Revised January, 1977 Report ESL-P-687

Encoding Message Lengths for Data Transmission*

by

Roger J. Camrass

and

Robert G. Gallager

Massachusetts Institute of Technology
Department of Electrical Engineering
and Computer Science
and
Electronics Systems Laboratory

Abstract

We consider two familiar techniques for encoding message lengths. One technique breaks messages into packets, with each but the last packet of a message having the same length. The message length is encoded by specifying the last packet and its length. The other technique uses a special bit sequence called a flag to terminate the message and slightly re-encodes the message to prevent the flag from appearing within the message. For a geometric message length distribution and for properly chosen parameters, we show that the packet strategy is optimal in the Huffman coding sense and that the flag strategy is very close. Moreover, we show that for a given expected message length the expected code word lengths are quite insensitive to the message length distribution.

---

*This work was supported under Grant NSF/ENG76-24447. Roger J. Camrass is presently at Plessey Telecommunications Limited, Beeston, Nottingham, NG 9 1LA, England.

Key Words: Source Coding, Multiplexing, Data Networks

Appeared in IEEE Transactions on Information Theory, July, 1978

Introduction

Encoding the lengths of messages is a problem that arises frequently in binary data transmission systems. One might think that from a protocol standpoint all that is required is to precede the message with the ordinary binary number representation of its length. There are two disadvantages to such a strategy. The first is that this representation requires a fixed number of bits and thus puts an upper limit on allowable message lengths; the second is that the encoder might not be able to store the entire message while determining its length. There are two common strategies, which we call the packet strategy and the flag strategy, used in practice to avoid the above problems. The point of this note is to show that when the parameters of these strategies are appropriately chosen, and when the distribution of the message lengths is geometric, then the packet strategy is an optimal encoding and the flag strategy is almost optimal.

Packet Strategies

A packet strategy is a strategy in which each message is broken into one or more packets before transmission. We assume that the packets have some maximum length L and that each message is segmented into as many L bit packets as possible with the final packet containing what is left over. If the packets are to be decoded back into messages, then a protocol is clearly necessary to distinguish the last packet and to encode the number of message bits (from 1 to L) in the last packet. Figure 1 shows a simple

form for this protocol. Each non-final packet is preceded by a one and the final packet is preceded by a zero plus a length specifier of approximately $\log_2 L$ bits.

In the Figure, the transmitted sequence, with its length protocol, is shown preceded by a sequence of idle bits followed by a start bit. This should be regarded as a separate protocol to indicate where each message starts; such protocols are necessary for synchronous transmission in which there is sometimes no message to send. Gallager [1] treats message starting protocols in the more general context of networks and shows their relationship to addressing.

With the strategy of Figure 1, the protocol that specifies the length of a message consists of a single bit preceding each packet plus the length specifier. More precisely, for a message of length m, there are $\lfloor (m-1)/L \rfloor$ 1's preceding the non-final packets (where $\lfloor x \rfloor$ denotes the integer part of x) and then a 0 and a length specifier preceding the final packet. This entire set of protocol bits then can be regarded as the unary code for $\lfloor (m-1)/L \rfloor$ (i.e. $\lfloor (m-1)/L \rfloor$ 1's followed by a 0) followed by an encoding of the integers 1 to L. Now assume that the probability mass function on the message lengths is given by

$$P(m) = (1-a)a^{m-1} \quad ; \quad m \geq 1 \tag{1}$$

It is shown in Gallager and VanVoorhis [2] that the optimal binary source code (in terms of minimum expected length) for the integers

with the distribution in (1) is formed as follows: define L to satisfy

$$a^L + a^{L+1} \leq 1 < a^L + a^{L-1} \quad ; \tag{2}$$

then form the unary code for $\lfloor (m-1)/L \rfloor$ and the Huffman code for [m-1] modulo L; the concatenation of these two codes is the desired optimal code for the integers. We observe that this is precisely the code being used in the packet strategy above, and thus the packet strategy is optimal (in terms of minimal expected number of protocol bits) if L is chosen according to (2).

In [2] it is shown that the expected length, $\bar{n}$, of this code exceeds the entropy of the distribution by a quantity that fluctuates between .025 and .033 for mean message lengths $\bar{m} = (1-a)^{-1}$ greater than 12 or so. Furthermore, the entropy of the distribution is $\mathcal{H}(a)/(1-a)$ where $\mathcal{H}$ is the binary entropy function. For large $\bar{m}$, this entropy is approximated by $\log_2 \bar{m} + \log_2 e + 0(1/\bar{m})$, yielding, for future comparison,

$$1.468 \leq \bar{n} - \log_2 \bar{m} + 0(1/\bar{m}) \leq 1.475 \tag{3}$$

It is interesting to note that the packet length L that satisfies (2) is approximated by $L = \bar{m} \ln 2$, which is sometimes inconveniently large. It should also be emphasized that the strategy is only optimal in the Huffman coding sense of mapping message lengths into code words. One could further

reduce the expected number of protocol bits by jointly encoding several message lengths at a time or by jointly encoding the message length and the message data. This is of purely academic interest, however, given the redundancy of a few hundredths of a bit per message.

The optimal strategy above is particularly simple to implement when L is a power of 2 since the Huffman code for the last packet length is just a binary number representation. We now evaluate what happens with the restriction that L is required to be a power of 2. The expected number of protocol bits is given by

$$\bar{n} = E\left[\left\lfloor \frac{m-1}{L} \right\rfloor\right] + 1 + \log_2 L \qquad (4)$$

$$= \frac{1}{1-a^L} + \log_2 L$$

Define $\beta = L/\bar{m}$. Since $\bar{m} = (1-a)^{-1}$, we have $a^L = e^{-\beta} + 0(1/\bar{m})$ and

$$\bar{n} = \log_2 \bar{m} + \frac{1}{1-e^{-\beta}} + \log_2 \beta + 0(1/\bar{m}) \qquad (5)$$

This is minimized, subject to the power of 2 restriction on L, by constraining β to $.48 \leq \beta < .96$. These restrictions determine a unique L for each $\bar{m}$ and lead to

$$1.471 \leq \bar{n} - \log_2 \bar{m} + 0(1/\bar{m}) \leq 1.565 \qquad (6)$$

In other words, restricting L to a power of 2 and letting the length specifier be simply the binary representation of the length of the last packet, costs less than .1 bits per message.

Finally, let us eliminate the assumption that the message length distribution is geometric; we continue to assume that the mean message length, $\bar{m}$, is known. Since $x-1 < \lfloor x \rfloor \leq x$, we can bound $\bar{n}$, as given by (4), by

$$(\bar{m}-1)/L + \log_2 L < \bar{n} \leq \bar{m}/L + 1 + \log_2 L \tag{7}$$

Choosing L to be the power of 2 between $\bar{m}/2$ and $\bar{m}$, (7) can be further bounded as

$$.913 + 1/L \leq \bar{n} - \log_2 \bar{m} \leq 2 \tag{8}$$

This result is somewhat related to universal coding, except that the objective of a universal code is to minimize (over code choices) the maximum redundancy over a set of probability distribution. Here instead we have approximately minimized (over code choices) the maximum expected length over all distributions with given mean.

Flag Strategies

A flag strategy is a strategy in which a unique bit pattern (a flag) of, say, r bits is used to indicate the end of the message (see Figure 2). To prevent premature terminations of the messages, the source must slightly re-encode the messages to avoid appearances of the flag within the messages. This re-encoding is done as follows: if r-1 consective bits of the message stream match the first r-1 bits of the flag, then an insertion of a bit is made into the message, the insertion being the complement of the final

flag bit. The decoder, upon seeing this r-1 bit pattern in the decoded data, either removes the next inserted bit (which is recognizable as the complement of the final flag bit) or accepts the flag if the next bit is the final flag bit (see Figure 3).

We shall consider the bit pattern of a 0 followed by r-1 1's to be our flag. Camrass [3] gives a more complete discussion of the issues involved in choosing a flag. The IBM synchronous data link control (SDLC) procedure [4] uses 0111 1110 as a flag, but the 0 at the end has no function in specifying message length (it allows for other distinguishable control characters with more than 6 contiguous ones).

In analyzing the flag strategy, we shall assume that the messages are composed of independent equiprobable binary digits but that the message distribution is arbitrary with mean message length $\bar{m}$. The expected number of bits used to specify the message length, $\bar{n}$, is r (the flag length) plus the expected number of insertions. For a message of length m bits, we note that the first r-2 bits cannot be followed by insertions, whereas each subsequent bit is followed by an insertion with probability 2^{-r+1}. Thus $E(I|m)$, the expected number of insertions, given a message length m, is $(m-r+2)2^{-r+1}$ for $m \geq r-2$ and 0 otherwise. It is convenient to bound this for all $m \geq 1$ by

$$(m-r+2)2^{-r+1} \leq E(I|m) \leq m\,2^{-r+1} \tag{9}$$

Averaging this over m, we have

$$r + (\bar{m}-r+2)2^{-r+1} \leq \bar{n} \leq r + \bar{m}\, 2^{-r+1} \tag{10}$$

The right hand side of (10) is minimized, subject to the integer constraint on r, by

$$r = 1 + \lfloor \log_2 \bar{m} \rfloor \tag{11}$$

With this choice of r, the upper and lower bounds in (10) are approximately equal for large $\bar{m}$, and we have

$$\bar{n} = r + \bar{m}\, 2^{-r+1} + 0((\log \bar{m})/\bar{m}) \tag{12}$$

This expression fluctuates depending on how close $\log_2 \bar{m}$ is to an integer, and we have

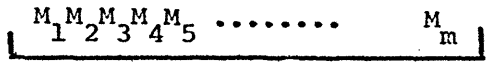
$$1.914 \leq \bar{n} - \log_2 \bar{m} + 0((\log \bar{m})/\bar{m}) \leq 2 \tag{13}$$

Conclusion

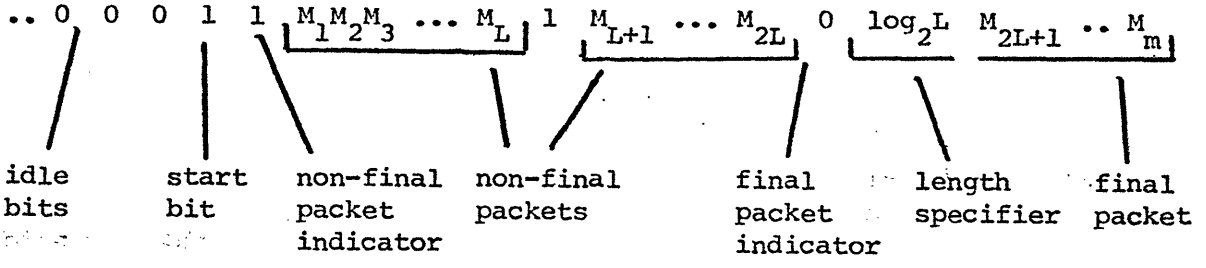
We have formed the expected number of bits $\bar{n}$ required to represent message length using packets (3), (6), (8), and using flags (13). For all practical purposes the strategies are equally efficient and insensitive to the message length distribution. The flag strategy has one practical

advantage for some applications in which messages come into the encoder sequentially with no prior indication of length. The flag strategy can encode such messages with no delay and virtually no storage, while the packet strategies incur a packet's worth of both delay and storage. The flag strategy has the disadvantage, however, that the number of protocol bits is dependent on the data; this causes a slight increase in the second moment of the encoded message length which in turn increases queueing delays (see Camrass [3]).

Message (m bits)



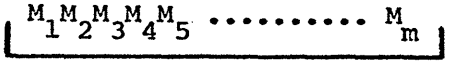
Transmitted block (message and protocol)



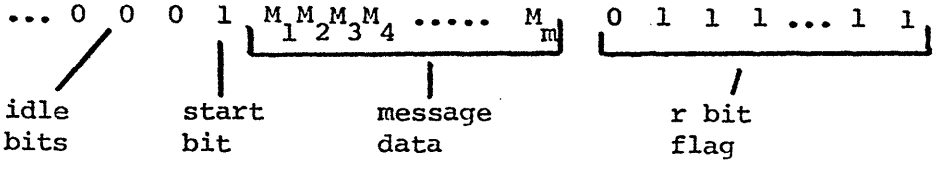
Packet Strategy

Figure 1

Message (m bits)

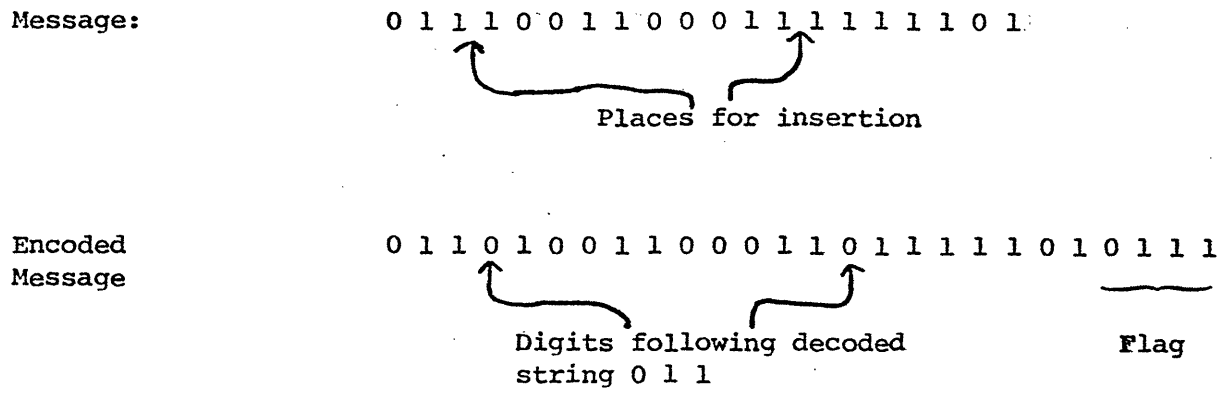


Transmitted block (message with protocol)



Flag Strategy

Figure 2



Examples of Insertions and Deletions
for the flag 0 1 1 1

Figure 3

[1] R. G. Gallager, "Basic Limits on Protocol Information in Data Communication Networks", IEEE Trans. Inform. Theory, Vol. IT-22, No. 4, July, 1976, pp. 385-398.

[2] R. G. Gallager and D. C. VanVoorhis, "Optimal Source Codes for Geometrically Distributed Integer Alphabets", IEEE Trans. Inform. Theory, Vol. IT-21, March, 1975, pp. 228-230.

[3] R. J. Camrass, "Protocol Problems Associated with Simple Communication Networks", Report ESL-R-673, Electronics Systems Laboratory, M.I.T., July, 1976.

[4] R. A. Donnan and J. R. Kersey, "Synchronous Data Link Control: A Perspective", IBM Systems Journal, May, 1974.