Behavior Research Methods & Instrumentation
1974, Vol. 6, No. 2, 149-154

Hierarchically distributed processing for psychology

CLIFFORD B. GILLMAN, DAVID J. LAPIN, and PAUL B. BUCKLEY*
University of Wisconsin, Madison, Wisconsin 53706

The advantages of loosely coupled hierarchical computer networks for psychological research are discussed. Hardware methods for interconnecting computers are considered, and software communication protocols are detailed. A hierarchy of interconnected computers is formed: (1) large campus computer for statistical analyses, (2) medium scale computer for data concentration and program development, and (3) laboratory computers for data collection. This hierarchy provides efficient computing at all levels and insures maximum computing power at minimum cost.

The laboratory computer has emerged as a powerful flexible tool for psychological research. A natural extension of this tool is the interconnection of laboratory computers to increase this power. The connection of computers into networks has become a popular topic in computer science as well. Invariably, computers are interconnected to share the work of computing efficiently, and the resulting networks can vary in size from two minicomputers to a system as large as the ARPANET. When work is shared between computers, this is called *distributed processing*. The purpose of this paper is to discuss ways in which laboratory computers may be interconnected for communication with each other. It will then be argued that a hierarchical network provides the optimal organization for laboratory computing in psychology. Finally, the operation of two such systems at the University of Wisconsin are described.

There are two basic methods for interconnection of computers. By the first method, the computers share one or more banks of memory. This may be accomplished by direct memory access channels or by multiport memories, but either method is expensive and requires the computers to be in close proximity. Furthermore, the operating systems in each computer must be closely interlocked to insure that memory is shared correctly, i.e., that one computer is not modifying a memory location as it is being read by the other. Operations of one computer depend so closely upon operations of another, that such computers are called *tightly coupled*. The second method of communication between computers requires that each computer act as a peripheral device to the other. There is simply a program module in each computer that handles input-output to the other computer, either by direct memory access or through registers. Such computers are called *loosely coupled*, and this organization places many fewer constraints on the operating systems of each computer. Long distance communication is possible between loosely coupled computers. It is occasionally necessary and desirable to tightly couple two or more computers for sufficient speed and computational power (e.g., the array processor ILIAC IV), but this is seldom the case in psychology.

*Requests for reprints should be sent to Clifford B. Gillman, Waisman Center on Mental Retardation and Human Development, University of Wisconsin, 2605 Marsh Lane, Madison, Wisconsin 53706. This work was supported, in part, by Grant HD-03352 from the National Institutes of Health.

Before discussing hierarchically organized networks of loosely coupled computers, one must first summarize the methods for communicating between a central processor and any peripheral device. Communication is performed in parallel or serially, by synchronous or asynchronous means. When transmitting or receiving a data word or a command word, the bits may be sent simultaneously along separate wires (parallel transmission) or sequentially in fixed order along a single line (serial transmission). These modes of transmission differ in the number of wires required–i.e., cost–and in maximum possible transmission speed. Synchronization reflects the ability of the receiving device to acquire timing information from the transmitting device. The receiver must know when to sample data and the transmitter must know when to change to the next bit. In order to synchronize the operations of transmitter and receiver, timing information must be included as part of the message between them. The transmitter may send a series of "synchronization characters" at the beginning of each message, to signal the beginning of the message and to permit the receiver to determine when to sample. Alternately, a separate timing signal may be transmitted in a single message. The danger of this method is that errors in timing within the message are cumulative. The potential for desynchronization provides potential for desynchronization provides a practical limit to message The potential fo desynchronization provides a practial limit to message length, but when timing circuitry in the receiver and transmitter are accurate and stable, this method provides extremely rapid transmission rates. Asynchronous transmission is accomplished by surrounding each character to be transmitted with a "start bit" and one or two "stop" bits. The start bit acts as a warning signal and synchronization agent for the receiver, which then samples at a previously agreed upon rate. As long as transmitter and receiver agree on the transmission rate, no other synchronization is required and timing errors are noncumulative. This mode of transmission allows data characters to be separated by varying amounts of time, giving rise to the name asynchronous transmission. The number of stop bits is determined by the rate of transmission. Since each character must include two or three extra bits, maximum effective transmission speed is reduced accordingly. Most computers communicate with each other at high data rates in the serial synchronous mode. Teletypes, cathode ray tube (CRT) terminals, and other slow peripheral devices communicate in the serial asynchronous mode. Techniques for loosely coupling networks are not limited by mode or speed of transmission, but merely require an appropriate peripheral handler in each computer to obey the correct communication protocol.

An effective hierarchical network for psychological research connects several laboratory computers to a medium scale computer for communication and data concentration. In turn, the medium scale computer is connected to a large, campus computer for further processing of data. Although several medium scale computers may be connected to a single large computer, and large computers may be connected interregionally, discussion is restricted to the details of a hierarchical network formed by connecting laboratory minicomputers to a single medium scale computer, which is connected to one large computer.

Although medium scale computers are not unusual in psychology, the use of a medium scale computer as the cornerstone of a hierarchical network is not common. More typically, medium scale computers have been used in time-sharing applications to collect data from several experiments simultaneously. Such multitask time-sharing systems share two major disadvantages that are circumvented by a hierarchical system. A hierarchical system like the one we discuss combines the benefits of a dedicated laboratory computer with those of a larger multitask processor, and it is this combination that is important. The most glaring disadvantage of a time-shared computer occurs when that computer fails to operate, at which point all experiments terminate. This inconvenience is avoided by using dedicated minicomputers to collect data, and requiring that each minicomputer be connected to its own data storing device so that data may continue to be collected when the central computer fails. This storage device is usually inexpensive (e.g., tape cassette or Teletype paper tape punch), and is only activated when the central computer fails to acknowledge a message. The stored data may be entered into the system at a later time, so that no data are lost in a hierarchical system. The second disadvantage of a time-shared system is the large software overhead due to the presence of several simultaneous experiments. A complicated operating system is required, but more importantly, programs for control of experiments and data collection must be written to be compatible with a multiprogam environment. One must always consider possible activities of other programs while writing each program. This is not an optimal situation. Programs to collect data and control experiments should be independent of all other experimental programs in the system if possible. The hierarchical system solves this problem by assigning all data collection and control programs to dedicated laboratory minicomputers. Communication with other computers is initiated and completely controlled by the minicomputer, although most of the requisite software resides in the central medium scale computer. Thus, execution of an experiment does not depend on its priority, nor does it depend on which other experiments are in progress.

We are committed to the proposition that laboratory minicomputers provide a powerful flexible cost-effective tool for psychological research in almost any area. The potential for timing and scheduling experimental procedures and collecting analog or digital data is limited only by the size of the minicomputer and programmer skill. The goal of a hierarchical network of computers is to minimize these two constraints. By allocating large numerical calculations and data storage to larger computers, the minicomputer is left to control experiments and collect data. Taking the complicated programming away from the minicomputer often removes constraints on experimental parameters. For example, the necessity to compute complicated algorithms rapidly in the midst of an experiment (e.g., Fourier transformations) may place unacceptable constraints upon the scheduling of experimental events. These constraints may be minimized by passing the data to another more powerful computer and waiting for the answer, especially if the larger computer is very fast and not very busy. The laboratory program is now dependent upon the operations of another computer, but it is often impossible to conduct such experiments at all otherwise. Furthermore, several laboratories may share one or more utility programs in the larger computer, which reduces program development time enormously. The ability to send data, initiate a program in another computer, and continue to work until the result is returned is most useful.

A problem for the minicomputer that stands alone in a laboratory is the necessity for peripherals, especially input-output devices and data storage facilities. The central processor comprises only a small proportion of the cost of a fully equipped laboratory minicomputer. This makes acquisition of a computer based laboratory quite inaccessible to investigators. Furthermore, independent laboratories with full sets of peripheral devices seldom use these devices continually or concurrently, and the net result is unnecessary redundancy. A central medium scale computer can make its full complement of peripheral devices available to each connected laboratory. Programs or data are accepted from minicomputers immediately and are sent to the appropriate device when it is free. The cost of a

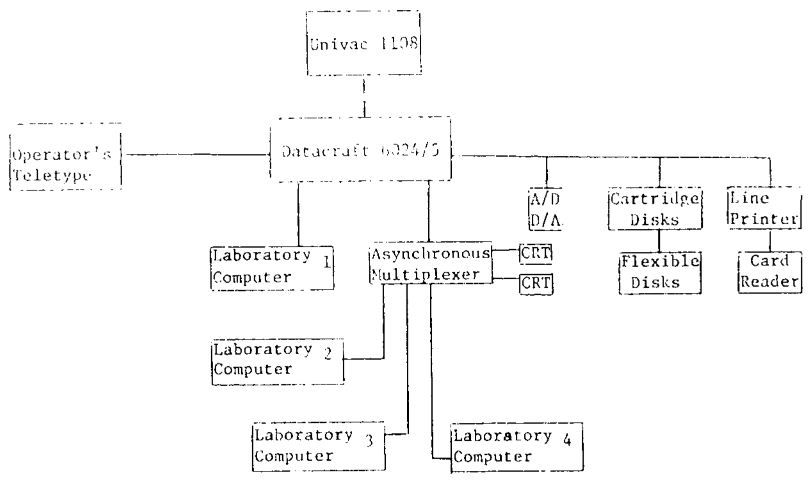

Fig. 1. Hierarchical system at the Waisman Center on Mental Retardation and Human Development.

new peripheral device can be shared by users to allow acquisitions otherwise impossible. New laboratories can be added to the system for the price of the central processor, laboratory interface, and an inexpensive storage devices for emergencies. At current prices, we compute that sharing peripheral devices in a hierarchical system becomes cost effective when three or more computer based laboratories are involved.

There are other benefits to interconnecting minicomputers through a medium scale computer. First, the major part of the communication protocol may be implemented in the medium scale computer to conserve memory in minicomputers, where it is at a premium. Second, the minicomputer may have enough memory to execute sophisticated programs whose source code is too large to be compiled readily in the minicomputer. Program development is usually the most time consuming part of laboratory computing, and using the compiler or assembler requires multiple passes through each program, especially if optimization is performed. Cross-assemblers and cross-compilers available on a medium scale computer can cut program development time in half. Third, well developed higher level languages like FORTRAN are available on medium scale computers, but are used in minicomputers with difficulty. Cross-compilers can be written to optimize for any minicomputer's instruction set. Our PDP-8 cross-assembler is even written in FORTRAN. Thus programmers can use high level languages and a variety of available utility routines to reduce programming effort. Fourth, minicomputers often face a problem in sending data to the large computer, where general statistical programs are available. Most investigators rely on such general statistical packages for detailed data analysis. Data can be transferred anywhere in a hierarchical network, e.g., to the laboratory that has the plotter, but usually data are sent from the laboratory to the medium scale computer for summary and reformatting, and control cards are then added to submit a job to the large campus computer. The medium scale computer acts as a remote job entry port to the large computer and receives the results of the statistical analyses.

We have developed two hierarchically distributed processing systems for use by psychologists at the University of Wisconsin, one in the Psychology Department and one at the Waisman Center on Mental Retardation and Human Development. Both systems are essentially the same, and Fig 1 describes the organization of the Waisman Center network. The medium scale computer is a 32K Datacraft 6024/5 (24-bit word) with paper tape, punched card, and line printing capacity. Cartridge disks are available for general and system use, and flexible "floppy" disks are used for private storage of data and programs by individual investigators. The Psychology Department installation has substituted an industry compatible magnetic tape drive for the flexible disks. Analog data collected off-line on FM tape may be inserted directly into the analog-to-digital conversion facility. Similarly, complex stimulus sequences can be generated digitally and converted to analog form for recording on tape. Communication to the university's Univac 1108 is accomplished through a serial synchronous interface over telephone lines at 4800 bits/sec. Communication to laboratory computers is done in two ways. One PDP-12 is connected in parallel, processor-to-processor, to allow the high data transfer rates required by certain neurophysiological experiments. All other laboratory computers communicate in serial asynchronous mode through an asynchronous multiplexer. This multiplexer can service up to 16 laboratories simultaneously, at data rates of 110, 150, 300, 600, 1200, 2400, 4800, or 9600 bit/sec. Each laboratory may choose its own transmission rate and is connected by cable to the multiplexer. At the maximum data rate, a 4K minicomputer with 12-bit words can transmit or receive its entire memory in approximately 5 sec, which most laboratories find adequate. All serial asynchronous communication obeys the Electronic Industry Association RS-232 standard, so that interfaces are commercially available for most minicomputers, and almost any CRT may be plugged

Table 1

Type	Octal	ASCII Char	Meaning
1	01	SOH	Start of File (SOF)
3	02	STX	Start of JOB (SOJ)
3	04	EOT	End of Transmission
3	06	ACK	Acknowledge
2	20	DC0	Data–Form 0 (text string)
2	21	DC1	–Form 1 (single precision logical)
2	22	DC2	–Form 2 (double precision logical)
2	23	DC3	–Form 3 (floating point–three word)
2	24	DC4	–Form 4 (floating point–FORTRAN)
3	25	NAK	Negative Acknowledge
3	26	SYN	Ready
1	27	ETB	Delete File if it exists and start new one (DFL)
3	34	FS	End of File (EOF)
1	32	SUB	Initiate Program (SOP)
1	31	EM	Send File (SEND)

Type 1–Followed by six frames of file name
Type 2–See data formats below
Type 3–No message characters

into the multiplexer. We have found the CRT very useful for program development and file editing. These CRT terminals are also connected to the Univac 1108 computer through acoustic couplers (modems) for interactive programming. Graphics capability and plotting are available through the Univac computer.

The Datacraft operating system is DMS III, a multitask foreground background system well suited for real time applications. All communication between computers is accomplished by foreground programs, while background is reserved for progam development and data analysis, either in the batch mode or interactively through the terminal handler ACRONIM. Communications between the Datacraft and Univac computers obey the Univac 1004 symbiont for remote job entry stations. This communication protocol is available in several Univac technical documents. Software design for communication between the Datacraft and laboratory computers began with this protocol, but the current version bears little resemblance to the original. Laboratory communication protocols are described in detail to aid investigators who may design similar systems. These protocols are independent of the communications interface or of the brand of computer, and have been designed for both 8-bit and 12-bit interfaces. Control characters were chosen from the standard ASCII set so that a CRT terminal may be substituted anywhere in the communication system as an ACRONIM terminal or for communication program debugging. This means that the Teletype connected to most minicomputers can be easily used as a Datacraft terminal as needed. Dialogue between a satellite laboratory computer and the Datacraft computer is *always* initiated and controlled by the laboratory.

MESSAGE FORMAT

Each message between the Datacraft and the Satellite computer consists of three parts: a function code which indicates how to treat the remainder of the message, the message itself, and a terminator (which may or may not contain a checksum depending on the function code).

The function code is the first frame of the message. No matter how large the frame, only the low order 7 bits are used for the function code. Of these 7 bits, the high order bit (Bit 6 in Datacraft notation) indicates the presence or absence of a checksum frame and the remaining six are interpreted as indicated in Table 1.

OPERATIONAL DIALOGUE

Operationally, the dialogue works as follows (refer to Fig. 2):

(1) Each message is ACKed when the computer is prepared to accept the next message.

(2) Any message which has a checksum and is improperly received is NAKed. It is then resent. No other messages are NAKed.

(3) Any time a message makes no sense, an EOT is sent. This is the only message that is not ACKed. Instead, an EOT is sent in response. Only after consecutive EOTs are sent in each direction, the dialogue reverts to its *initial state.*

(4) In the *initial state*, the only acceptable message is the READY message from the satellite to the Datacraft. This is ACKed and the dialogue enters the *idle state.*

(5) in the *idle state*, the SEND, SOJ, SOF, DLF, or SOP messages may be sent from the satellite to the Datacraft. In the first case, the dialogue immediately enters the *send state.* In the last case, the dialogue immediately enters the *program state.* In the remaining cases, the message is ACKed and the dialogue enters the *receive state.*

(6) In the *send state*, an SOF message is sent from the Datacraft to the satellite. This is ACKed by the satellite. The Datacraft then sends DATA messages (which are ACKed by the satellite) until the end of the file being sent. The Datacraft sends an EOF message which is ACKed, and the dialogue reverts to the *idle state.*

Table 2
Data Formats

8-Bit Interface	12-Bit Interface
Text Type	
Three frames per Datacraft word	Three frames (truncated to 8 bits per Datacraft word)
Single Precision Type	
Two frames (low order 6 bits each) per Datacraft word, placed in the low order portion and zero filled	One frame per Datacraft word, placed in the low order portion and zero filled
Double Precision Type	
Four frames (low order 6 bits each) per Datacraft word	Two frames per Datacraft word
Three-Word Floating Point Type	
Six frames (low order 6 bits each) per Datacraft double precision real number (two words)	Three frames per Datacraft double precision real number (two words)

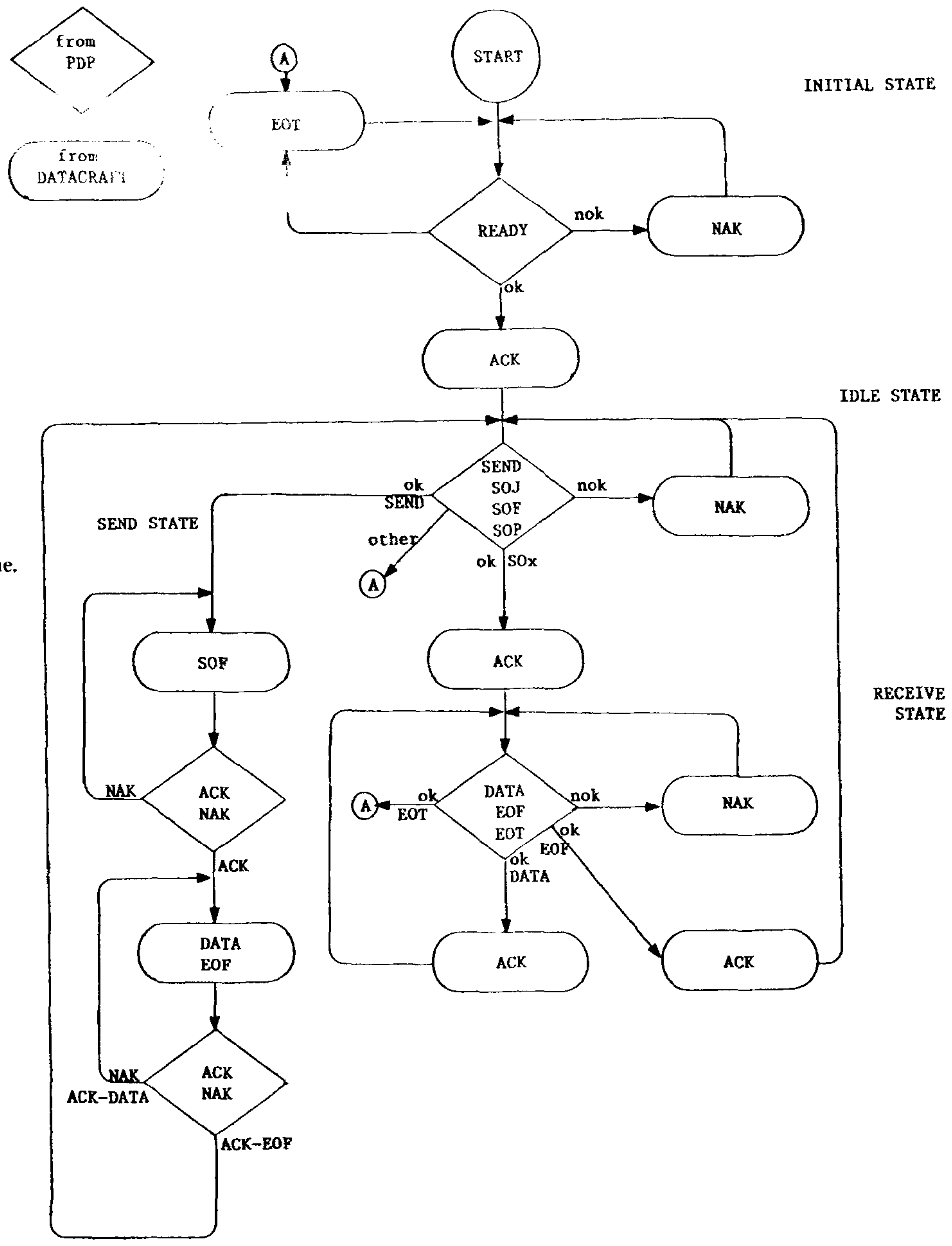

Fig. 2. Operational dialogue.

(7) In the *program state*, whenever the program reads an LUN assigned to the satellite computer, the Datacraft sends an ACK message and accepts a DATA, EOF, or EOT message. An EOT message aborts the program with Abort Code 37 (terminal abort). An EOF message sets the appropriate status bit. The LUN remains busy until an acceptable (ACK able) message is received by the Datacraft. Whenever the program writes an LUN assigned to the satellite, the DATA or EOF message is sent to the satellite. The LUN remains busy until the ACK is received. If the program is exited in any way, the dialogue reverts to the *idle state*. If the program is aborted, the EOT message is sent.

(8) In the *receive state*, the satellite sends DATA, EOF, or EOT messages. An EOF message causes an EOF to be written to the file. The message is ACKed and the dialogue reverts to the *idle state*. The EOT message causes the file to be deleted and an EOT message to be sent. DATA messages are written into the file and ACKed.

Services available to an individual laboratory include the ability to (1) load any program or data file, (2) store data or programs on disk or magnetic tape, (3) send a data file to the campus computer and initiate statistical analyses, and (4) initiate computations on larger computers. Implementing the entire protocol for a laboratory computer (including ACRONIM terminal capacity) requires roughly 2K of memory for a PDP-12 computer. However, this communication software must be resident in memory only when communication is desired, e.g., for data storage. A version which only allows ASCII and single precision binary transmission requires only 650_{10} memory locations in a PDP-8/e computer. An even more abbreviated version for transmission of 18-bit binary words is currently being designed and should require even less minicomputer memory. Each laboratory weighs the capabilities needed against the memory requirements to determine which subset of the protocol to be programmed. Those parts of the protocol not implemented in the minicomputer are simply ignored and do not affect the system.

In summary, we have implemented a hierarchically distributed system, in which laboratory computers may communicate with each other and with two larger computers. Real-time experimental control is performed in the laboratory by minicomputers whose timing is independent of the other elements of the system. The laboratory may call upon other resources as needed. When communicating with the network, the laboratory computer has powerful language processors, editors, file management, cross-assemblers, and statistical programs at his disposal, and essentially unlimited data storage is available. A medium scale computer sits at the center of a star shaped array, responding to communication requests in the foreground and performing program development and data analyses in the background. Data are concentrated at this point for transmission to a larger computer when necessary. This hierarchical organization allows full computing power to each laboratory, yet requires only a minimum minicomputer configuration in the laboratory. Sophisticated programming remains in the domain of professional programmers, the laboratory computer is devoted to what it does best, and computer based laboratories become more readily available to psychologists.