Network Working Group J. Bouknight
Request for Comments: 76 J. Madden
NIC 5180 G. Grossman
University of Illinois
28 October 1970

# Connection-By-Name: User-Oriented Protocol

## I. Introduction

Shortly after the first of the year, 1971, the Center for Advanced Computation (CAC) at the University of Illinois will begin to use the facilities of the ARPA network. We are the first of a small class of network nodes whose chief characteristic is that the node is a port to the network only. All computational power for these nodes will be taken from other nodes on the network, ILLIAC IV for example.

An important characteristic of most of the users at our Center is a lack of sophistication about data communication techniques and practices. The user will eventually be in the majority of those using the network from all nodes but the problem is ours, almost from the start.

In our discussions with our prospective users of the network as we designed our port facility, we found that the greatest confusion and consternation arose over having to deal with network protocol at the "nitty-gritty" level of sockets, links, etc. While most of them have been acclimated to computer systems at the file and device-by-name level where the software system handles details, here on the current version of the network, the user handles all details.

Thus, we were compelled to seek a user level interface to network protocol where all user protocol is handled symbolically with system procedures making the translation into host-to-host protocol.

Currently, connections are established by exchange of known socket numbers for the four loose ends of the connection. This requires either that the user or process always know all socket numbers he will use at his or other installations OR that his NCP (and/or related software) remember them for him, allowing him to reference them symbolically.

We propose a more general solution to the "telephone book" approach of obtaining socket numbers for user or processes. Only the host, at each site, knows its socket number space at any given instant in time as well as the status of the user or process to which a socket number assigned. Additionally, most permanently assigned devices and/or processes are known by standard mnemonic labels such as DSK (disk), LP (line printer), CR (card reader), TECO (PDP-10 text editor), etc. In most systems, all other communications are done through files or pseudo files, known only to the user by their names and not by their internal mechanism. In other words, most intrasystem communication at the user level is by symbolic reference to both devices and process.

We propose facilities, by extension of the current protocol, that will allow users to use the network on a connection-by-name basis as they already do in their host system. In the remainder of this paper we will present the suggested extensions to the current protocol and give an example of its usage in a dialogue between a user at CAC, controlling two processes; one at UTAH, and one at PAOLI (ILLIAC IV construction site).

## II. Proposed Extensions to Protocol

Let us define a class of syntax elements for use in our proposed extensions to the protocol. (This syntax is expressed in the metalanguage of the ALGOL-60 report.)

```
<label> ::= <usercode>/<filename>|<device name>

<devicename> ::= <string>

<usercode> ::= <string>

<filename> ::= <string>|<filename>/<string>

<string> ::= <char>|<char> <string>

<char> ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|0|1|2|
           3|4|5|6|7|8|9|.|,
```

A standard set of <devicenames> should be established to reference line printers, card readers, etc. - those hard peripherals with fixed processing tasks. A beginning set of <labels> might be:

| | |
|---|---|
| LP | line printer |
| CR | card reader |
| CP | card punch |
| PTR | paper tape recorder |
| PTP | paper tape punch |
| MT | magnetic tape |
| DSK | disk |
| TTY | teletype compatible terminal |

The format of <usercode> is that of the responding host for the current discussion. Future discussions about foreign-user usage of host facilities may result in a standard format for the entire network.

Most systems can identify files by one <string> plus the <usercode>. Others, such as the Burroughs B6500 use multifile identifiers where many <strings> may be used in the <label>. The set of <char> is that proposed in RFC 66, i.e., ASCII.

The proposed extensions involve a "request" for information and several variants of a "response" to the request.

A. Request for Socket Number for this Label

<RFSNL> <my socker #> <0> <label>

The RFSNL is sent on the control link to the destination host requesting the socket number of the attached <label>.

B. Acknowledgement of Request

Upon receipt of an <RFSNL>, the destination host returns one of three responses:

<AORP> <desired socket#> <your socket #>

<AORN> <desired socket#> <your socket #>

<AORN> <0> <your socket #>

The first response returns the requested socket number and signifies that the user, device, or process exists. The second response returns the requested socket number but signifies that the user, device, or process is not currently available for connection. The last response signifies that no such user, device, or process exists.

C. Discussion

The above extensions to the protocol are intended to enhance user acclimation to network usage. The element of strangeness is subdued and, in fact, for user of the B6500 erased. Attached to this RFC is an appendix containing a preliminary description of the user language of the network port facility being brought up at the CAC. We now present a sample user session on the CAC facility and detail how the protocol is used to establish the proper communication paths.

## III. Example of User Dialogue

Assume a user residing at CAC, whose site code is URBANA. His terminal is an alphanumeric CRT terminal and we assume solution of code conversion problems for network communications.

The sample user session will involve the setting up of two processes at two host sites with control from the third host site. All operations can be accomplished with the current protocol plus the proposed extensions.

In addition, we also assume that some form of standard user code is in use for all host sites uniquely identifying every network user when he is present.

Output keyed by systems will be underlined. Comments are offset to the right for legibility. All statements about the UTAH system are purely hypothetical.

| User Dialogue | Comments |
|---|---|
| | The user moves to the terminal, applies power and types: |
| HELLO | |
| | The CAC system responds for login purposes with: |
| USER= GROSSMAN<br>------ | |
| | for the user's code. |
| 1437 TR7/GROSSMAN LOGGED IN<br>___________________________ | |
| LINE PRINTER DOWN TILL 1600<br>___________________________ | |
| | This acknowledges proper usercode and sends any appropriate notes on system status. |
| ! LINK TO ILLIAC | |
| | The exclamation point (!) is the escape character which flags direct input to the PDP-11 OS: |
| | User requests connection to the ILLIAC IV node. NCP operations establish link from user terminal to B6500 MCP. |

```
   1437 TR7/GROSSMAN LINKED ILLIAC
   -------------------------------
                                 Completes response.

   ? EXECUTE DISK/PRINT; FILE DISK = ALPHA@UTAH REMOTE QUEUE; END

                                 1. DISK/PRINT lists text files from
                                 disk to B6500 line printer.
                                 2. REMOTE files on the B6500 will refer
                                 to files going to/coming from the
                                 network.
                                 3. ALPHA@UTAH specifies that a
                                 connection is to be made via the
                                 network to a file GROSSMAN/ALPHA from
                                 the UTAH node.
                                 4. QUEUE specifies periodic attempt to
                                 complete the connection.

                                 The B6500 will ask for the socket
                                 number associated with GROSSMAN/ALPHA
                                 until an AORP is received.

                                 The language is that of the monitor for
                                 the B6500
   ! FLAG ILLIAC =#
                                 All data received or sent on the link
                                 to ILLIAC must now be prefaced by the #
                                 character.

   ! LINK TO UTAH

   1441 TR7/ GROSSMAN LINKED UTAH
   ------------------------------
                                 User now links into UTAH PDP-10 system.

   #1410: DISK/PRINT BOJ 1441
   --------------------------
                                 System message stating beginning-of-job
                                 for DISK/PRINT on B6500.

   ^C

    . R PIP
    -
                                 User will run PIP on a listing file.
```

```
   * NETWKR:ALPHA@ILLIAC <- DSK:FIL.TMP
   -
                                  NETWRK is network file type for UTAH
                                  system.  Mechanism for file control
                                  basically same as for B6500 system.
                                  Since PIP will be sending to the
                                  network, it does not request a socket #
                                  from the B6500 NCP but instead
                                  instructs its NCP to acknowledge any
                                  request for GROSSMAN/ALPHA from ILLIAC
                                  with the socket number PIP will send
                                  from.  As soon as the B6500 NCP tries
                                  again to find GROSSMAN/ALPHA from UTAH,
                                  success occurs and the socket numbers
                                  are exchanged with subsequent
                                  connection establishment.

   *
   -
                                  PIP completes the task and terminates
                                  the connection to the B6500.

   #14: DISK/PRINT EOJ 1448
                                  B6500 acknowledges completion of task.

   #? TO SPO: SAVE LIST GROSSMANHA FOR MAIL(U OF I/GROSSMAN)
                                  User sends message to B6500 operator.

                                  User logs out of UTAH.
   JOB 10, USER GROSSMAN@URBANA TY68 AT 1448 ON 22-NOV-70
   ------------------------------------------------------

   FILES DELETED: 0, FILES SAVEDL RUNTIME 0 MIN 12 SEC
   ---------------------------------------------------
                                  System logout listing.

   ! END UTAH

   1449 TR7/GROSSMAN DELINKED UTAH
   -------------------------------
                                  Link to UTAH system now dropped.

   # FROM SPO: LISTING MAILED
   --------------------------
                                  B6500 operator response.
```

```
! LEAVE
                                    User desired to log out of CAC system.

1450 TR7/GROSSMAN DELINKED ILLIAC
---------------------------------
                                    Link to ILLIAC system new dropped.

1450 TR7/GROSSMAN LOGGED OUT
----------------------------
                                    Session over.
```

## Syntax and Semantics for the Terminal User Control Language for the Proposed PDP-11 ARPA Network Terminal System

by

G. R. Grossman

### Prefatory Notes

The following document represents a first attempt at providing a control language for the terminal user of the PDP-11 network terminal system. This language is deemed sufficiently powerful to provide the user with a minimal facility for attaching to remote host computers over the ARPA network, initiating processes, and routing data flow to local peripheral devices.

The hardware system as envisioned will comprise a PDP-11/20 with a least 8k of core, a small disk (512 kilobytes of storage), a console teletype, and optional card readers, line printers, DECtapes, User terminals, card punches, storage scopes, etc.

The executive system will consist of a basic driver system which will control autonomous processes and interrupt-driven device service routines. The system will keep tables in core and on the small disk for logging peripheral usage, keeping track of connections on the network, queuing up of tasks that cannot be immediately performed, storing attributes of remote hosts, etc.

Since network hosts handle communications in character-at-a-time or message modes, and may or may not echo characters over the network, the system takes this into account when handling connections to specific hosts. If the connection is in message mode, minimal line-by-line editing facility (character and line deletion) is provided.

A means for the user to change flag and message transmit characters is provided to prevent incompatibilities which may arise between the PDP-11 and other hosts.

This document does not describe control card syntax for card reader usage, nor does it describe the operator’s control language. These will be described in later documents.

### Character Set

```
<character> ::= <letter> | <digit> | <special> | <space>

<letter>    ::= A | B | ... | Y | Z
```

```
<digit>     ::= 0 | 1 | ... | 8 | 9

<special>   ::= ! | " | # | $ | % | & | ’ | ( | ) | * | + | , | - |
                . | / | : | : | < | = | > | ? | @ | [ |  | ] | ^ |
                  |‘ | { | <bar> | }
```

Identifiers

```
<identifiers> ::= <letter> | <identifier> <letter> |
                  <identifier> <digit>
```

Semantics: Identifiers are used to designate peripheral units, host computers, etc. No identifier may exceed 8 characters in length.

Numbers

```
<integer> ::= <digit> | <integer> <digit>
```

Semantics: <integer> are the only form of number allowed in the control language. They must not exceed 2^15-1.

Peripheral Designator

```
<peripheral designator> ::= <device class> <device number> | OPR

<device number>         ::= <digit> | <digit> <digit> |

<device class>          ::= CR | CP | LP | DT | TR | SS
```

Semantics: Peripheral designators name specific peripheral devices. Device lasses designate classes of peripherals.

OPR designates the operator’s console teletype. The classes of peripherals corresponding to the device classes are given on the following table:

| <device class> | type of peripheral |
|---|---|
| CR | card reader |
| CP | card punch |
| LP | line printer |
| DT | DECtape |
| TR | terminal |
| SS | storage scope |

File Label

```
<file label>     ::=  <tape label> |  <tape label> / <tape file name>

<tape label>     ::=  <identifier>

<tape file name> ::=  <identifier>
```

Semantics: File labels provide the means for designating tape files symbolically. If the <tape label> form is used, the designated file is assumed to occupy the entire tape.

Flagged Control Statement

```
<flagged control statement>  ::= <flag> <control statement>

<flag>                       ::= <special>
```

Semantics: <Flagged control statement>s arc the user’s names of communicating with the PDP-11 system. The <flag> must be the system default flag (!) or a substitute which the user provides by means of the <flag statement>. Input to the system which does not begin with a <flag> will be passed on to the process to which the user is connected, if any.

Control Statements

```
<control statement>  ::= <link statement> |

                         <copy statement> |

                         <end statement> |

                         <user statement> |

                         <status statement> |

                         <out statement> |

                         <to statement> |

                         <escape statement> |

                         <back statement> |

                         <delete statement> |

                         <transmit statement> |
```

```
                               <lock statement> |

                               <unlock statement> |

                               <assign statement> |

                               <label statement> |

                               <create statement> |
```

Link Statement

```
   <link statement> ::= LINK TO <host> <q>

   <q>              ::= <empty> |

                        QUEUE   |

                        QUEUE    <integer>
```

Semantics: The Link statement directs the system to set up a connection between the user’s unit and a remote host. The <q> construct allows the user to specify that, if the connection cannot be set up immediately, the system is to keep trying. If the QUEUE form is used, the system will keep trying indefinitely. If the QUEUE integer form is used, the system will try for integer minutes.

Copy Statement

```
   <copy statement>   ::= COPY <source> TO <dest> <q>

   <source>           ::= NETWORK |

                          <file label> |

                          <source class> |

                          <source device>

   <source class>     ::= CR | TR | SS |

   <source device>    ::= <source class> <device number>

   <dest>             ::= NETWORK

                          <file label> |
```

```
                          <dest class> |

                          <dest device>

   <dest class>       ::= CP | LP | TR | SS

   <dest device>      ::= <dest class> <device number>
```

Semantics: The <copy statement> directs the system to set up a connection between the <source> and <dest> and copy records of information between them. If the <device class> or <device> form is used for either <source> or <dest>, the copy process cannot begin until a unit is assigned to the user. If the <file label> form is used, the copy process can likewise not proceed until the system has access to a properly labeled tape. if the NETWORK form is used, a connection to a remote process must be pending.

The <q> construct has the same meaning as for the <link statement>, with the additional provision that the condition that caused the process to be incomplete may be the lack of a device assignment.

End Statement

```
   <end statement>   ::= END
```

Semantics: The <end statement> causes the current connection to be terminated.

User Statement

```
   <user statement>  ::= USER = <identifier>
```

Semantics: The <user statement> is used during the log in process to allow the user to identify himself.

Status Statement

```
   <status statement> ::= STATUS <device class> |

                          STATUS <peripheral designator>
```

Semantics: The <status statement> allows the user to interrogate the system as to the status of a device or class of devices.

Out Statement

```
<out statement> ::= OUT|LEAVE
```

Semantics: The <out statement> allows a user to log out of the system. If the OUT form is used, all queued process initiated by the user are terminated. The LEAVE from does not terminate such pending queued processes so long as these processes do not directly involve the user’s terminal.

To Statement

```
<to statement> ::= TO CON :<text> |  TO <user> : <text>
```

Semantics: The <to statement> allows the user to send a message to the operator or another logged-in user.

Flag Statement

```
<flag statement> ::= FLAG = <special>
```

Semantics: The <flag statement> allows the user to define the character which the system recognizes as preceding a control statement as distinguished from a message to a remote process to which he may be attached. The default flag character is "|".

Back Statement

```
<back statement> ::= BACK ? {ascii special or control character}
```

Semantics: The <back statement> allows the user to define the character which, in control or message mode, causes the system to "forget" the previous input character. The default backspace character is RUBOUT (ASCII 1778).

Delete Statement

```
<delete statement> ::= DELETE = {ASCII special or control character}
```

Semantics: The <delete statement> allows the user to define the character which, in control or message mode, causes the system to "forget" the previous line of input. The default delete character is ASCII VT (control K).

Transmit Statement

<transmit statement> ::= TRANSMIT = {ASCII special or control character}

Semantics: The <transmit statement> allows the user to define the character which, in control or message mode, causes the system to begin interpreting the control statement or to transmit the message. The default transmit character is carriage return.

Lock Statement

<lock statement> ::= LOCK

Semantics: The <lock statement> causes the system to prevent any user or process but the process to which the user is currently attached from sending messages to the user’s terminal.

Unlock Statement

<unlock statement> ::= UNLOCK

Semantics: The <unlock statement> reverses the action of a previous <lock statement>.

Assign Statement

<assign statement> ::= ASSIGN <assign device> <q>

<assign device> ::= LP | DT | CP

Semantics: The <assign statement> causes the system to attempt to assign a device not currently in use to the user. The <q> construct has the same meaning as for the <link statement>.

Label Statement

<label statement> ::? LABEL DT <device number> <tape label>

Semantics: The <label statement> causes the system to write a new label on the DEC tape specified.

Create Statement

```
<create statement> ::= CREATE <tape file name> ON <tape label>
```

Semantics: The <create statement> causes the system to create a new file named <tape file name> on the DEC tape labeled <tape label>.

Purge Statement

```
<purge statement> ::= PURGE <tape label> |

                      PURGE <tape file name> ON <tape label>
```

Semantics: The <purge statement> causes the system to delete all tape directory information on the DEC tape or tape file specified.

[ This RFC was put into machine readable form for entry ]
[ into the online RFC archives by Gottfried Janik 2/98 ]