Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document Number 220
CCTC-WAD Document Number 7501

Networking Research in Front Ending and Intelligent Terminals

H6000 Software Specifications

January 10, 1977

CAC Document Number 220
CCTC-WAD Document Number 7501

Networking Research in Front Ending and Intelligent Terminals

H6000 SOFTWARE SPECIFICATIONS

by
Peter A. Alsberg
Geneva G. Belford
John D. Day
Gary R. Grossman
Steven F. Holmgren

Prepared for the
Command and Control Technical Center
WWMCCS ADP Directorate
Defense Communications Agency
Washington, D.C. 20305

under contract
DCA100-76-C-0088

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

January 10, 1977

Approved for Release: Peter A. Alsberg
Peter A. Alsberg, Principal Investigator

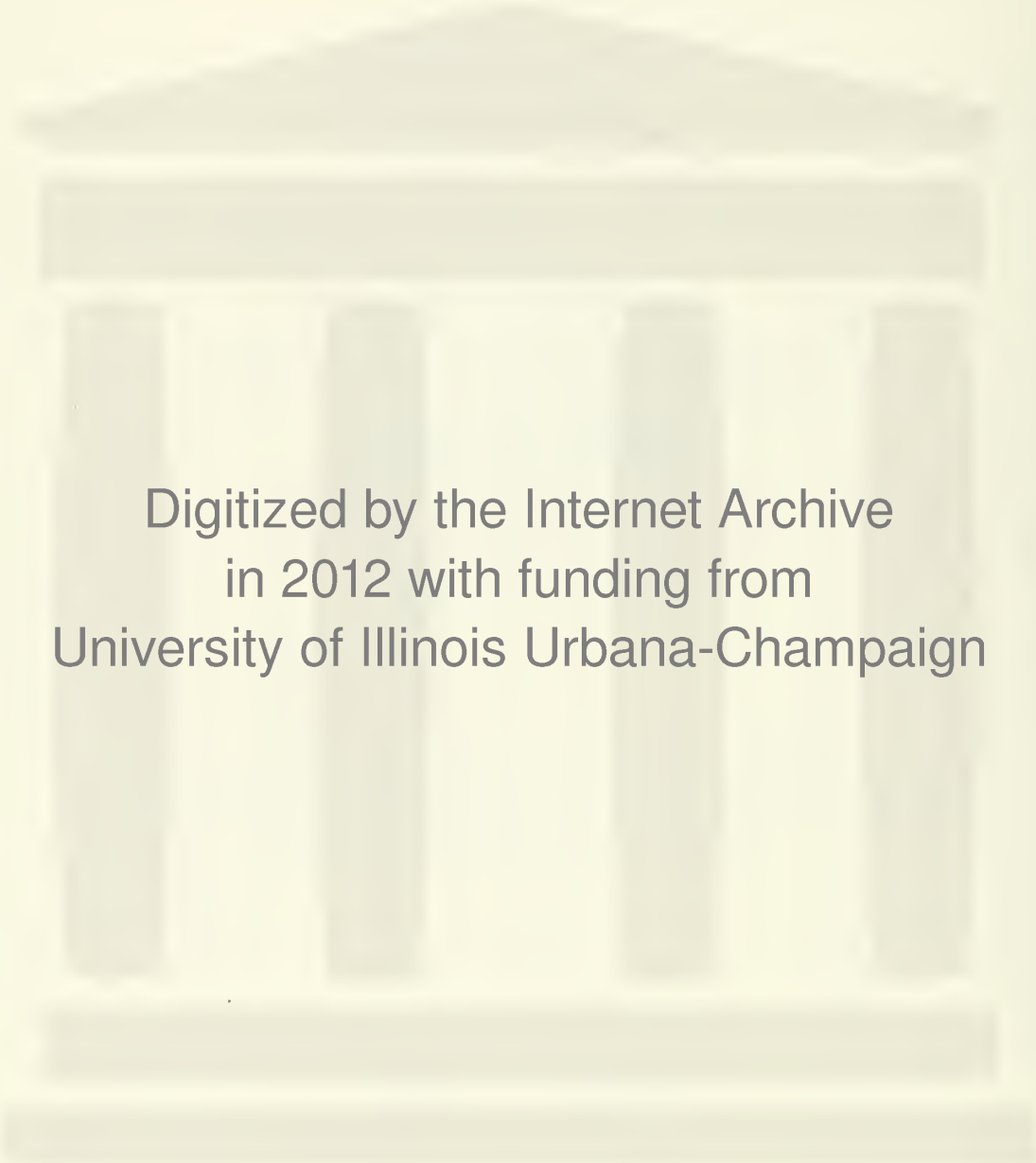

TABLE OF CONTENTS

SUMMARY

Background

Under contract DCA100-76-C-0088, the Center for Advanced Computation (CAC) of the University of Illinois at Urbana-Champaign is investigating the capabilities of network front ends. As a part of that contract, an experimental network front end is being developed to interface a WWMCCS H6000 to the ARPA Network and to conduct experiments with a Host-to-Front-End Protocol. The experimental network front end is being developed on a DEC PDP-11/70. The CAC has previously enhanced the UNIX operating system for PDP-11's to act as a mini-host on the ARPA Network. The network software developed by the CAC will be further enhanced under this contract to support the proposed ARPA Network Host-to-Front-End Protocol (HFP).

For complete specifications the reader will need to consult the set of supporting documents currently being prepared by the CAC. These documents are

1. Host-to-Front-End Protocol (CAC Document 219),

2. ARPA Host-Host Process-to-Service Protocol (CAC Technical Memorandum 80),

3. Program Access Process-to-Service Protocol (CAC Technical Memorandum 81), and

4. Server Virtual Terminal Process-to-Service Protocol (CAC Technical Memorandum 82).

The present document is designed to provide an overview of requirements and to fill in some necessary details not contained in the other documents. There are three major sections, which contain:

1. an overview of required software,

2. a description of the computer link between the H6000 and the PDP 11/70, and

3. specifications for the H6000 software required to support Telnet (primarily the User Telnet facility).

Software Requirements

Introduction

At present, the storage, maintenance, and processing requirements of host-resident network software represent a significant burden on WWMCCS hosts. Offloading a major portion of this network software to a front end should reduce the extent and complexity of host-resident software. As a result, host performance should improve considerably. Proper design of front-end and interface software should also yield improved security.

The network software can be thought of as a set of services provided to host processes or users. These services allow the network and the various hosts connected to it to be conveniently used. Offloading shifts the major burden of providing these services from the local host to the front end. A certain amount of software, however, must necessarily remain in the host, in order for a user or process to gain access to the front end and its services. In this section we summarize the host software requirements. Details are given either in the later sections of this document or in one or more of a set of supporting documents currently being prepared by the CAC.

Residual Service Software

The major services to be offloaded to the front end are:

1. the Network Control Program (NCP), which controls access to the network and to remote hosts on the network, and

2. Telnet, which provides an interface between terminals (which may be of widely differing types) and interactive processes on remote hosts.

The NCP is a system software module that implements the ARPA Network Host-Host and Initial Connection Protocols. Most of the NCP functions will be moved to the front end. These functions include flow control on connections, execution of connection protocol, maintenance of socket tables, sequencing of protocol states, and other complex support requirements. The current H6000 NCP should be replaced by a simple software module that maps the existing NCP software interface into HFP Messages whose ultimate destination is the NCP module in the front end. (The format of these Messages is specified in CAC Technical Memorandum 80, which defines the ARPA Network Host-Host Process-to-Service Protocol.) No modifications will then be required for programs that currently communicate with the host-resident NCP.

The Telnet service is usually implemented as two separate services: a User Telnet facility and a Server Telnet facility. User Telnet accepts input from terminals and initiates connections to ARPA Network hosts. Server Telnet accepts those connections. Normally, a user of a Telnet facility invokes a User Telnet "program" at his local ARPA Network host and is connected to a server Telnet "program" at a remote ARPA Network host. The result is that the user´s terminal appears to be connected to the remote host as if it were a local terminal at the remote host.

User Telnet can be almost completely offloaded to the front end. Only a simple relay process need be implemented in the H6000. This relay process is specified in fairly complete detail in a later section of this document.

The situation with regard to Server Telnet is more complex. There may be advantages to leaving more than the minimal amount of software in the host. Specification of the host software required for Server Telnet must await the completion of the CAC Technical Memorandum 82, which defines the Server Virtual Terminal Process-to-Service Protocol.

HFP Software

In order for the host to be able to send messages requesting services to the front end, and for the front end to be able to respond, a communications protocol must be specified. This protocol is the Host-to-Front-End Protocol (HFP), which is defined in CAC Document 219. By means of this protocol, logical channels are set up between host and front end, and messages are transmitted on these channels. Provisions are made for flow control and for out-of-sequence signaling.

The host must contain a software module, the Channel Protocol Module (CPM), which manages the logical channels and serves as a multiplexor. Messages from front end services to host processes are obtained from the lower-level communications link and transmitted to the correct process. In the other direction, the host CPM accepts messages from host processes and properly formats them for transmission on the lower-level communications

link to the front end. A CPM in the front end manages the other end of each logical channel.

In addition to the CPM itself, software must exist in the host to provide an interface between host processes and the host CPM. A suggested set of system primitives to implement such an interface is given in CAC Document 219.

Lower-level Communications Link

The host CPM must also interface with the low-level communications link to the front end. We have therefore provided a description of this link in the section below entitled "H6000 to PDP-11/70 COMPUTER LINK."

In brief, both the H6000 and the PDP-11/70 will be provided with sophisticated interfacing devices which communicate by means of tested ARPANET IMP-to-Host data transmission techniques. For this experimental project, we are attempting to use proven, off-the-shelf technology as much as possible. On the H6000 side, an Asynchronous Bit Serial Interface (ABSI) will be used. The ABSI is a Honeywell standard product modified by MIT to connect an H6180 to the ARPA Network. On one side, the MIT ABSI connects to an ARPA Network IMP. On the other side it connects to an H6000 IOM (I/O Multiplexor). On the PDP-11 side, an interface similar to that used to connect a PDP-11 to an ARPA Network IMP will be used. The IMP side of the PDP-11 interface and the IMP side of the ABSI can then be connected to enable PDP-11 to H6000 communications.

H6000 TO PDP-11/70 COMPUTER LINK

Introduction

This section describes a medium speed (800 Kbps) computer-to-computer link to be used to connect a Honeywell H6000 to a PDP-11/70. A description of the link hardware architecture is presented. Programming considerations for both ends of the link are described. A final subsection outlines interface conventions specific to this research project.

The link has two components: an H6000 Asynchronous Bit Serial Interface (ABSI) and a PDP-11 interface.

MIT's Electronic Systems Laboratory has modified an ABSI interface to connect it to an ARPA Network IMP [1]. Devices designed to interface a PDP-11 (as a host) to an ARPA Network IMP are available from a number of vendors. The link will be composed of an MIT-modified ABSI interface connected to a PDP-11 host-to-IMP interface.

Data will be passed between the interfaces using ARPA Network IMP-to-host data transmission techniques. These techniques include strategies for serially transferring data, for transmitting signals to indicate machine status, and for handling some error situations.

H6000 Asynchronous Bit Serial Interface (ABSI)

General Description

The ABSI connects a peripheral device to two IOM Common Peripheral Interchange (CPI) channels [Honeywell document number 43A130524 Rev H1]. Two CPI channels are required for full duplex operation: one for read operations from the front end and one for write operations to the front end. The ABSI has a maximum transfer rate of 800,000 bits/sec. The MIT ABSI attempts to meet CPI specifications. Known exceptions are summarized in reference 3, pages 11-13.

Operation

The MIT ABSI accepts six commands:

1. REQUEST STATUS
2. RESET STATUS
3. READ (from front end)
4. WRITE (to front end)
5. SET HOST DOWN
6. SET HOST UP

Each command returns a major status and substatus:

1. READY
   - COMPLETE MESSAGE
   - FRONT END DOWN
   - HOST DOWN
   - INCOMPLETE MESSAGE
2. DATA ALERT
   - PARITY ERROR
   - FRONT END DOWN
   - HOST DOWN
   - INCOMPLETE MESSAGE
3. COMMAND REJECT
   - INVALID OPERATION CODE
   - PARITY ERROR IN COMMAND SEQUENCE
   - FRONT END DOWN
   - HOST DOWN
   - INCOMPLETE MESSAGE
4. BUSY

The command and status codes are listed in reference 3. Much of the discussion that follows is also taken from reference 3.

REQUEST STATUS. This command returns the same status as did the previous command on the same channel. Previous COMMAND REJECT status, however, will be ignored, so that if COMMAND REJECT status is received in reply to REQUEST STATUS, it indicates that there was an error in the transmission of the REQUEST STATUS itself. Note that the status remembered from the previous command may indicate the substatus FRONT END DOWN even if the front end has since come up. To correctly obtain the current front end up/down status, RESET STATUS should be used. A similar situation exists for the substatus HOST DOWN. Note also that the status returned by this command may be undefined if it is the first command given after initialization of the ABSI.

RESET STATUS. This command is identical to REQUEST STATUS, except that any settable status condition from the previous command will either be reset or reinterrogated after the status is returned. Resettable status is defined as:

1. any DATA ALERT condition,
2. INCOMPLETE MESSAGE substatus,
3. FRONT END DOWN substatus if the front end is no longer down, or
4. HOST DOWN substatus if a SET HOST UP command has subsequently been given.

READ (from front end). This command will start reading a message from the front end. It will terminate under any of the following conditions.

1. The LAST FRONT END BIT line was raised by the front end to indicate transmission of the last data bit.

2. The host up/down relay is changed to the down state by either of the following:

    a. manual HOST DOWN pushbutton on the interface, or
    b. the SET HOST DOWN command on the write channel.

3. The front end goes down (according to its up/down relay).

4. The CPI Channel decides to terminate the command. Note that termination will not actually occur until just after a complete character (6 bits) has been transmitted to the channel. Possible reasons for termination by the channel are:

    a. The storage buffer indicated in the H6000 DATA DCW has been filled and there are no additional DATA DCWs.

    b. The channel is masked by the software (see IOM specification, Honeywell document number 44A177715).

    c. Some other condition occurs which causes masking, such as a second "connect" to the channel before the first DCW list has completed.

5. An "initialize" signal has been received (page 13). Note that all transmissions are aborted

immediately and no status is stored. During initialization, the "CHN POX" lines to both CPI Channels are opened. This will cause each channel to behave as though the ABSI had been disconnected. After the initialize signal is no longer present, the channel will be ready to accept a new command regardless of its previous state. The initialize signal also forces the host up/down relay to open (i.e., indicate HOST DOWN).

In normal operation, only cases 1 (COMPLETE MESSAGE, no errors) and 4a (INCOMPLETE MESSAGE, no errors) will occur. If the latter occurs, the termination status will be READY with substatus INCOMPLETE MESSAGE. The complete message may be constructed by appending the data read by the next READ command to the end of the data read by the present READ command. This permits a message to occupy two or more physical buffers. The ABSI does not raise the READY FOR NEXT FRONT END BIT line until a READ command is issued. Thus, there is no way to determine that the front end has initiated the transmission of a message unless a READ command is issued. It is expected that the software will normally keep a READ command pending whenever the system is in operation. The expected method for aborting such a read command is to issue a SET HOST DOWN command on the WRITE channel. As mentioned above, pushing the HOST DOWN pushbutton or initializing the ABSI will also abort a READ command.

WRITE (to front end). This command initiates the

transmission of a message to the front end. Normally, termination will occur only on an end-of-message condition. Note, however, that all of the conditions listed under READ (from front end) for aborting the READ command also apply except 1 and 2b. The LAST HOST BIT signal will be raised with the last bit transmitted only if termination is caused by the channel rather than by the ABSI. Also note that termination under conditions 4b and 4c (the channel becomes masked) will take place only after the next character (6 bits) has been sent to the front end.

SET HOST DOWN. This command will open the relay closed by the SET HOST UP command. If a READ operation is taking place, it will be aborted when the relay opens. It is possible that the SET HOST DOWN command will not take effect if followed too closely by a SET HOST UP command (i.e., the relay will not have time to open completely). Whenever the relay is open (or opening), READ and WRITE commands will be rejected with the status COMMAND REJECT and the substatus HOST DOWN. The SET HOST DOWN command may be simulated at any time with the manual HOST DOWN pushbutton.

SET HOST UP. This command causes the HOST MASTER READY line to be connected to the HOST READY TEST line by closure of a relay. The software must make sure that no READ or WRITE command is issued to either channel until the relay has had a chance to settle. The delay time is not specified here, since relay characteristics may vary. The relay in the prototype ABSI takes about 500 usec, but the programmer should probably allow 100 msec to be safe. This command may be simulated at any time by pushing

the manual HOST UP button.

Initialization. Initialization of the ABSI resets it to a state in which it is ready to receive commands from both channels. Any operations in progress will be aborted without returning status. An initialize signal can be generated by any of the following.

1. The manual initialize button is pushed.

2. Either IOM channel opens a relay ("POX" line) which is used by the peripheral to check that the IOM channel is connected and has power applied. Note that this relay is opened by the "SYSTEM INITIALIZE" button on the IOM bootload console. The same effect is obtained if either of the channel cables is disconnected.

Special Interrupts. When the front end comes up (closes its FRONT END READY relay) a "special interrupt" will be signalled on the READ channel. Note that bounce in the relay contacts may cause several interrupts to be sent to the software; the software must be prepared to handle this. The software should delay at least 100 msec after each interrupt and not consider the front end up until this interval has passed.

Status. The major status and substatus values are defined in Section 5 of Reference 3 with the following addendum.

The PARITY ERROR substatus of the DATA ALERT major status can occur only on the WRITE channel; the IOM will indicate a READ channel parity error in the channel status rather than the peripheral status field, since the error is detected by the channel rather than the ABSI. PARITY ERROR substatus indicates that a

character has been received from the IOM channel with bad parity. The WRITE command is aborted and the LAST HOST BIT is not sent to the front end. The software should indicate to the front end that this message should be ignored by immediately WRITING a second message whose length is greater than the maximum legal message length. Since the previous WRITE did not raise the LAST HOST BIT, this second message will be appended to the first, causing the front end to discard it. The original message may then be transmitted as though nothing happened.

It should be noted that PARITY ERROR substatus will usually be returned only if a logic element fails. Parity errors should occur infrequently.

ABSI Physical Characteristics

The ABSI will be a self-contained rack-mountable unit with its own power supplies and cooling fans. There will be two AMP 201622-1 male receptacles which mate with standard Honeywell IOM cables from the CPI channels. The rest of the physical characteristics are vendor dependent.

PDP-11/70 Interface

General Description

The PDP-11 side of the computer-to-computer link has receive and transmit components. These components will provide full duplex, serial, Direct Memory Access (DMA) communications with the ABSI.

PDP-11 DMA operation consists of interface data transfer to or from core memory. This transfer is accomplished by preempting the PDP-11 bus for short periods of time. While the bus is preempted, communication with memory takes place. Processor intervention is not required while a transfer is in progress.

Processor interrupts are generated at the completion of a transfer and under certain error conditions.

Each interface component will have four addressable registers: a Control Status Register, a Data Buffer Register, a Memory Address Register, and a Byte Count Register.

Operation

Normal operation of each of the components (receive and transmit) will consist of four steps.

1. The Memory Address Register is loaded with a buffer address.
2. The Byte Count Register is loaded with the size of the buffer.
3. The Control Status Register is loaded with a value which enables interrupts and initializes the transfer.

4. An interrupt is generated when the transfer is complete or when an error occurs. The Control Status Register may be examined to determine results.

Certain special interrupts are defined.

1. NON-EXISTENT MEMORY
   A non-existent core memory address was put into the Memory Address Register.

2. INPUT BUFFER FULL
   The count field of the input component is zero. Software intervention is required.

3. HOST READY TRANSITION
   The HOST READY control signal has been turned off by the other machine. This signal indicates that the host is down.

The LAST FRONT END BIT is used to notify the H6000 that the current message is complete. This notification may be inhibited by setting the DISABLE END MESSAGE bit in the Transmit Control Status Register. This permits the transfer of a message which occupies two or more buffers. To indicate message completion, the bit should be reset on the last transfer.

Physical Characteristics

The PDP-11 interface will be a self-contained rack-mountable unit. It will have its own power supplies and cooling fans. Suitable connections will be available for standard PDP-11 input and output bus cables. The rest of the packaging is vendor dependent.

Link Conventions

Several conventions are required to deal with special situations.

Establishing Communications

Each front end and host interface will have its own hardware READY signal. When a machine is coming up, the READY indicator should be turned on.

If the front end is coming alive, the ready signal will cause the generation of a special interrupt in the ABSI. If the H6000 is coming alive, the signal will cause an error interrupt in the front end.

When a READY interrupt has occurred, the following steps should be taken.

1. Any in-progress transfers from the other machine should be aborted.
2. Send four (4) HFP NOP messages to the other machine.
3. Old messages for transmission to the other machine should be transmitted.

Once four HFP NOP messages have been transferred normal operation of the interface may begin.

No Response

Each side of the connection should be prepared to receive a message from the other promptly. If a message has been in transit to the other machine for N seconds, the other machine

should be declared NOT RESPONDING. An attempt should be made to establish communication as defined above. If communication cannot be established, then operator intervention is to be requested. Because of the experimental nature of this project, N must be a readily modified variable. The initial value of N will be 30 seconds. If the operator declares the other machine DEAD, any further requests to communicate with the machine should be met with a HOST DEAD response until communication is established. If a lack of response by the other machine has caused operator intervention to be requested, but the other machine responds before the operator intervenes, then the operator should be notified.

Maximum Message Size

The maximum message size of any single machine-to-machine message is an installation parameter. This parameter should be adjusted for maximum throughput and buffer utilization. A suggested initial value is 3600 bits.

Messages greater than the maximum message size should be discarded by the receiver. If, in sending a message, the sender notices that an error has occurred, the sender should immediately send a message of length greater than the maximum. Since the transmission of the first message was incomplete, the receiver will see the second message as a continuation of the first and will discard both.

Line Integrity

The connection between the H6000 and the PDP-11 uses ARPA Network

IMP-to-host data transmission conventions. The only integrity check on the link is an IOM-to-ABSI parity check. MIT's experience is one parity error on the average of every five days. The same IOM to ABSI parity check will also be used for this project. If reliability is unacceptable, it will be necessary to add error detection and correction techniques.

OFFLOADING TELNET

Introduction

This section describes the offloading of the ARPA Network Telnet protocol from a Honeywell H6000 host to a DEC PDP-11/70 front end. The use of HFP to support Telnet functions is also described.

The Telnet protocol uses the concept of a Network Virtual Terminal (NVT). The NVT defines standard characteristics for terminals using the protocol. Examples of such characteristics are line length, character encoding, and end-of-line characters. A Telnet protocol implementation has two main components: User Telnet and Server Telnet. A User Telnet implementation resides on one network host and maps specific terminal characteristics to and from the NVT. The NVT-encoded inputs are passed across the network to a Server Telnet implementation which maps them into a form indigenous to the local operating system. These inputs are then passed on to interactive programs. The process is reversed in the other direction. This mechanism allows terminals on one host to communicate with programs on another.

The offloading of User Telnet and the offloading of Server Telnet are discussed separately. User Telnet is specified in some detail. Server Telnet is only discussed briefly here; details will appear in forthcoming documents.

Offloading User Telnet

Overview

An ARPA Network User Telnet implementation already exists for handling terminals connected to the network front end. If host terminals and processes could masquerade as terminals directly connected to the front end, this front end User Telnet implementation could be used. This strategy has several advantages.

1. Host and front end terminal users are presented with a uniform User Telnet interface.
2. Very little, if any, Telnet-specific software is required in the host.
3. The residual host software is small.
4. Other programs in the front end could be accessed from the host in the same way.

The Program Access Process-to-Service Protocol (CAC Technical Memorandum 81) allows host terminals to appear to the front end as if they were terminals directly connected to the front end. Thus, User Telnet may be completely offloaded by using the Program Access Service to gain access to the User Telnet program in the front end.

The Relay Process

The advantage of a maximum offloading strategy is that only a simple relay process needs to be implemented in the H6000. The role of the relay process is to transmit properly formatted information between a host process or terminal and the Program

Access Service in the front end. HFP Messages are used, as specified below.

The Relay Process software must

1. ensure that security information is properly transmitted in BEGIN Commands;

2. properly handle half-duplex terminals (via the GO AHEAD facility);

3. transform interrupt characters into SIGNAL Commands;

4. flush data when instructed to do so (via SIGNAL Commands); and

5. accept data from terminals with differing character sets and properly transform them for transmission to the front end.

HFP Channel-Level Messages

This section describes how the various HFP Messages are used in offloading User Telnet. A familiarity with the Program Access Process-to-Service Protocol is assumed. All ASCII characters are 7-bit ASCII right-justified in a 9-bit field.

BEGIN Command and Response. The format and semantics of the BEGIN Command and Response are as described in the Program Access Process-to-Service Protocol specification, with the

following qualifications. The Security field of the BEGIN Command contains a 36-bit count, followed by a string of ASCII characters (each right-justified in 9 bits). The string consists of

a 12-character user ID,

a 12-character password, and

a 4-character SCC.

The last character of the SCC is padding. The value of the count field is 28. The Command field of the BEGIN Command contains the lower case ASCII string "telnet" followed by a line-feed.

TRANSMIT Command and Response. The format and semantics of the TRANSMIT Command and Response are as described in the Program Access Process-to-Service Protocol specification with the following qualifications. The Command TEXT consists of

a 9-bit Control field,

a 27-bit Character Count field, and

a variable length Data field.

The Control field contains a bit (the "GO AHEAD" bit) which facilitates control of half-duplex terminals. The Character Count field contains the number of ASCII characters in the Data field. The Data field contains a string of 9-bit ASCII characters. This string is either a command to the Telnet program in the front end or data to be transmitted over the network by the Telnet program. APPENDIX I contains a description of the user interface to the Telnet program in the front end.

Other Commands and Responses. All other Commands and Responses are as described in the Program Access Process-to-Service Protocol specification.

Sample Operation

The following table lists a sequence of HFP Commands and Responses that show:

1. the execution of the front end Telnet program,
2. a connection to a foreign host,
3. the closing of that connection, and
4. destruction of the HFP communications channel.

The TEXT fields of the Commands are enclosed in parentheses. ASCII strings are indicated by quotation marks. All of the TRANSMIT Responses are assumed to indicate successful receipt of the immediately preceding TRANSMIT Command from the other end; details such as sequence numbers are omitted.

Host		Front End
BEGIN(name,pass, "telnet")	->	
	<-	BEGIN Response, success
	<-	TRANSMIT ("UNIX User Telnet")
TRANSMIT Response	->	
TRANSMIT ("connect Multics")	->	
	<-	TRANSMIT Response
	<-	TRANSMIT ("Attempting Connection")
TRANSMIT Response	->	
	<-	TRANSMIT ("Connection Open")
TRANSMIT Response	->	
	<-	TRANSMIT ("Multics 30.2: Load = 42.0")
TRANSMIT Response	->	
TRANSMIT ("login Joe")	->	
	<-	TRANSMIT Response
	<-	TRANSMIT ("Password: [illegible]")
TRANSMIT Response	->	
TRANSMIT ("Joepasswd")	->	
	<-	TRANSMIT Response
	<-	TRANSMIT ("Ready")
TRANSMIT Response	->	
TRANSMIT ("logout")	->	
	<-	TRANSMIT Response
	<-	TRANSMIT ("CPU = 3 sec")
TRANSMIT Response	->	
TRANSMIT ("^bye")	->	
	<-	TRANSMIT Response
	<-	END, normal
END Response	->	

Offloading Server Telnet

The Process-to-Service Protocol for offloading the server side of a Virtual Terminal Protocol (Server Telnet) will be defined in a forthcoming document (CAC Technical Memorandum 82). This protocol allows some flexibility in the choice of facilities to be offloaded. Implementation details which are needed for this experimental study will be specified in a brief memorandum which will be produced as an addendum to the H6000 Software Specifications.

The functions of Server Telnet are:

1. manipulation of network connections,
2. transmission of data,
3. mapping between local and network representations,
4. negotiation of Telnet options,
5. handling of special control functions, and
6. interfacing to the local system.

This last function is necessarily resident in the host. The first two are best offloaded to the front end. The remaining functions may be implemented in either the host or the front end. Some of these, such as the handling of special control functions, can conveniently be offloaded in their entirety. Other functions, such as the negotiation of options, necessarily require some residual host software. Local installation constraints may also affect the decision as to where to put each function.

REFERENCES

1. Specifications for the Interconnection of a Host to an IMP. BBN Document Number 1822.

2. Digital Equipment Corporation Peripherals Handbook, DR11-B Specification, pp 4-195 to 4-199.

3. Asynchronous Bit Serial Interface (ABSI) for Connecting the Honeywell H6180 to an ARPA IMP, John Ward, March 3,1975, Electronic Systems Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139.

APPENDIX I - UNIX USER TELNET

Caveat:

The format of this appendix is that of a Command description from the Unix Programmers' Manual. It is essentially documentation for the Unix User Telnet program. We include it here unedited for the reader's information.

DATE

11/09/75

NAME

Telnet - Converse with a host via the ARPA Network

SYNOPSIS

telnet

DESCRIPTION

Telnet allows a user to communicate with a foreign timesharing system on the ARPA Network. Its controlling actions can be divided into two classes. The first is a set of commands related to the opening, telnet protocol (see Telnet Protocol Specification NIC 18639) conversation with, and the termination of connections with foreign Network servers. The second class refers to the treatment of characters obtained from the standard input file.

The Connection related commands are:

con[nect] The connection command asks to initiate a connection the host specified, either via the hostname (see July 1975 Arpanet Directory for a complete list of host names) or octal host number as specified by the ´-h´ option. A specific socket may be requested with the ´-s´ option.

clo[se] The close command asks that the current connection be closed.

bye Exits the telnet program.

end Exits the telnet program.

bre[ak] Send a Telnet defined break command to the server.

abo[rt] Send a Telnet defined abort output command to the server.

ayt Send a Telnet defined Are You There command.

goa[head] Send a Telnet defined Go Ahead command.

syn[ch] Send a Telnet defined Synch sequence.

The Terminal related commands are:

cha[rmode] As the characters are read into the program send them to the network.

msg[mode] Wait for the endline character to be detected before send the characters off to the foreign host (default).

ech[o] Turn local echoing on or off. Turned on by 'echo on' turned off by 'echo off'. Default is on.

fla[g] < new flag character > Change the flag character. The flag character is used to get the 'attention' of the telnet program while a connection is open. The default is ''^''. This is used to change terminal parameters and execute local UNIX programs while a connection is open. For example the 'close' command is always executed with a flag char in front (^close).

eli[ne] Used to change the endline character. The default is carriage return. (see msgmode)

ten[ex] This mode is particularly useful for people conversing with a Tenex. It puts the local telnet in character mode, and sends a 'half' command down the open! connection. What this means is that every character type will be sent to the host but the echoing is done locally. This results in halving the system load for character-at-a-time users. Since each character doesnt have echoed from the foreign host. To this point it seems to work well with the exception of passwords, which can be gotten around by typing ''^echo off' <password> ''^echo on'. Please try this, it gives you faster apparent response, and eases the load on the system.

wai[t] <number of secs> This commands suspends gathering input from the terminal for n seconds (specified in octal). Its main use is in define strings where one would like to wait until he is sure a foreign host has finished the last command.

There is subclass of terminal commands that was implemented in hopes of satisfying most of the multiple character set problems that result from using one timesharing system thru one or more timesharing systems, and to in some sense allow the user to customize his keyboard into a character set 'mappable' to frequently used character strings. Effectively these commands allow one to map any single character into an arbitrary length, arbitrary content character string. For example my control b makes a connection to bbn, and logs in (ahh true lazyness). Admittedly this much bell and whistling may be more than a user needs, but we'll try it for awhile and see how much use it gets.

def[ine] Define a character to any string of arbitrary characters and ask if it is to be echoed to the terminal when invoked. The command is self documenting. Backslash may be used to escape any character.

sav[e] Save the current character mappings in a file named
'<home dir>/character_set'.

loa[d] Load a character map set from '<home dir>/character_set'. Done automatically at initial execution if a character_set file already exists.

prt Print the current set of character mappings.

SCENARIO

This is an example of the use of the define command. It maps the cntrl b character into a connection to bbn, execution of the 'tenex' command, and goes thru the bbn login sequence. It is important to remember that each line of the define string is terminated with a cr and an lf, so be careful when defining characters.

```
define
Character to be mapped:^B
Enter String (end with cntrl c)
con bbn<cr><lf>
^wait 10<cr><lf>
^tenex<cr><lf>
^echo off<cr><lf>
log holmgren pass acct <cr><lf>
^echo on<cr><lf>
^C
Do you want this string echoed(y or n):n
```

FILES

/dev/net/<hostname>

DIAGNOSTICS

Many, nothing too wierd though.

BUGS

Of course, although none are currently known.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
CAC Document Number 220
CCTC-WAD Document Number 7501

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
Networking Research in Front Ending and Intelligent Terminals - H6000 Software Specifications

5. TYPE OF REPORT & PERIOD COVERED
Research

6. PERFORMING ORG. REPORT NUMBER
CAC #220

7. AUTHOR(s)
P.A. Alsberg et al.

8. CONTRACT OR GRANT NUMBER(s)
DCA100-76-C-0088

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Command and Control Technical Center
WWMCCS ADP Directorate 11440 Isaac Newton Sq. N.
Reston, VA 22090

12. REPORT DATE
January 10, 1977

13. NUMBER OF PAGES
33

14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*

15. SECURITY CLASS. *(of this report)*
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*
Copies may be requested from the address given in (11) above

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*
No restriction on distribution

18. SUPPLEMENTARY NOTES
None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
network front end
network protocol

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
The CAC is engaged in an investigation of the benefits to be gained by employing a network front end. A DEC PDP-11/70 is being used as front end for connecting a Honeywell 6000 host to the ARPANET. This document presents an overview of the software required in the H6000 for accessing the ARPANET through the front end. Some implementation-specific details not supplied in other documents are presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*