CONF-830577--4

PARTICIPATION OF VAX VMS COMPUTERS IN IBM FILE-TRANSFER NETWORKS

Richard C. Raffenetti
Argonne National Laboratory
Argonne, Illinois 60439

CONF-830577--4

DE83 017945

DISCLAIMER

DISCLAIMER

The following pages are an exact representation of what is in the original document folder.

PARTICIPATION OF VAX VMS COMPUTERS IN IBM FILE TRANSFER NETWORKS

Richard C. Raffenetti
Argonne National Laboratory
Argonne, Illinois 60439

ABSTRACT

Communications software written at Argonne National Laboratory enables VAX VMS computer systems to participate as end nodes in a standard IBM file-transfer network. The software, which emulates the IBM Network Job Entry (NJE) protocol, has been in use at Argonne for over two years, and is in use at other installations. The basic NJE services include transfer of "print" and "punch" files, job submittal, execution of remote commands, and transmission of user-to-user messages. The transmit services are asynchronous to the user's VMS session and received files are automatically routed to a designated user directory. Access to files is validated according to the VMS protection mechanism. New features which were added recently include application level software to transfer general, sequential files and to bridge the "electronic mail" systems of VMS and VM/CMS. This paper will review the NJE emulator and describe the design and implementation of the sequential file transfer service. The performance of the emulator will be described. Another paper at this symposium will describe the mail bridge.

INTRODUCTION

This paper is a second report (1) on the features and capabilities of software written at Argonne National Laboratory which permits a VAX VMS system to participate as a peer member of an IBM file transfer network, known as NJE. The software was originally written to give VAX VMS users convenient access to the significant resources in Argonne's central complex of IBM computers. With five VAX systems now participating and others planned, the communications facility has proved valuable for access to resources everywhere in the network. For example, one of the VAX systems is a member of a nationwide DECnet-based network.

In this report we are describing new features of the software. We have studied and improved its performance and have added the capability to transfer general, sequential files. Support features and new applications or services are now in place. We describe briefly the user applications which operate in conjunction with the NJE networking service.

It is our goal to make the NJE emulator software general enough to communicate successfully with all of the systems which support the protocol providing there have been no site-dependent changes to the IBM software. We have made the software widely available by putting it into the public domain. The channels for distribution are described.

IBM FILE TRANSFER NETWORKS

The IBM file transfer network which we refer to as NJE (for Network Job Entry) is implemented in standard 370, 30xx, and 43xx operating systems (2). For example, in the major IBM operating systems the support is in JES2 or JES3 (for MVS systems) and RSCS (for VM systems). (The older IBM operating environments using HASP and ASP also support NJE networking.) Other IBM file transfer protocols are used in some but not all of the IBM systems and therefore are less universal.

The NJE protocol is in contrast to the Remote Job Entry (RJE) protocol which forms the basis for other VAX to IBM communication alternatives (e.g. HASP workstation, 2780/3780). The RJE protocol creates a remote system viewed not as a peer of the IBM host but simply as a workstation having a printer and card reader, and sometimes a card punch and/or a console. All computers which participate in an NJE network are peers so far as they support the full NJE protocol.

The NJE protocol transfers "objects" in the classes of SYSOUT (print or punch), job, or console command. The standard print object consists of a sequence of records limited to 133 characters in length; the standard punch object has records limited to 80 characters. Jobs are similar to punch and may be sent to the input queue of a remote host. The output from jobs are routed to the job origin or optionally to other network destinations. The console commands are short messages and their transmission is immediate. That is, a console command is interleaved into the flow of a SYSOUT or job object.

Information which does not conform to these object categories is communicated by transforming it to a punch object before it is entered onto the network and then performing the inverse transformation after it is received. The programs which do this kind of transformation are higher-level services which are independent of the NJE networking services.

The network transmits objects by a store-and-forward mechanism. If transmission is through an intervening node, a SYSOUT or job object is completely transferred to the intermediate node, and then to the destination node. After each intermediate transfer, the receiving node assumes responsibility for the object, and relieves the previous sending node of responsibility. Messages or commands are not stored but are forwarded immediately.

The NJE networking protocol is based on a BISYNC line protocol and provides a high degree of reliability. Cyclic-redundancy-counts are computed to guard against errors in each BISYNC record, and each record contains a sequence byte to ensure that records are not lost. The protocol is half-duplex so that each record is acknowledged in turn. Each BISYNC record can contain both the data moving in one direction and the response regarding the data moving in the opposite direction (piggybacking). Finally, NJE protocols include data byte compression which permits increased effective transfer rates in the case where a file has identical data bytes in sequence.

THE VAX VMS NJE PROTOCOL EMULATOR

The VAX VMS implementation of a protocol emulator for NJE communication only permits the VMS system to be an end node in an NJE network. By being an end node the VMS system is freed from having to temporarily store files and from having to maintain knowledge about the network topology. At Argonne the network is arranged in a star topology with a large IBM batch system as the central node. Five VAX VMS computers located around the laboratory connect directly to the batch node, ANLOS. In addition three VM/370 systems which support many interactive users are connected to the same batch node. Two of the VM systems are IBM computers not located at Argonne. Figure 1 contains a schematic diagram of the topology. Except for the non-Argonne (OFF-SITE) nodes, the names are the identifiers by which the various nodes are known. The operating systems are noted for each node.

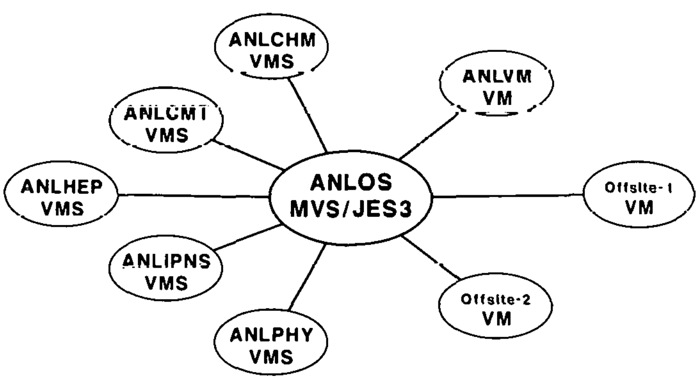

Figure 1: Argonne's NJE Network Topology

NJE Emulator Software

Important features of the NJE protocol emulator are its asynchronous operation, the automatic routing of information to users, and adherance to the file system protection scheme. Figure 2 is a schematic description of the emulator organization. In the figure, the ovals represent processes and the rectangles represent system mailboxes. NJE-SERVER is an image which requires many privileges and runs in its own detached process. It runs continuously, managing the line traffic to and from the communication device, the flow of commands from users, messages to users, and files to and from user file directories. Six tasks, known as line-driver, file-in, file-out, message-in, message-out, and broadcast, are dispatched by the main program in order of priority when there is work for a task module to perform. The line-driver task is highest priority and the other tasks are lower, the order being the same as the above sequence. Event flags as well as a program status word post the need for and the completion of the work of each task. NJE-USER is an installed, privileged image which is executed in a user's process. This image is executed by command procedures which validate the user commands. NJE-USER passes the NJE commands to the server process through the transmit mailbox after validating access to any files which have been named for transmission. Access to the transmit mailbox is limited by requiring privilege so that unauthorized commands may not be written to the communication server. The image NJE-RECEIVER runs in its own detached process and receives information from the server process by reading messages passed through the receive mailbox which also allows only system access. The receiver is a new feature of the emulator which currently provides a bridge between the IBM and VAX VMS mail or note facilities (see below). The receiver provides a mechanism whereby determinate transformations may be carried out automatically on received information.

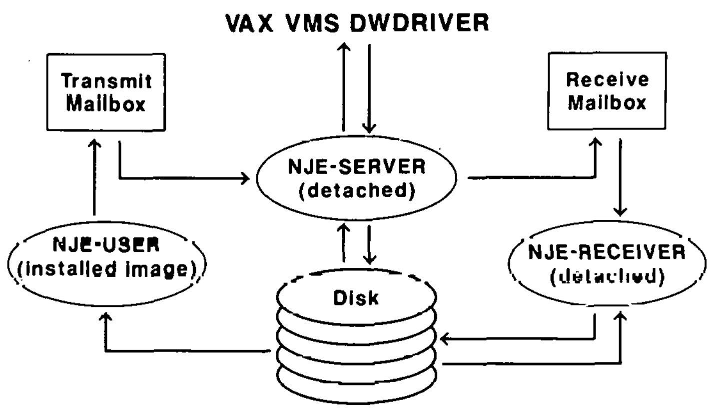

Figure 2: Schematic Organization of the Emulator Software

The server process handles the six major tasks as described above. When a SYSOUT object is received, the server process looks up the destination in a table provided by the system manager. Each VAX VMS user who will use the NJE services has an entry in the table. If the destination is a user, the object is written to a designated directory. If the object is a console command it may be regarded as a user message or as a node-oriented command. Messages are written to the designated user's terminal with the broadcast system service. The NJE emulator validates and executes commands and sends the response as a message to the origin. The valid commands are the internal NJE management commands which report data to tell how the NJE programs are functioning.

NJE Emulator Hardware

The hardware used to connect to the IBM system consists of the DUP-11 unibus device, a pair of synchronous modems, and a dedicated telephone circuit. At Argonne, the connection to the IBM computer is through an IBM 3705 telecommunications processor "front end." Figure 3 is a schematic diagram of the hardware. The maximum data rate of a DUP-11 in this configuration is 9600 baud and at Argonne all of the VAX VMS NJE connections are run at that rate. The data rate is governed by the modems. On a system with a busy unibus, the bus request level for the DUP-11 may have to be elevated if there are frequent errors in the reception or transmission of records.

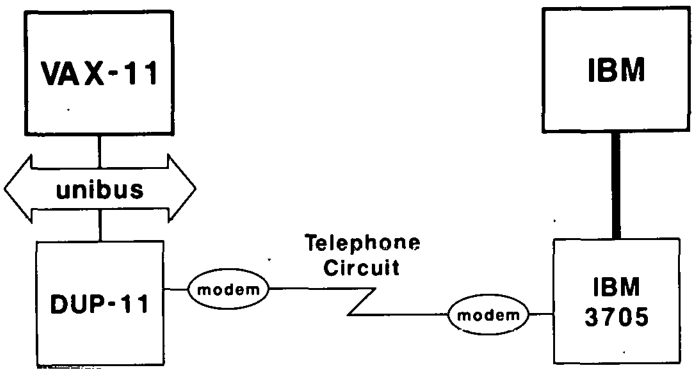

Figure 3: Schematic Diagram of the Hardware

NJE EMULATOR PERFORMANCE

There are two parts to the overhead incurred by the operation of the emulator. The first is the use of the central processor by the emulator software. The second is the processing carried out by the system to handle input to and output from the synchronous interface.

Central Processor Overhead

Since the original software was put into service, the central processor performance has been measured and the server code modified to lower the overhead incurred by the original version (3). A program for performance measurement and evaluation, which was obtained from the DECUS VAX SIG tape (4) is the tool which was used to determine which sections of the server software use large fractions of the processing. The modifications lowered the overhead by a factor of about five. The central processor overhead of the current implementation in an acceptable range for the existing systems.

The emulator software performance data presented here is representative and applies to the transmission of well-defined, sample data files. The data was collected from a VAX-11/780 system during mid-afternoon on a weekday. The system had about 20 interactive users and a total of about 50 processes. The communications circuit was clocked by modems at 9600 bits per second. In the standard implementation at Argonne, the server process is run at a base priority of five to ensure that it obtains all of the central processing share that it needs.

Table 1 displays representative data obtained from the current version of the NJE emulator when a file of non-compressible character data is sent as a stream of 80-byte records. (A sequence of bytes can be compressed if two or more blanks or three or more identical non-blank characters follow one after the other.) In Tables 1 and 2 the symbol CP stands for the central processor time used by the NJE server during the transmission of 4096 80-byte records. The symbol EL stands for the elapsed time. The RATE is the effective transmission rate which is computed from the data size (2.5 Mbits) and the elapsed time. The %CP is the percentage of elapsed time which the value of CP represents. The CP overhead is the number of central processor seconds it takes to prepare and transmit or to receive a megabit (2**20 bits) of data.

Table 1: Performance for Transmission of Noncompressible Data

	Transmit	Receive
CP (sec.)	22.4	16.0
EL (sec.)	375	408
RATE (bits/sec)	6991	6425
%CP	6.0	3.9
CP Overhead (sec./Mbit)	9.0	6.4

Table 2 displays the same data for transmission of 4096 80-byte records each of which is entirely compressible (all bytes of a record are the same). The %CP is quite high in this case but the overhead is lower than for non-compressible transfers. It would be unusual for users to have very highly compressible data. In text files, blanks are the dominant repeated character, while in binary data, the zero is common. We have observed a 15% compression in general, compiled program listings.

Table 2: Performance for Transmission of Compressible Data

	Transmit	Receive
CP (sec.)	16.9	12.7
EL (sec.)	55	64
RATE (bits/sec)	47662	40960
%CP	30.7	19.8
CP Overhead (sec./Mbit)	6.8	5.1

When the server is idle, very short messages are exchanged by the hosts at a rate of one exchange each two seconds. The central processor utilization during the time when no files are being transmitted is negligible.

IO Processor Overhead

The IO processing is a second major factor in the evaluation of the NJE emulator performance. There are two aspects related to the device. The DUP-11 was chosen largely by default. It is a supported VMS device and the driver is capable of carrying out a general binary synchronous communication. Because the DUP-11 is not a DMA device, then during file transmission there is significant interrupt processing. The interrupt load on the system is shown by the monitor utility to be in the range of 5%. Our development plans include substitution of another synchronous DMA interface for the DUP-11.

To emulate the NJE protocol, the binary capability of the DUP-11 driver is needed. With that interface, the VMS driver does not recognize the end of a BISYNC record. A receive buffer is allocated for the QIO read and the request completes only when the buffer has filled, meaning that following the data bytes the buffer will contain pad characters (hex FF) in accord with the line's mark state. The NJE emulator provides for this situation by posting small or large receive buffers depending on whether the communication is idle or not. However, during transmit or receive processing the time during which there is no productive data transfer is in the range of 15-20% of the total line time. A more intelligent driver for the DUP-11 would boost the effective transfer rates for non-compressible data to around 8000 baud for both transmit and receive processing. More effective use of the line capacity will raise the %CP time used by a similar amount while the central processor overhead should remain the same.

THE USER INTERFACE

The user interface (5,6) to the NJE networking facilities is described by command procedures. The command procedures serve three purposes. First, the commands require tailoring to the specific networking environment and therefore it is useful to keep that information apart from the compiled code. Second, many of the commands are named and oriented toward the batch IBM system and permit users to submit and control batch jobs as well as route output to central high-speed printers. Third, by using a command procedure it is possible to do command validation processing and give the user quick feedback if the command cannot be acted upon. The commands which have been implemented are fairly robust and prompt the user for missing or incorrect parameters. Comprehensive help documents have been added to the system help facility to describe all of the commands, their parameters, and their modifiers.

Because the network is IBM-oriented, the control information as well as the data consists of 8-bit EBCDIC characters and translation from or to VAX ASCII is carried out by the emulator software whenever the data is textual. Transfer of print or punch record streams is assumed to require translation; transfer of general sequential files requires user direction as to whether or not the translation is done. The character translation is directed by tables which were devised at Argonne but these can easily be changed.

TRANSFER OF SEQUENTIAL FILES

There are two aspects to the transfer of data to different computers in a network. These are transmission and data conversion. In the following we describe the transmission of sequential files, leaving the data conversion topic for a later section.

We had provided special programs to transform graphics metafiles to and from 80-byte records which could be transmitted in the NJE network. However, there was a need for transfer of more general files. A plan was written for the development of a similar set of programs which would transfer general sequential files. The goal in the implementation was that Fortran programmers could move their files between systems and do productive work on either system without needing to gain special knowledge of the respective file systems. Argonne National Laboratory is a scientific establishment and the Fortran language is a major tool of DEC and IBM users both. The files which are manipulated in and around Fortran programming efforts were held to be of greatest importance in the realization of a file transfer system.

File Transfer Protocol

An IBM program, called Bulk Data Transfer, IUP #5796-PKK (7,8), was in use to transfer data files between the VM/CMS interactive users and the Argonne MVS/JES3 batch system. The plan was to emulate the protocol of the Bulk Data Transfer programs. A useful side effect of this choice is that, like the NJE protocol emulator itself, no changes to standard IBM software are necessary for successful communication and file transfer to VAX VMS systems. (The IBM version of the programs has been extended locally to permit transfer of logical records exceeding 32767 bytes in length, which was not supported. Another extension allowed the program to be run under VM/370. These extensions are fully compatible.)

In the bulk data transfer protocol, a header record contains IBM OS data control block (DCB) parameters of the file and a trailer record contains information such as record count and date by which successful transfer can be confirmed. The data records of the file are converted to IBM's variable, blocked, spanned record format (RECFM=VBS). The block size is 80 bytes, permitting transfer in the NJE network under the category of punch. Transfer of standard VAX VMS files in an NJE network requires tables to define how to map VMS Record Management Services (RMS) parameters (9) into and out of the DCB parameter categories.

Note that, although the VM file system is different from both the IBM OS system and the VAX VMS system, the mapping to IBM VM file systems is ignored because the transport protocol is based on the IBM OS file system parameters. The mapping from an OS to a VM system and consequently from VAX VMS to VM is contained in the bulk data transfer programs which run in the VM system.

<u>File System Correspondence</u>

An important component in the design is the correspondence which is drawn between the attributes of files of either system. Files in the Fortran development environment include unformatted data, formatted output, and formatted data and also source code files created and modified by standard editors. In the VAX the standard file extensions .FOR, .LIS, .MAP, .DAT, and .LOG represent this group of file types. All but the .DAT files could be sent by the existing NJE print or punch commands as appropriate. The Fortran open statement gives users the flexibility to assign file attributes according to their requirements, and so the .DAT file characteristics may vary among the range of available choices. Moreover, because file name and file type extensions are assignable, it is not possible to draw inferences as to file characteristics from components of the file specification.

The achievement of a useful mapping starts from the analysis of the file types which each system provides to users. Table 3 lists the symbolic characters which make up the RECFM to specify the record types and alternative carriage control choices. Table 4 lists the tokens by which each of the equivalent items are specified to VAX RMS. The stream record types are not included because they did not exist when our plan was drawn up and because at Argonne there is yet no need to support them. Because neither file system is robust enough to support all file types of the other system without ambiguity, it is first necessary to consolidate elements of these tables where possible. Consolidation will decrease the effective number of correspondences between systems.

Table 3: IBM Record Format Specifications

Record Types:

- F - Fixed-length
- FS - Fixed-length, standard
- V - Variable-length
- VS - Variable-length, spanned
- U - Undefined

Carriage Control:

-   - No control, none implied
- A - ANSI (ASA) control
- M - machine control

Table 4: VAX RMS Record Format Specifications

Record Types:

- FIX - Fixed-length
- VAR - Variable-length
- VFC - Variable with fixed portion
- UDF - No record type specified

Carriage control:

-   - No control, none implied
- CR - Implied CR-LF
- FTN - ANSI (ASA) control
- PRN - Printer control (only VFC)

On the IBM side, the difference between F and FS is not in the data and so the difference need not be maintained between different systems. Therefore we handle FS data records as equivalent to F. The U or undefined recordtype is a variation of V where all of the data bytes are under programmer control and records are never blocked. We therefore handle U records similar to V logical records. If data bytes have control significance, then they will be transmitted and can be used on the receiving system. The M carriage control is a superset of A carriage control and cannot be supported directly on VAX systems. Thus it is useful to treat M as a variation of A, detect it, and map it the same as A.

On the VAX side we have chosen to consolidate the VFC type into the VAR type except for the special case of VFC with PRN. If a programmer uses a fixed control area of VFC, then it is unlikely that he would expect to do the same on an IBM system because of the absence of a similar record type. Moreover, VMS programs and utilities do not often create files in the VFC category and the Fortran language does not permit access to the fixed fields. Some VMS editors (SOS, WYLVAX) do use the fixed field for line numbers. But on IBM systems with different editors, the fixed field would not be needed and it is natural to want only the variable fields to be accessible. The other consolidation we make is to map the PRN carriage control information to FTN carriage control as well as can be done. The PRN data would not be understood by IBM equipment and FTN is likely to duplicate ordinary usage of PRN. We have determined that the type UDF signifies that RMS was not used to create the file and for the present we classify UDF files as not transmittable. Tables 5 and 6 show the complete mappings which were proposed and incorporated in the file transfer programs. There is a table for mapping file or record attributes from an IBM system to a VAX (Table 5) and from a VAX to an IBM system (Table 6). In the following we discuss other aspects of the design choices and the effects they have on users.

Table 5: IBM to VAX Record Attribute Mappings

IBM	VAX	
RECFM	RFM	RAT
F,FB	FIX	CR
FA,FBA	FIX	FTN
FM,FBM	FIX	FTN
FS,FBS	FIX	CR
FSA,FBSA	FIX	FTN
FSM,FBSM	FIX	FTN
U	VAR	CR
UA	VAR	FTN
UM	VAR	FTN
V,VB	VAR	CR
VA,VBA	VAR	FTN
VM,VBM	VAR	FTN
VS,VBS	VAR	none

Table 6: VAX to IBM Record Attribute Mappings

VAX		IBM
RFM	RAT	RECFM
FIX	none	FB
FIX	CR	FB
FIX	FTN	FBA
VAR	CR	VB
VAR	FTN	VBA
VFC	none	VB
VFC	CR	VB
VFC	FTN	VBA
VFC	PRN	VBA
VAR	none	VBSB
UDF	not an RMS file	not transmittable

After the file type consolidations are considered, the sole remaining asymmetry stems from the IBM file system physical blocking factor. This detail is not

relevant to the operation of a program apart from how effectively the program uses the disk device and memory for buffer space. The blocking factor is not even required in the job control language (JCL) records for an existing file. The design choice was to add the blocking attribute always when a file was being transmitted into the IBM environment even if there would be only one logical record per block. Therefore, if an unblocked file which originated in an IBM system were moved to a VAX and then back to an IBM system, it would differ from the original by addition of the blocking attribute. Block sizes are assigned according to the existing recommendations for the mix of different disk drive track lengths in the Argonne central computer environment.

There is no ambiguity in the mapping of unformatted data files written by Fortran programs where the OPEN statement is not used to change the default RMS RFM and RAT parameters. For IBM systems, the essential characteristic of the record format is that it is variable spanned (VS). On VMS systems, an unformatted file written by a Fortran program is variable with no carriage control attribute. The data records in either system are written in segments with embedded control words describing how the segments are recombined. The segmentation schemes are different so that it is necessary to detect such files and handle their records differently from those of all other file types where the records are not assumed to have embedded control information. In the VAX, the absence of carriage control is assumed to imply segmented data; in the IBM system, it is the presence of the spanned attribute. Although these choices result in no ambiguity for files written by Fortran programs, the segmentation mechanism in the VAX VMS system is not an RMS characteristic; therefore, it is not possible to distinguish segmented files from other non-segmented files having the same RMS attributes.

Server Process Modifications

Modifications to the NJE server process to support general file transfer turned out to be minimal. The conversion of the user file to a transmittable file would be done in a user process at command execution and the converted file would be sent by the server. Because the intermediate or temporary files would proliferate, an option was added to the server whereby a file could be sent and then deleted. This would eliminate the clutter of useless files.

Experiences

Whereas the original goal of the file transfer implementation project was met, we have encountered some unforeseen needs which happily have also been satisfied. These needs turned out to be involved with VAX to VAX transfers of image and object files.

Transmitting image files (.EXE) from one VAX VMS system to another would help to speed up the task of making quick updates to our networking software. Image files have fixed length records with record attribute of "NONE". Our mapping maintains the fixed records but the record attribute would be changed to "CR" at a receiving VAX. It turns out that the image activator ignores the record attribute and we have been able to update networking software images even when the weather was too hot or too cold to go out. More important than the weather, to use the network is faster and more efficient because the VAX VMS sites at Argonne National Laboratory are widely dispersed.

In the case where the version of VMS on two VAX computers is different there can be run time library incompatibilities which render transferred image files useless. To avoid the problems caused by having duplicate source code on many machines, it is desirable to transfer object files. Object files have the same attributes as Fortran unformatted data files (RFM=VAR and RAT=NON) but the records are not segmented. (Segmentation is not an attribute of the RMS file system.) When the above attributes are encountered, the file transfer emulator software assumes the presence of segmentation control words and transfer does not work. However, we learned that the RMS attribute can be changed from no carriage control to CR without affecting the integrity of an object file with regard to the linker and even to the ANALYZE utility. The CONVERT utility and the file definition language provide the mechanism. Therefore, one can change the carriage control attribute of an object file before it is sent and when the file arrives at another VAX it can be linked, added to an object library, etc. using the standard utilities.

DATA CONVERSION SERVICE

The VAX and the IBM computers have different internal formats for integer and real Fortran data types. The networking simply provides for the correct communication of a file made from a sequence of 8-bit bytes. To address the data conversion requirements which would arise, a set of subroutines was written and added to the libraries of the IBM batch and CMS systems. All of the Fortran data types of both machines are represented. The conversions which are practical have been implemented so that users can write simple programs to convert their transmitted data records. The subroutines perform record-oriented conversions, permitting an entire data record to be converted in one invocation if all of the data is uniform in type.

GRAPHICS DATA SERVICE

In order to facilitate the production of graphics output at Argonne a common graphics data file (metafile) was implemented both on the IBM and on the VAX VMS computers. Because of the data format, users do not have to run data conversion programs for graphics data exchanged between the computers. Output from graphics software which runs on one variety of computer can be plotted on hardware attached to the other variety using simple commands. This service has the effect of making the local graphics systems on both the IBM and the VAX VMS systems machine-independent as well as device-independent.

MAIL DELIVERY SERVICE

One of the natural applications for a system which is able to transfer files is the communcation of electronic messages. The mail systems used by VAX VMS users (MAIL) and by IBM CMS users (VM/SP2 NOTE or PROFS) were connected by programs implemented on the VAX. NJE is the vehicle by which the formatted messages are transmitted. The connection to the MAIL command is automatic both for transmission and reception. The implementation of this application is described elsewhere in these proceedings (10).

DEVELOPMENT PLANS

Future directions for the NJE software include support of a better device and a more intelligent device driver. A promising alternative is the DMF32's synchronous port. That device's DMA capability would essentially remove the interrupt processing load which exists with the DUP-11. In addition, line speeds up to 19.2k baud are possible. A driver to use the device's general byte synchronous capability will be needed. Higher speeds and lower overhead will make other applications and services practical, where the present, medium data rates and overhead are not sufficient. Other interfaces will be considered as they become available.

Another goal is the completion of the NJE functionality at the VAX end. There is no support in the NJE emulator to accept jobs to be submitted to the VAX batch queues. Considerations of resource control, job ownership, and user validation will have to be studied.

SUMMARY

The VAX VMS NJE protocol emulator enables a VAX VMS computer system to participate as a peer member and as an end node in an IBM file transfer network. The software which has been developed at Argonne National Laboratory is used to attach several VAX VMS systems to a central IBM-based complex of computers under the control of JES3 and VM/370. The system has been in use at Argonne since January 1981 and is undergoing continuing development and refinement. Refinement has benefitted from the experiences of users at other sites who use different IBM operating systems and environments which have their own idiosyncrasies. At other installations VAX VMS systems are successfully connecting to JES2 and VM with the NJE emulator software.

The basic NJE networking enables users to utilize remote hardware. The general, sequential file transfer service forms the basis for a mail delivery service and a data conversion service. Device-independent graphics software in each environment have identical interfaces and with the common graphics data provides a machine-independent graphics service.

The current version of the VAX VMS NJE emulator software is being made ready for distribution coordinated by the National Energy Software Center (NESC) located at Argonne National Laboratory. The Center serves as the software exchange and information center for computer software developed under U. S. Department of Energy (DOE) sponsorship. The address is:

National Energy Software Center
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439
(312) 972-7250.

BIBLIOGRAPHY

(1) Engert, D. E. An IBM NJE Protocol Emulator for VAX/VMS, Proceedings of the Digital Equipment Computer Users Society, Vol. 7, No. 4, USA Spring 1981.

(2) Hutchinson, J. M. Job Networking Facilities, IBM Washington Systems Center Technical Bulletin GG22-9042-00, Gaithersburg, Maryland, March 1980.

(3) Raffenetti, R. C. and Zelle, B. R. Friendly Neighborhood Computer Project: Performance Measurements and Improvements for VAX VMS NJE Networking, Argonne National Laboratory Technical Memorandum 394, February 1982.

(4) Beander, B. The PME Performance Measurement and Evaluation Package, Digital Equipment Corporation, DECUS SIG tape, Fall 1979.

(5) Baker, W. E., Bertoncini, P. J., Engert, D. E., Raffenetti, R. C., Rynes, P. E., and Wei, S. Friendly Neighborhood Computer Project: Extension of the IBM NJE Network to DEC VAX Computers, Argonne National Laboratory Technical Memorandum 383, February 1982.

(6) Bertoncini, P. J., Engert, D. E., Raffenetti, R. C., and Rynes, P. E. ibid. (revised), November 1982.

(7) International Business Machines Corporation, Bulk Data Transfer: Program Description and Operations Manual, International Business Machines Corporation, SH20-2088-0, May 1978.

(8) International Business Machines Corporation, Bulk Data Transfer: Systems Guide, International Business Machines Corporation, LY20-2367-0, May 1978.

(9) Digital Equipment Corporation, VAX-11 Record Management Services Reference Manual, Digital Equipment Corporation, Maynard, Massachusetts, May 1982.

(10) Osudar, J. The NJE Mail Bridge Between VAX VMS and IBM VM/CMS Systems, Proceedings of the Digital Equipment Computer Users Society, Vol. 9, No. 4, USA Spring 1983.

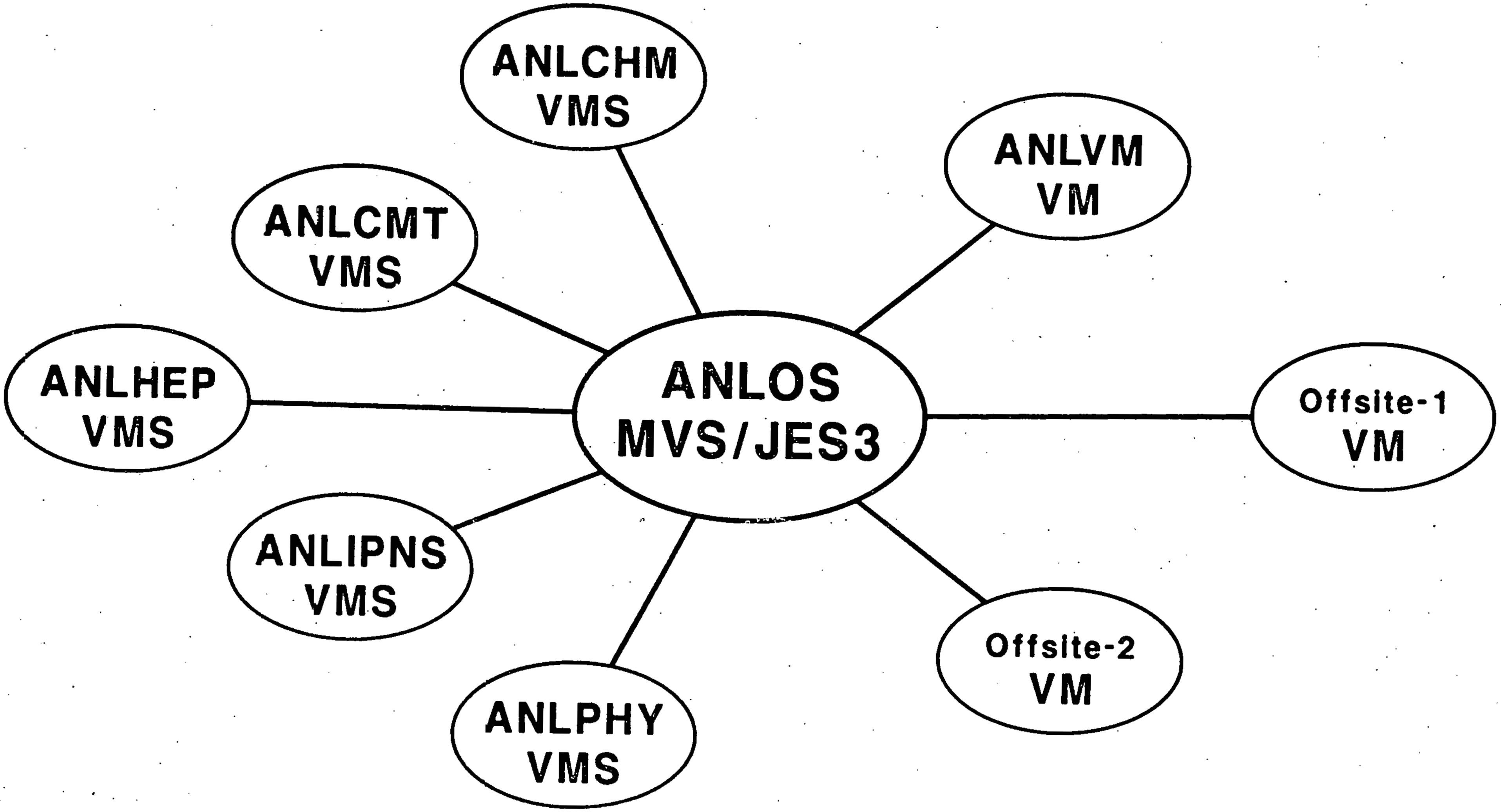
ANLCHM
VMS
ANLVM
VM
ANLCMT
VMS
ANLHEP
VMS
ANLOS
MVS/JES3
Offsite-1
VM
ANLIPNS
VMS
Offsite-2
VM
ANLPHY
VMS

Figure: 1

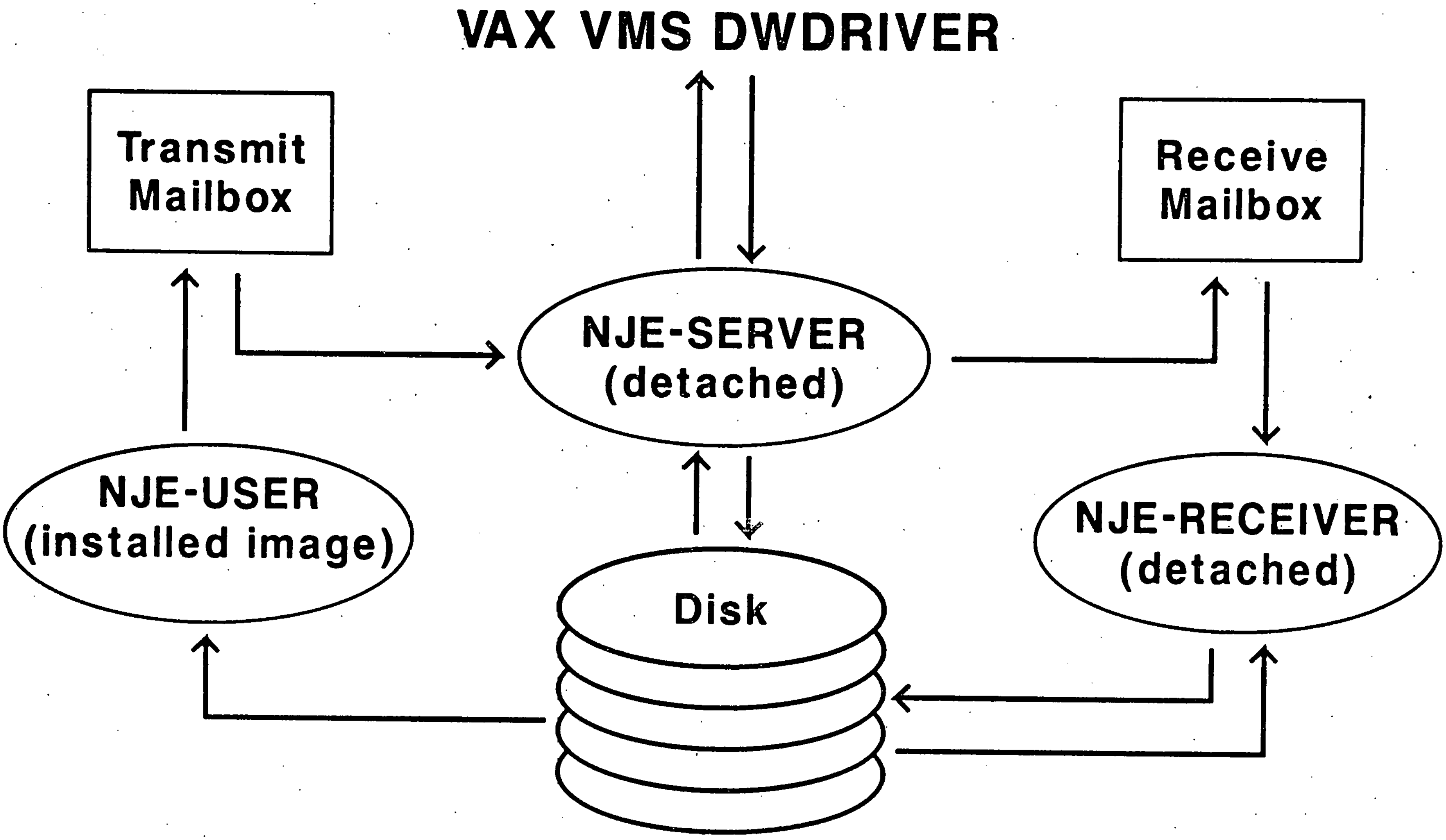
VAX VMS DWDRIVER
Transmit Mailbox
Receive Mailbox
NJE-SERVER (detached)
NJE-USER (installed image)
NJE-RECEIVER (detached)
Disk

Figure: 2

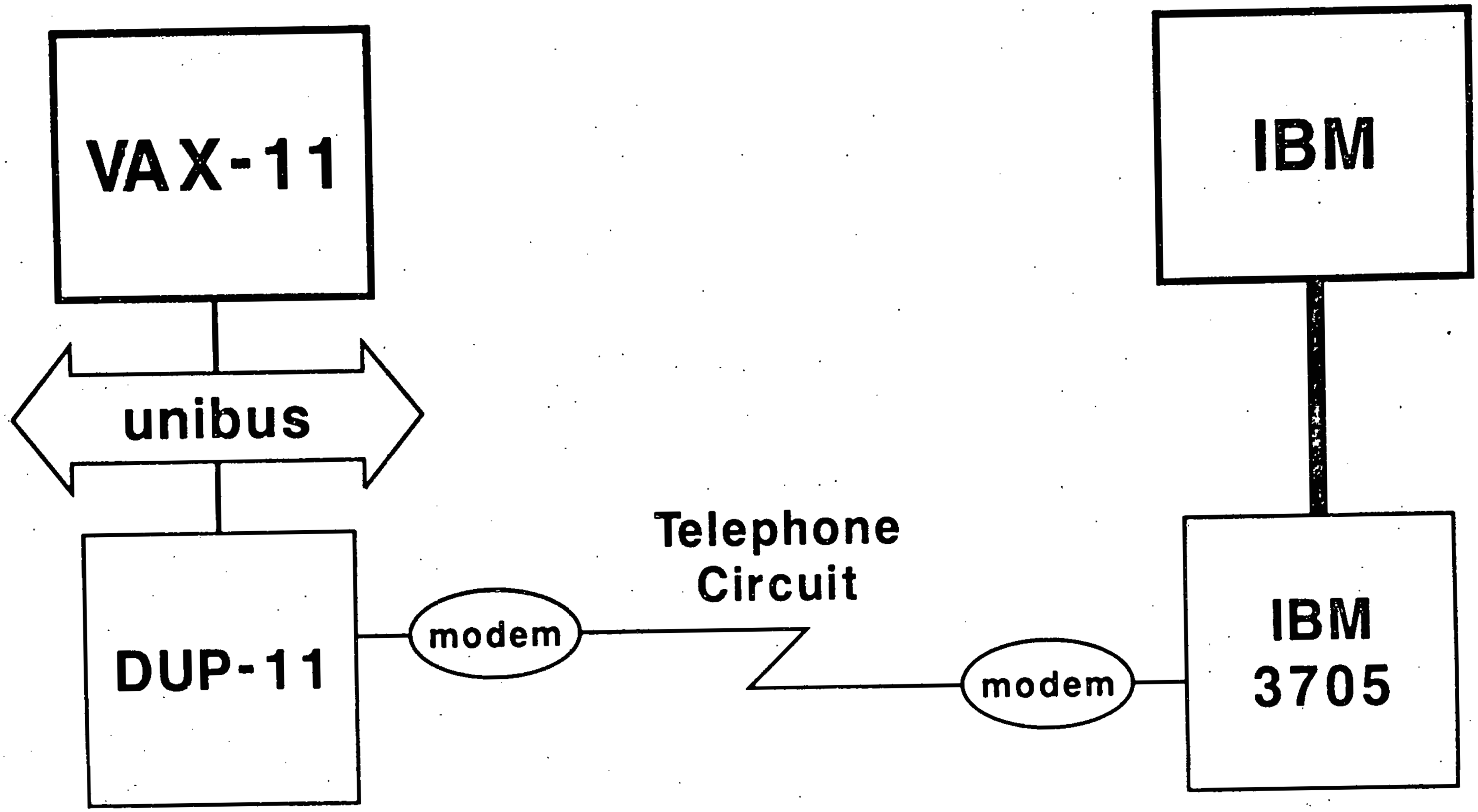
VAX-11
unibus
DUP-11
modem
Telephone
Circuit
modem
IBM
IBM
3705

Figure: 3

old master

PARTICIPATION OF VAX VMS COMPUTERS IN IBM FILE-TRANSFER NETWORKS

Richard C. Raffenetti
Argonne National Laboratory
Argonne, Illinois 60439

ABSTRACT

Communications software written at Argonne National Laboratory enables VAX VMS computer systems to participate as end nodes in a standard IBM file-transfer network. The software, which emulates the IBM Network Job Entry (NJE) protocol, has been in use at Argonne for over two years, and is in use at other installations. The basic NJE services include transfer of "print" and "punch" files, job submittal, execution of remote commands, and transmission of user-to-user messages. The transmit services are asynchronous to the user's VMS session and received files are automatically routed to a designated user directory. Access to files is validated according to the VMS protection mechanism. New features which were added recently include application level software to transfer general, sequential files and to bridge the "electronic mail" systems of VMS and VM/CMS. This paper will review the NJE emulator and describe the design and implementation of the sequential file transfer service. The performance of the emulator will be described. Another paper at this symposium will describe the mail bridge.

INTRODUCTION

This paper is a second report (1) on the features and capabilities of software written at Argonne National Laboratory which permits a VAX VMS system to participate as a peer member of an IBM file transfer network, known as NJE. The software was originally written to give VAX VMS users convenient access to the significant resources in Argonne's central complex of IBM computers. With five VAX systems now participating and others planned, the communications facility has proved valuable for access to resources everywhere in the network. For example, one of the VAX systems is a member of a nationwide DECnet-based network.

In this report we are describing new features of the software. We have studied and improved its performance and have added the capability to transfer general, sequential files. Support features and new applications or services are now in place. We describe briefly the user applications which operate in conjunction with the NJE networking service.

It is our goal to make the NJE emulator software general enough to communicate successfully with all of the systems which support the protocol providing there have been no site-dependent changes to the IBM software. We have made the software widely available by putting it into the public domain. The channels for distribution are described.

IBM FILE TRANSFER NETWORKS

The IBM file transfer network which we refer to as NJE (for Network Job Entry) is implemented in standard 370, 30xx, and 43xx operating systems (2). For example, in the major IBM operating systems the support is in JES2 or JES3 (for MVS systems) and RSCS (for VM systems). (The older IBM operating environments using HASP and ASP also support NJE networking.) Other IBM file transfer protocols are used in some but not all of the IBM systems and therefore are less universal.

The NJE protocol is in contrast to the Remote Job Entry (RJE) protocol which forms the basis for other VAX to IBM communication alternatives (e.g. HASP workstation, 2780/3780). The RJE protocol creates a remote system viewed not as a peer of the IBM host but simply as a workstation having a printer and card reader, and sometimes a card punch and/or a console. All computers which participate in an NJE network are peers so far as they support the full NJE protocol.

The NJE protocol transfers "objects" in the classes of SYSOUT (print or punch), job, or console command. The standard print object consists of a sequence of records limited to 133 characters in length; the standard punch object has records limited to 80 characters. Jobs are similar to punch and may be sent to the input queue of a remote host. The output from jobs are routed to the job origin or optionally to other network destinations. The console commands are short messages and their transmission is immediate. That is, a console command is interleaved into the flow of a SYSOUT or job object.

Information which does not conform to these object categories is communicated by transforming it to a punch object before it is entered onto the network and then performing the inverse transformation after it is received. The programs which do this kind of transformation are higher-level services which are independent of the NJE networking services.

The network transmits objects by a store-and-forward mechanism. If transmission is through an intervening node, a SYSOUT or job object is completely transferred to the intermediate node, and then to the destination node. After each intermediate transfer, the receiving node assumes responsibility for the object, and relieves the previous sending node of responsibility. Messages or commands are not stored but are forwarded immediately.

The NJE networking protocol is based on a BISYNC line protocol and provides a high degree of reliability. Cyclic-redundancy-counts are computed to guard against errors in each BISYNC record, and each record contains a sequence byte to ensure that records are not lost. The protocol is half-duplex so that each record is acknowledged in turn. Each BISYNC record can contain both the data moving in one direction and the response regarding the data moving in the opposite direction (piggybacking). Finally, NJE protocols include data byte compression which permits increased effective transfer rates in the case where a file has identical data bytes in sequence.

THE VAX VMS NJE PROTOCOL EMULATOR

The VAX VMS implementation of a protocol emulator for NJE communication only permits the VMS system to be an end node in an NJE network. By being an end node the VMS system is freed from having to temporarily store files and from having to maintain knowledge about the network topology. At Argonne the network is arranged in a star topology with a large IBM batch system as the central node. Five VAX VMS computers located around the laboratory connect directly to the batch node, ANLOS. In addition three VM/370 systems which support many interactive users are connected to the same batch node. Two of the VM systems are IBM computers not located at Argonne. Figure 1 contains a schematic diagram of the topology. Except for the non-Argonne (OFF-SITE) nodes, the names are the identifiers by which the various nodes are known. The operating systems are noted for each node.

Figure 1: Argonne's NJE Network Topology

NJE Emulator Software

Important features of the NJE protocol emulator are its asynchronous operation, the automatic routing of information to users, and adherance to the file system protection scheme. Figure 2 is a schematic description of the emulator organization. In the figure, the ovals represent processes and the rectangles represent system mailboxes. NJE-SERVER is an image which requires many privileges and runs in its own detached process. It runs continuously, managing the line traffic to and from the communication device, the flow of commands from users, messages to users, and files to and from user file directories. Six tasks, known as line-driver, file-in, file-out, message-in, message-out, and broadcast, are dispatched by the main program in order of priority when there is work for a task module to perform. The line-driver task is highest priority and the other tasks are lower, the order being the same as the above sequence. Event flags as well as a program status word post the need for and the completion of the work of each task. NJE-USER is an installed, privileged image which is executed in a user's process. This image is executed by command procedures which validate the user commands. NJE-USER passes the NJE commands to the server process through the transmit mailbox after validating access to any files which have been named for transmission. Access to the transmit mailbox is limited by requiring privilege so that unauthorized commands may not be written to the communication server. The image NJE-RECEIVER runs in its own detached process and receives information from the server process by reading messages passed through the receive mailbox which also allows only system access. The receiver is a new feature of the emulator which currently provides a bridge between the IBM and VAX VMS mail or note facilities (see below). The receiver provides a mechanism whereby determinate transformations may be carried out automatically on received information.

Figure 2: Schematic Organization of the Emulator Software

The server process handles the six major tasks as described above. When a SYSOUT object is received, the server process looks up the destination in a table provided by the system manager. Each VAX VMS user who will use the NJE services has an entry in the table. If the destination is a user, the object is written to a designated directory. If the object is a console command it may be regarded as a user message or as a node-oriented command. Messages are written to the designated user's terminal with the broadcast system service. The NJE emulator validates and executes commands and sends the response as a message to the origin. The valid commands are the internal NJE management commands which report data to tell how the NJE programs are functioning.

NJE Emulator Hardware

The hardware used to connect to the IBM system consists of the DUP-11 unibus device, a pair of synchronous modems, and a dedicated telephone circuit. At Argonne, the connection to the IBM computer is through an IBM 3705 telecommunications processor "front end." Figure 3 is a schematic diagram of the hardware. The maximum data rate of a DUP-11 in this configuration is 9600 baud and at Argonne all of the VAX VMS NJE connections are run at that rate. The data rate is governed by the modems. On a system with a busy unibus, the bus request level for the DUP-11 may have to be elevated if there are frequent errors in the reception or transmission of records.

Figure 3: Schematic Diagram of the Hardware

NJE EMULATOR PERFORMANCE

There are two parts to the overhead incurred by the operation of the emulator. The first is the use of the central processor by the emulator software. The second is the processing carried out by the system to handle input to and output from the synchronous interface.

Central Processor Overhead

Since the original software was put into service, the central processor performance has been measured and the server code modified to lower the overhead incurred by the original version (3). A program for performance measurement and evaluation, which was obtained from the DECUS VAX SIG tape (4) is the tool which was used to determine which sections of the server software use large fractions of the processing. The modifications lowered the overhead by a factor of about five. The central processor overhead of the current implementation in an acceptable range for the existing systems.

The emulator software performance data presented here is representative and applies to the transmission of well-defined, sample data files. The data was collected from a VAX-11/780 system during mid-afternoon on a weekday. The system had about 20 interactive users and a total of about 50 processes. The communications circuit was clocked by modems at 9600 bits per second. In the standard implementation at Argonne, the server process is run at a base priority of five to ensure that it obtains all of the central processing share that it needs.

Table 1 displays representative data obtained from the current version of the NJE emulator when a file of non-compressible character data is sent as a stream of 80-byte records. (A sequence of bytes can be compressed if two or more blanks or three or more identical non-blank characters follow one after the other.) In Tables 1 and 2 the symbol CP stands for the central processor time used by the NJE server during the transmission of 4096 80-byte records. The symbol EL stands for the elapsed time. The RATE is the effective transmission rate which is computed from the data size (2.5 Mbits) and the elapsed time. The %CP is the percentage of elapsed time which the value of CP represents. The CP overhead is the number of central processor seconds it takes to prepare and transmit or to receive a megabit (2**20 bits) of data.

Table 1: Performance for Transmission of Noncompressible Data

	Transmit	Receive
CP (sec.)	22.4	16.0
EL (sec.)	375	408
RATE (bits/sec)	6991	6425
%CP	6.0	3.9
CP Overhead (sec./Mbit)	9.0	6.4

Table 2 displays the same data for transmission of 4096 80-byte records each of which is entirely compressible (all bytes of a record are the same). The %CP is quite high in this case but the overhead is lower than for non-compressible transfers. It would be unusual for users to have very highly compressible data. In text files, blanks are the dominant repeated character, while in binary data, the zero is common. We have observed a 15% compression in general, compiled program listings.

Table 2: Performance for Transmission of Compressible Data

	Transmit	Receive
CP (sec.)	16.9	12.7
EL (sec.)	55	64
RATE (bits/sec)	47662	40960
%CP	30.7	19.8
CP Overhead (sec./Mbit)	6.8	5.1

When the server is idle, very short messages are exchanged by the hosts at a rate of one exchange each two seconds. The central processor utilization during the time when no files are being transmitted is negligible.

IO Processor Overhead

The IO processing is a second major factor in the evaluation of the NJE emulator performance. There are two aspects related to the device. The DUP-11 was chosen largely by default. It is a supported VMS device and the driver is capable of carrying out a general binary synchronous communication. Because the DUP-11 is not a DMA device, then during file transmission there is significant interrupt processing. The interrupt load on the system is shown by the monitor utility to be in the range of 5%. Our development plans include substitution of another synchronous DMA interface for the DUP-11.

To emulate the NJE protocol, the binary capability of the DUP-11 driver is needed. With that interface, the VMS driver does not recognize the end of a BISYNC record. A receive buffer is allocated for the QIO read and the request completes only when the buffer has filled, meaning that following the data bytes the buffer will contain pad characters (hex FF) in accord with the line's mark state. The NJE emulator provides for this situation by posting small or large receive buffers depending on whether the communication is idle or not. However, during transmit or receive processing the time during which there is no productive data transfer is in the range of 15-20% of the total line time. A more intelligent driver for the DUP-11 would boost the effective transfer rates for non-compressible data to around 8000 baud for both transmit and receive processing. More effective use of the line capacity will raise the %CP time used by a similar amount while the central processor overhead should remain the same.

THE USER INTERFACE

The user interface (5,6) to the NJE networking facilities is described by command procedures. The command procedures serve three purposes. First, the commands require tailoring to the specific networking environment and therefore it is useful to keep that information apart from the compiled code. Second, many of the commands are named and oriented toward the batch IBM system and permit users to submit and control batch jobs as well as route output to central high-speed printers. Third, by using a command procedure it is possible to do command validation processing and give the user quick feedback if the command cannot be acted upon. The commands which have been implemented are fairly robust and prompt the user for missing or incorrect parameters. Comprehensive help documents have been added to the system help facility to describe all of the commands, their parameters, and their modifiers.

Because the network is IBM-oriented, the control information as well as the data consists of 8-bit EBCDIC characters and translation from or to VAX ASCII is carried out by the emulator software whenever the data is textual. Transfer of print or punch record streams is assumed to require translation; transfer of general sequential files requires user direction as to whether or not the translation is done. The character translation is directed by tables which were devised at Argonne but these can easily be changed.

TRANSFER OF SEQUENTIAL FILES

There are two aspects to the transfer of data to different computers in a network. These are transmission and data conversion. In the following we describe the transmission of sequential files, leaving the data conversion topic for a later section.

We had provided special programs to transform graphics metafiles to and from 80-byte records which could be transmitted in the NJE network. However, there was a need for transfer of more general files. A plan was written for the development of a similar set of programs which would transfer general sequential files. The goal in the implementation was that Fortran programmers could move their files between systems and do productive work on either system without needing to gain special knowledge of the respective file systems. Argonne National Laboratory is a scientific establishment and the Fortran language is a major tool of DEC and IBM users both. The files which are manipulated in and around Fortran programming efforts were held to be of greatest importance in the realization of a file transfer system.

File Transfer Protocol

An IBM program, called Bulk Data Transfer, IUP #5796-PKK (7,8), was in use to transfer data files between the VM/CMS interactive users and the Argonne MVS/JES3 batch system. The plan was to emulate the protocol of the Bulk Data Transfer programs. A useful side effect of this choice is that, like the NJE protocol emulator itself, no changes to standard IBM software are necessary for successful communication and file transfer to VAX VMS systems. (The IBM version of the programs has been extended locally to permit transfer of logical records exceeding 32767 bytes in length, which was not supported. Another extension allowed the program to be run under VM/370. These extensions are fully compatible.)

In the bulk data transfer protocol, a header record contains IBM OS data control block (DCB) parameters of the file and a trailer record contains information such as record count and date by which successful transfer can be confirmed. The data records of the file are converted to IBM's variable, blocked, spanned record format (RECFM=VBS). The block size is 80 bytes, permitting transfer in the NJE network under the category of punch. Transfer of standard VAX VMS files in an NJE network requires tables to define how to map VMS Record Management Services (RMS) parameters (9) into and out of the DCB parameter categories.

Note that, although the VM file system is different from both the IBM OS system and the VAX VMS system, the mapping to IBM VM file systems is ignored because the transport protocol is based on the IBM OS file system parameters. The mapping from an OS to a VM system and consequently from VAX VMS to VM is contained in the bulk data transfer programs which run in the VM system.

<u>File System Correspondence</u>

An important component in the design is the correspondence which is drawn between the attributes of files of either system. Files in the Fortran development environment include unformatted data, formatted output, and formatted data and also source code files created and modified by standard editors. In the VAX the standard file extensions .FOR, .LIS, .MAP, .DAT, and .LOG represent this group of file types. All but the .DAT files could be sent by the existing NJE print or punch commands as appropriate. The Fortran open statement gives users the flexibility to assign file attributes according to their requirements, and so the .DAT file characteristics may vary among the range of available choices. Moreover, because file name and file type extensions are assignable, it is not possible to draw inferences as to file characteristics from components of the file specification.

The achievement of a useful mapping starts from the analysis of the file types which each system provides to users. Table 3 lists the symbolic characters which make up the RECFM to specify the record types and alternative carriage control choices. Table 4 lists the tokens by which each of the equivalent items are specified to VAX RMS. The stream record types are not included because they did not exist when our plan was drawn up and because at Argonne there is yet no need to support them. Because neither file system is robust enough to support all file types of the other system without ambiguity, it is first necessary to consolidate elements of these tables where possible. Consolidation will decrease the effective number of correspondences between systems.

Table 3: IBM Record Format Specifications

Record Types:

F - Fixed-length
FS - Fixed-length, standard
V - Variable-length
VS - Variable-length, spanned
U - Undefined

Carriage Control:

- No control, none implied
A - ANSI (ASA) control
M - machine control

Table 4: VAX RMS Record Format Specifications

Record Types:

FIX - Fixed-length
VAR - Variable-length
VFC - Variable with fixed portion
UDF - No record type specified

Carriage control:

- No control, none implied
CR - Implied CR-LF
FTN - ANSI (ASA) control
PRN - Printer control (only VFC)

On the IBM side, the difference between F and FS is not in the data and so the difference need not be maintained between different systems. Therefore we handle FS data records as equivalent to F. The U or undefined recordtype is a variation of V where all of the data bytes are under programmer control and records are never blocked. We therefore handle U records similar to V logical records. If data bytes have control significance, then they will be transmitted and can be used on the receiving system. The M carriage control is a superset of A carriage control and cannot be supported directly on VAX systems. Thus it is useful to treat M as a variation of A, detect it, and map it the same as A.

On the VAX side we have chosen to consolidate the VFC type into the VAR type except for the special case of VFC with PRN. If a programmer uses a fixed control area of VFC, then it is unlikely that he would expect to do the same on an IBM system because of the absence of a similar record type. Moreover, VMS programs and utilities do not often create files in the VFC category and the Fortran language does not permit access to the fixed fields. Some VMS editors (SOS, WYLVAX) do use the fixed field for line numbers. But on IBM systems with different editors, the fixed field would not be needed and it is natural to want only the variable fields to be accessible. The other consolidation we make is to map the PRN carriage control information to FTN carriage control as well as can be done. The PRN data would not be understood by IBM equipment and FTN is likely to duplicate ordinary usage of PRN. We have determined that the type UDF signifies that RMS was not used to create the file and for the present we classify UDF files as not transmittable. Tables 5 and 6 show the complete mappings which were proposed and incorporated in the file transfer programs. There is a table for mapping file or record attributes from an IBM system to a VAX (Table 5) and from a VAX to an IBM system (Table 6). In the following we discuss other aspects of the design choices and the effects they have on users.

Table 5: IBM to VAX Record Attribute Mappings

IBM	VAX	
RECFM	RFM	RAT
F,FB	FIX	CR
FA,FBA	FIX	FTN
FM,FBM	FIX	FTN
FS,FBS	FIX	CR
FSA,FBSA	FIX	FTN
FSM,FBSM	FIX	FTN
U	VAR	CR
UA	VAR	FTN
UM	VAR	FTN
V,VB	VAR	CR
VA,VBA	VAR	FTN
VM,VBM	VAR	FTN
VS,VBS	VAR	none

Table 6: VAX to IBM Record Attribute Mappings

VAX		IBM
RFM	RAT	RECFM
FIX	none	FB
FIX	CR	FB
FIX	FTN	FBA
VAR	CR	VB
VAR	FTN	VBA
VFC	none	VB
VFC	CR	VB
VFC	FTN	VBA
VFC	PRN	VBA
VAR	none	VBSB
UDF	not an RMS file	not transmittable

After the file type consolidations are considered, the sole remaining asymmetry stems from the IBM file system physical blocking factor. This detail is not

relevant to the operation of a program apart from how effectively the program uses the disk device and memory for buffer space. The blocking factor is not even required in the job control language (JCL) records for an existing file. The design choice was to add the blocking attribute always when a file was being transmitted into the IBM environment even if there would be only one logical record per block. Therefore, if an unblocked file which originated in an IBM system were moved to a VAX and then back to an IBM system, it would differ from the original by addition of the blocking attribute. Block sizes are assigned according to the existing recommendations for the mix of different disk drive track lengths in the Argonne central computer environment.

There is no ambiguity in the mapping of unformatted data files written by Fortran programs where the OPEN statement is not used to change the default RMS RFM and RAT parameters. For IBM systems, the essential characteristic of the record format is that it is variable spanned (VS). On VMS systems, an unformatted file written by a Fortran program is variable with no carriage control attribute. The data records in either system are written in segments with embedded control words describing how the segments are recombined. The segmentation schemes are different so that it is necessary to detect such files and handle their records differently from those of all other file types where the records are not assumed to have embedded control information. In the VAX, the absence of carriage control is assumed to imply segmented data; in the IBM system, it is the presence of the spanned attribute. Although these choices result in no ambiguity for files written by Fortran programs, the segmentation mechanism in the VAX VMS system is not an RMS characteristic; therefore, it is not possible to distinguish segmented files from other non-segmented files having the same RMS attributes.

Server Process Modifications

Modifications to the NJE server process to support general file transfer turned out to be minimal. The conversion of the user file to a transmittable file would be done in a user process at command execution and the converted file would be sent by the server. Because the intermediate or temporary files would proliferate, an option was added to the server whereby a file could be sent and then deleted. This would eliminate the clutter of useless files.

Experiences

Whereas the original goal of the file transfer implementation project was met, we have encountered some unforeseen needs which happily have also been satisfied. These needs turned out to be involved with VAX to VAX transfers of image and object files.

Transmitting image files (.EXE) from one VAX VMS system to another would help to speed up the task of making quick updates to our networking software. Image files have fixed length records with record attribute of "NONE". Our mapping maintains the fixed records but the record attribute would be changed to "CR" at a receiving VAX. It turns out that the image activator ignores the record attribute and we have been able to update networking software images even when the weather was too hot or too cold to go out. More important than the weather, to use the network is faster and more efficient because the VAX VMS sites at Argonne National Laboratory are widely dispersed.

In the case where the version of VMS on two VAX computers is different there can be run time library incompatibilities which render transferred image files useless. To avoid the problems caused by having duplicate source code on many machines, it is desirable to transfer object files. Object files have the same attributes as Fortran unformatted data files (RFM=VAR and RAT=NON) but the records are not segmented. (Segmentation is not an attribute of the RMS file system.) When the above attributes are encountered, the file transfer emulator software assumes the presence of segmentation control words and transfer does not work. However, we learned that the RMS attribute can be changed from no carriage control to CR without affecting the integrity of an object file with regard to the linker and even to the ANALYZE utility. The CONVERT utility and the file definition language provide the mechanism. Therefore, one can change the carriage control attribute of an object file before it is sent and when the file arrives at another VAX it can be linked, added to an object library, etc. using the standard utilities.

DATA CONVERSION SERVICE

The VAX and the IBM computers have different internal formats for integer and real Fortran data types. The networking simply provides for the correct communication of a file made from a sequence of 8-bit bytes. To address the data conversion requirements which would arise, a set of subroutines was written and added to the libraries of the IBM batch and CMS systems. All of the Fortran data types of both machines are represented. The conversions which are practical have been implemented so that users can write simple programs to convert their transmitted data records. The subroutines perform record-oriented conversions, permitting an entire data record to be converted in one invocation if all of the data is uniform in type.

GRAPHICS DATA SERVICE

In order to facilitate the production of graphics output at Argonne a common graphics data file (metafile) was implemented both on the IBM and on the VAX VMS computers. Because of the data format, users do not have to run data conversion programs for graphics data exchanged between the computers. Output from graphics software which runs on one variety of computer can be plotted on hardware attached to the other variety using simple commands. This service has the effect of making the local graphics systems on both the IBM and the VAX VMS systems machine-independent as well as device-independent.

MAIL DELIVERY SERVICE

One of the natural applications for a system which is able to transfer files is the communcation of electronic messages. The mail systems used by VAX VMS users (MAIL) and by IBM CMS users (VM/SP2 NOTE or PROFS) were connected by programs implemented on the VAX. NJE is the vehicle by which the formatted messages are transmitted. The connection to the MAIL command is automatic both for transmission and reception. The implementation of this application is described elsewhere in these proceedings (10).

DEVELOPMENT PLANS

Future directions for the NJE software include support of a better device and a more intelligent device driver. A promising alternative is the DMF32's synchronous port. That device's DMA capability would essentially remove the interrupt processing load which exists with the DUP-11. In addition, line speeds up to 19.2k baud are possible. A driver to use the device's general byte synchronous capability will be needed. Higher speeds and lower overhead will make other applications and services practical, where the present, medium data rates and overhead are not sufficient. Other interfaces will be considered as they become available.

Another goal is the completion of the NJE functionality at the VAX end. There is no support in the NJE emulator to accept jobs to be submitted to the VAX batch queues. Considerations of resource control, job ownership, and user validation will have to be studied.

SUMMARY

The VAX VMS NJE protocol emulator enables a VAX VMS computer system to participate as a peer member and as an end node in an IBM file transfer network. The software which has been developed at Argonne National Laboratory is used to attach several VAX VMS systems to a central IBM-based complex of computers under the control of JES3 and VM/370. The system has been in use at Argonne since January 1981 and is undergoing continuing development and refinement. Refinement has benefitted from the experiences of users at other sites who use different IBM operating systems and environments which have their own idiosyncrasies. At other installations VAX VMS systems are successfully connecting to JES2 and VM with the NJE emulator software.

The basic NJE networking enables users to utilize remote hardware. The general, sequential file transfer service forms the basis for a mail delivery service and a data conversion service. Device-independent graphics software in each environment have identical interfaces and with the common graphics data provides a machine-independent graphics service.

The current version of the VAX VMS NJE emulator software is being made ready for distribution coordinated by the National Energy Software Center (NESC) located at Argonne National Laboratory. The Center serves as the software exchange and information center for computer software developed under U. S. Department of Energy (DOE) sponsorship. The address is:

National Energy Software Center
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439
(312) 972-7250.

BIBLIOGRAPHY

(1) Engert, D. E. An IBM NJE Protocol Emulator for VAX/VMS, Proceedings of the Digital Equipment Computer Users Society, Vol. 7, No. 4, USA Spring 1981.

(2) Hutchinson, J. M. Job Networking Facilities, IBM Washington Systems Center Technical Bulletin GG22-9042-00, Gaithersburg, Maryland, March 1980.

(3) Raffenetti, R. C. and Zelle, B. R. Friendly Neighborhood Computer Project: Performance Measurements and Improvements for VAX VMS NJE Networking, Argonne National Laboratory Technical Memorandum 394, February 1982.

(4) Beander, B. The PME Performance Measurement and Evaluation Package, Digital Equipment Corporation, DECUS SIG tape, Fall 1979.

(5) Baker, W. E., Bertoncini, P. J., Engert, D. E., Raffenetti, R. C., Rynes, P. E., and Wei, S. Friendly Neighborhood Computer Project: Extension of the IBM NJE Network to DEC VAX Computers, Argonne National Laboratory Technical Memorandum 383, February 1982.

(6) Bertoncini, P. J., Engert, D. E., Raffenetti, R. C., and Rynes, P. E. ibid. (revised), November 1982.

(7) International Business Machines Corporation, Bulk Data Transfer: Program Description and Operations Manual, International Business Machines Corporation, SH20-2088-0, May 1978.

(8) International Business Machines Corporation, Bulk Data Transfer: Systems Guide, International Business Machines Corporation, LY20-2367-0, May 1978.

(9) Digital Equipment Corporation, VAX-11 Record Management Services Reference Manual, Digital Equipment Corporation, Maynard, Massachusetts, May 1982.

(10) Osudar, J. The NJE Mail Bridge Between VAX VMS and IBM VM/CMS Systems, Proceedings of the Digital Equipment Computer Users Society, Vol. 9, No. 4, USA Spring 1983.