Engin

C

Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA ILLINOIS 61801

CAC Document Number 240
CCTC-WAD Document Number 7518

Networking Research in Front Ending and Intelligent Terminals

ENFE Final Report

September 30, 1977

CAC Document Number 240
CCTC-WAD Document Number 7518

Networking Research in Front Ending and Intelligent Terminals

ENFE FINAL REPORT

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

September 30, 1977

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

TABLE OF CONTENTS

SUMMARY

ackground

Under contract DCA100-76-C-0088, the Center for Advanced omputation of the University of Illinois at Urbana-Champaign has nvestigated the capabilities of network front ends. As a part f that contract, an experimental network front end (ENFE) has een developed to interface a World-Wide Military Command and ontrol System (WWMCCS) H6000 to the ARPA network and to conduct xperiments with the proposed ARPANET Host-to-Front-End Protocol. total of 194.28 man-months were expended over a period of 12 onths (1 October 1976 to 30 September 1977).

An experimental network front end (ENFE) was a primary ontract deliverable. Delay in GFE hardware delivery signifiantly decreased the work that could have been accomplished. igital Equipment Corporation delivered the ENFE development minicomputer (DEC PDP-11/70) three months late. This slowed contruction of critical ENFE operating system software. Associated omputer Consultants delivered the Honeywell-DEC communications ink three months late, which delayed software installation, esting, and evaluation. Therefore, only a crude evaluation of NFE capabilities is available at this time. Despite these deays, however, all of the essential work contracted for has been uccessfully completed. The ENFE was built, tested, and is urrently operational.

NFE Research Program

The ENFE research program was organized into two teams. he first team implemented the ENFE. The ENFE uses a DEC DP-11/70 computer running a Unix general purpose operating sys- em. The Unix operating system capabilities were expanded to rovide more general support for Host-to-Front-End Protocol (HFP) oftware. Measurement software was added to Unix to support ests and expériments.

The second team was concerned with protocol issues. This eam revised the HFP specifications, generated Telnet protocol ptions for Honeywell VIP terminals, conducted offloading stu- ies, followed AUTODIN II protocol developments, and constructed plan for research into alternative front-end architectures.

Individuals and small groups were drawn from both of hese teams to generate an experiment plan and to carry out and nalyze ENFE tests and experiments.

There were some risks associated with the ENFE research rogram. The state of the art in network communications is eared to machine/terminal interaction rather than achine/machine interaction. A generalized machine-to-machine rotocol like HFP had never been used to facilitate communica- ions between a large computer and a front end. Furthermore, nix was not designed to support high speed message switching. iven the state of the art in HFP and the architecture of Unix, he Unix ENFE and HFP software were strictly experimental. The rimary product of this research program was experience with the

host front-ending problem.

All work (except the multi-host study) performed under the contract has already been thoroughly documented (Tables 1 and 2). Thus, this final report will abstract those reports produced.

Table 1
Contract Deliverables

AC Document Number	CCTC-WAD Document Number	Title	Date
220	7501	DRAFT H6000 Software Specifications	15 Nov. 1976
221	7502	DRAFT Experimental Network Front End Functional Description	15 Dec. 1976
220	7501	FINAL H6000 Software Specifications	10 Jan. 1977
221	7502	FINAL Experimental Network Front End Functional Description	15 Jan. 1977
227	7509	DRAFT Experimental Network Front End Experiment Plan	28 Mar. 1977
227	7509	FINAL Experimental Network Front End Experiment Plan	16 May 1977
232	7512	DRAFT Alternative Architecture Research Plan	16 May 1977
233	7515	DRAFT Experimental Network Front End Software Functional Description	1 July 1977
233	7515	FINAL Experimental Network Front End Software Functional Description	1 Aug. 1977

AC Document Number	CCTC-WAD Document Number	Title	Date
232	7512	FINAL Alternative Architecture Research Plan	30 Sept. 1977
239	7517	UNIX/ENFE Experimental Performance Report	30 Sept. 1977
230	7511	Offloading ARPANET Protocols to a Front End	30 Sept. 1977
241	7519	ENFE Nassi-Shneidermann Flow Charts	30 Sept. 1977
242	7520	ENFE Listings and Object Code	30 Sept. 1977
240	7518	ENFE Final Report	30 Sept. 1977

Table 2
Unscheduled Reports Delivered

AC Technical Number	CCTC-WAD Document Number	Title	Date
219	7503	Host to Front End Protocol/Version I	19 Aug. 1977
80	7504	ARPANET Host-Host Process-to-Service Protocol Specification	17 Mar. 1977
81	7505	Program Access Process-to-Service Specification	10 Mar. 1977
82	7506	Server Virtual Terminal Process-to Service Protocol Specification	17 Mar. 1977
84	7507	Illinois Inter-Process Communication Facility for Unix	1 Apr. 1977
94	7514	Telnet Data Entry Terminal Option	27 June 1977

ENFE SOFTWARE ARCHITECTURE

General Description

The offloaded network software can be thought of as a set of services provided to host (H6000) processes or to users. These services allow the network and the various hosts connected to the network to be conveniently used. A complete functional description of the ENFE software architecture is contained in CAC Document No. 233. The key features are summarized below.

Host-to-Front-End Communications

A basic mechanism must be provided to support communication between host processes and front-end services. This mechanism is the Host-to-Front-End Protocol (HFP), which is defined in CAC Document 219 (ARPA Request for Comments (RFC) 710). The HFP specification distinguishes two protocol layers: the channel protocol and the process-to-service protocols.

Channel Protocol. By means of the channel protocol, logical channels are set up between host processes and the front-end services, and messages are transmitted on these channels. Provisions are made for flow control and for out-of-sequence signaling. The channel protocol defines five types of HFP Messages:

1. BEGIN, which sets up logical channels;

2. END, which terminates logical channels;

3. TRANSMIT, which transmits data;

4. SIGNAL, which provides a means for synchronizing the ends of a logical channel, for interrupting the other end, and for flushing data from the other end of the channel; and

5. EXECUTE, which provides a means for passing service-specific information "out of band;" i.e., outside the strict sequencing required for TRANSMIT Messages.

ach Message type can be either a Command (requesting that the ction defined by the Message be taken) or a Response (indicating hether the action was taken and, if not, providing some explana-ion). The HFP specifications use the capitalized word, Message, o refer to these Message types.

Channel Protocol module. The front end contains a oftware module, the Channel Protocol module (CPM), which manages he logical channels and serves as a bi-directional multiplexor. he host also contains a CPM which similarly manages the other nds of the logical channels.

rocess-to-Service Communications

Communications between a host process and a front-end ervice may be divided into three stages:

1. communications between the host process and the host CPM (described in CSC Document No.

R493700056-2-1, "Host to Front-End Processor Protocol Interface Functional Description"),

2. communications between the host CPM and the front-end CPM (described in CAC Document No. 220, "H6000 Software Specifications" and CAC Document No. 219, "Host-to-Front-End Protocol"), and

3. communications between the front-end CPM and a front-end service (described in CAC Document No. 233, "Experimental Network Front End Software Functional Description").

Process-to-Service Protocols. The process-to-service rotocols specify the content, sequencing, and type of HFP Mesages by which host processes communicate with front-end serices. The process-to-service protocols implemented in the ENFE re:

1. ARPANET Host-Host Process-to-Service Protocol (CAC Technical Memorandum No. 80),

2. Program Access Process-to-Service Protocol (CAC Technical Memorandum No. 81), and

3. Server Virtual Terminal Process-to-Service Protocol (CAC Technical Memorandum No. 82).

Service Structure. Each front-end service implements one rocess-to-service protocol. All front-end services execute ithin their own address spaces; i.e., as user-level programs.

Each program is structured as a finite state machine that ccepts two types of inputs. HFP Message inputs are generated by rocesses in the host requesting action from the front-end serices. I/O completion event inputs are generated by the system

n response to service-initiated device I/O operations. Each nput is associated with a specific HFP logical channel. The nput type and current channel state determine the immediate ction and next channel state. Most actions result in the ransmission of data to another destination and in the generation f an HFP Response indicating the success or failure of the ac-ion.

Host-Host Service module. The ARPANET Host-Host Service HHS) module enables programs running in the host to use the RPANET Network Control Program (NCP) in the front end. It im-lements the ARPANET Host-Host process-to-service protocol.

Program Access Service module. The Program Access Service PAS) module enables programs running in the host to execute rbitrary programs in the front end. It implements the Program ccess process-to-service protocol.

Server Virtual Terminal Service module. The ARPANET erver Virtual Terminal Service (SVTS) module enables programs on ıe host to be accessed by terminals connected to other hosts on ıe ARPANET. It implements the ARPANET Server Virtual Terminal rocess-to-service protocol. It also implements the ARPANET Tel-et protocol described in NIC Document No. 15372.

PROTOCOL SPECIFICATIONS

ost-to-Front-End Protocol

The performance of data communications tasks such as terinal handling and network protocol interpretation can impose a ignificant load on a host. Some of these tasks can be performed y the front end for the host. The Host-to-Front-End Protocol HFP) defines a form of communication between the host and the ront end to enable this "offloading" of services.

Thus, the HFP provides specifications for:

1. a channel protocol,

2. individual Commands and Responses,

3. the process-to-CPM interface, and

4. the service-to-CPM interface.

ı addition, the HFP provides specifications for specifying rocess-to-service protocols.

Each HFP Message contains a HEADER carrying channel pro-ocol information and may contain TEXT carrying process-to-rvice protocol information. Process-to-service protocols use P Messages to carry information between a process and a service odule. The HFP Message types are:

1. BEGIN Command/Response,

2. END Command/Response,

3. TRANSMIT Command/Response,

4. SIGNAL Command/Response, and

5. EXECUTE Command/Response.

RPANET Host-Host Process-to-Service Protocol

CAC Technical Memorandum No. 80 specifies a process-to-ervice protocol for providing ARPANET Host-Host Protocol and nitial Connection Protocol services to a process through the FP. The Host-Host Protocol is the basic inter-process communica-ion protocol for the ARPANET (ARPANET NIC Document 8246). The rogram which implements it in each host is the Network Control rogram (NCP). The service described here provides an interface, hrough the HFP, between a process in a host and an NCP in a ront end. This enables the host process to establish and use RPANET connections.

ogram Access Process-to-Service Protocol

CAC Technical Memorandum No. 81 specifies a process-to-rvice protocol for the execution of, or attachment to, arbi-ary programs in the front end. The intent was to provide a neral mechanism that would allow the host to access terminal-iented front-end services. Examples of such services are User lnet and teleconferencing.

The protocol assumes that the program access service

itself is completely offloaded to the front end. The only software remaining in the host is a relay process that passes properly formatted data between a host terminal or process and the Program Access Service module in the front end. HFP TRANSMIT Commands are used for this data transmission.

Server Virtual Terminal Process-to-Service Protocol

CAC Technical Memorandum No. 82 specifies a protocol for offloading the server side of a virtual terminal protocol; e.g., Server Telnet for the ARPANET. This protocol allows some flexibility in the degree of offloading that may be achieved. Although the protocol is applicable to a general virtual terminal service, the discussion in the specification is in terms of the ARPANET Telnet Protocol, which is currently the only such protocol widely used.

The functions of the typical Server Telnet implementation include:

1. manipulating network connections,
2. negotiating Telnet options,
3. mapping between local terminal representations and network virtual terminal representations,
4. transmitting data over connections,
5. handling special control functions, and

6. interfacing remote terminals so that they appear as if they were local terminals.

The server virtual terminal process-to-service protocol ıses HFP Messages to carry information between the residual part ·f Server Telnet in the host (the "process") and the Server Virual Terminal Service module (the "service") in the front end.

ther Process-to-Service Protocols

Further study of the problem of offloading Telnet led us o conclude that a single, symmetric protocol should be designed o handle both User and Server Telnet services. We have contructed, but have not implemented, such a protocol (CAC Technial Memorandum No. 1Ø3, "Network Virtual Terminal Process-toervice Specification"). This protocol specification, in addiion to specifications for three process-to-service protocols onstructed in connection with our study of strategies for ofloading the ARPANET File Transfer Protocol, is appended to CAC ocument No. 23Ø, "Offloading ARPANET Protocols to a Front End."

ɘlnet Data Entry Terminal Option

Under the current contract, we have been tasked to proide facilities for attaching data entry terminals, specifically ıe 77Ø5 VIP terminal, to the ENFE and the Telnet software. Howver, the Telnet protocol was originally designed to support sime, scroll-mode terminals. To get the maximum amount of usefulɘss out of a data entry terminal, the Telnet protocol needed to

e extended. Fortunately, the Telnet protocol has a built-in echanism, the "option negotiation" mechanism, to allow such xtension. We have therefore defined an option to support data ntry terminals. In effect, this option defines the Network Virual Data Entry Terminal. This option supports a minimal set of seful functions common to most data entry terminals and also llows a number of highly sophisticated functions to be negotiatd. Details of this option may be found in "Data Entry Terminal ption," CAC Technical Memorandum No. 94 (ARPANET RFC 731).

EXPERIMENTATION

xperimental Network Front End Experiment Plan

This document (CAC Document No. 227) described our preliminary plans for the ENFE tests and experiments. The three main ections identified:

1. goals of the experimentation,
2. tools for experimentation, and
3. scenarios for specific experiments.

Goals. The goals of the experiment plan fell into three ategories:

1. performance testing,
2. fine-tuning of the system, and
3. investigating benefits to be gained from design changes.

ests were proposed to determine whether the system works as it s supposed to and is able to provide the front-ending facilities eeded in the short term by the WWMCCS community. In particular, hroughput and terminal support tests were given high priority.

Tools. In this document we described an optimal set of ools for thoroughly understanding the front end. The set we lanned to implement included:

1. timing mechanisms for generating interrupts as specified by the experimenter and for timestamping messages,

2. artificial traffic generators (involving both software and hardware) at the three front-end

interfaces (the interfaces to the host, to the network, and to the terminals), and

3. software to collect data as the experiments are run.

ther tools described in this document such as a simulation proram and queueing theory analysis are essential to a follow-on tudy of how the front-end design may be improved.

Specific Tests. In the last section of the report, we resented a set of scenarios for specific tests. These were ecessarily tentative, particularly as to details. We also mapped ut a more comprehensive program of experimentation than we exected to be able to carry out under the current contract. Tests id experiments will continue under a follow-on contract (the iase B work).

NIX/ENFE Experimental Performance Report

CAC Document No. 239 reports on the results of the Phase program of testing and experimentation.

Experimentation Software. The front-end tests used a onfiguration of software modules similar to the standard ENFE onfiguration, except that local and foreign host processes were mulated by processes resident in the ENFE itself. These ocesses served as message generators, sending messages to each her via the standard ENFE modules. An extra copy of the Chanel Protocol module was also included in the ENFE to interface e "local host" message generator to the front-end's Channel

rotocol module.

In order to make accurate timing measurements, a pro-rammable clock was attached to the PDP-11/70. System calls were mplemented to enable the experiment software to utilize this lock to get clock readings and to schedule interrupts. Times-amping software was built into the front end at various points. his software inserted clock readings (timestamps) into the texts f messages as they were transmitted through the front end. In his way the progress of a message through the ENFE could be easured to a high degree of accuracy.

Small modifications were made in the standard Unix moni-oring facilities to provide for monitoring of kernel-level as ell as user-level processes. This allowed a detailed analysis f processor usage.

<u>Experiment</u> <u>Results</u>. A large part of the Phase A testing d evaluation task involved testing the software to make certain at it operates correctly. A certain amount of fine-tuning ore than was anticipated in the Experiment Plan) was also car-ied out. The Phase A measurements reported here have allowed us identify which portions of the system need to be made more ficient and to draw broad conclusions regarding the system chitecture.

The measurements reported here include:

1. timing tests of the Inter-Process Communication (IPC) primitives.

2. timing of the progress of single messages

through the front end,

3. monitoring of processor usage, and

4. saturation throughput using several different configurations.

From timing the IPC primitives, we found that it requires minimum of five to six milliseconds to relay a message from one ront-end process to another. The single-message timing measure-ents indicate that (except in the case of the NCP) the time a essage spends being processed by the modules is a small fraction f the time required to relay the message between modules. Moni-oring the processor usage by the Channel Protocol module and by he Host-Host Service shows that 85 to 90 percent of CPU time is xpended in system calls. Furthermore, kernel-level monitoring hows that about half of this time is just in the overhead of aking system calls. We conclude that, as long as the front-end rchitecture requires Unix system calls to relay messages from ne module to another, no dramatic improvement in the efficiency f the front-end services can be expected.

Obtaining meaningful throughput measurements has been ade difficult by the self-contained experimentation configura-ion, which included two passages through the NCP. In this con-iguration, we found that throughput is severely limited by the CP. To investigate the extent of this limitation, we have sent essages as fast as possible from the ENFE through the Urbana IMP an 11/50 at Urbana. With this configuration, each message is andled only once by the ENFE NCP. The maximum throughput

easured to date with this configuration is about 50 kilobaud alreadyenough to saturate the ARPANET) when messages are sent y the 11/50 to the ENFE. However, when messages are sent from ne ENFE the throughput is roughly 40 percent less. We attribute nis difference to inability of the slower 11/50 to receive data apidly.

To further investigate the factors affecting throughput, e have separately exercised the two major portions of the message path in the self-contained configuration. The front-end ortion of the path (from the message generator to the Host-Host ervice) has a message throughput that is four to five times eater than that of the network portion (from a message generaor through the NCP to the IMP and back through the NCP to a message receiver). These results provide corroboration of the limting effect of the NCP.

IMPLICATIONS OF EXPERIMENT RESULTS

Multi-Host Study

In this study we examine the impact of a multi-host confi-guration on a network front end (NFE). This impact is a function of both the capacity and the structure of the NFE. In this study, the NFE examined is the WWMCCS Phase A Experimental NFE ENFE). Our estimates show that the WWMCCS Phase A ENFE can sup-port a multi-host configuration. Minor protocol changes may be equired.

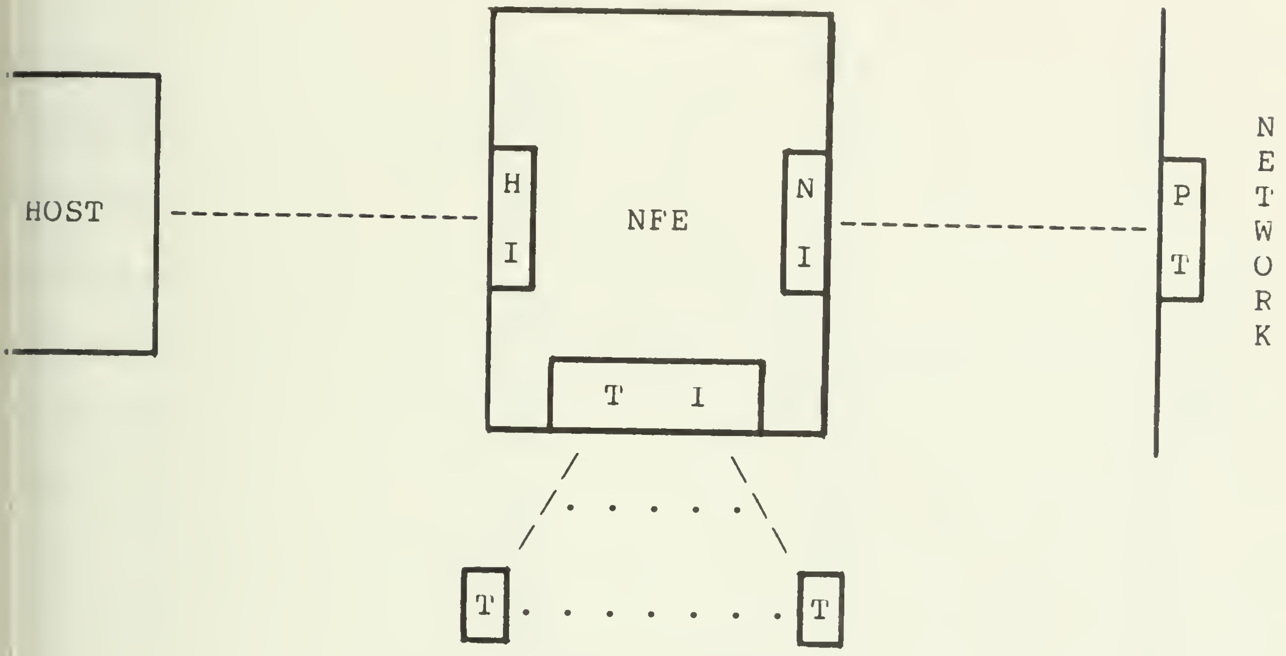
Figure 1 shows a model NFE in a single-host configuration. he NFE communicates with the host via a host interface. In the NFE this is the Asynchronous Bit Serial Interface (ABSI). The FE communicates with the network via a network interface. In he ENFE this is the IMP-11A. The NFE communicates with termi-als via terminal interfaces. In the ENFE, these are DH-11 and V-11 interfaces.

Figure 2 shows an NFE in a multi-host configuration. The ifference between this and the single-host configuration is that the multi-host configuration there are additional host inter-ces (ABSI's in the ENFE).

We discuss the quantitative and qualitative effects of of lti-host configurations on the NFE. We first deal with the antitative effects. In so doing, we develop a method for termining the resources required in the NFE for a given load

.mposed by the many hosts. We then deal with the qualitative ef-
'ects on the NFE and the protocols that it uses.

Single-Host NFE Configuration

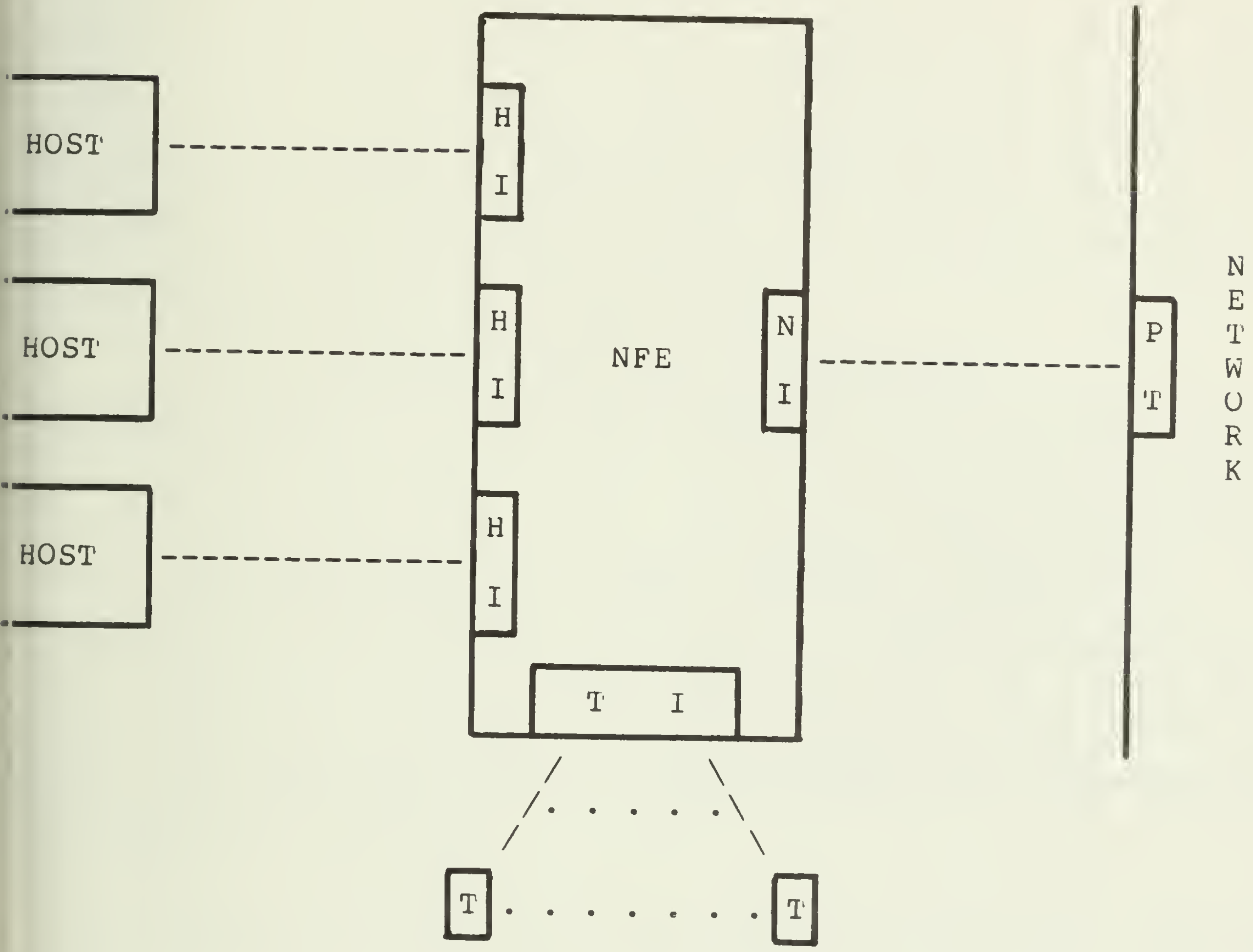


ere:

HI	=	Interface to the host
NI	=	Interface to the network
TI	=	Interfaces to terminals
PT	=	Port
T	=	Terminal

Figure 1.

Multi-Host NFE Configuration



ere:

HI	=	Interface to the host
NI	=	Interface to the network
TI	=	Interfaces to terminals
PT	=	Port
T	=	Terminal

Figure 2.

When we ask whether an NFE can support a given multi-host onfiguration, we are asking, in part, whether the resources vailable in the NFE are sufficient to support the load imposed y the hosts. Put another way, if Rlimit is the amount of a iven resource R which is available in the NFE and Rreq is the otal amount of the resource R which is required to support the oad imposed by the hosts, we must have:

$$\text{Rreq} \le \text{Rlimit, for all resources R.}$$

o determine whether or not this condition holds, we must compute req. To do this, we break the NFE down into its component odules, and determine the resources required by each module. We hen analyze the load in terms of the use of NFE modules which it ntails. Finally, we obtain the total resource requirement by umming the resources required by each module for supporting the oad imposed by the hosts.

Let Rmodule[m] be the amount of resource R that is required y module m. Then Rreq is computed by the summation

$$\text{(1)} \qquad \text{Rreq} = \sum_{m=1}^{M} \text{Rmodule}[m],$$

here M is the number of modules in the NFE. Rmodule[m] will in eneral consist of two parts: the fixed part, Rfixed[m], and the ariable part, Rvar[m]. Thus

$$\text{(2)} \qquad \text{Rmodule}[m] = \text{Rfixed}[m] + \text{Rvar}[m].$$

fixed[m] is the amount of resource R which is required by module

independently of the load. Rvar[m] is the amount of resource R hich is required by module m and which varies with the load.

We can usually determine Rfixed[m] directly from m by measrement. To determine Rvar[m], we first determine the amount of esource R that is required by module m for each unit of load. nis we call Rload[m]. We then determine the total load that enails the use of module m. This we call Lmodule[m]. Then

$$\text{(3)} \qquad \text{Rvar}[m] = \text{Rload}[m] * \text{Lmodule}[m].$$

module[m] will be the sum of the loads which are imposed by each f the hosts and which entail the use of module m. We let L[h,m] e the load which is imposed by host h and which entails the use f module m. If H is the number of hosts,

$$\text{(4)} \quad \text{Lmodule}[m] = \sum_{h=1}^{H} L[h,m],$$

$$\text{(5)} \quad \text{Rvar}[m] = \text{Rload}[m] * \sum_{h=1}^{H} L[h,m],$$

$$\text{(6)} \quad \text{Rmodule}[m] = \text{Rfixed}[m] + (\text{Rload}[m] * \sum_{h=1}^{H} L[h,m]), \text{ and}$$

$$(7)\quad Rreq = \sum_{m=1}^{M} (Rfixed[m] + (Rload[m] * \sum_{h=1}^{H} L[h,m])).$$

e note that the model we have developed employs a linear relaionship between load and resource consumption. In the real orld, such a relationship does not always hold, particularly ith respect to time-limited resources such as processor time.

We now discuss this method in more detail. In an NFE, the esources we consider are of two kinds: space-limited resources nd time-limited resources. The space-limited resource we conider is primary memory. The time-limited resources we consider e processor time and interface bandwidth. We will discuss the omputation of Rreq for primary memory in some detail. We will scuss the other two resources only insofar as their treatment ffers from that of primary memory.

We first consider the primary memory resource C. Let Climit the total available memory in the NFE. We want to compute eq, the total amount of memory required to support the load imsed by the hosts. Then we can determine whether

$$Creq \leq Climit.$$

this relation holds, then the memory in the NFE is sufficient support the load imposed by the hosts.

Consulting equation (4), we see that the first step in comting Creq requires the determination of L[h,m] for each host h

nd for each module m. This information can be conveniently epresented by a matrix:

$$\begin{matrix} L[1,1] & L[1,2] & \cdots & L[1,M] \\ L[2,1] & L[2,2] & \cdots & L[2,M] \\ \vdots & \vdots & & \vdots \\ L[H,1] & L[H,2] & \cdots & L[H,M] \end{matrix}$$

ach row represents the load which is imposed by a given host. ach column represents the load which entails a given module. We ote that the sum of column m is Lmodule[m].

To determine the L[h,m], we need two sets of information hich involve the services which the NFE performs for the hosts. e let U[h,s] be the number of simultaneous uses of service s by ost h. This information can also be represented by a matrix:

$$\begin{matrix} U[1,1] & U[1,2] & \cdots & U[1,S] \\ U[2,1] & U[2,2] & \cdots & U[2,S] \\ \vdots & \vdots & & \vdots \\ U[H,1] & U[H,2] & \cdots & U[H,S] \end{matrix}$$

here S is the number of different services performed by the NFE. ich row represents the use of services by a given host. Each olumn represents the use of a given service across all hosts.

We let E[s,m] be the number of unit loads imposed on module m for each use of service s. Again the information can be presented by a matrix:

$$\begin{matrix} E[1,1] & E[1,2] & \cdots & E[1,M] \\ E[2,1] & E[2,2] & \cdots & E[2,M] \\ \vdots & \vdots & & \vdots \\ E[S,1] & E[S,2] & \cdots & E[S,M] \end{matrix}$$

ach row represents the load on the modules of the NFE for each se of a given service. Each column represents the load on a iven module when each service is used once.

Then

$$(8) \qquad L[h,m] = \sum_{s=1}^{S} U[h,s] * E[s,m].$$

hat is,

$$L = UE.$$

Consulting equation (5), we see that the next step in com- ting Creq requires the determination of Cload[m] for each dule m. This is simply the amount of memory required by module m for each instance of whatever module m does. If a copy of dule m is required for each instance of whatever module m does, en Cload[m] is the total memory used by each copy of module m. the copies of module m all share the same reentrant program t each has a separate copy of the data space, then Cload[m] is e size of the data space for a copy of module m. If there is ly one copy of module m which allocates table space and buffer ace for each instance of whatever module m does, then Cload[m] i the size of the table space and buffer space allocated for ch instance of whatever module m does.

Consulting equations (6) and (7), we see that the final eps in computing Creq require the determination of Cfixed[m]

or each module m. This is simply the amount of memory required y module m independent of how many instances there are of whatever module m does. If a copy of module m is required for each nstance of whatever module m does, then Cfixed[m] is zero. If he copies of module m all share the same reentrant program but ach has a separate copy of the data space, then Cfixed[m] is the ze of the reentrant program. If there is only one copy of dule m which allocates table space and buffer space for each nstance of whatever module m does, then Cfixed[m] is the size of dule m when no such table space and buffer space is allocated.

The computation of Creq thus consists of the following 9 steps:

1) construction of the matrix U ,

2) construction of the matrix E,

3) computation of the matrix L = UE,

4) computation of Lmodule[m] using equation (4),

5) determination of Cload[m],

6) computation of Cvar[m] using equation (5),

7) determination of Cfixed[m],

8) computation of Cmodule[m] using equation (6), and

9) computation of Creq using equation (7).

note that steps 1, 2, 3, and 4 are independent of the resource der consideration. Therefore these steps need be performed ly once for all resources.

If Climit $\geq$ Creq, the total memory in the NFE will be adeate. But if the NFE is implemented on a mini-computer, we may

lso have to take into account the limitation on the amount of emory that can be addressed by each module.

We let Caddr be the address limit imposed on each module by ne structure of the NFE hardware. Then the condition that must old is

$$\text{Caddr} \geq \text{Cmodule}[m], \text{ for all modules } m.$$

his requires no additional computation, since the Cmodule[m] ere computed in step 7 above.

We now turn our attention to the time-limited resources: ocessor time and interface bandwidth.

Let us first consider the processor time resource P. This source will be treated differently from primary memory in three ys:

1) We are more likely to be interested in the fractional utilization of P than in absolute measures in seconds. That is, Plimit will be 1 and Pload[m], Pfixed[m], Pmodule[m], and Preq will be expressed in fractions of the available processor time.

2) For most modules m, Pfixed[m] will be very small or zero. Exceptions will be those modules which implement communication protocols that require continuous activity for maintaining communication, even when there is no data to be sent or received.

3) Our simple linear model of the relation between load and resource use may have to be replaced by a more sophisticated model such as queueing theory.

The last resource we consider is the interface bandwidth source B. This resource will be treated differently from pri-

ary memory in the three ways mentioned above under processor ime. Further, interfaces are specialized resources as opposed o universal resources such as memory space and processor time. his has two consequences:

1) Use of each interface will probably be limited to a single module.

2) The computations for each interface, or kind of interface, will have to be made separately.

We now apply our modeling method to a concrete example using umerical data. We set ourselves the task of computing Creq for ne ENFE in a hypothetical multi-host configuration. Our purpose s to illustrate the application of the method. Therefore we ill oversimplify wherever it suits our convenience.

We employ the 9-step method discussed above.

In step 1, we construct the matrix U. We recall that U[h,s] s the number of simultaneous uses of service s by host h. We ust therefore define the services performed by the ENFE. There e five:

1) HH performs ARPANET Host-Host Protocol interpretation for each of the local hosts.

2) SVTn serves as intermediary between the local hosts one one hand and remote terminals connected via the ARPANET on the other.

3) SVTt serves as intermediary between the local hosts on one hand and local terminals attached to the ENFE on the other.

4) UVTh serves as intermediary between terminals attached to the local hosts on one hand and remote hosts connected via the ARPANET on the other.

5) UVTt serves as intermediary between local termi-

nals attached to the ENFE on one hand and remote hosts connected via the ARPANET on the other.

e note that the service UVTt does not involve the local hosts. t only involves local terminals attached to the ENFE. To acount for the ENFE resources consumed by this service, we introuce a "phantom host" T. Let there be 3 local hosts, and let the se of ENFE services be as shown in Figure 3. This then is the atrix U.

		service				
		HH	SVTn	SVTt	UVTh	UVTt
h	1	10	5	5	10	-
o	2	12	6	7	4	-
s	3	9	3	11	6	
t	T	-	-	-	-	15

Figure 3. U[h,s]

In step 2, we construct the matrix E. We recall that E[h,s] the number of unit loads imposed on module m for each use of ervice s. We must therefore examine the structure of the ENFE identify the modules in the ENFE, determine what constitutes a nit load for each module, and determine how many unit loads are posed on each module for each use of each service.

There are 9 modules in the ENFE that are pertinent to this scussion:

1) CPM is the Host-to-Front-End Protocol (HFP) channel protocol module. Its unit load is a logical channel.

2) HHS is the HFP service module that enables the local hosts to access the ARPANET NCP. Its unit load is a logical channel and the associated du-

plex ARPANET connection.

3) NCPD is the portion of the ARPANET NCP which is implemented as a daemon process. Its unit load is a duplex ARPANET connection.

4) NCPK is the portion of the ARPANET NCP which is implemented as part of the Unix operating system kernel. Its unit load is a simplex ARPANET connection.

5) SVTS is the HFP service module that enables the local hosts to access both remote terminals connected via the ARPANET and local terminals attached to the ENFE. Its unit load is a logical channel and the associated terminal.

6) UTH is the program that enables terminals attached to the local hosts to access remote hosts connected to the ARPANET. Its unit load is a terminal and the associated ARPANET connection.

7) UTT is the program that enables terminals attached to the ENFE to access remote hosts connected via the ARPANET. Its unit load is a terminal and the associated ARPANET connection.

8) TD is the Unix terminal device driver. It enables terminals attached to the ENFE to access other modules in the ENFE. Its unit load is a terminal.

9) PA is the HFP service module that enables the local hosts to access programs in the ENFE (such as UTH). Its unit load is a logical channel and the associated program.

ven the functions performed by each of these modules, the ma-

tix E is as shown in Figure 4.

module

service	CPM	HHS	NCPD	NCPK	SVTS	UTH	UTT	TD	PA
HH	1	1	1	2	0	0	0	0	0
SVTn	1	0	1	2	1	0	0	0	0
SVTt	1	0	0	0	1	0	0	1	0
UVTh	1	0	1	2	0	1	0	0	1
UVTt	0	0	1	2	0	0	1	1	0

Figure 4. E[s,m]

In step 3, we compute the matrix L = UE. This is shown in igure 5. We now have the load imposed on each module by each ost. For example, host 1 imposes 3Ø unit loads on the CPM.

		module								
		CPM	HHS	NCPD	NCPK	SVTS	UTH	UTT	TD	PA
h	1	3Ø	1Ø	25	5Ø	1Ø	1Ø	Ø	5	1Ø
o	2	29	12	22	44	13	4	Ø	7	4
s	3	29	9	18	36	14	6	Ø	11	6
t	T	Ø	Ø	15	3Ø	Ø	Ø	15	15	Ø

<u>Figure 5. L[h,s]</u>

In step 4, we compute Lmodule[m] by summing the columns of L For example, the sum for the CPM column of L is:

Lmodule[CPM] = 3Ø + 29 + 29 + Ø = 88.

now have the total load imposed on each module by all hosts.

In step 5, we determine Cload[m] for each module m. For example, by examining the ENFE program listings, we find that for ech logical channel, the CPM requires 14 bytes of table space. Terefore, Cload[m] for the CPM is 14 bytes.

In step 6, we compute Cvar[m] as the product of Lmodule[m] ad CLoad[m]. For example, for the CPM we have:

Cvar[CPM] = 88 * 14 = 1232 bytes.

In step 7, we determine Cfixed[m]. For example, by examining te memory maps of the ENFE modules, we find that the CPM reires 12522 bytes, exclusive of the memory required per logical

channel. We note that the modules NCPK and TD are actually implemented as parts of the Unix operating system. To account for the memory used by the operating system, we introduce another module called UNIX. We include the Cfixed[m] for NCPK and TD in the Cfixed[m] for UNIX.

In step 8, we compute Cmodule[m] as the sum of Cvar[m] and Cfixed[m]. For example, for the CPM we have:

Cmodule[CPM] = 1232 + 12522 = 13754 bytes.

We now have the total memory required by each module under the load that is defined by matrix U.

The results of steps 4 through 8 are shown in Figure 6.

Module m	Lmodule[m]	Cload[m]	Cvar[m]	Cfixed[m]	Cmodule[m]
CPM	88	14	1232	12522	13754
HHS	31	32	992	10982	11974
NCPD	80	58	4640	17864	22504
NCPK	160	202	32320	0	32320
SVTS	37	56	2072	13162	15234
UTH	20	1294	25880	2974	28854
UTT	15	3976	59640	7183	66823
TD	38	192	7296	0	7296
PA	20	32	640	13522	14162
UNIX	0	0	0	71452	71452

Figure 6.

Finally, in step 9, we compute Creq as the sum of the Cmodule[m]. We find that

Creq = 284373 bytes.

For a PDP-11/70

Climit = 2**22 = 4194304 bytes.

Hence we have, for the load defined by matrix U,

$$\text{Creq} < \text{Climit}.$$

All of the foregoing deals with the quantitative aspects of deciding whether an NFE can support a multi-host configuration. But there are also qualitative aspects to this question. These qualitative aspects deal with the structure of the NFE hardware and software and with the structure of the protocols that the NFE uses. To facilitate our discussion, we will consider the ENFE and the protocols that it uses.

We first consider the structure of the ENFE hardware. The hardware basis of the ENFE is the PDP-11/70 computer. A multi-host configuration will require the addition of ABSI's to the ENFE hardware. The UNIBUS structure of the PDP-11/70 makes the addition of ABSI's relatively easy.

We next consider the structure of the ENFE software. The software must take into account the existence of multiple ABSI's and of multiple hosts connected through them. There are two effects which must be considered:

1) the effect on the Unix operating system, and

2) the effect on the CPM and on the service modules.

The existence of multiple ABSI's presents no problem in the Unix operating system. The structure of the I/O system permits multiple devices of the same kind to be driven by a single re-entrant device driver without confusion. It will permit the CPM to determine on which ABSI a given message arrived. Therefore the CPM will be able to determine from which host the message was

received. It will also permit the CPM to direct a message to the proper ABSI, hence to the proper host.

The existence of multiple ABSI's, and of multiple hosts connected through them, may have some effect on the structure of the CPM and of the service modules. This effect can be very small if a minor change is made to the HFP. We first discuss the problem.

The current version of the CPM and of the service modules uses logical channel numbers to identify the separate logical communications channels between the CPM and the service modules. These are the same logical channel numbers that are used in communication between the CPM in the host and the CPM in the ENFE. If there are multiple hosts, and if no change is made in the current policy for assigning logical channel numbers, confusion will result in the ENFE when two or more hosts use the same logical channel number.

One solution is to add a host field to the state tables in the CPM and in the service modules. A host field would also have to be added to all communications between the CPM and the service modules. This host field would be used to distinguish logical channels with the same logical channel numbers but from different hosts. This solution would require substantial alteration of the CPM and the service modules.

Another solution is to allot to each of the hosts a disjoint subset of the logical channel name space. This would require that the CPM check the logical channel number as it receives each

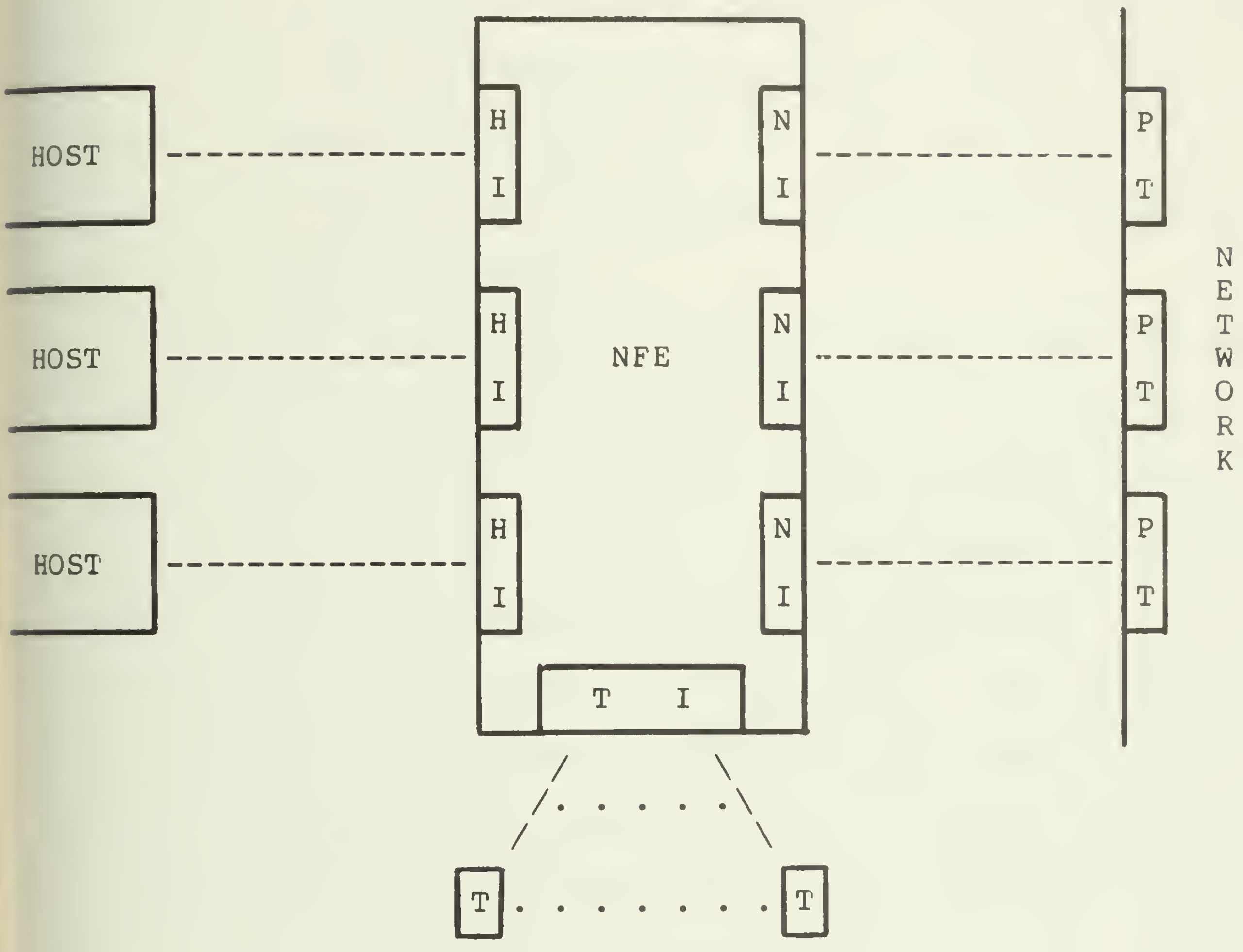
message. This would ensure that the logical channel number in the message matches the host from which it was received. The CPM would also have to use the logical channel number to direct each outgoing message to the proper ABSI. This solution requires relatively minor changes to the CPM. It requires no change at all to the service modules.

We last of all consider the structure of the protocols that the ENFE uses. Two protocols could affect, or could be affected by, a multi-host configuration: the HFP and the ARPANET Host-Host Protocol.

The only change which might be required in the HFP is in the way that logical channel numbers are assigned. Currently the (single) host may attempt to establish a logical channel using any 28-bit number whose high-order bit is zero. In a multi-host configuration, additional high-order bits could be used for identifying which logical channels belong to which hosts. This is not a significant change to the structure of the HFP or to its implementation.

The ARPANET Host-Host Protocol assumes a one-to-one correspondence between network ports, hosts, and NCP's. This means that confusion could result if a multi-host configuration used a single network interface as shown in Figure 2. It might be necessary to have a network interface for each host as shown in Figure 7. This situation should be avoided in the design of future networks such as AUTODIN II.

Multi-Host NFE Configuration
(Multiple Network Interfaces)



where:

HI	=	Interface to the host
NI	=	Interface to the network
TI	=	Interfaces to terminals
PT	=	Port
T	=	Terminal

Figure 7.

OFFLOADING STRATEGIES

In CAC Document No. 230, "Offloading ARPANET Protocols to a Front End," we presented a broad survey of offloading strategies for the most important ARPA Network protocols. This survey should be useful as a basis both for the future design of expanded front ends and for quantitative studies to determine optimal offloading strategies.

Offloading the Telnet Protocol. We discussed in detail the trade-offs involved in different degrees of offloading and provided a brief analysis of the potential for offloading each of the Telnet options. We also discussed which process-to-service protocols were needed to implement the various schemes. The symmetry of this protocol allowed us to develop a new process-to-service protocol (the network virtual terminal PSP) designed to efficiently implement an intermediate level of Telnet offloading. (The maximum offloading strategy adopted for the ENFE was implemented with two separate PSP's - one for the user side and one for the server side).

Offloading the File Transfer Protocol. We identified two major aspects of the File Transfer Protocol (FTP) that were candidates for offloading: the data transfer process and the bookkeeping and marker handling required for restarting a transfer that has aborted. Since FTP makes use of the Telnet protocol, the Telnet functions can also be offloaded. Considering only the User FTP, we found that there are eight possible offloading schemes that differ from one another in major ways.

With regard to the offloading of Server FTP, we confined

ourselves to examining the individual FTP Commands, classifying them as to whether they must be handled in the host or can be handled in the front end.

To make specific the schemes for offloading FTP, we designed three new process-to-service protocols: a User FTP PSP, a Server FTP PSP, and a File Access PSP. The latter provides a general facility for transferring files between host and front end.

Offloading Other ARPANET Protocols. Although our major contractual obligation was to study the offloading of FTP (and Telnet as a natural conjunct of this), we also looked briefly at three other protocols: Remote Job Entry (RJE), Teleconferencing, and Network Graphics.

ALTERNATIVE ARCHITECTURES

Alternative Architecture Research Plan

We developed a research plan (CAC Document No. 232) leading to the specification of a network front end (NFE) designed to meet WWMCCS needs through the 1980's. In CAC Document 232, we briefly reviewed the current state of the art as it affects NFE development, identified some promising directions for research, and presented a detailed plan for conducting the research.

State of the Art. We concluded that the NFE must be modular, efficient, and multi-level secure if it is to meet WWMCCS needs. Three groups of systems were considered relevant to NFE development:

1. existing network access systems,
2. secure systems, and
3. high-bandwidth communications systems.

Examination of these three groups revealed that there does not exist, nor will there exist in the near future, a system which will meet WWMCCS needs.

Research Directions. We presented some promising ideas for research which might lead to solutions to NFE problems. Two alternative hardware architectures were presented for solving the bandwidth problem. An alternative software architecture, the Hub System, was presented which may solve the problems of producing a modular, efficient, and multi-level secure system.

Research Plan. We presented a research plan with four phases:

1. the preparation phase,

2. the research phase,

3. the prototype phase, and

4. the specification phase.

The preparation phase will produce a set of technical constraints for the design of the WWMCCS NFE and will select a set of design features to be studied in the research phase. The selection will be performed through mathematical modeling using the technical constraints. The research phase will design and construct a Research NFE that will be used for evaluating architectural concepts through an iterative process of implementation, testing, and measurement. The prototype phase will design and construct a Prototype NFE which will serve as the basis for specifying the WWMCCS NFE. The specification phase will develop the WWMCCS NFE specfication.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
REPORT NUMBER CAC Document Number 240 CCTC-WAD Document Number 7518	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
TITLE (and Subtitle) Networking Research in Front Ending - Final Report		5. TYPE OF REPORT & PERIOD COVERED Research
		6. PERFORMING ORG. REPORT NUMBER CAC #240
AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s) DCA100-76-C-0088
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
CONTROLLING OFFICE NAME AND ADDRESS Command and Control Technical Center WWMCCS ADP Directorate, 11440 Isaac Newton Sq., N Reston, Virginia 22090		12. REPORT DATE September 30, 1977
		13. NUMBER OF PAGES 44
MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
DISTRIBUTION STATEMENT (of this Report) Copies may be requested from the address given in (11) above.		
DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) No restriction on distribution.		
SUPPLEMENTARY NOTES None.		
KEY WORDS (Continue on reverse side if necessary and identify by block number) Network front end Network protocol		
ABSTRACT (Continue on reverse side if necessary and identify by block number) The CAC has been engaged in an investigation of the benefits to be gained by employing a network front end. A DEC PDP-11/70 was used as front end for connecting a Honeywell 6000 host to the ARPANET. All work (except the multi-host study) performed under the contract has already been thoroughly documented. Thus, this final report abstracts those reports produced.		

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)