INWG PROTOCOL NOTE #5

C. Sunshine October 1975

Issues In Communication Protocol Design--Formal Correctness (DRAFT)

Carl A. Sunshine
Digital Systems Lab
Stanford University
October, 1974

INTRODUCTION

This paper discusses some of the important issues in communication protocol design bearing on the correct operation of the protocol. The possibility of loss, duplication, failure to deliver, and out-of-order delivery of messages is analyzed. The problem of confusion between packets from different incarnations of a connection is discussed in companion papers [TO], [BE], and [DA]. Issues of efficiency (bandwidth, delay, retransmission rate, buffer requirements, flow control) will be discussed in a later paper.

Section 1 formally describes the basic failure modes of a simple positive acknowledgement/retransmission protocol without sequencing, flow control, fragmentation, or opening and closing of connections. Section 2 introduces sequencing.

1. BASIC PROTOCOL

DEF: A positive ACK/retransmission communication (PAR) protocol:

Consists of a SENDING DISCIPLINE, a RECEIVING DISCIPLINE, and a TRANSMISSION MEDIUM for transmitting MESSAGES (packets, letters, finite length bit sequences) between users. Purpose is to mask the effect of errors in the transmission medium from the end users of the protocol.

SENDING DISCIPLINE as follows:

Each message submitted by user is assigned a unique identifier. (Ignore the problems of an infinite ID space in this formal definition.) The message is transmitted, and a copy is retained.

Arriving ACK's are checked, and damaged ones discarded*.

When an ACK referencing this identifier is received, the retained copy is discarded (and the user notified of success). If no ACK is received within the retransmission timeout period K, the copy is again transmitted and the cycle repeated. If the retry count has been exceeded, retransmission is suspended (and the user notified).

ACK's for discarded messages are ignored.

RECEIVING DISCIPLINE as follows:

Each message received from the transmission medium is checked for integrity and discarded if damaged*.

If not damaged, then add the message's ID to the list of received message ID's and transmit an ACK referencing the identifier. If the identifier is already in the list, discard the message as a duplicate. Otherwise deliver the message to the user.

TRANSMISSION MEDIUM:

Characterized by such parameters as delivery time (delay), maximum message lifetime in medium, maximum bandwidth, (non unity) loss probability, and (non unity) damage probability.

The complications of addressing, routing, and multiplexing many connections over a single path are ignored here--the protocol is defined for a single connection.

The protocol is said to be INITIALIZED when both sides have empty received message lists and no messages have been sent. (How to reliably accomplish this is discussed in *[TO]*, *[BE]*, and *[DA]*.)

The protocol is functioning CORRECTLY if the protocol has been initialized, and the sending and receiving disciplines are followed without error on both sides. Otherwise the protocol is functioning INCORRECTLY.

*Strictly speaking, it may not be possible to detect all possible damage, resulting in occasional acceptance of a faulty packet or ACK. However, this probability can be made arbitrarily small by use of checksums and other coding techniques.

Definition of failure modes:

DEF: A protocol FAILS TO DELIVER a message if it is possible that a message submitted to the protocol is not successfully delivered.

DEF: A protocol is said to LOSE messages if it can report successful delivery of a message when in fact the message has not been successfully delivered.

DEF: A protocol is said to generate DUPLICATES if a single message submitted by the user to the sending discipline can result in more than one copy of the message delivered by the receiving discipline. (Of course, if the user submits the same message twice, both copies will be delivered at the other end--this is not duplication.)

THM 1: A correctly functioning PAR protocol with infinite retry count never fails to deliver, loses, or duplicates messages.

COR 1A: A correctly functioning PAR protocol with finite retry count never loses or duplicates messages, and the probability of failing to deliver a message can be made arbitrarily small by the sender.

PROOF:

DUPLICATION: No duplicate message generated by the sending discipline or transmission medium will ever be delivered to the user, because in checking the list of received message ID's, the receiving discipline will discard them.

LOSS AND FAILURE TO DELIVER:

There is a nonzero probability that the transmission medium will successfully transmit a message. Hence an infinite retry count implies eventual successful delivery with probability one. (However this may take a long time if the transmission medium is highly unreliable!)

For finite retry counts, the count may be exceeded before successful transmission. However, the user is notified that the message may not have been delivered (it also may have been delivered if the ACK's are lost), and it is up to him to command the protocol to reset the retry count and continue, or give up. The protocol never reports successful delivery falsely, and the user can make the probability of failure to deliver arbitrarily small by increasing the retry count or changing other parameters of the protocol.

QED

THM 2: A PAR protocol with infinite or finite retry count that is functioning incorrectly because the received message ID list is lost (receiver crashes and restarts) will either lose messages, generate duplicate messages, or fail to deliver messages, and the failure probability cannot be made arbitrarily small by the sender.

PROOF: Suppose the protocol was initially functioning correctly. Without loss of generality, let side A be sending message X to side B.

Suppose that when B fails, it loses its received message ID list, but then continues to function normally. Suppose the original transmission of X arrived intact at B and was delivered, but the ACK was damaged or delayed. Then B fails, clearing its received message ID list. A retransmission of X then arrives, and is not detected as a duplicate, hence is delivered to the user.

Alternately, suppose that when B receives any message from A after failing, it notifies A of the failure, and rejects any messages until the protocol is initialized again. In this case A reinitializes the protocol (by some foolproof means beyond the scope of this analysis). But then A must decide what to do about X:

If A sends X, it may be a duplicate as above.

If A doesn't send X, and reports success, the message may be lost (if B failed before receiving a good copy of X)?

If A notifies the user of the failure and the uncertain fate of X, the user has the same possibilities for failure:

Continue trying to send X which may result in a duplicate as above. (This couldn't happen in THM 1.)

Give up which may be a failure to deliver X. Furthermore, the sender cannot make the probability of failure to deliver arbitrarily small by changing parameters available to him, since this failure depends on the reliability of the receiver.

QED

THM 3: A PAR protocol that is functioning incorrectly because the sending discipline loses track of ID's used or messages pending (sender crashes and restarts), will either lose messages or fail to deliver messages.

PROOF:

LOSS: If the sender loses track of ID's, and reuses an ID for a new message, the receiver will ACK it but discard the message as a duplicate. However, the sender will receive the ACK and report successful delivery.

FAILURE TO DELIVER: If the sending discipline loses messages that have been transmitted, but not yet ACKed, it ceases to retransmit them, and they are not delivered. Furthermore the user may not even be notified of the failure.

QED

DISCUSSION: Theorems 1-3 demonstrate the fundamental limitations of PAR protocols: they successfully mask errors in the transmission medium, but not surprisingly, they cannot guarantee reliable transmission when part of the protocol itself is violated due to failure of one side or the other. The information maintained at both sides of the protocol is necessary for correct functioning.

Many protocol designers persist in trying to get around this fundamental limitation by introducing more complicated control mechanisms, usually involving reinitializing the connection. This is really just a special case of opening and closing connections. The issue of (re)initialization and termination is seperable from the issue of reliability within a connection, and theorems 2 and 3 show that given certain types of failure, there can be no guaranteed reliability with PAR type communication protocols.

Those desiring greater reliabilty may implement failure recovery schemes at a higher (user) level (where they meet the same problems), or reduce the possibility of failure with self-checking or redundant machines, back-up stores, checkpointing, or other means.

2. SEQUENCING

The basic PAR protocol above does not concern itself with sequencing. In particular, it is possible due to alternate routing or loss and retransmission within the transmission medium for messages to arrive in a different order from which they were sent. To guarantee in-order delivery, the basic PAR protocol must be augmented with a sequencing mechanism.

DEF: A Sequencing PAR (SPAR) protocol is a PAR protocol with the following additions:

...NG DISCIPLINE: The sending discipline maintains a ...UENCE NUMBER (SN). Each message submitted by the user has SN attached (along with ID), and then SN is incremented. (To facilitate fragmentation or storage allocation, the sequence number may not be on a message basis, but rather on a byte basis as in [OEKA]. However this does not alter the basic behaviour of the mechanism, and will not be considered further.)

RECEIVING DISCIPLINE: The receiving discipline maintains an expected sequence number (ESN). After discarding damaged packets, the message's ID and SN determine the action to be taken according to Table 1.

TABLE 1:

	message SN : ESN		
	lower	equal	higher
id new	can't	ACK, deliver to user, INC, ENTER	discard as out of order
id old	ACK, discard	can't	can't

where ACK means transmit an ACK referencing ID;
INC means increment ESN;
ENTER means enter the message's ID in the received message ID list;
can't means this case cannot occur.

The protocol is said to be INITIALIZED when SN and ESN are equal to each other (may be different in the two directions), and no messages have been sent, and both sides have empty received message ID lists.

Note that in this SPAR protocol, the sequence number and identifier maintain redundant information. In particular, the receiving discipline never needs to check the received message ID list for duplicates, because the ESN screening does this. Since it is easier to remember a single ESN than a potentially infinite list of ID's, the ID can be dropped entirely from the SPAR protocol, with the sequence number performing both the duplicate detection and sequencing functions:

SENDING DISCIPLINE:

The sending discipline maintains a sequence number (SN). Each message submitted by the user has SN attached, and then SN is incremented. The message is transmitted, and a copy retained.

Arriving ACK's are checked, and damaged ones discarded.

when an ACK referencing this SN is received, the retained copy is discarded (and the user notified of success). If no ACK is received within the retransmission timeout period K, the copy is again transmitted and the cycle repeated. If the retry count has been exceeded, retransmission is suspended (and the user notified).

ACK's for discarded messages are ignored.

RECEIVING DISCIPLINE:

The receiving discipline maintains an expected sequence number (ESN).

Each message received is checked for integrity, and discarded if damaged.

if not damaged, compare the message's SN with ESN.

If less, transmit an ACK referencing the message's SN and discard the message as a duplicate.

If equal, transmit an Ack, deliver the message to the user, and increment ESN.

If greater, discard the message as out of order. (For greater efficiency, the receiving discipline may choose to keep some number of out of order messages for a time. The costs and benefits of such schemes will be discussed in a later paper.)

The protocol is said to be INITIALIZED when SN and ESN are equal to each other (may be different in the two directions) and no messages have been sent.

DEF: A SPAR protocol delivers messages OUT OF ORDER if it is possible for messages to be delivered in a different order than they were submitted.

Theorems 1-3 carry over straightforwardly to SPAR protocols.

COR 1B: A correctly functioning SPAR protocol with infinite retry count never fails to deliver messages, loses messages, duplicates messages, or delivers messages out of order.

PROOF: The first three parts are proved as in theorem 1 with the sequence number acting as ID. If a message ever arrives at the receiving discipline before one of its predecessors, the ESN check will cause it to be discarded. Only the next message in order can be delivered to the user. QED

COR 2B: A SPAR protocol that is functioning incorrectly because the receiving discipline loses ESN, will either lose messages, duplicate messages, or fail to deliver messages.

COR 2C: A malfunctioning SPAR where ESN and SN become desynchronized may completely fail to deliver messages.

PROOF: Desynchronization may occur if either the sending or receiving discipline fails to maintain SN or ESN as specified. If ESN winds up below the sequence number of all outstanding messages, the "expected" sequence number will never appear at the receiving discipline, and no message will be accepted. QED

Even if SN and ESN are lost or misset, a malfunctioning SPAR will not deliver messages out of order as long as the ESN screening in the receiving discipline is obeyed. (Duplicates may be delivered as in COR 2B.)

As a practical consideration, the infinite sequence number space assumed for SPAR is just as unrealistic as the infinite unique ID space for PAR. However, a finite sequence number space (with wrap around) can be made to function as if it were infinite, if certain constraints on the number of messages pending, and suitable modulo definitions of "less" and "greater" for sequence number comparisons are adopted. Such a scheme (also including flow control) is described in [ceka] and [cedasu] where ESN corresponds to the receiver's left window edge.

REFERENCES

BE — D. Belsnes, "Note on Single Message Communication", Stanford University, September 1974, INWG Protocol Note #3.

CEDASU — V. Cerf, Y. Dalal, and C. Sunshine, "Specification of Internet Transmission Control Program" (DRAFT), Stanford University, August 1974.

CEKA — V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. C-20, No. 5, May 1974, pp 637-48.

DA — Y. Dalal, "More on Selecting Sequence Numbers", Stanford University, October 1974, INWG Protocol Note #4.

TO — R. Tomlinson, "Selecting Sequence Numbers" (DRAFT), BBN, August 1974, INWG Protocol Note #2.