DRAON : An Intelligent Local Area Network

Tadashi Ae, Tetsuya Toi, Reiji Aibara, and Noriyasu Takahashi

Faculty of Engineering, Hiroshima Univercity
Saijo, Higashi-Hiroshima, 724 Japan

Abstract

This paper discusses the design and fabrication of a local area network which keeps distributed systems off from system deadlocks in resource sharing with low overhead.

We propose an Intelligent Network that extends the network concept in its communication capability. This intelligent network provides the upper layer protocols, such as transport protocol and session protocol, which are normally carried out by host computers, and offers deadlock-free resource access method to its client, i.e., a distributed operating system.

The design specification of DRAON, which is a kind of entirely distributed controlled intelligent networks, is briefly introduced. DRAON is a network of intelligent nodes each of which performs communication procedures with high performance, and resource lock/unlock operations in an inborn manner.

Finally, an experimental resource sharing system using a prototype-DRAON is presented as an example of applications of loosely coupled distributed computing systems.

Key Words:local area network, resource sharing, system deadlock, intelligent network, distributed operating system.

1 Introduction

First, the logical structure of DRAON will be discribed for introduction. In section 2, the access control strategy for the deadlock-free operations, which has employed by DRAON to avoid the system deadlocks in resource sharing, will be described. In section 3, the network configuration of DRAON will be described. Finally, in section 4, a distributed system which may be constructed using DRAON will be discused.

An intelligent network DRAON has the logical structure just as a centralized operating system, as shown in Figure 1, where a process on a host computer which is connected to DRAON can access any resource(s) that may be settled far away through the communication link which are supported by DRAON.

In Figure 1, the network manager controls the whole network and the link controller sets up, maintains, and tears down the communication link(s). It should be noted that the link controller may set up and maintain more than one communication links simultaneously.

An application process attempting to use the shared resource(s) must submit a command on demand (e.g. Send) to the network manager, as shown in Figure 2. A command is a kind of record type data which contains both indirect and direct object of the command action, and the restriction on execution time, shown as follows:

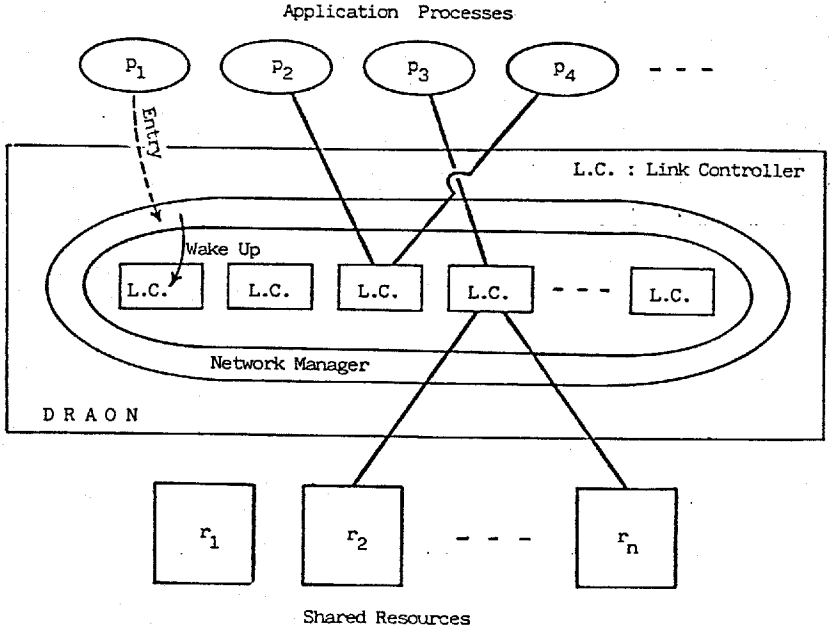


Figure 1 Logical Structure of DRAON

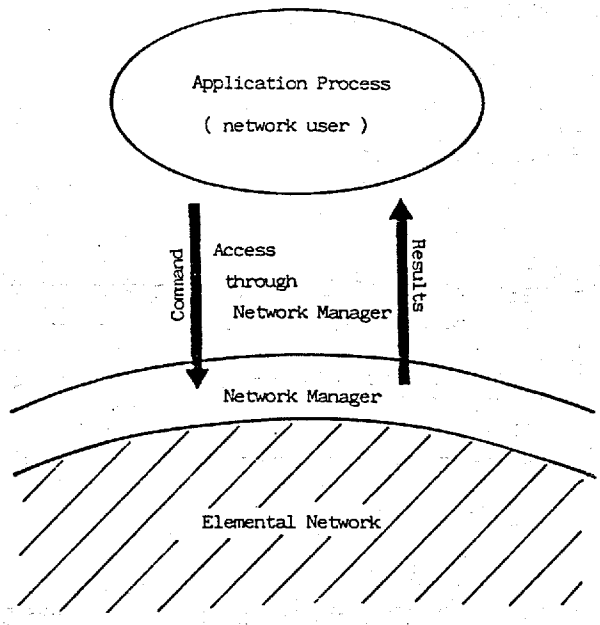


Figure 2 Network Access through Network Manager

```
action ( arg 1 , arg 2 , arg 3 )

where  action is the operation of a
                                  command,
       arg 1  is indirect object(s) of
                        the command action,
       arg 2  is direct object(s) of the
                           command action,
       arg 3  is the restriction on
                           execution time.
```

Some examples are shown as follows:

```
Send ( disk.BOX , listfile , 5 )
```

that means "begin to transfer the file named listfile on the local strage to the shared file server named BOX within 5 seconds, and save the file on it."

```
Send ( printer.PEN ,
                    disk.NOTE:textfile , 20 )
```

that means "ask the file server named NOTE to transfer the file named textfile on it to the print server named PEN within 20 seconds." Figure 3 illustrates this operation.

```
Order ( disk.NOTE , /ABORT/ , )
```

that means "make the execution of the submitted command abort on the file server named NOTE."

The network manager replies to the request from the application process and decides a unique link identifier (link ID) with the submitting process name and accepting time, and subsequently wakes up one of idle link controllers. The waked-up link controller checks the status of the requested resources, if any, after starting a local timer managed by the link controller itself. If all resources requested in the submitted command are available at that time, the link controller sets up new communication links as many as the indirect objects in the command, after having obtained their access rights. If some of resources requested in the command have been in use by another process, the link controller waits until they are released within the limits of the time restriction. When the execution of the command terminates, the link controller falls asleep. On the other hand, if the link controller can not confirm the completion of the command within the limits of the time restriction, it makes the execution terminate and reports a time-out error to the network manager.

The only thing that the client of DRAON (i.e., the distributed operating system, and further, the application process) must do when it wishes to use the communication facility is to make a request in the form of a system command. Subsequently, DRAON automatically checks the status of the requested resources, if any, and the communication link(s) is established if all the resources are available at that time. The communication links are also automatically released and the resources become available for other processes when a current user completes its work on them.

Consequently, it is not necessary for the client of DRAON to take care to create mutual exclusion and to guard against the system deadlocks and, moreover, the client can use various services of DRAON only by the system commands. Since every system command has its maximum permissible time in which DRAON must start to carry out the command, DRAON recognizes the persistency of the process for that operation. Furthermore, real-time applications can expect the high-speed response to DRAON, since the network level solution for the deadlocks makes the overhead extremely low.

From another point of view, DRAON looks

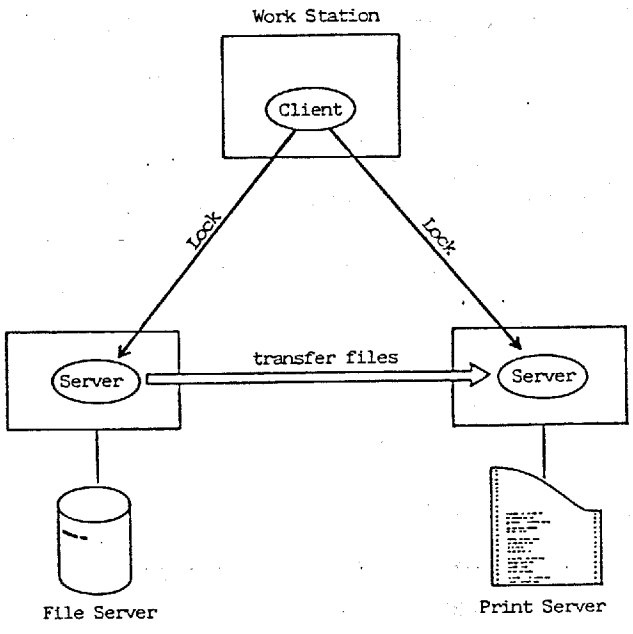


Figure 3 Remote File Transfer

In the latter approach, every application process that wishes to use more than one resources at a time must request them simultaneously, and no additional requests are never permitted before the release of the resources obtained previously, as shown in Figure 6. Therefore, the application process will obtain the access rights either for all resources that it has requested or for none of them. This strategy is called the lump-request lump-acquisition method which completely excludes the system deadlocks from DRAON.

The defect of this method is to restrict the behavior of application processes on accessing to the shared resources under the condition that the resources to be used together must be requested simultaneously. However, it is easier to implement this method in a distributed system than the former method.

3 Network Configuration

Figure 7 shows the DRAON network configuration, where DIN (DRAON Intelligent Node) is a network controller which carries out the communication protocols and realizes the system functions. It is generally composed of high-performance microprocessor(s), input/output controller(s), DMA controller, control memory, buffer memory, e.t.c.

Figure 8 shows the DRAON software hierarchy, where Data Transceiver provides an error-free communication channel, Link Controller provides reliable node-to-node communication, Network Manager is responsible for setting up, managing, and tearing down process-to-process connections, and also handles certain aspects of address conversion, access control, and recovery, SCI (System Command Interpreter) is a part of a distributed operating system, which performs high-level protocols mentioned in the following section, and the contents of Application Process includes to the users.

DRAON has various faculties on each DIN for its autonomous operations, described as follows:

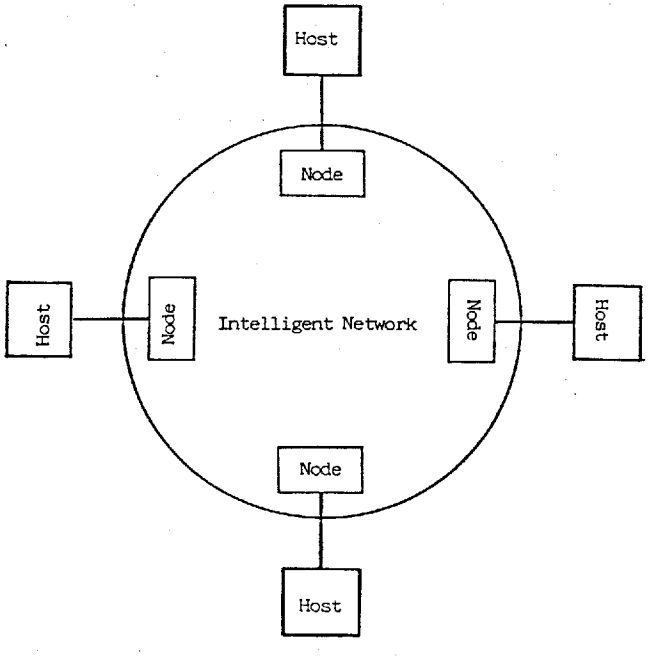
Real-time Clock

Every link controller has a real-time clock and manages it. A job command, which is submitted by an application process in the form of a system command mentioned in section 1, must be done within the time restriction of the command. Each link controller sets its own local timer, and after that, imposes time restriction on the following action. When time-out error occurs, the link controller terminates the execution of command, and notifies the occurrence of the failure to its client through the network manager.

If it is necessary, all local clocks on link controllers in a network are synchronized with each other by setting to the master clock.

Address Management

Since DRAON contains a number of various type resources, the designation of them based on their device names or physical addresses will decrease the flexibility and the usefulness of the system. Therefore, DRAON assigns a logical resource name to each actual shared devices at its system generation. The



Each node consists of a network interface and a communication processor.

Figure 7 DRAON Network Configuration

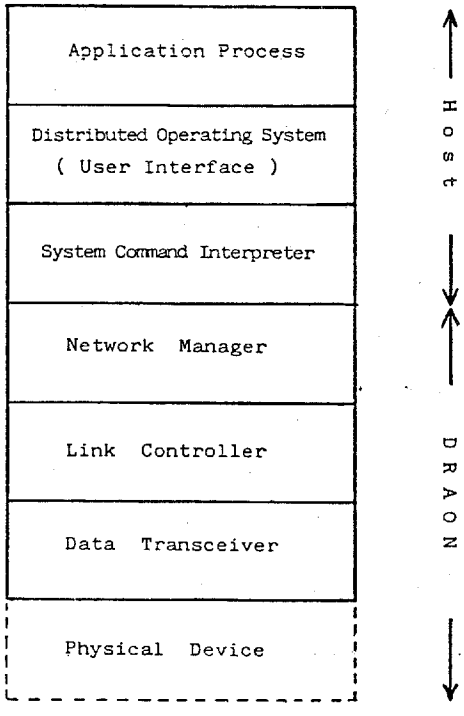


Figure 8 Software Hierarchy

like a command interpreter (see Figure 4) which accepts commands from its users, and which interprets, executes them, and subsequently returns the results, if necessary, as shown in Figure 2.

2 Access Control for Deadlock-Free Operations

Two access control strategies which prevent DRAON from falling into undesirable system deadlocks will be discussed in this section. One is a deterministic approach where the information table which maintains the access situation of all processes for every shared resource is used for forecasting deadlock situation, and another is a nondeterministic approach which is easily implemented at the cost of unrestricted access for the shared resources.

In the former approach, DRAON has a table called the Resource Access Right Table (shortly RART) which indicates what process is using each resource now, as shown in Figure 5.

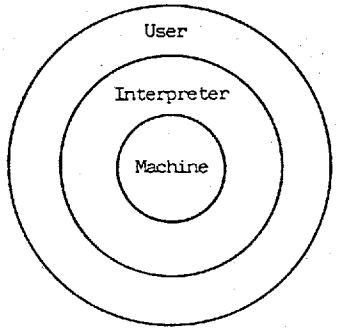
When an application process attempting to use shared resource(s) offers a request to the network manager, the table is checked by the link controller assigned to the process instead of the process itself. If the corresponding resource is available at that time, the status of the resource maintained in the table is changed from "idle" state to "in use" state. On the other hand, if it is not available (i.e., is in use), the application process waits until the request is accepted when the resource becomes available.

Therefore, since DRAON always recognizes the condition of resource assignment in the system, DRAON easily detects the critical situation before any deadlocks occur. If these situations occur, DRAON automatically tries to avoid them upon restricting the accesses for shared resources. Even in this situatiuon, the application processes do not have to deal with anything except their proper works.

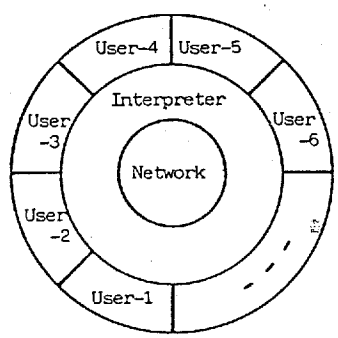
The defect of this approach are the high overhead and the difficulty for its implementation in distributed environment especially because of their complexity about the maintainance of consistency on the distributed multiple tables in the decentralized control scheme [1][2].

Resource	r_1	r_2	r_3		r_n
Status	p_8	idle	p_5		idle

Figure 5 Resource Access Right Table



Language Interpreter on Centralized System



Network Interpreter on Distributed System

Figure 4 Network Command Interpreter and Language Interpreter

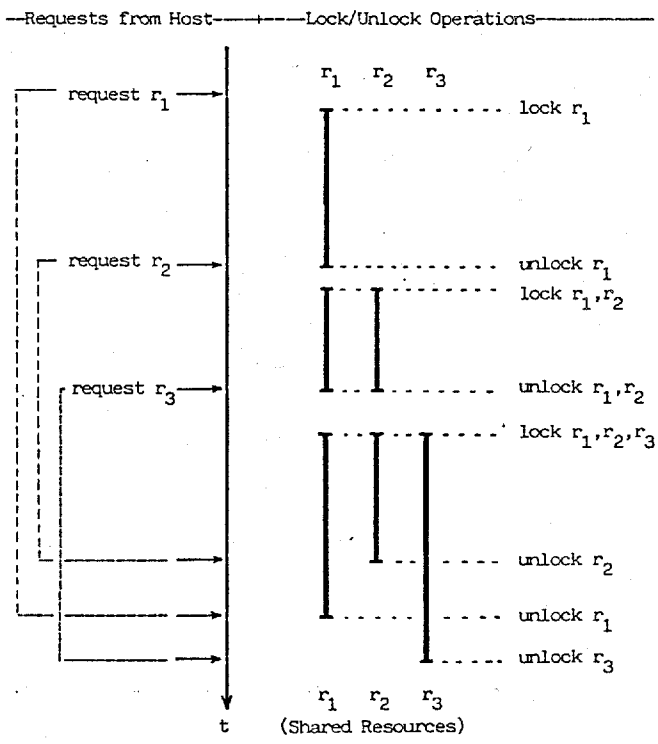


Figure 6 Lump-Request Lump-Acquisition Method

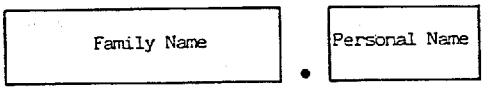


Figure 9 Logical Resource Name

i)	printer.A printer.B . . .	designates each resource
ii)	printer.	designates any one of the same devices
iii)	printer/2	designates any two of the same devices
iv)	printer.-	designates all of same devices

Figure 10 Designation of Resources

logical resource name which is unique in the system consists of the family name and the personal name, as shown in Figure 9, the former indicates the resource type (e.g., disk, printer, or processor), and the latter indicates individual attribute or identifier. This management enables application processes to designate the shared resources with some ambiguity, as shown in Figure 10. In this case, DRAON recognizes the logical name and converts it into an actual physical address using the address mapping table which maintains the correspondence between the logical names and the physical addresses. Figure 11 illustrates the address management of DRAON.

System Tuning

DRAON has a user-programable part in its network manager, because DRAON must be adapted to any type of underlying network and it has complicated functions which a type of DIN cannot provide all together. Subsequently, DRAON adapts itself flexibly to various application environments. At the system generation, a size of address mapping table, a set of functions which is provided, error recovery method, and access control method must be prescribed according to user's demands.

As the future subject, to make the best use of user-programable ability, a tuning support tool, which logs the frequency of failure on the establishment of communication links and of time-out errors, e.t.c., and supports the generation of the most desirable system according to a format chart that the users of system draws up for their requirements, will have to be developed.

As mentioned above, DRAON has the partial autonomic control structure which the conventional local area networks do not have provided. This structure introduces low system overhead, light load on hosts, and efficient resource utilization into distributed systems. Figure 12 shows the communication sequence between a host and a remote resource

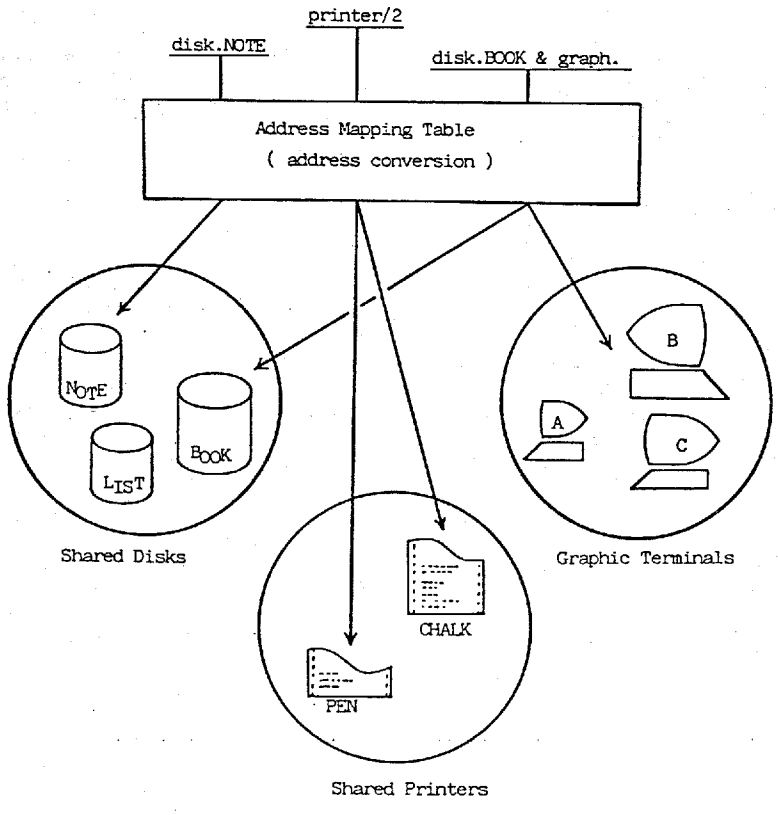


Figure 11 Address Management of DRAON

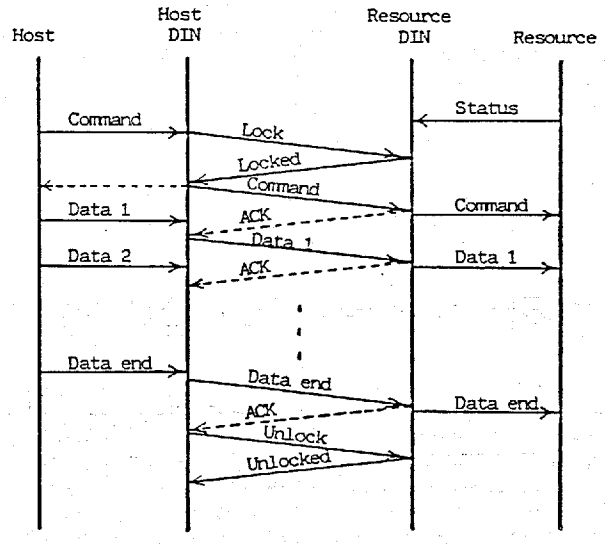


Figure 12 Communication Sequence

through DRAON, where it is obvious that DRAON reduce amounts of communication between a host and its network node.

4 Distributed System using DRAON

In this section, the construction of distributed system using DRAON is discussed. First, we have to consider what kind of system is most appropriate for being constructed using DRAON. DRAON has the characteristics which are the low overhead access control ability, and the well-defined interface to its upper layer. Therefore, a typical example of distributed systems using DRAON, which contains several work stations and various kind of shared resources, and offers useful abilities such as file transfer, resource sharing, electronic mail service, time-sharing and remote job entry service based on the processing server, and so on. Figure 13 illustrates a distributed system using DRAON.

Each work station [3][4] is usually composed of a super personal computer with a high resolution wide disply, Winchester type disk unit, pointing device, and communication interface. Each shared resource with controller is called a server as against a work station, which offers various services to users on the work stations, for example, file strage service by file servers [5][6], printing service by print servers, job entry service by processing servers, internetwork communication service by communication servers, and so on.

Users of the distributed system work anytime on data processing, document processing, data base retrieval, time sharing service and remote job entry, communicating other work stations and using the shared resources in the system.

On the other hand, each server provides these users with each special ability. The file server provides large-capacity storage ability, and it loads/saves the private program/data files according to the request from its users. The print server provides high-speed letter quality printing ability, the processing server provides high-performance processing ability in the form of time-sharing or remote job entry service, and the communication server provides reliable internetwork communication ability which enables its users to communicate with each other though the same or different type networks. Moreover, other information processing equipments just as plotters, OCRs, high-resolution graphic displays are considered as useful servers, which may provide the special data input/output ability for users.

In order to realize these systems mentioned above, a distributed operating system, as shown in Figure 8, has to provide the high-level protocols [7] such as file transfer protocol, virtual terminal protocol, interprocess communication protocol, remote job entry protocol, electronic mail protocol, and so on.

SCI and a operating system must provide these high-level protocols by request, since DRAON itself supports end-to-end data communications.

Then, more detailed explanation of each protocol is as follows [8]:

File Transfer Protocol

The most common uses of local area networks at present are for transferring files between stations. Since there is a need for programs to read a variety of incompatible files, systems must define a system standard format and provide a mapping from and to each existing file format. FTP provides this

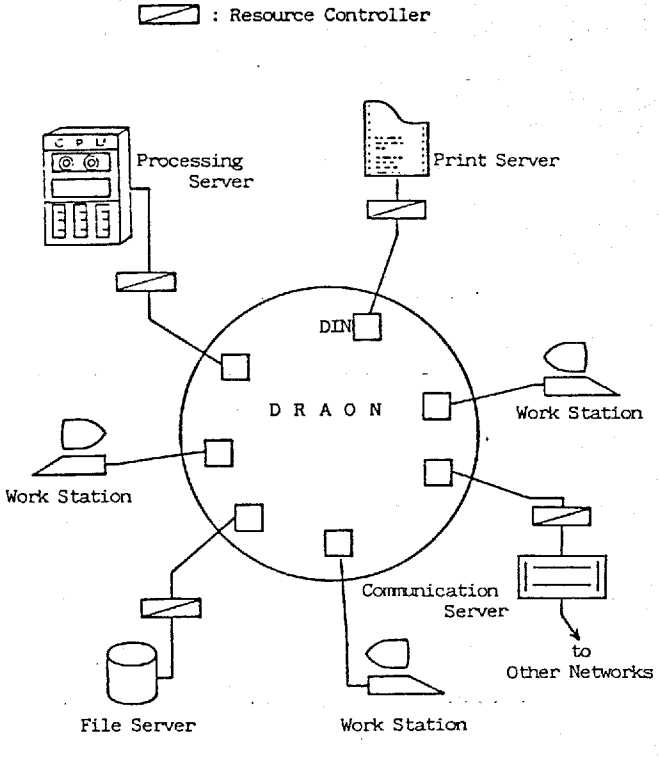


Figure 13 Distributed System on DRAON

conversion and other procedures in file transfer. Another aspect of file transfer is file manipulation such as create, delete, copy, append, rename on remote files. Most file transfer protocols on local area networks also support them.

Virtual Terminal Protocol

Since dozens of types of terminals may be used in a system, and no two of which are identical, systems must prevent such difficulties to hide terminal idiosyncrasies from application (i.e., user) programs. Therefore, systems employ protocols which are known as virtual terminal protocols and attempt to map real terminals into a hypothetical terminal. When this type of protocol is used, the designers invent a fictitious virtual terminal into which all real terminals are mapped. Application programs output vertual-terminal characters, which are mapped onto the real terminal's character set by the operating system at the destination.

Interprocess Communication Protocol

An application program on host may work communicating with other programs on remote hosts for data exchange or synchronization. IPC provides the data format, synchronization method, and procedures for the communications between application processes.

Remote Job Entry Protocol

When users have RJE service upon remote host machines, they have to transfer job-files, and subsequently receive the results of their jobs. RJEP defines the job-control language, file format, and entry procedures. RJEP is really supported by FTP to transfer job-files.

Electronic Mail Protocol

The electronic mail services enable users to create, edit various mail such as private, circular, bulletin, and to send or receive on normal or express delivery, and to save it on a file server, if necessary. EMP provides the mail format and procedures for systems.

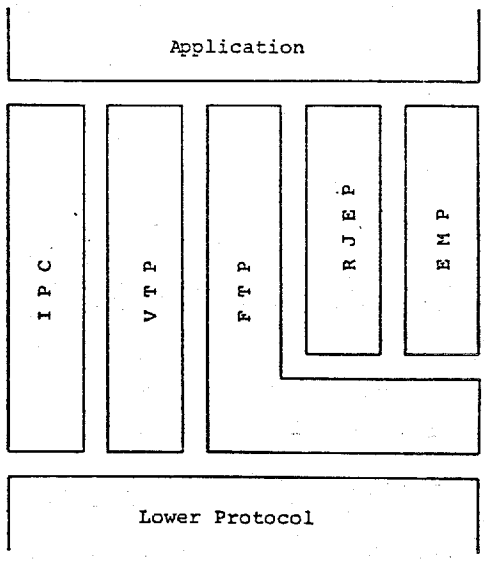


Figure 14 Relationship among High-Level Protocols

Figure 14 shows the relationship among these protocols.

Since these high-level protocols support the communication between the application processes (just as user process, server process, e.t.c.), the standardization of high-level protocols among the distinct networks enables the users to communicate each other through various networks.

In Figure 15, we demonstrate a prototype of DRAON which has been fabricated at our laboratory.

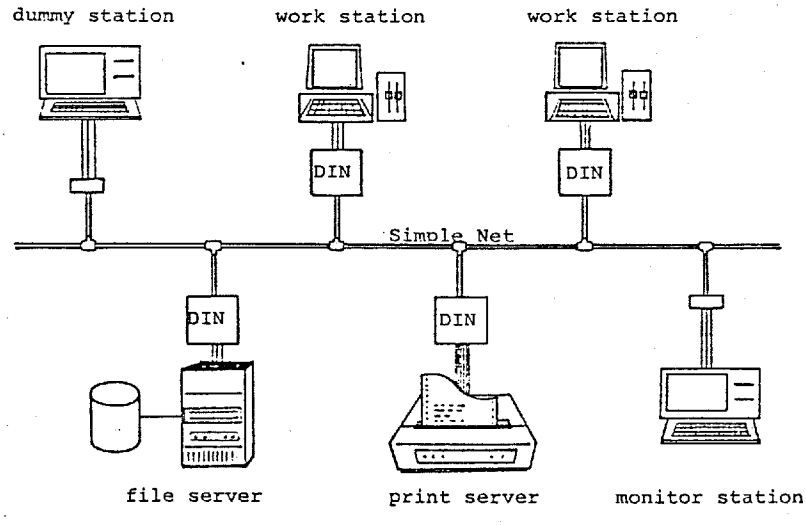


Figure 15 Configuration of Pilot System

References

[1] G.Gardarin and W.W.Chu; "A Reliable Distributed Control Algorithm for Updating Replicated Databases," Proc. Sixth Data Communications Symposium, Nov. 1979, California, pp.42-51.
[2] T.Hirota et al; "Protocol Assignment Method Using Logical Time in Multiple Data Access," 24th IPS of Japan, National Meeting, 2F-4, pp.461-462, Feb. 1982. (in Japanese)
[3] N.Kamibayashi; "Object Oriented User Interface Model," International Micro Computer Applications Conferrence, pp.25-36, Dec. 1982. (in Japanese)
[4] I.Ohta, et al; "Multi-functional Workstations," Journal of IPS of Japan, vol.24, No.10, pp.1247-1254, Oct. 1983. (in Japanese)
[5] A.D.Birrell and R.M.Needham; "A Universal File Server," IEEE Trans. on Software Engineering, Vol.SE-6, No.5, pp.450-453, Sep. 1980.
[6] M.Yoshida et al; "Implementation of File Server in a Local Network," 27th IPS of Japan, National Meeting, 4J-7, pp.839-840, Oct. 1983. (in Japanese)
[7] G.V.Bochmann; "Architecture of Distributed Computer Systems," Springer, Inc., N.Y. 1976.
[8] Y.Matsushita; "Computer Network," Baifuukan, Tokyo, 1983. (in Japanese)