when flags are longer than messages.

We do not give examples with larger average message length: except in very heavy traffic the improvement in performance brought by the use of optimal length flags do not warrant the increased complexity. It seems more sensible to send "0" 's when the line is idle, and to prefix every message with a "1".

Table 4.2

Optimal ν_1 as a Function of the Load

$Eb=8 \quad Eb^2=64 \quad \nu_2=2$

λ/Eb	optimal ν_1	Ew for optimal ν_1	Ew for $\nu_1=2$	Ew for $\nu_1=10$
.05	3	1.73	1.74	1.95
.10	3	1.99	2.01	2.43
.15	3	2.28	2.31	2.92
.20	3	2.61	2.65	3.43
.25	3	2.97	3.04	3.97
.30	3	3.40	3.50	4.55
.35	3	3.89	4.02	5.18
.40	4	4.45	4.65	5.87
.45	4	5.11	5.41	6.64
.50	4	5.90	6.33	7.52
.55	4	6.87	7.50	8.55
.60	4	8.09	9.00	9.79
.65	5	9.66	11.02	11.34
.70	5	11.69	13.88	13.36
.75	5	14.55	18.23	16.13
.80	6	18.81	25.67	20.24
.85	6	25.83	41.26	27.01
.90	7	39.71	94.71	40.48
.95	9	80.72	∞	80.84

Table 4.3

Optimal ν_1 as a Function of the Load

$Eb=5 \quad Eb^2=30 \quad \nu_2=2$

λ/Eb	optimal ν_1	Ew for optimal ν_1	Ew for $\nu_1=2$	Ew for $\nu_1=10$
.05	3	1.69	1.69	2.05
.10	3	1.89	1.90	2.59
.15	3	2.12	2.14	3.13
.20	3	2.38	2.41	3.66
.25	3	2.67	2.72	4.20
.30	3	3.00	3.09	4.74
.35	3	3.39	3.52	5.30
.40	3	3.85	4.04	5.89
.45	3	4.39	4.66	6.52
.50	4	5.03	5.44	7.22
.55	4	5.79	6.44	8.02
.60	4	6.75	7.76	8.97
.65	4	7.99	9.60	10.15
.70	5	9.62	12.30	11.67
.75	5	11.84	16.71	13.75
.80	6	15.19	25.17	16.83
.85	6	20.61	47.92	21.91
.90	7	31.26	321.00	32.02
.95	9	62.36	∞	62.42

Chapter 5

Encoding of Message Origins

1. Introduction

In the previous chapters we have examined ways to encode the message contents and lengths, and to differentiate between idle and message bits. We will study here how to encode message origins and destinations in a simple case. The model is as follows:

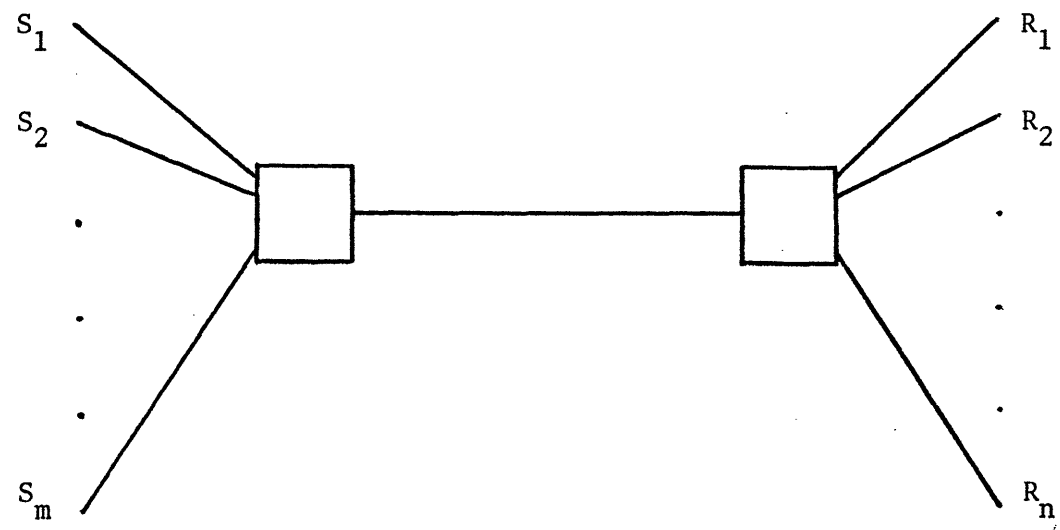


Figure 5.1: The Model

Messages are sent from the asynchronous sources S_i , i=1,2,..m , to a concentrator containing an infinite buffer. From there they are transmitted over a noiseless binary synchronous link to a "deconcentrator" which sends the messages to their destinations, R_i , i=1,2,...n . We observe that in general the destinations must be indicated by the sources to the concentrator, the origins and destinations must be indicated to the deconcentrator, while the origins alone need to be indicated to the

receivers.

To simplify the model, we can associate a virtual source and receiver with each source-receiver pair, as in the following figure.

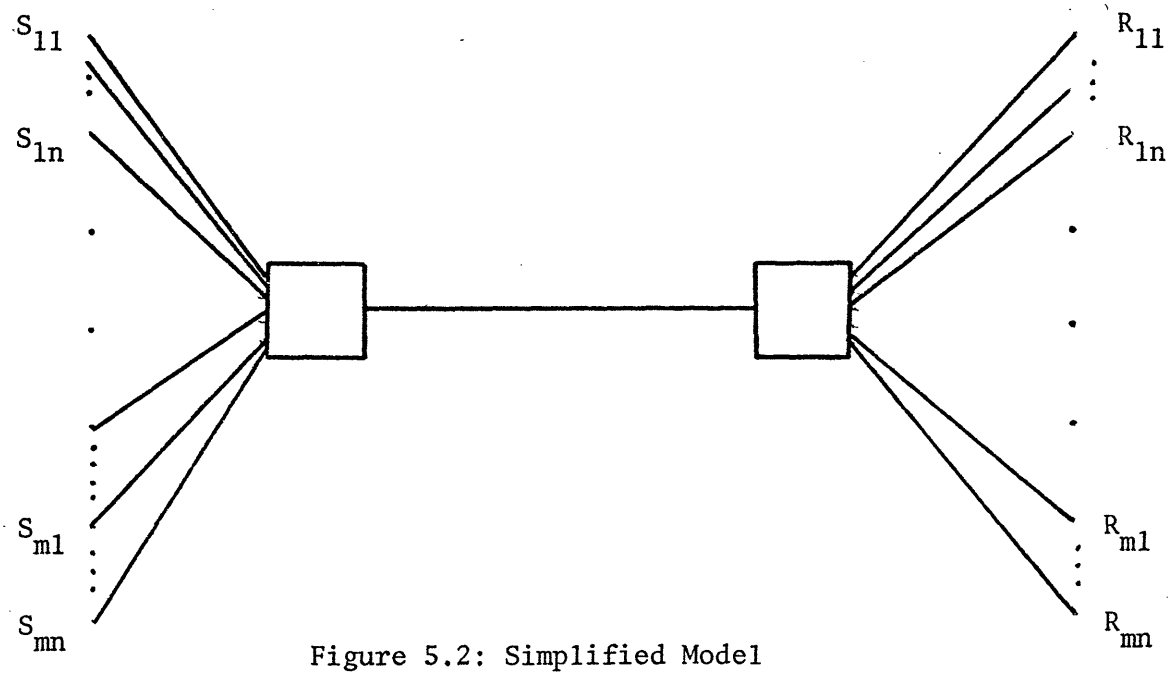


Figure 5.2: Simplified Model

Each source sends messages only to the corresponding receiver so it is enough to indicate to the deconcentrator the message origins. We will consider only this reduced problem.

2. Basic Idea.

Assume now that there are M independent sources, and that messages from source i arrive at the concentrator in a Poisson manner at rate λ, so that, as seen by the concentrator, the probability that the next message comes from source i is $1/M$. Does this imply that we need at least an average of $\log_2 M$ bits per message to indicate the origins to the deconcentrator? The negative answer to this question justifies the existence of this chapter.

If the messages were sent out by the concentrator in the order they were received, $\log_2 M$ would be a lowerbound to the average number of bits per message. However, although in general messages from a given source must be sent in the order they were received, to insure the intelligibility of the sequence of messages, there is no reason for messages from different sources to be transmitted in this fashion. It is precisely the possibility of reordering the messages that permits a decrease in the amount of information. We will illustrate by two examples how easily this can be done.

In both cases we assume as in Chapter 4 that each message contains a codeword indicating its length, that the sources are ergodic, that the mean interemission time of source i is $E(a_i)$, and that the mean length of messages from source i is $E(b_i)$. In both techniques we queue the messages in a special buffer according to their origins.

In technique I we transmit a "0" if buffer i is empty; if not, we transmit a "1" followed by a message. We go then to buffer $(i+1) \bmod M$ and repeat the process.

In technique II we still transmit a "0" if the buffer is empty; if it is not empty we transmit all messages present, prefixing then with a "1". We go to buffer $(i+1) \bmod M$ and repeat the procedure.

In both cases, if the receiver is initially synchronized and if there is no transmission error, the receiver will be able to recognize the origins of all messages.

By a reasoning similar to the one in Section 1 of Chapter 4, we obtain the result that the average number of protocol bits (the "0"s and

the "1"s) per message is equal to

$$\frac{1 - \sum_{i=1}^{M} E\, b_i / E\, a_i}{\sum_{i=1}^{M} 1/E\, a_i} \qquad (1)$$

for all techniques resulting in a stable system. One sees that in heavy traffic ($\sum_{i=1}^{M} E\, b_i / E\, a_i \tilde{<} 1$) this quantity will be very small. We recognize that amongst the protocol bits, some indicate that the line is "idle" (all buffers are empty), while others effectively indicate the origin of the messages, but the receiver is incapable of differentiating between these two kinds.

The conceptual difficulty of defining a "protocol bit" that we met in Chapter 4 reappears even more strongly here. We could try to reintroduce the concept of "low priority bit" from Chapter 4 but this does not appear to lead to very useful results. We will rather use two other approaches: in Section 3 we will modify the model and neglect completely the idle bits, concentrating on the study of how the reordering of the messages can decrease the amount of information necessary to specify their origins. In Sections 4 to 7 we will analyze some strategies to transmit the messages and their origins in an efficient manner, the goal being to minimize the expected message delay.

3. A Simplified Model

A. Introduction

To avoid the difficulties associated with the presence of idle times in the usual queueing model, but still be in a position to study the influence of the reordering of the messages on the amount of information necessary to specify their origins, we study the following model where we keep the number of messages in the queue constant.

B. Notation and Description of the Model

At time 0 a buffer contains $N-1$ messages, of which m_j came from source j, $j = 1,2,\ldots,M$.

At time $i + \frac{1}{4}$, $i = 0,1,\ldots$, one and only one new message enters, it comes from source j with probability p_j independently of the initial content of the buffer and of the previous arrivals. We denote its origin by X_i.

At time $i + \frac{3}{4}$ one and only one message is removed from the buffer. We denote its origin by Y_i.

We denote by S_i the state of the buffer at time i, i.e. S_i is a M-tuple $(m_1, m_2,\ldots,m_M)$, $\sum_{j=1}^{M} m_j = N - 1$, where m_j is the number of messages from source j present in the buffer at time i. One sees that the number of possible values of S_i is $\binom{N+M-2}{N-1}$ [Feller, 1968, p. 38], which we denote by σ. We index in some way the values of S_i, and denote them by $s_1, s_2,\ldots,s_\sigma$. The probability distribution of S_o is known a priori. We denote it by the row matrix Π_o, whose j^{th} component is equal to $\Pr(S_o = s_j)$.

Similarly, S_i^+ denotes the state of the buffer at time $i + \frac{1}{2}$. N messages are present in the buffer at that time, so that S_i^+ can take $\sigma^+ := \binom{N+M-1}{N}$ different values denoted $s_1^+, s_2^+, \ldots, s_{\sigma^+}^+$.

Very often we will need to deal with sequences of inputs and outputs. $X_{[i,j)}$ denotes the sequence $(X_i, X_{i+1}, \ldots, X_{j-1})$ and we define $Y_{[i,j)}$ in a similar fashion.

It will prove useful to define a function U (for Unordering) whose domain is the set of sequences $X_{[i,j)}$ and $Y_{[i,j)}$ and whose values are M-tuples. The k^{th} component of $U(X_{[i,j)})$ is the number of X_n in $X_{[i,j)}$ that are equal to k.

We can use U immediately to verify the relation

$$S_i + U(X_{[i,j)}) - U(Y_{[i,j)}) = S_j \qquad i \leq j \qquad (1)$$

If a suitable probability distribution has been defined, $H(Y_{[i,j)})$ denotes the entropy of $Y_{[i,j)}$, i.e.

$$H(Y_{[i,j)}) := - \sum_{y_{[i,j)}} \Pr(Y_{[i,j)} = y_{[i,j)}) \log_2(\Pr(Y_{[i,j)} = y_{[i,j)}))$$

To avoid the introduction of more symbols, we also use H in the following sense: if c is a s-tuple, $c = (c_1, c_2, \ldots, c_s)$, with non negative components, we define $H(c) := - \sum_{i=1}^{s} c_i \log_2 c_i$. The meaning of $H(.)$ will always be clear from the context.

C. Objective

The problem we wish to study is to find an "optimal" way of making the Y_i's known to an observer watching the output of the buffer.

This involves two distinct points: first, at time $i + \frac{1}{2}$ the

transmitter must decide what message to send out, i.e. the value of Y_i. There is a constraint on Y_i: one can only send out a message that is actually in the buffer. Mathematically this translates into the statement: "all components of S_{i+1} must be non negative," and was implicitly taken into account when we determined the number of states. Second, the receiver must be able to recognize Y_i. To that effect we allow a binary codeword of (variable) length n_i to be transmitted in front of every message, and we require that the knowledge of the codewords transmitted at time $j + \frac{3}{4}$, $j=0,1,2,\ldots,i$, and of $Y_{[0,i)}$ uniquely specifies Y_i.

Our objective will be to minimize the "expected number of protocol bits per message," $h := \limsup_{T\to\infty} E\left(\frac{1}{T}\sum_{i=0}^{T-1} n_i\right)$ over all possible encoding strategies, i.e. the choice of the message to be sent next, and the choice of the codewords indicating what message is sent.

We will give some examples in Section D and a lower bound in Section E. Finally we show in Section F how dynamic programming can be used to find the "optimal" choice of the message to be sent next.

D. Examples of Strategies

The end of the previous section may be made clearer by considering the following strategies.

STRATEGY I

We transmit the messages in the order they entered the buffer; this is the only choice if $N=1$. The probability that $(Y_i=k) = p_k$, $i \geq N$, thus the best we can do is to use a Huffman code to indicate the message origins, and the average number of protocol bits per message, h,

will be bounded by

$$H(p) \leq h < H(p) + 1$$

where $p := (p_1, p_2, \ldots, p_M)$.

STRATEGY II

We do the following: at time .75 we send a Huffman codeword specifying S_o^+ ; at times .75, 1.75, ..., N -.25 we transmit in some prespecified order (e.g. all messages from source 1, followed by messages from 2, etc.) all messages present in the buffer at time .5 . Note that no codewords are needed at times 1.75, 2.75, ..., N - .25 . At time N + .75 we transmit a codeword specifying S_N^+ and repeat the procedure. The probability that $S_{kn}^+ = (m_1, m_2, \ldots, m_M)$, $k \geq 1$, is equal to $\frac{N!}{m_1! \ldots m_M!} p_1^{m_1}, \ldots, p_M^{m_M}$, $m_j \geq 0$, $\sum_{j=1}^{M} m_j = N$, thus $\frac{H(p^{*N})}{N} \leq h < \frac{H(p^{*N}) + 1}{N}$ where $H(p^{*N})$ denotes the entropy of the multinominal probability distribution.

It is of interest to examine how this expression behaves as N increases. We can write

$$\frac{H(p^{*N})}{N} = - \sum_{j=1}^{M} p_j \log_2 p_j - \frac{1}{N} \log_2 N! + \frac{1}{N} \left\{ \sum_{j=1}^{M} \sum_{m=1}^{N} \binom{N}{m} p_j^m (1-p_j)^{N-m} \log_2 m! \right\}$$

To get a lowerbound we use the log-convexity of the gamma function to obtain

$$\frac{H(p^{*N})}{N} \geq - \sum_{j=1}^{M} p_j \log_2 p_j - \frac{1}{N} \log_2 N! + \frac{1}{N} \sum_{j=1}^{M} \log_2 \Gamma(1 + Np_j)$$

The use of Stirling's formula [Feller, 1968, p. 52], tight if $Np_j \geq 1$

$$\log(\Gamma(1+x)) \geq \log \sqrt{2\pi} + (x + \tfrac{1}{2}) \log x - x \log e$$

yields

$$\frac{H(p^{*N})}{N} \gtrsim \frac{M-1}{2N} \log_2 (2\pi N) + \frac{1}{2N} \sum_{j=1}^{M} \log_2 p_j \qquad (2)$$

To obtain an upperbound, we use Stirling's formula for $\log m!$ together with the inequality

$$\log m \leq \log Np_j + \frac{m - Np_j}{Np_j} \log e$$

This yields

$$\log m! \leq \log_2 \frac{\sqrt{2\pi Np_j}}{e} + m \left(\log \frac{Np_j}{e^2} + \frac{\log e}{2Np_j} \right) + \frac{m^2}{Np_j} \log e$$

This does not hold at $m=0$ when $Np_j < .43$ but is otherwise satisfied. Using this in the formula for $H(p^{*N})$, and using Stirling's approximation for $\log N!$, we obtain

$$\frac{H(p^{*N})}{N} \lesssim \frac{M-1}{2N} \log_2 (2\pi eN) + \frac{1}{2N} \sum_{j=1}^{M} \log_2 p_j$$

We can thus conclude that for this strategy, the expected number of protocol bits per message is equal to $\frac{M-1}{2N} \log_2 N + 0\left(\frac{1}{N}\right)$.

STRATEGY III

Here we note that at time $i + \frac{1}{2}$ there is at least one source such that $\lfloor \frac{N+M-1}{M} \rfloor$ messages from it are stored in the buffer. We send the binary representation of the index of this source, then the $\lfloor \frac{N+M-1}{M} \rfloor$ messages. The average number of protocol bits per message is bounded by:

$$\frac{\log_2 M}{\lfloor \frac{N+M-1}{M} \rfloor} \leq h < \frac{(\log_2 M) + 1}{\lfloor \frac{N+M-1}{M} \rfloor}$$

Here for large N, h is approximately equal to $\frac{M}{M+N-1} \log_2 M$ which is better than in Strategy II. However, for small N, II may be

better.

The two following strategies will be studied for the case $M=2$ and $p_1 = p_2 = .5$. A comparison of all strategies for this case appears in Table 5.1.

STRATEGY IV

Strategy IV is essentially polling: one transmits as many messages from source 1 as possible, until none remains in the buffer. One transmits then a run of messages from source 2, then 1 again, etc. The end of a run can be indicated by a flag as studied in Chapter III.

If $N=1$, each run has a geometric probability distribution. In general, the probability distribution of a run is the distribution of the sum of N independent geometric random variables, and thus a Pascal distribution:

$$\Pr(\text{run} = n) = \left(\frac{1}{2}\right)^n \binom{n-1}{n-N} \qquad n=N, N+1, \ldots$$

Its mean is equal to $2N$, so we can bound the expected number of protocol bits per message by

$$\frac{\text{Entropy of run}}{2N} \leq h < \frac{\text{Entropy of run} + .56}{2N}$$

The upper bound holds if the assumptions made in Section 4 of Chapter III are satisfied.

We now turn our attention to evaluating the entropy. This can be done numerically; results appear in Table 5.1. To obtain asymptotic results we note that the entropy is equal to $2N - \sum_{n=N}^{\infty} \left(\frac{1}{2}\right)^n \binom{n-1}{n-N} \log_2 \binom{n-1}{n-N}$.

Writing $\log \binom{n-1}{n-N} = \sum_{i=1}^{N-1} \log (n-N+i) - \log (N-1)!$, we see, from the convexity of $\log (n-N+i)$, that $-\log \binom{n-1}{n-N}$ is concave. By Jensen's

inequality we can lowerbound the entropy by $2N - \log_2 \binom{2N-1}{N}$ and, using Stirling's approximation, by $\frac{\log_2 4\pi N}{2}$. Writing $\log \binom{n-1}{n-N} = \log (n-1)! - \log (n-N)! - \log (N-1)!$, using the convexity of the first term, using Stirling's approximation together with the formula $\log_e x \leq x-1$ for the second term, and Stirling's approximation for the third, one can upperbound the entropy by $(\log_2 4\pi e^2 N)/2$. Thus, for Strategy IV, h behaves like $\frac{\log_2 N}{4N} + 0\left(\frac{1}{N}\right)$. This is about twice as good as Strategy II.

STRATEGY V

As mentioned earlier, we study this strategy only for $M=2$ with $p_1 = p_2 = .5$. Suppose that at time $i + .5$ we know that only messages from source j ($j = 1$ or 2) are in the buffer. We can then send N of them without any codeword, and the distribution of S^+_{N+i} will be binomial. We then alternate between messages from 1 and 2, until this becomes impossible because the buffer contains only one kind of message. We then signal the end of the run, e.g. by a flag.

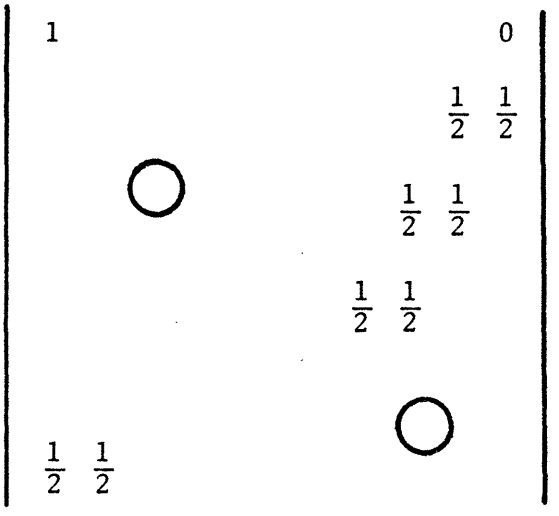
The expected number of protocol bits per message is thus bounded by

$$\frac{\text{Entropy of run}}{N + E(\text{run})} \leq h < \frac{\text{Entropy of run} + .56}{N + E(\text{run})}$$

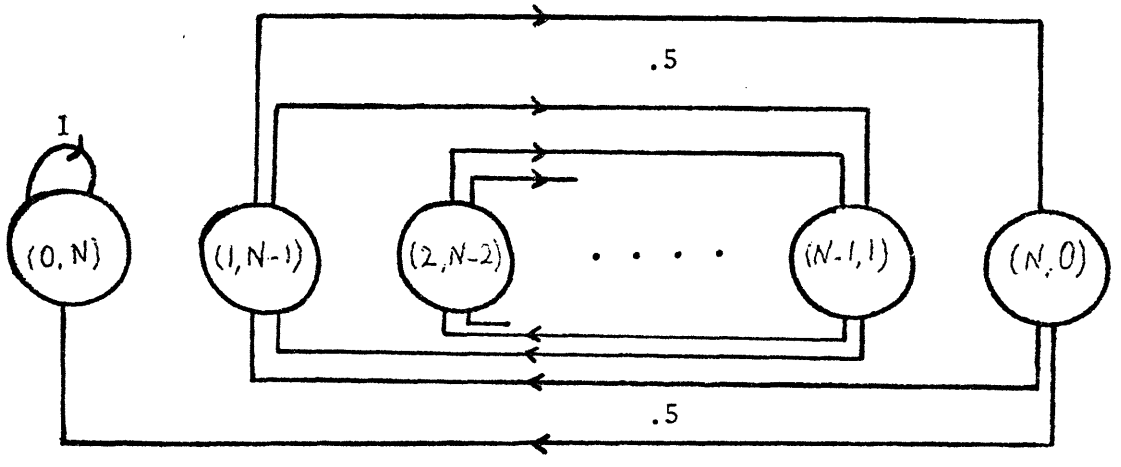
The upperbound holds if the assumptions made in Section 4 of Chapter III are satisfied.

It is of primary importance to study the statistics of the run. Assume that we try to send a message from source 1 at odd times, and a message from source 2 at even times. S^+_i performs a non-stationary

random walk: with probability .5 , $S^+_{i+1} = S^+_i$ whereas with probability .5 , $S^+_{i+1} = S^+_i + (-1, 1)$ if i is odd, and $S^+_{i+1} = S^+_i + (1, -1)$ if i is even. A run stops if $S^+_i = (0, N)$ with i odd, or (N, 0) with i even. However, we note that as far as the statistics of the remaining time in the run is concerned, being in state $S^+_i = (k, N-k)$ at time i is equivalent to being in state $S^+_{i+1} = (N-k, k)$ at time i+1 . We can thus describe the process by the (N+1, N+1) transition matrix



corresponding to the stationary process:



It is entirely feasible to compute the distribution of the time until trapping in state $(0,N)$ if the initial probability distribution of the state is binomial, by classical Markov chain methods e.g. [Howard, 1971, vol. I]. Results appear in Table 5.1. Fortunately, the mean time until trapping has a simple form. Denoting by $g(m_1, m_2)$ the mean time until trapping if the initial state is (m_1,m_2) one finds the relations

$$g(0, m_2) = 0$$

$$g(m_1, m_2) = 1 + \frac{1}{2}\,(g(m_2+1, m_1 -1) + g(m_2, m_1)) \qquad m_1 > 0$$

The solution to this system of equations is

$$g(m_1, m_2) = 2m_1\ (2m_2 + 1)$$

Averaging on the binomial distribution of the initial state, one finds that the expected run is equal to N^2. It is now easy to upperbound the entropy of the run: by [Gallager, 1968, p. 507] it is upperbounded by $(N^2 + 1)\, H\left(\frac{1}{N^2 + 1}\right)$ where H is the binary entropy, i.e. $H(x) := H((x,1-x))$. This bound is extremely close to the actual value (the relative difference is less than 1%), indicating that the probability distribution of the run is nearly geometric. From the results of Section 4 of Chapter III, fixed-length flags will be almost optimal.

Because $x\, H\left(\frac{1}{x}\right) \leq \log_2 ex$, h is upperbounded by

$$h \leq \frac{\log_2 (e(N^2 + 1)) + .56}{N^2 + N}$$

The presence of N^2 in the denominator makes this scheme markedly superior to all others. Note that it is the combination of two features that makes it efficient:

-- the fact that it does not attempt to send a message for N time units after detecting that no such message is present;

-- the fact that it alternates between sources.

Strategy IV (polling) has the first feature, but not the second; we have seen that the expected run is equal to $2N$. If one uses pure alternating, the expected run will be equal to $1 + \frac{1}{2} g_0 + \frac{1}{2} g_1 = 2N$, instead of $N+N^2$ when both features are present.

There are strategies for which h behaves like $(k \log(N))/N^2 + 0\left(\frac{1}{N^2}\right)$ even when $M > 2$. We describe now such a strategy for the symmetric case $(p_i = \frac{1}{M}, \quad i=1,2,\ldots,M)$. It is a generalization of Strategy V.

One removes one message from each source in cycles (say $1,2,3,\ldots M,1,2,\ldots$) until this becomes impossible. One transmits then $M-1$ codewords indicating the number of messages from each origin remaining in the buffer, and those N messages. This being done the distribution of the buffer state is multinomial and we start the procedure again, removing one message from each source in cycles. We call the number of messages transmitted during the cyclic part of this strategy a <u>run</u>.

If one uses a flag strategy as described in Section 4 of Chapter III to indicate the end of a run, h will be upperbounded by

$$h \leq \frac{\log_2 (e(E(\text{run}) + 1)) + .57 + (M-1)\ [\log_2 N]}{E(\text{run}) + N}$$

If one can show that E(run) is proportional to N^2 the desired result will be obtained. E(run) can be computed as in Strategy V. If $g(m_1,\ldots,m_M)$ denotes the expected run length if the initial state is $(m_1,\ldots,m_M)$, one has the relations

$$g(0, m_2, \ldots, m_M) = 0$$

$$g(m_1,m_2, \ldots, m_M) = 1 + \frac{1}{M} (g(m_2+1, m_3, \ldots,m_1-1) + g(m_2, m_3+1,\ldots,m_1-1) \ldots + g(m_2, m_3,\ldots m_1))$$

$$m_1 > 0$$

This can be solved numerically. For M=3 we obtain the expression

$g(m_1, m_2, m_3) = \frac{3m_1 (3m_2 + 1)(3m_3 + 2)}{3 (m_1 + m_2 + m_3) + 1}$. E(run) is equal to the average of g(.) over the multinomial distribution of the initial state. If M=3 we obtain $E(run) = N(N^2 + 1)/(3N + 1)$, which is approximately equal to $N^2/3$ for large N, as desired.

We are unable to solve this system of equation for all N, but can lowerbound E(run) by the following method.

Let $(m_1^j, m_2^j, \ldots, m_M^j)$ denote the state of the buffer at time j + .5. Assume that at time .5 the state distribution is multinomial, and start removing the messages in cycles. In order to obtain the

bound we remove the constraint that the m_i^j 's must be non negative. Thus the buffer state performs a non-stationary random walk and

$$\Pr(\text{run} \leq j) = \Pr\left(\min_{0 \leq k \leq j} (m^k_{(k+1) \bmod M}) \leq 0\right) \qquad j=0,1,\ldots$$

$$\leq \sum_{i=1}^{M} \Pr\left(\min_{k \in \mathbb{N} : 0 \leq i+kM-1 \leq j} (m_i^{i+kM-1}) \leq 0\right)$$

$$j=0,1,\ldots$$

We recall a version of Kolmogorov's inequality [Karlin, 1975, p. 280]: If $a_1, a_2, \ldots$ form a martingale and have a mean $Ea > 0$ then $\Pr(\min(a_1, a_2, \ldots, a_n) \leq 0) \leq \frac{\mathrm{Var}(a_n)}{(Ea)^2}$. Here for each i the m_i^{i+kM-1} 's, $k=0,1,\ldots,$ form a martingale and have mean $\frac{N+i-1}{M}$ and variance $(N+i+kM-1)\frac{1}{M}(1 - \frac{1}{M})$.

Thus

$$\Pr(\text{run} \leq j) \leq M \left(\frac{(N+j)\frac{1}{M}(1 - \frac{1}{M})}{(\frac{N}{M})^2} \right) \qquad j=0,1,\ldots$$

and $$\Pr(\text{run} > x) \geq \max\left(0, \frac{N^2 - (N+x)M(M-1)}{N^2}\right) \qquad x \geq 0$$

$$E(\text{run}) = \int_0^\infty \Pr(\text{run} > x)\, dx$$

$$\geq \frac{1}{2} \frac{(N - M(M-1))^2}{M(M-1)} \qquad N \geq M(M-1)$$

This shows that $E(\text{run})$ increases at least proportionally to N^2 for large N, as desired.

	Strategy			
	II	III	IV	V
N = 1	1	1	1	1
2	.75	1	.678	.599
3	.604	.5	.519	.390
4	.508	.5	.423	.274
5	.440	.333	.358	.203
6	.389	.333	.312	.157
7	.350	.250	.276	.126
8	.318	.250	.248	.103
9	.292	.200	.226	.086
10	.271	.200	.208	.073
11	.252	.167	.192	.063
12	.237	.167	.179	.055
13	.223	.143	.167	.048
14	.211	.143	.158	.043
15	.200	.125	.149	.038
16	.190	.125	.141	.035

Table 5.1

$\lim_{T\to\infty} \frac{1}{T} H\left(Y_{[0,T)}\right)$ as a function of N

M=2 $\quad p_1 = p_2 = .5$

E. A Lower Bound on h:

We have shown in Section D that simple strategies (i.e. II) can make h decrease like $\frac{M}{M+N-1}$; more complicated strategies (i.e. V) yield a decrease proportional to $\frac{\log_2 N}{N^2}$. We will show here that h cannot decrease faster than $((M-1)/(M+N-1))^2$. We will use in the sequel many standard relations between information theoretic expressions; they can be found in [Gallager, 1968].

Assume that we have decided on a feasible strategy. We have that for all T ,

$$\frac{1}{T} E \sum_{i=0}^{T-1} n_i \geq \frac{1}{T} H(Y_{[0,T)})$$

thus

$$h \geq \limsup_{n\to\infty} \frac{1}{T} H(Y_{[0,T)})$$

$$\geq \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \frac{1}{t} H(Y_{[it,(i+1)t)} \,|\, Y_{[0,it)}) \qquad t=1,2,3,\ldots$$

(in fact we have equality, but this requires a little proof)

$$\geq \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \frac{1}{t} H(Y_{[it,(i+1)t)} \,|\, Y_{[0,it)}, S_{it})$$

$$t=1,2,3,\ldots \qquad (3)$$

We now lowerbound $\frac{1}{t} H(Y_{[it,(i+1)t)} \,|\, Y_{[0,it)}, S_{it})$:

We have $\frac{1}{t} H(Y_{[it,(i+1)t)} \,|\, Y_{[0,it)}, S_{it}) \geq \frac{1}{t} I(X_{[it,(i+1)t)}; Y_{[it,(i+1)t)} \,|\, Y_{[0,it)}, S_{it})$

where $I(A;B) := H(A) - H(A|B)$

$= H(B) - H(B|A)$

$$\geq \frac{1}{t} I(U(X_{[it,(i+1)t)});U(Y_{[it,(i+1)t)}) \mid Y_{[0,it)}, S_{it})$$

by the Data Processing Theorem

[Gallager, 1968, p. 80].

$$= \frac{1}{t} (H(U(X_{[0,t)})) - H(U(X_{[it,(i+1)t)}) \mid U(Y_{[it,(i+1)t)}), Y_{[0,it)}, S_{it}))$$

by independence of the X_i's.

Repeating relation (1)

$$U(X_{[it,(i+1)t)}) = S_{(i+1)t} + U(Y_{[it,(i+1)t)}) - S_{it}$$

and remembering that $S_{(i+1)t}$ can take σ different values, we see that for every $U(Y_{[it,(i+1)t)})$ and S_{it}, $U(X_{[it,(i+1)t)})$ can take at most σ different values.

Thus $H(U(X_{[it,(i+1)t)}) \mid U(Y_{[it,(i+1)t)}), Y_{[0,it)}, S_{it}) \leq \log_2 \sigma$

Writing $H(U(X_{[it,(i+1)t)})) = H(p^{*t})$ as in Section D, and replacing in (3) one obtains

$$h \geq \max_{t=1,2,..} \frac{1}{t} (H(p^{*t}) - \log_2 \sigma) \qquad (4)$$

This can easily be computed.

We are interested in an asymptotic relation for large N. Using (2)

$$H(p^{*t}) \geq \frac{M-1}{2} \log_2(2\pi t) + \frac{1}{2} \sum_{j=1}^{M} \log_2 p_j$$

with $t = \frac{e}{2\pi} \left(\frac{\sigma^2}{\prod_{j=1}^{M} p_j} \right)^{\frac{1}{M-1}}$ in (4)

one obtains (neglecting the integer constraint on t)

$$h \geq \frac{2\pi(\log_2 e)\left(\prod_{j=1}^{M} p_j\right)^{\frac{1}{M-1}}}{e\,\sigma^{2/M-1}}$$

For $M=2$, $\sigma=N$,

so $$h \geq \frac{2\pi\ (\log_2 e)\ p_1\ p_2}{e\ N^2}$$

$$= \frac{.834}{N^2} \qquad \text{if } p_1 = p_2 = \frac{1}{2}$$

One can show that

$$\sigma := \binom{N+M-2}{M-1} \leq \left(e\ \frac{M+N-2}{M-1}\right)^{M-1}$$

so

$$h \geq \frac{2\pi(\log_2 e)\left(\prod_{j=1}^{M} p_j\right)^{\frac{1}{M-1}}}{e^3}\ \left(\frac{M-1}{M+N-2}\right)^2$$

F. "Optimal" Strategy

As explained in Section C, a strategy consists of two parts:

-- a rule to determine the value of Y_i ;

-- a code to indicate the value of Y_i .

The first part is the most interesting. We will gain some insight into it by assuming that non integer codeword lengths can be used subject only to the Kraft inequality [Gallager, 1968, p. 47]; in that case it is very easy to solve the second part.

Let's assume that one has decided how to select that Y_i 's; then for all encoding strategies

$$E \sum_{i=0}^{T-1} n_i \geq H(Y_{[0,T)})$$

$$= \sum_{i=0}^{T-1} \sum_{y_{[0,i)}} -\Pr(Y_{[0,i)}=y_{[0,i)}) \left(\sum_{y_i} \Pr(Y_i=y_i \,|\, Y_{[0,i)}=y_{[0,i)}) \log_2 (\Pr(Y_i=y_i \,|\, Y_{[0,i)}=y_{[0,i)})) \right)$$

This lowerbound can be achieved by using at time i a codeword of length

$$-\log_2 (\Pr(Y_i=y_i \,|\, Y_{[0,i)}=y_{[0,i)}))$$

$$\text{if } Y_i=y_i \text{ and } Y_{[0,i)}=y_{[0,i)}$$

This codeword provides just enough information to enable the receiver to recognize Y_i. A consequence of this is that the conditional probabilities

$$\Pr(S_i=s_j \,|\, Y_{[0,i)}=y_{[0,i)} \text{ and codewords transmitted between 0 \& i})$$

$$= \Pr(S_i=s_j \,|\, Y_{[0,i)}=y_{[0,i)})$$

Note that this is not true for all encoding strategies: in Strategy II, the codeword transmitted at time .75 specifies not only Y_o, but also S_o^+. Thus in general $\Pr(S_1=s_j \,|\, Y_o=k$, codeword transmitted at .75) $\neq \Pr(S_1=s_j \,|\, Y_o=k)$.

Now that we have "solved" the second part of the problem, we can turn our attention to the first part: how should we choose the Y_i's so as to minimize

$$\limsup_{T\to\infty} \frac{1}{T} H(Y_{[0,T)}) = \limsup_{T\to\infty} -\frac{1}{T} \sum_{i=0}^{T-1} \sum_{y_{[0,i)}} \Pr(Y_{[0,i)}=y_{[0,i)}) \left(\sum_{y_i} \Pr(Y_i=y_i \,|\, Y_{[0,i)}=y_{[0,i)}) \log_2 (\Pr(Y_i=y_i \,|\, Y_{[i,0)}=y_{[i,0)}) \right) \quad (4)$$

It turns out that this can be done by dynamic programming. Unfortunately we need first to give some more definitions:

Σ^s denotes the unit simplex of $\mathbb{R}^s$

u denotes a column matrix of suitable dimension (depending on the context) with all components equal to 1 .

e_k denotes a row matrix of suitable dimension (depending on the context) with all components equal to 0 , except the k^{th} , which is equal to 1.

$\Pi_i(y_{[0,i)})$ is a M-tuple whose j^{th} component is equal to $\Pr(S_i=s_j \mid Y_{[0,i)}=y_{[0,i)})$.

Similarly,

$\Pi_i^+(y_{[0,i)})$ is a σ^+-tuple whose j^{th} component is equal to $\Pr(S_i^+=s_j^+ \mid Y_{[0,i)}=y_{[0,i)})$.

By independence of the X_i , one can write:

$$\Pi_i^+(y_{[0,i)}) = \Pi_i(y_{[0,i)})\ P \qquad (5)$$

where P is a (σ,σ^+) stochastic matrix whose element $P_{ij} = p_k$ if $s_j^+ = s_i + e_k$, and 0 if there is no such k.

EXAMPLE: $M = 2$ $N = 2$

$\sigma = 2$ $\sigma^+ = 3$

if $s_1 = (1,0)$ $s_2 = (0,1)$

$s_1^+ = (2,0)$ $s_2^+ = (1,1)$ $s_3^+ = (0,2)$

then

$$P = \begin{pmatrix} p_1 & p_2 & 0 \\ 0 & p_1 & p_2 \end{pmatrix}$$

***

A <u>policy</u> α , $\alpha=1,2,\ldots\tau$ (τ will be defined later), is charac-

terized by a (σ^+, M) policy matrix A^α with the following properties:

1) $A^\alpha_{ij} = 0$ or 1

2) $\sum_{j=1}^{M} A^\alpha_{ij} = 1$

3) $A^\alpha_{ij} = 1$ only if the state S^+_i contains a message from source j.

The significance of this is that if at time $k+.5$ the state is s^+_j, one will choose $Y_k := m$ such that $A_{jm} = 1$. Properties 1) and 2) guarantee that a unique such m exists, and 3) guarantees that only messages that are actually in the buffer may be sent.

Matrix A^α has the following additional properties, which are easy to verify

1) A^α is stochastic

2) If policy α is used at time i, the conditional probability that $Y_i = k$ given $Y_{[0,i)} = y_{[0,i)}$ is equal to the k^{th} component of $\Pi^+_i(y_{[0,i)})A^\alpha$, or (by (5)) of $\Pi_i(y_{[0,i)})P\,A^\alpha$.

EXAMPLE: $M = 2$ $N = 2$ as before.

There are only two policies, 1 and 2, with

$$A^1 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad A^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

In both cases, one transmits a "1" in state (2,0) and a "2" in state (0,2) (there is no other choice); policy 1 transmits a "1" in state (1,1), whereas policy 2 transmits a "2."

If $\Pi_i(y_{[0,i)}) = (\rho_1, \rho_2)$, and if policy 1 is used,

$$(\Pr(Y_i=1 \mid Y_{[0,i)}=y_{[0,i)}),\ \Pr(Y_i=2 \mid Y_{[0,i)}=y_{[0,i)}))$$

$$= (\rho_1 \ \rho_2) \begin{pmatrix} p_1 & p_2 & 0 \\ 0 & p_1 & p_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= (\rho_1 + \rho_2 \ p_1, \ \rho_2 \ p_2) \qquad ***$$

Note that the number , τ , of policies can be quite large: if messages from k origins are present in state s_j^+ , the j^{th} row of a policy matrix can take k distinct values. The number of states with messages from k origins present is in turn equal to $\binom{M}{k}\binom{N-1}{k-1}$ (with $\binom{a}{b} := 0$ if $a < b$), where the first factor is the number of distinct choices of k origins, and the second factor represents the number of ways of distributing N messages between k origins, in such a way that each origin receives at least one message. This last number is equal to the number of ways of distributing $N-k$ messages between k origins. Thus there are $\prod_{k=1}^{M} k\binom{M}{k}\binom{N-1}{k-1}$ distinct policies.

EXAMPLE:

We have seen that if $M=N=2$, there are 2 policies. In the seemingly innocuous case $M=4$, $N=8$, there are about $6.22\ 10^{73}$ policies.

***

Associated to policy α we define M (σ^*,σ) <u>transition matrices</u> $B^{\alpha,k}$, $k=1,2,\ldots,M$, by

$$B_{ij}^{\alpha,k} = \begin{cases} 1 & \text{if and only if} \\ & \left\{ \begin{array}{l} s_j = s_i^+ - e_k \\ A_{ik}^{\alpha} = 1 \end{array} \right. \\ 0 & \text{otherwise} \end{cases}$$

These matrices have the following properties; they are proved by direct examination.

1) $B^{\alpha,k}\, u$ = k^{th} column of A^{α} ;

2) If policy α is used at time i, $\Pr(Y_i=k \mid Y_{[0,i)}=y_{[0,i)})$ $= \Pi_i^+(y_{[0,i)})\, B^{\alpha,k}\, u$;

3) If policy α is used at time i,

$$\Pi_{i+1}((y_{[0,i)},k)) = \frac{\Pi_i^+(y_{[0,i)})\, B^{\alpha,k}}{\Pi_i^+(y_{[0,i)})\, B^{\alpha,k}\, u} \qquad (6)$$

(this is Bayes' rule).

Property 2) justifies the appellation of "transition matrix." Using (5), (6) can be written as

$$\Pi_{i+1}((y_{[0,i)},k)) = \frac{\Pi_i(y_{[0,i)})\, P\, B^{\alpha,k}}{\Pi_i(y_{[0,i)})\, P\, B^{\alpha,k}\, u}$$

EXAMPLE: $M = 2$ $N = 2$ as before.

Associated to policy 1 (defined earlier), we have the matrices

$$B^{1,1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad B^{1,2} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

As an illustration, $B^{1,1}_{2,2} = 1$ because state (0,1) can be obtained from (1,1) by removing (1,0), and because if policy 1 is used, a "1" is transmitted if the state is (1,1).

Say policy 1 is used at time i,

$$\Pi_i(y_{[0,i)}) = (\rho_1, \rho_2)$$

$$\Pi_i^+(y_{[0,i)}) = (\rho_1^+, \rho_2^+, \rho_3^+) = (\rho_1 p_1, \rho_1 p_2 + \rho_2 p_1, \rho_2 p_2)$$

Then if $Y_i = 1$

$$\Pi_{i+1}((y_{[0,i)},1)) = \left(\frac{\rho_1^+}{\rho_1^+ + \rho_2^+}, \frac{\rho_2^+}{\rho_1^+ + \rho_2^+}\right)$$

$$= \left(\frac{\rho_1 p_1}{\rho_1 + \rho_2 p_1}, \frac{\rho_1 p_2 + \rho_2 p_1}{\rho_1 + \rho_2 p_1}\right)$$

whereas if $Y_i = 2$

$$\Pi_{i+1}((y_{[0,i)},2)) = (0,1)$$

Similar expressions result if policy 2 is used. ***

Although we are interested in minimizing $\limsup_{T\to\infty} \frac{1}{T} H(Y_{[0,T)})$ it is easier to first minimize $\frac{1}{T+1} H(Y_{[0,T+1)})$ for some fixed T. We have from (4)

$$H(Y_{[0,T+1)}) = -\sum_{i=0}^{T} \sum_{y_{[0,i)}} \Pr(Y_{[0,i)}=y_{[0,i)})\, C_{T-i}(y_{[0,i)})$$

where $C_{T-i}(y_{[0,i)}) := -\sum_{y_i} \Pr(Y_i=y_i \mid Y_{[0,i)}=y_{[0,i)}) \log_2(\Pr(Y_i=y_i \mid Y_{[0,i)}=y_{[0,i)}))$ is called the <u>expected immediate cost</u> at time i, given that $Y_{[0,i)} = y_{[0,i)}$.

Defining $D_0(y_{[0,T=1)}) := 0$

$$D_{i+1}(y_{[0,T-i)}) := C_i(y_{[0,T-i)}) + \sum_{y_{T-i}} \Pr(Y_{T-i}=y_{T-i} \mid Y_{[0,T-i)}=y_{[0,T-i)}) \cdot D_i((y_{[0,T-i)},y_{T-i})) \quad (7)$$

We have that $H(Y_{[0,T+1)}) = D_{T+1}$. D_i is called the <u>cost to go</u> at time T-i+1. Using Bellman's principle of optimality [Bellman, 1957] we see that this expression can be minimized by going backward in time: at time $T-i$, for every sequence $y_{[0,T-i)}$, we should find a strategy such that the resulting values of $\Pr(Y_{T-i}=k \mid Y_{[0,T-i)}=y_{[0,T-i)})$,

$k=1,2,\ldots M$, minimize $D_{i+1}(y_{[0,T-i)})$.

In a first step we will minimize $H(Y_{[0,T+1)})$ over all strategies consisting of using at time i a policy $\alpha(y_{[0,i)})$. We will show later that nothing is gained by using more general strategies.

At time T the receiver has computed $\Pi_T(y_{[0,t)})$. If the transmitter, which can also compute $\Pi_T(y_{[0,T)})$, decides to use policy α , one checks that

$$C_0(y_{[0,T)}) = H(\Pi_T(y_{[0,T)}) \; P \; A^{\alpha})$$

One sees that there is a policy $\alpha_0(\Pi_T(y_{[0,T)}))$, depending on $y_{[0,T)}$ through $\Pi_T(y_{[0,T)})$, that minimizes $H(\Pi_T(y_{[0,T)}) \; P \; A^{\alpha})$ over all policies. We denote the minimum by $V_1(\Pi_T(y_{[0,T)}))$. Thus V_1 (called the minimal cost to go at time T) is defined by

$$V_1(\Pi) := \min_{\alpha} H(\Pi \; P \; A^{\alpha})$$

$$= H(\Pi \; P \; A^{\alpha_0(\Pi)}) \tag{8}$$

It is aesthetically pleasant to define $V_0(\Pi) := 0$ (9)

EXAMPLE: $N = 2$ $M = 2$ as before

Let $\Pi(y_{[0,T)}) = (\rho_1, \rho_2)$.

If policy 1 is used,

$$C_0(y_{[0,T)}) = H\left((\rho_1, \rho_2) \begin{pmatrix} p_1 & p_2 & 0 \\ 0 & p_1 & p_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} , \; 1 \right)$$

$$= H(p_2\rho_2)$$

whereas if policy 2 is used

$$C_0(y_{[0,T)}) = H(p_1\rho_1)$$

One sees that policy 2 minimizes the expected immediate cost if

$$p_1 \leq \rho_2$$ ***

As we have seen earlier, the number of policies can be enormous. We show now that at most $M!$ policies need to be considered when one minimizes $C_0(y_{[0,T)})$.

THEOREM I

Let α_0 be a policy minimizing the expected immediate cost $H(\Pi\ P\ A^{\alpha})$ for a given Π in Σ^{σ} . Denote the i^{th} component of ΠP by ρ_i . Let $(\tau_1, \tau_2, \ldots, \tau_M) := \Pi\ P\ A^{\alpha_o}$.

For the given Π , for all i such that $\rho_i > 0$ define the relation $\hat{>}$ on $\{1,2,\ldots,M\}$ by :

if $A^{\alpha_o}_{ij} = 1$, then $j \hat{>} k$ for all $k \neq j$ such that s_i contains a message from k .

Then $\hat{>}$ is a partial ordering of $\{1,2,\ldots,M\}$.

Proof:

We must prove that if

$$j_1 \hat{>} j_2$$
$$j_2 \hat{>} j_3$$
$$\vdots$$
$$j_{n-1} \hat{>} j_n$$

then it is not true that $j_n \hat{>} j_1$. Assume to the contrary that $j_n \hat{>} j_1$.

Without loss of generality, assume that state s_i^+ contains messages from j_i and $j_{(i \bmod n)+1}$ $i=1,2,\ldots n$, and that $A^{\alpha_o}_{ij_i} = 1$.

Because α_0 is optimal

$$-\tau_{j_1}\log_2\tau_{j_1} - \tau_{j_2}\log_2\tau_{j_2} \leq -(\tau_{j_1}-\rho_1)\log_2(\tau_{j_1}-\rho_1) - (\tau_{j_2}+\rho_1)\log_2(\tau_{j_2}+\rho_1) \quad (10.a)$$

otherwise $H(\Pi P A^{\alpha})$ would be reduced by making $A^{\alpha_o}_{1j_1} = 0$ and $A^{\alpha_o}_{1j_2} = 1$.

Relation (10.a) can be rewritten as

$$(\tau_{j_1}-\rho_1)\log_2(\tau_{j_1}-\rho_1) - \tau_{j_1}\log_2\tau_{j_1} \leq \tau_{j_2}\log_2\tau_{j_2} - (\tau_{j_2}+\rho_1)\log_2(\tau_{j_2}+\rho_1) \qquad (10.b)$$

The function $x \log_2 x - (x+\rho)\log_2(x+\rho)$ decreases with increasing x for $\rho>0$, so (10.b) implies

$$\tau_{j_1} - \rho_1 \geq \tau_{j_2}$$

Similarly

$$\tau_{j_i} - \rho_i \geq \tau_{j_{(i \bmod n)+1}} \qquad i=1,2,\ldots,n$$

Adding these inequalities one obtains

$$-\sum_{i=1}^{n} \rho_i \geq 0$$

which is a contradiction.

Q.E.D.

Because a partially ordered finite set can be totally ordered, we have the following theorem:

THEOREM II

There is a policy α minimizing $H(\Pi\ P\ A^{\alpha})$ which has the form

-- define an ordering $\hat{>}$ on $\{1,2,\ldots,M\}$
-- $A^{\alpha}_{ij} = 1$ if j is represented in s_i^+ and if $j \hat{>} k$ for all $k \neq j$ represented in s_i^+.

There are at most $M!$ such policies.

Q.E.D.

An algorithm that comes naturally to mind, but which does not quite work, to define the ordering $\hat{>}$ is the following:

-- for $j=1,2,\ldots,M$ compute from Π the probabilities p_k^j, $k=1,2,\ldots,M$, that at time $i+.5$ the buffer contains at least one

message from source k, but none from sources $k_1, k_2, \ldots, k_{j-1}$.

Let $k_j := \min \{k : p_k^j \geq p_m^j \quad m=1,2,\ldots,M\}$.

-- Define $\hat{>}$ on $\{1,2,\ldots,M\}$ by $k_1 \hat{>} k_2 \hat{>} \ldots \hat{>} k_M$.

The idea behind this algorithm is to send a message from the origin that is the most likely to be represented in the buffer. If this is impossible (because no such message is in the buffer), we try the next most likely origin and so on.

Here is a counterexample showing that the resulting policy is not necessarily optimal.

EXAMPLE: $N = 2 \qquad M = 3$

$$p_1 = .2 \quad p_2 = .6 \quad p_3 = .2$$

$$s_1 = (1,0,0) \quad s_2 = (0,1,0) \quad s_3 = (0,0,1)$$

$$\Pi = (.475, .05, .475)$$

$$s_1^+ = (2,0,0) \quad s_2^+ = (1,1,0) \quad s_3^+ = (1,0,1)$$
$$s_4^+ = (0,2,0) \quad s_5^+ = (0,1,1) \quad s_6^+ = (0,0,2)$$

One finds

$$\Pi^+ = (.095, .295, .19, .03, .295, .095)$$

$$p_1^1 = .58 \quad p_2^1 = .62 \quad p_3^1 = .58 \qquad k_1 = 2$$

$$p_1^2 = .285 \quad p_2^2 = 0 \quad p_3^2 = .285 \qquad k_2 = 1$$

$$p_1^3 = 0 \quad p_2^3 = 0 \quad p_3^3 = .095 \qquad k_3 = 3$$

$$\text{so} \quad 2 \hat{>} 1 \hat{>} 3$$

The resulting $H(.) = -.62 \log_2 (.62) - .285 \log_2 (.285) - .095 \log_2 (.095)$

$= 1.26$

However, the ordering $1 \hat{>} 3 \hat{>} 2$

results in the cost

$$H(.) = -.58 \log_2 (.58) - .39 \log_2 (.39) - .03 \log_2 (.03) = 1.13$$

***

At time $T-1$, or more generally at time $T-i$, $i=1,2,\ldots,T$, the receiver has computed $\Pi_{T-i}(y_{[0,T-i)})$. The transmitter must find a policy $\alpha(y_{[0,T-i)})$ minimizing (from (7))

$$C_i(y_{[0,T-i)}) + \sum_{k=1}^{M} \Pr\,(Y_{T-i}=k \,|\, Y_{[0,T-i)}=y_{[0,T-i)}) \cdot V_i(\Pi_{T-i+1}((y_{[0,T-i)},k)))$$

We have seen earlier that if policy α is used

$$C_i(y_{[0,T-i)}) = H(\Pi_{T-i}(y_{[0,T-i)}) \; P \; A^{\alpha})$$

$$\Pr\,(Y_{T-i}=k \,|\, Y_{[0,T-i)}=y_{[0,T-i)}) = \Pi_{T-i}(y_{[0,T-i)}) \; P \; B^{\alpha,k} \; u$$

$$\Pi_{T-i+1}((y_{[0,T-i)})) = \frac{\Pi_{T-i}(y_{[0,T-i)}) \; P \; B^{\alpha,k}}{\Pi_{T-i}(y_{[0,T-i)}) \; P \; B^{\alpha,k} \; u}$$

Thus policy $\alpha(y_{[0,T-i)})$ must minimize

$$H(\Pi_{T-i}(y_{[0,T-i)}) \; P \; A^{\alpha}) + \sum_{k=1}^{M} \Pi_{T-i}(y_{[0,T-i)}) \; P \; B^{\alpha,k} \; u \; V_i\left(\frac{\Pi_{T-i}(y_{[0,T-i)}) \; P \; B^{\alpha,k}}{\Pi_{T-i}(y_{[0,T-i)}) \; P \; B^{\alpha,k} \; u}\right)$$

Clearly there is an optimal policy, $\alpha_i(\Pi_{T-i}(y_{[0,T-i)}))$ which depends on $y_{[0,T-i)}$ only through $\Pi_{T-i}(y_{[0,T-i)})$. We define $V_{i+1}(\Pi)$, the minimal cost to go at time $T-i$, by

$$V_{i+1}(\Pi) = \min_{\alpha} \left(H(\Pi\ P\ A^{\alpha}) + \sum_{k=1}^{M} \Pi P\ B^{\alpha,k}\ u\ V_i \left(\frac{\Pi\ P\ B^{\alpha,k}}{\Pi\ P\ B^{\alpha,k}\ u}\right)\right) \tag{11}$$

$$= H(\Pi\ P\ A^{\alpha_i(\Pi)}) + \sum_{k=1}^{M} \Pi\ P\ B^{\alpha_i(\Pi),k}\ u\ V_i \left(\frac{\Pi P B^{\alpha_i(\Pi),k}}{\Pi P B^{\alpha_i(\Pi),k}\ u}\right)$$

At this stage we have done the following: we know how to minimize $H(Y_{[0,T)})$ in a recursive fashion over all strategies consisting of using at time i a policy $\alpha(y_{[0,i)})$. We have seen that in fact there is an optimal policy that depends on $y_{[0,i)}$ only through $\Pi_i(y_{[0,i)})$. We will now prove some properties of the V_i's.

THEOREM III

$V_i(\Pi)$ is a continuous function of Π.

Proof:

By continuity of $H(.)$ and induction on i.

Q.E.D.

THEOREM IV

Let A be a (s,t) stochastic matrix.

Then: $\Pi\ u\ H\left(\frac{\Pi A}{\Pi u}\right)$ is a concave function of Π for Π in the set of s-tuples with non-negative components.

Proof:

Let Π_1 and Π_2 be two such s-tuples;

let $(q_1^1, \ldots, q_t^1) := \Pi_1 A$

$(q_1^2, \ldots, q_t^2) := \Pi_2 A$

Then: for $\lambda \in [0,1]$

$$\lambda\Pi_1 \; u \; H\left(\frac{\Pi_1 A}{\Pi_1 u}\right) + (1-\lambda)\;\Pi_2 \; u \; H\left(\frac{\Pi_2 A}{\Pi_2 u}\right) -$$

$$- (\lambda\Pi_1 + (1-\lambda)\Pi_2) \; u \; H\left(\frac{(\lambda\Pi_1 + (1-\lambda)\Pi_2)A}{(\lambda\Pi_1 + (1-\lambda)\Pi_2)u}\right)$$

$$= \lambda \sum_{j=1}^{t} q_j^1 \log_2 \frac{\left(\sum_{i=1}^{t} q_i^1\right)\left(\lambda q_j^1 + (1-\lambda)\; q_j^2\right)}{q_j^1\left(\sum_{i=1}^{t} q_i^1 + (1- \;)\; q_i^2\right)}$$

$$+ (1-\lambda) \sum_{j=1}^{t} q_j^2 \log_2 \frac{\left(\sum_{i=1}^{t} q_i^2\right)\left(\lambda q_j^1 + (1-\lambda)\; q_j^2\right)}{q_j^2\left(\sum_{i=1}^{t} \lambda q_i^1 + (1-\lambda)\; q_i^2\right)}$$

≤ 0 because $\log_e x \leq x-1$

Q.E.D.

If $s=t$ and A is the unit matrix, this gives the well known result that $H(\Pi)$ is a concave function of Π for Π in Σ^s.

COROLLARY IV.1: Let A be a (s,t) stochastic matrix and C be a (r,s) nonnegative matrix.

Then: $\Pi Cu \; H\left(\frac{\Pi CA}{\Pi Cu}\right)$ is a concave function of Π for Π in the set of s-tuples with non negative components.

Proof:

The components of ΠC are non negative and the composition of a concave function and a linear function is concave.

Q.E.D.

COROLLARY IV.2: For all (s,σ) non negative matrix C, for all $i \geq 0$, $\Pi Cu \; V_i\left(\frac{\Pi C}{\Pi Cu}\right)$ is a concave function of Π, for Π in the set of s-tuples with non negative components.

Proof:

By induction on i :

$V_0 = 0$ thus V_0 is concave

$$V_{i+1}(\cdot) = \min_{\alpha} \left(H(\cdot PA^{\alpha}) + \sum_{k=1}^{M} \cdot PB^{\alpha,k} \; u \; V_i \left(\frac{\cdot PB^{\alpha,k}}{\cdot PB^{\alpha,k} \; u} \right) \right)$$

$$\text{so} \quad \Pi Cu \; V_{i+1}\left(\frac{\Pi C}{\Pi Cu}\right) = \min_{\alpha} \left(\Pi Cu \; H\left(\frac{\Pi CPA^{\alpha}}{\Pi Cu}\right) + \sum_{k=1}^{M} \Pi CPB^{\alpha,k} \; u \; V_i \left(\frac{\Pi CPB^{\alpha,k}}{\Pi CPB^{\alpha,k} \; u} \right) \right)$$

The terms in the right hand side member of this equation all are concave by the previous corollary and induction on i. The minimum of a set of concave functions is concave.

Q.E.D.

We are now in a position to prove that nothing is gained by using more general strategies than what we have considered until now, i.e. strategies where at time $T-i$ one uses a policy determined by $y_{[0,i)}$.

THEOREM V

Denote by $D_{i+1}(y_{[0,T-i)})$ the cost to go at time $T-i$ if one uses a given causal strategy (i.e. $\Pr(Y_i=k \mid Y_{[0,i)}=y_{[0,i)}, X_{[0,i)}=x_{[0,i)}, X_{[i,T+1)}=x_{[i,T+1)}) = \Pr(Y_i=k \mid Y_{[0,i)}=y_{[0,i)}, X_{[0,i)}=x_{[0,i)})$. Let $D_0 := 0$.

Then: $D_i(y_{[0,T-i+1)}) \geq V_i(\Pi_{T-i+1}(y_{[0,T-i+1)}))$ $\quad i=0,1,2,\ldots,T+1$

Proof:

By induction on i.

-- $D_0 \geq V_0$

-- Suppose $D_i \geq V_i$, then from (7)

$$D_{i+1}(y_{[0,T-i)}) = C_i(y_{[0,T-i)}) + \sum_{k=1}^{M} \Pr(Y_{T-i}=k \mid Y_{[0,T-i)}=y_{[0,T-i)}) \cdot D_i((y_{[0,T-i)},k))$$

$$\geq C_i(y_{[0,T-i)}) + \sum_{k=1}^{M} \Pr(Y_{T-i}=k \mid Y_{[0,T-i)}=y_{[0,T-i)}) \cdot V_i(\Pi_{T-i+1}((y_{[0,T-i)},k))) \qquad (12)$$

Let the (σ^+,M) matrix A^* be defined by

$$A^*_{nj} := \Pr(Y_{T-i}=j \mid S^+_{T-i}=s^+_n, Y_{[0,T-i)}=y_{[0,T-i)})$$

An instant of reflexion will convince the reader that A^* can be written as a convex combination of policy matrices:

$$A^* = \sum_{\alpha=1}^{\tau} c_\alpha A^\alpha \quad , \quad c_\alpha \geq 0 \quad \sum_{\alpha=1}^{T} c_\alpha = 1$$

Defining similarly the M (σ^+,σ) matrices $B^{*,k}$ $k=1,2,\ldots,M$ by $B^{*,k}_{nj} := \Pr(S_{T-i+1}=s_j \ , \ Y_{T-i}=k \mid S^+_{T-i}=s_n, Y_{[0,T-i)}=y_{[0,T-i)})$

one has that $B^{*,k} = \sum_{\alpha=1}^{\tau} c_\alpha B^{\alpha,k}$

As before $\Pr(Y_{T-i}=k \mid Y_{[0,T-i)}=y_{[0,T-i)}) = \Pi^+_{T-i}(y_{[0,T-1)})B^{*,k}u$

and by causality $= \Pi_{T-i}(y_{[0,T-1)})P\,B^{*,k}u$

and, $$\Pi_{T-i+1}((y_{[0,T-1)},k)) = \frac{\Pi_{T-i}(y_{[0,T-1)})P\,B^{*,k}}{\Pi_{T-i}(y_{[0,T-1)})P\,B^{*,k}u}$$

Thus from (12)

$$D_{i+1}(y_{[0,T-i)}) \geq H\left(\sum_{\alpha=1}^{\tau} c_\alpha(\Pi_{T-i}(y_{[0,T-i)})P\,A^\alpha)\right)$$

$$+ \sum_{k=1}^{M} \left(\sum_{\alpha=1}^{\tau} c_\alpha(\Pi_{\tau-i}(y_{[0,T-i)})P\,B^{\alpha,k}u)\right) V_i\left(\frac{\sum_{\alpha=1}^{\tau} c_\alpha(\Pi_{T-i}(y_{[0,T-i)})PB^{\alpha,k})}{\sum_{\alpha=1}^{\tau'} c_\alpha(\Pi_{T-i}(y_{[0,T-i)})PB^{\alpha,k}u)}\right)$$

By Theorem IV and Corollary IV.2, the right hand side is a concave function of $(c_1, c_2, \ldots, c_\tau)$, and thus takes its minimal value at a vertex of Σ^τ, say e_β.

Thus

$$D_{i+1}(y_{[0,T-i)}) \geq H(\Pi_{T-i}(y_{[0,T-i)})P\ A^\beta) + \sum_{k=1}^{M} \Pi_{T-i}(y_{[0,T-i)})P\ B^{\beta,k}u \cdot V_i\left(\frac{\Pi_{T-i}(y_{[0,T-i)})P\ B^{\beta,k}}{\Pi_{T-i}(y_{[0,T-i)})P\ B^{\beta,k}u}\right)$$

$$\geq V_{i+1}(\Pi_{T-i}(y_{[0,T-i)})) \qquad \text{by (11)}$$

Q.E.D.

$V_i(\Pi)$ is naturally a nondecreasing function of i; the next theorem says something about the behavior of the increase.

THEOREM VI [Odoni, 1969]

$$\min_{\Pi}\ (V_{i+1}(\Pi) - V_i(\Pi)) \geq \min_{\Pi}\ (V_i(\Pi) - V_{i-1}(\Pi))$$

$$\max_{\Pi}\ (V_{i+1}(\Pi) - V_i(\Pi)) \quad \text{Max}_{\Pi}\ (V_i(\Pi) - V_{i-1}(\Pi))$$

Proof:

From (11)

$$V_{i+1}(\Pi) = H(\Pi\ P\ A^{\alpha_i(\Pi)}) + \sum_{k=1}^{M} \Pi PB^{\alpha_i(\Pi),k}u\ V_i\left(\frac{\Pi PB^{\alpha_i(\Pi),k}}{\Pi PB^{\alpha_i(\Pi),k}u}\right)$$

$$V_i(\Pi) \leq H(\Pi\ P\ A^{\alpha_i(\Pi)}) + \sum_{k=1}^{M} \Pi PB^{\alpha_i(\Pi),k}u\ V_{i-1}\left(\frac{\Pi PB^{\alpha_i(\Pi),k}}{\Pi PB^{\alpha_i(\Pi),k}u}\right)$$

$$\text{so } V_{i+1}(\Pi) - V_i(\Pi) \geq \sum_{k=1}^{M} \Pi PB^{\alpha_i(\Pi),k}u \left(V_i\left(\frac{\Pi PB^{\alpha_i(\Pi),k}}{\Pi PB^{\alpha_i(\Pi),k}u}\right) - V_{i-1}\left(\frac{\Pi PB^{\alpha_i(\Pi),k}}{\Pi PB^{\alpha_i(\Pi),k}u}\right)\right)$$

and $\min_{\Pi} (V_{i+1}(\Pi) - V_i(\Pi)) \geq \min_{\Pi} (V_i(\Pi) - V_{i-1}(\Pi))$.

The other statement is proved by replacing $\alpha_i(\Pi)$ by $\alpha_{i-1}(\Pi)$.

Q.E.D.

Because the V_i 's are increasing, it is inconvenient to work with them numerically. We note that $\alpha_i(\Pi)$ will still minimize the right hand side member of (11) if $V_i(\Pi)$ is translated by a constant. This leads to the definition of $\hat{v}_i$ and v_i as follows.

$$v_o(\Pi) := 0$$

$$\hat{v}_{i+1}(\Pi) := \min_{\alpha} \left\{ H(\Pi P A^{\alpha}) + \sum_{k=1}^{M} \Pi P B^{\alpha,k_u} v_i\left(\frac{\Pi P B^{\alpha,k}}{\Pi P B^{\alpha,k_u}}\right) \right\} \quad i=0,1,\ldots \tag{12}$$

$$v_{i+1}(\Pi) := \hat{v}_{i+1}(\Pi) - \hat{v}_{i+1}(e_1)$$

One checks by induction that $v_{i+1}(\Pi) = V_{i+1}(\Pi) - V_{i+1}(e_1)$, and that $v_i(e_1) = 0$ for all i .

$v_i(\Pi)$ can be interpreted as the relative cost of having a state probability vector Π at time $T-i+1$.

Theorem VI can be rewritten as

$$\min_{\Pi} (\hat{v}_i(\Pi) - v_{i-1}(\Pi)) \leq \min_{\Pi} (\hat{v}_{i+1}(\Pi) - v_i(\Pi)) \leq \hat{v}_{i+1}(e_1)$$

$$\leq \max_{\Pi} (\hat{v}_{i+1}(\Pi) - v_i(\Pi))$$

$$\leq \max_{\Pi} (\hat{v}_i(\Pi) - v_{i-1}(\Pi))$$

We turn now to the discussion of the infinite T case. It is natural to assume that there exists functions α_∞ and v_∞ , and a constant g , such that

$$\lim_{i\to\infty} \alpha_i = \alpha_\infty$$

$$\lim_{i\to\infty} v_i = v_\infty$$

$$\lim_{i\to\infty} \hat{v}_i(e_1) = g$$

Then one would expect from (11) that the following relation holds:

$$g + v_\infty(\Pi) = \min_\alpha \left[H(\Pi PA^\alpha) + \sum_{k=1}^{M} \Pi PB^{\alpha,k}{}_u \, v_\infty \left(\frac{\Pi PB^{\alpha,k}}{\Pi PB^{\alpha,k}{}_u} \right) \right] \tag{13}$$

$$= H(\Pi PA^{\alpha_\infty(\Pi)}) + \sum_{k=1}^{M} \Pi PB^{\alpha_\infty(\Pi),k}{}_u \, v_\infty \left(\frac{\Pi PB^{\alpha_\infty(\Pi),k}}{\Pi PB^{\alpha_\infty(\Pi),k}{}_u} \right)$$

The optimal strategy would be to use the policy $\alpha_\infty(\Pi_i(y_{[0,i)}))$ at all times i, and one expects $\lim_{T\to\infty} \frac{1}{T} H(Y_{[0,T)}) = g$.

This is made precise in the following theorem.

THEOREM VII

If there exists a bounded real valued function v_*, a function α_* and constants g_1 and g_2 such that for Π in Σ^σ

$$g_1 + v_*(\Pi) \le \min_\alpha \left[H(\Pi PA^\alpha) + \sum_{k=1}^{M} \Pi PB^{\alpha,k}{}_u \, v_* \left(\frac{\Pi PB^{\alpha,k}}{\Pi PB^{\alpha,k}{}_u} \right) \right]$$

$$= H(\Pi PA^{\alpha_*(\Pi)}) + \sum_{k=1}^{M} \Pi PB^{\alpha_*(\Pi),k}{}_u \, v_* \left(\frac{\Pi PB^{\alpha_*(\Pi),k}}{\Pi PB^{\alpha_*(\Pi),k}{}_u} \right) \tag{14}$$

$$\le g_2 + v_*(\Pi)$$

Then:

-- the entropy $H_a(Y_{[0,T)})$ of $Y_{[0,T)}$ corresponding to using policy $\alpha_*(\Pi_i(y_{[0,i)}))$ at all times i has the property that

$$g_1 \le \liminf_{T\to\infty} \frac{1}{T} H_a(Y_{[0,T)}) \le \limsup_{T\to\infty} \frac{1}{T} H_a(Y_{[0,T)}) \le g_2 \tag{15}$$

and

$$-- \; g_1 \leq \liminf_{T\to\infty} \frac{1}{T} H_b(Y_{[0,T)}) \qquad (16)$$

where $H_b(Y_{[0,T)})$ results from a causal strategy.

Proof:

Let $\Omega := \sup_{\Pi} v_*(\Pi) - \inf_{\Pi} v_*(\Pi)$.

We define $D_o(\Pi) := 0$

$$D_{i+1}(\Pi) := H(\Pi PA^{\alpha_*(\Pi)}) + \sum_{k=1}^{M} \Pi PB^{\alpha_*(\Pi),k}_{u} \cdot D_i\left(\frac{\Pi PB^{\alpha_*(\Pi),k}}{\Pi PB^{\alpha_*(\Pi),k}_{u}}\right) \qquad i=0,1,\ldots$$

From (7): $H_a(Y_{[0,T)}) = D_T(\Pi_0)$.

We have the relation

$$D_0(\Pi) \leq v_*(\Pi) - \inf_{\Pi} v_*(\Pi)$$

and by induction on i and (13)

$$D_i(\Pi) \leq i\, g_2 + v_*(\Pi) - \inf_{\Pi} v_*(\Pi) \qquad i=1,2,\ldots$$

We can conclude that

$$\frac{D_T(\Pi_0)}{T} \leq g_2 + \frac{\Omega}{T}$$

thus proving that

$$\limsup_{T\to\infty} \frac{1}{T} H_a(Y_{[0,T)}) \leq g_2$$

i.e. the right hand part of (15).

We also have the relation

$$v_*(\Pi) - \sup v_*(\Pi) \leq V_0(\Pi)$$

and by induction on i and (14)

$$i\ g_1 + v_*(\Pi) - \sup v_*(\Pi) \leq V_i(\Pi) \qquad i=1,2,\ldots$$

We can conclude that

$$i\ g_1 - \Omega \leq V_i(\Pi_0) \tag{17}$$

Now, if (16) is not true, there is a strategy such that

$$g_1 = \lim_{T\to\infty} \inf \frac{1}{T} H_b(Y_{[0,T)}) + 2\varepsilon \qquad \varepsilon>0$$

But there is a $T \geq \frac{\Omega+1}{\varepsilon}$ such that

$$\frac{1}{T} H_b(Y_{[0,T)}) \leq \lim_{T\to\infty} \inf \frac{1}{T} H_b(Y_{[0,T)}) + \varepsilon$$

For that T,

$$T\ g_1 \geq H_b(Y_{[0,T)}) + \Omega + 1$$

$$\geq V_T(\Pi_0) + \Omega + 1$$

which contradicts (17).

Thus (16) is true, and the left hand part of (15) follows.

Q.E.D.

This theorem asserts that if one can find functions v_*, α_*, and constants g_1 and g_2, e.g. by using algorithm (12), one can bound the optimal performance, and one can find a strategy performing within $g_2 - g_1$, of the optimum. Theorem VI guarantees that $g_2 - g_1$ does not increase as one progresses in algorithm (12). Note that convergence can be hastened in (12) by damping [Schweitzer, 1971], i.e. defining $v_{i+1}(\Pi) := \lambda(\hat{v}_{i+1}(\Pi) - \hat{v}_{i+1}(e_1)) + (1-\lambda)\, v_i(\Pi)$ for some well chosen λ in $(0,1]$.

COROLLARY VII.1: If there is a bounded real valued function v_∞, a function α_∞ and a real number g such that (13) is satisfied, the strategy consisting of using policy $\alpha_\infty(\Pi_i(y_{[0,i)}))$ at all times i

is optimal, and $\lim_{T\to\infty}\frac{1}{T}H(Y_{[0,T)}) = g$.

Proof:

Make $g_1 = g_2 = g$ in Theorem VII.

Q.E.D.

Note that nothing in this corollary guarantees the existence of an optimal strategy.

Note also that if a policy $\alpha(\Pi_i)$ is used at all times i, the Π_i's themselves form a stationary Markov Process in the simplex of $\mathbb{R}^\sigma$, and the probability distribution of Π_i can be computed. Our problem can be seen as a Markovian decision theory problem with observable state (i.e. Π_i) . These problems have been extensively studied especially in the finite dimensional case (see [Kushner, 1971]). Contrary to what is usually done, the proof of Theorem VII carefully avoids the use of the stationary distribution of the Π_i's, which is not guaranteed to exist, because the hypotheses are not very restrictive.

EXAMPLE: $M = 2$ $N = 2$ as before

Let $\Pi = (\rho_1\rho_2)$.

Equation (12) takes the form, where we use $v_\infty(\rho_1)$ in place of $v_\infty((\rho_1,\rho_2))$:

$$g + v_\infty(\rho_1) = \min\left[H(p_1+\rho_1p_2) + (1-\rho_1)p_2\ v_\infty(0) + (p_1+\rho_1p_2)v_\infty\left(\frac{\rho_1p_1}{p_1+\rho_1p_2}\right),\right.$$

$$\left. H(\rho_1p_1) + \rho_1p_1\ v_\infty(1) + (1-\rho_1p_1)\ v_\infty\left(\frac{p_1+\rho_1(p_2-p_1)}{1-\rho_1p_1}\right)\right]$$

The first argument in $\min(.,.)$ corresponds to policy 1, the second to policy 2.

***

We have solved numerically this example for different values of p_1 by discretizing the unit simplex of $\mathbb{R}^2$ (51 points) and using the algorithm (12). Results appear in Table 5.2.

Table 5.2

$g := \lim_{T\to\infty} \frac{1}{T} H(Y_{[0,T)})$ corresponding to an optimal strategy

M=2 N=2

p_1	g
.5	.60
.6	.58
.7	.51
.8	.41
.9	.25
.95	.14

In all cases, an optimal strategy turns out to be:

use policy 2 when $\rho_1 \leq .5$

Note that, if $p_1 = .5$, this is exactly what Strategy V of Section C does.

For $p_1 \neq .5$ this result shows that the strategy of always minimizing the expected immediate cost is not optimal.

It would be pleasant to prove analytically that the strategy described above is optimal. In the case $p_1 = p_2 = .5$,this would involve finding a bounded function v_∞ and g verifying

$$g + v_\infty(\rho_1) = H(\tfrac{\rho_1}{2}) + \tfrac{\rho_1}{2} v_\infty(1) + \tfrac{1}{2}(2-\rho_1)\, v_\infty(\tfrac{1}{2-\rho_1})$$

for $\rho_1 \leq .5$, and a similar expression for $\rho_1 \geq .5$. By symmetry one

expects $v_\infty(x) = v_\infty(1-x)$, so v_∞ and g must satisfy

$$g + v_\infty(\rho_1) = H(\frac{\rho_1}{2}) + \frac{\rho_1}{2} v_\infty(0) + \frac{1}{2}(2-\rho_1) v_\infty\left(\frac{1-\rho_1}{2-\rho_1}\right) \quad ,$$

$$\rho_1 \leq .5$$

In this expressions, all the arguments of v_∞ are between 0 and .5. Once this function is found, one should prove that it satisfies (13).

Before closing this section, we make a brief historical review. Our problem is essentially the problem of controlling a Partially Observable Markov Process. We solve it by working in the simplex of $\mathbb{R}^\sigma$, where the Π_i 's form a Markov Process if a policy $\alpha_i(\Pi_i)$ is used at times i . The problem is thus "reduced" to a Markov decision problem with observable state. The idea of doing this has become classical starting with [Drake, 1962]. One can find more references in Section 4 of [Platzman, 1976]. This last work is an attempt to control Partially Observable Markov Processes without making the transformation to the Π space, and is also an excellent review of the state of the art.

We should point out that the Partially Observable Markov Processes studied in the literature are simpler than what is considered here, because their immediate cost is only a function of the state of the original process, and the policy. Thus the expected immediate cost at time T-i if policy α is used has the form

$$C_i(y_{[0,T-i)}) = \Pi_{T-i}(y_{[0,T-i)}) \cdot q^\alpha$$

for some column σ-tuple q^α .

This compares with $C_i(y_{[0,T-i)}) = H(\Pi_{T-i}(y_{[0,T-i)})\ P\ A^\alpha)$ in our case. However the nice properties of continuity and concavity of the functions

$V_i(\Pi)$ in the simpler problem are conserved here. [Platzman, 1976] gives sufficient conditions on the matrices $PB^{\alpha,k}$ for an optimal solution to exist in the simpler case; it seems that these conditions would still be sufficient here. However, they are extremely cumbersome to verify.

G. Suggestions for Future Work.

Although it is not of immediate practical use, it would be worthwhile to prove that an optimal solution exists, that it verifies (13), and that v_∞ is continuous and concave.

It would be especially interesting to find analytic expressions for v_∞ and α_∞, at least for simple cases. We conjecture that $v_\infty(e_k) = v_\infty(e_1)$, $k=1,2,\ldots,\sigma$, i.e. that the relative values of perfect state knowledge are the same, regardless of the state.

One should try to prove or disprove the possibility that an $\alpha_\infty(\Pi)$ always belongs to the special class of policies considered in Theorem II.

Finally, we assumed until now that the p_i's were known. One should find robust strategies (e.g. a minimax strategy) that could be used when the source statistics are imperfectly known.

4. Analysis of Practical Strategies

A. Notation and Organization

Throughout Sections 4 to 8 we will consider a model where source i, $i=1,2,..\ M$, emits messages in a Poisson manner with rate $\lambda_i := 1/Ea_i, \lambda_T := \sum_{i=1}^{M} \lambda_i$, where every message contains a codeword indicating its length and where the lengths of the messages from source i have a probability distribution B_i. We assume that the message lengths and interarrival times are independent random variables. We will attempt to compute the expected message waiting time for different strategies indicating the message origins.

In Section B we will quickly study the equivalent of strategy I of Section 3.B: the concentrator transmits the messages in the order they were received, and prefixes each of them with a codeword indicating its origin.

In Section 5 we analyse some variants of Strategy II of Section 3.B; periodically the concentrator sends a codeword indicating the state of its buffer, then empties it. This will lead to a source coding problem of independent interest that will be treated in Section 6.

Section 7 will see the computation of the average message waiting time in cyclic strategies, where the concentrator serves all messages from source i present in the buffer, then all messages from source $i+1$, and so on. Finally we will discuss all results in Section 8.

B. Analysis of the First-In-First-Out Strategy

We send the messages in the order they were received, prefixing a message from source i with a codeword of length n_i. We must also specify what to do when the line is idle. In that case we use the same policy as in Section 2 of Chapter 4, i.e. we insert a flag of length ν_1 at the end of a busy period, then flags of length ν_2 if no arrival occurred during the transmission of the previous flag. Note that the flags and the codewords must be chosen jointly, so that the probability of an insertion in a message will depend on the origin of the message. We denote by p_i^j the probability that the flag of length ν_j causes an insertion in a message from source i.

We will use the formulas developed in Chapter 4 to compute the average message delay with the following identification:

$b' = 0$ (we include the message lengths in f_1 and f_2)

f_2 = message length + codeword length + possible insertion due to the flag of length ν_1

thus

$$Ef_2 = \frac{1}{\lambda_T} \sum_{i=1}^{M} \lambda_i (Eb_i + n_i) + p_i^1)$$

$$Ef_2^2 = \frac{1}{\lambda_T} \sum_{i=1}^{M} \lambda_i (E(b_i+n_i)^2 + p_i^1 + 2p_i^1 E(b_i+n_i))$$

f_1 will be defined as in Section 4.4 with

$c_j = \nu_j \quad j=1,2.$

$d_1 = d_2$ = message length + codeword length + possible insertion due to the flag of length ν_2

thus $$Ef_1 = -\frac{1}{\lambda_T} + \nu_1 + \frac{e^{-\lambda_T\nu_1}}{1-e^{-\lambda_T\nu_2}}\nu_2 + \frac{1}{\lambda_T}\sum_{i=1}^{M}\lambda_i(Eb_i + n_i + p_i^2)$$

$$Ef_1 + \frac{1}{2}\lambda_T Ef_1^2 = \nu_1^2 + 2\nu_1/\lambda_T \sum_{i=1}^{M}\lambda_i(Eb_i + n_i + p_i^2)$$

$$+ \frac{e^{-\lambda_T\nu_1}}{1 - e^{-\lambda_T 2}}\left\{\nu_2^2 + 2\nu_2/\lambda_T \sum_{i=1}^{M}\lambda_i(Eb_i+n_i+p_i^2)\right\}$$

$$+ \frac{1}{\lambda_T}\sum_{i=1}^{M}\lambda_i(E(b_i+n_i)^2 + p_i^2 + 2p_i^2E(b_i+n_i))$$

and one obtains from formula (1) of Chapter 4 that the average message delay is given by

$$E(d) = \frac{1}{\lambda_T}\sum_{i=1}^{M}\lambda_i(Eb_i+n_i) + \frac{1}{2}\frac{\sum_{i=1}^{M}\lambda_i(E(b_i+n_i)^2 + p_i' + 2p_i'E(b_i+n_i))}{1 - \sum_{i=1}^{M}\lambda_i(Eb_i+n_i+p_i')}$$

$$+ \frac{1}{2}\frac{\nu_1^2 + \frac{2\nu_1}{\lambda_T}\sum_{i=1}^{M}\lambda_i p_i^2 + \frac{e^{-\lambda_T\nu_1}}{1-e^{-\lambda_T\nu_1}}(\nu_2^2 + 2\frac{\nu_2}{\lambda_T}\sum_{i=1}^{M}\lambda_i p_i^2) + }{\nu_1 + \frac{e^{-\lambda_T\nu_1}}{1 - e^{-\lambda_T\nu_2}}\nu_2 +}$$

$$\frac{\frac{1}{\lambda_T}\sum_{i=1}^{M}\lambda_i(p_i^2 - p_i')}{\frac{1}{\lambda_T}\sum_{i=1}^{M}\lambda_i(p_i^2 - p_i')} \qquad (3)$$

If $\nu_1 = \nu_2$ the last term simplifies to

$$\frac{\nu_1}{2} + \sum_{i=1}^{M}\frac{\lambda_i}{\lambda_T}p_i' \qquad (4)$$

It is of interest to see how $E(d)$ behaves in light and in heavy traffic. In light traffic, the second term is negligible, so one sees that the codewords should come from a Huffman code, so as to minimize $\sum_{i=1}^{M} \lambda_i n_i$. ν_1 and ν_2 should be small, say $\nu_1=\nu_2=1$, or $\nu_1=\nu_2=2$, as we will discuss in Section 4.7. If all λ_i 's are more or less equal,

$$E(d) \cong \frac{1}{\lambda_T} \Sigma\lambda_i \, Eb_i + \log_2 M + 1.5$$

and increases with $\log_2 M$.

In heavy traffic the second term will dominate, and it will be of primary importance to maximize its denominator, thus again using a Huffman code, and using a large ν_1 . If all λ_i 's are more or less equal, we can have stability if $\Sigma\lambda_i(Eb_i+\log_2 M) < 1$. Thus if Eb_i is of the order of $\log_2 M$ or smaller, the maximum throughput of the system will be much reduced by the presence of the codewords.

5. Strategies Indicating the Buffer State.

A. Introduction

We study in this section a class of strategies where periodically the concentrator samples the buffer, makes known the state of the buffer to the deconcentrator, then transmits all the messages that were present in the buffer at the sampling time.

In addition to the notation introduced in Section 4.A, we call the time intervals between two sampling points the scanning times, and we denote them by s_i , i=1,2,.. We denote by m_j^i the number of

arrivals from source j during s_i, and by ν_τ the (variable) length of the codeword used to indicate the state of the buffer, i.e. the m_j^i's $j=1,2,..M$ at the end of s_i. Note that s_i is known to the receiver.

Thus an interesting problem is to find a code minimizing $E(\nu_\tau|s_i)$. This code will be very complicated, because it will jointly encode the m_j^i's. However, since the m_j^i's are conditionally independent given s_i, nothing will be lost by encoding separately, except some redundancy.

If we encode the m_j^i separately, the problem is to find a minimum average codeword length code for a Poisson random variable. This is still challenging because the number of codewords will be infinite, so that Huffman's procedure [Huffman, 1952] cannot be applied directly. We solve this problem in Section 6.

Here our goal is to find the average message delay, and we proceed to do so.

B. Statistics of the Scanning Times

Because the arrivals are Poisson, the scanning times form a Markov chain which is irreducible because, for any value of s_i, there is a non zero probability of no arrivals during s_i.

We have the relation

$$E(e^{-xs_{i+1}}|m_1^i,m_2^i\ldots m_M^i,s_i) = E(e^{-x\nu_\tau}|m_1^i m_2^i\ldots m_M^i,s_i) \cdot \prod_{j=1}^{M} (B_j^*(x))^{m_j^i} \qquad x \geq 0 \qquad (5)$$

Of course we want to average this, which is possible analytically only if $E(e^{-x\nu_T}|m_1^i\ldots m_M^i,s_i)$ has a sufficiently simple form. In particular

it is not possible if ν_T results from the algorithm of Section 6. We will restrict ourselves to strategies where $E(e^{-x\nu_T} | m_1^i, \ldots, m_M^i, s_i)$ has the form

$$\prod_{j=1}^{M} V_{0j}^*(x) \; (V_{1j}^x(x))^{m_j^i} \qquad (6)$$

We will also require that $\sum_{j=1}^{M} E\, \nu_{0j} > 0$. Otherwise infinitely many scanning times could take place in no time. Without causing any difficulty we could add a factor $(V^*(x))^{s_i}$ in (6), but this would be fruitless. We will examine codes that have the above property after finishing the analysis of the scanning times statistics.

We can now average (5) on m_j^i, the number of Poisson arrivals during s_i, to obtain

$$E\left(e^{-xs_{i+1}} \middle| s_i\right) = \left(\prod_{j=1}^{M} V_{0j}^*(x)\right) \exp\left(s_i \sum_{j=1}^{M} \lambda_j (V_{1j}^*(x) B_j^*(x) - 1)\right)$$

Denoting $\left(E\, e^{-xs_i}\right)$ by $S_i^*(x)$ we have $\qquad \mathrm{Re}\, x \geq 0 \qquad (7)$

$$S_{i+1}^*(x) = \left(\prod_{j=1}^{M} V_{0j}^*(x)\right) S_i^*\left(\sum_{j=1}^{M} \lambda_j (1 - V_{1j}^*(x)\, B_j^*(x))\right) \qquad \mathrm{Re}\, x \geq 0$$

Defining $V_o^*(x) := \prod_{j=1}^{M} V_{0j}^*(x)$

$$f^0(x) := x$$

$$f^1(x) := \sum_{j=1}^{M} \lambda_j (1 - V_{1j}^*(x)\, B_j^*(x)) \qquad \mathrm{Re}\, x \geq 0$$

$$f^i(x) := f^1(f^{i-1}(x)) \qquad i \geq 1 \qquad \mathrm{Re}\, x \geq 0$$

We can rewrite (7) as

$$S_i^*(x) = \prod_{j=0}^{i-1} V_o^*(f^j(x))\ S_o^*(f^i(x)) \qquad \text{Re } x \geq 0$$

We will show now that if $\rho_T := \sum_{j=1}^{M} \lambda_j (E v_{1j} + E b) < 1$ and if $\lim_{x \downarrow o} \prod_{i=o}^{\infty} V_o^*(\rho_T^i x) = 1$, then, for x real, $S^*(x) := \lim_{i \to \infty} S_i^*(x)$ is independent of S_o^* and is continuous at 0. Thus [Feller, 1966, p. 431] the process s_i is positive recurrent and S^* is given by

$$S^*(x) = \prod_{j=o}^{\infty} V_o^*(f^j(x)) \qquad \text{Re } x \geq 0$$

which is suitable for numerical computation. The proof is simple: by convexity one has immediately that $f^1(x) \leq \sum_{j=1}^{M} \lambda_j \left[\frac{d}{dy} (V_{1j}^*(y) B_j^*(y)) \Big|_{y=0} \right] x$ $= \rho_T x$. Thus $f^i(x) \leq \rho_T^i x$ and $\lim_{i \to \infty} f^i(x) = 0$, so $\lim_{i \to \infty} S_o^*(f^i(x)) = 1$. To be able to use the reference just mentioned we need $\lim_{x \downarrow o} \prod_{j=1}^{\infty} V_o^*(f^j(x))$ $= 1$, which is insured by $\lim_{x \downarrow o} \prod_{j=o}^{\infty} V_o^*(\rho^j x) = 1$, because $V_o^*(x)$ is decreasing and upperbounded by 1 for $x \geq 0$. Note that this condition and the continuity of $S^*(x)$ at 0 are guaranteed if $E\, v_o < \infty$ but this is not necessary.

Note also that if $\rho_T = 1$, $f^i(x) = x + o(x)$; thus if $\prod_{j=o}^{\infty} V_o^*(f^j(x))$ converges to a number different from 0, S^* will depend on S_o^*, whereas if $S^*(x) = 0$ $x > 0$, S^* is not the Lapalce-Stieltjes transform of a probability distribution. At any rate, the process s_i is not positive recurrent if $\rho_T = 1$.

From (5), if the process is positive recurrent, S^* satisfies the relation

$$S^*(x) = V_o^*(x)\ S^*\left[\sum_{j=1}^{M} \lambda_j (1 - V_{1j}^*(x)\ B^*(x))\right] \qquad (8)$$

and is a decreasing function of x, as is V_o^*.

Thus (8) implies that

$$x > \sum_{j=1}^{M} \lambda_j (1 - V_{1j}^*(x)\ B_j^*(x)) \qquad x > 0 ,$$

Dividing by x and taking the limit as $x \downarrow o$, we have that $\rho_T \leq 1$.

We can conclude that $\rho_T < 1$ is a necessary condition for the s_i process to be positive recurrent. We are unable to prove that it is sufficient; we still need a condition on V_o^*. From now on we will assume that $\rho_T < 1$ and $\sum_{j=1}^{M} E\, v_{oj} < \infty$, and we will consider only the stationary system (i.e. $S_o^* = S^*$).

Taking the values at $x=0$ of the derivatives of S^* in (8) one finds

$$E\, s = \frac{\sum_{j=1}^{M} E\, v_{oj}}{1 - \sum_{j=1}^{M} \lambda_j (E\, v_{1j} + Eb_j)}$$

$$E\, s^2 = (Es)^2 + \frac{(Es) \sum_{j=1}^{M} \lambda_j\, E(v_{1j} + b_j)^2 + \sum_{j=1}^{M} \mathrm{var}\,(v_{oj})}{1 - \left[\sum_{j=1}^{M} \lambda_j (E\, v_{1j} + Eb_j)\right]^2} \qquad (9)$$

C. Description of the Code

A coding scheme that satisfies (6) is to use a unary code to encode each m_j^i. In that case $V_{oi}^*(x) = V_{1i}^*(x) = e^{-x}$. Note that

it is not necessary to transmit all the codewords at the beginning of the scanning time. We can transmit first the codeword specifying m_1^i, then the m_1^i messages from source 1, etc. A more efficient form of the same code is to prefix every message with a "1", and to transmit at "0" when all messages from source 1 have been transmitted. This has a favorable effect on the message waiting times.

We will consider a generalization of this strategy, using flags. We transmit first all messages from source 1, then a flag of length ν_1, then all messages from source 2, etc. Under the usual assumptions (see Chapter 3)

$$V_{0j}^*(x) = \exp(-x\nu_j) \quad , \quad V_{1j}^*(x) = 1 - 2^{-(\nu_j - 1)} (1 - e^{-x}) \qquad j=1,2...M$$

D. The Waiting Time

If the service discipline for each source is first in first out, the waiting time w_i of a message arriving from source i u units of time before the end of a scanning time of length z is equal to u plus the sum of the lengths and insertions of the messages from sources $1,2,...i-1$ that arrive during z, plus the sum of the lengths and insertions of the messages from i that are already in the queue, plus the flags $1,2,...i-1$ plus a possible insertion. Thus

$$E(e^{-xw_i} \mid u,z) = e^{-xu} \exp\left[-z \sum_{j=1}^{i-1} \lambda_j (1 - V_{1j}^*(x)\ B_j^*(x))\right] \cdot$$

$$\exp(-(z-u)\ \lambda_i (1 - V_{1i}^*(x)\ B_i^*(x)) \cdot \prod_{j=1}^{i-1} V_{0j}^*(x) \cdot V_{1i}^*(x) \qquad \mathrm{Re}\ x \geq 0$$

u is uniformly distributed between 0 and z , because the arrivals are Poisson, so

$$E(e^{-w_i x} | z) = \prod_{j=1}^{i-1} V_{0j}^*(x) \exp(-z \sum_{j=1}^{i-1} \lambda_j (1 - V_{1j}^*(x) B_j^*(z))) .$$

$$V_{1i}^*(x) \frac{\exp(-z\lambda_i (1 - V_{1i}^*(x) B_i^*(x))) - \exp(-zx)}{z(x + \lambda_i (V_{1j}^*(x) B_j^*(x) - 1))}$$

Using the statistics of z developed in Appendix B one obtains:

$$W_i^*(x) := E(e^{-xw_i}) = \frac{\prod_{j=1}^{i-1} V_{0j}^*(x) \left[S^* \left[\sum_{j=1}^{i} \lambda_j (1 - V_{1j}^*(x) B_j^*(x)) \right] - S^* \left[x + \sum_{j=1}^{i-1} \lambda_j (1 - V_{1j}^*(x) B_j^*(x)) \right] \right]}{Es(x + \lambda_i (V_{1i}^*(x) B_i^*(x) - 1))} \cdot V_{1i}^*(x)$$

Differentiating one obtains the moment

$$Ew_i = \sum_{j=1}^{i-1} Ev_{0j} + \frac{E s^2}{E s} \left[\frac{1 + \rho_i}{2} + \sum_{j=1}^{i-1} \rho_j \right] + E v_{1i} \qquad (10)$$

where $\rho_i := \lambda_i (EV_{1i} + Eb_i)$

One can find from this an expression for the average message waiting time, $Ew := \frac{1}{\lambda_T} \sum_{i=1}^{M} \lambda_i Ew_i$. In general this expression is quite long to write and depends on the ordering of the sources. The only statement that we are able to make about the ordering that minimizes the

average waiting time is that if $Ev_{0j} = Ev_{0i}$ and $\rho_i = \rho_j$ one should have $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$, or equivalently, $Ev_{11} + Eb_1 \leq Ev_{12} + Eb_2 \leq \ldots \leq Ev_{1M} + Eb_M$, as expected.

If $\lambda_i = \lambda$, $Ev_{0i} = Ev_0$ and $\rho_i = \rho$ and $Ev_{1i} = Ev_1$, one checks that

$$Ew = \frac{M-1}{2} Ev_0 + \frac{Es^2}{2Es} (1 + \rho_T)$$

If in addition $E(v_{1j} + b_j)^2 = \Theta$ and var $v_{0j} = \sigma^2$, we can use (9) and (10) to obtain

$$Ew = \frac{M-1}{2} Ev_0 + \frac{MEv_0}{1 - \rho_T} \frac{1+\rho_T}{2} + \frac{\lambda_T\Theta + M\sigma^2/Es}{2(1-\rho_T)} + Ev_1$$

$$= -\frac{Ev_0}{2} + \frac{MEv_0}{1 - \rho_T} + \frac{\lambda_T\Theta + M\sigma^2/Es}{2(1 - \rho_T)} + Ev_1 \qquad (11)$$

Thus if in our coding scheme we use flags of length ν, and if the sources are identical, the average waiting time is given by

$$Ew = 2^{-(\nu-1)} - \frac{\nu}{2} + \frac{M\nu}{1 - \lambda_T(Eb + 2^{-(\nu-1)})} + \frac{\lambda_T(Eb^2 + 2Eb\ 2^{-(\nu-1)} + 2^{-(\nu-1)})}{1 - \lambda_T(Eb + 2^{-(\nu-1)})}$$

One sees that in light traffic the first two terms will dominate, especially when M is large. In heavy traffic, the presence of the protocol does not affect the capacity of the line if one chooses ν large enough.

For finite source alphabets, the Huffman coding algorithm [Huffman, 1952] yields a minimum expected codeword length code satisfying the prefix condition. Although it cannot be applied directly to countably infinite alphabets, its optimality can be used to develop optimal codes for these sources, as [Golomb,1966] and [Gallager and Van Voorhis, 1975] did in the case of geometric probability distributions. We show that for a large class of probability measures, including those whose tail decreases faster than geometrically with a ratio equal to .618, the coding problem can be reduced to a finite one, to which Huffman's procedure is applicable. This result hinges on the observation that if the tail of a probability measure decreases monotonically, no matter how fast, the codeword lengths of an optimum code must not increase faster than linearly, with slope 1, for otherwise some prefixes will not be used. This leads to the coding procedure developed in Theorem 1.

Theorem 1

Let $p(.)$ be a probability measure on the set of nonnegative integers. Assume there is a nonnegative integer m such that for all $j > m$ and $i < j$, the following hold:

$$p(i) \geq p(j) \tag{1a}$$

$$p(i) \geq \sum_{k=j+1}^{\infty} p(k) \tag{1b}$$

Then a binary prefix condition code with minimum average codeword length for the alphabet consisting of the nonnegative integers with the above probabilities is obtained by the following procedure:

Consider the reduced alphabet with letters $0,1, \ldots, m+1$ whose probabilities are

$$p_1(i) = p(i) \qquad i \leq m$$

$$p_1(m+1) = 1 - \sum_{i=0}^{m} p(i)$$

Apply Huffman's coding procedure [Huffman,1952] to this reduced alphabet. Denote by $C_1(i)$ and $\ell_1(i)$ respectively the codeword and codeword length for letter i ($C_1(i)$ is a sequence of $\ell_1(i)$ binary symbols) $0 \leq i \leq m+1$.

From there, construct the codewords $C(i)$ for the original alphabet by

$$\begin{aligned} C(i) &= C_1(i) && i \leq m \\ C(i) &= \{C_1(m+1),(i-m-1)*0,\ 1\} && i > m \end{aligned} \qquad (2)$$

where $n*0$ denotes a sequence of n 0's.

Moreover, with this procedure the average codeword length $\bar{\ell}$ for the original alphabet is given by

$$\bar{\ell} = E(i) + \ell_1(m+1) - m + \sum_{i=0}^{m} (\ell_1(i) - \ell_1(m+1) + m - i)p(i)$$

$$\text{where } E(i) = \sum_{i=0}^{\infty} ip(i) < m+2$$

Proof

It is a simple matter to check that $\bar{\ell}$ is as given, and that, because of hypothesis (1b), $E(i)$ is finite:

$$E(i) = \sum_{i=0}^{m} \sum_{k=i+1}^{\infty} p(k) + \sum_{i=m+1}^{\infty} \sum_{k=i+1}^{\infty} p(k)$$

$$\leq \sum_{i=0}^{m} \sum_{k=i+1}^{\infty} p(k) + \sum_{i=m+1}^{\infty} p(i-1)$$

$$< m+2$$

The codewords $C_1(i)$ satisfy the prefix condition, so it is clear that the codewords $C(i)$ also do. We show now that this code has minimum average length, using the same technique as Gallager and Van Voorhis [3].

Let the letters $0,1 \ldots, m+r$ of the "r-reduced" alphabet have probabilities:

$$p_r(i) = p(i) \qquad i < m+r$$

$$p_r(m+r) = \sum_{i=m+r}^{\infty} p(i)$$

The hypothesis ensures that, as long as r is greater than or equal to 1, the smallest probabilities are $p_r(m+r-1)$ and $p_r(m+r)$. Applying Huffman's procedure to the r-reduced alphabet, one verifies that the codeword lengths of the first $m+r$ letters in this alphabet are the same as the lengths of the corresponding code-words given in (2). So, denoting by $\overline{\ell_r}$ the average codeword length for the r-reduced alphabet, $\overline{\ell_r}$ converges to $\overline{\ell}$ as r grows.

Let $\overline{\ell_o}$ be the minimum average codeword length for the original alphabet, the minimum being taken over all uniquely decodable codes, so that $\overline{\ell} \geq \overline{\ell_o}$. We claim that $\overline{\ell_r} \leq \overline{\ell_o}$ because we can obtain a uniquely decodable code for the r-reduced alphabet by taking as codewords for letters 0 to $m+r-1$ the corresponding codewords in the optimum code,

and choosing as codeword for letter $m+r$ the shortest remaining codeword in the optimum code. The average codeword length of the code so obtained is not larger than $\overline{\ell_o}$, and is not smaller than $\overline{\ell_r}$, since Huffman's procedure yields an optimal code. We conclude that $\overline{\ell_r} \leq \overline{\ell_o}$, but $\overline{\ell_r}$ converges to $\overline{\ell}$ as r increases, so $\overline{\ell} \leq \overline{\ell_o}$. Recalling that $\overline{\ell} \geq \overline{\ell_o}$, $\overline{\ell} = \overline{\ell_o}$. Q.E.D.

The question then arises: how rapidly must $p(.)$ decrease in order to satisfy the hypothesis? A sufficient condition is that it satisfies $p(i) \geq p(i+1) + p(i+2)$ for large i; a weaker condition is that it decreases at least as fast as g^i where $g = \frac{1}{2}(\sqrt{5}-1) = .61803$ If $p(i) = p(i+1) + p(i+2)$, then $p(i) = p(0)\, g^i$, and hypothesis (1b) is satisfied with equality for all i and $j = i+1$.

In particular, the coding procedure developed in Theorem 1 is optimum when the probability measure is Poisson:

$$p(i) = \frac{\lambda^i e^{-\lambda}}{i!} \qquad i=0,1,\ldots$$

The only problem is to find the smallest suitable value for m (as defined in Theorem 1). One checks easily that $p(i)$ increases with i to a maximum value of $p(r)$, where $r = \lceil \lambda \rceil - 1$, and then decreases ($\lceil x \rceil$ denotes the smallest integer not smaller than x). If n is the smallest positive integer such that $p(n) \leq p(0)$, the smallest we can hope m to be is $n-1$ (a smaller m will not satisfy hypothesis (1a)). Fortunately, this is so, and we can upperbound this m by $\lceil e\lambda \rceil - 1$, as the following theorem will show. The size of the reduced alphabet for which we must execute Huffman's procedure and maintain a codeword table is thus a reasonable function of λ. In table 5.3 we present λ as a

m	upper limit of λ for that m	m	upper limit of λ for that m
0	1.0000	15	6.8004
1	1.4142	16	7.1770
2	1.8171	17	7.5531
3	2.2133	18	7.9289
4	2.6051	19	8.3043
5	2.9937	20	8.6794
6	3.3800	21	9.0542
7	3.7643	22	9.4287
8	4.1471	23	9.8030
9	4.5287	24	10.177
10	4.9092	25	10.550
11	5.2888	26	10.924
12	5.6676	27	11.298
13	6.0458	28	11.671
14	6.4234	29	12.044

Table 5.3

Relation between λ and m for Poisson distributions

function of m for small values of λ. In particular, if $\lambda \leq 1$, then $m=0$ so that the optimum code is unary and its average codeword length is equal to $1+\lambda$.

Theorem 2

If $p(i) = \frac{\lambda^i e^{-\lambda}}{i!}$ $\qquad i=0,1,\ldots$

and m is the smallest nonnegative integer such that $p(m+1) \leq p(0)$, then

a) $\lceil 2\lambda \rceil - 2 \leq m \leq \lceil e\lambda \rceil - 1$

b) $p(i) \geq \sum_{j=i+1}^{\infty} p(j) \qquad i > m$

and thus (1) is satisfied by this m.

Proof

a) We will first upperbound m. By Stirling's inequality [Feller, 1968, p. 52]

$$i! > \left(\frac{i}{e}\right)^i \cdot (2\pi i)^{\frac{1}{2}} > \left(\frac{i}{e}\right)^i \qquad i=1,2,3\ldots$$

If $i \geq e\lambda$, then $i! > \lambda^i$, so that $p(0) > p(i)$ and thus $m+1 \leq \lceil e\lambda \rceil$. (A more careful analysis shows that, when λ is large, m is approximately equal to $e\lambda - \frac{1}{2} \log 2\pi e\lambda - 1$.)

To lowerbound m, we note that the logarithm function is concave downward so that $\log \frac{i+1}{2} = \log \left(\frac{1}{i} \sum_{j=1}^{i} j \right) \geq \frac{1}{i} \sum_{j=1}^{i} \log j = \frac{1}{i} \log i!$ If $p(i) \leq p(0)$, then $\frac{1}{i} \log i! \geq \log \lambda$ so that

$$\log \frac{i+1}{2} \geq \log \lambda , \tag{3}$$

and thus $\frac{m+2}{2} \geq \lambda$.

b) $$\sum_{j=i+1}^{\infty} p(j) = \frac{e^{-\lambda}\lambda^{i}}{i!}\left[\frac{\lambda}{i+1} + \frac{\lambda^{2}}{(i+1)(i+2)} + \ldots\right]$$

$$\leq p(i)\left[\frac{\lambda}{i+1} + \frac{\lambda^{2}}{(i+1)^{2}} + \frac{\lambda^{3}}{(i+1)^{3}} + \ldots\right]$$

$$= p(i)\,\frac{\frac{\lambda}{i+1}}{1 - \frac{\lambda}{i+1}}$$

From inequality (3), if $p(i) \leq p(0)$, $\frac{\lambda}{i+1} \leq \frac{1}{2}$ and $\frac{\frac{\lambda}{i+1}}{1 - \frac{\lambda}{i+1}} \leq 1$.

This yields the desired result.

Q.E.D.

7. Analysis of Cyclic Strategies

A. Introduction

We give in Sections A to F a complete analysis of the average message waiting time for two important cyclic queueing systems. No explicit reference to the application of these systems to the encoding of message origins is made before Section G.

Communication and computer systems in which a single server is shared among several queues are common. For example, in a concentrator, messages arriving from many sources must be sent on one output line. In time-shared computer systems, a central computer must provide service to several users. The queueing models presented here may be useful in the analyses of these and similar problems.

Consider a node with M incoming communication links and one outgoing link. Digital messages arriving on link i are queued in a buffer i of infinite capacity. Periodically a "server" empties the queues and transmits the messages on the outgoing link. We will study the average waiting time in each queue under two service disciplines. In the first, referred to as the "please wait" discipline, the server serves only those messages already present in queue i when he arrives, then switches to queue $i+1$, which takes a random time, and goes on in

cycle, visiting each queue once in a cycle. In the second discipline, the "exhaustive service" discipline, the server empties queue i completely, then spends a random time switching to $i+1$ and continues the cycle. The random time between queues can be viewed as being used for transmission of protocol.

In both cases the i^{th} queue is characterized by a Poisson input with parameter λ_i messages per time unit and a service time with mean $\frac{1}{\mu_i}$ time units per message and second moment Θ_i. The switching times to queue i have mean ν_i time units and variance σ_i^2. We assume all interarrival, service and switching times to be independent.

Approximate studies have been made by [Leibowitz, 1961] and [Kruskal, 1969]. [Cooper and Murray, 1969] and [Cooper, 1970] studied both disciplines in the case of zero switching times. [Eisenberg, 1972] considered a more general configuration for the server cycle and allowed non zero switching times. He solved the problem of the exhaustive service discipline . [Konheim and Meister, 1974] solved the discrete-time equivalent of the exhaustive service problem in the case where the queues are identical. In addition, numerous authors referred to in [Eisenberg, 1972] studied the system of queues when $M=2$.

This research was pursued before the publication of [Carsten et al, 1977], which analyzes the "please wait" case by a method similar to ours. The rate of convergence of the algorithm presented in the paper just mentioned is not as claimed there, as will be shown in Section F.

Our solution differs from previous studies in the fact that we use a direct approach, without trying to find the Laplace-Stieltjes transforms of the waiting time distributions. We will show that we can find all average waiting times by solving a single system of about M^2

linear equations and we present a practical method of doing so. We remark that our results can be applied to the case of zero switching times and have a very simple form when the queues are identical.

In many communication systems, like computer networks, beside transmitting messages, one must also convey their origins or destinations. This can significantly increase the incurred delay. We will show how the previous queueing disciplines can be applied to reduce this overhead.

In Section B we present some relations valid for both disciplines. The "please-wait" case is treated in Section C and the "exhaustive-service" discipline in Section D. In Section E we present the simple modification that must be made to the previous results when the arrival processes are compound Poisson processes. In Section F, we propose to use an iterative algorithm to solve the system of equations and show that it converges. The application described above will be treated in Section G.

B Some Relations Valid for Both Disciplines

Results in this section are very general. They hold not only for the two service disciplines we consider, but also for many others, e.g. if one limits in some way the number of messages that can be served in one scan, as long as the system of queues remains stable.

We consider the system as being in the stationary state and the server as undergoing an alternance of switching periods of length c_i ($-\infty < i < \infty$) and service periods of length t_i, the i^{th} service period being spent in queue $i \bmod M$. (See Fig.5.3) From there we define the

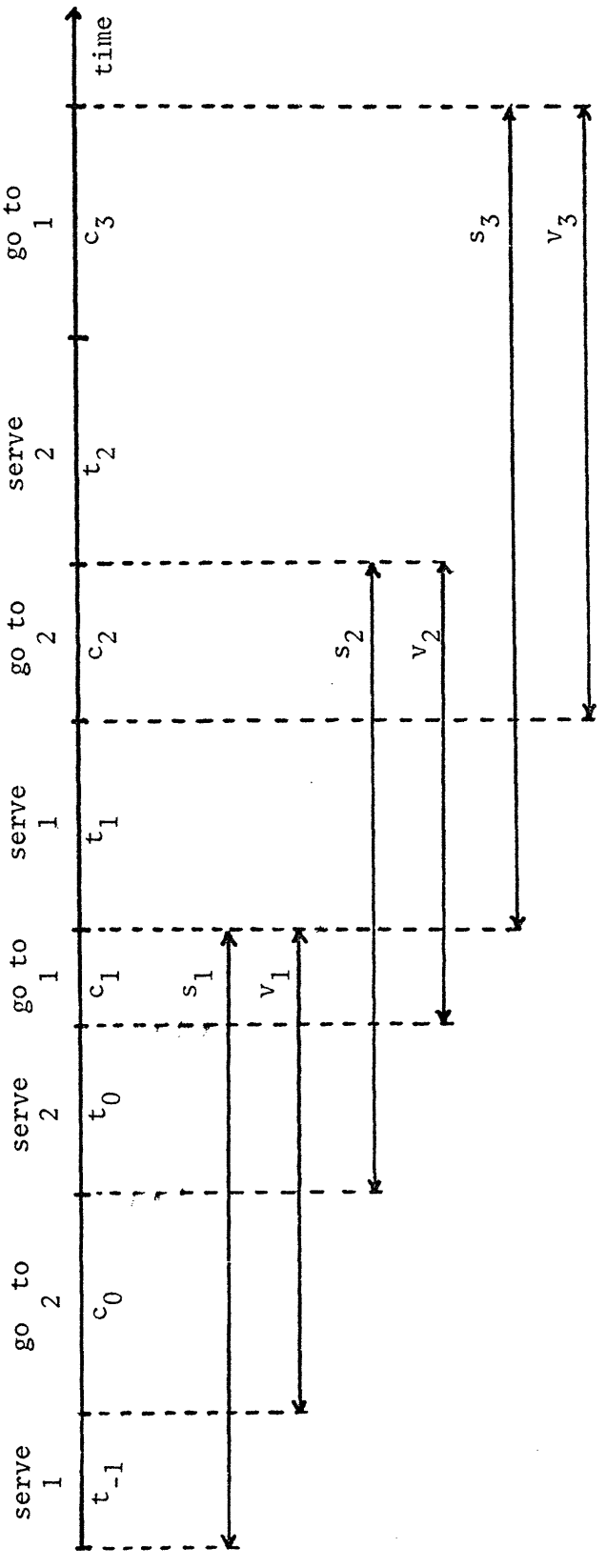


Figure 5.3: Notation (M=2)

i^{th} scanning time by

$$s_i := t_{i-M} + \sum_{k=i-M+1}^{i-1} (c_k + t_k) + c_i$$

$$= \sum_{k=i-M}^{i-1} (t_k + c_{k+1}) \qquad (1)$$

and the i^{th} intervisit time by

$$v_i := \sum_{k=i-M+1}^{i-1} (c_k + t_k) + c_i \qquad (2)$$

$$= s_i - t_{i-M}$$

In the steady state, we have the following relations between the means and variances of the service period lengths:

$$E[t_i] = E[t_{i \bmod M}] \qquad (3)$$

$$\text{var}\ (t_i) = \text{var}\ (t_{i \bmod M})$$

and similarly for the switching, intervisit and scanning times. From (3) the average of (1) is independent of i, and

$$E[s_i] = E[s]$$

We can find the value of $E[s]$ by the following reasoning. Let T be the time for n scanning times relative to queue M to take place. Say $T = s_M + s_{2M} + \ldots + s_{nM}$. Denote by m_j^{in} the number of messages arriving in queue j during T, by m_j^{out} the number of messages leaving queue j and by $\ell_{j1}, \ell_{j2} \ldots \ell_{jm_j^{out}}$ the lengths of these

messages.

We then have

$$\frac{T}{n} = \sum_{j=1}^{M} \left[\frac{1}{n} \sum_{i=1}^{n} c_{j+M(i-1)} + \frac{1}{n} \sum_{i=1}^{m_j^{out}} \ell_{ji} \right]$$

$$= \sum_{j=1}^{M} \left[\frac{1}{n} \sum_{i=1}^{n} c_{j+M(i-1)} + \frac{T}{n} \frac{m_j^{in}}{T} \frac{m_j^{out}}{m_j^{in}} \frac{1}{m_j^{out}} \sum_{i=1}^{m_j^{out}} \ell_{ji} \right]$$

Let us see what happens as n goes to infinity. We show in Section H that if

$$\sum_{i=1}^{M} \rho_i < 1 \quad (\rho_i := \frac{\lambda_i}{\mu_i}) ,$$

the queueing system is stable and the process $\{s_{iM}, i=\ldots,-1,0,1\}$ is ergodic; thus $\frac{T}{n}$ goes to $E[s]$ with probability one as n increases. By the law of large numbers, $\frac{1}{n} \sum_{i=1}^{n} c_{j+M(i-1)}$ goes to ν_j, $\frac{m_j^{in}}{T}$ to λ_j and $\frac{1}{m_j^{out}} \sum_{i-1}^{m_j^{out}} \ell_{ji}$ to $\frac{1}{\mu_j}$, all with probability one. $\frac{m_j^{out}}{m_j^{in}}$ goes to 1 if the system is stable. So we obtain:

$$E[s] = \frac{\sum_{j=1}^{M} \nu_j}{1 - \sum_{j=1}^{M} \rho_j} \tag{4}$$

This expression is meaningful only if $\sum_{i=1}^{M} \rho_i < 1$, as expected.

One finds similarly that

$$E[v_j] = (1 - \rho_{j \bmod M})\, E[s]$$

and

$$E[t_j] = \rho_{j \bmod M}\, E[s]$$

From now on we will assume $\sum_{i=1}^{M} \rho_i < 1$, $\sum_{i=1}^{M} \nu_i > 0$ and we will use the index j where we should use $j \bmod M$.

C Waiting Times for the "Please Wait" Discipline

We proceed in three steps. First we will express the average waiting times as functions of the moments of the scanning times. We find then a relation between the moments of the scanning times and those of the service period lengths. Finally we show that these are related to the solution of a certain system of linear equations.

Suppose we observe a message entering queue i and we note that it arrives u units of time before the end of a scanning time (relative to i) of length z and that it finds n messages in front of it. u , z and n are random variables. For a first in first out service, and conditioned on n , u and z , the Laplace-Stieltjes transform of the distribution of the waiting time of our message is

$$E(e^{-w_i x} \mid n,u,z) = (B_i^*(x))^n\, e^{-ux}$$

where B_i^* is the Laplace-Stieltjes transform of the distribution function of the service time of a message in queue i .

We will now remove the conditioning. Averaging on n, the number of Poisson events in a period of length $z-u$, we obtain

$$E(e^{-w_i x}|u,z) = \sum_{n=0}^{\infty} \frac{(\lambda_i(z-u))^n}{n!} (B_i^*(x))^n e^{-\lambda_i(z-u)} e^{-ux}$$
$$= e^{\lambda_i(z-u)[B_i(x)-1]} e^{-ux}$$

The arrival process being Poisson, u is uniformly distributed between 0 and z so

$$E(e^{-w_i x}|z) = \frac{1}{z} \frac{e^{\lambda_i z[B_i^*(x)-1]} - e^{-zx}}{x + \lambda_i B^*(x) - \lambda_i}$$

If the scanning times relative to the i^{th} queue have a distribution function $Pr[s_i \leq x] = S_i(x)$ with Laplace-Stieltjes transform S_i^*, we show in Appendix A that $Pr[z \leq x] = \int_o^x y/E[s]\, dS_i(y)$ (this would be a well known result of renewal theory if the scanning times relative to the i^{th} queue were independent); from there

$$W_i^*(x) = \frac{1}{E[s]} \frac{S_i^*(\lambda_i(1-B_i^*(x))) - S_i^*(x)}{x + \lambda B_i^*(x) - \lambda}$$

Differentiating one finds the average waiting time in queue i :

$$E[w_i] = \frac{E[s_i^2](1+\rho_i)}{2E[s]} = E[s]\frac{(1+\rho_i)}{2} + \frac{(1+\rho_i)\,var(s_i)}{2E[s]} \tag{5}$$

Let us find now a relation between $var(s_i)$ and $var(t_i)$. If n_i is the (random) number of messages present in queue i when the service

starts, t_i is the sum of n_i independent service times,

so $$E[t_i \mid n_i=n] = \frac{n}{\mu_i}$$

$$E[t_i^2 \mid n_i=n] = n[\Theta_i - \frac{1}{\mu_i^2}] + n^2 \frac{1}{\mu_i^2}$$

n_i in turn is the number of arrivals in queue i during s_i

so $$E[n_i|s_i=s] = \lambda_i s$$

$$E[n_i^2|s_i=s] = \lambda_i^2 s^2 + \lambda_i s$$

and

$$E[t_i|s_i=s] = \rho_i s \tag{6}$$

$$\text{var}(t_i) = \lambda_i \Theta_i E[s] + \rho_i^2 \text{var}(s_i) \tag{7}$$

As announced we now reduce the problem of evaluating (5) to solving a system of linear equations.

From (1) we have

$$\frac{\text{var}(s_i)}{E[s]} = \sum_{k=i-M}^{i-1} \sum_{j=i-M}^{i-1} R_{kj} \tag{8}$$

where $$R_{ij} := \frac{E[(t_i + c_{i+1})(t_j + c_{j+1})] - E[t_i + c_{i+1}]\, E[t_j + c_{j+1}]}{E[s]} \tag{9}$$

clearly $R_{ij} = R_{ji}$

$$R_{ij} = R_{i+kM,\, j+kM} \qquad k = \ldots -1,0,1,\ldots \tag{10}$$

but in general $R_{ij} \neq R_{ij+M}$

The reason for dividing by $E[s]$ in (8) appears before formula (16). From (9) and then (7) and (8) we obtain

$$R_{ii} = \frac{\text{var}\,(t_i)}{E[s]} + \frac{\text{var}\,(c_{i+1})}{E[s]}$$

$$= \lambda_i \theta_i + \rho_i^2 \sum_{j=i-M}^{i-1} \sum_{k=i-M}^{i-1} R_{jk} + \frac{\sigma_{i+1}^2}{E[s]} \qquad (11)$$

If $i > j$ $\quad R_{ij} = E[t_i\,(t_j + c_{j+1})] - E[t_i]\,E[t_j + c_{j+1}]$

$$= E[\,E[t_i \,|\{t_k\}\,,\ \{c_{k+1}\}\,,\ k < i\ ]\,(t_j + c_{j+1})]$$

$$- E[\,E[t_i|\{t_k\}\,,\ \{c_{k+1}\}\,,\ k<i]]\,E[t_j + c_{j+1}]$$

The outside expectations are on the t_k's and c_{k+1}'s , $k < i$.

By (6) and (1)

$$R_{ij} = E[\rho_i \sum_{k=i-M}^{i-1} (t_k + c_{k+1})(t_j + c_{j+1})]$$

$$- \rho_i \sum_{k=i-M}^{i-1} E[t_k + c_{k+1}]\,E[t_j + c_{j+1}]$$

$$= \rho_i \sum_{k=i-M}^{i-1} R_{kj} \qquad i > j \qquad (12)$$

If we define the set I as $\{(i,j) \in Z^2 : 1 \le i \le M\,,\ i-M+1 \le j \le i\}$ we can obtain a system of M^2 linear equations in the M^2 unknowns R_{ij} $(i,j) \in I$ by rewriting (11) and (12) as

$$R_{ii} = \rho_i^2 \left[\sum_{j=1}^{M} R_{jj} + 2 \sum_{j=i-M+1}^{i-1} \sum_{k=i-M}^{j-1} R_{jk}\right] + \lambda_i \theta_i + \frac{\sigma_{i+1}^2}{E[s]} \qquad (13)$$

$$R_{ij} = \rho_i \left(\sum_{k=i-M}^{j} R_{jk} + \sum_{k=j+1}^{i-1} R_{kj} \right) \qquad \begin{matrix} (i,j) \in I \\ i \neq j \end{matrix} \qquad (14)$$

and using relation (10) where necessary. We present in Section 6 a practical way of solving this system.

From (5), (8) and (11) we obtain for the average waiting time in queue

$$E[w_i] = \frac{E[s](1+\rho_i)}{2} + \frac{(1+\rho_i)}{2\rho_i^2} \left[R_{ii} - \left(\lambda_i \Theta_i + \frac{\sigma_{i+1}^2}{E[s]} \right) \right] \qquad (15)$$

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

For example when $M=2$ we have the system

$$R_{11} = \rho_1^2 [R_{11} + R_{22} + 2R_{10}] + \lambda_1 \Theta_1 + \frac{\sigma_2^2}{E[s]}$$

$$R_{22} = \rho_2^2 [R_{11} + R_{22} + 2R_{21}] + \lambda_2 \Theta_2 + \frac{\sigma_1^2}{E[s]}$$

$$R_{10} = \rho_1 [R_{21} + R_{22}]$$

$$R_{21} = \rho_2 [R_{10} + R_{11}]$$

which yields

$$R_{11} = \frac{(\lambda_1 \Theta_1 + \frac{\sigma_2^2}{E[s]})(1-\rho_1\rho_2-\rho_2^2(1+\rho_1\rho_2)) + (\lambda_2 \Theta_2 + \frac{\sigma_1^2}{E[s]})\rho_1^2(1-\rho_1\rho_2+2\rho_1)}{(1-\rho_1-\rho_2)(1+\rho_1+\rho_2+\rho_1\rho_2(1+\rho_1+\rho_2+2\rho_1\rho_2))}$$

and

$$E[w_1] = \frac{E[s](1+\rho_1)}{2} + \frac{(1+\rho_1)\left[(\lambda_1\theta_1+\frac{\sigma_2^2}{E[s]})(1+\rho_1\rho_2+2\rho_1\rho_2^2+2\rho_2^3)+(\lambda_2\theta_2+\frac{\sigma_1^2}{E[s]})(1-\rho_1\rho_2+2\rho)\right]}{2(1-\rho_1-\rho_2)(1+\rho_1+\rho_2+\rho_1\rho_2(1+\rho_1+\rho_2+2\rho_1\rho_2))}$$

In the case of vanishing switching times so that $E[s]$ and $\frac{\sigma_i^2}{\overline{s}}$ become null, the system (13), (14) remains valid and

$$E[w_i] = \frac{1}{2\rho_i^2}[R_{ii} - \lambda_i\theta_i] \qquad i = 1,\ldots,M \qquad (16)$$

In the important case where the queues are identical, or more precisely if $\rho_i = \rho$ and $\lambda_i\theta_i + \frac{\sigma_i^2}{E[s]} = \lambda\theta + \frac{\sigma^2}{E[s]}$, $i = 1,2,\ldots M$ we find that for $(i,j) \in I$

$$R_{ij} = \frac{R_{ii}}{1-(M-1)\rho} \qquad i \neq j$$

and

$$R_{ii} = \frac{(1-(M-1)\rho)\left[\lambda\theta+\frac{\sigma^2}{E[s]}\right]}{(1+\rho)(1-M\rho)}$$

so that

$$= \frac{M}{2(1-M\rho)}(\nu(1+\rho)+\lambda\theta)+\frac{\sigma^2}{2\nu}$$

$$E[w_i] = E[s]\frac{(1+\rho)}{2} + \frac{M}{2(1-M\rho)}\left[\lambda\theta + \frac{\sigma^2}{E[s]}\right] \quad / \quad i=1,2,..,M \qquad (17)$$

The ν_i's need not be equal for relation (17) to hold. We see that the part of the delay due to the switching times is equal to

$$E[s]\frac{(1+\rho)}{2} + \frac{M}{2(1-M\rho)}\frac{\sigma^2}{E[s]} = \frac{M\nu(1+\rho)}{2(1-\rho M)} + \frac{\sigma^2}{2\nu}$$

If the queues are not identical, the overhead is more difficult to assess. However, if the σ_i^2's are all zero, one deduces from formula (15) that the existence of switching times causes an extra delay

$\frac{E[s](1+\rho_i)}{2}$ for messages in the i^{th} queue. Other moments, like the average queue lengths and the means and variances of the number of customers served in one scan can easily be computed from the previous results.

D. Waiting Time for the "Exhaustive Service" Discipline

The method used in this section is very similar to the one used in Part C.

The customers present in queue i when t_i starts have arrived during v_i. [Avi-Itzhak, Maxwell and Miller, 1965] and [Eisenberg, 1972] found an expression for the average waiting time in queue:

$$E[w_i] = \frac{\lambda_i \Theta_i}{2(1-\rho_i)} + \frac{E[v_i^2]}{2E[v_i]} = \frac{\lambda_i \Theta_i}{2(1-\rho_i)} + \frac{E[v_i]}{2} + \frac{\text{var}\ (v_i)}{2E[v_i]} \quad (18)$$

If n customers are present in queue i when service starts we can regard t_i as composed of n independent "M/G/1 busy periods" [Takács, 1962] each with mean $\frac{1}{\mu_i(1-\rho_i)}$ and second moment $\frac{\Theta_i}{(1-\rho_i)^3}$.

Using this observation and a reasoning similar to the one used in Part C, one finds

$$E[t_i | v_i=v] = \frac{\rho_i}{1-\rho_i}\ v \quad (19)$$

$$\text{var}\ (t_i) = \frac{\lambda_i \Theta_i}{(1-\rho_i)^2} E[s] + \left(\frac{\rho_i}{1-\rho_i}\right)^2 \text{var}\ (v_i) \quad (20)$$

Let us now find the system of equations:

from (2)

$$\frac{\text{var}\ (v_i)}{E[s]} = \sum_{j=i-M+1}^{i-1} \sum_{k=i-m+1}^{i-1} K_{jk} + \frac{\sigma_i^2}{E[s]} \tag{21}$$

where $$K_{ij} := \frac{E[(t_i + c_i)(t_j + c_j)] - E[t_i + c_i]\ E[t_j + c_j]}{E[s]}$$

and has the same properties as R_{ij} in (9) (10).

Using (20), (21), (19) and (2)

$$K_{ii} = \frac{\lambda_i \Theta_i}{(1-\rho_i)^2} + \left(\frac{\rho_i}{1-\rho_i}\right)^2 \sum_{j=i-M+1}^{i-1} \sum_{k=i-m+1}^{i-1} K_{jk} + \left(\frac{\rho_i}{1-\rho_i}\right)^2 \frac{\sigma_i^2}{E[s]}$$

$$+ 2\ \frac{\rho_i}{1-\rho_i}\ \frac{\sigma_i^2}{E[s]} + \frac{\sigma_i^2}{E[s]}$$

$$K_{ii} = \frac{1}{(1-\rho_i)^2} \left[\lambda_i \Theta_i + \frac{\sigma_i^2}{E[s]} + \frac{\rho_i^2}{(1-\rho_i)^2} \sum_{j=i-M+1}^{i-1} \sum_{k=i-M+1}^{i-1} K_{jk}\ \right] \tag{22}$$

and by (19) and (2)

$$K_{ij} = \frac{\rho_i}{1-\rho_i} \sum_{k=i-M+1}^{i-1} K_{kj} \qquad i > j \tag{23}$$

Defining the set J by $J = \{(i,j) \in Z^2 : 1 \leq i \leq M\ ,\quad i-M+2 \leq j \leq i\}$

We obtain a system of $M(M-1)$ linear equations in the unknown K_{ij} $(i,j) \in J$ by rewriting (22) and (23) as

$$K_{ii} = \frac{1}{(1-\rho_i)^2}\left[\lambda_i\theta_i + \frac{\sigma_i^2}{E[s]}\right]$$

$$+ \frac{\rho_i^2}{(1-\rho_i)^2}\left[\sum_{\substack{j=1\\ j\neq 1}}^{M} K_{jj} + 2\sum_{j=i-M+2}^{i-1}\sum_{k=i-M+1}^{j-1} K_{jk}\right] \qquad (24)$$

$$K_{ij} = \frac{\rho_i}{1-\rho_i}\left[\sum_{k=i-M+1}^{j} K_{jk} + \sum_{k=j+1}^{i-1} K_{kj}\right] \qquad i > j \qquad (25)$$

From (18), (21) and (22)

$$E[w_i] = \frac{E[s](1-\rho_i)}{2} + \frac{K_{ii}(1-\rho_i)}{2\rho_i^2} - \frac{(1+\rho_i)}{2\rho_i^2}\left[\lambda_i\theta_i + \frac{\sigma_i^2}{E[s]}\right]$$

$$i=1 \ldots M$$

As in part C this solution remains valid when the switching times vanish.

When $\rho_i=\rho$ and $\lambda_i\theta_i + \frac{\sigma_i^2}{E[s]} = \lambda\theta + \frac{\sigma^2}{E[s]}$ we obtain for $(i,j) \in J$

$$K_{ij} = \frac{\rho K_{ii}}{1-(M-1)} \qquad i > j$$

$$K_{ii} = \frac{(1-(M-1)\rho)\left[\lambda\theta + \frac{\sigma^2}{E[s]}\right]}{(1-\rho)(1-M\rho)}$$

so that

$$E[w_i] = E[s]\frac{(1-\rho)}{2} + \frac{M}{2(1-M\rho)}\left[\lambda\theta + \frac{\sigma^2}{E[s]}\right] \qquad (26)$$

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

The difference between the result for the "please wait" discipline (18)

and this one is $\rho E[s]$. This corresponds to the fact that the fraction of messages arriving in a queue that the server is emptying, i.e. ρ , is delayed an extra scanning time in the "please wait" case.

E. Generalization to Compound Poisson Processes

To be complete, we investigate here the simple modifications that must be brought to the previous theory when the arrival processes are modeled as compound Poisson processes. This is sometimes a realistic model when data sources emit messages in clusters separated by long idle periods. In this case the i^{th} queue is characterized by the following statistics: clusters of messages arrive in a Poisson manner, at a rate of λ_i clusters per unit of time. A cluster is composed of a random number of messages. Let the mean number and mean square number of messages in a cluster be ξ_i and ζ_i respectively. The message lengths and switching times have the same means and variances as in previous sections, and we assume all interarrival, service and switching times, and the number of messages in a cluster, to be independent.

If we consider the set of messages present in a cluster as a supermessage, with mean length and mean square length of ξ_i/μ_i and $\xi_i\Theta_i + \frac{1}{\mu_i}\frac{1}{2}(\zeta_i-\xi_i)$ [Karlin and Taylor, 1975, p. 13] respectively, the supermessages will arrive in a Poisson manner so that the analysis of sections 2, 3 and 4 remains valid, as far as the scanning, intervisit and service period lengths, and the waiting times of the supermessages are concerned. All we need to do is replace $\frac{1}{\mu_i}$ by $\frac{\xi_i}{\mu_i}$ and Θ_i by

$\xi_i\Theta_i + \frac{1}{\mu_i^2}(\zeta_i - \xi_i)$ in all formulas.

The average waiting time of a message is equal to the average waiting time of the corresponding supermessage, plus a term taking into account the average time necessary to serve other messages in the same cluster. The average extra delay suffered by the n^{th} message served in a cluster is equal to $(n-1)\frac{1}{\mu_i}$, so the average sum of the extra delays suffered by all messages in a cluster contianing exactly n messages is equal to $\frac{n(n-1)}{2\mu_i}$. Averaging on n and dividing by the average number of messages in a cluster yields an average message extra delay of $\frac{\zeta_i - \xi_i}{2\xi_i}\frac{1}{\mu_i}$.

F. Properties of the Systems of Equations

In the first part of this section, we present alternate forms for the systems of equations (13) (14) and (24) (25). These new systems contain more unknowns but have a simpler structure, which is useful when the time comes to solve them numerically. In the second part we show that all systems considered in this paper can be solved by an efficient iterative algorithm.

Using equation (12) we can rewrite (11) as

$$R_{ii} = \rho_i \sum_{j=i-M}^{i-1} R_{ij} + \lambda_i\Theta_i + \frac{\sigma_{i+1}^2}{E[s]} \qquad (27)$$

Defining the set I' by $I' := \{(i,j) \in Z^2 \mid 1 \leq i \leq M, \; i-M \leq j \leq i\}$ we

can obtain a set of $M(M+1)$ equations in the unknowns R_{ij} $(i,j) \in I'$ by rewriting (12) and (27) as

$$R_{ij} = \rho_i \left(\sum_{k=i-M}^{j} R_{jk} + \sum_{k=j+1}^{i-1} R_{kj} \right) + \delta_{ij} \left(\lambda_i \theta_i + \frac{\sigma_{i+1}^2}{E[s]} \right) \qquad (28)$$

$$\left(\delta_{ij} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases} \right)$$

and using relation (10) when necessary.

Similarly the equations (22) (23) can be rewritten as

$$K_{ij} = \frac{\rho_i}{1-\rho_i} \left[\sum_{k=i-m+1}^{j} K_{jk} + \sum_{k=j+1}^{i-1} K_{kj} \right]$$

$$+ \delta_{ij} \frac{1}{(1-\rho_i)^2} \left[\lambda_i \theta_i + \frac{\sigma_i^2}{E[s]} \right] \qquad (29)$$

or

$$K_{ij} = \rho_i \left[\sum_{k=i-M+1}^{j} K_{jk} + \sum_{k=j+1}^{i} K_{kj} \right] + \delta_{ij} \frac{1}{1-\rho_i} \left[\lambda_i \theta_i + \frac{\sigma_i^2}{E[s]} \right]$$

$$\text{for } (i,j) \text{ such that } 1 \leq i \leq M,\ i-M+1 \leq j \leq i \qquad (30)$$

The system (28) can be rewritten in matrix form as

$$R = AR + B \qquad (31)$$

where R is a column matrix formed by the R_{ij}, $(i,j) \in I'$. A straightforward computation of the solution of (31) can become quite lengthy, A being a $M(M+1)$ by $M(M+1)$ matrix. Instead, the form of equation (31) suggests an iterative procedure, wherein the n^{th} estimate of $R, \hat{R}_n$, is expressed in terms of the $(n-1)^{th}$ estimate by

$$\hat{R}_n = A\hat{R}_{n-1} + B$$

By inspecting equation (28) one checks that each iteration requires only M^3+M^2+M additions and M^2+M multiplications. The variables that need to be stored are the elements of $\hat{R}_n$ and $\hat{R}_{n+1}$, together with the ρ_i's and the $\lambda_i\Theta_i + \frac{\sigma_{i+1}^2}{E[s]}$'s, i.e. a total of $2M(M+2)$ variables. A variant to the algorithm exists (see the specialized texts, e.g. [Varga, 1962]) that reduces this number to $M(M+3)$. In either case this is far from the M^4 that one could expect. It is known that $\hat{R}_n$ converges to the solution R when the norms of all eigenvalues of A are less than 1. Fortunately, this is the case when the system of queues is stable, as we shall see.

If $\rho_i > 0$ $i=1,2 \ldots M$, one can check that the matrix A is an irreducible nonnegative matrix in the sense that all its elements are nonnegative and it cannot be rewritten as

$$A = \begin{vmatrix} A_1 & 0 \\ A_3 & A_2 \end{vmatrix} \qquad \text{(with } A_1 \text{ and } A_2 \text{ square)}$$

by any permutations of rows followed by the same permutations of columns. Among the numerous properties of this type of matrix [Gantmacher, 1960, Ch. 13], we use the following: the eigenvalue of A with the largest norm, α, is real, positive, and bounded as follows:

$$\min_k \frac{(A)_k R}{(R)_k} \leq \alpha \leq \max_k \frac{(A)_k R}{(R)_k} \qquad (32)$$

for all non zero vectors R with elements ≥ 0.

We denote by $(A)_k$ and $(R)_k$ the k^{th} row of A and R. Now, if we use

in (32) a vector R with its elements R_{ij} set equal to $\rho_i\rho_j$, we find that

$$\alpha = \sum_{i=1}^{M} \rho_i < 1 \qquad (33)$$

If some ρ_i's = 0, one verifies easily that relation (33) still holds. A similar algorithm can be used to find the solution of the systems (13) (14), (24) (25), (29) (30). One finds by the same method the following relations about the dominant eigenvalue α.

Systems	Relations
(13) (14)	$1 > \sum_{i=1}^{M} \rho_i \geq \alpha \geq \left(\sum_{i=1}^{M} \rho_i \right)^2$
(24) (25)	$1 > \sum_{i=1}^{M} \rho_i > \max_{\substack{k \\ \rho_k \neq 0}} \frac{\sum_{i=1}^{M} \rho_i - \rho_k}{1 - \rho_k} \geq \alpha \geq \min_k \left(\frac{\sum_{i=1}^{M} \rho_i - \rho_k}{1 - \rho_k} \right)^2$
(29)	$1 > \sum_{i=1}^{M} \rho_i > \max_{\substack{k \\ \rho_k \neq 0}} \frac{\sum_{i=1}^{M} \rho_i - \rho_k}{1 - \rho_k} \geq \alpha \geq \min_k \frac{\sum_{i=1}^{M} \rho_i - \rho_k}{1 - \rho_k}$
(30)	$1 > \sum_{i=1}^{M} \rho_i = \alpha$

G. Application to the Encoding of the Message Origins

In the light of the strategy used in Section 5 it is clear how the cyclic strategies developed here can be used to indicate the message origins. It suffices to queue the messages from origin i in a special buffer that is emptied in a cyclic fashion, and to indicate the end of the service with a flag of length ν_i. If the probability of insertion is known, it is possible to apply the previous results to compute the system performances.

In particular, if the queues are identical and the probability of insertion equal to $2^{-(\nu-1)}$ one obtains from (17) and (26)

$$Ew = 2^{-(\nu-1)} + \frac{M\nu(1 + \lambda(Eb + 2^{-(\nu-1)}))}{2(1 - M\lambda(Eb + 2^{-(\nu-1)}))} + \frac{\lambda_T(Eb^2 + 2Eb\ 2^{-(\nu-1)} + 2^{-(\nu-1)})}{2(1 - M\lambda(Eb + 2^{-(\nu-1)}))} \quad (\ )$$

for the "please wait" discipline, and

$$Ew = 2^{-(\nu-1)} + \frac{M\nu(1 - \lambda(Eb + 2^{-(\nu-1)}))}{2(1 - M\lambda(Eb + 2^{-(\nu-1)})} + \frac{\lambda_T(\Theta + 2Eb\ 2^{-(\nu-1)} + 2^{-(\nu-1)})}{2(1 - M\lambda(Eb + 2^{-(\nu-1)}))}$$

for the "exhaustive service." The first term takes into account the possible insertion in front of a message. Here b refers to the length of a message, exclusive of any insertion.

We note that in light traffic the first two terms will dominate in both cases, whereas the presence of the protocol does not affect the capacity of the link if long enough flags are used.

H. Condition for Stability

We show here that if $\Sigma \rho_i < 1$, the queueing system is stable, and the process $\{s_{jM+i},\ j=\ldots,-1,0,1,\ldots\}$ formed by the lengths of the scanning times relative to queue i is ergodic.

To keep the argument short, we will prove these results only in the case where, with probability one, all service and switching times take only a countable number of values, so that the state spaces of the Markov Processes defined below are countable.

We define $d_k := (t_k\ ,\ c_{k+1}\ ,\ t_{k+i}\ ,\ \cdots\ t_{k+M-1}\ ,\ c_{k+M})^T$. The d_k's form a non stationary Markov Process and by (6)

$$E[d_{k+1}|d_k=d] = \begin{pmatrix} 0 \\ 0 \\ \\ \\ 0 \\ 0 \\ \nu_k \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 & & & & \\ 0 & 0 & 0 & 1 & & & \\ & & & 0 & 1 & & \\ & & & & & & \\ & & & & & & 1 \\ \rho_k & \rho_k & \rho_k & & & & \rho_k \\ 0 & 0 & 0 & & & & 0 \end{pmatrix} d$$

for the "please wait" case. If the "exhaustive service" discipline is used, the expression is similar except that the first ρ_k in the square matrix above is replaced by 0, and the others by $\rho_k/(1-\rho_k)$ (by (19)). In both cases we can write

$$E[d_{k+1}|d_k=d] = B_k + A_k\, d$$

We consider now the process d_{i+kM} , k=...,-1,0,1,... for i fixed. It forms a stationary Markov chain, all values of the form $(0 , c^*_{i+1} , 0 , c^*_{i+2} ,..., 0 , c^*_{i+M})^T$, where the c^*_j 's have non zero probability, are accessible in one step from all states, so the process is either recurrent or transient. One finds that

$$E\, d_{i+(k+1)M} \Big| d_{i+kM} = d \;= C_i + A_{i+M-1}\, A_{i+M-2} \cdots A_i\, d$$

for some C_i . If the eigenvalue of $A_{i+M-1}\, A_{i+M-2} \cdots A_i$ with the largest norm, α , is less than one, for any initial conditions,the mean of d_{i+kM} is uniformly bounded, so the process is positive recurrent. Using the same technique as in Section F, specifically formula (32) with test vector $(\rho_i, 0, \rho_{i+1}, 0, \ldots \rho_{i+M-1}, 0)^T$, one checks that α is < , = , > 1 with $\sum_{j=1}^{M} \rho_j$.

If the d_{i+kM} are ergodic, then, a fortiori, so are the s_{i+kM} 's because they are equal to the sum of the elements of the d_{i+kM} 's.

8. Comparison of the Practical Coding Schemes

In the previous sections we have analysed four different practical coding schemes. Which one is the best? If the input statistics are known, the performances can be computed and the various parameters optimized. Only then can one decide. It is however possible to make some general statements, as we will do here. For the sake of simplicity we assume that all sources have the same statistics, that all flags have length ν and that the probability of insertion is $2^{-(\nu-1)}$.

For convenience we reproduce here the formulas for the average waiting time:

I. First In First Out: (formulas 3 & 4 of Section 4)

$$Ew = \frac{\nu}{2} + 2^{-(\nu-1)} + En + \frac{M\lambda(E(b+n)^2 + 2\,E(b+n)\,2^{-(\nu-1)} + 2^{-(\nu-1)})}{2(1 - M\lambda(E(b+n) + 2^{-(\nu-1)}))}$$

We recall that n is of the order of $\log_2 M$.

II. Sampling: (formula 11 of Section 5)

$$Ew = -\frac{\nu}{2} + 2^{-(\nu-1)} + \frac{M\nu}{1 - M\lambda(Eb + 2^{-(\nu-1)})} + \frac{M\lambda(Eb^2 + 2Eb\,2^{-(\nu-1)} + 2^{-(\nu-1)})}{2(1 - M\lambda(Eb + 2^{-(\nu-1)}))}$$

III. Please Wait: (formula 5 of Section 7)

$$Ew = 2^{-(\nu-1)} + \frac{M\nu(1 + \lambda(Eb + 2^{-(\nu-1)})}{2(1 - M\lambda(Eb + 2^{-(\nu-1)}))}$$

$$+ \frac{M\lambda(Eb^2 + 2Eb\ 2^{-(\nu-1)} + 2^{-(\nu-1)})}{2(1 - M\lambda(Eb + 2^{-(\nu-1)})}$$

IV. Exhaustive Service: (formula 26 of Section 7)

$$Ew = 2^{-(\nu-1)} + \frac{M\nu(1 - \lambda(Eb + 2^{-(\nu-1)})}{2(1 - M\lambda(Eb + 2^{-(\nu-1)}))}$$

$$+ \frac{M\lambda(Eb^2 + 2Eb\ 2^{-(\nu-1)} + 2^{-(\nu-1)})}{2(1 - M\lambda(Eb + 2^{-(\nu-1)}))}$$

One sees immediately that, when the different origins have the same statistics, strategy III is better than II if $M > 1$, but not as good as strategy IV. The relative difference between III and IV is generally small. If $M > 1$, the overhead in strategy II is double the overheads in III and IV. If $M=1$, II is equivalent to III.

In light traffic, strategy I is better than IV, because $\frac{\nu}{2} + \log_2 M \leq \frac{M}{2}$. However, strategy IV performs better in heavy traffic; if ν is large enough, the presence of the protocol does not diminish the traffic that strategy IV can handle. All of this is consistent with what was said in Section 3: in light traffic it is hardly possible to reorder the messages, thus strategy I must be almost optimal. In contrast, strategy IV works well when many messages from each origin are served in every scan, because the flag is used only once for each batch. Note that as indicated in Chapter 3 , strategies II, III and IV would work better if the flag lengths were allowed to vary from message to

message in a batch, according to the probability (as computed by the receiver) that the batch will terminate after the present message.

The observation that Strategy I works well in light traffic and Strategy IV in heavy traffic suggests a hybrid scheme, similar to what [Hayes, 1976] and [Capetanakis, 1977] use in another context. The idea is to group the M origins in M' groups ($M' \leq M$), say origins 1 and 2 in group 1, 3 and 4 in group 2, etc. Strategy IV (or II or III) is used to differentiate between the groups, while prefixes are used to indicate the origins inside of a group. In the example just mentioned, messages from odd origins would be prefixed with a "0", the others with a "1". By varying the size of M' one obtains a continuum of possibilities, ranging from $M' = 1$ (optimal in light traffic) to $M' = M$ (best in heavy traffic). The performances of this scheme can be obtained by modifying in a trivial fashion the results for Strategy IV (or II or III).

Another point that we will investigate is the relation between the average message waiting time and the average number of protocol bits per message, denoted by h, which is equal to $1/M\lambda - Eb$ (formula (1) of Section 2). To be able to compare these results with those of Section 3 we will rather compute the relation between the average number of protocol bits per message and the average number of messages waiting for service, Em, which by Little's formula [Little, 1961] is given by $Em = M \lambda Ew$.

As we have noted earlier, some of the protocol bits convey information about idle times, and some about message origins. In Section 3, all protocol bits transmit information about the origins. The

comparison with Section 3 will still be meaningful in heavy traffic, where the encoding of the origins uses up most of the protocol bits. This is clear in the case of Strategy I.

There,

$$Em = \frac{M\lambda\nu}{2} + M\lambda 2^{-(\nu-1)} + M\lambda En + \frac{M\lambda(E(b+n)^2 + 2E(b+n)\ 2^{-(\nu-1)} + 2^{-(\nu-1)})}{2(h - En - 2^{-(\nu-1)})}$$

or

$$h = En + 2^{-(\nu-1)} + \frac{(E(b+n)^2 + 2E(b+n)\ 2^{-(\nu-1)} + 2^{-(\nu-1)})}{\frac{Em}{M\lambda} - \frac{\nu}{2} - 2^{-(\nu-1)} - En}$$

The first term represents information about the origins. As Em increases, so does the optimal ν and h tends to En, as should be.

In the case of Strategy II, the third term in the formula for Ew will dominate in heavy traffic. We will thus have

$$Em \tilde{=} \frac{M\nu}{h - 2^{-(\nu-1)}}$$

$$h \tilde{=} 2^{-(\nu-1)} + \frac{M\nu}{Em}$$

Optimizing on ν and neglecting the integer constraint, one finds that the optimal ν is given by $\nu = \log_2 (2 \log_e 2\ Em/M)$. This value of ν justifies the approximation of Ew by the third term in the formula above. Using this value in the formula for h, one obtains

$$h = \frac{M}{Em} \log_2 (2e(\log_e 2) \frac{Em}{M})$$

which has exactly the same form as what was found for Strategy II of Section 3.D, except that a factor $\frac{1}{2}$ is missing here. This is easy to explain qualitatively: the only difference between the situations in Section 3.0 and in this section is that the number of messages

served in one scan is variable here, which causes a loss of efficiency because $\frac{\log x}{x}$ is a convex function.

The cases of Strategies III and IV are similar, we treat IV only. The second term in the expression for Ew will eventually dominate. Neglecting the term $2^{-(\nu-1)}$ in the numerator, we obtain

$$Em \tilde{=} \frac{M\nu(1 - \lambda Eb)}{2(h - 2^{-(\nu-1)}}$$

or

$$h \tilde{=} 2^{-(\nu-1)} + \frac{M\nu}{2}(1 - \lambda Eb)$$

The optimal ν is given by

$$\nu = \log_2 \left(\frac{4(\log_e 2)}{1 - \lambda Eb} \frac{Em}{M} \right)$$

and the resulting h is equal to

$$h = \frac{(1 - \lambda Eb)}{2} \frac{M}{Em} \log_2 \frac{4e(\log_e 2)\, Em}{(1 - \lambda Eb)\, M}$$

This is about twice as efficient as Strategy II, but less efficient by a factor of two than the comparable strategy of Section 3.D.

We can thus conclude that although in heavy traffic strategy IV is the most efficient of the strategies we analyzed, it is probably far from being optimal, as indicated by the results of section 3. Nevertheless enormous gains can be realized by using it in heavy traffic, as illustrated in the following numerical example.

Fixed length messages arrive at a

concentrator in a Poisson manner, at a rate λ on each of M input lines. We want to transmit on a noiseless binary, synchronous output link not only the messages, but also their origins.

Usually this is done by prefixing messages with an address. In some cases this scheme significantly increases the average delay incurred by the messages, as a numerical example will show.

Let us use as time unit the interval necessary to transmit one bit on the output link and let us take $M=16$, the length $\frac{1}{\mu} = 50$ and $\lambda = \frac{1}{1000}$. If we naively forget about the addresses, we obtain from the formula of the mean waiting time in a $M/D/1$ queue:

$$E[w] = \frac{1}{2} \frac{M\lambda \left(\frac{1}{\mu}\right)^2}{(1-M\rho)} = 100$$

If we use a 4 bit address and prefix all messages with a "1" to distinguish them from idle periods during which we transmit "0" 's, the length becomes 55 (a 10% overhead) but the delay becomes

$$E[w] = \frac{1}{2} \frac{16 \frac{1}{1000} (55)^2}{1 - \frac{16\ 55}{1000}} + .5 \cong 202$$

(the term .5 takes into account the synchronous nature of the output link). The presence of the addresses doubles the mean waiting time in queue.

Another simple way of transmitting the origin of the messages is to use the cyclic, exhaustive service discipline. We queue messages in a buffer corresponding to their origin, prefix them with a "1" so that

their length is now 51 bits, process every queue in turn and when it is empty transmit a "0" . Our "switching time" has thus mean $\nu = 1$ and variance $\sigma^2 = 0$. From formula (26) of Section 7.

$$E[w] = \frac{1}{2}\frac{16\ 1\ (1 - 51/1000)}{1 - \frac{16\ 51}{1000}} + \frac{1}{2}\frac{16\ 1/1000\ (51)^2}{1 - \frac{16\ 51}{1000}} \cong 154$$

The improvement is due to the fact that this way of transmitting the address is naturally adaptive. When many messages are waiting in queue, few bits per message are needed to indicate the origin. Of course, this strategy works well only when the traffic is heavy, but this is precisely the time when it is worth reducing queueing delays. As the traffic growth heavier, this scheme works better and better.

9. Suggestions for Future Work

We have shown in Section 8 that the "sampling" and "polling" strategies behave in the same way in the fixed length queue and variable length queue cases. Unfortunately we know from Section 3 that they are rather inefficient. One would expect that the efficient strategies for the fixed queue length case will also perform well in a variable length queue environment. Their analysis is not easy, because they introduce much memory in the queueing system, but should be attempted.

On a more abstract level, the state of a queue can be regarded as forming a partially observable Markov process when the input process is Poisson. One should be able to use the same method as in Section 3 and determine a strategy that minimizes the entropy of the output sequence,

REFERENCES

ABRAMSON, N. and F.F. KUO, eds., 1973,
Computer Communication Networks. Englewood Cliffs. N.J.: Prentice-Hall, 1973.

AVI-ITZHAK, B., MAXWELL, W.L. and L.W. MILLER, 1965,
"Queueing with Alternating Priorities," Operations Research, vol. 13, pp. 306-318, 1965.

BELLMAN, R., 1957,
Dynamic Programming. Princeton, N.J.: Princeton University Press, 1957.

CAMRASS, R.J. and R.G. GALLAGER, 1976,
"Encoding Message Length for Data Transmission," Report ESL-P-687, 1976.

CAPETANAKIS, J.I., 1977,
"The Multiple Access Broadcast Channel: Protocol & Capacity Considerations," Ph.D. Dissertation, Dept. of Elec.Eng. and Comp.Science, M.I.T., Cambridge, MA, 1977.

CARSTEN, R.T., NEWHALL, E.E. and M.J.M. POSNER, 1977,
"A Simplified Analysis of Scan Times in an Asymmetrical Newhall Loop with Exhaustive Service," IEEE Trans. Communications, vol. COM-25, pp. 951-958, September 1977.

CERF, V.G., 1974,
"An Assessment of ARPANET Protocols," Network Information Center, #304889, Stanford Research Institute, Menlo Park, CA, April 1974.

COOPER, R.B., 1970,
"Queues Served in Cyclic Order: Waiting Times," Bell System Technical Journal, vol. 49, pp. 399-413, 1970.

———— and G. MURRAY, 1969,
"Queues Served in Cyclic Order," Bell System Technical Journal, vol. 48, pp. 675-690, 1969.

CROCKER, S.D., HEAFNER, J., METCALFE, R. and J. POSTEL, 1972,
"Function-Oriented Protocols for the ARPA Computer Network," AFIPS 1972 SJCC Proceedings, vol. 40, Atlantic City, N.J.: AFIPS Press, pp. 271-279, 1972.

DAVIES, D.W. and D.L. BARBER, 1973,
Communication Networks for Computers. New York: Wiley, 1973.

DONNAN, R.A. and J.R. KERSEY, 1974,
"Synchronous Data Link Control: A Perspective," IBM Systems Journal, May 1974.

DOOB, J.L., 1953,
Stochastic Processes. New York: Wiley, 1953.

DRAKE, A.W., 1962,
"Observation of a Markov Process through a Noisy Channel," Sc.D. Dissertation, Dept. of Elec.Eng., M.I.T., Cambridge, MA, 1962.

EISENBERG, M., 1972,
"Queues with Periodic Service and Changeover Time," Operations Research, vol. 20, pp. 440-451, 1972.

FELLER, W., 1966,
An Introduction to Probability Theory and Its Application, vol. II, New York: Wiley, 1966.

————, 1968,
Idem, vol. I, 3rd edition. New York: Wiley, 1968.

FUCHS, E. and P.E. JACKSON, 1969,
"Estimates of Distributions of Random Variables for Certain Computer Communications Traffic Models," Proc. ACM Symp. Problems on the Optimization of Data Communication Systems, Pine Mountain, GA, pp. 202-225, Oct. 1969.

GALLAGER, R.G., 1968,
Information Theory and Reliable Communication. New York: Wiley, 1968.

————————, 1976,
"Basic Limits on Protocol Information in Data Communication Networks," IEEE Trans.Inform. Theory, vol. IT-22, pp. 385-398, July 1976.

————————, 1977,
"A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE Trans. Communications, vol. COM-25, pp. 73-85, Jan. 1977.

————————, 1978,
"Variations on a Theme by Huffman," submitted to IEEE Trans.Inform. Theory.

———————— and D.C. VAN VOORHIS, 1975,
"Optimal Source Codes for Geometrically Distributed Integer Alphabets," IEEE Trans.Inform. Theory, vol. IT-21, pp. 228-230, March 1975.

GANTMACHER, F.R., 1960,
The Theory of Matrices, vol. II. New York: Chelsea, 1960.

GERLA, M. and L. KLEINROCK, 1977,
"On the Topological Design of Distributed Computer Networks," IEEE Trans. Communications, COM-25, pp. 48-60, Jan. 1977.

GOLOMB, S.W., 1966,
"Run Length Encodings," IEEE Trans.Inform. Theory, vol. IT-12, pp. 399-401, July 1966.

HAYES, J.F., 1976,
"Adaptive Polling," Technical Memorandum 76-3116-I, Bell Laboratories, Holmdel, N.J., 1976.

HOWARD, R.A., 1960,

Dynamic Programming and Markov Processes. Cambridge, MA: M.I.T. Press, 1960.

————————, 1971,
Dynamic Probabilistic Systems, vols. I and II. New York: Wiley, 1971.

HUFFMAN, D.A., 1952,
"A Method for the Construction of Minimum Redundancy Codes, " Proc. IRE, vol. 51, pp. 251-252, Sept. 1952.

JELINEK, F., 1968,
"Buffer Overflow in Variable Length Coding of Fixed Rate Sources," IEEE Trans.Inform. Theory, IT-14, pp. 400-501, 1968.

———————— and K. SCHNEIDER, 1972,
"On Variable Length to Block Coding," IEEE Trans.Inform. Theory, vol. IT-18, pp. 765-774, Nov. 1972.

KARLIN, S. and H.M. TAYLOR, 1975,
A First Course in Stochastic Processes. New York: Academic Press, 1975.

KINGMAN, J.F.C., 1970,
"Inequalities in the Theory of Queues," J.R.Statis.Soc., vol. B32, pp. 102-110, 1970.

KLEINROCK, L., 1965,
Queueing Systems, vol. I: Theory. New York: Wiley, 1965.

————————, 1976,
Idem, vol. II: Computer Applications. New York: Wiley, 1976.

————————, NAYLOR, W.E. and H. OPDERBECK, 1976,
"A Study of Line Overhead in the ARPANET," Commun.Ass. Computing Machinery, vol. 19, pp. 3-13, Jan. 1976.

———————— and H. OPDERBECK, 1977,
"Throughout in the ARPANET - Protocols and Measurement," IEEE Trans. Communications, COM-25, pp. 95-103, Jan. 1977.

KLERER, M. and G.A. KORN, eds., 1967,
Digital Computer User's Handbook. New York: McGraw-Hill, 1967.

KONHEIM, A.G. and B. MEISTER, 1974,
"Waiting Lines and Times in a System with Polling," J.A.C.M., vol. 21, pp. 470-490, 1974.

KRUSKAL, J.B., 1969,
"Work-Scheduling Algorithms: A Nonprobabilistic Queueing Study (with Possible Application to No 1 ESS)," Bell System Technical Journal, vol. 48, pp. 2963-2974, 1969.

KUSHNER, H.J., 1971,
Introduction to Stochastic Control. New York: Holt, Reinehart and Winston, 1971.

LEIBOWITZ, M.A., 1961,
"An Approximate Method for Treating a Class of Multi-queue Problems," I.B.M. Journal of Research and Development, vol. 5, pp. 204-209, 1961.

LEWIS, P.A.W. and P.C. YUE, 1972,
"Statistical Analysis of Series of Events in Computer Systems," Statistical Computer Performance Evaluation, W. Freiberger, ed. New York: Academic Press, pp. 265-280, 1972.

LITTLE, J.D.C., 1961,
"A Proof of the Queueing Formula: $L = \lambda W$," Operations Research, vol. 9, pp. 383-389, 1961.

NIELSEN, P.T., 1973,
"A Note on Bifix-free Sequences," IEEE Trans.Inform. Theory, IT-19, pp. 704-706, Sept. 1973.

ODONI, A.R., 1969,
"On Finding the Maximal Gain for Markov Decision Processes," Operations Research, vol. 17, pp. 857-860, 1969.

PLATZMAN, L.K., 1977
"Finite-memory Estimation and Control of Finite Probabilistic Systems," Ph.D. Dissertation, Dept. of Elec.Eng. and Comp. Science, M.I.T., Cambridge, MA, 1977.

POUZIN, L., 1973,
"Presentation and Major Design Aspects of the CYCLADES Computer Networks," Proc. 3rd Data Comm. Symp., 1973.

RUBIN, I., 1976,
"Data Compression for Communication Networks: The Delay-Distortion Function," IEEE Trans.Inform. Theory, IT-22, pp. 655-664, Nov. 1976.

RUDIN, W., 1966,
Real and Complex Analysis. New York: McGraw-Hill, 1966.

SCHALKWIJK, J.P.M. and K.A. POST, 1973,
"On the Error Probability for a Class of Binary Recursive Feedback Strategies," IEEE Trans.Inform. Theory, vol. IT-19, pp. 498-512, July 1973.

SCHWARTZ, M., 1977,
Computer Communication Network Design and Analysis. Englewood Cliffs, N.J.: Prentice Hall, 1977.

SCHWEITZER, P.J., 1971,
"Iterative Solution of the Functional Equations of Undiscounted Markov Renewal Programming," J.Math.Anal.Appl., vol. 34, pp. 495-501, 1971.

SEGALL, A., 1977,
"The Modeling of Adaptive Routing in Data-Communication Networks," IEEE Trans. Communications, vol. COM-25, pp. 85-94, Jan. 1977.

TAKACS, L., 1962,
Introduction to the Theory of Queues. New York: Oxford University Press, 1962.

TYMES, L., 1971,
"Tymnet - A Terminal Oriented Communications Network," AFIPS Conf. Proc., vol. 38, 1971.

VANDERMEY, J.E., 1976,
"The Architecture of a Transparent Intelligent Network," Proc. National Telecommunication Conference, 1976.

VARGA, R.S., 1962,
Matrix Interative Analysis. Englewood Cliffs, N.J.: Prentice Hall, 1962.

WYNER, A.D., 1974,
"On the Probability of Buffer Overflow under an Arbitrary Bounded Input-Output Distribution," SIAM J.Appl.Math., vol. 27, pp. 544-570, Dec. 1974.

Appendix A

A Theorem about Random Sums

We prove here a theorem that is used in Section 4 of Chapter 3.

Let (Ω , S , P) be a probability space. We recall that if $x : \Omega \to \mathbb{R}$ is a measurable function, $E|x|<\infty$,and if $\mathcal{B}$ is a σ-algebra included in S , $E(x|\mathcal{B})$ is defined as a $\mathcal{B}$-measurable function such that $\int_B E(x|\mathcal{B})\, dP = \int_B x dP$ for every B in $\mathcal{B}$. One can show [Doob, 1953, pp. 16 and 32] that $E(x|\mathcal{B})$ exists, that any two versions of it are equal almost everywhere, and that if z is a $\mathcal{B}$-measurable function with $E|xz|<\infty$, $E(xz|\mathcal{B}) = z\, E(x|\mathcal{B})$ almost everywhere. These facts are used below.

Let m be a measurable function $m : \Omega \to \mathbb{N}^+$

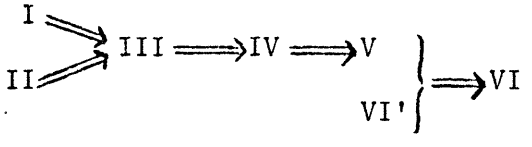
$b_1, b_2, \ldots$ be a sequence of measurable functions $E(|b_i|)<\infty$ $i \in \mathbb{N}^{++}$ $b_i : \Omega \to \mathbb{R}$

$\mathcal{B}_i$ be the smallest σ-algebra for which b_i is measurable

$\mathcal{B}^i$ be the smallest σ-algebra for which b_{i+1}, $b_{i+2}, \ldots$ are measurable

$\mathcal{B}^{(i)}$ be the smallest σ-algebra for which $b_1, b_2, \ldots, b_i$ are measurable

then



where

I $E\, b_i = a$ if $P(m \geq i) > 0 \qquad i \in \mathbb{N}^{++}$

m is independent of the b_i's

II the b_i's are mutually independent

$E\, b_i = a$ if $P(m \geq i) > 0 \qquad i \in \mathbb{N}^{++}$

$\{\omega : m(\omega) = i\} \in \mathcal{B}^{(i)}$

i.e. m is a Markov time

III $E\, b_i = a$ if $P(m \geq i) > 0 \qquad i \in \mathbb{N}^{++}$

$E(I_{m=i} | \mathcal{B}^i) = E(I_{m=i})$ a.e.

i.e. the event $m=i$ is independent of $b_{i+1}, b_{i+2}, \ldots$

IV $E\, b_i = a$ if $P(m \geq i) > 0 \qquad i \in \mathbb{N}^{++}$

$E(I_{m<i} | \mathcal{B}_i) = E(I_{m<i})$ a.e.

i.e. the event $m<i$ is independent of b_i

V $E(b_i\, I_{m \geq i}) = a\, E(I_{m \geq i})$

VI $E\left(\sum_{i=1}^{m} b_i\right) = a\, E(m)$

VI' $E\left(\sum_{i=1}^{m} |b_i|\right) < \infty$

VI' is a technical condition to insure that $E\left(\sum_{i=1}^{m} b_i\right)$ is well defined.

Proof

I $\Longrightarrow$ III this should be clear;

II $\Longrightarrow$ III it is enough to show that

$$\forall B \in \mathcal{B}^i \,,\ \int_B E(I_{m=i})\, dP = \int_B I_{m=i}\, dP$$

or $P(\{\omega : m(\omega) = i\})P(B) = P(\{\omega : m(\omega)=i\} \cap B)$

This follows from the fact that $B \in \mathcal{B}^i$, and that by II

$$\{\omega : m(\omega) = i\} \in \mathcal{B}^{(i)}$$

and the b_i's are mutually independent.

III $\Longrightarrow$ IV

$$E(I_{m<i} | B_i)$$

$$= \sum_{j=1}^{i-1} E(I_{m=j} | B_i)$$

$$= \sum_{j=1}^{i-1} E(I_{m=j}) \quad \text{by III because } B_i \subset \mathcal{B}^j \text{ for } j<i$$

$$= E(I_{m<i})$$

IV $\Longrightarrow$ V

$$E(b_i I_{m\geq i}) = E(E(b_i I_{m\geq i} | B_i))$$

$$= E(b_i E(i_{m\geq i} | B_i))$$

$$= E\, b_i E(I_{m\geq i}) \quad \text{by IV}$$

$$= a\, E(I_{m\geq i}) \quad \text{by IV}$$

V, VI' $\Longrightarrow$ VI

$$E\left(\sum_{i=1}^{m} b_i\right) = E\left(\sum_{i=1}^{\infty} b_i I_{m\geq i}\right)$$

$$= \sum_{i=1}^{\infty} E(b_i I_{m\geq i}) \quad \text{by VI'}$$

$$= \sum_{i=1}^{\infty} a\, E(I_{m\geq i}) \quad \text{by V}$$

$$= a\, E(m)$$

Note that we do not need to assume $E(|b_i|)<\infty$ and VI' if $P(b_i \geq 0) = 1$ $i \in \mathbb{N}^{++}$, and if we allow

the value ∞ .

If the b_i 's are independent and identically distributed, and some technical conditions are met, it is well known that I $\Rightarrow$ VI, while III $\Rightarrow$ VI (also known as Wald's theorem) is proved at different places in [Feller, 1966]. [Doob, 1953] proves that II $\Rightarrow$ VI .

The theorem given here is very simple, and its hypothesis minimal; that IV $\Rightarrow$ VI is somewaht surprising, we give an example illustrating it.

prob.	m	b_1	b_2	b_3
3/16	1	0	-1	2
1/16	1	16	7	2
1/16	2	0	-1	0
3/16	2	0	7	0
4/16	3	0	-1	0
4/16	3	0	-1	2

We have $E\, b_1 = E\, b_2 = E\, b_3 = 1$

$E\, m = 9/4$

$P(m < 2 \mid b_2 = -1) = \frac{1}{4} = P(m < 2)$

$P(m < 3 \mid b_3 = 2\ ) = \frac{1}{2} = P(m < 3)$

Thus, surely enough,

$$E \sum_{i=1}^{m} b_i = \frac{3}{16}\,0 + \frac{1}{16}\,16 + \frac{1}{16}\,(-1) + \frac{3}{16}\,7 + \frac{4}{16}\,(-1) + \frac{4}{16}\,1$$

$$= 9/4 = Em\ Eb_1$$

although

$$P(m=2 \mid b_3=2) = 0 \neq P(m=2) = \tfrac{1}{4} ,$$

thus hypothesis III is not met.

Appendix B

We prove here a theorem used in Sections 5 and 7 of Chapter 5. The method is similar for both cases, we will give the details for Section 5 and sketch the proof for Section 7.

We know that if $\rho_T < 1$ and if $\sum_{j=1}^{M} E\, v_{0j} < \infty$, the process $\{s_i, m_j^i\}$, $i=0,1,\ldots$ (j fixed) is Markovian and positive recurrent, thus ergodic; $E\, s_i$ and $E\, m_j^i$ are finite. For x given, consider the random variables

$$z_i(s_i, m_j^i) := m_j^i \, I_{s_i \leq x}$$

The z_i process is also ergodic, because if a set A of sequences $\{z_i\}$ is shift invariant, so is the set $A' := \{\{s_i, m_j^i\} : \{z_i(s_i, m_j^i)\} \in A\}$ A' has the same probability as A, i.e. 0 or 1.

Theorem

The limit, as the time increases, of the fraction $f(t)$ of messages from origin j that arrived in the queue during scanning times of length less than or equal to x is almost surely equal to

$$\frac{1}{E(s)} \int_0^x y \, dS(y) .$$

Proof: Denoting by $\alpha(t)$ the number of complete scanning times up to time t, we have that

$$\frac{\sum_{i=0}^{\alpha(t)-1} m_j^i I_{s_i \leq x}}{\sum_{i=0}^{\alpha(t)} m_j^i} < f(t) < \frac{\sum_{i=0}^{\alpha(t)} M_j^i I_{s_i \leq x}}{\sum_{i=0}^{\alpha(t)-1} m_j^i}$$

By the strong ergodic theorem, the ratio of the numerators over $\alpha(t)$ goes with probability one to $E\, m_j^i I_{s_i \leq x} = \lambda_j \int_o^x y\, dS(y)$ while the ratio of the denominator to $\alpha(t)$ goes with probability one to $\lambda_j E s$.

Q.E.D.

Note that this would be a well known result of renewal theory if the scanning times were independent, and if the arrivals did not interact with the lengths of the scanning times.

The proof for Section 7 goes along the same lines, the main difference is that the process $\{m_j^i, S_i\}$ must be replaced by a process of larger size, similarly as we did for the d_i process in Section , to retain the Markov property.

Appendix C

This appendix contains the listing of FORTRAN IV subroutines MOHUFF and LSEQ1 which implement respectively Steps I and II of the algorithm presented in Section 4.C of Chapter II.

MOHUFF is a straight translation in FORTRAN of the algorithm given in Step I. It works best when the symbols are listed in order of decreasing probabilities.

LSEQ1 computes the largest root of the equation $A^*(s)\ B^*(-s) = 1$, using the Newton-Raphson algorithm [Klerer and Korn, 1967, p. 2-59]. Because this algorithm works best with functions whose ratio of the second derivative to the first derivative has small absolute value, the subroutine computes the largest root of the equation $\log A^*(s) + \log B^*(-s) = 0$.

In lines 14 to 18 the program searches for a starting point larger than the largest root. Because the Laplace-Stieltjes transforms of probability distributions are log-convex, the sequence of values produced by the algorithm from this starting point will converge monotonely to the largest root. The algorithm itself occupies lines 19 to 28.

Function evaluations take place in lines 33 to 66. Subroutine INTTIM, which must be provided by the user, computes $\log A^*(s)$ and $\frac{d}{ds} \log A^*(s)$. If IND = 1, $B^*(s)$ is set equal to the lowerbound developed in Section 4.B of Chapter II, and the program computes s_u. If IND = 2, $B^*(s) = \sum_{i=1}^{c} p_i e^{smi}$, and the program computes the corresponding s^o. When m_i is constant, the same objective is attained more efficiently by setting IND to 3.

```
      SUBROUTINE MOHUFF(S,P,N,L,CW,AL,IND,PR)                          MOHU0001
C  COMPUTE MODIFIED HUFFMAN CODE                                       MOHU0002
C  S IS INPUT EXPONENT                                                 MOHU0003
C  P IS INPUT PROBABILITIES                                            MOHU0004
C  N IS INPUT SIZE                                                     MOHU0005
C  I IS OUTPUT LENGTH                                                  MOHU0006
C  CW CONTAINS THE FIRST 32 BITS OF OPTIMUM CODEWORD                   MOHU0007
C  AL IS OUTPUT AVERAGE CODEWORD LENGTH                                MOHU0008
C  IND AND PR ARE WORK AREAS                                           MOHU0009
C  IF S GT 10 THE OPTIMAL CODE FOR S = INFINITY IS RETURNED            MOHU0010
      REAL*4 P(N),PR(N),MIN1PR,MIN2PR,AL,SEXP,S,EXP                   MOHU0011
      INTEGER*4 CW(N),A,I                                             MOHU0012
      INTEGER*2 L(N),IND(N),D(32),IND1,IND2                           MOHU0013
      IF(S.GT.10.)GOTO11                                              MOHU0014
      SEXP=EXP(S)                                                     MOHU0015
C     INITIALIZE                                                      MOHU0016
      DO10 I=1,N                                                      MOHU0017
      PR(I)=P(I)                                                      MOHU0018
      IND(I)=I                                                        MOHU0019
      CW(I)=0                                                         MOHU0020
10    L(I)=0                                                          MOHU0021
13    A=N                                                             MOHU0022
C     FIND 2 SMALLEST                                                 MOHU0023
23    IF(PR(A).LE.1.E75)GOTO24                                        MOHU0024
      A=A-1                                                           MOHU0025
      GOTO23                                                          MOHU0026
24    I=A                                                             MOHU0027
      IND1=0                                                          MOHU0028
      MIN1PR=1.E75                                                    MOHU0029
      MIN2PR=1.E75                                                    MOHU0030
22    IF(PR(I).GE.MIN2PR)GOTO20                                       MOHU0031
      IF(PR(I).GE.MIN1PR)GOTO21                                       MOHU0032
      IND2=IND1                                                       MOHU0033
      MIN2PR=MIN1PR                                                   MOHU0034
      IND1=I                                                          MOHU0035
      MIN1PR=PR(I)                                                    MOHU0036
```

```
      GOTO20                                                            MOHU0037
21    IND2=I                                                            MOHU0038
      MIN2PR=PR(I)                                                      MOHU0039
20    I=I-1                                                             MOHU0040
      IF(I.GT.0)GOTO22                                                  MOHU0041
C     EXIT                                                              MOHU0042
      IF(IND2.EQ.0)GOTO40                                               MOHU0043
C     UPDATE                                                            MOHU0044
      PR(IND2)=SEXP*(PR(IND1)+PR(IND2))                                 MOHU0045
      PR(IND1)=2.E75                                                    MOHU0046
      DO30 I=1,N                                                        MOHU0047
      IF(IND(I).EQ.IND1)GOTO31                                          MOHU0048
      IF(IND(I).NE.IND2)GOTO30                                          MOHU0049
      CW(I)=CW(I)+CW(I)                                                 MOHU0050
      GOTO32                                                            MOHU0051
31    CW(I)=CW(I)+CW(I)+1                                               MOHU0052
      IND(I)=IND2                                                       MOHU0053
32    L(I)=L(I)+1                                                       MOHU0054
30    CONTINUE                                                          MOHU0055
      GOTO23                                                            MOHU0056
11    DO12 I=1,N                                                        MOHU0057
      PR(I)=1.                                                          MOHU0058
      IND(I)=I                                                          MOHU0059
      CW(I)=0                                                           MOHU0060
12    L(I)=0                                                            MOHU0061
      SEXP=1.                                                           MOHU0062
      GOTO13                                                            MOHU0063
C     COMPUTE AVERAGE LENGTH                                            MOHU0064
40    AL=0.                                                             MOHU0065
      DO 50 I=1,N                                                       MOHU0066
50    AL=AL+P(I)*L(I)                                                   MOHU0067
      RETURN                                                            MOHU0068
      ENTRY PRINTH                                                      MOHU0069
C  PRINTH PRINTS THE 32 LEFT-MOST BITS OF THE OPTIMAL CODEWORDS         MOHU0070
      WRITE(6,99)                                                       MOHU0071
      DO60 I=1,N                                                        MOHU0072
```

```
      IND1=L(I)                                                          MOHU0073
      IF(IND1.GT.32)IND1=32                                              MOHU0074
      DO70 IND2=1,IND1                                                   MOHU0075
      A=CW(I)/2                                                          MOHU0076
      D(IND2)=CW(I)-A-A                                                  MOHU0077
      IF(D(IND2).GE.0)GOTO70                                             MOHU0078
      A=A-1                                                              MOHU0079
      D(IND2)=-D(IND2)                                                   MOHU0080
70    CW(I)=A                                                            MOHU0081
60    WRITE(6,100)I,P(I),L(I),(D(IND2),IND2=1,IND1)                      MOHU0082
      RETURN                                                             MOHU0083
99    FORMAT(' ',8X,'PROBABILITIES',5X,'LENGTHS',2X,'CODEWORDS')         MOHU0084
100   FORMAT(' ',I5,3X,E13.6,6X,I4,4X,32I1)                              MOHU0085
      END                                                                MOHU0086
```

```
      SUBROUTINE LSEQ1(IND,S,P,N,L,IER)                                  LSEQ0001
C  COMPUTE LARGEST ROOT OF ALOG(A(S)*B(-S)) = 0                          LSEQ0002
C  IF THIS ROOT IS GT 10. 10.001 IS RETURNED                            LSEQ0003
C  USE NEWTON-RAPHSON                                                   LSEQ0004
C  IF NO CONVERGENCE STOP AFTER 99 ITERATIONS AND SET IER TO 0          LSEQ0005
      REAL*4 P(N),S,F,FD,A,AD,AUX,  DEL1,DEL2,ABS,      ALOG,EXP       LSEQ0006
      INTEGER*4 IER,IND,K,M                                            LSEQ0007
      INTEGER*2 L(N),I                                                 LSEQ0008
      IER=1                                                            LSEQ0009
      DEL2=1.E75                                                       LSEQ0010
      I=0                                                              LSEQ0011
      IF(S.LT.0.)S=1.                                                  LSEQ0012
      M=1                                                              LSEQ0013
      GOTO(601,602,604),IND                                            LSEQ0014
501   IF(F.GE.0..AND.FD.GT.0.)GOTO502                                  LSEQ0015
      S=S+.5                                                           LSEQ0016
      IF(S.GT.10.)GOTO4                                                LSEQ0017
      GOTO(601,602,604),IND                                            LSEQ0018
1     I=I+1                                                            LSEQ0019
      IF(I.GE.100)GOTO6                                                LSEQ0020
      M=2                                                              LSEQ0021
      GOTO(601,602,604),IND                                            LSEQ0022
502   SO=S                                                             LSEQ0023
      S=SO-F/FD                                                        LSEQ0024
      DEL1=ABS(S-SO)                                                   LSEQ0025
      IF(DEL1.LE.1.E-4.AND.DEL1.GE.DEL2)RETURN                         LSEQ0026
      DEL2=DEL1                                                        LSEQ0027
      GOTO1                                                            LSEQ0028
4     S=10.001                                                         LSEQ0029
      RETURN                                                           LSEQ0030
6     WRITE(6,100)                                                     LSEQ0031
      IER=0                                                            LSEQ0032
      RETURN                                                           LSEQ0033
601   F=0.                                                             LSEQ0034
      FD=0.                                                            LSEQ0035
      A=.6931472+S                                                     LSEQ0036
```

```
      AD=.6931472/A                                                    LSEQ0037
      DO600 K=1,N                                                      LSEQ0038
      IF(P(K).LE.0.)GOTO600                                            LSEQ0039
      AUX=P(K)**AD                                                     LSEQ0040
      F=F+AUX                                                          LSEQ0041
      FD=FD+ALOG(P(K))*AUX                                             LSEQ0042
600   CONTINUE                                                         LSEQ0043
      AUX=ALOG(F)                                                      LSEQ0044
      FD=AUX/.6931472-FD/F/A                                           LSEQ0045
      F=AUX/AD                                                         LSEQ0046
      CALL INTTIM(S,A,AD)                                              LSEQC047
      F=F+A                                                            LSEQ0048
      FD=FD+AD                                                         LSEQ0049
      GOTO(501,502),M                                                  LSEQ0050
602   F=0.                                                             LSEQ0051
      FD=0.                                                            LSEQ0052
      DO603 K=1,N                                                      LSEQ0053
      AUX=S*L(K)                                                       LSEQ0054
      IF(AUX.GT.170.)AUX=170.                                          LSEQ0055
      AUX=P(K)*EXP(AUX)                                                LSEQ0056
      F=F+AUX                                                          LSEQ0057
603   FD=FD+AUX*L(K)                                                   LSEQ0058
      CALL INTTIM(S,A,AD)                                              LSEQ0059
      FD=FD/F+AD                                                       LSEQ0060
      F=ALOG(F)+A                                                      LSEQ0061
      GOTO(501,502),M                                                  LSEQ0062
604   CALL INTTIM(S,A,AD)                                              LSEQ0063
      F=L(1)*S+A                                                       LSEQ0064
      FD=L(1)+AD                                                       LSEQ0065
      GOTO(501,502),M                                                  LSEQ0066
100   FORMAT(' NO CONVERGENCE IN LSEQ1')                               LSEQ0067
      END                                                              LSEQ0068
```