THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING BEING SKEWED DIFFERANTLY FROM THE TOP OF THE PAGE TO THE BOTTOM.

THIS IS AS RECEIVED FROM THE CUSTOMER.

STANDARDIZATION OF BASIC FILE MANAGEMENT SERVICES AND I/O DATA TRANSFERS FOR HETEROGENEOUS MINI COMPUTER NETWORKS

by

DOLAN M. MCKELVY

B. S., UNITED STATES AIR FORCE ACADEMY, 1971

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

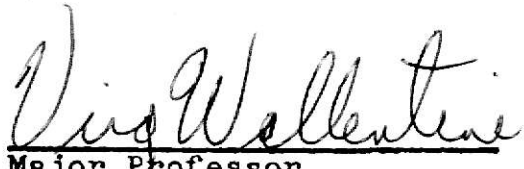
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1976

Approved by



Major Professor

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I wish to express my sincere thanks and appreciation to my major advisor Dr. Virgil E. Wallentine for his continued guidance and direction. I also want to thank the other members of my committee, Dr. William J. Hankley, and Dr. Myron A. Calhoun, for their assistance during finalization of this report.

To my wonderful wife, Pat, and my children, Amy and Shawn, I owe a very special thanks for their understanding and support throughout this endeavor.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 REPORT OBJECTIVES

This report is an initial attempt to develop standard network control blocks for basic file management services and data transfer Inputs/Outputs (I/O) for use in mini-computer networks of heterogeneous machines with real-time operating systems. One control block has been designed to handle capabilities such as creating, opening, closing, and deleting files, and another control block designed to handle the file I/O for data transfers. Transformations between the network standard control blocks and the machine dependent parameter blocks are presented. A network conversion process is modularly described. Each module in the conversion process is defined in detail using current documentation techniques. Sample hand simulation traces are used to display the designs conceptual correctness.

1.2 MOTIVATION/SIGNIFICANCE

The increasing interest and potential in computer networks coupled with technical advancements and a large variety of marketed mini-computers have created a tremendous need for standardization of file management services and associated parameters. Standardization of file management services in itself is an overwhelming task in light of the variety of computers on the market. However, by limiting the scope of possible computer architectures to basically the sixteen bit ASCII oriented mini-computers with real-time operating systems, a practical standardization of parameters

for file management services and I/O can be realized.

This research is of particular importance at the present time because of a U.S. Army research grant now underway at KSU on Functionally Distributed Computer Systems(R6). The major research effort involves design and implementation of a prototype mini-computer network of heterogeneous machines which will support a distributed data base management system. One primary concern is designing software to control inter-machine and inter-process communication. Figure 1 shows a current design overview of the inter-process communication which will be implemented as a set of modules below the local operating system on each machine. The parameter variations in file management and I/O capabilities between heterogeneous machines in the network requires some type of conversion and standardization of parameter information. This conversion process and use of standard control blocks is shown between the network user task and the sending/receiving function of the I/O request to/from the network resource controller. It is hoped that the network standard control blocks and the conversion modules and submodules designed in this report will help serve to further define that portion of the inter-process communication dealing with conversion and network standardization of file management and I/O parameter information.

The network user task 1 in Figure 1 is <u>not</u> aware of the network and associated processing. All I/O or file management services (FMS) requests are trapped by the local operating system and the conversion process as defined in

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

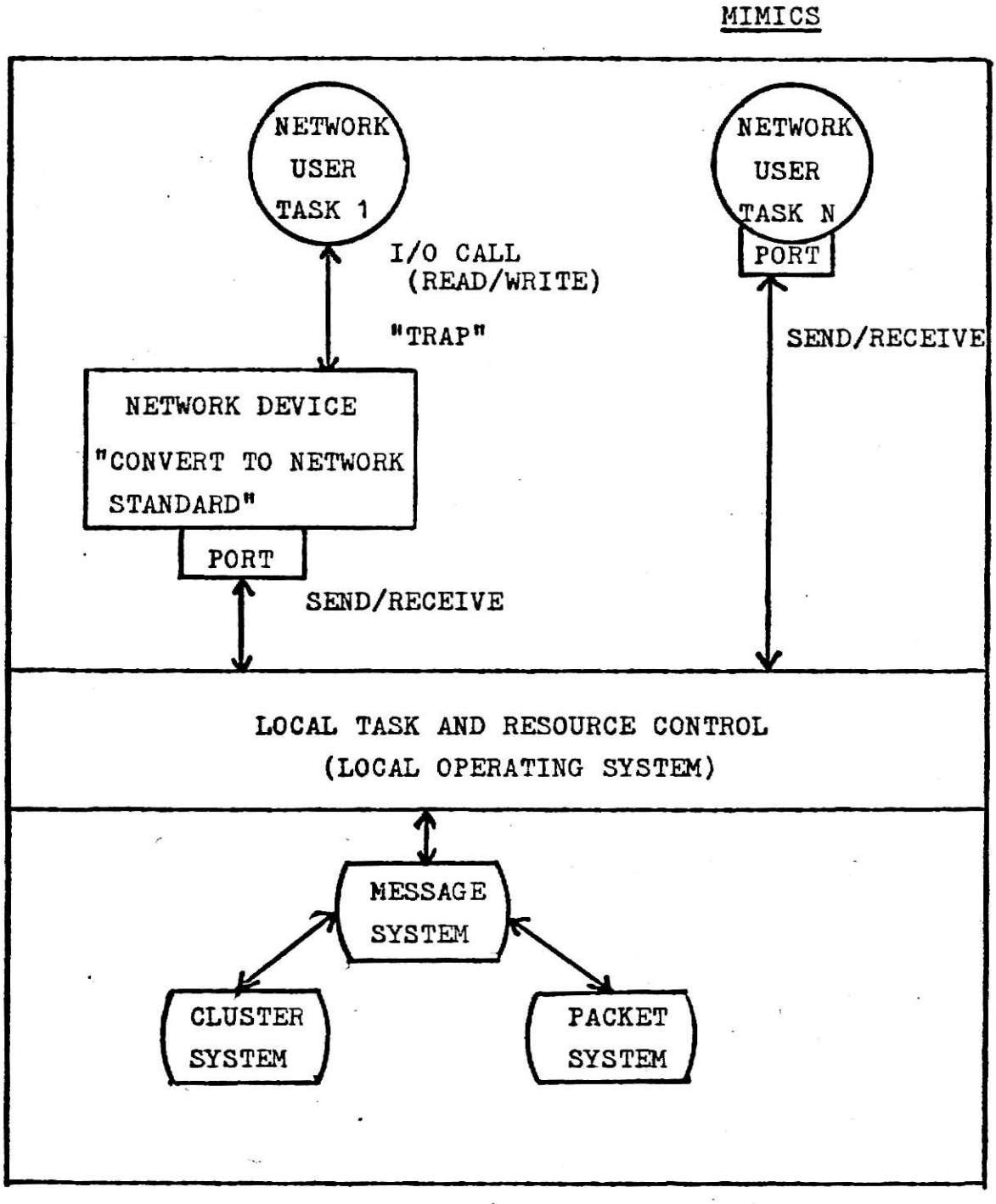


FIGURE 1:

Local Operating System Implementation of MIMICS

TASK 1---Unaware of the network.

TASK N---Aware of the Inter-Process Communications.

this report is performed. The network control block is then sent via the send/receive commands from a port through the network to the destination machine. The message system is responsible for describing the message and for sending variable length records throughout the network. The cluster system transfers accross the local network variable length records from memory to memory. The packet system breaks up messages and transports them one at a time to remote computers. Both the packet system and the cluster system are transparent to the software above them in the system. The network user in task N in Figure 1 is aware of the network and uses a network control language to eliminate the need for a conversion of the initial request into network standard form.

1.3 APPROACH TO PROBLEM

The first step taken in approaching this task was to investigate several representative mini computers (Data General Nova(R1), Interdata 8/32(R4), Interdata 7/16(R3)) and identify their file structures, file management services, I/O commands, associated parameters, characteristics, and limitations. This information was then integrated into network standard control blocks. Transformation tables were set up to handle parameter conversions between the network standard control blocks and machine dependent parameter blocks. While this detailed study was in progress, high-level modular design of the overall network conversion process was also being done. With high-level modules designed in a portable manner, only the logic for the lowest level command modules

required machine dependence.

Use of these standard network control blocks can be divided into two levels;

1. A low-level use for transformation to and from machine dependent parameter blocks so current programs not running on a network can be run without changes. This can be done by having all file operations which are trapped by the local operating system inspected by network software and handled internally if network action is required (See Figure 1).
2. A higher-level use, whereby a network control language is designed which uses the network standard directly and thus requires only transformations from the network standard to the destination machine. No machine to network conversions would be necessary except for status returns. This report is limited to providing only the lowest level type transformations; however, the network standard control blocks to be used by the higher level are developed.

1.4 REPORT STRUCTURE

This report is structured in the following top-down manner. We present the proposed network standard control block structures and define in detail all of their parameters. We then demonstrate their use through sample mappings between specific machines and the network standard control blocks. Using these transformations as a basic foundation, there is a discussion of overall network flow, describing where and

how each conversion is used. A hierarchical modular structure is developed to portray the necessary machine independent high-level modules and the machine dependent lowest level command submodules. English HIPO charts (Hierarchical Input, Porcessing, Outputs) are used to document each module. PASCAL type algorithms are used in the lowest level module HIPOs to explicitly define the conversion steps necessary. Example commands are then used to hand simulate the applicable algorithms for inter-machine mappings showing network flow and conceptual correctness. Lastly, there is a summarization of what has been accomplished thus far, a discussion of overall problems, suggested areas requiring future study, and possible approaches to any continued research.

The attached appendices contain the following information: Appendix A contains general characteristics of each machine, along with the actual transformations between each command and the network standard control block. Appendix B contains the machine dependent parameter blocks with the descriptions and settings for each parameter. Appendix C further defines each machines parameter block by command, showing what specific parameters are used for each command. Appendix D is a mapping between machine dependent file structures. Appendix E defines the lowest level command modules in HIPO form for the individual commands on each machine.

2. PROPOSED NETWORK STANDARD CONTROL BLOCKS

This section contains network standard control blocks to help standardize file management services and I/O capabilities. A general overview of concepts involved with designing the control blocks is presented to familiarize the reader with some of the design considerations. We then portray and briefly define each resulting network control block in Figures 2 and 3. Detailed descriptions of all associated parameters are given in Tables 1 and 2. Lastly, the transformations in appendix A are used as examples to show exactly how to interpret them.

2.1 CONCEPTUAL CONSIDERATIONS IN CONTROL BLOCK DESIGN

Our initial project definition called for a network control block to contain the intersection of compatible capabilities between machines in the network. However, a detailed study revealed very few straight forward compatibilities. The idea of emulating certain basic capabilities was discussed but later dismissed because of its restrictive nature. Rather than designing a standard control block with limited emulation capabilities, it was decided to design a general control block containing the parameters required for a union of all network machine capabilities. This would then provide for complete sharing and utilization between homogeneous machines in the network, while at the same time isolating distinct differences between architectures. Later, emulation could then provide additional capabilities on a needed basis for specific commands and machine types. In

addition to this major change in design philosophy, other information such as a machine type indicator and machine dependent status are retained in the control block to allow for more complete error analysis, possible recovery action, and increased processing speed.

2.2 NETWORK CONTROL BLOCK DESCRIPTIONS

As a result of extensive research, integration of similar parameters, and the conceptual philosophy used, the network standard control blocks described in Figures 2 and 3 were designed. They provide for a union of those capabilities presently existing on the three mini-computers studied. General mnemonic descriptions accompany each Figure, and detailed descriptions are given in Tables 1 and 2.

NFMB

NFMB-CODE (1)	NFMB-MD__ (1) AP 3 BF 2 FT 3	NFMB-STNT (1)
NFMB-STAT (1)	NFMB-LUNM (1)	NFMB-PKEY (2)
NFMB-SZFL (1)	NFMB-SZNX (1)	NFMB-LRCL (2)
NFMB-MTPS (1)	NFMB-MTPD (1)	NFMB-FDDV (3)
NFMB-FDVL (4)	NFMB-FDDR (20)	NFMB-FDFN (10)
NFMB-FDEX (3)	NFMB-FDVR (1)	

FIGURE 2:

Network Standard File Management Control Block (NFMB)

Mnemonics

NFMB-CODE	Specifies desired File Management command.
NFMB-MD__	Modifications for access, buffering, and file type.
NFMB-STNT	Network standard status response.
NFMB-STAT	Destination machine dependent status obtained.
NFMB-LUNM	Logical unit number assigned to file/device.
NFMB-PKEY	Read/Write protection keys.
NFMB-SZFL	File size parameter in bytes.
NFMB-SZNX	Index block size in bytes.
NFMB-LRCL	Logical record length in bytes.
NFMB-MTPS	Machine type of source machine.
NFMB-MTPD	Machine type of destination machine.
NFMB-FDDV	Device portion of file descriptor.
NFMB-FDVL	Volume portion of file descriptor.
NFMB-FDDR	Directory portion of file descriptor.
NFMB-FDFN	Filename portion of file descriptor.
NFMB-FDEX	Extension indicating content of file.
NFMB-FDVR	Version number of file described.

Table 1 (Page 1 of 2)

NFMB DETAILED MNEMONIC DESCRIPTIONS

NFMB-CODE	File management service code, 1--CREATE 2--OPEN/ASSIGN 4--CLOSE 8--DELETE
NFMB-MD__	The modification argument, AP--Access Privileges, 0--Shared RD/WR 1--Shared RD, Exclusive WR 2--Shared RD only 3--Shared WR only 4--Exclusive RD/WR 5--Exclusive RD, Shared WR 6--Exclusive RD only 7--Exclusive WR only BF--Buffering Provided, 0--System supplied 1--User supplied FT--File Type, 0--Contiguous 1--Chained (sequential) 2--Indexed (random)
NFMB-STNT	The network standard status response, 0--Successful completion of request. 1--Illegal function/filename. 2--Illegal logical unit/channel. 3--Volume error. 4--File non-existent on volume. 5--File already exists. 6--Protection error. 7--Access conflict--request denied. 8--File descriptor error. 9--Logical unit number assignment error. 10-Size error. 11-Device error. 12-Other--Not a common network error. 13-NO-MAPPING condition at destination machine.
NFMB-STAT	Actual destination machine dependent error status.
NFMB-LUNM	Logical unit number associated with file/device.
NFMB-PKEY	Protection write and read keys used to protect access to the files, 00--General access allowed XX--Requires exact match
NFMB-SZFL	Size parameter specified in bytes. Contiguous file--number of bytes/file. Chained file-----number of bytes/block. Indexed file-----number of bytes/data block.

Table 1 (Page 2 of 2)

NFMB-SZNX	Index block size---number of bytes/index block.
NFMB-LRCL	The number of bytes/logical record, or certain machine dependent information.
NFMB-MTPS	Machine type of source machine. 1---Interdata 8/32 2---Interdata 7/16 3---Nova 4-256---Other architectures
NFMB-MTPD	Machine type of destination machine with the same code values as NFMB-MTPS.
NFMB-FDDV	The device name portion of the file descriptor.
NFMB-FDVL	The volume on which the desired file resides.
NFMB-FDDR	The directory for the desired file.
NFMB-FDFN	The filename string of the file descriptor.
NFMB-FDEX	The extension portion of the file descriptor, indicating the content of the file.
NFMB-FDVR	The file version number portion of the file descriptor.

NIOB			
	NIOB-CODE (1)	NIOB-FUNC (2)	NIOB-STNT (1)
	NIOB-STDD (1)	NIOB-STDI (1)	NIOB-LUNM (1)
	NIOB-SADR (4)	NIOB-EADR (4)	NIOB-RREC (4)
	NIOB-NMBT (4)	NIOB-MTPS (1)	NIOB-MTPD (1)

FIGURE 3:

Network Standard I/O Control Block (NIOB)

Mnemonics

NIOB-CODE Specifies type of I/O request.

NIOB-FUNC Describes I/O function and characteristics.

NIOB-STNT Network standard status response.

NIOB-STDD Device dependent status obtained at destination machine.

NIOB-STDI Device independent status from destination machine.

NIOB-LUNM Logical unit number designating file for I/O.

NIOB-SADR Starting byte of the data buffer.

NIOB-EADR Last byte of the data buffer.

NIOB-RREC Random record number (0 relative).

NIOB-NMBT Number of bytes in transfer.

NIOB-MTPS Machine type designator for source machine.

NIOB-MTPD Machine type designator for destination machine.

Table 2 (Page 1 of 2)

NIOB DETAILED MNEMONIC DESCRIPTIONS

NIOB-CODE I/O operation code, 0--Data transfer
≠ 0--I/O command (REWIND, etc)

NIOB-FUNC I/O function code,

BIT		
0	Function---	0--Write 1--Read
1-3	Access Mode---	0--Sequential, current pointer 1--Random, use record number 2--Line, ASCII line 3--Free Form, variable size 4--Direct Block, Contiguous BLK
4	Format Type---	0--ASCII 1--Binary
5	Formatting ---	0--Yes, by device/file, bit 4 1--No
6	I/O Wait---	0--Proceed processing 1--Wait for I/O completion
7	I/O Connect---	0--Wait for connect 1--Immediate reject if busy

NIOB-STNT The network standard I/O status.

0--Successful I/O completion.
1--Illegal channel.
2--Illegal request.
3--File not opened yet.
4--Protection conflict.
5--End of medium.
6--End of file.
7--Out of space.
12-Other--Not a common network error.
13-NO-MAPPING condition at destination machine.

NIOB-STDD Actual device dependent status obtained at the destination machine.

NIOB-STDI Actual device independent status obtained at the destination machine.

NIOB-LUNM Logical unit number designating file for I/O.

NIOB-SADR Starting byte of the data buffer where data is to be put or taken from.

NIOB-EADR Ending byte of the data buffer whose address is in NIOB-SADR.

Table 2 (Page 2 of 2)

NIOB-RREC	The random record number to be used for random I/O. It is zero relative.
NIOB-NMBT	The number of bytes to be transferred or already transferred.
NIOB-MTPS	The machine type of the source machine. 1---Interdata 8/32 2---Interdata 7/16 3---Nova 4-256---Other architectures
NIOB-MTPD	The machine type of the destination machine with the same types as NIOB-MTPS above.

2.3 TRANSFORMATION TABLE ANALYSIS USING EXAMPLES

In this section we present the mapping between each mini-computer's unique calling parameters and the generalized standard blocks. Appendix A contains these transformation mappings ordered by machine and command. To show exactly how these mappings work, we use the following examples to specify a mapping direction and use the transformations from Appendix A for the Nova OPEN and CLOSE commands, the Interdata 8/32 I/O command, and the Interdata 7/16 CREATE command to describe the mapping with the machine dependent parameter blocks in Appendix B.

EXAMPLE #1--A Nova OPEN command going from machine dependent to network standard.

The transformation table shows an OPEN is represented by NFMB-CODE = 2. The Nova has three separate OPEN commands each describing different access privileges. If .OPEN is used, then it is converted to the network standard by setting NFMB-MDAP = 0 to indicate an access privilege of shared reading and writing. If .EOPEN is used, the network standard for shared reading and exclusive writing is one, so NFMB-MDAP is set to one. For .ROPEN, NFMB-MDAP = 2 indicates shared reading only is requested. Since there are no other access privileges on the Nova, nothing maps into the additional network standard access privileges indicated by values from three to seven. The channel number specified in the OPEN command is placed directly in NFMB-LUNM. The filename and extension, which are pointed to by accumulator zero, are

placed into their perspective locations in the NFMB. The content of accumulator one, which would only be used by another Nova, is placed into NFMB-LRCL in case the destination machine is a Nova. Having completed these assignments to the NFMB, the conversion to network standard is completed for the Nova OPEN command, and the NFMB is ready for shipment to the destination machine.

EXAMPLE #2--An Interdata 8/32 I/O command from machine dependent to network standard.

According to the transformation table, when the Interdata 8/32 function code has bit zero = 0, then NIOB-CODE is set to zero to indicate a data transfer I/O. The other bit settings of the Interdata 8/32 function code are then placed into their appropriate counterparts in the network standard I/O block. Bits one through three of the NIOB-FUNC byte have additional settings into which the Interdata 8/32 cannot be mapped since it does not support line I/O, Free Form I/O, or Direct Block I/O. The logical unit number is placed directly into NIOB-LUNM and the starting and ending buffer locations are also loaded directly into the NIOB. If the I/O is random, then the random record number is placed in NIOB-RREC. In order to conform to other network standard parameter requirements, the number of bytes to be transferred is computed by subtracting the starting buffer address from the ending address and placing the result into NIOB-NMBT. Once these assignments are made, the NIOB is ready for shipment to the destination machine.