CONF-811040--102

AN AUTOMATIC DATA-ACQUISITION AND COMMUNICATIONS COMPUTER NETWORK FOR FUSION EXPERIMENTS*

C. O. Kemper
Computer Sciences at
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37830

A network of more than twenty computers serves the data acquisition, archiving, and analysis requirements of the ISX, EBT, and beam-line test facilities at the Fusion Energy Division of Oak Ridge National Laboratory. The network includes PDP-8, PDP-12, PDP-11, PDP-10, and Interdata 8-32 processors, and is unified by a variety of high-speed serial and parallel communications channels. While some processors are dedicated to experimental data acquisition, and others are dedicated to later analysis and theoretical work, many processors perform a combination of acquisition, real-time analysis and display, and archiving and communications functions. A network software system has been developed which runs in each processor and automatically transports data files from point of acquisition to point or points of analysis, display, and storage, providing conversion and formatting functions as required. The network software simultaneously serves investigators and programmers by providing the transport of data files, system control and command files, programs, and hard-copy output from any system to any other system in response to a simple command entered on any terminal on any processor in the network. The network also includes an interface to the large central site computers at the Laboratory and an interface to the National Magnetic Fusion Energy network. This paper will briefly describe the components and procedures which enable the network to function.

Figure 1 presents an overview of the network topology at the Fusion Energy Division site as it stands in 1981. Each small circle represents a processor or node, and the three-character mnemonic within each circle is the name by which the node is known to both the users and the network software. For example, MET is the computer at the Medium Energy Test Facility, while INB, ICL, ISA, ISB, and ISS all serve the Impurities Study Experiment tokamak project. All nodes which appear within the large surrounding circle are PDP-11 family processors, manufactured by Digital Equipment Corporation, operating under the manufacturer's RSX11M or IAS real-time executives, and each contains and executes a copy of a common communications software package developed and maintained on the RSX node. Only the RSX node is primarily dedicated to communications; all of the other PDP-11's are primarily data-acquisition systems, with some secondary usage for program development. Consequently, the network software has been designed to impose only minimum demands on a node's processor time and memory, and requires no operating system modifications. In order to further reduce processor loading, most communication lines used are high-speed serial, sychronous lines managed by microprocessor interfaces (DMC-11's).

Those nodes which appear outside the large circle in Figure 1 do not use the common network package, but instead interface to the network with software specifically designed for each processor and its applications. 840, for example, is a PDP-11 timesharing system (RSTS/E) with a parallel interface to the network, while END is a Nuclear Data data acquisition system interfaced by CAMAC hardware. CEX is a PDP-8, and 8-32 is a Perkin-Elmer 8-32 processor, both used for ISX data acquisition, and both interfaced by serial, asynchronous lines. FED and X-10 are both large-scale, multiprocessor PDP-10 timesharing systems, which support most data-base and theoretical applications. The FED system shares the Fusion Energy computer center with the RSX communications system, to which it is linked by a memory port (DLIO). A 9.6 KB synchronous leased-line connects RSX to X-10, some seven miles away at the Laboratory's central computing facility, which also includes several large IBM systems, to which jobs may be submitted from either PDP-10. FED is also a node in the National Magnetic Fusion Energy network, permitting FED users to access that nation-wide system of large centers.

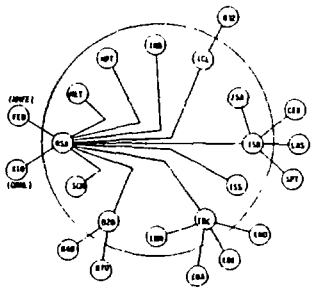

Fig. 1. FED/ORNL Network in 1981

Regardless of the processor or its operating system, or its position in or connection to the network, the communications system presents a single, constant interface to the user or to a user program. This interface was designed to implement system-to-system dataset transfer, rather than remote file or device access. Consequently, it is specified as a transfer command of the form DESTINATION=SOURCE, as shown in the example in Figure 2. The destination or output side, and the source or input side of the transfer command each consist of a three-character node name, followed by an underscore character, followed by a dataset specification in whatever syntax is required by that particular node. In the example given, a file is to be transferred from the ISB PDP-11 to the FED PDP-10, so the source side specification of device, user identification, and file name, type and version follow the rules of FILES-11 syntax, while the destination side specification of device, file name and extension, and the project-programmer number are given in TOPS-10 syntax. The communications software never parses either dataset specification on intermediate nodes, but only determines the source and destination node names for purposes of routing. The source specification is parsed only on the source node where the dataset is to be accessed, and the destination specification is parsed only on the destination node where

*Research sponsored by the Office of Fusion Energy, U.S. Department of Energy, under contract W-7405-eng-26 with Union Carbide Corporation.

the new copy is to be created. In all cases, the parsing logic used is that provided by the host operating system, so that changes or extensions to a file system syntax are available to a network user, but do not require changes in the communications system. In addition, all routine decisions by the network depend on whether or not the command is accompanied by data; if it is not, the command is routed to the source node, while if data accompanies the command, the entire block is routed to the destination. Consequently, it does not matter where in the network a transfer command originates; the transfer will be accomplished as specified. It also does not matter to the network software whether a command originates at a user terminal, an operator console, or in some executing program, such as a data acquisition or process display task. Host operating system facilities for inter-program communication are used to pass a program-generated command to the network in exactly the same form as a user would type it, so that the network software and application programs are each protected from changes in the other.

In addition to specifying the *movements* of datasets from system to system, the transfer command also specifies the reformatting or conversion of data required by differences between the source and destination processors. Source code, for example, is stored on a PDP-10 as 7-bit ASCII, five bytes per 36-bit memory word, with imbedded carriage control characters, while a PDP-11 stores source code as 8-bit ASCII, two bytes per 16-bit word, with carriage control implied by the record structure. In the absence of user-specified switches appended to the transfer command, the network software assumes that a data file is ASCII source, and will reformat the file in transit from the source format to the destination format if those processors are of different types. The switches by which a user may override or qualify such conversions are listed in Figure 2, together with descriptions of their effect. The binary (/BI) and halfword (/TH and /EH) switches are of particular interest. The binary switch suppresses all conversion, forcing the destination copy to a bit-for-bit image of the source dataset, regardless of word or byte boundaries. This feature allows both transport and backup of load images, object files, and other nonsource datasets between dissimilar computers. The truncate and extend halfword switches are commonly used in the interchange of archived experimental data between 16-bit processors and the PDP-10's, allowing the users' analytic programs easy access to the data in either machine environment.

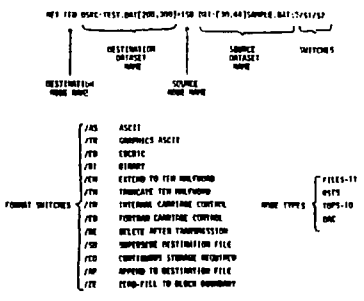

Fig. 2. Network Transfer Command

As noted above, all PDP-11's in the network execute a copy of a common communications software package, developed and maintained on the RSX node. This package performs most of the routing and conversion functions of the network, while the non-PDP-11 machines contain only sufficient software to interface the network to the users and file system on that processor. Figure 3 presents an overview of the functional structure of the communications package as it appears in any PDP-11 node. In broadest terms, the package consists of a receiver task, a spooler, and a sender task. The receiver task buffers a communications link to the file system, placing each incoming dataset in a queue file on the system's spooling disk. The spooler task processes each queue entry, making routine decisions and converting data if necessary, based upon the transfer command in the entry. The result is a transmission queue file on the spooling disk, which the appropriate link sender task will in turn buffer out onto another communications link. Note that each incoming command and accompanying dataset are accumulated in their entirety on disk by the receiver before being accessed by the spooler, which in turn processes the dataset completely before transmission can begin. This store-and-forward procedure was chosen for its simplicity in development and its high reliability in operation. The inherent delays in store-and-forward processing have been partially compensated by the use of an efficient file system, very fast disks, and high-speed communications lines, and additionally compensated by very careful software design to eliminate both in-core movement and disk-to-disk copying of data. The host real-time, multi-tasking operating systems also provide a natural means of multiplexing all the communications links on a given node; each link receiver places incoming datasets in files named after the adjacent node from which the dataset came, and since the spooler is installed in the operating system under each possible adjacent node name, many copies of the spooler can execute simultaneously but asynchronously. Each spooler in turn places its output in files named after the adjacent nodes to which the datasets are bound, so that the link senders are also multiplexed. The only interaction between software components serving different communications links is in competition for processor time, memory, and I/O cycles, and the priority-driven structure of the operating system readily allows operator regulation of the competition to maximize any chosen throughput characteristic.

The network includes several types of communications links, using several different protocols. A number of drivers, protocol modules, and sender/receiver pairs have been implemented to support these various links, but all of those modules are multi-functional and interchangeable. The drivers are multi-unit drivers, the protocol modules are shareable, reentrant codes, and the senders and receivers can be installed and executed under multiple names. All of them present the same interface to the spooler so that links may be added, removed, or have hardware or protocols modified, without impact on the spooler or on other links. In addition, the user interface program, NET, appears to the spooler as a receiver from the host node, and a complementary task appears to the spooler as a sender to the host node. This latter task handles datasets from the network bound for the host node, including user storage, unit-record peripherals, terminals, etc.

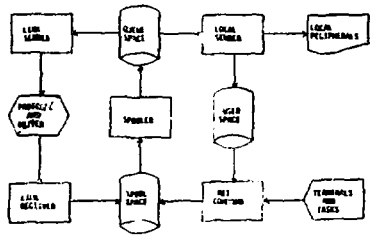

Fig. 3. Network Node Data Flow Outline

Figure 4 presents an example of the network's handling of a simple file transfer request. A user on node AAA wishes to move the file SRCDAT on node BBB to node CCC. He enters on a terminal on AAA the transfer command "CCC_DSTDAT=BBB_SRCDAT." The NET task places the command in a file AAA.SPL and requests the execution of the spooler under the name AAASPL. The spooler determines that the command must be moved to the source node BBB, and renames the file containing the command to BBB.QUE, since the network routing tables indicate the existence of a direct link to BBB. (If no direct link existed, the spooler would use instead of BBB the name of a node connected to AAA but closer to BBB.) The BBBSND link sender then passes the file across the link to AAARCV, the link receiver executing in BBB, which results in the command file being reproduced under the name AAA.SPL on node BBB. The spooler again executes under the name AAASPL, but this time determines that the host node is the user's source. Consequently, the spooler accesses the user's data-file SRCDAT and appends the data to the command, performing any required conversion at the same time. The entire output is then placed in file CCC.QUE for transmission to the destination node. (Again, there is a possibility of indirect routing.) After the file is reproduced by BBBRCV on node CCC as BBB.SPL, it is again encountered by the spooler, executing now under the name BBBSPL, which determines that the file has arrived at the user's destination node and consequently renames the file to CCC.QUE. The local node sender task will then create the user's destination file DSTDAT and move the user data to that file. The transfer is then complete.

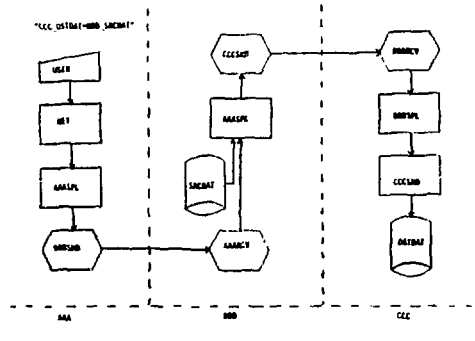

Fig. 4. Network File Transfer Example

Finally, Figure 5 illustrates the use of the network for data acquisition and archiving by the ISX experiment. The tokamak is monitored by software in the ISB PDP-11/34 and three PDP-8's, CEX, LAS, and SPT, using a large and changing variety of CAMAC interfaces. As each shot occurs, the experimental data are accumulated in both interface and main memories. The interface programs on ISB then begin processing their data out to archive files on disk, while the PDP-8's begin transmitting across serial, asynchronous 9.6 KB lines to ISB. The incoming data files are received, checked by the spooler, and stored by the local sender as archive files on disk. The archive disk is dual-ported, so that a second PDP-11/34, named ISA, can provide a wide range of displays and graphics for the control room personnel, while at the same time ISB both continues to acquire data and transmit the acquired files to the FED PDP-10. The communications link from ISB to RSX is a serial, synchronous, 1 MB line, with protocol managed by microprocessors, while the link from RSX to FED is a parallel memory-access port (DLIO). The shot files are archived at FED as they arrive, both to provide additional control room displays and to remain on-line for subsequent theoretical analysis.

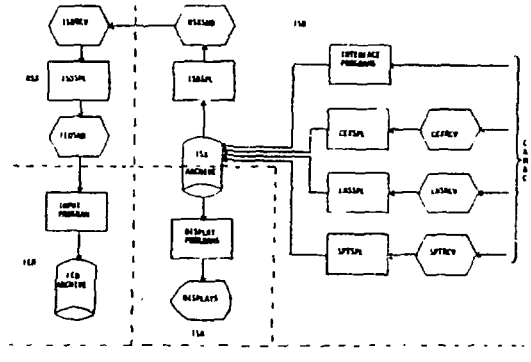

Fig. 5. Data Acquisition Example

The network thus summarized has proven itself a reliable and effective research tool, but it is not a static system, nor is it the product of a single design and implementation effort. The network has evolved over some eight years from a single PDP-11, supporting card reader, line printer, and one link, and due to a changing mix of experiments and user needs, both hardware and software are frequently added, removed, and modified. The future direction of this evolution will probably see the gradual inclusion of more, and more powerful, microprocessors, together with increasing reliance on commercial networking and operating system software for network integration and load distribution.