Network Working Group
Request for Comments: 705
NIC# 33644

FRONT - END PROTOCOL

B6700 VERSION

2 September 1975

This is a working document which has been developed as the specification and guideline for design of a Burroughs B6700 attachment to an ARPA-Style network.

The approach is to utilize a front-end processor with a new protocol for network operation. That protocol, described herein, has been built upon the concepts expressed by M.A. Padlipsky, et al, in NIC# 31117, RFC# 647.

This proposed, site-specific, FEP implementation is the work of Gerald Bailey and Keith McCloghrie of NSA and of David Grothe of ACC. It has already sustained some corrections provided by MAP. It will be helpful if interested networkers will review and provide comments to us.

Comments to BRYAN@ISI.

Roland Bryan - ACC

Network Working Group
Request for Comments: 705
Front-End Protocol: B6700 Version

WORKING DOCUMENT

FRONT-END PROTOCOL

PREFACE

This document describes the protocol to be used for connecting a general-purpose computer system (host) to an ARPANET-like network via a "front-end" computer. The main body of the document is aimed at a reader who is not conversant with all the details of network protocols. However, a paragraph marked with [n], refers a reader familiar with network protocols to the n-th item of Appendix A which will amplify that particular paragraph. Further information on the network protocols referred to in this document can be obtained from the Network Information Center.

Appendix B contains diagrams showing the transitions between the different connection states. Appendices C and D give the implementation details of this protocol in the Front-End and the Hosts.

This protocol is predicated upon the assumption that for each host, a line protocol, at a lower level, will be established between the device-driver modules in the Host and the Front-End, and that this line protocol provides Front-End Protocol with error-free transmissions.

INTRODUCTION 2

A host computer may be connected to a network for a variety of reasons. Network connection may be an attempt to expand the usefulness of the Host to the community of users which it serves by making network resources available to them. Conversely, the services which the Host provides may be made available to a larger community of users, with the network providing

the method of access to those services.

In order for members of a network community to communicate in an intelligent

way, there must exist a set of protocols. The implementation of these protocols in a host computer is typically called the Network Control Program (NCP). The size and complexity of the NCP is proportional to the number and

complexity of protocols which it implements. For an ARPANET like network, both the number and complexity are substantial.

WORKING DOCUMENT

WORKING DOCUMENT

A host which directly connects into the network must assume the responsibili
ty
for implementing this set of protocols. That is the "price of admission"
to become a network host. It is not necessary to implement every protocol
and every option in every host, but even in the simplest case -- implementat
ion
of an NCP is not a small task. The intrusion into the normal operating
environment of the host is also not small.

An alternative method for network connection is to connect the host to some
intermediate processor, and in turn, directly connect that processor to the
network. This approach is called "Front-Ending." There are many arguments
which may be posed to justify a host connection to a network through a front
-
end processor. The most obvious being that the responsibility for
implementation of the network protocols (the NCP) can be delegated to the
front-end (FE), thereby reducing the impact on the host.

The purpose of this document is not to justify Front-Ending as a philosophy,
but rather, to introduce a protocol for communications between a host and
a front-end processor which is providing it network access. The Front End
Protocol (FEP) is intended to permit the host to make use of the network
through existing protocols, without requiring that it be cognizant of the
complexities and implementation detail inherent in their execution.

The FEP is sufficiently general to permit its implementation in the host
to be in terms of the function the host is performing, or the services
which it is providing. Of primary consideration in specification of FEP
was that it must provide the host with a sufficiently robust command
repertoire to perform its network tasks, while buffering it from the
details of network protocols.

Before a detailed description of the command structures is undertaken it
seems appropriate to introduce several of the concepts upon which the FEP
is predicated.

The following section serves to briefly describe the FEP commands, and to
elaborate on the concepts of addressing and types of connections provided.

WORKING DOCUMENT

Commands (General) 3b

1. BEGIN Command

This command is sent from the host to the front-end processor. Its function is to direct the establishment of one or more network connections. The type and number of connections is specified in the BEGIN command string.

2. LISTEN Command

Through this command the host indicates its willingness to accept requests for connection arriving from other hosts. It directs the front-end processo
r
to LISTEN for any such connection requests. The number and type of connections are specified in the command string.

3. RESPONSE Command

The front-end processor uses the RESPONSE command to indicate to the host th
at
a particular path specified in a BEGIN or LISTEN command is now open or that the open attempt failed.

4. MESSAGE Command

Message text passing between the host and its front-end processor is sent in this command string. The MESSAGE command is bi-directional, and is the same

for host or front-end.

5. INTERRUPT Command

The INTERRUPT command is sent by either the host of FE. Its most common use
is
to convey that the user wishes to terminate what he is doing - i.e., he has depressed the Control-C, ATTN, or INT key.

6. END Command

One or more connections may be closed by either the FE or the host issuing this command. The connection(s) which are affected by the action of the END

are specified in the command string.

7. REPLY Command

This command is required to be sent by both the host and FE to acknowledge receipt of all command types (except REPLY). The success or failure of the command being acknowledged is conveyed in the REPLY command string.

WORKING DOCUMENT

Connections 3c

In order to engage in a meaningful conversation, the parties involved must be connected. A network connection is defined by the ARPA Host-Host Protocol document (Nic #8246) as follows : "A connection couples two processes so that output from one process is input to the other. Connections are defined to be unidirectional, so two connections are necessary if a pair of processes are to converse in both directions." The components of a connection, the sockets, are defined: "... a socket forms the reference for one end of a connection, and a connection is fully specified by a pair of sockets. A socket is identified by a Host number and a 32-bit socket number. The same number in different Hosts represents different sockets."

The existing network protocols incorporate prescribed strategies for selecting socket assignments, pairing sockets to form connections, and in the number of connections required to implement the protocol.

Conversations, in most cases, are bi-directional. Thus to simplify the Host's procedures in these cases, FEP permits duplex connections on which the Host can both send and receive. Send only and Receive only connections are also available for those situations where communication is one-way.

Thus, FEP provides the flexibility to reduce complexity in the Host, in addition to accommodating existing protocols and allowing for the development of new protocols.

Addressing 3d

Conversations in FEP are uniquely identified at initiation by some combination of Host address, Index number, Path number and Socket assignment. The Host address and Socket assignment are required to form the connection(s); thereafter the Index and Path are sufficient to identify the conversation.

Host Address

If, through the BEGIN command, the local Host explicitly directs the creation of network connection(s), it must specify the address of the foreign host to which it desires communication. If the local host indicates a willingness to communicate, through the LISTEN command, the Front-End processor will supply

the address of the connecting foreign host(s) in its RESPONSE command(s).

WORKING DOCUMENT

WORKING DOCUMENT

Socket

A socket is either a send socket or a receive socket. This property is called the socket's gender. The sockets at either end of a network connection must be of opposite gender. As previously defined a socket forms the reference for one end of a network connection. To the extent possible, the FEP shields the Host from the responsibility of assigning sockets for individual conversations. However, because the socket is a fundamental part of the addressing mechanism of the network, the Host may need to be aware of socket assignments when establishing connections.

It is through a "well-advertised" socket that a host provides services to other members of the network community. The Initial Connection Protocol (ICP) [1] is used to first connect to the well-advertised socket in order to exchange the number of a presently unused socket which is then used for the connections required so that the well-advertised socket can be freed for others attempting to connect.

When establishing a conversation (with a BEGIN or LISTEN command) the Host indicates in the value of the CONN-TYPE field whether the socket specified is to be employed directly, or to be used as an initial connection socket.

Index/Path Addressing 3e

Indexes are values assigned by the local Host to identify network conversations. When conversations are established (with the BEGIN or LISTEN commands) the Host must specify an index value. This value will be associated with the resultant conversations for their duration.

It is often necessary to affiliate conversations [2]. To accommodate this, data paths are defined such that each index has one or more path(s) associated with it (a path can not exist except as a subordinate to an index) and all network communication is transmitted on some path.

The maximum number of indexes which may be in use at any one time, and the maximum number of paths within one index are installation parameters.

Index 0 is reserved for controlling other indexes, and logically represent t
he
"pipe" through which all other indexes "flow."

WORKING DOCUMENT

Addresses in FEP command strings are conveyed by the pair of fields "INDEX" and "PATH." In commands which cause new indexes to be opened, or new data paths to be added to an existing index (BEGIN or LISTEN), the PATH field indicates the first path to be acted upon by this command. For those commands which do not create new paths or indexes, if PATH is 0, then all paths associated with this INDEX are addressed; if PATH is non-zero, only th
e
specific path within the specified INDEX is addressed.

Path Types 3f

A path can be one of three types:

a. DUPLEX - both the Host and the FE can issue MESSAGE commands on the path.

b. SEND - only the Host can issue MESSAGE commands on the path.

c. RECEIVE - only the FE can issue MESSAGE commands on the path.

The paths within an index may be a mixture of path types but one BEGIN/ LISTEN must be used for each contiguous set of the same type.

An FEP path is analogous to a network connection with the following exceptio
n.
Network connections are always simplex. This is true for paths of type SEND or RECEIVE. However, a DUPLEX path is formed by the FE connecting two local sockets to two foreign sockets. This is a "duplex connection" which is composed of two network (simplex) connections.

Modes of Establishing a Path 3g

One or more paths are established by the action of a single BEGIN or LISTEN command, with the mode specified in the CONN-TYPE field of the command. Each of the path types is established in one of two modes - directly or via ICP. The gender of the path (its ability to receive or send or both) is not affected by the mode.

When any of the path types is specified with the ICP mode, the socket value in the SOCKET field is used as the "well-advertised" socket and an actual working socket will be exchanged according to the Initial Connection Protoco
l.
When the direct mode is indicated, the specified socket is used as the worki
ng
socket.

WORKING DOCUMENT

In either mode, when multiple paths are indicated, the next higher socket number values of the appropriate gender are selected for each path. [3]

Translation 3h

When the Host sets up a path(s) (with a BEGIN or LISTEN command) it identifies what type of translation or data-mapping it requires the FE to perform on all data transmitted on this path(s). This is specified by two values - one giving the format of the data transmitted between the FE and the network, the other giving the format of the data between the Host and the FE. [4]

Flow Control 3i

All commands (except REPLYs) must be REPLYED to by the receiver. The sender is blocked from sending more commands on the same path until a REPLY has been received. The REPLY command serves two functions: it indicates the success/failure of the last transmission on the path, and it also indicates a willingness of the receiver to accept more data on that path. Receipt of any valid REPLY on an open path is sufficient to unblock it for END or INTERRUPT commands. Thus a receiver who will not (or can not) accept more data (MESSAGE commands) on a given path need not block the sender from ENDing the path if he desires. An indication of "READY" in the reply serves to unblock the path for MESSAGE commands also.

In the normal case, the REPLY performs both functions concurrently. However, when the receiver is not ready to accept more data, he can REPLY indicating only success/failure of the last command which should be sufficient to allow the sender to free the transmission buffer, requeue the command for retransmission if necessary, etc. and wait for another REPLY command announcing the receiver’s ability to accept more data.

Exceptional Conditions 3j

When a command is received and can not be executed, the REPLY command is used to notify the sender of the command. To do this, the bits of CODE field of the REPLY are set to show the CATEGORY of the error and its TYPE within that category (see Section 3h).

WORKING DOCUMENT

WORKING DOCUMENT

COMMANDS 4

Introduction 4a

All communications between the Host and the FE is performed by means of commands. The commands are given names for documentation purposes but are distinguished by the binary value of the first field of the command string. Command strings will be padded with zeros up to the next multiple of an installation defined parameter. (This value will be dependent on the capabilities of the hardware interface between the Host and the FE.)

Field lengths within a command string are specified as some number of bits. These information bits will be right-justified within the least number of bytes needed to hold them. The size of a byte will be an installation parameter which will normally be 8 bits but other values will be accommodate
d
as necessary.

The values and meanings of the CODE field of the REPLY command are given for each command within the following descriptions:

1: BEGIN 4b

Format

BEGIN INDEX PATH HOST SOCKET TRANS-TYPE CONN-TYPE NPATHS

Use

This command is sent only from the Host to the FE. Its function is to direc
t
the FE to establish one or more logical connections (paths) on the specified index between the Host and the FE.

Its use has three different modes (depending on the value of the PATH field)
:

mode (a) - to set up a new index and to direct the FE to attempt to establish network connections for the one or more paths specified within this index.

WORKING DOCUMENTS

mode (b) - to attempt to establish network connections for an existing (but at present closed) path within the already set-up index.

Mode (c) - to attempt to establish network connections for one or more new paths within the already set-up index.

Parameters

a) BEGIN is an 8-bit field with the value 1.

b) INDEX is a 16-bit field, specifying the index. Note that the value 0 is reserved for special use (see Section 4).

c) PATH is an 8-bit field, specifying the path(s) which are to be established. Its value identifies the mode of the BEGIN (see above) :

mode (a) - its value must be 1.

mode (b) - its value must be that of the path to be "re-opened."

mode (c) - its value must be exactly one greater than the current number of paths defined within this index.

d) HOST is a 32-bit field specifying the foreign host with which connections are to be established.

e) SOCKET is a 32-bit field, specifying the first or only socket at the foreign host to which connections are to be made.

f) TRANS-TYPE is a 16-bit field which directs the FE to perform this type of translation on all data (i.e. TEXT in the MESSAGE command string) sent on every path being established by this command. The first 8 bits specify the format of the data on the network side; the second 8 bits specify the format of the data on the Host side. The values assigned to the particular formats (eq. ASCII, EBCDIC etc.) are installation parameters; however, the

value 0 will always mean "bit string" and thus if either of the 8-bit sub-fields contains 0, then no mapping will be performed.

g) CONN-TYPE is an 16-bit field, specifying the type and mode of connection(s) to be established for the specified path(s). Its value informs the FE how to associate sockets with indexes/paths (see Sections 2f and 2g).

Value	Type	Mode
7	Duplex	via ICP
6	Duplex	direct
5	Receive	via ICP
4	Receive	direct
3	Send	via ICP
2	Send	direct

h) NPATHS is an 8-bit field, specifying the number of paths which this command directs the FE to attempt to establish connections for. If the BEGIN is of mode (b) then its value must be 1. Otherwise the sum of its value and the value of the PATH field is the new current number of paths plus one.

Error CODES in REPLY

Category	Type	Meaning
3	1	PATH invalid for new index
3	2	PATH invalid for old index
3	3	PATH already open
3	4	HOST unknown
3	5	TRANSLATION-TYPE invalid
3	6	CONNECTION-TYPE invalid
3	7	NPATHS invalid for old path on old index
3	8	Specified socket inconsistent with CONN-TYPE
3	9	INDEX invalid, not ready for business
4	1	No new connections - FE full
4	2	No new connections - closing down soon

WORKING DOCUMENT

2: LISTEN 4c

Format

LISTEN INDEX PATH HOST SOCKET TRANS-TYPE CONN-TYPE NPATHS

Use

This command is sent only from the Host to the FE.

Its function is to direct the FE to "listen," i.e., to hold the specified paths pending until such time as a request for connection (RFC) is received from the network to the specified local socket. then to set up connections and to respond with a RESPONSE command for each path.

Its use has three different modes (depending on the value of the PATH field) :

mode (a) - to set up a new index and to listen on the specified local socket in order to establish connections for the specified paths.

mode (b) - to listen on the specified socket in order to establish connections for the specified, existing (but at present closed) path within the already set-up index.

mode (c) - to listen on the specified socket in order to establish connections for the specified new path(s) within the already set-up index.

By use of the HOST parameter, the FE can be directed to accept RFCs from any host or only from the specified host.

Parameters

a) LISTEN is an 8-bit field with value 2.

b) INDEX is a 16-bit field specifying the index.

c) PATH is an 8-bit field specifying the first of the one or more paths which are to be held pending receipt of a RFC. Its value identifies the mode of the LISTEN (see above) :

WORKING DOCUMENT

mode (a) - its value must be 1.

mode (b) - its value must be that of the existing path.

mode (c) - its value must be exactly one greater than the current number of paths within this index.

d) HOST is a 32-bit field specifying the host from which RFCs are to be accepted; a value of 0 implies from any host.

e) SOCKET is a 32-bit field specifying the local socket on which the FE is to listen for RFCs.

f) TRANS-TYPE is a 16-bit field specifying the type of translation the FE is to perform on all data sent on every path established as a result of this command. Its values are the same as in the BEGIN command.

g) CONN-TYPE is an 16-bit field specifying the type and mode of the connection(s) to be established for the specified path(s) when an RFC is received. Its values are the same as in the BEGIN command.

h) NPATHS is an 8-bit field specifying the number of paths which this command associates with the specified index and which are to be established. If the LISTEN is of mode (b) then its value must be 1. Otherwise the sum of its value and the value of the PATH field is the new current number of paths plus one, within this index. Thus its value is the number of extra RFCs for which the FE is listening on this socket.

Error CODEs in REPLY

Category	Type	Meaning
3	1	PATH invalid for new index
3	2	PATH invalid for old index
3	3	PATH already open
3	4	HOST unknown
3	5	TRANSLATION-TYPE invalid

WORKING DOCUMENT

WORKING DOCUMENT

3	6	CONNECTION-TYPE INVALID
3	7	NPATHS invalid for old path on old index
3	8	Specified socket inconsistent with CONN-TYPE
3	9	INDEX invalid, not ready for business
3	10	Socket already in use.
4	1	No new listens - FE full
4	2	No new listens - closing down soon

3: RESPONSE 4d

Format

RESPONSE INDEX PATH CODE HOST SOCKET

Use

This command is sent only from the FE to the Host - once per path specified in
a BEGIN or a LISTEN command.

For paths specified in a BEGIN, it is sent to indicate the success or failure
of the connection attempt. For paths specified in a LISTEN, it is sent at the time when the FE has received a matching RFC and has established the connection.

The HOST and SOCKET parameters are purely informational which the Host can ignore if it so desires. Their contents are only guaranteed if the connection
attempt succeeded.

Parameters

a) RESPONSE is an 8-bit field with value 3.

b) INDEX is a 16-bit field specifying the index.

c) PATH is an 8-bit field specifying the particular path.

d) CODE is a 16-bit field indicating the outcome of the connection attempt:

WORKING DOCUMENT

WORKING DOCUMENT

Value	Meaning
0	Path successfully established.
1	Local IMP dead.
2	Foreign IMP inaccessible.
3	Foreign Host dead.
4	Foreign Host not responding.
5	Connection refused.

e) HOST is a 32-bit field specifying the foreign host to which the connection has been made.

f) SOCKET is a 32-bit field specifying the socket at the foreign host. If the connection type is simplex, then it is the only foreign socket for this path; if duplex, then it is the lower of the two foreign sockets.

Error CODES in REPLY

Category	Type	Meaning
3	11	INDEX unknown
3	12	PATH unknown
3	13	CODE invalid

4: MESSAGE 4e

Format

MESSAGE INDEX PATH COUNT PAD TEXT

Use

This command is sent by either the Host or the FE to transmit data on the specified path and index.

Parameters

a) MESSAGE is an 8-bit field with value 4.

b) INDEX is a 16-bit field specifying the index.

WORKING DOCUMENT

WORKING DOCUMENT

c) PATH is an 8-bit field specifying the path. Note that the value 0 is used in the broadcast option (see Section 3j).

d) COUNT is a 16-bit field specifying the number of bits of data in the TEXT field.

e) PAD is an n-bit field, where n is an installation parameter. It contains only padding (in the present protocol specification) and can be used to enable the host to have the TEXT field start on a convenient boundary.

f) TEXT is a field containing COUNT bits of data being transmitted on this path.

Error CODES in REPLY

Category	Type	Meaning
2	1	This option not implemented
3	12	PATH unknown
3	14	No connection opened in this direction
3	15	PATH blocked at this time, resent later
3	16	PATH suspended at this time, resent later
3	17	PATH closed
3	17	COUNT too large
4	3	Error in transmitting data, resend command
4	4	Data lost, resent command.

5: INTERRUPT 4f

Format

INTERRUPT INDEX PATH CODE

Use

This command is sent by either the Host or the FE.

Its most common use is to pass the information that a terminal user has pressed his INT (or ATTN or Control-C) key, thereby requesting his applications program to quit what it is doing for him.[5]

WORKING DOCUMENT

WORKING DOCUMENT

Parameters

a) INTERRUPT is a 8-bit field with value 5.

b) INDEX is a 16-bit field specifying the index.

c) PATH is an 8-bit field specifying the path on which the INTERRUPT is transmitted. Note that the value 0 is used in the broadcast option (see Section 3j).

d) CODE is a 16-bit field. It has no defined meanings as yet and should contain 0.

Error CODES in REPLY

Category	Type	Meaning
2	1	This option not implemented
3	11	INDEX unknown
3	12	PATH unknown
3	14	No connection opened in this direction
3	15	PATH blocked at this time, resend later
3	17	PATH closed.

6: END 4g

Format

END INDEX PATH CODE

Use

This command is sent by either the Host or the FE, to terminate a connection
.
If PATH is 0, then the index and all its paths are terminated, otherwise jus
t
the specified path of the index is terminated.

Parameters

a) END is an 8-bit field with value 6.

b) INDEX is a 16-bit field specifying the index.

WORKING PARAMETER

WORKING DOCUMENT

c) PATH is an 8-bit field containing the path to be closed, or 0 if the whole index is to be closed.

d) CODE is a 16-bit field indicating the reason for the closure:

Value	Meaning
0	Normal close
1	Retries exhausted
2	Foreign Host failure
3	Foreign IMP failure
4	Network failure
5	Local IMP failure.

The "Retries exhausted" code indicates that the FE has been retrying a transmission to the foreign host without success.

Error CODES in REPLY

Category	Type	Meaning
3	11	INDEX unknown
3	12	PATH unknown
3	13	CODE unknown
3	15	PATH blocked at this time, resend later
3	17	PATH closed.

7: REPLY 4
h

Format

REPLY INDEX PATH CODE

Use

This command is sent by both the Host and the FE to acknowledge receipt of every other type of command (including those on index 0, see Section 4) and/or
to unblock that particular direction of an opened path for the transmission of another command.

Note that the INDEX and PATH fields contain exactly the same as those of the command being replied to.

WORKING DOCUMENT

WORKING DOCUMENT

Parameters

a) REPLY is an 8-bit field with value 7.

b) INDEX is a 16-bit field specifying the index.

c) PATH is a 8-bit field specifying the path.

d) CODE is a 16-bit field indicating the success/failure of the command being REPLYed to, and the sender's readiness for more commands on the same path. It is divided into four subfields - STATUS, COMMAND, CATEGORY, and TYPE.

1) STATUS is 4 bits wide

bit 0 (right-most) - READY
bit 1 - NOT-READY
bit 2 - ACK
bit 3 - NAK

ACK=1 indicates that the sender (of the REPLY) has accepted the command (being REPLYed to). NAK=1 indicates that the sender has discarded the command (with the reason given by the settings of the other bits of the CODE field).

NOT-READY=1 indicates that the sender (of the REPLY) is willing to receive an END or INTERRUPT on this path. READY=1 indicates that MESSAGE commands will also be received.

Normally only one REPLY command will be sent for each other command. However MESSAGE, INTERRUPT, RESPONSE and invalid END commands can be replied to by a REPLY with ACK (or NAK)=1 and NOT-READY=1 and another REPLY, some time later, with READY=1. [6]

The ACK and NAK bits are mutually exclusive and should never both be on simultaneously, and similarly the READY and NOT-READY bits.

Note that the READY/NOT-READY bit settings are only relevant when a path is open.

WORKING DOCUMENT

WORKING DOCUMENT

2) COMMAND is 4 bits wide. It indicates the command for which this is a REPLY :

Value	Meaning
0	any of the following
1	BEGIN
2	LISTEN
3	RESPONSE
4	MESSAGE
5	INTERRUPT
6	END

The value 0 is defined for cases where a Host does not wish to incur any overhead required to fill in the non-zero value.

3) CATEGORY is 3 bits wide. It specifies the category of the error indicated by the ACK bit being off :

Value	Meaning
1	Command not recognized
2	Option not implemented
3	Invalid
4	Action failed.

Its value is relevant only when NAK=1.

4) TYPE is 5 bits wide. It specifies which error occurred. Its value is relevant only when NAK=1. The possible values and meanings for the various errors and their corresponding CATEGORY subfield values are given under the description of each command.

Sequencing 4i

Once communication between the Host and the FE has been established and each side is "Ready for Business" (see Section 4b) the Host may at any time send BEGIN or LISTEN commands for new indexes. The FE will acknowledge a BEGIN or

WORKING DOCUMENT

WORKING DOCUMENT

LISTEN with a REPLY and the index is then set-up providing that the REPLY indicates no errors. Other BEGIN or LISTEN commands for the new paths on the same index may be sent at any time after the index is set-up.

The FE will send a RESPONSE command for each path on completion of its attempts to fulfill the Host's instructions. If an attempt failed (indicated by the CODE field) then the path remains closed and another BEGIN or LISTEN for that path can be sent. If the attempt was successful, then MESSAGE or INTERRUPT commands can be sent after the Host has REPLYED to the RESPONSE.

An INTERRUPT or END command may be sent on any opened path after receiving a REPLY for the previous command on the same path in the same direction. A MESSAGE command may be sent if in addition the READY bit was on in the last REPLY command.

New paths on the same index may be opened at any time after the index has been set-up, or particular paths may be ENDed and then have new BEGINs or LISTENs for them issued. An index remains set-up, even if all its paths are closed, until an END command with PATH=0 is issued for the index.

Communication between the Host and the FE is terminated by an END with INDEX=0 and this will abort any remaining open paths and indexes.

Broadcasting 4j

Broadcasting is an optional feature of the protocol. If it has been enabled by the installation parameter, then the Host may send a MESSAGE or INTERRUPT

command on a particular index with PATH=0. This instructs the FE to send the data contained in the TEXT field of the MESSAGE command (or an interrupt) on every network connection corresponding to an open path of the specified index.

This feature will only occur on MESSAGEs from the Host to the FE (since no utilization of it in the other direction is envisaged).

A broadcast MESSAGE is replied to with one or two REPLYs in the same way as any other MESSAGE command. Flow control within the index is maintained as if broadcast MESSAGEs were sent on a separate path, i.e., flow control on other paths is not directly affected.

WORKING DOCUMENT

WORKING DOCUMENT

Note that for a broadcast MESSAGE command the FE will perform translation on the data for each path in accordance with the BEGIN or LISTEN which initiated that path. Thus, care must be taken when all paths of the particular index do not have the same format on the Host side specified in their TRANS-TYPE (see Section 6b).

Index 0 5

Introduction 5a

Index 0 provides a control link between the Host and the FE, and thus has no network connections directly associated with it. The commands on this index are used to establish and terminate the connection between Host and FE and to control other indexes.

Path 0 5b

Path 0 of Index 0 is used to pass global commands - i.e., those which do not refer to any particular index or path. The currently defined commands are :

MESSAGE INDEX=0 COUNT PAD TEXT

where TEXT = COMMAND [PARM1] [PARM2]
COMMAND is 8 bits long
PARM1 and PARM2 are 16 bits long

a) COMMAND=1 , PARM1=Hostid

This is the "Ready for Business" command which must be sent by both Host and FE to establish communication between them. Count gives the length of the TEXT field as usual. If COUNT=8, then just the COMMAND field is present. If COUNT=24, then both the COMMAND and Hostid are present.

The FE will never send a Hostid. The Host may send its Hostid in the event that the FE is connected to more than one IMP or if alternate routes to the network exist (e.g., via patch panels).

Until both sides have sent this command no other command is valid.

b) COMMAND=2 , PARM1=M , PARM2=N

WORKING DOCUMENT

WORKING DOCUMENT

This is the "CLOSING" command which is a purely information indicati
on
that the sender’s FEP module has been told that communication will b
e
terminated in M minutes for a period of N minutes (N=0 implies
unknown).

No action is required of the receiver, however he may be able to
distribute the information to his users.

c) COMMAND=3

This is the "CONTINUE" command which indicates that any previous
CLOSING command is now no longer true.

END INDEX=0 PATH+0 CODE

This command terminates the connection between the HOST and FE. All
other paths/indexes are automatically aborted and the FE will close
all network connections. The values of the CODE field are the same
as in the general END command.

Path 1 5c

Path 1 is reserved for commands specific to a particular path or index. No
commands are presently defined; they will be at a later date when more
experience has been gained on the need for them.

Path 2 5d

Path 2 of Index 0 is used for Operator-to-Operator communication between the
Host and the FE. It is an optional feature which is enabled by an installat
ion
parameter.

MESSAGE commands are formatted in the normal manner with the sender requesti
ng
that the TEXT field be displayed to the receiver’s system operator.

Scenarios
6

The following scenarios are included to provide the reader with a "feeling"
for
FEP in a varied set of applications. The examples selected relate to existi
ng
ARPANET protocols or other networking applications, and do not represent an
exhaustive list of capabilities.
6a

WORKING DOCUMENT

WORKING DOCUMENT

Fields which are variable or not relevant are not shown (for purposes of clarity) in the command strings in the following examples. 6b

Host Implementation of User TELNET 6c

```
BEGIN ndxa,PATH=1,host,SKT=1,,CONN-TYPE=duplex+ICP,NPATHS=1
```

The User TELNET process in the Host causes the BEGIN command to be issued. When a successful RESPONSE has been returned by the FE, a typical duplex TELNET connection will have been made to the Host specified in the BEGIN.

Host is Providing Server TELNET 6d

```
LISTEN ndxa,PATH-1,HOST=0,SKT=1,,CONN-TYPE=duplex+ICP,NPATHS=32
```

With this one command the Server TELNET process in the Host has conditioned the FE to LISTEN on Socket 1 (the well-advertised TELNET socket) and to establish as many as 32 duplex data paths. The FE, through the RESPONSE command, will report each connection as it occurs. Path 1 will represent the first such duplex connection, etc. The Host may then manage the data paths individually. Individual paths may be ended and placed back into a LISTENing state by the Host. If at any time an END command specifying this INDEX with a PATH of 0 were to be sent by the Host, all connections would be dissolved, and for all practical purposes, the Host is no longer willing to provide Server TELNET services.

Host is Providing Server FTP 6e

```
LISTEN ndxa,PATH=1,HOST=0,SKT=3,,CONN-TYPE=duplex+ICP,NPATH=1
```

As soon as a RESPONSE for this LISTEN comes from the FE, the Host Server FTP process should select a new INDEX and issue a new LISTEN for ndxb on socket 3.
The duplex connection which has been made is the control path for the file transfer. Based upon control information passed between server and user on ndxa (path 1) the server FTP will either:

```
BEGIN ndxa,PATH=2,(hostid etc. from response),NPATHS=1
```

OR

```
LISTEN ndxa,PATH=2,(hostid etc. from response),NPATHS=1
```

WORKING DOCUMENT

When a RESPONSE command has been received to the previous command, the data connection (PATH 2) will have been made and transfer of data may begin. The values for TRANS-TYPE and CONN-TYPE for the LISTEN or BEGIN will be derived from the exchange of information on the control path.

Host Is User FTP 6f

BEGIN NDXA,PATH=1,HOSTID,SKT-3,,CONN-TYPE-duplex+ICP,NPATH=1

when a RESPONSE to this command has been returned by the FE the control path will have been connected. The Host, after exchanging information on the control path, may then proceed by issuing a BEGIN or LISTEN as in the Server

FTP example.

Teleconferencing 6g

An INDEX with n PATHs permits up to n otherwise disassociated conversations to be affiliated. Each path can be manipulated individually, or all paths as a group. With the broadcast option -- a MESSAGE command specifying INDEX but not specifying PATH will be broadcast to all open paths on that index. Thus each host may direct its messages to any or all parties.

A "conference" may be initiated by any host who issues a LISTEN with multiple duplex paths on an agreed upon (but not necessarily well advertised) socket.

When some foreign host connects, an ordinary TELNET connection exists. If, however, a third or forth or more parties connect, the hosts already engaged in the conversation may elect to inform the late comers of the members already involved. Each host may then elect to connect to as many other hosts as he desires. (The parties could agree as to who would BEGIN and who would LISTEN).
Following this scheme [it is not a protocol] all parties participate equally, there is no moderator. Each host decides to whom he will speak. Using the initial LISTEN, a variation on this would permit the LISTENer to be moderator and require that he relay messages to other parties, as desired.

In summary, the data path mechanism permits a group of users to select an agreed upon socket, appoint a "moderator," and at a prescribed time engage in a conference without benefit of special protocol implementations in the FE or in any of the hosts (except possibly the moderator).

WORKING DOCUMENT

WORKING DOCUMENT

Example of the Use of Simplex Connections
6h

The Simplex connection types permits a host to LISTEN on a group of simplex sockets of a particular gender. If the host supported a group of line printers, for example, the Line Printer Applications Program could perform a

LISTEN on a socket advertised to be his "Printing Socket," specifying as many receive paths as he had printers. Foreign hosts could then connect (via ICP) to his print socket. They would be given an appropriate working socket value and then connect to an available printer. In this way up to n foreign hosts may be connected to his n printers at all times. All that any needs to know to avail themselves of printing services is the server host's print socket. [1]

Host Implementation 7

Concepts 7a

The Front-End Protocol permits a Host to make use of the network through existing low-level protocols, without requiring that it be cognizant of the implementation details of those protocols.

Implementation of FEP in the Host is in terms of the function it is performing or the service it is providing. Information regarding sockets is available to the sophisticated user, but can be ignored if not relevant to the problem at hand.

The Host should provide the equivalent of a BEGIN, LISTEN, MESSAGE, INTERRUPT, and END command. In other words, the human user or applications level process has at its disposal the full power of FEP.

The FEP module in the Host serves as a control mechanism to multiplex/demultiplex traffic between itself and the FE. In appearance and function it is much like any multi-line interface driver. It handles REPLYS, reports errors, etc. The FEP module must also assume the responsibility for assignment of indexes. This could easily be implemented as a "GETINDEX" subroutine which would allow a user to ask for an index to be assigned to him. The user could then proceed to do BEGINs, LISTENs, etc. on that index.

A server process makes itself available to the network at large by issuing an appropriate LISTEN. The Host FEP module would not have to be aware of what servers were implemented or in operation. The server process, when it was

activated, could do a "GETINDEX," followed by a LISTEN on its well-advertise
d
socket, and then proceed from there. The Host FEP module simply associates
indexes to processes and passes the incoming traffic to the appropriate proc
ess
for analysis and response. It maintains flow between itself and the FE thro
ugh
the generation and receipt of REPLYs.

The type of data structures, or format of information employed in the implementation of the FEP commands in the Host is, of course, up to the implementor. BEGIN could be a macro call, with the various information passed as parameters to the Host FEP module -- which would then arrange it into a command for delivery to the Front-End processor. The important consideration is not how the commands are implemented, but simply that their

function is provided. It might be desirable, for instance, to implement the Host such that the front-end processor looks like a special I/O device. In this case, it may be appropriate to implement a form of OPEN [for BEGIN or LISTEN], a GET or PUT [for MESSAGE], CLOSE [for END], etc...

Regardless of the implementation details, it appears that, while it is the responsibility of one control module to assign and manage INDEXes, data path
s
are entirely the responsibility of the process which "owns" the INDEX.

Installation Parameters 7b

To package the software for the FE for a given Host, that Host supplies a number of parameters defining what FE capabilities it requires. These parameters are input to a system-generation procedure to produce the particu
lar
FE code.

The parameters are:

Byte Size

This gives the size in bits, into a multiple of which each and every field of a command string will be right-justified (i.e., the information bits come last, preceded by as many padding bits as are needed to complete the least integral number of bytes).

Its value will normally be 8 but other values will be accommodated as necessary.

WORKING DOCUMENT

Command String Padding

This gives the size in bits of the width of the hardware interface between the Host and the FE, such that every command string transmitted in either direction will have padding appended to complete the least multiple of this width.

In the typical implementation, this parameter will be 0 and any padding required will be appended/discarded by the line protocol underlying FEP.

Pad Field Length

This gives the size in bits of the PAD field in the MESSAGE command. This enable a Host to have the TEXT field start on a convenient boundary.

Its value can be anywhere in the range 0 to 64.

Maximum of MESSAGE

This gives the maximum length of a MESSAGE command string.

Because buffer allocation in the FE is based on this parameter, its value should be chosen with care.

Maximum number of Indexes

This gives the maximum number of indexes which may be set-up at any one time.

Maximum Number of Paths

This gives the maximum number of paths within one index which may be open at any one time.

Translation Types

This gives the set of required values and meanings of the TRANS-TYPE field of the BEGIN/LISTEN commands. The TRANS-TYPE field is divided into two 8-bit subfields; the first giving the format of data on the

network side; the second giving the format of data on the Host side. The FE is required to translate between these formats all data contained in the TEXT field of MESSAGE commands.

This parameter specifies the required formats and their values in th
e
8-bit subfields. The value 0 is reserved to mean "bit-string" and when it appears as either (or both) of the subfields it implies no translation is to be done.

Broadcast Option

This specifies whether the Host wants to be able to use the Broadcas
t
feature (see Section 3j).

Operator-to-Operator Communication Option

This specifies whether the Host wants the ability to send messages to the FE operator or to have the Host’s operator receive messages from the FE.

Other options may be included in the protocol at some later date and these w
ill
be available through installation parameters similar to the Broadcast option
.

Note that all of these parameters affect the size and complexity of the FE code. Thus it is important that their values be chosen carefully so as to maximize FE efficiency while minimizing Host implementation effort.

For descriptions of individual Host implementations and a list of the option
s
available so far, see Appendix D.

FE Implementation 8

FEP is device independent. For the present however, an initial implementati
on
will be accomplished using the DEC PDP/11 computer as the FE device and the front-end software is to be based upon an extended version of the original E
LF
system developed at SCRL.

For more detailed information, see Appendix C.

by : 9

G. W. Bailey (BAILEY@OFFICE-1)
K. McCloghrie (MCCLOGHRIE@OFFICE-1)

WORKING DOCUMENT

APPENDIX A 10

References

[1] ICP is used in this document in a less strict manner than specified in NIC 7101, in that it is not necessarily two simplex connections that are set up as the result of the exchange of the socket number on the initial connection.

[2] An example of connections needing to be affiliated, is in the implementation of FTP, where the control connection and the data connection have a defined relationship in their socket assignments.

[3] Note that a range of socket numbers is reserved for use by an index when it is set-up (cf. AEN).

However, socket numbers for the paths of an index are not necessaril
y
contiguous. For instance, when the next path opened after a SEND path is another SEND path, or when a path other than the first of an index is opened with ICP specified. Nevertheless, if a protocol requires contiguous sockets, then the opening of the paths in a logi
cal
manner will provide the contiguity.

[4] One possible translation will be from a Network Virtual Terminal on the network side to a local terminal type on the Host side.

[5] The FE will directly equate the INTERRUPT command with the Host-Host protocol INR/INS commands.

[6] Note that the READY indication in a REPLY is, in the general case, not directly related to a network RFNM; unless it is heavily loaded,

the FE will be buffering possibly more than one message (in either direction) until flow control mechanism allow the messages to be sen
t
on.

However, it is possible that a particular Host might wish to have knowledge of receipt of a previous message before transmitting the next. In this case, the FEP implementation could be set up to only indicate READY after receiving the RFNM and possibly only send RFNMs after receiving a REPLY containing an ACK.

WORKING DOCUMENT

WORKING DOCUMENT

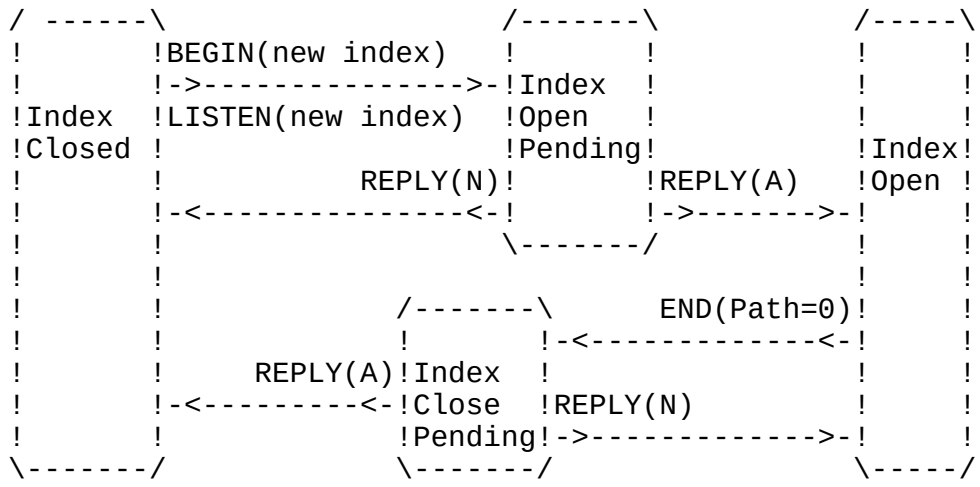
APPENDIX B

State Diagrams

In the state diagrams below the following notation is used:

```
REPLY(A)    - REPLY with ACK=1, READY/NOT-READY irrelevant
REPLY(N)    - REPLY with NAK=1, READY/NOT-READY irrelevant
REPLY(R)    - REPLY with ACK=0, NAK=0, READY=1
REPLY(A+R)  - REPLY with ACK=1, READY=1
REPLY(N+R)  - REPLY with NAK=1, READY=1
REPLY(A+NR) - REPLY with ACK=1, NOT-READY=1
REPLY(N+NR) - REPLY with NAK=1, NOT-READY=1
```

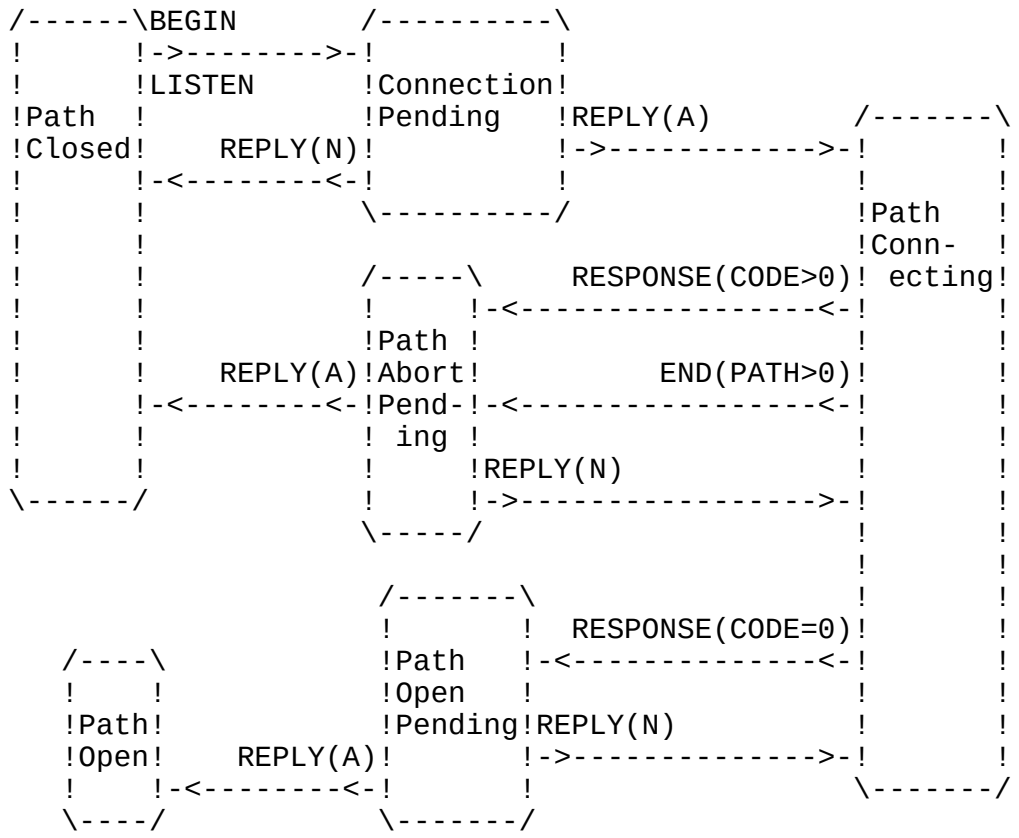
State Diagram for INDEX



WORKING DOCUMENT

APPENDIX B (continued)

State Diagram for Whole Path

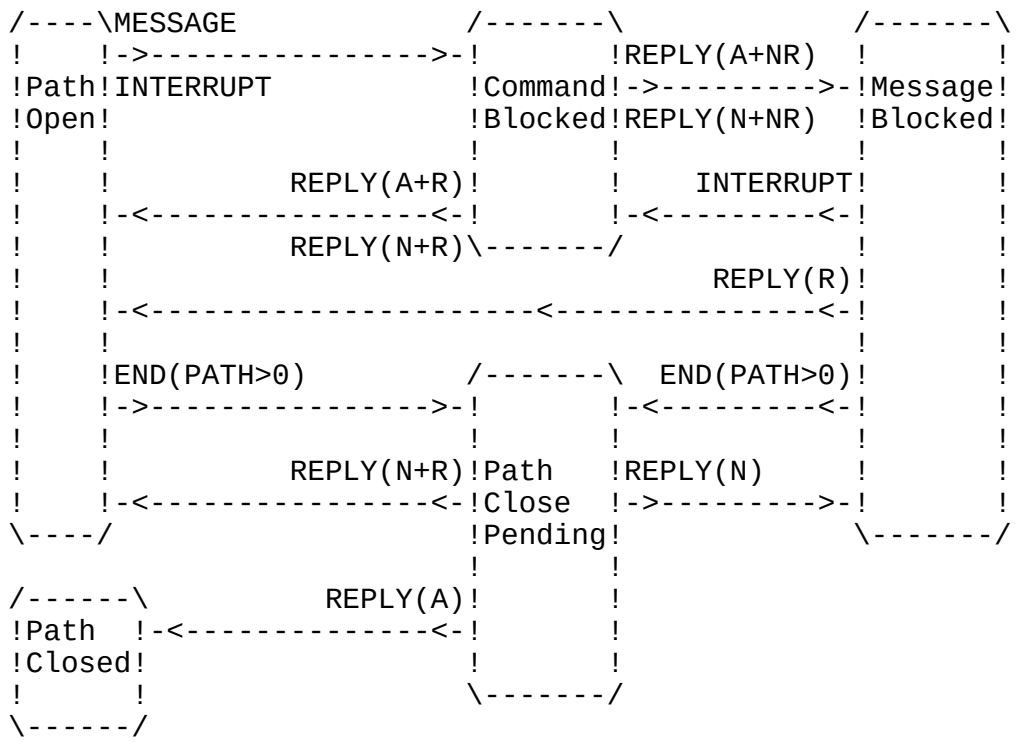


WORKING DOCUMENT

WORKING DOCUMENT

APPENDIX B (continued)

State Diagram for Each Direction of Path



WORKING DOCUMENT

WORKING DOCUMENT

APPENDIX C

Front-End Implementation

Introduction

A Network Access System (NAS), developed for a DEC PDP/11 computer, supports the current Imp-Host, Host-Host and ICP protocols. The implementation of these protocols facilitate process-process communications across the network and multi-user TELNET access to foreign hosts. This NAS provides the FE environment in which FEP is implemented.

The NAS system is comprised of a Kernel or executive section and a Network Control Program (NCP) plus a collection of modules to support device interfaces, handle terminals, and implement applications, as appropriate. The software is modular and extensible.

The KERNEL

The Kernel of the system consists of a set of functional modules which perform the task of resource management in a multiprocessing environment. This allows processes to be created, vie for processor service according to priority, intercommunicate, and be terminated. System primitives exist for various tasks such as process creation and synchronization, storage allocation, and sharing of the interval timer.

The term process used here describes an autonomous sequence of states brought about by the PDP-11 processor; a process' state is characterized by the set of processor registers, a stock, and process-owned storage areas. Process share storage areas which are accessed only (eq. pure code). Processes also share storage areas which may be updated (eq. control tables). In this case an allocation mechanism is utilized to prevent simultaneous ownership of an updatable storage area. The storage area is thus viewed as a sequentially sharable resource which is allocated by the process, modified, and then released.

Processes are given control of the processor by a single procedure called the Dispatcher. Processes are said to be in a ready state or in a waiting state. When a process blocks itself, control is given to the highest priority ready process.

WORKING DOCUMENT

Each process has an associated input message queue. This queue is the vehicle for interprocess communication. A process is blocked (put into a wait state) when its input message queue becomes empty (voluntary wait), or when an interrupt occurs (involuntary wait) because a higher priority process is to receive control of the processor. A process may voluntarily block itself waiting for any signal, or it may block itself for a specific event to be posted to its input message queue.

The Network Control Program

The NCP provides "third level" protocol functions to local processes. It contains a process which decodes and passes messages which have been received from the IMP and placed on the IMP-Host queue. This process interacts with other processes which call the NCP to establish connections or to transmit data. Thus the NCP is essentially divided into two parts:

1) a process which handles incoming messages from the network, interprets IMP-Host and Host-Host control messages, and forwards regular messages on established connections; and

2) a set of primitives which allow local processes to establish connections to other processes across the network, and to perform requests for data to be transferred on these connections.

There are two primary data structures used by the NCP to monitor the status of network connections. The first is called the Host Table, and describes that which is peculiar to each given host; the second is referred to as a Connection Table and contains all information on the state of a local NCP socket (connection). Connection Tables may be created either through external requests (e.q., an RFC is received from a remote host) or through internal requests (e.g., a local process performs a LISTEN).

Flow control is that portion of the NCP which governs the flow of data on connections. There are two procedures which perform this task; one which handles receive connections and one which handles send connections. These procedures receive control when an event has occurred which may now make it possible to transfer data on a connection.

Both send and receive flow control procedures have the responsibility of moving data between local process buffers and messages being received or transmitted over the network. In addition, they handle the formatting and unpacking of

messages received. Local processes are unaware that data is being transmitted as discrete messages.

The NCP watchdog process monitors the state of network connections, checking for error conditions and performing garbage collection tasks. It receives control at periodic intervals and scans the list of known hosts, looking for existing connections. For each host to which an input or output connection exists, the Watchdog causes a Host-Host NOP message to be sent. Thus if a remote Host crashes while data is being awaited, local processes are informed of the error condition. The NCP takes notice of the remote crash when it receives a IMP--Host type 7 control message (Destination Host Dead). It then automatically closes all connections to that Host, and notifies using processes of that fact.

A second function of the NCP Watchdog is to check for connections hung because of an outstanding RFNM. If a RFNM is not received for a specified interval, the message is discarded, and the associated connection closed.

The FEP Handler

The Front-End Protocol is implemented as a collection of related, but specialized processes which manage network connections on the one side, and manage FEP paths and indexes on the other. Some FEP processes are NCP users. They cause network connections to be made, rule on incoming RFCs, and both accept and generate network data. Other FEP processes support the Host. These processes parse incoming commands, create indexes and paths, control the generation of replies and generally manage the paths. Certain FEP processes control specialized tasks such as translation of data, servicing of LISTEN commands and generation of RESPONSE commands.

Two data structures provide control information for FEP activities. An Index Table exists for each active index. Each Index Table associates one or more Path Table entries. Information in the Path Table reflects the state of the path, the translation type specified for data on this path, and necessary information to associate the path to any appropriate NCP Connection Tables. The Path Table is the common interface for all of the FEP modules. Most FEP processes are activated to service some event which is usually associated to a path. The action of the process will likely be dictated by the state of the path as indicated by the Path Table entry, and may result in altering the state of the path or the activation of one or more other FEP processes.

WORKING DOCUMENT

Two message queues provide Host input and output to the FEP modules. A line protocol mechanism services these queues. Commands from the Host are placed on the FEP Input queue by the line protocol process and the FEP Host Input process is signaled. When an FEP Host Output module places a Command for the Host on the host Output queue it signals the line protocol process.

The FEP implementation is basically Host independent down to the level of th
e
Host Input and Host Output queues.

The Line Protocol Mechanism

The device interface and the line protocol between the FE and the Host are installation dependent. Because of this dependency, only a general discussi
on
of the Line Protocol Mechanism is possible in this context. Detailed descriptions of the specific line protocols are included in the section for each Host.

The communications discipline and physical device characteristics may vary considerably from host to host. All FEP line protocols, however, will show certain common characteristics. The interface between the FEP handler and t
he
Line Protocol Mechanism will always be Host Input and Host Output queues. A
ll
line protocol mechanisms will be expected to guarantee the integrity of the data. This implies some form of flow control, error detection/correction an
d
retransmission capability, as well as normal transmit/receive responsibiliti
es.
The Line Protocol Mechanism will be expected to report failure after unsuccessfully attempting to perform an I/O operation. The number of retrie
s
etc. before reporting failure is an installation parameter. The FEP Handler works only in terms of FEP commands. The line protocol may provide for bloc
k
transfers where each physical block is comprised of one or more FEP commands
.
If such is the case, it is encumbent upon the Line Protocol Mechanism to deblock the incoming Host commands before placing them in the Host Input que
ue.

The Line Protocol Mechanism will, in the general case, not manage any buffer
s.
After successfully transmitting a command to the Host it is responsible for reporting the I/O complete, but the buffer space is freed or reused only by the FEP process which "owns" that space. The FEP Handler might use buffer assignment to control the rate of incoming traffic. When the FEP Host Input

queue is ready to accept an additional command, it would acquire a buffer an
d
signal the Line Protocol Mechanism, passing it a pointer to a buffer. This

is effectively a "read" request. When the line protocol handler has filled the buffer, it adds it to the Host Input queue and signals I/O complete to the appropriate FEP process.

If the nature of the physical connection is such that the FE must accept unsolicited input, it may be necessary for the Line Protocol Mechanism to have its own buffer pool, in addition. If this is the case, it must be entirely managed by the line handler and transparent to the FEP Handler.

Data Translations

The TRANS-TYPE provisions in FeP may be employed for at least two general services. First, it can be used for normal character set substitutions. Th
is
is where, in the general case, there is a one-to-one relationship between th
e
two character sets.

The second service addresses the problem of data transformation. In this ca
se,
there need not be a one-to-one relationship between incoming data and outgoi
ng
data.

The translation mechanism uses a token (e.g., a character) from the incoming data stream to index into a translation table. The result may be one of the following:

a) do nothing, drop the character
b) output the character unchanged
c) substitute input character by output character
d) substitute input character by output string
e) activate a procedure indicated by the table
f) change the translation
g) test the translation mode and do any of the above depending on the result.

For each translation/transformation required by the Host a translation table

must be defined. For simplicity and clarity the TRANS-TYPE field in the FEP commands allows the user to specify Host side and Network side as independen
t
entities. In actual execution the Host/Network pair addresses a translation table which must have been previously defined. Note that for a duplex path two translation tables are necessary (A->B is not the same as A<-B).

A collection of "standard" character sets will be addressed initially (EBCDI
C,
ASC117, ASCII8, BCD, etc.) and at least NVT. As new requirements are define
d
these will be added to a library which will then be available to subsequent users.

WORKING DOCUMENT

APPENDIX D

Host Implementations

To be written at a later date.