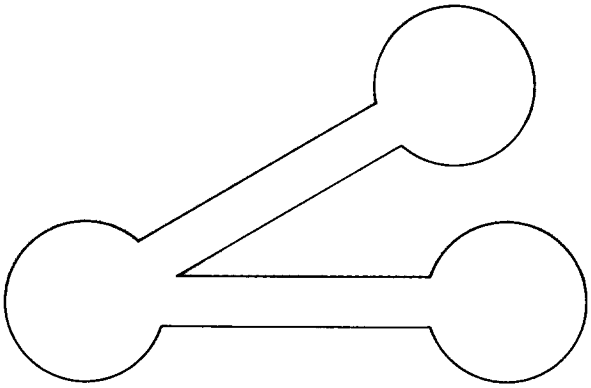
A File Transfer Protocol (FTP)

Michel GIEN
IRIA, BP. 105, 78150 Rocquencourt, France

This paper presents a File Transfer Protocol (FTP) which can be used to transfer files between heterogeneous computer systems connected by a communication network. The basic mechanisms as well as the set of protocol commands and responses have been defined in the context of a general architecture. A simplified architecture, to help first implementations is also proposed. This can easily be extended to the more general one. The proposed protocol is completely compatible (even identical in some areas) with the currently proposed Virtual Terminal Protocol (VTP).

Keywords: File management, file transfer, data transfer, high level protocols, distributed systems, network control language.



Michel Gien is Project Engineer at IRIA. Graduated Engineer from Ecole Centrale des Arts et Manufactures (Paris) in 1971, he worked for the CII-HB Research Center in Grenoble before joining IRIA in 1973. Attached to the Computer Center, he took an active part in the development and operation of CYCLADES, an experimental Computer Network linking universities and research centers in France. He is also involved in the IRIA participation to the EIN project (European Informatics Network). The design of network protocols is one of his main interests.

[1] This paper has been presented at the Computer Network Protocols Symposium, held in Liege (Belgium) in February 1978 and organized by the University of Liege. The permission to reprint this paper is gratefully acknowledged.

1. Introduction

The use of computer networks (1) generally starts with access to different systems from the same remote terminal. This is made possible by standard conventions to handle terminals over the network, provided by a Virtual Terminal Protocol (VTP) [2–6].

The necessity for some standard ways of handling files appears rapidly with the development of network usage, e.g. new software distribution, tape transfer, use of two computers at the same time (e.g. source program edited on one computer and compiled on another).

The VTP allows users to switch from one system to another, from the same terminal. To take full advantage of the network, the user also needs to be able to transfer his files easily between these systems. This raises the need for the provision of a Network File Management Service [7–15].

2. File management

2.1. Files

A *File* is a container of information, considered as a whole (i.e. it can be identified as such), without any reference to the meaning of the data it contains, as opposed to a *Data Base* which deals with the meaning and structure of the data itself, without reference to its "physical" representations within files.

Within a file, data is represented according to local operating system conventions (e.g. text characters can be coded in ISO code on eight bits, or in EBCDIC code . . .). A file is often structured in blocks of information units called *Logical Records* (or logical units). If a file is structured in logical records, then they represent the basic unit in which information can be accessed within the file.

2.2. File management

A File Management Service (FMS) usually provides two kinds of services: (a) a *File Access Method* which consists of a set of standard primitives to store data into files and retrieve it (often in terms of logical records); (b) a set of *File Maintenance Programs or Services* whose usual functions are (1) to make a copy of a file (i.e. move data from one container to another possibly with changes in its physical structure), (2) to delete a file (i.e. suppress all access to the file and its contents), (3) to rename a file (i.e. change its identification), (4) to reorganize a file (i.e. get back unused space), etc.

As opposed to *Data Base Management* which provides access to the actual information, file management permits manipulation of the "physical" representation of this information.

2.3. Network file management

A general *Network File Management Service* could consist of the extension of local file management services over a network. These services would be accessed through a standard File Management Access Method (FM-AM) which would include:
- a Network File Access Method, for accessing data contained in a remote network file, i.e. a remote file access method
- Network File Maintenance Services which would provide remote access to local file maintenance services
- a Network File Transfer Service to extend over the network the "copy" facility usually available locally.

The cooperation between the network File Management Access Method (FM-AM) and the File Management Systems (FMS), as well as between the File Management Systems themselves, would be ruled by means of a *File Management Protocol* (FMP) (see Fig. 1).

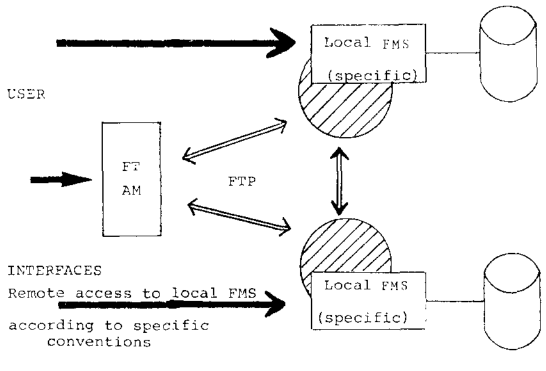


Fig. 2. Network file transfer.

3. File transfer

Except for the file transfer service, all other file management operations (i.e. file access and file maintenance) involve only one FMS at a time: the one handling the file. Remote access to local file management services of a system is provided to network users by means of a VTP (interactive or batch access), thus allowing the user to perform these file management functions. The burden, put on the user, is that he has to know the control language of every file system he wants to access. This would not be acceptable in the long run, but it is workable as an initial step.

On the contrary, file transfer operations, which involve cooperation between at least two systems (i.e. two file access methods), impose the definition of some common network-wide conventions, namely a *File Transfer Protocol* (FTP) to provide a file transfer service through a File Transfer Access Method (FT-AM) (see Fig. 2).

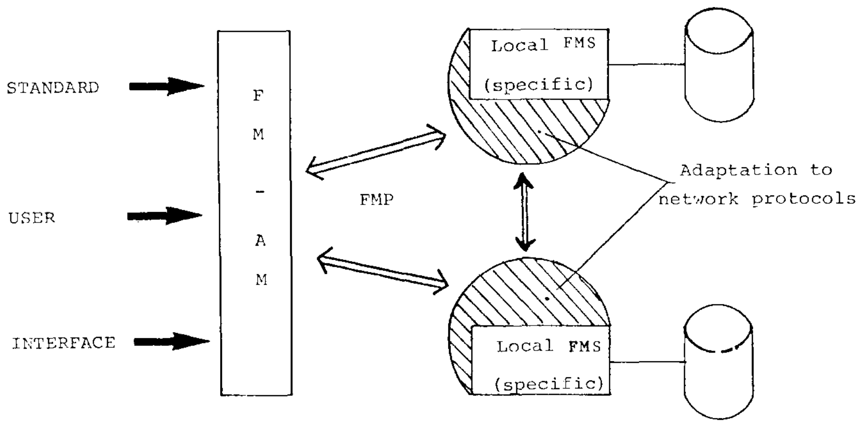


Fig. 1. Network file management.

4. File transfer standard conventions

Handling files, over a network, raises three main classes of problems. They concern: (a) File identification, (b) Data representation within files and (c) Mechanisms for transferring the data.

File identification problems have not been solved in their generality. No network-wide convention exists for naming a file. These would involve maintaining catalogs or directories of network files. The usual way of identifying a file over the network is to identify the system handling the file, using its address as known by the Transport service, and then use local conventions within that system. Standardization of files naming schemes is not specific to file transfer and is not of the concern of this paper.

Data representation within files depends on local FMS conventions. Files can be structured into logical records or not, and codes, byte sizes, etc. are usually different. One aspect of a file transfer service is that it must cater to some code conversions or file organization mapping between the systems involved, in order to give a useful service to the users. To make this possible, it is necessary to define some standard file organization and data representations that can be mapped on every system and that will be used as the *standard conventions*, to represent data travelling over the network.

The conventions defined for a minimum service cover only sequential files containing text information grouped into records, which constitute logical units (i.e. "text messages"), accessible individually and sequentially.

Except for homogeneous or compatible systems, other types of file organization and data representation are not considered. Nevertheless, a "*transparent*" mode of operation is provided, which allows the transfer of any kind of file between homogeneous or compatible file systems. Both standard and transparent modes of operation of the file transfer service, rely on the same synchronization, data transfer and error recovery *mechanisms*, which are defined in the following.

5. File transfer architecture

5.1. Architecture of a network File Management System (FMS)

A network file management system can be considered [7,10,11] as being composed of two kinds of components: (a) *Service Points*, providing access to local file handling facilities, and (b) *Control Points* which monitor and synchronize the operation of service points in order to perform actions on files, as requested by users (or application programs).

A sequence of file management operations, which involves one control point and one (or two) service point (s) is called: *Transaction.*

In order to complete a transaction, a dialog takes place between a control process (the *Controller*) at a control point and a server process (the *Server*) at a service point.

In the case of a file transfer one controller and two servers are involved: one controller at the control point, e.g. where the user is located; one server at the service point where the source file resides: the *Producer;* one server at the service point where the file has to be transfered: the *Consumer.* In this case, a dialog will also take place between the two servers (see Fig. 3). One server operates only on one file at a time and one controller and one server cooperate only for one transaction at a time. Several transactions could be handled in parallel involving the same control and service points: they will be considered here (i.e. in the model), as being performed by different instances of controller and server processes.

5.2. Communication means

To conduct their dialog, controller and server (s) communicate through the transport service, locally provided by a *Transport Station* (TS) [16–18] residing on each site. This transport service can be considered as an inter-process communication facility which establishes *Liaisons* between processes *Ports.*

On the liaisons established between file management ports, transport stations provide error control and flow control.

A liaison established between a controller and a server is called a *Control Liaison*; between two servers (for a file transfer) where it will be used to transfer actual data it is called a *Data Liaison* (see Fig. 3).

5.3. Simplified architectures for File Transfer

The general architecture proposed earlier, when applied to a file transfer, involves three liaisons (see Fig. 4). (1) A control liaison between the controller and the producer server, on which commands and replies C are exchanged; (2) A control liaison between

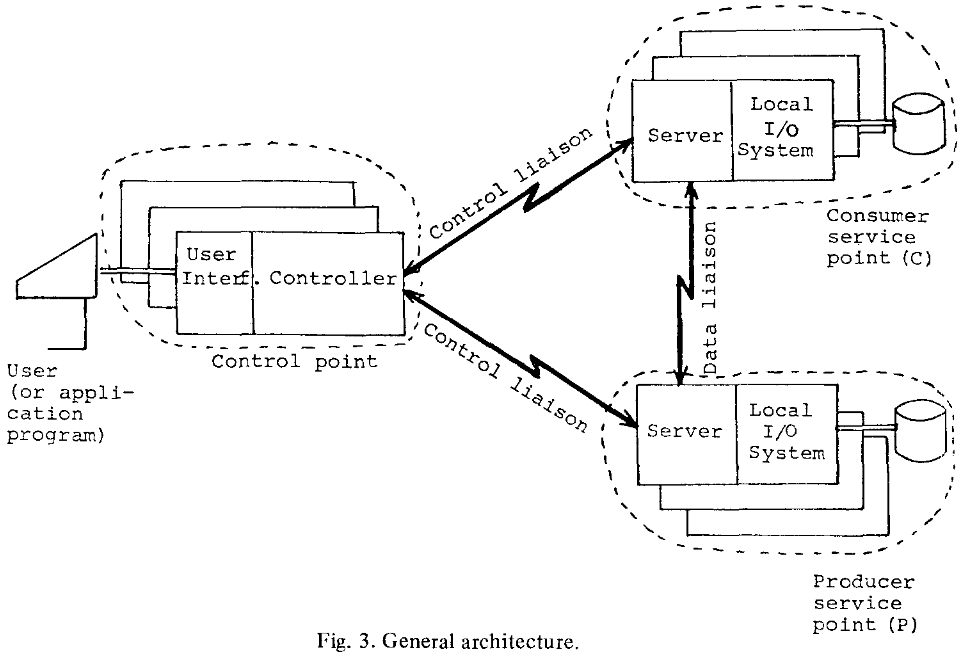


Fig. 3. General architecture.

the controller and the consumer server, on which commands and replies C′ are exchanged; (3) A data liaison between the producer and the consumer servers, on which data D are transferred.

A simplified model [7,13] can be derived from the one described earlier, where controller and server functions happen to be grouped within the same systems (see Fig. 5). In this case, user (or application program) requests must be directed to the site where one of the files resides.

This approach does not impact the basic mechanisms involved in the protocol; it simplifies synchronization and restart problems (in case one liaison is broken) between distant sites: only two distant points are involved and only two systems have to be controlled (instead of three), during a transaction.

Formats for exchanging commands and replies on the control liaison are different from formats used for transferring data on the data liaison. It is easy to distinguish them. Thus commands C′ and data D i.e. control and data, can be multiplexed on the same liaison to simplify again synchronization and restart problems between both ends (see Fig. 6).

This simple architecture allows easy experimentation on the basic File Transfer Protocol mechanisms.

As can be seen in figs. 4, 5, 6, the move to the general model where Control and Data transfer functions are well isolated, will have a greater impact on implementation and structure of the different modules involved in controller and server processes, than on the actual protocol primitives.

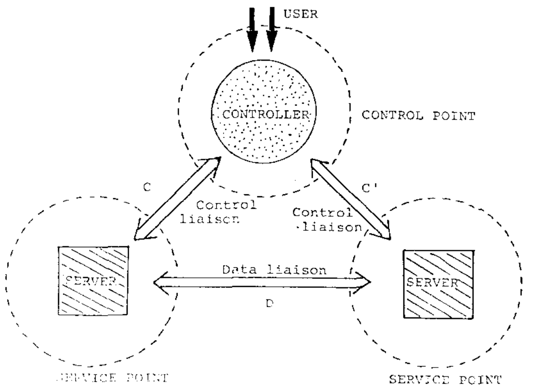


Fig. 4. File transfer general architecture.

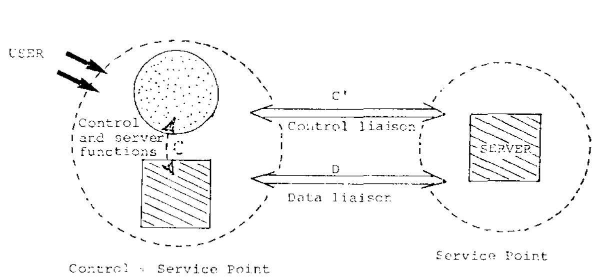


Fig. 5. File transfer simplified architecture.

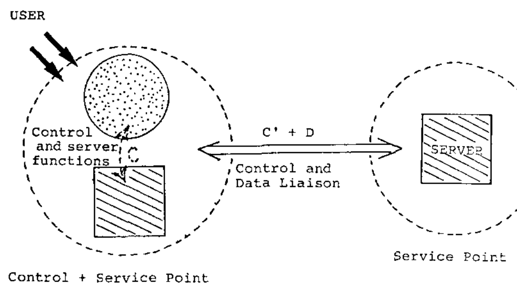


Fig. 6. Minimum architecture for file transfer.

6. User (application) interface

Even though they might be standardized elsewhere, user or application interface commands and responses are out of the scope of FTP. Nevertheless for the sake of clarity, it is assumed here that the user or application program gives to the controller which will be involved in the transaction, information on the nature of the operations to be performed as well as the necessary parameters associated with the files to be handled.

These parameters concern the file identification and will include: the identification of the site where the file resides (transport station address, network file transfer server port number) in order to access the service point concerned; the name of the file as known on that site; the access keys (account number, passwords . . .) necessary to access the file.
An example of commands appearing at the user interface could be:

– TRANSFER ⟨Producer file identification⟩ TO ⟨Consumer file identification⟩

– APPEND ⟨Producer file identification⟩ TO ⟨Consumer file identification⟩

7. File transfer primitives

File Transfer primitives are grouped into two classes: (1) commands/replies (exchanged on the control liaison in the general model) to monitor the file transfer operation; (2) data transfer primitives (exchanged on the data liaison in the general model) which allow a safe transfer of the actual data.

7.1. File transfer commands/replies

These primitives are exchanged in an alternate (demand/response) mode. They are used to initiate and terminate a transaction (i.e. a file transfer). They are:

D-SENDFILE ⟨Transparent/Standard Mode⟩ ⟨File Identification⟩ ⟨Rewind/Resume⟩ ⟨Mark⟩

and the correpsonding reply:

R-SENDFILE ⟨Reply Code⟩ ⟨File Structure⟩

or:

D-RECVFILE ⟨Transparent/Standard Mode⟩ ⟨File Identification⟩ ⟨File Structure⟩ ⟨Rewind/Append/Resume⟩

and the corresponding reply:

R-RECVFILE ⟨Reply Code⟩ ⟨Mark⟩

These pairs of commands and replies are used to initialize a file transfer, in either direction.

CLOSEFILE ⟨Code⟩

is used to terminate a transaction (i.e. a file transfer).

As parameters of these commands, file identification and file structure are given either in transparent mode when systems are homogeneous or compatible, or in standard mode when systems are incompatible. In this case, only a limited number of file structure types can be handled.

A position parameter indicates whether the transfer must be executed from the beginning of the file (Rewind) or if the file must be appended to an existing one (Append). A "Resume" indication allows to restart a transfer previously interrupted. In this case, the "Mark" parameter indicates a restart checkpoint from which to recover.

7.2. Data transfer primitives

DATA primitives are used to transfer the actual data (on the data liaison). The data blocks exchanged can be structured into *Records*, a concept which is similar to the VTP *Message* concept (2, 3, 4, 5). An *End of Record or Message* (EOR/EOM) indication is available to specify the end of a logical record. The last record indicates the *End of the File* (EOF).

1. In transparent mode, if specified by the initiation commands, (i.e. through the "mode" parameter of the D-SENDFILE/D-RECVFILE commands), the structure and representation of the data contained in the blocks exchanged (structured or not into records), is in a form known by the receiving end.
2. In standard mode, records contain data, structured

TABLE 1: Standard scenario

PRODUCER SITE (P)	CONSUMER SITE (C)
Following a User Request made at that site to transfer a file from one system (producer) to another site (consumer)	
- Contact with the server	- Logger listening for a contact
- - - - - - - - - -	- - - - - - - - - - →
	- Answer to the contact demand
← - - - - - - - - - -	- - - - - - - - - -
At this point the contact has been successful and a liaison established	
- Request for the consumer to be prepared to receive a file D-RECVFILE	
──────────	──────────→
	(- Allocate the necessary ressources) - Create (or append to) the file R-RECVFILE
←──────────	──────────
- Send Data DATA	
──────────	──────────→
DATA	
──────────	──────────→
- Put a checkpoint D-CHECKPOINT (n)	
──────────	──────────→
	- Put a corresponding Checkpoint R-CHECKPOINT (n)
←──────────	──────────
DATA	
──────────	──────────→
DATA	
──────────	──────────→
- Put a new checkpoint D-CHECKPOINT (n+1)	
──────────	──────────→
	- Put a new Checkpoint - Release the previous one R-CHECKPOINT (n+1)
←──────────	──────────
- Can release information related to the previous acknowledged Checkpoint (marker n) DATA	
──────────	──────────→
- When the end of file is reached CLOSEFILE	
──────────	──────────→
	- Close the file (and release all resources) CLOSEFILE
←──────────	──────────
- Close the file (and release all resources) - Close the liaison - - - - - -	- - - - - - - - - - →
← - - - - - - - - - -	- Close the liaison - - - - - -

TABLE 2: Recovery from a failure

PRODUCER SITE (P)	CONSUMER SITE (C)
- Data Transfer Phase :	
DATA ————	————→
DATA ————	————→
- Put a Checkpoint	
D-CHECKPOINT (n) ————	————
	- Put a corresponding Checkpoint
←————	R-CHECKPOINT n)
- Release Checkpoint (n-1)	
DATA ————	————→
DATA ————	————→
- System Failure	- Other end failure discovered - Save the last Checkpoint and stops
- Restart of the system	
- Recovery of the previous transfer	
- Contact	
- - - - - - - -	- - - - - - - -→
	- Answer to the contact demand
←- - - - - - - -	- - - - - - - -
D-RECVFILE <Resume> ————	————
	- Find the last acknowledged Checkpoint
←————	R-RECVFILE <Marker:n>
- Go to the Checkpoint <n>	
- Restart sending data :	
DATA ————	————→
⋮	

and represented according to standard conventions as specified in the "file-structure" parameter of the R-SENDFILE/D-RECVFILE primitives.

Control primitives (CTRL) are used to control the orderly progress of a transaction (i.e. a file transfer) e.g. to be able to recover later on, if a failure occurs. They are exchanged in alternate mode (on the data liaison). This exchange cannot occur in the middle of the transmission of a record, and can be activated only after an EOR indication has been transmitted. These primitives are: D-CHECKPOINT ⟨Mark⟩ sent by the producer after the setting up of a checkpoint and R-CHECKPOINT ⟨Mark⟩ sent by the consumer to acknowledge the setting up of a corresponding checkpoint.

7.3. Functional scenarios

A standard file transfer scenario is given in table 1. Table 2 gives an example of a transfer restarted after a failure occuring at one of the two ends.

8. Conclusion

A file transfer protocol has been defined which provides the necessary mechanisms to transfer a file safely from one place to another over a network.

Conversions between file structures can be performed so far only on a limited number of "standard" structures (e.g. sequential text files).

Otherwise file structures must be compatible and transfer be performed in transparent mode.

File identification problems have not been touched. They are considered to be more general problems, belonging to the area of a network job control language. Local conventions only are used here to identify a file.

References

[1] D.L.A. Barber, A European Informatics Network: achievement and propects, ICCC 76, Toronto, Aug. 76, 44-50.

[2] H. Zimmermann, Proposal for a Virtual Terminal Protocol (VTP), réseau Cyclades TER 533, Jan. 76, 16 p.

[3] N. Naffah, Protocole d'Appareil Virtuel type écran, réseau Cyclades IFR 503.1, Oct. 76, 61 p.

[4] EIN, Proposal for a scroll mode Virtual Terminal, edited by P. Schicker and H. Zimmermann, EIN/CCG/77/02, INWG note ≠≠ 62, Jan. 77, 45 p.

[5] EURONET, Data-Entry Virtual Terminal Protocol for EURONET, VTP-D/1, Jan. 77, 30 p.

[6] G. Schulze, A virtual terminal concept, PIX/VTP/GMD/77/02, Feb. 77, 27 p.

[7] N. Neigus, File transfer protocol, NIC 17759, ARPA RFC 542, July 73, 50 p.

[8] EPSS study group 2, File transfer protocol, HLP/CP(75) 3 issue 2, June 75, 46 p.

[9] DEC. DECNET, Digital network architecture, design specification for data access protocol, July 75, 31 p.

[10] P. Schicker/A. Duenki/W. Baechi, Bulk transfer function (proposal), EIN/ZHR/75/20, Sept. 75, 24 p.

[11] M. Gien, Proposal for a standard File Transfer Protocol, INWG note ≠≠, réseau Cyclades DAT 519, May 77, 48 p.

[12] A. Belloni/ M. Bozzetti/ G. Le Moli, A proposal for a Bulk Transfer Function, EIN/CREI/77/1, Jan. 77, 7 p. and EIN/CREI/77/3, Feb. 77, 24 p.

[13] J. Day/G. Grossman, An RJE Protocol for a Resource Sharing Network, NIC 38316, ARPA RFC 725, March 77, 26 p.

[14] W. Heinze/B. Butscher, File Transfer in the HMI Computer network, Third Europ. Network users' Workshop IIASA, Apr. 77, 13 p.

[15] M.N. Farza/G. Sergeant, Machine interprétative pour la mise en oeuvre d'un langage de commande sur le réseau Cyclades, thesis presented at the Fac. of Sciences, Univ. Paul Sabatier of Toulouse, réseau Cyclades LAN 511, Oct. 74, 231 p.

[16] V.G. Cerf/A. Mc Kenzie/R. Scantlebury/H. Zimmermann, Proposal for an International End-to-end Protocol, ACM Sigcomm. Comput. Comm. Review, Vol. 6, nr 1, ISO/TC97/SC6-Doc. 1282, INWG note ≠≠ 96, July 75, 29 p.

[17] H. Zimmermann, The Cyclades end-to-end protocol, 4th Data Comm. Symposium, Québec City, réseau Cyclades SCH 565, Oct. 75, 7-21.

[18] EIN, End-to-end protocol, revision 2, EIN/CCG/051175, Nov. 75, 22 p.