September 1981 LIDS-P-1132

PROTOCOLS FOR ENCODING IDLE CHARACTERS IN DATA STREAMS†

by

J. A. Roskind*
and
P. A. Humblet**

ABSTRACT

A memoryless source produces one of the characters 1, 0, i (i signifies "idle") every unit of time. The probabilities associated with the characters are

$$P(1) = P(0) = \frac{p}{2}$$

$$P(i) = 1-p$$

A transmitter encodes these characters as they are produced and generates 1's and 0's at a rate of one character per unit time. These 1's and 0's are then transmitted over an error free channel. A receiver is only required to reconstruct exactly the sequence of 1's and 0's produced by the source. The delay encountered by a bit is then the length of time between its generation by the source and its reconstruction by the receiver. For a given transmitter-receiver and a given value of p we can calculate the expected bit delay (if it exists).

The goal of this thesis is to find transmitters and receivers which result in small bit delay. The main result is the proof of the existence of a class of strategies with delay growing as $\log(\log(\frac{1}{1-p}))$ as $p \rightarrow 1$.

---

†This paper was sumbitted as a master's thesis in June 1980 at M.I.T.

*Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Partial support furnished by the John and Fannie Hertz Foundation.

**Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Support furnished by NSF-ECS 79-19880.

Bell Laboratories

subject: **PROTOCOLS FOR ENCODING IDLE CHARACTERS IN DATA STREAMS†**
Charging Case Number 11175-102,
Filing Case Number 20878

date: July 1, 1981

from: **J. A. Roskind***
P. A. Humblet**

TM 81-11217-10

MEMORANDUM FOR FILE

CHAPTER I

INTRODUCTION

1.0 Idle Encoding Problem

A memoryless source produces one of the characters 1, 0, i (i signifies "idle") every unit of time. The probabilities associated with the characters are

$$P(1) = P(0) = \frac{p}{2}$$

$$P(i) = 1-p \triangleq q$$

A transmitter encodes these characters as they are produced and generates 1's and 0's at a rate of one character per unit time. These 1's and 0's are then transmitted over an error free channel. A receiver is only required to reconstruct exactly the sequence of 1's and 0's produced by the source. The delay encountered by a bit is then the length of time between its generation by the source and its reconstruction by the receiver. For a given transmitter-receiver and a given value of p we can calculate the expected bit delay (if it exists).

The problem is most interesting when p is near 1 (q near 0). Although for any p, there exists transmitter-receiver pairs that offer finite expected delay, for all known encoding strategies the expected delay is forced arbitrarily high by taking p sufficiently close to (but not equal to) 1. In contrast to this, we note that when p is 1, the problem is obviously solvable with zero delay! This rather large discontinuity leaves open the question of whether there

exists a strategy holding the expected delay below some bound not dependent on p. The section that follows will, however, show that at least a small discontinuity does exist in the achievable delay.

There are two motivations for studying this problem. The first is a problem encountered when data is being received at a node synchronously based on one clock and transmitted synchronously based on a second clock. If we take into account the probabilistic relative drift of the two clock rates, then we are effectively forced to randomly include idles in the data stream. The second motivating problem is the efficient encoding of bursty data sources [2]. The problem was actually formulated by R. R. Boorstyn in this context.

Chapter II of this thesis will provide the necessary background in the use of flags in data streams. Flags are essential to all idle encoding schemes that will be discussed. Chapter III will present a description and analysis of a known scheme that has delay $\log\frac{1}{q}$ as $q \to 0$. Chapter IV contains the analysis and description of the main result of the thesis. A class of strategies are shown to exist that have delay growing like $\log(\log\frac{1}{q})$ as $q \to 0$.

1.1 Lower Bounds on Delay

In order to obtain a lower bound, we will assume that the receiver in this system has the assistance of a genie. The genie supplies the receiver with a copy of the sources output one unit of time after it is produced. Clearly any encoding scheme that might have worked without the genie may still be used. Hence the optimal encoding scheme in this new system has delay which under bounds that of the original system.

The problem now reduces at each point in time to the following: The transmitter must encode the character produced by the source (either a 1, a 0, or an i) into the characters 1 and 0 for transmission over the channel. All past information is known by both the transmitter and the receiver (by way of the genie) and hence is irrelevant to the transmission.

Without loss of generality we can assume the transmitter encodes the source character 1 into the channel character 1. We then consider the two possible encodings of the source character

0. If it too is encoded into a channel character 1, then clearly the delay of the system is always one unit of time. If it is encoded into a channel character 0, then there is a symmetric situation in which we can encode the i into either 1 or 0. Assume we choose to encode i to a 1. Clearly then when the source produces a 1, and the transmitter sends it, it will encounter one unit of delay, whereas a source character 0 will encounter no delay. Since the source produces 1's and 0's with equal probability, the expected delay is then .5 units of time. Notice that the above argument only holds if there is some non-zero probability of an i occurring. This then clearly establishes the discontinuity of achievable delay as p approaches 1.

Higher order lower bounds may be computed by using a genie that delivers every n units of time a copy of the previous n source characters. The resulting bounds are piecewise polynomial of order $n-1$. To generate such a bound requires consideration of all possible mappings from 3^n strings to 2^n strings. Such computation is doubly exponential in complexity and hence not feasible for n beyond 3 or 4.

CHAPTER II

FLAG STRATEGIES

This chapter is devoted to the study of flag strategies. They are essential to our solutions and will be analyzed in some detail. We will first define them and illustrate their use by some examples. We later explain the concept of decoding delay associated with flag strategies. Finally, we will introduce the notion of random flags and analyze the associated decoding delay.

2.0 Fixed Single Flag Strategies

Fixed single flag strategies involve the use of a prespecified sequence of characters (a flag) on a data channel to indicate the occurrence of some event. If the event does not take place the characters are used to send some other information. If the information being sent is accidentally encoded into the beginning of the specified flag sequence, then the transmitter inserts a character into the data stream to distinguish the sequence from the actual flag. The receiver on the other hand is able to detect such an insertion and effectively remove it before decoding the remaining channel sequence.

Such strategies can be understood most simply by examining an example relevant to the idle encoding problem. Assume the data channel is binary (with characters 1 and 0). The flag sequence is defined to be 1000. The information that is to be encoded is a sequence of bits (1's and 0's) which are encoded into channel characters in the obvious way.

Sample Sequence # 1

Information	α_i	—	0	1	0	0	1	1	1	0	0	0	0	0	...
Time	i	—	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Channel contents	x_i	—	0	1	0	0	1	1	1	1	0	0	1	0	0	0

The explanation of this encoding is as follows:

The bits $\alpha_1 \ldots \alpha_4 = 0100$ are placed directly onto the channel (characters $x_1 \ldots x_4$). At this point it should be noted that the last 3 bits $x_2x_3x_4 = 100$ very nearly forms a flag sequence. The transmitter must perform a bit insertion and so $x_5 = 1$. The receiver also notices sequence $x_2x_3x_4 = 100$ and can conclude that since $x_5 = 1$, it was an insertion and $x_2x_3x_4$ are actually

data. Information transmission continues as α_5 through α_9 are placed on the channel as x_6 through x_{10} (as stated, x_5 is an insertion). Again there is an insertion at $x_{11} = 1$ (as $x_8x_9x_{10}$=100) and then information $\alpha_{10}\alpha_{11}\ \alpha_{12} = 000$ is sent as characters $x_{12}x_{13}x_{14}$. Notice that a convention is assumed here as channel characters $x_{11}x_{12}\ x_{13} = 100$ are *not* considered to be even possibly the start of a flag. Channel character x_{11} was an insertion and has been recognized as such by both the receiver and transmitter. The convention is then that insertions are excluded from use as channel history when checking for flags (Channel history at time 14 is $x_{10}x_{12}x_{13}$=000).

As a second example we will encode the same information, with flags placed on the channel after channel characters x_3 and x_{13}.

Sequence # 2

Information	α_i	—	0	1	0	0	1	1	1	0	0								...
Time	i	—	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Channel controls	x_i #	—	0	1	0	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0

Here we have bits $\alpha_1\alpha_2\alpha_3 = 010$ placed on the channel as $x_1x_2x_3$. At this point, some temporal event has occurred and it is desired that a flag be sent. The channel characters $x_4...x_7 = 1000$ represent that flag. Data bits $\alpha_4...\alpha_9$ are then placed on the channel as characters $x_8...x_{13}$. At this point it is desired that another flag be placed on the channel. Channel character x_{14} must be an insertion. Channel characters x_{15}—$x_{18} = 1000$ form the flag.

From these examples it can be seen that the convention of precluding the use of insertions in a flag sequence has various effects. In the first example we saved one bit insertion by not having to insert $x_{14} = 1$. In the second problem the convention cost an extra bit of transmission as x_{14} could not be part of the flag (alternative convention would allow $x_{14}x_{15}x_{16}\ x_{17}$=1000 to be a flag). There is also some uncertainty in the receiver in the second example as to whether the event which triggered the flag transmission occurred after x_{13} or x_{14} (both would result in the same channel sequence).

Although the convention just mentioned had little effect on a strategy with a well chosen flag, it can have a tremendous effect on performance with an ill chosen flag. By "well chosen"

we mean bifix free [1], in the sense that no prefix of the flag root ("root" $\triangleq$ first F bits of an $F+1$ bit flag) is a suffix of the flag root. The niceness of a bifix free flag lies in the fact that insertions can never occur during a flag transmission, and so an insertion would also imply that the next possible insertion is at least F bits away.

As an example of difficulty arising from an ill chosen flag consider the extreme case where a channel with binary characters and binary information has the flag 1110 (Convention-insertions can be part of a flag).

Sample Sequence #3

Information	α_i	=	1 0 0 1 1 1 0 0 1 0 ...
Time	i	=	1 2 3 4 5 6 7 8 9 ...
Channel Characters	x_i	=	1 0 0 1 1 1 1 1 1 1 1

Notice that $\alpha_1 ... \alpha_6$ are transmitted straightforwardly. However, when this alternate convention is applied, we see that not only is $x_7 = 1$ an insertion ($x_4x_5x_6$=111), but so is $x_8 = 1$ an insertion as $x_5x_6x_7 = 111$, and x_9 and x_{10} ad infinitum. The point of this example is that the choice of what convention to use and what flag sequence to use can heavily impact upon the associated strategy's performance.

As final examples we will examine the application of the former convention to a fixed flag which has a root which is not bifix free. This convention is that an insertion bit (once detected by the receiver) is effectively deleted from the data stream. All further flag transmissions would skip by the bit insertion as though there were no such bit. For this example assume a flag of the form 10100 (note: the root 1010 has a suffix 10 which is also a prefix) with binary information.

Sample Sequence #4

Information	α_i	1 0 1 0 0 0 ...
Time	i	1 2 3 4 5 6 7
Channel Data	x_i	1 0 1 0 1 0 0 ...

Explanation: Information bits $\alpha_1 ... \alpha_4$ are placed directly on the channel ($x_1 ... x_4$=1010). The channel character $x_5 = 1$ is an insertion. Since x_5 is an insertion, it is no longer possible for α_1 to be the first bit of a flag and so α_5 is sent as $x_6 = 0$. Notice that the effective past history of the channel is now $x_1x_2x_3x_4x_6 = 10100$ with the provision that x_1 is not the first bit of a flag.

(Note: x_5 is deleted). With this in mind we see that $\alpha_6 = 0$ is placed directly on the channel as x_7 without the need for an insertion.

As a further example suppose the same sequence of information bits are transmitted but it is desired that a flag be sent starting at channel character 3.

Sample Sequence #5

Information	α_i	1 0 1 0 0 0 ...
Time	i	1 2 3 4 5 6 7 8 9 10 11 12
Channel data	x_i	1 0 1 0 1 1 0 0 1 0 0 0 ...

Explanation: Information bits $\alpha_1\alpha_2 = 10$ are placed immediately onto the channel as $x_1x_2 = 10$. Starting at time 3 it is desired that a flag be sent and so this is started by sending $x_3x_4 = 10$ (the first two bits of 10100). At time 5 however an insertion is required and is placed on the channel $x_5 = 1$. The receiver's detection of this insertion leaves an effective past history of $x_1...x_4 = 1010$ at time 5 with the provision that x_1 is not the first bit of a flag. At time 6 the transmitter resumes sending the flag started at time 3 by sending $x_6x_7x_8 = 100$. At time 8 the effective past history of the channel is $x_3x_4x_6x_7\ x_8 = 10100$ which is a flag and is identified as such. Channel characters $x_9...x_{12} = 1000$ then carry the information $\alpha_3...\alpha_6$. Notice that this convention can be used with any flag without ever deadlocking in a continuous insertion state. The exact reason for using such a convention will become clear later, but a major feature is the ability to handle (nonambiguously) insertions in the middle of flags. The conventions of the above example will be used in Section 2.2.

2.1 Decoding Delay of a Fixed Single Flag Strategy

The decoding delay associated with a strategy is defined as the time interval between the placement of a bit of information on the data channel by the transmitter and its positive decoding by the receiver as a bit of information. Looking back at sample sequence #1 we see that bit $\alpha_2 = 1$ is placed on the data channel as x_2 but is not positively decoded as the data bit 1 until x_5 is received and so it has decoding delay 3. Notice that until $x_5 = 1$ was received,there was still a possibility (as far as the receiver is concerned) that x_2 was the first bit of a flag. With this in mind it is clear that $\alpha_3 = x_3 = 0$ has delay 2 and $\alpha_4 = x_4 = 0$ has delay 1. We also see that

$\alpha_5 = x_6 = 1$ has delay 1. The reason for this delay is that as a 1, it is potentially the first bit of flag.

Having defined the decoding delay of a data bit, we can consider the expected decoding delay for a bit when a specific flag and strategy (with associated conventions) are chosen. For example if we use an $(F+1)$ bit flag of the form 100...00 and the conventions of sample sequence #1 and #2, the delay encountered by a bit during decoding is no more than the waiting time until a 1 is next transmitted over the channel. If we assume that insertions are rare, the expected decoding delay approaches the expected waiting time for the next information bit to be a 1. If the information source is memoryless with equal a priori probabilities of 1's and 0's, we then have the expected decoding delay is 2 bits.

2.2 Fixed Length Random Flags

Although single fixed length flag schemes can be described rather nicely, the analysis of their performance can be quite complex. Specifically the probabilities of insertions at close points in time are very dependent. In order to overcome these dependencies as much as possible we will introduce the concept of fixed length random flags. The idea is that at each point in time a sequence defining a flag root is selected randomly (selections are independent with a uniform distribution) from all possible sequences of the fixed length. Both transmitter and receiver know the chosen roots and the necessary insertions and identifications can be made. The following ALGOL program illustrates the details of such a protocol. It is assumed that LENGTH and FLAGLST are predefined constants representing the length of the flag root and the infinite list of randomly generated flag roots. It is assumed without loss of generality (as far as probabilistic analysis is concerned) that every flag ends with a "0" and that insertions are always "1". The conventions used here were also used in the end of Section 2.0.

It was somewhat difficult to arrive at a consistent and well defined protocol for this section. We are supplying the ALGOL program so as to be sure that algorithms used by the receiver and transmitter are totally defined.

```
PROCEDURE MAIN;
      COMMENT: This is system environment
      BEGIN
            STRING  FLAGLST [1:∞, 1:Length];
                  COMMENT: All flags for all time;
            WHILE TRUE DO
                  BEGIN
                        READ NEXTCHAR;
                        ENCODE (NEXTCHAR);
                              COMMENT: This is the way we
                                    drive the transmitter.  NEXTCHAR
                                    may be a "1", a "0", or an "F".
                  END;
```

```
PROCEDURE ENCODE (ARG); VALUE ARG; STRING ARG;
    BEGIN COMMENT: This is the transmitters environment;
        OWN STRING HISTORY [1:Length];
        OWN INTEGER TIME:=1;
            COMMENT: HISTORY maintains a list of
                transmitted bits. TIME is used to
                keep track of which flag is relevant;
        INTEGER I;
        STRING FLAG [1:Length];

        PROCEDURE TRNSMT (BIT), VALUE BIT; STRING BIT;
            BEGIN
                INTEGER I;
                AGREE:=TIME > Length;
                    COMMENT: No insertions till TIME > Length;
                FOR I: = 1 TO Length WHILE AGREE DO
                    AGREE:= HISTORY [I] = FLAGLST [TIME-Length, I];
                    COMMENT AGREE tell us if we matched
                    HISTORY with a flag prefix;
                IF AGREE THEN
                    CHANNEL ("0");
                    COMMENT: A "0" following a flag prefix
                    is an insertion. A "1" would be
                    used for an actual flag;
                CHANNEL (BIT);
                    Comment: Then we send the actual bit;
                TIME:=TIME+1;
                FOR I:=1 TO Length - 1 DO
                    HISTORY [I]:=HISTORY [I+1];
                HISTORY [Length]:=BIT;
                    COMMENT: Shift the HISTORY vector;
        END;
        COMMENT; Notice this routine only increments
            TIME when HISTORY is shifted. This keeps the
            FLAGLST in sync with HISTORY;
        COMMENT: Now for th . body of PROC ENCODE ... ;
        IF ARG = "F" THEN
            BEGIN COMMENT: We are being asked to send a flag;
                FOR I:=1 TO Length DO
                    FLAG[I]:=FLAGLST[TIME,I];
                    COMMENT: First save a copy of the flag prefix;
                FOR I:=1 TO Length DO
                    TRNSMT(FLAG[I]);
                    COMMENT: Then send the prefix with
                    any neceɔɔary insertions;
                CHANNEL ("1");
                    COMMENT: Then complete the actual flag
                    with a "1". No time increment is
                    necessary as "1" is not added to
                    HISTORY;
            END;
        ELSE TRNSMT(ARG);
                COMMENT: If we are not sending a flag,
                    then we're sending data;
        END; COMMENT END of ENCODE;
```

```
PROCEDURE CHANNEL(BIT); VALUE, BIT; STRING BIT;
    COMMENT The channel just passes info from
        the receiver to the transmitter.
    RCV(BIT)

PROCEDURE RCV(DATA);VALUE, DATA; STRING DATA;
    BEGIN COMMENT; This is the receiver's environment
        OWN STRING HISTORY[1:Length];
        OWN INTEGER TIME:=1, HSTRLNGTH:=0
        OWN BOOLEAN INSERT:=FALSE;
            COMMENT: HISTORY and TIME function the
                same as in the transmitters environment.
                The integer HSTRLNGTH indicates the number
                of bits in HISTORY that might be
                actual data bits. The boolean INSERT
                indicates that the last bit received
                was an insertion.;
    INTEGER I;
    BOOLEAN AGREE;
    IF INSERT=FALSE THEN
        COMMENT; If there wasn't just an insertion,
                then we must check to see if
                HISTORY is a flag prefix.;
        BEGIN
                AGREE:=TIME > Length;
                FOR I=1 TO Length WHILE AGREE DO
                    AGREE:=HISTORY[I]= FLAGLST[TIME-Length, I];
                COMMENT:AGREE tells us if we actually
                    have a match;
                IF AGREE THEN
                    BEGIN COMMENT: Must be a flag or insertion;
                        IF DATA=0
                        THEN INSERT:=TRUE
                        ELSE BEGIN
                            HSTRLNGTH:=0;
                            WRITE "FLAG RECEIVED";
                            END;
                        RETURN;
                    END;
                END;
                ELSE INSERT:=FALSE;
                COMMENT: Clear the INSERT Flag;
                COMMENT: Now in any case we don't have
                    a flag prefix in HISTORY so we
                    shift in the new bit
                IF HSTRLNGTH=Length
                    THEN WRITE (HISTORY[1]);
                    ELSE HSTRLNGTH:=HSTRLNGTH+1;
                FOR I:=1 TO Length-1 DO
                    HISTORY[I]:=HISTORY[I+1];
                HISTORY [ Length ]:=DATA;
                TIME:=TIME+1;
        END; COMMENT: END OF RCV;
        END;
```

Note the receiver function presented in the ALGOL program (RCV) introduces decoding delay of at least $F - 1$.

The purpose of the program is to clarify the protocol description, but it should be clear that a more "intelligent" receive function could be defined to achieve minimal decoding delay, (i.e., writing a data bit as soon as it is determined that it is not part of a flag).

As an example we can consider the following channel data sequence that could occur with this strategy, and the resulting interpretation by an "intelligent" receiver. The flags at each point are 4 bits long and are designated $f_k (k=1,2,...,)$ where k references the location of the first bit of that flag.

time	t	—	1	2	3	4	5	6	6′	7	8	8′	9	10	11	...
Channel data	X_t	—	0	1	0	0	0	1	1	1	0	...				
	f_1	→	1	0	0	1										
	f_2	→	0	1	0	1										
	f_3	→	0	0	0	0										
	f_4	→	1	1				1	0							
	f_5	→	0					1	1	1						
	f_6	→	0	1				0	0							
	f_7	→	0					1	1	1						

time	1	2	3	4	5	6	6′	7	8	8′	9	10	11	12	13	...
f_t (output)	0	1	-	-	-	0	0	-	-	0	1	1	...			

Explanation - At time $t = 1$, bit $x_1 = 0$ is received. This bit is different from the first bit of $f_1 = 1001$ and so $y_1 = 0 = x$, is the output at time $t = 1$.

Similarly $x_2 = 1$ is immediately passed by the receiver ($y_2 = 1$).

At time $t = 3$, however, $x_3 = 0$ is seen on the channel and the receiver cannot conclude that x_3 is not part of $f_3 = 0000$. The receiver must wait for further information ($y_3 = -$).

The bit $x_4 = 0$ might still be part of a flag transmission $f_3 = 0000$ as $x_3x_4 = 00$. Again the receiver must wait ($y_4 = -$). With the receipt of $x_5 = 0$, the possibility still exists that $x_3x_4x_5 = 000$ are the first 3 bits of $f_3 = 0000$, and so we have $y_5 = -$.

Finally $x_6 = 1$ is seen on the channel and the following conclusions may be drawn:

1) x_6 is an insertion bit and does not represent data;

2) x_3 was not part of f_3, as f_3 was not sent;

3) x_4 was not part of f_3 (and could not be part of f_4);

4) x_5 was not part of f_3 (or f_4) but might be part of $f_5 = 0111$ as $x_5 = 0$.

As a consequence of these conclusions the output at time $t = 6$ is $y_6 = 00 = x_3x_4$. The time is not incremented at this point (by convention) as the bit was an insertion.

With the receipt of $x_{6'} = 1$ there is still the possibility that $x_5x_{6'} = 01$ are the first two bits of $f_5 = 0111$ (notice x_6 is effectively deleted). Again as $x_7 = 1$ the possibility remains that $x_5x_6x_7 = 011$ is the start of f_5. The output are respectively $y_{6'} = -$ and $y_7 = -$.

When bit $x_8 = 0$ is observed it is clearly an insertion and the receiver is able to write (with due consideration of $f_{6'}$ and f_7) $y_8 = 011$. Again with the example of the random flag strategy we see that decoding delay is still well defined. The decoding delays associated with the example are

Name	Value	RCV Algorithm		Intelligent Algorithm	
		When Decoded	Delay	When Decoded	Delay (minimal)
x_1	0	4	3	1	0
x_2	1	5	3	2	0
x_3	0	6	3	6	3
x_4	0	7	4	6	2
x_5	0	8	4	8	4
$x_{6'}$	1	9	4	8	2
x_7	1	10	4	8	1

2.3 Expected Decoding Delay

Having defined the protocol we will now consider the expected decoding delay of this system when an "intelligent" receiver is used. The averaging is taken over all possible choices of flags and all possible data sequences.

There are three causes for decoding delay. The first is that the sequence of channel bits, around the data bit of interest, may match the prefix of one or more flags for some length of time. Until there is a discrepancy between the channel sequence and *every* possible flag sequence containing that data bit, the decoding of that bit as an actual data bit must be delayed. The second cause for delay is that if it is necessary to look at k additional data bits (as just

mentioned), there may be anywhere from 0 to k insertions between those data bits. Finally there is the possibility that even though the receiver only has to look at k data bits, it may have to look at the channel bit that follows to insure that it is an insertion. We now define K to be the random variable corresponding to the number of data bits that must be examined. We define L to be the random variable corresponding to the number of insertions among the k data bits, and we define M to be the number of insertions that follow the last data bit that must be observed (clearly M is 0 or 1).

We will now proceed to derive the joint distribution of K, L and M. To this end, we will first compute the marginal distribution

$$Pr\{K = x\}$$

Let the j^{th} data or flag root bit placed on the channel be called α_j, and let the sequence P_j be the potential flag root sequence that would start with α_j and continue through α_{j+F-1}. The event that no additional data bits need be observed, following α_j, in order to decode α_j as a data bit is then:

$$\begin{aligned}\{K < 1\} &= \{(\alpha_j \text{ differs from the first bit of } P_j)\} \\ &\cap \{\alpha_{j-1}\alpha_j \text{ differs from the first 2 bits of } P_{j-1})\} \\ &\vdots \\ &\cap \{\alpha_{j-(F-2)}\ldots\alpha_j \text{ differ from the first } F-1 \text{ bits of } P_{j-(F-2)}\}\end{aligned}$$

Now, since the flags are chosen randomly and independently, the probability of the above event will take the product form. Finally using the assumption that the flag root selection is based on a uniform distribution over sequences of length F, we have

$$Pr(K < 1) = \frac{1}{2} \cdot \frac{3}{4} \cdot \ldots \frac{2^{F-1}-1}{2^{F-1}}$$

$$= \prod_{j=1}^{F-1} \frac{2^j-1}{2^j}$$

$$\left(\text{Note } \prod_{j=a}^{b} f(j) \triangleq 1 \text{ when } b < a\right)$$

More generally we have

$$Pr(K < x) = \prod_{j=x}^{F-1} \frac{2^j - 1}{2^j}$$

Now we make use of the equation

$$Pr(K = x) = Pr(K < x+1) - Pr(K < x)$$

to get the desired marginal distribution

$$Pr(K = x) = \begin{cases} \frac{1}{2^x} \prod_{j=x+1}^{F-1} \frac{2^j - 1}{2^j} & \text{when } x \leq F - 1 \\ 0 & \text{when } x > F-1 \end{cases}$$

Having derived the marginal distribution on K (the number of data bits that must be observed in order to conclude that the current bit is a data bit) we will now derive the conditional distribution

$$Pr(L = y | K = x)$$

Recalling now that L is the number of insertions following a data bit but prior to the last data bit that must be observed, we see the distribution straightforwardly. When $K = x$ there are exactly x places that an insertion can occur. These insertions are independent of each other and the event $K = x$. This can be seen as an insertion after α_{j+i} $(i < x)$ depends only on $f_{j+i-F+1}$, but the event that $K = x$ is totally, independent of $f_{j+i-F+1}$ (as all root compositions are independent). The probability of any such insertion is then:

$$Pr(\text{insertion}) = 2^{-F} \triangleq r$$

So the above conditional distribution is binomial for $0 \leq L \leq K$ and is specifically

$$Pr(L = y | K = x) = \binom{x}{y} r^y (1 - r)^{x-y}$$

Finally we will find the distribution of M, the number of insertions that follow the kth data bit (note: the only possibilities are 0 or 1) given that we must observe exactly k additional data bits to decode the current data bit as such. (Note: M is totally independent of L given K, and the

conditioning is just a formality).

$$Pr(M_x = 1 | K = x \& L = y) = \frac{Pr(M_x=1 \& K=x)}{Pr(K=x)}$$

$$= \frac{Pr(K=x \,|\, M_x=1) Pr(M_x=1)}{Pr(K=x)}$$

$$= \frac{\prod_{j=x+1}^{F-1} (1-2^{-j}) 2^{-F}}{\frac{1}{2x} \prod_{j=x+1}^{F-1} (1-2^{j})} = 2^{x-F}$$

We now obtain: (for $0 \leq y \leq x \leq F-1$)

$$Prob(\{K=x\} \cap \{L=Y\} \cap \{M=1\}) = (2^{x-F})$$

$$\cdot \binom{x}{y} r^y (1-r)^{x-y}$$

$$\cdot \left(\frac{1}{2^x} \prod_{j=x+1}^{F-1} (1-2^{-1}) \right)$$

$$= 2^{-F} \binom{x}{y} r^y (1-r)^{x-y} \prod_{j=x+1}^{F-1} (1-2^{-j})$$

and

$$Prob(\{K=x\} \cap \{L=y\} \cap \{M=0\}) = (1-2^{x-F})$$

$$\cdot \binom{x}{y} r^y (1-r)^{x-y}$$

$$\cdot \left(\frac{1}{2^x} \prod_{j=x+1}^{F-1} (1-2^{-j}) \right)$$

$$= (2^{-x} - 2^{-F}) \binom{x}{y} r^y (1-r)^{x-y} \prod_{j=x+1}^{F-1} (1-2^{-j})$$

Now we calculate the expected decoding delay: (for a given F)

$$E(K+L+M) = E(K+L+M|M=1)\cdot P(M=1) + E(K+L+M|M=0)\cdot P(M=0)$$

$$= \left[2^{-F}\sum_{x=0}^{F-1}[x+1+rx]\prod_{j=x+1}^{F-1}(1-2^{-j})\right]$$

$$+ \left[\sum_{x=0}^{F-1}(1^{-x}-2^{-F})x(1+r)\prod_{j=x+1}^{F-1}(1-2^{-j})\right]$$

$$= 2^{-F}\sum_{x=0}^{F-1}(1+x2^{-x})\prod_{j=x+1}^{F-1}(1-2^{-j}) + \sum_{x=0}^{F-1}x2^{-x}\prod_{j=x+1}^{F-1}(1-2^{-j})$$

It can be shown by induction that

$$2^{-F}\sum_{x=0}^{F-1}(1+x2^{-x})\prod_{j=x+1}^{F-1}(1-2^{-j}) = F2^{-F}$$

which has the interpretation of the expected number of insertions that follow a bit (i.e., $E(L+M)$).

The second term in the sum is defined

$$A(F) \triangleq \sum_{x=0}^{F-1}x2^{-x}\prod_{j=x+1}^{F-1}(1-2^{j})$$

and may be interpreted as the expected number of bits that must be examined (i.e., $E(K)$). By examination of $A(F)$ it can be seen that the following recursive relationship is satisfied:

$$A(F+1) = (1-2^{-F})A(F) + F2^{-F} \quad (A(0)=0)$$

If we now define

$$B(F) \triangleq E(K+L+M)$$

we have

$$B(F) = F2^{-F} + A(F)$$

and from the recursion relationship we see

$$B(F+1) = B(F) + \left[\frac{F}{2} - B(F)\right]2^{-F} + F2^{-2F} + 2^{-F-1}$$

This relation allows us to verify by induction that $B(F) \leq F/2$, and thus to conclude that $B(F)$ is a monotone increasing and convergent sequence.

Following are some numerical results:

F	$A(F)$	E(decoding delay) $= B(F)$
0	0	0
1	0	.5
2	.5	1
3	.8750	1.25
4	1.1406	1.3906
5	1.3193	1.4756
6	1.4344	1.5281
7	1.5057	1.5604
8	1.5486	1.5799
9	1.5738	1.5914
10	1.5883	1.5981
11	1.5665	1.6019
12	1.6011	1.6041
⋮	⋮	⋮
∞	1.6067	1.6067

CHAPTER III

3.0 Single Flag Encoding Scheme [2]

Having discussed the nature of flags we are now in a position to describe and analyze one class of transmission schemes that solves the idle encoding problem. In this scheme the transmitter maintains a queue of data bits to be transmitted and sends them off one at a time whenever possible. If at some time the queue becomes completely empty and it comes time for a bit to be sent, the transmitter then sends some prespecified $F+1$ bit flag. The receiver is of course constantly checking for flags and does not confuse such sequences with data. Note that while a flag is being transmitted the bits which are produced by the source will queue up and transmission of data will very likely proceed once the flag has been totally sent.

3.1 Analysis of Single Flag Encoding Schemes

The analysis of this scheme will be performed in two different ways. The first is only an approximate method which requires some intuition about the behavior of random walks. The second way utilizes the theory of queues and is the same method as is used in the later text to analyze the non-deterministic buffer maintenance scheme (although the single flag scheme is a much simpler application).

A typical graph of queue size vs. time for the single flag strategy is given in Fig. 2. The key point to all the analysis that follows is that the start of a new flag represents a renewal point of the random process defined by the queue size. Every time a flag is sent the queue is completely empty. In both methods of analysis the behavior of the process will only be looked at between consecutive renewal points but the expected average delay will be correct for the process as a whole.

The probabilistic drift shown in Fig. 2 between the flags (of length F) is caused by two factors. The first factor is a downward drift (the queue decreases by one) and corresponds to the source producing an idle and the transmitter sending the next bit from the queue. The second factor is an upward drift and corresponds to the source producing a bit and the transmitter not

being permitted to send the next queued bit as it must make a bit insertion

Clearly if we are to have finite expected mean delay in this system, the mean queue size at the transmitter must be finite. It is necessary then that the probability of downward drift exceed the probability of upward drift.

Specifically:

$$Prob\ (insertion\ \&\ no\ idle) < Prob\ (\ no\ insertion\ \&\ idle) \tag{1}$$

Since the probability of an idle being generated by the memoryless source is independent of the sequence of bits produced by the source, (1) simplifies to:

$$Prob\ (insertion) < Prob\ (idle) = q \tag{2}$$

Method I

In order to simplify the analysis it is desirable to make the probability of an insertion independent of the probability of an insertion at any other point in time. Notice for example that if the only flag used is an $F+1$ bit flag of the form 1000 ... 001, then two consecutive insertions could never occur and so the insertion probabilities are clearly dependent. In order to achieve this independence we will choose, at each point in time, the form of the next $F+1$ bit flag. The choice is made at random from a uniform distribution of all possible F bit binary words. (Note: the receiver is synchronized with the transmitter in the selection of these flags). We are not advocating the use of randomized strategies, but they do lend themselves to easy modeling. The probability of an insertion at any point in time is then

$$Prob\ (\ insertion) = \frac{1}{2^F} \tag{3}$$

Combining (2) and (3) we get the stability constraint on our choice of F:

$$\frac{1}{2^F} < q \tag{3}$$

or equivalently

$$\log_2 \frac{1}{q} < F \tag{5}$$

Note that although it is necessary (in order to simplify the analysis) to randomize the selection of flags, the implication that a deterministic scheme can match the performance of the system is not lost. There must exist infinitely many deterministic sequence of flags that match the randomized performance. It is not clear, however, that a single fixed flag scheme can match the randomized performance.

Clearly now the expected delay per bit is at most the sum of the expected receiver decoding delay (known (Section 2.3) to be less than 1.7 bits) and the expected queueing delay. Looking at Fig. 2 we can conclude, if indeed the graph is representative of the process, that the average queueing delay is about $F/2$ (and certainly order of F). The figure would be accurate if the downward drift probability was much greater than the upward drift probability. This would make it very unlikely for the queue to increase up past F. In order to assure this "much greater than" requirement we make use of the fact that the insertion probability decreases exponentially as F increases (see (3)). Instead of barely satisfying the stability criterion given in (5) we set F at:

$$F = \log_2 \frac{1}{q} + m \tag{6}$$

where m is some well chosen number greater than 1. That is, if we fix m at say 20, then the average queuing delay will certainly be $F/2$ as the upward drift probability is 10^{-6} times the downward drift probability. Probably though we could use a smaller m and not increase F so much, thereby achieving smaller delay.

In any event, when m is fixed appropriately, the queueing delay is $F/2$ with F set as in (6). It follows that the queuing delay grows as the order of $\log 1/q$ as q approaches zero. This being the case, the expected total bit delay then grows as $\log 1/q$ as q goes to zero.

Method II

As a second and more rigorous method we will here go about analyzing the expected queuing delay per bit far more exactly. We will find the expected total queuing delay between renewal points (empty queue) and then divide that by the expected total number of bits processed between renewal points. As in the first method we then add the receiver decoding delay to give the expected total delay per bit.

The "total queuing delay between renewal points" is defined to be the sum of the individual queueing delays encountered by each of the bits processed between renewal points. This total may be calculated easily by using the fact that when there are k bits in a queue, then *each* bit accumulates another unit of delay, and the total grows by k. Hence the total queuing delay is simply the sum of the queue lengths achieved during a given period.

In order to calculate the expected total queuing delay and the expected total number of bits processed we must examine the markov chain given in Fig. 1. The states of the markov chain correspond to possible queue sizes. Transitions with probability s correspond to a bit of data being transmitted over the channel and an idle being produced by the markov source. Transitions with probability r represent an inserted bit being sent over the channel while a bit was produced by the markov source and added to the queue. The state denoted RENEW represents reaching of the next renewal point.

Let $T(n)$ be the expected total time till trapping when started in state r. We then have the following equations

$$T(k) = 1 + sT(k-1) + rT(k+1) + (1-r-s)T(k) \tag{7}$$

$$T(0) = 0 \tag{8}$$

Solving for $T(k)$ subject to the initial condition of (8) we get $T(k)$ is linear in k and in fact

$$T(k) = (\frac{1}{s-r})k \tag{9}$$

Similarly let $D(n)$ be the expected total delay till trapping in RENEW when started in state

n. We now have equations

$$D(k) = k + sD(k-1) + rD(k+1) + (1-r-s)D(k) \tag{10}$$

$$D(0) = 0 \tag{11}$$

Solving the difference equation subject to (11) we get

$$D(k) = \frac{1}{2(s-r)} k^2 + \frac{s+r}{2(s-r)^2} k \tag{12}$$

It should be clear that a simple deterministic approximation to the activity of the queue during the flag transmission (with q small) is quite accurate. (More rigor will be introduced when the buffer maintenance scheme is analyzed). With such a deterministic assumption we have that the queue size after flag transmission is exactly the flag length, and the total delay during this period is exactly

$$\frac{F(F+1)}{2} = \text{Total Delay During Flag} \tag{13}$$

Now we just add up the total delay and divide by the total number of bits and get

$$\text{Exp Bit Queuing Delay} = \frac{\frac{F(F+1)}{2} + D(F)}{p(F+T(F))} \tag{14}$$

which may be simplified using (9) and (12) to

$$\text{Exp Bit Queuing Delay} = \frac{\frac{F+1}{2} + \left[\frac{1}{2(s-r)} F + \frac{s+r}{2(s-r)^2}\right]}{p(1 + \frac{1}{s-r})} \tag{15}$$

From the markov chain and its interpretation we have

$$s = \text{Prob (idle \& no insertion)} = q(1 - \frac{1}{2^F}) \tag{16}$$

$$r = \text{Prob (data bit \& insertion)} = (1-q)\frac{1}{2^F} \tag{17}$$

and so

$$s-r = q - \frac{1}{2^F} \tag{18}$$

$$s+r = q + \frac{1-2q}{2^F} \tag{19}$$

As we did in the initial analysis (6) if we let

$$F = \log_2 \frac{1}{q} + m \tag{20}$$

where m is of the order of 5 or 10 (the exact optimal choice isn't worth worrying about) then

$$q >> \frac{1}{2^F} \tag{21}$$

Combining this with (18) and (19)

$$s-r \approx q \tag{22}$$

$$s+r \approx q \tag{23}$$

and so from (21) we see asymptotic behavior of delay as q approaches 0 of

$$\frac{1}{2} \log_2 \frac{1}{q} \tag{24}$$

CHAPTER IV

4.0 Buffer Maintenance Encoding Schemes

The single flag encoding scheme that we looked at is clearly inefficient but it is not clear quantitatively how much of an improvement can be achieved. The root of the problem is that although we use a flag very rarely, it is quite large. The penalty for its transmission is "paid" (i.e., the queue size increases) at one point in time. The aim of the buffer maintenance strategies to be discussed in this chapter is to have small "flags" which are sent more frequently. The penalty is then paid over a period of time and the resulting delay is smaller.

A second interpretation of the previous phenomena is offered by information theory [3]. The presence (absence) of a flag indicates to the receiver that the queue is (not) empty. This event has a small (large) probability, and therefore large (small) self information which results in encoding by a long (short) codeword. The resulting codeword length variance is large and causes unfavorable consequences for delay. Our aim to overcome this problem is to use short "flags" to indicate an event whose probability (as seen by the receiver) is not extreme. In order for such probabilities not to be extreme (as viewed by the receiver) the receiver must "track" the state of the queue more closely and so more frequent protocol transmissions are necessary.

4.1 Deterministic Buffer Maintenance Scheme

In this encoding scheme there are basically two types of "flags". We have a flag (denoted F) and a class of refill strings (denoted S). The function of F is the same as the flag in the single flag encoding scheme. That is, when the transmitter queue is completely empty, F is sent. A refill string is sent at intervals of l bits (l is an integer parameter of the scheme that must be optimized). The class of refill strings is exactly $0^* 1$, i.e., 1 or 01 or 001 or ... or $0^n 1$ or ... . By intervals of l we mean that if the final bit of a refill string was sent at time t, then the bit transmitted at time $t + l + 1$ is the first bit of another S. Notice that the placement of an S in the transmitted data stream has priority over data bits, F, and bit insertions (i.e., bits

added to the data stream to prevent mistaking data for an F). The function of a refill string is to increase the number of bits queued in the transmitter to at least N (N is some fixed number found to be optimal for this scheme). This is achieved by simply sending 0's (assuming it is time to refill) until the queue contains N bits and then sending a 1, which terminates the refill string.

A clearer view of what this scheme is intended to do can be gotten from Fig. 3. The graph is of queue size vs. time. It begins with the queue empty. The transmitter proceeds to send a flag (F) and completes the transmission at Pt. 1. The transmitter then starts transmitting queued up bits over the channel. The downward drift of the queue is caused by the arrival of idles at the transmitter. Every l bits during this downward drift (between Pt 1 and Pt 2) an S is sent. Due to the fact that the queue size is greater than N, the S is simply 1, and the effect on the drift is not noticeable. At Pt 3 however, the queue size is well below N and the S sent has a noticeable effect (the queue size is brought back up to N). These cycles of :

1) Starting with queue size of $\sim N$

2) Probabilistically drifting downward during l bits of transmission (of data and inserted bits)

3) Sending an S to raise the queue size back up to N (assuming we're below N)

are repeated for some time. Eventually Pt 4 is reached and a large number of idles (i.e., much more than expected) causes the queue to become completely empty. At this point an F is sent and a graph similar to the one shown follows.

There is some difficulty in performing an exact analysis of this scheme. The points at which the F is sent represent renewal points of the process. In order to analyze this scheme it then suffices to analyze the behavior of the scheme between renewal points. Analysis here is not simple because the process is not stationary (an S every l bits). The system does cycle but there is no true renewal point. If during the period of time between consecutive S's, the queue size always decreased, then there would always be N bits in the buffer when the last bit of an S is sent, and that point would be a renewal point. Unfortunately the queue size can also rise

occasionally during such an interval.

4.2 Randomized Buffer Maintenance Scheme

In order to get an exact analysis of the delay of a buffer maintenance system, we will now look at a similar scheme where the transition probabilities in the queue size are no longer time dependent between long flags. In this system instead of having l fixed at some l_0, we allow intervals between refill strings to be a sequence of independent random variables $\{l\}$, geometrically distributed with mean l_0. The transmitter and receiver both know $\{l\}$. The reason for the choice of this distribution is the ease with which a markov model can produce it.

The choice of flag roots will also be made random to facilitate analysis. As was shown earlier (Section 2.3) the average decoding delay is quite small (less than 1.61). Note also that as with random flags, infinitely many deterministic sequences of intervals must be able to match the expected performance of the random system. The expectation of performance is taken over all possible $\{l\}$, all possible flag roots, and all possible source sequences.

One final issue to clarify is the priority of placement of a refill sequence versus a long flag. The rule is that refill sequences have priority. They are initially identified by their temporal locations in the data stream (and are self delimiting in their lengths), thus they can be effectively deleted from the channel history as far as detection of long flags (and long flag insertions) are concerned.

Having established the nature of the strategy, we can see that its behavior is cyclically broken into two parts. The first part commences when the transmitter queue is completely empty and the source produces an idle. This state (to be referred to as the trap state) forces the transmitter to send a flag. Some time later the flag will be completed, the queue will have some random number of bits awaiting transmission and the body of our strategy will begin to function. This marks the beginning of the second part of the scheme. This part will eventually terminate in the trap state and the system will repeat the cycle.

The key point to the analysis that follows is that the trap state is a renewal point of the stochastic system. For this reason it suffices to analyze the performance of the system between

trap states in order to evaluate the long term performance of the system.

The mean time that an information bit spends in the buffer is given by the formula:

$$\text{Mean bit queueing delay} = \frac{\text{Mean Total Delay between Renewals}}{\text{Mean Total Number of Bits between Renewals}}$$

4.3 The Model

The source model is given in Section 1.0.

The probability of starting a refill string at any point in time is defined as $S = l_0^{-1}$. We also define (for convenience)

$$S' \triangleq 1 - S$$

The probability that an insertion will be necessary is defined as $T' = 2^{-F}$. Again for convenience

$$T \triangleq 1 - T'$$

Note that this is the probability conditioned on the fact that a refill string is *not* about to begin (refill has priority), and also conditioned on the fact that the last character sent was not an insertion (two consecutive insertions can not occur).

The size of the queue that a refill string will establish is defined to be N.

Having defined these values we see that the state and transitions of the system may be represented by the infinite markov chain shown in Fig. 4. The number associated with each state indicates the number of bits in the transmitter's queue at that time. The state at the far left marked "-1" denotes the trap states. The set of states $C(k)$ ($k \in 0, 1, \ldots N-1$) are the ordinary states of the system. From these states it is possible to transmit a bit (and move to $C(k)$ or $C(k-1)$), or transmit an insertion (and move to $B(k)$ or $B(k+1)$), or to begin a refill string (and move to $F(k)$ or $F(k+1)$). The set of states $F(k)$ ($k \in 0, 1, \ldots N$) correspond to the transmission of a refill string. This set of states can only be left when $k = N$ (i.e., the refill is complete). The presence in one of the states $B(k)$ ($k \in 0, 1, \ldots N-1$) indicate that the last transmission was an insertion (and hence the next may not be). The top set

of states $A(k)$ ($k \in 0, 1, \ldots N$) are also refilling states except they carry the information that the last non-refill character was an insertion or the end of a flag.

The previous discussion has distinguished between the various states when the queue size is less than N. For larger queue sizes there need be no separate state for a refill string, as the refill string is but one bit. For $k \in N, N+1, \ldots$ the states $G(k)$ and $H(k)$ are distinguished only by the fact that presence in $G(k)$ implies the last non-refill bit (perhaps the last bit) was an insertion or the end of a flag.

As an example of a transition and the reason for the probability being assigned to it consider $C(0)$ to $B(1)$ which can be seen to have probability $S'T'p$. First we see "S'" indicates a refill is *not* to be initiated and hence the question of an insertion arises. The "T'" indicates that an insertion is sent over the channel (therefore we move to some $B(\cdot)$ state). Finally the "p" indicates that a bit arrived just prior to the transition and so the queue will increase in size (we didn't transmit anything from the queue).

As a subtle side note, it can be pointed out that the distinction between states $A(\cdot)$ and $F(\cdot)$, $B(\cdot)$ and $C(\cdot)$, and $G(\cdot)$ and $H(\cdot)$ is necessitated by the protocol rule that bit insertions are deleted from the effective past history of the channel (hence two insertions *cannot* occur consecutively). If this rule is not observed, at some point in time *during* an actual flag transmission, a single channel bit might be required to be both an insertion (to disclaim the existence of an earlier potential flag) and also be the next bit of the actual flag! The only solution to such a dilemma would be to forget about the flag that was being sent, send the concluding bit of the earlier flag (i.e. claim the earlier flag was being sent all along!), and push whatever data we have just redefined as the start of a flag, back onto the queue for transmission. Hence the net effect would be to induce an uncertainty in the delay of order of the size of the flag. Such uncertainty has no effect on the analysis of the single flag encoding scheme, but would be intolerable for the buffer maintenance scheme.

The other part of the system that must be modeled is the behavior of the system after trapping and while the flag is being sent. This analysis (given in 4.8) is nearly identical to what

could have been given for the operation of the single flag encoding scheme. The analysis is shown here, however, in order to add rigor and assure the accuracy of the result.

4.4 Expected Time to Trap

Given the markov chain description of the body of this system (Fig. 4), we can easily characterize the expected time to trap when starting in a given state. In words, the motivation for all the equations that follow is that the expected time to trap from any state is one more than the weighted average of the expected times to trap from the states that can next be reached by some transition.

First we define the boundary condition

$$C(-1) = 0 \tag{1}$$

as this is the trap state.

Then we consider the equations for A, B, C and F, valid for $k < N$

$$A(k) = 1 + qA(k) + pA(k+1) \tag{2}$$

$$B(k) = 1 + SqA(k) + SpA(k+1) + S'qC(k-1) + S'pC(k) \tag{3}$$

$$C(k) = 1 + S'TqC(k-1) + S'TpC(k) + S'T'qB(k) + S'T'pB(k+1) + SqF(k) + SpF(k+1) \tag{4}$$

$$F(k) = 1 + qF(k) + pF(k+1) \tag{5}$$

Next we give the boundary conditions that offer the transition from A, B, C and F to G and H.

$$A(N) = 1 + qG(N) + pG(N+1) \tag{6}$$

$$F(N) = 1 + qH(N) + pH(N+1) \tag{7}$$

$$C(N-1) = 1 + S'TqC(N-2) + S'TpC(N-1) + S'T'qB(N-1) + S'T'pG(N) + SqF(N-1) + SpF(N) \tag{8}$$

$$H(N) = 1 + S'TqC(N-1) + (S'Tp + Sq)H(N) + SpH(N+1) \tag{9}$$
$$+ S'T'qG(N) + S'T'pG(N+1)$$

Finally now we state the equations of G and H

$$G(k) = 1 + SqG(k) + SpG(k+1) + S'qH(k-1) + S'pH(k) \tag{10}$$

$$H(k) = 1 + S'TqH(k-1) + (S'Tp+Sq)H(k) + SpH(k+1) \tag{11}$$
$$+ S'T'qG(k) + S'T'pG(K+1)$$

A comparison of (9) and (11) reveals that a more simple statement equivalent to (9) is that

$$C(N-1) = H(N-1) \tag{12}$$

where $H(N-1)$ is simply the extension of the solution to the difference equation. With a bit more analysis it can also be seen that (8) is equivalent to

$$H(N) = C(N) \tag{13}$$

with the obvious extension of C.

The solutions to (10) and (11) are readily seen to be linear in k. A simple substitution allows us to find the following result.

$$H(k) = \frac{(k-N+1)(1+T')}{S'-p(T'+1)} + H(N-1) \tag{14}$$

$$G(k) = \frac{(k-N)(1+T')+1}{S'-p(T'+1)} + H(N-1) \tag{15}$$

The boundary conditions (6) and (7) and the previous result give us

$$A(N) = \frac{1+S'}{S'-p(T'+1)} + H(N-1) \tag{16}$$

$$F(N) = \frac{1+T'+S'}{S'-p(T'+1)} + H(N-1) \tag{17}$$

The solution to (2) and (3) are again seen to be linear in k, more precisely

$$A(k) = \frac{N-k}{p} + A(N) \tag{18}$$

$$F(k) = \frac{N-k}{p} + F(N) \tag{19}$$

with $A(N)$ and $F(N)$ given in (16) and (17)

The equations for $B(k)$ and $C(k)$ are harder to solve. Substituting the expression (3) for $B(k)$ in the expression for $C(k)$ (4) yields the following relation

$$\begin{aligned} C(k) = 1 + S'T' + C(k-1)(S'Tq + S'T'S'q^2) \\ + C(k)(S'Tp + S'T'S'2pq) \\ + C(K+1)(S'T'S'p^2) \\ + S(qF(k) + pF(k+1)) \\ + S'T'S(q^2A(K) + 2pqA(K+1) + p^2A(K+2)) \end{aligned} \tag{20}$$

Note that the sum of the coefficients of the C's, F's and A's on the right hand side is equal to 1, as it should be.

Looking at (20) we see C is the solution to a second order system driven by linear and constant terms (A and F). We therefore know that C has the form

$$C(k) = \beta(k+1) + \gamma(r_1^{k+1}-1) + \delta(r_2^{k+1}-1) \tag{21}$$

which already satisfies (1). Moreover r_1 and r_2 are roots of

$$\sigma(r) \triangleq ur^2 + vr + w = 0 \tag{22}$$

$$\text{with} \qquad u \triangleq S'T'S'p^2$$

$$v \triangleq -1 + S'Tp + S'T'S'2pq$$

$$w \triangleq S'T'S'p^2$$

Note that $\sigma(0) > 0$, $\sigma(1) < 0$ and $u > 0$. This implies r_1 and r_2 are real and positive and lie

on opposite sides of 1. These facts will be recalled later.

A simple substitution of (21) into (20) reveals

$$\beta = -\frac{1}{p} \tag{23}$$

Now by eliminating A and F from (20) by using (14) through (17) we see

$$0 = 1 + S'T' + \frac{1}{p}(S'Tq + S'T'Sq^2 - S'T'S'p^2) \tag{24}$$

$$+ \frac{ST'}{S'-p(T'+1)} - S'T'S + \frac{q}{p}(S+S'T'S)$$

$$+ S(1+S'T')\left[\gamma r_1^N + \delta r_2^N + \frac{1+S'}{S'-p(T'+1)}\right]$$

Note that we have also made use of (12) to eliminate $H(N-1)$, and (21) to eliminate $C(N-1)$. Simplifying we obtain

$$\gamma r_1^N + \delta r_2^N = -\frac{1+S'}{S'-p(T'+1)} - \frac{1}{S(1+S'T')}\left[\frac{1}{p} + \frac{ST'}{S'-(T'+1)}\right] \tag{25}$$

$$\triangleq X$$

In order to obtain a second relation that will completely determine γ and δ, we must use the last boundary condition (13) with (12) in (14), and use the definition of $C(N-1)$ based on (20). The resulting equation is

$$\gamma(r_1^{N+1} - r_1^N) + \delta(r_2^{N+1} - r_2^N) = \frac{S'}{p(S'-p(1+T'))} \tag{26}$$

$$\triangleq Y$$

We can then write

$$\gamma = \frac{1}{r_1^N}\left[\frac{Y - X(r_2 - 1)}{r_1 - r_2}\right] \tag{27}$$

$$\delta = -\frac{1}{r_2^N}\left[\frac{Y - X(r_1 - 1)}{r_1 - r_2}\right]$$

Now back substituting into (21) we see

$$C(N-1) = -\frac{N}{p} - (\gamma + \delta) + X \tag{28}$$

which by way of (12) and (15) gives us

$$G(k) = \frac{(k-N)(1+T') + 1}{S' - p(T'+1)} - \frac{N}{p} - (\gamma+\delta) + X \tag{29}$$

4.5 Small q Analysis - Expected Time to Trap

As this thesis was motivated by the peculiar behavior of the system (Chapter I) as q approaches zero, we will examine the behavior of our scheme for small q, as we adjust the parameters S, T and N accordingly.

Examining the denominator of the expressions for G and H, one sees that to maintain a stable system, S and T' must approach zero at least as fast as q. Thus we let

$$S = sq \tag{1}$$

$$T' = tq, \text{ with } s + t < 1 \tag{2}$$

H and G can then be written

$$H(k) = \frac{(k-N+1)(1+O(q))}{q\ (1-s-t)} + H(N-1) \tag{3}$$

$$G(k) = \frac{k-N+1}{q(1-s-t)}\ (1+O(q)) + H(N-1) \tag{4}$$

Our next task is to compute $H(N-1)$. We will compute successively X, Y, r_1, r_2, γ, and δ.

$$X = \frac{-1}{q}\left[\frac{2}{1-s-t} + \frac{1}{s}\right](1+O(q)) \,. \tag{5}$$

$$Y = \frac{1+O(q)}{q(1-s-t)} \tag{6}$$

next u, v and w defined in (4.4.22) can be rewritten so that the roots r_1 and r_2 of the equation $ur^2 + vr + w = 0$ are

$$r_1 = \frac{(1+s+t) - \sqrt{(1+s+t)^2-4t}}{2t} + O(q) < 1 \tag{7}$$

$$\frac{1}{r_1} = \frac{(1+s+t) + \sqrt{(1+s-t)^2-4t}}{2} + O(q) > 1 \tag{8}$$

$$r_2 = \frac{(1+s+t) + \sqrt{(1+s-t)^2-4t}}{2t} + O(q) > 1 \tag{9}$$

$$\frac{1}{r_2} = \frac{(1+s+t) - \sqrt{(1+s+t)^2-4t}}{2} + O(q) < 1 \tag{10}$$

$$r_1 - r_2 = \frac{-\sqrt{(1+s-t)^2-4t}}{t} + O(q) \tag{11}$$

Using (4.4.28) we see that

$$C(N-1) = \left[-\frac{N}{p} - \frac{1}{qr_1^N}\left\{\frac{\frac{-1}{1-s-t} - \frac{1}{s} + r_2\left(\frac{2}{1-s-t} + \frac{1}{s}\right)}{r_1 - r_2}\right\}\right. \tag{12}$$

$$+ \frac{1}{qr_2^N}\left\{\frac{\frac{-1}{1-s-t} - \frac{1}{s} + r_1\left(\frac{2}{1-s-t} + \frac{1}{s}\right)}{r_1 - r_2}\right\}$$

$$\left. - \frac{1}{q}\left[\frac{2}{1-s-t} + \frac{1}{s}\right]\right](1 + O(q))$$

If we assume that N is also going to be large (as will be forced later) then the above is dominated by the r_1^{-N} term and

$$C(N-1) = \frac{1}{qr_1^N}\left[\frac{\frac{-1}{1-s-t} - \frac{1}{s} + r_2\left[\frac{2}{1-s-t} + \frac{1}{s}\right]}{r_1 - r_2}\right] \tag{13}$$

$$\cdot (1 + O(q)) \cdot (1 + O(r_1^N))$$

Which leads to the final (and usable) result

$$G(k) = \left[\frac{k}{q(1-s-t)} - \frac{1}{qr_1^N}\left[\frac{\frac{-1}{1-s-t} - \frac{1}{s} + r_2\left[\frac{2}{1-s-t} + \frac{1}{s}\right]}{r_1 - r_2}\right]\right] \tag{14}$$

$$\cdot (1+O(q))\ (1+O(r_1^N))$$

4.6 Expected Total Delay Till Trapping

The average total delay till trapping is just the expected value of the sum of the delays encountered by all bits processed during a renewal period. The way this sum is calculated is by keeping effectively a running total of the total delay, and noting that the sum may be updated by simply adding the current queue size to the running sum. (If there are k bits in the queue, then one unit of time later *each* bit will accrue one additional bit delay, or a net increase of k bit delays). The difference equations which must then be solved to calculate this expectation during the activity of the body of the scheme are nearly identical to those given in Section 4.4. The only change is that each equation is driven by a linear term k, instead of the constant term 1.

For completeness we will list these equations. First the low end boundary condition that says no additional delay is accrued after trapping.

$$\underline{C}(-1) = 0 \tag{1}$$

Note that all function names in this section will be underlined to distinguish them from Section

4.4, but still relate them to Figure 4.

The equations relating $\underline{A}$, $\underline{B}$, $\underline{C}$ and $\underline{F}$ are read directly from Figure 4.

$$\underline{A}(k) = k + q\ \underline{A}(k) + p\underline{A}(k+1) \tag{2}$$

$$\underline{B}(k) = k + Sq\underline{A}(k) + Sp\underline{A}(k+1) + S'q\ \underline{C}(k-1) + S'p\ \underline{C}(k) \tag{3}$$

$$\underline{C}(k) = k + S'Tq\ \underline{C}(k-1) + S'Tp\ \underline{C}(k) + S'T'q\ \underline{B}(k) + S'T'p\ \underline{B}(k+1) \tag{4}$$
$$+ Sq\ \underline{F}(k) + Sp\ \underline{F}(k+1)$$

$$\underline{F}(k) = k+q\ \underline{F}(k) = p\ \underline{F}(k+1) \tag{5}$$

Next we list the transition equations that relate the boundary of $\underline{A}$, $\underline{B}$, $\underline{C}$ and $\underline{F}$ to that of $\underline{G}$ and $\underline{H}$

$$\underline{A}(N) = N + q\ \underline{G}(N) + p\ \underline{G}(N+1) \tag{6}$$

$$\underline{F}(N) = N + q\ \underline{H}(N) + p\ \underline{H}(N+1) \tag{7}$$

$$\underline{C}(N-1) = N-1 + S'Tq\ \underline{C}(N-2) + S'Tp\ \underline{C}(N-1) + S'T'q\ \underline{B}(N-1) \tag{8}$$
$$+ S'T'p\ \underline{G}(N) + Sq\ \underline{F}(N-1) + Sp\ \underline{F}(N)$$

$$\underline{H}(N) = N + S'Tq\ \underline{C}(N-1) + (S'Tp+Sq)\ \underline{H}(N) + Sp\underline{H}(N+1) \tag{9}$$
$$+ S'T'q\ \underline{G}(N) + S'T'p\ \underline{G}(N+1)$$

Finally the equations for $\underline{G}$ and $\underline{H}$

$$\underline{G}(k) = k + Sq\ \underline{G}(k) + Sp\ \underline{G}(k+1) + S'p\ \underline{H}(k-1) + S'p\ \underline{H}(k) \tag{10}$$

$$\underline{H}(k) = k + S'Tq\ \underline{H}(k-1) + (S'Tp + Sq)\ \underline{H}(k) + Sp\ \underline{H}(k+1) \tag{11}$$
$$+ S'T'q\ \underline{G}(k) + S'T'p\ \underline{G}(k+1)$$

The solutions to (10) and (11) are clearly going to be quadratic. Some manipulations yield the result

$$\underline{H}(k) = \frac{(k^2-(N-1)^2)(1+T')}{2\Delta} + \frac{k-(N-1)}{2\Delta^2}\left[(1+T')(Sp+S'q) + pTT'\right] \tag{12}$$

$$\underline{G}(k) = \underline{H}(k-1) + \frac{k}{\Delta} + \frac{p}{S'\Delta^2}\left[SS' + pT'\right] \tag{13}$$

where $\Delta = S' - p(T'+1)$

Similarly it can be seen that the solutions to (1) and (4) are of the form

$$\underline{A}(k) = \frac{N^2-k^2}{2p} + \frac{k-N}{2p} + \underline{A}(N) \tag{14}$$

$$\underline{E}(k) = \frac{N^2-k^2}{2p} + \frac{k-N}{2p} + \underline{E}(N) \tag{15}$$

Furthermore, using the boundary conditions (6) and (7) we see that

$$\underline{A}(N) = \underline{H}(N-1) + a \tag{16}$$

$$\underline{E}(N) = \underline{H}(N-1) + f \tag{17}$$

Where

$$a = N\left[\frac{S'+1}{\Delta}\right] - \frac{p(1+T')}{2\Delta} + p\left[\frac{(1+T')(Sp+S'q)+pTT'}{2\Delta^2} + \frac{S'q+ST'p}{S'\Delta^2}\right] \tag{18}$$

and

$$f = N\left[\frac{S'+q(1+T')}{\Delta}\right] - \frac{q(1+T')}{2\Delta} + \frac{(1+p)}{2\Delta^2}\left[(1+T')(Sp+S'q)+pTT'\right] \tag{19}$$

As in Section 4.4, we can solve for $\underline{C}(k)$ in (4) by substituting the expression (3) and eliminating the presence of $\underline{B}(k)$. The resulting difference equation is then:

$$0 = k(S'-2S'T\Sigma) - S'T'pS + SF(k) + S'T'SA(k) \tag{20}$$

$$+ u\,\underline{C}(k+1) + v\,\underline{C}(k) + w\underline{C}(k-1)$$

where u, v and w are as in (4.4.22). We then have a second order linear difference equation, driven by a quadratic function. Consequently the solution must be of the form:

$$\underline{C}(k) = \alpha(k^2-1) + \beta(k+1) + \gamma(r_1^{k+1}-1) + \delta(r_2^{k+1}-1) \tag{21}$$

where r_1, r_2 are roots of

$$ur^2 + vr + w = 0$$

Notice that we have already satisfied (1) by virtue of the form expressed in (21). Also notice that r_1 and r_2 are identical to those in the previous section.

Setting to zero the coefficients of the quadratic and linear terms in (20) quickly yields

$$\alpha = -\frac{1}{2p} \tag{22}$$

$$\beta = \frac{1}{2pS(1+S'T')}\left[1 + S' - S'T'(S + 2p)\right] \tag{23}$$

A comparison of (9) and (11) gives us the boundary condition equivalent to (9) of

$$\underline{C}(N-1) = \underline{H}(N-1) \tag{24}$$

It can also be seen that (8) is equivalent to

$$\underline{C}(N) = \underline{H}(N) \tag{25}$$

where $\underline{C}(N)$ is the extension of $\underline{C}$ using (2), (3), (4) and (5) (Note: several quantities evaluated in this extension have no significance to the model. For example $\underline{E}(N+1)$, $\underline{D}(N+1)$ etc). Using (24) and (25) we can arrive at the equation:

$$\gamma(r_1^{N+1} - r_1^N) + \delta(r_2^{N+1} - r_2^N) = \frac{S'(2N-1)}{2p\Delta} - \beta + \frac{(1+T')(Sp+S'q) + pTT'}{2\Delta^2} \triangleq Y \tag{26}$$

Notice that in the above equation that γ and δ are the only unknowns.

A second equation may be derived by setting the constant term in (20) to zero and making use of (14), (15), (16), (17) and (24). This leads eventually to

$$\gamma r_1^N + \delta r_2^N = -N(\beta + \frac{1}{2p}) - \frac{1}{S(1+S'T')}\left[S'T'S'p \right. \tag{27}$$

$$+ Sf + S'T'Sa + \frac{1}{2p}\left[S'p(T+2S'T'q)-1\right]$$

$$\left. + \beta\left[2S'S'T'p + S'Tp-1\right]\right]$$

$$\triangleq X$$

By solving (26) and (27) simultaneously for γ and δ we complete the determination of $\underline{C}(k)$ using (21). Thus we have defined $\underline{C}(N-1)$ and, by virtue of (24), also defined $\underline{H}(N-1)$. This then completes the determination of the quantities $\underline{H}(k)$ and $\underline{G}(k)$ (using (12) and (13)) which are the key points of the expected delay analysis.

Specifically

$$\gamma = \frac{1}{r_1 N}\left[\frac{Y - X(r_2-1)}{r_1 - r_2}\right] \tag{28}$$

$$\delta = \frac{-1}{r_2^N}\left[\frac{Y - X(r_1-1)}{r_1 - r_2}\right] \tag{29}$$

$$\underline{H}(N-1) = -(\gamma+\delta) - \frac{1}{2p}(N^2-2N) + \beta N + X \tag{30}$$

4.7 Small q Analysis - Expected Total Delay Till Trapping

Continuing with the earlier small q analysis we take

$$S = sq \tag{1}$$

$$T' = tq \quad \text{with} \quad s + t < 1 \tag{2}$$

and see that

$$\Delta = q(1-s-t)\ (1 + O(q)) \tag{3}$$

$$\underline{H}(k) = \left[\frac{k^2-(N-1)^2}{2q(1-s-t)} + \frac{(k-(N-1))\ (1+s+t)}{2q(1-s-t)^2}\right](1+O(q))$$

$$+ \underline{H}(N-1) \tag{4}$$

$$\underline{G}(k) = \left[\underline{H}(k-1) + \frac{k}{q(1-s-t)} + \frac{s+t}{q(1-s-t)^2}\right](1 + O(q)) \tag{5}$$

Looking now to (4.6.18) and (4.6.19) we see that

$$a = \left[N\left[\frac{2}{q(1-s-t)}\right] + \frac{1+s+t}{q(1-s-t)^2}\right](1 + O(q)) \tag{6}$$

$$f = \left[N\left[\frac{1}{q(1-s-t)}\right] + \frac{1+s+t}{q(1-s-t)^2}\right](1 + O(q)) \tag{7}$$

Moving on to (4.6.22) and (4.6.23) we have

$$\alpha = -\frac{1}{2}(1 + O(q)) \tag{8}$$

$$\beta = \frac{1}{sq}(1 + O(q)) \tag{9}$$

From (4.6.26) we see that

$$Y = \left[\frac{2N-1}{2q(1-s-t)} - \frac{1}{sq} + \frac{1+s+t}{2q(1-s-t)^2}\right](1 + O(q)) \tag{10}$$

and from (4.6.27)

$$X = \left[N\frac{t-1}{(1-s-t)sq} - \frac{1+s+t}{q(1-s-t)^2} + \frac{1+s-t}{qs^2}\right](1 + O(q)) \tag{11}$$

Now if we assume N is to be large, then it is clear that (4.6.30) is dominated by γ (Note: it was assumed that $r_1 < r_2$, and it was shown in Section 4.4 that $0<r_1<1<r_2$). We would then have

$$\underline{H}(N-1) = -\gamma \cdot (1 + O((r_1)^N)) \tag{12}$$

$$= \frac{1}{r_1^N}\left[\frac{1}{r_2-r_1}\right]\left[Y + X(1-r_2)\right]\left[1 + O(r_1^N)\right]$$

but

$$Y + X(1-r_2) = N(1 + \frac{t-1}{s}(1-r_2))\left[\frac{1}{q(1-s-t)}\right]\left[1 + O(q)\right]\left[1 + O(\frac{1}{N})\right] \tag{13}$$

and finally

$$\underline{G}(k) = \left[\frac{k^2-2k}{2q(1-s-t)} + \frac{k(2+s-t)}{q(1-s-t)}\right. \tag{14}$$

$$\left. + \frac{N}{qr_1^N}\left[\frac{s+(t-1)(1-r_2)}{s(r_2-r_1)(1-s-t)}\right]\right](1 + O(q))\left[1 + O(\frac{1}{N})\right]$$

4.8 Flag Transmission

In this section we evaluate the probability distribution of the time τ it takes to transmit a flag, together with the expected time and expected delay until the next flag will be transmitted. We assume that if, while a flag is being sent, it comes time to send a refill string, then the shortest possible S is sent (i.e., one bit long).

Together with each bit of a flag root, we may have an insertion with probability T'. After each flag root bit or insertion we may have a geometric run of refill bits, with mean $\frac{S}{S'}$. The Z transform of the probability distribution of the time it takes to transmit a flag is then given by

$$\mathbf{T}(z) = z^{F+1}\left[T\frac{S'}{1-Sz} + T'\left[\frac{S'}{1-Sz}\right]^2\right]^F \tag{1}$$

We then have the mean

$$E(\tau) = \left.\frac{\partial \mathbf{T}}{\partial z}\right|_{z=1} = F + 1 + F\,\frac{S}{S'}\,(1 + T') \tag{2}$$

as well as

$$E(\tau(\tau-1)) = \frac{\partial^2 \mathbf{T}}{\partial z^2}\bigg|_{z=1} = F(F+1)(1+2\frac{S}{S'}(1+T')) \tag{3}$$

$$+ 2F(1+T')\left[\frac{S}{S'}\right]^2$$

$$+ F(F-1)\left[\frac{S}{S'}(1+T')\right]^2$$

If we let m denote the number of information bits stored in the buffer when the flag is completely transmitted, we have that it has a probability distribution with Z transform

$$\mathbf{T}(q+pz) \tag{4}$$

Combining these results with the result given in the previous two sections we have that the mean time between flags is

$$E(\tau) + E(G(m)) \tag{5}$$

and the mean total delay is

$$E\left[p\,\frac{\tau(\tau-1)}{2}\right] + E(\underline{G}(m)) \tag{6}$$

4.9 Expected Queueing Delay

Finally now we can evaluate the performance of this scheme as a function of N, l_0, F and q. Then for a given q, we could optimize the choice of N, l_0, and F. Note that as a consequence of this, for any q, the performance of the optimal buffer maintenance scheme will always meet or exceed the performance of the fixed single flag strategy. This can be seen easily by taking $N = -1$ and l_0 very large and reducing the buffer maintenance scheme to a single flag strategy.

Specifically the performance of the system is given as the ratio of (4.8.5) to (4.8.6) (see equation at the end of Section 4.2). Note that G and $\underline{G}$ are given in (4.4.29) and (4.6.10). The distribution of m may be gotten by taking the inverse Z transform of (4.8.4). The remaining expectations are evaluated explicitly in (4.8.2) and (4.8.3).

4.10 Expected Queueing Delay-Small q Analysis

For fixed F, the variance of τ decreases linearly with q. Therefore the mean time until the next flag is just

$$(F + 1 + G(F + 1))(1 + O(q)) \tag{1}$$

and the mean delay is

$$\left[\frac{F(F+1)}{2} + \underline{G}(F + 1)\right](1 + O(q)) \tag{2}$$

Making use of the equation given at the end of Section 4.2 we take the quotient of (1) and (2) to get the expected queueing delay

$$\left[\frac{\frac{F(F+1)}{2} + \underline{G}(F+1)}{F + 1 + G(F+1)}\right](1 + O(q)) \tag{3}$$

Now since

$$T' = 2^{-F} \tag{4}$$

and

$$tq = T' \tag{5}$$

we have that

$$F = \log_2 \frac{1}{q} + \log_2 \frac{1}{t} \tag{6}$$

We wish to choose N such that

$$\left(\frac{1}{r_1}\right)^N \sim \left[\log_2 \frac{1}{q}\right]^2 \tag{7}$$

The motivation for (7) lies in making the numerator of (3) such that $\underline{G}$ dominates by a factor of N. The natural choice is then

$$N = \log_{\frac{1}{r_1}} \left[\log_2 \frac{1}{q}\right]^2 \tag{8}$$

$$= \frac{1 + \log_2(\log_2 \frac{1}{q})}{\log_2 \frac{1}{r_1}}$$

Expanding (4.5.14) and (4.7.14) we then have

$$G(F+1) = \left[\frac{\log_2 \frac{1}{q} + 1}{q(1-s-t)}\right. \tag{9}$$

$$\left. - \frac{\left[\log_2 \frac{1}{q}\right]^2}{q} \left[\frac{\frac{-1}{1-s-t} - \frac{1}{s} + r_2\left[\frac{2}{1-s-t} + \frac{1}{s}\right]}{r_1 - r_2}\right]\right]$$

$$\cdot (1 + O(q)) \cdot \left[1 + O(r_1^N)\right]$$

and

$$\underline{G}(F+1) = \left[\frac{\left[\log_2 \frac{1}{q}\right]^2 - 2\log_2 \frac{1}{q}}{2q(1-s-t)} + \frac{\log_2 \frac{1}{q}(2+s+t)}{q(1-s-t)}\right. \tag{10}$$

$$\left. + \frac{N\left[\log_2 \frac{1}{q}\right]^2}{q} \left[\frac{s+(t-1)(1-r_2)}{s(r_2-r_1)(1-s-t)}\right]\right] \cdot (1 + O(q))$$

$$\cdot \left[1 + O(\frac{1}{N})\right]$$

Substituting (9) and (10) back into (3) we get the delay grows as

$$N \frac{s + (1-t)(r_2-1)}{-1+t+r_2(1+s-t)} \tag{11}$$

We will show now that the coefficients of N is quite small, without worrying about getting its smallest value. Let us just assume we take t to be small, say 10^{-3}. Take s to be close to 1 but still satisfying (4.5.2). Looking now at equation (4.5.8) we see that

$$\frac{1}{r_1} \approx \frac{1}{1+s} \approx 2 \tag{12}$$

Using this in (8) then gives

$$N \approx \log_2(\log_2 \frac{1}{q}) \tag{13}$$

Looking back at (4.5.9) we see that t being small implies that r_2 is quite large (at least in comparison with the other terms in (11)). Therefore we have the delay in (11) is simply

$$N \frac{s+(1-t)(r_2-1)}{-1+t+r_2(1+s+t)} \approx \frac{N}{1+s} \approx \frac{N}{2} \tag{14}$$

Hence the final result (combining (13) and (14)) is that the average delay of this scheme asymptotically grows no faster than

$$\frac{1}{2} \log_2(\log_2 \frac{1}{q}) \tag{15}$$

as $q \rightarrow 0$.

Acknowledgement

I would like to thank Bob Boorstyn for originating the problem addressed by this thesis, and discussing at length several aspects of the problem. I appreciate the discussions I've had with many members of the Mathematics Center of Bell Laboratories. I am grateful for an extremely swift initial entry of the text of this paper by the word processing center at Bell Laboratories, Murray Hill. Finally I would like to thank David Roskind Jr. for his extensive and repeated proof reading of this paper.

J. A. Roskind*

MH-11217-JAR/PAH-df

P. A. Humblet**

Att.
Figures 1-4
References 1-3

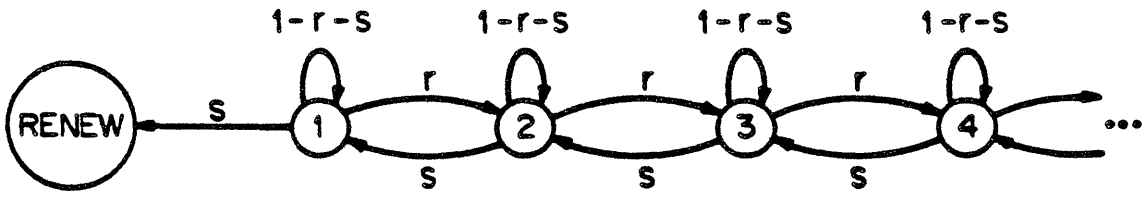

FIGURE 1

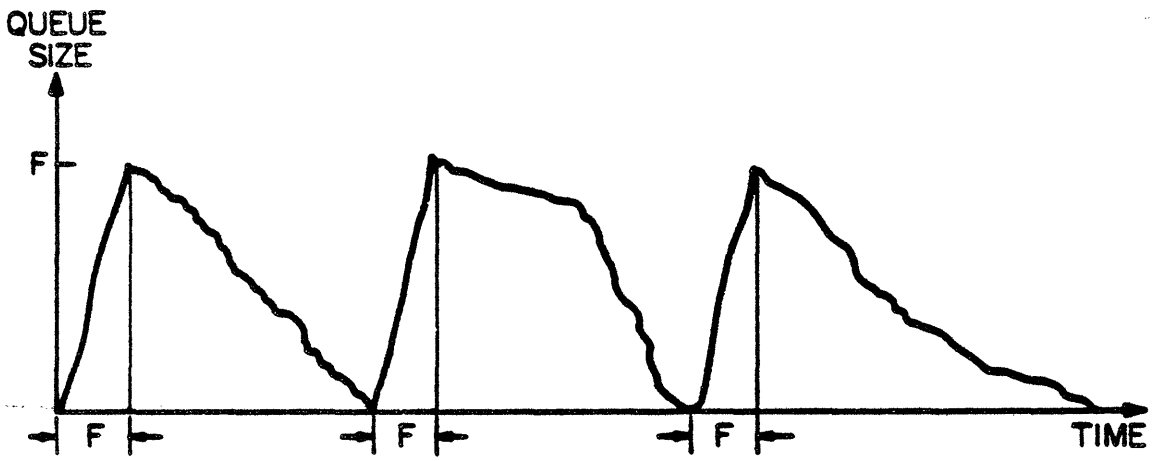

FIGURE 2

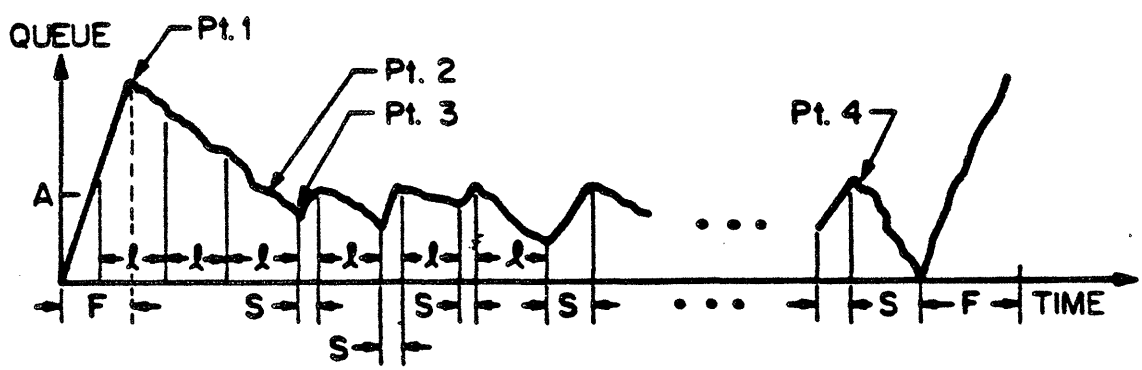

FIGURE 3

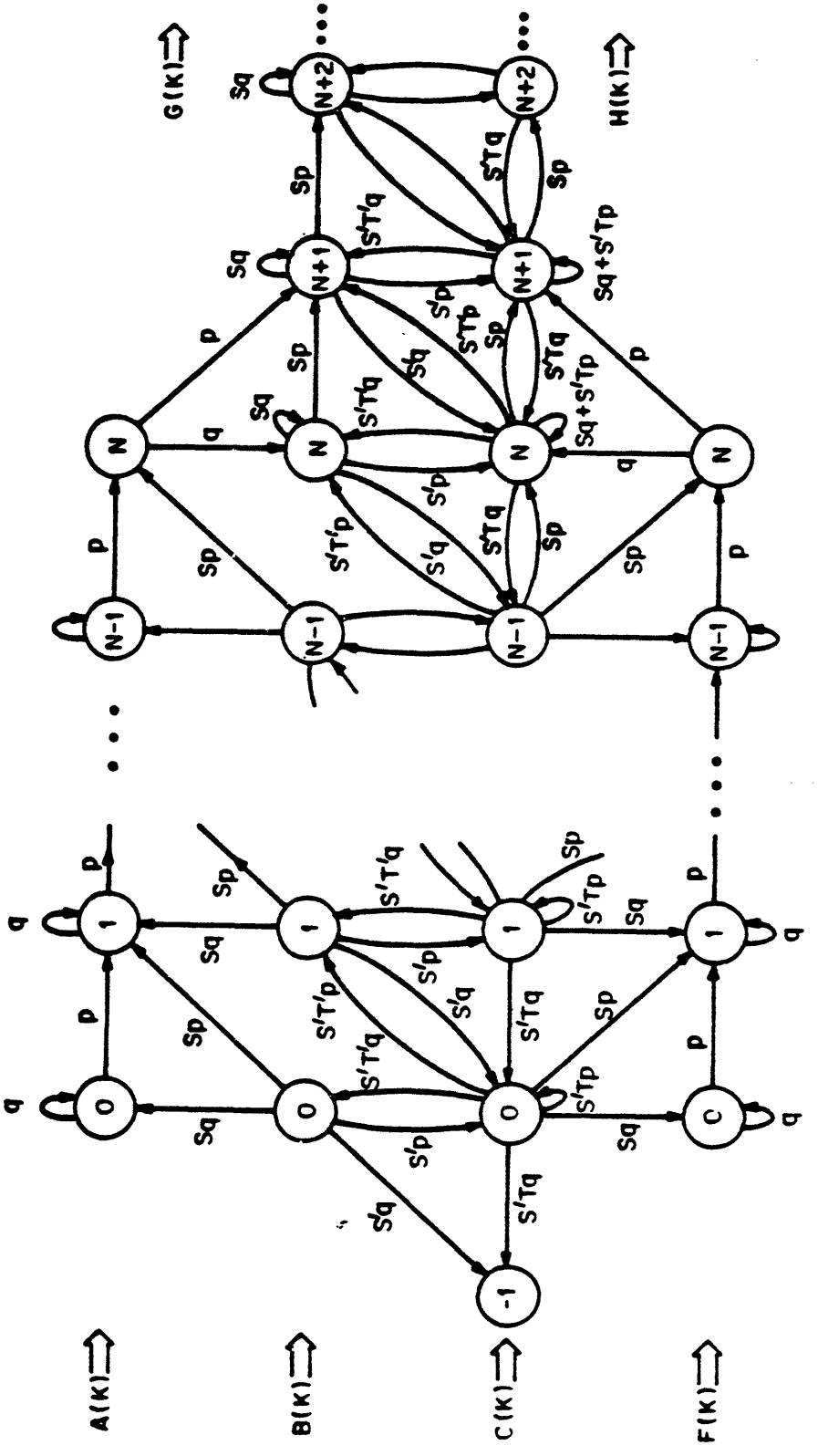

FIGURE 4

REFERENCES

[1] P. T. Nielsen, "A Note on Bifix-free Sequences," *IEEE Transactions on Information Theory*, vol. IT-19 (Sept. 1973), pp. 704-706.

[2] R. R. Boorstyn and J. F. Hayes, "Delay and Overhead in the Encoding of Bursty Sources," *Proceedings of the International Conference on Communications* (June, 1980).

[3] R. G. Gallager, *Information Theory and Reliable Communication*, (New York: John Wiley and Sons, 1968).