THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 51, No. 6, July-August, 1972
Printed in U.S.A.

An Experimental Interconnection of Computers Through a Loop Transmission System

By C. H. COKER

(Manuscript received October 13, 1971)

Two laboratory computers have been interconnected through an addressed-block data transmission system (ring) as described by J. R. Pierce and implemented by W. J. Kropfl. This paper gives an idea of the equipment, programming, and protocols of communication through that system.

I. INTRODUCTION

J. R. Pierce has described a digital communications system in which addressed messages are transmitted through a hierarchy of interconnecting loops or rings.[1] Components for one ring have been implemented by W. J. Kropfl.[2] We have used the ring to interconnect two laboratory computers.

1.1 *User's View of the Ring*

To the user, the system resembles a high-speed telegraph service. A message, headed by a destination address, can be "put on the wire," and a moment later, it will be delivered to the addressee. Inside the network, the message is multiplexed onto a loop of circulating message blocks. If the addressee is on the same loop as the sender, the message travels around until it reaches him. If the addressee is on another ring, the message is passed from ring to ring, up and down the hierarchy, until it reaches him. Thereupon it is removed and its place on the loop marked "empty."

But these internal details are invisible to the user. From his viewpoint, he can transmit directly to anyone he designates–without prior negotiation with the network to place a call, without worries of maintaining the connection for possible further communication, or of breaking it when finished.

II. HARDWARE: THE BASIC INTERFACE

Electrical communication with the network terminal ("B" station[1,2]) is simple. To send a message, the user signals the terminal that he is ready to transmit (A in Fig. 1). Milliseconds later, depending on network timing and traffic, the terminal begins sending back clock pulses (B), by which the sender is to shift into the terminal first an address and then data (C).

The message propagates through the network, ultimately to the addressee. If his receive ready line (D) is set, his terminal delivers the message, headed by the sender's address, serially (E) with clock pulses (B) for shifting.

The main constituents of a computer interface to this terminal are a shift register (F), for matching the parallel format of the computer to the serial ring; parallel full-computer-word buffers (G, H), for matching the narrow timing tolerances of the shift register to the more asynchronous responses of the computer; and logic (J), for controlling

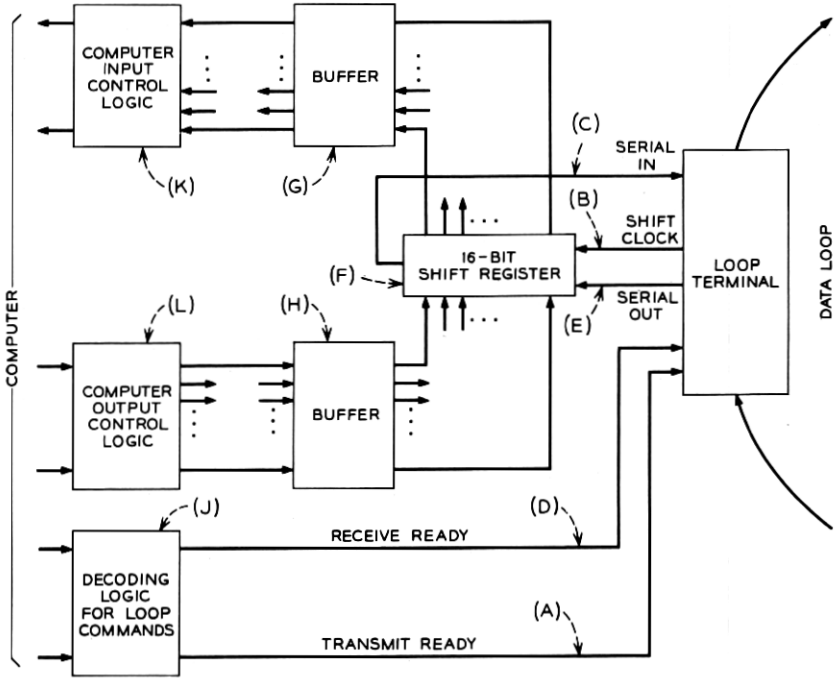

Fig. 1—Basic interface between a computer and the data loop.

read or write lines to the station, and for getting data into (K) and out of (L) the computer. Not shown in the figure are a counter, to segment serial bit streams into computer words, and assorted components to control buffer loading.

A basic interface for computer input-output at a fundamental level (through an "I/O bus") requires about 40 7400-series integrated circuit chips. Depending on the machine, some or most of these interface operations can be accomplished with less special design, using standard computer options.

The speed of the ring allows, in most cases, computer input-output to be done by block transfers through data channels, or to be programmed word-by-word. Some machines, however, are not fast enough to keep up with the T1 carrier loop system, transmitting word-by-word.

2.1 *Options: Buffering to Receive Unexpected Messages*

The simple interface above is adequate for most computers. However, for the very small and the very large, certain changes are desirable.

In a dedicated (one-user-at-a-time) computer, the situation can be controlled so that incoming messages are correctly anticipated and the computer is always prepared to receive them. In a time-shared multi-job computer, however, messages may arrive at any time–even when the computer is attempting to send a message.

One way to guarantee reception of ill-timed messages is to provide separate data channels of the computer for input and output (K and L in Fig. 1), so that the computer can prepare for both sending and receiving at the same time. A less expensive alternative is to provide means to rescind an output command upon appearance of an incoming message. When it recognizes an incoming address, the loop station produces a pulse that can be used for a computer interrupt. The interface must have enough buffer capacity to store incoming data until the interrupt request is honored and the channel or program readied for input. In the computer of this study (a DDP-516), worst-case delay for the sequence (without contemporaneous I/O in other data channels) is 30 μs. Three-word buffers (G in Fig. 1) are sufficient. Depending on the computer, buffers to receive messages by interrupt will require 10 to 20 7400-series chips.

2.2 *Hardware Addressing: Rejection of Intruding Messages*

A different type of unexpected message problem occurs when a machine that is prepared to receive a message from one sender gets a message from another. If the receiver is an unsophisticated device,

the device might be unable to detect the intrusion, or to recover from it in time to receive the desired message.

Hardware to discard messages from any but a designated sender is relatively simple using the 8-bit parallel output of Kropfl's station. Furthermore, the interface can be structured to demultiplex data and addresses onto separate lines, and to perform a similar multiplexing function for transmission.

The address of a correspondent can then be designated once, and will remain the same until redefined. The user has the option of treating the interfaced ring as an addressed-block system, by setting a new address for each transmission; or of treating it as a direct line, by leaving the address the same for as long as he wants. Data blocks can be any size–possibly much larger than a loop block. At the user's option the interface can reject intruding messages, or set off alarms or program interrupts.

Engineering for such features is straight-forward. About 10 integrated circuit chips are required if addresses of correspondents are to be set manually; 20 chips, if they are to be changeable under program control.

III. USE OF THE EXPERIMENTAL LOOP

Two identical Honeywell DDP-516 computers of the Bell Laboratories Acoustics Research facility have been connected to a local ring. The computers have 16 thousand 16-bit words of 0.96-μs memory; 800 thousand words of 3.3-megabit disk memory; hardware multiply, divide, double precision, and floating point; and printers, card readers, and analog-to-digital and digital-to-analog conversion equipment. One of the machines has 300- and 2000-baud *Dataphone*® data sets. The two machines are used for a variety of on-line applications in speech analysis, synthesis, and perception research.

3.1 *The Experimental Interface*

The two computer interfaces use 4-μs/word data channels and provide hardware multiplexing of data and addresses. In addition, the interface recognizes a special bit in the data block, and upon receipt of a block with that bit set, causes a program interrupt. These interrupt or "command blocks" simplify synchronization and provide more positive control of a remote computer in program debugging.

Figure 2 is a photograph of a loop station installed above its interface circuit. The interface is done in a card logic used for other devices on this computer. Engineering time was approximately 1.5 man-months.

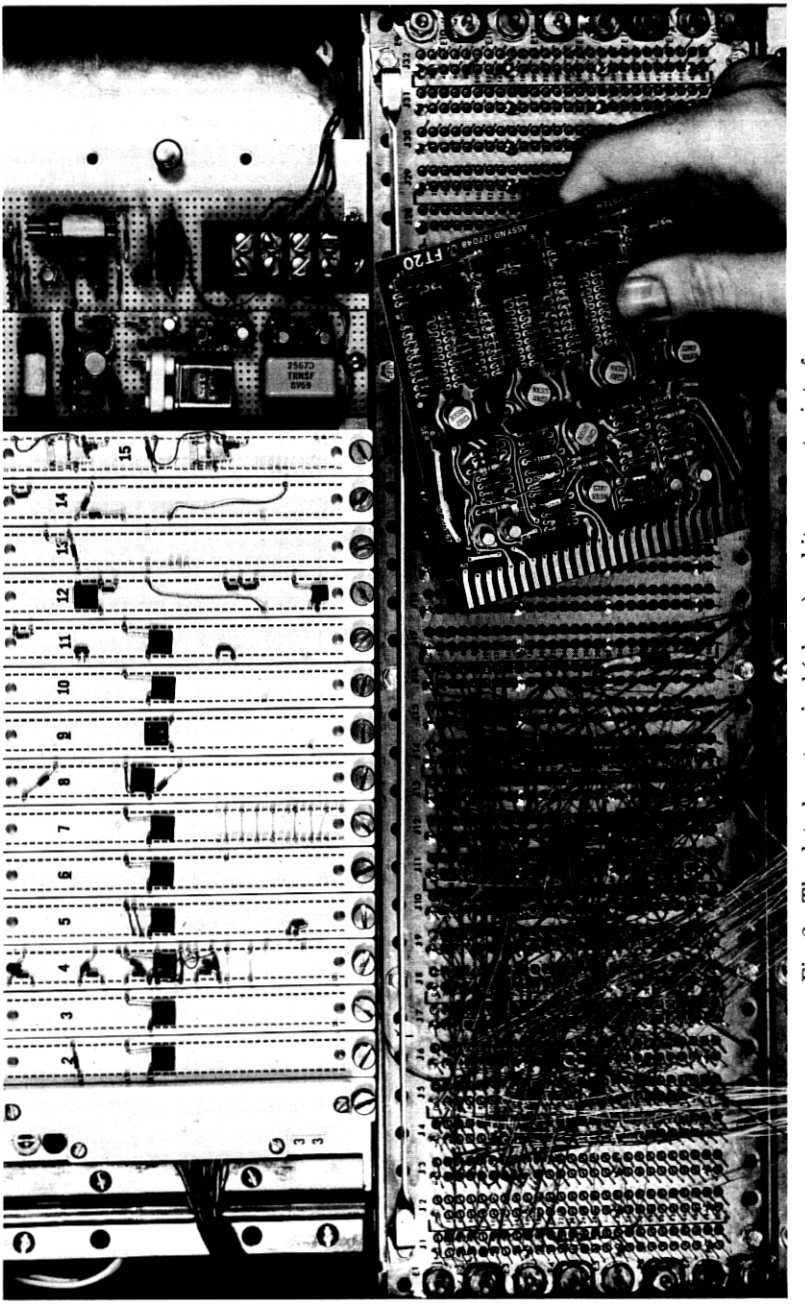

Fig. 2—The data loop terminal (above) and its computer interface.

IV. SOFTWARE

Programming for the ring system is essentially the same as for other means of transmission. Problems of error detection, correction, and retransmission; problems of format compatability between dissimilar computers; problems of distinguishing data from control information; etc., are in no way different for the ring. Users are free to transmit or receive as the network permits. There are no required self-disciplines, except perhaps to discourage nuisance calls or "junk mail."

4.1 *Program Synchronization*

A point that deserves mention is a result of high-speed transmission in general, not specifically of the ring system. Without programmed precautions, the speed of the T1 loop allows a slow receiver to be overrun by a fast sender, whereas with slower transmission, the channel itself might have been the limiting factor.

A trivial solution, appropriate for simple receiving terminals, is simple open-loop control of transmission rate. The sender, knowing the limitations of the receiver, waits for a fixed time after transmitting each block before sending the next. With more capable receivers, a basic "dialogue" procedure that is good practice for other transmission media also works well for the loop. In an initial "handshake," sender and receiver agree that a fixed amount of data will be sent, after which the sender will hold, awaiting a "go-ahead" message from the receiver. Normally, the sender will retain the transmitted data while waiting, in case the reply is a request for retransmission.

4.2 *Software Multiplexing of Addresses*

Hardware insertion and removal of addresses and rejection of intrusions is attractive for a computer as well as for a simple device–especially in an open-shop, real-time environment where computer users want direct, low-level control of I/O. Hardware addressing is not necessary, however, for program efficiency. Multiplexing can be done in software in several ways, depending on the particular interface. At the worst, it is no more complicated than copying data to or from a transmission buffer headed by address information. This processing is quite modest, compared to the translation, reformatting, packing, and unpacking frequently done for storage and input-output media.

4.3 *An Example Program*

A system utility program written for the loop provides a means to copy data or programs from the disk of one machine to that of the other,

and to perform several other functions. For transmission, the same program is used in both machines. Transmission can be controlled from either end.

The idling sequence of both machines is an attempt to input from the typewriter. The originator of a transaction gets a command from the keyboard and sends it through the network as a "command block," which interrupts the responding computer and takes it out of the idling sequence. The responder appends a coded acceptence or rejection and returns the message. The originator checks the reply and, if there is no error and the responder agrees on the amount, format, and disposal of data, then both computers enter the data sequence. The action proceeds by toggling between sending a block of data and sending back an acknowledgment.

In each computer, attempts to read control replies, data, and acknowledgments are subject to fixed time limits. Failure to receive the message in time is taken as an error. Errors of any kind are reported on the typewriter where the command was originated. Responsibility for requesting status, or restarting transmission, are left to the operator.

The program consists of 400 instructions in assembly language. Approximately 100 of these are ring I/O and error checking, 150 are typewriter I/O and command interpretation, and 150 are communications with the disk and printer, and the sequencing of subroutine calls to implement the commands.

4.4 *User Access to the Loop*

Commands of the loop utility program allow a user's program to be transported to a remote computer, loaded and placed into execution. Subsequently, a special interrupt-command block can cause the remote program to be aborted, dumped onto disk, and the general loop utility restored for continued remote operation. This allows both machines to be operated from one console, even in most program debugging.

There are no restrictions or special disciplines for use of the loop. It can be used in direct access by user programs, either in assembly language with a nine-instruction sequence, or in FORTRAN using existing library subroutines. Two calls define the address of a correspondent and transfer any amount of data up to 200 thousand bits, using data-loop blocks as they become available. For error checking, users may echo all transmissions, use simple checksums or use a burst-resistant multiple-error-detecting subroutine developed for magnetic tape.

Most of our uses of the loop involve the transmission of a program

or block of data 200 thousand bits or less in length. Transmissions are sufficiently infrequent to account for a very low average bit rate.

If necessary to reduce congestion, Kropfl's station can be made to impose a limit on peak transmission rate. All of our uses, thus far, are consistent with a peak rate limit of 50 thousand bits/second. Most transmissions would last only one or two seconds with that constraint.

Even a very demanding requirement–transmission of simple computer-generated motion pictures with our graphics system–can be done within

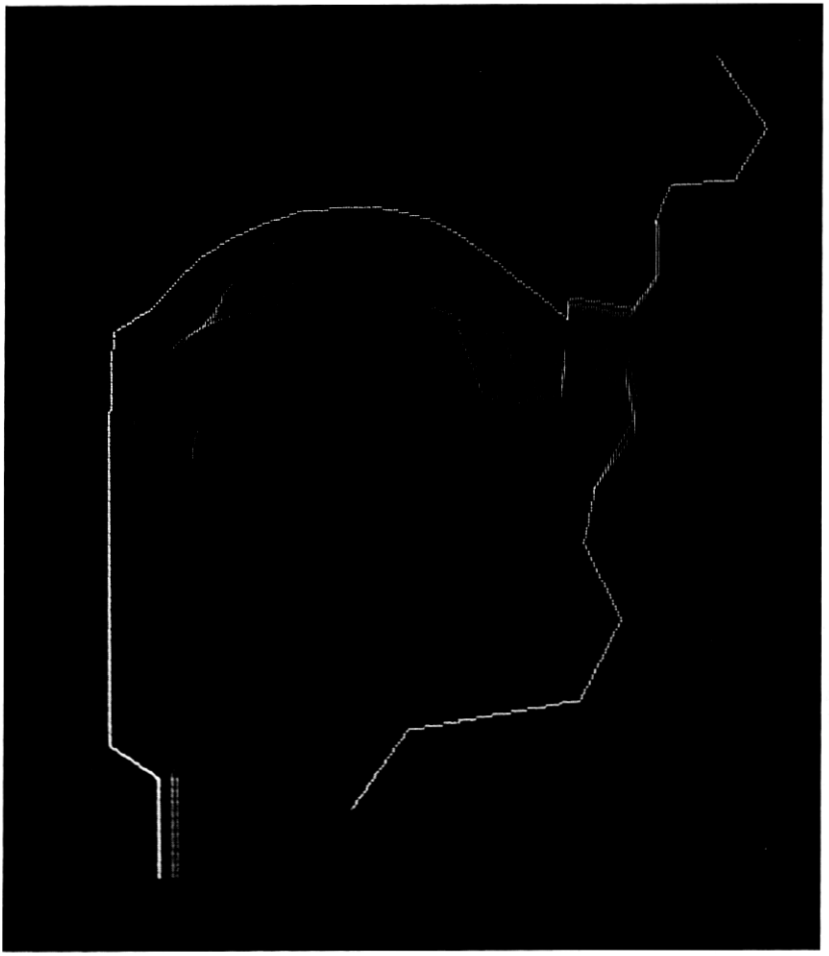

Fig. 3—The loop system supports rapid computer graphics. Using a dot-by-dot transmission code, real-time motion pictures of this complexity can be transmitted at 50 kilobits/second. Using a vector code, they would require only 10 kilobits.

this limit. The example in Fig. 3, drawn as a series of separately controlled dots, requires less than 3000 bits per frame for the part that moves. With a vector scope, it could be drawn with 600 bits per frame–less than 10 kilobits/second, 1/120 of the loop capacity.

V. OTHER APPLICATIONS

The above applications involve transmissions between essentially equal computers. Addressed-block transmission is potentially very useful between unequal correspondents also.

In remote-batch operation of a large computer, an interface from the remote terminal to the addressed-block system is the same as that for voice-grade and leased-line services. With conventional transmission the main computer has a separate modem and interface for each trunk. These are connected to a fairly large special processor whose job is to sort the simultaneously incoming streams of data into separate messages, and present them sequentially to the computer; and to perform an inverse function for output. These operations are inherent properties of addressed-block transmission! Messages are forced "into line" getting onto and passing through the network. They arrive at the computer and leave sequentially, through a single interface.

The availability of low-cost but powerful processors is making on-line computers desirable for every laboratory. But to be most useful, a machine should have access to a variety of expensive but infrequently used peripheral devices. A local data loop will allow a number of small and intermediate computers to share a pool of special equipment. Communication instead of duplication combines the economy and versitility of centralization with the on-line computing power of separate machines.

VI. SUMMARY

We have interfaced two laboratory computers to an experimental addressed-block data transmission system. The project went smoothly; there were no disappointments or surprises. Programming for the system is equally pleasant and uncomplicated. The general-purpose data transmission program for the system was written and debugged in a week. We are presently extending the system to other computers. We see addressed-block transmission as a simple but convenient solution to our computer communications needs.

REFERENCES

1. Pierce, J. R., "Network for Block Switching of Data," B.S.T.J., this issue, pp. 1133–1145.
2. Kropfl, W. J., "An Experimental Data Block Switching System," B.S.T.J., this issue, pp. 1147–1165.