Network Working Group T. P. Skinner
Request for Comments: 58 MIT Project MAC
June 1970

Logical Message Synchronization

At the last network meeting, the question of logical and physical message distinctions was raised. An argument was made in favor of never running two logical messages together as one or more physical messages. Another method of stating this is that a logical message must begin on a physical message boundary. This did not, however, solve the problem of locating the end of a logical message. A rather poor technique was suggested by myself which consisted of using the first partial physical message as an indication of the last physical message of the logical message. This technique was thrown out for a number of very valid reasons. The solution that seemed most pleasing was the inclusion of some sort of a bit count or data type specification to precede the logical message. Most everyone seemed to like this even though it was stated in a very general way.

As of this writing it appears that it is desired to completely sever the relation between physical and logical messages. This certainly is aesthetically pleasing. However, we are now forced to view the network as a virtually infinite bit stream with no physical delineations. It may well do to transmit a logical header and bit count for each message as long as there are no errors along the line. If, however, a bit is dropped, the problem of synchronization is compounded by the fact that we have no ability to search for the beginning of a logical message other than brute force. An error of this type could be introduced by faulty host or user software/hardware as well as the imp itself. This would involve the shifting of the message bit by bit and seeing if the data looked reasonable. This could certainly be time-consuming as well as introducing the possibility of false synchronism.

I can think of several solutions to the problem at the moment. None of them seems to be very good. Upon losing synchronism, a user could send some form of error message to the other host. The other host could then in return cease sending and wait for a message to continue from the troubled user. This would allow the troubled user to flush out all waiting input. He would then be assured that the next bit started a logical message. The problems here are in assuring synchrony due to input/output buffering in the network and at both hosts. How, for example, can the troubled host be assured he has all the pending data? Once he is sure, he can then resume input assuming all is OK.

Another partial solution requires the original restriction that logical messages always start on physical boundaries. A user then merely has to examine the beginning of each physical message to see if it fits the pattern of a logical message header. This technique is a lot safer than examining the entire input stream as well as being quite a bit faster.

I have not intended to suggest a solution to the problem, but merely bring it to light. If we want to restrict logical messages to begin on physical boundaries we must plan this early in the game. (It probably works out that way in most cases anyway.) Other schemes can be tried later. We must, however, face up to this problem fairly soon.