# REMOTE FILE ACCESS IN RPCNET

F.Caneschi - L.Lenzini - M.Sommani

Nota Interna C79-7

A cura di: Fausto Caneschi - Luciano Lenzini - Marco Sommani
( CNUCE Istituto del C.N.R. )

# REMOTE FILE ACCESS IN RPCNET

F. Caneschi L. Lenzini M. Sommani

CNUCE - Institute of C.N.R.
(National Research Council)
Pisa, Italy.

A description is given of a Remote File Access (RFA) system implemented on computers connected via the RPCNET network and running under the VM/370 operating system. The RFA system enables CMS users to access data stored on CMS minidisks anywhere in the network in the same way as for data stored locally. The system network facilities are described briefly and some considerations are also made about the possibility of extending the RFA system to other operating systems.

## INTRODUCTION.

RPCNET, a computer network connecting Italian research and educational institutions, became operational in October, 1976 ([1], [2], [3], [4]). At that time, it was decided to implement a Remote File Access (RFA) system for CMS virtual machines ([8], [9]), with the main purpose of testing the RPCNET facilities. This RFA system was designed to enable CMS users to access data stored on CMS minidisks anywere in the network in the same way as for data stored locally. The RFA system was also designed in order to eliminate one of the main faults of CMS, namely, the impossibility for two CMS virtual machines to have read-write access to the same minidisk without spoiling its contents.

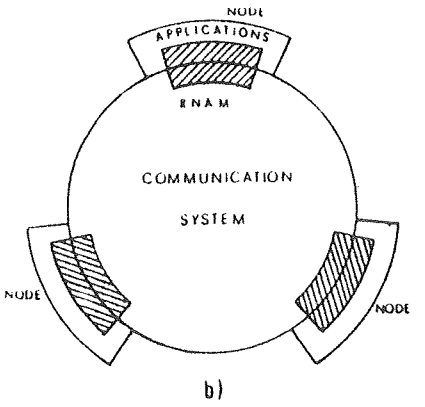
The RFA system was thus designed and implemented, and became operational by October, 1977. It was quickly evident that the RFA system indeed resolved successfully the needs of users having to simultaneously update data stored in a single location using programs running on different computers. Therefore, it was decided to investigate the possibility of implementing the RFA system on operating systems other than CMS. Since the RFA system had been designed in strict reference to CMS concepts, it was necessary to re-define the system in a more general sense.

Section 3 of the present paper contains the description of the CMS implementation, operational since October, 1977. This description is given in terms of the general RFA system design concepts, which are presented informally in section 2. A formal description of the general RFA system design concepts can be found in [7]. Section 1 contains the description of some RPCNET basic features, necessary in order to understand the rest of the paper. Further information about RPCNET can be found in [1], [2], [3] and [4]. Section 4 contains a detailed description of the new CMS commands introduced by the present implementation.

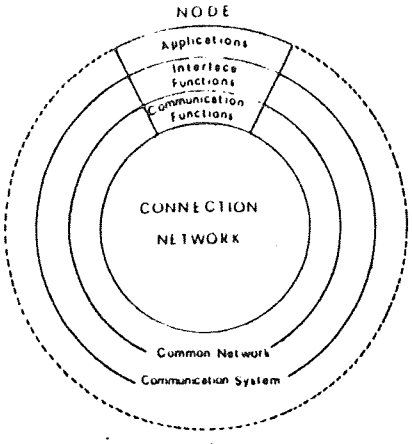
## 1. RPCNET ARCHITECTURE.

The RPCNET architecture has been designed to provide a machine independent method for communications between Applications via logical channels.

The architecture of RPCNET is structured in three well defined layers (Figure 1): the innermost "Common Network" layer, which is based on the "packet store and forward" technique, the next outer "Interface Functions" layer, which supports the "logical channels", and the outermost layer of network "Application" programs, which use the logical channels to communicate with each other. The Common Network provides the packet switching capability by using a physical communication medium ("Connections Network") together with packet transmission on a full-duplex basis, and routing and reconfiguration facilities ("Communication Functions Layer"). 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




b)




a)

Figure 1

or duplication of packets. These characteristics of the Common Network are caused by such factors as the use of alternative routes for packets between nodes in cases of failure, and retransmission on data links when errors are detected. The Interface Functions build through the Common Network a "logical channel" facility which provides error detection of lost packets, and the disassembly and reassembly of messages longer than packet length. The combination of the Interface Functions layer and the Common Network constitutes the Communication System. A set of directives called RNAM (REEL Network Access Method) allows the Applications to use the network facilities ([4], [5]).

The RPCNET "logical channel" allows error detection when messages are lost. A check is also made for breakages in the logical channels. The most convenient error detection scheme provided by RPCNET, from the Applications viewpoint, is the ability of the sender to detect that a sent message has been lost. It is normally the sender who can most easily recover when messages are lost.

The RPCNET logical channel allows only one operation to be specified at a time. The SEND, RECEIVE, BREAK and TESTLC RNAM directives are relevant to the RFA system. SEND is used to send a message which will be buffered (given sufficient storage) at the receiving side until a matching RECEIVE is made 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 receive state to one that is in send state. The break message is received in an asynchronous fashion, and no special operation of 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 requests that RPCNET verify whether sent messages have arrived at the other end of the logical channel.

When any message is sent, there are three possible error detection options. The first is "no error detection", the second is "definite response", and the third is "exception response". A SEND directive issued with "definite response" does not complete until the related message has reached the destination network Node and the corresponding acknowledgement is received. The other two response forms allow the SEND to complete as soon as the message leaves the local network Node. In the case of "exception response", the sending Application (the one which issued the SEND directive) is notified asynchronously upon detection of message loss. It is clear that a SEND directive can be issued only when the previous directive has been completed.

Other functions of interest here are those used for setting up and closing down logical channels between Applications. When an Application wants to connect to another, it performs a BIND operation. An Application wishing to be connected by a "bind request" will

perform an INVITE operation. At any time, either Application can perform an UNBIND to release the logical channel.

## 2. RFA SYSTEM ARCHITECTURE.

The principal requirements that characterize the RFA system design, are the following:

- user programs must be able to access remote files in the same way as for local ones, without having to be rewritten or recompiled;
- in order to avoid conflicts between programs sharing the same data, the physical access to every resource must be controlled at each network Node by a single process..

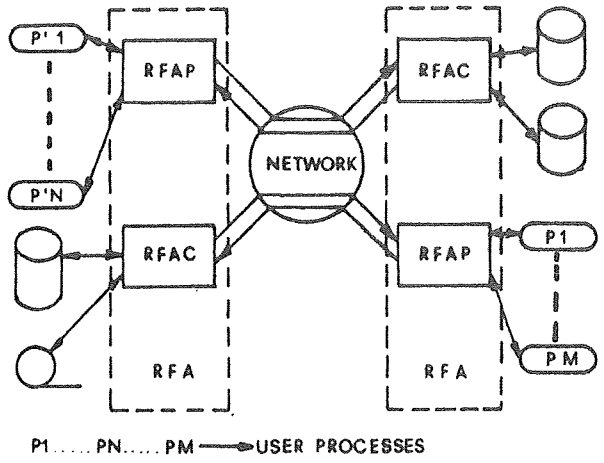



Figure 2

In accordance with the requirements outlined above, the RFA system was structured into two components (Figure 2):

- the Program Interface (RFAP), that traps all access requests made by the user processes, and resides on the user's Host;
- the Access Controller (RFAC), that receives the access requests from remote RFAPs and performs the necessary I/O operations for accessing physical resources.

The RFAP must then convert all requests coming from user processes into requests which are significant for the RFAC. RFAP must also map all remote resources into concepts recognized by the operating system where it resides.

According to the RPCNET terminology, the RFAP with the associated user processes at one side and the RFAC with the corresponding physical resources at the other end are two Applications which exchange information through logical channels.

Early in the RFA system design stage, special care was paid to resolving the problem of how many logical channels should be set up between each RFAC and RFAP pair. Since every request originated by an RFAP and then directed to an RFAC requires the following information transfers:

- request transfer (from the RFAP to the RFAC);
- data transfer (needed only for data transfer requests);
- answer transfer (from the RFAC to the RFAP);

it was obviously concluded that the information exchanged between RFAC and RFAP must flow in both ways. If all requests issued by user processes were handled by the corresponding RFAP in a strictly sequential way, the "change direction" facility could be used and a single logical channel between RFAP and RFAC would suffice. This situation corresponds to the VM/370 implementation, which is described in section 3. However, in most cases an RFAP will have to interface and then to multiplex requests coming from more than one process; any process in turn can issue requests in an asynchronous way. Consequently, the sender side can never know whether or not the change direction indication must be sent. It is thus necessary to set up at least two simplex logical channels between each RFAC and RFAP pair, each one being in charge of transferring information in one direction only.

It should be noted that in the discussion above it has been assumed that when the RFAP issues a data transfer request, the user process has already provided all buffers necessary for the operations; otherwise, it is advisable to set up an additional logical channel for the data transfer; this solution allows also a transfer of data between two RFAC's governed by an RFAP located in a third node ([7], [13]). Although this facili-

ty is not present in our implementation, the RFA system architecture can easily accommodate it.

As was specified in section 1, SEND operations can be issued both in "no response" and in "definite response" mode. Of course, if the "no response" mode is chosen, the RFA system must provide its own data flow control protocol. In an implementation where a single logical channel is set up between RFAC and RFAP and the "change direction" facility is used, SEND operations can be issued in "definite response" mode, without a notable diminishing in performance.

The information flowing through the logical channels that couple the RFAPs and the RFACs can be classified into five categories:

- resource reservation requests;
- file system manipulation requests (such as erasing and renaming files);
- data transfer requests;
- data transfers;
- answer transfers.

All requests in the first three categories are concluded by an answer transfer which specifies whether the request has been executed correctly by the RFAC. In an implementation where the "change direction" facility is used, the answer transfer has also the function of returning the "send state" to the RFAP, thus enabling it to issue a new request.

Data transfers are caused by data transfer requests. All implementations must define the type of data which can be requested (volumes, files, records, items, catalogs, index files etc.); if no additional logical channel is set up, data transfers must take place in the time between the data transfer request and the corresponding answer transfer. Transfers of large quantities of data, such as volumes and files, are likely to be requested by programs that do not provide all the necessary buffers at the moment of the request; these transfers should therefore be allowed only by those implementations where an additional logical channel can be set up for data transfers. In the present VM/370 implementation, only sets of records and minidisk directories can be transferred; in both cases, the necessary user buffers are always ready when the request is issued. Since RPCNET already provides a service for the interchange of spool files among different network nodes ([3], [6], [4]), the RFA service has been designed mainly to allow programs running in different network nodes to share the same data bases; therefore, priority has been given to the transfer of small amounts of data.

File system manipulation requests can be easily implemented only if both RFAP and RFAC run under the same operating system. In the present implementation, the operating system is always CMS, therefore all CMS file system manipulation functions (namely, the ERASE and RENAME functions) have been implemented. In a more general situation, the way in which RFAP maps the remote file system will determine which requests will be allowed.

Resource reservation requests are needed in order to enable the RFAC to avoid access conflicts between different user processes. The RFAC must, therefore, be able to distinguish one process from another. Because of the particular nature of the VM/370 implementation, different user processes (i.e. different CMS virtual machines) always communicate with the RFAC through different logical channels. Thus, it is not necessary to include particular information in the request in order to identify the issuing process; in other implementations, however, other solutions will have to be found ([7]). The implementation design must also state which resources can be reserved: in the present implementation, this is only possible for minidisks and CMS file identifiers. When the RFAC can satisfy a reservation request immediately, a positive answer is returned to the RFAP; otherwise, there are two different answer possibilities:

- a negative answer is sent immediately;
- a positive answer is sent when the resource becomes available.

Since, in an implementation using the "change direction" facility, the second solution would make the logical channel unavailable for new requests until the resource becomes available, at the present, only the first answer method has been implemented. The implementation of the second answer method will most probably be the first extension of the RFA system, although it will require basic architecture modifications, e.g., changing the properties of the logical channels.

## 3. IMPLEMENTATION.

As most of the RPCNET Hosts are IBM 370 computers running under the VM/370 operating system ([8]), it was natural that the first RFA system implementation should be developed under the CMS operating system ([9], [10]). Some modifications in the RFA system architecture which has been described here were required by the VM/370 virtual machine environment. Since a VM/370 virtual machine is a software simulation of the IBM 370 hardware, each CMS user has his own set of virtual peripherals and his own file system stored on "minidisk(s)". Thus, each virtual machine can be considered as a different computer and the RFAP and the RFAC can be implemented on all the virtual machines. However, since CMS is designed for a single user and is not multiprogrammable, it may be undesirable to have both RFAC and RFAP running on the same virtual machine. In fact, a solution of this type, although feasible, may cause problems to the CMS user, who may dislike having a public service such as RFAC running on his private virtual machine. Therefore, in the VM/CMS implementation, RFAP is run on all the user CMS virtual machines, while RFAC is only run on particular CMS virtual machines.

The standard CMS implementation allows a user to enlarge or restrict the file system available to his own operating system by means of the ACCESS and RELEASE standard commands. These commands, respectively, add and remove the file directory of a minidisk to and from the active directory of the operating system. These features were thus extended to allow a CMS user to also use the ACCESS command when including remote files in his active directory. This means that a set of remote files is presented to CMS as a minidisk. Moreover, a virtual minidisk address is assigned to the set of remote files, since the ACCESS command uses it as a parameter. In order to create and destroy this association, two new CMS commands, NETATT and NETDET, have been implemented. The NETATT command must be used in order to associate a minidisk address to a set of files known to the RFAC. Since, in the present implementation, the RFAC is also a CMS Virtual Machine, the set of files must also be a minidisk for the RFAC.

The present implementation is characterized by the following peculiarities:

- each RFAP interfaces a single user process;
- all user requests are synchronous;
- data transfer requests are only issued when the necessary user buffers have been provided.

As already noted in section 2, in this type of environment a single logical channel between each RFAP and RFAC pair using the "change direction" facility is sufficient.

In the CMS implementation, however, one logical channel is set up for each NETATTached minidisk. In other words, the BIND and UNBIND operations are triggered by the NETATT and NETDET commands respectively. Thus, if a CMS user NETATTaches twice the same remote minidisk by specifying two different addresses, two logical channels are set up. The RFAC will therefore handle all reservation requests for resources located on the minidisk as coming from two different processes. This is, in fact, the correct behaviour for the RFAC because, from the user's CMS virtual machine point of view, the two addresses correspond to two different minidisks.

The RFAC maintains reservation tables for two kinds of resources: minidisks and CMS file identifiers. Minidisks must be reserved explicitly when the NETATT command is issued, while file identifiers can be reserved implicitly or explicitly, as will be explained below.

In the present implementation, the minidisk reservation mode is a two letter keyword, where the first letter (R for Read or W for Write) determines whether the minidisk is requested in read-only or read-write mode and the second (S for Shared or E for Exclusive) determines whether other users may have access to the minidisk or not. When a minidisk is requested in exclusive mode, no file identifier reservation table is maintained by the RFAC. When a minidisk is released, all the file identifier reservations issued for it are released too.

File identifiers can be reserved explicitly by means of the new CMS command NETRES or implicitly when the first read or write command is issued. A file identifier reserved explicitly must be released by means of the NETREL command; file identifiers reserved implicitly are released when the file is closed. The file identifier reservation mode is a

two letter keyword, where the first letter (R for Read or W for Write) determines whether the file is requested in read-only or read-write mode and the second (W for Write, R for Read or E for Exclusive) determines the ways in which other users are allowed to use the file (reading and writing, only reading, or not at all). Implicit reservation modes are RW if the first operation is a READ, and WR if it is a WRITE.

In the present implementation, eight requests can be issued by user programs:

NETRES Explained above and in the next section.

NETREL Explained above and in the next section.

GETDIR A new copy of the NETATTached minidisk directory is obtained. This request is sent every time the user issues the ACCESS command for the minidisk.

READ One or more records are read from a file. The parameters sent to the RFAC are the file identifier, the record number, the number of records that must be read and the input buffer length. This request causes the implicit reservation of the file identifier in RW mode if it had not already been reserved.

WRITE One or more records are written on a file. The parameters sent to the RFAC are the file identifier, the record number, the number of records that must be written and the output buffer length. This request causes the implicit reservation of the file identifier in WR mode if it had not already been reserved.

CLOSE Releases a file reserved implicitly.

ERASE Deletes a file from a minidisk. The request is not executed if another user has already reserved the file identifier.

RENAME Alters the name of a file. The request is not executed if another user has reserved the old and/or the new file identifier.

The internal protocol used in the present implementation is quite simple, because requests transferred to the RFAC consist simply in the parameter lists needed by the corresponding CMS directives. Future implementations will probably use the more general protocols, described in [7]. CMS directives appearing in the protocol are READFST, RD UF, WRBUF, FINIS, ERASE and RENAME ([11]).

## 4. NEW CMS COMMANDS.

This section contains the detailed description of the four new CMS commands introduced by the RFA system. Information about the CMS operating system can be found in [9] and [10], although this section is intended mainly for people already familiar with CMS.

### 4.1. NETATT.

Remote CMS minidisks can be added to the virtual machine configuration by means of the NETATT command.

```
+----------+-----------------------------+
|          |                             |
| NETATT   | [AS] ccu1 [DASD] ccu2       |
|          | [OF] vmid [[AT] destid]     |
|          | [[THROUGH] vmid2]           |
|          | [([rsvmode] [[PASS]         |
|          | password] [)]]              |
|          |                             |
+----------+-----------------------------+
```

where:

ccu1 is the minidisk virtual address, as it will be known by the local CMS;

ccu2 is the minidisk virtual address as it is defined in the directory of vmid;

vmid is the name of the remote virtual machine that has the minidisk in question defined in its configuration;

destid is the standard RPCNET destination identifier ([3]), composed of a node name and a host name separated by a blank. Destid identifies the computer where vmid is defined. If there is no specification, the local computer is assumed;

vmid2 is the name of the virtual machine, defined at destid, on which RFAC is run. If there is

no specification, the name HANDLER is assumed;

rsvmode is the two letter reservation mode, explained in section 3. If there is no specification, the WS mode is assumed;

password is the link password of the minidisk ([12]). If there is no specifiation, it is assumed that the minidisk has no password.

Usage Notes

1. The minidisk defined by the NETATT command can be used as a normal CMS minidisk by all CMS commands except FORMAT, DDR, DISK and TAPE ([10]).

2. If the remote minidisk address is specified in a CP command, the minidisk is not found, since the NETATT definitions are known to CMS only.

3. CP commands such as SYSTEM RESET or IPL ([12]) are equivalent to a NETDET command.

4.2. NETDET.

Minidisks defined and reserved by means of the NETATT command can be released by means of the NETDET command.

| NETDET | ccu |
|---|---|

where:

ccu is the address that was defined by means of the NETATT command.

Usage Notes

1. If, when the NETDET command is issued, the remote minidisk is ACCESSed with some mode-letter, the corresponding CMS RELEASE function is also performed.

4.3. NETRES.

File identifiers can be reserved by means of the NETRES command.

| NETRES | fileid rsvmode |
|---|---|

where:

fileid is the normal CMS file identifier, consisting of a filename, a filetype and a filemode separated by blanks ([9]).

rsvmode is the two letter reservation mode, explained in section 3.

Usage Notes

1. NETRES is a CMS module generated in the transient area ([11]); as such it can be invoked from within a program, as many other CMS commands.

2. Two consecutive NETRES from the same user on the same file identifier are not allowed (see NETREL).

3. The ERASE command is rejected if the file identifier is already reserved by another user. The RENAME command is rejected if the old and/or the new file identifier is reserved by another user.

4.4. NETREL.

File identifiers, reserved by means of the NETRES command, can be released by means of the NETREL command.

| NETREL | fileid |
|---|---|

where:

fileid is the normal CMS file identifier, consisting of a filename, a filetype and a filemode separated by blanks ([9]).

Usage Notes

1. NETREL is a CMS module generated in the transient area ([11]); as such it can be invoked from within a program, as many other CMS commands.

REFERENCES

[1] F. Caneschi, E. Ferro, L. Lenzini, M. Martelli, C. Menchi, M. Sommani, F. Tarini, "Architecture of and the Service Facilities Provided by RPCNET - The Italian Computer Network for Education and Research Institutions", Proceedings of the ICCC 4th International Conference for Computer Communication, (September 1978).

[2] F. Caneschi, E. Ferro, L. Lazzeri, L. Lenzini, M. Martelli, C. Menchi, M. Sommani, F. Tarini, G. Torrigiani, "RPCNET: Architektura i Servis", Avtomatika i Vycislitel'naja Technika, 6 (1978) 44-55.

[3] F. Caneschi, E. Ferro, L. Lenzini, M. Martelli, C. Menchi, M. Sommani, F. Tarini, "The RPCNET Applications User Manual", Technical Report CSN-031, IIASA, (1978).

[4] F. Caneschi, "A Computer Network: Structure and Protocols of the RPCNET", Professional Paper PP.78-12, IIASA, (December 1978).

[5] A. Fusi, "RNAM - Macro Instructions for Application Programs", RPCNET Internal Document US012-00, (July 1975).

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

[7] F. Caneschi, M. Sommani, "Remote File Access: a Data Communication Protocol for a Computer Network", Research Report, IIASA, (to be published).

[8] "IBM Virtual Machine Facility /370: Introduction", File No. S370-20, Order No. GC20-1800.

[9] "IBM Virtual Machine Facility /370: CMS User's Guide", File No. S370-39, Order No. GC20-1819.

[10] "IBM Virtual Machine Facility /370: CMS Command and Macro Reference", File No. S370-36, Order No. GC20-1818.

[11] "IBM Virtual Machine Facility /370: System Logic and Problem Determination Guide Volume 2", File No. S370-37, Order No. SY20-0887.

[12] "IBM Virtual Machine Facility /370: CP Command Reference for General Users", File No. S370-36, Order No. GC20-1820.

[13] P. Schicker, A. Duenki, W. Baechi, "Bulk Transfer Function (Proposal)", Forschungsproject COST-11, European Informatics Network, EIN/ZHR/75/20, (September 1975).

STAMPATO PRESSO IL
SERVIZIO TECNOGRAFICO
DEL CNUCE