PROJECT MAC

January 11, 1974

Computer Systems Research Division

Request for Comments No.45

NETWORK USER'S SUPPLEMENT (NUS) TO THE MPM

by Michael Padlipsky

This RFC comprises the first installment of first drafts of the Network User's Supplement (NUS) to the MPM. As the material is in first draft form, comments on information thought to be lacking would be appreciated - in addition to comments on the material present.

Length estimates of forthcoming drafts are given in the Table of Contents. Note that some of the usage writings (e.g. "net_mail" and "neted") have already been issued as RFC's or MTB's. Chapters 3, 4, and 5 will probably be issued as separate RFC's, as I complete the constituent drafts.

Note that it is intended in the next iteration to strengthen the distinction between issues relating to the ARPA Network per se and those relating to potential other, ARPA-like networks to which Multics machines may be attached in the future. Aid in spotting areas where this distinction needs to be made would be appreciated.

DRAFT TABLE OF CONTENTS

Network Users' Supplement

to the

Multics Programmers' Manual

Foreword: Plan of the Manual

The structure of this Supplement differs somewhat from that of the Multics Programmers' Manual in that it is organized strictly along functional lines rather than partly so and partly alphabetically by command/subroutine name. This approach was chosen in order to focus on and clarify the context of the actual commands and subroutines involved in dealing with the ARPA Network on Multics. Context is particularly important in the Network area, because - unlike the case when dealing with a general purpose time-sharing system, even one as sophisticated as Multics - there is little or no "common knowledge" which the reader may be assumed to have in regard to how things ought to be, for the area itself is brand new. Thus, to take an everyday example, the description of an intra-system "mail" command need presuppose knowledge only of file creation and "addressing" (recipient naming) conventions. However, to use an inter-system mail command, the reader must also know enough about the existence of "Network Control Programs" on each system, possibly alien addressing conventions, and even the Network protocol for sending "mail" that he can recognize error messages as to their context and either try again immediately, try again "later", or contact a Network systems programmer.

This need for context should not be interpreted by the reader as implying that the Network is abnormally difficult to use. Rather, it should be understood that (except for completely rote matters) some knowledge of the underlying mechanisms' nature will be quite useful in terms of preventing wasted effort and aiding the development of new applications. Note that we are by no means suggesting that detailed knowledge of Network protocols is a prerequisite of Network use. The point is that some knowledge of general Network mechanisms will prove to be an asset in determining "what went wrong where".

Of course, readers who wish to turn directly to the writeups of such tools as the Network mail command or the File Transfer command are at liberty to look them up in the Index and do so. However, for the benefit of those who are interested in the Network in the abstract - as well as those who prefer not to use tools by rote - this manual begins with a brief discussion of the history and structure of the Network, followed by a discussion of Network "protocols" which, while not highly technical, serves to introduce several concepts appealed to later and places the particular tools offered into perspective. (The introductory chapter also includes a short bibliography.) The next chapter offers brief overviews of the various portions of system software which make up the supported Network implementation on Multics; this chapter should be of particular interest to subsystem developers. The third chapter addresses itself to material of interest to users who access Multics from the Network; it does not attempt to serve as a replacement for the main body of the MPM, nor even as a summary; instead, it documents those issues which must be taken into consideration by Network users in addition to those addressed in the MPM. Chapter 4, on the other hand, deals with use of the Network from Multics; here, some familiarity with the material dealt with in Chapter 1 is probably necessary. The fifth chapter is of interest almost exclusively to those who are developing subsystems which use the Network from Multics, giving as it does calling sequence information on the primitives of the Multics Network implementation; it should be read in conjunction with Chapter 2. Finally, the sixth chapter (which will not be furnished in the initial release of the manual) will offer students of systems a view of the design of the Multics "Network Control Program".

History and Structure of the ARPA Network*

Introduction

[Early in 1969,] the Advanced Research Projects Agency of the Department of Defense (ARPA) began implementation of an entirely new venture in computer communications: a network that would allow for the interconnection, via common-carrier circuits, of dissimilar computers at widely separated, ARPA-sponsored research centers. This network, which has come to be known as the ARPA Network, presently includes approximately [40] nodes and is steadily growing. Major goals of the Network are (1) to permit resource sharing, whereby persons and programs at one research center may access data and interactively use programs that exist and run in other computers of the network, (2) to develop highly reliable and economic digital communications, and (3) to permit broad access to unique and powerful facilities which may be economically feasible only when widely shared.

The ARPA Network is a new kind of digital communication system employing wideband leased lines and message switching, wherein a path is not established in advance and instead each message carries an address. Messages normally traverse several nodes in going from source to destination, and the etwork is a store-and-forward system wherein, at each node, a copy of the message is stored until it is safely received at the following node. At each node a small processor (an Interface Message Processor, or IMP) acts as a nodal switching unit and also interconnects the research computer centers, or Hosts, with the high band-width leased lines. [(The development of the "IMP sub-net" was performed by Bolt, Beranek, and Newman Inc.)]...

*Adapted from Ornstein, S.M., F.E. Heart, et al. "The Terminal IMP for the ARPA Computer Network", Proceedingsof the 1972 SJCC, p. 243 ff., with permission.

The Developing Network

The initial installation of the ARPA Network, in 1969, consisted of four nodes in the western part of the United States. A geographic map of [a recent configuration of the] ARPA Network is shown in Figure 1. Clearly, the most obvious development has been a substantial growth, which has transformed the initial limited experiment into a national assemblage of computer resources and user communities. The Network has engendered considerable enthusiasm on the part of the participants, and it is increasingly apparent that the Network represents a major new direction in both computer and communications technology.

Figure 2 is a logical map of the Network, where the Host computer facilities are shown in ovals, all circuits are 50 kilobits, and dotted circuits/nodes represent planned installations. On this figure certain nodes are listed as a "316 IMP"; this machine is logically nearly identical to the original IMP, but can handle approximately two-thirds of the communication traffic bandwidth at a cost savings of approximately one-half. The original IMP includes a Honeywell 516 computer, and more recently Honeywell began to market the 316 computer as a cheaper, downward-compatible machine. As the Network has grown, sites were identified which did not require the full bandwidth of the original IMP, and a decision was made to provide an IMP version built around the 316 computer. Also shown in Figure 2 are certain nodes listed as "TIP"; this new machine is discussed [below. (Site abbreviations shown on Figures 1 and 2 are explained in Section 4.1.)]

As the Network has grown, a great deal of work has been concentrated on the development of Host-to-Host protocol procedures. In order for programs within one Host computer system to communicate with programs in other Hosts,

agreed-upon procedures and formats must be established throughout the Network. This problem has, as predicted, turned out to be a difficult one. Nonetheless protocol procedures have evolved and are being accepted and implemented throughout the Net. At the present writing, many of the Hosts have working "Network Control Programs" which implement this protocol. Protocol development is more fully reported [elsewhere], but we wish to make a general observation on this subject: the growth of the Network has dynamically catalyzed an area of computer science which has to do with the quite general problem of how programs should intercommunicate, whether in a single computer or between computers. Thus the evolution of the Host-to-Host protocol represents a side benefit of the Network that reaches well beyond its utility to the Network alone. [(See Section 1.2 for further discussion of the Host-to-Host Protocol, and "higher-level" protocols based upon it.)]

Since both hardware and software Network connections must be implemented by each Host, it is important that the external characteristics of the IMP be relatively stable. This stability has been carefully maintained, while at the same time internal operation of the IMP program has undergone extensive revision and improvement. For example, trouble reporting, statistics gathering, and test procedures have been substantially improved. In addition to improvements that have already been incorporated into the program, there have also been extensive studies of performance and message flow control. These studies have pointed up areas of vulnerability to perverse heavy traffic patterns and have suggested still other possible improvements in the routing and flow control mechanisms....

Somewhat belatedly in the Network design, the need to connect a single

IMP to several Hosts was recognized. This required multiple Host interfaces at the IMP as well as more complex IMP software. Further, the various Host computers at a site are often physically distant from one another, thus requiring an increase in the maximum allowable physical separation between a Host and its IMP. To connect to an arbitrarily remote Host would have meant a communications interface capable of attachment to common-carrier circuits via modems. It would furthermore have required cooperative error control from the Host end. At the time, BBN chose not to modify the logical way in which the IMP dealt with the Host and instead provided more sophisticated line drivers which would handle distances of up to two thousand feet. Several such "Distant Host" installations are now working in the network. Unfortunately, as the Network has grown, new sites have appeared where still greater Host/IMP distances are involved. The [Distant Host] scheme does not include error control, and use of this scheme over greater distances is not appropriate. [BBN has since specified a "Very Distant Host" interface to meet these objections.]

Another facility which has been tested is the ability of the IMPs to communicate over 230.4 kilobit phone lines instead of 50 kilobit lines. A short, fast link was incorporated into the Network for a brief period and no problems were encountered. To date, Network loading has not justified upgrading any operational Network circuits to 230.4 kilobits, but this will be considered as loading rises.

Substantial effort has gone into traffic and trouble reporting. A Network Control Center (NCC) has been built at Bolt Beranek and Newman Inc. in Cambridge, where a small dedicated Host computer receives reports each minute from every IMP on the Network. Traffic summaries and status and trouble reports are then generated from this material....

[Modes of] Access

During the early phases of Network development a typical node consisted of one or more large time-shared computer systems connected to an IMP. The IMPs at the various sites are connected together into a subnet by 50 kilobit phone lines and the large Host computers communicate with one another through this subnet. This arrangement provides a means for sharing resources between such interconnected centers, each site potentially acting both as a user and as a provider of resources [("Server")]. This total complex of facilities constitutes a nationwide resource which could be made available to users who have no special facilities of their own to contribute to the resource pool. Such a user might be at a site either with no Host computer or where the existing computer might not be a terminal-oriented time-sharing system.

A great deal of thought went into considering how best to provide for direct terminal access to the Network. One possibility, which would have essentially been a non-solution, was to require a user to dial direct to the appropriate Host. Once connected he could, of course, take advantage of the fact that that Host was tied to other Hosts in the Net; however, the Network lines would not have been used to facilitate his initial connection, and such an arrangement limits the terminal bandwidth to what may be available on the switched common-carrier networks.

A similar solution was to allow terminals to access the Network through a Host at a nearby node. In such a case, for example, a worker in the New England area wishing to use facilities at a California site might connect into a local Boston Host and use that Host as a tap into the Network to get at the facilities in California. This approach would have required Hosts to provide hardware access facilities for many terminals which would use their systems only in passing. For many Hosts, the kinds of terminals which can be connected

directly are limited in speed, character set, etc. In terms of reliability, the user would have been dependent on the local Host for access: when that Host was down, his port into the Network would be unavailable. Furthermore, the Hosts would have been confronted with all of the problems of composing terminal characters into Network messages and vice versa as well as directing these messages to the proper terminals and remote Hosts. Time-sharing systems are generally already overburdened with processing of characters to and from terminals and many are configured with front end processors employed explicitly to off-load this burden from the main processor. Increasing the amount of such work for the Hosts therefore seemed unreasonable and would have resulted in limiting terminal access. Instead, a completely separate means for accessing the Network directly seemed called for: an IMP with a flexible terminal handling capability - a Terminal IMP, or TIP.*

The TIP, as developed by BBN, comprises an IMP and a "Multi-line Controller", with additional memory in the IMP to allow for the code which controls the various types of terminals attached via the MLC, and which performs the various functions dictated by Network protocols. It does not, however, go beyond the terminal support level to such areas as furnishing local file storage. To further bridge the gap between the terminal and the full-spectrum Host, the Center for Advanced Computation of the University of Illinois has developed the ARPA Network Terminal Support system (ANTS). ANTS comprises an IMP and a PDP-11; thus, it is able to offer more computing power to those users who require more amenities than the TIP furnishes, while still running to a full-scale Host.

The Network, then, may be viewed as a collection of "Server" Hosts and "User" Hosts, where the latter may range from simple terminal support systems, through more sophisticated terminal support systems, to other large time-sharing

* End of adapted material.

systems which also play Server roles. All of these Hosts communicate in terms of various protocols which have been developed by the Network Working Group, a loose confederacy of systems programmers and designers from the several Hosts. Section 1.2 offers a closer look at Network protocols, for those interested.

(Figure 1 will go here)

(Figure 2 will go here)

"Levels" of Protocol

Introduction

To the user, perhaps the most striking feature of the ARPA Network is the scope of the Server Hosts it makes available. This heterogeneity of operating systems runs the gamut from such one of a kind early time-sharing systems as the TX2 at M.I.T.'s Lincoln Laboratory to such one of a kind recent systems as the ILLIAC IV at NASA's Ames research facility; among others, it includes Multics, TENEX, OS/VS/TSO, TSS/360, and DEC System 10/50, running on Honeywell 6180s, DEC PDP10s, and IBM 360s and 370s. Each of these Servers has, of course, its own native conventions and preconceptions. To allow them to communicate, it has been necessary to evolve a series of common conventions - or "protocols" - which each uses for Network transactions. Figure 1 offers an abstract metaphoric attempt to place these protocols in context, and the balance of this Section expands upon it to the extent necessary to allow the reader to appreciate what underlies his use of the Network. (To become an expert on the protocols, deeper study is, of course, required; see the Bibliography in Section 1.3.)

IMP-IMP Level

Naturally, none of the Host to Host communication could proceed were there not an IMP subnet. Although the subnet - discussed in Section 1.1 - is a "given" even to those system programmers who implement Network Control Programs (NCP's), it should be noted for present purposes that the IMPs communicate with on another over leased lines in general (although satellite links also come into play) in terms of "packets" of bits. These packets are approximately 1000 bits long, and up to eight packets may comprise a single "message" from the Hosts' point of view. The IMP subnet accepts messages from Hosts, breaks them into packets if necessary, and routes the packets to the intended desination IMP by

passing them to whichever neighboring IMP is instantaneously determined to be "best" (provided there is more than one neighboring IMP, of course). At the destination IMP (determined by the contents of a particular field in the "leader" of the message) packets are reassembled as necessary and when complete are passed on to the Host.

IMP-Host Level

From the Host's point of view, the IMP is connected to it as an I/O device; physically, the Host's channel or port connects to a "special interface" (for which Host personnel are responsible), which in turn connects to an IMP port via a cable. (On the ARPA Network, personnel of BBN's Network Control Center are responsible for the IMP and its port.) Each message the Host sends to the IMP for Network transmission begins with a "leader" which contains the destination and a "link" number. When the IMP delivers the message, the link number is preserved for use in the Host-to-Host protocol, but the destination field is replaced by the address of the originating Host. For such control functions as the Host or the IMP's announcing that it has come up or is about to go down, a "for-Host" or "for-IMP" bit is set in the leader and an appropriate "type" field conveys the particular information desired.

It should be noted that the disposition of the message is reflected by the IMP to the Host in one of two ways: If the message was delivered successfully, a "Request For Next Message" (RFNM) type IMP-Host message is sent for that link in question. If the message was not delivered (perhaps because the destination Host or IMP was down or because the message was "lost" in the subnet), an "Incomplete Transmission" type message is sent, with a subfield giving more detailed information (so that the Host can retransmit the message if it has preserved a copy and if the circumstances require it). Except for messages routed

via high-altitude satellite transmission paths, transmission time is normally under .5 second, although confirmation may take considerably longer as the receiving IMP must allow sufficient time for its Host to accept the message.

Host-Host Level

To allow processes on given Hosts to communicate with one another, a Host-to-Host protocol must exist. For historical reasons, the link field of the Host-IMP leader did not offer a rich enough address space. Therefore, the Host-to-Host protocol establishes a "socket" space, and each Host's Network Control Program (NCP) is responsible for allowing the association of its processes with socket numbers. NCPs communicate with one another in order to open, close, and control data flow over sockets. The NCP transactions are transmitted over a distinguished link (called the "control link"), and follow a prescribed format as to operation codes ("commands"), socket fields, and link fields (the latter being necessary in the opening of a read socket relative to the receiving Host). Given the means for opening a read-write socket pair, processes may then communicate as they see fit, although certain protocols have been prescribed governing the communication between special types of processes.

Note that data flow is controlled by the receiving Host, as it knows best what sort of buffering capacity it can furnish. The NCP discipline is that the sending Host is only permitted to send as many bytes (of a size established when the socket connection is opened) as have been allocated to it by the receiving Host via the appropriate NCP command ("ALL") for the socket in question. Thus, when one is logged in to a "Server" Host from a "User" Host, the rate at which the user actually sees data displayed on his terminal is primarily a function of the NCP on the "User Host (which is the Host which has physical control over his terminal), not of the "Server" Host (which is the Host on which the computation of interest is running).

User-User Level

With the ability to establish a socket connection between processes on Network Hosts, the field is clear for furnishing really interesting abilities to users. For we can then come up with protocols for causing particular types of processes to be created on our behalf on Server Hosts, and become directly logged-in users, or have files transmitted from one Host to the other, or appear to be remote job entry terminals, or the like. Fundamental to all these "user level" protocols is the ability to establish communications with "the right sort of process"; and just as the NCP's are defined to be what is at the other end of a distinguished link, various special-purpose processes are defined to be at the other end of certain distinguished sockets. The general Initial Connection Protocol (ICP), then, is a series of NCP commands and definitions of sockets such that a request for connection to the defined socket will result in the transmission over that connection of another socket number which in turn will become the base of an even-odd (read-write) socket pair over which the desired function can be performed. (If that's a bit cryptic, consider the fact that it is each system's NCP which must "know" about the association of sockets with its processes; therefore, it must be free to specify the actual socket numbers to use, depending on the sort of process it is dealing with.)

Naturally enough, the first user-level protocol to be defined was the one which allows direct login to the foreign system. An ICP to socket 1, then, is defined as placing the user's process in communication with the Server Hosts' "logger" process, which will cause the creation of a process which operates according to the dictates of the "Telnet Protocol" (from telecomunications network). Telnet is the dominant user-level protocol, both in terms of frequency of use and importance to other user-level protocols (which are in general compatible with it, and in some sense embedded in it). Following the same reasoning as detailed in Section 1.1 in support of the notion that small terminal-support Hosts should exist rather than forcing Servers to control terminals directly, the

Telnet Protocol avoids the problem of requiring each of n Servers to be able to control all of m terminal types (the so-called "n by m problem") by defining a "Network Virtual Terminal" (NVT) - a common intermediate representation into which the User Host will translate the physical terminal input relative to itself, and from which the Server Host will translate into the form expected by its system. (The converse translation process occurs on output from the Server Host via the User Host to the terminal, of course.) Thus, such generic functions as "interrupt the (Server) process running on my behalf" may be assigned Telnet control codes (defined byte sequences "outside" the character set), relieving the user of the necessity of knowing whether the foreign Server actually looks for a line condition ("Break" or "Attention") or a specific "control character" internally. Rather he learns the characteristics of the "User Telnet" at his disposal (whether via a TIP, an ANTS, or a full-scale Host) and uses them for dealing with all Servers. See Figure 2.

Another function of interest provided by the Telnet Protocol is the ability to negotiate "options" between the User and Server sides of the Telnet Protocol implementation to control such factors as line-length, echo mode, and the like. The ability to alter the NVT default assumptions in such areas is particularly useful in view of the fact that the NVT takes a "least common denomination" view of terminals, and this mechanism allows users at relatively powerful terminals to get their money's worth, provided the program they are communicating with is prepared to cooperate. As might be expected, Server Hosts and User Hosts vary wide in terms of which particular negotiated options they support. (See Sections 3.8 and 4.1 for Multics assumptions.)

The existing protocol for File Transfer (FTP) is rather more complex than the Telnet Protocol, for it does not avail itself of the view that there should be a "Network Virtual File". Rather, it acknowledges the fact that many of the Hosts on the Network structure files rigidly and provides means for describing

such structures as well as transferring the relevant bits. A default file is a string of ASCII characters, however; hence, the ability to send "mail" from system to system is a natural outgrowth of the FTP. As with the Telnet Protocol, "User" side implementations are a per-Host option in the FTP area (see Section 4.4 for a description of "net_mail", and 4.2 for the Multics "User FTP"), while the "Server FTP's" must, of necessity, present a uniform interface. Uniform naming and accounting conventions have yet to be developed for mail, but the facility is quite widely used nonetheless.

Other user-level protocols on which there is a fair degree of consensus are a Remote Job Entry protocol and a Graphics protocol. Still under development are a "File Access Protocol" (which allows selective referencing of foreign files, in a manner akin to the Multics I/O switch, as opposed to the in toto transfers of the FTP), a "Unified User-Level Protocol" (which defines a common command subset for the performing of generic functions on the various Servers, allowing all user-level protocols to be invoked via Telnet connections and allowing invocation of foreign functions by program), and various attempts to specify general frameworks for system-to-system and process-to-process communication.

Although it might not merit being called a protocol, there is also a common context editor command on most Servers. Part of the Unified User-Level Protocol effort, the command is known as "neted", and is documented in Section 3.4.

"LEVELS" of Protocol

LEVEL	TALKS OVER	TALKS IN
IMP-IMP	Leased lines	Packets
IMP-Host	Cable/SPIF/Channel	links
Host-Host	Control links	NCP commands
User-User	Sockets	Bytes

USER-LEVEL Protocols

"OFFICIAL"	IN PROGRESS
• TELNET/ICP	• FILE ACCESS
• FILE XFER/"MAIL"	• PROCESS-PROCESS
• RJE	• UNIFIED USER-LEVEL PROTOCOL
• GRAPHICS	

Figure 1

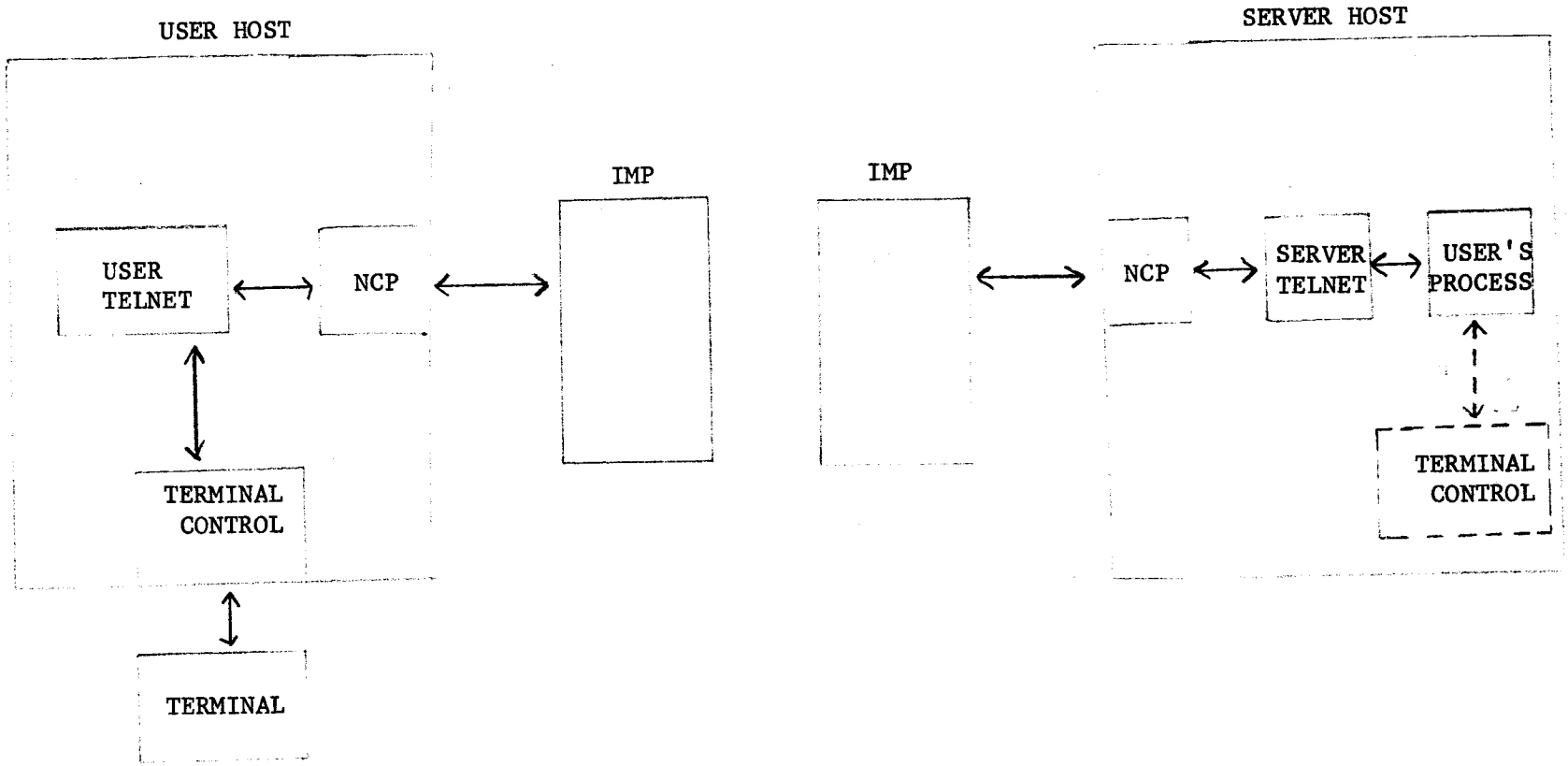

Figure 2. User and Server Functions of the Telnet Protocol

Note that the broken lines around "Terminal Control" in the Server Host are meant to suggest that the Server Telnet has replaced the local terminal in the process operating on the Server Host on behalf of the user.

BIBLIOGRAPHY

Listed here are a few of the basic bibliographical items about the ARPA Network. Note that the material indicated as "available from" some Network source is primarily intended to be made available to organizations which are associated with the Network rather than to individual users of Network Hosts. The latter usually refer to copies of the information involved which are held by their local Network personnel. In certain circumstances, however, individual users may wish to purchase their own copies if they are not clearly eligible to receive them on a supported basis.

The Network Information Center (NIC) is funded by ARPA; it distributes Network Working Group "Requests for Comments" and the memoranda and notes of various NWG interest groups as well as material indicated herein.

A. Network New User's Packet
Available from: Mr. Jean Iseli
MITRE Corp.
1820 Dolly Madison Blvd.
Westgate Research Park
McClean, Va. 22101

B. Papers

In the AFIPS Conference Proceedings, Spring Joint Computer Conference 1970 (pp. 543 ff.):

1. L.G. Roberts, B.D. Wessler, "Computer Network Developments to Achieve Resource Sharing".

2. F.E. Heart, R.E. Kahn, S.M. Ornstein, "The Interface Message Processor for the ARPA Computer Network".

3. L. Kleinrock, "Analytic and Simulation Methods in Computer Network Design".

4. H. Frank, I.T. Frisch, W. Chou, "Topological Consideration in the Design of the ARPA Computer Network".

5. C.S. Carr, S.D. Crocker, V.G. Cerf, "Host-Host Communication Protocol in the ARPA Network".

In the AFIPS Conference Proceedings, Spring Joint Computer Conference 1972 (pp. 243 ff.):

1. S.M. Ornstein, F.E. Heart, et al., "The Terminal IMP for the ARPA Computer Network".

2. H. Frank, et al., "Computer Communication Network Design - Experience with Theory and Practice".

3. S.D. Crocker, et al., "Function-Oriented Protocols for the ARPA Computer Network".

4. R.H. Thomas, D.A. Henderson, "McRoss - A Multi-Computer Programming System".

5. L.G. Roberts, "Extension of Packet Communication Technology to a Hand-Held Personal Terminal".

C. Protocols Notebook
D. Resource Notebook

} One copy per Network site available through:

Network Information Center
Stanford Research Institute
333 Ravenswood Ave.
Menlo Park, Ca. 90

Overview of the Multics Network Attachment

The sections comprising this chapter deal with various portions of the supported software on Multics which implement the Multics attachment to the Network. This overview section is provided to place them in perspective one to another.

The IMP as I/O device is managed by a "ring 0" (hardcore supervisor) module called the IMP DIM (Device Interface Module). The IMP DIM performs a small number of Host-to-Host protocol functions, in addition to its major role as performer of the Host-IMP protocol. Another ring 0 module, called the NCP, is responsible for the balance of the Host-to-Host protocol functions. The Multics "logger" is a user-ring module in the Answering Service process; the same module also services Initial Connection Protocol contacts on the File Transfer Protocol (FTP) Server socket. The Server functions of the Telnet Protocol are performed by a module which replaces the local teletypewriter "DIM" in user processes and supports the same generic function calls on the user I/O streams as does the local version. The User functions of the Telnet Protocol are embodied in a free-standing command, as are the User functions of the File Transfer Protocol. The Server functions of the File Transfer Protocol are performed by a "login responder"/"process overseer" mechanism to which control is passed by the logger on FTP socket ICPs.

There are two methods for performing non-user-level-protocol operations and implementing new protocols: I/O system interface modules ("IOSIMs") and direct calls to the NCP primitives. Both usually hinge on the ability to perform the Initial Connection Protocol, for which a subroutine is also furnished (net_connect_). As dealt with in more detail in 4.0, "outgoing" access to the Net from Multics is administratively controlled.

The IMP DIM

Part of the hardcore supervisor, the IMP DIM (Device Interface Module) primarily exists in order to perform the Host-IMP. That is, it contains the code which manages the "leader" portion of IMP messages, issues I/O connect instructions for the full-duplex "Asynchronous Bit-Serial Interface" which connects the IMP to a pair of H6180 IOM common peripheral channels, enforces the discipline that no further messages are to be transmitted on a given "link" until the IMP subnet has furnished a "RFNM", and manages the retransmission of messages when needed.

It should be noted that managing the leader has rather deep implications. Not only does this management cover the proper assignment of link numbers on messages to be written, it also entails the association of link numbers with socket numbers (and hence with processes) on messages which are read. A natural extension of such functions which was also a practical necessity is the IMP DIM's role in enforcing the Host-to-Host Protocol flow control discipline, particularly in processing the NCP "ALL" command. This role avoids the need to "wakeup" the Network Daemon (that process in which most types of the Host-Host commands are processed) every time a small recipient Host is willing to allow a few more characters to be transmitted.

The IMP DIM's buffering strategy on input is such that Multics permits quite large allocations (in the NCP sense), by means of placing input on receipt from the read channel of the ABSI as rapidly as possible into pageable buffers managed by IMP DIM code which runs at "call time" in the user's process. (The actual servicing of ABSI interrupts is, of course, performed at

"interrupt time" and involves wired down buffers.) The buffering strategy for writing also involves a mixture of pagable and wired buffers, although at the present time the wired buffer strategy does not utilize maximum IMP message-size buffers as does the read side.

The IMP DIM is callable only from ring 0, which is to say that it is called by the NCP rather than directly by user code.

The NCP and the Network Daemon

There are two contexts in which the Multics Network Control Program must be viewed: as a ring 0 module which interfaces with the IMP DIM on the one hand and the user ring on the other and as a process which exists to guarantee that Host-to-Host Protocol commands will be processed when sent to Multics. The process is called the Network Daemon, and can be dealt with briefly: The IMP DIM wakes up the Network Daemon for all messages on the "control link" other than those which it manages itself at interrupt time. In response to these wakeups, the Daemon calls in to the NCP module in ring 0, where they are duly processed, and the Daemon returns to a "blocked" state. Also, various control functions such as initializing the Network software and turning it off if necessary are performed in the Daemon.

The NCP is rather less straightforward, as it must manage the "socket space" of the Host-to-Host Protocol on behalf of any and all user (and system) processes which deal with the Network on Multics, in addition to processing the bulk of the NCP commands (such as interrupt the process associated with a given socket, reset all table entries associated with a given Host, and the like). At the present level of detail, it is sufficient to observe that the socket management is based on the notion of a socket's being in a particular "state", and when that state changes the NCP directs a wakeup to the process controlling the socket.

User-callable entry points in the NCP are discussed in Section 2.6 and detailed in Section 5.4.

A few rather cryptic points which will be of interest to Network systems programmers at other Hosts would be appropriate to mention here: a) The Multics NCP does not queue requests for connection. The implication of this from the

user's point of view is that on attempts to login to Multics from the Network, a "refused" status usually means that the request for connection happened to arrive at a time when another one was being serviced, and a subsequent login attempt will probably be honored. b) No timeouts are imposed at the NCP level, although such user-level timeouts as the Answering Service's two minutes to attempt a login after "dialup" are of course reflected through. c) The Multics implementation requires that communications over the Net with each Host must be specifically enabled. Thus, a newly attached Network Host will not succeed in communicating with a Multics Host until communications have been administratively enabled on the Multics side.

The Logger

In the present implementation of Multics, the only process which is privileged to create processes is the Answering Service. Thus, in order to fulfill the logger function of the Telnet Protocol and the Server function of the File Transfer Protocol, there must be Network code in the Answering Service Process.

This code services the wakeups sent by the Network Daemon to the Answering Service when requests for connection are received on socket 1 (the Telnet ICP contact socket) or socket 3 (the FTP ICP contact socket). Just as the Answering Service initially has control over all local terminal ports, it initially has control over the virtual ports (actually socket pairs) which Network logins will use. (And instead of calling the local tty DIM for such ports, it calls a Network tty DIM, which furnishes the same generic functions as does the local version.)

When the login is authenticated, control of the socket pair is passed off from the Answering Service to the user process. This fact is significant in the design of Network protocols, as it differs from many other systems' views (in that they have a process per logical port already in existence), and makes it awkward to transmit arguments across the process boundary. For example, in doing Network mail, Multics requires an initial login equivalent rather than immediately accepting the FTP "mail" command both for reasons of accounting and authentication, and because the addressee of the mail is then directly known to the created user process.

As a convenience to users coming in over the Net from upper-case-only terminals via User Hosts which do not furnish case-mapping, a pre-login "map" command is furnished; see Section 3.2.

User Processes

There are two classes of user processes of interest in the Network context: those which are operating on behalf of a local user, and those which are operating on behalf of a remote (or "Network") user. This section will touch on the "supported" aspects of both classes, and the remaining sections of this chapter will introduce the software for specialized applications - primarily for "outgoing" use of the Net.

Coming in to Multics from the Network, the primary environment the user will encounter is that of the "Server Telnet". That is, a direct Network login to Multics, actuated by performing an Initial Connection Protocol (via a "User Telnet" on some Network Host), results in the creation of a process on the user's behalf, just as would a directly dialed local login. The process created for the Network login is identical to that created for the local login except for the presence of a terminal control module which communicates with the Network Control Program rather than with teletypewriter line controlling software in the Multics supervisor. (This approach is facilitated by the design of Multics, which follows the discipline that the system's command processor and all system commands perform I/O on the symbolically-named streams "user_input" and "user_output", with the "attachment" of these streams to particular physical or login devices being accomplished at process creation time.) Thus, the major terminal control module in a Network user's process is a program called "nttydim" as opposed to the local user's "ttydim" - through which the user's I/O streams are attached. ("DIM" - or device interface module - is an earlier term for "IOSIM".) It is in nttydim that the Server aspects of the Telnet protocol are performed. See also Chapter 3.

The File Transfer Server process also employs the Network version of the "ttydim" for I/O. It takes advantage of another feature of the Multics design in order to place the process into the proper environment for performing FTP

"commands", the substitutability of command processors. See also Section 3.4.

For the two most common outgoing uses of the Network, a "User Telnet" (Section 4.1) and a "User FTP" (Section 4.2) command are furnised. It should be noted, however, that the ability to use these commands is not automatically conferred upon all Multics users. The point here is a matter of ARPA policy, in order to prevent small, experimental Server Hosts from being swamped by casually curious users on extremely large Hosts. As the small Servers often do not have much access control, the large Hosts such as Multics which can prevent unlimited use of their User Telnets and FTP's do so. (Permission to make outgoing use of the Net is coordinated by the Multics Network Technical Liason.) Should a locally-registered user happen to find himself at another Network site, however, no restrictions are placed on his "incoming" access to Multics from the Net.

IOSIMS

Not all use of the Network from Multics is for the purpose of logging into or transferring files with another Server Host. To facilitate special-case applications, two I/O System Interface Modules (IOSIMS) are furnished - as well as the NCP primitives discussed in the next section. (The former may be viewed as "packaged" ways of doing process to process communications, and the latter as "unpackaged" ways.)

The IOSIMS differ in that one deals only with ASCII data transmission and the other does not. They have the following features in common: One may cause, at "attach" time, either an initial connection protocol to be perfromed to a foreign Host, or the establishment of a socket connection to a known foreign socket, or the establishment of a socket which will "listen" for connections from a specified Host. Once the stream is attached through a Network IOSIM, normal Multics I/O system generic functions may be performed. In particular, calls exist which govern the interpretation of data transmitted over the connection in such aspects as read delimiter character, "canonicalization", and the like. See also Sections 4.3.1 and 4.3.2.

Users have written programs employing the IOSIMs for such purposes as printing out Multics listings on a TIP's line printer, and communicating with a foreign system which is not a Network Host but which is locally dialed up from a TIP port.

Primitives

For applications where even the IOSIMs make too many implicit assumptions about the nature of the transactions being performed, the discrete entry points of the NCP and several "free-standing" subroutines may be employed by user programs or subsystems. The NCP primitives allow for precise management of sockets, from activation through establishing byte sizes and sending or accepting Host-to-Host protocol "requests for connection" to causing Host-to-Host protocol "closes" to be sent; see Sections 5.0 and 5.4. The subroutines allow for the performing of the Initial Connection Protocol or the sending or accepting of requests for connection (Section 5.1), the management of a process "socket space" (Section 5.2), and the performing of the 9-to-8 and 8-to-9 bit conversions necessary to go to and from Multics' internal representation of ASCII from and to the Network's representation (Section 5.3).

The User Telnet and FTP employ these primitives.