< POSTEL, TELNET-COVER,NLS;7, >, 23-OCT-74 12:56 JBP ;;;; 1

Request for Comments: 659 Jon Postel
NIC# 31177 SRI-ARC
Online file: [SRI-ARC]<POSTEL>RFC659,TXT 18 October 1974

Announcing Additional Telnet Options 1a

This is to announce the set of Telnet Options defined in Requests For Comments 651 through 658 (NIC 31154 through 31161) are part of the offical Telnet Options and should be added to the protocol Notebook. 1b

RFC 651 is a revision of the Status Option to utilize the subcommand feature and reduce the number of bits required to convey the status information. This revised Status Option replaces the previous Status Option. 1c

The others RFCs (652 through 658) are new Options for controling the behavior of the format effector characters Carriage Return, Horizontal Tab, Form Feed, Vertical Tab, and Line Feed. 1d

Your attention is also called to RFC 645 (NIC 30899) "Network Standard Data Specification Syntax" which was prepared some time ago but has not received wide circulation. 1e

Unfortunately the Network Information Center can not provide hardcopy documents. Currently the author is responsible for distribution of RFCs. 1f

The documents are available online as: 1g

[ISI]<DCROCKER>STATUS-OPTION-REVISION,RNO 1g1
[ISI]<DCROCKER>NAOCRD,RNO 1g2
[ISI]<DCROCKER>NAOHTS,RNO 1g3
[ISI]<DCROCKER>NAOHTD,RNO 1g4
[ISI]<DCROCKER>NAOFFD,RNO 1g5
[ISI]<DCROCKER>NAOVTS,RNO 1g6
[ISI]<DCROCKER>NAOVTD,RNO 1g7
[ISI]<DCROCKER>NAOLFD,RNO 1g8
[SRI-ARC]<POSTEL>RFC645,TXT 1g9
[SRI-ARC]<POSTEL>RFC659,TXT (This Announcement) 1g10

(If these files are not found online they may have been archived, in this case the files should be accessable via the "interogate" command.) 1g10a

telnet-cover-letter

(J24296) 24-OCT-74 12:46;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

tip access letter

< POSTEL, TIP-LETTER.NLS;2, >, 21-OCT-74 14:31 JBP ;;;; 1

Dick: The following is the information to be conveyed to RML to get me set up for tip access, as i understand it this has to be sent via us mail preferably on letterhead, the tip access controls are scheduled to be come effective 1dec74. 1a

24 OCT 74 1b

Charles Pearce
Chief, Data Systems Branch/ELD
Range Measurement Laboratory
Patrick Airforce Base
Florida 32925 1c

In response to the "Very Important News" about TIP Access i am supplying the following information: 1d

Name Identifier: POSTEL 1d1

Full Name: Jonathan B, Postel 1d2

password Requested: JBP 1d3

Address: 1d4

Stanford Research Institute
Augmentation Research Center
Menlo Park, California 94025 1d4a

Telephone: (415) 326-6200 x3718 1d5

Network Address: POSTEL@ISI 1d6

Tips to be accessed: ALL 1d7

Organization Name:SRI-ARC 1d8

Authorizing Individual: Dick Watson 1d9

Address: 1d10

Stanford Research Institute
Augmentation Research Center
Menlo Park, California 94025 1d10a

Telephone: (415) 326-6200 x3718 1d11

tip access letter

(J24297) 24-OCT-74 12:51;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /RWW([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

GLOSSARY

< POSTEL, GLOSSARY.NLS;3, >, 26-SEP-74 16:00 JBP ;;;; ; ;

A.1 Glossary

Abbreviations

AEN
another eightbit number

ALL
A host to host protocol command to allocate buffer space to the sending NCP in the receiving NCP.

ANTS
ARPA Network Terminal System

ARPA
Advanced Research Projects Agency of the Department of Defense

ARPANET
Advanced Research Projects Agency Computer Network

ASCII
American Standard Code for Information Interchange. The character encoding used in the network.

BBN
Bolt, Beranek, and Newman, Inc. Cambridge, Massachusetts.

BKY
The operating system used at Lawrence Berkeley Laboratories for the CDC 6600 computer.

CCBS
Center for Computer-based Behavorial Studies at University of California, Los Angeles.

CDC
Control Data Corporation

CLS
A host to host protocol command to close the connection.

DEC
Digital Equipment Corporation

DMS
Dynamic Modeling System, A host computer on the ARPANET at MIT.

EBCDIC
Extended Binary Coded Decimal Interchange Code. The character encoding used primarily by IBM computer systems.

FCP
File Control Program

FTP
File Transfer Protocol

IBM
International Business Machines

ICP
Initial Connection Protocol

IPC
interprocess communication

IMP
Interface Message Processor

LBL
Lawrence Berkeley Laboratory

MCP
The operating system for the Burroughs 6700.

MIT
Massachusetts Institute of Technology

Multics
Multiplexed Information and Computing Service, the operating system for the Honeywell 6180 computer designed and implemented at MIT's project MAC.

NCC
Network Control Center at BBN.

NCP
Network Control Program

NIC
Network Information Center at the Augumentation Research Center of Stanford Research Institute, Menlo Park, California.

OS/MVT
An IBM operating system for the 360 series of computers.

PDP
Programmed Digital Processor

RAND
The RAND Corporation

RFNM
Ready For Next Message

RFC
request for connection

RTS
Receiver to Sender request for connection. A host to host protocol command.

SDC
System Development Corporation

STR
Sender to Receiver request for connection. A host to host protocol command.

TCP
Terminal Control Program

TENEX

The operating system designed and implemented by BBN for the DEC PDP 10 computer.

TIP
Terminal Interface Processor

UCLA
University of California, Los Angeles

UCSB
University of California, Santa Barbara

UCSD
University of California, San Diego

UI
University of Illinois

VM
An IBM operating system for the 370 series of computers.

Terms

another eightbit number
The user program specified portion of the socket number.

ARPA Network Terminal System
A particular small host system designed to interface a wide variety of terminals and peripherals to the ARPA network. This system was designed and implemented by the Center for Advanced Computation at the University of Illinois. The system operates on a DEC PDP11 computer.

connection
The form of interprocess communication provided to the user level processes by the NCPs in the host computers. A connection is a logical simplex stream of data from one port of one process to another port of another process in the network.

control message
A message (of the regular type) that contains host to host commands.

File Control Program
That module in the operating system that controls the access to files by the user processes.

File Transfer Protocol
The protocol that specifies the communication interaction required to move blocks of data (files) between host computers in the network.

full duplex

A channel in which data can flow in both directions simultaneously.

half duplex
A channel in which data can flow in both directions, but may only flow in one direction at a time.

header
The control information at the begining of a packet.

host
A computer attached to an IMP. A host does not necessarily offer services to other computers in the network.

Initial Connection Protocol
The sequence of actions taken by user level programs to establish a pair of connections between a user program and a service program.

Interface Message Processor
The packet routing computers which are the nodes of the ARPA network. An IMP is connected to between 1 and 5 other IMPs and to between 0 and 4 hosts. No more than 7 total IMPs and hosts can be connected.

interprocess communication
The facility for one process to communicate with another process.

leader
The first 32 bits of a message, containing address and control information. The most important fields in the leader are: the message type, the link number, and the host address.

link number
A parameter in the leader that selects a logical communication channel between the source and destination hosts.

message
The unit of transmission between a host and an IMP, up to 8096 bits.

Network Control Program
The program module added to the operating system that interfaces the user processes to the IMP and controls the communication between hosts by implementing the host to host protocol.

packet
The unit of transmission between IMPs, up to 1008 bits.

port
The input or output identifier associated with a particular data stream of a process. For example

a Fortran logical unit number or a data set reference number, or an assembly languague data control block.

prefix
: A 40 bit block immeadiately following the leader and containing the byte size and number of bytes of following text.

process
: A program in execution with its associated address space, registers and location counter.

protocols
: The rules of behavior, in particular, the allowed formats and sequences of communication between two processes.

regular message
: A message from the host to the IMP or from the IMP to the host that is the normal data carrying type. When following the host to host protocol a regular message may carry either a set of control messages or a users data.

request for connection
: Either of the host to host protocol commands STR or RTS.

Ready For Next Message
: A message from the IMP to the host indicating that the previously sent message on the same link number as this RFNM was received by the destination IMP and has begun transmission into the destination host.

socket
: The terminus of a connection. The network wide name of an input or output port associated with a process.

Telnet
: The protocol (or the programs that implement it) that specifies the communication interaction such that a user on one system gains access to the services of a second system as if he were a local user of the second system.

Terminal Interface Processor
: An extension of the IMP to allow a variety of terminals to access the ARPA network. The TIP contains the NCP and User-Telnet programs as well as the terminal handling code in the same processor as the IMP. In addition there is a BBN constructed multi-line controller for up to 63 terminal.

simplex

A channel in which data can flow in one direction only.

Terminal Control Program
: The program module in the operating system that controls the flow of data between the interactive terminals and the user processes.

virtual
: Being something in effect, but not in actuallity. For example a virtual memory might be one that a user process accesses as if it were a large linear core resident set of memory words, when in actuallity the memory is managed by the operating system using paging and maping such that only a small portion of the users set of memory words are in core at any particular time.

GLOSSARY

(J24298) 24-OCT-74 12:57;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([ACTION]) ; sub-Collections: SRI-ARC; Clerk: JBP;

FTPCODES

<POSTEL>FTPCODES.NLS;1, 26-SEP-74 13:23 JBP ; 1

Jon Postel
24 OCT 74

Revised FTP Reply Codes

1a

This document describes a revised set of reply codes for the File Transfer Protocol. 1b

The aim of this revision is to satisfy the goal of using reply codes to enable the command issuing process to easily determine the outcome of each command. The user protocol interpreter should be able to determine the success or failure of a command by examining the first digit of the reply code. 1c

An important change in the sequencing of commands and replies which may not be obvious in the following documents concerns the establishment of the data connection. 1d

> In the previous FTP specifications when an actual transfer command (STOR, RETR, APPE, LIST, NLIST, MLFL) was issued the preliminary reply was sent after the data connection was established. This presented a problem for some user protocol interpreters which had difficulty monitoring two connections asynchronously. 1d1
>
> The current specification is that the preliminary reply to the actual transfer commands indicates that the file can be transferred and either the connection was previously established or an attempt is about to be made to establish the data connection. 1d2

This reply code revision is a modification of the protocol in described in RFC 542, that is to say that the protocol implementation associated with socket number 21 (decimal) is the protocol specified by the combination of RFC 542 and this RFC. 1e

A note of thanks to those who contributed to this work: Ken Pogran, Mark Krilanovich, Wayne Hathway, and especially Nancy Neigus. 1f

Nancy Neigus
Ken Pogran
Jon Postel
24 OCT 74

A New Schema for FTP Reply Codes 1g

Replies to File Transfer protocol commands were devised to ensure the synchronization of requests and actions in the process of file transfer, and to guarantee that the user process always knows the state of the Server. Every command must generate at least one reply, although there may be more than one; in the latter case, the multiple replies must be easily distinguished. In addition, some commands occur in sequential groups, such as USER, PASS and ACCT, or RNFR and RNTO. The replies show the existence of an intermediate state if all preceding commands have been successful. A failure at any point in the sequence necessitates the repetition of the entire sequence from the beginning. 1h

Details of the command-reply sequence will be made explicit in a state diagram. 1h1

An FTP reply consists of a three digit number (transmitted as three alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is intended for the human user. It is intended that the three digits contain enough encoded information that the user-process (the User-PI described in RFC 542) will not need to examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be server-dependent, so there are likely to be varying texts for each reply code. 1i

Formally, a reply is defined to contain the 3-digit code, followed by Space <SP>, followed by one line of text (where some maximum line length has been specified), and terminated by the TELNET end-of-line code. There will be cases, however, where the text is longer than a single line. In these cases the complete text must be bracketed so the User-process knows when it may stop reading the reply (i.e. stop processing input on the TELNET connection) and go do other things. This requires a special format on the first line to indicate that more than one line is coming, and another on the last line to designate it as the last. At least one of these must contain

the appropriate reply code to indicate the state of the transaction. To satisfy all factions it was decided that both the first and last line codes should be the same. 1j

Thus the format for multi-line replies is that the first line will begin with the exact required reply code, followed immediately by a Hyphen, "-" (also known as Minus), followed by text. The last line will begin with the same code, followed immediately by Space <SP>, optionally some text, and TELNET <eol>. 1j1

For example:

```
123-First line
Second line
  234 A line beginning with numbers
123 The last line
```

1j1a

The user-process then simply needs to search for the second occurrence of the same reply code, followed by <SP> (Space), at the beginning of a line, and ignore all intermediary lines. If an intermediary line begins with a 3-digit number, the Server must pad the front to avoid confusion. 1j2

This scheme allows standard system routines to be used for reply information (such as for the STAT reply), with "artificial" first and last lines tacked on. In the rare cases where these routines are able to generate three digits and a Space at the beginning of any line, the beginning of each text line should be offset by some neutral text, like Space. 1j2a

This scheme assumes that multi-line replies may not be nested. We have found that, in general, nesting of replies will not occur, except for random system messages (called spontaneous replies in the previous FTP incarnations) which may interrupt another reply. Spontaneous replies are no longer defined; system messages (i.e. those not processed by the FTP server) will NOT carry reply codes and may occur anywhere in the command-reply sequence. They may be ignored by the User-process as they are only information for the human user. 1j3

The three digits of the reply each have a special significance. This is intended to allow a range of very simple to very sophisticated response by the user-process. The first digit denotes whether the response is good, bad or incomplete. (Referring to the state diagram) an unsophisticated user-process will be able to determine its

next action (proceed as planned, redo, retrench, etc.) by simply examining this first digit. A user-process that wants to know approximately what kind of error occurred (e.g. file system error, command syntax error) may examine the second digit, reserving the third digit for the finest gradation of information (e.g. RNTO command without a preceding RNFR.) 1k

There are four values for the first digit of the reply code: 1k1

1yz Positive Preliminary reply 1k2

The requested action is being initiated; expect another reply before proceeding with a new command. (The user-process sending another command before the completion reply would be in violation of protocol; but server-FTP processes should queue any commands that arrive while a preceeding command is in progress.) This type of reply can be used to indicate that the command was accepted and the user-process may now pay attention to the data connections, for implementations where simultaneous monitoring is difficult. 1k2a

2yz Positive Completion reply 1k3

The requested action has been successfully completed. A new request may be initiated. 1k3a

3yz Positive Intermediate reply 1k4

The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The user should send another command specifying this information. This reply is used in command sequence groups. 1k4a

4yz Transient Negative Completion reply 1k5

The command was not accepted and the requested action did not take place, but the error condition is temporary and the action may be requested again. The user should return to the beginning of the command sequence, if any. It is difficult to assign a meaning to "transient", particularly when two distinct sites (Server and User-processes) have to agree on the interpretation. Each reply in the 4yz category might have a slightly different time value, but the intent is that the user-process is encouraged to try again. A rule of thumb in determining if a reply fits into the 4yz or the

5yz (Permanent Negative) category is that replies are 4yz if the commands can be repeated without any change in command form or in properties of the User or Server (e.g. the command is spelled the same with the same arguments used; the user does not change his file access or user name; the server does not put up a new implementation.) 1k5a

5yz permanent Negative Completion reply 1k6

The command was not accepted and the requested action did not take place. The User-process is discouraged from repeating the exact request (in the same sequence). Even some "permanent" error conditions can be corrected, so the human user may want to direct his User-process to reinitiate the command sequence by direct action at some point in the future (e.g. after the spelling has been changed, or the user has altered his directory status.) 1k6a

The following function groupings are encoded in the second digit: 1k7

x0z Syntax - These replies refer to syntax errors, syntactically correct commands that don't fit any functional category, unimplemented or superfluous commands. 1k7a

x1z Information - These are replies to requests for information, such as status or help. 1k7b

x2z Connections - Replies referring to the TELNET and data connections. 1k7c

x3z Authentication and accounting - Replies for the logon process and accounting procedures. 1k7d

x4z Unspecified as yet 1k7e

x5z File system - These replies indicate the status of the Server file system vis-a-vis the requested transfer or other file system action. 1k7f

The third digit gives a finer gradation of meaning in each of the function categories, specified by the second digit. The list of replies below will illustrate this. Note that the text associated with each reply is suggestive, rather than mandatory, and may even change according to the command with which it is associated. The reply codes, on the other hand, should strictly follow the specifications

in the last section; that is, Server implementations should not invent new codes for situations that are only slightly different from the ones described here, but rather should adapt codes already defined. If additional codes are found to be necessary, the details should be submitted to the FTP committee, through Jon Postel. 1k8

A command such as TYPE or ALLO whose successful execution does not offer the user-process any new information will cause a 200 reply to be returned. If the command is not implemented by a particular Server-FTP process because it has no relevance to that computer system, for example ALLO at a TENEX site, a Positive Completion reply is still desired so that the simple User-process knows it can proceed with its course of action. A 202 reply is used in this case with, for example, the reply text: "No storage allocation necessary." If, on the other hand, the command requests a non-site-specific action and is unimplemented, the response is 502. A refinement of that is the 504 reply for a command that IS implemented, but that requests an unimplemented parameter. 1k8a 1k9

200 Command okay 1k9a
500 Syntax error, command unrecognized
[This may include errors such as command line too long.] 1k9b
501 Syntax error in parameters or arguments 1k9c
202 Command not implemented, superfluous at this site. 1k9d
502 Command not implemented 1k9e
503 Bad sequence of commands 1k9f
504 Command not implemented for that parameter 1k9g 1k10

110 Restart marker reply.
In this case the text is exact and not left to the particular implementation; it must read:
MARK yyyy = mmmm
where yyyy is user-process data stream marker, and mmmm is Server's equivalent marker. (note the spaces between the markers and "=".) 1k10a
211 System status, or system help reply 1k10b
212 Directory status 1k10c
213 File status 1k10d
214 Help message (on how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.) 1k10e 1k11

120 Service ready in nnn minutes 1k11a

220	Service ready for new user	1k11b
221	Service closing TELNET connection (logged off if appropriate)	1k11c
421	Service not available, closing TELNET connection. [This may be a reply to any command if the service knows it must shut down.]	1k11d
125	Data connection already open; transfer starting	1k11e
225	Data connection open; no transfer in progress	1k11f
425	Can't open data connection	1k11g
226	Closing data connection; requested file action successful (for example, file transfer or file abort.)	1k11h
426	Connection trouble, closed; transfer aborted.	1k11i
227	Entering [passive, active] mode	1k11j
		1k12
230	User logged on, proceed	1k12a
530	Not logged in	1k12b
331	User name okay, need password	1k12c
332	Need account for login	1k12d
532	Need account for storing files	1k12e
		1k13
150	File status okay; about to open data connection.	1k13a
250	Requested file action okay, completed.	1k13b
350	Requested file action pending further information	1k13c
450	Requested file action not taken: file unavailable (e.g. file not found, no access)	1k13d
550	Requested action not taken: file unavailable (e.g. file busy)	1k13e
451	Requested action aborted: local error in processing	1k13f
452	Requested action not taken: insufficient storage space in system	1k13g
552	Requested file action aborted: exceeded storage allocation (for current directory or dataset)	1k13h
553	Requested action not taken: file name not allowed	1k13i
354	Start mail input; end with <CR><LF>.<CR><LF>	1k13j

In this section, the command-reply sequence is presented. Each command is listed with its possible replies; command groups are listed together. Preliminary replies are listed first (with their succeeding replies under them), then positive and negative completion, and finally intermediary replies with the remaining commands from the sequence following. This listing forms the basis for the state diagrams, which will be presented separately. 1m

ICP 1m1
120 1m1a
220 1m1a1
220 1m1b
421 1m1c
Logon 1m2
USER 1m2a
230 1m2a1
530 1m2a2
500, 501, 421 1m2a3
331, 332 1m2a4
PASS 1m2b
230 1m2b1
202 1m2b2
530 1m2b3
500, 501, 503, 421 1m2b4
332 1m2b5
ACCT 1m2c
230 1m2c1
202 1m2c2
530 1m2c3
500, 501, 503, 421 1m2c4
Logoff 1m3
QUIT 1m3a
221 1m3a1
500 1m3a2
REIN 1m3b
120 1m3b1
220 1m3b1a
220 1m3b2
421 1m3b3
500, 502 1m3b4
Transfer parameters 1m4
SOCK 1m4a
200 1m4a1
500, 501, 421, 530 1m4a2
PASV 1m4b
227 1m4b1
500, 501, 502, 421, 530 1m4b2
ACTV 1m4c

227 1m4c1
202 1m4c2
500, 501, 421, 530 1m4c3
BYTE, MODE, TYPE, STRU 1m4d
200 1m4d1
500, 501, 504, 421, 530 1m4d2
File action commands 1m5
ALLO 1m5a
200 1m5a1
202 1m5a2
500, 501, 504, 421, 530 1m5a3
REST 1m5b
500, 501, 502, 421, 530 1m5b1
350 1m5b2
STOR 1m5c
125, 150 1m5c1
(110) 1m5c1a
226, 250 1m5c1b
425, 426, 451, 552 1m5c1c
532, 450, 452, 553 1m5c2
500, 501, 421, 530 1m5c3
RETR 1m5d
125, 150 1m5d1
(110) 1m5d1a
226, 250 1m5d1b
425, 426, 451 1m5d1c
450, 550 1m5d2
500, 501, 421, 530 1m5d3
LIST, NLST 1m5e
125, 150 1m5e1
226, 250 1m5e1a
425, 426, 451 1m5e1b
450 1m5e2
500, 501, 502, 421, 530 1m5e3
APPE 1m5f
125, 150 1m5f1
(110) 1m5f1a
226, 250 1m5f1b
425, 426, 451, 552 1m5f1c
532, 450, 550, 452, 553 1m5f2
500, 501, 502, 421, 530 1m5f3
MLFL 1m5g
125, 150 1m5g1
226, 250 1m5g1a
425, 426, 451, 552 1m5g1b
532, 450, 550, 452, 553 1m5g2
500, 501, 502, 421, 530 1m5g3
RNFR 1m5h
450, 550 1m5h1

```
      500, 501, 502, 421, 530                                  1m5h2
      350                                                      1m5h3
   RNTO                                                         1m5i
      250                                                      1m5i1
      532, 553                                                 1m5i2
      500, 501, 502, 503, 421, 530                             1m5i3
   DELE                                                         1m5j
      250                                                      1m5j1
      450, 550                                                 1m5j2
      500, 501, 502, 421, 530                                  1m5j3
   ABOR                                                         1m5k
      225, 226                                                 1m5k1
      500, 501, 502, 421                                       1m5k2
   MAIL                                                         1m5l
      354                                                      1m5l1
         250                                                  1m5l1a
         451, 552                                             1m5l1b
      450, 550, 452, 553                                       1m5l2
      500, 501, 502, 421, 530                                  1m5l3
Informational commands                                          1m6
   STAT                                                         1m6a
      211, 212, 213                                            1m6a1
      450                                                      1m6a2
      500, 501, 502, 421, 530                                  1m6a3
   HELP                                                         1m6b
      211, 214                                                 1m6b1
      500, 501, 502, 421                                       1m6b2
Miscellaneous commands                                          1m7
   SITE                                                         1m7a
      200                                                      1m7a1
      202                                                      1m7a2
      500, 501, 530                                            1m7a3
   NOOP                                                         1m7b
      200                                                      1m7b1
      500                                                      1m7b2
```

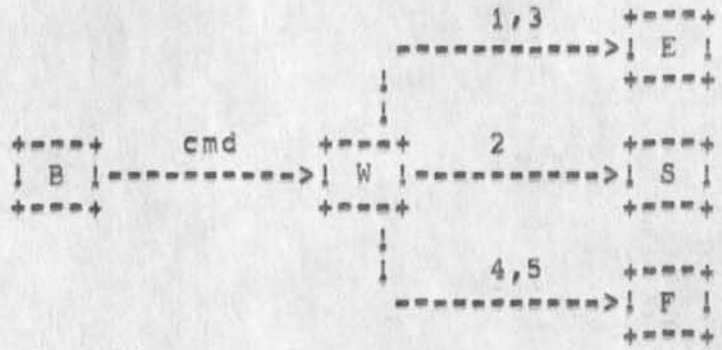
FTP State Diagrams 1n

Here we present state diagrams for a very simple minded FTP implementation. Only the first digit of the reply codes is used. There is one state diagram for each group of FTP commands or command sequences. 1o

The command groupings were determined by constructing a model for each command then collecting together the commands with structurally identical models. 1p

For each command or command sequence there are three possible outcomes: success (S), failure (F), and error (E). In the state diagrams below we use the symbol B for "begin", and the symbol W for "wait for reply". 1q

We first present the diagram that represents the largest group of FTP commands: 1r

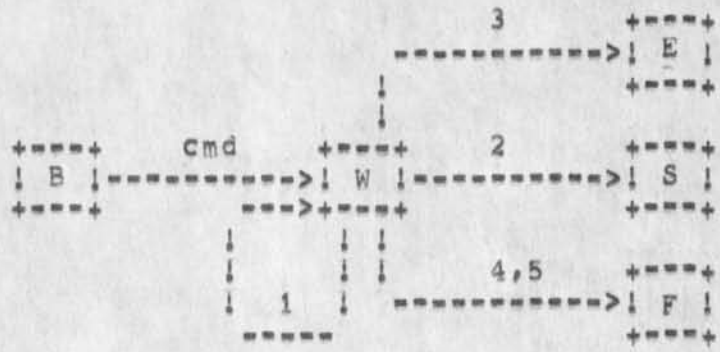


1r1

This diagram models the commands: 1r2

ABOR, ACTV, ALLO, BYTE, DELE, HELP, MODE, NOOP, PASV, QUIT, SITE, SOCK, STAT, STRU, TYPE. 1r2a

The other large group of commands is represented by a very similar diagram: 1s



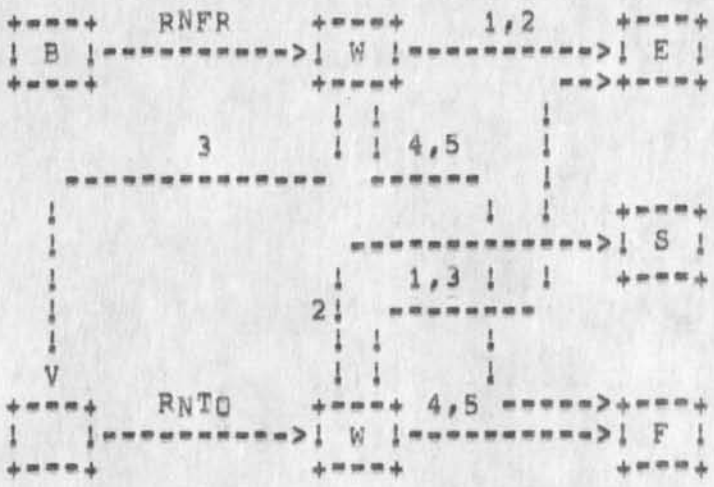
1s1

This diagram models the commands: 1s2

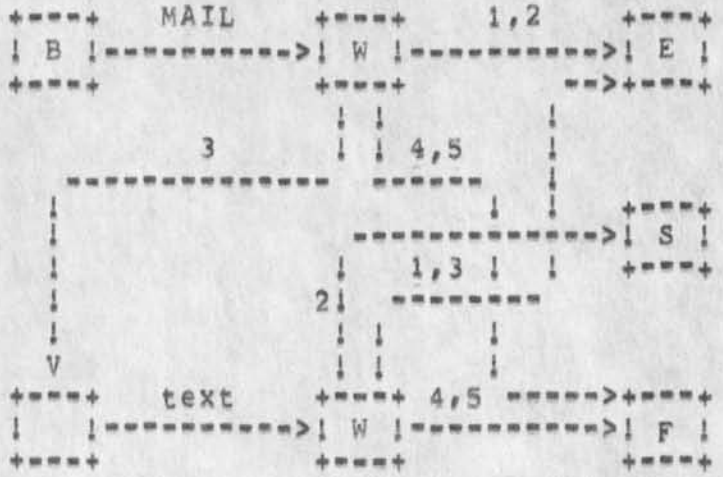
APPE, (ICP), LIST, MLFL, NLST, REIN, RETR, STOR. 1s2a

Note that this second model could also be used to represent the first group of commands, the only difference being that in the first group the 100 series replies are unexpected and therefore treated as error, while the second group expects (some may require) 100 series replies. 1t

The remaining diagrams model command sequences, perhaps the simplest of these is the rename sequence: 1u


RNFR
1,2
3
4,5
1,3
2
RNTO
4,5
B
W
E
W
S
F

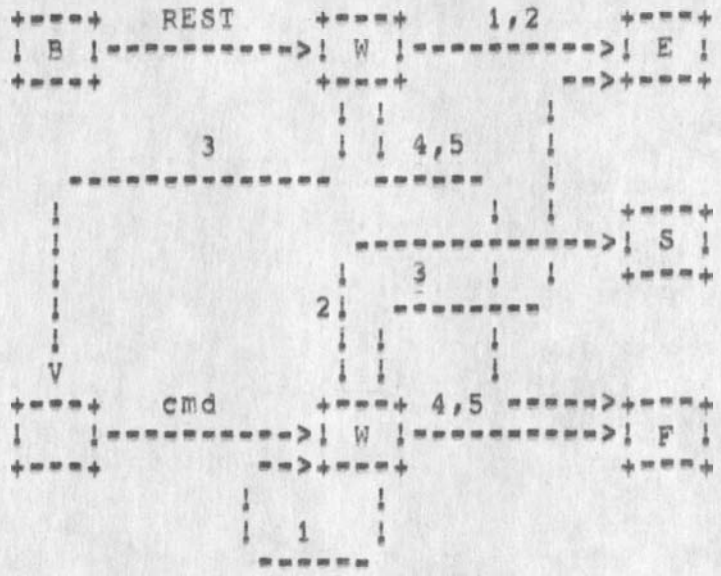
A very similar diagram models the Mail command: 1v



1v1

Note that the "text" here is a series of lines sent from the user to the server with no response expected until the last line is sent, recall that the last line must consist only of a single period. 1v2

The next diagram is a simple model of the Restart command: 1w

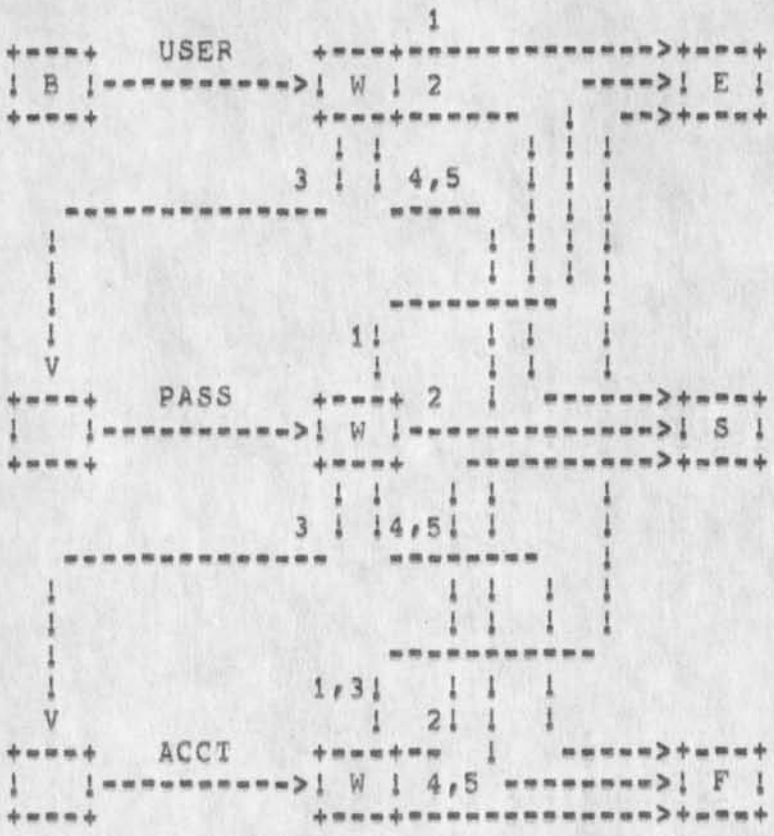


1w1

Where "cmd" is APPE, STOR, RETR, or MLFL. 1w1a

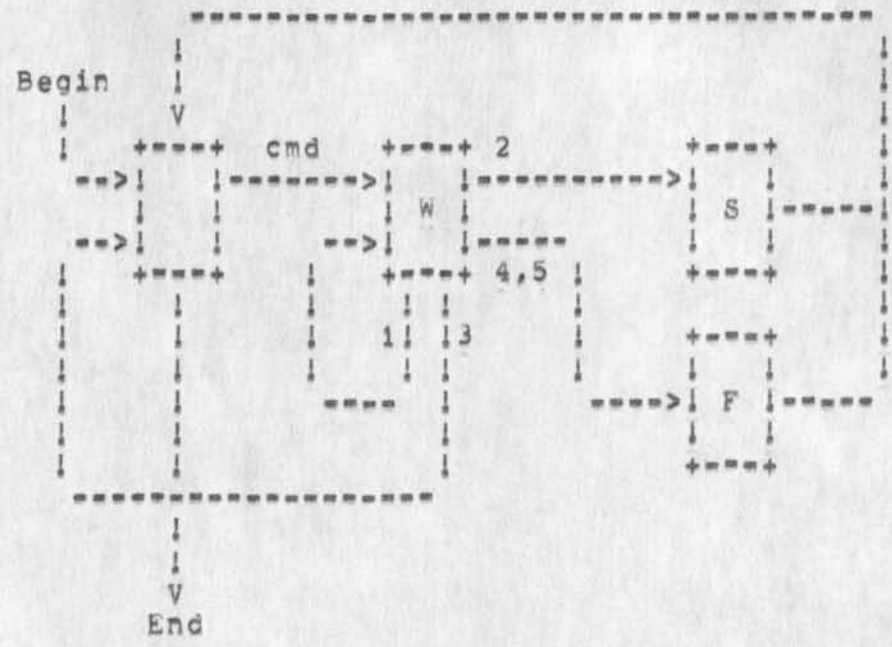
We note that the above three models are similar, in fact the Mail diagram and the Rename diagram are structurally identical. The Restart differs from the other two only in the treatment of 100 series replies at the second stage. 1x

The most complicated diagram is for the Logon sequence: 1y



1y1

Finally we present a generalized diagram that could be used to model the command and reply interchange: 1z



1z1

FTPCODES

(J24299) 24-OCT-74 13:00;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

FTPCODES

<POSTEL>FTPCODES.NLS;1, 26-SEP-74 13:23 JBP ; 1

Jon Postel
24 OCT 74

Revised FTP Reply Codes

1a

This document describes a revised set of reply codes for the File Transfer Protocol. 1b

The aim of this revision is to satisfy the goal of using reply codes to enable the command issuing process to easily determine the outcome of each command. The user protocol interpreter should be able to determine the success or failure of a command by examining the first digit of the reply code. 1c

An important change in the sequencing of commands and replies which may not be obvious in the following documents concerns the establishment of the data connection. 1d

> In the previous FTP specifications when an actual transfer command (STOR, RETR, APPE, LIST, NLIST, MLFL) was issued the preliminary reply was sent after the data connection was established. This presented a problem for some user protocol interpreters which had difficulty monitoring two connections asynchronously. 1d1
>
> The current specification is that the preliminary reply to the actual transfer commands indicates that the file can be transferred and either the connection was previously established or an attempt is about to be made to establish the data connection. 1d2

This reply code revision is a modification of the protocol in described in RFC 542, that is to say that the protocol implementation associated with socket number 21 (decimal) is the protocol specified by the combination of RFC 542 and this RFC. 1e

A note of thanks to those who contributed to this work: Ken Pogran, Mark Krilanovich, Wayne Hathway, and especially Nancy Neigus. 1f

JBP 24-OCT-74 13:32 24300

Nancy Neigus
Ken Pogran
Jon Postel
24 OCT 74

A New Schema for FTP Reply Codes

19

Replies to File Transfer protocol commands were devised to ensure the synchronization of requests and actions in the process of file transfer, and to guarantee that the user process always knows the state of the Server. Every command must generate at least one reply, although there may be more than one; in the latter case, the multiple replies must be easily distinguished. In addition, some commands occur in sequential groups, such as USER, PASS and ACCT, or RNFR and RNTO. The replies show the existence of an intermediate state if all preceding commands have been successful. A failure at any point in the sequence necessitates the repetition of the entire sequence from the beginning.

1h

Details of the command-reply sequence will be made explicit in a state diagram.

1h1

An FTP reply consists of a three digit number (transmitted as three alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is intended for the human user. It is intended that the three digits contain enough encoded information that the user-process (the User-PI described in RFC 542) will not need to examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be server-dependent, so there are likely to be varying texts for each reply code.

1i

Formally, a reply is defined to contain the 3-digit code, followed by Space <SP>, followed by one line of text (where some maximum line length has been specified), and terminated by the TELNET end-of-line code. There will be cases, however, where the text is longer than a single line. In these cases the complete text must be bracketed so the User-process knows when it may stop reading the reply (i,e, stop processing input on the TELNET connection) and go do other things. This requires a special format on the first line to indicate that more than one line is coming, and another on the last line to designate it as the last. At least one of these must contain

the appropriate reply code to indicate the state of the transaction. To satisfy all factions it was decided that both the first and last line codes should be the same. 1j

Thus the format for multi-line replies is that the first line will begin with the exact required reply code, followed immediately by a Hyphen, "-" (also known as Minus), followed by text. The last line will begin with the same code, followed immediately by Space <SP>, optionally some text, and TELNET <eol>. 1j1

For example:

```
123-First line
Second line
  234 A line beginning with numbers
123 The last line
```
1j1a

The user-process then simply needs to search for the second occurrence of the same reply code, followed by <SP> (Space), at the beginning of a line, and ignore all intermediary lines. If an intermediary line begins with a 3-digit number, the Server must pad the front to avoid confusion. 1j2

This scheme allows standard system routines to be used for reply information (such as for the STAT reply), with "artificial" first and last lines tacked on. In the rare cases where these routines are able to generate three digits and a Space at the beginning of any line, the beginning of each text line should be offset by some neutral text, like Space. 1j2a

This scheme assumes that multi-line replies may not be nested. We have found that, in general, nesting of replies will not occur, except for random system messages (called spontaneous replies in the previous FTP incarnations) which may interrupt another reply. Spontaneous replies are no longer defined; system messages (i.e. those not processed by the FTP server) will NOT carry reply codes and may occur anywhere in the command-reply sequence. They may be ignored by the User-process as they are only information for the human user. 1j3

The three digits of the reply each have a special significance. This is intended to allow a range of very simple to very sophisticated response by the user-process. The first digit denotes whether the response is good, bad or incomplete. (Referring to the state diagram) an unsophisticated user-process will be able to determine its

next action (proceed as planned, redo, retrench, etc.) by simply examining this first digit. A user-process that wants to know approximately what kind of error occurred (e.g. file system error, command syntax error) may examine the second digit, reserving the third digit for the finest gradation of information (e.g. RNTO command without a preceding RNFR.) 1k

There are four values for the first digit of the reply code: 1k1

1yz Positive Preliminary reply 1k2

The requested action is being initiated; expect another reply before proceeding with a new command. (The user-process sending another command before the completion reply would be in violation of protocol; but server-FTP processes should queue any commands that arrive while a preceeding command is in progress.) This type of reply can be used to indicate that the command was accepted and the user-process may now pay attention to the data connections, for implementations where simultaneous monitoring is difficult. 1k2a

2yz Positive Completion reply 1k3

The requested action has been successfully completed. A new request may be initiated. 1k3a

3yz Positive Intermediate reply 1k4

The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The user should send another command specifying this information. This reply is used in command sequence groups. 1k4a

4yz Transient Negative Completion reply 1k5

The command was not accepted and the requested action did not take place, but the error condition is temporary and the action may be requested again. The user should return to the beginning of the command sequence, if any. It is difficult to assign a meaning to "transient", particularly when two distinct sites (Server and User-processes) have to agree on the interpretation. Each reply in the 4yz category might have a slightly different time value, but the intent is that the user-process is encouraged to try again. A rule of thumb in determining if a reply fits into the 4yz or the

5yz (Permanent Negative) category is that replies are 4yz if the commands can be repeated without any change in command form or in properties of the User or Server (e.g. the command is spelled the same with the same arguments used; the user does not change his file access or user name; the server does not put up a new implementation.) 1k5a

5yz Permanent Negative Completion reply 1k6

The command was not accepted and the requested action did not take place. The User-process is discouraged from repeating the exact request (in the same sequence). Even some "permanent" error conditions can be corrected, so the human user may want to direct his User-process to reinitiate the command sequence by direct action at some point in the future (e.g. after the spelling has been changed, or the user has altered his directory status.) 1k6a

The following function groupings are encoded in the second digit: 1k7

x0z Syntax - These replies refer to syntax errors, syntactically correct commands that don't fit any functional category, unimplemented or superfluous commands. 1k7a

x1z Information - These are replies to requests for information, such as status or help. 1k7b

x2z Connections - Replies referring to the TELNET and data connections. 1k7c

x3z Authentication and accounting - Replies for the logon process and accounting procedures. 1k7d

x4z Unspecified as yet 1k7e

x5z File system - These replies indicate the status of the Server file system vis-a-vis the requested transfer or other file system action. 1k7f

The third digit gives a finer gradation of meaning in each of the function categories, specified by the second digit. The list of replies below will illustrate this. Note that the text associated with each reply is suggestive, rather than mandatory, and may even change according to the command with which it is associated. The reply codes, on the other hand, should strictly follow the specifications

in the last section; that is, Server implementations should not invent new codes for situations that are only slightly different from the ones described here, but rather should adapt codes already defined. If additional codes are found to be necessary, the details should be submitted to the FTP committee, through Jon Postel. 1k8

A command such as TYPE or ALLO whose successful execution does not offer the user-process any new information will cause a 200 reply to be returned. If the command is not implemented by a particular Server-FTP process because it has no relevance to that computer system, for example ALLO at a TENEX site, a Positive Completion reply is still desired so that the simple User-process knows it can proceed with its course of action. A 202 reply is used in this case with, for example, the reply text: "No storage allocation necessary." If, on the other hand, the command requests a non-site-specific action and is unimplemented, the response is 502. A refinement of that is the 504 reply for a command that IS implemented, but that requests an unimplemented parameter. 1k8a
1k9

200 Command okay 1k9a
500 Syntax error, command unrecognized
[This may include errors such as command line too long.] 1k9b
501 Syntax error in parameters or arguments 1k9c
202 Command not implemented, superfluous at this site. 1k9d
502 Command not implemented 1k9e
503 Bad sequence of commands 1k9f
504 Command not implemented for that parameter 1k9g
1k10

110 Restart marker reply.
In this case the text is exact and not left to the particular implementation; it must read:
MARK yyyy = mmmm
where yyyy is User-process data stream marker, and mmmm is Server's equivalent marker. (note the spaces between the markers and "=".) 1k10a
211 System status, or system help reply 1k10b
212 Directory status 1k10c
213 File status 1k10d
214 Help message (on how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.) 1k10e
1k11

120 Service ready in nnn minutes 1k11a

220	Service ready for new user	1k11b
221	Service closing TELNET connection (logged off if appropriate)	1k11c
421	Service not available, closing TELNET connection. [This may be a reply to any command if the service knows it must shut down.]	1k11d
125	Data connection already open; transfer starting	1k11e
225	Data connection open; no transfer in progress	1k11f
425	Can't open data connection	1k11g
226	Closing data connection; requested file action successful (for example, file transfer or file abort.)	1k11h
426	Connection trouble, closed; transfer aborted.	1k11i
227	Entering [passive, active] mode	1k11j 1k12
230	User logged on, proceed	1k12a
530	Not logged in	1k12b
331	User name okay, need password	1k12c
332	Need account for login	1k12d
532	Need account for storing files	1k12e 1k13
150	File status okay; about to open data connection.	1k13a
250	Requested file action okay, completed.	1k13b
350	Requested file action pending further information	1k13c
450	Requested file action not taken: file unavailable (e.g. file not found, no access)	1k13d
550	Requested action not taken: file unavailable (e.g. file busy)	1k13e
451	Requested action aborted: local error in processing	1k13f
452	Requested action not taken: insufficient storage space in system	1k13g
552	Requested file action aborted: exceeded storage allocation (for current directory or dataset)	1k13h
553	Requested action not taken: file name not allowed	1k13i
354	Start mail input; end with <CR><LF>.<CR><LF>	1k13j

In this section, the command-reply sequence is presented. Each command is listed with its possible replies; command groups are listed together. Preliminary replies are listed first (with their succeeding replies under them), then positive and negative completion, and finally intermediary replies with the remaining commands from the sequence following. This listing forms the basis for the state diagrams, which will be presented separately. 1m

ICP 1m1
120 1m1a
220 1m1a1
220 1m1b
421 1m1c
Logon 1m2
USER 1m2a
230 1m2a1
530 1m2a2
500, 501, 421 1m2a3
331, 332 1m2a4
PASS 1m2b
230 1m2b1
202 1m2b2
530 1m2b3
500, 501, 503, 421 1m2b4
332 1m2b5
ACCT 1m2c
230 1m2c1
202 1m2c2
530 1m2c3
500, 501, 503, 421 1m2c4
Logoff 1m3
QUIT 1m3a
221 1m3a1
500 1m3a2
REIN 1m3b
120 1m3b1
220 1m3b1a
220 1m3b2
421 1m3b3
500, 502 1m3b4
Transfer parameters 1m4
SOCK 1m4a
200 1m4a1
500, 501, 421, 530 1m4a2
PASV 1m4b
227 1m4b1
500, 501, 502, 421, 530 1m4b2
ACTV 1m4c

```
      227                                                    1m4c1
      202                                                    1m4c2
      500, 501, 421, 530                                     1m4c3
   BYTE, MODE, TYPE, STRU                                    1m4d
      200                                                    1m4d1
      500, 501, 504, 421, 530                                1m4d2
File action commands                                         1m5
   ALLO                                                      1m5a
      200                                                    1m5a1
      202                                                    1m5a2
      500, 501, 504, 421, 530                                1m5a3
   REST                                                      1m5b
      500, 501, 502, 421, 530                                1m5b1
      350                                                    1m5b2
   STOR                                                      1m5c
      125, 150                                               1m5c1
         (110)                                               1m5c1a
         226, 250                                            1m5c1b
         425, 426, 451, 552                                  1m5c1c
      532, 450, 452, 553                                     1m5c2
      500, 501, 421, 530                                     1m5c3
   RETR                                                      1m5d
      125, 150                                               1m5d1
         (110)                                               1m5d1a
         226, 250                                            1m5d1b
         425, 426, 451                                       1m5d1c
      450, 550                                               1m5d2
      500, 501, 421, 530                                     1m5d3
   LIST, NLST                                                1m5e
      125, 150                                               1m5e1
         226, 250                                            1m5e1a
         425, 426, 451                                       1m5e1b
      450                                                    1m5e2
      500, 501, 502, 421, 530                                1m5e3
   APPE                                                      1m5f
      125, 150                                               1m5f1
         (110)                                               1m5f1a
         226, 250                                            1m5f1b
         425, 426, 451, 552                                  1m5f1c
      532, 450, 550, 452, 553                                1m5f2
      500, 501, 502, 421, 530                                1m5f3
   MLFL                                                      1m5g
      125, 150                                               1m5g1
         226, 250                                            1m5g1a
         425, 426, 451, 552                                  1m5g1b
      532, 450, 550, 452, 553                                1m5g2
      500, 501, 502, 421, 530                                1m5g3
   RNFR                                                      1m5h
      450, 550                                               1m5h1
```

```
      500, 501, 502, 421, 530                                  1m5h2
      350                                                      1m5h3
   RNTO                                                         1m5i
      250                                                      1m5i1
      532, 553                                                 1m5i2
      500, 501, 502, 503, 421, 530                             1m5i3
   DELE                                                         1m5j
      250                                                      1m5j1
      450, 550                                                 1m5j2
      500, 501, 502, 421, 530                                  1m5j3
   ABOR                                                         1m5k
      225, 226                                                 1m5k1
      500, 501, 502, 421                                       1m5k2
   MAIL                                                         1m5l
      354                                                      1m5l1
         250                                                  1m5l1a
         451, 552                                             1m5l1b
      450, 550, 452, 553                                       1m5l2
      500, 501, 502, 421, 530                                  1m5l3
Informational commands                                          1m6
   STAT                                                         1m6a
      211, 212, 213                                            1m6a1
      450                                                      1m6a2
      500, 501, 502, 421, 530                                  1m6a3
   HELP                                                         1m6b
      211, 214                                                 1m6b1
      500, 501, 502, 421                                       1m6b2
Miscellaneous commands                                          1m7
   SITE                                                         1m7a
      200                                                      1m7a1
      202                                                      1m7a2
      500, 501, 530                                            1m7a3
   NOOP                                                         1m7b
      200                                                      1m7b1
      500                                                      1m7b2
```

Jon Postel
24 OCT 74

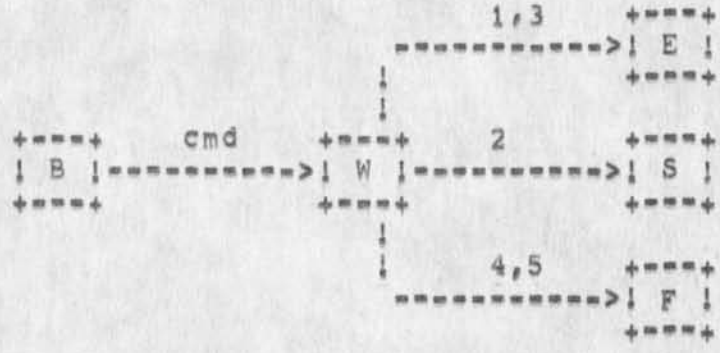
FTP State Diagrams 1n

Here we present state diagrams for a very simple minded FTP implementation. Only the first digit of the reply codes is used. There is one state diagram for each group of FTP commands or command sequences. 1o

The command groupings were determined by constructing a model for each command then collecting together the commands with structurally identical models. 1p

For each command or command sequence there are three possible outcomes: success (S), failure (F), and error (E). In the state diagrams below we use the symbol B for "begin", and the symbol W for "wait for reply". 1q

We first present the diagram that represents the largest group of FTP commands: 1r

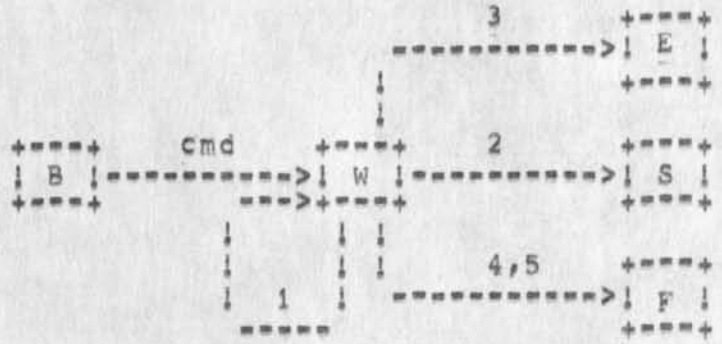


1r1

This diagram models the commands: 1r2

ABOR, ACTV, ALLO, BYTE, DELE, HELP, MODE, NOOP, PASV, QUIT, SITE, SOCK, STAT, STRU, TYPE. 1r2a

The other large group of commands is represented by a very similar diagram: 1s



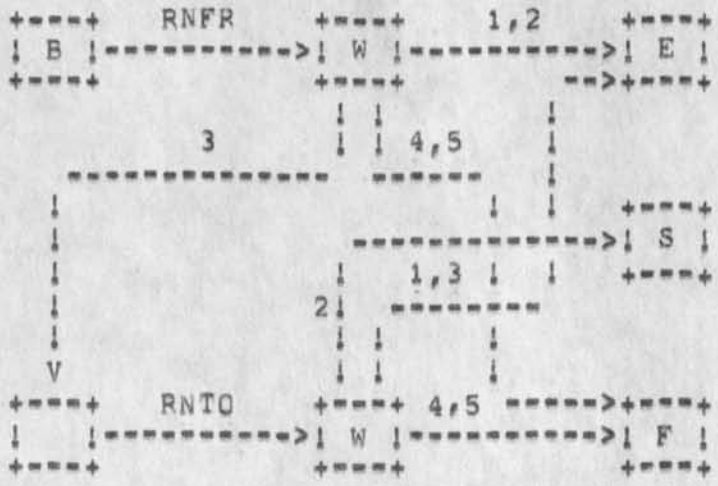
1s1

This diagram models the commands: 1s2

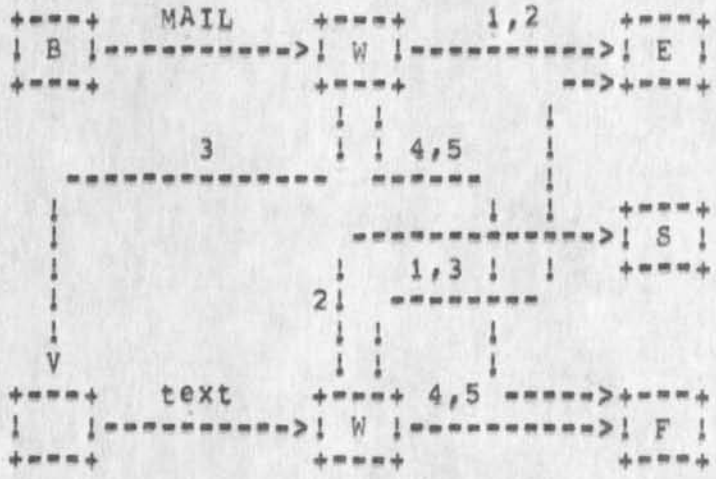
APPE, (ICP), LIST, MLFL, NLST, REIN, RETR, STOR, 1s2a

Note that this second model could also be used to represent the first group of commands, the only difference being that in the first group the 100 series replies are unexpected and therefore treated as error, while the second group expects (some may require) 100 series replies. 1t

The remaining diagrams model command sequences, perhaps the simplest of these is the rename sequence: 1u


RNFR
1,2
3
4,5
1,3
2
RNTO
4,5
B
W
E
S
W
F

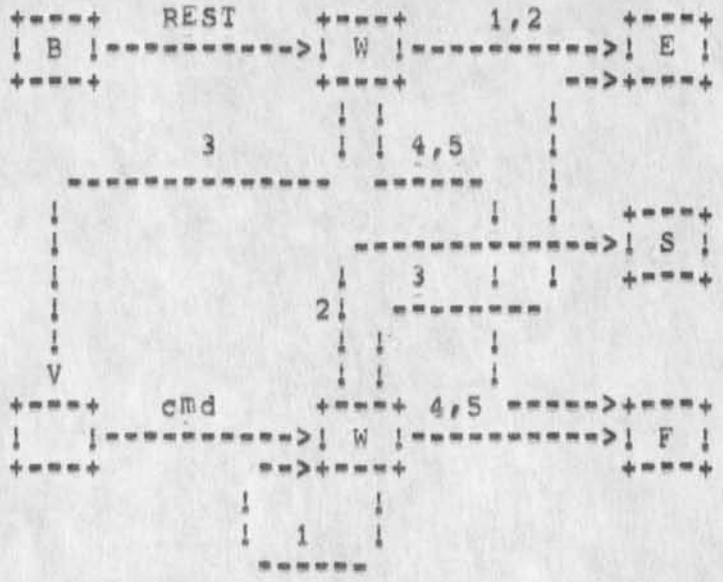
A very similar diagram models the Mail command: 1v



1v1

Note that the "text" here is a series of lines sent from the user to the server with no response expected until the last line is sent, recall that the last line must consist only of a single period. 1v2

The next diagram is a simple model of the Restart command: 1w

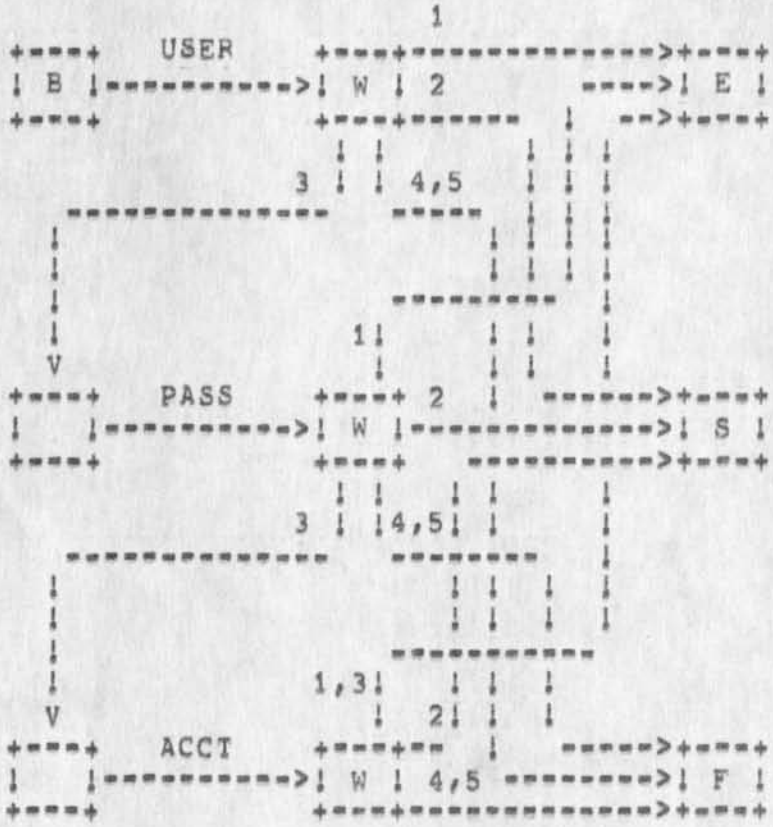


1w1

Where "cmd" is APPE, STOR, RETR, or MLFL. 1w1a

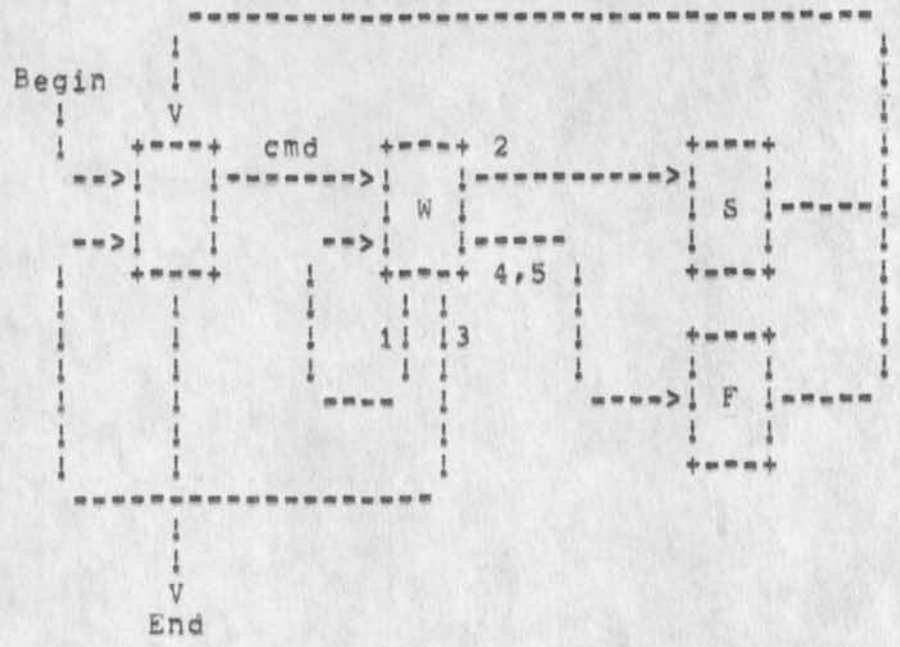
We note that the above three models are similar, in fact the Mail diagram and the Rename diagram are structurally identical. The Restart differs from the other two only in the treatment of 100 series replies at the second stage. 1x

The most complicated diagram is for the Logon sequence; 1y



1y1

Finally we present a generalized diagram that could be used to model the command and reply interchange: 1z



1z1

FTPCODES

(J24300) 24-OCT-74 13:32;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

<POSTEL>FTPSPEC.NLS;1, 26-SEP-74 12:56 JBP ; 1

INTRODUCTION 1a

The File Transfer protocol (FTp) is a protocol for file transfer between hosts (including Terminal Interface Message Processors (TIPs)) on the ARPA Computer Network (ARPANET). The primary function of FTP is to transfer files efficiently and reliably among hosts and to allow the convenient use of remote file storage capabilities. 1a1

The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs. 1a2

The attempt in this specification is to satisfy the diverse needs of users of maxi-hosts, mini-hosts, TIPs, and the Data-computer, with a simple, and easily implemented protocol design. 1a3

This paper assumes knowledge of the following protocols described in NIC #7104: 1a4

The host-host Protocol 1a4a

The Initial Connection Protocol 1a4b

The Telnet Protocol 1a4c

DISCUSSION 1b

In this section, the terminology and the FTP model are discussed. The terms defined in this section are only those that have special significance in FTP. 1b1

Terminology 1b2

ASCII 1b2a

The ASCII character set as defined in NIC #7104. In FTP, ASCII characters are defined to be the lower half of an eight-bit code set (i.e., the most significant bit is zero). 1b2a1

access controls 1b2b

Access controls define users' access privileges to the use of a system, and to the files in that system. Access controls are necessary to prevent unauthorized or accidental use of files. It is the prerogative of a server-FTP process to provide access controls. 1b2b1

byte size 1b2c

The byte size specified for the transfer of data. The data connection is opened with this byte size. The data connection byte size is not necessarily the byte size in which data is to be stored in a system, nor the logical byte size for interpretation of the structure of the data. 1b2c1

data connection 1b2d

A simplex connection over which data is transferred, in a specified byte size, mode and type. The data transferred may be a part of a file, an entire file or a number of files. The path may be between a server-DTP and a user-DTP, or between two server-DTPs. 1b2d1

data socket 1b2e

The passive data transfer process "listens" on the data socket for an RFC from the active transfer process (server) in order to open the data connection. The server has fixed data sockets; the passive process may or may not. 1b2e1

EOF 1b2f

The end-of-file condition that defines the end of a file being transferred. 1b2f1

EOR 1b2g

The end-of-record condition that defines the end of a record being transferred. 1b2g1

error recovery 1b2h

A procedure that allows a user to recover from certain errors such as failure of either host system or transfer process. In FTP, error recovery may involve restrarting a file transfer at a given checkpoint. 1b2h1

FTP commands 1b2i

A set of commands that comprise the control information flowing from the user-FTP to the server-FTP process. 1b2i1

file 1b2j

An ordered set of computer data (including programs), of arbitrary length, uniquely identified by a pathname. 1b2j1

mode 1b2K

The mode in which data is to be transferred via the data connection. The mode defines the data format during transfer including EOR and EOF. The transfer modes defined in FTP are described in Section 3D. 1b2k1

NVT 1b2l

The Virtual Terminal as defined in the ARPANET Telnet Protocol. 1b2l1

NVFS 1b2m

The Network Virtual File System. A concept which defines a standard network file system with standard commands and pathname conventions. FTP only partially embraces the NVFS concept at this time. 1b2m1

pathname 1b2n

pathname is defined to be the character string which must be input to a file system by a user in order to identify a file. Pathname normally contains device and/or directory names, and file name specification. FTP does not yet specify a standard pathname convention. Each user must follow the file naming conventions of the file systems he wishes to use. 1b2n1

record 1b2o

A sequential file may be structured as a number of contiguous parts called records. Record structures are supported by FTP but a file need not have record strucutre. 1b2o1

reply 1b2p

A reply is an acknowledgment (positive or negative) sent from server to user via the Telnet connections in response to FTP commands. The general form of a reply is

a completion code (including error codes) followed by a text string. The codes are for use by programs and the text is usually intended for human users. 1b2p1

server-DPT 1b2q

The data transfer process, in its normal "active" state, establishes the data connectioin by RFC to the "listening" data socket, sets up parameters for transfer and storage, and tranfers data on command from its PI. The DTP can be placed in a "passive" state to listen for, rather than initiate, an RFC on the data socket. 1b2q1

server-FTP process 1b2r

A process or set of processes which perform the function of file transfer in cooperation with a user-FTP process and, possibly, another server. The functions consist of a protocol interpreter (PI)and a data transfer process (DTP). 1b2r1

server-PI 1b2s

The protocol interpreter "listens" on socket 3 for an ICP from a user-PI and establishes a Telnet communication connection. It receives standard FTP commands from the user-PI, sends replies, and governs the server-DTP. 1b2s1

Telnet connections 1b2t

The full-duplex communication path between a user-PI and a server-PI. The Telnet connections are established via the standard ARPANET Initial Connection Protocol (ICP). 1b2t1

type 1b2u

The data representation type used for data transfer and storage. Type implies certain transformations between the time of data storage and data transfer. The representation types defined in FTP described in Section 3B. 1b2u1

user 1b2v

A human being or a process on behalf of a human being wishing to obtain file transfer service. The human user may interact directly with a server-FTP process, but use of a user-FTP process is preferred since the protocol design is weighted towards automata. 1b2v1

user-DTP 1b2w

The data transfer process "listens" on the data socket for an RFC from a server-FTP process. If two servers are transferring data between them, the user-DTP is inactive. 1b2w1

user-FTP process 1b2x

A set of functions including a protocol interpreter, a data transfer process and a user interface which together perform the function of file transfer in cooperation with one or more server-FTP processes. The user interface allows a local language to be used in the command-reply dialogue with the user. 1b2x1

user-PI 1b2y

The protocol interpreter initiates the ICP to the server-FTP process, initiates FTP commands, and governs the user-DTP if that process is part of the file transfer. 1b2y1

The FTP Model 1b3

With the above definitions in mind, the following model (shown in Figure 1) maybe diagrammed for an FTP service. 1b3a

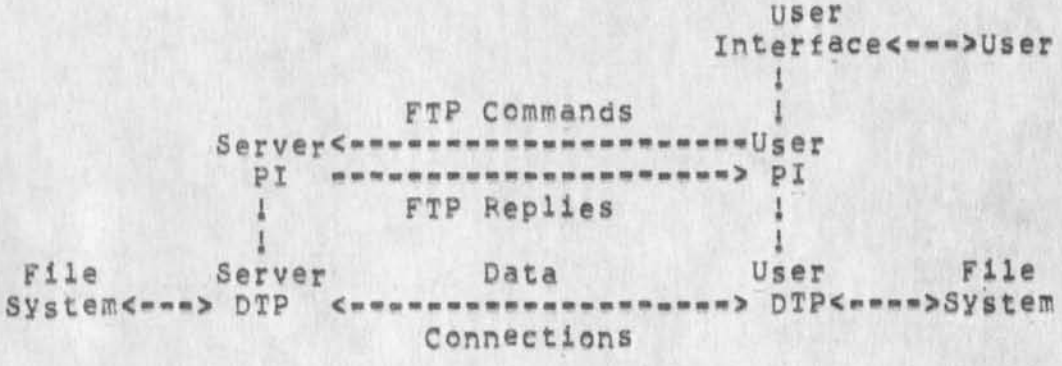


Figure 1 1b3b

In the model described in Figure 1, the user-protocol interpreter initiates the Telnet connections. At the initiation of the user, standard FTP commands are generated by the user-PI and transmitted to the server process via the TELENT connections. (The user may establish a direct Telnet connection to the server-FTP, from a TIP terminal for

example, and generate standard FTP commands himself, by-passing the user-FTP process.) Standard replies are sent from the server-PI to the user-PI over the Telnet connections in response to the commands. 1b3c

The FTP commands specify the parameters for the data connection (data socket, byte size, transfer mode, representation type, and structure) and the nature of file system operation (store, retrieve, append, delete, etc.). The user-DTP or its designate should "listen" on the specified data socket, and the server initiate the data connection and data transfer in accordance with the specified parameters. It should be noted that the data socket need not be in the same host that initiates the FTP commands via the Telnet connections, but the user or his user-FTP process must ensure a "listen" on the specified data socket. It should also be noted that two data connections, one for send and the other for receive, may exist simultaneously. 1b3d

In another situation a user might wish to transfer files between two hosts, neither of which is his local host. He sets up Telnet connections to the two servers and then arranges for a data connecton between them. In this manner control information is passed to the user-PI but data is transferred between he server data transfer processes. Following is a model of this server-server interaction. 1b3e

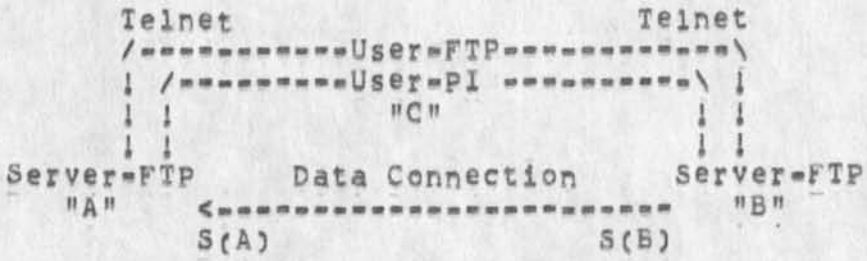


Figure 2 1b3f

The protocol requires that the Telnet connections be open while data transfer is in progress. It is the responsibility of the user to request the closing of the Telnet connections when finished using the FTP service, while it is the server who takes the action. The server may abort data transfer if the Telnet connections are closed without command. 1b3g

DATA TRANSFER FUNCTIONS 1c

Files are transferred only via the data connection(s). The Telnet connection is used for the transfer of commands, which describe the functions to be performed, and the replies to these commands (see Section 4C). Several commands are concerned with the transfer of data between hosts. These data transfer commands include the BYTE, MODE, and SOCket commands which specify how the bits of the data are to be transmitted, and the STRUcture and TYpE commands, which are used to define the way in which thhe data are to be represented. The transmission and representation are basically independent but "Stream" transmission mode is dependent on the file structure attribute and if "Compressed" transmission mode is used the nature of the filler byte depends on the representation type. 1c1

Data Representation and Storage 1c2

Data is transferred from a storage device in the sending host to a storage device in the receiving host. Often it is necessary to perform certain transformations on the data because data storage representations in the two systems are different. For example, NVT-ASCII has different data storage representations in diffeent systems. PDP-10's generally store NVT-ASCII as five 7-bit ASCII characters, left-justified in a 36-bit word. 360's store NVT-ASCII as 8-bit EBCDIC codes. Multics stores NVT-ASCII as four 9-bit characters in a 36-bit word. It may be desirable to convert characters into the standard NVT-ASCII representation when transmitting text between dissimilar systems. The sending and receiving sites would have to perform the necessary transformations between the standard representation and their internal representations. 1c2a

A different problem in representation arises when transmitting binary data (not character codes) between host systems with different word lengths. It is not always clear how the sender should send data, and the receiver store it. For example, when transmitting 32-bit bytes from a 32-bit word-length system to a 36-bit word-length system, it may be desirable (for reasons of efficiency and usefulness) to store the 32-bit bytes right-justified in a 36-bit word in the latter system. In any case, the user should have the option of specifying data representation and transformation functions. It should be noted that FTP provides for very limited data type representations. Transformations desired beyond this limited capability should be performed by the user directly or via the use of the Data Reconfiguation Sevice (DRS, RFC #138, NIC #6715; and RFC #437, NIC #13701).

Additonal representation types may be defined later if there is a demonstrable need. 1c2b

Data representations are handled in FTP by a user specifying a representation type. This type may implicitly (as ASCII or EBCDIC) or explicitly (as in Local byte) define a byte size for interpretation which is referred to as the "logical byte size." This has nothing to do with the byte size used for transmission over the data connection(s) (called the"transfer byte size") and the two should not be confused. For example, NVT-ASCII has a logical byte size of 8 bits but an ASCII file might be transferred using a transfer byte size of 32. If the type is Local byte, then the TYPE command has an obligatory second parameter specifying the logical byte size. 1c2c

The types ASCII and EBCDIC also take a second (optional) parameter; this is to indicate what kind of vertical format control, if any, is associated with a file. The following data representation types are defined in FTP: 1c2d

ASCII Format 1c2d1

This is the default type and must be accepted by all FTP implementations. It is intended primarily for the transfer of text files, except when both hosts would find the EBCDIC type more convenient. 1c2d1a

The sender converts the data from his internal character representation to the standard 8-bit NVT-ASCII representation (see the Telnet specification). The receiver will convert the data from the standard form to his own internal form. 1c2d1b

In accordance with the NVT standard, the CRLF sequence should be used, where necessary, to denote the end of a line of text. (See the discussion of file structure at the end of Section 3B). 1c2d1c

Using the standard NVT-ASCII representation means that data must be interpreted as 8-bit bytes. If the BYTE command (Section IV,A,2) specifies a transfer byte size different from 8 bits, the 8-bit ASCII characters should be packed contiguously without regard for transfer byte boundaries. 1c2d1d

The Format parameter for ASCII and EBCDIC types is discussed below. (See Section 3B5.) 1c2d1e

EBCDIC Format 1c2d2

This type is intended for efficient transfeer between hosts which use EBCDIC for their internal character representation. 1c2d2a

For transmission the data are represented as 8-bit EBCDIC characters. The character code is the only difference between the functional specifications of EBCDIC and ASCII types. 1c2d2b

End-of-line (as opposed to end-of-record -- see the discussion of structure) will probably be rarely used with EBCDIC type for purposes of denoting structure, but where it is necessary the NL character should be used. 1c2d2c

The Format parameter for ASCII and EBCDIC types is discussed below. (See Section 3B5.) 1c2d2d

Image 1c2d3

The data are sent as contiguous bits which, for transfer, are packed into transfer bytes of the size specified in the BYTE command. The receiving site must store the data as contiguous bits. The structure of the storage system might necessitate the padding of the file (or of each record, for a record-structured file) to some convenient boundary (byte, word or block). This padding, which must be all zeroes, may occur only at the end of the file (or at the end of each record) and there must be a way of identifying the padding bits so that they may be stripped off if the file is retrieved. The padding transformation should be well publicized to enable a user to process a file at the storage site. 1c2d3a

Image type is intended for the efficient storage and retrieval of files and for the transfer of binary data. It is recommended that this type be accepted by all FTP implementations. 1c2d3b

Local-byte Byte-size 1c2d4

The data is transferred in logical bytes of the size specified by the obligatory second parrameter, Byte-size. The value of Byte-size must be a decimal integer; there is no defauult value. The logical byte size is not necessarily the same as the transfer byte

size. If there is a difference in byte sizes, then the logical bytes should be packed contiguously, disregarding transfer byte boundaries and with any necessary padding at the end. 1c2d4a

When the data reaches the receiving host it will be transformed in a manner dependent on the logical byte size and the particular host. This transformation must be invertible (that is an identical file can be retrieved if the same parameters are used) and should be well publicized by the FTP implementors. 1c2d4b

This type is intended for the transfer of structured data. For example, a user sending 36-bit floating-point numbers to a host with a 32-bit word could send his data as Local-byte with a logical byte size of 36. The receiving host would then be expected to store the logical bytes so that they could be easily manipulated; in this example putting the 36-bit logical bytes into 64-bit double words should suffice. 1c2d4c

A character file may be transferred to a host for one of three purposes: for printing, for storage and later retrieval or for processing. If a file is sent for printing, the receiving host must know how the vertical format control is represented. In the second case, it must be possible to store a file at a host and then retrieve it later in exactly the same form. Finally, it ought to be possible to move a file from one host to another and process the file at the second host without undue trouble. A single ASCII or EBCDIC format does not satisfy all these conditions and so these types have a second parameter specifying one of the following three formats: 1c2e

non print 1c2e1

This is the default format to be used if the second (format) parameter is omitted. Non-print format must be accepted by all FTP implementations. 1c2e1a

The file need contain no vertical format information. If it is passed to a printer process, this process may assume standard values for spacing and margins. 1c2e1b

Normally, this format will be used with files destined for processing or just storage. 1c2e1c

Telnet format controls 1c2e2

The file contains ASCII/EBCDIC vertical format controls (i.e., CR, LF, NL, VT, FF) which the printer process will interpret appropriately. CRLF, in exactly this sequence, also denotes end-of-line. 1c2e2a

Carriage Control (ASA) 1c2e3

The file contains ASA (Fortran) vertical format control characters. (See NWG/RFC #189 Appendix C and Communications of the ACM vol. 7 no. 10 p. 606, October 1964.) In a line or a record, formatted according to the ASA Standard, the first character is not to be printed. Instead it should be used to determine the vertical movement of the paper which should take place before the rest of the record is printed. The ASA Standard specifies the following control characters: 1c2e3a

Character	Vertical Spacing
blank	Move paper up one line
0	Move paper uo two lines
1	Move paper to top of page
+	No paper movement, overprint.

1c2e3a1

Clearly there must be some way for a printer process to distinguish the end of the structural entity. If a file has record structure (see below) this is no problem; records will be explicitly marked during transfer and storage. If the file has no record structure, the CRLF end-of-line sequence is used to separate printing lines, but these format effectors are overridden by the ASA controls. 1c2e3b

A note of caution about parameters: a file must be stored and retrieved with the same parameters if the retrieved version is to be identical to the version originally transmitted. Conversely, FTP implementations must return a file identical to the original if the parameters used to store and retrieve a file are the same. 1c2f

In addition to different representation types, FTP allows the structure of a file to be specified. Currently two file structures are recognized in FTP: file-structure, where there is no internal structure, and record-structure, where the file is made up of records. File-structure is the default, to be assumed if the STRUcture command has not been

used but both structures must be accepted for"text" files (i.e., files with TyPE ASCII or EBCDIC) by all FTP implementations. The structure of a file will affect both the transfer mode of a file (see Section 3D) and the interpretation and storage of the file. 1c2g

The "natural" structure of a file will depend on which host stores the file. A source-code file will usually be stored on an IBM 360 in fixed length records but on a PDP-10 as a stream of characters partitioned into lines, for example by CRLF. If the transfer of files between such disparate sites is to be useful, there must be some way for one site to recognize the other's assumptions about the file. 1c2h

With some sites being naturally file-oriented and others naturally record-oriented there may be problems if a file with one structure is sent to a host oriented to the other. If a text file is sent with record-structure to a host which is file oriented, then that host should apply an internal transformation to the file based on the record structure. Obviously this transformation should be useful but it must also be invertible so that an identical file may be retreieved using record structure. 1c2i

In the case of a file being sent with file-structure to a record-oriented host, there exists the question of what criteria the host should use to divide the file into records which can be processed locally. If this division is necessary the FTP implementation should use the end-of-line sequence, CRLF for ASCII, or NL for EBCDIC text files, as the delimiter. If an FTP implementation adopts this technique,it must be prepared to reverse the transformation if the file is retrieved with file-structure. 1c2j

Establishing Data Connections 1c3

The mechanics of transferring data consists of setting up the data connection to the appropriate sockets and choosing the parameters for transfer -- byte size and mode. Both the user and the server-DTPs have default data sockets; these are the two sockets (for send and receive) immediately following the standard ICP Telnet ,i.e., (U+4) and U+5) for the user-process and (S+2), (S+3) for the server. The use of default sockets will ensure the security of the data transfer, wiithout requiring the socket information to be explicitly exchanged. 1c3a

The byte size for the data connection is specified by the byte command, or, if left unspecified, defaults to 8-bit

bytes. This byte size is relevant only for the actual transfer of the data; it has no bearing on representation of the data within a host's file system. The protocol does not require servers to accept all possible byte sizes. Since the use of various byte sizes is intended for efficiency of transfer, servers may implement only those sizes for which their data transfer is efficient including the default byte size of 8 bits. 1c3b

The passive data transfer process (this may be a user-DTP or a second server-DTP) shall "listen" on the data socket prior to sending a transfer request command. The FTP request command determines the direction of the data transfer and thus which data socket (odd or even) is to be used in establilshing the connection. The server, upon receiving the transfer request, will initiate the data connection by RFC to the appropriate socket using the specified (or default) byte size. When the connection is opened, the data transfer begins between DTP's and the server-PI sends a confirming reply to the user-PI. 1c3c

It is possible for the user to specify an alternate data socket by use of the SOCK command. He might want a file dumped on a TIP line printer or retrieved from a third party host. In the latter case the user-PI sets up Telnet connectioins with both server-PI's and sends each a SOCK command indicating the fixed data sockets of the other. One server is then told (by an FTP command) to "listen" for an RFC which the other will initiate and finally both are sent the appropriate transfer commands. The exact sequence of commands and replies sent between the user-controller and the servers is defined in Section 5B. 1c3d

In general it is the server's responsibility to maintain the data connection -- to initiate the RFC's and the closes. The exception to this is when the user-DTP is sending the data in a transfer mode that requires the connection to be closed to indicate EOF. The server must close the data connection under the following conditions: 1c3e

The server has completed sending data in a transfer mode that requires a close to indicate EOF. 1c3e1

The server receives an ABORT command from the user. 1c3e2

The socket or byte size specification is changed by a command from the user. 1c3e3

The Telnet connections are closed legally or otherwise. 1c3e4

An irrecoverable error condition occurs. 1c3e5

Otherwise the close is a server option, the exercise of which he must indicate to the user-process by an appropriate reply. 1c3f

Transmission Modes 1c4

The next consideration in transferring data is choosing the appropriate transmission mode. There are three modes: one which formats the data and allows for restart procedures; one which also compressesthe data for efficient transfer; and one which passes the data with little or no processing. In this last case the mode interacts with the structure attribute to determine the type of processing. In the compressed mode the representation type determines the filler byte. 1c4a

All data transfers must be completed with an end-of-file (EOF) which may be explicitly stated or implied by the closing of the data connection. For files with record structure, all the end-of-record markers (EOR) are explicit,including the final one. 1c4b

Note: In the rest of this section, byte means "transfer byte" except where explicitly stated otherwise. 1c4c

The following transmission modes are defined in FTP: 1c4d

Stream 1c4d1

The data is transmitted as a stream of bytes. There is no restriction on the representation type used; record structures are allowed,in which case the transfer byte size must be at least 3 bits! 1c4d1a

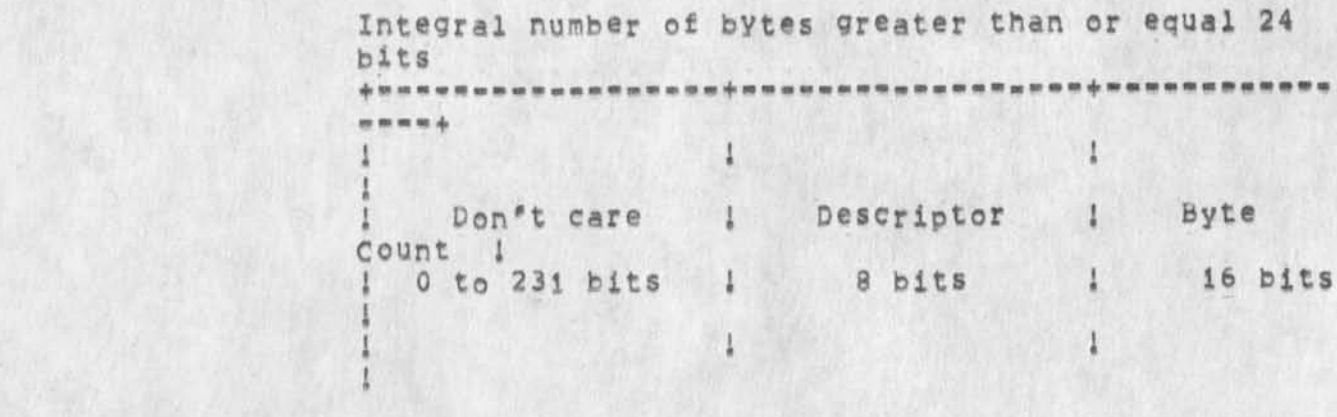
In a record structured file EOR and EOF will each be indicated by a byte control code of whatever byte size is used for the transfer. The first byte of the control code will be all ones,the escape character. The second byte will have the low order bit on and zeroes elsewhere for EOR and the second low order bit on for EOF; that is, the byte will have have value 1 for EO_R and value 2 for EOF. EO_R and EOF may be indicated together on the last byte transmitted by turning both low order bits on, i.e., the value 3. If a byte of all ones was intended to be sent as data, it should be repeated in the second byte of the control code. 1c4d1b

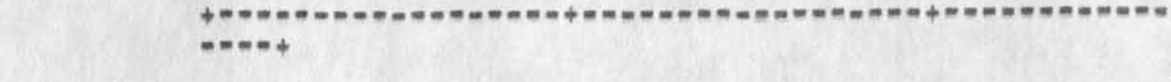
If the file does not have record structure, the EOF is indicated by the sending host closing the data connection and all bytes are data bytes.

Block

The file is transmittred as a series of data blocks preceded by one or more header bytes. The header bytes contain a count field, and descriptor code. The count field indicates the total length of the data block in bytes, thus marking the beginning of the next data block (there are no filler bits). The descriptor code defines: last block in the file (EOF) last block in the record (EOR), restart marker (see section 3E) or suspect data (i.e., the data being transferred is suspected of errors and is not relilable.) This last code is not intended for error control within FTP. It is motivated by the desire of sites exchanging certain types of data (e.g., seismic or weather data) to send and receive all the data despite local errors (such as "magnetic tape read errors"), but to indicate in the transmission that certain portions are suspect). Record structures are allowed in this mode, and any representation type may be used. There is no restriction on the transfer byte size. 1c4d2a

The header consists of the smallest integral number of bytes whose length is greater than or equal to 24 bits. Only the least significant 24 bits (right-justified) of header shall have information; the remaining most significant bits are "don't care" bits. Of the 24 bits of header information, the 16 low order bits shall represent byte count, and the 8 high order bits shall represent descriptor codes as shown below. 1c4d2b





1c4d2b1

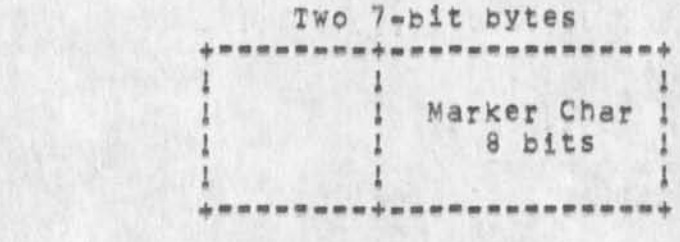
The descriptor codes are indicated by bit flags in the descriptor byte. Four codes have been assigned, where each code number is the decimal value of the corresponding bit in the byte. 1c4d2c

Code	Meaning
128	End of data block is EOR
64	End of data block is EOF
32	Suspected errors in data block
16	Data block is a restart marker

1c4d2c1

With this encoding more than one descriptor coded condition may exist for a particular block. As many bits as necessary may be flagged. 1c4d2d

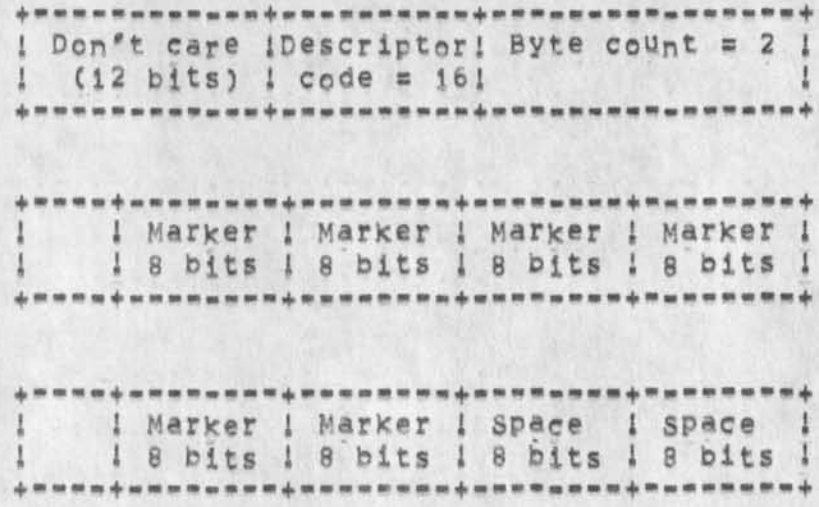
The restart marker is embedded in the data stream as an integral number of 8-bit bytes representing printable characters in the language being used over the Telnet connection (e.g., default -- NVT -- ASCII). These marker bytes are right-justified in the smallest integral number of transfer bytes greater than or equal to 8 bits. For example, if the byte size is 7 bits the restart marker byte would be one byte right-justified per two 7-bit bytes as shown below: 1c4d2e



1c4d2e1

If the transfer byte size is 16 or more bits, the maximum possible number of complete marker bytes should be packed,right-justified, into each transfer byte. The restart marker should begin in the first marker byte. If there are any unused marker bytes, these should be filled with the character SP (Space,in the appropriate language). SP must not be used within

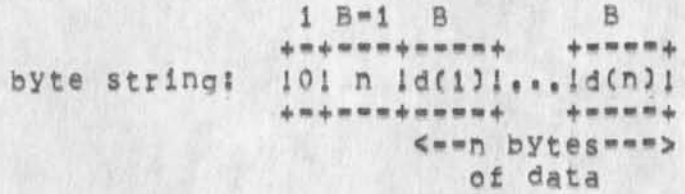
a restart marker. For example, to transmit a six-character marker with a 36-bit transfer byte size, the following three 36-bit bytes would be sent: 1c4d2f



1c4d2f1

Compressed 1c4d3

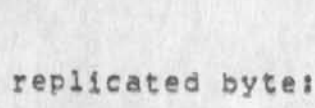
The file is transmitted as series of bytes of the size specified by the BYTE command. There are three kinds of information to be sent: regular data, sent in a byte string; compressed data, consisting of replications or filler; and control information, sent in a two-byte escape sequence. If the byte size is B bits and n>0 bytes of regular data are sent, these n bytes are preceded by a byte with the left-most bit set to 0 and the right-most B-1 bits containing the number n. 1c4d3a



string of n data bytes d(1),..., d(n)
count n must be positive

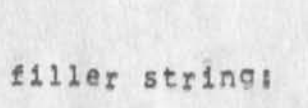
1c4d3a1

To compress a string of n replications of the data byte d, the following 2 bytes are sent: 1c4d3b



1c4d3b1

A string of n filler bytes can be compressed into a single byte, where the filler byte varies with the representation type. If the type is ASCII or EBCDIC the filler byte is SP (Space,ASCII code 32,, EBCDIC code 64,) If the transfer byte size is not 8, the expanded byte string should be filled with 8-bit SP characters in the manner described in the definition of ASCII representation type (Section 3B). If the type is Image or Local byte the filler is a zero byte, 1c4d3c



1c4d3c1

The escape sequence is a double byte, the first of which is the escape byte (all Zeroes) and the second of which contains descriptor codes as defined in Block mode. This implies that the byte size must be at least 8 bits, which is not much of a restriction for efficiency in this mode. The descriptor codes have the same meaning as in Block mode and apply to the succeeding string of bytes. 1c4d3d

Compressed mode is useful for obtaining increased bandwidth on very large network transmissions at a little extra CPU cost. It is most efficient when the byte size chosen is that of the word size of the transmitting host, and can be most effectively used to reduce the size of printer files such as those generated by RJE hosts. 1c4d3e

For the purpose of standardized transfer, the sending host will translate his internal end of line or end of record denotation into the representation prescribed by the transfer mode and file structure, and the receiving host will perform the inverse translation to his internal denotation. An IBM 360 record count field may not be recognized at another host, so the end of record information may be transferred as a two byte control code in Stream mode or as a flagged bit in a Block or Compressed mode descriptor. End of line in an ASCII or EBCDIC file with no record structure should be indicated by CRLF or NL, respectively. Since these transformations imply extra work for some systems, identical systems transferring non-record structured text files might wish to use a binary representation and stream mode for the transfer. 1c4e

Error Recovery and Restart 1c5

There is no provision for detecting bits lost or scrambled in data transfer. This issue is perhaps handled best at the NCP level where it benefits most users. However, a restart procedure is provided to protect users from gross system failures (including failures of a host, an FTP-process, or the IMP subnet). 1c5a

The restart procedure is defined only for the block and compressed modes of data transfer. It requires the sender of data to insert a a special marker code in the data stream with some marker information. The marker information has meaning only to the sender, but must consist of printable characters in the default or negotiated language of the Telnet connection. The marker could represent a bit-count, a record-count, or any other information by which a system may identify a data checkpoint. The receiver of data,if it implements the restart procedure, would then mark the corresponding position of this marker in the recieving system, and return this information to the user. 1c5b

In the event of a system failure, the user can restart the data transfer by identifying the marker point with the FTP restart procedure. The following example illustrates the use of the restart procedure. 1c5c

The sender of the data inserts an appropriate marker block in the data stream at a convenient point. The receiving host marks the corresponding data point in its file system and conveys the last known sender and receiver marker information to the user, either directly or over the Telnet connection in a 251 reply (depending on who is the sender).

In the event of a system failure, the user or controller process restarts the server at the last server marker by sending a restart command with server's marker code as its argument. The restrart command is transmitted over the Telnet connection and is immediately followed by the command (such as RETR, STOR or LIST) which was being executed when the system failure occurred. 1c5d

FILE TRANSFER FUNCTIONS 1d

The communication channel from the user-PI to the server-PI is established by ICP from the user to a standad server socket. The user protocol interpreter is responsible for sending FTP commands and interpreting the replies received; the server-PI interprets commands, sends replies and directs its DTP to set up the data connection and transfer the data. If the second party to the data transfer (the passive transfer process) is the user-DTP then it is governed through the internal protocol of the user-FTP host; if it is a second server-DTP then it is governed by its PI on command from the user-PI. 1d1

FTP Commands 1d2

General Principles 1d2a

The File Transfer Protocol follows the specifications of the Telnet protoccol for all communications over the Telnet connection - see NIC #7104. Since, in the future, the language used for Telnet communication may be a negotiated opton, all references in the next two sections will be to the "Telnet end of line code", currently one may take these to mean NVT-ASCII and CRLF. No other specifications of the Telnet protocol will be cited. 1d2a1

FTP commands are "Telnet strings" terminated by the "Telnet end of line code". The command codes themselves are alphabetic characters terminated by the character SP (Space) if parameters follow and Telnet-EOL otherwise. The command codes and the semantics of commands are described in this section; the detailed syntax of commands is specified in Section 5C, the reply sequences are discussed in Section 5D, and scenarios illustrating the use of commands are provided in Section 6. 1d2a2

FTP commands may be partitioned as those specifying access-control identifiers, data transfer parameters, or FTP service requests. Certain commands (such as ABOR, STAT, BYE) may be sent over the Telnet connections while a data transfer is in progress. Some servers may not be

able to monitor the Telnet and data connectioons simultaneously, in which case some special action will be necessary to get the server's attention. The exact from of the "special action" is related to decisions currently under review by the Telnet committee; but the following ordered format is tentatively recommended: 1d2a3

User system inserts the Telnet "Interrupt Process" (IP) signal in the Telnet stream. 1d2a3a

User system sends the Telnet "Synch" signal 1d2a3b

User system inserts the command (e.g., ABOR) in the Telnet stream. 1d2a3c

Server PI,, after receiving "IP", scans the Telnet stream for exactly one FTP command. 1d2a3d

(For other servers this may not be necessary but the actions listed above should have no unusual effect.) 1d2a4

Access Control Commands 1d2b

The following commands specify access control identifiers (command codes are shown in parentheses). 1d2b1

User Name (USER) 1d2b1a

The argument field is a Telnet string identifying the user. The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the Telnet connections are made (some servers may require this). Additional identification information in the form of a password and/or an account command may also be required by some servers. Servers may allow a new USER command to be entered at any point in order to change the access control and/or accounting information. This has the effect of flushing any user, password and account information already supplied and beginning the login sequence again. All transfer parameters are unchanged and any file transfer in progress is completed under the old acccount. 1d2b1a1

Password (PASS) 1d2b1b

The argument field is a Telnet string identifying

the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress typeout. It appears that the server has no foolproof way to achieve this. It is therefore the responsibility of the user-FTP process to hide the sensitive password information. 1d2b1b1

Account (ACCT) 1d2b1c

The argument field is a Telnet string identifying the user's account. The command is not necessarily related to the USER command, as some sites may require an account for login and others only for specific access, such as storing ffiles. In the latter case the command may arrive at any time. There are two reply codes to differrentiate these cases for the automaton: when account information is required for login, the response to a successful PASSword command is reply code 331; then if a command other than ACCounT is sent, the server may remember it and return a 331 reply, prepared to act on the command after the account information is received; or he may flush the command and return a 433 reply asking for the account. On the other hand, if account information is not required for login, the reply to a successful PASSword command is 230; and if the information is needed for a command issued later in the dialogue, the server should return a 331 or 433 reply depending on whether he stores (pending receipt of theACCounT command) or discards the command, respectively. 1d2b1c1

Reinitialize (REIN) 1d2b1d

This command terminates a USER, flushing all I/O and account information, except to allow any transfer in progress to be completed. All parameters are reset to the default settings and the Telnet connection is left open. This is identical to the state in which a user finds himself immediately after the ICP is completed and the Telnet connections are opened. A USER command may be expected to follow. 1d2b1d1

Logout (ByE) 1d2b1e

This command terminates a USER and if file transfer is not in progress, the server closes the Telnet connection. If file transfer is in progress, the connection will remain open for result response and the server will then close it. If the user-process is transferring files for several USERs but does not wish to close and then reeopen connections for each, then the REIN command should be used instead of BYE. 1d2b1e1

An unexpected close on the Telnet connection will cause the server to take the effective action of an abort (ABOR) and a logout (BYE). 1d2b1e2

Transfer Parameter Commands 1d2c

All data transfer parameters have default values, and the commands specifying data transfer parameters are required only if the default parameter values are to be changed. The default value is the last specified value, or if no value has been specified, the standard default value as stated here. This implies that the server must "remember" the applicable default values. The commands may be in any order except that they must precede the FTP service request. The following commands specify data transfer parameters. 1d2c1

Byte size (BYTE) 1d2c1a

The argument is a decimal integer (1 through 255) specifying the byte size for the data connection. The default byte size is 8 bits. A server may reject certain byte sizes that he has not implemented. 1d2c1a1

Data socket (SOCK) 1d2c1b

The argument is a host-socket specification for the data socket to be used in data connection. There may be two data sockets, one for transfer from the "active" DTP to the "passive" DTP and one for "passive" to "active". An odd socket number defines a send socket and an even socket number defines a receive socket. The default host is the user host to which Telnet connections are made. The default data sockets are (U+4) and (U+5) where U is the socket number used in the Telnet ICP and the Telnet connections are on sockets (U+2) and (U+3). The server has fixed data soickets (S+2) and (S+3) as

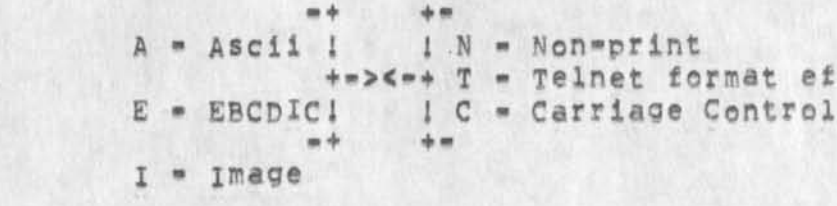
well, and under normal circimstances this command and its reply are not needed. 1d2c1b1

Passive (PASV) 1d2c1c

This command requests the server-DTP to "listen" on both of his data sockets and to wait for an RFC to arrive for one socket rather than initiate one upon receipt of a transfer command. It is assumed the server has already received a SOCK command to indicate the foreign socket from which the RFC will arrive to ensure the security of the transfer. 1d2c1c1

Representation Type (TYPE) 1d2c1d

The argument specifies the representation type as described in Section 3B. Several types take a second parameter. The first parameter is denoted by a single Telnet character, as is the second Format parameter for ASCII and EBCDIC; the second parameter for local byte is a decimal integer to indicate byte size. The following codes are assigned for type: 1d2c1d1



1d2c1d1a

The default representation type is ASCII Non-Print. If the Format parameter is changed, and later just the first argument is changed, Format then returns to the Non-print default. 1d2c1d2

File Structure (STRU) 1d2c1e

The argument is a single Telnet character code specifying file structure described in Section 3B. The following codes are assigned for structure: 1d2c1e1

F - File (no record structure)
R - Record structure 1d2c1e1a

The default structure is File (i.e., no records). 1d2c1e2

Transfer Mode (MODE) 1d2c1f

The argument is a single Telnet character code specifying the data transfer modes described in Section 3D. The following codes are assigned for transfer modes: 1d2c1f1

S - Stream
B - Block
C - Compressed 1d2c1f1a

The default transfer mode is Stream. 1d2c1f2

FTP Service Commands 1d2d

The FTP service commands define the file transfer or the file system function requested by the user. The argument of an FTP service command will normally be a pathname. The syntax of pathnames must conform to server site conventions (with standard defaults applicable), and the language conventions of the Telnet connection. The suggested default handling is to use the last specified device, directory or file name, or the standard default defined for local users. The commands may be in any order except that a "rename from" command must be followed by a "rename to" command and the restart command must be followed by the interrupted service command. The data, when transfered in response to FTP service commands, shall always be sent over the data connection, except for certain informative replies. The following comands specify FTP service requests: 1d2d1

Retrieve (RETR) 1d2d1a

This command causes the server-DTP to transfer a copy of the file, specified in the pathname, to the server - or user-DTP at the other end of the data connection. The status and contents of the file at the server site shall be unaffected. 1d2d1a1

Store (STOR) 1d2d1b

This command causes the server-DTP to accept the

data transfered via the data connection and to store the data as a file at the server site. If the file specified in the pathname exists at the server site then its contents shall be replaced by the data being transfered, A new file is created at the server site if the file specified in the pathname does not already exist. 1d2d1b1

APPend (with create) (APPE) 1d2d1c

This command causes the server-DTP to accept the data transfered via the data connection and to store the data in a file at the server site. If the file specified in the pathname exists at the server site, then the data shall be appended to that file; otherwise the file specified in the pathname shall be created at the server site. 1d2d1c1

Allocate (ALLO) 1d2d1d

This command may be required by some servers to reserve sufficient storage to accommodate the new file to be transfered. The argument shall be a decimal integer representing the number of bytes (using the logical byte size) of storage to be reeserved for the file. For files sent with record strcture a maximum record size (in logical bytes) might also be necessary; this is indicated by a decimal integer in a second argument field of the command. This second argument is optional, but when present should be separated from the first by the three Telnet characters SP R SP. This command shall be followed by a STORe or APPEnd command. The ALLO command should be treated as a NOOP (no operation) by those servers which do not require that the maximum size of the file be declared beforehand, and those servers interested in only the maximum record size should accept a dummy value in the first argument and ignore it. 1d2d1d1

Restart (REST) 1d2d1e

The argument field represents the server marker at which file transfer is to be restarted. This command does not cause file transfer but "spaces" over the file to the specified data checkpoint. This command shall be immediately followed by the appropriate FTP service command which shall cause file transfer to resume. 1d2d1e1

Rename from (RNFR) 1d2d1f

This command specifies the file which is to be renamed. This command must be immediately followed by a "rename to" command specifying the new file pathname. 1d2d1f1

Rename to (RNTO) 1d2d1g

This command specifies the new pathname of the file specified in the immediately preceding "rename from" command. Together the two commands cause a file to be renamed. 1d2d1g1

Abort (ABOR) 1d2d1h

This command indicates to the server to abort the previous FTP service command and any associated transfer of data. The abort command may require "special action", as discussed in Section 4B, to force recognition by the server. No action is to be taken if the previous command has been completed (including data transfer). The Telnet connections are not to be closed by the server, but the data connection must be closed. An appropriate reply should be sent by the server in all cases. 1d2d1h1

Delete (DELE) 1d2d1i

This command causes the file specified in the pathname to be deleted at the server site. If an extra level of protection is desired (such as the query, "DO you really wish to delete?"), it should be provided by the user-FTP. 1d2d1i1

List (LIST) 1d2d1j

This command causes a list to be sent from the server to the passive DTP. If the pathname specifies a directory, the server should transfer a list of files in the specified directory. If the pathname specifies a file then the server should send current information on the file. A null agument implies the user's currrent working or default directory. The data transfer is over the data connection in type ASCII or type EBCDIC. (The user must ensure that the TYPE is appropriattely ASCII or EBCDIC>) 1d2d1j1

Name-List 1d2d1k

(NLST) - This command causes a directory listing to be sent from server to user site. The pathname should specify a directory or other system-specific file group descriptor; a null argument implies the current directory. The server will return a stream of names of files and no other information. The data will be transferred in ASCII or EBCDIC type over the data connection as valid pathname strings separated by CRLF or NL. (Again the user must ensure that the TYPE is correct.) 1d2d1k1

Site Parameters (SITE) 1d2d1l

This command is used by the server to provide services specific to his system that are essential to file transfer but not sufficiently universal to be included as commands in the protocol. The nature of these sevices and the specification of their syntax can be stated in a reply to the HELP SITE command. 1d2d1l1

Status (STAT) 1d2d1m

This command shall cause a status response to be sent over the TELENT connection in the form of a reply. The command may be sent during a file transfer (along with the Telnet IP and Synch signals -- see Section 4B) in which case the server will respond with the status of the operation in progress, or it may be sent between file transfers. In the latter case the command may have an argument field. If the argument is a pathname, the command is analogous to the "list" command except that data shall be trasfereed over the Telnet connection. If a partial pathname is given, the server may respond with a list of file names or attributes associated with that specification. If no arrgument is given, the server should return general status information about the server FTP process. This should include current values of all transfer parameters and the status of connections. 1d2d1m1

Help (HELP) 1d2d1n

This command shall cause the server to send helpful information regarding its implementation status over the Telnet connection to the user. The command

may take an argument (e.g., any command name) and return more specific information as a response. The reply is type 0xx, general system status. It is suggested that HELP be allowed before entering a USER command. The server may use this reply to specify site-dependent parameters, e.g., in response to HELP SITE. 1d2d1n1

NoOp (NOOP) 1d2d1o

This command does not affect any parrameters or previosly entered commands. It specifies no action other than that the server send a 200 reply. 1d2d1o1

Miscellaneous Commands 1d2e

There arre several functions that utilize the services of file transfer but go beyond it in scope. These are the Mail and Remote Job Entry functions. It is suggested that these become auxiliary protocols that can assume recognition of file transfer commands on the part of the server, i.e., they may depend on the core of FTP commands. The command sets specific to Mail and RJE will be given in separate documents. 1d2e1

Commands that are closely related to file transfer but not proven essential to the protocol may be implemented by servers on an experimental basis. The command name should begin with an X and may be listed in the HELP command. The official command set is expandable from these experiments; all experimental commands or proposals for expanding the official command set sdhould be announced via RFC. An example of a current experimental command is: 1d2e2

Change Working Directory (XCWD) 1d2e2a

This command allows the user to work with a different directory or dataset for file storage or retrieval without altering his login or accounting information. Transfer parameters are similarly unchanged. The argument is a pathname specifying a directory or other system dependent file group designator. 1d2e2a1

FTP Replies 1d3

General principles 1d3a

The server sends FTP replies over the Telnet connection in response to user FTP commands. The FTP replies constitute the acknowledgment or completion code (including errors). The FTP-server replies are formatted for human or program interpretation. Single line replies consist of a leading three-digit numeric code followed by a space, followed by a one-line text explanation of the code. For replies that contain several lines of text, the first line will have a leading three-digit numeric code followed immediately by the character "-" (Hyphen, ASCII code 45.), and possibly some text. All succeeding continuation lines except the last are constrained not to begin with three digits; the last line must repeat the numeric code of the first line and be followed immediately by a space. For example: 1d3a1

```
100-First Line
Continuation Line
Another Line
100 Last Line
```
1d3a1a

It is possible to nest (but not overlap) a reply withiin a multi-line reply. The same format for matched number-coded first and last lines holds. 1d3a2

The numerric codes are assigned by groups and for ease of interpretation by programs in a manner consistent with other protocols such as the RJE protocol. The three digits of the code are to be interpreted as follows: 1d3a3

Each Telnet line delimited by a numeric code and the TELENT EOL (or group of text lines bounded by coded lines) that is sent by the server is intended to be a complete reply message. It should be noted that the text of replies is intended for a human user. Only the reply codes and in some instances the first line of text are intended for programs. 1d3a4

The assigned reply codes relating to FTP are: 1d3b

DECLARATIVE sPECIFICATIONs 1e

Minimum Implementation 1e1

In order to make FTP workable without needless error messages, the followng minimum implementation is required for servers: 1e1a

```
TYPE - ASCII Non-print
MODE - Stream
STRUCTURE - File
            Record
BYTE - 8
COMMANDS - USER, BYE, SOCK,
           TYPE, BYTE, MODE, STRU,
             for the default values
           RETR, STOR,
           NOOP.
```

1e1a1

The initial default values for transfer parameters are: 1e1b

```
TYPE - ASCII Non-print
BYTE - 8
MODE - Stream
STRU - File
```

1e1b1

All hosts must accept the above as the standard defaults. 1e1c

Connections 1e2

The server Protocol interPreter shall "listen" on Socket 3. The user or user protocol interpreter shall initiate the full-duplex Telnet connections performing the ARPANET standard initial connection protocol (ICP) to server socket 3. Server- and user- processes should follow the conventions of the Telnet protocol as specified in NIC #7104. Servers are under no obligation to provide for editing of command lines and may specify that it be done in the user host. The Telnet connections shall be closed by the server at the user's request after all transfers and replies are completed. 1e2a

The user-DTP must "listen" on the specified data sockets (send and/or receive); these may be the default user sockets (U+4) and (U+5) or a socket specified in the SOCK command. The server shall initiate the data connection from his own fixed sockets (S+2) and (S+3) using the specified user data socket and byte size (default - 8 bits). The direction of the transfer and the sockets used will be determined by the FTP service command. 1e2b

When data is to be transferred between two servers, A and B (refer to the diagram in Section 2), the user-PI, C, sets up

Telnet connections with both server-PI's. He then sends A's fixed sockets, SA, to B in a SOCK command and B's to A; replies are returned. One of the servers, say A, is then sent a PASV command telling him to "listen" on his data sockets rather than initiate an RFC when he receives a transfer service command. When the user-PI receives an acknowledement to the PASV command, he may send (in either order) the corresponding service commands to A and B. Server B initiates the RFC and the transfer proceeds. The command-reply sequence is listed below where the messages are vertically synchronous but horizontally asynchronous: 1e2c

User-PI - Server A	User-PI - Server B
C>A : ICP	C>B : ICP
C>A : SOCK HOST-B, SKT-S(B) SKT-S(A)	C>B : SOCK HOST-A,
A>C : 200 Okay	B>C : 200 Okay
C>A : PASV	
A>C : 200 Okay	
C>A : STOR	C>B : RETR

1e2c1

The data connection shall be closed by the server under the conditions described in Section 3c. If the server wishes to close the connection after a transfer where it is not required, he should do so immediately after the file transfer is completed. He should not wait until after a new transfer command is received because the user-process will have already tested the data connection to see if it needs to do a "listen"; (recall that the user must "listen" on a closed data socket before sending the transfer request). To prevent a race condition here, the server sends a secondary reply (257) after closing the data connection (or if the connection is left open, a "file transfer completed" reply (252) and the user-PI should waait for one of these replies before issuing a new transfer command. 1e2d

Commands 1e3

The commands are Telnet character string transmitted over the Telnet connections as described in Section 4B. The command and semantics are described in Section5C. The command syntax is specified here. 1e3a

The commands begin with a command code followed by an argument field. The command codes are four or fewer

alphabetic characters. Upper and lower case characters are to be treated identically. Thus any of the following may represent the retrieve command: 1e3b

RETR Retr retr ReTr rETr 1e3b1

This also applies to any sumbols representing parameter values, such as A or a for ASCII type. The command codes and the argument fields are separated by one or more spaces. 1e3c

The argument field consists of a variable length character string ending with the character sequence CRLF (Carriage Return, Linefeeed) for NVT-ASCII representation; for other negotiated languages a different end of line character might be used. It should be noted that the server is to take no action until the end of line code is received. 1e3d

The syntax is specified below in NVT-ASCII. All characters in the argument field are ASCII characters including any ASCII represented decimal integers. Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied. 1e3e

The following are all the currently defined FTP commmands: 1e3f

USER SP <username> CRLF 1e3f1

PASS SP <password> CRLF 1e3f2

ACCT SP <acctno> CRLF 1e3f3

REIN CRLF 1e3f4

BYTE SP <byte size> CRLF 1e3f5

SOCK SP <host-socket> CRLF 1e3f6

PASV CRLF 1e3f7

TYPE SP <type code> CRLF 1e3f8

STRU SP <structure code> CRLF 1e3f9

MODE SP <mode code> CRLF 1e3f10

RETR SP <pathname> CRLF 1e3f11

STOR SP <pathname> CRLF 1e3f12

APPE SP <pathname> CRLF 1e3f13

ALLO SP <decimal integer> [SP R SP <decimal integer>] CRLF 1e3f14

REST SP <marker> CRLF 1e3f15

RNFR SP <pathname> CRLF 1e3f16

RNTO SP <pathname> CRLF 1e3f17

ABOR CRLF 1e3f18

DELE SP <pathname> CRLF 1e3f19

LIST [SP <pathname>] CRLF 1e3f20

NLST [SP <pathname>] CRLF 1e3f21

SITE SP <string> CRLF 1e3f22

STAT [SP <pathname>] CRLF 1e3f23

HELP [SP <string>] CRLF 1e3f24

NOOP CRLF 1e3f25

The syntax of the above argument fields (using BNF notation where applicable) is: 1e3g

<username> ::= <string> 1e3g1

<password> ::= <string> 1e3g2

<acctno> ::= <string> 1e3g3

<string> ::= <char>|<char><string> 1e3g4

<char> ::= any of the 128 ASCII characters except CR and LF 1e3g5

<marker> ::= <pr string> 1e3g6

<pr string> ::= <pr char>|<pr char><pr string> 1e3g7

<pr char> ::= any ASCII code 33, through 126,, printable characters 1e3g8

<byte size> ::= any decimal integer 1 through 255 1e3g9

<host-socket> ::= <socket>|<host number>, <socket> 1e3g10

<host-number> ::= a decimal integer specifying an ARPANET host. 1e3g11

<socket> ::= decimal integer between 0 and (2::32)-1 1e3g12

<form code> ::= N|T|C 1e3g13

<type code> ::= A[SP <form code>]|E [SP <form code>]|I| 1e3g14

1 SP <byte size> 1e3g15

<structure code> ::= F|R 1e3g16

<mode code> ::= S|B|C 1e3g17

<pathname> ::= <string> 1e3g18

Sequencing of Commands and Replies 1e4

The communication between the user and server is intended to be an alternating dialogue. As such, the user issues an FTP command and the server responds with a prompt primary reply. The user should wait for this initial primary success or failure response before sending further commands. 1e4a

Certain commands require a second reply for which the user should also wait. These replies may, for example, report on the progress or completion of file transfer or the closing of the data connection. They are secondary replies to file transfer commands. 1e4b

The third class of replies are informational and spontaneous replies which may arrive at any time. The user-PI should be prepared to receive them. These replies are listed below as spontaneous. 1e4c

One important group of spontaneous replies is the connection greetings. Under normal circumstances, a server will send a 300 reply, "awaiting input", when the ICP is completed. The user should wait for this greeting message before sending any commands. If the server is unable to accept input right away, he should send a 000 "announcing FTP" or a 020 "expected delay" reply immediately and a 300 reply when ready. The user will then know not to hang up if there is a delay. 1e4d

The table below lists alternative success and failure

replies for each command. These must be strictly adhered to; a server may substitute text in the replies, but the meaning and action implied by the code numbers and by the specific command reply sequence cannot be altered. 1e4e

COMMAND-REPLY CORRESPONDENCE TABLE 1e4f

FTP SCENARIOS 1f

TIP User wanting to transfer file from host X to local Printer: 1fi

a) TIP user opens Telnet connections by ICP to host X socket 3. 1fia

b) The following commands and replies are exchanged: 1fib

TIP host X 1fic

<------- 300 Awaiting input CRLF 1fid

USER username CRLF -------> 1fie

<------- 330 Enter Password CRLF 1fif

PASS passwsord CRLF -------> 1fig

<------- 230 User logged in CRLF 1fih

SOCK 65538 CRLF -------> 1fii

<------- 200 Commmand received OK CRLF 1fij

RETR this,file CRLF -------> 1fik

(host X initiates data connection to TIP socket 65538, i.e., PORT 1 receive) 1fil

<------- 250 File transfer started CRLF 1fim

<------- 252 file transfer completed CRLF 1fin

BYE CRLF -------> 1fio

<------- 231 User logged out CRLF 1fip

c) host X closes the Telnet and data connections. 1fiq

Note: The TIP user should be in line mode. 1fir

1f1s

User at host U wanting to transfer files to/from host S: 1f2

In general the user will communicae to the server via a mediating user-FTP process. The following may be a typical scenario. The user-FTP prompts are shown in parentheses, '----->' represents commands from host U to host S, and '<-----' represents replies from host S to host U. 1f2a

JBP 24-OCT-74 13:36 24301

FTPSPEC

(J24301) 24-OCT-74 13:36;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;

MTR6722

<POSTEL>MTR6722,NLS;1, 26-SEP-74 13:12 JBP ;
Survey of Network Control Programs in the ARPA Network

1.0 Introduction

1.1 Purpose of this report

This report describes the function of a Network Control Program (NCP) in the Advanced Research Projects Agency Computer Network (ARPANET), and surveys several representative implementations of such programs. This work was accomplished as part of MITRE task 810A.

The network control program is the operating system module that interfaces user programs to the communications network by providing system calls to invoke communications functions specified by the network wide host to host protocol.

The purpose of this report is to document the implementation of the network control program and the optional strategies used in different implementations. This report should be of use to individuals implementing network control programs for other computers, and to individuals designining network control programs for other computer networks.

The survey investigated network control program implementations in eleven systems: TENEX at Bolt Beranek and Newman (BBN); Multics at Massachusetts Institute of Technology (MIT); the IBM systems at University of California, Santa Barbara (UCSB), University of Califoria, Los Angeles (UCLA), The RAND Corporation (RAND), Systems Development Corporation (SDC); the Burroughs system at University of California, San Diego (UCSD); the DEC system at the center for computer-based Behavorial Studies (CCBS) at UCLA; the Control Data system at Lawerence Berkeley Laboratory (LBL); the ARPA Network Terminal System (ANTS) at University of Illinois(UI); and the Terminal Interface Processor (TIP) at BBN.

The following table summarizes the organizations (places), operating systems, and computer hardware included in the survey.

Place	System	Computer
BBN	TENEX	DEC PDP 10
MIT	Multics	H 6180
UCSB	OS/MVT	IBM 360/75
UCLA	OS/MVT	IBM 360/91
RAND	OS/MVT	IBM 370/158
SDC	VM	IBM 370/145
UCSD	MCP	B 6700
CCBS	DEC	DEC PDP 10
LBL	BKY	CDC 6600
UI	ANTS	DEC PDP 11
BBN	TIP	H 316

1.2 ARPANET Overview

The ARPANET is an advanced computer communications system connecting together a set of computer centers in the United States and Europe. The following description of the network touches on several aspects: the physical implementation, the scope and size, and the functional goals.

The ARPANET is implemented using packet transmission technology. At each network site there is an Interface Message Processor (IMP) which is a store and forward packet routing computer. Each IMP is connected to between 1 and 5 other IMP's via common carrier circuits. These circuits are normally 50 kilobit per second channels. The IMP's are Honeywell 316 or 516 computers modified and programed by Bolt, Beranek, and Newman (BBN) <Heart>. Also connected to an IMP may be between 0 and 4 hosts. A host is a computing sysyem which currently ranges in size from a DEC PDP11 to a IBM 370/195. A Terminal Interface Processor (TIP) is an IMP and a minihost combined in one processor <Ornstein>.

The ARPANET has grown almost continuously from the time it began with the installation of the first IMP in September 1969. As of this writing the network consists of 48 IMPs of which 21 are

TIPs, and 54 hosts. The network extends from Hawaii in the west to London and Norway in the east. One communication channel has been upgraded to 230.4 kilobits per second, and the communications between California and Hawaii, and between the United States and Europe are via satellite channels. The satellite channel to Europe is 7.2 kilobits per second, while the satellite channel to Hawaii is the normal 50 kilobits per second.

The goal of the ARPA computer network is for each computer to make every local resource available to any computer in the network in such a way that any local program available to local users can be used remotely without degradation. That is, any program should be able to call on the resources of other computers much as it would call a subroutine. The resources which can be shared in this way include software and data, as well as hardware <Roberts>.

The process of sucessful communication requires the use of some rules of behavior in order to permit the communicating entities to properly interpret the conversation. These rules of behavior may include both constraints on the sequencing of the units of conversation, as well as the structure and content of the communication (e.g. the grammar and the meaning). In the ARPANET these rules of communication behavior are called protocols.

Communications in the ARPANET are of two types; those associated with two directly connected entities (e.g. IMP to IMP, IMP-host, system-process) and those between more widely separated entities (e.g. system to system, process to process). This second type of communication is sometimes said to be supported by a virtual communications channel. For example the virtual process to process communication channel is really a process-system, host-IMP, IMP to IMP, IMP-host, system-process channel.

The protocols in the ARPANET build up the capabilities of the network in a series of levels or layers. The lowest of these is the IMP to IMP protocol which provides for reliable communication among the IMPs. This protocol

handles transmission error detection and correction, flow control to avoid congestion, and routing.

The next level is the IMP-host protocol which provides for the passage of messages between hosts and IMPs in such a way as to create virtual communication paths between the hosts. With the IMP-host protocol, a host has operating rules which permit it to send messages to specified hosts on the network and to be informed of the dispensation of those messages. In particular, the IMP-host protocol constrains the hosts in their transmissions in order to make make good use of available communications capacity without denying such availability to other hosts.

The next higher level is the host to host protocol, implemented by the network control program. The host to host protocol is the set of rules whereby hosts construct and maintain communications between user processes running on separated computer systems. One process requiring communications with another on some remote computer system makes requests on its local operating system to act on its behalf in establishing and maintaining those communications using the host to host protocol <Crocker>.

If this brief introduction to the ARPANET is not sufficient, the reader is urged to turn to the references <Heart>, <Roberts>, <Crocker>, and <Ornstein>, before reading the body of this report.

1.3 Report Organization

The report is organized to first discuss the general characteristics of operating systems and network control programs, and then to discuss the specific implementations surveyed.

First is a discussion of the purpose of a network control program. To place this in some context the relevant characteristics of an operating system are discussed.

A network control program is then discussed in abstract terms, that is without reference to any particular machine or operating system. This description of an idealized NCP is then used as a reference in the descriptions of the surveyed systems.

The findings of the survey are presented next, first in terms of the system structure and NCP implementation strategy, and finally in terms of the physical characteristics of the various implementations.

The final section summarizes the findings of the survey.

At the end of the report there is a glossary of terms and a list of references.

2.0 The Network Control Program (NCP) Environment

This section discusses the purpose of the NCP and how it relates to the operating system and other protocols.

2.1 Purpose of a Network Control Program

The function of network control program (NCP) is to implement the host to host protocol. That is the NCP is to provide a common interface across the various operating systems to the user level processes, and to provide to the operating systems the means to communicate among themselves the control information necessary to establish, regulate, and terminate the communication between user processes.

The development of the network in the computer science research environment provided an ample collection of well developed interactive timesharing systems for the initial set of systems to become part of the network. These early interactive systems generally possessed a process structure and interprocess communications mechanism. It seemed that the network should naturally extend these interprocess communications mechanisms to allow communication between processes in different systems. There were some difficulties, however, for the various systems did not all have a

uniform scheme for interprocess communication or even a common way of naming the destination of a communique. Several of the systems that were later added to network did not allow communication between processes, indeed, a few systems did not support the concept of a process (or so their programmers claimed).

A standard interprocess communication mechanism and a standard naming scheme were needed. Further there was a need for a common language such that the operating systems of the various hosts could talk to each other about the interprocess communications they support.

These needs are filled by the host to host protocol as implemented by the network control programs of the various hosts. The host to host protocol specifies a language of commands with parameters which are exchanged between NCPs to arrange, manage, and terminate process to process communication. The host to host protocol also specifies a common name space called sockets for indicating the source and destination of interprocess communications. The host to host protocol provides an interprocess communication mechanism called connections.

2.2 Relationship of the NCP to Telnet and File Transfer

Two functional capabilities are desirable in a computer network: to be able to use interactive terminals with programs on remote computers "as if you were there", and to transfer between computers large collections of data or files (which may be programs or data). The first capability is provided by the Telnet protocol and the second by the File Transfer Protocol (FTP) <NIC7104>.

2.2.1 Telnet

The implementation of Telnet is to the NCP almost indistinguishable from any pair of communicating processes in the network. At the computer where the human user sits at his terminal there is a program called User-Telnet which talks to the terminal on one side and talks to the NCP on the other side. At the

computer where the serving program is located there is a program called Server-Telnet which talks to the NCP on one side and talks to the serving program on the other side as if it (Server-Telnet) were a terminal. This last requirement, that the serving program believes it is talking to a terminal and not the network, is a tricky one. Some systems have constructs which allow a process to act as a terminal to another process, in these systems this is a simple requirement to implement. In other systems however the implementation of this capability has been so difficult that the Server-Telnet function has been implemented in the system with the NCP. The importance of this requirement is that it permits programs constructed for use from interactive terminals with no thought of the network to be used by remote users via the network.

2.2.2 File Transfer

The File Transfer Protocol (FTP) is designed to fill the need for a mechanism to transfer files containing programs or data between computers. The FTP utilizes a Telnet connection (pair) to allow the exchange of control information (requests and replies) and a separate data connection for the actual transmission of the file. The NCP is not normally aware of the fact that this data connection is used for the FTP.

2.3 Description of an Operating System

An operating system consists of program modules which augument the hardware and provide an environment for processes. Among the operating system modules of interest are a terminal control program (TCP), a file control program (FCP), and of course, a network control program (NCP). The interfaces between these operating system modules and user processes take the form of system calls and returns, and sometimes pseudo interrupts. System calls are implemented in a variety of ways, but often it is a special hardware instruction that invokes a system call (e.g. SVC, UUO, JSYS, MME). In higher level programming languages a system call is often indistinguishable from a subroutine call.

In some cases the form of the system call is quite different for each different module of the system.

A process is a program in execution with its associated address space, a location counter, some general registers, and usually some open files (or devices) . Processes may be created by users, though there are often processes which have been programmed by systems programmers for particular functions, and some of these may be initiated by the system when it begins running. Some processes may have access to greater (or lesser) capabilities than those created by normal users. In general there may be elaborate regulations provided by the system to control passing of the capability or permission to access particular resources between processes.

One important aspect of an operating system that has a great impact on the implementation of network functional capabilities is the provision for interprocess communication. Generally processes are viewed as independent computational units that need interact only with the operating system in a few very constrained ways (i.e. via system calls), however often it would be useful to build a new capability based on a combination of existing programs. One way of extending the usefulness of the process structure is to allow processes to communicate between themselves such that several processes may cooperate to accomplish a computational goal. The form of communication supported by an operating system very much influences the extent to which processes actually cooperate and, therefore, the extent to which use of the ARPANET is a natural extension of the programming enviroment. The network host to host protocol seeks to make available to processes a particular form of interprocess communication called connections.

There are many other features and essential functions of operating systems which will be ignored in this discussion because they are not relevant to the implementation of an NCP. Any multiprogramming or multiple process system must have a scheduler, some form of memory management, and provide for accounting, security, protection, and privacy.

3.0 Description of a Network Control Program

This section begins with a general overview of the operation of a network control program and gradually refines the definition. The functional components of an NCP are described first in a general way followed by a description of a typical set of system calls for the user-NCP interface and a description of the operation of the NCP and finally a detailed description of the functions of each NCP component.

3.1 NCP Functions:

The NCP must provide several functions to interface between the user process on one side and the IMP on the other side. Among these functions are device handling, formatting, error control, flow control, multiplexing, and synchronization.

The IMP is connected to the host computer much as any input or output device (since the IMP is a full duplex device it might be interfaced as two simplex devices). This implies that there must be in the lowest levels of the system a program module to control the IMP interface on the input output instruction and interrupt level.

The IMP-host protocol requires that a standard format be used for messages exchanged between the host and the IMP. The standard format has a 32 bit leader at the beginning of each message that contains control information indicating such things as the message type, source or destination host and logical link number. In addition to the leader the host to host protocol requires an additional 40 bits of prefix information for each message. This additional prefix indicates the byte size and number of bytes in the text of the message.

As data becomes available to send to the IMP, it is formatted into messages and queued for transmission to the IMP, and as messages are received from the IMP, they are queued for processing.

The messages received from the IMP are of various types. The two most frequently received are REGULAR and RFNM. A REGULAR message is used to transfer data. A RFNM (request for next message) is used to indicate that the previously sent message on this

logical link to this host was sucessfully received by the destination IMP. There are several other message types to indicate error conditions.

The REGULAR messages are of two catagories: user data or NCP to NCP control information. These are distinguished by the logical link number in the leader. All NCP to NCP control messages are transmitted on logical link zero. All user data messages are transmitted on a logical link number in the range 2 through 71.

The NCP control messages have several functions: to establish connections between pairs of processes in the network, to regulate the flow of data over these connections, to terminate connections, and to convey some special signals between the NCPs.

The leader of a data message must be examined to determine to which user process buffer the text of the message should be appended.

The user process interacts with the network by issuing system calls to the NCP to establish, use and terminate connections. When the user process issues calls which cause the NCP to send data to a distant process the NCP must include information in the leader that will enable the receiving NCP to determine for which process the data is intended.

The sending and receiving processes and hosts might not operate on the same sized quanta of information and they might operate at differing speeds. In such a situation it is natural to use buffers to smooth the flow of information and to allow each entity to operate using its prefered quantum size. It is the responsibility of the NCPs to manage these buffers and to regulate the flow from sending to receiving host such that the receiving host is able to buffer all the data sent with out undue difficulty.

Many of the NCP control messages received and many of the system calls will require the NCP to send NCP control messages to the foreign host.

The NCP must maintain information about each active connection in a connection table. The elements of each entry in this table include information about the user process using the connection, the foreign

host, the buffer location and fullness, and the state of the connection.

If any errors are detected the NCP must act to protect itself from harmful consequences, but it must also act to provide reliable service to all the user processes. In any case the NCP should record the relevant information about the error and the circumstances (e.g. time and day). The NCP must also report abnormal events to the the computer operator and be able to receive instructions from the computer operator.

The NCP should make available to users and the computer operator the status of hosts and connections.

The NCP should gather statistics on the usage of various elements of the protocol and resources allocated to it (e.g. buffers).

3.2 System Calls

The following discussion of the NCP and system calls is at a level comparable to that in the basic specifications of the IMP-host protocol <BBN1822> and the host to host protocol <McKenzie>. Note that the following sections are modeled closely on Network Working Group Request for Comments note number 55 <Newkirk>.

The system calls assumed to be available to user written processes are described.

LISTEN(PORT,AEN,CODE)

The local socket of this process with this AEN is associated with this process PORT. A return value is given in CODE. If there is a pending call the connection may be opened immediately and the process notified, if there is no matching pending call the NCP will notify the process when a matching RFC arrives.

CONNECT(PORT,AEN,FS,CODE)

The local socket of this process with this AEN is associated with this PORT, and the

specified foreign socket (FS) is also associated with this local socket, defining a connection. If there is a pending call matching these parameters the connection is opened and the process so notified. A return value is given in CODE. If there is no matching pending call the NCP communicates this request to the foreign host and notifies the local process when a matching request is received and the connection is opened.

SEND(PORT,BUFFER,LENGTH,CODE)

The data starting at BUFFER and extending LEGNTH bits is transmitted on the connection associated with this PORT in accordance with the allocation values. CODE is set with a return value.

RECEIVE(PORT,BUFFER,LENGTH,CODE)

Data received on the connection associated with this PORT is stored into the processes address space starting at BUFFER an extending for LEGNTH bits. A return value is set in CODE.

CLOSE(PORT,CODE)

Activity on the connection associated with this PORT is stopped. A return value is set in CODE.

INTERRUPT(PORT,CODE)

A special interrupt signal refering to the connection associated with this PORT is sent on a logically parallel data path. A return value is set in CODE.

STATUS(PORT,INFO,CODE)

The relevant status information from the connection table entry associated with the PORT is returned in INFO. A return value is set in CODE. This allows a user program to monitor the state of a connection, of special interest are the allocation values and the NCP buffer used and free values.

3.3 NCP Operations

Presented here are discriptions of the operations conducted during the three major phases of network usage: opening, communicating, and closing.

Opening

In order to establish a connection for data transmission, a pair of RFC's must be exchanged. An RTS must go from the receive side to the send side, and an STR must be issued by the send side to the receive side. In addition, the receive side in its RTS must specify a link number, and the send side in its STR must specify a byte size. These RFC's (RFC is a generic term encompassing RTS and STR) may be issued in either order.

A provision must also be made for queuing pending calls (i.e. RFC's which have not been dealt with by the user program). Thus, when a user is finished with a connection, he may choose to examine the next pending call from another process and decide to either accept or refuse the request for connection. A problem develops because the user may choose to not examine his pending calls; thus they will merely serve to occupy queue space in the NCP. Several alternative solutions to this problem are discussed later.

Utilizing the framework of the typical system calls described above, at least four temporal sequences can be envisioned for obtaining a successfully opened connection:

The user process may issue a LISTEN indicating that it is willing to connect to any process which sends an RFC specifying this local socket. When an RFC of interest arrives the NCP responds with a matching RFC and notifies the user processs of the now open connection. The user can, of course, inspect the parameters of the connection (using the STATUS system call, for example) to determine if it really wants the

connection, and if not the user can CLOSE the connection.

If upon processing a user request for a LISTEN, the NCP discovers that a pending call exists for this local socket, the NCP immediately sends the matching RFC and notifies the user of the open connection.

The user may issue a CONNECT, specifing a particular foreign socket that he would like to connect to. An RFC is issued. If the other NCP accepts the request, it answers by returning an RFC. When this acknowledging RFC is received the connection is opened.

When processing the CONNECT, the NCP may discover that a pending call exists from the specified foreign socket to the local socket in question. An acknowledging RFC is issued and the connection is opened.

In all the above cases the user is notified when the connection is opened, but data flow cannot begin until buffer space is allocated and an ALL command is transmitted.

Any of these connection scenarios will be interrupted if either the other NCP sends a CLS command when an RFC is expected or the user issues a CLOSE system call before the connection is opened, as discussed under Closing.

Communicating

Data can only flow when a connection is fully opened (i.e. when two RFC's have been exchanged). It is assumed that the NCP's have buffers for receiving incoming data and that there is some meaningful quantity which they can advertise on a per connection basis in ALL commands indicating the amount of data they can handle. It is noted that the sending side regulates its transmission according to that amount.

When a connection is opened, a connection

table entry field called their-allocation-values is set to zero. The receive side will decide how much space it can allocate and send an ALL message specifying that space. The send side will increment their-allocation-values by the allocated space and will then be able to send messages of length less than or equal to their-allocation-values. When messages are transmitted, the length of the message is subtracted from their-allocation-values. When the receive side allocates more buffer space (e.g. when a message is taken by the user, thus freeing some system buffer space), the number of bits newly available is sent to the send side via an ALL message.

Thus, their-allocation-values is never allowed to become negative and no transmission can take place if their-allocation-values equals zero.

Notice that the lengths specified in ALL messages are increments not the absolute size of the receiving buffer. This is necessitated by the asychronous nature of the flow control protocol. The values in the ALL command can be quite large, thus providing the facility for an essentially infinite bit sink, if that may ever be desired.

Closing

Just as two RFC's are required to open a connection, two CLS's are required to close a connection. Closing occurs under various circumstances and serves several purposes. To simplify the analysis of race conditions, four cases are distingusished: aborting, refusing, termination by receiver, and termination by sender

A user aborts a connection when he issues a CONNECT and then a CLOSE before the connection was opened. TYpically a user will abort following an extended wait for the acknowledgement; the NCP may also abort for him if he blows up.

A connection is refused when the NCP sends a CLS as a response to an arriving RFC. This may occur if a user has issued a connect and an RFC arrives from some other foreign socket.

After a connection is established, either side may terminate. The required sequence of events suggests that attempts to CLOSE by the receive side should be viewed as requests which are always honored as soon as possible by the send side. Any data which has not yet been passed to the user, or which continues over the network, is discarded. Requests to CLOSE by the send side are honored as soon as all data transmission is complete.

Aborting

Three cases are distinguished:

In the simplest case an RFC is sent followed later by a CLS. The other side responds with a CLS and the attempt to connect ends.

The foreign process may accept the connection concurrently with the local process aborting it. In this case, the foreign process will believe the local process is terminating an open connection.

The foreign process may refuse the connection concurrently with the local process aborting it. In this case, the foreign process will believe the local process is acknowledging its refusal.

Refusing

After an RFC is received, the local host may respond with an RFC or a CLS, or it may fail to respond. (The local host may have already sent its own RFC). If the local host sends a CLS, the local host is said to be refusing the request for connection.

CLS commands must be exchanged to close a connection, so it is necessary for the local host to maintain the connection table entry until an acknowledging CLS is returned.

Termination by the Receiver

When the user on the receive side issues a CLOSE system call, his NCP accepts and sends a CLS command immediately. Data may still arrive, however, and this data should be discarded. The send side, upon receiving the CLS, should immediately terminate the data flow.

Termination by the Sender

When the user on the send side issues a CLOSE system call, his NCP must accept it immediately, but may not send out a CLS command until all the data in the local buffers has been passed to the the foreign host. It is thus necessary to test for both buffer empty and RFNM received before sending the CLS command. The CLS must be acknowledged before the connection table entry can be deleted.

In this presentation several topics have been mentioned which should be further explained, among these are pending call queues, and connection states.

Pending Call Queues

It is essential that some form of queuing for pending RFC's be implemented. A simple way to see this is to examine a typical connection establishment sequence. One side issues a LISTEN, the other a CONNECT. If the LISTEN is issued before the RFC coming from the remote CONNECT arrives, all is fine. However, due to the asynchronous nature of the network, events may not occur in this sequence. If calls are not queued, and the RFC comes

before the LISTEN is issued, it will be refused; if it arrives later it will be accepted.

Unless one has infinite queue space, it is desireable to have some mechanism for purging the queues of old RFC's which the user never bothered to examine. An obvious but informal method is to note the time of arrival of each RFC, and then to periodically refuse all RFC's which have been queued longer than some arbitrary limit. Another action which should be included in any purging scheme is for the NCP to send a CLS on any pending connection when a user logs out or blows up.

The following scheme may be used to reduce the number of queued requests. When a CONNECT is issued, the NCP assumes that this local socket wants to talk to the specified foreign socket and to that socket only. It therefore purges from the pending call queue all non-matching RFC's by sending CLS's. Similarly, when the connection is in the RFC SENT state (a CONNECT has been issued and an RFC sent) all non-matching RFC are refused. If a LISTEN is issued and results in an open connection, the remainder of the pending calls are not removed from the queue, in the expectation that the user may wish to accept these requests in the future.

Connection states

Since the sequence of use of a connection involves many events and the legality and interpretation of many of these events is dependent on the preceeding activity, the NCP must remember, for each connection, where it is in the sequence. To keep this knowledge concisely the notion of a state is used. It has often been attempted to construct a state transition diagram to illustrate the possible state sequences of a connection, but to accurately take into account the many possibilities the diagram

would be overly complex, thus often a simpler diagram is used that shows only the main lines of the primary sequences.

The states that are typically present are:

NOT ACTIVE

This is not really a state, but the fact that a connection is not in the connection table at all.

LISTENING

The local socket is associated with a process port, and the NCP is waiting for an RFC to this local socket from any foreign socket in any host. When an RFC does arrive for this local socket a matching RFC is sent and the connection is set to the OPEN state.

RFC SENT

This state indicates that the local socket is associated with a process port, an RFC has been sent to a specific foreign socket in a specific foreign host, and no matching RFC has yet been received. This state would be entered if the user process issued a CONNECT call and there was no matching RFC in the pending call queue. When a matching RFC does arrive, the NCP completes the initialization and marks this connection in the OPEN state.

RFC RECEIVED

An RFC has been received for which there was no matching entry in the connection table. This is a pending call. If a user process issues a matching CONNECT or LISTEN it will be satisfied at once. The local NCP will send an RFC and the connection will be marked in the OPEN state.

OPEN

RFC's have been exchanged and the connection is open. Transmission may begin subject to the constraints of the buffer allocation quantities.

ALLOCATION WAIT

To transmit data on a send connection there must be some positive allocation values (i.e. buffer space in the receiving host), if the allocation value (either bits or messages) has fallen to zero then the sender must wait until an ALL command arrives to increment the allocation values.

RFNM WAIT

After sending data on a connection the sender is not permitted to send additional data until the corresponding (IMP to host) RFNM command is received. When a RFNM is received the state changes to either OPEN or ALLOCATION WAIT depending on the allocation values.

CLS SENT

The user program has issued a CLOSE system call and the NCP has sent a CLS command to the foreign host. This cannot be done on a send connection until all the data is sent that the user process has previously output, and until a RFNM has been received for the last message of that data.

CLS RECEIVED

A CLS command has been received from the foreign host. If this is a send connection the NCP notifies the user process at once and answers with a CLS command, moving the connection

to the CLOSED state possibly discarding data sent by the user but not yet transmitted by the NCP. If this is a receive connection, the NCP must wait until the user process has read all the received data. The NCP then sends a CLS to the foreign host, and notifies the user process.

CLOSED

The connection has been closed by an exchange of CLS commands. This is a transitory state and the connection should be deleted from the connection table shortly.

3.4 Functions of NCP Components

The following are the NCp functional components (program modules) and the tasks they carry out.

IMP Input Routine

Read messages from the IMP, and turn them over to the Network Interpreter Routine.

IMP Output Routine

Write messages to the IMP having received them from the Output Scheduler Routine.

Network Interpreter Routine

Analyze and act on messsages from the network, including maintaining connection table entries, composing replying messages, and exchanging information with the System Call Interpreter Routine, the Error and Statistics Routine, and the Output Scheduler Routine.

Output Scheduler Routine

Queue messages for delivery to the IMP Output Routine and to maintain a sent messages queue in case retransmisssion is called for.

SYstem Call Interpreter Routine

Analyze and act on system calls from the user processes, including maintaining connection table entries, composing messages to foreign hosts, and passing messages to the Output Scheduler Routine.

Error and Statistics Routine

Record and report on detected errors in the program or protocol and the use thereof. Gather and record statistics of interest.

In the following paragraphs each of the functional components is explained. While there are likely to be a number of unusual events not explicitly discussed, the majority of frequent events are described. One comment that is important at this point is that the NCP should be constructed to be resilient in the face of errors. That is to say that when an error is detected the NCp should act to protect itself from any harmful effects, but should also act in a manner consistent with achieving for the user process the most reliable and consistent communications possible.

IMP Input Routine

There must be someplace in the system to handle at the machine instruction level and interrupt level each device attached to the central processing unit. This is true for line printers, disks, terminals, and multiplexor channels as well as for the IMP or rather the IMP-host Special Interface.

The IMP-host communication is a full duplex channel (i.e. simultaneous transmission in both directions) and it is often easier to interface the IMP to the computer as two independent devices.

On input the routine has a buffer available for the longest message that can be received and has a pending instruction to read from the IMP. When an end of input interrupt occurs the routine checks the length and signals the Network Interpreter Routine that a message is ready. The routine then gets a new buffer and starts a new read operation.

The amount of buffering depends on the rate and frenquency at which the IMP Input Routine and the Network Interpreter Routine operate. Two buffers are recommended. The manner in which the routine signals the Network Interpreter Routine varies in various systems, it may be anything from raising a flag to a pseudo-interrupt.

IMP Output Routine

For output from the host to the IMP this routine is supplied by the Output Scheduler Routine with the starting address and the length of a data buffer to move to the IMP. When the transfer completes the routine frees the buffer and signals the Output Scheduler Routine.

Network Interpreter Routine

As a message from the network is processed the leader should be examined to check the link number field. If the value is zero then the message is a host to host control message. If the link number is (curently) 2 through 71 the message is a data message associated with an open connection. If the link number is other that these two catagories the message is either part of another protocol (e.g. Message Switching Protocol <Bressler>) or an error.

If the NCP is aware that another protocol is being used in parallel with the host to host protocol it can turn over any message belonging to that protocol to the appropriate program on the basis of this link number inspection.

A message from the network is processed by examining the type field. The action taken for each type is indicated. The two types expected most frequently are REGULAR and RFNM.

REGULAR

This is a regular message, it is passed

to the next phase in the input message analysis.

ERROR IN LEADER

This message indicates that there has been an error in a previous host to IMP message such that the IMP could not decipher the leader. This is the only response that will be received to one of the unanswered messages on the sent messages queue, it will take careful detective work to determine which message. If the message was related to this response can be determined, the message and leader should be checked for correctness and retransmitted.

IMP GOING DOWN

The IMP is warning of of an impending service outage, parameters in the message leader tell something about how soon and how long, so the user processes can be notified.

NOP

This no operation message is discarded.

RFNM

Ready for next message on this link. This message is used to confirm the transmission of messages from the host to the destination IMP. There should be an associated message on the sent message queue which can now be discarded.

The link number should be used to locate a connection table entry. The state field of that table entry should indicate RFNM WAIT. This state should be changed to either OPEN or ALLOCATION WAIT depending on the amount of data ready to send and the allocation values. If there is data to send and

the allocation values are positive the data send subroutine should be called.

DEAD

This is an indication that the destination host or IMP is dead. Normally the corresponding message on the sent message queue is discarded, and this host is marked dead in the host status table.

Note that some very recent work has been done on making the process to process communication more reliable in the face of network and operating system errors. Among the techniques is to treat a DEAD response as a temporary service interruption that will be quickly repaired (e.g. in a few minutes) and thus to retransmit the message associated with the DEAD response.

If the destination host (or IMP) is really dead then the NCP must close all of the connections to that host and notify any processes effected. The connection table must be updated.

ERROR IN DATA

There has been an error in the transmission of a previous host to IMP message, but the leader was preserved, so the link field can be used to attempt to associate this message with a message on the sent message queue. Once the associated message is determined, it is retransmitted.

INCOMPLETE

In this case the destination may be alive but the message was not delivered, the message is retransmitted.

RESET

The IMP has dropped and raised its ready line. If at the time of this occurance the IMP held a message to be transmitted to the host, or if a message transmission was in progress, data has been discarded. Similarly if a message was being transmitted to the IMP at the time of this occurance, it was discarded. The host and the IMP should at this point send each other several NOP messages to clear the line and reestablish the flow of messages. Some of the messages on the sent messages queue may need to be retransmitted.

UNASSIGNED

There are several messages types that are not assigned any meaning currently, these should be treated as NOPs, that is ignored.

The next phase in the analysis of a message from the network is to determine whether this is a host to host command or a data message associated with an open connection.

Suppose the current message has a link number in the range 2 through 71 identifying it as a data message associated with an open connection.

The link number is used to find a connection table entry and find the buffer associated with this connection, then checking the allocation and buffer space available the data is copied from the message into the processs buffer. Of course the proper checking is done to see that the connection is open, etc. and the allocation values are updated as appropriate possibly sending an ALL command. If the process has requested some notification when data arrives then the appropriate notice is given.

If the link number is zero then this is a

host to host command, and contains one or more commands. Some of the commands are trivial while others are quite complex and require the mantenance of state information in the connection table.

NOP

This command is discarded.

RTS

This is a receiver to sender request for connection. The connection table is searched for a matching entry (due to a CONNECT or LISTEN system call). If a match is found the connection is opened (sending an STR if it has not been done earlier). If the connection state was LISTENING the matching STR is sent, otherwise the connection state should be RFC SENT. In either case the connection state is set to OPEN.

If no match was found then the information is added to the table, creating a new entry. This is the case of a pending call. The connection state is set to RFC RECEIVED.

STR

This is a sender to receiver request for connection. The connection table is searched for a matching entry. If a matching entry is found the connection is opened (sending an RTS if it has not been done earlier). If the connection state is LISTENING the matching RTS is sent, otherwise the state should be RFC SENT. In either case the state is set to OPEN.

If no match was found then the information is added to the table, creating a new entry. This is the

case of the pending call. The connection state is set to RFC RECEIVED.

CLS

This is a command to close the connection. If the connection is in the state CLS SENT, the matching CLS has been sent and the connection state set to CLOSED. Otherwise the state is set to CLS RECEIVED.

If this is a send connection the associated process is informed and any unsent data is discarded. The matching CLS is then sent and the connection state set to CLOSED.

If this is a receive connection there may be data received and buffered which has not yet been read by the associated process. Thus the information that the connection is now closed must be flagged so that the process can be informed as it finishes reading the accumulated data. The matching CLS can be sent as soon as the NCP has flagged the connection entry. The connection state is also set to CLOSED.

At this point the state should be CLOSED, the data buffers should be empty, and the process aware that the connection is closed, thus the connection table entry can be deleted.

ALL

This is an allocation of buffer space for messages and bits that may be sent on the associated connection. The connection table entry fields for their-allocation-values is updated by adding the just received quantities to the values in the

table entry. If the connection state was ALLOCATION WAIT it is changed to OPEN. If there is data waiting to be sent the data send subroutine is called.

GVB

This is the give back command, it requires the return (in a RET command) of a portion of the current allocation for the associated connection. This is done by building a RET command and asking the Output Scheduler Routine to send it. The connection state should change to ALLOCATION WAIT if the allocation values have been reduced to zero.

RET

This is the return command, in answer to a give back command associated with this connection. The allocation values are now decremented by the amounts indicated in the return command.

INR

This is a command to interrupt the process associated with the receive connection indicated by this link number.

INS

This is a command to interrupt the process associated with the send connection indicated by this link number.

ECO

This command requires that an echo reply command be sent. The ERP command containing the received parameter is constructed and turned

over to the Output Scheduler Routine.

ERP

This is a response to an echo command and the data received should be exactly that which was sent.

ERR

This command indicates that the sender of this command has detected an error. This command and the date and time are turned over to the Error and Statistics Routine for recording. The data portion of the command indicates which of the messages and connections are involved since these connections may have to be resynchronized.

RST

This is a host to host reset command. This indicates that the sending host has cleared all of its tables of information, i.e. all connection are dissolved. Thus, all the tables of information relating to the sending host are cleared, and a RRP command. The RRP is composed and turned over to the Output Scheduler Routine to be sent. Also any user process that may be effected are notified.

RRP

This is a response to a reset command previously sent.

Data Send Subroutine

This subroutine checks to see if there is buffered data and allocation available, and the state is OPEN. If so the subroutine forms a message whose length is the minimum of the data available, the

allocation available and the maximum allowed message size. This message is turned over to the Output Scheduler Routine. The allocation values are updated, and the state is set to RFNM WAIT.

Output Scheduler Routine

As messages ready for transmission to the various remote hosts are turned over to the Output Scheduler Routine, they are queued and delivered one at a time to the IMP Output Routine. It may be appropriate to order the queue according to some priority (e.g. host to host commands first), but this is optional. As the messages are sent by the IMP Output Routine they should be placed on a sent messages queue. As RFNMs are received the corresponding messages can be deleted from the sent messages queue. Other responses (e.g. INCOMPLETE) will cause a message on the sent messages queue to be indicated for retransmission.

System Call Interpreter Routine

As a user process issues system calls the NCP (eventually) must be invoked to service these requests. Some calls will be control requests (CONNECT, LISTEN, CLOSE, ...) while others will be data requests (SEND, RECEIVE).

For the control requests the NCP must check the status of table entries referenced in the call or create new entries. Some calls may require composition and sending of NCP commands to other hosts.

For the data flow requests the NCP must check tables and buffers and move data from system buffers to user process buffers or vice versa as the buffer space or data availability permits. This may result in the NCP sending to the other host either a data message or an allocate command depending on the direction of data flow.

The actions taken by the NCP to satisfy each of the system calls is now indicated.

LISTEN

The NCP searches the connection table for a pending call which matches this request. If a match is found the table entry is completed with the information supplied by this call; namely, the process identification and the port identification. A buffer should be assigned and initialized. The state should be RFC RECEIVED, a matching RFC should be sent and the connection moved to the OPEN state. The user process should be notified of the now open connection.

If there is no matching pending call a new table entry is created, filling in the values for local socket, process, and port identification. The state should be set to LISTENING.

CONNECT

The connection table is searched for a pending call. If a pending call is found the state should indicate RFC RECEIVED. A matching RFC is sent and the state updated to OPEN. A buffer should be assigned and initialized, and the remaining table entries filled in. The user process should be notified of the now open connection.

If no matching pending call was found a new table entry is created and filled in with the supplied information. The NCP sends an RFC and sets the table entry state to RFC SENT.

SEND

The indicated data is copied from the users buffer to the NCP buffer (being concatenated to any data already there) associated with this port. The buffer

can be found from the the connection table entry associated with this port. The state is checked and if it is OPEN the allocation values are checked if either is zero the state is set to ALLOCATION WAIT. If there is space allocated the data send subroutine is called.

RECEIVE

The NCP moves data from the NCP buffer indicated in the connection table entry associated with this port to the users buffer up to the limit of either the amount specified or the amount available. The amount of data actually moved is indicated to the process. The NCP also checks to see if this frees a sufficient amount of buffer space to send an allocate command. If so an ALL is formatted and turned over to the Output Scheduler Routine.

CLOSE

The NCP will try to close this connection as soon as it can be sure the data flow has stopped. If this is a send connection, the NCP will wait until all the data issued by the user in SEND system calls has been transmitted to the remote host and a RFNM returned from the last message. This condition can be checked by ascertaining that the NCP buffer for this connection is empty and the state is OPEN (or even ALLOCATION WAIT). Once this all-data-transmitted condition has been met the NCP can begin to close the connection. The System Call Interpreter Routine forms a CLS command and turns it over to the Output Scheduler Routine. The connection state is set to CLS SENT.

If this is a receive connection the user process clearly does not want any more data even if there is some it has

not read, so any buffered data or any that arrives following the CLOSE call is discarded, and no new allocates are sent. The NCP sends a CLS at once to notify the sending NCP and process to stop their transmission and to close the connection. The connection state is set to CLS SENT.

INTERRUPT

For a send connection the NCP forms a INS command, and for a receive connection the NCP forms a INR command; the command is forwarded to the Output Scheduler Routine.

STATUS

The NCP returns, in the INFO argument, data from the connection table entry associated with this port. This system call has no effect on the state of connections or buffers, and no information is transmitted to the network because of it.

Error and Statistics Routine

In case of any error condition detected a record should be added to a log file indicating the date, time, leader, and other circumstances of the error (e.g. NCP control message, portion of the content of the message).

Errors in use of the host to host protocol should also be reported back to the offending host using the ERR command. Any ERR commands received should be logged.

Certain kinds of errors and errors that occur with high frequency should be reported to a system operator via an on-line console.

The Connection Table

The entries of the connection table should contain the following fields:

link number

local socket

foreign socket

foreign host

process identification

port identification

buffer address

state of connection

byte size

allocation values

date and time

In summary, the functions of the NCP are restated: to provide to the user processes a form of interprocess communication called connections. In carrying out this function the NCP must implement mechanisms for error control, flow control (allocates), multiplexing (sockets and links), and synchronization (interrupts).

4.0 Basis of Survey Comparisons

This section describes the model NCP discussed in the previous section as if were an actual implementation. For purposes of comparison the format is the same as that used in the descriptions of the surveyed systems.

Model system

System

The operating system is a timesharing system which provides for user process, a file system, and interprocess communication. Each user process has an independent (virtual) address space, a set of general registers, a location counter, and a set of open files. The system provides system calls to open, read or write, and close files, terminals or connections. Interprocess communications are supported by a

mechanics similar to the network connections. There is also a pseudo interrupt facility such that the system can cause the location counter of a process to be set to the processes interrupt address.

NCP

The network control program is implemented as part of the operating system, though it is programmed in an way that would allow it to be run as a user program (except for the privileged interaction with the IMP input and output handler). The communication between the NCP and the user processes uses the existing interprocess communication system calls.

Points of comparison

System Calls

The system calls Listen, Connect, Send, Receive, Close, Interrupt, and Status are available to users. These call are similar to other input and output service call.

Return Characteristics

These system calls are nonblocking, with the option of blocking until completion.

Programming Languages

The network system calls are available to programmers in every programming language on the system.

RFC Queueing Policy

Requests for connection are queued until either the local process issues a Connect system call or a timeout period elapses. When a Connect system call is issued on a local socket all requests queued on that socket are refused execept the matching request. Requests queued on a local socket opened via a local Listen system call are retained in the queue.

Timeout Policy

Queued RFCs are timed out after ten minutes, a missing CLS is timed out after two minutes.

Connection States

The following states are used: Not Active, Listening, RFC Sent, RFC Received, Open, Allocation Wait, RFNM Wait, CLS Sent, CLS Received, Closed.

Allocation Policy

The allocation policy is to carefully control the flow of data using the bit count of the allocation to allow exactly the buffer space reserved on a per connection basis. In particular the initial allocation is a large number of messages (100) and as many bits as available in the connection buffer. As the data flows, the allocation values are adjusted whenever the values fall to a lower bound expressed as a fraction of the initial allocation (e.g. two-thirds). When the allocation is adjusted it is set to the maximum values then available.

Interrupt Treatment

The network interrupt signals INS and INR cause the user program associated with the connection to begin executing at its interrupt address.

Retransmission Policy

The incomplete transmission reply from the IMP will cause the indicated message to be retransmitted.

Error Treatment

A log file is kept of all unusual occurances, among the things entered into this log are all ERR messages received, all ERR messages sent, and any

information about internal errors detected.

Measurement and Status Information

The NCP keeps running totals on the number of times each host to host command is sent or received, on the number of messages sent and received of the number of bits sent and received, and the number of each type of IMP to host and host to IMP message received or sent. The NCP keeps an accounting log in which is recorded the information about each connection when the connection is closed. The data recorded is the user name and account number, the foreign host, the number of messages and bits sent and received, the elapsed time and the time and day when the connection was closed.

The NCP connection table is accessed by a status display program which displays the foreign host, the socket numbers, the link number, the allocation values, and connection state, for each connection.

Operator Interaction

The operator can close any connection or send a reset to any host, can stop, continue, or reinitialize the NCP. The operator is informed of errors by the NCP.

Experimental Protocols

There is a provision to pass messages which arrive on designated links to other programs, to provide for testing of experimental protocols.

5.0 The Survey

TENEX

System

The TENEX system was designed and built by Bolt, Beranek, and Newman, Inc. (BBN) to operate on

the Digital Equipment Corporation (DEC) PDP-10 computer. The BBN modifications include a special instruction to implement system calls (JSYS), and paging hardware to implement a virtual memory but also to allow the physical core memory to be expanded beyond the 256 kilowords directly addressable by the 18 bit instruction address field.

The TENEX system provides a user environment which is embodied in a job. A job can be made up of several forks arranged in a tree structured hierarchy. Each fork is a program in a virtual machine and corresponds closely to the concept of a process.

The TENEX file naming conventions are organized for consistent naming of all types of devices and files. Network sockets are one type of file name and can be used anywhere a file name argument is called for. The file structure is rather flat in that each user has one directory which contains all his files, though the file names do consist of three parts - name, extension, and version.

NCP

The TENEX NCP consists of five modules: the Interrupt Service Routine, the Message Packaging Routine, the Control Routine, the File System Interface Routine, and The Server-Telnet Routine. The bulk of the work is concentrated in two routines: Message Packaging, and Control.

The Message Packaging Routine not only formats the data into host-IMP messages, it also deals with the IMP-host protocol and RFNM waits. The Control routine takes care of flow control on open connections, connection establishment and termination, and connection table management.

Points of comparison with the general model

System Calls

The network system calls are implemented as regular file system calls. Within this framework the functions suggested in the

general model are provided. In addition to the suggested calls are calls to dump the buffer (i.e. transmit now), and to control the maximum allocation size. There is also a call to cause a pair of connections to be treated as a terminal (TTY:).

Return Characteristics

The system calls are blocking, but there are options for pseudo interrupts and immediate return.

Programming Languages

Both LISP and BCPL have features for network communication as well as assembly language.

RFC Queueing Policy

All requests for connection which do not immediately match are queued. The limit for such queued requests is approximately 100 requests. Unmatched requests are timed out.

Timeout Policy

Queued (unmatched) RFCs are timed out after two minutes, local Listenss are not timed out. A missing matching CLS is timed out after two minutes. A missing RFNM is timed out after two minutes.

Connection States

Fourteen connection states are used. The three additional states are a finer level of detail in the sequences of events for opening and closing connections, for example there is a state for "CLS received but waiting for RFNM for last data sent".

Allocation Policy

The initial allocation issued for a connection is two full messages plus a user specified buffer, this usually totals

approximately 24,000 bits. The initial message count is two messages. The policy is to maintain the allocation values as close to these initial (maximum) values as possible.

Interrupt Treatment

The network interrupt signal causes a pseudo interrupt to the indicated process.

Retransmission Policy

The incomplete transmission response to a message causes the NCP to retransmit the message. There is no limit on the number of times this may occur.

Error Treatment

An online log is made of detected program errors, detected protocol errors, and ERR commands received. No ERR commands are sent. No statistical information is kept on errors.

Measurement and Status Information

There are no provisions for measurement in the NCP. Status information is maintained by the NCP such that user programs can obtain information on hosts up, connections open, and states of connections. There is a program called Netstat that users can run to display this information at their terminal.

Operator Interaction

The computer operator is informed online about program and protocol errors detected. The operator can suspend and continue network service, and can reinitialize the NCP. A systems programmer can watch the activity on a Telnet connection.

Experimental Protocols

There is a provision for turning over messages with link numbers outside the ranges assigned to the host to host protocol to another program. This allows experimental protocols to be tested in parallel with the host to host protocol.

Multics

System

The Multics (for Multiplexed Information and Computing Service) system was designed and built at project MAC at the Massachusetts Institute of Technology. The system is currently implemented on the Honeywell 6180.

A major emphasis in the Multics system is on the controled sharing of information (programs and data). The directory structure is quite general, allowing a hierarchy of directories with each level containing the names of segments or other directories.

The term file is not used in the Multics environment, rather the term segment is appropriate. A segment is a linear address space which may contain either a program or data. The set of segments known to a process are all directly accessible to (authorized) users. Each segment is subject to an elaborate set of access controls.

In addition to segment access controls there are protection rings which are an extension of the master/slave or supervisor/user state concepts in many systems. In the Multics system, programs executing in one ring are protected from programs executing in higher numbered rings, except the programs running in higher numbered rings are permitted to make calls to prespecified entry points in the programs executing in the lower numbered rings.

System calls are identical to calling any program segment. Some processes, however, can access more deeply into the system or are privedged to control system resources. The method of interaction with the network is by

means of special system calls, different from those that handle other input output operations.

NCP

The Multics network control program is organized in three components: the Network Daemon, the NCP, and the IMP DIM (for IMP Device Interface Module). Taking these in reverse order, the function of the IMP DIM is to manage all the frequent operations of the host to host protocol (e.g. data messages and allocates) and the low level input and output operations with the IMP. The NCP module performs the less frequent operations (e.g. opening connections) and services the user program system calls. The Network Daemon deals with administrative functions, and responds to host to host control commands.

The IMP DIM contains the code which manages the leader portion of IMP messages, issues input and output instructions for the full duplex Asynchronous Bit Serial Interface (ABSI) which connects the IMP to a pair of H6180 IOM (input output controller) common peripheral channels. The IMP DIM manages the assignment of link numbers on messages to be written, and it associates link numbers with socket numbers (and hence processes) on messages which are read. A natural extension of such functions, which was also a practical necessity, is enforcing the host to host protocol flow control discipline, particularly in processing the NCP ALL command. In part because the IMP DIM operates at interrupt time this role avoids the need to wakeup the Network Daemon every time a small recipient host is willing to allow a few more characters to be transmitted.

The NCP interfaces with the IMP DIM on one hand and with the user processes on the other. It must manage the socket space of the host to host protocol on behalf of any and all user (and system) processes which deal with the network on Multics and process the bulk of the NCP commands (such as interrupt the process associated with a given socket, reset all table entries associated with a given host, etc.). The socket management

is based on the notion of a socket being in a particular state, and when that state changes the NCP directs a wakeup to the process controlling the socket.

The Network Daemon is a process in the user ring which processes the host to host control commands, sometimes calling on the NCP to complete the processing. The IMP DIM wakes up the Network Daemon to process the control commands in the same way it wakes up user processes to process data messages.

Points of comparison with the general model

System Calls

Multics has two additional system calls: activate and deactivate. A socket must be activated before it can be used in any other system call, in a sense the activate call indicates to the NCP that the user program is interested in this local socket and reserves it to the user process. Deactivate then releases the local socket and frees any associated resources. Also the echo function is reserved to privileged processes only.

The NCP primitives (the discrete entry points of the NCP and several free standing subroutines) allow for precise management of sockets, from activation through establishing byte sizes and sending or accepting host to host protocol requests for connection to causing host to host protocol CLSs to be sent. The subroutines provide for performing the initial connection protocol or sending or accepting requests for connection, the managing a process socket space, and performing the 9-to-8 and 8-to-9 bit conversions necessary to go to and from Multics' internal representation of ASCII from and to the networks representation.

Return Characteristics

The system calls are nonblocking and a

wakeup signal occurs when the requested action is completed. System call routines also post status information which can be inspected by the process.

Programming Languages

PL/1, Lisp, and BCPL have interfaces to the NCP primitive functions. Fortran programs can access the network function via special subroutines. Assembly language programs can invoke the network functions.

RFC Queueing Policy

Requests for Connection are not queued.

Timeout Policy

The Multics NCP does not timeout an step or state.

Connection States

Multics uses a few more connection states than the example description, but these additional states primarily make explicit the fine structure of events during connection establishment and termination. For example there is a "cls-read" state to indicate the connection is closed on the foreign side but that the local process is still reading buffered data. Two other states are the "active" state corresponding to socket selected by an "activate call, but not used as yet; and the "broken" state. The broken state indicates that this socket has been involved with some anomaly occuring within the Multics network software, any previous connection state has been destroyed.

Allocation Policy

Multics issues large allocation quantities relying on a large input buffer used in a pooled fashion. The total input buffer space is a segment set to allow 225 full sized (8000 bit) messages. The initial

allocation for each receive connection is 25 messages and 65000 bits. The allocation is updated only when one of the values falls to an established lower bound, and then it is raised to the upper bound.

The IMP DIM's buffering strategy on input is such that Multics permits quite large allocations (in the NCP sense), by means of placing input on receipt from the read channel of the ABSI as rapidly as possible into pageable buffers managed by the IMP DIM code which runs at "call time" in the user's process. (The actual servicing of the ABSI interrupts is, of course, performed at "interrupt time" and involves wired down buffers.) The buffering strategy for writing also involves a mixture of pageable and wired buffers, although at the present time the wired buffer strategy does not utilize maximum IMP message-size buffers as does the read side.

Interrupt Treatment

An incoming interrupt signal (INS or INR) is first read by the IMP DIM which passes it to the Network Daemon process. The Network Daemon calls in the NCP to process the command. The NCP determines the relevant process and sends it an "interprocess signal" (ips). This is basically a PL/1 on condition which is raised, the particular condition being "QUIT".

Retransmission Policy

On incomplete transmission Multics attempts to retransmit the unfortunate message up to three times.

Error Treatment

Multics does not send ERR commands when it detects protocol violations. Any ERR commands received are merely counted.

Measurement and Status Information

Few measurement features are included in the network software beyond the meters in the basic Multics software.

The operator and processes can access a data base which indicates which hosts are currently communicating, a user can determine the status of his own connections.

Operator Interaction

The NCP reports online about program reinitializations, IMP going down messages and IMP crashes. The system operator can force a Telnet user connections closed, suspend and continue, or reinitialize the NCP operation.

Experimental Protocols

Multics does allow for experimental protocols by allowing for messages with a given link number in the leader to be diverted to another protocol process.

IBM 360/75 MVT

System

The University of California, Santa Barbara Computer Center operates an IBM 360/75 with the MVT operating system. This system is primarily batch oriented, however at Santa Barbara it supports the Culler-Fried On Line Mathematical System.

NCP

The network control program was designed with several objectives. First and most predominant was to keep the software independent of and entirely sepatate from the operating system. Second was to make the services of the NCP available to any task in the system. Implicit in this requirement is the need for some sort of interprocess communication mechanism. Although

such a facility exists in most multiprogrammed systems, it is conspicuously absent from OS MVT; thus the NCP must itself provide whatever portion of such a mechanism it requires to communicate with tasks that use its services. Third, the NCP must provide a means for subordinating network activity in the host to the operational requirements of a multiprogrammed systems. This means that the operations staff must be able to control the NCP's activity, and obtain status information from the NCP. Finally the NCP must provide a record of its activity for debugging and tuning the NCP and for statistical information on network use <White2>.

The NCP is inserted into the system as a normal job. Once in execution the NCP assumes control of selected portions of the operating system. All modifications made to the system by the NCP are made dynamically and are transparent to the operating system. The NCP terminates execution only at the operators request, and as it does so, it extracts itself from the system, undoing the modifications it made at initiation. The NCP employs the operating system specify-task-abnormal-exit macro instruction to assure the extraction process is performed even if the NCP terminates abnormally.

THE NCP operates in supervisor state and with a protection key of zero. Obtaining this status is part of the initialization process and is accomplished with the aid of an installation provided supervisor call.

Tasks in the system communicate with the NCP by means of a supervisor call (SVC). So that the NCP can detect the issuance of its SVC the NCP instates itself as the systems SVC first level interrupt handler (FLIH). In this capacity the NCP examines every SVC interrupt which occurs, intercepting and processing the one of interest to the NCP, while allowing all other SVCs to proceed to the systems FLIH where they are processed normally. The NCPs presence is an overhead of ten instructions per SVC.

The NCP assumes the IMP to be attached to the

360 as two devices, one input and one output. In transfering data to or from the IMP the NCP bypasses the conventional execute channel program facility of the operating system Instead it executes directly the I/O machine instructions, so that the NCP can process the I/O interrupts which occur, the NCP instates itself as the systems I/O FLIH. In this capacity the NCP examines every I/O interrupt which occurs, intercepting and processing those generated by the IMP interface devices, while allowing the other interrupts to proceed to the systems FLIH and by processed normally. The NCPs presence is an overhead of ten instructions per I/O interrupt.

points of comparison with the general model

System Calls

In addition to the normal system calls there is a call to perform the host name to host number mapping, and a call to find out the status of a host. The echo function is not available as a system call.

Return Characteristics

The system calls are nonblocking and use a wakeup type return. The mechanism is called Wait and Post.

Programming Languages

There are provisions for network system calls in PL/1, Fortran, and of course assembly language.

RFC Queueing Policy

Requests are queued up to ten for each local socket, then succeding requests are refused.

Timeout Policy

There are no timeout periods for request for connection either locally or remotely

initiated. An echo reply is timed out after 4 minutes, and a matching close is timed out after 15 seconds.

Connection States

The only additional state is called "I/O pending".

Allocation Policy

The NCP reserves a 2048 bit buffer for each open connection and allocates strictly according to the space available in that buffer plus any additional space set by the user program. The message allocation is set very large (65,000) and therefore is not a factor in controling message flow. A give back command might be sent if the user issues a receive system call with the length allowed to be variable and a large maximum length. Such a call would allocate some buffer space in the users address space. If the call was satisfied with less than the maximum length the left over space in the users address space buffer would be deallocated by means of a give back command.

Interrupt Treatment

When a network interrupt command is received a note is made, and the return values of the next user send or receive system call will carry this note. Thus the network interrupt will not be effective if the user process is stuck in a computation loop.

Retransmission Policy

No retransmission is attempted.

Error Treatment

While received error commands are logged, no error commands are sent when a protocol error is detected. No statistics on errors are kept.

Measurement and Status Information

The computer operator can find out which hosts are currently communicating, and see the connections currently open. Both the users and the operator can determine the states of active connections, but neither can find out the allocation values, though they can determine the number of bytes pending input or output.

There are two host status tables, the first is maintained by the NCP by sending ECHO commands to each host once every four minutes, the second table (network accessible on socket 15) is maintained by attempting a connection to the server-Telnet socket of each host onec every 15 minutes.

Operator Interaction

The NCP reports to the operator changes in the IMP status. The operator may close all connections, or send a reset to a specific site, as well as suspend and continue or reinitialize the NCP operation.

Experimental Protocols

There is no provision for experimental protocols.

IBM 360/91 MVT

System

The University of California, Los Angeles, Campus Computing Network (CCN) operates an IBM 360/91 computer using the OS/MVT operating system. This is primarilly a batch oriented system. However, at CCN there are several interactive subsystems. Among these are URSA and TSO.

URSA is a locally developed interactive system using display terminals which allows users to compose program and data data sets (files) and to submit data sets to run as jobs in the batch

job stream. The URSA system provides users (and the computer operator) extensive job and system monitoring capabilities.

TSO is the IBM provided Time Sharing Option for its OS/MVT systems. This provides the facilities generally found in an interactive timesharing system.

CCN has developed a general purpose and powerful interprocess communication facility called the Exchange <Braden>. This facility is used to implement the network software.

NCP

The network related software is organized into three catagories: the NCP, the protocol routines, and the user level programs. The user level program interact with the protocol routines via the Exchange facility. The protocol routines obtain services from the NCP by making subroutine calls on the NCP.

The NCP is organized into three sections: the IMP input and output section, the NCP section, and the Logger section. The logger section performs the Initial Connection Protocol (ICP).

There are protocol routines for several function oriented protocols, for example User-Telnet, Server-Telnet, File Transfer, and Remote Job Service.

Points of comparison with the general model

System Calls

The user program makes request via the Exchange facility to protocol routines. These requests are generally in terms of higher level things than the system calls suggested in the general model. For example the user level requests are for such things as "open a Telnet connection pair", or "send this line of EBCDIC characters". The protocol routine to which such a request is addressed of course has

the facilities to deal with the NCP in the terms of the suggested calls.

Return Characteristics

The Exchange provides options for either blocking or nonblocking behavior, and for a wakeup or interrupt type of return.

Programming Languages

Only assembly language currently allows convenient access to the network (through the Exchange macro), but work is being done to make network access from PL/1 and Fortran possible.

RFC Queueing Policy

Requests for connection are refused at once unless the local socket matches an active instance of a protocol routine, or the request is for an ICP socket. Requests to open sockets are refused, except for requests to open ICP sockets, which are queued.

Timeout Policy

Requests by local processes (protocol routines) are not timed out by the NCP. A queued RFC from a remote host is timed out after one minute. A missing matching CLS is timed out after one minute. A missing RRP or ERP is timed out after 30 seconds. The initial connection protocol is timed out if the 32 bit number is not sent or received within one minute. Of course the protocol routines can time out any operation they request.

Connection States

The suggested connection states are used with a slight elaboration in the connection establishment and termination phases.

Allocation Policy

The allocation is handled by the protocol routines, which may each have a different strategy. Generally the existing protocol routines follow strict dedicated space policies using circular buffers.

Interrupt Treatment

The NCP notifies the appropriate protocol routine on a network interrupt. The protocol routine can notify the user level process via the Exchange.

Retransmission Policy

If an incomplete transmission response is received the NCP attempts retransmission up to five times.

Error Treatment

When a protocol error is detected the NCP composes an ERR message and sends it to the other host. An online report is made of both the ERR messages sent and the ERR messages received. Any errors detected in the operation of the NCP are reported online.

Measurement and Status Information

There are some facilities for measuring network usage in an accounting sense, but no facilities in the NCP to measure particular protocol functions. The things that can be messured are: per connection the number of bits and messages, the user identification, the protocol routine used, the time of day and the elapsed time the connection was open. There is status information on hosts currently communicating, and connections currently open.

Operator Interaction

The operator is informed online about

error conditions and program or network crashes. The operator is informed if a message is received from a host not in the host table. The operator may force a Telnet connection pair closed, may reinitialize the NCP, or may stop and remove the NCP from the system. It may be possible for the operator or a systems programmer to spy or advise on a connection but this depends on the protocol routine being used.

Experimental Protocols

While at the time of the survey there were no experimental protocols facilities, such facilities would be easy to add as a new protocol routine.

IBM 370/158 OS/MVT

System

The RAND Corporation operates an IBM 370/158 computer. The operating system at the time of this survey was the OS/MVT system with HASP, there were plans to change to the VS2 Release 2 system. In any case the information in this report is on the NCP implementation with the OS/MVT system.

NCP

The network control program is the program written at UCSB. The network is interfaced in a way that allows all programs running under OS/MVT to access and be accessed by the network.

The system at RAND is much the same as at UCSB. There are some differences from the UCSB situation at the next level of software, however. At RAND there is sustantial use of the Wylber and Milten software packages to provide interactive computer services to local users. The NCP communicates with Milten via the operating system to provide network access to this interactive facility. There is also a Network Access Program (NAP) through which local users can access the network.

Points of comparison with the general model

System Calls

Same as UCSB.

Return Characteristics

Same as UCSB.

Programming Languages

Assembly language.

RFC Queueing Policy

Same as UCSB.

Timeout Policy

Same as UCSB.

Connection States

Same as UCSB.

Allocation Policy

Same as UCSB.

Interrupt Treatment

Same as UCSB.

Retransmission Policy

Same as UCSB.

Error Treatment

Same as UCSB.

Measurement and Status Information

The status information available is basically the same as that as the UCSB implementation, however the status information is not available on socket 15 nor is the ICP polling done.

Operator Interaction

Same as UCSB, except that the operator can spy on a connection.

Experimental Protocols

Same as UCSB.

IBM 370/145 VM

System

The Systems Development Corporation has two computers connected to the ARPANET, an IBM 370/158 and an IBM 370/145. The model 158 uses the same software as the RAND Corporation computer so this section will discuss the model 145 only. The operating system being used Virtial Machine or VM system, which is much like the CP system used on the IBM 360/67.

NCP

The NCp for the 145 at SDC, although still in design at the time this information was obtained, is presented to the extent that the design exists. The general idea is to use as much of the UCSB NCP as possible and thereby to reduce the work required to get the 145 on the network. The approach is to dedicate a virtual machine to the NCP and have that virtual machine own the IMP. Other virtual machines will communicate with the NCP by associating their (virtual) card readers and line printers with the NCP virtual machine's line printer and card reader. This design is similar to the design used at Lincoln Laboratories for a 360/67 CP/CMS system <Winett>.

Points of comparison with the general model

System Calls

The system call interface will be different than the general model due to the communications between the users virtual machine and the NCP virtual machine being an association between one

machine's virtual card reader and the other machine's virtual line printer. Further the intention is to allow the user to control only Initial Connection Protocol connection pairs of sockets (i.e. Telnet data streams).

Return Characteristics

The calls can be constructed to be either blocking or nonblocking and pseudo interrupt returns are possible.

Programming Languages

Since the calls are communicated as data to (virtual) line printers and from (virtual) card readers, any language that has provision for input and output to such devices can be used to communicate with the NCP.

RFC Queueing Policy

Same as UCSB.

Timeout Policy

Same as UCSB.

Connection States

Same as UCSB.

Allocation Policy

Basically the same as UCSB, but there may be some adjustment of the buffer sizes.

Interrupt Treatment

The treatment of the interrupt is not defined at this time, and may be more difficult that in other systems due to the virtual machine structure.

Retransmission Policy

Same as UCSB.

Error Treatment

Same as UCSB.

Measurement and Status Information

Same as UCSB.

Operator Interaction

The communication between the computer operator and the NCP has not been designed. It is planned to be similar the the UCSB implementation.

Experimental Protocols

Same as UCSB.

Burroughs 6700 MCP

System

The University of California, San Diego Computer Center operates a Burroughs 6700 computer system. The operating system is the Burroughs supplied Master Control Program (MCP). The system is a dual processor system.

The Burroughs machine provides hardware segmentation mechanism. The operating system is written in a version of Algol.

The system supports both batch and interactive modes of processing, and many users find it convenient to mix these modes. The interactive executive program is called CANDE (for Command and Edit).

NCP

The network software is principally contained in a program called the Network Message Control System (NETMCS).

The flow of data from and to the network is as follows: messages from the IMP first are read by a Micro 820 minicomputer which is the IMP-host special interface, the data is reformatted and

passed to the DCP, the DCP then passes the data to the NETMCS program where it is examined and acted on. If the data were to be intended for a service program (e.g. CANDE) the data would be reformatted and passed back to the DCP which would then route it to the service program. In the oposite direction the path is exactly reversed.

Points of comparison with the general model

System Calls

There is a system call to perform the Initial Connection Protocol. There are system calls to establish a character code to be used. There is a system call to set the mode of interaction to character at a time or line at a time. There are a pair of calls for sending and receiving data. Basically the calls are set up to provide a higher level interface than that described in the model. Here the user is presented with a mechanism where the elments are the ICP established Telnet connection pairs.

Return Characteristics

The system calls may be either blocking or non blocking and may wakeup or interrupt the process on completion. The interrupt return feature is not normally used however. The primary mechanism for process to process or process-system communication is placing messages on a queue and reading messages from a queue.

Programming Languages

The programmers best interface to network functions is in the Algol language, but the network functions can also be accessed from Fortran, Cobol, Basic, and PL/1.

RFC Queueing Policy

All foreign RFCs are queued and if not

referenced by a local process in three minutes are answered by a CLS. RFCs for local sockets in the range 0 through 255 are queued forever.

Timeout Policy

Requests for connection are refused after three minutes, except requests to sockets which are Listened to.

Connection States

The states RFNM WAIT and ALLOCATION WAIT are not used because these conditions are implemented using the event mechanism. There is a state for "socket owned by process but not in use".

Allocation Policy

The allocation policy is essentially to allocate an infinite space and message count. In the past the largest values that would fit in the allocate command fields were used, but recently this has been changed to 10 messages and 32,000 bits due to problems with some hosts which seemingly were not able to deal correctly with the large values. The system would send a give back command if too much data were to be queued. The space for this buffering is drawn from a pool of buffers commonly shared by several of the system modules, these buffers may be backed off to the disk if they become large. This would be done by the buffer management module without the awareness of the NCP.

Interrupt Treatment

The network interrupt signal is ignored.

Retransmission Policy

There is no retransmission of messages by the NCP. If an incomplete transmission response is received it is acted on as if it were a RFNM, except that an error

indication is displayed to the computer operator.

Error Treatment

When a protocol error is detected both a log entry is made and a ERR message is sent to the other NCP. When an ERR command is received it is both logged and reported online to the operator.

Measurement and Status Information

There is available to processes a data base which shows the host currently communicating and the connections currently open. There are no special facilities in the NCP for measuring the network performance, however, programs which use the network are charged on a per packet basis.

Operator Interaction

The computer operator may force a connection (Telnet pair actualy) closed, or the operator may spy on the traffic flowing on a connection. The operator can stop, start ,and reinitialize the NCP. The NCP will report online about some types of errors detected.

Experimental Protocols

There is no provision for experimental protocols.

TIP

System

The Terminal Interface Message Processor (TIP) is a special case of a host computer in the ARPANET. The TIP is an extension of an IMP to perform the host to host and Telnet protocols for a set of up to 63 terminals. This is done in the Honeywell 316 computer with a total of 28 kilowords of core memory, 16 kilowords for the IMP and 12 kilowords for the TIP.

NCP

points of comparison with the general model

System Calls

Because of the specialization of the TIP to the User-Telnet function it is not clear that this is a relevant question. The human user has commands available which are reflected by program actions to listen (RFH, RFS), connect (STH, STS, ICP), send and receive (implicit), close (C), and interrupt (SS). In addition the reset host to host command can be sent.

Return Characteristics

The internal mechanism for communication between program modules is to set a flag and give up.

Programming Languages

The only programming language used in the TIP is an assembly language.

RFC Queueing Policy

No queueing of RFCs is done by the TIP.

Timeout Policy

Local requests for connection are timed out after 45 seconds. A matching CLS is timed out after 45 seconds. A RFNM is timed out after 30 seconds.

Connection States

The TIP uses a eight state description of a connection: 0) try to open, 1) RFC sent, 2) RFC received -- try to reply, 3) solid connection, 4) try to close, 5) CLS sent, 6) CLS received -- try to reply, 7) no connection. The information about allocation wait and RFNM wait is managed with flags.

Allocation Policy

The TIP follows a strict policy of allocating exactly the space available in per terminal dedicated buffers. The initial allocation is one message and the buffer size number of bits.

The total buffer space is 4000 words of 16 bits each. For each terminal there is one buffer for data from the terminal to the IMP and two equal sized buffers for data from the IMP to the terminal. These buffers can be tailored to the set of terminals specifically for each TIP.

Interrupt Treatment

The interrupt does not serve any great purpose in this special environment since the only "process" to be interrupted is the human user. However the TIP does process the INS command.

Retransmission Policy

The TIP does retransmit a message if it receives an incomplete transmission response from the IMP. The TIP will attempt retransmission indefinitely.

Error Treatment

The TIP neither keeps a log of ERR messages received or sends ERR messages when it detects protocol errors. However some protocol violations and program traps cause information to be sent to the Network Control Center (NCC). In these cases the TIP program keeps going generally by faking the missing event or ignoring the extra event.

Measurement and Status Information

There is no provision for measurement or status information in the TIP itself. The NCC and the Tenex supported RSEXEC program however do provide status information.

Operator Interaction

The provision for operator interaction is via a debugging program resident in the IMP. This program can be accessed either from a dedicated local teletype or from the NCC via the network. This allows the operators at the NCC to examine and change any memory cell in the TIP. This is then a quite powerful tool, and allows the NCC operator to close a connection, force an allocate, examine the data buffers, and many other functions.

The operators at the NCC can suspend and continue operation of the TIP, or they can cause the program to be reloaded over the network from the NCC.

Experimental Protocols

There is no provision for experimental protocols in the TIP.

ANTS

System

The ARPA Network Terminal System (ANTS) was conceived at the University of Illinois Center for Advanced Computation as a flexible means to interface a wide range of interactive terminals and perpherial devices to the ARPA Network. The system has been constructed with this goal in mind, and communication between program modules receives careful attention thruout the system design.

This is a small system and it is not designed to permit users to run programs on it, but rather to enable users to reach larger computers in the ARPANET.

The system is implemented on a Digital Equipment Corporation PDP 11/50, but it is designed to run on a range of the PDP 11 series. There are several instances of the system running on the 11/45.

The version of the system surveyed is the ANTS MARK II system, a redesigned and reimplemented system based on the "quick and dirty" ANTS MARK I.

NCP

The NCP is the central part of this system, the system is tailored to the needs of the host to host protocol. Communication between program modules is by means of data paths which use a flow control mechanism very similar to the host to host protocol.

Points of comparison with the general model

System Calls

All of the suggested system calls are available to programs which call on the NCP.

Return Characteristics

The system call is a form of interprocess communication which can be best thought of as message and reply. The call (message) is either blocking or nonblocking at the callers request. The reply will wakeup a blocked process.

Programming Languages

The system programming language is PEESPOL, an ALGOL like language. The programs which call on the NCP are also written in PEESPOL.

RFC Queueing Policy

Requests for connection are queued if the socket is assigned, otherwise the request is refused. The only limit to queuing of requests is the lack of memory space.

Timeout Policy

There are no timeouts.

Connection States

The suggested states are used with the addition of several states in the close sequence for conditions such as "CLS received but data to be read by local process", Also the states RFNM WAIT and ALLOCATION WAIT are represented by a separate bit and the allocation value entries respectively.

Allocation Policy

The message allocation is set to either 100 messages and kept near that value or set to the largest value, in either case the message allocation plays no important role in flow control. The bit allocation is an exact transform of the data path allocation set up between the NCP and the calling program.

Interrupt Treatment

There is no pseudo interrupt feature in the system communication scheme.

Retransmission Policy

The NCP does retransmit a message when the IMP responds with either an incomplete transmission or a data error. The same message will be retransmitted a unlimited number of times.

Error Treatment

When protocol errors are detected ERR messages are sent, and both ERR messages sent and received are logged on the operators console.

Measurement and Status Information

The NCp does maintain a table of "hosts up" on the basis of recent communication and resets. It is also possible for the operator to determine which connections

are open, No measurement facilities are provided.

Operator Interaction

The operator is informed of every Telnet connection pair established. There are no current provisions for operator intervention with the operation of the NCP.

Experimental protocols

There is a simple way to route messages to and from program modules that are using experimental protocols.

DEC PDP10 Monitor

System

The Center for Computer-based Behavioral Studies (CCBS) at the University of California, Los Angeles operates a DEC PDP-10 Computer with a modified version of a DEC supplied operating system. The system is configured with a PDP-15 processor to act as a terminal concentrator. The PDP-10 and the PDP-15 communicate via shared memory.

NCP

The network control program is implemented in the PDP-15. The PDP-15 supports local user access to the network with a User-Telnet program, as well as allowing remote users to access the PDP-10 via Server-Telnet, or Server-FTP.

Points of comparison with the general model

System Calls

At the time of the survey there was no means for user written programs to access the network.

Programming Languages

No programming languages have provisions for calling on network functions. The NCP itself is written in assembly language.

RFC Queueing Policy

Unmatched RFCs are not queued.

Timeout Policy

The listen of the Server-Telnet "process" is never timed out. Requests for connection from other hosts that do not match local requests are refused at once. A missing matching CLS is timed out after 30 seconds, as is a missing matching ERP.

Connection States

The connections states are the same as those in the general model, except that the closed state is not explicitly represented since at that stage the table entry can be deleted. The allocation wait state is not explicity a state since this information is checked by consulting the allocation value.

Allocation Policy

The allocation policy is to allocate the largest possible values with the expectation that data will be processed faster than it can be sent. Additional allocate messages are sent when the message value falls to 16 messages, then the allocation is incremented to 100 messages and the maximum possible number of bits.

Interrupt Treatment

For Telnet connections (the only type presently supported) all data is forwarded to the PDP-10 as if it were from a local terminal. The PDP-10 either processes the data or discards it if there is buffer overflow, but all "important" characters (e.g. control c) are examined and are

effective. In this situation the processing of the network interrupt signal is irrelevant.

Retransmission Policy

No retransmission of messages occurs.

Error Treatment

When a protocol error is detected an ERR command is sent. When an ERR command is received it is ignored. As a debugging tool two circular buffers (one for input one for output) are kept of the most recent network traffic.

Measurement and Status Information

There is thus far no measurement of the NCP performance. The status data is limited to the currently connected (and logged in) Telnet users, but there is consideration being given expanding this to have status data on the current connection state and allocation values.

Operator Interaction

The operator is informed of most NCP program failures, and of network failures. The operator can reinitialize the NCP, and can remove the NCP from the system. The operator can inspect the running NCP using a DDT debuging program in the PDP-15.

Experimental Protocols

There is no provision for experimental protocols.

BKY

System

The Lawrence Berkeley Laboratories operate a system composed of three principal processor and several peripheral processors. The major machines in the complex are Control Data

Corporation products. There is a CDC 6400, a CDC 6600, and a CDC 7600. These machines are interconnected with 12 megabit per second channels and share a commom disk file. There is a front end computer connected to both the 6400 and 6600 for supporting access by interactive users. The front end can handle up to 128 lines at up to 9600 bits per second. Also connected to the 6600 is an IBM photostore of 10^12 bits capacity.

The operating system for this complex is necessarily distributed, but the major portion is on the 6600. The system actually executes in the set of perpheral processors on the 6600 with tables and data in the 6600 memory.

NCP

The NCp as part of the operating system executes in a perpheral processor of the 6600. The NCP has table and data space in the 6600 memory.

Points of comparison with the general model

System Calls

The user programs have available system calls for all the suggested functions except echo and status, in addition there are calls for assigning and releasing a socket, for determining a unique socket number.

Return Characteristics

The system calls are nonblocking and wakeup the program when completed. A program can determine the outcome of a system call by examining information posted in the users space.

Programming Languages

programs written in Fortran can access the network through special subroutine calls.

RFC Queueing Policy

All requests to assigned sockets are queued.

Timeout Policy

An unmatched request for connection will be timed out in 4 minutes. An unmatched echo or reset or close will be timed out in 2 minutes.

Connection States

The states used are the same as those in the general model (a few have different names) with a few elaborations in the close sequence. The states for ALLOCATION WAIT and RFNM WAIT are not used as distinct states, but separate bit flags are used for these conditions.

Allocation Policy

The allocation policy is to allocate exactly the available portion of a circular buffer in the users address space on a per connection basis. The initial size of this buffer and hence the allocation is chosen by the user. The message count part of the allocation is kept positive by the NCP.

Interrupt Treatment

The NCP increments an interrupt counter in the users argument and result area upon receipt of a host to host interrupt signal. This is effective in the case of Server-Telnet since the program receives a wakeup from the system every half second and when the INS is received to check for such conditions.

Retransmission Policy

The NCP retries the transmission of a message up to 5 times if the imcomplete transmission response is received.

Error Treatment

The NCP does not send error messages, but it does log a wide variety of events including detected errors and ERR messages received.

Measurement and Status Information

The log (called the NCP dayfile) receives information on each connection as it is terminated including usage statistics. As the NCP was just completed at the time of this survey no measurements have been done. There is quite a bit of information that could be extracted from the log (dayfile) to provide measurements of various aspects of the NCP performance and usage, but this has not been done.

Operator Interaction

The operator can examine the log (dayfile) and can control the execution of the NCP (start or stop, reinitialize or remove), the operator can also kill the job associated with a particular Telnet connection pair, or examine the core of any job (including the NCP).

Experimental Protocols

The NCP does not now have any special provisions for experimental protocols.

6.0 Physical Characteristics

TENEX

programming and Maintenance Costs

The NCP was written by two top systems programmers working together for two months for a total of 16 man-weeks, including the time spent debugging. The machine time used is estimated to be about 40 connect-time hours.

Program maintenance is estimated to be about three man-days per month.

Program Size

code size

The program is about 4K 36 bit words or 144,000 bits. This includes a portion of the Server-Telnet function.

table size

The space set aside for tables is 1512 words of 36 bits or 54,432 bits.

buffer size

The total buffer space reserved to the NCP is 4K words of 36 bits or 144,000 bits.

total size

The total of the above figures is 9.5K words of 36 bits or 342,432 bits.

Performance Measurements

A measurement of the transmission and CPU bandwidth was made on an unloaded TENEX system in November 1972, the TENEX system has been improved since that time especially by reducing the input and output system overhead.

Data was sent from a user program through the NCP to the IMP and then back the reverse path to the same user program. When sending short blocks (50 to 200 bytes) of 8 bit bytes a throughput of 20 to 30 kilobits per second (KBS) was achieved. Sending large blocks (400 to 800 bytes) of 36 bit bytes a throughput of 60 KBS was achieved <Murray>.

Multics

Programming and Maintenance Costs

The programming and debugging effort required two man-years. The program was operational for most of the development period yet not complete. The continuing maintenance requires about 2 man-days per month.

Program Size

code size

The program size is 22K 36 bit words or 792,000 bits.

table size

The tables are included in the preceeding program size.

buffer size

The buffer space is 3K 36 bit words or 110,592 bits.

total size

The total size is then 25K 36 bit words or 902,592 bits.

It should be noted that of this total only 5K words of program and 3K words of buffers are wired down, the remainder is paged.

Performance Measurements

Very little performance measurement has been done, but it is known that typical throughput using the file transfer protocol has been 12KBS with Tenex and 8KBS Multics to Multics. (A reason for the low value Multics to Multics is that Multics restricts the maximum output message size to a one packet message, this to avoid an old IMP or IMP-host interface bug.)

Another measure of interest is system overhead: input or output with the network cost (when last measured some time ago) 1.2 times the same input or output operations to local devices. This value may have decreased due to changes since the time of the last measurement.

IBM 360/75 MVT

Programming and Maintenance Costs

The NCP was written by one top systems programmer in 26 weeks, including debugging time. The maintenance of the NCP requires about

4 man-days per month. About $3500.00 worth of computer time was used for testing the NCP.

Program Size

code size

The code of the NCP is 52K 8 bit bytes or 416,000 bits.

table size

The total size allocated for tables is 53K bytes or 424,000 bits.

buffer size

The buffers are included in the tables.

total size

The total size is then 105K bytes or 840,000 bits.

Performance Measurements

A throughput of 70KBS has been achieved in sending a large core resident block of data to the IMP and back. This transfer used 2 percent of the CPU bandwidth.

It takes one tenth of a second to send a full 8000 bit message from a user program to the IMP and back to the user program.

IBM 360/91 MVT

Programming and Maintenance Costs

The NCP programming required 1.5 man-years, plus additional time creating the environment for the protocol programs (called the ICT environment). Maintenance requires about 1 man-day per month.

Program Size

code size

The program is 15K bytes or 120,000 bits.

table size

The tables are dynamically allocated, and consist of the following sections: for each activly communicating foreign host there is a 32 byte pointer block and a 256 byte link table, for each connection there is an 80 byte table. At a time of typical use this might require 5K bytes of table space or 40,000 bits.

buffer size

The buffer space is also dynamically allocated, typical allocations being 256 bytes per connection. At a time of typical use this might require a total of 5K bytes or 40,000 bits.

total size

The total size in typical use is 25K bytes or 200,000 bits. (However, it should be noted that the NCP is run in a 140K byte region 1,120,000 bits).

Performance Measurements

No careful measurement of NCP performance has been done but it is known that on a typical day there may be on the average 3 Network RJS users and 6 Network TSO users and that over the 8 hour prime shift on such a day about one and one half percent of the CPU usage can be charged to the NCP.

IBM 370/158 OS/MVT

Programming and Maintenance Costs

The NCP is based on the UCSB program. Modification to adapt the NCP to this environment took four man-weeks. Maintenance of the NCP requires about 1 man-day per month.

Program Size

code size

The size of the code is 35K 8 bit bytes or 280,000 bits.

table size

The size of the tables is 12K 8 bit bytes or 96,000 bits.

buffer size

The buffers are included in the tables.

total size

The total size is then 47K 8 bit bytes or 376,000 bits.

Performance Measurements

NCP performance measurements are not part of the normal procedures, but one day of NCP usage was analyzed. On a typical day in the 12 hours from 6 am to 6 pm there were 118 network sessions which involved a total data transfer of 5.96 million bytes. During this same period the NCP job can be charged for 484 CPU seconds (including interrupt handling). These facts suggest the following conclusions: the NCP consumes 1.1 per cent of the CPU, and the NCP overhead cost of transfering 1000 bytes is about .088 seconds.

IBM 370/145 VM

Programming and Maintenance Costs

The NCP is under development still but the estimated time to convert the UCSB supplied program to this environment is six man-months.

Program Size

The program is not complete, so the finished size is unknown but it is estimated to be about 20 per cent larger than the UCSB 360/75 MVT implementation but less than 100K bytes (800,000 bits). This appears to be impossible but perhaps the code will be 20 per cent larger but the tables and buffers will be much smaller.

Performance Measurements

No performance measurements have been done.

Burroughs 6700 MCP

Programming and Maintenance Costs

The NCP was written in three man-months. About three man-days per month are required for program maintenance.

Program Size

code size

The NCP code is about 15K 48 bit words or 720,000 bits. The NCP, Telnet, FTP and RJE programs are combined together in one module (the NETMCS) which is 25K words in size.

table size

The table space is typically 6K words or 288,000 bits.

buffer size

The buffers are included in the tables.

total size

The total size is then 21K words for the NCP or 1,008,000 bits. It should be noted that the entire NETMCS requires 31K words but can run in 10K to 16K of core due to the segmentation structure.

Performance Measurements

On one typical day the NCP used 3% of the CPU of one of the dual processors over the eight hour prime shift.

TIP

Programming and Maintenance Costs

The TIP program was written in about two months

by one person. There has been significant modification and improvement since the program was first operational however. The current maintenance requires about one to one and a half man-days per month per site, with currently 20 sites, this requires one person full time.

Program Size

code size

The program is about 6K 16 bit words or 96,000 bits.

table size

The tables require about 2K words or 32,000 bits.

buffer size

The buffer space is 4K words or 64,000 bits.

total size

The total size for the above is then 12K words or 196,000 bits.

Performance Measurements

Because the TIP is oriented to terminal use the throughput measurements made with larger host are not applicable. The interest in the case of the TIP is the total or maximum combined throughput. BBN has arrived at a formula that indicates the throughput constraints on the TIP. Recalling that the TIP is an IMP too, the competition for CPU use is between traffic due to IMP to IMP phone lines (L), hosts (H) and the collection of terminals (T). The formula is $L + H + 15T < 600$ KBS. Where H, L, and T are full duplex rates, that is, a 50KBS full duplex line counts as 50 in the formula. The maximum total terminal traffic is about 80KBS, for example eight 9600 baud display terminals doing output only (assuming sufficient buffer space is available).

ANTS

Programming and Maintenance Costs

The ANTS MARK II NCP was designed in two man weeks, coded in six man weeks, and debugged in seven man weeks, for a total of fifteen man weeks. Note that the designers had the longer experience with ANTS Mark I, however.

Program Size

code size

The program is 4K words of 16 bits or 64,000 bits.

table size

Tables are dynamically assigned from the system free memory pool, but a typical load might require 1000 bytes of 8 bits each or 8000 bits.

buffer size

There are IMP input and output buffers of 256 bytes each for 4096 bits total.

total size

Thus the total size is about 76,000 bits.

Performance Measurements

There have been no performance measurements as such, but it is noted that the system can support 9600 bit per second terminals with no apparent degradation of the terminal speed.

DEC PDP10 Monitor

Programming and Maintenance Costs

The NCP and Server-Telnet programs together were written in six man-months. The User-Telnet and Server-FTP are included in the code in the PDP-15. The program was only recently completed and a normal maintenance level has not been established.

Program Size

code size

The NCP program is 12K 18 bit words or 216,000 bits.

table size

The table size is 12K words or 216,000 bits.

buffer size

The buffers are included in the tables.

total size

The total size is then 24K words or 432,000 bits.

Performance Measurements

No performance measurements have been performmed.

BKY

Programming and Maintenance Costs

The NCP coding effort took one man-year including a month of debugging time.

Program Size

code size

The code is about 8K words of 12 bits each, or 96,000 bits.

table size

The tables are reserved 2K words of 60 bits each, or 120,000 bits.

buffer size

The NCP also has an input and output buffer to the IMP, each capable of containing one

full message, or a total buffer of 16,000 bits.

total size

The total number of bits then is 232,000.

Performance Measurements

Since the NCP was completed just at the time of the survey no performance measurement is available.

Summary Chart

The following chart summarizes, by system, the size of the program, tables, buffers, and total of these, for each NCP.

	Code	Table	Buffer	Total
TENEX	144	54	144	342
Multics	792	-	110	902
360/75 416	424	-	840	
360/91 120	40	40	200	
370/158	280	96	-	376
370/145	-	-	-	800
B6700	720	288	-	1008
TIP	96	32	64	196
ANTS	64	8	4	76
DEC10	216	216	-	432
BKY	96	120	16	232

Note: The table entries are thousands of bits.

7.0 Summary

The findings of the survey of network control programs are presented in this section. These remarks might best be labeled opinion rather than conclusions because of the sparseness and inconsistency of the evidience.

First note that the ARPANET works. The NCPs in the host do communicate among themselves, and user level programs in the various hosts do communicate with other user level programs in other hosts.

This has been accomplished by a uncoordinated group of

systems programmers working for diverse organizations geographically distributed. These programmers established a working group and used a series of technical memos (called Request for Comments) to exchange information and viewpoints on the developing protocols. Several meetings were held to gather concensus on protocol issues and adopt proposals for implementation. The specifications produced by this process are loose in several respects, the two most important areas being the functional specification of the user process to NCP interface (system calls) and the requirement for queueing of requests for connection.

The current host to host protocol has several flaws in addition to the weak positions on the user interface and queueing cited above. Error control is present in a very limited sense, the ERR command is useful for reporting detected protocol violations, but such violations should arise only due to program bugs, and the ERR command is not employed by many of the implementations. The hosts should have a means of ensuring that the data transmitted is received correctly and that the messages transmitted all arrive. The first problem could be attacked by an end to end checksum, and the second by a message sequence number.

Another problem is in the flow control aspect, while the allocate mechanism is constructed to allow quite flexible buffer management, many of the implementations have chosen to use a very simple strategy, often one that requires an allocate for each message, thus insuring a host to host round trip delay between messages of the same conversation. Similarly the host to host protocol requires a host to destination IMP round trip delay between each message of the same conversation by requiring the NCP to wait for a RFNM to each message on a logical link before sending another message on the same logical link. These constraints limit the throughput achievable on any particular connection.

The performance measurements of network control programs has been very spotty and informal. There should be some consistency and regularity to performance measurements. There needs to be a standard set of experiments defined and these experments should be performed regularly.

Generally network control programs implemented as core resident modules are capable of higher throughput rates and cause smaller delays, this is not necessarily due to better coding but usually due to avoidance of the paging or swapping overhead incurred with a nonresident program. On the other hand the nonresident program may have the advantage of not tying up core memory when not in use and may be able to have only the active subroutines in core thus using a smaller portion of core even when in use.

There is no authority to designate network control programs complete or correct. There should be a mechanism for a third party to review and certify NCPs, as the situation stands each implementation is correct only on the word of its implementer.

The documentation of network control programs is spotty, often there is no documentation (other than the code) of the program, however see <BBN91>, <White1>, <Winett>, <Wong>. Further it is sometimes difficult to find documentation on the user program interface (system calls). This latter problem is serious in that it tends to prevent users from constructing inovative applications of the network facilities.

A.1 Glossary

Abbreviations

AEN
another eightbit number

ALL
A host to host protocol command to allocate buffer space to the sending NCP in the receiving NCP.

ANTS
ARPA Network Terminal System

ARPA
Advanced Research Projects Agency of the Department of Defense

ARPANET
Advanced Research Projects Agency Computer Network

ASCII
American Standard Code for Information Interchange. The character encoding used in the network.

BBN
Bolt, Beranek, and Newman, Inc. Cambridge, Massachusetts

BKY
The operating system used at Lawrence Berkeley Laboratories for the CDC 6600 computer.

CCBS
Center for Computer-based Behavorial Studies at University of California, Los Angeles.

CDC
Control Data Corporation

CLS
A host to host protocol command to close the connection.

DEC
Digital Equipment Corporation

DMS
Dynamic Modeling System. A host computer on the ARPANET at MIT.

EBCDIC
Extended Binary Coded Decimal Interchange Code. The character encoding used primarily by IBM computer systems.

FCP
File Control Program

FTP
File Transfer Protocol

IBM
International Business Machines
ICP
Initial Connection Protocol
IPC
interprocess communication
IMP
Interface Message Processor
LBL
Lawrence Berkeley Laboratory
MCP
The operating system for the Burroughs 6700.
MIT
Massachusetts Institute of Technology
Multics
Multiplexed Information and Computing Service, the operating system for the Honeywell 6180 computer designed and implemented at MIT's project MAC.
NCC
Network Control Center at BBN.
NCP
Network Control Program
NIC
Network Information Center at the Augumentation Research Center of Stanford Research Institute, Menlo Park, California.
OS/MVT
An IBM operating system for the 360 series of computers.
PDP
Programmed Digital Processor
RAND
The RAND Corporation
RFNM
Request For Next Message
RFC
request for connection
RTS
Receiver to Sender request for connection. A host to host protocol command.
SDC
System Development Corporation
STR
Sender to Receiver request for connection. A host to host protocol command.
TCP
Terminal Control Program
TENEX

The operating system designed and implemented by BBN for the DEC PDP10 computer.

TIP
Terminal Interface Processor

UCLA
University of California, Los Angeles

UCSB
University of California, Santa Barbara

UCSD
University of California, San Diego

UI
University of Illinois

VM
The IBM operating system for the 370 series of computers.

Terms

another eightbit number
The user program specified portion of the socket number.

ARPA Network Terminal System
A particular small host system designed to interface a wide variety of terminals and peripherals to the ARPA network. This system was designed and implemented by the Center for Advanced Computation at the University of Illinois. The system operates on a DEC PDP11 computer.

connection
The form of interprocess communication provided to the user level processes by the NCPs in the host computers. A connection is a logical simplex stream of data from one port of one process to another port of another process in the network.

control message
A message (of the regular type) that contains host to host commands.

File Control Program
That module in the operating system that controls the access to files by the user processes.

File Transfer Protocol
The protocol that specifies the communication interaction required to move blocks of data (files) between host computers in the network.

full duplex

A channel in which data can flow in both directions simultaneously.

half duplex
A channel in which data can flow in both directions, but may only flow in one direction at a time.

header
The control information at the begining of a packet.

host
A computer attached to an IMP. A host does not necessarily offer services to other computers in the network.

Initial Connection protocol
The sequence of actions taken by user level programs to establish a pair of connections between a user program and a service program.

Interface Message Processor
The packet routing computers which are the nodes of the ARPA network. An IMP is connected to between 1 and 5 other IMPs and to between 0 and 4 hosts.

interprocess communication
The facility for one process to communicate with another process.

leader
The first 32 bits of a message, containing address and control information. The most important fields in the leader are: the message type, the link number, and the host address.

link number
A parameter in the leader that selects a logical communication channel between the source and destination hosts.

message
The unit of transmission between a host and an IMP, up to 8096 bits.

Network Control program
The program module added to the operating system that interfaces the user processes to the IMP and controls the communication between hosts by implementing the host to host protocol.

packet
The unit of transmission between IMPs, up to 1008 bits.

port
The input or output identifier associated with a particular data stream of a process. For example a Fortran logical unit number or a data set

reference number, or an assembly languague data control block.

prefix
A 40 bit block immeadiately following the leader and containing the byte size and number of bytes of following text.

process
A program in execution with its associated address space, registers and location counter.

protocols
The rules of behavior, in particular, the allowed formats and sequences of communication between two processes.

regular message
A message from the host to the IMP or from the IMP to the host that is the normal data carrying type. When following the host to host protocol a regular message may carry either a set of control messages or a users data.

request for connection
Either of the host to host protocol commands STR or RTS.

Request For Next Message
A message from the IMP to the host indicating that the previously sent message on the same link number as this RFNM was received by the destination IMP and has begun transmission into the destination host.

socket
The terminus of a connection. The network wide name of an input or output port associated with a process.

Telnet
The protocol (or the programs that implement it) that specifies the communication interaction such that a user on one system gains access to the services of a second system as if he were a local user of the second system.

Terminal Interface Processor
An extension of the IMP to allow a variety of terminals to access the ARPA network. The TIP contains the NCP and User-Telnet programs as well as the terminal handling code in the same processor as the IMP. In addition there is a BBN constructed multi-line controller for up to 63 terminal.

simplex
A channel in which data can flow in one direction only.

Terminal Control Program
: The program module in the operating system that controls the flow of data between the interactive terminals and the user processes.

virtual
: Being something in effect, but not in actuallity. For example a virtual memory might be one that a user process accesses as if it were a large linear core resident set of memory words, when in actuallity the memory is managed by the operating system using paging and maping such that only a small portion of the users set of memory words are in core at any particular time.

A.2 References:

<BBN1822>
Bolt, Beranek, and Newman, Inc. "Specifications for the Interconnection of a HOST and an IMP," Report No. 1822, Cambridge, Massachusetts, Revised March 1974. Also contained in NIC 7104.

<BBN91>
Bolt, Beranek, and Newman, Inc. "The Terminal Interface Message Processor Program," Technical Information Report No. 91, Cambridge, Massachusetts, Revised November 1973.

<Braden>
Braden, R.T., and S.C. Feigin. "Programmers Guide to the Exchange," Campus Computing Network, University of California, Los Angeles, California, March 1972.

<Bressler>
Bressler, R., and D. Walden. "A Proposed Experiment with a Message Switching Protocol," RFC 333, NIC 9926, Augmentation Research Center, Stanford Research Institute, Menlo Park, California, May 1972.

<Crocker>
Crocker, S.D. et.al. "Function-Oriented Protocols for the ARPA Computer Network," AFIPS Conference Proceedings, 40:271-279, SJCC, 1972.

<Heart>
Heart, F.E. et.al. "The Interface Message Processor for the ARPA Computer Network," AFIPS Conference Proceedings, 36:551-567, SJCC, 1970.

<McKenzie>
McKenzie, A.A. "Host/Host Protocol for the ARPA Network," NIC 8246, January 1972. Also contained in NIC 7104.

<Murray>
Murray, H. "TENEX Bandwidth," RFC 415, NIC 12407, Augmentation Research Center, Stanford Research Institute, Menlo Park, California, November 1972.

<Newkirk>
Newkirk, J. et.al. "A Prototypical Implementation of the NCP," RFC 55, NIC 4757, Augmentation Research Center, Stanford Research Institute, Menlo Park, California, 19 June 1970.

<NIC7104>
Network Information Center, "Current Network Protocols," NIC 7104, Augmentation Research Center,

Stanford Research Institute, Menlo Park, California, Revised June 1973.

<Ornstein>
Ornstein, S.M., et.al. "The Terminal IMP for the ARPA Computer Network," AFIPS Conference Proceedings, 40:243-254, SJCC, 1972.

<Roberts>
Roberts, L.G. and B.D. Wessler. "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, 36:543-549, SJCC, 1970.

<White1>
White, J.E. "An NCP for the ARPA Network," Computer Research Laboratory, University of California, Santa Barbara, California, December 1970. Also available as NIC 5480.

<White2>
White, J.E. "Dynamic Extension of OS/360 for a Network Environment," Computer Research Laboratory, University of California, Santa Barbara, California.

<Winett>
Winett, J.W., and A.J. Sammes. "An Interface to the ARPA Network for the CP/CMS Time-Sharing System," Technical Note 1973-50, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts, November 1973.

<Wong>
Wong, J. "Network Control Program (NCP)," SEX Notebook Section 25.3, SPADE Group, Computer Networks Research Project, Computer Science Department, University of California, Los Angeles, California, January 1972.

MTR6722

(J24302) 24-OCT-74 13:48;;;; Title: Author(s): Jonathan B. Postel/JBP; Distribution: /JBP([ACTION]) ; Sub-Collections: SRI-ARC; Clerk: JBP;