INDRA Note 754
IEN 100
May 3rd 1979

See also IEN 101-102

Comparison of the DIN FTP and the NI FTP

C. J. Bennett
P. L. Higginson

ABSTRACT: This note assesses the DIN FTP proposed for AUTODIN II according to design aims developed in IEN 99. The facilities available in the DIN FTP are compared with those in the NI FTP.

The FTP proposed by BBN for AUTODIN II is conceptually closely related to NI FTP. Many of its basic concepts are derived from it, including the fundamental notions of the conceptual file store and parameter negotiation. Many of the attributes used are extensions or restrictions of NI FTP attributes, although the coding is rather different. Thus the approach taken in this note is to view the DIN FTP as an alternative approach to realising the same design aims. The two FTPs are therefore compared according to the design requirements developed in IEN 99. Additionally, where the FTPs differ significantly in the facilities offered to the user, these facilities are compared directly.

DIN FTP AND THE NI FTP DESIGN REQUIREMENTS

(i) Transport service independence. The DIN FTP makes the following assumptions about the underlying transport service:

(a) The transport service will provide a synchronised sequenced stream of octets between the two ends. This is the same as the NI FTP.

(b) All recoverable errors are handled by the transport service. (This is by implication, as the point is not explicitly discussed. This is certainly the level of service provided by TCP). As the DIN FTP incorporates all the recovery mechanisms of the NI FTP, this defect can be easily remedied, but as the specification stands it cannot be used above X25 or similar services, which use a transport service RESET involving loss of data.

(c) The identity of the host of the connection initiator will be made available by the transport service (as all references to the 'Controlling Host ID' and 'Active Host ID' are by citation). This is of course a reasonable assumption but not a necessary one. For example, the UCL NI FTP project is investigating NI FTP above mapped concatenated virtual calls. In this situation, the only addressing information which could be supplied from a transport service is the address of the last transport protocol convertor.

(d) Any implementation using the UserID parameter assumes the transport service will perform authentication actions at all sites involved in the transfer. This assumption is only true of the Secure TCP; hence implementations using this parameter can only be used on AUTODIN II. We will return to the issue of access control below.

(ii) Supportable applications. The DIN FTP has removed several options of the NI FTP which are useful in this regard. These include:

(a) The ability to mix codes in a file. There are several applications where this is useful, such as job output and graphics. An example of more immediate interest is mixed FAX and text messages. This restriction is arbitrary and could easily be removed from the DIN FTP (though the FileDataType parameter must be redefined to be bit-coded).

(b) Knowledge of the output device type. This prevents the DIN FTP from being used in spooling applications.

(c) The distinction between selectors and modifiers. This is a property of the conceptual filestore. Its use is not very apparent in the NI FTP as such, but it enables other protocols such as Job Transfer and File Management protocols to be built easily. (Consider the difference between 'Select File with FileName X' and 'Modify (selected) FileName to X'.) The topic of other protocols will be returned to later.

(d) The ability to readily trace file transfers in system logs through the Monitor Message parameter. This is particularly useful for error tracing and for following up user complaints.

(iii) Operating system independence. The DIN FTP has removed several attributes which will make it difficult to implement it on some operating systems. In particular we note the following NI FTP attributes omitted in the DIN FTP which may cause problems in this regard:

(a) Maximum Transfer Size. This refers to the amount of data that will actually be transferred; the Maximum File Size is the size the resultant file may not exceed. The two are conceptually different, and may have different effects. On some systems (eg IBM's OS360, GEC 4080) the Maximum File Size, reserved at file creation time, sets a limit for all eternity, and must cover all anticipated appends. The DIN FTP's MaximumFileSize parameter is in fact a MaximumTransferSize parameter.

(b) Filename Password. Files may legitimately be accessed (even on TENEX) by giving an appropriate key without the user being recognised as the owner of that file. Some IBM systems can require a legitimately logged in user to provide an additional file-specific password before access is granted to a file.

(c) Account Password. Usernames and Accountnames are conceptually orthogonal. In systems where a 'username' corresponds to a superdirectory shared between many different users, these may distinguished by accounts with a separate level of password protection. An example of this kind of double lock is the superuser mode on UCL's UNIX.

The issue here is not one of meeting the peculiarities of particular operating systems. The rationale behind such features of the NI FTP is that they represent different fundamental concepts, and that even though some systems may not find the distinctions important, it was foreseen that some implementations and applications would find them very necessary. Nevertheless, it should be stressed that the NI FTP design group had experience of a wide range of operating systems, most of which are commonly used in the UK. The attributes introduced were all found to be either immediately necessary, or represented areas in which it was felt "escape routes" should be provided.

(iv) Extendability. There are two aspects to this question. One is the provision of "escape routes", mentioned above. The NI FTP spec identifies several areas where such escape routes are needed, notably: Access Control (via the 'Account Password', which, as we saw above, can be used to cover secondary levels of protection); File Description (via 'Kinship'); Data coding (via the 'Private Codes' selector); and special processing (via 'Special Options'). With the exception of Private Codes, these are omitted from the DIN FTP. The DIN FTP does provide a 'HostHardwareTypes' escape parameter; this extends a feature which is not supported by NI FTP for reasons discussed below.

The second aspect is protocol extension. As discussed above, the NI FTP readily extends to new attributes during the negotiation phase. The inclusion of new data transfer management commands (level 1) is more difficult than with DIN FTP, but the memoryless transfer model means that very few such commands should really be needed. The mechanisms for negotiating sets of protocol extensions (with option sets, relational operators and so on) are very similar in both protocols. (A minor point is that the NI FTP has attempted to bit-encode option selectors and operators. This allows, for example, a natural extension of the protocol to include LT and GT operators.)

FACILITIES COMPARISON

(i) The three party transfer model. The approach taken by the DIN FTP is not actually incompatible with the NI FTP, and one could quite reasonably implement an NI FTP (or a whole family of implementations) which used a private Controller/Active protocol, with extensions such as 'SourceFile', 'SourceUser', etc, and necessary additional commands such as 'Disconnect'.

(ii) Access Control. We have presented above our reasons for believing that the DIN FTP is inadequate as a general-purpose FTP in this regard. The (very strong) access control available on AUTODIN II is transport service specific, and implementations which support it are not portable. Use of this facility will tend to create a "closed community" of FTP implementations. The network-independent attributes provided for access control will not always be sufficient. The three party model used also allows the Active site to inspect the access control parameters issued by the controller for the Passive, which could lead to security problems when authentication is an important issue. (Obviously this cannot happen in a two-party model).

Access control is not really an FTP issue. Different systems can choose different methods: source host lists, recall addresses, and password protection are three. An FTP should have hooks for providing information to systems where the access control is password based, but this is simply a channel for contending with one type of access control. The actual requirements for access control are implementation-specific. The argument that single-host ownership of files is a major impediment to distributed file sharing is undeniable, but we feel that it is not the job of an FTP to solve this problem. As and when solutions become available they should be exploited by FTP implementations, but the best that the protocol specification can do is to recognise the existing state of chaos.

(iii) The DIN FTP provides formal grades of implementation. The NI FTP is more flexible, as it allows specific implementations to suit their own needs and facilities, and it is only recommended that all implementations support a defined set of defaults, while 'core' DIN FTP attributes are all required. Thus a very specific NI FTP implementation can be built for a specialised device which does not support unnecessary default values. (We at UCL intend to exploit this feature of NI FTP in our FAX project).

At a more philosophical level, this reflects the design environments from which the FTPs emerged - the NI FTP is being offered as a basis for standardisation for any group. These cannot be controlled by any central or regulating body; the DIN FTP assumes the more closed and controlled environment of AUTODIN II.

(iv) The Data Transfer Protocol. We are unconvinced that the formal separation of a data transfer protocol is a useful extension. As far as the FTP is concerned, it is little different from the NI FTP records during the data transfer phase, but complicates the formatting of NI FTP commands considerably. An NI FTP SFT, as generated by the current TENEX version, may occupy 78 octets complete with headers; with the DTP the equivalent SFT requires 28 sets of headers instead of 2, and occupies 131 octets. (With data segments, the NI FTP is more efficient for records of less than 126 bytes, and the DTP becomes more efficient than NI FTP for records exceeding 189 octets, though the difference is marginal). Where efficiency may become important is in data compression. The NI FTP breakeven point for compression is 3 data bytes; for the DIN FTP it is between 5 and 7 bytes (depending on how many control headers are involved).

However, the basic point at issue is whether the DTP is the right place to provide hooks for more sophisticated protocols. We contend that most such protocols (job entry, FAX, graphics, mail etc) will rely on primitives manipulating whole files. Hence the point of departure should be the conceptual file store and the attributes and negotiation mechanisms used to control a file within it. Once this has been accepted, protocols performing other file-based functions can use these common descriptions and mechanisms.

(v) Partial File Transfers. This is a useful facility available only in the DIN FTP. The associated abilities to describe file structure as indexed, variable record etc, and to nominate the keylength field length are also useful. The fact that the NI FTP is restricted to sequential files is recognised to be a major limitation. In the NI FTP model, an extension to handle partial file transfer could negotiate the pair of transfer delimiters during the negotiation phase, and transfer a single sequential section of the file during the transfer phase. Thus the feature is supportable by extension, but requires the definition of several new parameters.

(vi) Multiple file transfers. Both protocols can readily support multiple file transfers, though the details vary. The NI FTP requires the passive side to be reinitialised each time, which protects against parameter interdependencies persisting between transfers (see (vii) below). Thus the facility becomes a problem for the designer of the user interface. The same remark, of course, apples to multiple partial transfers.

(vii) Parameter Negotiation. This is much more formal in DIN FTP than NI FTP. In particular, parameters are explicitly grouped into different commands (Login, TransactionDescription and TransferDescription). The reasons for preserving this distinction beyond the user interface rather than issuing a single SFT, which allows an operating system total freedom to make its own weird assumptions about the relationships of parameters, are unclear.

The whole problem of interactions between parameters is potentially very complex, and it was for this reason that the NI FTP design precludes multiple attempts at SFT after a rejection. The parameter settings left from the last transfer, or negotiation attempt, may have undesired consequences on the next.

The DIN FTP also allows an attribute to be specified several times within a command, which the NI FTP prohibits (except for one experimental version). The specification uses multiple citations of SourceFileName as the example for this. Since it is not intended that this ability be used to set ranges of restrictions (eg ByteSize ge 8 and le 36 but ne 22), this case seems to be the only possible use outside of statistics collection.

(viii) File Management Primitives. The NI FTP group has taken the attitude that file management is the proper function of another (compatible) protocol providing the complete range of facilities and using the same conceptual filestore structure. Hence the NI FTP's file manipulation, as expressed by the Mode of Access attribute, is solely related to copying files. The access modes of NI FTP are all supportable by the DIN FTP, with the exception of 'destructive read'. DIN FTP provides two file management primitives: Delete and ListNames. Delete is, of course, easy to implement. ListNames is a directory interrogation facility, and is primarily useful for obtaining lists of filenames for subsequent multiple file transfer. This is of more limited use with the NI FTP's two-party model than with the DIN FTP's three party model, and can only be used to full advantage where grouped file specification is possible.

(ix) Foreground and background mode. In practise, we can see very few differences between them, as all three parties must be involved in the entire negotiation phase. Essentially, the difference is that the Controller/Active and Active/Passive connections do not have to be simultaneously open but can be opened as needed (eg the controller may have to answer a LoginRequest from a passive in background mode) that is, in background mode all three parties retain records of the transaction specification. This is a consequence of the three-party model; in the NI FTP it is a user interface issue and an implementation may be either foreground or background.

(x) Host Type. There are two ways this type of parameter can be used. It is clearly useful for a user interface to offer shorthand profiles for various parameter settings. The DIN FTP takes the alternative approach in which the parameter allows transfers between identical systems in efficient ways, but these are not defined within the DIN FTP specification. We do not feel this is desirable, and we question the implied assumption that it is not necessary. Noncommunicating families of FTPs can build up local interpretations for parameters of this sort. When, at a later date, it becomes possible for two such groups to interconnect, the local interpretations will be found to be incompatible. This facility again reflects the fact that the DIN FTP is developed for the AUTODIN II environment; in our opinion, this type of facility should be implemented in the user interface via a profile mechanism in a more open environment.

(xi) Format Effectors. All additions made by the DIN FTP to the NI FTP list can be adequately described within that list (with the ANSI Fortran Format setting). The DIN FTP restriction that bits 1, 2 and 4 cannot interact is unnecessary (For example, it is perfectly reasonable to mix imbedded Horizontal Tabs with the reqiuirement that End-of-Record implies end of line). We doubt whether the ability to change format settings during the transmission of data is a capability which will be widely used. In any case, we do not think that there should be a mandatory TextFormatter between records, and feel that the restriction that DTP segments should only contain whole lines is unnecessary. The Tabstops attribute may be a useful idea. As is currently defined, however, it requires the receiver to keep table space for it; we recommend a fixed tab interval that can be changed as an option. This approach could easily be supported by the NI FTP. We note that the NI FTP's earlier breakeven point for compression makes compression an attractive approach to this problem.

(xii) Rollback. Extending Rollback to allow the sender to request the receiver to rollback is probably a good idea. However, it is potentially more difficult for the receiver to roll back than it is for the sender as his internal checkpoints may not correspond at all to the FTP's CheckPoints (which are likely to correspond to internal checkpoints of the sender). Hence he may have to keep track of at least a window's worth of FTP checkpoints, whether acknowledged or not. Additionally, he must be able to retrieve the stored data. Some acknowledgement strategies (eg Ack a full window only) will sometimes