INDRA Note 698
IEN 59
October 23rd 1978

The UCL Transnet File Transfer Implementation

C. J. Bennett

ABSTRACT: This note describes the UCL plans for implementing a transnet file transfer service between sites in the US and sites in the UK based on the Network Independent File Transfer Protocol. Some familiarity with both the NI-FTP and the future UCL configuration is assumed in this document.

Introduction

The future UCL connection to the body of the ARPANET [Kirstein78] relies on the existence of adequate communication protocols with mappings taking place between protocols at the same level at points where no end-to-end direct communication is possible. Access from hosts at the UCL site to the ARPANET will be through TCP eventually, although NCP access may be retained in the short term via the NCP tunnel. Access to hosts not running TCP will require an NCP/TCP translation service. On the UK side, various UK and European networks will be accessed through X25, and hence a TCP/X25 protocol convertor is required; in the short term, access to EPSS via the current SWITCH gateway will use a mapping between the NCP and the EPSS bridge.

For supporting end-to-end services above transport level in this situation, we require that the mappings preserve the invariant properties required of the end-to-end transport service - namely reliability, sequencing, and error detection - and that they support the service parameters required, such as the appropriate throughput/delay characteristics, by implementing flow control strategies appropriate to the nature of the service. We also require that the service protocol itself be network-independent; that is, the properties it requires of the transport service should be an absolute minimum.

File transfer has been selected as the pilot service for a study of the implementation of internet services based on these principles. There are several reasons for this. Firstly, a network independent file transfer protocol (referred to in this document as the NI-FTP) has been defined [HLPG77] which is currently being implemented on several sites on EPSS. This fulfils one of the main requirements of such a project, and it provides a natural set of implementations against which a transnet version can be tested. Secondly, it is a service with simple and clearly defined service requirements: maximise bandwidth and minimise end-to-end delay, given that the transmitter can keep the transmitting interfaces permanently busy. Thirdly, it is clear that a file transfer service must be provided, and neither of the two alternative methods of doing this - staged transfers of complete or partial files, and direct mappings between local file transfer protocols - are likely to produce an acceptable service. In some circumstances, staging is unavoidable. For instance, transfers using the UCL PDP9 as a HASP RJE terminal must be based on the current ARPANET FTP, due to the exponentially large amount of work that would be needed to alter the current situation. However, such staging can be kept to a maximum of two intermediate stores by use of the NI-FTP; this question will be investigated in more detail later in this document.

The main aim of this note is to outline the implementation plans for the NI-FTP project. The first stage - the production of a TOPS version of the NI-FTP - is well in hand. Later stages will be coordinated with the development of the new UCL configuration.

Implementation of the NI-FTP on the ARPANET.

The NI-FTP is currently being written on the TOPS-20 system at ISIE. It is being written as a user program in BCPL, but all systems interfaces for the filing system and for the network, use JSYS calls directly. The implementation is based on the state diagrams given in the appendix to the NI-FTP specification. As much code as possible is being written so that it is common to both the initiator and responder sides of the FTP; in this way, both sides are being developed simultaneously. As it turns out, everything except the state machines for the two sides, and the user interface for the initiator side, can be written in a common version for level 0, which handles the transfer initiation and specification. Levels 1 and 2, which form the control of the transfer, and the actual transfer itself, are being written as two separately runnable modules, for sender and receiver, which will be activated appropriately on entering the data transfer state.

The user interface assumes, as in the ARPANET FTP implementations on TOPS, that the transfer will take place in the user's real time, partly for ease of implementation, partly because of the realtime nature of the TOPS system. The user is prompted to provide details of the file specification. Currently, it is planned to support the transfer of both IA5 files (with variable record size based on recognised EOL delimiters) and binary files (the selection of record size being a matter for study); create, replace, delete and append facilities; data compression; and packed or aligned transmission, with acceptable byte sizes of 7 or 8 (for character files) and 36 for binary. Enhancements of the working system may take place at a future date.

It is planned that the FTP will be runnable above both TCP and NCP; for this reason, the network interface is being left to the end of the development of the NI-FTP. A simulated packet exchange takes place between the two sides via the TOPS file system. A testbed program has been developed which runs the two sides alternately, and this has proved to be a useful aid in its development, coupled with the BEDIT binary edit program.

The principal problem that must be resolved before the NI-FTP can be interfaced to either TCP or NCP is the question of standard socket numbers. Currently, there is no control on the allocation of TCP socket numbers, and so there is, temporarily at least, no theoretical difficulty. However, the only machine on which TCP runs and on which we have an account is SRI-KA; it is unlikely that this situation will be changed until version 4 is released early next year at which time the user interface to TCP will also be changed. For this reason, effort will be concentrated on getting up an NCP version, and here the allocation of socket numbers becomes much more difficult administratively. Once socket numbers have been allocated, demultiplexing can be achieved through the use of multiple connections on listening sockets on the responder side.

So far as is known to us, this implementation is the only one definitely planned for the ARPANET, although there is a possibility that a version will be developed for the NORD-10 at NDRE. An implementation under UNIX is planned as a student project at UCL, and this may be installed on our SYSTIME and interfaced to TCP if it is successful. However, the only local ARPANET tests that can be made with any certainty will be between versions of this implementation running on different TOPS or TENEX machines. Tests against a different implementation will have to await the development of a support module interfacing the NCP to EPSS in the UCL PDP9.

Establishing the NI-FTP Service Support

The various points at which transport services are to be mapped contain service support modules for the FTP. As far as the transport service is concerned, these will appear as the destination FTP - they will occupy the same standard sockets and so forth. Their function, however, is much closer to that of gateways in a virtual circuit catenet. They must support the establishment of an end-to-end connection, and must ensure that all packets are delivered in order and without loss or duplication. They must also handle functions such as flow control, fragmentation, and possibly intermediate reassembly. Finally, they must initiate and transmit 'Transport Service Resets' to connections affected by network failures.

A virtual circuit catenet structure provides a means of solving these problems fairly readily. An address space of known NI-FTPs and service support modules (SSMs) is established, which will be related to the usual host IDs via a simple mapping function. Routing will be fixed by routing tables in the SSMs, with alternate paths built in in case a transfer is initiated after a failure at some intermediate node. As the routing tables need only consist of destination transport spaces plus destination NI-FTPs in the current transport space, they need not be very large. In order to notify the two ends of connection establishment and clearance, one must augment the protocol to include CALL ESTABLISH and CALL CLEAR packets and their responses. It is a matter for further study whether to withhold the establishment or clearing of each individual connection until it is known that the next step is established or cleared - that is, the connection can either be established piecewise, or end-to-end via a domino effect on each individual link.

The simplest way to maintain sequencing and ordering across transport service boundaries is to handle each connection separately, by using each transport service to demultiplex its traffic beforee handing it to the SSMs. The exact strategy to be followed at this stage depends heavily on the details of each transport service, and the operating systems of the machines in which the SSMs are running; it is possible that in some machines it may only be possible to support a fixed and limited number of connections.

The other two crucial areas handled by the SSMs are fragmentation and flow control. Flow control on a hop-by-hop basis should ideally be controlled by the SSM, and this is perfectly feasible when it is interfacing to TCP or X25, but not with the NCP, which can cause the SSM to drop packets if it has no buffers available. Thus a different strategy must be evolved for each transport service, and in cases where this may be difficult frequent transport service reset may be expected. We do not intend to offer RR (Restart Request) support in the first version of the NI-FTP, but hopefully, transport service reset will be sufficiently rare for this not to be a major drawback. End-to-end flow control can be based on the Mark facility supplied by the transfer. This is intended to be used to reflect the differing properties of the real file stores, and using it for this purpose, while not violating the protocol, does bend it somewhat - it may mean that in practise all transnet transfers musy be disc-to-disc.

Fragmentation is of course a classic gateway function. One feature of the NI-FTP project is that it will be possible to study fragmentation/reassembly tradeoffs in great detail - in particular, optimal choice of initial packet size and intermediate reassembly under virtual circuit conditions. Clearly, it will affect the choice of buffer sizes in the SSMs, and consequently the interaction of flow control strategies between adjacent transport services.

It is expected that at least two and probably three SSMs will be built in the course of the project. One will support an NCP/TCP conversion, one a TCP/X25 conversion, and the third an NCP/EPSS conversion. On current timescales it would seem that the third is the one which will be built first, although it is unlikely that it will become a permanent part of the system, or that it will support all facilities. This one will be part of the PDP9 SWITCH system; the TCP/X25 SSM will be built on an LSI under MOS, and the NCP/TCP SSM will be built either on a TOPS/TENEX or on another MOS system.

The NCP/EPSS conversion will mainly be used to provide access to EPSS for tests of the ARPA implementation against EPSS ones, for a short period only. For this reason, rather than building a full SSM, we may patch in a SWITCH %CONNECT call into the ARPA implementation, so that we can use the existing terminal conversion facilities to set up an end-to-end call without modifying the SWITCH system. If this decision is made, testing with EPSS should be possible as soon as the ARPANET implementation becomes operational.

Staged Transfers

Some of the motivation for installing staged transfers has been mentioned already. Two particular reasons are that it may not be possible to add NI-FTP service without difficulty, where an FTP service already exists; and that the user wishes to do a transfer to a device other than a disc. This possibility may, of course, be perfectly reasonable using the NI-FTP, but the perversion of the Mark facility suggested here for the transnet service may make it difficult to do such a transfer transnet.

Where staging is necessary, the use of the transnet file transfer facility should make it unnecessary to have more than two intermediate storages at the most. The transfer can be thought of as taking place in three stages: from 'local' source device to a disc local to the NI-FTP; from disc to disc in the transnet transfer, and from remote disc to remote destination device. The source and destination devices may be anything, and the first and last sections of the transfer will be done by the local network's local FTP.

A user with sufficient patience and with terminal access to all four hosts could, of course, conduct such staging manually. Automation of the procedure would require some changes to the NI-FTP implementation described above. The user interface would have to be extended to include a 'staging' option, whereby the user names the true source and destination of the transfer, and also the NI-FTP host that the transfer must be routed through. An interface to the local transfers would have to be built. The initiator NI-FTP would have to provide a mechanism for delaying and queuing transfer requests. Some form of error reporting would have to be developed. Finally, it may be necessary to break down the transfer so that only partial files are staged; this possibility may create some serious design problems.

Conclusions

The NI-FTP project falls into three major phases. The first - implementation of the NI-FTP on the ARPANET - is well in progress. The second - the development of the SSMs - is in the design stage, but development depends on the overall development of the UCL plan. The final stage - staged file transfers - will only be tackled after the first two stages are reasonably complete.

References

[HLPG77] - Higher Levels Protocol Group: A Network Independent File Transfer Protocol, INWG Protocol Note 86
[Kirstein78] - P. T. Kirstein: Some Problems in the Interconnection of Computer Networks. INDRA Note 697, Annex A.