Reliable Full-Duplex File Transmission over Half-Duplex Telephone Lines

W. C. LYNCH
Case Institute of Technology, Cleveland, Ohio

A field-proven scheme for achieving reliable duplex transmission over a half-duplex communication line is presented, and to demonstrate the difficulty of the problem, another similar scheme, which is only slightly unreliable, is also presented. A flowchart for the reliable scheme and some interesting examples are given.

KEY WORDS AND KEY PHRASES: telephone communication, half duplex, transmission, error correction, full duplex, telephone errors
CR CATEGORIES: 3.81, 4.41, 6.35

1. Introduction

Since data transmission over telephone lines has been an accomplished fact for many years, one would assume that practical and adequate techniques for the control of errors in this noisy medium would have been worked out and promulgated. Confidence in such an assumption appears to be ill-placed with respect to products offered by the major computer manufacturers. The standard hardware and software packages seem to adequately detect errors, but the automatic correction of such errors is quite another matter.

The purpose of this article is to describe an adequate scheme for both the detection and correction of transmission errors over a half-duplex telephone line. This is a difficult problem due to the necessity of sending control, error, and verification information over the same noisy line that first caused an error.

2. Half-Duplex Lines

The transmission scheme presented here is with reference to standard voice grade telephone lines. Such lines may be either a dial-up switched line transmitting at 2000 baud (bits/second) using a 201A modem, or a 2400 baud hard wire line using a 201B modem.

Such lines are called half-duplex because they may transmit in either direction, one direction at a time. By contrast, a simplex line may transmit only one way, while a (full-) duplex line may transmit both ways at the same time. An unpleasant feature of half-duplex lines is the time required to switch directions. A reversal time on the order of 200msec is required to allow echoes to die out. This reversal time significantly complicates the process of maximizing the effective (error-free) data transmission rate.

To avoid confusion it should be observed that the terms simplex, half-duplex, and duplex apply independently at each level of organization. Thus we may have full-duplex transmission of files while the records of that file are interchanged on a half-duplex basis. Simplex file transmission may occur on a full-duplex line. Descriptions of transmission schemes are often inexplicit on this point (e.g. half duplex typically means half duplex by *file*, not record).

3. Reliable Simplex Transmission

The transmission scheme is first presented so as to form a simplex channel, i.e. it will pass data in only one direction. Then we extend the algorithm to cover duplex file transmission (both ways at once).

It is assumed throughout that the system has perfect error detection. To be more precise, if a message is sent from A to B, B will either miss the message completely (*drop* the message), or if it does not drop the message, will be able to determine whether or not the message is in error. For messages on the order of dozens of characters in length, a horizontal and vertical parity scheme similar to that employed on magnetic tape seems to be more than adequate. Odd parity should always be used in order to allow the modems to resynchronize themselves.

It is further assumed that there are no message drops. Later in the paper the algorithm is extended to cover the message drop case, and thus lift this restriction.

To each message sent from A to B we attach an extra bit called the alternation bit. This bit is of course subject to the error checking. After B receives the message it decides if the message had no errors (is error-free). It sends back to A a verification message (*verify bit*) indicating to A whether or not the immediately preceding A → B message was error-free. After A receives this verification one of three possibilities holds: (1) the A → B was good, (2) the A → B message was bad, (3) the verification was in error so that A does not know whether the A → B message was good or bad. In cases (2) and (3) A simply resends the same A → B message as before. In case (1),

A *fetches* the next message to be sent, and sends it, *inverting the setting of the alternation bit* with respect to the previous A → B message.

Whenever B receives a message that is not in error it compares the alternation bit of this new message to the alternation bit of the most recent error-free reception. If the alternation bits are equal the new message is not accepted. If they differ the new message is accepted. The verify from B to A indicates error-free reception independently of the acceptance of the error-free reception.

We can now prove two things. Every message fetched by A is (1) received error-free at least once by B and (2) is accepted at most once by B. Thus every message fetched by A is accepted exactly once by B.

First, for A to fetch a new message to send, two things must occur in sequence. The current A → B message must be received error-free at B, and then the subsequent verification (indicating error-free reception) must be received error-free at A. A will not advance to a new message until this happens. Each message is thus received error-free at least once by B.

B may receive error-free the same message many times due to erroneous B → A messages. B will accept a message, though, only when its alternation bit differs from the alternation bit of the previous accepted message. Thus B accepts only the *first* error-free transmission. Subsequent error-free transmissions of the same message are discarded since their alternation bits have not been inverted. Thus B can accept at most one copy of each message A fetches.

B thus accepts precisely one copy of each message fetched by A.

Initialization of this scheme depends upon A and B agreeing on an initial setting of the alternation bit. This is accomplished by an A → B message whose error-free reception (but not necessarily acceptance) forces B's setting of the alternation bit. Multiple reception of such a message cannot do harm.

4. Some Extensions of the Simplex Scheme

The simplex scheme described in Section 3 can easily be extended to duplex operation. As B sends the verify back to A, it attaches a message for A. This B → A message contains an alternation bit. The A → B message that follows contains the verify information for the previous B → A transmission. Both simplex channels operate essentially independently of each other.

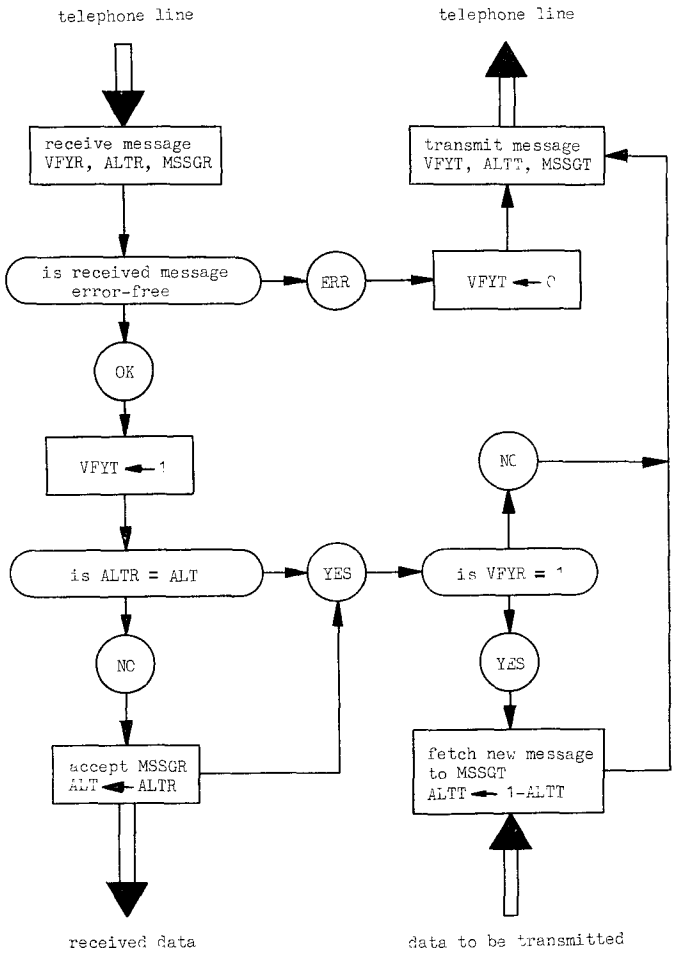
Another obvious extension is to block the data into fairly long messages. This will reduce the effect of the 200 millisecond line reversal time, but it will also increase the probability of error. Often other factors, such as available core storage, limit the block size before the error rate does.

If the information being transmitted is alphanumeric, say FORTRAN programs, it may pay to eliminate blanks by the use of tab and carriage return characters. A modest investment in processing may pay big dividends in improved effective transmission rate.

For special purposes it is possible for a terminal to "lie." A terminal may, at any time, treat an error-free message as though it were in error. This tactic may be useful where, for example, the receiving terminal is temporarily out of buffer space and has no place to file the reception.

This brings us to the topic of message synchronization. The receiving terminal does not have to respond with the verify immediately. It may wait any amount of time before responding. Such a delayed response could allow for extended processing without the danger of another message arriving, since the direction of the messages must strictly alternate. The "real time" problems in the system are thus simplified.

If a message is completely dropped (a condition which we assumed in Section 3 not to occur) the system will hang up.



Legend

VFYR	verify bit in received message
ALTR	alternation bit in received message
MSSGR	data text of received message
VFYT	verify bit in transmitted message
ALTT	alternation bit in transmitted message
MSSGT	data text of transmitted message
ALT	alternation bit of most recent accepted message

FIG. 1. Flowchart for the reliable transmission algorithm

Since drops may result from broken or intermittent connections, this condition must be treated. One solution is to have, say, A send a verify indicating erroneous reception. This would be sent after 5 seconds of quiet time. This will, under most conditions, revive the system without damage (it's one of those "white lies"). Processing time at both A and B should be kept under a second or two in all cases if a 5 second time-out is used. Too long a time-out interval results in wasted time when message drops occur, while too short an interval may restrict the time available for message processing. If the processing time exceeds the time-out interval, messages in opposite directions may "cross" on the line. The resulting "garbage" may cause improper recovery. The bound on processing time is important.

A flowchart for the entire scheme appears in Figure 1.

Figure 2 depicts a typical exchange of information on a noisy line. The odd numbered lines list the state of the message just received by terminal A and the even numbered lines list the state of the message just received by B. The column headings denote the following quantities: *mssg*, the name of the data message transmitted (e.g. AB3 is the third data message from A to B); *err*, whether or not a transmission error was detected; *verfy*, the setting of the verify bit (1 indicates previous message was error-free); *alt*, the setting of the alternation bit; *accpt*, whether or not the message was accepted. Parentheses around a quantity indicates the information as sent (it is unreliable due to transmission errors).

You may construct your own example by performing the following steps.

1. Fill in the *err* column at random with OK's and ERR's
2. On line $n + 1$ *verfy* is 1 if and only if *err* = OK on line n.
3. If line $n = 1$ has *err* = OK and *verfy* = 1 then *mssg* on line $n + 2$ = (1 + *mssg* on line n) and *alt* on line $n + 2$ = (1 − *alt* on line n). Otherwise *mssg* on line $n + 2$ = *mssg* on line n and *alt* on line $n + 2$ = *alt* on line n.
4. Parenthesize *mssg*, *verfy*, and *alt* on all lines with *err* = ERR.
5. Leave *accpt* blank if *err* = ERR. Set *accpt* to YES if *alt* and *alt* on the previous line where *err* = OK disagree. Set *accpt* to NO if *alt* and *alt* on the previous line where *err* = OK agree.

The entire scheme described in this section has been implemented on a UNIVAC 1004, model A. This is a plugboard programmed machine with 961 bytes of storage. The system has been in operation for over two and a half years at Case Institute of Technology. UNIVAC 1004's and UNIVAC 1107's may be used in any combination as the terminals. The operation has been conducted using 201A modem, dial-up service. One such dial-up connection currently in use involves a distance of 100 miles, including crossing state boundaries and passing through both the Bell and General Telephone systems. Line noise on such a dial-up connection has posed no operational problem to this system.

This implementation uses both the blocking and blank compression techniques described earlier. Blanks are removed from the end of each record, and a message consists of as many records as can be accommodated in a 300-character block. Effective print speeds of 150–180 lines/min are attained when the transmission consists of FORTRAN programs and data.

5. An Inadequate Scheme

To underscore the difficulty in constructing adequate schemes for reliable transmission, I will present and analyze a reasonable-looking but inadequate scheme published by one of the major computer manufacturers in a system information manual.

This scheme is again a full-duplex alternating message direction, symmetric terminal scheme, as is the one presented in Section 4. Instead of each message containing both a verify and an alternation bit, as in Section 4, each

	terminal A					terminal B				
	mssg	*err*	*verfy*	*alt*	*accpt*	*mssg*	*err*	*verfy*	*alt*	*accpt*
1	BA1	OK	1	1	YES					
2						AB1	OK	1	1	YES
3	BA2	OK	1	0	YES					
4						AB2	OK	1	0	YES
5	(BA3)	ERR	(1)	(1)	—					
6						(AB2)	ERR	(0)	(0)	—
7	BA3	OK	0	1	YES					
8						(AB2)	ERR	(1)	(0)	—
9	BA3	OK	0	1	NO					
10						AB2	OK	1	0	NO
11	BA4	OK	1	0	YES					
12						(AB3)	ERR	(1)	(1)	—
13	(BA4)	ERR	(0)	(0)	—					
14						(AB3)	ERR	(0)	(1)	—
15	BA4	OK	0	0	NO					
16						AB3	OK	1	1	YES
17	(BA5)	ERR	(1)	(1)	—					
18						(AB3)	ERR	(0)	(1)	—
19	BA5	OK	0	1	YES					
20						AB3	OK	1	1	NO
21	BA6	OK	1	0	YES					
22						AB4	OK	1	0	YES

FIG. 2. A typical message exchange on a noisy line

	terminal A			terminal B		
	mssg	*err*	*ack*	*mssg*	*err*	*ack*
1	BA1	OK	1			
2				AB1	OK	1
3	BA2	OK	1			
4				AB2	OK	1
5	(BA3)	ERR	(1)			
6				AB2	OK	0
7	BA3	OK	1			
8				AB3	OK	1
9	BA4	OK	1			
			3a			
1	BA1	OK	1			
2				AB1	OK	1
3	BA2	OK	1			
4				(AB2)	ERR	(1)
5	(BA2)	ERR	(0)			
6				AB2	OK	0
7	BA2	OK	1			
8				AB3	OK	1
9	BA3	OK	1			
			3b			

FIG. 3. A message exchange in the second scheme

message contains only one control bit, called the acknowledge bit. The operational rules for both terminals are:

1. If the previous reception was error-free, the acknowledge bit of the next transmission is one; if the reception was in error the bit is zero.
2. If the acknowledge bit of the previous reception was zero, or the previous reception was in error, retransmit the old message; otherwise fetch a new message for transmission.

The question of when to accept an error-free reception is left open. This question, in fact, has no consistent resolution. Consider the message exchanges depicted in Figures 3a and 3b. Specifically, should the message received at line 7 be accepted by A? A is presented with exactly the same information in 3a and 3b! A is forced to guess which situation is the one that has occurred. The penalty for a wrong guess is either dropping a message or accepting a duplicate of a message.

If A consistently assumes that 3a represents the situation, A will pick up message duplicates in the (rare) case when two errors occur in sequence as in 3b. Such errors, while rare, do occur, and their rareness will make it extremely difficult to catch the flaw in the system. This inadequate scheme will work *almost* all of the time.

6. Conclusion

A field-proven scheme for achieving reliable full-duplex transmission over noisy half-duplex telephone lines has been presented. The sensitivity of the algorithm and the difficulty of the problem have been illustrated by contrasting the algorithm with another, slightly different algorithm. This modified algorithm fails in rare cases and gives rise to operation which is faulty enough to degrade its usefulness, and not faulty enough to permit it to be easily debugged.

An interesting problem is posed by these two algorithms. The adequate scheme used two bits of control information (verify and alternation bits) per message while the inadequate scheme used only one bit (the acknowledge bit). In Section 3, three states were described for the received message, and the control bits of the next transmission encoded into the two control bits the total information concerning which of the three states held on reception. This leads to the conjecture that at least two control bits are required for any adequate scheme of this sort, and that only one control bit will never do. The reliable duplex transmission problem would, of course, have to be better formalized before it could be claimed that such a conjecture were "proven."

On the Design of Display Processors

T. H. Myer
Bolt Beranek and Newman Inc, Cambridge, Mass.
AND
I. E. Sutherland*
Harvard University, Cambridge, Mass.

The flexibility and power needed in the data channel for a computer display are considered. To work efficiently, such a channel must have a sufficient number of instructions that it is best understood as a small processor rather than a powerful channel. As it was found that successive improvements to the display processor design lie on a circular path, by making improvements one can return to the original simple design plus one new general purpose computer for each trip around. The degree of physical separation between display and parent computer is a key factor in display processor design.

KEY WORDS AND PHRASES: display processor design, display system, computer graphics, graphic terminal, displays, graphics, display generator, display channel, display programming, graphical interaction, remote displays
CR CATEGORIES: 2.44, 6.22, 6.29, 6.35

1. Introduction

In mid-1967 we specified a research display system. This paper describes some of the problems we encountered and some conclusions we have drawn. The display will be an adjunct to an SDS-940 time-shared computer system. The chief purpose for the display and the parent computer is programming research.

When we first approached the task, we assumed we had merely to select one of the several available commercial displays. This proved possible with the analog equipment that constitutes a *display generator*; we found several display generators that combined good accuracy, resolution, and speed. However, the control part of the display, which we have come to call the *display processor*, was another story. We were not completely happy with the command repertoire of any of the commercial systems we saw; we were not sure just how to couple the display to our computer, and above all, we had serious doubts about what a display processor should *be*.

This work was sponsored by the Advanced Research Projects Agency under ARPA Order No. 627, Amendment No. 2, and conducted under Contract No. AF19(628)-5065, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, Bedford, Massachusetts 01730.
* And Bolt Beranek and Newman Inc, Cambridge, Mass.