Network Working Group M. A. Padlipsky
Request for Comments: 647 MITRE-TIP
NIC: 31117 November 1974

A PROPOSED PROTOCOL FOR CONNECTING HOST COMPUTERS TO ARPA-LIKE NETWORK VIA DIRECTLY-CONNECTED FRONT END PROCESSORS

by Michael Padlipsky

with the advice and general agreement of

Jon Postel and Jim White (SRI-ARC), Virginia Strazisar (MITRE)

and the general agreement of

Tom Boynton (ISI), Frank Brignoli (NSRDC), John Day (CAC), Bob Fink (LBL), Carl Ellison (UTAH), Eric Harslem (RAND), Wayne Hataway (AMES-67), John McConnell (I4-TENEX), Ari Ollikainen (UCLA), Dave Retz (SCRL), Al Rosenfeld (CASE), Stan Taylor (BRL), Doug Wells (MIT-MULTICS).

All affiliations are shown only for identifying ARPANET involvement, and do not necessarily denote organizational approval of the positions taken here.

No -- repeat NO -- expression of general agreement should be taken to apply to Appendix 2, which is exclusively a personal viewpoint.

INTRODUCTION

As this proposal is written, in the fall of 1974, the ARPA network has achieved sufficient acceptance that a rather large number of organizations are currently planning either to attach their general purpose computer systems directly to the ARPANET or to interconnect their systems employing "ARPANET technology". The authors have been in touch with efforts sponsored by the Air Force systems command, the Naval Ship Research and Development Center, the Defense Communications Agency ("PWIN" -- the Prototype World-Wide Military command and Control system Intercomputer Network), ARPA (The National Software Works), the AEC, and other government agencies. A common characteristic of these networks and the sub-networks is the presence of a number of systems which have no counterparts on the current ARPANET; thus, hardware "special interfaces" (between the host and the network Interface Message Processor) and -- more important -- Network Control Programs cannot simply be copied from working versions. (Systems include CDO 6600's, XDS Sigma 9's, Univac 494's, 1107's, 1108's, 1110's, and IBM 370's running operating systems with no current ARPANET counterparts.) Because it is also widely accepted that the design and implementation of an NCP for a "new" system is a major undertaking, an immediate area of concern for networks which employs as much off-the-shelf hardware and software as is practicable. This paper addresses two such approaches, one which apparently is popularly assumed as of now to be the way to go and another which the authors feel is superior to the more widely known alternative.

FRONT-ENDING

In what might be thought of as the greater network community, the consensus is so broad that the front-ending is desirable that the topic needs almost no discussion here. Basically, a small machine (a PDP-11 is widely held to be most suitable) is interposed between the IMP and the host in order to shield the host from the complexities of the NCP. The advantages of this fundamental approach are apparent: It is more economic to develop a single NCP. "Outward" (User Telnet) network access is also furnished by the front end acting as a mini-Host. The potentiality exists for file manipulations on the mini-Host. Two operating systems are in advanced stages of development on the ARPANET for PDP-11's which will clearly serve well as bases for network front ends; thus, the hardware and software are copiable. So if we consider a model along the following lines

```
Host *** Front End --- IMP --- Network
```

everything to the right of the asterisks may almost be taken as given.

(Caveat: Note the "almost" well in last sentence neither ANTS nor ELF -- the two systems alluded to above -- is a completely finished product in the estimation of either their respective developers or of the knowledgeable ARPANET workers who have contributed to this report. Both are capable of being brought to fruition, though, and in a reasonable amount of time. We will assume ELF as the actual front-end system here for two reasons: apparent consensus, and current activity level of the development team. However, we have no reason to believe that readers who prefer ANTS would encounter substantive difficulties in implementing our proposal on it.)

(Explanatory notes: ANTS is an acronym for ARPA Network Terminal Support system; it was developed at the Center for Advanced Computation (CAC), University of Illinois. ELF is not an acronym (It is said to be German for "eleven"); it was designed at the Speech Communications Research Lab (SCRL), Santa Barbara, California.)

THE RIGID FRONT-END ALTERNATIVE

Referring back to the model above, the popular view of the asterisks is to have the front-end system simulate a well known device for each Host (typically a remote job entry station along the lines of the 200UT on the CDC 6600), effectively requiring no software changes on the Host system. We characterize this approach as "rigid" because an immediate implication is that the Host system is constrained to handle data to and from the network only in fashions which its system already provides. (E.g., if you simulate a card reader, your data will necessarily be treated as batch input if a terminal, necessarily as time-sharing input.) Now, it may be argued that Host software changes are only being shunned in order to "get on the air" quickly, and may be introduced at a later date in order to allow unconstrained channelling of network data within the Host; but this reasoning may surely be refuted if it can be shown that an alternative exists which is essentially as quick to implement and does not require the waste motion of constructing known-device simulation hardware and software for each new Host, only to eventually avoid the simulation in the Host.

The major advantage which might be claimed for the rigid front-end approach other than quickness to implement would be embarrassing if true. That is, the possibility exists that either the "new" Host's operating systems or system programming staffs are so intractable that avoiding Host software changes is a necessity rather than a desire. We certainly hope neither is the case and have no reason to believe it to be so, but we must acknowledge that such possibilities exist as meta-issues to this report.

DISAVANTAGES OF THE RIGID FRONT-END ALTERNATIVE

The rigidity argument sketched above merits some amplification. The major disadvantage of interfacing with the Host only in fixed ways lies in a loss of functionality. Granted that "Telnet" and "RJE" functions can be performed (though we have deep reservations about file transfer) by simulating a known device there are more things in practice and in theory than just using the Hosts' time-sharing and batch monitors. "Teleconferencing" is an instance which comes immediately to mind. Graphics is another. Neither fits naturally into the setting a typical operating system is likely to assume for a Telnet or RJE connection. Further, the ARPANET is just beginning to evolve a view of "process-to-process" protocols where cooperating programs on dissimilar systems communicate over network sockets in a true use of sockets as interprocess communication media. It is difficult to conceive of viewing a (simulated) line printer as an abstract "port" without considerable contortion of the extant operating system. To attempt to summarize this cluster of objections, a simulation of a known device may be cheaper than a large enough number of phone calls, but it's not networking.

For that matter, it is by no means clear that the goal of Host software changes can even met. In the case of one particular system on the ARPANET where a PDP-15 was employed as a front end to a PDP-10, one of the authors discovered that on attempting to login over the net he was confronted by an interrogation as to the type of terminal he was at -- the front end having been attached at the wrong point in the PDP-10's terminal handling code. (Being a battle-scarred veteran of Telnet protocol development, he gave suitable answers for describing a "Network Virtual Terminal". Unfortunately, however, the NVT apparently had no counterpart in the Hosts' normal complement of local terminals. And when he tried such Telnet control functions as "don't echo, I'm at a physically half-duplex terminal" things really got confused). As it happens, he later found himself in the neighbrohood of the Host in question, and found himself spending an afternoon attempting to explain the philosophy and importance to the Telnet protocol of the NVT. The site personnel were both appreciative and cooperative, and although we have not had occasion to verify it, we assume that the site is probably now usable from the ARPANET. The important point, though, is that operating systems tend to make extensive, very often unconscious, assumptions about their operating environments. This observation is particularly true when it comes to terminal types, and the problem is that there is simply no guarantee that the several systems in question could even "do the right thing" if they were front-ended by simulating a known device -- unless, of course, the simulation of the device in the mini were so painstaking that all we'd get would be an expensive way of adding an RJE station, period.

Less abstract considerations also apply. For one thing, a mini-computer -- even with "third-generation" software -- is not as free and convenient an environment to program in as a full-scale Host therefore, implementing the several simulations will not be trivial pieces of software engineering. Further, if the simulation software is prepared by front-end experts, they will encounter repeated start-up transients in learning enough about the expectations of the several Host in order to perform their tasks. For that matter, it is clear that if personnel from the several Host are barred from active participation in attaching to the network there will be natural (and understandable) grounds for resentment of the "intrusion" the network will appear to be; systems programmers also have territorial emotions, it may safely be assumed.

On a still more practical level, it should be noted that the potential need to simulate more than one known device -- and even the potential complexity of any single device simulation -- may well lead to a requirement for a larger PDP-11 configuration than would otherwise be reasonable. And although there are other reasons for arguing that each front-end processor ought to be as big a configuration as possible, we must acknowledge that dollars do matter. Also on the topic of numbers, it should be further noted that the line speed available for known-device simulations can be quite low. The 200UT, for example, is on a 4800 baud line, which is rather a mismatch with a 50,000 baud communication subnet. (Of course, there's always the 40,800 baud line into the 6600 -- but it is'nt expected to have interactive devices on it, so the extant software won't send the data to the "right place"....) And no experienced ARPANET protocol designer would be willing to overlook the possibility that there will probably have to be a flow control discipline between the Host and the front-end processor anyway, so the no change to Host software goal becomes rather dubious of fulfillment.

After all that, it is perhaps gratuitously cruel to point out still another level of difficulty, but we feel quite strongly that it should be addressed. For, it must be admitted, the question must be asked as to who will do the front-end implementations. This sort of thing is scarcely within the purview of CAC of SCRL. But, as will be urged in Appendix 2, it is of the outmost importance that whoever performs the task already have ARPANET expertise, for we know of no case where "outsiders" have successfully come aboard without having become "insiders" in the process, which is neither an easy nor cost effective way to proceed.

In light of the above, it is at least reasonable to consider an alternative to the rigid front-end approach, for regardless of the weight the reader may attach to any particular cited disadvantage, in total they at least suggest that the known-device simulation tactic is not a panacea.

THE FLEXIBLE FRONT-END ALTERNATIVE

Our alternative approach is based on a principle which actually has been around since at least a month before the ARPANET began running User and Server Telnets on a regular basis. The principle is that it would be nice to off-load as much as possible of the NCP from the Host, because Hosts are supposed to have better things to do with their cpu cycles than field control messages from other Hosts -- especially when 90% of the control messages are merely ALL(ocate) commands. This insight led to the notion that all a Host "really" has to do is associate sockets with processes (and, of course, pass data along socket connections). And the flexible front-end approach is no more than an updating of these 1971 ideas to the following: Drop the hard and fast goal that there will be NO changes to Host software in favor of the more realistic goal of making MINIMAL changes to the Host attach the front-end processor to any convenient high-speed "channel" (/ "port" / "multiplexer" / "line" / "cable"); let the fro nt-end processor handle the NCP; define an extremely compact protocol for the Host and front-end to follow (the H-FP); and let the Host H-FP module distribute the data appropriately within its operating system, because the H-FP will make it clear where the data should go and if you have to ram the data into the teletype buffers, it's still cleaner than trying to do interprocess communication over a card reader. (The H-FP is detailed in less bald terms in Appendix 1). Now that might sound rather uncompromising -- and almost surely sounds rather cryptic -- but between the advantages it engenders and the more comprehensive description which follows, we feel that it does represent a superior basis for solving the overriding problem of how best to attach "new" Hosts to an ARP-like net.

ADVANTAGES OF THE FLEXIBLE FRONT-END ALTERNATIVE

The primary advantage of the flexible front-end alternative is precisely its flexibility: Although minimal implementations may be envisioned on given Hosts, the most minimal of the implementations is still as powerful as the rigid front-end approach; and as the need for more functions is perceived, they may be coded for quite easily with our approach. This is so because programs in the Host can "get their hands on" data from the net (and send data to the net) in a natural fashion -- it is not the case that only those things done on a given system with the data from, say, a card reader, can conveniently be done here. Thus, in contrast to the rigid front-end approach, the flexible front-end approach "is networking". Indeed, it should be noted that a major "real" ARPANET server site has expressed an interest in implementing the H-FP based on some five minutes' worth of the blackboard explanation with two of the authors.

Another advantage of our new approach is that it involves personnel at the various new sites in the process of coming aboard the network. Not only does this involvement have merit psychologically (if known-device simulation were employed, the network could represent an alien intrusion forced upon them, to site systems types), but it is also technically preferable to have per-site coding done by "experts", which would not be the case if the per-site tailoring were done exclusively in the mini. Recall the PDP-15 to PDP-10 attempt discussed earlier. That case may fairly be viewed as one of the front-ending's having been performed in ignorance of the conventions of both the Host's operating system and of the ARPANET? Not only should that sort of thing be avoided by the expedient of involving experts on the target operating systems in the process of attaching to the network but there are practical considerations as well: we estimate that adding a minimal Host-Front End Protocol routine in a given operating system would require no longer than the same few man months to develop than would the adding of a new known-device simulation package to the mini. So that we foresee scheduling advantages in addition to the more abstract ones already asserted. Further, it ought to be a more friendly environment to program in on the Host than in the mini. (This is not to say the ELF does not appear to be good environment to program in; rather, it is to make the "obvious" claim that if the big systems did not furnish convenient programming environments we wouldn't have them.)

As touched on earlier, another point which bears further examination is the area of flow control. The known-device simulation approach appears to assume that this too will be handled by the mini, and that the simulation will be aware of whatever flow control discipline the host and the physical device being simulated follow. However, when the one device "everybody knows" will be simulated (CDC 200UT) operates on a 4800 bit-per-second line, and the IMP subnetwork operates on a 50,000 bps lines, some attention must be paid to the mismatch -- especially in view of the fact that only one process in the Host is typically associated with a known device, but the network actually transmits data on behalf of many processes. Our approach, on the other hand, allows for a very direct, simple flow control discipline to be imposed, without getting involved in per-Host idiosyncrasies. (The option to go more elaborate -- potentially more efficient -- flow control disciplines is also provided.) Thus, we can simply pick the beat line speed available on a particular Host, and attach to it.

Notice one other level of practical advantages: The min's H-FP module can be furnished along with its operating system by the same network "insiders" who are furnishing the operating system itself. Thus, a critical task need not be subjected to the perils of subcontracting. Indeed, this approach lends itself far more readily to subcontracting than other, if subcontracting must be done for the per-cost software for with the PDP-11 being almost always the same, network "insiders" can be used in conjunction with site personnel to build Host H-FP modules either through commercial consulting contracts or even from within the ARPANET community. (The latter possibilities exists because another fact about system programmers is that -- although they resent "invasions" -- they tend to enjoy getting inside new and different systems, if only to feel superior to them in contrast with their own.)

The strengths of the flexible front-end approach, then, tend to arise in exactly those areas of weakness of the rigid front-end approach. Perhaps most important of all, though, is the fact that it "makes sense" to almost every single experienced member of the ARPANET community with whom it has been discussed. So, we might reason, if the ARPANET is desirable, it is desirable because efforts of those who made it work and if they have gained insights into networking in general in the process, their opinions deserve particular attention.

RECOMMENDATIONS

The protocol specified in Appendix 1 is felt to be around 90% complete. We are aware that we have not specified all the codes that will be needed to describe conditions of which the Host and Front-End must apprise each other, for example. But we think that, in general the protocol "Woks". We stand willing to discuss it with cognizant decision makers in the various interested organizations, and, for that matter, to continue to debate it with our technical peers. At this stage, however, the dominant makers avert the apparent stampede to the rigid front-end approach and evaluate the flexible front-end alternative in light of the preceding arguments and the following protocol specification.

APPENDIX 1. THE HOST-FRONT END PROTOCOL

ASSUMPTIONS

The physical connection of the front end (FE) to the Host is assumed to be made over the "best" port (or channel, line, etc.) available on the Host, where "best" covers both line speed and quality of software available to physically manage the line. The choice should be made by site personnel. Hardware interfacing capability is assumed to be straightforward; it is, at least, no more complex for the H-FP than for known-device simulation. The connection is assumed to be sufficiently closely coupled that a simple, explicit acknowledgment H-FP command will offer satisfactory flow control. That is, distances are assumed to be short and bit rates high; thus, the same assumptions are made here as are made in the case of Local IMP-Host interfaces: that error checking and flow control are not first-order problems.

On the software level, buffering is assumed to be adequate in the Host to accept at least a full (8096 bit) IMP-IMP message-- although the FE could probably get around this constraint if it absolutely had to. Given only a minimal H-FP module in the Host, the FE will allow the same level of Telnet and RJE functioning as would the known-device simulation, as follows: The FE will always shield the Host from the NCP commands and the simplex sockets they deal with, dealing instead with a repertoire of but five H-FP commands and conversing over duplex data streams with the appropriate management of Network sockets left to the FE. (The commands are described below; we continue with the discussion of assumptions here, but some readers may prefer to study the commands before continuing with the balance of this section.) For Telnet, although subsequent analysis may lead to a more sophisticated treatment, the present assumption is that the FE will normally refuse all "negotiated options" and strip all Telnet control codes from the data it passes to the Host (unless the Host orders it to pass an unaltered Telnet stream); on a pre-installation basis, the FE will also map from Telnet ASCII to the Host's desired character set. Telnet "interrupt process" controls are handled by an H-FP command, discussed below.

For RJE, because the ARPANET RJE Protocol is only known to have been implemented on one Host in the ARPANET and is generally considered to be too cumbersome, the standard socket for RJE will be reserved for future use, and a special designator will indicate to the Host that input on the given connection is to be treated as data in the format and job control language of its own "batch" system. Again, character set mapping will be available on a per-installation basis.

For file transfer, however, a further assumption must be made about Host software. This is because the FE cannot be expected to manipulate the Host's file system; therefore, if the host whishes to participate in file transfer activities its H-FP module must be able to access the Host's file system for both sending and receiving files. Again, the FE will be able to shield the Host from the details of the underlying protocols to a large extent; but the Host must be able to handle FTP "stor" and "retr" commands, which will be passed over the (single) connection opened between the FE and the Host for file transfer. (FTP "user" and "pass" commands might also be desirable. As with Telnet, the FE will manage the Various Network sockets involved so as to allow the Host to operate on only the H-FP connection, and will again optionally perform character set mapping. Note that Hosts may refuse to open FTP connections until and unless they choose to, with no impact on the FE.

The Host's H-FP module, in short, will interpret the commands of the protocol, distribute Telnet data to and from the appropriate points within its operating system where terminal I/O is expected, distribute RJE data like manner, and when it is able to do so handle FTP as sketched above and amplified on below. It will, also on a when-desired basis, support calls from its system's user processes for unspecified purposes I/O on ARPANET sockets to allow for such functions as teleconferencing and other process exploitations of the Net. Our overriding assumption is that the initial H-FP module for a given Host (which does not require FTP or unspecified socket capability) will not be appreciably harder to implement than a known-device simulation; that it will offer extensibility to more interesting uses of the network than the alternative has been sketched here and will be returned to after the H-FP commands are described.

FORMAT OF THE COMMANDS

All communication between FE and Host is performed in terms of H-FP commands. The fields of the several commands are one or more "bytes", where a byte is per-installation parameter of 8, 9, 12, 16, 18, 32, 36, 48, 60 or 64 bits width, according to the coding convenience of the given Host's H-FP module implementers? (6 bit bytes are not supported because they do not offer enough room to express all the values anticipated for certain code fields machines with 6 bit internal byte structure can specify 12 bit H-FP bytes and still be able to use their natural byte oriented instructions.) Values for fields will be right-justified within their (potentially several) byte widths. Note that the list of byte sizes is 1) not meant to be exhaustive, and 2) probably unnecessarily extensive -- as 8,9, and 12 are probably the only "reasonable" sizes in actual practice (but if a particular machine is better suited for handling whole words rather than fractions thereof, the FE can certainly make life more convenient for it.)

Although the commands are given names for documentation purposes, the value transmitted in the first byte of each command will be the binary representation of the number shown before its name in the next section. (i,e., the command field is one byte wide.)

COMMANDS

(Note that all commands may be sent by either the FE or the Host.)

1. BEGIN INDEX HOST SOCKET TRANSLATION-TYPE CONNECTION-TYPE

The begin command establishes a "connection" between the Host and the FE. Regardless of internal representation, the duplex data stream the connection represents will be referred to by the value specified in the next (INDEX) field that is, for example, the FE will send input from and receive output for a given Telnet connection "on" a given INDEX, even though it is actually managing two "sockets" for the purpose in its dealings with the Network.

a) INDEX is a two-byte field. Both the Host and the FE may choose arbitrary values for it when opening connection with a BEGIN command (H-FP implementations will probably simply increment INDEX by 1 whenever they need a new connection); however, the value of 0 is reserved to apply to the "global" connection between the Host and the FE -- thus, when either machine "come up" the first thing it does is send a BEGIN for INDEX=0. (The END and ACKNOWLEDGE commands also follow this convention; for that matter, there is no reason why the MESSAGE command could not also, should it be desired to extend the FE's functions in the future. At present, however, this is merely a potential extension.) Note that all other fields should be set to 0 for INDEX 0 BEGINS.

b) HOST is a two-byte field. It specifies the Host number associated with the socket in the next field. On FE to Host BEGINS this is purely informational. However, on Host to FE BEGINS it is necessary to enable the FE to identify the foreign Host with which to communicate at the NCP level.

c) SOCKET is a four-byte field. If SOCKET=1, a Telnet connection is to be established. If SOCKET=3, an FTP connection is to be established. If SOCKET=5, an ARPANET RJE Protocol connection is to be established (no known current utility). If SOCKET=77, a Host-specific connection is to be established for RJE/batch. All other values are for connections for unspecified purposes, to be opened at the NCP level according to the CONNECTION-TYPE field. Note that sockets 1, 3, 5 and 77 are "known about" and special-cased by the FE.

d) TRANSLATION-TYPE is a one-byte field. From FE Host, it is informational. From Host to FE, it specifies character set mapping if desired, or characterizes the data to be transmitted over the connection. 0 request / specifies ASCII data 1; binary data (note that this value will not be sent from FE to Host under current assumptions, and that word size is to be a per-installation parameter); 2, mapping of ASCII to/from local character set. Other types will be defined if needs are identified.

e) CONNECTION-TYPE is a one-byte field. For FE to Host BEGINS it is informational. For Host to FE BEGINS it instructs the FE as to which kind of NCP connection discipline to follow. 1 requests a duplex connection (i.e., that the Initial Connection Protocol of the ARPANET be employed) 2, a simplex connection (i.e., that the appropriate ARPANET "request for connection" Host-Host Protocol commmand be employed for the gender of the socket at hand). Note that this extended use of the H-FP will be of interest when (and if) User-level programs on the Host begin to use the Network. (The FE will open 8-bit connections at the Network level unless otherwise directed.)

2. ACKNOLEDGE INDEX CODE

The ACKNOWLDEGE command is multi-purpose. It must be sent in response to all commands from the other machine (other than ACKNOWLEDGES, of course), and is primarily used to indicate the success or failure of the command just received on INDEX. Note that this implies that each MESSAGE on a given INDEX must be ACKNOWLEDGEd before the next can be sent.

a) INDEX is as above.

b) CODE is a two-byte field. CODE=0 indicates success / acceptance of the command most recently received for INDEX. CODE=1 indicates failure /rejection of the most recent command. (E.g., if a MESSAGE, buffering was unavailable so the other machine must retransmit; if a BEGIN, the indicated protocol / socket cannot be serviced.) CODE=3 indicates an invalid or inactive INDEX has been used. CODE=4 indicates (HOST to FE) that no mapping is to be performed on the connection just opened. Other values (for such meanings as "foreign Host down", "undefined type requested" and the like) will be assigned as identified.

3. MESSAGE INDEX COUNT PAD TEXT

The MESSAGE command is employed for the transmission of data.

a) INDEX is as above.

b) COUNT is a two-byte field which specifies the number of bits of data in the TEXT field.

c) PAD is a 1-to-n-byte field. Its width is a per-installation parameter used to enable the TEXT field to start on a word boundary if the local H-FP implementers so desire. (This is not only a kindness, but it's also a placeholder if we decide to go to a flow control mechanism involving sequence numbers.)

d) TEXT is a field wherein byte structure is coincidental. It consists of COUNT bits of data to be sent to the process implicitly associated with INDEX by a BEGIN command (which has not been ENDed.)

4. INTERRUPT INDEX

The INTERRUPT command, when sent from the FE to the Host, indicates that an FCP interrupt command (INS or INR) has been received for the process associated with INDEX; the Host should interrupt the associated process and whatever fashion is "normal" to it. (The most common use of the NCP is in Telnet, where it is defined as being the functional equivalent of having struck a terminal's ATTN, INT, of BREAK key, or input a "control-c" on certain character-at-a-time systems; essentially, it requests a "quit button" push. Note that the FE will take care of the associated Telnet control code in the input stream.) When sent from the Host to the FE (in process to process applications), it will indicate that an appropriate NCP interrupt be sent, according to the gender of the socket associated with INDEX.

5. END INDEX CODE

The END command is used to terminate a connection. It may be sent either because one system or the other is about to go down, or because the FE have received an NCP "CLS" command or because the destination system or IMP has gone down, or at the behest of a Host user process.

a) INDEX is as above. Note that if INDEX=0 the END refer to the "global" connection between the Host and the FE in such case, the high-order bit of CODE will be set to 1 and the low-order bits will specify the number of the minutes to shutdown if this information is available. (Furnished because the associated IMP often informs the FE of such a condition.)

b) CODE is a two-byte field. CODE=1 indicates the general "close" case (either received or ordered) 2, foreign systems has gone down; 3, foreign IMP has gone down; 4, local IMP has gone down. Other values will be assigned as identified.

EXTENSIBILITY

Simplicity and compactness being major goals of the protocol, the small repertoire of commands just presented represent "all there is". Recall that we are specifically omitting from consideration such issues as error and flow control, which could turn the H-FP into another Host-Host Protocol. (should error and flow control prove desirable in practice, we have, of course, thought of some suitable mechanism within the H-FP framework; but they are not considered germane in the present context.) The primary intention here is to specify a protocol, which lends itself to minimal initial implementations in the Hosts, on the same time scale as would have otherwise been required for known-device simulations -- but which offers great flexibility in the use of the network than would be achieved through known-device simulation.

The astute reader will have noticed that most of the commands have been specified with an eye toward the future. Because the same protocol, which allows the Host and the FE to communicate can easily allow user processes on the Host to use the Network, we have tried to encourage this desirable end by furnishing all the necessary hoods and handholds for it in the FE's H-FP module through the broad definitions of the commands. A Hosts's H-FP module can furnish a trivial interface for user programs in terms of a very few entry points (open, read, write, and close appear to be the minimal set) and allow the user program considerable flexibility in its use of the net. For example, a "User" FTP program could be straightforwardly created even for a Host, which did not choose to field the BEGINs on socket 3 (necessary for "Server" FTP capability), and files could still be "pulled" to the Host even if they could not be "pushed" to it. (the FE will be required to recognize and special-case BEGINs on socket 3, but that's a small price to pay). So, if the specification of the h-FP command repertoire seems somewhat more complex than it need be, remember that not all of it has to coped with on any given Host -- and that any give host ca take advantage of more functions as it desires. (Although it's not really within the present scope, we stand willing to invent per-Host H-FP to user program interfaces on request.)

FTP

To amplify a bit on the problem of file transfer, it must be observed that in general only a file system can manage its files. This borders on tautology and is difficult to deny. Therefore, although the FE can shield the Host from a great deal of the mechanism included in the FTP for functions not directly germane to the transferring of files, Host's operating system and place or extract a given file, even though it "has" the file's name available to it. There is no in-principle reason why the H-FP module on the Host can't invoke an appropriate routine when it receives a BEGIN on socket 3, though. (The FE will handle all the type and mode negotiations, pass the "stor" or "retr" line along, and be ready to transmit or receive on the appropriate socket but "somebody" in the Host has to receive or transmit the MESSAGE to or from the right place.) But if that seems hard to do on any particular Host, its H-FP module can merely negatively ACKNOWLEDGE any BEGINs for socket 3. The real point to be noted is that the H-FP still allows in principle for User FTP, as explained above, even so -- and that the simulation of known device offers neither (User nor Server FTP) function.

(Files could, of course, be transferred into the FE, then somehow gotten into the Host "later" -- perhaps by faxing up a batch job -- but that route requires either an awful lot of buffering in the mini or a very sophisticated file system there, or both. It also requires an awful lot of per-Host information in each FE -- or perhaps human intervention. We're not saying it can't be done... eventually. But it's not going to be clean, or quick, or easy, or cheap.)

SUMMATION

Several important themes have unavoidably been dealt with piecemeal in the foreign attempt to specify the H-FP in the abstract. To gather the threads together, it might be useful to consider the various ways in which the protocol can be employed, in the context of their ARPANET counterparts. A. "SERVER" FUNCTIONS: There are, in essence, three levels on which a Host can use the H-FP to fulfill ARPANET "Server" functions. 1) For Hosts which choose to take FULL advantage of the flexibility of the H-FP, all "fourth level" (user process to user process) protocols can be managed by the Host. The FE will perform NCP (Host-Host protocol) and IMP-Host protocol functions (the associated IMP will, of course, perform IMP-IMP protocol functions), thus shielding the Host from the necessity of implementing a full-blown NCP with the attendant complexity of being aware of the 11 to 14 "states" of a socket, flow control, retransmission, and the like (as well as shielding it from the IMP-Host protocol, with the attendant complexity of mapping "links" to/from "sockets", dealing with message types forming and parsing "leaders", and the like). This mode of use is effected by giving the "no mapping" code when the Host acknowledge a BEGIN on socket 1 and 3 (and by simply accepting BEGINs on all other sockets). 2) For Hosts which choose to take PARTIAL advantage of the flexibility of the H-FP, many aspects of the fourth level protocols (in particular Telnet and FTP) can be managed by the FE on the Host's behalf, by virtue of making assumptions about which Telnet and/or FTP "commands" are to be permitted and only referring search matter as the association of data which processes and/or file names to the Host. (Note that the CODE field of the ACKNOWLEDGE command furnishes the mechanism for conveying such error information as "file not found" from the Host to the FE, which in turn will send out appropriate FTP error messages.) This mode of use is effected by simply accepting (with code 0) BEGINs on sockets 1 and/or 3 (and doing as one chooses for all other sockets); that is, fourth level shielding is anticipated to be commonplace, and is the FE's default case. 3) For Hosts which choose to take NO advantage of the flexibility of the H-FP, the "private" RJE/batch connection type will still provide for the desirable functions of load sharing and transferring files even though other fourth level protocols were to be rejected by a given Host (by refusing BEGINs on all sockets other than 77). Even in this most restricted case, the ability to upgrade to either of the broader base is additively implicit in the H-FP, with no changes required to the FE's own H-FP module -- whereas it would entail considerable alteration of the Host's operating system had the first step been a known-device simulation. B. "USER" FUNCTIONS: 1) On the "User" side, a Host could again elect to handle such fourth level protocols as Telnet and FTP itself. However, particularly in the Telnet case, there is no real need for this, as a User Telnet "comes with" the FE and it is unnecessary to burden the Host with such use unless so many of its local terminals are hardwired that it would be expensive to access the FE directly. (Note that for a User FTP, the Host's H-FP module would, as discussed above, in all likelihood require a user program callable interface.) 2) On a less ambitious level, the FE could be induced to perform the same shielding as it offers the Server FTP (cf. case A2, above), given an "FTP mapping" TRANSLATION-TYPE on the BEGIN command or the previously suggested special casting by the FE on socket 3. 3) Finally, "User" functions could be completely finessed, as per case A3.C. PROCESS TO PROCESS FUNCTIONS: Irrespective of the positions taken in A and B, given only a user program callable interface to the Host's H-FP module, all other fourth level protocols which might evolve -- or, simply, general use of sockets as interprocess communication ports -- can be achieved directly. Again, this would fundamentally be an "add-on" to the system, not an alteration of existing software.

APPENDIX 2 - SOME NOTES ON IMPLEMENTERS

INTRODUCTORY DISCLAIMER

This appendix represents strictly the personal views of one of the authors; I (now that I can admit to being Mike Padlipsky) have not even permitted the other authors to agree with the views expressed here, much less disagree with them, for they are insights which I've gained the hard way during nearly four years of involvement with the ARPANET and I feel they need saying -- regardless of the polite fiction of refraining from finger pointing. Please note at the outset, however, that I am motivated not by a sense of vindictiveness -- nor even of righteous indignation -- but rather by a desire to present some history in the hope that the reader will not be condemned to repeat it. Note also that even though it makes the prose more convoluted than it might otherwise have been, the convention will be observed of "naming no names". I am not, I repeat, out to get these guys; merely to get away from them and their like in the future. (The reader can stop here with no loss to the main argument of the paper.)

SEVERAL HORROR STORIES FROM THE WONDERFUL WORLD OF NETWORKING

Consider first the tale already told of the PDP 15/PDP 10 front ending effort. Having been involved in the writing of both the "old" (1971) and the "new" (1973) Telnet Protocols, I feel a certain sense of shame by the association that they were not so compellingly clear that the power of the Network Virtual Terminal / common intermediate representation approach could not have been missed, ever by system programmers operating in pretty much of a vacuum with respect to contact with knowledgeable ARPANET workers. Having said that -- and meant it -- I still feel we did a good enough job for average-plus system types to cope with. (The fact that numerous Hosts are on the Net is evidence of this.) Unfortunately, however, average-minus system types do exist and must also be contended with. Therefore, if we do not make a concerted effort to "idiot proof" our protocols, we may anticipate further repetitions of the sad state the site under discussion found itself in before it happened upon them. (And, it must regretfully be observed, support of the "real" ARPANET has deteriorated to the point that the massive effort required to over-explain ourselves probably could not be launched in the prevailing climate. More on this point later.)

Case in point number two is potentially far graver than a mere "philosophical" muddle over bringing one site aboard. It involves an attempt by one of the Armed Services to network a large number of large machines using the ARPANET as a model. The implementation of the software house with no known ARPANET expertise. The communications subnet and the hardware interfacing to the Hosts was subcontracted to a well-known hardware manufacturer with no known ARPANET expertise. (As an aside, but because it's so startling I can't forbear, the "system architect" for the target network is still another well-known hardware manucfacturer (!), with, of course, no known ARPANET expertise.) To make a long, continuing story short, it is currently the case that the "real" ARPANET system whose hardware corresponds most closely to the machines being netted here (even though it is benchmarked at a rather lower "mips" (million instructions per second) rate than the target net's machines) can transfer files at rates in excess of 28,000 bits per second (following the rather cumbersome full ARPANET FTP) from a small configuration developement machine to a lightly loaded (but still running in excess of 20 users) service machine one Network "hop" away, while the new system achieves rates which I am apparently not permitted to quantify but are very considerably lower even though only one process is being run on each machine -- also one "hop" away

-- and the protocol for file transfer is nowhere near so general as in the ARPANET. Given a year or two, the situation can presumably be rectified, but at present it is fair -- if somewhat fanciful -- to say that if the Japanese were capable of only like level of technology transfer they'd still be trying to make up their balance of trade with those cute little parasols on matchsticks.

Yet what has gone amiss here in Horror Story 2? I submit that the choice of subcontractors was based upon a misapprehension of the level of technological sophistication associated with the ARPANET, and that what was (is?) needed is a subcontract to a knowledgeable ARPANET source (and I don't mean to the usual, profit-marking place -- though I guess I trust them for the subnet), rather than to "outsiders". (I don't even mean to any particular place on the Net; maybe what's needed is to form a meta-place out of the whole Net. More on this, too, later.) The real point is that the model was essentially ignored by the putative model-followers, and -- demonstrably -- it shouldn't have been.

Case three should go a long way toward dispelling any impressions that might be building in the reader's mind that I'm some sort of hardcore ARPANET chauvinist. For even "insiders" have blown some. This is actually a dual case, for it involves two unsuccessful attempts to furnish terminal support mini-Hosts for the Net. In one case, the choice of machine was faulty; even with additional core memory field retrofitted, buffers cannot be furnished to support reasonable data rates without imposing considerable unnecessary Host overhead in the processing of too frequent Host-Host Allocation commands. Nor is there enough room to furnish more than a rudimentary command language in the mini. Now these were knowledgeable, reasonably well managed "insiders" -- but they were contractually not in a position to heed the technical intuitions of several of themselves and the technical intuitions of many of their colleagues throughout the Network Working Group that they'd been painted into a corner.

In the second sub-case, the hardware and contractual obligations appear to have been right, but ill-considered choice of implementation language and inadequate management have prevented the project's full completion to this time (some two years after its inception). Again, there was forewarnings from the NWG, in that we had tried to alert them quite early about the language issue. (On the management level, we could only sympathize -- and in some cases empathize -- but it is at least a tentacle position to take that the ARPANET as a whole happened despite, not because of, management.) (I guess I am an ace system programmer chauvinist.)

The final case to be cited here involves another military effort.

This one I'm not even sure I'm supposed to know about, much less talk about. But I can say that it involves a subcontractor's attempt to attach several special purpose machines to a major ARPANET server by means of an internally invented set of machines and protocols. My information suggests that when asked why they failed to follow the apparently obvious course of using ARPANET technology (facilities for which do, of course, already exist on the target server), the subcontractors essentially replied that they hadn't felt like it. They also made their approach work yet, and it's been something like a couple of years they've been trying.

Then three's the fad to simulate RJE terminals... but to use that as Horror Story 5 would be begging the question -- for now.

SOME MORALS

Rather than search out any more dirty linen, let's pause and look for the lessons to be learned. In the first place, it borders on the obvious that for any technical project the "right" technicians must be found and empowered to perform it. Despite the generation of over-sell on the "power of computers", they still absolutely require intelligent, competent programming -- which in turn requires intelligent, competent programmers. And, at the risk of gilding the ragweed, not all self-professed programmers are intelligent and/or competent.

In the second, and more interesting, place, all unknowing the ARPANET has attracted or engendered an "in-group" of extremely good system types -- who have learned through some sort of natural selection process to work well together despite the immense handicap of the heterogeneity of our various "nome" systems' assumptions. We not only have developed a common tongue, but some of us even like each other. (It should be noted that Appendix 1 was specified on a Wednesday afternoon and a little bit of a Thursday morning. Jon and Jim and I had been there before.) It seems quite clear to me that the organizations for whom this report is intended should avail themselves of the expertise which exists in the NWG; we've got a reasonable track record, after all, especially in comparison to others who have attempting networking. Many of us also feel quite strongly that we didn't get a chance to finish the job on the ARPANET, and would like to be given the chance to "do it right" -- especially in view of the errors which have been committed in our name. (This is particularly important because the old gang is beginning to scatter. For myself, I expect this will be my last RFC. Well, at least I've tried to make the most of it.) The ARPANET is no more a finished product than ANTS or ELF -- but all of them could and should be.

In the final place now, a rather trite moral must be drawn: Technical competence is extremely difficult to assess a priori. (I'm inordinately fond of saying "Don't ask me what I'm going to say, I haven't said it yet" myself.) But "track records" ARE important, and competence CAN be demonstrated -- to a suitable jury of technical peers. Therefore, beware of plausible sounding subcontractors who tell you "It's easy". In our field, and particularly in getting all those strange machines which were developed by people who by and large didn't talk to each other to "talk" to each other, it's NOT easy. I'm willing to claim that it will be easier letting some NWG types do it with the H-FP approach, but it might never be really easy -- where "never" means for the next 10 years or so, until "real" networking comes off the shelf with the operating system (which itself scarcely comes off the shelf today) -- but don't get me started on The Manufacturers.

BEYOND THE PAIN PRINCIPLE

So it's not easy. It's also not impossible. Indeed, the time appears to be ripe right now avoiding generating a whole new generation of horror stories, by sensitizing decision makers to technical realities and "doing things right" this time around. Having seized this occasion to say some things to that end which I think are important, I must in good conscience stand ready to defend the assertions I've made of error in some camps and of correctness in what I might loosely call "our" camp. I do so stand, with a right good will. If any reader desires more corroborative detail -- or merely to see if I rant like this in contexts other than RFCs (or even to have a go at my explanation of the common intermediate representation principle), well, I'm still in the ARPANET Directory -- even though the phone number's different (try 703-790-6375). The mailbox remains accurate (even though there is no "ARPANET mail protocol" it's marvelous how stopgaps endure).

[This RFC was put into machine readable form for entry]
[into the online RFC by Helene Morin, Viagenie,12/1999]