JARECT Vol. 9, Telecommunication Technologies (1983), H. Inose (ed.)
OHMSHA, LTD. and North-Holland Publishing Co.
© OHMSHA, LTD. 1983

2.5 Distributed File and Job Management of Network Oriented Operating Systems

Hidehiko TANAKA* and Tohru MOTO-OKA*

Abstract

The Network-oriented Operating System (NOS) concept is based on a message-oriented interprocess communication facility which can be used independently of process locations. Distributed file and distributed job management systems are implemented on the experimental computer network, TECNET, as a set of system processes on the system nucleus. The structure of these management systems is discussed and the implementation results shown. The enhancement of system reliability with respect to system failure is also discussed.

2.5.1 Introduction

Since the success of the ARPA network project, many computer networks have been developed. In the past 10 years, basic networking concepts such as interprocess communication beyond machine boundaries, communication protocol, layered structure of network architecture, and virtual devices such as network virtual terminals have been developed. At present most computer manufacturers support network applications through such network architecture products as SNA. The internetworking of existing computer networks is going to be an indispensable technology in the future.

However, most computer network applications are limited to remote time sharing system usage, file transfer, or remote job entry. With distributed processing, only very simple computer cooperation such as that involving fixed job partition between host and satellite computers has been realized so far. More complicated distributed processing based on the general interprocess communication facility has not been investigated, except for homogeneous computer networks. This article is concerned with general distributed processing for heterogeneous computer networks.

Our research on computer networks began in 1972.[1,3] The goals of this project were to : define the basic operating system structure appropriate for many kinds of network applications ; develop a model for network utility management of file and job management, and, test the feasibility of a network oriented operating system (NOS) through real implementation.

The project began by making an experimental computer network testbed (TECNET) in the laboratory, using a few minicomputers and medium-scale/large-scale computers. In 1979, the implementation of an extended version of a file management system that supports the multipled file handling feature used to enforce system reliability was completed. The second phase of network oriented operating systems research, began in 1979, along with research on hardware computer architecture most suited to distributed processing.

2.5.2 System Architecture of NOS[3]

The structure of the NOS is shown in Fig. 2.5.1. It is assumed that all computers in the network have a four layered operating system consisting of: a system nucleus, a distributed file management

* Department of Electrical Engineering, The University of Tokyo, Bunkyo-ku, Tokyo 113.

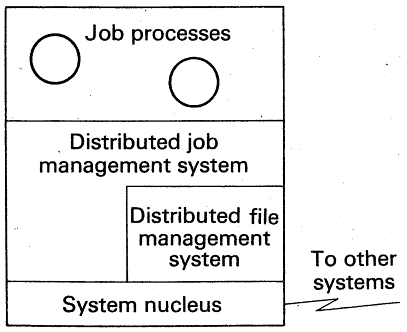

Job processes
Distributed job management system
Distributed file management system
System nucleus
To other systems

Fig. 2.5.1 NOS system structure.

system, a distributed job management system and job processes. The functions of the system nucleus are process scheduling, multiprogramming and primitive operations support, such as the provision of an interprocess communication facility, interrupt handling, and virtual memory control.

All other functions are implemented by many kinds of processes that communicate with each other through primitive operations for interprocess communications, such as SEND msg/RECEIVE msg. These operations activate explicit message transfer. Memory sharing among processes is not permitted in the NOS, even if the processes are in the same computer. Although the interprocess communication primitives issued by users are checked for their capability, those issued by system processes, such as device processes and distributed file and job management processes, are not checked.

Because the location of each process is transparent to the user of the NOS interprocess communication facility, some management or application processes can be implemented easily. Although all current network architectures are based on the connection-oriented interprocess communication, our interprocess communication facility is message-oriented, and each message transfer is controlled directly by SEND/RECEIVE primitives. This characteristic is especially effective for achieving more tightly coupled control systems for network applications or some transaction-oriented applications. The NOS file management system enables users to access remote files as easily as local files. When allocating file resources, this manager guarantees deadlock-free network wide allocation by following a predetermined file request cycle. NOS jobs consist of distributed process sets, which are controlled by the distributed job management system. A high-level system description language was developed to implement the NOS and with this language, a programmer can directly use a system nucleus service, such as interprocess communication.

2.5.3 File Management

File management structure

The objectives of the NOS file management system are:

(a) Easy access to distributed files
(b) Fully distributed control
(c) Built-in protection against deadlock
(d) Extension to distributed DBMS
(e) Multiple copy support

The structure of the NOS file system is shown in Fig. 2.5.2.

The user requests a file through an access interface (FACI). When the file access command is

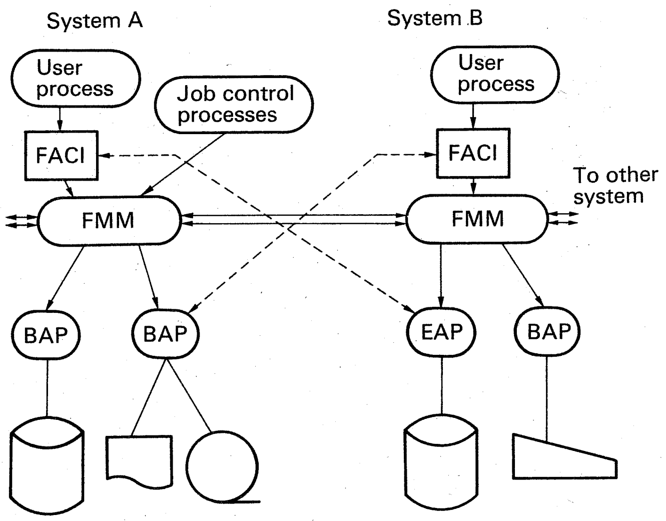


Fig. 2.5.2 Structure of NOS distributed file management system.

"OPEN," the local File Main Manager (FMM) communicates with a remote FMM and locates the Access Process that controls the physical access to the required file. FACI keeps the process number of the access process and transmits file access commands, such as "READ" or "WRITE," when they are issued by the user process. The processing of "CLOSE" is via the FMMs, as is "OPEN." There are two kinds of access processes—Basic Access Process (BAP) and Extended Access Process (EAP). BAP's support conventional file access, such as disk read/write, line-printer output, and card-reader input. EAP's support higher level access, such as a search operation for data that satisfies some condition. EAP's process data where it exists, thus minimizing the volume of transmitted data and making remote file access efficient.

Distributed file system

It is assumed that every user has a home site where the main file directory is located. When a user issues a file access command, the assigned file name is specified. Using this name as a key, the system searches a User File Name Table (UFNT) until the owner name, the file name assigned by the owner, and the home site of the owner are found. The physical location and disk address of the file can be obtained by searching an Owner File Name Table (OFNT) at the system pointed to by the home address of the owner. The access control field in the OFNT is used to check the file access authorization. When a user wants to enter a shared file into his UFNT, he must get the owner's name and the file name assigned by the owner. It is assumed that the user does this by referencing data dictionaries or printed manuals.

An example of file access is shown in Fig. 2.5.3. In this example, user "a" accesses file "ufn" owned by "b." The owner's file name is "nnn," and the owner's home system is "Y."

Deadlock handling When a user is going to use several shared files at a time, he faces a deadlock problem. It was decided to assign all the required resources free of deadlock at one time. In the TECNET network, all files stored are numbered sequentially. Prior to the start of processing, the user requests all files needed and the File Main Manager creates a file request command with a list of required file names. This command is circulated along a fixed loop of systems. Each FMM checks the availability of requested files stored in that system, and forwards the command to the next system. As a result of these transfers, the file request is granted only if all files are checked

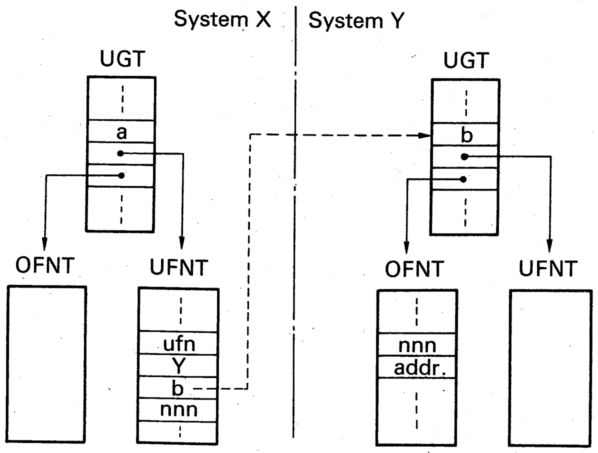


Fig. 2.5.3 Example of remote file access.

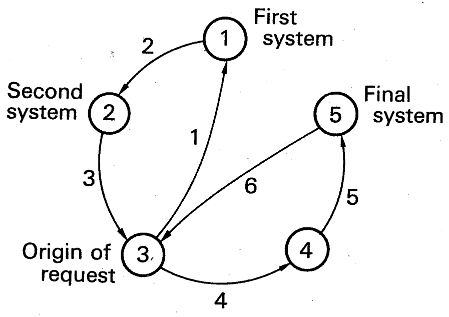


Fig. 2.5.4 Example of file request routing.

successfully. Otherwise, the request fails and a failure message is returned to the user. Figure 2.5.4 shows an example of the request circulation. Generally, any system can be skipped if it has no files named in the list.

File management protocol Several commands that are standard in the network were provided, as shown in Table 2.5.1. These commands are issued by the FACI when the user process uses this interface. GETFN is used to get a physical file location, owner name, and owner file name, and is sent to the user's home FMM. OBTAIN is a command used to get the file access right for all files to be used. FREE is used to disable the file access right. CATALOG is used to catalog a specified file into its UFNT with the name assigned. PERMIT is used to permit sharing of the file with other users. All the commands are accompanied by their corresponding reply messages.

The basic access methods are sequential-access and direct-access. At present, the Virtual Sequential Access Method is supported by a processor, but this can only be used locally. The open modes are Input, Output, and Update.

Multiple copy feature

The reliability of the file system can be improved by having file copies which enables the user to access a copy of the file that could not normally be accessed because of system failure. In the network

Table 2.5.1 Distributed file commands and job control commands

File commands	Job control commands
GETFN	set MJMP initial data
OBTAIN	request to create LJMP
FREE	request to remove LJMP
OPEN	request to remove MJMP
CLOSE	set LJMP initial data
CATALOG	request to create jobprocess
UNCATALOG	request to start jobprocess
PERMIT	request to stop jobprocess
INHIBIT	request to remove jobprocess
READ	request to abort jobprocess
WRITE	request to quit jobprocess
	report of jobprocess end

environment, another benefit of having copies is that communication costs decrease, because the effective distance to the file gets shorter when copies are placed at several sites. However, the existence of copies should be transparent from the user's point of view. That is, the contents of all copies should be consistent, even if many users try to update the file concurrently.

In the following discussion, two kinds of files are assumed : logical and physical files. A logical file is the original file from the user's point of view and is independent of the existence of copies. Physical file is used to indicate a copy of the original file.

Distributed file directory with copy feature The OFNT used previously is broken into a new OFNT and several Physical File Node Tables. The OFNT contains all the copy location information. When a user wants to use a logical file, he selects one of the physical files, the choice of which is dependent on the location of the user process and the availability of each copy. This selection algorithm plays an important role in reducing the cost of distributed access when the decomposition of queries expressed in some high-level DML is concerned. Accordingly, a method of getting all the location information for copies at the time of the access-right request (OBTAIN) is provided. This method will be useful for inplementing distributed DBMS. The PFNT maps an owner file name to the location information for a physical file (disk name, address, etc.). This table is placed at the site where the physical copy is stored.

Distributed file protocol with copy feature When OBTAIN is used, the location information for all copies is returned with the reply message while the file access-right check is performed. A new command, CREATE, creates a physical file and its PFNT. To associate the PFNT with an OFNT, the OBTAIN command is used with an OFNT registration flag set after the CREATE command. To add a copy of a logical file, the command, ADDCOPY, is used to make up a physical copy and register it into the OFNT. The command, DLTCOPY, deletes a copy. To get the contents of the UFNT, OFNT, and PFNT at some FMM, the DISP command is sent to that FMM. CREATE, DELETE, ADDCOPY, DLTCOPY, and DISP commands are added, and OBTAIN/FREE commands are modified to facilitate the copy feature. A user can create a logical file with a few copies placed at several sites by using these commands.

2.5.4 Job Management

Network job management structure

In the network environment, a job is defined as a set of processes that are located at several computer systems and that communicate with each other to perform a function for the user. Accordingly, the job management system in the network environment should manage jobs across the computer boundary. The architecture of the network job management system of the NOS is shown in Fig. 2.5.5. Each job is managed by a Master Job Management Process (MJMP) and several Local Job

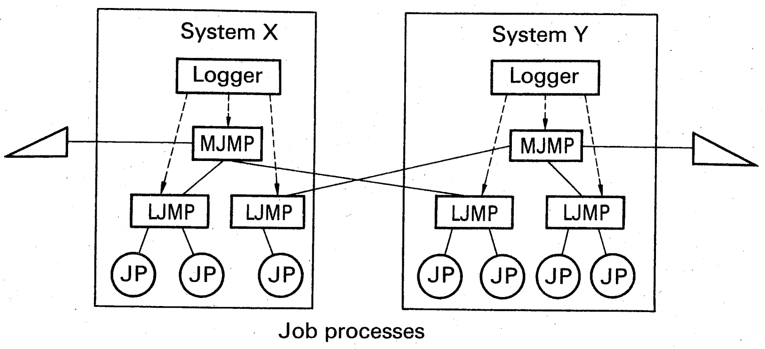


Fig. 2.5.5 Structure of distributed job management system.

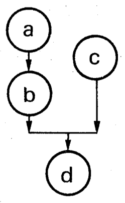
Management Processes (LJMP's), which are located at each computer system in which some job processes of the job are running. The MJMP manages all LJMP's and is the job management center. The hierarchy structures of the MJMP and LJMP's are made up at job creation time for each job with the help of a logger process. A network job control language was designed to define these network job structures.

Network job control language

A set of job control statements sufficient to realize the arbitrary structure of a network job on the computer network is provided. Figure 2.5.6 shows an example of controlling the execution order of processes. Figure 2.5.7 shows a very simple example of a job that reads file data from an OLDF at site OKITAC and writes the processing results into NEWF at site PPS.[2)]

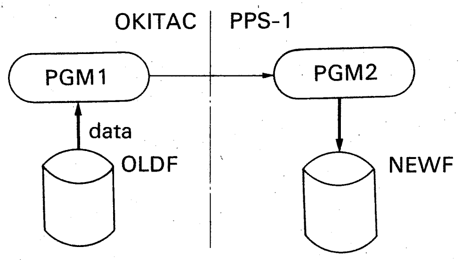
Distributed job management protocol

Communications among the MJMP, LJMP, and logger define the distributed job management protocol. 12 basic commands are provided, as shown in Table 2.5.1. These commands are followed by their appropriate responses. When a job is started, the MJMP sends a "request to start job-process" command to the LJMP and thereby indirectly controls the process. LJMP's monitor job processes that are local to the site, provide management information reports, such as accounting and termination status of each process, and send them to the MJMP by a "report of job-process end" command. The division of job management work between the MJMP and LJMP's is made so that local management tasks are dedicated to the LJMP's, and others that are common to all systems are left to the MJMP. Accordingly, this protocol is independent of specific implementation.



```
$EXEC  fa,  a,@PPS
$EXEC  fb,  b,@PPS/a
$EXEC  fc,  c,@OKI/
$EXEC  fd,  d,@PPS/b,c
```

Fig. 2.5.6 Example of process execution order control.



```
$JOB  USR1@OKI
$ALLOCATE  CORE=10@OKI
$ALLOCATE  PRCS=01@PPS
$FD  OLDF,FN=MSF1,DISP=(OLD,DEL)
$EXEC , PLIB(PGM1)@OKI
$FD  NEWF,FN=MSF2,DISP=(NEW,CAT),
       UNIT=@PPS.DK.SYS,SP=10
$EXEC , ELIB(PGM2)@PPS/
$STEP
$END
```

Fig. 2.5.7 Example of network job description.

2.5.5 Reliability Considerations

Principles

Reliability is essential to preserve the autonomy of each system and to manage failure when using distributed facilities. As the basic structure of the NOS is based on the message oriented interprocess communication, processes cannot send or receive data without permission of other processes. Accordingly, a failure of one process is well shielded from the others. This interprocess communication facility is the only way of achieving intersystem interaction, so the autonomy of each system is preserved. The process name registration scheme is to statically secure interprocess communications and is efficient in terms of communication overhead. This scheme is also important for the maintenance of independence between programs and processes.

Reliability enhancement of the distributed file management system

A multiple copy feature is introducde to make the distributed file usage reliable. However, other enhancements are needed to make the distributed file management system itself more reliable. These enhancements include:

(a) Distributed file directory enhancement

(b) A consistent update mechanism

If a system failure occurs when the user is reading a file, the access history of the file should be kept in the file-using system in order to change the physical file dynamically, even though the location information of alternative files can be easily retrieved, because it was set in the local area of the FACI when the OBTAIN command was used.

In a multiple-file environment, a network or computer system failure can cause inconsistency of file content. One inconsistency is internal to the file; that is, the original file's contents are inaccurate. Another inconsistency is external, which means there are discrepancies in the contents of copies. Because these inconsistencies occur when the file is updated, some corporation rule among copies is needed to preserve consistency. For example, a decision can be made as to whether the update transaction can be forwarded or not, depending on the conditions:

(a) Whether the user can communicate with some specified site.

(b) Whether the user can communicate with more than half the sites which store the physical files.

If the above conditions are not met, the transaction is aborted. Each update transaction is assigned a sequence number at the file-request stage (OBTAIN) that uniquely identifies the processing order of transactions for the file. The update operation should be done for all copies, though some systems that store the copy may have failed, and the update operation cannot always be performed at exactly the same time. Accordingly, the transaction and the sequence number are kept at some sites, and can be forwarded to the failed systems as soon as they recover. To ensure that files are updated consistently, even when failures occur, mechanisms such as the "Two Phase Commit" procedure developed by Lindsay of IBM should be provided.[5]

Failure handling of the distributed job management system

Failures can be classified as follows:

(a) Internal Failure of a Job Process
(b) Computer Systems Failure
(c) Network Failure

Depending on which of these three cases is involved, the failure is handled by the MJMP and/or LJMP's by exchanging management messages.

2.5.6 Implementation

Languages

To implement the NOS for each member of TECNET, assembler languages and micro assembler language were used for kernels, and a language, NPLAN, for control process description. NPLAN is an ALGOL-like high-level language and is based on a language called PLAN developed by Takeichi.[4] The NPLAN compiler, written in PASCAL, produces V-codes, which are interpreted by a microprogram of PPS-1.[2] A few primitive operations were added to PLAN so that NPLAN can be used as the basis for writing programs in the network environment. Statements related to the Primitive operations are compiled into V-code "KCALL" (kernel call), the parameters of which include an operation identifier, the process name of the opposite side, a message identifier, and a buffer identifier, etc. The V-code interpreter, written in microcode, expands the KCALL to the corresponding primitive operations written in the microprogram. For example, a SEND statement can be written as:

Cstatus : = Send (Hostname, Processname, Waitmode, Message)

Hostname and Processname define the destination process, and Waitmode specifies whether this process (source) waits for the end of the send communication event or processes the next statements. "Send" is an internally-defined function procedure. Cstatus, therefore is the resulting status of the Send execution. The V-code interpreter is a virtual stack machine that is provided for system portability. The size of this interpreter is about 1500 microprogram steps.

Implementation results

Table 2.5.2 shows an NOS implementation result. Implementation is performed for three machines: OKITAC 4300C, PPS-1, and FACOM R. A simplified version of the NOS nucleus is implemented on the FACOM R. The distributed file manager and job manager are implemented on the OKITAC and PPS-1. Table 2.5.2 shows the result of two versions of the distributed file manager. The "Single" version does not have the Copy feature; the "Multi" version has the Copy feature at the PFNT level. Multiple UFNT and OFNT support, consistency maintenance, and job recovery features

Table 2.5.2 NOS element program size.

Program element	OKITAC 4300C		PPS-1			
	Single	Multi	Microprogram		V-code	
			Single	Multi	Single	Multi
Nucleus	3 700		3 000			
(comm.)	3 400		1 600	4 000		
(others)	1 600		400	(System data)		
FMM	1 900	3 100			4 500	6 100
BAP	1 550	4 000	717	719		
FACI	900	1 550	141	805		
MJMP	4 000				5 000	
LJMP	500				1 300	
Logger	750		500			
Loader	1 500		250			

are not yet supported. By implementing the Copy feature, the size of the FMM increased 30 ~ 40 percent, and that of the FACI, 70 percent for OKITAC, and 400 percent for the PPS-1. This 400 percent increase includes the increase in code as a result of improving the user interface. The amount of code for the PPS-1 program is expressed in terms of microcode or V-code (16 bit/step). The processing time of user file commands is measured for PPS-1 and OKITAC 4300C. For example, GETFN needs 8.8 ms and 1.4 ms, OBTAIN 23.0 ms and 3.8 ms, and CREATE 13.4 ms and 1.7 ms, respectively. The time needed for interprocess communication is about 2.0, 8.5 and 41 ms for PPS-local, OKITAC-local and PPS-OKITAC communications.

2.5.7 Conclusion

This paper presents the structure of a network oriented operating system and demonstrates its feasibility by showing implementation results obtained with the experimental computer network TECNET. The elements of the NOS are the system nucleus, which enables interprocess communication beyond machine boundaries, the distributed file management system, which permits multiple files, and the distributed job management system, which expands the concept of local processing to a virtual network environment yet preserves the autonomy of each local system.

The next step involves research on hardware computer architecture which is oriented to distributed processing, and the development of the high level network-oriented programming language DIPROL.

References

1) H. Tanaka and T. Moto-oka: An Experimental Computer Network TECNET, Papers of Technical Group on Electronic Computers, IECE Japan, *EC-73-57* (Dec. 1973) [In Japanese].

2) T. Moto-oka and Y. Yamamuro: Polyprocessor System PPS-1, *JIP 15* (July 1974) 557 [In Japanese].

3) H. Tanaka and T. Moto-oka: Network Oriented Operating System-Process Control and Distributed File in TECNET, Pacific Area Computer Network Symposium (Aug. 1975) 171.

4) M. Takeichi: A Mini Compiler of a Mini Language, bit, '6, No.8-13 (1974) [In Japanese].

5) B. G. Lindsay, et al.: Notes on Distributed Database, Research Report, IBM Research Lab., San Jose, *RJ2571* (June 1979).