REMOTE FILE ACCESS: A DATA-ACCESS PROTOCOL FOR COMPUTER NETWORKS

F. Caneschi and M. Sommani
Institute of the National Research Council (CNUCE)
Pisa, Italy

RR-80-43
December 1980

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
Laxenburg, Austria

Research Reports, which record research conducted at IIASA, are independently reviewed before publication. However, the views and opinions they express are not necessarily those of the Institute or the National Member Organizations that support it.

FOREWORD

The idea of connecting two or more computers in order to make them work together is as old as the concept of digital computing itself. Recent years have proved that computer networks are suitable instruments for achieving this goal.

Computer interconnection is assuming clear structure owing to the significant contributions toward standardizing communication procedures that have been made by a number of national and international organizations and institutions. Although lower-level procedures are already well established, higher-level ones are still being developed.

The Informatics Task at the International Institute for Applied Systems Analysis (IIASA) has made several studies of standard protocols and their performance. This study concentrates on a protocol for a higher-level family, the file transfer protocol. While a restricted implementation was carried out in connection with the work on RPCNET, the Italian computer network for education and research institutions, the protocol as defined in this paper has a more general meaning. In fact, the assumptions the authors make about the nature of the network software are quite general and can be applied to a number of existing networks.

The authors started this work at the Institute of the National Research Council (CNUCE), Pisa, Italy and completed it at IIASA. This study was submitted for publication in September 1979; its issue was delayed for technical reasons.

PREFACE

A computer network may be seen as a set of components, some being hardware (physical lines, modems, and computers) and some being software (programs to perform the network functions). This report explores part of the software structure of a computer network.

Although seven protocol layers are recognized as independent entities in computer network architecture, in this report we consider, for the sake of simplicity, a three-layer architecture. The first layer drives the physical lines and sends data (divided into "packets") onto the lines toward their destinations. It also "routes" packets coming from the lines to their destinations and keeps the tables that describe the network topology up to date. When dealing with this layer, we speak of a "first-level protocol."

The second layer is the interface between the host computer and the network. It provides such services as maintaining connections, signaling errors, assembling packets into users' data units, and disassembling users' data units into packets for the network. The protocol related to this layer is commonly called the "end-to-end protocol" because the host computers are considered to be end users of the network.

The highest level of network architecture is the application level; any program that uses the network by means of the end-to-end services is considered to be an application. Application programs exchange information and perform services for users of the host computer. For instance, an application program can allow a user to "connect" his terminal (which is physically connected to a computer) to any computer in the network or can transfer a data file from one computer to another.

Although with appropriate restrictions any user can use the network, writing his application program to "talk" to another program following a private protocol, such services as terminal access or file transfer should be considered as "system" services and should not be managed by ordinary users.

This means that standard high-level protocols must be developed for such services.

This paper proposes a new system service, remote file access (RFA), which permits a user to have access to data files stored physically in remote computers without transferring either all the data or the associated program from one computer to another.

The logical-record-access philosophy of an RFA system suggests using it as a basis for distributed data-base systems, for both enquiry and update applications. From the networking point of view, an RFA system can be considered an application.

Unfortunately, no standards have been developed for the end-to-end protocol. We were thus obliged to make general assumptions, which led to a description of the interaction between the application protocol (that is, the remote file access protocol) and the end-to-end protocol that was not formally exact. Because most existing computer networks have similar interfaces between the applications and end-to-end layers, however, a future standard will not differ substantially from our assumptions.

We studied and developed the RFA protocol with a flexible structure in order to allow different types of implementation, such as the application made during the RPCNET project. This restricted use must not be viewed as the complete protocol, for it provided only a subset of the RFA facilities and differed significantly from the final version. For these reasons we did not compare the performance of the only present working example of an RFA system with other types of file transfer protocols. The RFA system provides services not offered by other, similar protocols.

F. CANESCHI
M. SOMMANI

CONTENTS

1 INTRODUCTION

Several studies have been made since the necessity for establishing standard protocols for computer networks (Hovey 1976; Folts and Cotton 1977) was recognized. An international agreement regarding the line protocol was formulated, and a half-duplex logical channel (HDLC) was developed and, in some cases, implemented.

The problem of a standard end-to-end protocol is still unresolved. X25 (CCITT Study Group VII 1976, 1977) is not an end-to-end protocol, although studies to build up a standard end-to-end protocol on an X25 basis (Csaba 1976) have been carried out.

To develop standardized high-level functions, we need to make some assumptions about the nature of the network making up the "sublayer" of a high-level protocol. Naturally, the greater the accuracy of these assumptions, the fewer changes need to be made during the implementation phase. The assumptions we make on the nature of the network or of the services provided by the end-to-end protocol are similar to those described in INWG (International Network Working Group) Protocol Note 86 (High-Level Protocol Group 1977) and are summarized in Section 3.

2 HIGH-LEVEL PROTOCOLS AND THE FILE TRANSFER PROTOCOL

A computer network, in its universal meaning, is a tool by which a number of programs running on different computers (hosts) can exchange information with each other. According to this definition, any user program, under certain conditions and with certain limitations, should be able to "access the network" in order to "talk" with another program in another host, following their own private protocol.

We can, however, separate from the set of all possible network applications some "system services" that should be managed not by user programs

but by the operating systems directly or by specialized subsystems. Such services obviously should use standard protocols – thus the need for standard high-level protocols.

Two network services were recently found to contain standard protocols: the terminal access with its related virtual terminal protocol (VTP) and the file transfer with its file transfer protocol (FTP). Both have been the subject of involved studies (High-Level Protocol Group 1977; Schulze and Börger 1978; Bauwens and Magnee 1978; Shicker and Duenki 1976; Shicker *et al.* 1975; and Gien 1978).

As our study is related to some aspects of the FTP, the next section focuses on the file transfer.

2.1 The File Transfer

The file transfer should provide a tool by which a user can transfer large amounts of data from one host to another in a network; the common denominator of all the studies on the FTP is the definition of the file transfer functions.

Put simply, the FTP problem is a consumer–producer problem. On the one side, a program asks, for instance, for a certain amount of data (normally ordered and called a file) from another program that runs in another host. On the other side, the "awakened" program starts sending data until the requested amount is reached. Obviously, the reverse operation is also possible: the requesting side may send data to the other side, which, in turn, has to provide the space (for instance, disk) for the data to be stored.

In FTP philosophy, the user initiates an event by means of a command which may be issued in a network job control language (NJCL) environment; from that point on, he is normally not involved with data transfer or similar issues until a complete file transfer has been performed.

Two aspects of this philosophy should be taken into consideration. The first is the master–slave relationship between the two sides of the connection. Once the first messages have been exchanged, the roles are assigned; they will not change for the rest of the transfer. The second is the "dynamic connection" concept. The connection established between the two application programs when one of them requested the network services will end when the data transfer is completed.

2.2 An Alternative Solution: The Remote File Access

From the networking point of view, an operating system is a user. It is possible, then, for a user's program to require access to remotely located data. Such a program (data base management, for example) does not require an entire file, but rather the possibility of accessing at any time any record of a complete file system, without necessitating the transmission of all the data in the system.

Moreover, if the transmission of some data is requested, this should not mean that when a file is "closed" (no longer used), the link established via the network should be closed, too.

Because a connection may remain active for an indefinite period of time, more than one user should be able to access the same file system (and even the same file, with some restrictions, of course) at the same time.

In RFA there are no fixed roles: a producer of data directed to a file system may at the same time be a consumer of data belonging to the same file system, without opening another network connection. A connection of this type is better considered as "static" than as "dynamic." Once the referred file system has been accessed, no data transfer is performed until explicitly requested. The connection path will be closed only on request when the entire file system has to be released.

As the name remote file access suggests, this system is especially designed for connection to nonlocal file systems, without necessitating the transfer of entire files.

3 THE NETWORK ENVIRONMENT

User programs should never be involved in network details. For instance, a user program should be able to refer to a file contained in a remote file system, without any definition other than the normal file identifier. Before executing the program, however, an NJCL command should request the "linkage" of that file system to the job. The link command should be as simple as possible; in some cases, only the host's network address, which holds the requested file system, should be added to the file identifier.

The RFA should be in charge of establishing the requested connection and setting up the proper entries in the system's tables, which describe the file system, to indicate its "remoteness."

As previously noted, we made some assumptions about the nature of the services provided by the network that should be used by the RFA function. We also noted that the assumptions we made are similar to those in INWG Protocol Note 86 (High-Level Protocol Group 1977), namely, the following:

1. The transport service regulates the information flow between network users in such a way as to provide a synchronization and a limitation of the demands on the receiver side(s).
2. If the transport service is not "error free," it should detect errors and signal them to both ends of the connection.
3. Any process, at any time, regardless of its role, should be able to close the connection with the counterpart.
4. User's (RFA's) information is delivered, correct and complete, and ordered.

5. The lack of availability of a remote counterpart should be distinguishable from a network break (error).

We made certain other assumptions primarily to better explain the protocol; it is not, however, affected dramatically by them. These assumptions include the following:

6. The primary service provided by the network to the application programs is the possibility of establishing and maintaining logical channels without taking into consideration the actual network architecture (datagram or virtual call).
7. The following operations are possible in a logical channel:
   a. CONNECT – to contact another application
   b. LISTEN – to wait for a contact by another application
   c. SEND – to send data
   d. RECEIVE – to receive data
   e. CLOSE – to close the connection
8. From the user's point of view, a logical channel may be
   a. Simplex (data flow is in one direction)
   b. Half-duplex (data flow is in one direction until one party, normally the sender, decides to "pass the ball" to the other party)
   c. Duplex (data flow is bidirectional; the sender can receive and the receiver can send)

Moreover, we suppose that in the CONNECT and LISTEN operations the user can specify the maximum amount of data to be transmitted in one operation. The network software has to find an agreement if the amounts are different in the two operations.

4 NETWORK DATA REPRESENTATION

As different computers and their operating systems may use different data representations, we recognized the necessity for a uniform "network" data representation early, when data transfer protocols were first studied and developed.

The RFA function, however, is not normally involved with physical data representations because it normally handles logical, not physical, records. This does not mean that a uniform logical record structure should not be studied; even if fewer differences exist in the logical record structures than in the physical ones (for different computers), some differences do exist. This problem, however, is not dealt with in this paper but will be the subject of further study.

In the following description we suppose that a record is a data entity

whose meaning is fully understood by the two parties that are "talking" using the network.

5 SYSTEM STRUCTURE

Figure 1 shows a proposed system structure similar to one implemented in ARPANET (Thomas 1974). In the figure, the term "network software" means all the lower-level services and protocols, such as line, reconfiguration, and end-to-end, that stay "below" the remote file access function and protocol. The network software could also be implemented in two or more different computers using front-end processors; this is completely transparent to the RFA.

The RFA is split into two parts, the remote file access program interface (RFAP) and the remote file access controller (RFAC).

The RFAP is in charge of establishing and handling the connection, that is, "trapping" all users' requests directed to a remote file system and performing the necessary network operations, such as SEND and RECEIVE.

The RFAC is the remote counterpart of the RFAP. The RFAC thus has to receive the requests coming from the RFAP(s) and to perform the necessary operations on the requested file system.

A characteristic of this architecture is its flexibility. A host may in fact choose between three possibilities:

1. Only the RFAP function is implemented. This option means that a local user may access remote file systems, but no remote user can access the local file system.
2. Only the RFAC function is implemented. This is the antithesis of the first option; remote users can access the local file system or systems, but local users cannot access remote file systems.
3. Both the RFAC and RFAP functions are implemented. This should be the "typical" situation. The center (host) permits both access to local files by remote users and access to remote files by local users.

We decided to split the RFA functions into two parts not only for flexibility but also because of two other characteristics of the RFA system design:

1. User programs must be able to access remote files in the same way that they access local ones, without their having to be rewritten or recompiled.
2. Physical access to every resource must be controlled at each host by a single process to avoid conflicts – between, for instance, programs sharing the same data.

The first characteristic refers directly to the RFAP, while the second concerns the RFAC. In addition, the first means that the RFAP should convert all

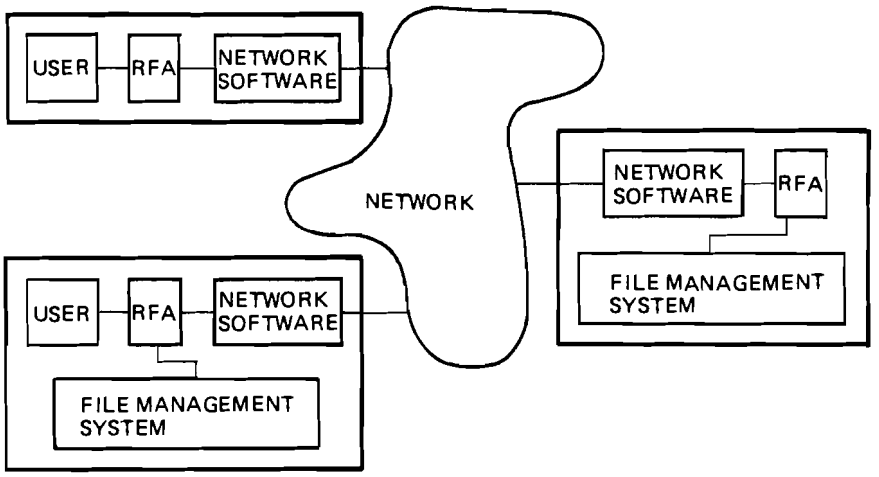

FIGURE 1 RFA system structure.

requests coming from user programs into requests that are meaningful for the RFAC; the RFAP should also perform, if necessary, a mapping between remote resources and concepts known to the local operating system.

5.1 Logical Channels Used by the RFA

Once having decided to split the RFA into two distinct components, we needed to determine how many logical channels we should use to perform the RFAP–RFAC communication. According to the system configuration, every request from the RFAP to the RFAC causes four information transfers:

1. Request transfer (from the RFAP to the RFAC)
2. Permission transfer (from the RFAC to the RFAP)
3. Data transfer (for data transfer requests only, from the RFAP to the RFAC, or vice versa)
4. Answer transfer (from the RFAC to the RFAP)

From these considerations, it follows that the information exchanged between the RFAP and the RFAC must flow in both directions. This situation is summarized in Figure 2. If we use a full-duplex (FD) logical channel, any RFAP can exchange information with an RFAC using one logical channel for both requests and data transfers.

If we use a half-duplex (HD) logical channel, the number of logical channels in use between each RFAP–RFAC pair depends strictly on the nature of the operating system under which the RFAP is implemented. Let us suppose,

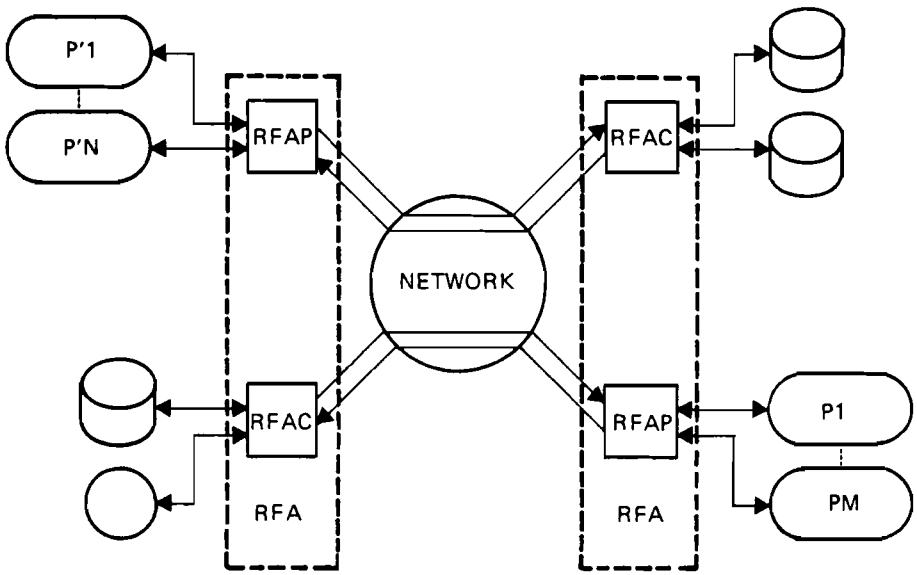

FIGURE 2 Connection between RFA components in different computers, showing the logical channels used. P'1 - - - P'N and P1 - - - PM represent user processes.

for instance, that the sender in the HD logical channel is allowed to reverse the roles of the participants via an indicator that we can term a "change direction" indicator. This is a real situation; see, for example, the RPCNET logical channel (Caneschi *et al.* 1978). In such a situation we have two possibilities.

In the first instance, the RFAP has to handle the requests for processes sequentially; that is, no process multiplexing is allowed or requested. In this case, a single HD logical channel can be used with the "change direction" facility because no confusion can arise as to the roles of the two parts of the connection. This case was implemented and tested in RPCNET (Caneschi *et al.* 1979).

In the second situation, the RFAP has to interface and to multiplex requests coming from a number of processes; every process, in turn, can issue asynchronous requests with respect to the other processes. In this case, with a single HD logical channel environment and the RFAP as the sender side, it is impossible to know when the "change direction" indication has to be issued. It therefore becomes necessary to use two logical channels, each one in simplex (SX) mode, between each RFAP-RFAC pair.

If the available logical channel is simplex it is necessary to set up two logical channels between each RFAP-RFAC pair. This situation is summarized in Figure 3, where columns represent the availability of a certain logical channel type and rows represent the characteristics of the system where the RFAP runs.

If it is desirable to separate the control messages from the actual data

Logical channel type / System	FD	HD	SX
MONO-USER	1 FD	1 HD	2 SX
MULTI-USER	1 FD	2 HD	2 SX

FIGURE 3 Logical channels vs. system configurations.

transfer, twice as many logical channels must always be provided. When, for example, users' programs do not provide all the buffers necessary for data transfer, the logical channel could be congested in such a way that no new request transfer could be performed. Moreover, although it is not the primary aim of an RFA service, setting up a logical channel could provide the transfer of data between two RFACs governed by an RFAP running on a separate host. This facility gives the RFA the full capabilities of a file transfer (see Figure 4), so that an RFA system can be considered as an extension of a file transfer system.

5.2 Resource Reservation in the RFA

As previously stated, in an RFA system more than one user is allowed to gain access to the same file system at the same time. This capacity stems from the "static" nature of an RFA connection and means that there is a high probability of keeping the requested resource busy for a long time. We should thus permit a number of users contemporary access not only to the same file system, but also to the same file.

From the point of view of the host's operating system, there can be an almost infinite number of possibilities for sharing resources among a number of users, including using passwords for reading and for writing and sharing a limited amount of resources (or all, or none), according to the owner's specifications. From this point of view, the RFA system (RFAC) is the owner of the required resources and can thus manage remote users' utilization of the file or files.

According to the RFA's nature, a remote user knows that a file he is using can be used by others at the same time or later, so a user should have the possibility of reserving files, file systems, or both. Reservation of an entire file system, although possible, should remain valid only for the duration of a session and should depend on implementation. Reservation of a single file, on

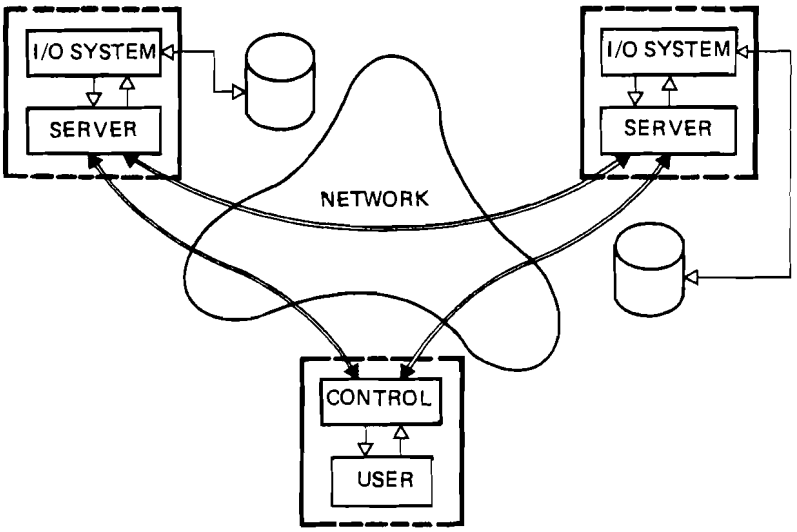

FIGURE 4 File transfer using the RFA.

the other hand, should be possible for a longer period of time and, to avoid conflicts between users, should be performed automatically by the RFAC when an operation is issued on that file. Due to this mechanism, it could be possible for two or more users to write in the same file at the same time (an operation that normally is forbidden in operating systems). This should be permitted only if all such users have issued a request for performing such an operation. This stated, the "natural" serialization of the operations (the RFAC is a single process) will prevent more than one user from writing at the same time on the same file.

The following are recognized reservation requests:

1. Write Exclusive: the user wants an exclusive read/write access and no other user can access the file or file system.
2. Read Exclusive: the user wants an exclusive access but will perform only read operations on the file or file system.
3. Write Shared Write: the user wants an exclusive access to the file or file system and allows other users to gain a write access to it.
4. Write Shared Read: the user wants a write access to the file or file system and forbids other users from writing into it.
5. Read Shared Write: the user wants a read only access to the file or file system and allows other users to write into it.
6. Read Shared Read: the user wants a read only access to the file or file system and forbids other users from writing into it.

The criteria by which the RFAC satisfies reservation requests vary from implementation to implementation. Using passwords is a simple (perhaps too simple) way to implement such a mechanism.

The recognized reservation requests for a single file are the same as those for a file system, with the only difference being one more parameter indicating whether the reservation must be permanent (maintained after the end of the session) or temporary. Reservation requests should normally be changed or deleted only by a request coming from the user who issued the reservation. It should be possible to have a sort of "super user" to prevent undesirable situations.

6 THE RFA PROTOCOL

The RFA protocol is simple: a detailed description of the actions taken by the RFAP and the RFAC can be made using the well-known technique of finite-state machines.

6.1 Finite-State Description

In the following description we assume that *in any state* a network's or partner's failure will cause *only local action.* This means that, in case of error, the tentatives – if any – for recovering do not provoke any special message exchange between the partners. If recovery is feasible, this will imply a re-transmission of the block or a change in the "window" configuration (see Appendix); if, on the other hand, recovery is not feasible, the connection will be declared closed.

Having made this clarification, we may describe the protocol state by state. It is perfectly symmetrical: the oriented graph describing it is the same for both the RFAP and the RFAC.

If in the IDLE state, a request for connection will give control to the CONNECTING state. For the RFAP, a request for connection is originated by the user who asked for the RFA services; for the RFAC, a request for connection comes from the network, originated by an RFAP.

If in the CONNECTING state, a negative answer coming from the network (for the RFAC) or from the counterpart (for the RFAP) will return control to the IDLE state.

If in the CONNECTING state, a positive answer will put control into the CONNECTED state.

If in the CONNECTED state, any legal request for data access will be executed. Control will remain in the CONNECTED state.

If in the CONNECTED state, a request to close the connection will return control to the IDLE state.

The protocol is summarized in the oriented graph of Figure 5.

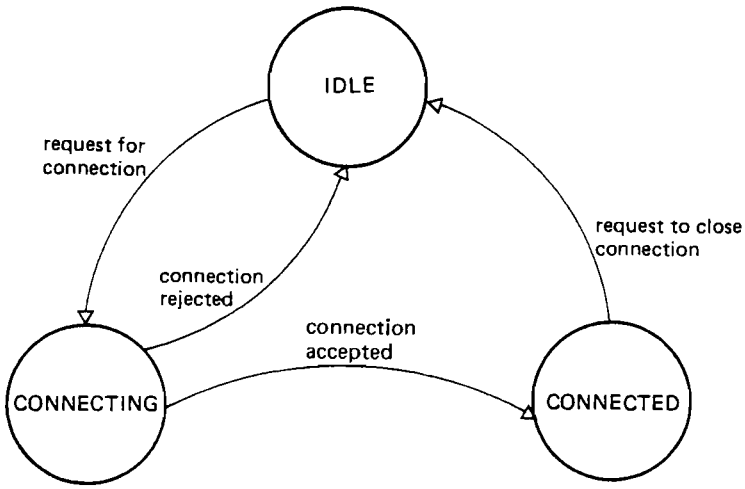

FIGURE 5 RFA connection protocol.

Any legal request issued in the CONNECTED state will consist of the four phases described in Section 5: request for operation; permission for operation; data transfer (if any); and end of operation (with completion code).

When a normal data transfer (rather than "bulk" data transfer; see Appendix) is performed, the user program accessing remote data is better notified of a network failure in the form of an I/O error indication. The program can reactivate the connection when this becomes possible and can decide the point from which to restart the operation. In this case, the RFAC does not need to note the broken connection.

The data transfer may, nevertheless, take a long time, resulting in more likelihood of an interruption in network service during this phase and less convenience in restarting this phase from the beginning. Although a long data transfer is in contrast to the normal RFA philosophy, the RFA system is, as we noted previously, able to provide a file transfer service. In the case of "bulk" data transfer, a second logical channel is established between the parties; it will be closed when the transfer has been performed. The rules governing this "service" logical channel are explained in the following section.

6.2 The RFA Data Channel Protocol

The service logical channel operation is opened by a request to send a file from one file system to another. The data transfer, if permitted, is performed using a dynamic window technique, as explained in detail in the Appendix. When using this technique, it is not necessary to establish restart markers to resume the transfer after a network fault; in fact, the situation of the current window is

saved at the time of the interruption and will be used as a restart indicator when transmission resumes.

As at least one network operation is active in any state, there is no possibility of deadlocks due to network or counterpart failure. In fact, in both cases, as noted in Section 3, the operation will be terminated by the local network software with an error indication, thus avoiding the necessity for the RFAP (or the RFAC) to wait for an indefinite period of time.

APPENDIX

The RFA Protocol: Detailed Description

The commands (operations) exchanged between the RFAP and the RFAC are listed and explained in detail in the sections that follow.

CONNECTION REQUEST

A CONNECT network operation is performed by the RFAP; the message associated with the CONNECT request will contain an indicator, called NORMAL/BULK; a number; and data.

The NORMAL/BULK indicator will have the NORMAL value in the case of an RFA "normal" connection or the BULK value if the operation is a data transfer on a "service" logical channel. The number (from 0 to 255) is associated by the RFAP with the user who requested the RFA services in the case of a bulk data transfer. The third field (data) is used for additional information, as will be explained subsequently.

The format of the CONNECT message is described in Figure 6, where

IND (1 byte) represents the indicator:
X'00' means NORMAL
X'10' means BULK

NAM (1 byte) is the number of the user of the RFAP, which will be used in the CONNECTED phase during a bulk data transfer by the two parties, as will be explained subsequently. If IND = NORMAL, this field has no meaning.

LEN (2 bytes) is the length of the next field (expressed in bytes).

DATA (variable length):
Byte 1 is the modifier. If IND = NORMAL, this byte indicates the type of reservation requested for the file system. If IND = BULK, this byte can be X'00' (new file transmission) or X'10' (old file retransmission).
Bytes 2 and following are supplementary information. If IND = NORMAL, these bytes contain a complete reference path (bytes 2-52). If IND = BULK and modifier = X'00', this field contains a

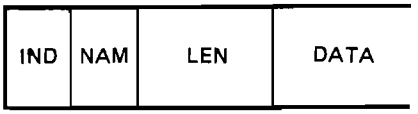

FIGURE 6 CONNECT message (RFAP side).

network address. If IND = BULK and modifier = X'10', the first two bytes contain a block number, the third byte contains the window, and from the fourth byte on, a network address is specified.

On the other side, the addressed RFAC has to keep at least one LISTEN network operation active at any time. This means that as soon as one LISTEN operation is closed by a corresponding CONNECT operation, the RFAC will ask the network for another LISTEN operation, in order to "keep an ear free" for another request.

When a CONNECT operation has been successfully closed, the RFAP will start sending to the connected RFAC messages containing the requests described in the following section.

SESSION MESSAGES (NORMAL SESSION)

Every message from the RFAP to the RFAC will have the general format of Figure 7, where

USN (1 byte) is the number of the user, assigned by the RFAP to associate the request with the user (see IDENTIFY operation).
REQ (1 byte) is the code pertaining to the request.
LEN (2 bytes) is the length of the next field.
MOD (variable length) is the additional information used by the RFAC to perform the operation specified in the REQ field.

On the other side, every message from the RFAC to the RFAP will have the general format described in Figure 8, where

USN (1 byte) is the number of the user.
OP (1 byte) is the operation code.
IND (1 byte) is the return code for the requested operation.
LEN (2 bytes) is the length of the next field.
DATA (variable length) is the result of the operation (if any).

Every message will be sent by the RFAP or the RFAC using a network SEND operation and will be received using the corresponding RECEIVE operation.

The data length is specified in the first part (header) of any message; thus if the network allows only a fixed amount of data to be transferred in one

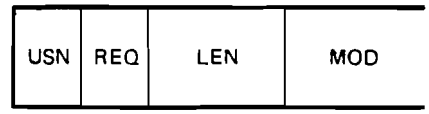

FIGURE 7 Session message (from RFAP to RFAC).

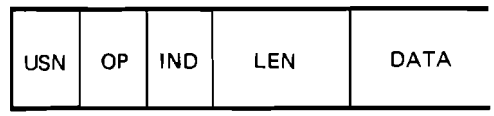

FIGURE 8 Session message (from RFAC to RFAP).

operation, the sender side will issue as many SEND operations as necessary, which will be coupled by a corresponding number of RECEIVE operations on the receiver side. As specified in Section 3, the network is supposed to deliver users' data in the order in which they are presented. In any case, the maximum data length permitted in one operation should be a local parameter; the RFAP has to decide whether to reject the requested operation or to open (if possible) a "service" channel if the data field is considered too long.

Having specified the general message format and rules, let us describe the message formats operation by operation.

IDENTIFY operation. The RFAP identifies a user who will work with the associated RFAC. This operation should be the first for any user and will be issued automatically by the RFAP for any request. Messages:

1. From the RFAP to the RFAC

USN user's number, which will be used in any subsequent operation
REQ IDENTIFY (X'D0')
MOD user's name (in the RFAP's operating system notation), which will be used in case of restart (see RESERVE and RELEASE operations)

2. From the RFAC to the RFAP

USN user's number
OP operation code (X'D0')
IND RFAC's return codes:
- X'00' operation performed
- X'F0' operation not performed; user's number already in use
- X'02' operation not performed; no space for new user

DATA this field has no meaning

CLEARID operation. The RFAP asks the RFAC to clear a user identification number that has been set up with the IDENTIFY operation. The actions taken by the RFAC are the same as those taken in the case of a request to close the

connection, but the logical channel is not closed. Messages:

1. From the RFAP to the RFAC

USN	user's number	
REQ	CLEARID (X'E0')	
MOD	this field has no meaning	

2. From the RFAC to the RFAP

USN	user's number	
OP	operation code (X'E0')	
IND	RFAC's return codes:	
	X'00'	operation performed
	X'F0'	no user with such a number
DATA	this field has no meaning	

OPEN operation. The RFAP asks the RFAC to open the specified file. Messages:

1. From the RFAP to the RFAC

USN	user's number	
REQ	OPEN (X'01')	
MOD	byte 1	action (opening for reading, X'10'; opening for writing, X'01')
	bytes 2 and following	file reference, number of records to be read or written, and so forth, according to the specifications of the operating system where the RFAC resides

2. From the RFAC to the RFAP

USN	user's number	
OP	operation code (X'01')	
IND	RFAC's return codes:	
	X'00'	operation performed
	X'01'	operation not performed; file already opened
	X'02'	operation not performed; file already reserved by another user (see RESERVE operation)
	X'04'	operation failed
	X'08'	reserved for future use
	X'F0'	no user with such a number
DATA	this field is meaningful only if IND = X'04'; in this case, it contains the return code issued by the operating system	

The RFAC reserves the specified file as RW if the operation is READ or as WR if the operation is WRITE. This temporary reservation is not performed if the file has already been reserved by the same user, and it will be canceled when the CLOSE operation is issued (see CLOSE operation).

WRITE operation. The RFAP asks the RFAC to write some data into the specified file. Messages:

1. From the RFAP to the RFAC

USN	user's number	
REQ	WRITE (X'00')	
MOD	bytes 1-16	file reference
	bytes 17-18	record number
	bytes 19 and following	data

2. From the RFAC to the RFAP

USN	user's number	
OP	operation code (X'00')	
IND	return codes:	
	X'00'	operation performed
	X'01'	file not previously opened
	X'02'	operation scheduled
	X'04'	operation failed
	X'F0'	no user with such a number
	others	free
DATA	return codes according to the RFAC's operating system	

READ operation. The RFAP asks the RFAC to read from the specified file. Messages:

1. From the RFAP to the RFAC

USN	user's number	
REQ	READ (X'02')	
MOD	bytes 1-16	file reference
	bytes 17-18	record number
	bytes 19-20	first record to be read

2. From the RFAC to the RFAP

USN	user's number	
OP	operation code (X'02')	
IND	return codes:	
	X'00'	operation performed
	X'01'	file not opened for reading
	X'02'	operation scheduled
	X'04'	operation failed
	X'F0'	no user with such a number
	others	free
DATA	if IND = X'00', this field contains the requested data; otherwise, it contains the error return code issued by the operating system	

CLOSE operation. The RFAP asks the RFAC to close the specified file. Messages:

1. From the RFAP to the RFAC

USN user's number
REQ CLOSE (X'04')
MOD bytes 1–16 file reference

2. From the RFAC to the RFAP
USN user's number
OP operation code (X'04')
IND RFAC's return codes:
X'00' operation performed
X'01' file not yet opened
X'04' operation failed
X'F0' no user with such a number
others free
DATA if IND = X'00', this field has no meaning; otherwise, it contains the error return code issued by the operating system

The RFAC cancels the reservation for the specified file if that reservation was caused by an OPEN operation.

ERASE operation. The RFAP asks the RFAC to erase the specified file. Messages:

1. From the RFAP to the RFAC
USN user's number
REQ ERASE (X'08')
MOD bytes 1–16 file reference

2. From the RFAC to the RFAP
USN user's number
OP operation code (X'08')
IND RFAC's return codes (same as for CLOSE operation, but X'01' has no meaning)
DATA system's return codes (see CLOSE operation)

RESERVE operation. The RFAP asks the RFAC to reserve the specified file. Messages:

1. From the RFAP to the RFAC
USN user's number
REQ RESERVE (X'10')
MOD bytes 1–16 file reference
byte 17 reservation codes (same as for the reservation of an entire file system; see Section 5.2):
WE (X'00') Write Exclusive
RE (X'01') Read Exclusive
WW (X'10') Write Shared Write
WR (X'20') Write Shared Read
RW (X'40') Read Shared Write
RR (X'80') Read Shared Read

byte 18 reservation type:
- TE (X'00') temporary reservation, which will be canceled at the end of the session
- PE (X'10') permanent reservation, which will remain after the end of the session; a permanent reservation cannot be issued for an entire file system

2. From the RFAC to the RFAP

USN user's number
OP operation code (X'10')
IND RFAC's return codes:
- X'00' reservation performed
- X'01' the requested type of reservation is not compatible with a reservation that has already been made by another user
- X'02' file already reserved by this user
- X'F0' no user with such a number

DATA this field has no meaning

RELEASE operation. The RFAP asks the RFAC to cancel a previously issued reservation. Messages:

1. From the RFAP to the RFAC

USN user's number
REQ RELEASE (X'20')
MOD bytes 1–16 file reference

2. From the RFAC to the RFAP

USN user's number
OP operation code (X'20')
IND RFAC's return codes:
- X'00' operation performed
- X'01' file not previously reserved
- X'F0' no user with such a number
- others free

DATA this field has no meaning

SEND operation. The RFAP asks the RFAC to send a specified amount of data (even to another RFAC). Messages:

1. From the RFAP to the RFAC

USN user's number
REQ SEND (X'40')
MOD
- bytes 1-16 file reference
- bytes 17-18 number of the first record to be sent
- bytes 19-20 number of records (if equal to X'FFFF', the whole file must be sent)

byte 21 — user's name that will be used in the secondary connection

bytes 22 and following — network address of the destination

2. From the RFAC to the RFAP

USN — user's number

OP — operation code (X'40')

IND — RFAC's return codes:

- X'00' — send operation completed
- X'01' — file reserved by another user for reading
- X'02' — operation scheduled
- X'04' — file not existing or not readable
- X'08' — connection rejected by the counterpart
- X'F0' — no user with such a number

DATA — this field has no meaning

The name that will be used during the data transfer is the name specified by the "master" RFAP in the command message. The details of the transmission protocol are explained in the following pages. When the transmission has been finished, a message is sent to the "master" RFAP both if the data transfer is a "three-party" connection (i.e., implies a data transmission between two RFACs) and if the RFAP is the actual receiver.

RECEIVE operation. The RFAP asks the RFAC to receive a specified amount of data (even from another RFAC). Messages:

1. From the RFAP to the RFAC

USN — user's number

REQ — RECEIVE (X'80')

MOD

- bytes 1-16 — file reference
- bytes 17-18 — number of records to be received (if equal to X'FFFF', the entire file must be received)
- byte 19 — user's number that will be used by the counterpart in the connection
- byte 20 — second modifier:
  - X'00' — data are overwritten; the old contents of the file are lost
  - X'01' — data are appended to the end of the file
  - X'02' — data are partially overwritten; the next two bytes indicate the first record that must be overwritten
- bytes 21-22 — meaningful only if the previous byte has a value of X'02'

2. From the RFAC to the RFAP

USN — user's number

OP operation code (X'80')
IND RFAC's return codes:
- X'00' transfer completed
- X'01' file system reserved by another user
- X'02' operation scheduled
- X'04' file temporarily reserved by another user
- X'08' file permanently reserved by another user
- X'F0' no user with such a number

DATA user's number used during the transfer (allowing the RFAP to release this name and to use it for another connection)

INFORMATION operation. The RFAP asks the addressed RFAC about a file transfer. The request can be sent both to the sender RFAC and to the receiver RFAC. Messages:

1. From the RFAP to the RFAC

USN user's number
REQ INFORMATION (X'C0')
MOD user's number for the transfer

2. From the RFAC to the RFAP

USN user's number
OP operation code (X'C0')
IND RFAC's return codes:
- X'00' file transmitted
- X'01' no transfer request from this user
- X'02' no transfer request with this number
- X'04' the file is being transmitted
- X'08' the file has been transmitted only partially, for the reason and in the amount specified in the DATA field
- X'10' no connection request by any other RFAC to date
- X'F0' no user with such a number

DATA meaningful only if IND = X'08'
- byte 1 reason codes:
  - X'01' no more space to store data
  - X'02' temporary network interruption; the transmission will resume as soon as possible
- bytes 2–5 number of bytes received

CLOSE CONNECTION REQUEST

When the user wants to "detach" the file system, he has to issue a command, which is translated by the RFAP into a network CLOSE operation or into a CLEARID operation if more than one RFAP's user is working on that connection. The RFAC will receive the CLOSE code as a termination of the currently

opened network operation and will close all currently opened users' files, cancel all previously issued temporary reservations, and close the logical channel.

As previously stated, in the IDENTIFY operation the RFAP sends two users' names to the corresponding RFAC. The first is a number used during the connection by the RFAP and the RFAC. The second is the actual name (in the RFAP's operating system notation) of the user. It can be an account number, a programmer's name, or any set of characters that unequivocally identifies this user to the RFAP's operating system. When a file is permanently reserved, this name is associated with the file name in such a way that if the same user later asks for another connection, his files will remain under his domain; a new connection request with the associated IDENTIFY operation will bring, generally, a new user's number, which is an RFAP parameter.

Data Channel Protocol

The data channel is established on request by means of a SEND or RECEIVE RFA operation when a bulk data transmission is to be started. The protocol ruling the data channel is a typical file transfer protocol. It can be considered as divided into three parts, the connection protocol, data transfer protocol, and restart protocol, which are discussed in the sections that follow.

CONNECTION PROTOCOL

The RFAC or RFAP that received the SEND request issues a network CONNECT operation to the destination RFAC or RFAP, which has been specified in the SEND message. Messages:

1. From the sender to the receiver

IND	BULK (X'10')	
NAM	user's number, as specified by the RFAP	
DATA	byte 1	modifier = X'00'
	bytes 2 and following	network address of the RFAP that requested the operation

2. From the receiver to the sender
   connection accepted
   connection rejected

The message exchange will be the same whether the sender is an RFAP or an RFAC. As the RFAC–RFAC connection is more usual, we will consider it in this section; rules for RFAP–RFAC connection (a particular case of this general situation) may be deduced from those discussed here.

If the connection request has been rejected, the sender RFAC will send an answer to the requesting RFAP, specifying code X'08' (see SEND operation). The receiver RFAC will send another answer to the requesting RFAP,

specifying the reason for the rejection.

If the connection request has been accepted, the sender RFAC will send an answer to the requesting RFAP, specifying code X'02' (see SEND operation). The receiver RFAC will send a similar answer (see RECEIVE operation). When the data transfer has been completed, both the sender and receiver sides will send another message to the master RFAP, specifying code X'00'.

At this point, the data transfer protocol will begin.

DATA TRANSFER PROTOCOL

Every data block (sent by means of a network SEND operation) will consist of a block header and data. The block header is a three-byte field where

byte 1	X'00'	data field contains data
	X'01'	last block sent (data field contains the last block)
	X'02'	last block acknowledged
	X'08'	message is a status message
	X'10'	request for status
bytes 2–3	block number, the number of the block being transmitted	

If the message is a status message,

byte 1	X'10'
bytes 2–3	block number referring to the left edge of the window
byte 4	window situation. A bit with a value of 1 means that the corresponding block has been received correctly; a bit with a value of 0 means that the corresponding block has not been received.

The receiver side will send a status message every half window, except when a request for status arrives from the sender. If no additional space for storing data is available, the receiver will send a "last block acknowledged" message and will discard any arriving block. When the sender receives this message, the connection will be closed.

If no status is received, the sender will send blocks until the end of the window has been reached and will then start a "time-out" mechanism, waiting for the status message. If the time-out expires, a request for status will be sent and the time-out will be started again. As soon as the last block is sent, the sender will make a request for status to the receiver.

Because at least one network operation is active in each state (both that of the sender and that of the receiver), both sender and receiver can be notified at any moment if the counterpart will not be available owing to a network or host failure. In this case, the window situation will be saved and the connection will be considered closed.

We may now describe the protocol, using the well-known technique of the state diagram.

Sender

- If in the IDLE state, a request to send data will pass control to the CONNECTING state.
- If in the CONNECTING state, a connection rejected condition will close the logical channel and will pass control to the IDLE state.
- If in the CONNECTING state, a connection accepted condition will pass control to the SENDING state.
- If in the SENDING state, blocks are sent according to the window situation. A status received condition will pass control to the UPDATE WINDOW state. A last block acknowledged received message will close the connection and will pass control to the IDLE state.
- If in the SENDING state, an end of window condition will pass control to the WAITING FOR STATUS state.
- If in the SENDING state, a block sent condition will not cause any change of state.
- If in the SENDING state, a last block sent condition will pass control to the SENDING REQUEST FOR STATUS state.
- If in the UPDATE WINDOW state, the window updated condition will pass control to the SENDING state.
- If in the WAITING FOR STATUS state, a time-out mechanism will be started: the time-out expired condition will pass control to the SENDING REQUEST FOR STATUS state.
- If in the WAITING FOR STATUS state, a status received condition will pass control to the UPDATE WINDOW state.
- If in the WAITING FOR STATUS state, a last block acknowledged received message will close the connection and will pass control to the IDLE state.
- If in the SENDING REQUEST FOR STATUS state, a request sent condition will pass control to the WAITING FOR STATUS state.

Receiver

- If in the IDLE state, a request to receive data will pass control to the CONNECTING state.
- If in the CONNECTING state and a connection message is received, that message is checked. If accepted, control will pass to the RECEIVING state; if rejected, control will return to the IDLE state.
- If in the RECEIVING state, a block received condition will pass control to the CHECK SEQUENCE NUMBER state.
- If in the CHECK SEQUENCE NUMBER state, a duplicate block or a block out of the current window will be discarded; control will return to the RECEIVING state.
- If in the CHECK SEQUENCE NUMBER state, an accepted block will pass control to the UPDATE WINDOW state.

- If in the SEND STATUS state, the message sent condition will pass control to the RECEIVING state.
- If in the UPDATE WINDOW state, a no status point condition will return control to the RECEIVING state.
- If in the UPDATE WINDOW state, a last block arrived condition (if no more blocks have to be received) will pass control to the SEND ACKNOWLEDGE state.
- If in the UPDATE WINDOW state and no more space is available for storing data, control will pass to the SEND ACKNOWLEDGE state.
- If in the UPDATE WINDOW state, a status point condition or a last block arrived condition (if some blocks still have to arrive) will pass control to the SEND STATUS state.
- If in the SEND ACKNOWLEDGE state, the message sent condition will close the connection and will pass control to the IDLE state.

The situation is summarized in the oriented graphs of Figures 9 and 10.

RESTART PROTOCOL

If transmission is interrupted by a network or host crash, the window is saved at both sides, if possible. Transmission will be resumed, when possible, using a restart protocol.

The problem of deciding when the transmission has to be restarted remains unresolved; a network may or may not have a mechanism to alert users when the connection has been reestablished. Although this is not a trivial problem, we can solve it only by making additional assumptions about the nature of the network, which would make our description more specific. We can, however, list some suggestions to implement such a mechanism; these might include the following:

1. The network alerts the users (RFACs and RFAPs) who were interrupted.
2. The RFACs and RFAPs, who have a "wait" option in the CONNECT and LISTEN network operations, can issue such operations and wait until they are completed, at which time the connection will be reestablished.
3. The RFACs, RFAPs, or both start a time-out and repeat, respectively, the CONNECT and LISTEN operations when the time-out expires. If the operations are closed without error, the connection is established; otherwise, the time-out is started again.
4. The sender tries to connect the receiver only when a new request for file transfer, directed to that node, is issued.

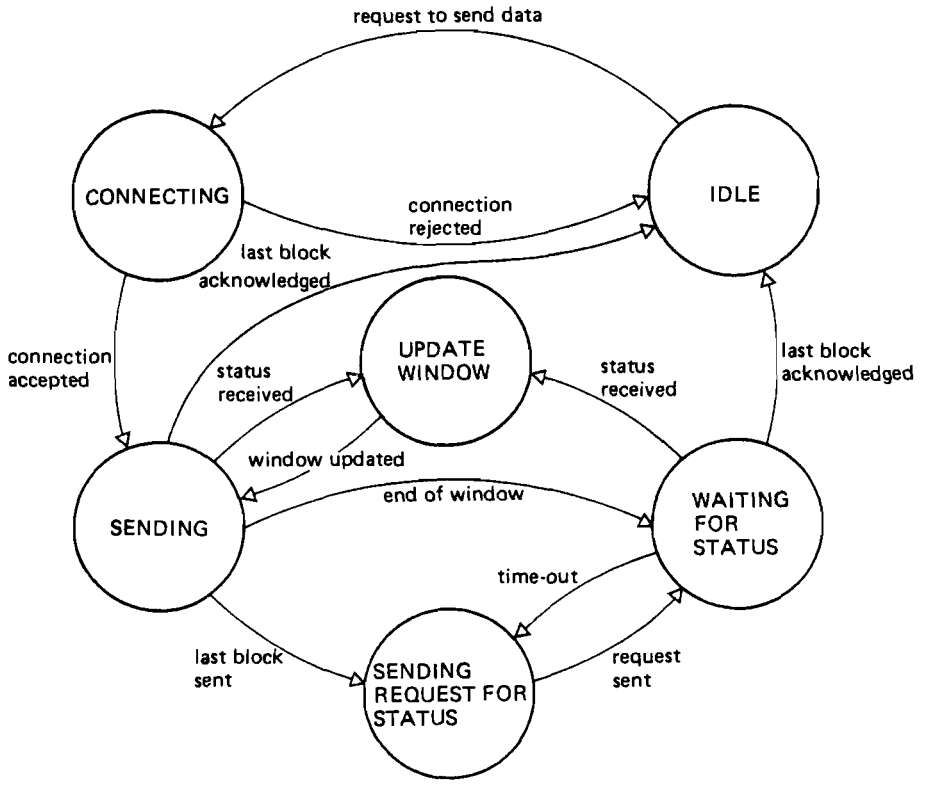

FIGURE 9 Data channel protocol: sender side.

In any case, the CONNECT network message for the sender will have the following format:

IND	BULK (X'10')	
NAM	user's number, as specified by the originator RFAP	
DATA	byte 1 (modifier)	X'10'
	bytes 2-3	block number related to the left window edge
	byte 4	window
	bytes 5 and following	network address of the originator RFAP

The receiver RFAC will look for an interrupted file transfer with the specified RFAP's name and number and will accept or reject the connection. Then the two windows are compared and the "more advanced" one is chosen as the actual window. A status message is sent back to the sender, and the normal data transfer protocol is started again. By "more advanced" window we mean the window that has the highest left edge, or, if the left edges of

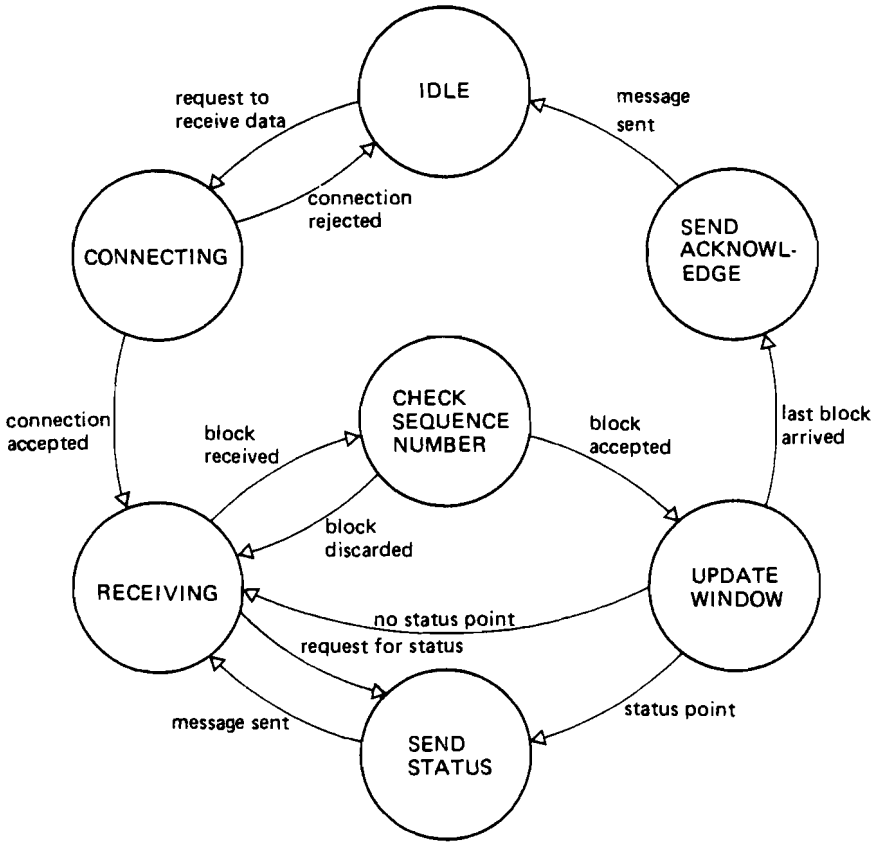

FIGURE 10 Data channel protocol: receiver side.

the two windows are the same, the window that contains more 1s in its configuration.

The two windows may be identical; the sender's window may be more advanced than the receiver's window; or the receiver's window may be more advanced than the sender's window. The first instance occurs when the network fails during a normal block transmission and the two RFACs have time to save their windows. The second can happen if the receiver's host crashes after sending a status message and before saving the updated window; in this case, the sender's window is true, while the receiver's window does not reflect the actual situation. The third case comes about if the network crashes during the transmission of a status message, or if the sender's host crashes after the receiver RFAC has sent a status message and before the sender is able to update its window; in this case, the receiver's window is true. The "most advanced window," in any case, reflects the true state of the transmission.

ACKNOWLEDGMENT

The authors would like to thank Dr. K. G. Beauchamp for his useful comments, which were especially helpful during the reviewing phase of this paper.

REFERENCES

Bauwens, E., and F. Magnee (1978) Definition of the virtual terminal protocol for the Belgian university network. Pages D3-1–D3-14, Proceedings of the Symposium on Computer Network Protocols, edited by A. Danthine. Liège, Belgium: Université de Liège.

Caneschi, F., E. Ferro, L. Lenzini, M. Martelli, C. Menchi, M. Sommani, and F. Tarini (1978) Architecture of and the service facilities provided by RPCNET – the Italian computer network for education and research institutions. Pages 695–701, Evolutions in Computer Communications, Proceedings of the ICCC (International Council for Computer Communication) Fourth International Conference on Computer Communication, Kyoto, Japan. Amsterdam: North-Holland Publishing Co.

Caneschi, F., L. Lenzini, and M. Sommani (1979) Remote file access in RPCNET. Pages 321–329, Proceedings of the European Conference on Applied Information Technology of the International Federation for Information Processing (IFIP), London. Amsterdam: North-Holland Publishing Co.

CCITT Study Group VII (1976) Draft Recommendation X25. Geneva: International Telegraph and Telephone Consultative Committee, International Telecommunication Union.

CCITT Study Group VII (1977) Editor's group on Level 3 of X25 "Proposed additional amendments and additions to X25," Temporary Document 75-E. Geneva: International Telegraph and Telephone Consultative Committee, International Telecommunication Union.

Csaba, L. (1976) Problems in the Design of the IIASA End-to-End Protocol. INWG Protocol Note 45. Teddington, UK: International Network Working Group, European Informatics Network, National Physics Laboratory.

Folts, H., and I. Cotton (1977) Interfaces: new standards catch up with technology. Data Communications June: 31–40.

Gien, M. (1978) A file transfer protocol (FTP). Pages D5-1–D5-7, Proceedings of the Symposium on Computer Network Protocols, edited by A. Danthine. Liège, Belgium: Université de Liège.

High-Level Protocol Group (1977) A Network Independent File Transfer Protocol. INWG Protocol Note 86. Teddington, UK: International Network Working Group, European Informatics Network, National Physics Laboratory.

Hovey, R. B. (1976) Packet switched networks agree on standard interface. Data Communications May/June: 25–39.

Schulze, G., and J. Börger (1978) A virtual terminal protocol based upon the "communication variable" concept. Pages D2-1–D2-6, Proceedings of the Symposium on Computer Network Protocols, edited by A. Danthine. Liège, Belgium: Université de Liège.

Shicker, P., and A. Duenki (1976) Virtual terminal definition and protocol. ACM (Association for Computing Machinery) Communication Review 6(4): 1–18.

Shicker, P., A. Duenki, and W. Baechi (1975) Bulk transfer function (proposal). EIN/ZHR/75/20.

Thomas, R. H. (1974) A resource sharing executive for the ARPANET. Pages 155–163, Proceedings of the 1973 National Computer Conference, New York. Montreal: AFIPS Press.

THE AUTHORS

Fausto Caneschi attended the University of Pisa and received his *Laurea* in 1975 in electronic engineering. Now researcher at CNUCE, an Institute of the National Research Council (CNR) of Italy, he worked at IIASA during 1978 and 1979 as an expert on protocols, particularly high-level protocols, for the Informatics Task of the System and Decision Sciences Area. He also supervised the IIASA–Pisa–Frascati connection, from organizational as well as technical viewpoints, and provided consultancy for IIASA computer users. Dr. Caneschi's work includes papers on computer networks and satellite communications.

Marco Sommani also attended the University of Pisa; he received his *Laurea* in mathematics in 1972. A researcher at CNUCE, he worked during 1979 at IBM's J. Watson Research Center, Yorktown Heights, New York, in the field of image processing. Dr. Sommani is particularly interested in computer science and applied mathematics, including computer languages and graphic optimization. His work includes several papers on computer networks, particularly file transfer protocols.

RELATED IIASA PUBLICATIONS

RM-74-18	An Adaptive Routing Technique for Channel Switching Networks, by D. E. Bell and A. Butrimenko.	$3.00	AS45
RM-76-07	Optimization of the HDLC I-Frame Structure and IIASA Data Communication Network, by Y. Masunaga.	$3.00	AS45
RM-76-31	Optimal HDLC I-Frame Structure in a Two–Way File Transfer and IIASA Data Communication Network, by Y. Masunaga.	$3.00	AS45
RR-75-11	On Stochastic Computer Network Control, by Yu. A. Rozanov.	$3.00	AS45