(84.2.17) DRAFT ICCC'84

INI: Internal Network in the ICOT Programming Laboratory and its Future

----Impact of the FGCS on Future Communication Networks----

Akihito Taguchi, Nobuyoshi Miyazaki, Akira Yamamoto
Hajime Kitakami, Katsuzoh Kaneko, Kunio Murakami

Institute for New Generation Computer Technology
(ICOT)

Mita-Kokusai Bldg. 21F
4-28 Mita 1-Chome Minato-ku Tokyo 108 Japan
Phone:03-456-3192 Telex:ICOT J32964

Abstract

This paper is concerned with the INI local area network now under development by ICOT; it also discusses various aspects of future intelligent knowledge-communication networks related to FGCS knowledge processing.

INI is a compound network consisting of several ETHERNETs (#1) and bridges. A simple and unique bridge protocol has been developed for interconnecting ETHERNETs. INI is three-layered. Instead of the conventional session-oriented protocols, INI has a group-communication broadcasting protocol corresponding to the protocols of layers 4 and 5 in the OSI model. This innovation is advantageous in network management. The communication group is a kind of abstract data type, which is useful in certain applications.

INI is the basis of the programming support environment for developing FGCS software. Since programming is the inherent task of computers and computer networks, more intelligent supports will have to be provided for programming offices than those provided by conventional office automation.

#1: ETHERNET is a trademark of the Xerox Corporation .

TABLE of CONTENTS:

1. Introduction

INI (Internal Network in ICOT) is a local area network which is being developed by ICOT (#2).

INI consists of ETHER-cables, LIAs (LAN Interface Adaptors) and bridges. A bridge connects a pair of ETHER-cables. INI is three-layered. INI has group-communication protocol corresponding to that of layers 4 and 5 in the OSI reference model proposed by ISO. This is used instead of conventional session-oriented protocols for introducing network controls, for instance. A simple and unique bridge protocol has been developed for the routing function.

INI is intended to be used as a data-bus between the PSI super-personal computer (#3) and the DELTA relational database machine. INI is also the basis of the programming support environment for software development in ICOT.

#2: ICOT is the central organization of the Japanese Fifth Generation Computer Systems project, abbreviated FGCS.

#3: Refer to Section 3.1 for PSI and DELTA.

2. Overview of INI

2.1 Objectives

INI is now under development primarily as a data-bus for PSI, DELTA and other intelligent terminals, and as the basis of a programming support environment.

INI is also intended for use as an experimental network in the following areas:

- .Developing various network technologies, such as network management protocols, integrated network operating systems and protocol translation
- .Developing advanced office automation and intelligent programming support environment
- .Application of FGCS software technology to network technology, e.g. an expert system for advanced network management and a consultation system for configuring a network system

2.2 Network Configuration and Architecture

(1) Physical Configuration

As shown in Fig.2.2-1, INI is a compound network consisting of ETHER-cables, LIAs and bridges. A bridge connects a pair of ETHER-cables. An LIA interfaces locally with several attached intelligent terminals, and executes communication protocols as their front-end communication processer. One or more LIAs perform some aspects of network management and are therefore called NCLIAs (Network-Control LIAs). Gateway processers with protocol-translation capabilities will be required for interconnection with a global packet-switching network.

(2) Layers and Protocols

As shown in Fig.2.2-2, in contrast to OSI, INI has basically three

layers: physical, logical and application. The application layer, resident in terminals, may be divided into sublayers according to OSI. The protocols of the physical and logical layers are executed in LIAs, which also execute certain application-layer protocols.

.Physical-layer Protocol

The physical layer corresponds to OSI layers 1,2 and 3. Its protocol is based on ETHERNET [DIX1] and is enhanced to support the bridge function.

.Logical-layer Protocol

The logical layer corresponds to OSI layers 4 and 5. For this layer, INI has a unique new "group-communication" broadcasting protocol, which is used in place of the conventional session-oriented protocols. A logical communication mediun, called a "communication group", is formed among several processes, precluding the necessity of establishing a session between each pair of processes, as required in session-oriented communications. A message sent by any [redacted] process is broadcast to all members of the group. In this paper, the term "session" is used to a communication group consisting of only two processes.

(3) Addresses

Fig.2.2-3 represents the LIA, terminal and process addresses in INI.

(4) Logical Structure and Interface of LIA

Fig.2.2-2 represents the logical structure of LIA. The LIA logically contains dummy processes corresponding to application processes in local terminals; these execute logical- and physical-layer protocols. There is also an application process that manages the network. These processes communicate with each other through a pre-defined communication group.

There are physical- and logical-level interfaces between an LIA and its terminal. IEEE488 is available for the physical-level interface.

The logical-level interface is conceptually located between an application process and its dummy process. It is analogous to the logical-layer protocol, but much simpler so as to reduce the burden on application processes. Dummy processes provide their application processes with reliable, transparent communication media like virtual circuits, so that application processes basically need not be concerned with data transmission control. In the logical interface, application processes are able to designate other application processes using symbolic names instead of process addresses, as discussed in Section 2.4.

(5) Reasons for Introducing LIAs

.The LIA executes logical-layer protocol so as to reduce the burden on its terminals. Even if this protocol must be modified, the terminals need not be changed. For example, the data-transmission command may be modified in order to superpose information required for network control. Such protocol modifications are occasionally necessary in experimental networks such as INI.

.The LIA serializes simultaneous data-transmission requests from its terminals to reduce the possibility of collisions on ETHER-cable and the resultant reduction of INI efficiency.

.LIAs, where network-management processes are placed, will perform the processing required for intelligent networks and even for integrated network operating systems. As shown in Fig.2.2-4, they will be equipped with a knowledge-processing capability similar to that of FGCS.

.The LIA may serve as an intelligent terminal controller for protocol translation and media conversion.

2.3 Protocols for Bridges and Group Communications

2.3.1 Bridge Protocol

INI is a tree-structured compound network consisting of several

subnetworks (ETHER-cables), as shown in Fig.2.3-1. Bridges are required to interconnect the subnetworks. A bridge must be as small a device as possible and not reduce ETHER-cable transmission capacity. So, bridge protocol needs to be as simple as possible. Conversely, it must be sophisticated enough to detect ineffective packets passing through it, which otherwise tend to cause inefficiency when the multicast facility is used, as mentioned below.

(1) Objectives

INI is tree-structured using bridges for the following reasons:

.To localize experimental high-traffic communications within a subnetwork in order not to disturb other ordinary communications.

.To enhance the flexibility of coaxial cable for installation, thus economizing on cable requirements.

(2) Addressing for Bridge Protocol

INI makes good use of ETHERNET multicast facility for communications among the subnetworks. As described in Section 2.2, the LIA address consists of a subnetwork address and a node-address-in-subnetwork. Table-2.3-1 shows the LIA addressing for inter-subnetwork communications. It should be noted that this addressing is consistent with that of ETHERNET.

A bridge need not know the INI topology. For routing, it has only to know one given range of subnetwork addresses and the subnetwork addresses of two subnetworks connected to itself. However, this means constraining subnetwork addressing in such a way that all the subnetwork addresses of subnetworks beneath a bridge must be in the assigned range, as exemplified in Fig.2.3-1.

It is also desirable that LIAs and application processes need not be conscious of INI topology. In other words, both inter- and intra-subnetwork communications appear the same to LIAs and application processes. The LIA only needs to know the LIA addresses and the subnetwork address of its own subnetwork. It uses an

individual-LIA-address-inter-subnetwork as the destination when sending a packet to another LIA in a subnetwork other than its own. An individual-LIA-address-within-subnetwork is used when a packet is sent to another LIA in the same subnetwork. In all other cases, the broadcast-LIA-address is used.

(3) Bridge Processing

The above addressing scheme minimizes the routing overhead, because it does not require bridges to distinguish between communication groups or to perform address conversion. The following points describe bridge processing:

.Packets with Individual-LIA-Address-Within-Subnetwork

Bridges reject these packets.

.Packets with Individual-LIA-Address-Inter-Subnetwork

A bridge discards these packets, if they are received from the subnetwork above (or beneath) the bridge and their subnetwork addresses are out of (or within) the preassigned range of subnetwork addresses. These packets are ineffective.

A bridge sends out these packets to the opposite subnetwork after setting their multicast bit off, if their subnetwork addresses are the same as that of the opposite subnetwork linked to the bridge. Otherwise, a bridge allows these packets to pass through unmodified.

.Packets with Broadcast-LIA-Address

Bridges allow these packets to pass through unmodified.

.Packets with Broadcast-LIA-Address-Within-Subnetwork

This means the broadcasting within one subnetwork specified by the subnetwork-address field.

A bridge performs the same processing as packets described above, except when their subnetwork addresses are the same as that of the opposite subnetwork. In this exceptional case, where the opposite subnetwork is the destination, the node-address-in-subnetworks of

these packets are changed to the one reserved for this purpose.

For group communications, broadcasting over the entire INI is naturally desirable. However, the last case above(broadcasting-within-subnetwork) is useful for re-transmission on communication groups. This is currently under study.

The facility corresponding to ETHERNET's multicast group is not supported over the entire INI by the bridge protocol; this facility can be realized using group communications.

(4) Further Considerations of Bridges

When a bridge can receive no more packets because of its buffer-busy, it discards new packets without notifying the sending LIA. Therefore, the group communication end-to-end protocol of the logical layer must be responsible for detecting the loss of packets and for re-transmiting them. Because collision is the main cause of bridge buffer-busys, so the introducing of the priority ETHERNET [Kom1] might be effective to reduce the possibility of buffer-busys. The highest priority will be assigned to bridges. The gathering of statistics accumulated in bridges, such as the number of discarded packets and collisions, is also studied as one of the network management functions.

Bridges are obviously weak point in INI from the viewpoint of reliability. Dual bridges and control protocols, such as switchover and echo-check, are also under study.

2.3.2 Group-Communication Protocol

Group-communication protocol relates to the logical layer and is therefore executed in an LIA.

A communication group is formed among several processes. A message sent by one member process is delivered to all the other member processes.

(1) Objectives

For gathering and propagation of information for network management, group communication based on broadcasting is more appropriate than conventional session-oriented communications which are designed for one-to-one transmissions. For example, databases distributed within networks, containing network management information such as network configurations and program modules to be remotely loaded, can be simultaneously updated or queried by group communication. Moreover, group communication is useful for implementing basic integrated network operating system functions, such as resource-locking and process-synchronization.

Group communication also makes good use of the physical-layer ETHERNET multicasting protocol so as to cut down on the volume of transmission data; session-oriented communication does not take advantage of this powerful facility. Even in interconnecting with global packet-switching networks, their broadcasting facility are also utilized.

In addition, programming with communication group is powerful for applications such as those listed below, because communication group is an abstract data type representing an object consisting of several sessions.

.Distributed database control

.Distribution of electronic mail and online documentation

.Distributed processing (Distributed algorithm)

.Knowledge gathering for knowledge-programming support environment

(2) Main Functions of Group-Communication Protocol

Table-2.3-4 explains the functions of commands.

(a) Formation and Termination of Communication Group

Fig.2.3-2 shows the command sequence for forming a communication group; this sequence is initiated at the request of a process. A CGFORM command is broadcast to all the destination processes of the communication group, called member processes. The CGFORM contains the

list of process addresses and a CGID (Communication-Group IDentifier). A CGID consists of the LIA address of the requesting process and a unique number, and so corresponds to only one communication group. Commands other than CGFORM, and data transmited over the communication group, are destined for each member process according to their CGID. Each LIA maintains the correspondence between each CGID and the process addresses of member processes in its terminals.

Member processes may not join in the communication group by returning a CGFORM-NACK command. Therefore, the formed one may include part of the destined member processes.

Any member process can request the termination of the communication group. Some kinds of conditions and authorization for termination requests are maybe introduced in the application layer.

(b) Transmission of Data

Only functions essential for providing reliable communication media are defined; these include segmenting/assembling, acknowledgment, flow control, re-transmission and sequencing. However, additional supports for the application layer, such as transmission mode, bracket control and commitment control, are not defined. For example, only the simultaneous transmission mode (analogous to the full-duplex mode in a conventional session) is defined. Additional aspects of transmission mode may be introduced in the application layer.

The following discusses certain problems of re-transmission control, which are related to both group communication and bridge protocols. The sender process broadcasts a packet of data with the broadcast-LIA-address, then waits for the responses (DTRANS-ACK or DTRANS-NACK). The sender process may have to re-transmit the same packet to some member processes. Five cases are possible, re-transmission to each

.process,

.terminal,

.LIA,

.subnetwork, or

.group of subnetworks.

The first is the simplest and has been adopted for the current version of INI, but here, the number of re-transmitted packets increases. The second and third could easily be implemented by making the LIA a little more sophisticated. The fifth is not desirable, since LIAs must necessarily know INI topology. The fourth, in which the broadcasting-within-subnetwork is used as mentioned in Subsection 2.3.1, is now under study. This technique should be sufficiently effective to reduce the number of re-transmitted packets.

2.4 Network Management

This section describes some aspects of network management in the current version of INI which apply the characteristic group communication. It should be pointed out again that group communication is a kind of data abstraction and so each network-management process need not know the number of member processes.

Logically speaking, network management protocols are regarded as those on a pre-defined communication group among the network-management processes.

Table-2.4-1 shows the network management commands.

(1)Down-Loading Protocol

One or more NCLIAs maintain program modules and configuration information for each LIA.

Each LIA, except for NCLIAs, broadcasts a DL-REQ (Down-Load request) command to NCLIAs when powered or when its terminal inspection mode is needed. It is possible for the LIA to receive DLAVAIL responses from several NCLIAs. Then, the LIA sends a DL-STRT command to one NCLIA, usually within the same subnetwork. After receiving the DL-STRT command, the NCLIA network-management process,

called a down-load server, begins the down-loading procedure as shown in Fig.2.4-1.

If copies of LIA information are held in several NCLIAs, the failure of one NCLIA can be recovered by another which has received the broadcast DL-REQ command. It should be mentioned that group communication is useful in the simultaneous update of these copies.

(2)Name-Resolution Protocol

Application processes in terminals may use symbolic names to designate other processes, especially application-server processes. The LIA network-management process translates a process name into a process address by using the name-resolution protocol. (Otherwise, each LIA should know the correspondence between process names and process addresses of all the servers in INI.)

Each network-management process knows only the correspondence between the process names and process addresses of servers in locally-attached terminals. This correspondence is part of the configuration-information down-loaded by NCLIA. The network-management process broadcasts the specified process names through the communication group with an N-RESOL command to query their corresponding process addresses. The other network-management processes return the process addresses with an N-REP command when one or more of the process names specified in the received N-RESOL command are the same as those of server processes in their terminals. The status of server processes such as "busy" and "over-loaded" are also reported using N-RESOL commands.

The same symbolic name may be assigned to several server processes of the identical service, thus ensuring that a backup will be available in case of failure of any server process. By making intelligent use of status information reported in N-REP commands, the network-management processes would contribute to network-wide load balancing. This is the first step towards integrated network operating system.

3. Impact of FGCS on Future Communication Networks

3.1 Some Aspects of FGCS Knowledge-Processing

(1) Knowledge Processing and Logic Programming

The Japanese FGCS project (1982-1991) is aimed at creating new basic technologies required to build intelligent systems capable of knowledge processing. Such a system is called a KIPS (Knowledge Information Processing System).

In addition to the inference mechanism and knowledge bases, KIPS ("expert system" or "consultation system") must have a natural intelligent man-machine interface such as natural language and graphics. To accomplish this, a quantum leap is required towards the processing of the "semantics" of data from that of the "representations" of data. This means that KIPS must be capable of manipulating knowledges.

"Knowledge" is one higher/broader concept, including both program and data/database. Programs and data differ in representation rather than in semantics. Program and data, respectively, might be said to be intensional and extensional representations of the same knowledge. Knowledge bases will contain a wide range of knowledges. The retrieval of data and the computation of programs are integrated into the higher concept of "inferential problem-solving" through knowledge bases.

The programming language required for KIPS should be a very-high-level, non-procedural one with symbol-processing capabilities. A logic programming language based on PROLOG [Kow1] has been adopted as the basis of the principal programming language for FGCS. PROLOG makes it possible to declaratively express programs as knowledges as well as data bases.

Because they are high-level and have non-procedural semantics,

logic programming languages also seem to be adequate for applying such techniques as specification, automatic verification and automatic program synthesis for intelligent programming systems. These techniques have been studied in theory but not in practice, because it is very difficult to use them in conventional procedural languages such as FORTRAN. Intelligent programming systems will possess and manipulate such knowledge bases as program-package libraries and so on.

PROLOG will also regulate most of the software and architecture of fifth generation computers. So, this principal language is called the "Kernel Language" and regarded literally as the "kernel" of the FGCS project.

The architecture of fifth generation computers should be higher-level-programming-language-oriented and far different from conventional von Neumann architecture. Two types are being researched and developed based on PROLOG machine: PIM (Parallel Inference Machine) based on reduction/data-flow mechanisms; and PSI (Personal Sequential Inference machine). Moreover, DELTA, a relational database machine, has been developed as the first step towards a knowledge-base machine.

Refer to [Yok1] for further details on the overall FGCS project.

(2) New Software Technology for FGCS and Networks

The same philosophy and technology as those for FGCS will be adopted in various fields of information processing. For example, future intelligent networks will provide intelligent services through KIPSs, and should employ the same kind of intelligent processing for themselves. Table-3.1-1 shows the correspondence between new software technologies and existing communication network technologies.

3.2 Prospects for Future Intelligent Networks

(1) Intelligent Network Control

Intelligent networks will have the capability of autonomic self-control such as adaptive routing and adaptive flow-control. These intelligent controls will be performed by KIPSs installed in the network.

To realize such controls, gathering of statistics from various points within the network and propagation of instructions, such as certain of IBM SNA commands [IBM1], are essential. The group communication proposed in this paper is useful for such controls, too.

The extra transmission required for these purposes will be rationalized and not disturb normal service transmissions, firstly because of the future use of high-capacity transmission media such as optical fibre, and secondly because the network load level (transmission rate) will need to be held far below the saturation point in order to guarantee reasonable throughput and transfer-delay [Bux1] [Koi1].

(2) Knowledge-Communications

Intelligent networks of the future generation will provide users with knowledge communications instead of conventional data communications while fifth generation computers will perform knowledge processing instead of data processing. In other words, they will transmit not "representations" but "semantics".

Knowledge communication means, firstly, intelligent services provided by intelligent servers and knowledge bases around the network, and secondly, the conversion of representations of knowledges such as protocol translations and media conversions. These will be much more intelligent than conventional capabilities, such as code-conversion and data-compression.

3.3 Prospects for Future Programming in Network Environments

(1) Future Programming

Most future programming will be "knowledge programming", based especially on logic programming, in which programmers will use a PWB (Programmer's Work Bench) to construct models and logic of their programs by using various knowledge bases on networks [Ste1]. In a future intelligent programming system as KIPS, knowledge bases will hold all knowledges necessary to programming, e.g. languages, methodologies and specifications of existing program packages. Moreover, they will have domain-oriented knowledges, such as experiential knowledges, experimental results, investigation findings and data maintained in individual divisions of organizations.

These knowledges are represented in logic programming languages and relational models, both of which are based on first order logic.

(2) Advanced Man/Network Interface

PWBs, such as PSI and STAR, will provide advanced man/machine interfaces and serve as interface ports [Tag1] [Smi1]. Moreover, the visualization of the entire network, including knowledge bases (remote or distributed on networks), is essential for future programming. Thereby, a programmer may simultaneously pay attentions to any knowledges in the network so as to facilitate creative programming. A.Macdonald also proposes "visual programming" [Mac1].

For network visualization, two multiplexing techniques, the multi-window display and the virtual circuit(or session) are primitive, but essential. As shown in Fig.3.3-1, several knowledge bases can be seen simultaneously. Communication groups and logically related sessions [Bre1], which evolved from virtual circuits, are more elegant, advanced concepts for realizing the simultaneous inquiry and gathering of knowledges.

(3) Program Transmission Considerations

One of the main problems in existing programming support systems is the difficulty of retrieving appropriate reusable modules from program libraries. This is because similar modules are not well clustered in the libraries. One approach to improving clustering is generic programming or parametrized programs [Arc1]. Using these techniques, an individual program may include all similar functions in itself, and so is fairly large. For example, TRIGO program may contain all trigonometric functions, and SORTER may sort a list of any type according to a given order-relation including ordinal- and alphabetical-orders.

Consequently, Knowledge gathering for knowledge programming will increase the network traffic tremendously, especially for the transmission of programs to be re-used. Therefore, a technique such as "partial evaluation" of knowledge (program) will be required to reduce volume of transmissions.

As formalized below, partial evaluation is a technique for automatically generating an optimized specific version of a generic program, a concept similar to ADA's "generic package".

```
.-------------------------.
|  TRIGO(sin,X)=TRIGO.sin(X)  |
|-------------------------|
```

In the above example, the specific version "TRIGO.sin(X)" is generated from the generic "TRIGO(T,X)" with respect to one given argument "sin". "TRIGO.sin(X)" is equal to the well-known trigonometric function "SIN(X)". It should be emphasized that logic programming languages are appropriate for this technique [Fut1]. The projection and selection of relational algebra might be said to be a kind of partial evaluation of knowledge represented in a relational model; it is also useful in reducing volume of transmissions.

(4) Programming Support Environments on Networks

The network environment can assist programmers with online documentation, communications between programmers, the exchange and brew of knowledges, and so on, as well as other intelligent office automation systems.

Programming might be said to be the native task of computers and computer networks, because its output is itself executable, testable and effective on computers again. Also, parts and instruments used in this task are available on computer network environments. Consequently, intelligent support environments for programming offices will be able to and have to provide much more than conventional office automation on internal networks. For example, a program made at a PWB (such as PSI) may not be efficiently executed on it; it must be sent to a suitable computer, such as a conventional FORTRAN machine in the network, to be tested. In future, the intelligent network environment will provide a kind of virtual machine that simulates an integrated computer equipped with an inference machine, a database machine and a super-computer, or which simulates a multiprocessor (such as PIM), by connecting various resources around the network. Such virtual machines will be useful for the development of practical KIPS applications, such as a diagnostic expert system for a nuclear reactor.

3.4 Design Guidelines for INI/CC Consultation System

INI/CC (Consultant for Configuring INI) is a consultation system, an application of KIPS, for determining the configuration of INI.

It should be noted that the FGCS Kernel Language based on PROLOG (mentioned in Section 3.1) may be viewed as the first approximation to a KIPS. It possesses primitive but essential knowledge-processing capabilities, such as inference function and symbol processing, and it can also declaratively express various types of knowledge. The authors mean to feed it experiential knowledges gained in the course of using INI in order to construct INI/CC. The guidelines for constructing INI/CC are briefly described below.

(1) INI/CC is used to determine the following:

- .Location of LIAs
- .INI topology (location of bridges and repeaters)
- .Subnetwork addresses
- .Range of subnetwork addresses assigned to each bridge
- .Disposition of cables
- .Location of server processes

(2) INI/CC takes into consideration the following:

- .Characteristics of resources(LIA, Repeater, Bridge, Cable)
- .Layout of rooms and desks
- .Locality of communications
- .Load on server processes
- .Load on bridge
- .Number of collisions

Various similar knowledge-based systems have been proposed, e.g., R1, for configuring VAX-11 computer systems [Mcd1], and ACE, for the maintenance of telephone-networks [Ves1].

4. Conclusion

In this paper, it has been shown that group communication and the LIA are useful for achieving intelligent network management, and that they are also powerful tools for visualizing the network so as to provide a better programming environment.

At present, the authors have completed the detailed design for group communication and network management and are implementing them. The bridge is in the detailed design phase.

In future, the fruits of the FGCS project should be incorporated into network technologies. While fifth generation computers are due to the surprising evolution of VLSI, the intelligent networks of the new generation will be made possible by optical fiber technology in addition to the FGCS technology.

5. Acknowledgment

The authors thank I. Yoshida and Y. Matsumura of OKI Ltd. for their cooperation, especially in the development of INI group-communication protocol, which is based on BANET that is primarily aimed at distributed database control [Yam1]. The authors also express their gratitude to Kazuhiro Fuchi, Director of ICOT Research Center, for providing the opportunity of this work.

REFERENCES

[Arc1] Archibald,J.L.,et.al.,"Abstract Design and Program Translator; New Tools for Software Design",IBM System Journal Vol.22 No.3,1983

[Bac1] Backus,J.,"Function Level Computing",IEEE Spectrum, Aug. 1982

[Bre1] Brenner,J.B.,"Preliminary View on Administration and Use of Logically Related Sessions",Contribution to ISO/TC97/SC16/WG2-no16,Aug. 1978

[Bux1] Bux,W.,"Local Area Subnetworks:A Performance Comparison",IEEE Transactions on Communications Vol.COM-29 No.10,Oct. 1981

[Cod1] Codd,E.F.,"Relational Database:A practical Foundation for Productivity",CACM Vol.25 No.2,Feb. 1982

[DIX1] DEC,Intel and Xerox,"The Ethernet,A Local Area Network,Data Link Layer and Physical Layer Specification,V 20",1983

[Fut1] Futamura,Y.,"Partial Computation of Programs",Lecture Notes in Computer Science 147 (Springer-Verlag),1983

[IBM1] IBM,"System Network Architecture Format and Protocol Reference Manual:Architecture Logic",IBM Form SC30-3112,Sept. 1979

[Koi1] Koide,S.,et.al.,"Performance of Network Architecture"(in Japanese),Information Processing Vol.25 No.1,Jan. 1984

[Kom1] Komachi,Y.,et.al.,"Priority Ethernet Modified with Reassignment Algorithm"(in Japanese),Journal of the Institute of Electronics and Communication Engineers of Japan Vol.J66-D No.1,Jan. 1983

[Kow1] Kowalski,R.A.,"PROLOG AS A LOGIC PROGRAMMING LANGUAGE",AICA, Sept. 1981

[Mac1] Macdonald,A.,"Visual Programming",Datamation Vol.28 No.11,Oct. 1982

[Mcd1] Mcdermott,J.,"R1: A Rule-Based Configurer of Computer Systems", Artificial Intelligence 19,1982

[Oht1] Ohta,K. and Oh-hara,Y.,"A Proposal for a Design Methodology for Conversion Algorism"(in japanese),Proc. of 25th National Conference of Information Processing Society of Japan,Mar. 1983

[Schi1] Schindler,S.,"OPEN SYSTEMS MANAGEMENT -A Tutorial Elaboration on the ISO Approach-",Conference on DATA and TELE COMMUNICATIONS INTERNATIONAL-JAPAN'82 ,Jan. 1982

[Schu1] Schultz,G.,et al,"Executable Description and Verification of SNA",IEEE Transactions Vol.COM-28 No.4,Apr. 1980

[Smi1] Smith,D.C.,et.al.,"Designing the Star User Interface", Byte, April 1982

[Ste1] Stefik,M.,et.al.,"Knowledge Programming in LOOPS",AI MAGAZINE Fall 1983

[Tag1] Taguchi,A.,et al,"Configuration and Design Philosophy for an Operating System of Personal Sequential Inference Machine(SIM)"(in

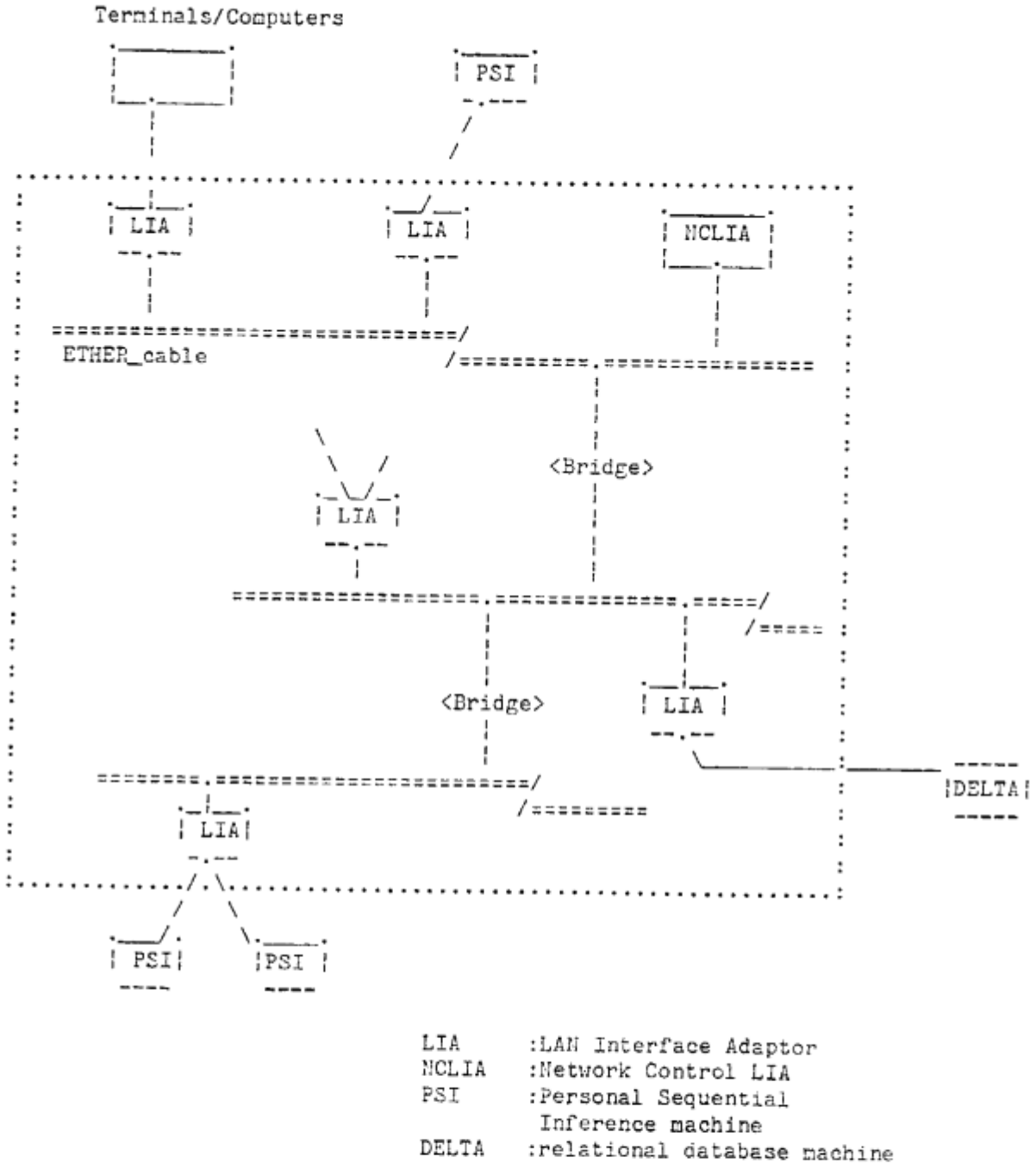
Japanese),Proc. of 26th National Conference of Information Processing Society of Japan,Oct. 1983

[Tak1] Takahashi,Y.,et al,"Protocol Description and Verification Techniques"(in Japanese),Information Processing Vol.20 No.7,July 1979

[Vec1]] Vezonder,G.T.,et.al.,"ACE:An Expert System for Telephone Cable Maintenance",Proc. of IJCAI'83,Aug. 1983

[Yam1] Yamazaki,H.and Yoshida,I.,"A Proposal for Broadcast Architecture Network(BANET)",Proc. of ICCC'82,Sep. 1982

[Yok1] Yokoi,T. "A Perspective of the Japanese FGCS Project",ICOT TM-0026, Sept. 1983



LIA :LAN Interface Adaptor
NCLIA :Network Control LIA
PSI :Personal Sequential Inference machine
DELTA :relational database machine

Fig.2.2-1 Physical Configuration of IMI

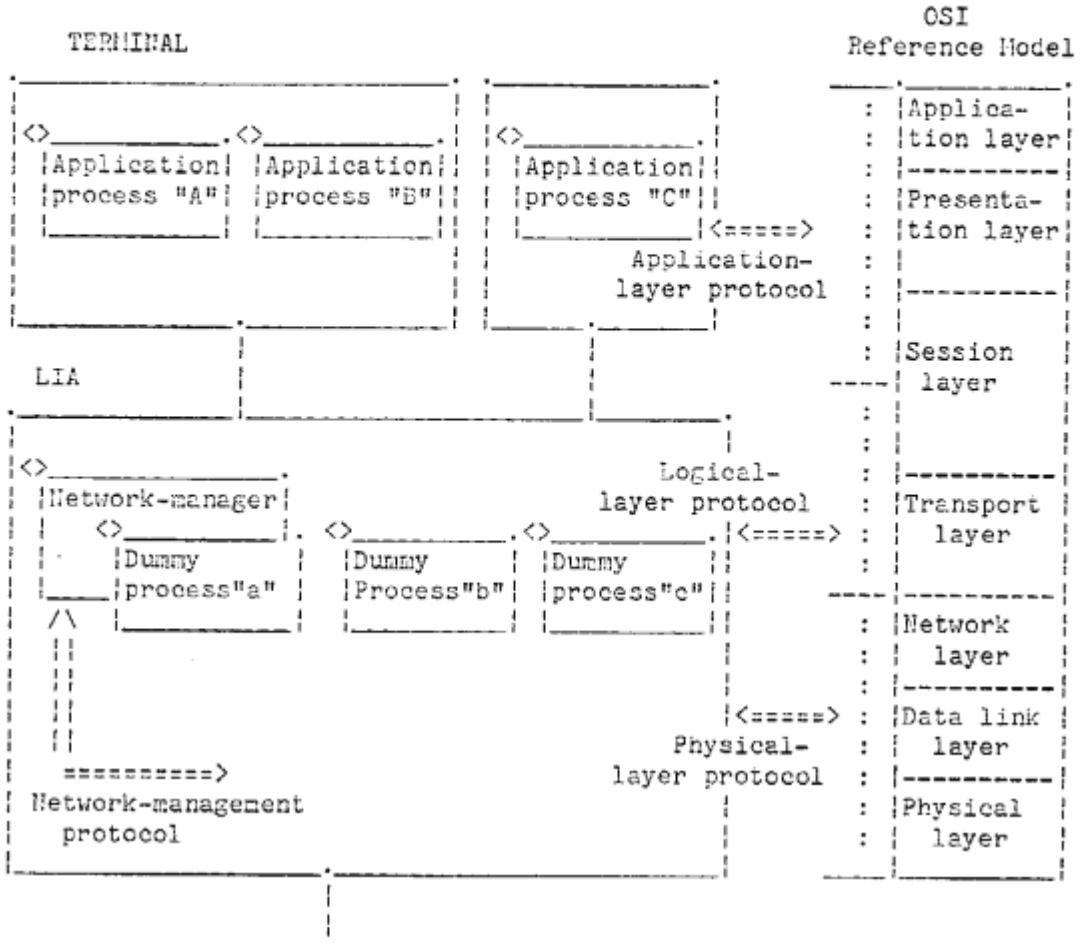


Fig.2.2-2 Logical Structure of LIA (in contrast to OSI Reference Model)

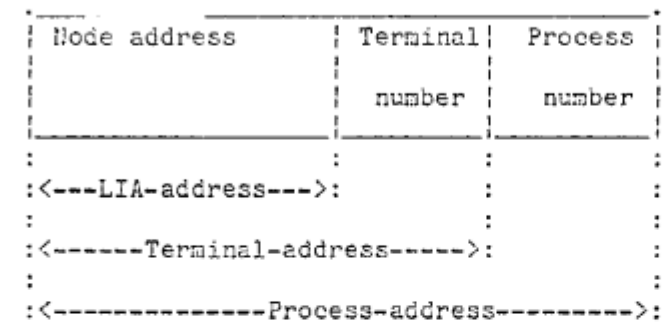


Fig.2.2-3 INI Addresses

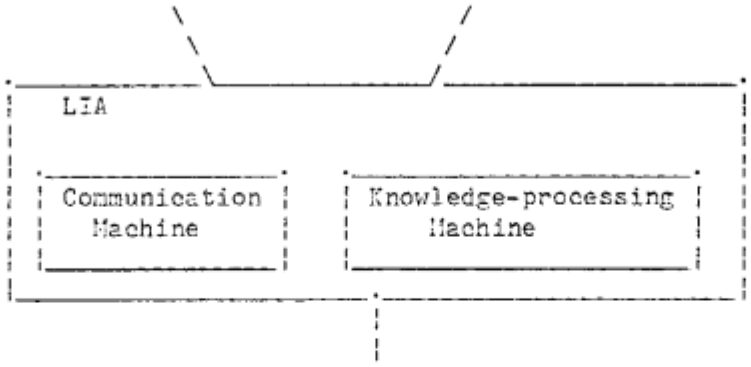


Fig.2.2-4 Future Intelligent LIA

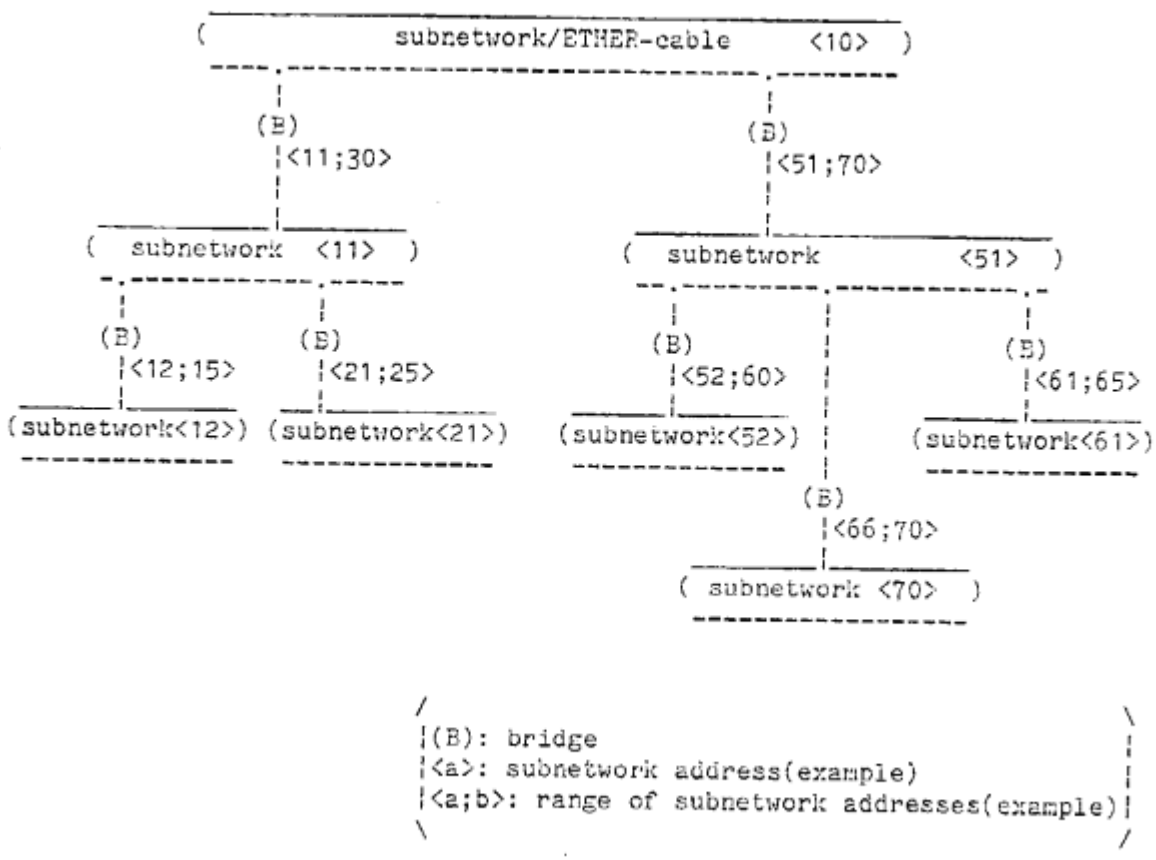


Fig.2.3-1 INI Topology

Table 2.3-1 Addressing for Bridge-Protocol

Type of LIA address	Destination address field of ETHERNET packet header			Meaning
		LIA-address		
	Multicast bit	Subnetwork address	Node-address-in-subnetwork	
Individual-LIA-address-within-subnetwork	off	specified	specified	Designates an LIA in the same subnetwork as the sending LIA.
Individual-LIA-address-inter subnetwork	on	specified	specified	Designates an LIA in a subnetwork other than that of the sending LIA.
Broadcast-LIA-address	on	Broadcast-address (Defined in ETHERNET)		Designates all LIAs in INI.
Broadcast-LIA-address-within-subnetwork	on	specified	not specified	Designates all LIAs within a subnetwork.

Table 2.3-4 Principal Commands of Group-Communication Protocol

Phase	Command-Name	Function	Principal Parameter
Communication-group-forming	Communication-group-formation (CGFORM)	Request to form a communication-group	.Process-addresses of member processes of the group .Process-address of requester .CGID of communication-group .Profile of communication
	Communication-group-acknowledgment (CGACK)	Affirmative response to CGFORM command	.Process-address of requester .Process-address of responder .CGID specified in CGFORM
	Communication-group-negative acknowledgment (CGNACK)	Negative response to CGFORM command	.Reason-code for rejection, in addition to those of CGACK command
	Communication-group-member (CGMEMBER)	Announce member processes to all the rest of the group	.Process-address of requester .Process-addresses of member processes of the group .The same CGID as CGFORM
Communication-group-dropping	Communication-group-drop (CGDROP)	Request to end communication-group	.Process-address of requesting member process of the group .CGID of group to be ended .Reason-code for termination
	Communication-group-drop-acknowledgment (CGDROP-ACK)	Response to approve CGDROP	.Process-address of requester .Process-address of responder .CGID
Data-sending	Data-transmission (DTRANS)	Transmission of data	.Process-address of sender .CGID .Sequence number .Process-address of receiver (For re-transmission only)
	Data-transmission-acknowledgment (DTRANS-ACK)	Response to DTRANS (Data received)	.Process-address of sender .Process-address of receiver .CGID .Sequence number of last received data
	Data-transmission-negative-acknowledgment (DTRANS-NACK)	Response to DTRANS (Data not received)	.Reason-code for rejection or discard of data
	Cancel-chain-of-data (CANCEL)	Request to discard chain of data to be received	.Process-address of sender .CGID .Process-address of receiver (For re-transmission only)
	CANCEL-acknowledgment (CANCEL-ACK)	Affirmative response to CANCEL command	.Process-address of requester .Process-address of responder .CGID

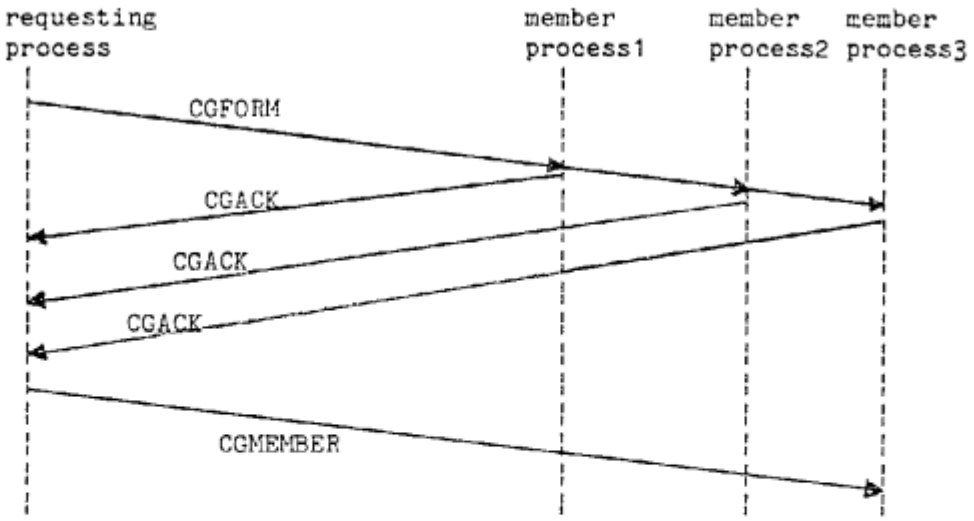


Fig.2.3-2 Command Sequence for Forming of Communication-Group

Table 2.4-1 Principal Commands of Network-Management Protocol

Protocol	Command-name	Function	Principal parameters
Down-loading	Down-loading request (DL-REQ)	Request down-loading to down-load servers	.LIA-address of requester as LIA-ID .CGID (pre-defined)
	Down-loading-available (DLAVAIL)	Response for DL-REQ to requester	.Process-address of down-load server .LIA-address of requester .CGID
	Down-loading-start (DL-STRT)	Request the server to start down-loading	.Process-address of down-load server .LIA-address of requester .CGID
	Down-loading-data (DL-DATA)	Transfer information down-loaded to the requester from the server	.Process-address of down-load server .LIA-address of requester .CGID .Information down-loaded .Loading-location within memory of requester LIA .First/middle/last flag .Start-location for execution (Only in last DL-DATA)
	Down-loading-complete (DL-COMP)	Acknowledgment from the requester to the server	.Process-address of down-load server .LIA-address of requester .CGID
Name-resolution	Name-resolution (N-RESOL)	Query process-addresses for process-names	.Process-address of requester .CGID (pre-defined) .List of process-names
	Name-reply (N-REP)	Reply to N-RESOL to provide process-addresses	.Process-address of requester .CGID .List of process-addresses/process-names .Status information of server processes

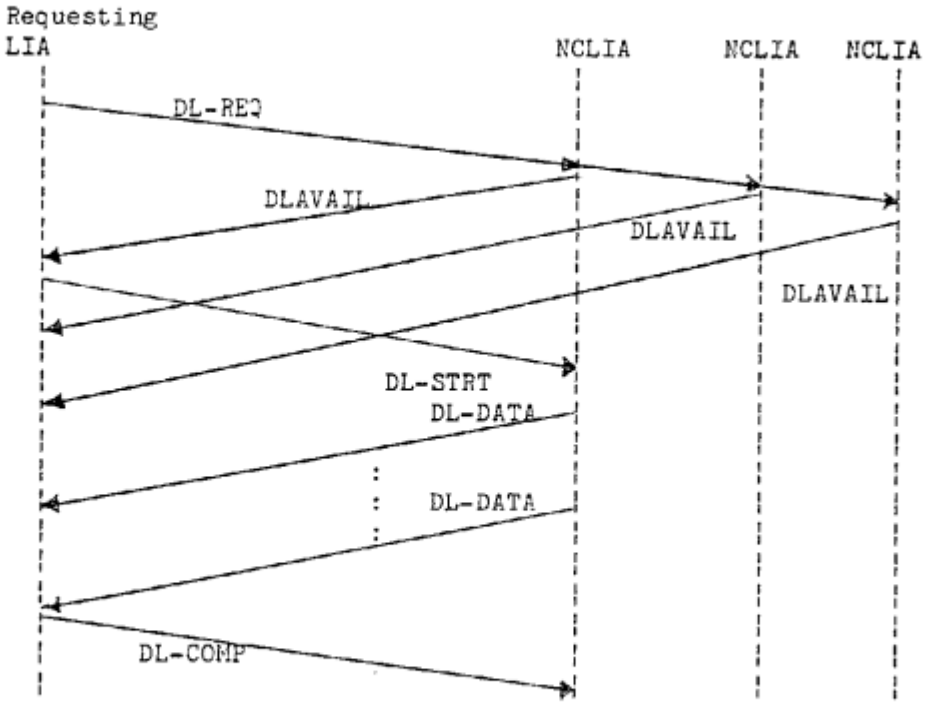


Fig.2.4-1 Command Sequence for Down-Loading

Table 3.1-1 Relation between New Technologies and Network Technologies

New/future technology (mainly software technology)	Technology in network	
	conventional technology	Field of applications

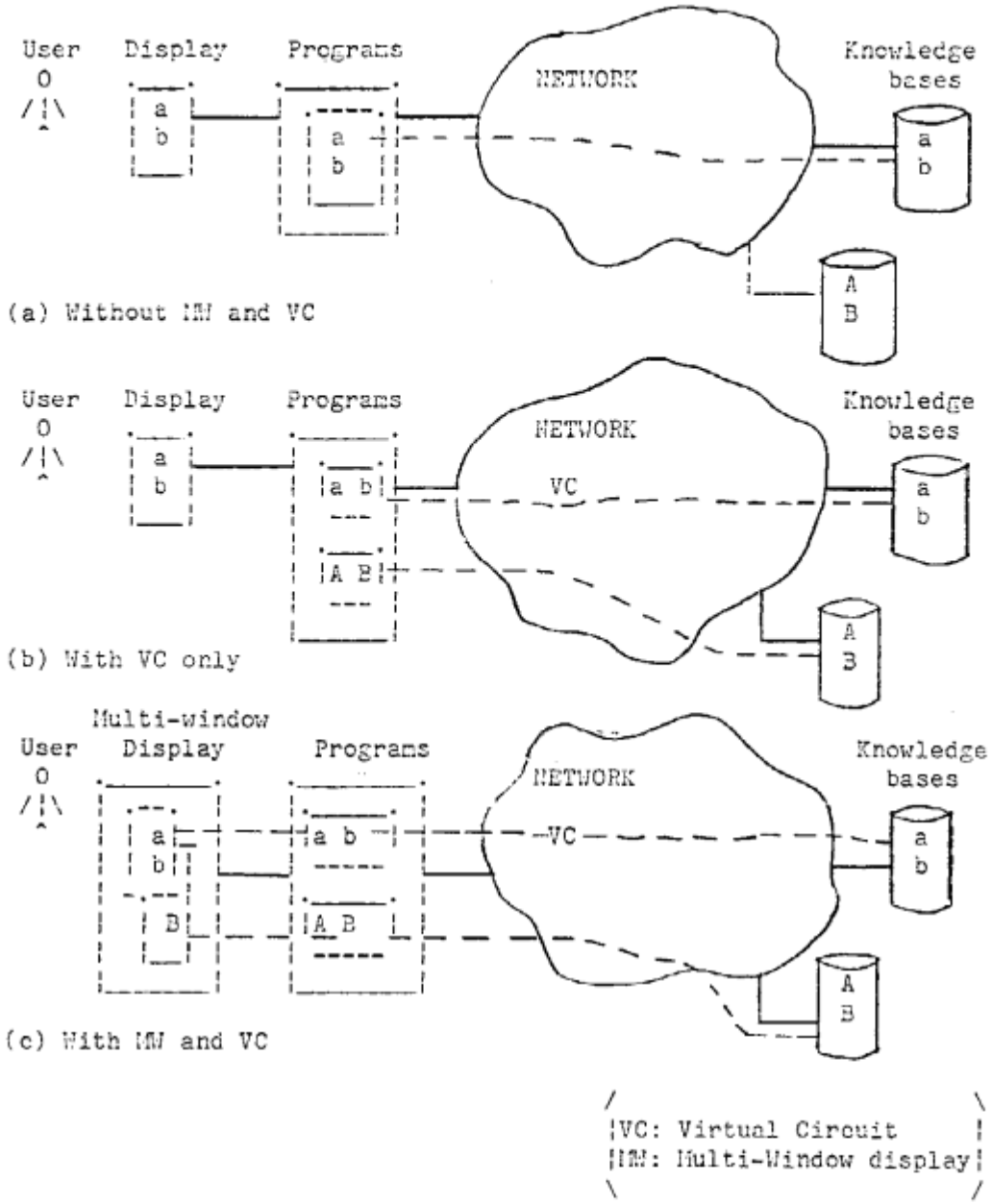


Fig.3.3-1 Visualization of Network Environment