INDRA Note 753
IEN 99
May 3rd 1979

NI FTP: Summary and Assessment

P. L. Higginson
C. J. Bennett

ABSTRACT: This note is a brief summary of the NI FTP, its design aims, and the ways in which those aims are achieved. A comparison of the NI FTP and the DIN FTP proposed for AUTODIN II is contained in IEN 100.

1. The NI FTP Protocol

This section is a brief summary of the important features of the NI FTP protocol.

The NI FTP was written by a committee of people from eight different organisations in the UK who between them used a wide variety of computer systems and hardware. Computers used by the group were manufactured, among others, by DEC, IBM, ICL, GEC, PRIME and CTL. Drafts were widely circulated and discussed outside the design group. While the initial aim was to produce a protocol for transferring files across EPSS, "network independence" became an overriding aim. This was because it was realised that EPSS would shortly be replaced by a different, X25-based network, and because most of the eight organisations had multiple systems or their own local networks, and the ability to use the FTP in all these areas was desired. There are about 15 implementations in progress, of which 3 have so far exchanged files. UCL has a TOPS20/TENEX implementation which has transferred files across the ARPANET, and tests have begun on interworking with implementations in EPSS.

The NI FTP is a two-party file transfer protocol. The transfer occurs in two phases. In the first, the transfer is defined and the attributes of the data to be transferred are negotiated. In the second, the data is actually transferred. Three levels of protocol are recognised: level 0, which covers the negotiation of file transfer parameters; level 1, which handles error and flow control during the transfer, along with any data or time dependent changes in the file; and level 2, which is the actual transfer of data.

The negotiation phase is simply structured to be an exchange of file attributes between the two sides. These attributes define a common definition of the file and the transfer within a "conceptual filestore" supported by the two sides. During this phase, agreement is reached on such things as character code, file size, file name and access rights, all of which are regarded as attributes of the file within the conceptual file store. It is up to the two ends to map these attributes into locally acceptable forms for their actual filestores; if this cannot be done, the transfer may be rejected.

The currently defined attributes cover a basic set of properties of sequential files. This set can be easily extended, and facilities are provided to enable implementations to grow in an upwards compatible manner. Every attribute has a default value, which allows an implementor to construct a transfer facility as simple or as complex as desired, in the knowledge that his facility will be able to interact easily with more complex ones. This is possible because quoting an unknown attribute or unacceptable value for an attribute need not of itself cause a protocol error. During the negotiation phase, the more complex implementation will reduce its attribute set to one acceptable to the simpler one.

The data transfer phase can treat the data totally transparently, but does provide a user record structure for text files if desired. It allows for data compression as an option, which can significantly reduce the amount of data actually transferred. Even if this is not used, the protocol overhead can be as low as 1 byte in 64. For text files, the existence of different degrees of formatting sophistication is recognised by allowing a set of common format effectors to be agreed in the negotiation phase. The data can be in a variety of codes, as agreed during the negotiation phase, and the particular code in use can be changed during the transfer phase, thus allowing the protocol to handle special files such as job output files, graphics files with embedded text, or symbolic files produced by a compiler for debugging purposes. Binary files can use byte sizes of any arbitrary size as agreed in advance.

Error control is provided by the use of data checkpoints. These are inserted by the sender at suitable points, and may be matched to the characteristics of the file source or destination storage devices. The acknowledgement of a data mark implies that the data has been securely stored by the receiver. The use of a data window of unacknowledged data marks enables the sender to detect problems at the receiving end which will cause it to suspend the flow of data. On recovery (or in any appropriate circumstance), the receiver can ask the sender to resume the transfer starting from any mark after the last mark acknowledged. This facility also protects the transfer against loss of data within the network, and can even be used to resume an interrupted transfer in a separate session.

Flow control is provided by allowing the receiver to request the sender to suspend the flow of data, and to tell it when to resume. This is primarily intended to cover device interruptions (eg mounting a magnetic tape). The use of these options is again subject to prior agreement during the initial negotiation phase, as with all the other attributes.

The mechanics of the data transfer is left to the transport mechanism. The protocol is defined so as to make minimal assumptions about the transport service. These are: the transport service will provide a synchronised sequential bytestream between the two sides with an acceptably low rate of error, and that if an unrecoverable error does occur, there exists some way for the transport service to notify the application process. Recovery would then be handled by the error control mechanism described above.

2. Assessment of the NI FTP.

The design of the NI FTP was governed by four basic objectives:

(i) The protocol should be independent of the communication medium.

(ii) It should be flexible enough to satisfy a diverse variety of applications.

(iii) It should be able to interface easily to a wide range of operating systems.

(iv) It should allow "escape routes" such that special features of a particular operating system can be exploited where necessary.

The following features of the protocol were designed to meet these objectives:

(i) The minimal requirements necessary from a transport service were identified and adhered to, along with the recognition that unrecoverable errors (at transport level) may occur.

(ii) The definition of the file is in terms of a conceptual file store. No restrictions were placed on the mapping of this file store to the local file store, either in the attributes specification or in the definition of the negotiation process. The inclusion of option sets and relational operators in the attribute specification allows extensions to be easily incorporated into existing implementations and into the protocol itself. A number of parameters provide the required "escape routes" for areas where operating system dependent peculiarities are anticipated.

(iii) Complete transparency for naming files and for data transfer is possible.

(iv) The "records" of data transfer used within the protocol enable mixing of control and data within a single connection. They need have no relation to the record structure (if any) used within the file, unless the user so desires. The only restriction is that the communication medium should deliver eight bit bytes of data in sequence.

(v) The formats of control commands and attributes are standardised in a machine-oriented form. The action of the protocol is defined as a single file transfer transaction. Hence the dominant need for flexible command formats occurs in the negotiation phase, and the level 0 commands are very loosely structured to allow indefinite extension. Only a very small number of commands must be exchanged during the data transfer; for this reason, a more rigid structure was adopted for level 1 commands to ease processing overhead.