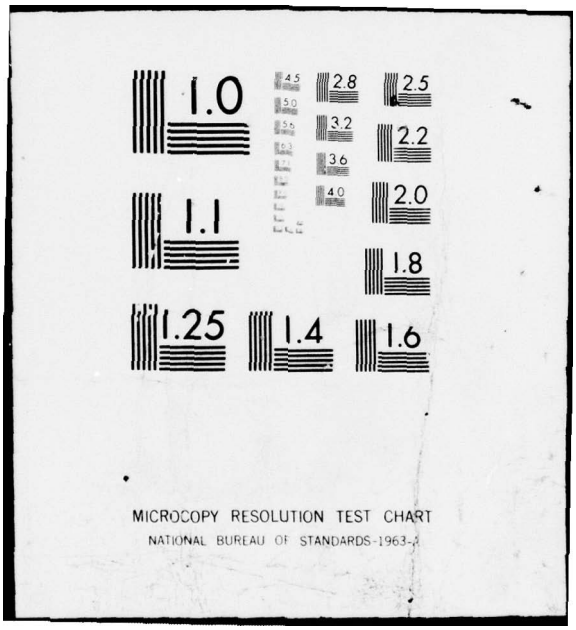
AD-A044 727 MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE F/G 9/2
THE DISTRIBUTED COMPUTER NETWORK PROJECT.(U)
JUL 77 D L MILLS N00014-67-A-0239-0032
UNCLASSIFIED TR-555 NL

| OF |
AD
A044 727

END
DATE
FILMED
10-77
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

COMPUTER SCIENCE
TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

Technical Report TR-555 July 1977

FINAL REPORT ON THE DISTRIBUTED COMPUTER NETWORK PROJECT

by

David L. Mills

This research was supported in part by Grant N00014-67-A-0239-0032 from the Office of Naval Research and by the Department of Computer Sciences, University of Maryland.

Abstract

This report summarizes the work done during the last year of the Distributed Computer Network (DCN) Project. The accomplishments during this period include further refinement of the operating systems and compilers, development of automatic fault-recovery and resource-assignment facilities and re-evaluation of certain concepts indigent to the DCN architectural design.

Table of Contents

Table of Contents

FINAL STATUS REPORT ON THE DCN PROJECT

This is the final status report on the DCN project, supported under ONR contract N00014-75-C-0872 for the period 1 June 1976 through 31 May 1977. The project has been supported during this period as a no-cost extension of last year's funding. The activities reported herein reflect the progress toward the goals set forth in the contract proposal and in previous status reports.

A current list of publications supported at least in part by the DCN project is appended. The list, in chronological order, includes only those publications appearing as formal technical reports and journal articles. At this time all of them, with the exception of the last three (14, 15, 16) have appeared in their respective proceedings. The last three have been microfilmed, but will not be produced in paper form, due to budgetary constraints. The second report (15) on the ULP language has been submitted for journal publication.

It is expected that at least one more technical report will be produced, in addition to this report. This report will describe the current state of the software and serve as a reference document for any possible further work. If time permits this report will be condensed and submitted for publication.

1. Goals and Objectives

The goals specified for the remaining period of the contract were contained in the status report of 26 September 1975 as amended by the letter of 3 February 1976. These include:

1. Continued development of the Virtual Operating System (VOS) and Basic Operating System (BOS) for the DCN to at least a point where its performance could be assessed relative to competing technologies.

2. Selected experimental DCN developments involving distributed resource allocation/control and fault recovery.

3. Investigation of high-level language features useful in the distributed environment.

4. Examination of various approaches to DCN implementations and evaluation of their performance and utility, where possible using the DCN experience as a guide.

The progress toward these goals since the last status report

is reported in this document. There are three parts: the first describing development of the DCN software to enhance its use as a measurement and evaluation tool, the second describing certain experimental resource management techniques and the third describing the results of certain experiments designed to test the principles of the DCN design. Discussion of a number of key points in the design of DCN-like networks and evaluation of a number of alternatives in implementation are contained in the summary and conclusions.

2. DCN Software Development

The DCN software has been continuously refined in minor ways in order to improve performance and provide features necessary for experiments. This activity is described in following sections.

2.1. VOS Activity

From the outset of the work on this contract, the notion of using portable virtual segments as the (only) intrinsic data type was thought to be of primary importance. Although a single-computer operating system embodying these concepts was operational as early as 1972, the integration of the virtual segment architecture into the network environment has proceeded very slowly since then. Part of the reason for this has been limited manpower and key personnel departures, and part has been the inherent difficulty in developing system components such as compilers, linkers, etc., to operate in this environment.

The VOS development effort during the past year has been minimal and aimed primarily at maintaining compatibility with the BOS in the areas of device support, network interface and user command operations. Specific activities are summarized in the following sections.

2.1.1. VOS Emulation

It was determined at an early stage that development of a complete program-preparation capability including text editors, assemblers, compilers and linkers was impractical with the resources available to the project. Instead, a virtual-machine emulator was proposed in which one of the DEC operating systems cold run with little or no modifications. Unfortunately, the structure of the system chosen, the Disk Operating System (DOS), turned out to be such as to make this strategy unworkable. This

was one of the reasons why Dr. John Shore of NRL withdrew his support from the project in 1975.

Digital Equipment Corporation has since replaced DOS with a new system called RT-11. This development, together with the possibility of using other systems, such as UNIX, have prompted a new evaluation of the virtual emulator approach. An LSI-11 computer and RT-11 system was made available to the project (at no cost to the project) in order to test this approach. From persual of the available documentation, including system manuals and source code, it appears that the objections originally raised in connection with DOS have been removed in RT-11.

There are two ways that RT-11 system components could be run in the VOS:

1. Background load module (.SAV) files could be mapped directly onto VOS segments at corresponding virtual memory addresses. This would require VOS emulation of direct-access input/output transfers (which are always called through the supervisor-interrupt interface) in much the same way as originally proposed for DOS. The advantage that RT-11 has over DOS is that the construction of the various system components is much more disciplined and violations such as appeared in DOS FORTRAN have been avoided. In addition, it seems to be practical to incorporate much more of the monitor-level software, such as portions of the keyboard monitor (KMON) and User Service Routine (USR), directly into the emulator, thus diminishing the total work of implementation.

2. Object module (.OBJ) files could be linked with an interface package to run directly in VOS without emulation. In fact, these files can be linked and run in the Basic Operating System (BOS) for the DCN using the same technique. A number of system components of RT-11 and other systems are already supplied in object-module form, among them the Paper Tape Software (PTS) linker and the RT-11 BASIC interpreter/compiler, and others are readily available. By replacing or modifying the input/output subroutines for these components it should be possible to avoid the necessity of an object-time emulator. Modules to operate in the VOS could be linked by the RT-11 linker in absolute load module (.LDA) format and loaded by means of the existing VOS/BOS loader onto a VOS virtual-segment file. The program could be run by simply mapping the segments of this file at corresponding virtual addresses. Modules to operate in the BOS could be linked by the RT-11 foreground relocatable (.REL) format, in which the relocation information is preserved. Through simple modifications of the existing VOS/BOS relocating loader, it

would be possible to dynamically load the run the program in the same way that cross-assembled/compiled programs are now run.

2.1.2. VOS File System

A further development planned for the VOS was a comprehensive virtual file system. For various reasons Dr. Lay and Mr. Pollizzi were unable to complete the implementation of the virtual file system according to the original design. However, various intermediate versions of this file system have become operational, although not with all the features planned.

In order to facilitate program development and networking experiments, the RT-11 compatible BOS file system is presently in use in the VOS. Its replacement by the developmental virtual file system is a matter only of re-linking the VOS.

2.2. BOS Activity

As explained in the last status report, the Basic Operating System (BOS) for the DCN was developed in order to facilitate experiments in network architecture and intercommunication. The intent was, and is, to maintain compatibility with the VOS in all areas except those for which the dynamic virtual-storage mapping facilities of the VOS are required. In order to reduce the resources required for development and maintenance it was specified that both the BOS and VOS were to use as many routines in common as practicable. However, it was also specified that the BOS was to operate with no restrictions on capability in all PDP11-compatible machines, including the LSI-11, while the VOS was to operate only in PDP11 machines including hardware relocation facilities of the PDP11/45 or equivalent.

In connection with certain experiments described later it became apparent that a third system would be highly desirable in addition to the BOS and VOS just mentioned. This system, actually a minor modification of the BOS, would provide an extension to the BOS address space by using the hardware relocation features of the PDP11/40 or equivalent. This extension, called the relocatable version of the BOS, provides a static relocation of each user process address space in blocks of 4K words. It is implemented as a conditional assembly of the BOS supervisor - no other system modules are affected.

The net result of these extensions has been to create a spectrum of three upwardly compatible operating system capable of operation in the distributed network environment. All three systems provide the same interface for interprocess communication, application-program interface and resource allocation. The relocatable version of BOS provides an expandable virtual address space organized as resident fixed-size segments for each user process. The VOS provides this feature and, in addition, automatic swapping between main and auxilliary storage of variable-size segments.

2.3. Application Program Interface

An application program interface has been designed and implemented for the BOS. The same interface can be used by the VOS, although a more sophisticated implementation would probably be more useful. The interface provides a mechanism to load and execute relocatable object modules produced by the Univac cross software and to interact with the DCN environment.

In order to support application programs the BOS user process provides an address-space partition that extends downward from the initial process stack pointer. The size of this partition is fixed in the absolute version of the BOS and can be increased in blocks of 4K words in the relocatable version of the BOS A relocatable object module can be loaded from any suitable network device or file, including Univac files (using the appropriate data transmission program). The module is loaded starting from the bottom of the partition (or elsewhere as specified). The region from the current stack pointer to the top of the loaded module can be used by the loaded program as necessary or, alternatively can be used as a new subpartition in which to load another program

This structure was designed to support dynamic overlays organized as a tree-structured set of segments. Current DCN software, including the Command Language Interpreter, File/Device Interchange Program, BASIC Interpreter and ULP Compiler are supported in this fashion. Any of these programs can be linked into the resident re-entrant system and shared by all of the user processes or, alternatively, can be dynamically loaded into a partition in one or more user processes separately.

A special feature allows linking to global sysbols defined by the resident re-entrant system to be resolved when a module is dynamically loaded. This is accomplished using a special system symbol table which can be extended dynamically during system operation. This feature is designed to provide linkage to common packages such as the FORTRAN Object-Time System, avoiding duplication and redundancy in partition and file storage.

2.4. Univac 1100 System Support

Several modifications to the DCN software in the Univac 1100/40 and 1108 were required to support the other activities described in this report. These include minor modifications to the cross assembler, link editor and data transmission programs.

2.4.1. Cross Software Modifications

The cross assembler was modified to include the special LSI-11 instructions and LSI-11-PDP11/40 floating-point instructions. All of the 1100-resident software was modified to use the new Univac-supported Omnibus elements, which include highly desirable features for cross-support software such as this.

2.4.2. Data Transmission Programs

New data transmission programs were constructed to provide much more secure transmission between the 1100 machines and the DCN network. It is interesting to point out that most of the failures encountered in data transmission between the 1100 and the DCN were not traceable to errors in the data transmission process itself but due to various 1100 hardware and software anomolies. As a practical matter the 1100 systems crash much more often than the DCN and transmission can be garbled by such things as operator messages to normal (human) terminals. It has been an extremely bothersome chore to gain relief from these problems by means of clever software in either or both the 1100 or DCN systems.

A measure of relief was achieved by means of special DCN data transmission programs written by Hans Breitenlohner of the Computer Science Center and modified by Mr. McMullen. These programs, which were invaluable during the process of transcribing the DCN data base from 1100 files to RT-11 files, cleverly filters out 1100 operator messages and codes the data according to the DCN standard described elsewhere. A useful if somewhat amusing sidelight to the operation of these programs is that operator messages appear in the form of audible Morse Code in systems which have only a single asynchronous serial line unit for both the operator's console and data transmission interface.

In another connection Mr. Breitenlohner extensively modified

the Univac C/SP software so that high-speed (7200 baud) synchronous remote-batch connections can be made under program control to both Univac systems now in use and greatly improved the error recovery characteristics of these systems and the DCN. In addition, a means has been provided to encode ASCII upper/lower case transmission between the Univac machines and the DCN, so that the high-speed upper/lower case Printronix printer connected to the DCN can be used as a remote 1100 print station. This printer, which was acquired from Department funds, was in fact used in this manner to print the latest reports ennumerated on the list of publications of the DCN project (see Appendix).

2.4.3. SIMPL-X1 Modifications

A small project carried out jointly with Dr. Richard Hamlet (assisted by Mr. McMullen during the last funding year) was concerned with the use of the SIMPL programming language in the user process for application programming. For reasons ennumerated below, it has not been possible to use SIMPL in the DCN software itself. However, Dr. Hamlet is now refining a new version of the SIMPL compiler in which some of the objections are removed. One of these is the production of relocatable object modules, a project which involved substantial changes to the code-generator portions of the compiler.

Unfortunately, it has not been possible to contribute support to Dr. Hamlet's work during the reporting interval (except for judicious arm-twisting of Mr. McMullen, who is not now supported on the DCN project) and the availability of the enhanced compiler was too late to allow evaluation in the DCN environment. It was the intent of this exercise that SIMPL-compiled programs could be compiled and linked to operate in both the BOS and VOS, in much the same way that assembler and ULP-compiled programs are now. Subject to bug extermination, it should be possible to do this soon.

2.5. System Support

During the last few years the development of inexpensive PDP11-compatible processors such as the LSI-11 and operating systems such as RT-11 have greatly improved the outlook for maintenance of DCN software directly in the PDP11 environment, rather than in the cross-software environment. The minor but annoying inconsistencies between the Univac cross assembler and linker and other PDP11 assemblers and linkers have excaberated the situation. Accordingly, the entire DCN data base, including all

source modules, has been transcribed to floppy disks and adapted to suit the RT-11 macro assembler and linker. An absolute load module (.LDA) can be conveniently constructed for any of the existing PDP11-compatible processors and most peripherals using RT-11 facilities. The DCN load module can then be loaded from RT-11 or by the standard initial-program load procedure. DCN application programs have full access to RT-11 files and can create and update these files as well. The equipment and operating system used for the transmission and conversion effort was provided without charge to the contract.

3. DCN Network Development

A number of additions and modifications to the set of common modules used both by the BOS and VOS have been made to provide features necessary for DCN operation as an integrated network. These include expanded and refined peripheral device support, an RT-11 compatible file system, automatic routing for the intercomputer communication subsystem, a form of automatic resource allocation for the process management subsystem, an application program interface compatible with the ULP compiler and a number of features providing transient-fault recovery and automatic reconfiguration.

3.1. Device Support

The number of types of devices supported has been increased in include the AED 6200 Floppy Disk, together with a special Morse Code/Radioteletype interface used in connection with a Master's thesis project. The device support modules for the interprocessor links (300-baud asynchronous, 7200-baud synchronous, Unibus Link) have been modified to provide channel-failure detection and reconfiguration. The 7200-baud synchronous link, presently connected through the Univac CS/P to both the 1100/41 and the 1108, was modified for upper/lower case operation and to provide better error detection and correction capabilities. The command language used to specify various device options and parameters was standardized and integrated into the resource assignment subsystem (see below). The direct-access disk modules for the fixed-head disk, moving-head disk cartridge and floppy disk were reconfigured to interface with a number of file systems, including the RT-11 compatible file system and the VOS virtual file system.

3.2. RT-11 Compatible File System

In order to obtain access to the many programs and system components available in the RT-11 Operating System, a file

subsystem was constructed which provides the capability to read and write RT-11 files. It was realized that the RT-11 file structure, which is contiguous by block, would not be appropriate for the shared multi-user access environment envisaged for the DCN. On the other hand, means for interfacing with the RT-11 system was deemed imparative if the DCN operating systems were to be eventually supported under RT-11, as in fact they now are. In any case, the VOS virtual file system was intended to be the principal (but not necessarily the only) file system available on the network. In fact, with the present designs, both the RT-11 and virtual file systems can co-exist on the same machine.

The RT-11 file system has been used to store data gathered on the DCN network for later processing by programs running on the RT-11 system itself. These data include all of the source text of the DCN system itself (about a megabyte of text) and occasional object modules produced by the Univac 1100 Cross Assembler and Cross Link Editor. BASIC programs have been exchanged between the BOS and RT-11 systems along with various data used in real-time satellite tracking and telemetry reduction in connection with the AMSAT OSCAR satellites.

3.3. Automatic Message Routing

In connection with the automatic reconfiguration facilities provided by the catalog subsystem, described presently, the interprocess message transmission subsystem was revamped to provide automatic channel failure detection and reconfiguration. This facility, which was surprisingly easy to install, includes the following:

3.3.1. Message Timeouts

As an option, any process waiting for a message or a reply to a previously sent message) can initiate a timeout. If no message (or reply) has been received during the specified interval, the message wait is terminated and the process is free to initiate recovery procedures. Most of the device service processes now make use of this feature to detect channel or device failures. User application programs can also make use of this feature.

3.3.2. Hello Messages

Device support processes at each end of every intercomputer link periodically send a hello message to their correspondent in the other machine. This causes the correspondent process in the other machine to send a reply containing the routing table to the originating machine. This reply is used to, among other things, initialize entries in the routing table of the originating machine. The entries in the routing table include both a link number (port address) to be used to send messages to another hostel and a "cost" or transmission delay estimate (for details see below).

Each time a reply is not received within the specified timeout interval (presently a few seconds) the cost is incremented. When a reply is received the corresponding link-number entry is updated and the cost recomputed. Using this information the interprocess message transmission subsystem attempts to choose the best (lowest cost) path, depending upon the instantaneous configuration and loading of the system.

3.3.3. Automatic System Catalog Update

Recalling from previous DCN publications, each hostel includes a system catalog, which contains the names and port addresses of all other port dictionaries and user catalogs in the network. Each hostel also includes a port dictionary, which contains the names and port addresses of the various processes running in that hostel, and one or more volume dictionaries which contain the name and location of each file on the corresponding direct-access volume in that hostel. The entries in these catalogs and dictionaries reflect the current configuration and access capabilities of the network and obviously must be preserved if parts of the network fail.

Periodically, each port dictionary process sends its name and address to the system catalog of every available hostel in the network, as indicated by the routing table. Present procedures involve a round-robin broadcast of this information with a period of a few seconds for each hostel in turn. The system catalog is cleared when a failed hostel is restarted, and an entry which is not renewed periodically by a broadcast from the corresponding port dictionary is removed. Thus, the entries in all system catalogs converge to identical values rapidly (within a minute or so for an eight-hostel network) after an on/offline transition on

the part of any hostel.

3.4. Resource Assignment

The DCN architectural design calls for management of all system resources by means of processes. This includes all accesses to the system data base, input/output devices, and user processes, in which all application programs are run. Thus, resource assignment and control is mechanized by means of interprocess messages, and assignment of a resource to a process amounts to the possession of the port address of the process controlling that resource.

All DCN processes are named according to a convention which assigns each process to a set sharing in common a set of attributes. For the most part these naming conventions follow the convention used in many of the DEC operating systems. In some cases, in particular the user process, the names reflect the characteristics of the machine hardware, such as whether hardware relocation or floating-point features are available and the size of the storage available (where applicable).

A user process accesses a resource by attempting to assign a port address of some process controlling that resource to a logical channel. This action sends a message to the system catalog for the hostel of the requesting process, thence to the port dictionary of the hostel of the resource-controlling process and finally to the resource-controlling process itself. A confirming reply message is returned directly to the requesting process, which then interacts directly with the resource-controlling process. Upon termination of these interactions, a disconnect message is sent to the resource-controlling process, after which it is again available for new requests.

At any time there may be a number of resource-controlling processes of a given type which can provide the same service, as indicated by the naming convention. For instance, there may be several available BOS and VOS user processes, perhaps on different machines, and some of these may run on processors with floating-point features and some may not. A user can specify which characteristics are required and which are "don't care" by including default indicators at appropriate points in the resource-controlling process name.

When a resource-controlling process is created (and at certain times thereafter) it sends its name and port address to the port dictionary of the parent hostel. The message containing this information includes a "cost", which is an arbitrary number

assigned either a-priori or computed by the process itself. This cost, which includes components due to the size of the memory partition assigned to the process and whether it is exclusively assigned to some user, is stored for each process in tables maintained by the port dictionary and can be accessed when a decision is made to allocate a particular process from a set of those providing similar services. In a similar fashion a cost assigned to the port dictionary itself is transmitted in the hello messages broadcast by the port dictionary to the set of system catalogs. This cost is intended to reflect the cost of using the processor and includes components due to the processor speed, total memory size, etc.

A request for initial connection to a resource-controlling process is sent to a system catalog process. The message contains a field indicating the maximum cost allowed by the user. The system catalog process searches its tables for the first occurance of a name which matches the one specified, with due regard for the "don't care" and defaults. Since only port dictionary names and user catalog names occur in these tables, this amounts to initial selection of a hostel in which to search for the process or file name itself. The hostel-name field of the requested process name is usually defaulted, in which case the search starts with the port dictionary process of the same hostel as that of the requesting process. If this field is specified then only the hostel indicated will be considered further.

The choice of port dictionary used to continue searching for a requested process is determined as that for which the sum of the cost of the port dictionary process (from the system catalog tables) and the cost of communication (from the routing table) is a minimum. When the choice is made the original request message is sent to the port dictionary process and is processed further there.

The port dictionary process searches its tables for the least-cost entry corresponding to the set of constraints implied in the requested process name, subject to the additional constraint that the total cost (system catalog + communication + port dictionary) is less than the maximum allowed by the user. If such an entry is found, a special control message is sent to the resource-controlling process. This message, which can include additional information as provided by the user, is processed by the resource-controlling process and either accepted, in which case the confirming reply message is routed through the port dictionary (to update the cost) and to the requesting process, or rejected, in which case the rejection reply is sent directly to the requesting process.

If no satisfactory entry is found by the port dictionary, a

rejection message is returned to the system catalog which requested the search. The system catalog resumes its own search for a suitable hostel, shoosing the next least expensive hostel, and again sends a search request if one is found. If no suitable hostels remain to be searched a rejection reply is returned to the requesting process.

It should be clear from the foregoing that this algorithm, which is used for all resource assignments in the DCN, cannot always yield access to the resource of least cost. This is due to two reasons. First, the cost and availability of resource-controlling processes, port dictionaries and communication channels is dynamic and can change with time. Second, the algorithm does not search all the port dictionaries, since it returns the address of the first resource-controlling process found in an ordered search constrained by selection criteria and maximum cost. Some of these apparent defects are due to the inherent difficulties in dealing with the nondeterministic environment in which the DCN is supposed to flourish, while others could be removed or at least diminished through refinements in the design. For instance a weighted-sum cost calculation specified by the user would allow better control of the allocation process. In another refinement a parallel search could be undertaken at the port dictionary level (although this would be expensive in communication channels and processing time). All things considered, although the existing heuristic algorithms could well be improved, their performance is adequate and they are compactly implemented. In fact, the system catalog and port dictionary processes share well over three-quarters of their procedure code.

3.5. An Experiment In Morse Code Decoding

A low-key investigation which has been going on for some time by the author and some of his students, is involved with real-time decoding of manually-sent Morse signals on high-frequency radio circuits. The activity has generated a number of experimental systems supported within the DCN framework. The latest version, described in [16], is implemented as a standard DCN device support routine and interfaces with the remaining system components in the same manner as other operator terminals. The techniques used in the design of the latest decoder are advanced and novel in their own right, as evidenced by an informal conference with personnel of the National Security Agency, and may be useful to the intelligence community. However, the value to the DCN effort lies in the lessons learned by the experimental dispersion of resources to handle the nontrivial computations involved.

The Morse decoding process can be conveniently separated into two functional modules: one to perform the digital sampling,

real-time filtering and detection functions, and the other to process the resultant intervals estimate the parameters and decode the characters. In one of the experimental systems an LSI-11 hostel and analog/digital interface were connected directly to receiving equipment locatd at the author's home. The real-time module was incorporated as a device process under the BOS in this machine and connected via a 300-baud telephone line to the decoding module, which ran as a BOS user process in one of the Department's PDP11/45 machines. The decoded output then was sent back to the LSI-11 for printing.

The main results of this experiment served to validate the real-time processing capability and data-link multiplexing in the DCN and to demonstrate the portability of the user process. The decoding module was transmitted between the LSI-11 and the PDP11/45 machines, loaded and run without change.

3.6. Experiments in Scheduling Algorithms

In a 1976 memo Mr. Ray Bryant and Mr. Robert Budd suggested using the DCN as a testbed for certain experiments in scheduling algorithms. During the Summer of 1976 these experiments were carried out (supported by another contract) with a DCN configuration using three PDP11 hostels. The experimental system was implemented in BASIC and required certain minor changes in the BOS supervisor, as described previously. The findings were reported by Dr. Agrawala in: Final Report on NASA Contract NAS 5-22878 (February 1977).

3.7. Experiments in Performance Evaluation

Also in Summer 1976 Mr. Jeff Mohr used the DCN to evaluate certain performance characteristics of the Univac 1100-Series machines. In his experiment, supported by another contract, a BASIC program ran in one of the Department's PDP11/45 machines connected via telephone lines to one of the Univac systems. The program simulating a demand terminal, sent timed probe messages to the Univac system and measured the responses. The findings were reported in a number of places, including: Agrawala, A., Mohr, J., and Flanagan, J. The error analysis of Exec 8 log tapes. Proc. CPEUG Conf., San Diego, November 1976.

3.8. The ULP Language

With 20-20 hindsight one can hardly resist sniping at the DCN construction technology, which is based in assembly language. Although there was considerable feeling among the participants

that a suitable high-level language should be used, the decision to use assembly language was based on necessity, since no other direct or cross-support software was available. Later, the desire to provide a compact system kernel that could be easily reconfigured for a number of PDP11-compatible machines, including the LSI-11, encouraged continued use of assembly language.

Although the cross-compiler SIMPL has been available for some time, it has not been used extensively for DCN software. This is due to three reasons:

1. Repeated experiments in which components of the file system, device support system, etc., were coded both in SIMPL and assembly language revealed rather large differences between SIMPL and assembly language, as much as four-to-one in some cases.

2. Much of the DCN software deals with control of time-critical hardware such as interprocessor links and interval timers. The code produced by the SIMPL compiler is gross, inefficient and completely unsuited to this kind of software.

3. There is no hope of bootstrapping SIMPL into the PDP11 itself, due to its large size, and SIMPL is not readily available on machines other than the Univac 1100.

4. The SIMPL compiler did not produce relocatable code (but, see above) and was exceedingly awkward to use in large multi-module systems such as the BOS and VOS, and impossible to dynamically load in a BOS hostel.

In response to these limitations, and with support from other sources at no cost to the project, the programming language ULP was designed and a compact compiler was constructed for it. The compiler exists in two forms, one in FORTRAN, the other in ULP itself and has been installed in several computer systems, including the Univac 1100, IBM System/370 and DEC PDP10 and PDP11.

The ULP language and compiler was designed to allow convenient modifications in the language and compiler and yet to be very compact. It now provides data types including character and floating and a segmented virtual-storage structure designed for use in the VOS virtual machine. It now is used as a cross compiler in several time sharing systems including the Univac 1100 and DEC PDP10, and as a direct compiler in RT-11 and the DCN, where it compiles itself in under 16K of memory.

Although a number of demonstration and application programs have been written in ULP and run in DCN user processes, none of

the DCN operating system software modules now in use is written in that language. It was determined that a wholesale rewrite of the DCN software would be preferred over piecemeal replacement, but that neither was possible with the resources available to the project.

On the other hand, new application programs have been written in ULP and an ULP-compatible interface is now installed in the DCN user process. The interface also supports the DEC FORTRAN subroutine calling convention, so that the RT-11 FORTRAN library can be linked with an ULP object module and the resulting load module run in a DCN user process.

4. Summary and Conclusions

The development of the DCN has made significant progress during this reporting interval, in spite of declining contract support and the departure of key personnel. It is the strongly expressed contention of this author that had the state of the development and documentation achieved now been the case in the Summer of 1975 that the course of the project, and, incidentally, the careers of the participants involved, might have been far happier.

In spite of these hindsights the only relevant questions are: what has been accomplished and how might this affect further work. Since it does not seem likely that further funding for work of this kind will be available, nor will be solicited by this investigator, considerations on these subjects may be the principal value of the work actually performed.

It is clear that the DCN as a base for network operations is a viable system. It does provide a base for application programs needing access to programs and files on the same and other computiing systems. These facilities are of course commercially available in software for the PDP11, in particular the DECNET software for the various PDP11 operating systems.

As evidenced from the available documentation, however, this software is extremely resource-intensive, rather inflexible, and does not provide automatic routing and resource assignment such as provided in the DCN. It should be pointed out that the compact structure and extensibility in the spectrum of three operating systems for the DCN allows the complete set of network capabilities in even very small systems such as the LSI-11 and GT-40. The BOS runs on the LSI-11 with 4K of memory and on the GT-40 with 8K of memory including a complete set of display file manipulation functions.

The BOS and VOS operating systems are robust and architecturally sound. The BOS in particular has been extensively used for general-purpose data collection and transmission and device control, and considerable experience has been accumulated in its use. The VOS operating system development has been disappointing and would have been hoped to proceed much faster than it did. Part of this is due to the departure of key personnel and enthusiasm for this aspect of the effort and part due to the growing feeling that the virtual machine and virtual storage architecture enhance the network rather than the other way around. In any case, the exceptional power attributed to the VOS user process by its virtual architecture allows a rich spectrum of experimentation independent of the network environment.

It is disappointing that the effort toward emulation of an existing operating system did not receive more emphasis early in the project. In spite of the fact that DOS turned out to be a less than ideal foundation for which to build an emulator, the effort should have been continued and switched to either RT-11 or UNIX. The key problem here appeared to be lack of familiarity and expertise in these systems. If one of these systems had been available (say for the program development of the DCN software itself) the approach to the problems involved would have been much better appreciated and informed.

The VOS file system is another area of disappointment. The designs of the three generations of file systems was very sophisticated and difficult enough to implement in a single CPU system, much less a multiple distributed system. In retrospect it would have been better to implement a simple fixed segment size system with no fancy protection or sharing mechanisms in the interest of quickly getting on with the other aspects of the network.

The apparent diversion of effort to create the BOS as distinct from the VOS has been the object of criticism. In fact, this led to a healthy assessment of project direction and a much more rapid progress toward the important issues of network architecture than otherwise would have been possible. In fact, it was possible to freeze the development of VOS in some aspects and BOS in others. Subsequently, the two systems have merged in all important aspects except where the virtual architecture is involved. In fact, all of the system software modules for device control, communications, resource allocation and so forth are identical. Only the modules for virtual storage operations are distinct.

Another object of criticism has been the inefficiencies inherent in the design of the serial-by-byte interprocess communication system. This was recognized very early to be a

problem when transmitting bulk data between processes on the same machine, since the overhead of the required supervisor primitives limited such transmission to about 500-5000 bytes/second depending upon the processor type. The suggestion made then was that inter-process messages involving bulk data transfers should be accomplished by mapping segments of virtual memory between processes, as coordinated by the serial-by-byte system. This would amount to adding a segment access method on top of the present system and most properly belongs in the area of VOS extensions. The VOS already has provisions in the file system and swapper processes designed to enable access to segments of a foreign hostel. Provisions should be easily added to effect bulk transfers from these hostels.

The entire set of questions relating to inefficiency are being finessed here, and should be considered in detail elsewhere. An interesting observation has been made in this connection with RT-11 running in the LSI-11 with the AED floppy disk. The AED floppy disk is much more efficient than the equivalent DEC device, providing over 1100 512-byte sectors per disk surface. Most RT-11 system programs are not fast enough to need more than two sectors per disk revolution or a data rate of 6000 bytes/second. This is of a rate comparable to those of the ARPAnet, the digital satellite (SPADE) and (potentially) the radio (ALOHA) channels, and about the maximum rate for the DCN serial-by-byte system in a PDP11/45. One is led to the conclusion that the performance of the DCN system using these interprocess communication channels should be roughly comparable to that of the LSI-11 system mentioned, at least at low degrees of multiprogramming. It is this author's experience that the real-time processing rates expected for compiling FORTRAN and linking object modules for test under the LSI-11 system are roughly comparable to those experienced on the Univac 1100 systems under moderate loading conditions. (When compared under conditions of heavy loading, the 1100 system is much slower than the LSI-11!)

It is in the areas of fault tolerance and resource allocation that most of the progress during this reporting period was made and it is these areas that seem most promising for future work. That the present system is robust and resilient to fault transients can be easily proved by simply powering down one machine in the network and then powering it up again. As demonstrated repeatedly in the laboratory, the communications and resource access subsystems do in fact respond to the transient as expected. There is a period of confusion as the various messages time-out and corrections are made to the catalog and dictionary tables, following which substantially normal operation ensues for the remaining operating fragment(s) of the network.

The most obvious problem with the transient fault recovery

design is that it doesn't go far enough. In order to be completely effective, every module in the system would have to respond to a fault in the same fashion as those already built to the specified design. The most critical area in which this should be done is in the file system. This could be accomplished most effectively in the virtual file system by carefully modifying the swapping processes and segment exception rouines. However, the practical requirement that all software, including compilers, editors and application programs respond in the required manner would seem to dampen the ardor of all but the most diligent, particularly in the University research environment.

The most interesting area for further work, at least to this author, is in the area of automatic resource allocation. In the body of this report the term "resource assignment" has been used intentionally. This is to emphasize the partial nature of the algorithms implemented. The existing mechanism, as mentioned above, works well in the experimental environment where the rate of requests and communication overheads are low. In more realistic systems the question of cost determination, heuristic algorithm design and optimality can be expected to play a major part. One of these questions involves prior cost computation and optimal resource location, using the methodologies of Operations Research. Other questions involve resource migration as determined by usage patterns. Still a third is an approach combining the two, taking into account the stochastically non-stationary environment in which network state information is available only after unpredictable nontrivial delays.

5. Appendix - Publications of the DCN Project

1. Lay, W.M., Mills, D.L., and Zelkowitz, M.V. Design of a distributed computer network for resource sharing. AIAA Computer Network Systems Conference, Huntsville, Alabama, April 1973.

2. Lay, W.M., Mills, D.L., and Zelkowitz, M.V. Operating systems architecture for a distributed computer network. ACM Conference on Trends and Applications of Minicomputer Networks, Gaithersburg, Maryland, April 1974.

3. Zelkowitz, M. V. Simulation and implementation of computer networks. Proc. Thirteenth Annual ACM Washington Chapter Symposium, June 1974.

4. Mills, D.L. Structured programming and compiling in a minicomputer environment. Computer Science Technical Report TR-339, University of Maryland, October 1974.

5. Mills, D.L. Structured programming and compiling in a minicomputer environment. Proc. IFIP TC-2 Working Conference on Software for Minicomputers, Lake Balaton, Hungary, September 1975.

6. Lay, W.M., Stebbens, A.K., and Pollizzi, J.A. PDP11/Univac 1108 cross assembler system. Computer Science Technical Report TR-422, University of Maryland, October 1975.

7. Mills, D.L. An overview of the Distributed Computer Network. Computer Science Technical Report TR-413, University of Maryland, October 1975.

8. Mohr, J. A graphics supervisor for the GT40. Computer Science Technical Report TR-440, November 1975.

9. Mills, D.L. The Basic Operating System for the Distributed Computer Network. Computer Science Technical Report TR-416, University of Maryland, January 1976.

10. Mills, D.L. Transient fault recovery in a distributed computer network. Computer Science Technical Report TR-414, University of Maryland, January 1976.

11. Mills, D.L. Dynamic file access in a distributed computer network. Computer Science Technical Report TR-415, University of Maryland, January 1976.

12. Mills, D.L. Transient fault recovery in a distributed

computer network. Proc. NBS IEEE Trends and Applications 1976 Symposium, Gaithersburg, Maryland, May 1976.

13. Mills, D.L. An overview of the Distributed Computer Network. Proc. AFIPS 1976 NCC, New York, N.Y., June 1976.

14. Mills, D.L. User's guide to the ULP language for the PDP11. Computer Science Technical Report TR-536, University of Maryland, May 1977.

15. Mills, D.L. The ULP language and compiler. Computer Science Technical Report TR-537, University of Maryland, May 1977.

16. Mills, D.L. Real-time recognition of manual Morse telegraphy using nonlinear estimation and Viterbi decoding. Computer Science Technical Report TR-554, University of Maryland, July 1977.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR-555	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Final Report on the Distributed Computer Network Project		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) David L. Mills		8. CONTRACT OR GRANT NUMBER(s) N00014-67-A-0239-0032
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science University of Maryland College Park, Md. 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Information Systems Program Code 437 Arlington, Va. 22217		12. REPORT DATE July 1977 13. NUMBER OF PAGES 21
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release. Distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Distributed Computer Network, Dynamic Resource Allocation, Fault Tolerant System, Virtual Segment Network

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes the work done during the last year of the Distributed Computer Network (DCN) Project. The accomplishments during this period include further refinement of the operating systems and compilers, development of automatic fault-recovery and resource-assignment facilities and re-evaluation of certain concepts indigent to the DCN architectural design.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified 7-14-77

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)