Network Working Group   R. Kalin
Request for Comments: 203   MIT Lincoln Lab
NIC: 7168   10 August 1971

Achieving Reliable Communication

ABSTRACT

A non-standard protocol, suitable for either second or third level use, is proposed with the intent of providing error resistant and highly reliable communication channels. Errors introduced by message garbling, message loss, and message pickup are considered. Measures for increasing throughput are also discussed.

AIMS AND LIMITATIONS

It is not our intent to propose the design of a perfect communication channel, rather it is our contention that in the real world there can be no perfect channels and that no amount of protocol can insure the error free transfer of information. Our goal is to explicate the various types of errors that are possible and to provide for each techniques of detection and recovery that, at a cost, can be made arbitrarily good. In this way the mean time between undetected errors can be made as large as necessary.

ERROR TYPES AND DETECTION

Over a message switching facility, such as the ARPA network, all transmission errors can be divided into two classes -- those that result in the loss of an expected message, and those that result in the picking up of an unexpected message. A single bit inversion can cause errors of both types. Error detection can therefore be divided into two components -- one which attempts to determine if the message just received is appropriate at that time, and another which attempts to determine if a message has been lost.

The detection of garbled input messages has been adequately covered by classical coding (elsewhere, mistakenly termed 'communication') theory. Internal message consistency can be determined through the use parity bits, checksum fields, or any of the various coding techniques available for adding some measure of redundancy. With relative simplicity, the likelyhood of an undetected error of this type can be made small enough so as to become inconsequential.

Because it is adequately covered elsewhere, no further discussion shall be given here.

The detection of a message’s external consistency, whether or not it can possibly follow the message that arrived just before it, can also be straight forward. Sequence numbers, if used, can be easily checked. A modulo N sequence field will allow detection of up to N-1 successive message losses. If several concurrent links are in use then sequencing can be maintained for each link. Multi-link single sequence schemes are more complicated and, although used between IMPs for transmission of message packets, they shall be ignored here.

The detection by a receiving host of a lost message can not be determined directly, but rather must be inferred from other observations. Any automatic correction scheme must be prepared to handle the possibility of faulty inference. Message loss would normally be inferred upon the arrival of a message that should follow the one expected. It might also be inferred by the fact that the message expected is long overdue.

ERROR CORRECTION

If a BCH or other error correcting code is used for transmission, errors detected in a message’s internal consistency can sometimes be corrected by the receiving host. In the event that this is not possible, the content of the message is of little use because it can not be relied upon. The only reasonable solution is that of discarding the message and relying upon the recovery procedures implemented for lost messages.

Errors of external consistency can also be treated in the same way. The message can be thrown away and the techniques for recovering lost messages relied upon. Over a critical channel, a slightly fancier technique can at times save some retransmissions. If message N is expected, but message N+1 arrives, there is no need to throw away message N+1 and then recover two messages, it could be saved, and only message N retransmitted.

On noisy channels the technique of saving out of sequence messages can be used to some advantage, especially if recovering from a lost message requires several messages of overhead. On the ARPA network, the measured error rate is so low that its advantages are outweighed by the increase in resident coding.

RECOVERING LOST MESSAGES

The simplest technique I know of for recovering lost can be defined by the following rules:

1) All undiscarded messages have reply messages.
2) Messages with coding errors that can not be corrected are discarded.
3) The receiver can determine if a message is in sequence.
4) Messages received that are out of sequence are discarded.
5) If no reply message is received in N time units since the last transmission, the last message sent is retransmitted (space need not be isochronic).
6) A new message is not sent until the reply for the last one has been received.

The above protocol, if run, is highly effective for continuous communication. Since by rule 6) only one message can be in transit at a time, the necessary sequencing information can be contained in a single bit. Unmodified, it is not suitable for finite communication, since rules 1) and 5) guarantee that there will be no 'last message'. The protocol also does not make very effective use of a pipelined channel, since there is only one message being sent at a time.

Channel throughput can be increased by several techniques, the first of which would be to disassemble the data stream into several (eg. four or eight) streams, transmit each using the above protocol, and then reassemble the streams at the far end. Another technique is to modify rules 5) and 6).

5a)   If no reply has been received to message M in N time units since the last transmission, then messages M, M+1,... are retransmitted.

6a)   There must be no more than L outstanding unreplied messages.

With L equal to one, this protocol degenerates into the first protocol. Increasing L increases throughput until the gain is outweighed by the time spent in error recovery. The larger L, the costlier error recovery. The value of N should be adjusted so that the reply time for a message is usually less than N plus the time to send L-1 messages. Increasing N too much will have the effect of lowering the response time to errors. Decreasing N increases the probability initiating unnecessary retransmissions.

A CRITICAL RACE

The above protocols leave unresolved the the particulars of starting and stopping a finite transmission. In opening a communication channel, what is the sequence number of the first message sent? What will be the first sequence number of the first message sent? What will be the first sequence number of the first reply received? At the end of transmission, how does one signal the ’last message’? The following two rules are introduced:

7) If the same message has been received K times (eg. 50), then it should be accepted as being ’in sequence’. The expected sequencing should be adjusted accordingly. K identical reply messages are then sent.

8) If no reply has been received in J seconds, then the retransmission of the last unreplied message should cease.

With these additional rules a finite transmission is started by repeatedly transmitting the first message until K identical reply messages are received. Sequencing is adjusted accordingly and then subsequent messages can be sent. A conversation is broken by quitting transmission after the reply to the last message you care about has been received. Eventually the other end will stop resending the reply. To avoid ambiguity, the variable J should be less than N times K. Problems will arise if the network crashes for J seconds, for there is a race condition over whether or not the lack of a reply is the result of a channel failure or the end of a conversation.