MEMORANDUM

To: D. Walden, W. Crowther, R. Alter, J. Cole, E. Belove

From: B. Cosell

Subject: PDP-1 User Interface to the Network

Date: 21 April 1972

This memo reflects my current thoughts about the design of the User Interface to the network. The memo is not in any way intended to be "user oriented", but rather is intended to be a system level discussion of the mechanism. A package for user access to the network is a different problem and I will worry about it later.

I see three major constraints governing the design of the User Interface:

1. Exec modifications must be kept to a minimum.

2. The interface must be very flexible.

3. There is no facility within the Exec for inter-program communication.

To preserve flexibility, the Exec will do no preprocessing of data either to or from the network. This, unfortunately, means that each user program will have to handle its own IMP/Host protocol, as well as its own Host/Host protocol. Communication between the Exec and network user programs will be effected by means of queues on the Fastrand.

All messages that come in from the network will get forwarded onto the "network input queue" which will consist of a single item addressed by an invariant number. The item will consist of a series of 16. word blocks, one per message. For each message, its block in the queue will contain:

Ø 1Ø	first nine words of the message (two of leader and seven of data).
11	drum address of message item
12	message length
13 14	date and time message received
15 16 17	SPARE

Messages of nine words or less will have no separate item associated with them. Both the queue and all messages will reside on third two.

To keep the drum from slowly filling up with messages that no user program wants, there will be a special user program whose purpose will be to garbage collect this queue. A message block will be removed from the queue, and the message it addresses removed from the drum, when the message is more than one minute old. Also, to make it a little easier for user programs to do their own garbage collection, it will remove any message whose drum address entry in the queue has its sign bit set. Thus, a user program would look at a message and when done with it just set the sign bit in the drum address word of its message's block, and the garbage collector will remove the block from the queue and the message from the drum. If the user program would like to retain the message, it would rewrite the drum address entry in its message's block with a sign bit and all zeroes, that is, "delete this message block but no item is associated with this message."

For output to the network, the procedure is similar. There will be a queue on third two addressed by an invariant number. Users will place the drum addresses of messages for the network onto this queue. The addresses will be removed from the queue when the message is sent to the network, and, once again to simplify garbage collection for the user program, if the message's address was placed on the queue without its sign bit set, the message itself will be expunged from the drum when it is sent.

In the event that the IMP is down or that the IMP is being used for some purpose which requires the exclusion of other user programs, the invariant numbers pointing to the input and output queues will both be set to zero. This will be the signal to user programs that they may not have access to the network.

Other Problems and Considerations

This scheme is not quite general enough for some of the uses we will be making of the network. In particular, it will be difficult to write user programs which conform to the current Host protocol. A single message from the net over the control link may contain different pieces to be distributed among different programs, and user programs transmitting over the control link cannot know if some other user program is using it at the time. Therefore, the Exec will break up messages over link one from the network into individual control messages so that they can be found by the programs awaiting them, and it will also maintain a "blocked control link" table for governing transmissions to the network.

Another problem is that for almost every use of the network, we will have to invent global mechanisms for assigning unique numbers, be they for links, ports, sockets, or whatever.

Looking over the scheme, I feel that I may have gone overboard in sacrificing user convenience and speed in favor of simplicity in the Exec. Within bounds, the balance can be shifted somewhat. I am relying on you to help with ideas on how to do this. Please forward to me any suggestions or improvements you can come up with.

APPENDIX: Sample User subroutines

These subroutines are not several things. They are not debugged; they have never even been assembled. They are not my thoughts as to what the user package for accessing the network should be. They do not include IMP/Host Protocol.

They are — 1) to find out how unwieldy the mechanism is (∿ 1ØØ words); 2) to provide an example of how to deal with shared structures on the PDP-1; and 3) to learn, from the readily apparent inadequacy of these subroutines, what the correct set of subroutines for the user package is.

BC/jm
Attach.

```
/SAMPLE SUBROUTINES FOR ACCESSING THE NETWORK VIA THE QUEUES

/SUBR "SEARCH" - SCANS INPUT QUEUE LOOKING FOR MESSAGES FOR
/    THIS USER
/USER SHOULD SUPPLY SUBROUTINE "CHECK" WHICH SKIPS IF "INBLK"
/   IDENTIFIES A MESSAGE FOR HIM
/GIVES R1 IF THE IMP IS DOWN
/GIVES R2 IF NOTHING ON THE QUEUE IS OF INTEREST
/GIVES R3 WITH DRA OF MSG IN I/O, MSG ID IN "INBLK"

SEARCH,   DAP SRCHX
SRCH3,    LIO (NETINQ)   /GET DRA OF THE QUEUE
          IWR
          DIO INQDRA
          SNI
          JMP SRCHX I    /IMP IS DOWN
          CLA            /SET TO BEGIN AT START OF QUEUE
SRCH2,    DAC SKIP
          LAW SGCMDS     /GET NEXT MSG ID
          LIO INQDRA
          SGI+5
           JMP RDERR
          LAC MSGDRA
          SPA            /THIS BLOCK REALLY THERE?
          JMP SRCH6      /IT'S WAITING TO BE DELETED
          JSP CHECK
          SKP I
          JMP SRCH1      /FOUND A GOOD ONE
SRCH6,    LAW 1Ø.        /LOOP TO NEXT MSG ID
          ADD SKIP
          JMP SRCH2

SRCH1,    LIO MSGDRA
          IDX SRCHX
SRCH4,    IDX SRCHX
SRCHX,    JMP .

INQDRA,   Ø
```

```
RDERR,      LAC ERCODE
            SAD (2000)
             JMP SRCH3      /LOOKS LIKE QUEUE MOVED (MAYBE BY GARBAGE COLLE(
            SAD (40000)
            JMP SRCH5       /LOOKS LIKE WE'VE HIT END OF THE QUEUE
            HLT             /BAH?
            JMP SRCH3       /GO TRY AGAIN IF CONTINUE HIT

SRCH5,      LAC QLEN        /ARE WE REALLY AT END?
            SUB SKIP
            SPA
            JMP SRCH4       /REALLY AT END - SO NOTHING FOR US
            HLT             /BAH??
            JMP SRCH3

QLEN,       0               /LENGTH OF QUEUE
RWITNM,     0               /QUEUE'S REWRITE NUMBER
INBLK,      .+9./           /MSG ID GOES HERE
MSGDRA,     0
INBLK+16./

SGCMDS,     QLEN            /READ LENGTH AND REWRITE NUMBER
            2
            I               /NOW SKIP STUFF WE'VE LOOKED AT ALREADY
SKIP,       .
            INBLK           /NOW READ MESSAGE ID
            16.
            -0
```

```
REMOVE,         DAP RMVX            /REMOVE ADDRESSED BY INBLK FROM QUEUE
                CLA
                JMP RMV1

EXPUNJ,         DAP RMVX            /HAVE GC EXPUNGE MESSAGE FROM DRUM
                LAC MSGDRA
RMV1,           IOR (400000)
                DAC RMVVAL
                LAC OWNWD           /SAVE UP USER'S OWNWD
                DAC SVOWN
                LAC (FLEXO IMP)                 /SET UP OF QUEUE'S
                DAC OWNWD
                LAC SKIP            /SET UP TO GET TO PROPER PLACE IN THE QUEUE
                DAC RRSKIP
                ADD (MSGDRA-INBLK)
                DAC RWSKIP
RMV2,           LAW RWCMDS
                LIO INQDRA
                SGI+5               /TRY TO REWRITE THE BEAST
                 JMP RMVERR
RMVRET,         LAC SVOWN           /PUT BACK OWNWORD
                DAC OWNWD
RMVX,           JMP .

RMVERR,         LIO (NETINQ)
                IWNR
                SNI
                 JMP RMVRET         /IMP HAS GONE DOWN
                DIO INQDRA
                LAW RMVRRD          /TRY TO CHECK IF OUR MSG IS STILL THERE
                SGI+5
                 JMP RMVRET         /GUESS NOT, SO WHAT MORE CAN WE DO??
                JMP RMV2            /OK, TRY AGAIN WITH NEW REWRITE NUMBER

SVOWN,          Ø                   /USER'S OWNWD SAVED HERE
RMVVAL,         Ø                   /ADDRESS TO PUT INTO QUEUE

RWCMDS,         IOR QLEN            /RE-WRITE LENGTH AND REWRITE #
                2
                I                   /LEAP UP TO WHERE OUR DRA SHOULD BE
RWSKIP,         .
                IOR RMVVAL          /AND WRITE CORRECT FLAVOR OF REMOVED DRA
                1
                -Ø

RMVRRD,         QLEN                /RE-READ LENGTH AND REWRITE #
                2
                I                   /LEAP UP TO OUR MSG BLOCK
RRSKIP,         .
                AND INBLK           /NO ERROR IF OUR MESSAGE IS STILL THERE
                16.
                -Ø
```

```
/PUT A MESSAGE'S DRA ON OUTPUT QUEUE TO THE NET
/JDA'D TO WITH DRA OF MESSAGE IN AC
/R1 => IMP IS DOWN
/R2 => MESSAGE SENT

FORNET,   0
          DAP RMVRET
          LAC OWNWD        /SAVE USER'S OWNWD
          DAC SVOWN
          LAC (FLEXO IMP)
          DAC OWNWD
FN1,      LIO (NETOTQ)     /GET DRA OF OUTPUT QUEUE
          IVNR
          SNI
          JMP RMVRET       /IMP IS DOWN
          DIO OTQDRA
          LAW FORCMD
          SGI+5            /GET LENGTH AND REWRITE NUMBER
           JMP FN1         /ONLY IMP DOWN CAN NAIL THIS ONE
          LAW I 2          /SET TO SKIP DATA CURRENTLY IN QUEUE
          ADD FORVAR
          DAC FORSKP
          IDX FORVAR       /AND BUMP LENGTH
          LAW FORRW
          SGI+5
           JMP FN1         /IMP'S DOWN OR SOMEBODY CHANGED IT UNDER US
          IDX RMVX
          JMP RMVRET       /DONE

FORCMD,   FORVAR           /READ LENGTH AND REWRITE NUMBER
          2
          -0

FORRW,    IOR FORVAR       /RE-WRITE LENGTH AND REWRITE NUMBER
          2
          I                /SKIP TO END OF ITEM
FORSKP,   .
          IOR FORNET       /CRAM OUR MSG ONTO THE END
          1
          -0

FORVAR,   0                0
OTQDRA,   0


          CONSTANTS
          START
```