**TELENET Under Single-Connection TCP Specification**

The necessary functions of a user-TELNET to run under TCP/IP on a microprocessor (as in a mobile packet radio terminal) are described. A detailed implementation specification for the mini-TELNET is presented in an ALGOL-like notation, along with a discussion of its user interface, and control and data structures.
TELNET under Single-Connection TCP Specification

D. E. Rubin
Digital Systems Laboratory
Stanford University
Stanford, California 94305

February 2, 1976

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2494, Contract No. MDA903-76C-0093.

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency or the United States Government.
Table of Contents

Section Subsection Page

1. Introduction ............................................................................. 1
2. The User TELNET .................................................................... 1
3. Data Structures ....................................................................... 3
4. Service Routines ..................................................................... 4
5. Pseudo-Algol Implementation Guide ...................................... 7

References .................................................................................. 14
1 Introduction

One of the goals of the evolving Packet Radio Network (PRNET) is to provide for network access through portable packet radio terminals. These terminals will implement packet radio software on a DEC LSI-11, INTEL 8088, or similar microprocessor. Owing to the unnecessary complexity and generality of the Transmission Control Protocol (TCP) used by the PRNET, a single connection subset (TCP0) has been designed specifically for portable PRNET terminal application [2].

Given a user TCP0, it is necessary to provide a means of interfacing a user at the portable interactive terminal to the TCP0 and thereby to the net to which he is attached. This interface should provide for establishment and control of network connections, and interactive data flow between the net and the user, as well as specification by the user of various echoing and command intercept options. In the Arpanet, these functions are provided by the TELNET protocol [3]. This paper provides an implementation specification for a TELNET to run under TCP0 without operating system support on a microprocessor. Design objectives were therefore small size, simplicity, and a structure appropriate for character at a time data streams at or slower than 120 cps. It is assumed that the reader is familiar with the TCP0 user interface [2], the TELNET protocol [3], and the Arpanet TIP user interface [4].

2 The User TELNET

All interactive traffic flow between the terminal and the network is mediated by the TELNET, which also performs user command interpretation and execution, presenting a subset of the standard Arpanet TIP user interface. Commands include:

1) "so <NET>,<TCP>,<SOCKET>": opens the full duplex connection specified by the supplied foreign net, tcp and socket numbers, and the implied local socket.

2) "sc": closes the network connection.

3) "at c": sets transmit character at a time mode.

4) "at l": sets transmit line at a time mode.

5) "ae l": sets local echo mode.

6) "ae r": sets remote echo mode.

7) "ai <character code>": sets a new command intercept character.
The TELNET consists of a single, non-reentrant, non-preemptable process, composed of a set of service routines and a single cyclic wait (polling) routine, from which the service procedures are dispatched on event completion. (Note that execution of the wait causes a context switch to the next in a circular queue of processes: TELNET, TCP0 SEND, TCP0 RECEIVE). Events include:

1) Change in connection or command handling status (e.g., connection opened, or bad command syntax).

2) Reception of new data over the network connection.

3) Reception of a new character from the terminal.

4) TCP0 send completion.

5) Terminal output completion.

An event scoreboard composed of logical flags is used to signal these events and control dispatch of the appropriate service routines. The flags corresponding to 1 through 5 above are:

1) STATUS flags: these control dispatch of the procedure COMPOSEMESSAGE. They include OPEN (connection opened), ERROR (connection error), CLOSED (connection closed), HOSTNR (host not responding), CANT (command cannot be serviced), and BAD (bad command syntax). Any of these but BAD can be set to TRUE by TCP0.

2) NEWDATA: dispatches either PROCESSDATA or PROCESSTELNETCMD, dependent upon whether the current input is a TELNET escape sequence. This flag is set to TRUE by TCP0 when new data awaits delivery to the user process.

3) TTYINPUT: dispatches either PROCESSCHAR or PROCESSCOMMAND dependent upon whether the current input is a user command intercept sequence. It is set to TRUE by the TTY input interrupt service routine.

4) SENDREADY: dispatches AWAKENSEND. This flag is set to TRUE by TCP0 when new data can be accepted for sending to the net.

5) PRINTERIDLE: dispatches AWAKENPRINTER. It is set to TRUE by the TTY output interrupt service routine.

Note that other conditions can inhibit the dispatch of any of the above service routines. In general, test conditions in the wait routine permit a routine call only if the routine can run to completion.
at that time, without the need for an intermediate event wait. For example, PROCESSDATA will not be called while there is no buffer space for the new character, even though NEWDATA signals data reception. For the one volatile event, TTYINPUT, a special flag, RINGBELL, is set should neither service routine be immediately dispatchable. This causes the terminal output driver to warn the user with a CONTROL-G (bell) that input is being rejected.

3 Data Structures

The data structures used by the TELNET are eight buffers:

TELNETMSGBUF[j]: two linear buffers (j=0 or 1) in which TELNET option negotiation messages destined for the net are placed for double buffered output. Associated with these buffers is a flag, TELNETCMRDREADY, which indicates that the current buffer (TELNETMSGBUF[j]) contains an unsent TELNET command. This flag inhibits dispatch of PROCESSTELNETCMD and execution of the user command "E" (echo mode) until its value is FALSE.

SENDBUF[i]: two linear buffers (i=0 or 1) in which the output stream of user data to the net is queued for double buffered output. Associated with these buffers is three variables: a). XMTMODE (= "C" or "L")—indicates whether character- or line-at-a-time transmission is in effect, b). CHARSTOSTORE— a flag that indicates to the procedure AWAKENSEND when it is time to SEND the contents of SENDBUF[i] to the net, c). BYTECOUNT—this counter is passed to the TCP0 in a SEND call and specifies the number of octets to be sent. Service routine dispatch for the TTYINPUT event is inhibited while SENDBUF[i] is full.

MESSAGEBUF: a linear buffer used for construction of user status messages destined for the terminal. Dispatch of COMPOSEMESSAGE is inhibited while this buffer is non-empty.

ECHOBUF: a circular queue containing the stream of characters to be locally echoed. All characters of a command intercept sequence are automatically enqueued here. Associated with this buffer is a variable ECHOMODE (= "R" for remote, "L" for local) which indicates whether locally generated characters are to be echoed.

RECEIVEBUF: a circular queue in which characters received over the network connection are placed. Dispatch of NEWDATA
event service procedures is inhibited while RECEIVEBUF is full.

COMMANDBUF: a linear queue for assembly of user commands. If it is overflowed, the user is informed with a "BAD" status message.

Note that there are two output streams to the network (TELNETMSGBUF[] and SENDBUF[]) and three output streams to the terminal (MESSAGEBUF, ECHOBUF, RECEIVEBUF). This scheme greatly simplifies buffer management and the overall control structure, as well as permitting a priority structure to be assigned to different characters, depending upon source (the streams are listed above from highest to lowest priority): characters of higher priority are processed for output before those lower in the hierarchy.

Also an important part of the control/data structure are two counting variables:

TELNETCOUNT: indicates how many bytes of a TELNET escape sequence have been received over the network connection (the start of such a sequence is signalled by an "IAC" in the TELNET input stream). On NEWDATA event it is used to select between PROCESSDATA (if=8) and PROCESSTELNETCMD (if>8).

COMMANDCOUNT: indicates how many bytes of a command intercept sequence have been received from the terminal (the start of such a sequence is signalled by an INTERCEPTCHAR in the terminal input stream). On TTYINPUT event, it is used to select between PROCESSCHAR (if=8) and PROCESSCOMMAND (if>8).

4 Service Routines
There are four categories of TELNET service procedures:

1) Net input handlers: PROCESSDATA, PROCESSTELNETCMD.
2) Terminal input handlers: PROCESSCHAR, PROCESSCOMMAND, PARSE, ASCII TO BINARY.
3) Net output handlers: AWAKENSEND, TELNETREPLY, SENDCHAR.
4) Terminal output handlers: TTYINPUT INTERRUPTSERVICE, TTYOUTPUT INTERRUPTSERVICE, COMPOSEMESSAGE, AWAKENPRINTER, TYPENEXT, TYPE.
Each list of routines is in top level to lower level order.

COMPOSEMESSAGE is called from the event wait loop when MESSAGEBUF is free and a STATUS flag has been set. For each set flag, status text is placed in MESSAGEBUF. All status flags are then reset to FALSE.

PROCESSDATA is called from the wait loop if a terminal-bound character arrives and RECEIVEBUF has space. The character is fetched from the TCP0 via a RECEIVE call, and then enqueued in RECEIVEBUF, unless it is "IAC", in which case TELNETCOUNT is incremented.

PROCESSTELNETCMD is called from the wait loop when a TELNET command byte arrives. It has two entry points, one chosen by the value of TELNETCOUNT. The first handles reception of the second byte of the escape sequence. If this byte is "IAC", then it is enqueued in RECEIVEBUF and TELNETCOUNT reset to zero, else it is saved in the variable COMMAND and TELNETCOUNT incremented. The second entry point handles reception of the third byte (stored into OPTION) and acts on the TELNET command received by sending out an appropriate TELNET reply, plus possibly setting a new ECHOMODE value.

TELNETREPLY is called from PROCESSTELNETCMD, and passed either "DONT" or "WONT". It assembles a TELNET reply in TELNETMSGBUF[j], using this parameter and OPTION, and sets TELNETCMDREADY to TRUE.

PROCESSCHAR is called from the wait loop when a new net-destined character arrives from the terminal and there is buffer room for it. If the new character is the command intercept character, INTERCEPTCHAR, then conditions are initialized for command collection, otherwise the character is prepared for transport to the net by a call on SENDCHAR.

PROCESSCOMMAND is called from the wait loop when a user command byte arrives from the terminal. The character is enqueued in ECHOBUF and COMMANDBUF. If a carriage return, PARSE is called.

SENDCHAR is called from PROCESSCHAR and PROCESSCOMMAND. The character, passed in the global variable CHAR, is enqueued in SENDBUF[i], and if local echo is set, it is also enqueued in ECHOBUF. If the character is special (e.g., a control character, carriage return, or escape), if the transmission mode is character at a time, or if SENDBUF[i] is full, CHARSTOSEND is set to TRUE, signalling that a SEND should be issued.

PARSE is called when COMMANDBUF contains a complete command. Syntax is checked; if not good, BAD is set to TRUE. Otherwise, the appropriate action is taken (e.g., issuing an OPEN, CLOSE, setting a new intercept character, etc.).

ASCII2TObINARY is called from PARSE, and converts an ASCII string in radix-10 to binary, of length in bits specified by a parameter. This
routine uses a global index into COMMANDBUF, STRINGORIGIN, to find the left end of the string. A global flag, BADSTRING, is set to TRUE if bad string syntax. STRINGORIGIN is returned pointing to the character past the string delimiter (CR or COMMA if good syntax).

AWAKENSENDD is dispatched from the event wait loop on signal from the TCP0 via SENDREADY. If there is a TELNET reply ready, it is sent via a SEND call. Otherwise, if CHARSTOSEND, then SENDBUF[i] is output via a SEND. In either case, buffer switching is performed after the SEND.

TTYINPUTINTERRUPTSERVICE and TTYOUTPUTINTERRUPTSERVICE are called via device interrupt. They set to TRUE the TTYINPUT and PRINTERIDLE flags respectively.

AWAKENPRINTER is called from the event wait loop when PRINTERIDLE. From highest to lowest priority it will do one of:

1) TYPE a CONTROL-u if RINGBELL is TRUE, and set RINGBELL to FALSE.

2) TYPE the next character of a carriage return expansion (if CRCOUNT>0). A variable, NULLSNEEDED, controls the number of null characters generated in the expansion.

3) TYPE the next space of a tab expansion (if TABCOUNT>0).

4) TYPENEXT the contents of MESSAGEBUF, if non-empty.

5) TYPENEXT the contents of ECHOBUF, and set TABCOUNT or CRCOUNT to the appropriate value if the character is a tab or carriage return.

6) TYPENEXT the contents of RECEIVEBUF.

TYPENEXT is passed a buffer, from which it dequeues the next character and then TYPEs it. A variable COLCOUNT is also maintained. This is used in setting TABCOUNT for tab expansion.

TYPE is passed a character which it sends to the terminal for printing. PRINTERIDLE is set to FALSE.
begin

TELNETINIT:

(initialize all buffers);
TELNETCOUNT-COMMANDCOUNT=0,
TABCOUNT=CRCOUNT=0;
BYTECOUNT=COLCOUNT=0;
i=j=0;
NULLSNEEDED-(whatever is appropriate for the device);
ECHOMODE-REQUESTEDMODE="R";
XMITMODE="C";
INTERCEPTCHAR="@";
(all flags but PRINTRIOLE set to FALSE);
PRINTRIOLE-TRUE;
(whatever other minor housekeeping details);

WAITLOOP:

(wait for signal);

if (OPEN|ERROR|CLOSED|BAD|BADHINT) and (MESSAGEBUF empty) then

COMPOSEMESSAGE;

if NEWDATA and (RECEIVEBUF not full) then

if (TELNETCOUNT.0) then

begin
NEWDATA=FALSE;
PROCESSDATA;
end

else if (not TELNETCMDREADY) then

begin
NEWDATA=FALSE;
PROCESSTELNETCMD;
end;

if TTYINPUT then

begin
if (SENDBUF[i] full) or (ECHOBUF full) then RINGBELL=TRUE else

if COMMANDCOUNT=0 then PROCESSCHAR else PROCESSCOMMAND;
TTYINPUT=FALSE;
end;

if SENDREADY then AWAKENSSEND;
Mini-Telnet Implementation

February 2, 1976

if PRINTERIDLE then AWAKENPRINTER;
go to WAITLOOP;

PROCEDURE COMPOSEMESSAGE;
begin if BAD then (place "BAD<CRLF><NULL STRING>" in MESSAGEBUF);
if CANT then (place "CANT<CRLF><NULL STRING>" in MESSAGEBUF);
if OPEN then (place "OPEN<CRLF><NULL STRING>" in MESSAGEBUF);
if ERROR then (place "CONNECTION ERROR ... etc");
if HOSTNR then (place "HOST NOT RESPONDING ... etc");
if CLOSED then (place "CLOSED ... etc");
BAD=CANT=OPEN=ERROR=HOSTNR=CLOSED=FALSE;
end;

PROCEDURE PROCESSDATA;
begin RECEIVE(CHAR,1,BYTESXFERRED);
if CHAR=IAC then TELNETCOUNT-1
else (enqueue CHAR in RECEIVEBUF);
end;

PROCEDURE PROCESSTELNETCMD;
case TELNETCOUNT of
 -1:
begin RECEIVE(COMMAND,1,BYTESXFERRED);
if COMMAND not equal (WILL|WONT|DO|DONT) then
begin TELNETCOUNT-0;
if COMMAND=IAC then (place "IAC" in RECEIVEBUF);
end else TELNETCOUNT-2;
end;
-2:
begin TELNETCOUNT-0;
RECEIVE(OPTION,1,BYTESXFERRED);
case COMMAND of
  WILL:
if (OPTION= ECHO) and (REQUESTEDMODE="R") then
  ECHOMODE="R"
else TELNETREPLY(DONT);
Mini-Telnet Implementation

February 2, 1976

-DO:
  TELNETREPLY(WONT);

-WONT:
  if (OPTION=ECHO) then
    if REQUESTEDMODE="R" then CANT-TRUE else
      ECHOMODE="L";
  -DONT:
    TELNETREPLY(WONT);
  end;

PROCEDURE TELNETREPLY(CMD);
begin
  (place "<IAC><CMD><OPTION>" into TELNETMSGBUF(j));
  TELNETCMDREADY-TRUE;
end;

PROCEDURE PROCESSCHAR;
begin
  (fetch tty character to CHAR);
  if CHAR=INTERCEPTCHAR then
    begin
      COMMANDCOUNT-1;
      (initialize COMMANDBUF);
      (enqueue CHAR in ECHOBUF);
      end
  else SENDCHAR;
  end;

PROCEDURE SENDCHAR;
begin
  if (ECHOMODE="L") then (enqueue CHAR in ECHOBUF);
  (enqueue CHAR in SENDBUF(i));
  BYTECOUNT-BYTECOUNT+1;
  if (CHAR a special character) or (XMITMODE="C") or (SENDBUF[i] full)
    then CHARSTOSEND-TRUE;
end;

PROCEDURE PROCESSCOMMAND;
begin
  (fetch tty character to CHAR);
  (fold CHAR to upper case);
  if COMMANDCOUNT=1 and CHAR=INTERCEPTCHAR then
begin
COMMANDCOUNT+3;
SENDCHAR;
end

else begin
enqueue CHAR in ECHOBUF);
if (COMMANDBUF not full) then
begin
enqueue CHAR in COMMANDBUF;
COMMANDCOUNT=COMMANDCOUNT+1;
if CHAR=CR then PARSE;
end
else begin
BAD=TRUE;
COMMANDCOUNT+0;
end;
end;

PROCEDURE PARSE:

if COMMANDBUF(2) not equal (SPACE|CR) then BAD=TRUE else
begin
COMMANDBUF(1) of
=C;
if COMMANDBUF(2)=CR then CLOSE else BAD=TRUE;

=O;
begin
BAD=STRING=FALSE;
STRINGORIGIN-3;
NET=ASCIITOBINARY(8);
TCP=ASCIITOBINARY(16);
SOCKET=ASCIITOBINARY(24);
if BADSTRING or COMMANDBUF(STRINGORIGIN-1) not = CR then
BAD=TRUE
else
begin
OPEN(NET,TCP,SOCKET,RETURNCODE);
if RETURNCODE not equal 0 then CANT=TRUE;
end;
end;
begin
BADSTRING=FALSE;
STRINGORIGIN=3;
TEMP=ASCII TO BINARY(7);

if BADSTRING or COMMANDBUF(STRINGORIGIN-1) not = CR then
  BAD=TRUE
else INTERCEPTCHAR=TEMP;
end;

if COMMANDBUF(4)=CR and COMMANDBUF(3)="C" | "L") then
  XM1MODE=COMMANDBUF(3)
else BAD=TRUE;

begin
if COMMANDBUF(4)=CR and COMMANDBUF(3)="L" | "R") then
  if ECHOMODE not equal COMMANDBUF(3) then
    if TELNETCMDREADY then BAD=TRUE else
      begin
        REQUESTEDMODE=COMMANDBUF(3)
        if REQUESTEDMODE"L" then
          (set <IAC><DNT><ECHO>
           in TELNETMSGBUF[j]))
        else (set <IAC><DO><ECHO>
           in TELNETMSGBUF[j]);

        TELNETCMDREADY=TRUE;
      end
    else BAD=TRUE;
  end;

(anything else):
BAD=TRUE;

COMMANDCOUNT=0;
end;

BINARY PROCEDURE ASCII TO BINARY(LENGTH);

begin
  (convert string from left to right starting at STRINGORIGIN in COMMANDBUF to binary, and stop when a delimiter is found or "LENGTH" bits is overflowed);

  if (LENGTH is overflowed) or (delimiter not equal (cr|",",)) then
    BADSTRING=TRUE;
Mini-Telnet Implementation

February 2, 1976

(leave STRINGORIGIN pointing to character after delimiter);
end;

PROCEDURE AWAKENSEND;

   if TELNETCMDREADY then
      begin
         TELNETCMDREADY=SENDREADY=FALSE;
         SEND(TELNETMSGBUF[j],3);
         j=(j+1) mod 2;
      end;

   else if CHARSTOSEND then
      begin
         CHARSTOSEND=SENDREADY=FALSE;
         SEND(SENDBUF[i],BYTECOUNT);
         BYTECOUNT=0;
         i=(i+1) mod 2;
      end;

PROCEDURE AWAKENPRINTER;

   if RINGBELL then
      begin
         RINGBELL=FALSE;
         TYPE(CONTROLG);
      end;

   else if CRCOUNT>0 then
      begin
         CRCOUNT=CRCOUNT+1;
         if CRCOUNT=2 then TYPE(LF) else
            begin
               TYPE(NULL);
               if CRCOUNT=NULLSNEEDED+2 then CRCOUNT=0;
            end;

   else if TABCOUNT>0 then
      begin
         TYPE(SPACE);
         TABCOUNT=TABCOUNT-1;
      end;

   else if (MESSAGEBUF not empty) then TYPENEXT(MESSAGEBUF)

   else if (ECHOBUF not empty) then
begin
if (next character is TAB) then TABCOUNT- (8-COLCOUNT) else
  if (next character is a CR) then CRCOUNT+1;
  TYPENEXT(ECHOBUF);
end

else if (RECEIVEBUF not empty) then TYPENEXT(RECEIVEBUF);

PROCEEDURE TYPENEXT(BUFFER);
begin
  (dequeue next character from BUFFER into CHAR); if (CHAR is printable) then COLCOUNT- (COLCOUNT+1) MOD 8 else
    if CHAR=CR then COLCOUNT+8;
    TYPE(CHAR);
end;

PROCEDURE TYPE(CHARACTER);
begin
  PRINTERIDLE-FALSE;
  (send CHARACTER to tty);
end;

PROCEDURE TTYINPUTINTERRUPTSERVICE;
begin
  (service the device);
  NEWDATA-TRUE;
end;

PROCEDURE TTYOUTPUTINTERRUPTSERVICE;
begin
  (service the device);
  PRINTERIDLE-TRUE;
end;

end.
References


Defense Communications Engineering Center

Dr. Harry Helm
DCLC, R-520
1860 Wiehle Avenue
Reston, VA 22090

General Electric

Dr. Richard L. Shuey
General Electric Research and Development Center
P. O. Box 8
Schenectady, New York 12301

Dr. A. Bell Isle
General Electric Company
Electronics Laboratory
Electronics Park
Syracuse, New York 13201

Mr. Ronald S. Taylor
General Electric Company
175 Curtner Avenue
San Jose, CA 95125

General Motors Corporation
Computer Science Department
General Motors Research Laboratories
General Motors Technical Center
Warren, MI 48090

Dr. George C. Dodd, Assistant Head
Mr. Fred Krull, Supervisory Research Engineer
Mr. John Boyse, Associate Senior Research Engineer

Hawaii, University of
The ALOHA System
2540 Dole Street, Holmes 486
Honolulu, Hawaii 96822

Professor Norman Abramson

Hughes Aircraft Company

Mr. Knut S. Kongelbeck, Staff Engr.
Hughes Aircraft Company
8430 Fallbrook Avenue
Canoga Park, CA 91304

Mr. Allan J. Stone
Hughes Aircraft Corporation
Bldg. 150 M.S. A 222
P. O. Box 90515
Los Angeles, CA 90009

Hughes Aircraft Company
Attn: B. W. Campbell 6/E110
Company Technical Document Center
Centinela and Teale Streets
Culver City, CA 90230

IBM

Dr. Patrick Mantey, Manager
User Oriented Systems
International Business Machines Corp.
K54-282, Monterey and Cottle Roads
San Jose, CA 95193

Dr. Leonard Y. Liu, Manager
Computer Science
International Business Machines Corp.
K51-282, Monterey and Cottle Roads
San Jose, CA 95193

Mr. Harry Reinstein
International Business Machines Corp.
1501 California Avenue
Palo Alto, CA 94303

Illinois, University of

Mr. John D. Day
University of Illinois
Center for Advanced Computation
114 Advanced Computation Bldg.
Urbana, Illinois 61801

Institut de Recherches d'Informatique et d'Automatique (IRIA)
Reseau Cyclades
78150 Rocquencourt
France

Mr. Louis Pouzin
Mr. Hubert Zimmerman
Information Sciences Institute, University of Southern California
4676 Admiralty Way
Marina Del Rey, CA 90291

Dr. Marty J. Cohen  
Mr. Steven D. Crocker  
Dr. Steve Kimbleton  
Mr. Keith Uncapher

London, University College  
Professor Peter Kirstein  
UCL  
Department of Statistics & Computer Science  
43 Gordon Square  
London WC1H 0PD, England

Massachusetts Institute of Technology  
Dr. J. C. R. Licklider  
MIT  
Project MAC - PTD  
545 Technology Square  
Cambridge, Massachusetts 02139

MITRE Corporation  
Mr. Michael A. Padlipsky  
MITRE Corporation  
1820 Dolly Madison Blvd.  
Westgate Research Park  
McLean, VA 22101

Network Analysis Corporation  
Beechwood, Old Tappan Road  
Glen Cove, New York 11542

Mr. Wushow Chou  
Mr. Frank Howard

National Bureau of Standards  
Mr. Robert P. Blanc  
National Bureau of Standards  
Institute for Computer Sciences and Technology  
Washington, D. C. 20234

Mr. Ira W. Cotton  
National Bureau of Standards  
Building 225, Room B216  
Washington, D. C. 20234

National Physical Laboratory  
Computer Science Division  
Teddington, Middlesex, England

Mr. Derek Barber  
Dr. Donald Davies  
Mr. Roger Scantlebury  
Mr. P. Wilkinson

National Security Agency  
9800 Savage Road  
Ft. Meade, MD 20755

Mr. Dan Edwards  
Mr. Ray McFarland

Norwegian Defense Research Establishment  
P. O. Box 25  
2007 Kjeller, Norway

Mr. Yngvar G. Lundh  
Mr. P. Spilling

Oslo, University of  
Prof. Dag Belsnes  
EDB-Sentret, University of Oslo  
Postbox 1059  
Blindern, Oslo 3, Norway

Rand Corporation  
1700 Main Street  
Santa Monica, CA 90406

Mr. S. Gaines  
Mr. Carl Sunshine

Rennes, University of  
M. Gerard LeLann  
Reseau CYCLADES  
U.E.R. d'Informatique  
B. P. 25A  
35031-Rennes-Cedex, France
Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, CA  94025

Ms. E. J. Feinler
Augmentation Research Center

Dr. Jon Postel
Augmentation Research Center

Mr. D Nielson  Director
Telecommunication Sciences Center

Dr. David Retz
Telecommunication Sciences Center

System Development Corporation

Dr. G. D. Cole
System Development Corporation
2500 Colorado Avenue
Santa Monica, CA  90406

Telenet Communications, Inc.
1666 K Street, NW
Washington, D. C.  20006

Dr. Holger Opderbeck
Dr. Lawrence G. Roberts
Dr. Barry Wessler

Transaction Technology Inc.

Dr. Robert Metcalfe
Director of Technical Planning
Transaction Technology Inc.
10880 Wilshire Blvd.
Los Angeles, CA  90024

Defense Communication Agency

Dr. Franklin Kuo
4819 Reservoir Drive
Washington, D. C.  20007

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA  94304

Mr. David Boggs
Dr. William R. Sutherland

STANFORD UNIVERSITY
Digital Systems Laboratory

Mr. Ronald Crane
Mr. Yogen Dalal
Ms. Judith Estrin
Professor Michael Flynn
Mr. Richard Karp
Mr. James Mathis
Mr. Darryl Rubin
Mr. Wayne Warren

Digital Systems Laboratory Distribution
Computer Science Department - 1 copy
Computer Science Library - 2 copies
Digital Systems Laboratory Library - 6 copies
Engineering Library - 2 copies
IEEE Computer Society Repository - 1 copy

Electrical Engineering

Dr. John Linvill