RPCNET: ARCHITECTURE AND SERVICES

F. Caneschi
E. Ferro
L. Lazzeri
L. Lenzini
M. Martelli
C. Menchi
M. Sommani
F. Tarini

CNUCE, Istitute of the National Research Council, Pisa,
ITALY

1. Project History and Objectives

REEL (REte di ELaboratori), a project to investigate concepts and experimental solutions for distributed processing problems, was formally established in June 1974 (Lenzini /1/), as a collaboration among the following Italian institutions:

- CNEN, Division for the Management of the Information Systems, Bologna.
- CNR, CNUCE Institute, Pisa.
- CSATA, Center of Studies for Advanced Technological Applications, Bari.
- IBM, Scientific Center, Pisa.
- University of Padua, Computing Center.
- University of Turin, Computing Center.
- (In 1975 the IBM Scientific Center in Venice joined the project, not as a project partner but as a user of the network facilities).

The result of this project was the design and implementation of a packet switching distributed control network named RPCNET (REEL Project Computer NETwork).

The RPCNET design was determined by the principal requirements of the REEL project partners:

a) Minimum of additional hardware

The minimum hardware required to get into to the network constituted by the appropriate features to drive telecommunication lines at a convenient speed (full duplex binary synchronous from 1200 to 9600 bps) and by a (single) intelligent processor on which to implement the RPCNET functions. The utilization of a separate processor (a Front End Processor or a Telecommunication Processor) is not mandatory.

b) Partner independence

Each center's should retain control of any resources placed at the disposal of the other network partners. Thus, each center can dynamically attach to/or detach from the network. The network should thus be able to reconfigure itself in a highly automatic way.

c) Minimum impact on the existing operating systems

The introduction of the new network capabilities into the operating system of each participating center should neither disrupt the user community nor the system maintenance but should act as a straightforward extension of the centers' existing concepts and facilities.

d) The network should not be limited only to terminal handling and/or to Remote Job Entry traffic

2. Network Architecture

RPCNET is conceptually divided into three layers (Fig.1.a):

Common Network

This layer provides the packet switching capability which drives the physical communication medium by using routing and reconfiguration strategies together with a well defined packet format.

The Common Network does not guarantee that packets will be delivered in the same order in which they were sent, nor does it guarantee that there will not be loss or duplication of packets. These losses arise from such factors as the use of alternative routes for packets between nodes in case of failure, and retransmission on data links when errors are detected. As shown in Fig.1a the Common Network is structured into a Connections Network and a Communications Functions layer.

The Connections Network provides the physical communication medium capable of carrying data from one processor to an adjacent one. The Connections Network is made up of full duplex circuits called Connections. These can be point-to-point synchronous data links or channel attachment subchannels.

The internal structure of the Communication Functions layer is shown Fig.2. The Network Connection Handler (NCH) is in charge of sending and receiving packets between two adjacent nodes. All NCHs have three control phases (a) the connection making phase for establishing a usable connection, (b) the data transfer phase, whose protocol details are hidden from the user of the NCH, and (c) the connection release phase. The control procedures for the data transfer phase are independent of the other two phases. For synchronous data links, NCH employs a logical protocol which is designed to detect transmission errors and to invoke retransmission of packets when any errors are detected. NCH supports an input queue for each Connection from which packets are taken for transmission.

The Packet Switcher (PS) routes packets towards their destinations by using a routing table that contains, for each destination, the corresponding NCH input queue. When a packet reaches its destination node, the packet is given to the Interface Function layer for further analysis.

The Common Network Manager (CNM) manages and updates the routing table for the node. To do this, packets are sent among CNMs in order to propagate topology changes through the network. Topology changes take place when activation and de-activation of connections or nodes occur.
Obviously this can be a consequence of a network operator intervention or of failure conditions.

Interface Functions Layer

This is the layer that provides the interface between the Common Network and the layer in which the network users reside (Applications layer).

Interface Functions basically provide: a) defining and undefining ports (or Logical Unit/LU) on the Communication System, b) extemporaneous services, such as query and mailing, to the end user, c) building and cancelling Logical Channels between Applications. These functions are performed by a component called Network Services Manager (NSM) (Fig.2). Once established, logical channels are maintained by a software component called Session Handler (SH) (Fig.2) which is in turn made up of three modules; Presentation Services (PR), Data Length Adapter (DLA) and Data Flow Control (DFC). The functions provided by the Session Handler are converted by the PR into messages sent via a full duplex message facility provided by the DLA module. The DLA disassembles messages into packets for transmission, and conversely reassembles received packets into messages and communicates through the DFC to the DLA at the other end of the Logical Channel. The DFC provides a window oriented scheme for pacing and error detection in case of possible loss of packets. The DFC is directly addressable by the Common Network for each defined logical channel in the node. Logical channels are the normal means of communication between processes using the network. The RPCNET Logical Channel is designed for optimum communication in one direction at a time, although the protocol allows messages to be sent in the opposite direction if they can fit into a single packet. The optimum direction can be switched by the Application currently in the sending state.

The Session Handler provides for error detection in case of loss of messages as well as for reordering the incoming messages in the correct sequence. Applications must accomplish error recovery when a message is lost. It is normally the sender who can most easily recover in the case of message lost.

It is worth noting that the first implementation of RPCNET was intended to insure that message loss occurred only with the loss of a network node containing the message. Also, the first routing algorithm minimized out of sequence messages. Thus although the error recovery is necessary, it is seldom invoked except to prevent "hang conditions" in the rare case of a crash of a node holding a message relevant to the host and terminal interconnection.

During a Session, Applications exchange units of information called BIUs (Basic Information Units), which consist of two parts: RU (Request/Response Unit) containing that part of the information which is the object of the communication and is transparent to the Communication System and the RH (Request/Response Header) which contains indications on the modalities of use and status of the Logical Channel. It is worth pointing out that a) RU is

limited in size and b) RU data integrity and sequentiality are maintained. The combination of the Interface Functions layer and the Common Network constitutes the Communication System.

Thus, the characteristics of a Logical Channel can be summarized as follows:

- it is driven with a half-duplex technique
- it does not provide an error-free connection
- error (loss of RU) is detected and signalled at both Logical Channel sides

Applications Layer

This most external layer contains the ultimate sources and destinations of information. The term Application is used here to indicate the generic network end user. More precisely, an Application is defined as any process or set of coordinated processes that access the Communication System in order to obtain network services. Towards the external boundary, Applications can be directly or indirectly attached to one or more end user physical devices which represent the ultimate sources or destinations of information. These devices, if present, are also included in the conceptual frame of the term Application. For example, a user terminal and the code supporting I/O operations on the processor to which the terminal is connected are considered to be a single process of an Application. By this definition, any information exchange across the Communication System must take place between Applications.

The addition of the Applications layer to the Communication System makes RPCNET a closed system.

3. Network Control, Data Flow and RNAM

Two software components, already defined, provide the control in each Node: the Network Services Manager (NSM), within the Interface Functions, and the Common Network Manager (CNM) within the Communication Functions. Each shares control of the respective layer in an equihierarchical way with the corresponding components in all other Nodes. Both are addressable as separate control points (Franchi /2/).

This separation of control gives rise to a logical Node configuration that allows for a variety of physical Node configurations.

The logical Node configuration is defined by stating that, in a Node, there is one and only one Common Network control point (CNM) and one, several, or even no Network Services Manager (NSM).

By distributing the software layers defined by the architecture on one or two processors, the physical Node configurations of Fig.3 are possible.

Fig.4 shows an example of the network with different Node configurations, together with its functional representation, from which it can be seen that the definition of Network Connection applies more precisely to communication channels connecting any two processors both of which contain a CNM component, i.e. two Network Nodes. Communication channels connecting physically distinct components within the same Node are called Internal Connections when they join different processors in the same Node, or telecommunication lines (or simply Lines) when they are used to attach devices to some processor.

Considering the topological layout of network hardware components, the Host role implies a processor having a single connection (Internal Connection) to an FEP. Data coming from many Applications possibly associated with device Lines are multiplexed/demultiplexed on this Connection.

A further level of multiplexing/demultiplexing is performed on data traffic of more Hosts to/from the single component that within the Communication Function layer has the capability of routing information. The presence of this component characterizes a processor as a Node.

The routing, the Host multiplexing/demultiplexing and the Application multiplexing/demultiplexing stage correspond to the three level address scheme of RPCNET.

A zero value of the second and third address fields (fig.5) identifies the addressable unit associated with the Common Network Manager (CNM). A zero value of the third address field identifies an NSM. A non-zero address is used to identify a network addressable unit associated with an Application. This unit is called LCT (Logical Channel Termination).

Before starting any network activity, an Application has

to ask the Communication System for an LU (Logical Unit), using the RNAM macro instruction OPENLU. This allows the Application to contact the NSM component and through its control point to send and receive messages and/or inquiries (MESSAGE and INQUIRE macros).

When an Application wants to connect to another, it issues a BIND operation. An Application wishing to be connected by a bind request, must issue an INVITE operation.

If the Application requested by the BIND exists and has issued an INVITE, the two Applications become addressable without the intervention of the respective NSMs.

The SH (Session Handler) component carries out the in-Session information exchange task by executing the SEND, RECEIVE and BREAK macros issued by the Applications.

The RPCNET logical channel allows only one operation to be specified at a time. SEND is used to send a message, and the message will be buffered at the receiving side (given sufficient storage) until a matching RECEIVE is done by the application there. Due to the nature of the half duplex logical channel, it is necessary to "change direction" before a receiver can send and a sender can receive. The sender is in charge of changing direction. A BREAK can send a message (restricted in length to fit within a common network packet) from an application which is in the receive state to one that is in the send state. The break message is received in an asynchronous fashion, and no special operation from the receiver is necessary.

The TESTLC operation is used by an application to control certain aspects of the logical channel error detection mechanism. In effect it verifies whether the messages sent have arrived at the other end of the logical channel. When a message is sent, this can be done in three possible ways or "modes". The first is by the no response mode, the second is by the definite response mode (definite acknowledgement of succesful receipt or of loss), and the third is by the exception response mode. A SEND sent by definite response does not end until an acknowledgement is received. The other two modes allow the SEND to end as soon as the message leaves the network node. In the case of exception response, an asynchronous call to the sending application is made (or it is scheduled as a task) upon detection of message loss.

The UNBIND and CLOSELU macros are used respectively to release a Logical Channel and a Logical Unit at the end of a Session. CLOSELU is not mandatory if mail and inquiry services are still requested.

4. Experimental Implementation

The network configuration on which the RPCNET software prototype is operating, as of October '77, is shown in Fig.6. The central processors available at the Node locations are IBM System/370s, and IBM System/7 as Front End Processor (or FEP). The operating system software on System/370s is VM/370 in some Nodes and OS-VS (SVS/HASP4) in the others.

Connections between these processors are leased telephone lines (Data Links). The System/370 Channel Attachment feature (Local Attachment) can be used between a System/7 and a System/370 when these are located back to back. This second type of connection allows a faster, parallel by byte, data transfer between processors.

The software providing the 'Communication System services and the RNAM access facilities is called CNS/VM under VM/370 (Fusi /3/) and CNS/VS under OS/VS (Gori /4/); CNS stands for Computer Network Subsystem.

The System/7 network software is an independent stand-alone system called NCS/7 (Network Control System for System/7). It is written in System/7 assembler language and is compatible with MSP/7, the IBM standard support. NCS/7 has been provided with its own Task, Storage, I/O and Command Management (Lazzeri /5/).

Each processor in the configuration of Fig.6 plays the role of Full Node (see Fig.3a). The Communication System software and RNAM are operational under CNS/VM, CNS/VS and NCS/7. This means that Applications running under the control of OS-VS, VM or NCS/7 can communicate with each other through logical channels by using RNAM.

5. RPCNET Services

The following facilities have been included with priority in the REEL project objectives.

- Exchange of spool files between OS-VS and VM in any combination.

- Interactive Terminal access.

- Remote Device Access

5.1 Spool File Exchange

The spool to spool RPCNET facility (Bertaina /6/) under VM is provided as an extension of the SPOOL and TAG console functions of the VM/370 Control Program. By using this extension, the VM/370 user at his virtual machine console can consider any destination of spool files as a virtual machine running under the processor to which he is attached (see Appendix).

Under CMS, the VM/370 Conversational Monitor System, spool information is not limited to being logically associated to input card decks or output print lines, but a CMS user can put on spool areas any information available on his private direct access space.
CMS users have at their disposal a special command, SEND, which is expanded in the equivalent SPOOL and TAG console functions.
VM users accessing local or remote card input devices can address any destination processor in the network by putting a TAG card in front of their card decks.

Under OS-VS operating systems, the HASP4 facilities were extended to allow the exchange of spool files, not only with other OS-VS systems but also with VM/370 Nodes.
A /*SEND card put in front of input card decks provides facilities analogous to those provided by a TAG card under VM, while a /*ROUTE card allows the routing of job outputs anywhere in the network.
In particular, jobs to be executed under OS-VS can be acquired both from the network and from the local, internal or RJE workstation card readers. In the same way, spool files to be printed or punched or to be sent otherwhere through the network can be acquired without OS-VS execution or scheduling.

These control cards have the following formats:

```
/*SEND node host ntuserid nn ty acct password

/*ROUTE  outty NET node host ntuserid nn ty
```

where:

'node' : destination network node;
'host' : destination network host;
'ntuserid' : -for print or punch destination, output file identifier;
-for input destination to a VM host, addressed virtual machine name;
-ignored in other cases;
'nn' : two-digit identifier of the addressed RJE workstation of the OS-VS destination host (ignored in other cases);
'ty' : may be PR, PU or RD, to specify print, punch or input destination (this specification cannot be chosen independently of 'outty' specificat.)
'acct' : output accounting code (for card deck local listing or duplication);
'password' : 1 to 8 byte password (for card deck local listing or duplication);
'outty' : may be PRINT or PUNCH for job output print image or card image data.

The end-to-end protocol adopted provides translation from VM to OS-VS spool format, and viceversa, for card image or print image data.

5.2 Interactive Terminal Access

For any particular interconnection of a terminal to an interactive operating system through RPCNET, there will be a Terminal application at the terminal side and a Host Application at the interactive system side. In RPCNET, two approaches were considered for the implementation. The first approach does not involve any modification of the Host system code, while the second normally requires some modification. In the first approach, one IBM System/7 Host Application drives a System/7 emulator of an IBM Start/Stop 370X/EP communication control unit. Thus the System/7 appears to the interactive computer to be a 370X/EP with start-stop terminals (Lazzeri /8/). The operating system can support terminals via the network if the operating system itself has software-to-access terminals via the 370X/EP. By contrast, in the second approach there are Host and Terminal Applications which reside in the VM/370 control program, and which are essentially additions to the VM/370 terminal access method.

Fig.7 illustrates the logical separation of the various Application components and their interconnection via RPCNET. The Terminal Application is made up of a line driver that maps a real terminal into a Virtual Terminal (or VT), a network command processor, and a module for communicating with the Host Application via the RPCNET logical channel. The protocol by which this communication is carried on is called Virtual Terminal Protocol, or VTP. For the System/7, only a line driver for the IBM 2741 interactive terminal has

been implemented so far. The VM/370 Terminal Application could use the access methods already provided for the terminals supported by VM/370.

The Host Application maps the Virtual Terminal Protocol into actions meaningful to the computer. In the case of the System/7, the Application maps the Virtual Terminal Protocol into 370X/EP emulator actions. In the case of VM/370, the mapping is into a virtual console for the virtual machines, at a level where both the VM control program and the user virtual machine access the virtual console. Thus, codes have been added to the terminal access method of VM/370 so that the RPCNET Virtual Terminal is inserted into the system.

Terminals attached to the VM/370 system as virtual machine consoles, can connect to any other VM in the network by issuing the command REELON, followed by the specification of the destination system. After this command (see Appendix), users can LOGON into any virtual machine of the destination system. A similar capability is under test for IBM 2741 terminals attached to System/7. By issuing the command REELON users can reach any VM/370, either directly or via a System/7 Front End, or any OS-VS via a System/7 Front End.

5.3 Remote devices access

The CMS user has at his disposal a set of commands by which he can attach to his virtual machine one or more minidisks defined in the directory of one or more virtual machines which reside in remote hosts.
When a remote device is attached, it can be accessed and used just like a local one, i.e., a list of the files in that,disk may be obtained, files may be read and edited, renamed and erased, simply by issuing standard CMS commands. The approach allows different users to read and write in the same device simultaneously, without destroying or modifying the master file directory of the disk.
A user may access a disk also in exclusive mode: in this case, other users cannot access the same device.
If a user wants to reserve some files, and not all the disk, he can do this simply by issuing a command from his virtual machine. A file reserved may be accessed by other users read/only, read/write, or cannot be accessed, depending on the mode of the reservation.

The attach command (NETATT) has the following format:

```
NETATT    < AS>ccu1<DASD>ccu2<OF>vmid1
          < THROUGH>vmid2(HANDLER)<AT>node      host      W(R)
          S(E)<PASS>passwd
```

where:

ccu1 — is the virtual address of the remote device for the local CMS.

ccu2	is the virtual address of the remote device for the remote CMS.
vmid1	is the name of the remote virtual machine which has the device to link.
vmid2	is the name of the virtual machine which handles the connection This virtual machine resides in the remote host and works disconnected. If not specified, HANDLER is assumed.
Node Host	are, respectively, node and host names
w(r)	is the access mode of the device (write or read): if not specified, w is assumed.
s (e)	is the access mode allowed to the other users. If e (exclusive) is specified, nobodyelse may link and access that device until it is detached; if not specified s (shared) is assumed.
Passwd	is the password of the device that must be linked. This parameter must always be specified.

After NETATT is issued, a user can access the device just like a CMS minidisk:

ACCESS ccu1 mode

where ccu1 is the same parameter issued in the command NETATT.

If a user wants to reserve a file (or also reserve a filename and filetype), he can issue the command NETRES, which has the following format:

NETRES filename filetype options

where 'options' are two characters: the first may be R or W, and indicates whether you want to access the file in read/only (R) or read/write (W). The second character indicates the access mode allowed to other users: i.e. R if other users can only read the file, but not write in it, W if others can write, and E if nobody else can use that file.

When the reservation of a file is no longer necessary, the user can issue the command NETREL, which has the following format:

NETREL filename filetype filemode

Finally, when a user wants to detach the remote device, he can release it by a CMS command, and then issue the command NETDET, otherwise issue NETDET directly: the effect is the same.

The format of the command NETDET is:

NETDET ccu1

where ccu1 has the meaning above described.

6. Conclusion

RPCNET has been implemented almost on schedule according to the initial design (Lenzini /1/). The objectives and requirements formulated by the project partners have largely been met.

The Communication System, RNAM and the following services: Interactive Terminal Access, File Transfer, Spool-to-Spool and Remote Access devices are all operational. At present, there are basically two ways of applying RPCNET. One is the use of the present prototype network, either in its current configuration, or with changes. The other is the development of independent replicas of RPCNET for internal use of the project partner's organizations.

For CNUCE, the main objective in terms of network usage is to turn the RPCNET prototype implementation into a network of CNR (National Research Council) Research Institutes, for the benefit of the National Research Council and possibly for the benefit of other scientific organizations.

As a matter of fact, the following centers:

ICITE - CNR Laboratory
Future center of CNR in Rome
Istituto Superiore della Sanita'

along with CNUCE will be the nucleus for such a network. RPCNET should therefore facilitate researches, providing the cheapest and effective computing services available on the network. It is mainly a service bureau type of operation for unsophisticated users who want to use RPCNET as a tool for their researches.

In order to solve problems such as documentation, RPCNET software maintenance, consulting services and standardization of the user procedures and in view of foreseen problems, a group of CNUCE System Programmers has been set up. This group expects to carry out the RPCNET services available within CNR by the end of this year.

Before concluding it is worthwhile pointing out that the design and implementation of RPCNET has allowed CNUCE to give valuable contributions within national and international committees treating technical matters in the area of computer networking. In particular a person from CNUCE is a member of the ETAG (Economic and Technical Aspects Group) group of EURONET and coordinates the interfacing of the Italian Host computers to EURONET.

7. Acknowledgements

RPCNET has been a team project and a large number of people have contributed to the ideas presented in this paper. Special acknowledgement is made to the Pisa IBM Scientific Center people for their outstanding contributions to the RPCNET project.

Appendix

To the user connected to a VM/370 Node via any terminal supported by the present VM/370 release, RPCNET provides the following facilities:

A -LOGON to any virtual machine in other Nodes, by simply issuing a special network command:

REELON <nodeid>

before the standard VM/370 LOGON.
For example from a terminal connected to Pisa via a point-to-point line, the user can logon a virtual machine in Turin (Torino). The messages exchanged at the terminal are the following:

```
vm/370 online
  -message sent by the VM/370 in Pisa
<attn>
  -attention to unlock the keyboard
reelon torino
RPCNET - VM/370 ONLINE TORI
  -message sent by the VM/370 system in Torino
<attn>
  -attention to unlock the keyboard
logon <vmid>
  vmid must be a virtual machine
  defined in the VM/370 directory in Turin
```

B -When logged on, a virtual machine user can send spool data files to any other system.

When the virtual machine is running CMS, the user has at his disposal a new command, SEND, which has the following format:

SEND <nodeid> <mvid> <mode> <fileid> [<variable text>

This command has the effect of transferring to the Node specified in "nodeid" the file specified in "fileid". The mode in which the file will be handled by the destination Node spool system is described by the "mvid" and "mode" fields.

Example:

SEND TORI MV2 RD ALFA BETA

the CMS file ALFA BETA is sent to the virtual reader (RD) of the MV2 virtual machine in Torino.

Similar facilities are available, either to non-CMS virtual machines or for special applications to CMS users, by using the VM/370 commands SPOOL and TAG, followed by an output operation on the virtual unit specified in both the commands.

Example:

```
SPOOL E TO RETE
TAG DEV E VENE VENE MYSELF PRT
```

After these commands have been issued, all files sent to the virtual printer on address E will be printed on the real printer of the system in Venice. MYSELF will be the output identifier together with RPCNET.

C -The CMS programmer has at his disposal a macro library and an access method to write application programs which exchange data with other application programs, via the network.
By using these communication primitives, a new CMS command, NETDEF, has been defined modifying the standard FILEDEF command in order to use the network without changing the already existing application programs. An example is given of the use of the NETDEF command where a FORTRAN program reads data residing in another Node and writes the results in a third Node. Suppose that the CMS user is in Pisa. He can issue the following commands:

```
NETDEF 01 INPUT APPL1 TORI
NETDEF 02 OUTPUT APPL2 VENE
```

and then:

```
LOAD FORTPROG
START
```

FORTPROG is a program that reads from the logical unit 1 and writes on the logical unit 2. APPL1 and APPL2 are network applications residing in Turin and Venice respectively.

D -The RPCNET user can take advantage of the local network facilities, such as remote job entry from batch terminals and remote job transfer to HASP and JES systems, as provided by RSCS (under VM/370).
RPCNET extends the services offered by RSCS, by allowing a routing of input and output data throughout the network. For example a user in Turin can send a job to Pisa to be run on an OS-VS/SVS system attached via RSCS to the VM/370 system in Pisa. The output may be returned either to the real printer in Turin, or to the reader of a virtual machine.

E -In every VM/370 Node, the RPCNET software runs as a special operating system (called CNS-VM) on a virtual machine. By logging on this virtual machine a terminal can become the network operator console. The network operator has at his disposal the following facilities:

1. Start and stop network connections, force down network ports and sessions, Node shutdown.
2. The sending of messages to any other network operator.

3. The inquiring about the status and activity of the Node.

References

/1/ L. Lenzini, G. Sommi, (1976), 'Architecture and Implementation of RPCNET', Proceedings of the 3rd ICCC Conference, Toronto, August 1976, p.605

/2/ P. Franchi, (1976), 'Distribution of Functions and Control in RPCNET', Proceedings of the 3rd Annual Symposium on Computer Architecture, Clearwater Fla., January 1976, p.130

/3/ A. Fusi, (1976), 'REEL Project: CNS/VM, The Virtual Machine Environment Computer Network Subsystem', Proceedings IEEE Computer Networks: Trends and Applications, Gaithersburg, November 1976, p.128

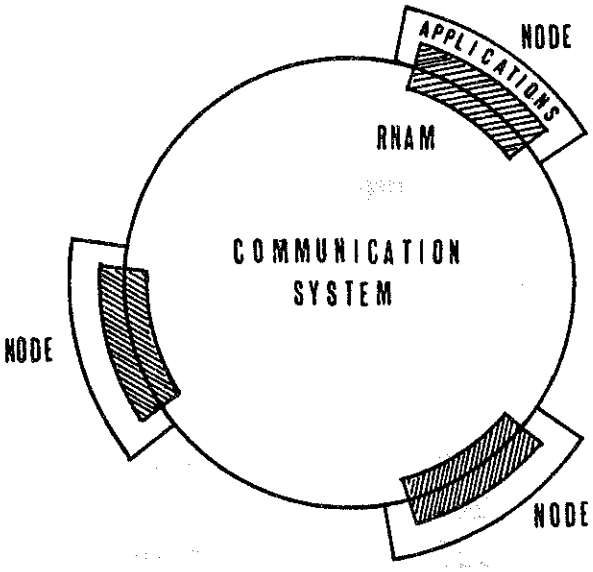
/4/ G. Gori, M. Maier, 'Design and Implementation of Software for a Distributed Control Computer Network', Proceedings of the International Symposium on Technology for Selective Dissemination of Information, Repubblica di S.Marino, September 1976, p.89

/5/ L. Lazzeri, L. Lenzini, A. Springer, (1976), 'NCS7, The Implementation of RPCNET on a Minicomputer', Submitted for publication

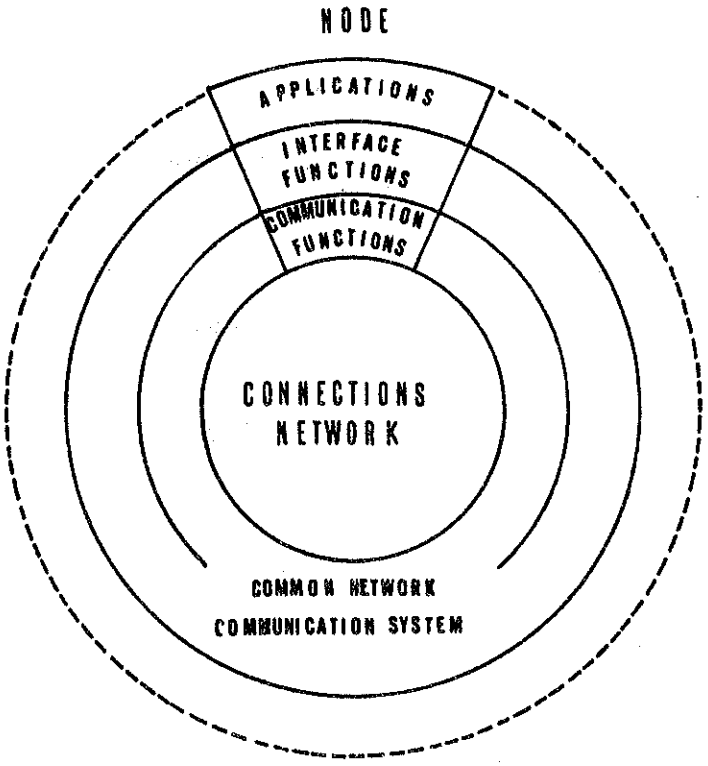
/6/ P. Bertaina, M. Magini, C. Paoli, F. Tarini, (1974), 'Spool To Spool Protocol: Initial Design', RPCNET Internal Document IS007-01, October 1974

/7/ L. Lazzeri, L. Lenzini, A. Springer, (1976),' Terminal Access to Host Computer through RPCNET' , Proceedings of the ICS77/ACM Conference, Liege, April 1977

/8/ L. Lazzeri, L. Lenzini, (1976), 'EPS7: 2703 Emulation Program for the System/7', Proceedings of the SEAS (SHARE European Association) Anniversary Meeting, West Berlin, September 1976


APPLICATIONS
NODE
RNAM
COMMUNICATION
SYSTEM
NODE
NODE

b)


NODE
APPLICATIONS
INTERFACE
FUNCTIONS
COMMUNICATION
FUNCTIONS
CONNECTIONS
NETWORK
COMMON NETWORK
COMMUNICATION SYSTEM

a)

Fig.1

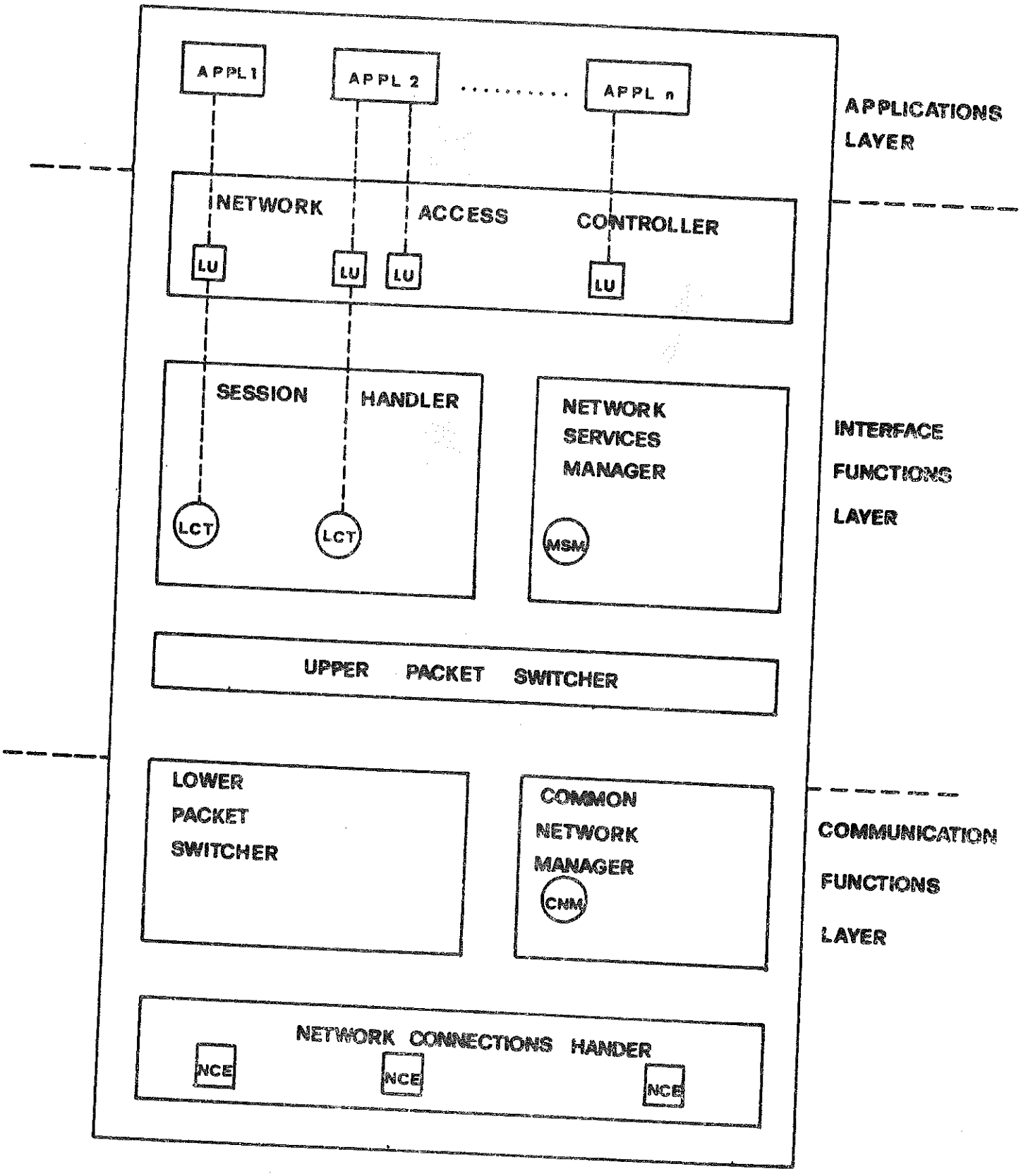
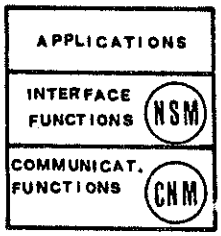

APPL 1
APPL 2
APPL n
APPLICATIONS LAYER
NETWORK ACCESS CONTROLLER
LU
LU
LU
LU
SESSION HANDLER
LCT
LCT
NETWORK SERVICES MANAGER
MSM
INTERFACE FUNCTIONS LAYER
UPPER PACKET SWITCHER
LOWER PACKET SWITCHER
COMMON NETWORK MANAGER
CNM
COMMUNICATION FUNCTIONS LAYER
NETWORK CONNECTIONS HANDER
NCE
NCE
NCE

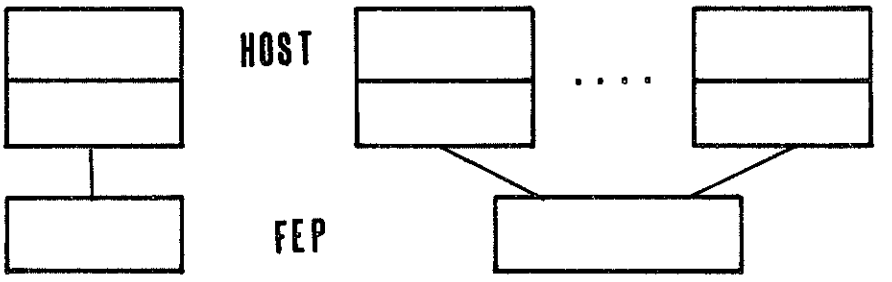
Fig. 2


APPLICATIONS
INTERFACE FUNCTIONS
NSM
COMMUNICAT. FUNCTIONS
CNM

a)


COMMUNICAT. FUNCTIONS
CNM

b)


HOST
FEP

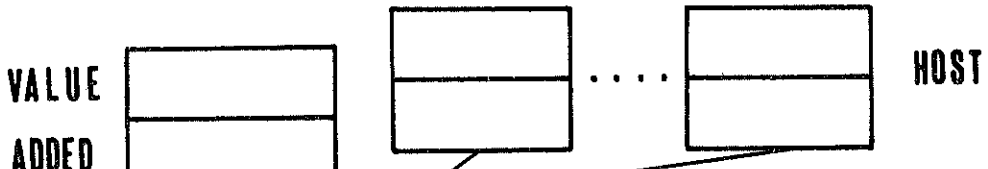

VALUE ADDED FEP
HOST

Fig.3

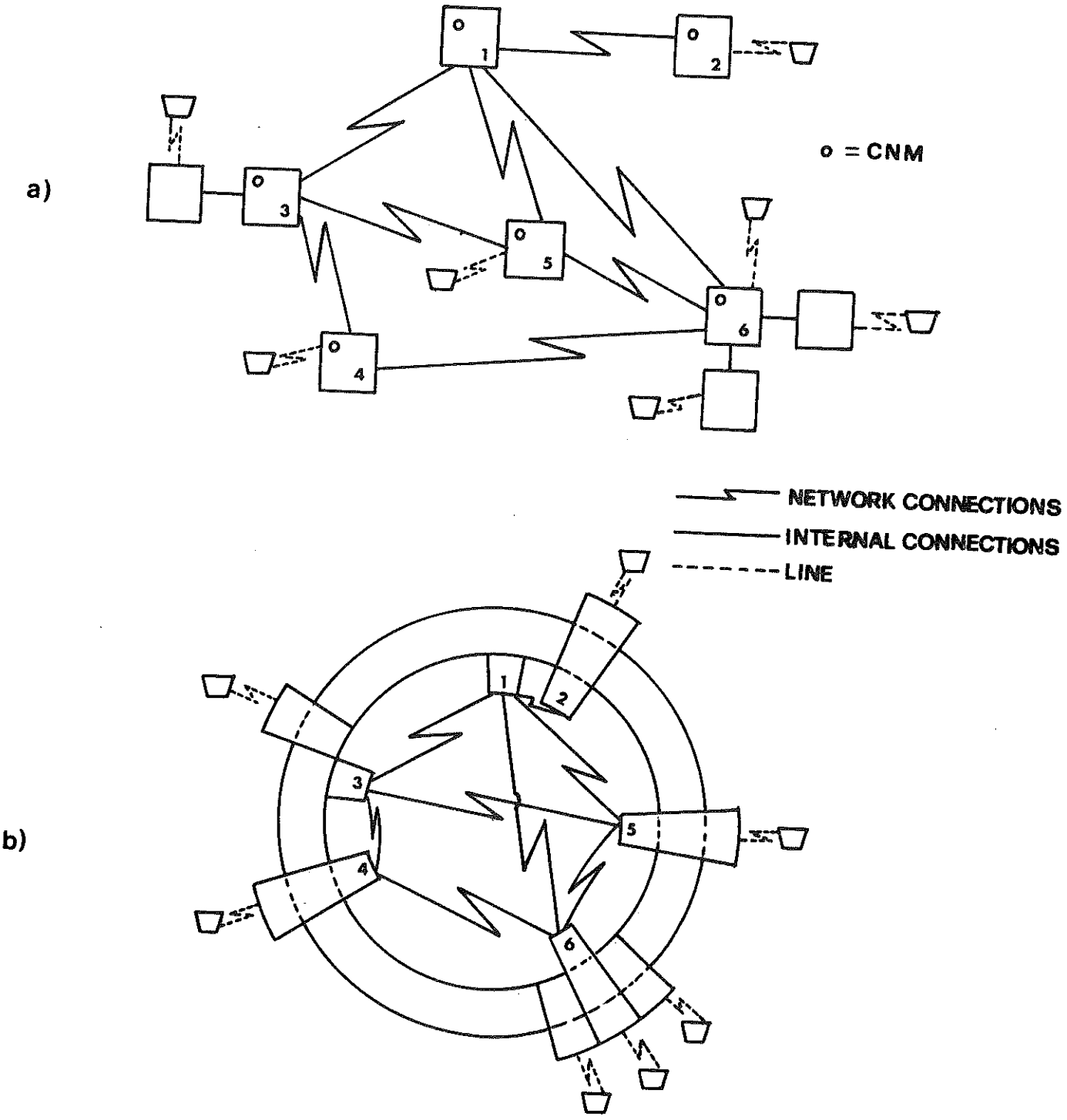


Fig. 4

Layout (a) and Functional Diagram (b) of a 6-Node, 9-Processors Network

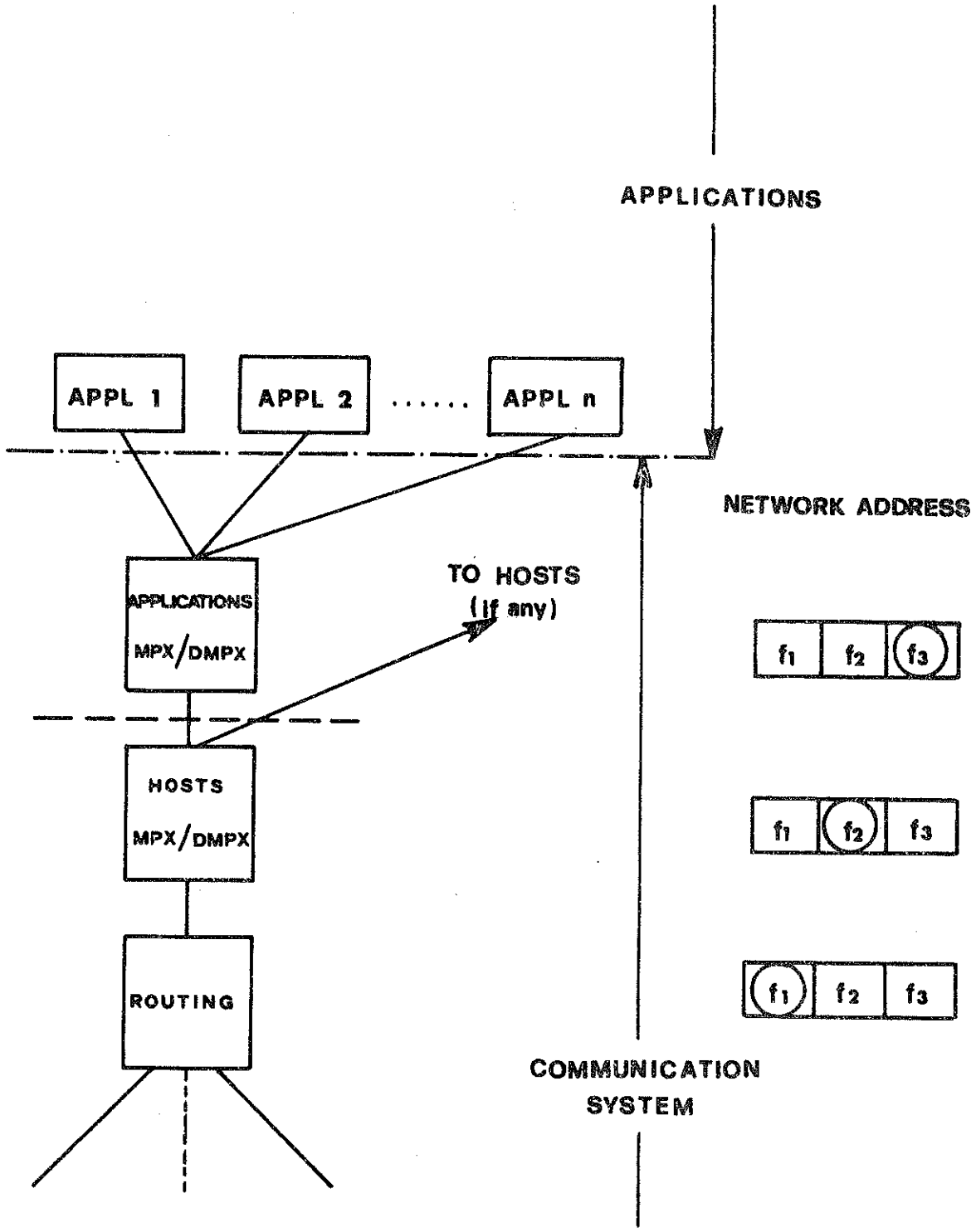


Fig. 5

Data flow vs Addressing Scheme in DPCNET

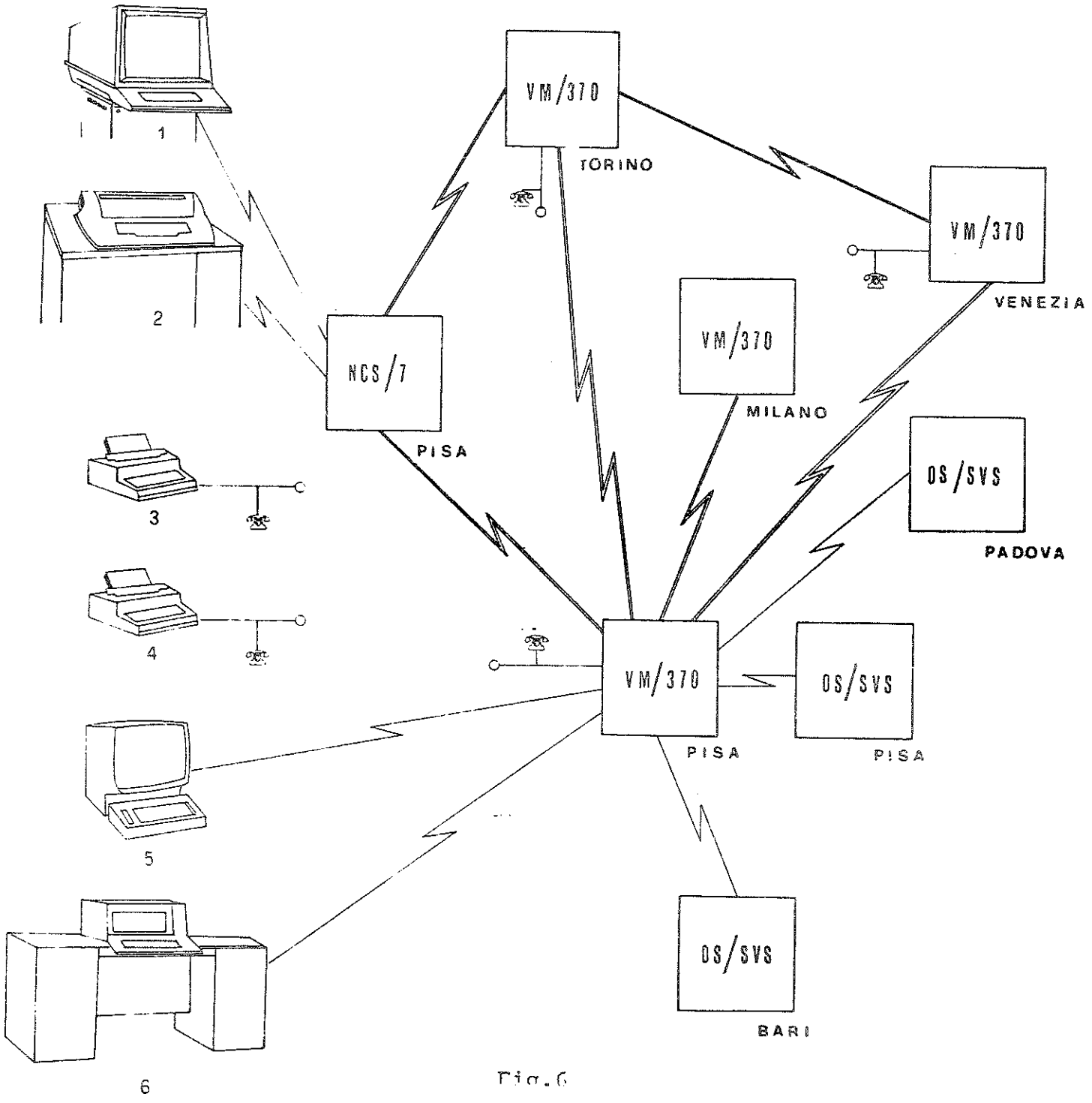

1
2
3
4
5
6
NCS/7
PISA
VM/370
TORINO
VM/370
VENEZIA
VM/370
MILANO
OS/SVS
PADOVA
VM/370
PISA
OS/SVS
PISA
OS/SVS
BARI

Fig.6

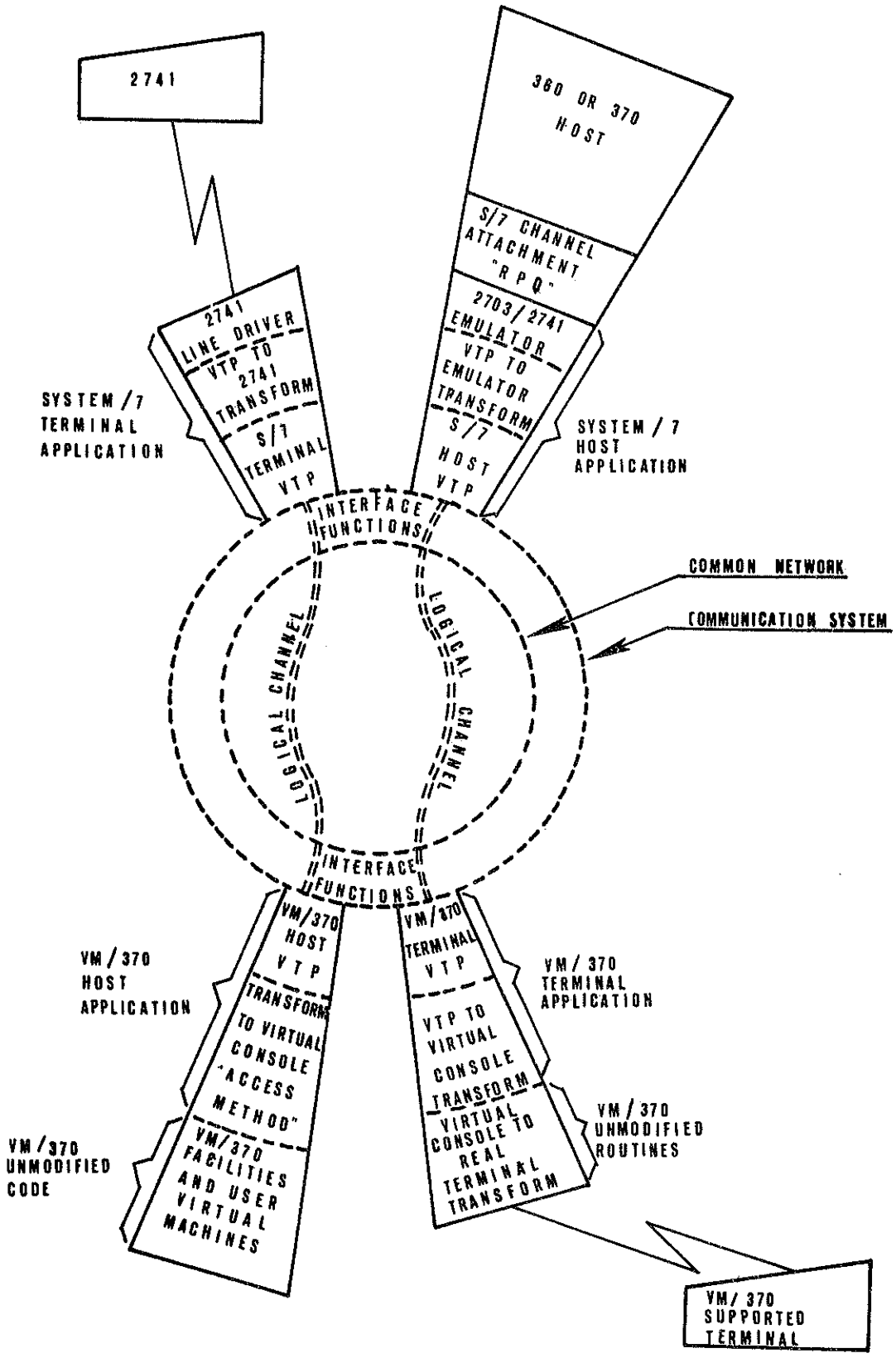

2741
360 OR 370 HOST
S/7 CHANNEL ATTACHMENT "RPQ"
2741 LINE DRIVER VTP TO 2741 TRANSFORM
2703/2741 EMULATOR VTP TO EMULATOR TRANSFORM
S/7 TERMINAL VTP
S/7 HOST VTP
SYSTEM/7 TERMINAL APPLICATION
SYSTEM/7 HOST APPLICATION
INTERFACE FUNCTIONS
LOGICAL CHANNEL
LOGICAL CHANNEL
COMMON NETWORK
COMMUNICATION SYSTEM
INTERFACE FUNCTIONS
VM/370 HOST VTP
TRANSFORM TO VIRTUAL CONSOLE "ACCESS METHOD"
VM/370 FACILITIES AND USER VIRTUAL MACHINES
VM/370 TERMINAL VTP
VTP TO VIRTUAL CONSOLE TRANSFORM
VIRTUAL CONSOLE TO REAL TERMINAL TRANSFORM
VM/370 HOST APPLICATION
VM/370 UNMODIFIED CODE
VM/370 TERMINAL APPLICATION
VM/370 UNMODIFIED ROUTINES
VM/370 SUPPORTED TERMINAL

Fig.7