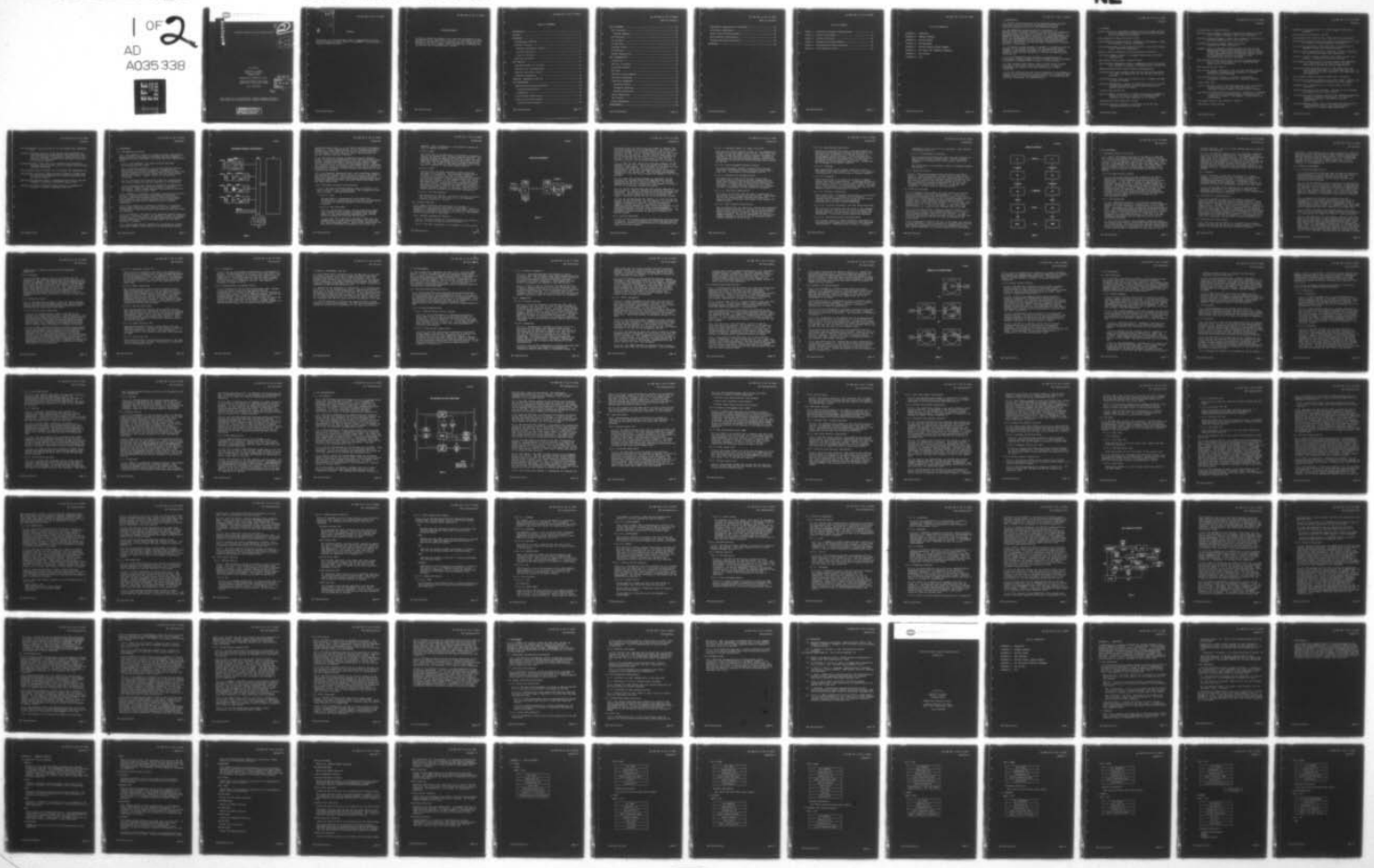
AD-A035 338 STANFORD RESEARCH INST MENLO PARK CALIF AUGMENTATION --ETC F/G 17/2
TERMINAL-TO-HOST PROTOCOL SPECIFICATION.(U)
JUL 76 J B POSTEL, L L GARLICK, R ROM

UNCLASSIFIED SRI-ARC-35940 NL

1 OF 2
AD
A035 338



STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U S A

Terminal-to-Host Protocol Specification

15-July-76

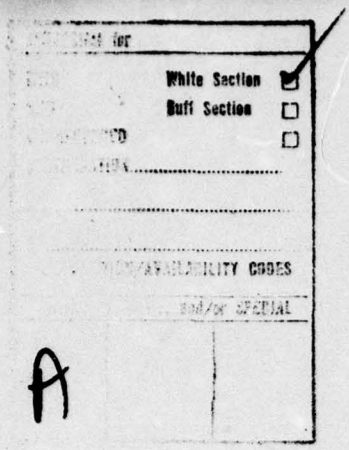
Jonathan B. Postel
Larry L. Garlick
Raphael Rom

Augmentation Research Center

Stanford Research Institute
Menlo Park, California 94025

(415) 326-6200

This report is the deliverable "Terminal-to-Host Protocol Specification" as specified in contract DCA100-76-C-0034.



PREFACE

This report is the deliverable "Type "A" Specification of the Terminal-to-Host Protocol (Final)" for AUTODIN II as specified in contract DCA100-76-C-0034.

ACKNOWLEDGEMENTS

We wish to extend our thanks to all those who contributed to the development of the ARPA Telnet protocol, to James White and David Maynard of SRI-ARC for their significant help in evaluating this document in its final stages, and to Beverly Boli of SRI-ARC for the tedious work of shaping this document.

TABLE OF CONTENTS

Table of Contents

LIST OF FIGURES

LIST OF APPENDICES

1. INTRODUCTION

1.1. This is the specification of the Terminal-to-Host Protocol (THP) which serves as both the terminal-to-terminal and terminal-to-process protocol for the AUTODIN II network.

1.2. The AUTODIN II system provides the capability for geographically distributed computers, called hosts, to communicate with each other. The hosts are a diverse set of computers of differing manufacture, speed, word size, and operating system. The AUTODIN II system provides a mechanism for communication between hosts, the Transmission Control Protocol (TCP) specifies how the hosts use this mechanism to provide communication services to processes, and finally, the THP provides a mechanism for offering services to human users.

1.3. The reader of this document is assumed to be familiar with the concepts of the AUTODIN II system, TCP, and operating system concepts. In particular the reader is expected to have read the AUTODIN II Specification [Reference 1] and the TCP Specification [Reference 2].

1.4. It is intended that the information presented here be sufficiently complete enough to allow a competent system programmer to implement a program module to carry out this protocol.

1.5. This document draws heavily from the ARPA Network protocol documents on the Telnet Protocol [Reference 3]. We wish to acknowledge the efforts of the many contributors to that collection.

1.6. In the following sections a brief background of the AUTODIN II network, and the protocol environment within it, is presented. This introduces the more detailed specification in the sections on THP function, environment, and implementation.

2. GLOSSARY

access circuit--an addressable hardware port on the LCM; the point at which communication lines for local terminal devices, TAC's, and hosts are attached.

ARC--Augmentation Research Center of SRI.

ASCII--American Standard Code for Information Interchange; a seven bit encoding of textual characters.

authorization table--a table used by the TCP to check S/P/T values during connection management.

BASIC--a simple programming system developed at Dartmouth.

BSL--Binary Segment Leader; an S-segment header used for PS-SIP communication.

CCU--Channel Control Unit; an MCCU or SCCU.

CMB--Connection Management Block; a bookkeeping block used by the THP (program) to maintain information about a connection.

EBCDIC--Extended Binary Coded Decimal Interchange Code.

echo-mode--the choice between local echo (at the user's terminal, TC or THP) and remote echo (at the server's THP, TC or process).

EOL--End-of-letter flag; used to indicate that a segment includes the end of a letter.

FED--Front-End Device; a method of connecting a host to a PS with minimum alteration to the host operating system to support protocol functions.

FL--TCP flush control bit.

host--a computer connected to a network that executes programs on behalf of its users but does not necessarily offer services to the other computers on the network.

host-entity--a host, FED, CCU, or TAC.

HSI--Host Specific Interface; a sub-module of the CCU that interfaces the TCP to the host.

ID--identifier or identification.

LCM--Line Control Module; controls communication between a PS and hosts, terminal equipment, TAC's, and remote PS's.

LCN--Local Connection Name; a handle that is given to the user as a shorthand identifier for a TCP connection.

line-mode--the choice between line-at-a-time and character-at-a-time interaction.

LISTEN--a type of connection opening in which some part of the destination address, send precedence, or TCC has been unspecified. This type of open allows a user or process to wait indefinitely until a connection attempt is made by a remote user, and is thus useful for open synchronization and server processes.

MCCU--Multiple Channel Control Unit; allows the interfacing of a host to a PS and provides up to 32 connections with remote hosts or subscribers.

NCC--Network Control Center.

NLS--oN Line System, developed at ARC, for text editing, document production, programming and debugging, and message distribution and cataloging.

NVT--Network Virtual Terminal; a standard, network-wide, intermediate representation of a canonical terminal.

octet--eight bits.

port--a name chosen from a universal name space (not necessarily unique, however) that identifies the stream of information passed between a process and the network.

precedence--an indication of the urgency of information; in AUTODIN II there are 16 precedence levels, divided into four categories. Category I, the highest, is specified to be non-blocking.

PS--Packet Switch of the AUTODIN II network.

PSN--packet switch network.

S-segment--the concatenation of a Binary Segment Leader and a T-segment.

S/P/T--Security, precedence, and TCC.

scanning-mode--the choice between record mode and stream mode.

SCCU--Single Channel Control Unit; allows the interfacing of a host to a PS with a single connection capability.

security--an indication of the sensitivity of information; in AUTODIN II there are 16 security levels.

segment--a data transmission unit comprised of header information for routing and optional text information.

server--an entity, usually a process, that offers service, for example, a time-sharing executive process.

SIP--Segment Interface Protocol (or the program that implements it); a sub-module of the host, TAC, or CCU that interfaces to the LCM for PS-TCP communication.

socket--an entity defining one end of a TCP connection; the inter-network-wide name of a process port which is a concatenation of network identifier, TCP identifier, and port identifier.

SRI--Stanford Research Institute, Menlo Park, California.

subscriber--the logical address of an access circuit, which at the PS level represents an end-user or a host computer.

T-segment--the concatenation of a TCP-TCP header and data (optional).

TAC--Terminal Access Control module. Consists of the following sub-modules: TC, THP, TCP, and SIP.

TC--Terminal Control module; A module of the operating system that interfaces terminals to processes. In the TAC it interfaces directly to the LCM for user-TAC communication.

TCC--Transmission Control Code; a user group code that serves to compartmentalize traffic and define controlled communities of interest among subscribers.

TCP--Transmission Control Protocol (or the program that implements it).

Telnet--an Arpanet protocol (or the program that implements it) that specifies the communication interaction such that a user on a terminal of one computer gains access to the services of another computer as if she were a local user of the second computer.

terminal-profile--a data table that indicates the settings of terminal control options, such as echoing, data blocking, and page formatting.

THP--Terminal-to-Host Protocol (or the program that implements it).

type-in-mode--the choice between entering commands (command mode) to the THP and exchanging text (send/receive mode) with the remote location.

UMB--User Multiplexing Block; a bookkeeping block used by the THP (program) to maintain information about a user.

user-group--a set of subscribers that belong to a community of interest or traffic compartment (see TCC).

3. BACKGROUND

3.1. Configuration Overview

3.1.1. The AUTODIN II system is a packet switched communication network linking a wide range of terminals and host computers. There are four main components of a packet switched communication system:

3.1.1.1. The Backbone: the packet switches and their interconnecting trunk lines.

3.1.1.2. The User Interface: the various interface devices that connect to subscriber equipment. Examples of the interface devices are the Terminal Access Controller (TAC), the Multiple Channel Control Unit (MCCU), and the Single Channel Control Unit (SCCU). Figure 1 shows the various user interfaces.

3.1.1.3. Protocols and conventions needed for communications and control between the various components in the backbone.

3.1.1.4. Protocols and conventions needed for communications and control between the various processes in the user environment. These include the host level protocol (TCP) and the terminal access protocol (THP).

3.1.2. AUTODIN II is designed to provide a communication service needed to support Interactive, Query/Response, Bulk Data Transfer, and Narrative applications to meet the man-to-man, man-to-computer, and computer-to-computer data transmission requirements of DoD users in CONUS and certain overseas subscribers.

3.1.3. Host computers are defined as computers, or front-end devices (FED) associated with computers, that are capable of simultaneously conducting multiple conversations with other hosts or terminals.

3.1.4. In addition, two types of host interface control units are provided: a SCCU and a MCCU. The purpose of these two interface control units is to allow certain host subscribers to interface to the network with little or no modification to their host hardware or software.

3.1.5. The program modules necessary for accessing the network will either reside in user hosts, or in user-provided network

SUBSCRIBER INTERFACE CONFIGURATIONS

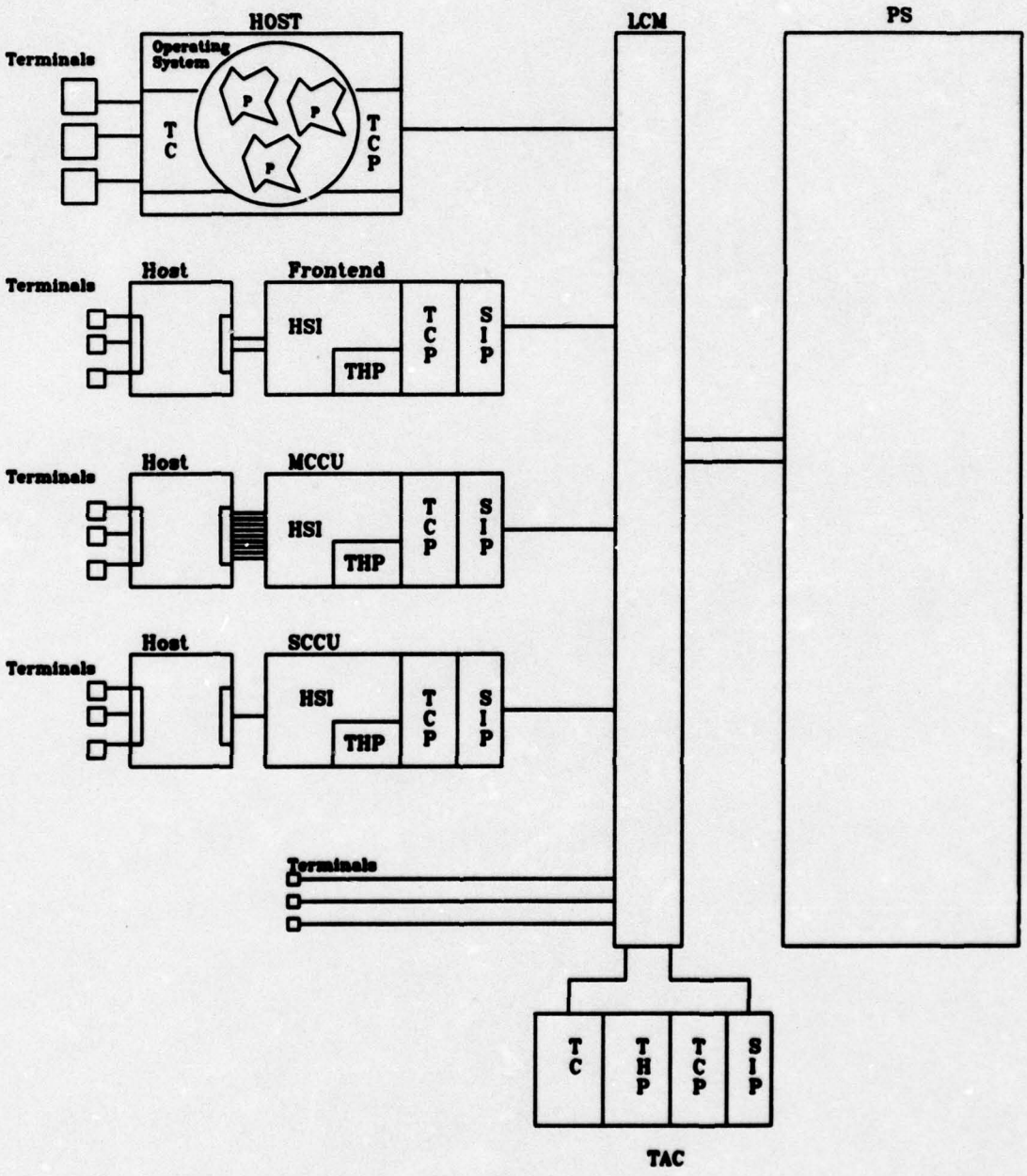


Figure 1

front-end devices (FED), or in the SCCU's and MCCU's provided by AUTODIN II. These program modules are the Transmission Control Program (TCP), which performs host-to-host protocol functions, and the Segment Interface Protocol (SIP), which performs the network interface protocol functions.

3.1.6. Terminals are defined as character oriented devices capable of conducting communications with only one destination at a time. Terminal subscribers will access the network through a TCP-SIP configuration using a Terminal-to-Host Protocol (THP). This protocol, which is similar to the ARPANET Telnet, interfaces the terminals to the TCP program module in the TAC to provide the mechanism for terminals to access foreign host computers via the communications network.

3.1.7. For terminals associated with a host subscriber computer, the THP/TCP/SIP program modules reside in the host, FED, SCCU, or MCCU as the case may be. For terminals not associated with a local host computer the THP/TCP/SIP program modules reside in the Terminal Access Control (TAC) functional module associated with the Packet Switch (PS).

3.1.8. Subscriber Interface Configurations

3.1.8.1. The above mentioned modules must be tailored to fit into a variety of user configurations. The configurations are shown in Figure 1 and described below.

3.1.8.1.1. Host

The most general configuration is the direct host configuration, in which the TCP and the THP are implemented in the host as operating system or service process functions.

3.1.8.1.2. Host Front-End

The next configuration shown, the Front-End Device (FED) case, requires a host specific interface (HSI) software module in the front-end which implements still another protocol between the front-end and the Host.

In most cases the front-end is intended to deal with the complications of network protocols, saving the Host both CPU cycles and memory space. The protocol used between the Host and the front-end is intended to be as simple as

possible, often a simulation of some device to which the Host is already interfaced.

3.1.8.1.3. MCCU

The third case is the MCCU which is intended to interface a host to the network with a minimal effect on the Host. The strategy is much the same as the front-end except that the physical interface is defined to be a set of up to 32 lines where each line is associated with one logical connection at a time. The intent is that each line may appear to the host as a terminal or some other simple device.

3.1.8.1.4. SCCU

The next case, the SCCU, provides a single-connection interface to the network. The principal intended use of the SCCU is to provide to a pair of hosts currently connected via a private line the less expensive alternative of using the network. In this situation each host would have an SCCU interface to the network. Other uses of the SCCU might include an interface for a synchronous remote job entry station. Implementation details for a SCCU are significantly simpler than for the other configurations. A separate appendix, Appendix H, contains a list of all the simplifications realizable with the SCCU configuration.

3.1.8.1.5. TAC and Terminals

The fifth case is the TAC. The TAC is intended to provide the TC, THP, TCP, and SIP services for the terminals directly connected to the LCM.

3.2. Protocol Overview

3.2.1. Before discussing the particulars of the THP, a description of the protocol environment is presented. First a scenario of a message transmission is outlined, then the protocol functions are discussed, and finally the structural relationship between the protocols is described.

3.2.2. Message Transmission Scenario

3.2.2.1. The major modules in the communication path between a user and a service process are shown in Figure 2.

3.2.2.2. The user (Subscriber 1) at Terminal 1,0 first enters

PROTOCOL ENVIRONMENT

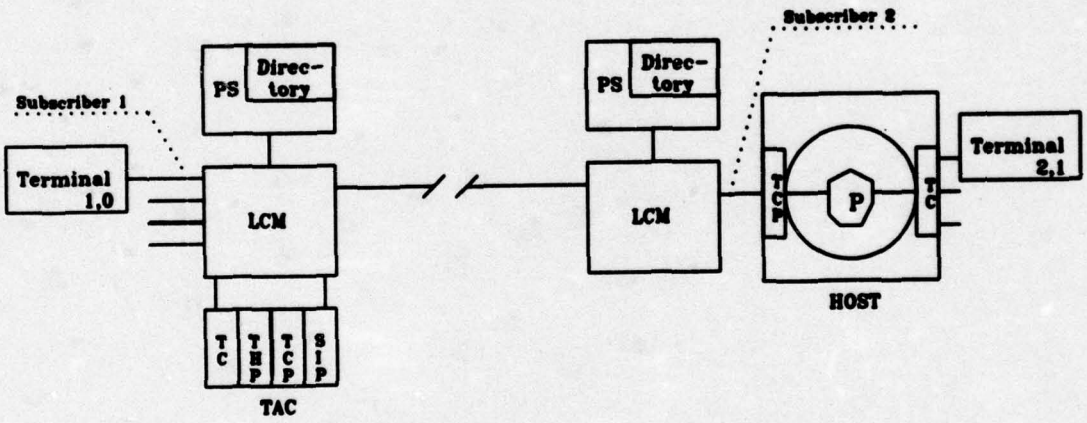


Figure 2

information about the destination she wishes to address, and the S/P/T parameters for the communication. The user's input is routed through the Line Control Module (LCM) to the TC module in the TAC. The TC delivers the data to the THP, which processes the information. The THP makes an OPEN request on the TCP specifying the destination and S/P/T as indicated by the user. The TCP sets up a table entry (a TCB) for the pending connection, but no network messages are transmitted.

3.2.2.3. When the user enters her first data message, the THP packages the text into a letter and passes the letter to the TCP with a SEND request. The TCP checks the TCB entry for this connection and finds that the connection is not yet established. The TCP then exchanges connection establishing messages with the distant TCP. The TCP passes the messages (segments) to the SIP, which forwards the segment to the PS via the LCM.

3.2.2.4. When this opening exchange is completed, the TCP reformats the original letter into segments which include sequencing and the S/P/T control information. As with all segments destined for a foreign TCP, this segment is passed to the SIP for forwarding through the network.

3.2.2.5. The PS verifies that the S/P/T values indicated in this message are valid by checking the Directory entry for this subscriber. On passing this test the PS routes the segment to another PS, and by travelling from one PS to another the segment eventually reaches the PS of the destination address.

3.2.2.6. At the destination the segment is routed from the PS through the LCM to the Host (in our example). In the Host the segment is processed by the TCP. The TCP reformats the segment, deleting the control information and possibly combining several segments to form a letter for delivery to the destination process. The TCP sends an acknowledgement (possibly combined with data) to the data originating TCP. The destination process acquires the data as the result of a RECEIVE request.

3.2.3. Protocol Functions

3.2.3.1. In this scenario several functions have been mentioned or implied. In the following we will review the functions and indicate the division of responsibility for implementing them among the protocol modules.

3.2.3.1.1. Conceptual Model for Higher Level User

Each protocol module provides a model of its communication facility to the next higher level of use. This model is an attempt to make the lower levels of protocol invisible to the user of the particular protocol module. Examples of these models are the inter-process communication facility provided by the TCP and the terminal-to-host model provided by the THP.

3.2.3.1.2. Device Dependent Terminal Control

The device dependent terminal characteristics include conversion of character coding, folding of long lines of text, etc. This function is performed by the TC.

3.2.3.1.3. Character Set Standardization

The community of users of the network utilize many different types of terminals and host computer systems. Among these terminals and systems several character sets are used. To enable users to interact with all other subscribers, a network standard character set is defined. The THP is responsible for any translation necessary.

3.2.3.1.4. Interaction Mode Control

The control by the user over the mode of interaction is accomplished via an interaction with the THP. These modes include the choice between character-at-a-time and line-at-a-time interaction, and between local echoing and remote echoing.

3.2.3.1.5. User Command Interpretation

The user should have the ability to control the interaction mode, connection opening and closing, and various options possible under THP as well as the device characteristics. These controls should be effected through the use of a command language. The THP is responsible for parsing the command language and carrying out the requested action, often by calling on the TC or TCP.

3.2.3.1.6. Multiplexing Connections

The use of the network by a host or TAC must usually be subdivided or multiplexed into many conversations so that the many users (terminals or processes) are able to simultaneously obtain services. The TCP provides this multiplexing function by presenting to the THP level the conceptual notion of connections, which is the basic mechanism for process-process communication. Connections may be thought of as logical circuits.

3.2.3.1.7. Ordering

Most applications of the network require data to be delivered in the order in which it is sent. This function is accomplished in the TCP by means of sequence numbers.

3.2.3.1.8. Packaging

Users usually wish to be spared the details of segment formats and network communication. A packaging function is performed by each module along the path to provide the user with a simpler view of the communication system. The major step in this packaging is the formatting into (and out of) segments, which is accomplished by the TCP.

3.2.3.1.9. Reliable Transmission

Users expect that all messages they present to the network will be delivered. To provide for the reliable transmission of messages both the PS and the TCP utilize sequence numbers, positive acknowledgements, retransmission, and flow control mechanisms.

3.2.3.1.10. Security, Precedence, and User Group Monitoring

The security, precedence, and user group of each message must be verified. The PS will do this by checking each message against the PS Directory at both the source and destination. The TCP may provide additional checking.

3.2.3.1.11. Multiplexing Hosts and Subscribers

The backbone network is shared among many subscribers by multiplexing the message flow according to the subscriber identification. Subscribers may be hosts or terminals

connected to TAC's; they are not processes. This function is provided by the PS.

3.2.3.1.12. Addressing and Routing

The PS provides addressing for Hosts and most terminals by means of the concept of Subscribers. The TCP provides addressing for processes and other terminal users with sockets.

The PS provides for the optimal routing of messages through the network.

3.2.4. Levels of Protocol

3.2.4.1. The protocol modules form a hierarchy, as shown in Figure 3. Users conceptually transmit text to other users. In actuality they exchange text with the supporting THP's. The THP's in turn conceptually exchange letters, but actually send and receive letters to and from the supporting TCP's. The TCP's conceptually exchange T-segments, but in fact these T-segments are transmitted via the SIP's to the PS's as S-segments. The PS's actually exchange packets.

3.3. Users and Processes

3.3.1. Throughout this document the terms "user" and "process" are used regularly and often interchangeably. There is no distinction made between the two entities, since a user can be replaced by a process in most of the occurrences without any loss of generality. Some examples may have slightly different details, since a user interacts through a terminal device. The examples are quite often of the flavor, " ... a user talks to a remote process ... ," and are not meant to imply that other alternatives, e.g. a user-to-user dialogue, are not allowed.

3.3.2. In the descriptions of protocol module interfaces, it is unnecessary to restrict the neighboring module to be a user or user's terminal handler, when it can be a process. In some instances descriptions speak of the user mainly to indicate that it is the user-side of the THP (or TCP), as opposed to the network-side.

3.3.3. Occasionally, however, the user is the human user and what is being described is what the human at a terminal sees or how he controls his interaction with the network components he uses. User command languages, for instance, apply only to human users.

NLS graphics

LEVELS OF PROTOCOL

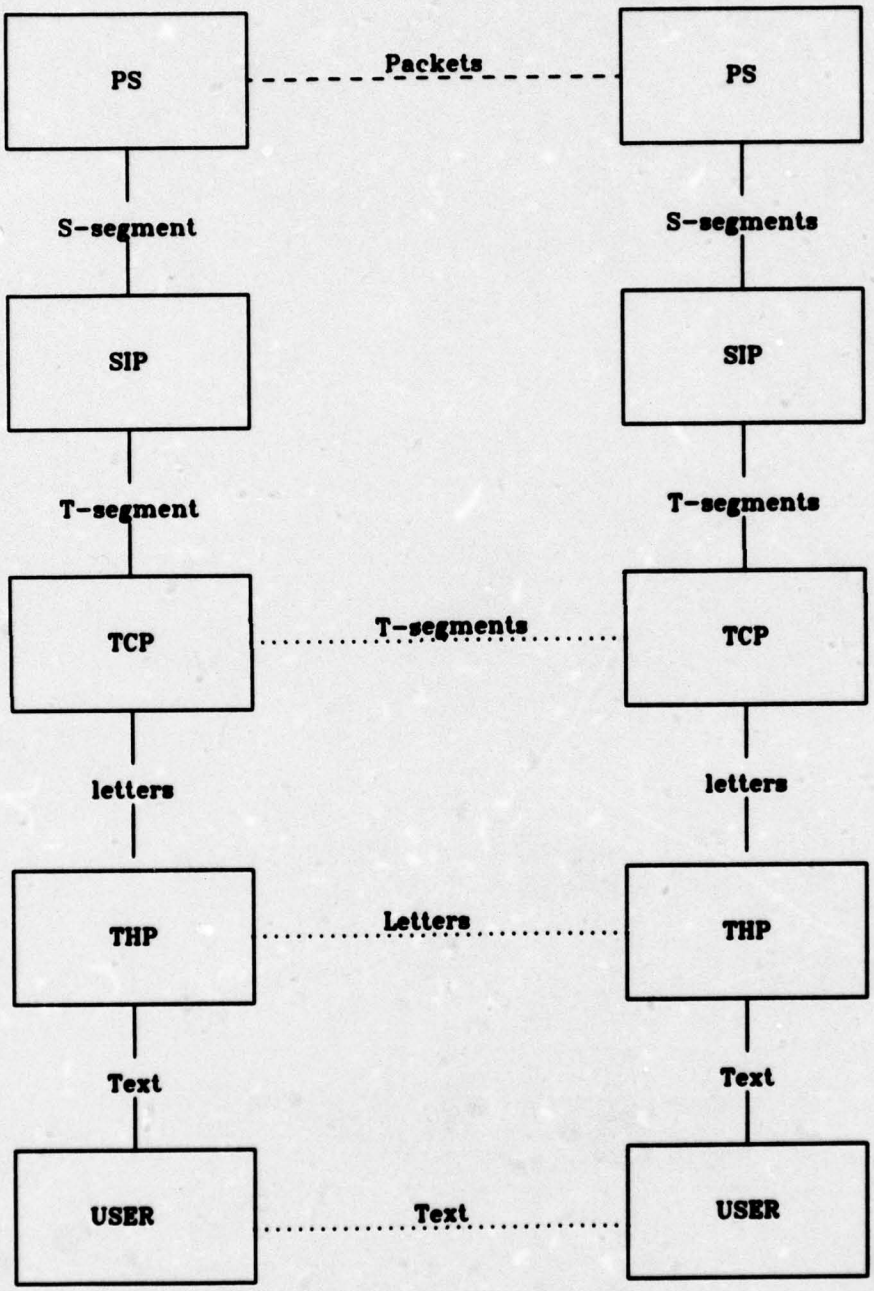


Figure 3

4. THP FUNCTIONS

4.1. Conceptual Model for the User

4.1.1. One of the primary functions of the THP is to give the user a view of the network that makes lower levels of protocol, i.e. TCP and PS-PS protocol, as invisible as possible. As stated earlier, a user conceptually passes text to another user. For the user to accomplish this task, he must have some model of how text is actually passed to a remote user. This model should clarify terminal, user, and process symmetry, explain how to address a user or process, and define what constitutes a user session.

4.1.2. Terminal-Process Symmetry

4.1.2.1. Two of the major services provided by the THP are terminal-to-terminal dialogues and terminal-to-process transactions. Of less importance are the process-to-process interactions. To the user, it should be equally easy to establish communication with a process as with another user's terminal. By making terminals and processes as similar as possible with respect to THP dialogues, the user has a simple conceptual model to understand. THP employs a model for terminals that makes the device-dependent characteristics invisible to the remote terminal or process, and essentially lets the terminal appear to be like a process with no distinguishing features.

4.1.3. User Addressing

4.1.3.1. Addressable entities on the network include both terminal users and processes. Terminal users of the AUTODIN II network are either subscribers (supported by a TAC) or host users. Addressable processes are entities found exclusively on hosts. The addressing scheme provided by the AUTODIN II PSN is only capable of addressing subscribers (TAC user or a distinct host), since it only provides for the subscriber ID in its S-segments and inter-switch packets. It is clear that user addressing requirements are not adequately met by the PS addressing mechanism.

4.1.3.2. The TCP provides a more powerful addressing scheme. The scheme is used only at the TCP-TCP level and co-exists with the PS addressing. The THP builds upon the TCP capabilities to present its own model of addressing to the user. A user address is the concatenation of subscriber ID, host user ID, and a

function selector. Not all of these address parts are required in every user address.

4.1.3.3. The simplest address, that of a subscriber supported by a TAC, is just the subscriber ID. This assumes a default (zero) user ID and function selector. The most complex address might arise when a non-default version of a host standard function is to be addressed. In this case all fields would be required--the subscriber ID would select the host, the user ID would select the standard function on the host, and the function selector would select the non-default version of the standard function. Appendix A supplies further detail on this matter.

4.1.4. Sessions

4.1.4.1. A session is more than the passing of a single message. It is defined to be the lifetime of an established communication path or connection. A session includes connection establishment, sequences of messages, and the disestablishment of the connection. Connections exist at the TCP level and they are conceptually similar at the THP level.

4.1.4.2. At the THP level, a session can be interrupted by connection preemption. The THP attempts to maintain the preempted connection in a suspended state so that resumption of the connection (session) can occur.

4.2. Character Set Standardization

4.2.1. The community of users of the network utilize many different types of terminals and host computer systems. Among these terminals and systems several character sets are used. To enable users to interact with all other subscribers, a standard character set is defined. This code set is seven-bit ASCII in an eight-bit field, plus some special characters to be described later in the document. The THP is responsible for any translation to or from the standard character set (an example of this is the translation of a network-standard erase line (EL) into the local character(s) that performs that function). The THP performs no other code conversion, such as EBCDIC to ASCII conversion, etc.

4.3. Terminal Controller Control

4.3.1. The user sees the THP as its interface to the network. Between the user and the THP may be several software or hardware modules that perform the function of terminal handler or

controller. These modules can be nearly invisible, in that the user never has to talk directly to them. However, these modules perform functions, such as duplex control or line formatting, that may require parameter setting by the user. The THP must provide, as part of its user command repertoire, a set of commands to set parameters in the terminal controller (see section 7.3.5 and Appendix B).

4.4. Connection Management

4.4.1. Interaction Mode Control

4.4.1.1. The control by the user over the mode of interaction is accomplished via interactions with the THP. These modes include the choice between character-at-a-time and line-at-a-time interaction, and the choice between local echoing and remote echoing.

4.4.1.2. The user should be able to instruct the THP when to send his text. This can be done in two ways: (1) upon receipt of a special character requesting "immediate" transmission of the data and, (2) on completion of a text unit, where a text unit is a user defined entity such as up to the end-of-line, every "n" characters, a full buffer, etc. These are the same functions required by the AUTODIN II specification [Reference 1].

4.4.1.3. Insofar as the mode of interaction and the availability of local buffer space permit, data should be accumulated in the process where it is generated until a complete line of data is ready for transmission, or until some locally-defined explicit signal to transmit occurs. This signal could be generated either by a process or by a human user.

4.4.1.4. Where possible, local echoing should be the rule. While this is not possible for highly interactive environments, it is appropriate for almost all line-at-a-time activities.

4.4.1.5. The motivation for the line-at-a-time and local echo modes is the high cost of remote echoing and the intention to keep the network traffic at the lowest possible level. Since many systems take some processing action at the end of each input line, it is reasonable to buffer data at its source and send it when the line termination character is detected. On the other hand, a user or process may sometimes find it necessary or desirable to provide data which does not terminate at the end of a line; therefore, methods should be provided to locally

signal that all buffered data should be transmitted immediately.

4.4.2. Packaging

4.4.2.1. The user does not wish to be burdened with the knowledge of the packaging schemes used by the protocols that support his dialogue. The THP performs the packaging function at both the THP-TCP and the THP-TC interfaces. The packaging must comply with any constraints upon letter sizes and text block sizes imposed by the TC or TCP. The connection's line mode, character-at-a-time or line-at-a-time, also influences letter sizes and text block sizes; for example in the character-at-a-time mode letters will often hold only one character and text may be delivered to the user in very small buffers.

4.4.3. Connection Control

4.4.3.1. The user must be able to direct the THP to perform some of the connection management functions. These functions include opening, closing, interrupting, flushing data, and entering and exiting a send-receive mode.

4.4.3.2. *Special Control Functions*

4.4.3.2.1. As stated before, a goal of the THP is the provision of standard interfacing of terminal devices to the network. Early experiences with this type of interconnection have shown that certain functions are implemented by most servers, but that the methods of invoking these functions differ widely. For a human user who interacts with several server systems, these differences are highly frustrating.

4.4.3.2.2. THP, therefore, defines a standard representation for the most common of these functions, as described below. These representations have standard, but not required, meanings; that is, a system which does not provide the function to local users need not provide it to network users and may treat the standard representation for the function as a No-operation. On the other hand, a system which does provide the function to local users is obliged to provide the same function to a network user who transmits the standard representation for the function.

4.4.3.2.3. Interrupt Process (IP)

This function is frequently used when a user believes his process is in an unending loop, or when an unwanted process was inadvertently activated. The resumption of execution is totally dependent on the interpretation of the interrupt by the receiving process. For a user-to-user connection, the interrupt could be used to get the attention of the remote user.

4.4.3.2.4. Abort Output (AO)

Many systems provide a function which allows a process, which is generating output, to run to completion (or to reach the same stopping point it would reach if running to completion) but without sending the output to the user's terminal. Further, this function typically clears any output already produced but not yet actually printed (or displayed) on the user's terminal. This command is needed to avoid unnecessary data traffic over the net if a user realizes he does not need the data.

4.4.3.2.5. Are You There (AYT)

Many systems provide a function which provides the user with some evidence that the system is still up and running. This function may be invoked by the user when the system is unexpectedly "silent" for a long time, and thus is useful for distinguishing a busy system from a dead one. This control does not bypass the normal data stream to perform the stated function.

4.4.3.2.6. Erase Character (EC)

Many systems provide a function which deletes the last undeleted character or "print position" from the stream of data being supplied by the user. This function is typically used to edit keyboard input when typing mistakes are made.

4.4.3.2.7. Erase Line (EL)

Many systems provide a function which deletes all the data in the current "line" of input. This function is typically used to edit keyboard input.

4.4.4. Preemption

4.4.4.1. Due to the precedence requirements of the AUTODIN II network, an established THP connection at any level lower than Category I can be preempted by a request for connection at any higher level. No facility exists at the TCP level to cause preemption of a TCP connection, thus the full responsibility for this function is assumed by the THP. This issue is discussed at length in 8.7.2.

4.4.5. Accountability of message receipt

4.4.5.1. Neither the THP nor TCP can guarantee that a message it has delivered in fact has been read by the user. Accountability for receipt of messages can only be achieved by a destination (human or process) response message. Without the response message, which really constitutes end-to-end acknowledgement, all that is guaranteed is that the remote THP has had a letter deposited in its buffer. During closing sequences, especially when the user gives an ABORT command, the THP may discard some of its previously received letters.

5. SECURITY, PRECEDENCE, AND TCC

5.1. The primary S/P/T consideration at the THP level is the need to preempt a connection in case a higher precedence connection becomes established. This affects only the terminal user. The terminal user can have only one active connection. If the current active connection is of lower precedence and a higher precedence connection becomes established, the lower precedence connection is suspended by the THP (see section 8.7.2).

5.2. Except for the preemption of connections, it is intended that the THP will not be concerned with S/P/T. Specifically the THP shall have no access to the authorization table or any other secure information in either the host or the TCP. It will merely pass all the information from the user (process) to the TCP and rely on the TCP to reject unauthorized use of connections. When a connection is established the THP will pass the S/P/T information to the user.

5.3. The THP endeavors to allocate its resources fairly between connections of the same precedence. The THP allocates resources to Category I connections in preference to all other connections.

6. THP ENVIRONMENT

6.1. The program that implements the THP is not a self-contained system. It has to communicate with the operating system under which it runs and with its neighboring processes. A description of the envisioned operating system environment in which the TCP operates is outlined below. Part of the THP's environment is its interface with its neighbors, the TCP on the network side and the user process or TC on the user's side. Both the specific inter-process communication mechanism and the conceptual models of interaction are discussed below.

6.2. Operating System Characteristics

6.2.1. Here we describe some operating system characteristics that are important for any implementation of THP. In no way is it implied that these characteristics are absolutely required of any operating system under which THP is to be implemented. Those characteristics that are most important to the foundation of the specification are discussed first, followed by some characteristics that would generally facilitate implementation of THP.

6.2.2. Strong Recommendations

6.2.2.1. Address spaces and data sharing

6.2.2.1.1. In an environment of cooperating processes in which large blocks of data are being passed between the processes, it is often important that the processes be able to pass data without copying it. It is therefore advisable that processes be able to share the physical address space in which the data resides or have some other mechanism for efficient data sharing.

6.2.2.2. Inter-process communication

6.2.2.2.1. The host inter-process communication mechanism should support asynchronous calls such that a module is not blocked waiting for I/O with neighboring modules. We recommend that a global event mechanism capable of passing event data be made available through the operating system, or by some subsystem running under the operating system. A more detailed discussion of interprocess communication is presented in section 6.3.

6.2.2.3. Resource Preemption

6.2.2.3.1. Some mechanism must be available to secure resources, both time and space, when high precedence activity is encountered. The operating system should have features in its process management, process scheduling, and space management that permit preemption. Some suggestions that relate to preemption are offered in the next section.

6.2.2.3.2. When in need of resources, the THP should first attempt to preempt resources it has itself allocated, prior to seeking preemption of resources allocated by the operating system to other processes. In an extremely heavy load, preemption within the THP may be impossible and the operating system's assistance might be needed to carry out preemption.

6.2.3. Suggestions

6.2.3.1. Process structure

6.2.3.1.1. It is not necessary to define the host, TAC, or CCU modules (SIP, TCP, THP, TC, HSI) or module subtasks as processes of the operating system. A process can be thought of as "a running program": the current code, resources, and context. While it may not matter if a process is known or managed directly by the operating system, it is conceptually useful to be aware of the hierarchy of processes. In the host, TAC, or CCU, the superior of a process is defined as the element that activates that process (usually some kind of scheduler).

6.2.3.2. Scheduling

6.2.3.2.1. Scheduling is performed at several levels in a multi-process environment and depending upon process structure, can be arbitrarily complex. Of concern here is the scheduling of top level processes (SIP, TCP, THP, TC, HSI) and the pitfalls to be avoided at all levels of scheduling. No mention is made here of host process scheduling; it is assumed that scheduling suggestions are helpful only for TAC'S, MCCU's, and SCCU's. Additional scheduling notes may be found in the discussion of THP Implementation.

6.2.3.2.2. TAC and CCU scheduling is primarily concerned with fairness in function and should be non-preemptive. Scheduling should be done using a round-robin scheme. To

insure some level of fairness between top-level processes, processes should limit their activity to some agreed upon maximum. To prevent deadlocks, processes should not block themselves waiting for resources to be freed or provided by other processes at that level.

6.2.3.2.3. In several of the TAC and CCU modules, special handling of high precedence activity is required. Scheduling according to the precedence of an activity should be handled in the respective processes, in a way that does not preempt a task in execution. For example, within the THP, no event should preempt the running of the from-user text handler. After the from-user text handler finishes with the present record, however, the scheduler will gain control and reexamine the THP event list for high precedence events before another process is scheduled.

6.2.3.3. Buffer management

6.2.3.3.1. Buffer management in the host, TAC, and CCU is a difficult problem and the "correct" algorithm will evolve only after performance measurements are made. The strategy must address several issues--flow control, deadlock avoidance, and throughput. As with scheduling, it is probably incorrect to handle all buffer allocation uniformly. Where special heuristics are required for solving allocation problems, the allocation should be handled at the process level (e.g. allocating buffers for packaging letters in the THP).

6.2.3.3.2. To facilitate buffer management we have adopted a policy in which the module that requests allocation of space is the one that releases it. Thus buffers allocated to the terminal handling process will be released by it, buffers allocated to the THP will be released by the THP, and buffers allocated to the TCP will be released by the TCP.

6.2.3.3.3. We envision at least two levels of space or buffer management--global management for allocation to the THP, TCP, and SIP modules, and local management for allocation within each of the modules. For instance, the THP space manager (a local manager) acquires space from the global manager and in turn allocates space to its modules, independently of the policy used by the other local managers.

6.2.3.3.4. The global manager is responsible for fairness among the user, THP, TCP, and SIP modules. Since no buffers

actually move across interfaces permanently, this fairness is straightforward to implement. The global manager cannot, however, address problems like deadlock prevention since it cannot determine how its allocations are used. Only the local managers, especially the user space manager, can attempt to distribute buffers to input and output traffic in such a way as to avoid deadlocks.

6.3. Inter-Process Communication

6.3.1. In implementing the THP protocol the need for asynchronous communication within a host-entity is very significant. To emphasize this point we will describe most inter-module communication in terms of events (with no distinguishing external characteristics). This is in contrast to other documents on host-to-host and process-to-process protocols, which describe inter-module communication in terms of procedure calls.

6.3.2. Process A must be able to signal Process B that event data is ready for it. Process A must regain control immediately following posting of that event with some indication that the event was posted successfully or unsuccessfully.

6.3.3. Posting events is an operation to be performed by the operating system (or some event sending mechanism running under the operating system). It enables the operating system to verify the authority of the event sender to communicate with the event receiver. It also informs the sender if an attempt is made to send an event to a non-existing process or to a process that has just crashed.

6.3.4. To facilitate event handling a process ID must be associated with each process. The process ID is the means by which the sender specifies the destination of the event. The operating system will attach the sender's ID to each event, thereby enabling the destination to distinguish among various sources he might be communicating with, and at the same time, prevent the sending process from masquerading as another.

6.3.5. Associated with each THP event are at least the two fields: "Subscriber ID" and "Transaction ID". All events are handled by the operating system in the same priority level. Each process (THP or TCP) should keep a list of events it has read but not yet serviced completely. Before starting on the highest priority listed event the process should read and enter into the list any newly received events.

6.3.6. Event data should be passed by value, i.e. copied from the sender's address space. The data should be copied into a sharable memory for fast access by the receiving process. It should be fixed in length and small so that allocation for event data is efficient. The event data could be thought of as being placed on the receiver's event list, which should be dynamic in length and never blocked.

6.4. Terminal and Process Models

6.4.1. In an attempt to achieve terminal-user and process symmetry it is important to have a firm conceptual model of the possible relationships between the terminal user, the TC, the THP, and the process. This is necessary if both user-to-user communication and user-to-process communication are to be encompassed in one model.

6.4.2. To that end, it is important to have a consistent model that enables processes designed to interact with local users to interact with remote users without having to undergo any modification for network communication.

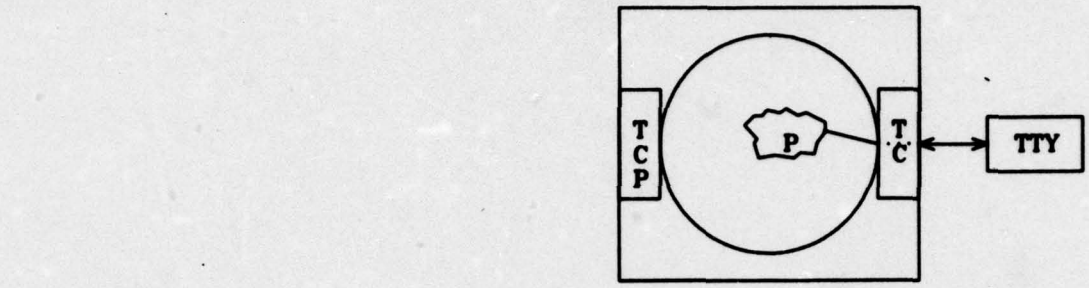
6.4.3. In a local interaction, a terminal oriented process calls upon the TC via an operating system call every time a terminal input/output operation is performed. This situation is described in Figure 4-a.

6.4.4. The introduction of the network into the terminal-process path can be described as an introduction of a pair of THP's at either side of the path (see Figure 4-b). Two important changes have occured: (1) the TC now mediates between two processes (one of them the THP) and, (2) there are two TC's involved: one at the process end, and the other at the terminal's end.

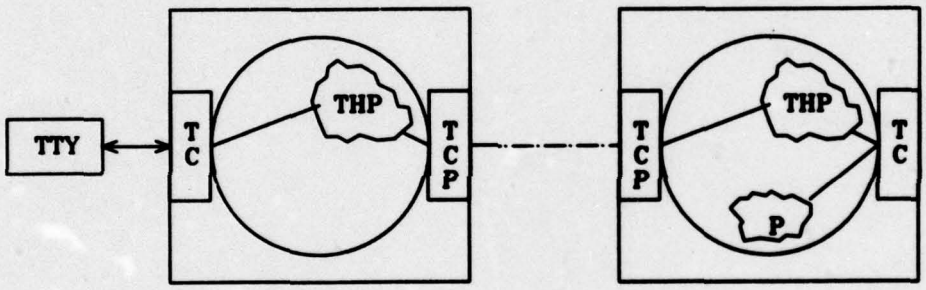
6.4.5. The TC at the terminal's end is a regular TC since it interfaces a physical terminal and a process. However, the TC at the process end is slightly different since in this case the THP acts as a terminal. We shall call such a process a pseudo terminal, and expect the TC to be able to distinguish between a real terminal and a pseudo terminal.

6.4.6. The third situation, showing user-to-user communication, is described in Figure 4-c. This situation does not involve any special operation on the part of the TC's at either end of the network, as both serve a physical terminal on one side and a process on the other. Further discussion of the TC's function is given in section 7.3.5.

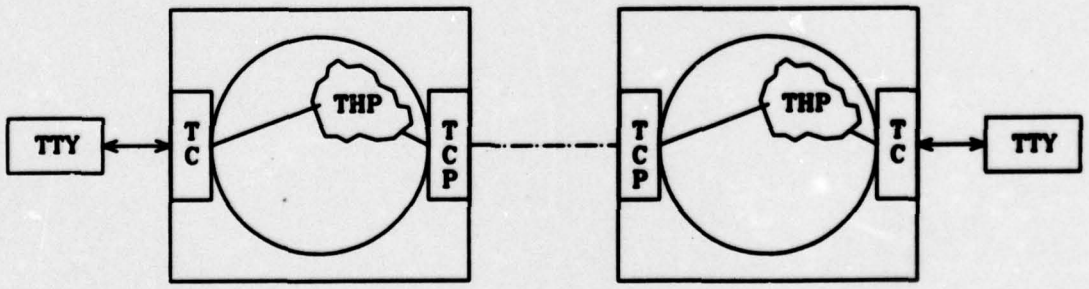
TERMINAL AND PROCESS MODELS



(a)



(b)



(c)

Figure 4

6.4.7. A set of conventions for terminals (or pseudo terminals) is necessary for implementing a model that maintains the terminal user and process symmetry discussed above. The concept of a Network Virtual Terminal is helpful for establishing these conventions.

6.4.8. Network Virtual Terminal

6.4.8.1. The THP has a simplified view of remote processes. Unless notified to the contrary, it assumes all remote terminals have the same characteristics. The default terminal device is called a Network Virtual Terminal (NVT). This concept is discussed in Reference 3.

6.4.8.2. An NVT is an imaginary device which provides a standard, network-wide, intermediate representation of a canonical terminal. All host-entities map their local device characteristics and conventions so as to appear to be dealing with an NVT over the network, and each assumes a similar mapping by the other party. This eliminates the need for keeping information about the characteristics of each other's terminals and terminal handling conventions in both the local and remote host-entity.

6.4.8.3. The Network Virtual Terminal is a bi-directional character device. The NVT has a printer and a keyboard. The printer responds to incoming data and the keyboard produces outgoing data which is sent over the THP connection and, if echoing is desired, to the NVT's printer as well.

6.4.8.4. The NVT is a canonical device and it's default characteristics may not be suitable for servicing all terminals. To provide for more elaborate conventions, a mechanism must be available to alter those set forth by the NVT model. This mechanism, called options negotiation, is fundamental to the THP implementation model.

7. THP INTERFACES

7.1. User Interface

7.1.1. The THP is the user's interface to the network and thus has several functions to perform in presenting a simple conceptual model at the user level (see 4.1). The interface that is actually provided for user interaction does much to strengthen or weaken that model. The user interface is presented below as a command language. While this shows a distinct orientation toward terminal users rather than processes, the language is also an adequate THP-process interface specification.

7.1.2. Command Language

7.1.2.1. A user views the network and its resources through a command language interface to the THP. It is desirable to keep all lower level protocols as invisible to the user as possible. Some suggestions about the general requirements of the command language interface are given below and in Appendix B. The exact command language syntax will not be specified, but rather will be left to the THP implementer.

7.1.2.2. The command language of the THP provides the user with a set of control functions by which he can affect the nature of his conversation over the network. The set of control functions can be divided into the groups listed below. (A complete description of the individual commands can be found in Appendix B.)

7.1.2.2.1. Connection Control: commands to open and close connections; set parameters such as destination address, security, precedence, and TCC; enter and exit send-receive mode.

7.1.2.2.2. Network Virtual Terminal Control: commands to set modes of NVT operation such as uppercase vs. upper- and lowercase, character-at-a-time vs. line-at-a-time, echo vs. no echo. These commands may result in altering the terminal profile in the TC and/or negotiating options with the foreign THP.

7.1.2.2.3. Data Transmission: commands that are allowed when in the text sending-receiving mode. These commands do not change the state of the connection or the terminal characteristics. No explicit command is needed for data transmission to occur once the text sending-receiving mode is

entered; however, there is a command, the THP escape character, to leave this mode.

7.1.2.3. It should be noted that providing a standard, consistent user interface to the network is important for the usability of the network. Users may access the network from many different sites and will expect to find the same network interface at each site. Documentation for users, such as user guides, must be current and complete. The documentation task is simplified if it supports a single, standard command language.

7.1.2.4. Some sites will not be able to support all the functions that are available at the most sophisticated site. Even so, the style of command language should be the same, even if the repertoire of commands is different. Some elements of command language style are recognition mode, prompt and feedback type, and help features. (See Reference 4)

7.2. TCP Interface

7.2.1. The communication with the TCP will follow the inter-process communication model described in 6.3, i.e. using events in an asynchronous manner. The set of events will include, among others, data to be transmitted, and control functions to the local TCP. The full set of events is described in Appendix C.

7.3. TC Interface

7.3.1. On the user's side the THP communicates with the Terminal Controller (TC). It is not the purpose of this specification to elaborate on the capabilities found in the TC of a host-entity. We must, however, present some hypothetical characteristics upon which to build a model of THP-TC communication. The TC is assumed to be a set of low level routines, probably offered by the operating system, which uses some asynchronous signalling mechanism (not necessarily the global event sending mechanism).

7.3.2. The THP hands the TC characters to be displayed on the user's terminal and receives from the TC text typed by the user. In addition the TC owns a data structure, called the terminal profile, in which physical parameters of the terminal are recorded. The TC shall offer a profile read-access mechanism to any process, and a modify-access mechanism to a selected set of processes, including the THP.

7.3.3. Processes may request to be informed by the TC whenever a

change is made to the profile. The TC will merely indicate that changes have been made to the profile but will not necessarily indicate which parameters were actually changed. It is the responsibility of the process to find out the actual change, probably by reading the profile and comparing it against a copy of its own.

7.3.4. In the following sections the functions of the TC are outlined and the terminal profile described.

7.3.5. TC Functions

7.3.5.1. Translation

7.3.5.1.1. THP's exchange text in the NVT character set, which is almost identical to the ASCII character set. Since various terminals may produce characters in different codes, such as EBCDIC, it is the task of the TC to translate characters coming from the THP into the character set of the terminal and vice versa.

7.3.5.1.2. Some terminals may have a smaller character set than the ASCII set; for example, terminals may not have lower case letters, etc. The TC may perform the case translation for such terminals if so desired by the user. The TC must also adopt a scheme for displaying non-printable characters; a process that sends a non-printable character expects the receiving user to be notified of the existence of such a character in its data stream. The exact translation is not specified, however.

7.3.5.2. Formatting

7.3.5.2.1. The NVT character set includes formatting characters such as the TAB, Form-Feed (FF), Vertical-Tabs (VT), etc. Some terminals have a built-in mechanism to perform the proper formatting when a formatting character is received. However, for terminals that do not have these capabilities it is the TC's responsibility to provide for simulation of such formatting. This requires that the TC keep track of the exact character position of the printing head, and know the page size and line width of the terminal it is serving. The TC is also expected to handle line overflows, i.e. to perform wrap-around when the text received from the THP overflows the line width of the terminal.

7.3.5.3. Line Mode Control

7.3.5.3.1. Two terminal line modes, character-at-a-time and line-at-a-time, must be supported by the TC. In character-at-a-time mode the TC must make the THP aware of any characters in its type-in buffer immediately. In line-at-a-time mode, the TC must hold the text in its buffer until an end-of-line character or a "send it now" character is detected.

7.3.5.4. Editing

7.3.5.4.1. Terminals, especially those working in a line-at-a-time mode, must provide for some line editing capabilities. (For terminals that do not have such capabilities a simulation must be provided via some printable convention.) A user must be able to instruct the TC to erase the last character (EC), or the last line (EL).

7.3.5.4.2. In some cases, frequently encountered in a character-at-a-time mode, line editing may not have local meaning since the entity that should be deleted may have already been transmitted. The TC should then translate the user's edit request to the appropriate NVT character and submit it to the THP, providing the user with appropriate echo.

7.3.5.4.3. The user should not be prohibited from sending characters that have special meaning to the TC, as text characters to the remote THP. In order to make this possible a Literal Escape character must be provided which instructs the TC to handle the next character as a text character even though it is usually treated as a special control character.

7.3.5.4.4. In order to facilitate execution of Abort Output command and proper handling of connection preemption, both the user and the THP should be able to flush the printing buffer of the TC.

7.3.5.5. Out-of-Stream Messages

7.3.5.5.1. In some cases processes need to insert text for the user, bypassing the normal text stream to the user's terminal. For instance, the operating system may need to display various messages, or the THP may need to notify the user when an interrupt occurs on the connection. Thus the TC

must provide special handling for text to be displayed to the user immediately.

7.3.5.6. Echoing

7.3.5.6.1. The THP provides for various echoing options. Echoing may be performed by the terminal, the local TC, the local THP, or even traverse the network. It is therefore necessary that the TC be able to inhibit its own echoing when required. This, of course, does not apply to half-duplex terminals, where the local echoing cannot be inhibitted.

7.3.6. Terminal Profile

7.3.6.1. To accommodate all the information about the physical properties and current settings for each terminal, the TC owns a data structure called the terminal profile. This data structure is made available to any process requiring information about the terminal through a set of calls on TC routines. In the configuration relevant here, the TC extracts data from the profile in order to perform the correct formatting, and the THP needs read access to the profile to assist its negotiation of options that deal with terminal characteristics (such as line width, echoing, etc.). The THP needs write access to the profile when an option that bears influence on the terminal parameters has been successfully negotiated.

7.3.6.2. Since the TC is the owner of the terminal profile it should be the only one that modifies it. Hence, the TC should provide functions callable by other processes to read or modify the profile, such as a READTERMPROFILE and a SETTERMPROFILE. In the host case these functions may be part of the operating system; in the TAC they could be provided by the TC itself. It should be noted that the TC itself never initiates a new setting for the terminal profile, as this must be done by the process that communicates with the terminal user. The TC does have responsibility for the initial settings of the profile, however.

7.3.7. TC Operation

7.3.7.1. The TC is a character oriented interface, that usually resides between a process and a physical terminal. Each terminal is assigned an ID (such as line number) and the "owning" process is known by some process ID. The TC passes characters typed on the terminal to the requesting process and

vice versa using these ID's. In addition, the process must be able to modify the profile of the terminal it is communicating with, so it can adapt the terminal characteristics to its operation.

7.3.7.2. When a terminal user is talking to a remote process it is necessary for the remote TC to mediate between a THP and a process. In this situation the remote THP acts like a pseudo terminal, i.e. its operation is limitted to submitting and receiving characters to and from the TC. The access to the terminal profile is somewhat more complex as both the process and the THP may need to modify it. (Note that a profile in this case is a profile of a pseudo terminal.)

7.3.7.3. As we do not assume that processes should distinguish between real and pseudo terminals, the TC must somehow distinguish between the two. The THP shall request that it be informed by the TC whenever changes are made to the profile by the process. This is done so that the THP has an updated version of the profile, and so that the THP can initiate option negotiation if the changes in the profile justifies it. Only a subset of the profile will contain characteristics that are negotiable by the THP, examples being echoing mode and tab simulation.

7.3.8. TC-THP Communication

7.3.8.1. Communication with the TC is assumed to be accomplished via signals and subroutines that the TC makes available to all processes. Signals and calls affecting the THP can be categorized into from-user text, to-user text, and terminal profile operations.

7.3.8.2. The TC sends a DATA-AVAILABLE signal when it has text from the user to send to the network. When available to do so, the THP responds by performing a READTC call to copy the data into a letter buffer. The THP calls WRITETC in the TC when it needs to pass text to the user.

7.3.8.3. As mentioned in the last section, the TC manages a per terminal data structure, the terminal profile. The profile may be read and modified through TC subroutines. READTERMPROFILE and SETTERMPROFILE subroutines make it possible for the THP to change the way the TC handles the terminal. This ability is required if the exact terminal characteristics are to be made known to the TC for proper terminal handling, and to the THP for negotiation of options.

8. THP IMPLEMENTATION

8.1. Introduction

8.1.1. The following sections discuss some of the implementation issues that are important in any THP. It is necessary to establish a model of implementation to give a framework for discussing the issues. We have chosen such a model, but the choice does not indicate that a particular implementation must conform to this model. Our model is modular and easily understood. It is also consistent with the interface requirements set forth in the TCP specification [Reference 2].

8.1.2. The THP is modelled as one superior process (external event handler) which interacts with its neighboring modules, the TC and the TCP, and schedules two inferior processes, the from-user text handler and the to-user text handler (see Figure 5). The THP is modelled as one instance, which handles the multiplexing of all THP connections. (Again, this is assumed for discussion only and does not imply that an instance of THP per user is not an acceptable implementation model.)

8.1.3. The processes of the THP access several data structures, most of which are shared. Communication between the three THP processes is through the intra-THP signal board. Two lists, the to-net letter list and the from-net letter list, hold entries for each pending SEND or RECEIVE letter. A Connection Management Block (CMB) holds per connection state information, and a User Multiplexing Block (UMB) holds per user information required for multiple connections and connection preemption. The data structures are discussed again in 8.2.

8.1.4. A set of subroutines are shared among the processes. They are invoked by call, in contrast to the processes which are scheduled. The various processes and subroutines are called the functional modules of the THP and are discussed in 8.3.

8.1.5. Figure 5 shows the relationships among the functional modules of the THP and the letter lists. Text and letter flow is shown in solid lines, signal-type control in dot-dashed lines, and subroutine call control in dashed lines. To aid in understanding the relationships, a brief scenario of a typical operation is given below.

8.1.6. The scenario is between a terminal user and a remote process from which the user obtains some service, such as information retrieval. The scenario will include a request by

THP MODULES AND DATA STRUCTURES

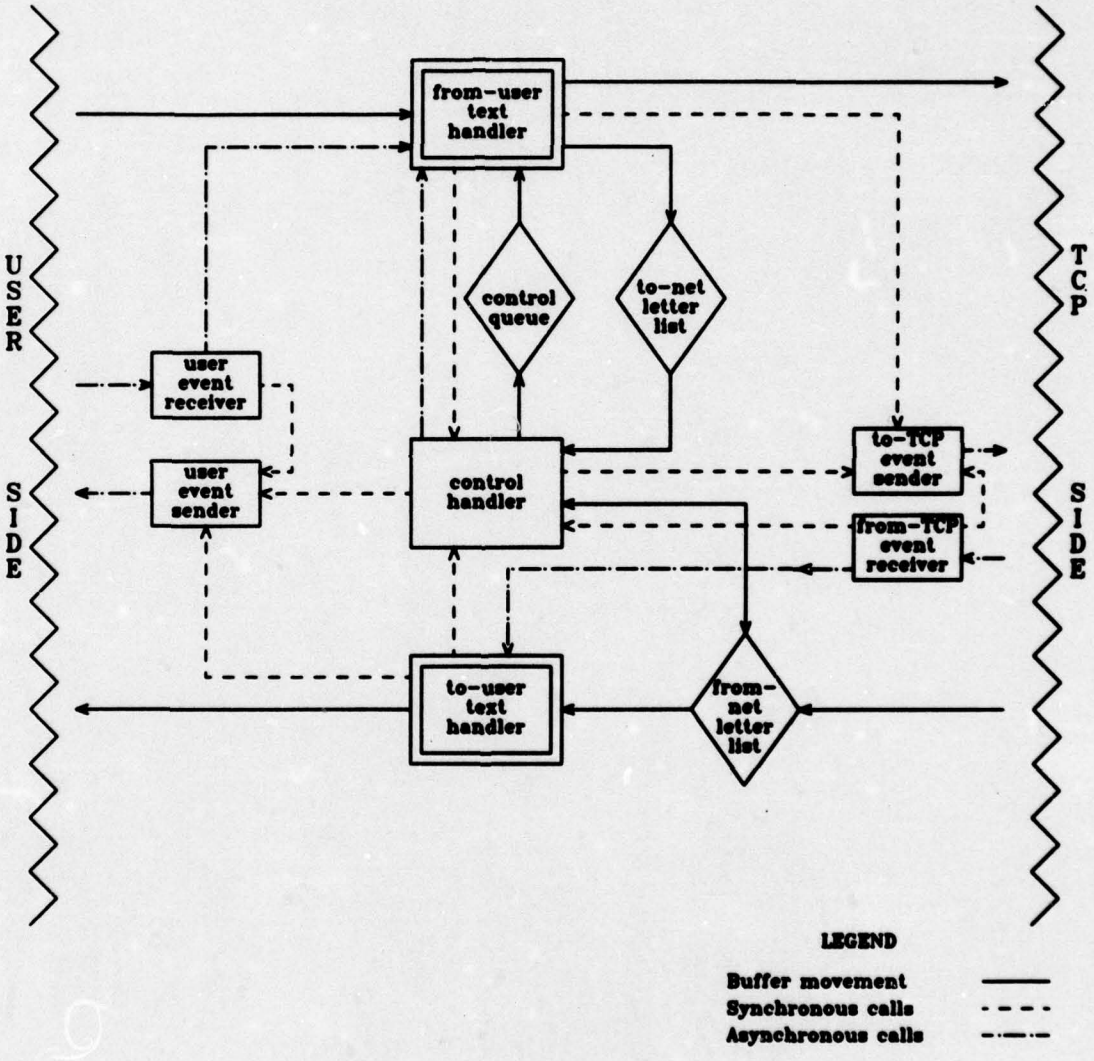


Figure 5

the user and a reply by the process. The connection establishment phase will not be covered. The user will be assumed to be at a line-at-a-time terminal, with the echo provided by the terminal device.

8.1.7. The user types a request on his terminal and terminates the request with an end-of-line character, such as a carriage return. The characters from the user are echoed by the terminal so no echo characters are sent from the TC to the terminal. When the TC receives the end-of-line character, it sends the line of text to the THP. This is done by signalling the THP that data is available and awaiting a READTC call from the THP.

8.1.8. When the THP runs, its external event handler determines if any new data from the TC is available. If there is, its address and length are copied to the CMB for this connection and the from-user text handler is signalled (through the intra-THP signal board) to copy the text to a letter buffer.

8.1.9. The from-user text handler is responsible for scanning the stream for any special control characters typed by the user and packaging the text into one or more letters. The letter buffer's address is added to the to-net letter list and a SEND event is sent to the TCP, via the to-TCP event sender. The End-of-Letter (EOL) flag is always set in the SEND event. The to-net letter list simply holds letter buffer addresses so that SEND return events from the TCP can be associated with a specific letter.

8.1.10. Once the SEND is performed by the THP, transmission is complete. Acknowledgement of letter delivery to a remote THP buffer is detected by the return event for the SEND, which is handled by the TCP event receiver. The return causes the letter buffer address to be removed from the to-net letter list and the buffer space freed, both by the THP control handler when called by the from-TCP event receiver.

8.1.11. The reply to the user's request arrives as an incoming letter from the TCP. The THP must have passed previous RECEIVE events to the TCP to make the TCP aware of its receive buffers. The THP keeps a record of its receive buffers by placing their address on the from-net letter list. A return from the RECEIVE event indicates that there is a letter or partial letter in the receive buffer, depending on the mode specified in the RECEIVE. The return event is fielded by the from-TCP event receiver, which signals the to-user text handler to process the letter.

8.1.12. The to-user text handler is responsible for scanning the

letter for controls and performing any character set translation that is required. Finally, it calls the TC to transfer text to the user's terminal, through a call to the user event sender. (The call on the TC is unspecified here. It should return without blocking and should indicate how much text was copied.) When the text has been copied, the letter buffer should be removed from the from-net letter list.

8.1.13. The transaction ends when the TC notices it has text for the user's terminal and sends the text, with whatever character code translation and formatting is necessary, to the terminal.

8.2. Data Structures

8.2.1. Understanding the data structures of a program is important to the understanding of its functional operation. In this section we present the data structures that the THP accesses.

8.2.2. External Event List

8.2.2.1. The THP's external event list is a data structure that provides a location for the event sending mechanism to deposit events and event data. It is an unordered list, each element of which holds the event sender's process ID and the address of a block of event data. The operating system provides primitives that allow a process both to create its own empty event list and to send events to other processes. This data structure may be supported in many ways and is used in this document to model the interprocess communication mechanism used for THP-TCP and some TC-THP communication.

8.2.3. Intra-THP Signal Board

8.2.3.1. The THP signal board provides an inter-process signalling capability for the intra-THP processes. An entry on the board is a single bit, settable by any THP functional module and resettable by the process that it signals. The signalling of processes is an important efficiency measure. It aids the THP and the process in determining what work, if any, it has pending. Since each process does work for all THP connections, it would be highly inefficient searching all the CMB's to determine that there is no work pending.

8.2.3.2. The following signal board entries and their destinations support process signalling:

from-user text work (from-user text handler)

control work (from-user text handler)

from-net letter work (to-user text handler)

8.2.4. Connection Management Block (CMB)

8.2.4.1. A shared data structure is required to handle bookkeeping on a per connection basis. The data structure should contain enough information to determine the state of a connection, locate any letter buffer associated with a connection, and process any control occurring on or for that connection. The data structure that supports these functions in the THP is called the Connection Management Block (CMB) which is created and maintained for the lifetime of a given connection. Details on the structure of the CMB are given in Appendix D.

8.2.5. User Multiplexing Block (UMB)

8.2.5.1. Since it is possible that a terminal user will have two connections open at any time, one low precedence and one high precedence, and since there is no limit on the number of connections that a process can have open, a special data structure, the UMB, is needed to locate the CMB's of each connection belonging to a user or process. Multiple connections are discussed further in section 8.7.3.

8.2.6. To-net Letter List

8.2.6.1. The to-net letter list holds the addresses of all the outstanding letters that have been sent via a SEND event but have not received return events (acknowledgements) from the TCP. Note that this requires that each SEND to the TCP is a full letter, i.e. it has the EOL flag set.

8.2.7. From-net Letter List

8.2.7.1. The from-net letter list is the list of committed receive buffers that the THP has offered for this connection through previous RECEIVE events.

8.2.8. Control Queue

8.2.8.1. The control queue is a data structure that is shared between the THP control handler and the from-net text handler for the purpose of storing controls that need to be inserted in the THP-THP data stream.

8.3. Functional Modules

8.3.1. The functional modules of the THP are a composite set of THP processes and subroutines. In conjunction with the data structures described previously, the module descriptions below help to present a comprehensive implementation model. Figure 5 is a useful diagram for viewing these interrelationships.

8.3.2. External Event Handler (process)

8.3.2.1. The external event handler is the top-level process of the THP. It examines the THP external event list and calls user or from-TCP event receiver, depending on the source of the event. The handler has the additional responsibility of scheduling the from-user and to-user text handlers.

8.3.3. User Event Receiver (subroutine)

8.3.3.1. The user event receiver, when called by the external event handler, is responsible for translating events from the process on the user's side (TC or other process) into calls and signals on the THP's functional modules. The most notable event will be the DATA-READY signal from the TC, indicating that data can be copied from the TC's user output buffer.

8.3.4. From-TCP Event Receiver (subroutine)

8.3.4.1. The user event receiver, when called by the external event handler, is responsible for translating events from the TCP into calls and signals on the THP's functional modules. Many of the events will be return events, corresponding to the call events previously sent by the THP. The event receiver must also support the THP-TCP event acknowledgement scheme by sending ACK/NACK events through calls on the to-TCP event sender.

8.3.5. User Event Sender (subroutine)

8.3.5.1. The THP user event sender is responsible for making calls upon TC subroutines and sending signals to the TC when instructed to do so by the other intra-THP modules.

8.3.6. To-TCP Event Sender (subroutine)

8.3.6.1. The to-TCP event sender is the module called to send events to the TCP. As described in the TCP specifications [Reference 2], the TCP "primitives" are available through events and return events. The event sender is responsible for sending the events listed in Appendix C.

8.3.7. From-User Text Handler (process)

8.3.7.1. The main functions of the from-user text handler are to scan the text from the user for controls, parse the command language when in command mode, package user text and THP controls into letters, and send the letters to the TCP. The handler is scheduled by the external event handler whenever there is text available from the TC to be packaged or a control needs to be sent. The pending work for the handler is located by following CMB chains, one for high precedence and one for low precedence, and examining each CMB for an indication of unpackaged text.

8.3.7.2. The handler must process the text stream as typed by the user (or simulated by a process). The stream includes controls as well as text bound for the destination user. The handler must scan the stream character-by-character since each character is a potential control character. The mode of input, command or send-receive, determines whether the text should be interpreted as commands to the THP or text for the destination user. When in the send-receive mode, the handler must scan for the the escape character that signals a return to the command mode.

8.3.7.3. When in the command mode, the handler must perform command language parsing. The handler must formulate user commands into calls on the control handler, if necessary. One of the commands is a mode switching command, which will put the mode back into the send-receive mode.

8.3.7.4. The from-user text handler is also responsible for adding THP-THP controls to the outgoing stream. The need to send a control is indicated by the control handler. A separate

shared data structure, the control queue, is used for this purpose. These controls are always records, even if the connection itself is in the stream mode (see 8.5).

8.3.7.5. The handler packages of text and controls into letters for the TCP. (The THP never sends partial letters, i.e. without setting the EOL flag in the SEND event.) Depending on the mode, stream or record, text will be copied directly into letters or further packaged into THP records (see 8.5 for details). Letter construction and transmission will be directed by the user in some instances, such as the Send-Now character in the line-at-time mode. Otherwise, the handler should send the letter as soon as it has packaged the last available character from the TC. When the packaging of text into a letter is complete, the to-TCP event sender is called to pass a SEND event to the TCP.

8.3.8. To-User Text Handler (process)

8.3.8.1. The to-user text handler has the basic responsibility of translating the letters received from the TCP into text for the user and controls for the THP. The letters must be scanned and parsed, and text for the user must be passed on to the TC.

8.3.8.2. Scanning and parsing input

8.3.8.2.1. The handler must perform data stream (letter content) scanning as specified in 8.5. When controls are encountered, the control handler is called to process them.

8.3.8.3. Packaging letters into text

8.3.8.3.1. Letters containing text should be made available to the user immediately. The handler does this by calling the user event sender, which then makes the appropriate call on the TC.

8.3.9. Control Handler (subroutine)

8.3.9.1. The control handler is the center of all control activity; generation or receipt of controls always results in a call on the control handler.

8.3.9.2. Control handling is the subject of section 8.6. The full set of user-THP, THP-THP, and THP-TCP controls are presented there.

8.3.9.3. When a THP control is to be sent, the control handler notifies the from-user text handler by signalling it through the intra-THP signal board and by adding the control to a shared data structure, the control queue.

8.3.9.4. A control may require that a message be passed to the user. In this case, the message is sent directly by the control handler, through a call to the user event sender.

8.3.9.5. Some controls require the generation of a special control event for the TCP. The control handler calls the from-TCP event receiver to generate such events.

8.4. NVT Model

8.4.1. The NVT has a printer and a keyboard. The NVT printer has an unspecified carriage width and page length and can produce representations of all 95 ASCII graphics (codes 32 through 126). Of the 33 ASCII control codes (0 through 31 and 127), and the 128 uncovered codes (128 through 255), the following have specified meaning to the NVT printer:

8.4.1.1. NULL (NUL)

A no operation.

8.4.1.2. Line Feed (LF)

Moves the printer to the next print line, keeping the same horizontal position.

8.4.1.3. Carriage Return (CR)

Moves the printer to the left margin of the current line.

8.4.2. In addition, the following codes shall have defined, but not required, effects on the NVT printer. Neither end of a THP connection may assume that the other party will take any particular action upon receipt or transmission of these:

8.4.2.1. BELL (BEL)

Produces an audible or visible signal (which does NOT move the print head).

8.4.2.2. Back Space (BS)

Moves the print head one character position towards the left margin.

8.4.2.3. Horizontal Tab (HT)

Moves the printer to the next horizontal tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.

8.4.2.4. Vertical Tab (VT)

Moves the printer to the next vertical tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.

8.4.2.5. Form Feed (FF)

Moves the printer to the top of the next page, keeping the same horizontal position.

8.4.3. All remaining codes do not cause the NVT printer to take any action.

8.4.4. The sequence "CR LF", in accordance with the definitions above, will cause the NVT to be positioned at the left margin of the next print line (as would, for example, the sequence "LF CR"). However, many systems and terminals do not treat CR and LF independently, and will have to go to some effort to simulate their effect. (For example, some terminals do not have a CR independent of the LF, but on such terminals it may be possible to simulate a CR by backspacing.) Therefore, the sequence "CR LF" must be treated as a single "new line" character and used whenever their combined action is intended; the sequence "CR NUL" must be used where a carriage return alone is actually desired; and the CR character must be avoided in other contexts. This rule gives assurance to systems which must decide whether to perform a "new line" function or a multiple-backspace that the THP stream contains a character following a CR that will allow a rational decision.

8.4.5. The NVT keyboard has keys, or key combinations, or key sequences, for generating all 128 ASCII codes. Note that although many have no effect on the NVT printer, the NVT keyboard is capable of generating them.

8.4.6. In addition to these codes, the NVT keyboard shall be capable of generating the following additional codes, which are described in section 8.6:

8.4.6.1. Synch, Break (BRK), Interrupt Process (IP), Abort Output (AO), Are You There (AYT), Erase Character (EC), Erase Line (EL)

8.4.7. The spirit of these "extra" keys, and also the printer format effectors, is that they should represent a natural extension of the mapping that already must be done from "NVT" into "local". Just as the NVT data byte 68 should be mapped into whatever the local code for "uppercase D" is, so the EC character should be mapped into whatever the local "Erase Character" function is. Further, just as the mapping for 124 is somewhat arbitrary in an environment that has no "vertical bar" character, the EL character may have a somewhat arbitrary mapping (or none at all) if there is no local "Erase Line" facility. Similarly for format effectors: if the terminal actually does have a "Vertical Tab", then the mapping for VT is obvious, and only when the terminal does not have a vertical tab should the effect of VT be unpredictable.

8.5. THP Data Stream Scanning

8.5.1. THP-to-THP data stream scanning is the interpretation of the contents of letters passed between THP's. This scanning, which is not to be confused with the scanning of terminal user's input, can be handled in one of two modes, which differ in the way text and controls are intermixed in the data stream. The first, the record mode, divides the data stream into records (of varying size) so that both text and control appear as records in the stream. The second mode, the stream mode, is unformatted and allows text to appear anywhere in the stream. The need to include controls in the stream is satisfied by a special escape character that signals the THP that a control follows.

8.5.2. THP's are required to implement both schemes, with the record-mode being the default. A special THP control enables the receiver switching from one mode to another; this is described in section 8.6.1.5.4.

8.5.3. Both directions of data flow need not be in the same mode; a THP may send letters in the record mode and receive letters in the stream mode. A THP may request that the remote THP start sending in a new mode by using a REQUEST-MODE control. This request, like an option negotiation request, may be rejected by

the foreign THP. However, the local THP may inform the remote THP of its intent to switch modes by sending a SET-MODE control. This indication is not a request; it must be accepted by the remote THP. This design allows the implementation of a THP with only one sending mode, which may be desirable in special configurations like the SCCU.

8.5.4. The RECORD Mode

8.5.4.1. This mode divides the data stream into records of varying sizes. A record consists of two parts: a header that carries information about the size of the record and its content, and a body that contains the data itself (text or controls). The advantage of using the record mode is that the receiving THP does not have to check each character it receives from the remote THP. If the record contains text to be transferred to the user the THP will transfer the entire text without further scanning. If the record contains THP control information the appropriate control handling module is called to process the rest of the record.

8.5.4.2. The size of records varies for a number of reasons. In control records, the size is determined by the number of parameters that have to be transmitted. For the DATA record, however, other rules apply since additional directives may be issued by the user. The user may instruct the THP of the manner in which records should be created. Line-at-a-time operation indicates that records should be made at the termination of each line. Records may also be terminated upon an explicit request from the user to send the data "now", or to send the data every x characters. However, records may not be broken, i.e., a letter must contain a whole number of records.

8.5.4.3. Precautions should be taken to avoid deadlocks. Deadlocks are a problem when a THP that operates in the record mode communicates with a character-at-a-time process. As a rule of thumb, stream mode should be used when the transmission mode is character-at-a-time.

8.5.4.4. The record header consists of the following octets (in this order):

Record Marker (RM)
High order bits of the record length
Low order bits of the record length
Record type

8.5.4.5. The first octet of the header is a record marker (RM), which is a begin-of-record mark. A THP, when expecting a new record, should verify that a RM appears as the next octet. The RM thus enables the THP to verify its synchronization with record boundaries; the RM also serves as a flag in the stream mode as described in the next section.

8.5.4.6. The next two octets comprise a 16 bit number that represents the length of the record-body in octets. The high order bit of the length is always zero and thus only 15 bits are used to describe the record length. This is designed so that normal arithmetic instructions can always be employed by a computer with 16 bits per word.

8.5.4.7. The record type describes the content of the record-body. Record types include THP controls and data. The complete set of control types and their formats are given in Appendix E.

8.5.4.8. A letter must contain a whole number of records. Thus, the beginning of a letter coincides with the beginning of a record, and the end of letter coincides with the end of a record. This can assist THP's in the synchronization of record boundaries. There is no limit to the number of records per letter.

8.5.5. The STREAM Mode

8.5.5.1. In the stream mode the input stream is considered to be an infinitely long sequence of octets. In order to allow THP-THP controls to be discriminated from data, a mechanism must be provided to flag THP controls in the stream.

8.5.5.2. To allow intermixing of controls with text in the stream, a control record is inserted into the stream. The begin-of-record mark (RM) acts as a flag for the remote THP that what follows is a single record, to be interpreted as a control. To avoid ambiguity, the bit pattern that represents the RM is outside the ASCII character set so that all ASCII characters are still available. The octets following the RM must obey the rules for records: two octets representing the length, a type octet, and (possibly) a body. At the end of this record the mode is switched back to the stream mode.

8.5.5.3. The mechanism described above requires a single control parser per THP for both record and stream mode. In the record mode the control parser is called for every control type

record, and in the stream mode the control parser is called when a RM is encountered in the data stream.

8.5.5.4. Since the RM has a special meaning to the receiving THP it cannot be a part of the data exchanged between the users. In special cases, when other than textual data is exchanged between users (e.g. binary data), and it is necessary to transmit a bit pattern identical to that of the RM, a record is required. This can be achieved by either switching to the record mode or sending a record of type DATA containing the RM octet.

8.5.5.5. The stream mode should be used when a character-at-a-time type communication exist between THP's, or whenever short messages dominate the conversation. Stream mode should be avoided if the data being transmitted is non-textual.

8.5.5.6. Controls are always transmitted in records. This, again, can be achieved by either switching to the record mode or sending a control type record in the stream mode.

8.5.5.7. The most important difference between the modes is that in the record mode all octets are guaranteed to belong to a record, whereas in the stream mode octets that do not fall within record boundaries are considered text.

8.6. THP Control Handling

8.6.1. THP-THP Controls

8.6.1.1. THP's exchange controls by sending records in the data stream. The controls are filtered out of the stream being passed to the user. Some of the controls are used to represent features commonly found on systems, others provide for the negotiation of NVT characteristics, and others support general THP-level connection management. More details of the exact record syntax for each control can be found in Appendix E.

8.6.1.2. RM

8.6.1.2.1. As already pointed out, the record marker (RM) has two functions. In the stream mode, it serves as a control escape flag and indicates that the following text should be interpreted as a control record. It also serves as a begin-of-record mark in the record mode. For more details, see 8.5.

8.6.1.3. Common Control Functions

8.6.1.3.1. Earlier, in 4.4.3.2, the need for a set of control functions that translate into common system features was discussed. The following controls support that set of functions:

Interrupt Process (IP)

The recipient THP should interrupt the local process to which the NVT is connected. The TC will translate the action into a meaningful message for the terminal user. This control can be used as part of the out-of-band signal for other protocols which use THP.

Abort Output (AO)

The recipient should allow the current process to (appear to) run to completion, but should not send its output to the user. It should also send a Synch to the remote user (see Appendix F). Since this will flush all letters in that direction of the connection, some rules must be followed to ascertain that the functions of flushed controls are performed. (see Appendix F)

Are You There (AYT)

The receiver send back to the remote user some visible (i.e., printable) evidence that the AYT was received. For a process, the evidence may be the status of the process. For a user, it should be the user's status as provided by the TC.

Erase Character (EC)

The recipient should delete the last undeleted character or "print position" from the data stream. Thus the local delete character sequence should be inserted into the data stream for the user terminal or process.

Erase Line (EL)

The recipient should delete characters from the data stream back to, but not including, the last "CR LF" sequence sent over the connection. Thus the local delete line sequence should be inserted into the data stream for the user terminal or process.

8.6.1.4. Option Negotiation Support

8.6.1.4.1. In section 8.8 details of negotiating options between THP's are discussed. Below is the set of THP-THP controls that support option negotiation.

DO

Requests that the other party perform, or confirms that you are expecting the other party to perform the indicated option.

DONT

Demands that the other party stop performing, or confirms that you no longer are expecting the other party to perform the indicated option.

WILL

Indicates the desire to begin performing, or confirms that you are now performing, the indicated option.

WONT

Indicates the refusal to perform or continue performing the indicated option.

XCONTROL

Negotiation of some options is an agreement to include a new control type. The XCONTROL record has the new control as a parameter, thus it allows for unrestricted extension of the control set (see Appendix G for details).

8.6.1.5. Connection Control

8.6.1.5.1. DATA

A DATA record is sent whenever text is being transmitted in the record mode or when the code for RM is being sent in the stream mode.

8.6.1.5.2. SUSPEND

The SUSPEND control is sent by the THP when a connection being suspended due to preemption. For a complete discussion of the generation of this control and the proper action to take when this control is received, see 8.7.2.

8.6.1.5.3. CONTINUE

The CONTINUE control is sent by the THP when a suspended connection is resumed. For a complete discussion of generation of this control and the proper action to take when this control is received, see 8.7.2.

8.6.1.5.4. SET-MODE

This control is used to declare that the mode on this direction of the connection has been set to either record or stream mode.

8.6.1.5.5. REQUEST-MODE

When a THP wishes to change the scanning mode for the incoming receive side of a connection, a REQUEST-MODE control is sent. This control is suggestive only and does not require any action by the recipient. It should not be repeated unless some change in the session is experienced.

8.6.1.5.6. DM

This control is the in-line portion of the SYNCH command (see Appendix F), i.e. it travels in the data stream. It should always be accompanied by a TCP Interrupt, the out-of-line portion of the SYNCH command.

8.6.1.6. Miscellaneous

8.6.1.6.1. NOP

No operation.

8.6.1.6.2. BREAK (BRK)

This control is provided because it is a signal outside the ASCII set which is currently given local meaning within many systems. It is intended to indicate that the Break Key or the Attention Key was hit. Note, however, that this

is intended to provide a 129th code for systems which require it, not as a synonym for the IP control.

8.6.1.6.3. STATUS-REQUEST

This control allows a THP to determine the status of the current THP options as viewed by the THP at the other end of the connection. It is a request for the status of options that are not currently in the default state.

8.6.1.6.4. STATUS-REPLY

This control contains the status of all the current THP options that are not in the default NVT states. Therefore the length of the record carrying this control is variable.

8.6.1.6.5. REPEAT

This control is used to signify the multiple occurrence of an octet (ASCII or binary unit). The control can be used when the text stream from the user or process includes a data octet that is repeated, but only makes sense if the character is repeated more than 6 times. This allows for the overhead of the REPEAT control, but not for the overhead of breaking a stream into records at the inappropriate place, such as before the last character.

8.6.1.7. Special Control Handling

8.6.1.7.1. A few special rules are necessary for the reliable operation of the THP protocol. Since both controls and data flow on the THP connection, it is dangerous to flush data without regard for what controls may be "in the pipe." Below are a few general rules and a specific procedure relating to Abort Output. For further information on THP SYNCH's and THP INTERRUPTs see Appendix F.

8.6.1.7.2. General Rules

Do not send on a connection until the last FL-INT is acknowledged (the return INT event has arrived).

Do not send a FL-INT on a connection until all previous DM's are acknowledged.

Do not send on a connection until the CONTINUE is acknowledged.

8.6.1.7.3. Abort Output

The receiver of an Abort Output (AO) control is obligated to flush the pipe in the direction of the sender of the AO. Flushing is accomplished by sending a TCP INT-FL. To synchronize the time of the TCP flush with a point in the THP data stream, a DM is sent in the THP data stream. The flushing of the data stream could cause unwanted flushing of THP controls. If the rules listed above are followed, this problem can be handled by a simple procedure--after flushing the path with a TCP INT-FL and THP DM, send a SET-MODE for the current scanning mode and if Remote Controlled Transmission and Echo (RCTE) is active, send an RCTE with the current settings.

8.6.2. User-THP Controls

8.6.2.1. The user may issue commands to control the connection, the TC, and the NVT. These were discussed earlier in the Command Language section 7.1.2.

8.6.3. THP-TCP Controls

8.6.3.1. TCP Primitives

8.6.3.1.1. The THP and TCP interact through the event mechanism supported by the operating system (see 6.3). The conventions established for the contents of the events to the TCP resemble a call/return mechanism, so one can speak of TCP primitives. The set of primitives support letter transmission, connection management controls, and messages to/from the PS. In many cases, the commands available to the user in the Command Language map directly into TCP primitives. A list of all TCP events can be found in Appendix C.

8.6.3.2. Event Exchange Support

8.6.3.2.1. A small subset of the events, the ACK and NACK events, are used to support reliable event exchange. They provide acknowledgement of the receipt of an event and indicate whether or not the event is syntactically correct.

8.7. Connection Management

8.7.1. Connection States

8.7.1.1. The THP uses connection state information to determine the appropriate action to take when it receives controls from the user, TC, or TCP. The state of a connection is stored in the Connection Management Block and should be available for read to any functional module. The state is modified only by the THP control handler. The following set of states is useful as a conceptual model for the user as well as an implementation model for the system programmer.

8.7.1.2. Open

8.7.1.2.1. A CONNECT or LISTEN command has been issued by the user, but no TCP synchronization has occurred. To move from this state to the ACTIVE state either a SEND from the user (CONNECT case), a connection attempt by a remote THP (LISTEN case), or initial options negotiation exchanges must occur.

8.7.1.3. Active

8.7.1.3.1. A TCP connection has been established (see Open state) and has not been preempted. The connection has a data path established to the user or process. For the terminal user, this is the only path to the THP; all text from the THP is associated with this connection and all type-ins from the user are bound for this connection.

8.7.1.4. Suspended

8.7.1.4.1. A previously active connection has been preempted by a higher Category connection and both THP's are synchronized with respect to the suspension of the connection. This state has two varieties, one for the locally suspended end of the connection and one for the remote end of the connection which has been notified of the suspension. (Note that a connection may be suspended both locally and remotely, a situation called simul-suspend.) To move back to the Active state, the initially suspended end must be resumed by its THP following the preemption, and the remote end must receive a THP-THP CONTINUE control (see 8.7.2).

8.7.1.5. Suspending

8.7.1.5.1. As described in 8.7.2 suspending a connection requires the handshake of both THP's so they are synchronized. This is an intermediate state between the Active and Suspended states.

8.7.1.6. Resuming

8.7.1.6.1. Resuming a preempted connection requires sending a CONTINUE to the remote THP and insuring that this message has been received. According to the rules cited in 8.6.1.7.2 nothing should be sent on a connection until the CONTINUE is acknowledged (the sending buffer is returned). The RESUMING state signifies that a CONTINUE was sent but not yet acknowledged.

8.7.1.7. Closed

8.7.1.7.1. A closed connection is either a non-existent or a closing connection. This state is the origin state of a connection. It can also be entered upon receipt of a user ABORT command or CLOSE command. If a CLOSE was issued, the state means that a deferred close is in progress. This is the only case in which the state is actually recorded, since in all other cases the bookkeeping information for the connection, the CMB, does not exist.

8.7.2. Preemption Handling

8.7.2.1. As stated earlier, one of the main responsibilities associated with connection management is the support of connection preemption. Preemption in this case is the preemption of a data stream, not of time and space resources. It is primarily a switching mechanism that allows a terminal user to be interrupted by higher priority activity and later to resume at the point of interruption.

8.7.2.2. First, it is important to remember that input or output for a terminal user can be associated with only one connection. The terminal user cannot be expected to interpret messages from several connections at once. A process, on the other hand, can handle the multiplexing of data streams in a number of ways, and need not be limited to one active connection like the terminal user.

8.7.2.3. There are three notable characteristics of connection

preemption in the AUTODIN II network--(1) preemption occurs only for terminal users, not for processes, (2) preemption of a Category I connection will never occur, and (3) no connection can be preempted by a connection at a precedence level lower than or equal to the precedence of the active connection.

8.7.2.4. To achieve preemption, the THP must always have one listening connection for each user (unless the active connection is of Category I precedence) and that connection must be in at least the TCP Open state at all times, waiting for connection attempts at precedence levels higher than the precedence level of the active connection. This listening connection serves as the path for preemption. The OPEN event that was sent to the TCP indicates unspecified security level, unspecified destination address, unspecified send precedence, receive precedence one category higher than the active connection, and unspecified TCC.

8.7.2.5. When a connection request arrives for the listening connection, the request forces the listening connection into the Active state. If another connection is already active, it must be suspended in favor of the higher priority connection. The preemption appears to be immediate for the local user. If another connection is in a non-Active state it should be closed using an ABORT command. Although closing such connections is not necessary for its reliable operation, the THP might not be able to dedicate the required resources for a long period. From this it is obvious that at most one connection can be in the open state, and only one connection in the preemption-related states.

8.7.2.6. The THP will retract all its receive buffers for the preempting connection so that the pipe from the remote THP is clear, and no more text is delivered until the connection is resumed. All text that is currently available in the THP will be buffered for the user and saved, and notification of the establishment of the new preempting connection will occur.

8.7.2.7. No more text will be sent from the user on the old preempted connection, rather a SUSPEND will be sent as the last letter from the preempted connection, and the connection is moved to the SUSPENDING state. The TCP knows nothing of these state changes. Figure 6 is useful in following the description of the suspending procedure described below.

8.7.2.8. The receiver of the SUSPEND must first suspend the output, which means signalling the user or process that it has

THP PREEMPTION DIAGRAM

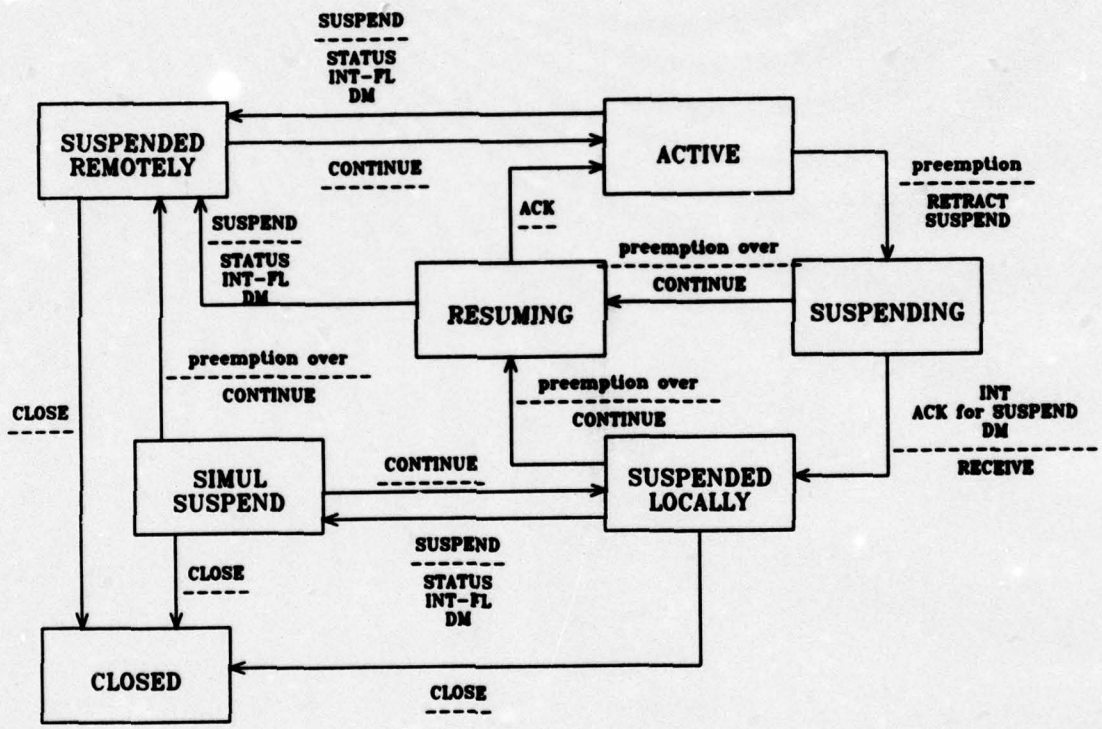


Figure 6

been suspended, and place a TCP STATUS command to find the exact amount of data received by the remote THP (Note that since the initiating THP has retracted its buffers, and since the SUSPEND letter carried an updated TCP acknowledgement, the return STATUS event will accurately indicate the amount of data received). The receiver of the SUSPEND must then place a TCP flushing interrupt (INT-FL) and a THP data mark (DM) in its output data stream.

8.7.2.9. The SUSPEND is then processed, which means that the user or process should not be allowed to create any more output traffic until the sender of the SUSPEND sends a CONTINUE. When the acknowledgement of the flush arrives the receiver of the SUSPEND will regain control of its send buffers, which must be scanned for undelivered THP controls. All controls should be saved for retransmission except for a possible undelivered SUSPEND which must be deleted and never retransmitted. The convention is that if a record was only partially delivered it is assumed by both sides not to have been delivered at all.

8.7.2.10. The receiver of the SUSPEND will move the connection to the remotely Suspended state and act as a very restrictive filter, Accepting only a small subset of controls (CONTINUE, CLOSE, INTERRUPT, DM). In this state it awaits one of three events--the resumption of the connection, a close of the connection, or preemption at its end of the connection.

8.7.2.11. When the sender of the SUSPEND receives the interrupt and realizes that his SUSPEND was received, it provides receive buffers once again so that it can receive THP controls. When it receives the DM, it has reached a synchronizing point, after which it expects no more text to be sent, and moves the connection to the locally Suspended state. It will continue receiving controls on that connection but if it receives text, it will immediately close the connection. In this state, if a SUSPEND is received, the above procedure is repeated but the connection moves to the Simul-Suspend state.

8.7.2.12. When the preempting connection is closed, the THP will attempt to resume the old preempted connection. First it will determine if the remote side closed, in which case it will inform the user. Otherwise, it will send a CONTINUE. Then it must determine if the remote side is itself in the suspended state, i.e. has sent a SUSPEND, in which case a Suspended remotely state is assumed. If it appears that the remote side is waiting for resumption, the THP will inform the user that

the connection is being resumed, and move the connection to the Resuming state.

8.7.2.13. The receiver of the CONTINUE will notify its user or process and remove restrictions on the output side.

8.7.2.14. The following rules are coupled with the above algorithm (in addition to those mentioned in 8.6.1.7.2):

8.7.2.14.1. A suspended connection that needs to be closed should be closed via the ABORT command to avoid deadlocks.

8.7.2.14.2. The arrival of a CONTINUE without a previous matching SUSPEND should be treated as a NOP.

8.7.3. Multiple Connections

8.7.3.1. Multiple connections are supported by the THP for both terminal users and processes. The limitations on the number of connections permitted and the precedence level of the connections is one area where the THP treats a terminal user and a process quite differently. The motivation for providing multiple connections for the terminal user is to provide a preemption path; while for the process, the motivation is to allow it to serve many connections at once. An important difference between a terminal user and a process is that the user cannot handle text from several connections; it is awkward to see a tag (holding the source and destination address) with each block of text displayed on one's terminal.

8.7.3.2. The terminal user may have at most two connections opened, of which only one can be active. The terminal user can only open one connection through commands to the THP. The second connection is used for a path for preemption; thus only the THP can open it. In fact, the user is not aware of having two connections unless an active low precedence connection becomes suspended as a result of preemption by a Category I connection. If a user already has a Category I connection opened, the THP will not bother making a preemption path available and as a result the user will have only one connection open. In the event that preemption occurs (see 8.7.2) both the preempted low precedence connection and the new preempting connection will be established, but only the new connection will be active (sending/receiving text with the user terminal). The preempted connection will now be in the suspended state.

8.7.3.3. A process may wish to multiplex many THP connections to itself. It can do this by making the same address (socket) the local end of several connections or by making a different address (socket) the local end of each connection. The THP does not explicitly limit the number of connections or the number of Category I connections that can be served by a process, nor does the THP ever preempt a process's connection. In reality, there must be some restriction on the number of connections due to finite resources.

8.8. Option Negotiation

8.8.1. The THP is primarily designed to provide terminal oriented network communication. Because this service is offered to a wide variety of terminals, the standard or default, set-up must adequately serve the least sophisticated terminal. There is no reason, however, why a more sophisticated terminal should not make use of its extra features if the process it is communicating with supports those features. The option negotiation is the mechanism that allows THP's to agree to use a more elaborate, or even completely different, set of conventions for their connection.

8.8.2. Negotiating options is not designed for sophisticated terminals and processes only, but rather provides a more general mechanism for the THP's to improve communication. For example, if the two terminals at both ends of a connection produce characters in the EBCDIC code, there is no reason why the text should be translated to ASCII at one end, and from ASCII at the other; the two parties can instruct their THP's to negotiate an option to transmit the text in binary and avoid the translation.

8.8.3. The basic strategy for setting up the use of options is to have either or both THP's initiate a request that some option take effect. The other THP may then either accept or reject the request. If the request is accepted the option immediately takes effect. If it is rejected, the associated aspect of the connection remains as specified for an NVT. Clearly, a THP may always refuse a request to enable, and must never refuse a request to disable some option since all THP's must be prepared to support the NVT.

8.8.4. The syntax of option negotiation has been set up so that if both THP's request an option simultaneously, each will see the other's request as the positive acknowledgment of its own.

8.8.5. The symmetry of the negotiation syntax can potentially

lead to nonterminating acknowledgment loops--each THP seeing the incoming commands not as acknowledgments but as new requests which must be acknowledged. To prevent such loops, the following rules prevail:

8.8.5.1. THP's may only request a change in option status; i.e., a THP may not send out a "request" merely to announce what mode it is in.

8.8.5.2. If a THP receives what appears to be a request to enter some mode it is already in, the request should not be acknowledged.

8.8.5.3. Whenever one THP sends an option command to a second THP, whether as a request or an acknowledgment, and use of the option will have any effect on the processing of the data being sent from the first THP to the second, then the command must be inserted in the data stream at the point where it is desired that it take effect. (It should be noted that some time will elapse between the transmission of a request and the receipt of an acknowledgement or rejection. Thus, a site may wish to buffer data, after requesting an option, until it learns whether the request is accepted or rejected, in order to hide the "uncertainty period" from the user.)

8.8.5.4. It is possible for requests initiated by processes to simulate a non-terminating request loop if the process responds to a rejection by merely re-requesting the option. To prevent such loops from occurring, rejected requests should not be repeated until something changes. Operationally, this can mean the process is running a different program, or the user has given another command, or whatever makes sense in the context of the given process and the given option. A good rule of thumb is that a re-request should only occur as a result of subsequent information from the other end of the connection or when demanded by local human intervention.

8.8.6. Option requests are likely to flurry back and forth when a connection is first established, as each party attempts to get the best possible service from the other party. Beyond that, however, options can be used to dynamically modify the characteristics of the connection to suit changing local conditions. For example, the NVT uses a transmission discipline well suited to the many line-at-a-time applications such a BASIC, but poorly suited to the many character-at-a-time applications such as NLS. A server process might elect to devote the extra processor overhead required for a character-at-a-time discipline

when it was suitable for the local process and would negotiate an appropriate option. However, rather than being permanently burdened with the extra processing overhead, it could switch (i.e., negotiate) back to NVT when the more taut control was no longer necessary.

8.8.7. The Mechanism of Negotiation

8.8.7.1. Two main ideas guide the mechanism of negotiation: (1) Any party may initiate or abort option negotiation at any time and, (2) THP's are not required to implement all or any part of the options.

8.8.7.2. Four controls are used in the context of option negotiation: DO, DONT, WILL, and WONT. Each of these controls must indicate the the option name it refers to and include relevant parameters (if any). The DO control requests the receiver of the control to use an option. The receiver must respond with a WILL control if he agrees to use the option, or with a WONT control if he does not agree. The WILL control is an offer by the sender to use an option. The receiver must respond with a DO if he agrees to use the option, or with a DONT if he does not agree. The full list of the currently defined options, their structure parameters and meaning, is given in Appendix G.

8.8.7.3. In summary, WILL XXX is sent by either THP to indicate its desire (offer) to begin performing option XXX; DO XXX and DONT XXX its positive and negative acknowledgments. Similarly, DO XXX is sent to indicate a desire (request) that the other THP (i.e., the recipient of the DO) begin performing option XXX; WILL XXX and WONT XXX the positive and negative acknowledgments. Since the NVT is what is left when no options are enabled, the DONT and WONT responses are guaranteed to leave the connection in a state which both ends can handle.

8.8.7.4. As stated before, THP's do not have to implement all the options. Thus, a THP that receives an offer or request to use an option unknown to it (i.e. not implemented), can respond negatively and leave the connection with its default convention with respect to that option.

8.8.7.5. At any time either party can disable further negotiation via the appropriate WONT or DONT.

8.9. Flow Control

8.9.1. The THP is conservative in handling flow control. For both directions of flow it has the ability to strictly control the flow of data. This is reasonable since the THP is completely responsible for all data passed to it. There is no way to recover previous data from its neighbors (TC or TCP) or from the remote THP with which it is conversing.

8.9.2. The THP is signalled when the TC has data for it to send, but it is the THP that actually initiates the transfer. When the THP must send data to the TC, it calls the TC, giving it the buffer address. The THP expects a transfer to occur immediately, even if it does not transfer the full buffer. Even if the user's terminal is blocked, there is no possiblity of the THP being overrun with incoming network traffic since it controls the TCP interface so stringently.

8.9.3. The THP's outgoing letters can be blocked by the TCP, so the THP must be careful not to accept data from the TC unless it has sufficient buffer space that can be used for packaging and queuing outgoing letters.

8.9.4. The flow control exerted on the traffic coming from the TCP is especially interesting, because the total receive buffer commitment, as indicated in the RECEIVE to the TCP, affects the TCP-TCP flow control and letter acceptance mechanism as well as the THP-TCP flow control mechanism. As discussed in Reference 2 the TCP will discard incoming traffic and advise the foreign TCP not to send letters if the user's (THP's) receive buffer commitment is zero. This should be avoided, since the normal control path to the THP, i.e. the contents of the letters, is shut if no letters arrive.

8.10. Buffer Management

8.10.1. The THP acquires buffer space from a global space manager, which also allocates buffers to the TCP and SIP. From this space the THP is responsible for allocating send and receive letter buffers to its connections.

8.10.2. Buffer management in the THP is closely related to flow control, as seen in the previous section. In addition to flow control considerations, a proper buffer allocation policy must speak to the issues of deadlock avoidance, fairness among connections, and buffer preemption.

8.10.3. Deadlock can occur on a connection when the local side of the connection is waiting for some input from the remote side, the contents of which determine what the next output from the local side is to be. This or a similar deadlock situation can happen whenever sufficient buffer space does not exist for one direction of flow on either side of a connection. The THP buffer allocation strategy must provide for some minimum buffer space per direction, rather than per connection. Otherwise one direction of flow could monopolize the buffers allotted for the connection.

8.10.4. Fairness of buffer allocation among connections can be achieved by keeping the allocation per connection similar in size. This assures a bandwidth that is equally distributed among the connections. This is true because the TCP flow control mechanism is heavily dependent on the receive buffer commitment.

8.10.5. It is necessary to assure a minimum bandwidth for high precedence connections. Part of this responsibility is assumed by the TCP (see Reference 2), but the THP must also contribute to giving high precedence connections a high level of service. To guarantee a minimum bandwidth, Category I connections must be guaranteed buffer space, unless all connections are Category I. (In this case, the connection should be closed due to the lack of resources, rather than risking deadlock on the other connections by reducing their buffer space.) If preemption of buffers held by lower precedence connections is necessary, then the lower precedence connection should be closed to recover its buffers.

9. MEASUREMENTS

9.1. To evaluate the extent to which the THP, TCP, and PSN can deliver services as required and to identify the dominant parameters that affect that service, a comprehensive set of performance measurements must be made. Measurement will also be a valuable tool for determining the initial settings for the values of THP parameters.

9.2. Measurement Implementation Philosophy

9.2.1. The THP level measurements may be accomplished by using instrumented versions of the THP process in selected locations. These special versions of the THP may collect statistics of various kinds that may be reported to a local file on a host, or via a connection from a TAC.

9.3. Performance Measurement

9.3.1. Throughput and delay should be measured as a function of precedence level, record verses stream mode, line mode, THP buffer allocation strategies, letter sizes, etc. Overhead should be evaluated in several modes of user activity.

9.4. Single Connection Measurements

9.4.1. Round trip delay times

9.4.1.1. The time from the moment the letter is sent by the THP to the time that the send buffer is released by the TCP.

9.4.1.2. In character-at-a-time remote echo mode the time from the moment the user types a character to the time that the user receives the echo.

9.4.1.2.1. This measurement should be parameterized by letter size to distinguish from one set of round trip times and another.

9.4.1.2.2. Network destination, current configuration, and traffic load may also be issues of importance that must be taken into account.

9.4.2. Letter Size Statistics

9.4.2.1. Histogram of letter size in both directions on the THP connection.

9.4.2.2. Ratio of text octets to total octets in letters. This is a measure of the overhead due to THP controls and records. This measurement should be parameterized by record verses stream mode.

9.4.3. Effective Throughput

9.4.3.1. This is the long-term rate at which text is processed through the THP. The time interval over which the measure is made is a parameter of the measurement experiment. The shorter the interval, the more bursty we would expect the measure to be.

9.4.3.2. The measurement should be made with a process generating artificial input as well as a human user at a keyboard generating input.

9.4.3.3. Since throughput may be dependent upon buffer allocation, this value must be recorded also.

9.5. Multiconnection Measurements

9.5.1. Statistics on user commands sent to the local THP.

9.5.2. Statistics of error or success codes returned.

9.5.3. Counter for each control over all letters emitted by the THP and another for letters accepted.

9.5.4. Percentage of data carrying letters.

9.5.5. Overall ratio of text octets to total octets in letters sent and received by the THP.

9.6. Minimum Measurement Facilities

9.6.1. The correct algorithms and parameter settings for flow control and buffer allocation must evolve from experience with THP implementations. Thus it is suggested that some measurement facilities in the THP be regularly used through initial implementation and acceptance testing and into the "production" phase.

9.7. Error Log

9.7.1. The THP should log in a file the protocol errors it detects. The error records should include a time stamp and the

THP record. This log should be examined daily by the computer operator or chief programmer, and errors due to local problems corrected. Errors due to remote problems should be reported to the remote site and to the NCC.

9.7.2. It is especially important to record violations of S/P/T in this error log (or another log). S/P/T violations should be reported to the NCC periodically.

9.8. Debugging Aids

9.8.1. When THP implementations are being tested, either initially or after a modification or parameter change, it is desirable to track the execution of the THP in some detail. In particular it is useful to log the control records in some cases the data for some or all of the messages sent and received by the THP. The THP should be prepared to log such information when a debugging flag is set.

10. REFERENCES

(1) Defense Communications Agency, "Specifications (Type A) for AUTODIN II Phase I," November 1975 (referred to as the AUTODIN II Specification).

(2) J. Postel, L. Garlick, R. Rom, "Transmission Control Protocol,
*** OFFICE-1 GOING DOWN IN " July 1976.60 MINUTES. ***

(3) Network Information Center, "TELNET Protocol Specification," August 1973, NIC Catalog Number 18639.

(4) C. Dornbush, K. Victor, C. Irby, "A Command Meta Language For NLS," January 1975. (SRI-ARC Catalog Number 22130.)

(5) V. Cerf, Y. Dalal, C. Sunshine, "Specification of Internet Transmission Control Program," INWG General Note #72, December 1974 (Revised).

(6) D. Rubin, "TELNET Under Single-Connection TCP Specification," Digital Systems Laboratory Technical Note 76, Stanford University, Stanford, California, February 1976.

(7) Cerf, V. and R. Kahn, "A Protocol for Packet Network Intercommunication," IEEE Transactions on Communication, Vol COM-20, No. 5, May 1974.

(8) C. Sunshine, "Interprocess Communication Protocols for Computer Networks," Digital Systems Laboratory Technical Note 105, Stanford University, Stanford, California, December 1975.

(9) Cerf, V., "ARPA Internetwork Protocols Project Status Report for the Period November 15, 1975 - February 15, 1976", Digital Systems Laboratory Technical Note 83, Stanford University, Stanford, California, 25 February 1976.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

Terminal-to-Host Protocol Specification

Appendices

15-July-76

Jonathan B. Postel
Larry L. Garlick
Raphael Rom

Augmentation Research Center

Stanford Research Institute
Menlo Park, California 94025

(415) 326-6200

LIST OF APPENDICES

APPENDIX A. ADDRESSING

A. Addressing is a general mechanism for routing messages (text, letters, segments, packets, ...) to a destination. The addressing requirements of each entity (user, THP, TCP, PS, ...) may be quite different since the destination entity is almost always different. In the AUTODIN II network, several layers of addressing exist. These layers are kept as invisible as possible to higher level protocols, in hopes of presenting one coherent addressing model to the user. Below, a few of the more important layers of addressing are discussed, in order of closeness to the terminal user.

B. User Addressing

1. A user address must be simple enough to permit destinations to be specified by subscriber ID, when the user is supported by a TAC. Yet it must be powerful enough to address an experimental version of a standard function on some host computer. To this end, a user address has the following form:

Subscriber ID. (16 bits) This field corresponds to a terminal user on a TAC or to a host-entity. It is a required part of the address.

User ID. (12 bits) This field is used for addressing host users or standard host functions and consists of two subfields as follows:

User ID Type FLag. (1 bit) A zero means the User ID Proper is a static ID, such as directory name; a one means the User ID Proper is dynamic, such as a process name or job number.

User ID Proper. (11 bits) Depending on the previous bit, this field may contain a static or dynamic ID. The field will be zero for a TAC subscriber.

Function Suffix. (4 bits) This field is used to select a particular function offered by the user above. It defaults to zero, suggesting that a user's well known address should have a zero Function Suffix.

2. Examples

User B is a terminal user supported by a TAC and always listens for a connection attempt on his default Function Suffix. The address of user B is <subscriber ID = B> <user ID = 0>

<Function Suffix = 0>. This is the simplest address that can be specified.

Suppose host D has a special process, C, that provides an experimental version of the standard function service, E. To access this service, an address like the following will be used:

<subscriber = D><userid = C's static ID><Function Suffix = E>

This represents one of the most complex types of user addressing offered. It demonstrates the use of a static User ID to provide a "well-known" address for a global addressing mechanism.

C. TCP Addressing

1. The TCP addressing scheme is based on sockets. A socket is the concatenation of a TCP identifier and a port, each of which is a 16 bit entity. The conversion of user addresses to TCP addresses is accomplished using the following rules:

(1) The Subscriber ID becomes the TCP identifier if a User ID is provided, i.e. if the Subscriber ID refers to a host.

(2) The User ID, if present, maps directly into the leftmost 12 bits of the port.

(3) The Function Suffix, if present, maps directly into the rightmost 4 bits of the port.

2. The result of these rules is that a TCP identifier is nothing more than a Subscriber ID and is at least enough to route a message to a TCP. In the TAC case, the TCP is able to route the message to the user (end subscriber) with the extra leader added by the PS. Routing messages in the host case, however, requires port information.

D. BSL Address

1. The destination address found in the Binary Segment Leader (BSL) always corresponds exactly to a subscriber address. This address is required and will be taken directly from the address provided by the user. In the TAC case, the address refers to a terminal connected to an access circuit. The destination PS will handle routing to the TAC and the adding of any leaders required for final routing to the user's terminal. For the host case, the address is that of the host's access circuit and cannot be used to address specific users or standard host functions.

APPENDIX B. COMMAND LANGUAGE

A. Connection Control Commands

CONNECT

Performs a fully specified OPEN; parameters must include foreign user address, send precedence, security level (absolute or ceiling), and TCC. If the THP must negotiate initial options, they should be done at this time, in which case the connection will be established at the completion of this command. Otherwise, no network traffic will be generated until the user sends text.

LISTEN

Performs a partially specified OPEN, in which any of the address, precedence, or TCC parameters may be unspecified.

CLOSE

Performs a deferred, graceful close of the connection. All previous text will be delivered to the remote THP prior to closing the connection.

ABORT

Performs an immediate, flushing close of the connection. No guarantee is given about the status of text previously sent on this connection.

SEND

Puts the user in the send-receive mode. This command is used as an escape from the command level. A successful CONNECT or LISTEN will automatically move the connection to the send-receive mode. To get back to the command level from the send-receive mode, a THP-escape is used (see below).

SET-SECURITY

Changes the security level of all new letters sent on this connection.

SYNCH

Causes the THP to send a TCP Interrupt and to insert a THP DM control in the data stream. This allows the user to clear his data path while in the command mode. A Synch character is also available for performing the same operation while in the send-receive mode. The semantics of this command are discussed in Appendix F.

B. Network Virtual Terminal Control

ECHO-MODE

Instructs the THP to set the echo mode of the connection. Parameters must provide for the specification of local or remote echoing.

TERMINAL-TYPE

Since it would be cumbersome for the user to specify all the terminal characteristics in individual THP commands, the terminal-type command is used to allow the specification of a default terminal profile. Examples of terminal types are 2741, TI, and scope. It is beyond the scope of this document to specify what set of terminals should have distinct default profiles.

DUPLEX-MODE

This command allows the user to select full- or half-duplex modes, independently of the TERMINAL-TYPE command. For full-duplex, the THP assumes that the terminal does not perform echoing. For half-duplex, the THP believes the terminal is doing the echoing and thus does neither local echoing nor any negotiating for remote.

LINE-MODE

Line modes include character-at-a-time and line-at-a-time. If the user specifies character-at-a-time mode, the remote scanning mode will be negotiated for stream mode. Character-at-a-time results in the transmission of each character as soon as possible, though not necessarily one per letter.

If the user selects line-at-a-time, the scanning mode will be negotiated for record mode. In line-at-a-time mode the buffer

will be transmitted upon receipt of a end-of-line, buffer overflow, or a "send now" character.

TRANSPARENT

This command disables the interpretation of all the special characters that may be typed in the send-receive mode, except the THP escape character. A user talking to a remote user or process that requires these special characters can be sure that the characters get passed to the remote user or process

[NO] RAISE

Causes lower case alphabetic characters to be transmitted as their upper case equivalents.

[NO] LOWER

Causes upper case alphabetic characters to be transmitted as their lower case equivalents.

SET-ESC-CHAR

Resets the THP ESCAPE character.

SET-SYNCH-CHAR

Resets the SYNCH character.

SET-BRK-CHAR

Resets the BREAK character.

SET-INT-CHAR

Resets the INTERRUPT character.

SET-AO-CHAR

Resets the AO character.

SET-AYT-CHAR

Resets the SYNCH character.

SET-LIT-ESC-CHAR

Resets the LITERAL ESCAPE character.

SET-SEND-NOW-CHAR

Resets the SYNCH character.

C. Data Transmission Control

Interrupt-Process character

Causes the THP IP control to be inserted into the data stream. The remote THP will notify remote process or TC using the interrupt facility provided by the operating system.

Abort-Output character

Causes the THP AO control to be inserted into the data stream.

The remote THP will allow the remote process to (appear to) run to completion, but will not send its output. Rather, it will send a Synch (TCP Interrupt and THP DM) to the user issuing the AO.

Are-You-There character

Causes the THP AYT control to be inserted into the data stream.

The remote THP will send back to the user some visible (i.e., printable) evidence that the AYT was received. For a process, the evidence may be the status of the process. For a user, it should be the user's status as provided by the TC.

Erase Character character

Causes the THP EC control to be inserted into the data stream.

The remote THP will cause deletion of the last preceding undeleted character or "print position" from the data stream. Thus the remote delete character sequence should be inserted into the data stream for the remote user terminal or process.

Erase Line character

Causes the THP EL control to be inserted into the data stream.

The remote THP will cause deletion of characters from the data stream back to, but not including, the last "CR LF" sequence sent over the connection. Thus the remote delete line sequence should be inserted into the data stream for the remote user terminal or process.

BREAK character

Causes the THP BREAK control to be inserted into the data stream. The remote THP should perform the necessary action to simulate the striking of the break key or attention key in the remote environment.

SYNCH character

Cause the THP to send a TCP Interrrupt and to insert a THP DM control in the data stream. This allows the user to clear his data path. The semantics of this command are discussed in Appendix F.

Literal Escape character

Causes the next character to be sent in the data stream without being examined for possible THP control meaning. The Literal Escape character *is not sent*.

THP Escape character

Moves the user into the command mode. In command mode text is interpreted as commands. To enter or reenter the send-receive mode, the user must issue the SEND command. No character is sent on the connection as a result of the receipt of this character.

Send-Now character

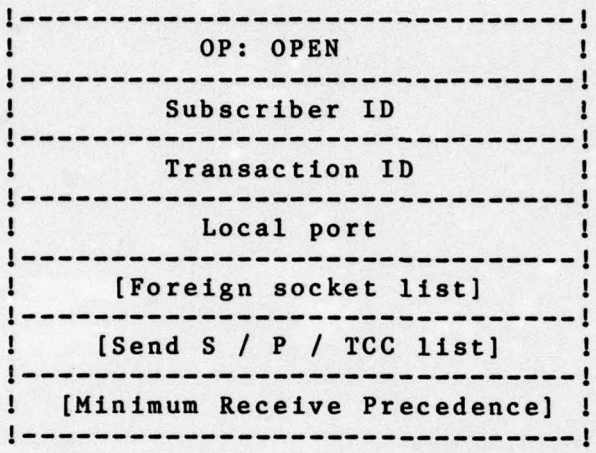
Causes whatever is in the TC or THP buffers to be sent immediately on the connection. This character should be sent only when in record mode (line-at-a-time). The Send-Now character itself is not sent on the connection.

APPENDIX C. THP-TCP EVENTS

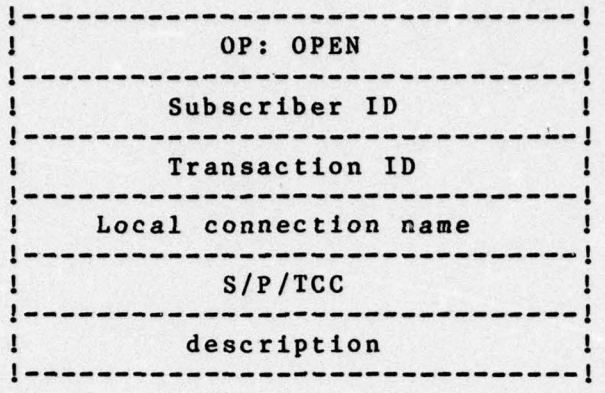
A. Events

1. OPEN

USER => TCP



TCP => USER

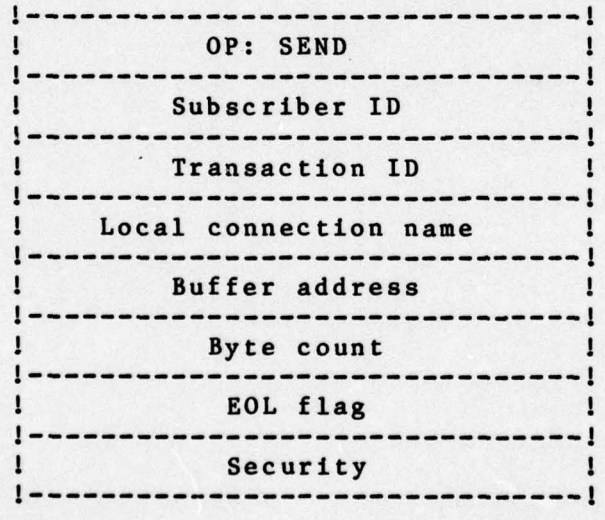


Possible description

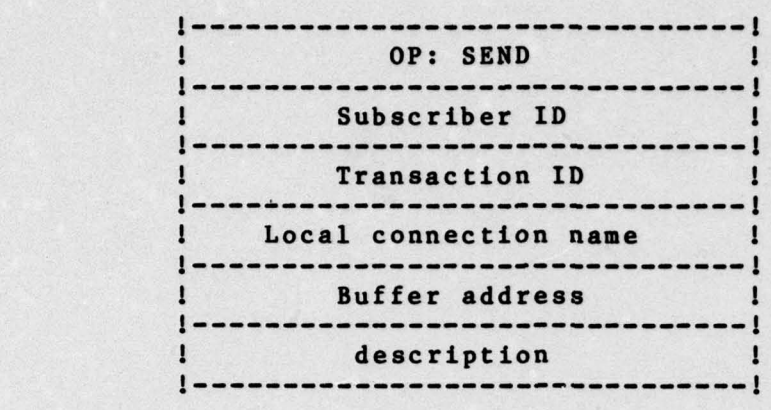
See "Error and TCP-to-User Event Codes."

2. SEND

USER => TCP



TCP => USER

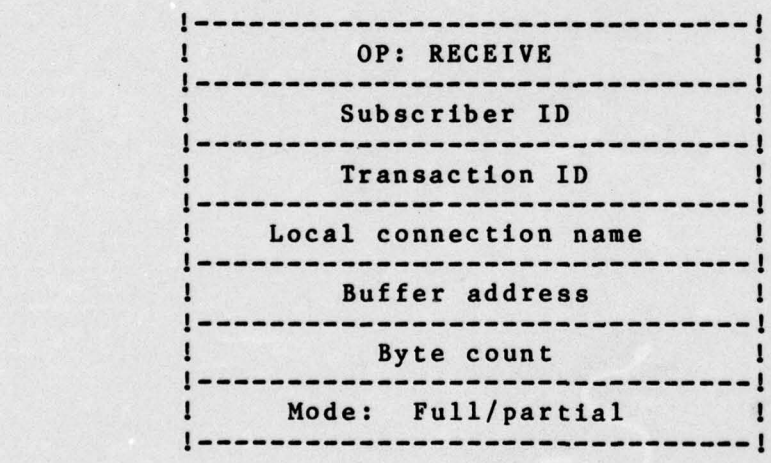


Possible description

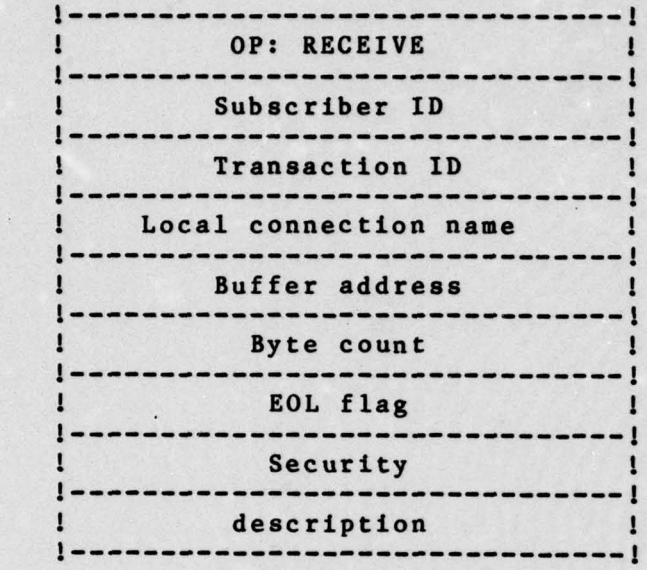
See "Error and TCP-to-User Event Codes."

3. RECEIVE

USER => TCP



TCP => USER

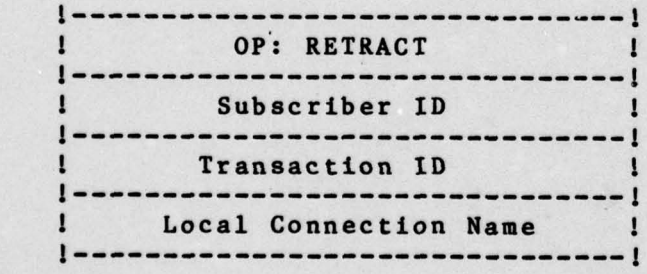


Possible description

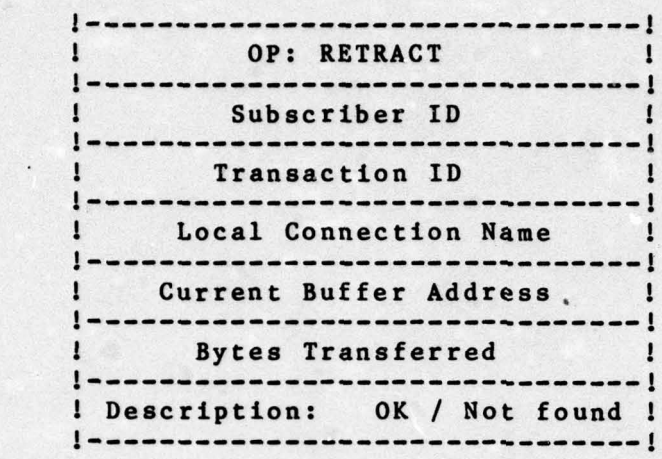
See "Error and TCP-to-User Event Codes."

4. RETRACT (all receive buffers)

THP => TCP

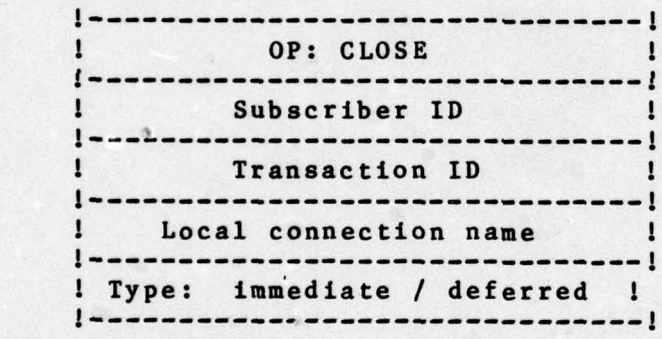


TCP => THP

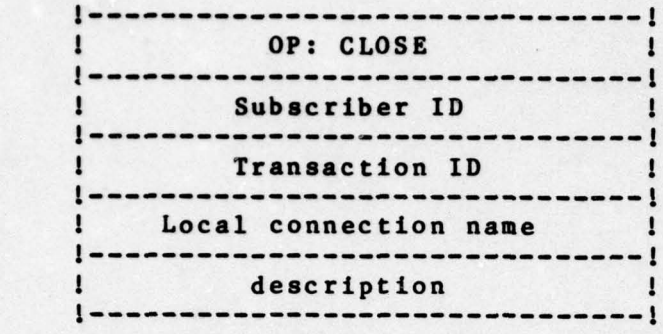


5. CLOSE

USER => TCP



TCP => USER

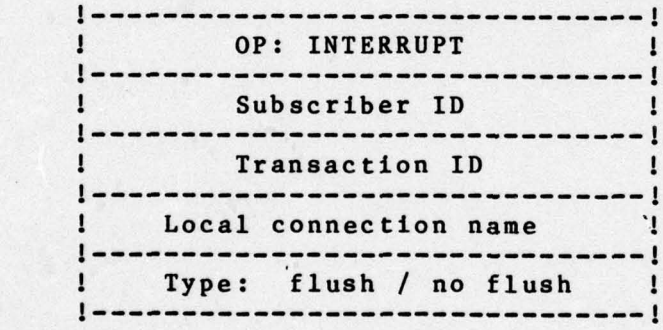


Possible description

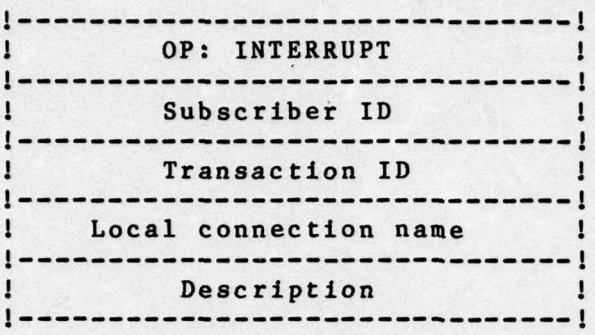
See "Error and TCP-to-User Event Codes."

6. INTERRUPT

USER => TCP



TCP => USER

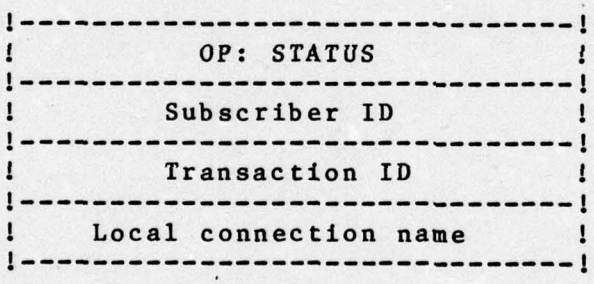


Possible description

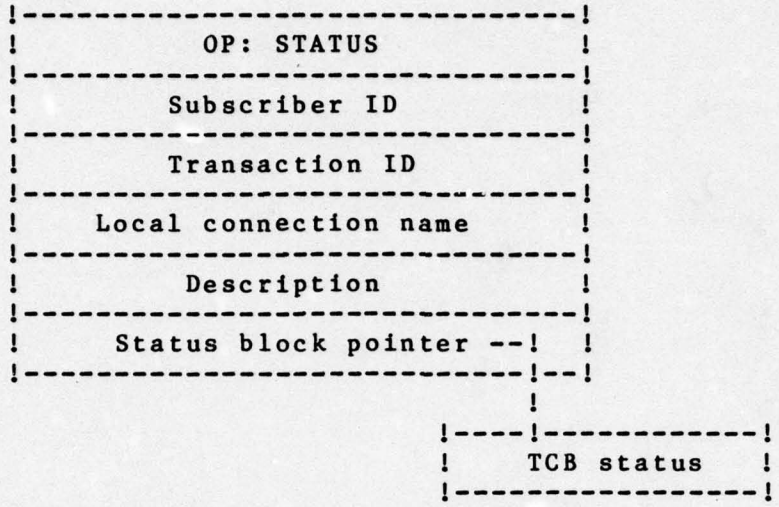
See "Error and TCP-to-User Event Codes."

7. STATUS

THP => TCP

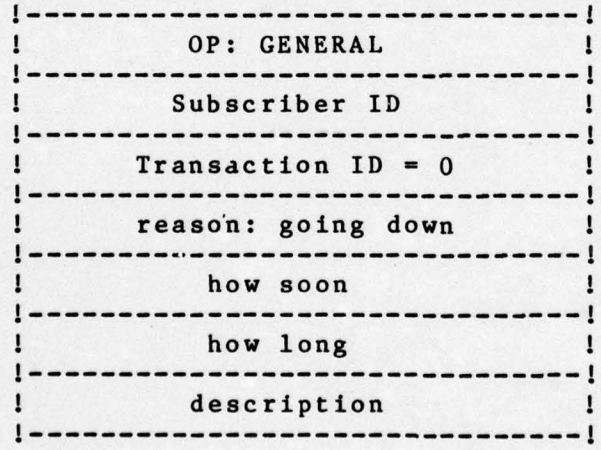


TCP => THP



8. GENERAL

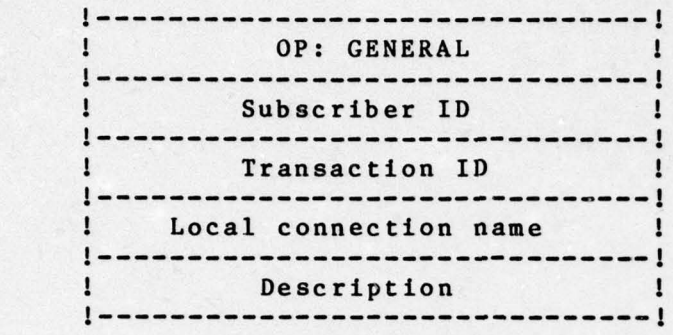
USER => TCP



Possible description

Scheduled maintenance
Panic
Reload
Emergency restart

TCP => USER

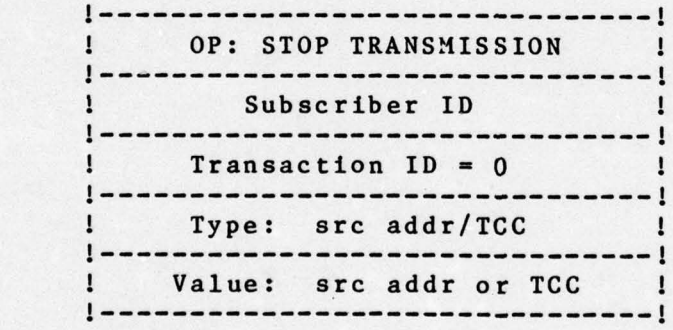


Possible description

See "Error and TCP-to-User Event Codes."

9. STOP TRANSMISSION

THP => TCP



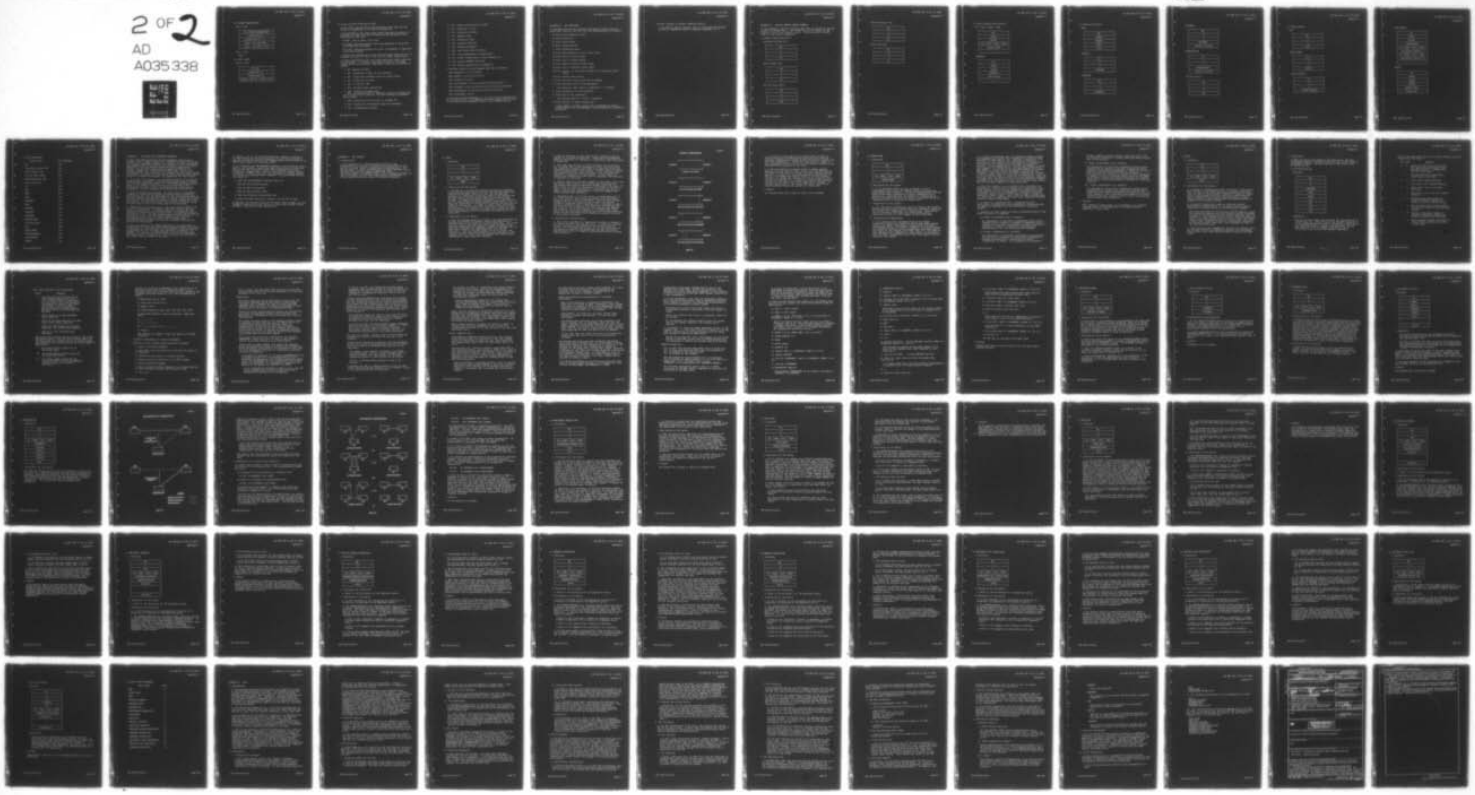
TCP => THP

None.

AD-A035 338 STANFORD RESEARCH INST MENLO PARK CALIF AUGMENTATION --ETC F/G 17/2
TERMINAL-TO-HOST PROTOCOL SPECIFICATION.(U)
JUL 76 J B POSTEL, L L GARLICK, R ROM

UNCLASSIFIED SRI-ARC-35940 NL

2 OF 2
AD
A035 338

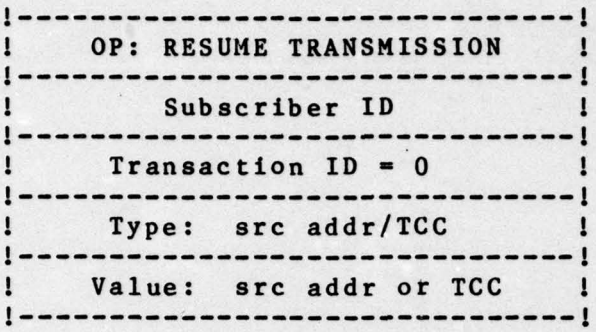


END

DATE
FILMED
3-77

10. RESUME TRANSMISSION

THP => TCP

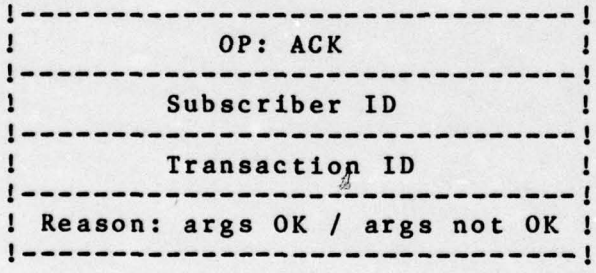


TCP => THP

None.

11. ACK / NACK

THP <=> TCP



B. Error and TCP-to-THP Event Codes

1. The event code specifies the particular event that the TCP wishes to communicate to the THP or foreign TCP.

2. In addition to the event code, three flags may be useful to classify the event into major categories and facilitate event processing by the user:

E flag: set if event is an error

L/F flag: indicates whether event was generated by Local TCP, or Foreign TCP or network

P/T flag: indicates whether the event is Permanent or Temporary [retry may succeed]

3. Events are encoded into 8 bits with the high order bits set to indicate the state of the E, L/F, and P/T flags, respectively.

4. Events specified so far are listed below with their codes and flag settings. A * means a flag does not apply or can take either value for this event. Additional events may be defined as needed.

0	0**	general success
1	ELP	connection illegal for this process
2	0F*	unspecified foreign socket has become bound
3	ELP	connection not open
4	ELT	no room for TCB
5	ELT	foreign socket unspecified
6	ELP	connection already open
	EFP	unacceptable SYN [or SYN/ACK] arrived at foreign TCP.
7	EFP	connection does not exist at foreign TCP
8	EFT	foreign TCP inaccessible [may have subcases]
9	ELT	retransmission timeout

Note: This is not a misprint, the local meaning is different from foreign.

10	E*P	buffer returned due to flush
11	OF*	interrupt to user
12	**P	connection closing
13	E**	general error
14	E*P	connection reset
15	E*P	security violation
16	E*P	illegal precedence
17	E*P	user group (TCC) violation
18	EFP	stop transmission at destination
19	E*P	buffer flushed due to stop transmission
20	OL*	Solicit RECEIVE from user
21	E*P	buffer flushed due to preemption

5. Possible events for each message type are as follows:

Type 0[general]: 2,11,12,14,20

Type 1[open]: 0,1,4,6,13,15,16,17

Type 2[close]: 0,1,3,13

Type 3[interrupt]: 0,1,3,5,7,8,9,12,13,15,16,17,18

Type 10[send]: 0,1,3,5,7,8,9,10,11,12,13,15,16,17,18,19,21

Type 20[receive]: 0,1,3,10,12,13,15,16,17,18,19,21

Type 30[status]: 0,1,13

6. Note that events 6(foreign), 7, 8, and 15-18 are generated at the foreign TCP or in the network, and these same codes are used in the Control Data Extension field of the T-segment header.

APPENDIX D. CMB STRUCTURE

A. The CMB contains the following information (field sizes are approximate only and may vary from one implementation to another):

16 bits: Local connection name

16 bits: Process ID

32 bits: Local socket

32 bits: Foreign socket

4 bits: Connection state

16 bits: Head Pointer to to-net letter list

16 bits: Head of control queue

16 bits: Tail of control queue

16 bits: Head of from-net letter queue

16 bits: Tail of from-net letter queue

16 bits: Pointer to last octet copied out of from-net letter queue

16 bits: Forward CMB pointer

16 bits: UMB back pointer (may not be needed)

4 bits: Security level of the connection

1 bit: Security level type (0 = absolute; 1 = ceiling)

4 bits: Precedence of the connection

8 bits: TCC of the connection

1 bit: Receive mode (0 = full; 1 = partial)

16 bits: Pointer to option states list

[each element includes: option name, performer of option, acknowledged or not acknowledged, accepted/rejected, parameters (if any)]

16 bits: Pointer to current terminal profile

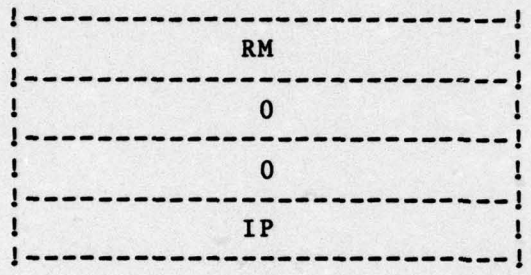
[the profile should include a copy of the profile maintained by the TC, additional special character mappings, etc.]

APPENDIX E. THP-THP CONTROL RECORD FORMATS

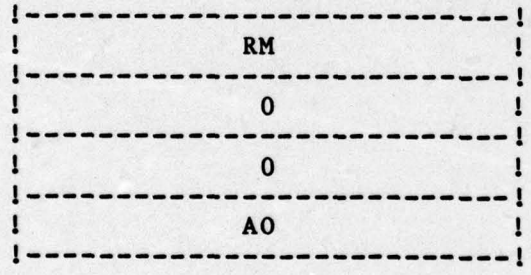
A. This appendix lists the controls that flow as records on the THP connection between THP's. The convention for the format is one octet per line of the control record. The last section assigns codes to the control mneumonics.

B. Common Control Functions

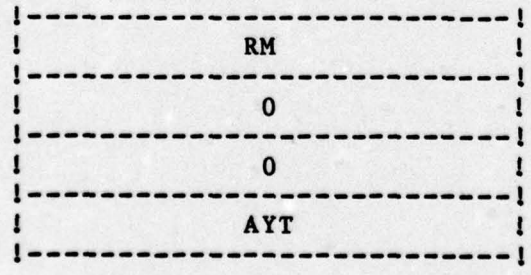
Interrupt Process (IP)



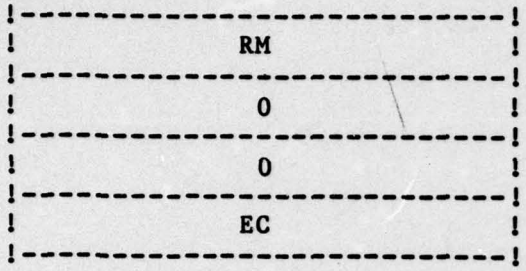
Abort Output (AO)



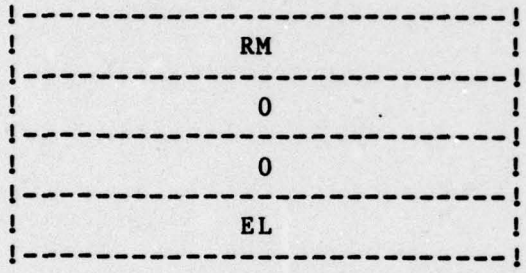
Are You There (AYT)



Erase Character (EC)

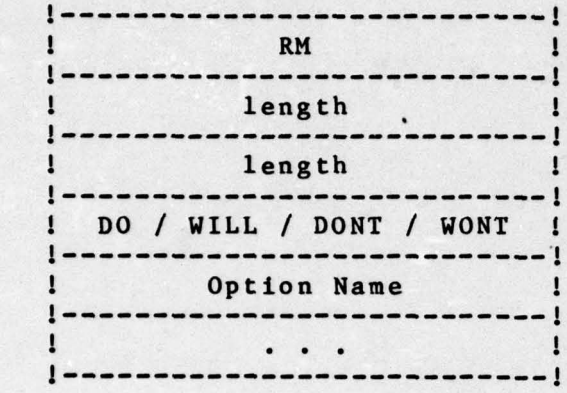


Erase Line (EL)

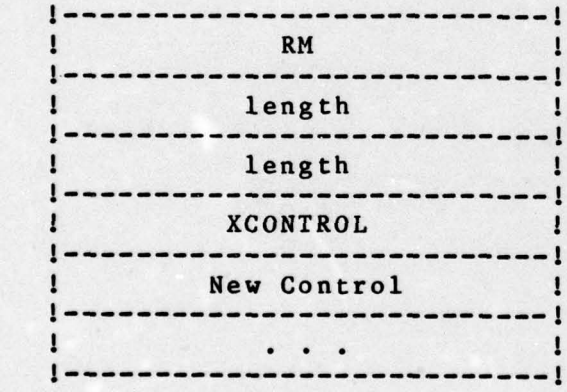


C. Option Negotiation Controls

DO / WILL / DONT / WONT

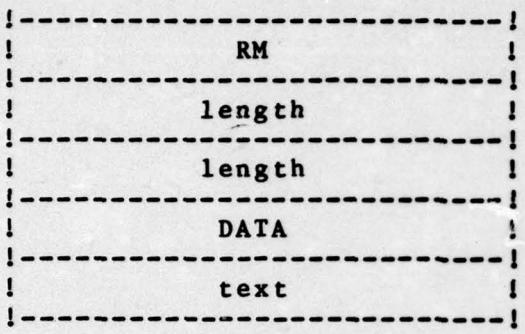


XCONTROL

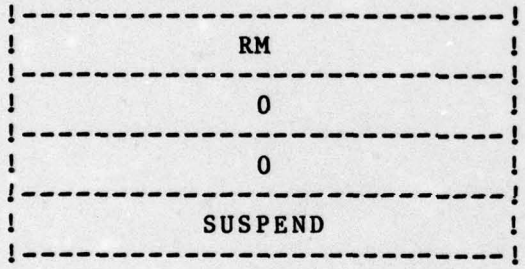


D. Connection Control

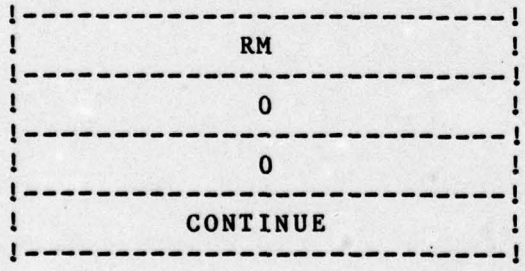
DATA



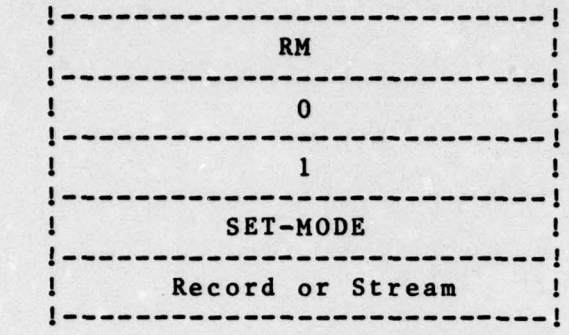
SUSPEND



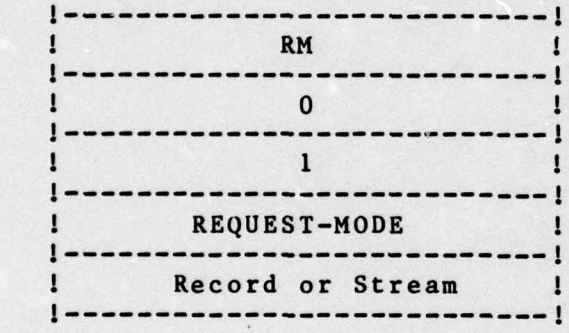
CONTINUE



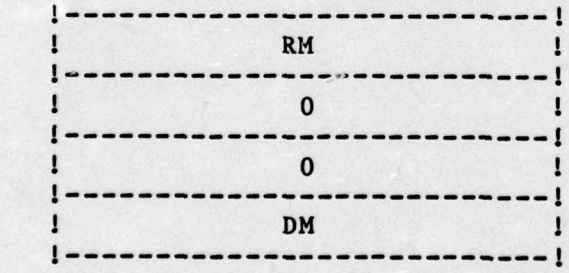
SET-MODE



REQUEST-MODE

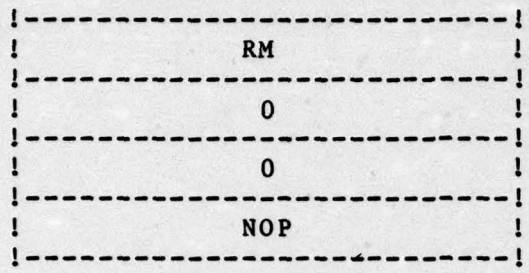


DM (Data Mark)

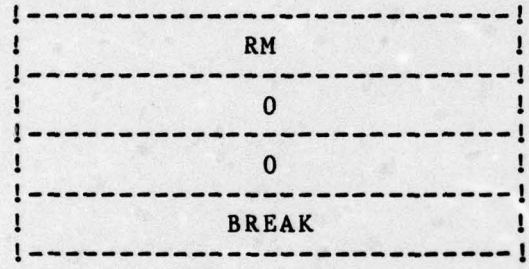


E. Miscellaneous

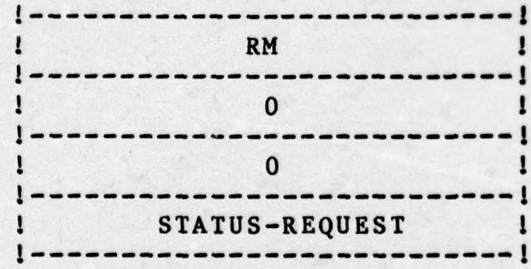
NOP



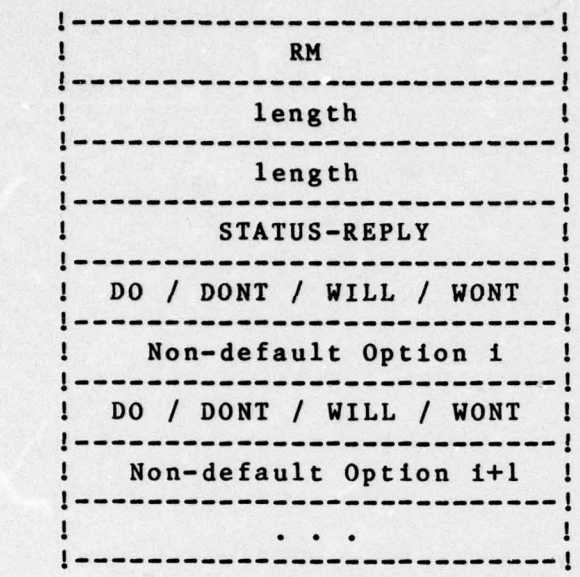
BREAK (BRK)



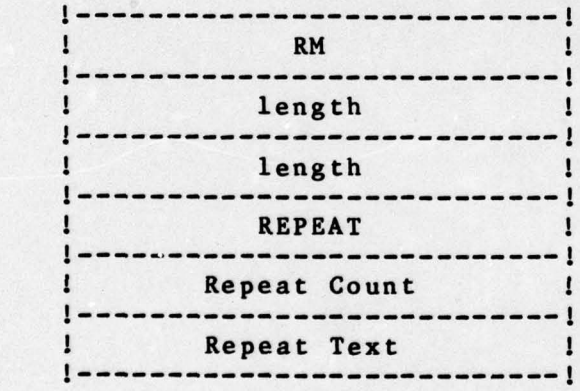
STATUS-REQUEST



STATUS-REPLY



REPEAT



F. Code Assignment

Control Type	Code (decimal)
RM	170
Interrupt Process (IP)	254
Abort Output (AO)	253
Are You There (AYT)	252
Erase Character (EC)	251
Erase Line (EL)	250
DO	249
WILL	248
DONT	247
WONT	246
XCONTROL	245
DATA	244
SUSPEND	243
CONTINUE	242
SET-MODE	241
REQUEST-MODE	240
DM (Data Mark)	239
NOP	238
BREAK (BRK)	237
STATUS-REQUEST	236
STATUS-REPLY	235
REPEAT	234

APPENDIX F. THP SYNCH AND INTERRUPT SEQUENCES

A. Most time-sharing systems provide mechanisms which allow a terminal user to regain control of a "runaway" process; the IP and AO functions described elsewhere are examples of these mechanisms. Such systems, when used locally, have access to all of the signals supplied by the user, whether these are normal characters or special "out of band" signals such as those supplied by the teletype "BREAK" key or the IBM 2741 "ATTN" key. This is not necessarily true when terminals are connected to the system through the network; the network's flow control mechanisms may cause such a signal to be buffered elsewhere, for example in the user's Host.

B. To counter this problem, the THP SYNCH mechanism is introduced. A SYNCH signal consists of a TCP Flush-Interrupt command coupled with the THP command DATA MARK (DM). The Flush-Interrupt command, which is not subject to the flow control pertaining to THP connections, is used to clear the data path and invoke special handling of the data stream by the process which receives it.

C. After receiving the interrupt, the data stream is immediately scanned for "interesting" signals as defined below, discarding intervening data. The THP DM command is the synchronizing mark in the data stream which indicates that any special signal has already occurred and the recipient can return to normal processing of the data stream. The DM is sent after the interrupt, which is enough to insure that it will not arrive ahead of the interrupt.

D. Implementers are warned that in some cases several SYNCH's may be sent in succession. In general, this will require that all letters received prior to a Flush-Interrupt be scanned for THP controls so a DM is not lost in the flush. "Interesting" signals are defined to be: the THP standard representations of IP, AO, and AYT (but not EC or EL); the local analogs of these standard representations (if any); all other THP commands; other site-defined signals which can be acted on without delaying the scan of the data stream.

E. Since one effect of the SYNCH mechanism is the discarding of essentially all characters (except THP commands) between the sender of the SYNCH and its recipient, this mechanism is specified as the standard way to clear the data path when that is desired. For example, if a user at a terminal causes an AO to be transmitted, the server which receives the AO (if it provides that function at all) should return a SYNCH to the user.

F. Finally, just as the TCP Flush-Interrupt command is needed at the THP level as an out-of-band signal, so other protocols which make use of THP may require a THP command which can be viewed as an out-of-band signal at a different level.

G. By convention the THP INTERRUPT sequence is to be used as such a signal. For example, suppose that some other protocol, which uses THP, defines the character string STOP analogously to the THP command AO. Imagine that a user of this protocol wishes a server to process the STOP string, but the connection is blocked because the server is processing other commands. The user should instruct his system to:

1. Send the THP INTERRUPT sequence, that is:

   Send the TCP Flush-Interrupt;

   Send the THP IP character;

   Send the THP Data Mark (DM);

2. Send the character string STOP; and

3. Send the other protocol's analog of the THP DM (if any).

H. The user (or process acting on his behalf) must transmit the THP INTERRUPT sequence of step 1 above to ensure that the THP IP gets through to the server's THP interpreter.

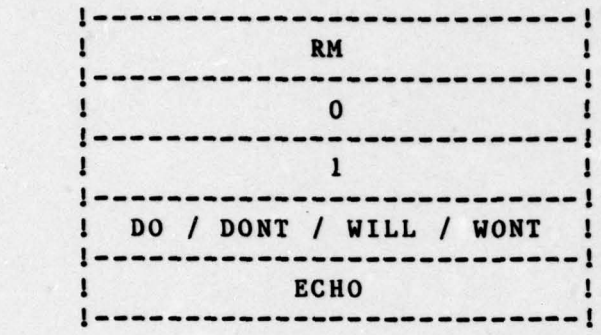
APPENDIX G. THP OPTIONS

A. Introduction

1. This appendix is a detailed specification for the implementation of options, along with the motivations for the options and some related implementation considerations. The options appear in order of importance--the most important first. The first five options (ECHO, BINARY-MODE, RCTE, INCLUDE-GO-AHEAD, and EXTENDED ASCII) are considered necessary as their performance facilitates communication considerably, or provide capabilities that cannot be achieved otherwise.

B. ECHO

1. Structure



2. Motivation for the option.

a. The NVT has a printer and a keyboard which are nominally interconnected (half duplex type) so that "echoes" need never traverse the network; that is to say, the NVT nominally operates in a mode where characters typed on the keyboard are (by some means) locally turned around and printed on the printer. In highly interactive situations it is appropriate for the remote process (command language interpreter, etc.) to which the characters are being sent to control the way they are echoed on the printer. In order to support such interactive situations, it is necessary that there be an option to allow the two ends of the connection to agree that characters typed on an NVT keyboard are to be echoed by the other end of the connection.

3. Description of the option.

a. When the echoing option is in effect, the end performing the echoing is expected to transmit (echo) data characters it receives back to the sender of the data characters. The option does not require that the characters echoed be exactly the characters received (for example, a number of systems echo the ASCII ESC character with something other than the ESC character). When the echoing option is not in effect, the receiver of data characters should not echo them back to the sender; this, of course, does not prevent the receiver from responding to data characters received.

b. The two direction of data flow on the connection need not operate at the same echoing mode. There are five reasonable modes of operation for echoing on a connection, which are shown in Figure G1.

c. If either user desires to echo characters to the other,or for the other THP to echo characters to him, that user gives the appropriate command (WILL ECHO or DO ECHO) and waits (and hopes) for acceptance of the option. If the request to operate the connection in echo mode is refused, then the connection continues to operate in non-echo mode. If the request to operate the connection in echo mode is accepted, the connection is operated in echo mode.

d. After a connection has been changed to echo mode, either end may demand that it revert to non-echo mode by giving the appropriate DONT ECHO or WONT ECHO command (which the other end must confirm to allow the connection to operate in non-echo mode). Just as each direction of a connections may be put in remote echoing mode independently, it must be removed from remote echoing mode separately.

e. Implementations of the echo option, as implementations of all other options, must follow the general loop preventing rules set forth in the main document. Also, so that switches between echo and non-echo mode can be made with minimal confusion (momentary double echoing, etc.), switches in mode of operation should be made at times precisely coordinated with the reception and transmission of echo requests and demands. For instance, if one end responds to a DO ECHO with a WILL ECHO, all data characters received after the DO ECHO should be echoed and the WILL ECHO should immediately precede the first of the echoed characters.

4. Implementation Consideration

a. This option provides the capability to decide on whether or not either end will echo for the other. It does not, however, provide any control over whether or not an end echos for itself; this decision must be left to the sole discretion of the systems at each end (although they may use information regarding the state of "remote" echoing negotiations in making this decision).

NLS Graphics

ECHOING CONFIGURATIONS

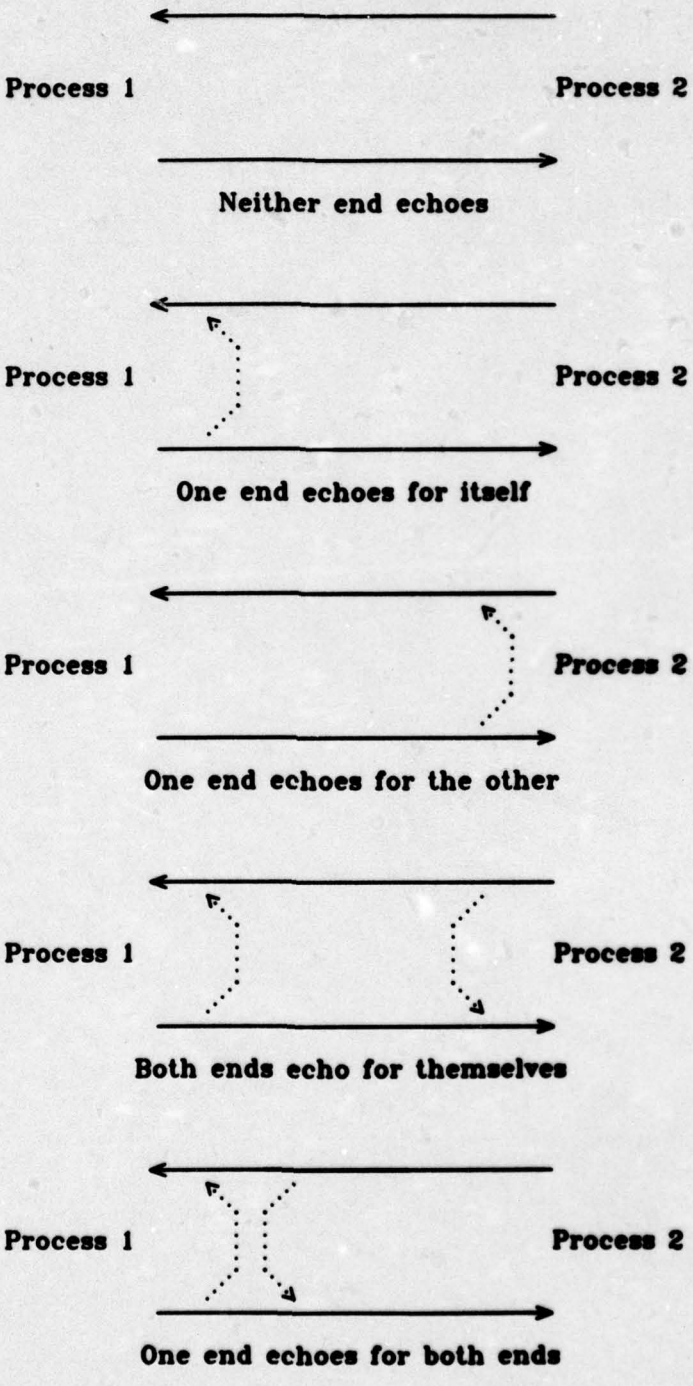


Figure G1

b. It should be noted that if both sides enter the mode of echoing characters transmitted by the other side, then any character transmitted in either direction will be "echoed" back and forth indefinitely. Therefore, care should be taken in each implementation that if one site is echoing, echoing is not permitted to be turned on at the other.

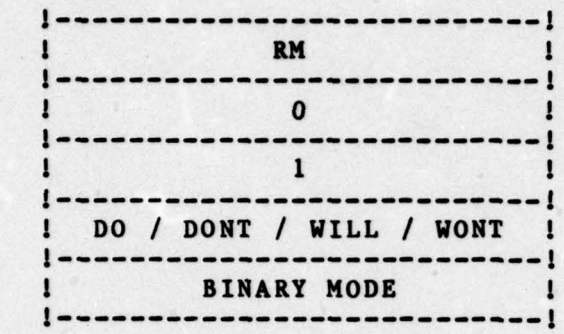
c. A user may explicitly request local or remote echoing. However, the user does not have control over the selection of the module in the local or remote site that actually does the echoing. A request for local echo should be directed by the THP to the TC (which may further direct it to the terminal itself); if neither the terminal nor the TC is able to echo, the THP must provide it. A request for remote echo should be directed by the remote THP to the remote TC or process; if neither can provide it, the remote THP itself can echo but should probably reject the request for remote echo.

5. Default

No echoing (which may or may not imply local echoing).

C. BINARY MODE

1. Structure



2. Motivation for the option.

a. It is sometimes useful to have available a binary transmission path within THP without having to utilize any other higher level protocol. The use of records within the THP protocol provides a mechanism of binary transmission in a natural way, requiring only the addition of a mechanism by which the processes involved can agree to interpret the octets transmitted over a connection as binary data.

3. Description of the option.

a. With the binary transmission option in effect, the receiver should interpret octets received from the transmitter which are not within a THP command as 8 bit binary data. This applies only to records of type DATA in a record mode, and to characters outside of a THP command in the stream mode.

4. Implementation suggestions.

a. It is foreseen that implementations of the binary transmission option will choose to refuse some other options (such as the EBCDIC transmission option) while the binary transmission option is in effect. However, if a pair of Hosts can understand being in binary transmission mode simultaneous with being in, for example, echo mode, then it is all right if they negotiate that combination. Some options in combination with the binary transmission option look very useful, such as negotiate line length.

b. It should be mentioned that the meanings of WONT and DONT are dependent upon whether the connection is presently being operated in binary mode or not. Consider a connection operating in, say, EBCDIC mode which involves a system which has chosen not to implement any knowledge of the binary command. If this system were to receive a DO BINARY, it would not recognize the BINARY option, and therefore would return a WONT BINARY, If the default for the WONT BINARY, were always NVT ASCII, the sender of the DO BINARY, would expect the recipient to have switched to NVT ASCII, whereas the receiver of the DO BINARY, would not make this interpretation. Thus, we have the rule that when a connection is not presently operating in binary mode, the default (i.e., the interpretation of WONT and DONT) is to continue operating in the current mode, whether that is NVT ASCII, EBCDIC, or some other mode.

c. This rule, however, is not applied once a connection is operating in a binary mode (as agreed to by both ends); this would require each end of the connection to maintain a stack, containing all of the encoding-method transitions which had previously occurred on the connection, in order to properly interpret a WONT or DONT. Thus, a WONT or DONT received after the connection is operating in binary mode causes the encoding method to revert to NVT ASCII.

d. It should be remembered that a connection has two directions, one going each way; operating in binary mode must be negotiated separately for each direction, if that is desired.

e. Consider now some issues of binary transmission both to and from a process or a terminal:

(1) Binary transmission from a terminal.

The implementer of the binary transmission option should consider how (or whether) a terminal transmitting over a connection with binary transmission in effect is allowed to generate all eight bit characters, ignoring parity considerations, etc., on input from the terminal.

(2) Binary transmission to a process.

The implementer of the binary transmission option should consider how (or whether) all characters are passed to a process receiving over a connection with binary transmission in effect. As an example of the possible

problem, TENEX intercepts certain characters (e.g., ETX, the Teletype control-C) at monitor level and does not pass them to the process.

(3) Binary transmission from a process.

The implementer of the binary transmission option should consider how (or whether) a process transmitting over a connection with binary transmission in effect is allowed to send all eight-bit characters with no characters intercepted by the monitor and changed to other characters. An example of such a conversion may be found in the TENEX system where certain non-printing characters are normally converted to a Circumflex (uparrow) followed by a printing character.

(4) Binary transmission to a terminal.

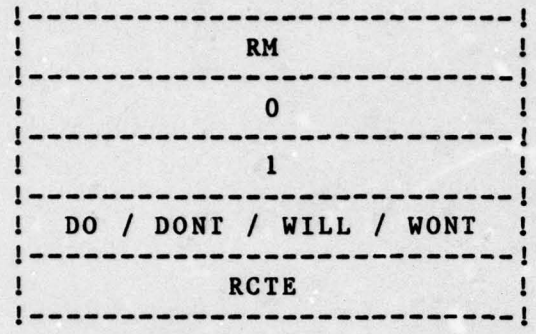
The implementer of the binary transmission option should consider how (or whether) all characters received over a connection with binary transmission in effect are sent to a local terminal. That issue may be the addition of timing characters normally inserted locally, parity calculations, and any normal code conversion.

5. Default

Won't switch to binary mode, if not already in it (continue existing mode), or switching back to NVT ASCII mode if presently in binary mode.

D. RCTE

1. Structure



2. Motivation for the option:

a. A problem of echoing delay might occur when a very distant user accesses a full-duplex host, e.g. through a satellite link. In order to save the many thousands of miles of transit time for each echoed character, while still permitting full server responsiveness and clean terminal output, an echo control similar to that used by some time-sharing systems is suggested for the entire network.

b. An important additional issue is efficient network transmission. Implementation of the remote control echoing scheme will eliminate almost all server-to-user echoing.

The option described in this section involves making a using host carefully regulate the local terminal printer according to explicit instructions from the foreign (serving) host. In addition, it also requests using hosts to buffer a terminal's input to the serving host until it forms a useful unit (with "useful unit" delimited by Break or Transmission characters as described below). Therefore, fewer messages are sent on the user-to-server path.

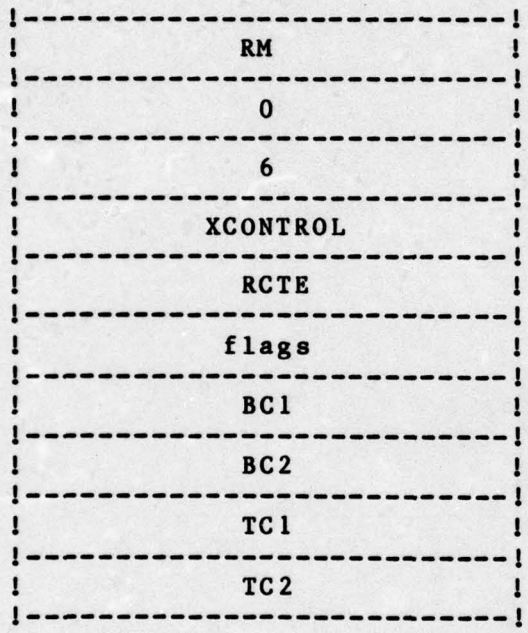
c. This option is only intended for use with full-duplex hosts. The Go-Ahead feature is completely adequate for half-duplex server hosts. Also, RCTE should be used in place of the ECHO option.

3. Description

a. When both hosts have agreed to use RCTE option they have essentially agreed to include an RCTE sub-type in the repertoir of XCONTROL records. The RCTE sub-type is described in the following section.

b. The RCTE sub-type

Structure



Explanation

The bits in the "flags" byte indicate the requested type of control and echo. The two bytes, BC1 and BC2, construct a 16 bit number indicating a set of characters to serve as as the Break Characters. Likewise, the two bytes, TC1 and TC2, construct a 16 bit number indicating a set of characters to serve as as the Transmit Characters.

Bits in the flags byte have the following meaning (bits are counted from the right):

Bit	value	Meaning
0	0	Ignore all other bits in this byte and repeat the last "flags" that was sent. Equals a 'continue what you have been doing'.
	1	Perform actions as indicated by other bits in this byte.
1	0	Print (echo) Break character
	1	Skip (don't echo) Break character
2	0	Print (echo) text up to Break character
	1	Skip (don't echo) text up to Break character
3	0	Continue using same classes of Break characters. (ignore the content of the BC1 and BC2 bytes)
	1	The two 8-bit bytes following this byte contain flags for the new Break classes.
4	0	Continue using same classes of Transmit characters. (ignore the content of the TC1 and TC2 bytes)
	1	Reset Transmit classes according to the two bytes following the Break classes bytes.

Byte value (decimal) and its meaning:

Value	Meaning
0	Continue what you have been doing Even numbers greater than zero (i.e., numbers with the right-most bit off) are in error and should be interpreted as equal to zero. When the "flags" is an even number greater than zero, Classes bytes TC1 & TC2 and/or BC1 & BC2 MUST NOT BE SENT.
1	Print (echo) up to AND INCLUDING Break character
3	Print up to Break character and SKIP (don't echo) Break character
5	Skip text (don't echo) up to Break character, but PRINT Break character
7	Skip up to and including Break character

Add one of the previous non-zero values to one of the following values, to get the total decimal value for the byte (Note that Classes may not be reset, without also resetting the printing action; so an odd number is guaranteed):

8	Set Break classes (using the two bytes [BC1 BC2])
16	Set Transmission classes (using the two bytes [TC1 TC2])
24	Set Break classes (using the two bytes [BC1 BC2]) and the Transmission classes (using the two bytes [TC1 TC2]).

Classes for Break and Transmission (The right-most bit of the second byte (TC2 or BC2) represents Class 1; the left-most bit of the first byte (TC1 or BC1) represents the currently undefined Class 16. Any class combination is legal):

1) Upper-Case Letter (A-Z)

2) Lower-case letters (a-z)

3) Numbers (0-9)

4) Format Effectors (<BS> <CR> <LF> <FF> <HT> <VT>)

5) Non-format effector Control Characters, <DEL> and <ESC>

6) . , ; : ? !

7) { [(< >)] }

8) ' " / \ % @ $ & # + - * = ^ _ | ~

9) <Space>

THP commands are ALWAYS to have the effect of a Break character.

c. Explicit description of control mechanism:

Overview of User Terminal Printing Action & Control

1. Agree to use RCTE commands

2. User holds echo printing until instructed by server to do otherwise

3. Server may send output to terminal printer.

4. Network output is printed up to an RCTE command

5. Server sends RCTE command

6. User acts upon the RCTE command up to a Break character or until receipt of output from the server host.

7. Go to (2)

Note: Output from the server host may occur at any time, in which case, the flow of control switches to (2) and then proceeds to (3), (4), etc.

Explanation:

Both Hosts agree to use the RCTE option. After that, the using host (DO RCTE) merely acts upon the Controlling (serving) host's commands and does not issue any RCTE commands unless and until it (using host) decides to stop allowing use of the option (by sending DONT RCTE).

Using host begins synchronization between the serving host and itself by suspending terminal echo printing until directed to do otherwise by the controlling host, thru an RCTE command.

The server may send output to the terminal printer, either in response to input from the user (in which case it is already synchronized with the terminal input) or spontaneously. In the latter case, flow of control automatically switches to (2) and continues from there. Output from the server is defined as completed when step (5) occurs. That is, text from the Server to the terminal printer MUST end with an RCTE command.

Any output from the server is printed on the terminal IMMEDIATELY. Again note that the end of such output is defined to be the occurrence of an RCTE command.

Server sends an RCTE command. The command may redefine Break and Transmission classes, Action to be performed on Break characters, and action to be performed on text. Each of these independent functions is controlled by separate bits in the flags byte.

A Transmission character is one which RECOMMENDS that the Using Host transmit all text accumulated up to and including its occurrence. (For network efficiency, Using hosts are DISCOURAGED (but not prohibited) from sending before the occurrence of a Transmission character, as defined at the moment the character is typed).

If the Transmission Classes bit (Bit 4) is on, the two bytes following the two Break Classes bytes will indicate what classes are to be enabled.

If the Bit is OFF, the Transmission classes remain unchanged. When the RCTE option is first initiated, NO CLASSES are in effect. That is, no character will be considered a Transmission character. (As if both TC1 and TC2 are zero.)

A Break character REQUIRES that the Using host transmit all text accumulated up to and including its occurrence and also causes the Using host to stop its print/discard action upon the User's input text, until directed to do otherwise by another RCTE command from the Serving host. Break characters therefore define printing units. "Break character" as used in this appendix does NOT mean THP Break character.

If the Break Classes bit (Bit 3) is on, the two bytes following the flags will indicate what classes are to be enabled. There are currently nine (9) classes defined, with room left for expansion.

If the bit is OFF, the Break classes remain unchanged. When the RCTE option is initiated, CLASSES 4, 5, and 9 are to be in effect. That is, Format Effectors, Non-format effector Control Characters and DEL, and Punctuation characters are to be Break characters.

The list of character classes, used to define Break and Transmission classes, are listed at the end of this document.

Because Break characters are special, the print/discard action that should be performed upon them is not always the same as should be performed upon the rest of the input text.

For example, while typing a filename to the TENEX operating system, the text of the filename should be printed (echoed), but the <escape> (if the name completion feature is used) should not be printed.

If Bit 1 is ON The Break character is NOT to be printed.

A separate bit (Bit 2) signals whether or not the text itself should be printed (echoed) to the terminal. If Bit 2 = 0, then the text IS to be printed.

Yet another bit (Bit 0 - right-most bit) signals whether or not any of the other bits of the command should be checked. If this bit is OFF, then the command should be interpreted to mean "continue whatever echoing strategy you have been following, using the same Break and Transmission classes."

This is particularly useful for the command that follows spontaneously generated output from the Serving host (such as "System Going Down") which needs to signal End-of-Message, but does not usually want to reset any other conditions.

Input from the terminal is (hopefully) buffered into units ending with a Transmission or Break character; and echoing of input text is suspended after the occurrence of a Break Character and until receipt of a Break Reset command from the Serving host. The most recent RCTE Break reset command determines the Break actions.

When a Break character is typed, the cycle of control is complete and action re-commences at (2). Action also automatically switches to (2) upon receipt of any text from the Server host.

Notes, Comments, Etc.:

Even-Numbered Commands, greater than zero, are in error, since they will have the low-order bit off. The command should be interpreted as equal to zero, which means that any Classes Reset bytes ([TC1 TC2] [BC1 BC2]) will be in error.

Serving hosts will generally instruct Using hosts NOT to echo Break Characters, even though it might be all right to echo most Break characters. For example, <cr> is usually a safe character to echo but <esc> is not. TENEX Exec is willing to accept either, during filename specification. Therefore, the using host must be instructed NOT to echo ANY Break Characters.

This is generally a tolerable problem, since the serving host has to send an RCTE command at this point, anyhow. Adding the Break character to the message (so that it appears to be echoed) will not cause any extra Network traffic.

The RCTE Option entails a rather large overhead. In a true character-at-a-time situation, this overhead is not justified. But on the average, it should result in significant savings in Network traffic.

Buffering Problems and Transmission vs. Printing Constraints:

There are NO mandatory transmission constraints. The Using host is allowed to send a character a time, though this would be a waste of RCTE. The Transmission Classes commands are GUIDELINES, so deviating from them, as when the User's buffer gets full, is allowed.

Additionally, the Using host may send a Break Class character, without knowing that it is one (as with type-ahead).

The problem with buffering occurs when printing on the user's terminal must be suspended, after the user has typed a currently valid Break Character and until a Break Reset command is received from the serving host. During this time, the user may be typing merrily along. The text being typed may be SENT, but may not yet be PRINTED.

In any case, when the buffer does fill and further text typed by the user will be lost, the user should be notified.

The Serving and Using hosts must carefully synchronize Break Class Reset commands with the transmission of Break characters. Except at the beginning of an interaction, the Serving host MAY ONLY send a Break Reset command in response to the Using host's having sent a Break character as defined at that time. This should establish a one-to-one correspondence between them. (A flag value of zero, in this context, is interpreted as a Break Classes reset to the same class(es) as before.) The Reset command may be preceded by terminal output.

Text should be buffered by the Using host until the user types a character which belongs to the transmission class in force at THE MOMENT THE CHARACTER IS TYPED.

Transmission Class Reset commands may be sent by the Serving host at ANY TIME. If they are frequently sent separate from Break Class Reset commands, it will probably be better to exit from RCTE and enter regular character at a time transmission.

It is not immediately clear what the Using host should do with currently buffered text, when a Transmission Classes Reset command is received. The buffering is according to the previous Transmission Classes scheme.

The Using host clearly should NOT simply wait until a Transmission character (according to the new scheme) is typed.

Either the buffered text should be rescanned, under the new scheme;

Or the buffered text should simply be sent as a group. This is the simpler approach, and probably quite adequate.

It is possible to define NO BREAK CHARACTERS except the THP commands (RM ...). This might actually be useful, as in the case of transmitting on carriage-return, with the Using host echoing (a controlled half-duplex).

Having the serving host send a THP command will allow the server to know when he may reset the Break classes, but the mechanism is awkward and probably should be avoided.

Sample Interaction:

"S:" is sent from Serving (WILL RCTE) host to Using host.
"U:" is sent from Using (DO RCTE) host to Serving host.
"T:" is entered by the terminal user.
"P:" is printed on the terminal.

Text surrounded by square brackets ([]) is commentary.
Text surrounded by angle brackets (<>) is to be taken as a single unit. E.g., carriage return is <cr>.
<XCONTROL> is an abbreviation for <EXTENDED CONTROL>.

The following interaction shows a Login to a TENEX, initiation of the EDIT editor, insertion of some text and the return to the Exec level.

An attempt has been made to give some flavor of the asynchrony of Network I/O and the user's terminal input. Many other possible combinations, using the same set of actions listed below, could be devised. The actual order of events will depend upon network and hosts' load and the user's typing speed.

A connection has already been opened, but the TENEX prompt has not yet been issued. The hosts first discuss using the RCTE option:

S: <RM> 0 2 <WILL> <RCTE>

U: <RM> 0 2 <DO> <RCTE>

S: TENEX 1.31.18, TENEX EXEC 1.50.2 <cr><lf>@ <RM> 0 6 <XCONTROL> <RCTE> 11 1 24 0 0

[Print the Herald and echo input text up to a Break Character, but do not echo the Break Character. Classes 4 (Format Effectors), 5 (Non-format effector Controls and <DEL>), and 9 (<space>) act as Break Characters. No transmission characters are set.]

P: TENEX 1.31.18, TENEX EXEC 1.50.2 <cr><lf>@

T: LOGIN AUTODIN <cr>

P: LOGIN

U: LOGIN <space>

U: AUTODIN <cr>

S: <space> <RM> 0 6 <XCONTROL> <RCTE> 0 0 0 0 0

P: <space> AUTODIN

S: <cr><lf> (PASSWORD): <RM> 0 6 <XCONTROL> <RCTE> 7 0 0 0 0

P: <cr><lf> (PASSWORD):

T: WASHINGTON 1000<cr>

[The password "WASHINGTON" is not echoed. Printing of "1000<cr>" is withheld]

U: WASHINGTON <space>

U: 1000<cr>

S: <space> <RM> 0 6 <XCONTROL> <RCTE> 3 0 0 0 0

S: <cr><lf> JOB 17 ON TTY41 7-JUN-73 14:13 <cr><lf>@ <RM> 0 6 <XCONTROL> <RCTE> 0 0 0 0 0

P: <space> 1000

[Printing is slow at this point; so the account number is not printed as soon as the server's command for it is received.]

P: <cr><lf> JOB 17 ON TTY41 7-JUN-76 14:13 <cr><lf>@

T: EDIT <esc><cr>

P: EDIT

U: EDIT<esc>

S: .SAV;1 <RM> 0 6 <XCONTROL> <RCTE> 0 0 0 0 0

P: .SAV;1

U: <cr>

S: <cr><lf><lf> Edit 3/14/73 DRO,KRK <cr><lf>: <RM> 0 6 <XCONTROL> <RCTE> 15 1 255 0 0

[The program is started and the EDIT prompt ":" is sent. At the command level, EDIT responds to every character.]

P: <cr><lf><lf> EDIT 3/14/73 DRO,KRK <cr><lf>:

T: IThis is a test line.<cr> This is another test line.<^Z> Q

["I" means Insert Text. The text follows, terminated by a Control-Z. The "Q" instructs EDIT to Quit.]

U: I

U: This is a test line.<cr>

S: I<cr><lf>* <RM> 0 6 <XCONTROL> <RCTE> 11 0 24 0 0

[EDIT prompts the user, during text input, with an asterisk at the beginning of every line.]

P: I<cr><lf> *This is a test line.

S: <cr><lf>* <RM> 0 6 <XCONTROL> <RCTE> 0 0 0 0 0

P: <cr><lf>* This is another test line.

U: This is another test line.<^Z>

U: Q

[Note that the "Q" will not immediately be printed on the terminal, since it is a Break character.]

S: ^Z<cr><lf>: <RM> 0 6 <XCONTROL> <RCTE> 15 1 255 0 0

[The returned "^Z" is two characters, not the ASCII Control-Z.]

S: Q<cr><lf>@ <RM> 0 6 <XCONTROL> <RCTE> 11 1 24 0 0

P: Q<cr><lf> @

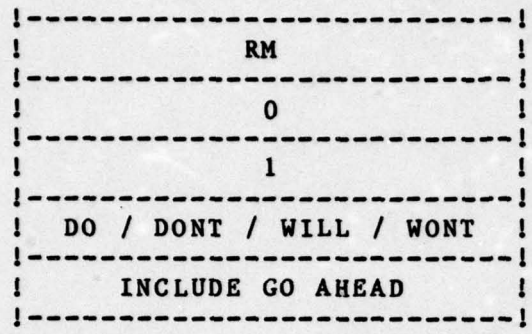
And the user is returned to the Exec level.

4. Default

Neither host asserts special control over the other host's terminal printer.

E. INCLUDE-GO-AHEAD

1. Structure



2. Motivation for the option.

a. While most of the terminals in use nowadays do not need (and do not have) a keyboard locking mechanism, there is no reason why terminals that do have that mechanism should not make use of it, if the process at the other end can support it. It is therefore desirable that the THP have an option with which processes involved can agree that one or the other or both include a GO-AHEAD signal in their transmission.

3. Description of the option.

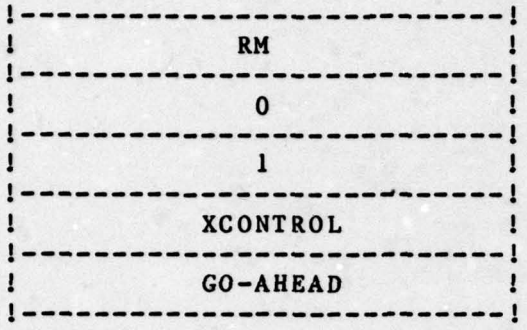
a. When both hosts have agreed to use INCLUDE-GO-AHEAD option they have essentially agreed to include a GO-AHEAD sub-type in the repertoir of XCONTROL records. The GO-AHEAD sub-type is described in the following section.

b. When the INCLUDE-GO-AHEAD option is in effect on the connection between a sender of data and the receiver of the data, the sender HAS TO transmit GA's.

c. Since including GA's applies only to one direction of the connection, it must be negotiated for each direction separately, if it is desired that GA's are included in both directions.

d. The GO-AHEAD sub-type

Structure



4. Implementation considerations.

a. As the INCLUDE-GO-AHEAD option is sort of a line-at-a-time mode, the sender of data which is including GO AHEADs should attempt to transmit characters in the record mode, consistent with any other agreements which are in effect (e.g., approximate transmit record size agreements).

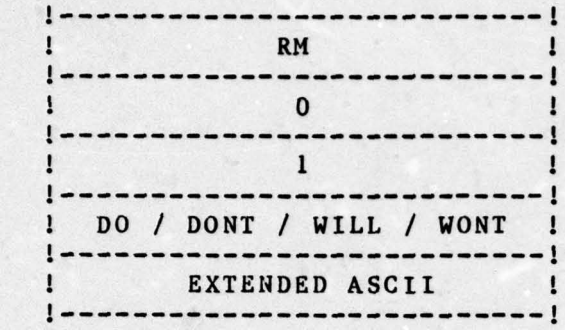
b. In many implementations it will be desirable to couple the INCLUDE-GO-AHEAD option to the echo option so that when the echo option is in effect, the INCLUDE-GO-AHEAD option is not in effect.

5. Default

GO-AHEADs are not included.

F. EXTENDED ASCII

1. Structure



2. Motivation for the option

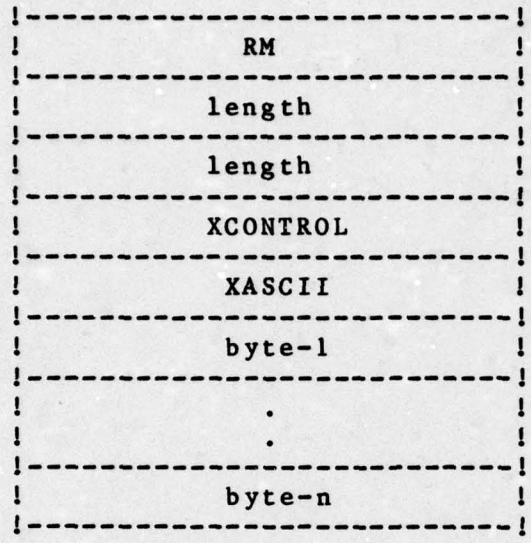
a. Several sites on the net may use keyboards which use almost all 128 characters as printable characters, and use one or more additional 'control' bits as command modifiers or to separate textual input from command input to programs. Without these additional bits, several characters cannot be entered as text because they are used for control purposes, such as the greek letter 'beta' which on a THP connection is Control-C and is sometimes used for stopping ones job. In addition there are several programs that require these additional bits to run effectively. Hence it is necessary to provide some means of sending characters larger than 8 bits wide. This option is to allow the transmission of such extended ASCII characters.

3. Description of the option.

a. When both hosts have agreed to use the Extended ASCII (XASCII) option they have essentially agreed to include an XASCII sub-type in the repertoir of XCONTROL records. The XASCII sub-type is described in the following section.

b. The XASCII sub-type

Structure



Explanation

This record indicates that the receiving end should interpret the bytes following the XASCII byte as beeing in the extended ASCII code.

The body is divided into pairs of bytes, the first being the 8 high-order bits and the second the 8 low-order bits; together they construct a 16 bit byte to be interpreted as a single extended ASCII character.

c. Experience has shown that most of the time, regular 7-bit ASCII is typed, with an occasional 'control' character used. Hence, it is expected that normal NVT ASCII will be used for 7-bit ASCII and that only when extended ASCII text must be sent will the XASCII record be used.

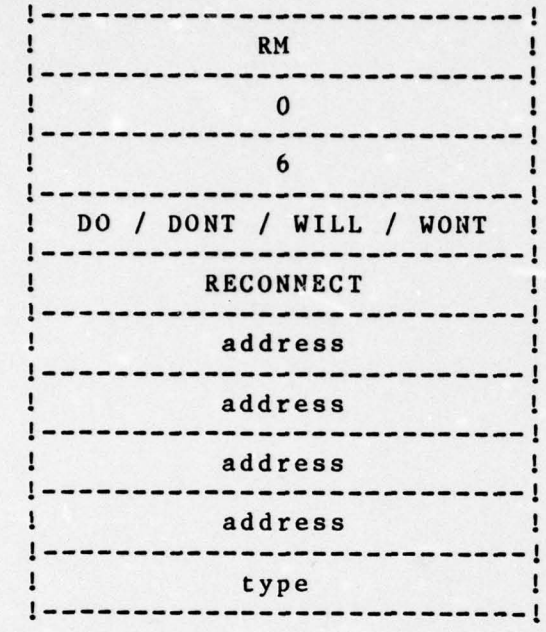
d. The exact meaning of these additional bits depends on the user program and will not be specified here.

4. Default

No Extended ASCII records are allowed.

G. RECONNECTION

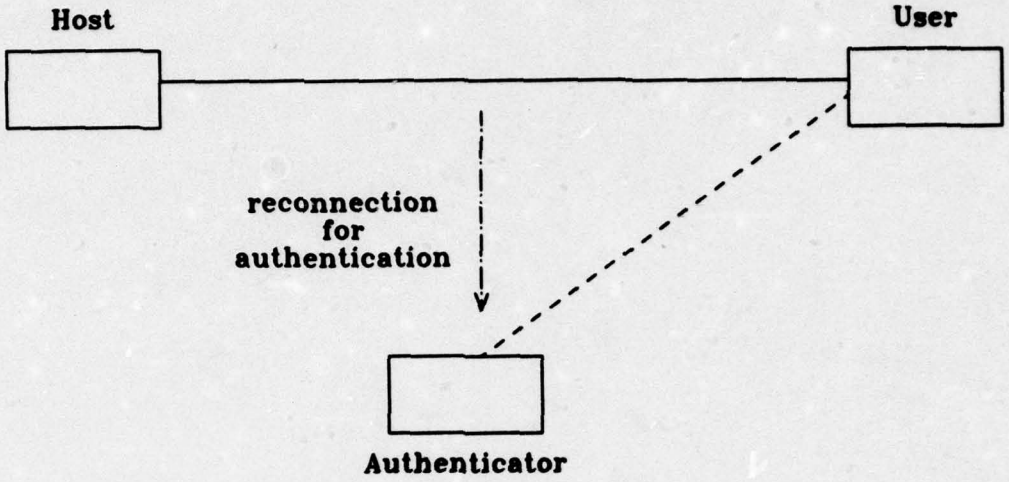
1. Structure

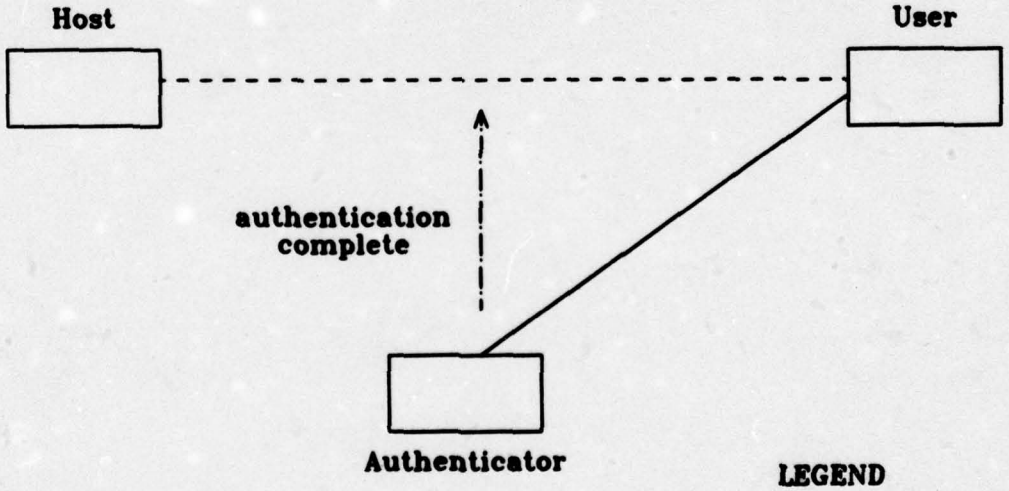


2. Motivation for the option.

a. There are situations in which it is desirable to move one or both ends of a communication path from one user or process to another. Users who are served by the the same TCP may use the MOVE CONNECTION mechanism (see TCP specifications). However, to move the connection across hosts is somewhat more complicated. The reconnection option provides such a mechanism.

RECONNECTION FOR AUTHENTICATION





LEGEND

Existing situation ———

Desired situation - - - - - -

Reconnection —·—·—

Figure G2

Imagine a scenario in which a user could use the same name and password (and perhaps account) to log into any server on the network. For reasons of security and economy it would be undesirable to have every name and password stored at every site. A user wanting to use a host that doesn't have his name or password locally would connect to it and attempt to log in as usual (see Figure G2). The host, discovering that it doesn't know the user, would hand him off to a network authentication service which can determine whether the user is who he claims to be. If the user passes the authentication test he can be handed back to the host which can then provide him service

If the user doesn't trust the host and is afraid that it might read his password rather than pass him off to the Authenticator he could connect directly to the authentication service. After authentication, the Authenticator can pass him off to the host.

The idea is that the shuffling of the user back and forth between host and Authenticator should be invisible to the user.

3. Description of a reconnection mechanism.

a. Assume that H1 wants to move its end of communication path A-C to port D at H3 (Figure G3-a). The reconnection proceeds by steps:

H1 arranges for the reconnection by sending to H2:

H1->H2: DO RECONNECT H3-D START

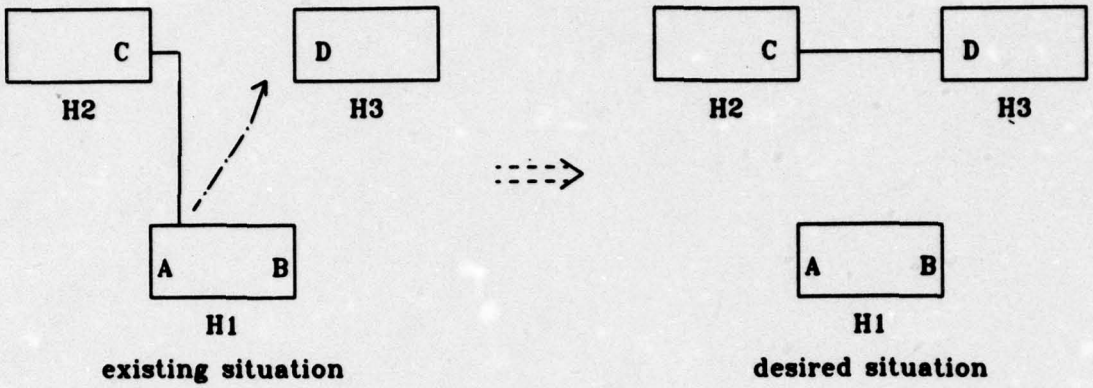
H2 agrees to reconnect and acknowledges with:

H2->H1: WILL RECONNECT H3-D START

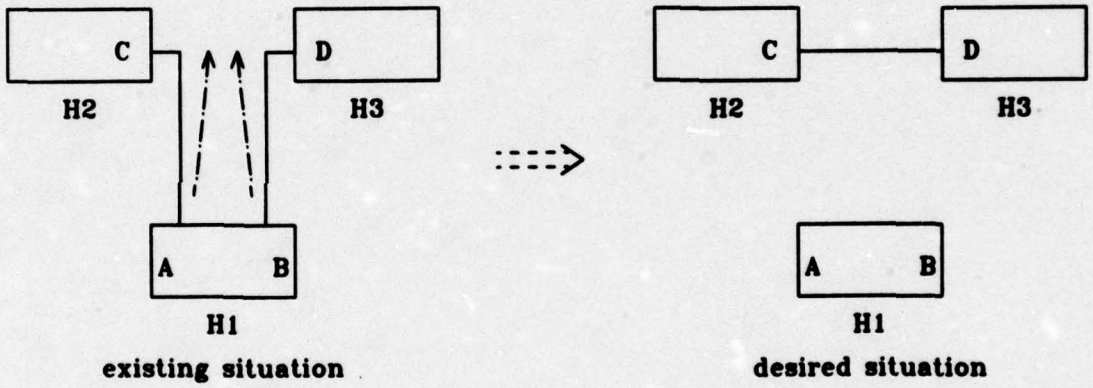
H1 notes that H2 has agreed to reconnect and closes the connection between A and C. H2, after sending the WILL, initiates a connection to H3-D.

In order for the reconnection to succeed H1 must, of course, have arranged for H3's cooperation. One way H1 could do this would be to establish the path B-D and then proceed through the reconnection protocol exchange with H3 concurrently with its exchange with H2 (See Figure G3-b). The dialogue between H1 and H3 will be:

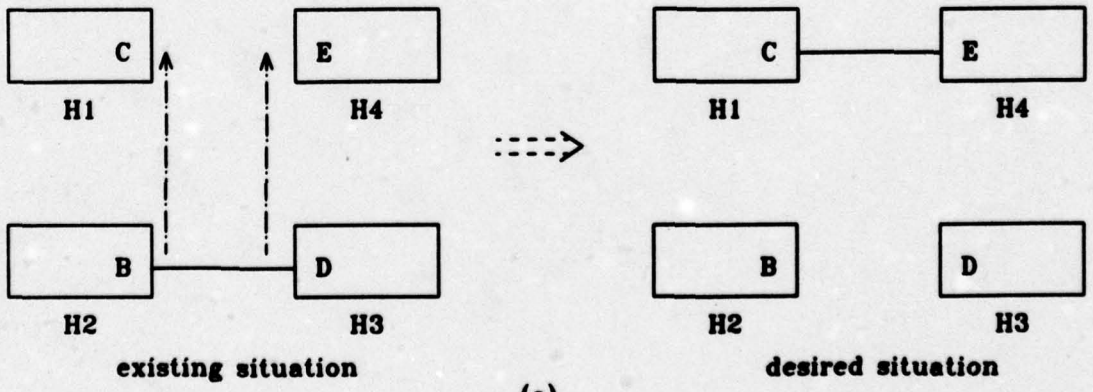
RECONNECTION CONFIGURATIONS



(a)



(b)



(c)

Figure G3

H1->H3: DO RECONNECT H2-C EXPECT

H3->H1: WILL RECONNECT H2-C EXPECT

H1 notified H3 to expect a connection from H2-C. Upon H3's agreement (WILL) he should start LISTENing for a connection from H2-C. H1 will close the connection between him and H3 only upon a receipt of a WILL from H2. This is done so that H1 may inform H3 (via a DONT) in case H2 refuses to perform the reconnection.

b. Either of the THP's may refuse or abort reconnection. H2 could respond to H1's DO with WONT; H1 can abort the reconnection by responding to WILL with DONT.

c. It is easy to insure that messages in transit are not lost during the reconnection. Receipt of the WILL message by H1 is taken to mean that no further messages are coming from H2; similarly receipt of DO from H1 by H2 is taken to mean that no further messages are coming from H1.

d. To complete the specification of the reconnection mechanism consider the situation in which two "adjacent" entities initiate reconnections. The situation is described in Figure G3-c. H2 and H3 simultaneously try to arrange for reconnection:

H2->H3: DO RECONNECT H1-C START/EXPECT

H3->H2: DO RECONNECT H4-E START/EXPECT

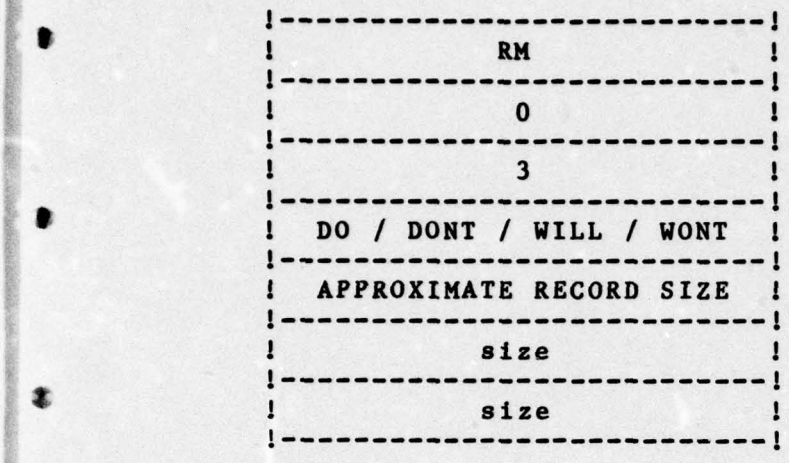
e. Thus, H2 sees a DO from H3 rather than an WILL or WONT in response to its DO to H3. This "race" situation can be resolved by having the reconnections proceed in series rather than in parallel: first one entity (say H2) performs its reconnect and then the other (H3) performs its reconnect. There are several means that could be used to decide which gets to go first. Perhaps the simplest is to base the decision on sockets: the entity for which the number formed by the 32 bit socket number is largest gets to go first. Once an ordering has been determined the reconnection proceeds as though no conflict occurred.

4. Default

No reconnection is allowed.

H. APPROXIMATE RECORD SIZE

1. Structure



2. Motivation for the option.

a. The THP protocol does not specify how many characters the transmitter of data should attempt to pack into the records or letters it sends, except for the limit of 2**15 characters per record. However, (1) some receivers may prefer incoming records to generally have some minimum size, for example, to lessen the burden of processing input interrupts; (2) some receivers may prefer incoming data records to generally have some maximum size, for example, to promote more efficiently utilization of input buffer space; (3) some transmitters may have maximum sizes for outgoing data messages, information which could be used to more efficiently utilize the receiver's input buffer space; and (4) some transmitters may desire to transmit some minimum size message, for example, to lessen the burden of processing output interrupts.

b. Therefore, it is desirable to have some mechanism whereby the THP's involved can attempt to agree on the approximate size of records transmitted over the connection. (It might be even more powerful to be able to negotiate approximate or even exact upper and lower bounds on message size. However, fixed bounds would sometimes be hard to manage, and specifying both upper and lower bounds, even approximately, seems overly complicated considering the expected payoff.)

c. It should be noticed that the approximate record size applies to one direction of the connection only, and has to be negotiated separately for each direction if it is desired to set approximate record sizes in both directions.

3. Description of the option.

a. With the option which specifies the approximate size of records transmitted over the connection, the transmitter attempts to send messages of the specified size unless some other constraint (for instance, an end of line) requires the message to be sent sooner, or characters for transmission arrive so fast that the message has to be bigger than the specified size. The option is to be used strictly to improve the STATISTICS (e.g., timing and buffering) of message reception and transmission -- the option does NOT specify any absolutes.

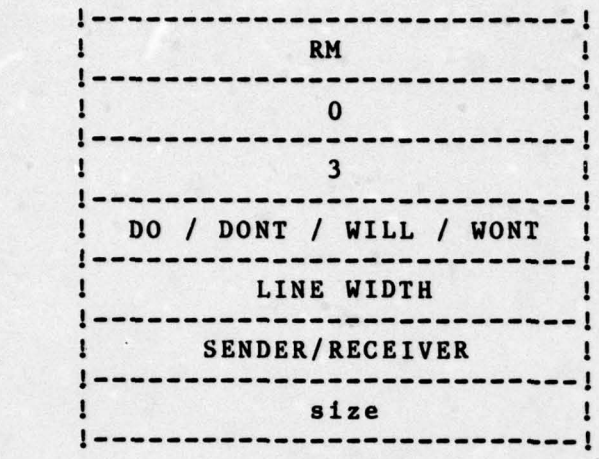
b. The two size octets specify a 16 bit number which is the approximate record size. With this option not in effect, record size is completely undefined as per the NVT specification.

4. Default

No attempt will be made to agree on a message size.

I. LINE WIDTH

1. Structure



2. Motivation for the Option

a. If we consider the example of a computer transmitting data over a connection to a terminal where the data is printed, then the computer is the data sender and the terminal is the data receiver. Continuing to use this example, this option could be used to negotiate the line width to be used when printing lines from the computer on the terminal. To negotiate line width on the other half of the connection the THP'S involved reverse their data sender and data receiver roles; this can be done unambiguously as the sender of a DO or DONT command can only be the data sender, thus defining the direction of the connection under discussion, and the sender of a WILL or WONT command can only be the data receiver.

b. There appear to be four cases in which it is useful for one end of a connection to communicate with the other about the output line width:

(1) the sender may wish the receiver to use its local knowledge of the printer width to properly handle the line width;

(2) the receiver may wish the sender to use its local knowledge of the data being sent to properly handle the line width;

(3) the sender may wish to use its local knowledge of the data being sent to instruct the receiver in the proper handling of the line width; and

(4) the receiver may wish to use its local knowledge of the printer width to instruct the sender in the proper handling of the line width.

c. An example of proper handling of the line length is for the receiver to wrap around lines sent by the sender so that the lines fit on the printer page. Another example of proper handling of the line length might be not folding lines even though they overflow the printer page, if that is what the user desires.

3. Description of the Option

a. The SENDER/RECEIVER entry indicates which end of the connection is to assume responsibility for line-width handling. The size indicates the desired line-width in characters. In addition the following conventions about the size are adopted:

A size of zero indicates a request or agreement, to handle line width with no explicitly suggested size.

A size of 255 suggests a line width of infinity.

b. If both ends suggest simultaneously (WILL and DO) the same party to handle line width, but offer different sizes, the minimum of the two sizes is assumed to be agreed upon.

c. The guiding rules are that

(1) if neither data receiver or data sender wants to handle output line-width considerations, the data receiver must do it, and

(2) if both data receiver or data sender want to handle output line-width considerations, the data sender gets to do it.

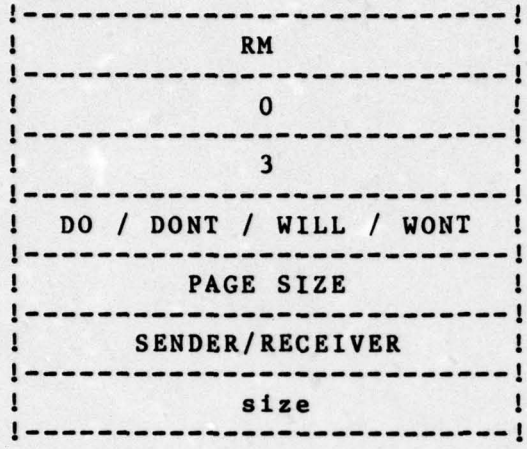
d. The reasoning for the former rule is that if neither want to do it, then the default in the option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver makes.

4. Default

In the absence of negotiation concerning which connection end, data sender or data receiver, is handling output line width considerations, neither end is required to handle line width consideration and neither end is prohibited from handling line width consideration but it is appropriate if at least the data receiver handles line width considerations albeit primitively.

J. PAGE SIZE

1. Structure



2. Motivation for the Option

a. In the following, we are discussing one direction of a full duplex connection. In this direction data passes from the data sender to the data receiver. If we consider the example of a computer transmitting data over a connection to a terminal where the data is printed, then the computer is the data sender and the the terminal is the data receiver. Continuing to use this example, this option could be used to negotiate the page size to be used when printing pages from the computer on the terminal. To negotiate page size in the other direction of the connection the THP's involved reverse their data sender and data receiver roles; this can be done unambiguously as the sender of a DO or DONT can only be the data sender, thus defining the direction of the connection under discussion, and the sender of a WILL or WONT can only be the data receiver.

b. There appear to be four cases in which it is useful for one end of a connection to communicate with the other about the output page size:

(1) the sender may wish the receiver to use its local knowledge of the printer page size to properly handle the page size;

(2) the receiver may wish the sender to use its local knowledge to the data being sent to properly handle the page size;

(3) the sender may wish to use its local knowledge of the data being sent to instruct the receiver in the proper handling of the page size; and

(4) the receiver may wish to use its local knowledge of the printer size to instruct the sender in the proper handling of the page size.

c. An example of proper handling of the page size is for the receiver to hold off further output until instructed to continue when the lines being printed are about to overflow the display screen.

3. Description of the Option.

a. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for page-size handling. The size indicates the desired page-size in lines. In addition the following conventions about the size are adopted:

A size of zero indicates a request or agreement to handle page-size with no explicitly suggested size.

A size of 255 suggests a page-size of infinity.

b. If both ends suggest simultaneously (WILL and DO) the same party to handle page size, but offer different sizes, the minimum of the two sizes is assumed to be agreed upon.

c. The guiding rules are that

(1) if neither data receiver or data sender wants to handle output page size considerations, the data receiver must do it, and

(2) if both data receiver or data sender want to handle output page size, the data sender gets to do it.

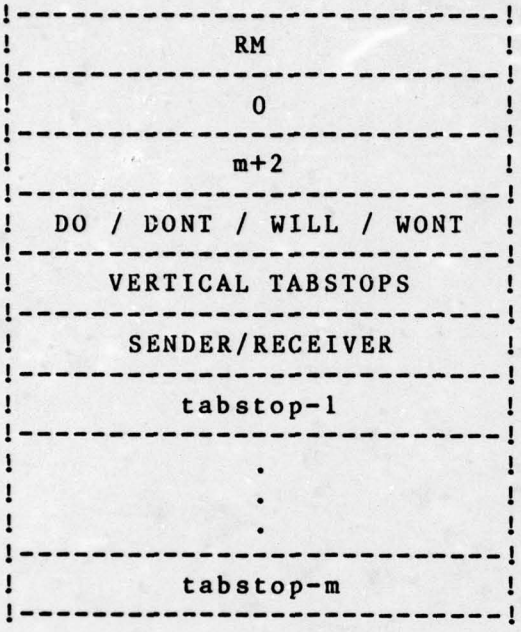
d. The reasoning for the former rule is that if neither wants to do it, then the default dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver makes.

4. Default

In the absence of negotiation concerning which connection end, data sender or data receiver, is handling output page size considerations, neither end is required to handle page size consideration and neither end is prohibited from handling page size consideration, but it is appropriate if at least the data receiver handles page size considerations albeit primitively.

K. VERTICAL TABSTOPS

1. Structure



2. Motivation for the Option

a. Refer to the motivation for the Page-Size option.

3. Description of the Option

a. In the following, we are discussing one direction of a connection, as described in the Page-Size option.

b. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for vertical tabstops handling. The tabstop-i indicates the line number, relative to the physical top of the printer page or terminal screen, that are to be set. If no tabstops appear following the SENDER/RECEIVER entry, it is interpreted as a request or agreement, to handle horizontal tabstops with no explicitly suggested column numbers.

c. The guiding rules are that:

(1) if neither data receiver nor data sender wants to handle output vertical tabstops, the data receiver must do it, and

(2) if both data receiver and data sender want to handle output vertical tabstops, the data sender gets to do it.

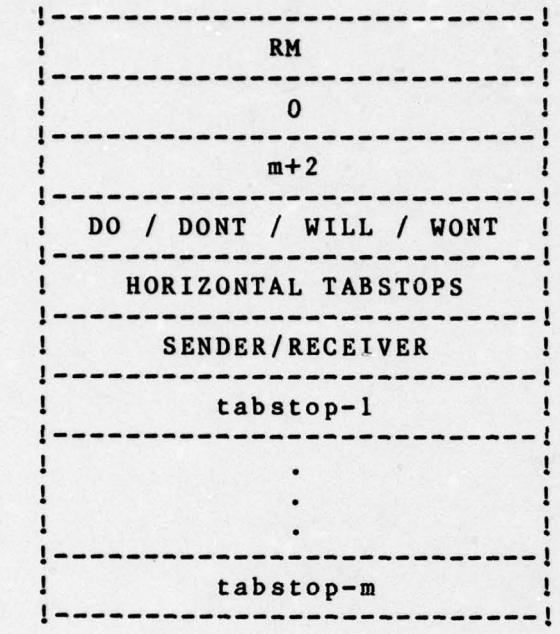
d. The reasoning for the former rule is that if neither wants to do it, then the default in the option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make. This is necessary due to the asynchrony of network transmissions.

4. Default

In the default absence of negotiations concerning which connection end, data sender or data receiver, is handling output vertical tabstop considerations, neither end is required to handle vertical tabstops and neither end is prohibited from handling them; but it is appropriate if at least the data receiver handles vertical tabstop considerations, albeit primitively.

L. HORIZONTAL TABSTOPS

1. Structure



2. Motivation for the Option

a. Refer to the motivation for the Page-Size option.

3. Description of the Option

a. In the following, we are discussing one direction of a connection, as described in the Page-Size option.

b. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for horizontal tabstops handling. The tabstop-i indicates the column numbers, relative to the physical left side of the printer page or terminal screen, that are to be set. If no tabstops appear following the SENDER/RECEIVER entry it is interpreted as a request or agreement, to handle horizontal tabstops with no explicitly suggested column numbers.

c. The guiding rules are that:

(1) if neither data receiver nor data sender wants to handle output horizontal tabstops, the data receiver must do it, and

(2) if both data receiver and data sender want to handle output horizontal tabstops, the data sender gets to do it.

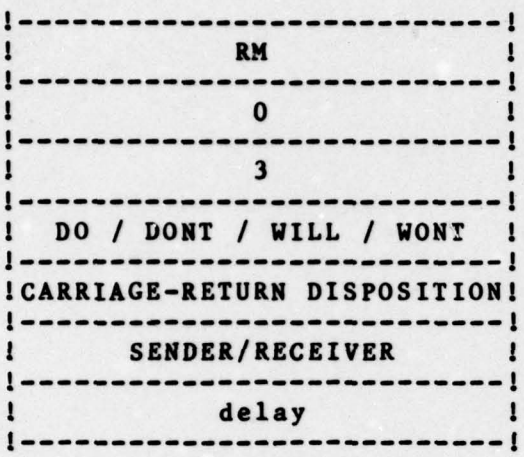
d. The reasoning for the former rule is that if neither wants to do it, then the default dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

4. Default

In the default absence of negotiations concerning which connection end, data sender or data receiver, is handling output horizontal tabstops, neither end is required to handle them and neither end is prohibited from handling them; but it is appropriate if at least the data receiver handles horizontal tabstops, albeit primitively.

M. CARRIAGE-RETURN DISPOSITION

1. Structure



2. Motivation for the Option

a. Refer to the motivation for the Page-Size option.

3. Description of the Option

a. In the following, we are discussing one direction of a connection, as described in the Page-Size option.

b. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for carriage-return disposition handling. The delay indicates the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character. In addition the following conventions about the delay are adopted:

A delay of zero indicates a request or agreement, to handle carriage-return disposition with no explicitly suggested delay.

A delay of 255 suggests the discarding of all carriage returns.

c. If both ends suggest simultaneously (WILL and DO) the same end to handle line width, but offer different sizes, the minimum of the two sizes is assumed to be agreed upon.

d. The guiding rules are that:

(1) if neither data receiver nor data sender wants to handle carriage-returns, the data receiver must do it, and

(2) if both data receiver and data sender want to handle carriage-returns, the data sender gets to do it.

e. The reasoning for the former rule is that if neither wants to do it, then the default dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

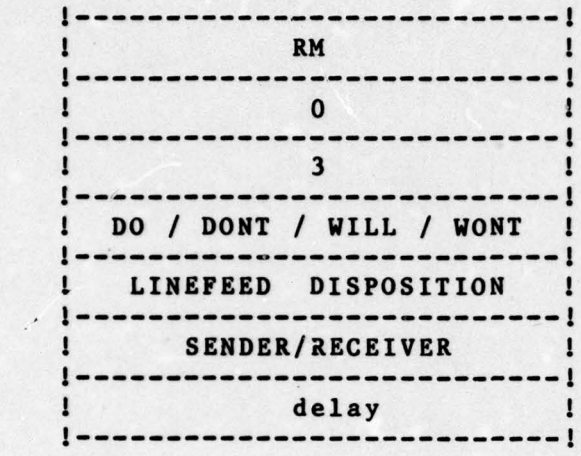
f. Note that carriage-return delays, controlled by the data sender, must consist of NUL characters inserted immediately after the character in question. This is necessary due to the asynchrony of network transmissions. Due to the end-of-line convention, with carriage-returns followed by a linefeed, any NULs that would otherwise be placed after the carriage-return must be placed after the linefeed, regardless of any modifications that may additionally be made to the line feed.

4. Default

In the default absence of negotiations concerning which connection end, data sender or data receiver, is handling output carriage-returns, neither end is required to handle carriage-returns and neither end is prohibited from handling them; but it is appropriate if at least the data receiver handles carriage-returns, albeit primitively.

N. LINEFEED DISPOSITION

1. Structure



2. Motivation for the Option

a. Refer to the motivation for the Page-Size option.

3. Description of the Option

a. In the following, we are discussing one direction of a connection, as described in the Page-Size option.

b. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for linefeed disposition. The delay indicates the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character. In addition the following conventions about the delay are adopted:

A delay of zero indicates a request or agreement, to handle linefeed disposition with no explicitly suggested delay.

A delay of 254 suggests that linefeed be simulated

A delay of 255 suggests the discarding of all linefeeds.

c. If both ends suggest simultaneously (WILL and DO) the same end to handle linefeed disposition, but offer different delays, the minimum of the two delays is assumed to be agreed upon.

d. The guiding rules are that:

(1) if neither data receiver nor data sender wants to handle output linefeeds, the data receiver must do it, and

(2) if both data receiver and data sender want to handle output linefeed disposition, the data sender gets to do it.

e. The reasoning for the former rule is that if neither wants to do it, then the default dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

f. Simulation is defined as the replacement of the linefeed character by new-line (see following) and enough blanks to move the print head (or line pointer) to the same lateral position it occupied just prior receiving the linefeed. To avoid infinite recursion, such simulation is allowed only for linefeed characters that are not immediately preceded by carriage-returns (that is, part of a new-line combination). It is assumed that linefeed simulation will be necessary for printers that do not have a separate linefeed (like the IBM 2741); in this case, end-of-line character padding can be specified through the CARRIAGE-RETURN DISPOSITION option. Any padding (0 < delay < 254) of linefeed characters is to be done for ALL linefeed characters.

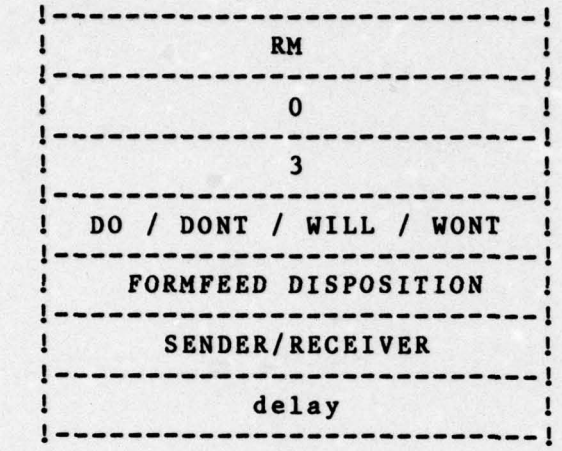
g. Note that delays, controlled by the data sender, must consist of NUL characters inserted immediately after the character. This is necessary due to the asynchrony of network transmissions. Additionally it may be necessary to add carriage-return padding or delay after the associated linefeed (see CARRIAGE RETURN DISPOSITION option).

4. Default

In the default absence of negotiations concerning which connection end, data sender or data receiver, is handling output linefeed considerations, neither end is required nor prohibited from handling linefeeds; but it is appropriate if at least the data receiver handles them, albeit primitively.

0. FORMFEED DISPOSITION

1. Structure



2. Motivation for the Option

a. Refer to the motivation for the Page-Size option.

3. Description of the Option

a. In the following, we are discussing one direction of a connection, as described in the Page-Size option.

b. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for form feed disposition. The delay indicates the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character. In addition the following conventions about the delay are adopted:

A delay of zero indicates a request or agreement, to handle horizontal tab disposition with no explicitly suggested delay.

A delay of 253 suggests that each occurrence of the character be replaced by carriage-return/line-feed

A delay of 254 suggests that form feed be simulated

A delay of 255 suggests the discarding of all form feeds.

c. If both ends suggest simultaneously (WILL and DO) the same end to handle form feed disposition, but offer different delays, the minimum of the two delays is assumed to be agreed upon.

d. The guiding rules are that:

(1) if neither data receiver nor data sender wants to handle output formfeeds, the data receiver must do it, and

(2) if both data receiver and data sender want to handle output formfeeds, the data sender gets to do it.

e. The reasoning for the former rule is that if neither wants to do it, then the default dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

f. Simulation is defined as the replacement of the formfeed character by enough line-feeds (only) to advance the paper (or line-pointer) to the top of the next page (or to the top of the terminal screen).

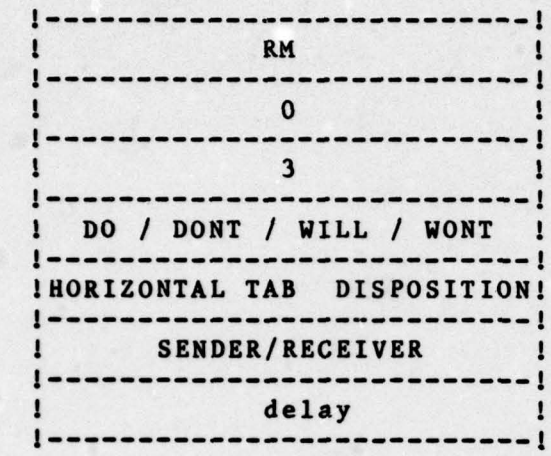
g. Note that delays, controlled by the data sender, must consist of NUL characters inserted immediately after the formfeed character. This is necessary due to the asynchrony of network transmission.

4. Default

In the default absence of negotiations concerning which connection end, data sender or data receiver, is handling output formfeeds, neither end is required to handle formfeeds and neither end is prohibited from handling them; but it is appropriate if at least the data receiver handles formfeed considerations, albeit primitively.

P. HORIZONTAL TAB DISPOSITION

1. Structure



2. Motivation for the Option

a. Refer to the motivation for the Page-Size option.

3. Description of the Option

a. In the following, we are discussing one direction of a connection, as described in the Page-Size option.

b. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for horizontal tab disposition. The delay indicates the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character. In addition the following conventions about the delay are adopted:

A delay of zero indicates a request or agreement, to handle horizontal tab disposition with no explicitly suggested delay.

A delay of 254 suggests that tabbing be simulated

A delay of 255 suggests the discarding of all tabs.

c. If both ends suggest simultaneously (WILL and DO) the same connection end to handle horizontal tab disposition, but offer different delays, the minimum of the two delays is assumed to be agreed upon.

d. The guiding rules are that:

(1) if neither data receiver nor data sender wants to handle output horizontal tab characters, the data receiver must do it, and

(2) if both data receiver and data sender wants to handle output horizontal tab characters, the data sender gets to do it.

e. The reasoning for the former rule is that if neither wants to do it, then the default dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

f. Simulation is defined as the replacement of the horizontal tab character by enough spaces to move the printer head (or line-pointer) to the next horizontal tab stop.

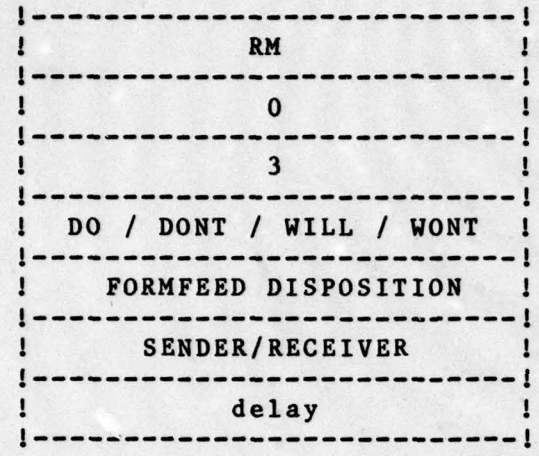
g. Note that delays, controlled by the data sender, must consist of NUL characters inserted immediately after the horizontal tab character. This is necessary due to the asynchrony of network transmissions.

4. Default

In the default absence of negotiations concerning which connection end, data sender or data receiver, is handling output horizontal tab character considerations, neither end is required to handle horizontal tab characters and neither end is prohibited from handling them; but it is appropriate if at least the data receiver handles horizontal tab character considerations, albeit primitively.

Q. VERTICAL TAB DISPOSITION

1. Structure



2. Motivation for the Option

a. Refer to the motivation for the Page-Size option.

3. Description of the Option

a. In the following, we are discussing one direction of a connection, as described in the Page-Size option.

b. The SENDER/RECEIVER entry indicates which connection end is to assume responsibility for vertical tab disposition. The delay indicates the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character. In addition the following conventions about the delay are adopted:

A delay of zero indicates a request or agreement, to handle vertical tab disposition with no explicitly suggested delay.

A delay of 253 suggests that each occurrence of the character be replaced by carriage-return/line-feed

A delay of 254 suggests that vertical tab be simulated

A delay of 255 suggests the discarding of all vertical tabs.

c. If both ends suggest simultaneously (WILL and DO) the same end to handle vertical tab disposition, but offer different delays, the minimum of the two delays is assumed to be agreed upon.

d. The guiding rules are that:

(1) if neither data receiver nor data sender wants to handle the output vertical tab characters, the data receiver must do it, and

(2) if both data receiver and data sender want to handle the output vertical tab characters, the data sender gets to do it.

e. The reasoning for the former rule is that if neither want to do it, then the default dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

f. Simulation is defined as the replacement of the character by enough line-feeds (only) to advance the paper (or line-pointer) to the next vertical tab stop.

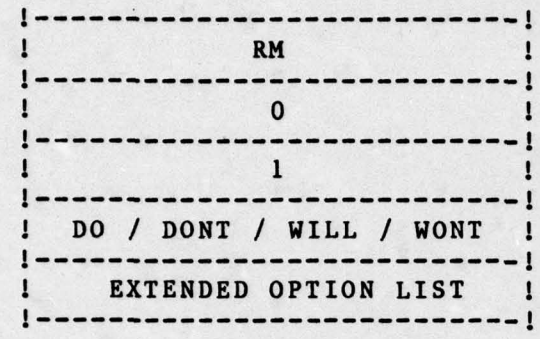
g. Note that delays, *controlled by the data sender*, must consist of NUL characters, inserted immediately after the vertical-tab character. This is necessary due to the asynchrony of network transmissions.

4. Default

In the default absence of negotiations concerning which connection end, data sender or data receiver, is handling output vertical tabstop considerations, neither end is required to handle vertical tabstops and neither end is prohibited from handling them; but it is appropriate if at least the data receiver handles vertical tabstop considerations, albeit primitively.

R. EXTENDED OPTION LIST

1. Structure



2. Motivation for the option

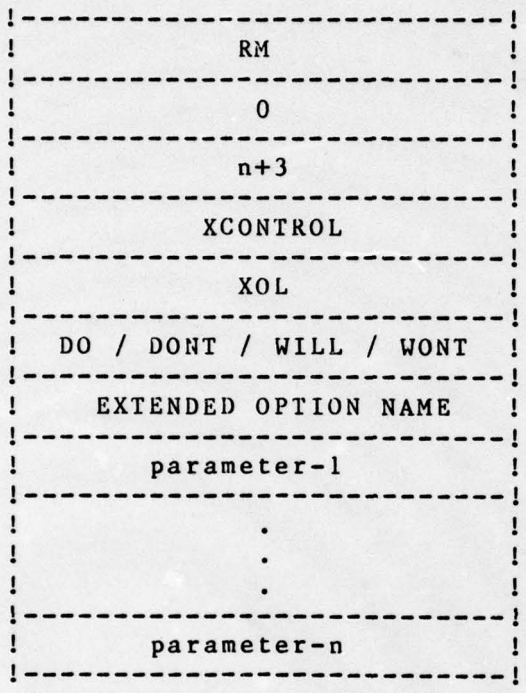
a. Eventually, a 257th option will be needed and there is currently no way to support it. The propsed option will extend the option list for another 256 options in a manner which is easy to implement.

3. Description of the option.

a. When both hosts have agreed to use the Extended Option List (XOL) option they have essentially agreed to include an XOL sub-type in the repertoir of XCONTROL records. The XOL sub-type is described in the following section.

b. The XOL sub-type

Structure



Explanation

The XOL sub-type indicates to the receiver that the following is an option negotiation record, for which the option name is taken from an "extended option list". The bytes following the record type are the regular DO/DONT/WILL/WONT byte, followed by the option name (from the extended list), followed by any parameters that are relevant for the negotiation.

4. Default

Negotiation of options on the "Extended Options List" is not permitted.

S. OPTION CODE ASSIGNMENT

Option Name	Code
ECHO	0
BINARY MODE	1
RCTE	2
INCLUDE-GO-AHEAD	3
EXTENDED ASCII	4
RECONNECTION	5
APPROXIMATE RECORD SIZE	6
LINE-WIDTH	7
PAGE-SIZE	8
VERTICAL TABSTOPS	9
HORIZONTAL TABSTOPS	10
CARRIAGE-RETURN DISPOSITION	11
LINEFEED DISPOSITION	12
FORMFEED DISPOSITION	13
HORIZONTAL TABS DISPOSITION	14
VERTICAL TAB DISPOSITION	15
EXTENDED OPTION LIST	16

APPENDIX H. SCCU

A. Introduction

1. This appendix describes modifications and changes to be made in the THP implementation on an SCCU. The appendix essentially traverses the main document and points out the places where changes should be made. Much of what follows are recommendations for SCCU THP implementation. Individual implementations may include more of what is suggested here up to a full scale THP. The reader of the appendix is assumed to be acquainted with the functions and implementation model of the THP that are presented in the main document.

2. The underlying assumptions for a SCCU THP implementation are that only one connection may exist at any one time, and that the THP has to look, from the network standpoint, as any other THP.

3. The limitation on the number of connections is imposed by the SCCU TCP that allows for only one connection. The need for identical network behavior of a SCCU THP stems from the requirement that non-SCCU users be able to communicate with a SCCU through a regular THP. A direct consequence of the single connection limit is eliminating preemption, since it is achieved by having a second connection listening at all times.

4. Much of the savings in SCCU THP implementation is a result of these assumptions. Introducing a second connection (e.g., for preemption purposes) requires the inclusion of the preemption mechanism, the connection multiplexing mechanism etc. This will make the SCCU less unique in its structure, and in fact more similar to a TAC. SCCU's are envisioned to support mainly process-to-process communication, for which preemption is not meaningful even in a full scale THP. If more than one connection is needed, it is recommended that the interface be via a TAC so that the number of connections is limited only by the availability of resources.

B. Background

1. Configuration Overview

a. The Single Channel Control Unit (SCCU), provides a single-connection interface to the network. The principal intended use of the SCCU is to provide to a pair of hosts currently connected via a private line the less expensive alternative of using the network. In this situation each host

would have an SCCU interface to the network. However, provisions are made for non-SCCU subscribers to communicate with SCCU hosts and vice versa.

b. Two types of SCCU configurations are envisioned, both involving a Host Specific Interface (HSI) module. The first configuration makes use of the TCP by extending requests to the THP; the second configuration allows the HSI a direct access to the TCP without THP's intervention. Although the capabilities of both configurations are the same, the first is intended to enable terminal oriented users of one host to appear as local (terminal) users to the other host. The second configuration is used when the raw connection is employed by a process for a more general activity. A combination of the two configuration will not be allowed to facilitate the operation of the TCP so it does not communicate with two different processes about the same connection. This appendix addresses the changes to be made to a THP for use in an SCCU.

2. Protocol Overview

a. As the SCCU is an integral part of the AUTODIN II network the need for the various protocol levels and functions still exists. The main task of the THP implementer on a SCCU is to design a module that looks like any other THP interface on the network side and has a subset of full-scale THP from the user's side.

b. The important points to remember when implementing a SCCU THP are: (1) Only one connection may exist at any one time, (2) The THP has to look, from the network standpoint, as any other THP and, (3) Preemption may not occur.

C. THP Functions

1. As mentioned above the function of the THP does not change for the SCCU. The THP has to appear like any other THP to all other THP's in the network, and compromises that are made, in design and implementation, may affect only the way the THP looks to its own user.

2. Conceptual Model for the User

a. One of the primary functions of the THP is to give the user a view of the network that makes lower levels of protocol as invisible as possible. The user who uses the THP has some

model of how text is actually passed to a remote user. This model should be identical for SCCU and non-SCCU users.

b. Network Virtual Terminal

i. The NVT is a network-wide feature of all THP's and thus must be maintained by the SCCU as well. However, some of the function of the NVT printer may not be implemented.

c. User Addressing

i. Addressing capabilities and syntax should not be altered by the SCCU implementation so that SCCU users can address any addressable entity on the network and can be addressed by any other network user.

ii. Since only one connection may exist in a SCCU at any one time, the dynamic port name may be selected independently by the process using the connection, or alternately provided by the SCCU without the process even knowing what it is. Static port names and standard function selectors should, however, be fixed so that users who need services offered by the host behind the SCCU can address it.

3. HSI Control

a. The HSI in the SCCU configuration assumes the tasks of the terminal controller in the TAC and host configurations. Although the HSI may not be connected directly to a user's terminal, but rather to a process running on behalf of the user or to the actual TC within the host, the HSI must be able to process all the requests submitted by the THP (such as READTERMPROFILE, SETTERMPROFILE, and DATA). The manner in which this can be achieved is left open to be determined separately for each SCCU HSI implementation.

4. Connection Management

a. The connection is the basic tool with which processes communicate over the network; a connection in the SCCU, in this respect, is no exception. It is important that the user have a consistent conceptual view of connections, whether on a SCCU or not. The internal representation of a connection may however, be different for the SCCU case.

b. Interaction Mode Control

i. Insofar as the mode of interaction and the availability of local buffer space permits, data should be accumulated in the SCCU until a complete line of data is ready for transmission, or until some explicit signal to send the data "now" occurs. This signal could be generated either by a process or by a human user.

ii. It should be noted that a character-at-a-time mode may create excessive delays for a user behind the terminal. This stems from the fact that in this mode each character constitutes a letter, and only one letter may be pending. Unless specifically needed, line-at-a-time mode should be used to avoid the delays and increase efficiency.

c. Preemption

i. As preemption may not occur on the SCCU, no preemption facilities need to be provided. This reduces the precedence checking in the THP, and eliminates the need for implementing any controls associated with preemption. The THP must be able to process preemption messages from the remote THP (i.e. if the remote THP is preempted); it is recommended though, that if preemption occurs on a connection, that the connection be closed immediately.

D. THP Environment

1. The THP has to communicate with the process it serves and with the operating system under which it runs. Part of the THP's environment is its interface with its neighbors, the TCP on the network side, and the HSI on the user's side. The interaction between the TCP and the THP could be modified only to the extent that the SCCU TCP is modified, i.e., the THP must be able to accept, and possibly react to, all events coming from the TCP. Internally, however, the environment is quite different--the operating system in the SCCU is not expected to offer all the services that a regular operating system does, and the THP-user communication should reflect the changes in the inter-process relationship.

2. Inter-Process Communication

a. Although process structure in the SCCU may be different from the host or TAC cases, some asynchronous signalling mechanism must be provided. This stems from the concern of creating

excessive wait time in the SCCU. If a complete asynchronous model cannot be implemented, provisions ought to be made so that the inbound part of the THP (the from-TCP letter handler), and the entire TCP and SIP are able to run as independent processes. This will insure that the user does not have to wait (maybe indefinitely) until all segments of incoming letters arrive, and that the THP will be able to process incoming controls (especially those that are processed out of line) independently of the user's activity.

3. Terminal and Process Models

a. Generally, the THP handles processes and terminals in the same way. This is achieved by having a Terminal Controller mediate between the THP and the process or terminal it serves. In the SCCU this interfacing is performed by the HSI which, therefore, assumes the responsibilities of the TC, and should be implemented in such a way that processes need not undergo any modifications in order to be able to serve a remote network user.

E. THP Interfaces

1. The THP interfaces with the TCP on the network side, and an HSI on the user's side. In most cases the behavior of the THP on those interfaces is identical for the SCCU and non-SCCU cases. The few exceptions are outlined below.

2. User Interface

a. A user views the network through a command language. Such a command language enables the user to instruct his THP to perform some control type actions on the connection. A SCCU in this respect is no exception. A command language must still exist, but may include less commands than a regular THP, reflecting the internal changes and the different nature of operation of a SCCU THP.

3. TCP Interface

a. Since a SCCU user and a non-SCCU user need the same kind of capabilities, essentially the same set of events is offered by the TCP. However, because of the special configuration, some of those events need fewer or different parameters, and some are used in a more restrictive manner.

4. HSI Interface

a. As emphasized before, the HSI assumes the role of a TC. This implies that the HSI maintains a terminal profile and provides the THP with read and write access to that data structure.

b. The nature of the communication between the HSI and the host it serves is not specified. However, since both the THP and the HSI are processes, and since a general event sending mechanism is available, we recommend that the THP and HSI make use of that mechanism for their communication. The tasks of the HSI are outlined below.

i. The HSI is responsible for maintaining a terminal profile data structure and for providing the THP with read and write access to it. (In maintaining the profile the HSI may have to communicate with the host it serves.)

ii. The HSI shall notify the THP of any changes made to the terminal profile. To achieve this, the HSI must itself be notified of any such changes by the host and forward the information to the THP.

iii. The functions of formatting, editing, echoing etc. must be provided by the HSI. It is not implied that the HSI perform those functions itself, but rather that it be able to make the appropriate arrangements that those functions be performed somewhere on the path between the THP and the terminal or process that uses the connection.

iv. The main function of the HSI is to convey data to and from a user. To facilitate this function the HSI should buffer such data and avoid signalling the THP for every available character, unless specifically asked to do so by the THP (for example, in a character-at-a-time mode). Furthermore, in order to save resources it is recommended that data buffers be passed between the THP and the HSI by reference, i.e. without actually copying the data from an HSI buffer to a THP buffer.

F. THP Implementation

1. Implementing a SCCU THP should follow the guidelines set forth in the main document. However, the assumptions made in the previous sections should lead to a more compact implementation. The changes in implementation stem mainly from the fact that only one connection exists, so all that is needed in the regular THP

to maintain the multiple-connection property is eliminated. Other changes reflect the additional assumptions made throughout this appendix.

2. Ordering of events should become easier since preemption may not occur. Event handling will also be simpler reflecting the process structure within the SCCU.

3. THP Data Structures

a. Connection Management Block (CMB)

i. The following entries may be deleted from the CMB:

Local socket
Pointer to to-net letter list
Pointer to last octet copied
Forward CMB pointer
UMB back pointer

ii. The following entries should be added to the CMB:

Local port
Pointer to letter buffer

b. User Multiplexing Block (UMB)

i. This data structure is not needed since only one connection exists.

c. To-net Letter List

i. One of the features of the SCCU TCP is its inability to handle more than a single letter at a time. This restriction means that the THP can have only one letter pending and thus the to-net letter list can have at most one entry, and implies waiting for delivery of each letter. The to-net letter list can therefore be replaced by a pointer to the single letter buffer. This pointer may reside in the CMB.

4. Functional Modules

a. Generally, the functional modules within the THP do not change, either in function or in structure. As a result of changes in the THP-HSI communication mechanism, operations on the THP-HSI interface might become simpler. In addition, the

external event handler does not have to sort the events according to connection and precedence.

5. THP Data Stream Scanning

a. Two scanning modes are available in the general THP, the record and the stream modes. Every THP should be able to receive data in either mode. In fact, the modes are designed in such a way that very little extra code or parsing is needed to have both implemented. On the send side, both modes need not be implemented.

b. In the SCCU configuration the record mode is desirable to facilitate throughput since only one outbound letter may be pending. In the inbound direction the record mode can increase throughput due to the simplified handling of DATA records since the HSI and the THP may share buffers and thus can transfer the data to the user with no further delays.

6. THP Control Handling

a. THP-THP Controls

i. Common Control Functions

All the THP-THP common control functions have to be provided both for the local and remote user. However, as stated in the main document, controls that are not provided to local users need not be provided to remote network users.

ii. Option Negotiation Support

Option negotiation must be supported by the SCCU as it is a network-wide feature. It is not necessary however, that all options be implemented. A few suggestions about which options to implement are offered in the Option Negotiation section of this appendix.

iii. Others

Some controls need not be implemented on the THP send side; the THP must however, be familiar with these controls to the extent that it reacts correctly to their occurrence. Below is a list of controls and justification for their exclusion.

SUSPEND

Close the connection

CONTINUE

This should not occur since the THP closes a suspended connection.

NOP

This control does not contribute to the reliable operation of a THP connection.

REPEAT

Need not be implemented in the outbound direction, as the need for data compression is not as important as the need to keep the THP concise.

XCONTROL

This control should be discarded if the THP does not implement any options that involve extended control.

7. Connection Management

a. In the regular THP much support is given to a preempted connection to insure a minimum level of control flow, and to administer multiple connections. Connection management in the SCCU is considerably reduced as a result of having a single non-preemptable connection. The SCCU THP will close the connection as soon as the remote THP indicates that the connection is preempted. Consequently, the number of states for a THP connection is reduced to include only the OPEN, ACTIVE, and CLOSE states.

8. Option Negotiation

a. Option negotiation is a network-wide feature and must therefore be supported by the SCCU in accordance with the rules set forth in the main document. Having only a few options may considerably facilitate the THP's operations.

i. The following options should be fully implemented on a SCCU:

ECHO
BINARY MODE
APPROXIMATE RECORD SIZE

ii. The following options may not be implemented on a SCCU:

RCTE
INCLUDE-GO-AHEAD
EXTENDED ASCII
RECONNECTION
EXTENDED OPTION LIST

iii. The following options should be implemented on the SCCU at least to the extent that the SCCU THP may request or agree that the remote THP perform the options, but will reject requests for it to perform them:

LINE WIDTH
PAGE SIZE
VERTICAL TABSTOPS
HORIZONTAL TABSTOPS
CARRIAGE-RETURN DISPOSITION
LINEFEED DISPOSITION
FORMFEED DISPOSITION
HORIZONTAL TAB DISPOSITION
VERTICAL TAB DISPOSITION

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: (14) SRI-ARC-35940, SRI-ARC-35941

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): (6) Terminal-to-Host Protocol Specification

5. TYPE OF REPORT & PERIOD COVERED: (9) Final rept.

6. PERFORMING ORG. REPORT NUMBER: 35940/35941

7. AUTHOR(s): (10) Jonathan B. Postel, Jon L. Garlick, Raphael Rom, Larry

8. CONTRACT OR GRANT NUMBER(s): DCA 100-76-C-0034 (15)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Augmentation Research Center
Stanford Research Institute
Menlo Park, California 94025

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

11. CONTROLLING OFFICE NAME AND ADDRESS
DCA/Defense Communications Engineering Center
1860 Wiehle Avenue, Reston, Virginia 22090
ATTN: Code R850

12. REPORT DATE: (11) 15 July 1976

13. NUMBER OF PAGES: 177 (12) 178p.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
N/A

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

[redacted]

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for Public Release; Distribution Unlimited.

18. SUPPLEMENTARY NOTES

N/A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Descriptors: Communication Networks, Data Transmission Systems.

Identifiers: Computer Networks.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This is the specification of the Terminal-to-Host Protocol (THP) which serves as both the terminal-to-terminal and terminal-to-process protocol for the AUTODIN II network.

The AUTODIN II system provides the capability for geographically distributed computers, called hosts, to communicate with each other. The hosts are a diverse set of computers of differing manufacture, speed, word size, and operating system. The AUTODIN II system provides a mechanism for communication between hosts, the Transmission Control Protocol (THP) specifies how the-

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

UNCLASSIFIED 407 215

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

hosts use this mechanism to provide communication services to processes, and finally, the THP provides a mechanism for offering services to human users.

The reader of this document is assumed to be familiar with the concepts of the AUTODIN II system, TCP, and operating system concepts. In particular, the reader is expected to have read the AUTODIN II Specification and the TCP Specification.

It is intended that the information presented here be sufficiently complete enough to allow a competent system programmer to implement a program module to carry out this protocol.

This document draws heavily from the ARPA Network protocol documents on the Telnet Protocol. We wish to acknowledge the efforts of the many contributors to that collection. (modified author abstract)