NETWORK ORIENTED OPERATING SYSTEM
——PROCESS CONTROL AND DISTRIBUTED FILE IN TECNET——

Hidehiko Tanaka and Tooru Motooka
University of Tokyo
Tokyo, Japan

1. INTRODUCTION

In our laboratory, an experimental computer network project, TECNET (Tokyo Experimental Computer Network), started four years ago. The primary aim of this project was to research the computer-to-computer communication system, to develop a network oriented operating system for the future use, and to study the problems in the network environment through the extensive use of the network.

We presented some of our research results as to the TECNET communication facility and the philosophy of Network Oriented Operating System (NOS), in the previous paper.[(1)~(5)] In this paper, we describe the process control mechanism in NOS, show the implementation, and discuss about the distributed file system. In NOS, we suppose that each element computer system of network has a similiar process control mechanism, especially a process-to-process communication primitives-set, even if the hardware architecture of element computers is different in machine languages, and peripherals. The TECNET has a wide variety of communication equipment, from computer-to-computer direct bus coupling, to a high speed data communication lines. Also, it has a wide selectivity of computer resources such as minicomputers, high speed large scale computer, multi-processor system, and microprogrammed processors with writable control storage. Accordingly, inspite of its small scale, TECNET brings us a useful experimental tools for computer network.

2. TECNET

In fig. 1, it is shown the system configuration of TECNET. Computer connections are made of four kinds of means. First one is the common bus connection, of which lines are composed of 12 pairs of lines, and which can transmit data at the speed of 20 k byte per second. Second one is the communication lines of which transmission control procedure is the basic ISO standard. Third one is the

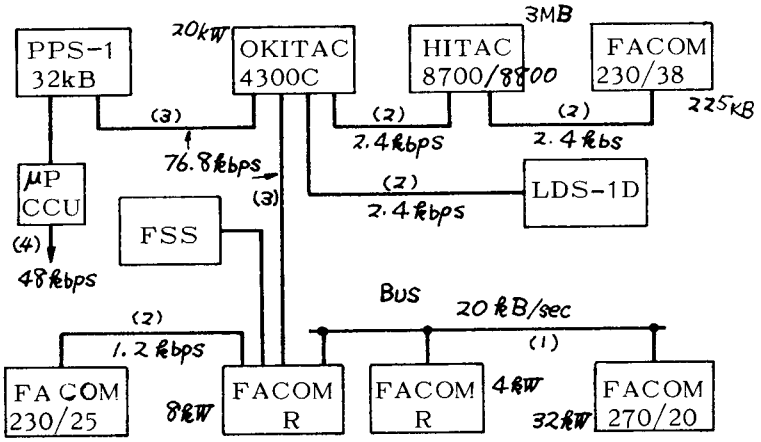

Fig. 1 TECNET

high speed code-transparent communication lines, of which transmission control procedure is the one expanded from ISO standard. The communication control units for this kind of lines are made of about 500 ICs and MSIs, can transmit data at the speed of about 200 kbps in full duplex. Almost all transmission errors are detected by cyclic redundancy check code of 16 bits, which is generated and tested by hardware. Fourth one is the line controlled by a special communication control unit which is connected to a poly-processor system. The function of this CCU is microprogrammed, henceforth can be changed easily. Accompanied by adequate hardware, this special CCU is highly adaptive, and has the ability to connect to almost any kind of other communication lines without incurring much overhead; this hardware is composed of mode changeable error checking MSI, transmitter and receiver LSIs of which function can be changed to adapt asynchronous transmission or synchronous transmission, and some frag pattern control circuits for high level data link control procedure.

HITAC 8700/8800 is a large scale computer system in the campus and provides both conversational and remote job entry services through communication lines of 2.4 kbps. FACOM 230/38, and PPS-1 are middle scale computer systems. FACOM 270/20, and 230/25 are small scale computers. OKITAC 4300C and FACOM Rs are minicomputers. OKITAC, FACOM Rs, and PPS-1, all reside in one room. But, other machines are placed in other separate buildings in the campus. PPS-1 is a poly-processor system which is composed of three processors and two main memory banks, each of which contains 16 kB IC-memories. The memory banks will be increased to 4 banks (up to 64 kB) by the end of this year. Each processor is controlled by 14 kinds of micro instructions of 24 bits wide, and has 15 registers, local memories of 64 words, and writable control store of 8 kWs. FSS is a figure-information-input equipment, flying-spots-scanner, which has the ability to sample 4096x4096 dots with 128 (7 bits) levels in a field. LDS-1D is a graphic display system of IMLAC Co.

The main functions of three minicomputers are the IO control of peripherals, communication control of lines, and command interpretation of user commands. OKITAC functions as network file control system as well. Substantial processing is done in the HITAC, FACOM 230/38, FACOM 230/25, FACOM 270/20 and PPS-1.

3. NETWORK ORIENTED OPERATING SYSTEM

The overall control system of the TECNET, which we call as NOS, has a philosophy which is network oriented. There are many processes running at various computer systems. For these processes to communicate each other freely, it must not be given trouble to send or receive information even if both processes are not within a same computer system. The NOS provides interprocess communication primitives which can be used for remote-process communication as well as for local-process communication. The characteristics of NOS is as follows.

(1) multi-process structure
(2) network-wide communication primitives
(3) process for each IO peripherals
(4) distributed file system

(1) means that processes can be created by user and are scheduled by multi-programming in a system. Also, the created processes are stopped or activated by higher level processes. The relationship among processes are expressed in a tree structure as fig. 2.

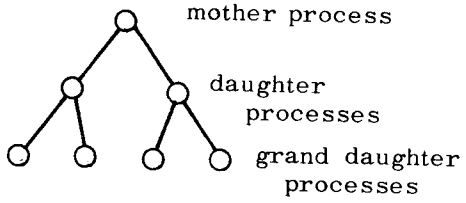

Fig. 2 Process Hierarchy

The higher processes govern the lower processes in the tree. According to (3), when a process wants to output some data, for example, the process sends the data to the output-process through inter-process communication primitives. (4) means that several files located at various computer system can be easily accessed as if they were local.

Primitive Operations

Processes interacts with other processes using primitives, and are controlled by primitives. The primitives provided in NOS are shown in Table 1.

CREATE	SEND
REMOVE	RECEIVE
START	CLOSE
STOP	SENSE
TIME	WAIT

Table 1 Primitives

CREATE and REMOVE primitives are the ones for a process to create or remove a daughter process. Process is placed by primitive START in a queue which contains a set of processes ready for running at any time. It is placed at a dormant state by primitive STOP, and blocked waiting for some event to occur by primitive WAIT. TIME is a primitive by which processes are started at a given time. SEND, RECEIVE, CLOSE and SENSE are the inter-process communication primitives.

4. PROCESS CONTROL

There are three kinds of process queues, a non real time process queue, a real time process queue, and a time designated process queue. Each non real time process is pre-assigned a static priority, and its queue is scheduled according to the priority. Real time processes have higher priority than non real time processes. Time designated processes are woken up when it becomes the designated time point, and preempt the processing of current process.

Virtual Processors

PPS-1 is a complex composed of three identical processors, each of which is controlled by a microprogram stored in a writable control store of 8 kWs (24 bits/W). Accordingly, the machine instruction set of each processor can be changed dynamically through loading other microprograms. We call it a virtual processor that is a processor of which machine instruction set has been settled by loading some microprograms. All processes in PPS have its own virtual processor attribute, VPA, and can be processed only in the virtual processor specified by the VPA. There are several virtual processors and their process queues in PPS.

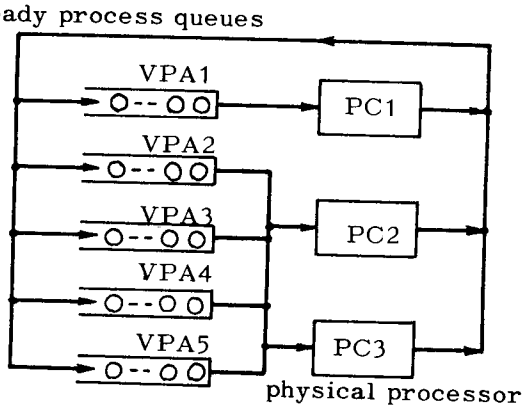

Fig. 3 Virtual Processors

It is a very difficult problem to determine dynamically which process queues should be assigned the physical processors. At present, the assignment is done using the priority scheme which is described in the previous section.

Network Primitives

If we try to extend all of the process control scheme of single computer system to the computer network, all of the primitive operations should function beyond the machine boundary. But, such a scheme is expected to be very complex and inefficient. In NOS, we are planning to experience the following two schemes.

(1) Full set of primitives is available over network

(2) Only the interprocess communication primitives are available over network

The latter scheme is more realistic than the former, and we are implementing this type of scheme at present. But the former also should be given some attention in the future.

Inter-process Communication

The primitives for inter-process communication are SEND, RECEIVE, CLOSE and SENSE. In our previous implementation, there were two other primitives, INFORM and SEARCH instead of SENSE. With this modification, the structure of inter-process communication became more consistent than the previous one.

When a process wants to send some data to othe process, it emits a SEND primitive with parameters which specify the addressed process name and the data block to be sent. If the corresponding process emits RECEIVE, the communication request matches and the data block is transferred from the sending block to the receiving buffer block which the receiving process specified in the RECEIVE PARAMETERS. If the addressed process does not want communicate, it can emit CLOSE primitive to refuse the request. This inter-process communication is extended to the remote process communication as well. Accordingly, a process need not distinguish the remote communication from the local one. The transmission control in case of remote communication is done implicitly within the communication primitives.

The error control procedure uses positive acknowledge method. When a correct data is made up in the main memory, the program is interrupted, and acknowledge message is made up and transmitted through communication line to the neighbor computer. When some error is detected in a received data by cyclic redundancy check code, any action is not activated, and the receiver unit expects for retransmission by time-out. The transmission is 8 bits byte oriented, code-transparent, and synchronous. The communication control units can connect 4 full duplex line circuits, and have the ability to transmit at a speed of 200 kbps in each line.

When some communication requests arrived at a process, and when some data transfer completed, the corresponding process is woken up if the process is in blocked state. The process can get the event-details by emitting SENSE primitives. Event details are kept in event details blocks (EDBs) and placed under monitor control. Processes cannot touch the EDBs directly, but get the information indirectly through SENSE primitive. When a process issued a SENSE, an EDB of the process is copied into the area designated by SENSE and deleted. In our previous implementation, the EDBs existed in user area itself which the process specified. Accordingly, process could touch the EDBs directly. But it was very difficult to manage and keep consistency of the EDBs.

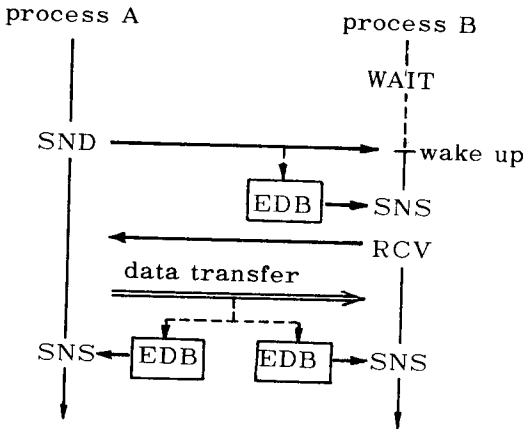

Fig. 4 Interprocess Communication

In fig. 4, it is shown an example of inter-process communication. When data transfer has completed, its EDBs at both sides are created at the same time in case of local process communication. But in case of remote communication, EDB at the receiving site is created at first. After the communication completed message is returned to the sending site, the EDB at the sending site is created with it.

Process Hierarchy

The relationship among processes is expressed by a tree structure. This process hierarchy controls the primitives. Only mother process can issue CREATE and REMOVE primitives to some daughter processes. STOP and WAIT primitives are permitted to issue from mother process to daughter processes, or to itself. But START and TIME primitives can be issued from any process to another process, because a process can check the process which emitted START toward it, when it is activated. The SENSE primitive is used for this check as well. Resource allocation is also controlled by this process hierarchy, and mother process can distribute her own resources among daughter processes.

Known Item Table

There are three kinds of process identifiers, ie., process name (PNM), process reference number (PRN), and process number (PNO). Process name is a symbolic name which user processes utilize to identify the process. Process number is a relative address of the process entry in a process management table which contains all process entries in the system. Process number which uniquely identifies the process in a system, cannot be used directly by user processes. Each user process has its own known item table which is composed of the process numbers of processes to which communication is authorized. Process reference number is a index with which a process refers to some other processes to communicate with. In fig. 5, three kinds of process identifiers are shown.

When a process wants to communicate with other process, it issues an ADDRESS command with process name as a parameter, which requests for the process number to be entered in its known item table. If granted, the process number is entered, and the relative address of the entry is returned to the original process. Process name is used only in ADDRESS command. For actual inter-process communication, the process reference number is used.

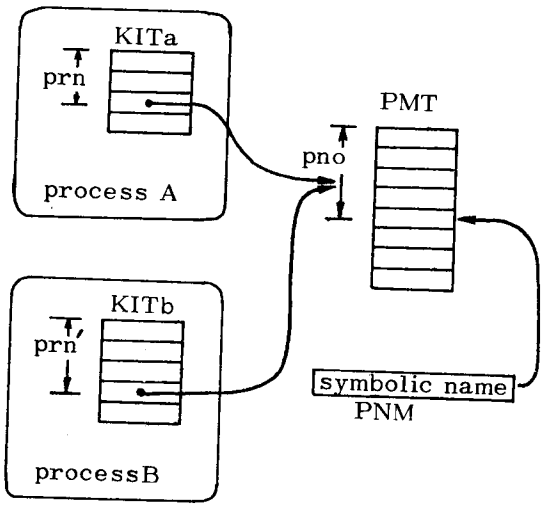

Fig.5 Process Identifiers

5. IMPLEMENTATION

NOSs are implemented in OKITAC 4300C, PPS-1 and FACOM R. Although NOSs in OKITAC and PPS are made of a full set of primitives, NOS in FACOM R is made of a inter-process communication primitives only. Several system programs are made using the NOS primitives.

OKITAC Version

A hardware timer, which we made to interrupt the CPU at any time interval from 100 μs to 6.4 sec which we can set with special IO instruction, interrupt CPU at time interval of 2.0 ms and wake up a time management routine. This routine tests the existence of real time processes, and the occurrence of some time out events. If either of these events is detected, dispatcher is activated, and it causes a process-switching from the present process to another process. Primitives, scheduler and dispatcher, which are the nucleus of NOS, are masked from any interruption during its execution, and coded with assembler language OKISAP 43. The size of nucleus is expected to be less than 8.5 kWs as follows. Communication primitives occupy 5.25 kWs in all, which include packet control routine for the communication lines used in case of remote process communication. Other primitives, scheduler and dispatcher total about 1.5 kWs. Interrupt routines total about 1.5 kWs, which are composed of interrupt routine from input-output peripherals and from three kinds of communication control units.

There exist many system processes which perform major operating tasks. These are as follows.

(1) peripheral (or terminal) control processes : cassetts magnetic tape, TTY, PTR and PDS-1D graphic display control processes
(2) HITAC control processes : conversational job, remote batch job, and TOOL control processes
(3) logger process
(4) system command analysis process
(5) file control process

PDS-1D graphic display terminal is connected to OKITAC with 2 pairs of communication lines. The display control process makes the code conversion between PDS and OKITAC standards, divides the output character sequence from HITAC , for example, into 80 characters blocks, and concatinates some blocks from PDS into a message to HITAC. TOOL is a online file management macro command system developed in HITAC system for easy use of file. TOOL control process issues adequate macro commands to HITAC in responce to the request for some files in HITAC system.

This process supports programmed job control facility. Logger process creates a user process when loggin command is entered from some terminals or other processors. Also it searches the ID number of process which exists in OKITAC system, and informs the other logger processes in some remote computer system of the number when it is asked from it. System command analysis process receives character strings in ISO 7 bits code from other processes, analyzes them, and performs appropriate procedures. File control process controls the distributed file system as will be described in next section.

PPS Version

PPS-1 is a microprogram controlled multi processor system. Machine languages are coded in microprograms in control storages. At the end of each machine instruction cycle, input-output interruption and timer interruption which are indicated in a status register, are tested by a microprogram. When either of these flags is on and the interrupt mask is off, it activates the interrupt routine of machine instruction level. The processors of PPS have associative memories for address translation, so they are suited for virtual memory control. Accordingly, NOS of PPS supports virtual address space of 24 bits. Secondary memory is a disk cartridge of 5 M bytes. Page size is 1 kbytes. Segment size is 64 kbytes, and user space is 256 kbytes for each user.

PPS-NOS also supports virtual machine environments. Each process has a virtual processor attribute. The control memory space of each physical processor is divided into 2 parts logically. One is a common area which stores the nucleus microprogram of NOS. The other is an individual area which stores a microprogram of a virtual processor. The contents of this area is changed dynamically when process is switched to some other process of which virtual processor attribute is different from the previous one.

We have developed 3 virtual processors, OKITAC emulator, FACOM R emulator, and fast fourier transform machine by this time. These sizes are 1.7 kWs, 1.0 kWs, and 1.6 kWs respectively. We are developing some other virtual processors, APL machine, IO processor, and so on. There are three kinds of IO processors. First one is a low speed peripheral processor which controls a line printer, card reader, character display, and teletypewriter. Second one is a disk control processor which controls cartridge disks up to 3. This processor is used for virtual memory control. Third one is a processor for communication control which controls the input and output of data from and to communication lines. These 3 kinds of processors could be gathered into a processor. But the matrix switch unit between processors and IO adapters permits only one-to-one connections and does'nt support one-to-many connections at a time. So, division of IO processing function into 3 processors is suited for PPS-1.

The nucleus of NOS, primitives, scheduler, dispatcher, and microprogram interrupt routine, is coded in microinstruction. This follows that primitives are processed in the same manner as machine instructions.

6. DISTRIBUTED FILE SYSTEM

In computer network environment, we should face a problem of many file systems of different structures distributed at various site in the network. To master the network resources perfectly, we should extend the fuction of each file system so that we can use all file systems as if they were a single file system. This extension is very difficult when file structures and file conventions are different each other. As each monitor system is closely related in NOS, this distributed file system can be realized easily. The structure of NOS file system is shown in fig. 6.

File Command

There are four basic file commands which user process can use : OPEN FILE, CLOSE FILE, READ FILE, and WRITE FILE. OPEN FILE command is to get the right of access and to exclude the other access requirement to the same file. CLOSE FILE command is to release the right. READ and WRITE command are used to read and write the content of the file after OPEN command is issued. File control process in each HOST takes the responsibility for controlling the access to its local file processes. Also, it has a duty to direct a OPEN requirement for some remote file

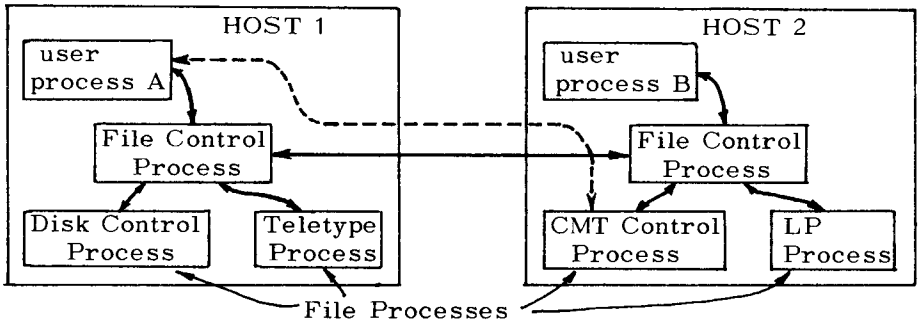

Fig. 6 NOS Distributed File System

to the remote file control process. After OPEN procedure has finished, the substantial file access is done directly between the user process and the required file process. File control process establishes the link between them. This direct file access is transparent for the remoteness of the file through global interprocess communication primitives.

File Concept

The term, file, in NOS is used in wide sense. It means input-output peripherals as teletype-writer or card reader, as well as the disk file. Accordingly, when a process wants to use type-writer for its output printing, the process OPENs the file "TTY" and WRITEs out the output characters to file "TTY". Element files in TECNET are CMT at OKITAC, HITAC file system through OKITAC, DISK at PPS, and other many peripherals. HITAC file system means the file facility at HITAC system which can be used through communication line. HITAC file control process in OKITAC shows the HITAC system as if it were a local disk file system.

Logical Name vs Physical Name

Each file has both logical name and physical name. Logical file name specifies the logical relationship among files and its logical structure. In contrast to this, physical file name is the physical address of the file which defines the detail location of the file. The logical structure of files is constructed over the physical structure of file control system. Our tentative version of NOS implements physical file name only. But, in our next version, we are intended to implement logical structure of files.

The structure of physical file name is

SYSNAME. DEVICE. OPNNAME. ACCSNAME

where SYSNAME means computer system name such as OKITAC, PPS, etc, DEVICE means the name of device such as DISK, CMT, etc, OPNNAME specifies the unit file for open procedure, and ACCSNAME specifies the unit record of the file in each access procedures as READ, WRITE. For example,

PPS. DISK. SYSA. R05

means a record R05 in a system file A resided in disk storage of PPS-1. The physical location in terms of disk address is specified through table-look-up in the directory file with SYSA. R05 as its key.

7. CONCLUSION

In this paper, we described the development of network oriented operating system, NOS for

the TECNET. We presented the phylosophy of NOS, the details of NOS nucleus part, the interprocess communication procedures, the implementation of NOS in TECNET, and the structure of distributed file system. At present, we are implementing the distributed file system of physical name version. Also, we are making ever continuing improvement for interprocess communication and virtual processor scheduling algorithm.

REFERENCES

(1) H. Tanaka, and T. Motooka : A Experimental Computer Network-TECNET, IECEJ, EC 73-57 (Dec. 1973). in Japanese.

(2) T. Motooka, and H. Tanaka : Polyprocessor Operating System, second meeting paper for basic research of large scale information systems, Aug. 1974. in Japanese.

(3) T. Motooka, and Y. Yamamuro : Polyprocessor PPS-1, Information Processing Society of Japan, Vol. 15, No. 7, July 1974, pp. 557-564.

(4) S. Ueda, H. Tanaka, and T. Motooka : Interprocess Communication Control System, IECEJ Conference 1974, No. 1685.

(5) T. Motooka, H. Tanaka, and S. Harada : Communication Control Unit For Computer NETWORK, IECEJ Conference 1974, No. 1687.