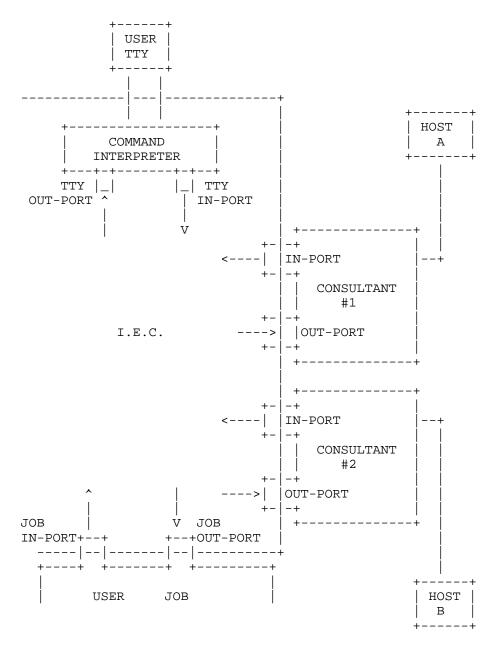   Inter-Entity Communication - An Experiment

    This note is an attempt to be a status report concerning an
    experiment based on the desire of users, at their consoles, to
    converse with one another, and perhaps to get some debugging
    assistance. The user might ask: "who can I talk to"; "can I show him
    what I have done", and "can I let him run my program?" Many time
    sharing systems provide capabilities such as these, within the bounds
    of their system. Almost all systems have a "WHO" or "SYSTAT", many
    have commands like "LINK" or "TALK", and some support more esoteric
    capabilities like controlling another user's program. At the last
    formal meeting of the Network Working Group, in October of 1971 at
    MIT, a group got together to talk about these features for Inter
    Entity Communications (IEC), and how they might be extended to span
    across Host boundaries.

    Subsequent development has proceeded in an ad hoc manner. The
    general design philosophy paralleled that of TELNET in terms of
    having both server and user programs. The server program would
    handle commands like "connect to user FOO", "where is user BAR", or
    "who is on your system?" An initial implementation of a server and
    user was brought up at MIT-DMCG, using a completely arbitrary
    protocol. Soon after that, in an effort to increase its usefulness,
    the protocol was modified to be compatible with that being used by
    the Resource Sharing Executive being developed at BBN-TENEX.

    The MIT user program used the concept of "ports" to help identify
    character streams entering and leaving an object. A pictorial
    diagram follows (FIGURE 1) showing a user teletype, his job and two
    consultants with whom he is conversing.

```
                        +------+
                        | USER |
                        | TTY  |
                        +------+
                          |  |
          --------------|---|---------------+
                          |  |               |
               +------------------+          |       +-------+
               |     COMMAND      |          |       | HOST  |
               |   INTERPRETER    |          |       |   A   |
               +---+-------+-+--+  |          |       +-------+
               TTY |_|       |_| TTY          |           |
          OUT-PORT ^         |  IN-PORT       |           |
                   |         |                |           |
                   |         V            |   +-------------+  |
                                      +-|-+   |             |  |
                              <----| |IN-PORT              |--+
                                      +-|-+   |             |
                                        |  |  CONSULTANT    |
                                        |  |     #1         |
                                      +-|-+   |             |
               I.E.C.           ---->| |OUT-PORT            |
                                      +-|-+   |             |
                                        |     +-------------+
                                        |
                                        |  +-------------+
                                      +-|-+ |             |
                              <----| |IN-PORT             |--+
                                      +-|-+ |             |  |
                                        |  |  CONSULTANT   |  |
                                        |  |     #2        |  |
                                      +-|-+ |             |  |
                   ^            |  ---->| |OUT-PORT        |  |
                   |            |        +-|-+ |           |  |
               JOB |            V  JOB    |   +-------------+  |
          IN-PORT+--+        +--+OUT-PORT |                    |
            -----|--|-------|--|----------+                    |
             +----+  +-------+  +---------+                     |
              |                          |           +------+
              |     USER      JOB        |           | HOST |
              |                          |           |  B   |
                                                     +------+
```

The user now has the option of opening or closing any of the ports he
wishes. While in conversation mode, he might turn off the ports
leading to the JOB. If he wished consultant 1 to control the job, he
might turn off the input ports from his own TTY and from consultant
2.

Towards this goal, the user interface provides the following set of
commands:

WHO   user supplies which host, and given a list of [user,
           teletype, jobs].

WHERE   user supplies identification of another user, and program
           tries to find him on all the servers it knows about (for
           1 server, that code was very easy to write!)

OPEN or CLOSE   user specifies which port to turn on or turn off.

PORT MAP   gives the user a picture of all his ports.

CONNECT   user specifies host, user, and port identification. If
           successful, results in an open connection to the
           specified user.

DISCONNECT user specifies port, and connection is cleanly broken.

The above description applies to the program at MIT-DMCG. Similar
ones will soon be available on the other ITS systems.

From TENEX, the user interface is through the RSEXEC subsystem. To
the user, the RSEXEC looks much like the standard TENEX EXEC, but not
limited to just the local system. With the exception of the concept
of PORTS, the command structure is similar to that previously
described:

@ WHERE   (is user) THOMAS

          Lists each "currently active" job of user Thomas. Each job
          is identified by its network site, job I.D. and attached
          terminals.

@ SITES   (of user) BRESSLER

          Lists all of the (currently accessible) network sites where
          user Bressler has an account.

@ LINK   (to TTY0 103 (AT SITE) UTAH-10

          Links the user's terminal to terminal 106 at the UTAH PDP-
          10.

@ WHO   Lists the users currently logged in at each (accessible)
          network site. (WHO has options for specifying selected
          sites.)

Supplementing the above services, the TENEX RSEXEC program provides a
set of files system tools. It is planned to integrate these services
with the FTP type protocols, and make these services available on
other non-TENEX systems.

Socket 245 (decimal) has been assigned to this experiment. As
mentioned above, these services are now (or will soon be) available
on many ITS and TENEX systems. In addition, at least one of these
services will be available on a non login basis. This will enable TIP
users to avail themselves of these communication facilities.

Further participation in this experiment is of course invited. It is
hoped that a service like this can play an important role in network
development. Sites are invited to experiment with the "conferencing"
possibilities of this experiment. We would be interested in knowing
what drawbacks are encountered. The protocol design will remain
flexible, and can be expanded to meet short comings that use will
discover. Areas of experimentation include integration with the mail
protocol, conference scheduling, and incorporating a picture oriented
graphics protocol, for graphics users to share screens.

Attached is a copy of the protocol currently used. At first glance,
it may appear hostile to non PDP-10s, but this was not intentional.
A new and more general protocol is being developed, but since this
one is operational, it seems useful to try using it.

                            INTERIM PROTOCOL

There are two parts to the RSEXEC protocol:

1. an initial connection protocol which specifies how a user program
   connects to the server program, and

2. a command protocol which specifies how the user process talks to
   the server process to get service.

Initial Connection Protocol

To connect to the server the user process connects to socket number
365 (octal) connection byte size = 32. The server program then
transmits two bytes and breaks the connection:

    byte 1 = socket number = X

    byte 2 = transaction number (meaningful to server)

The server and user programs complete the ICP by opening two 36 bit
"working" connections:

    U + 3 --> X

    U + 2 --> X + 1

    where U = the socket used by the user program to initiate the ICP.

After the two working connections are established the server is ready
to accept commands.

Note that the RSEXEC ICP is virtually identical to the official
ARPANET ICP, the single difference being transmission of the
transaction number.

Command Protocol

[Note on terminology:

ASCII   7 bit characters, packed 5 to a 36 bit word, with the low
            order bit 0. In all following examples the contents of a
            string are delimited with "/".

ASCIZ   ASCII, terminated with a character (7 bits) of zero.

SIXBIT   6 bit characters, packed 6 to a 36 bit word. A sixbit
            character + 60 (octal) = the equivalent ASCII character.

byte   unless otherwise stated is 36 bits.

XWD A,B   Half words. 18 high order bits = A, 18 low order bits =
            B.]

    USINF

To obtain information about a user at the server's site

1. user sends:

   byte 1: ASCII /USINF/
   byte 2->k: ASCIZ /USERNAME/

   2. server responds:

     neg ack: 1 byte = XWD 0, error #   ;no such user

     pos ack: byte 1:   -1
              byte 2->n: XWD job #, tty #
                              where tty # = -1 if job detached
              byte n+1: -1

   SSTAT

   To obtain the active users at the server's site

   1. user sends:
     byte 1: ASCII /SSTAT/

   2. server responds:

     neg ack: 1 byte = 0

     pos ack: 1 byte = -1
              followed by data blocks of the form

              a. 1 byte = -1   ;means end of transmission
              or
              b. byte 1:   XWD job #,tty #
                 byte 2:   SIXBIT /subsys name/
                 byte 3->n: ASCIZ /USERNAME/

   LINK

   To link to a user terminal at the server site

   1. user sends:

     byte 1: ASCII /LINK/
     byte 2: terminal #

   2. server responds:

     neg ack: 1 byte = 0

     pos ack: 1 byte = number N   ;means server will attempt link

   3. if positive acknowledgement server and user try to establish the two
      8 bit connections:

          N + 1 --> U

```
       N       --> U + 1
```

   where U is the socket used by the user to initiate the ICP.

These connections are to be used to carry text to and from the linked
tty at the server's site.

4. server responds (a second time):

   neg ack: 1 byte = 0   ;means can't establish connections or
                          ;couldn't make the link

   pos ack: 1 byte = -1   ;means link to tty established and
                          ;anything transmitted over the
                          ;connections will go to linked tty.

BREAK

To break a link previously established to a terminal at the server
site:

1. user sends:

   byte 1: ASCII /BREAK/

2. server responds:

   neg ack: 1 byte = XWD 0,error #

   pos ack: 1 byte = -1   ;link successfully broken

TERMINATE

To terminate connection with the server the user can either send a
single byte = 0 or just close the connections. The former is
preferred. The server responds by breaking the connections.

        [This RFC was put into machine readable form for entry]
     [into the online RFC archives by Hélène Morin, Viagénie, 12/99]