NETWORK WORKING GROUP NIC 5740
Request for Comments #97
John T. Melvin
Richard W. Watson
SRI-ARC
15 February 1971

A FIRST CUT AT A PROPOSED TELNET PROTOCOL

1 Introduction

This paper describes a first cut at a proposed Telnet protocol. _Telnet_ is a process which runs at a _user's_ _site_ and allows him to utilize a typewriter-like terminal to gain interactive service from a remote _server_ _site over the ARPA Network. This paper was motivated by our need to set specifications for a protocol which would allow online access to the Network Information Center (NIC). The Online System running at the Network Information Center we will refer to as NLS(NIC). On thinking about the problem of setting specifications for access to the NIC, we have tried to generalize our ideas so that they would apply to other systems with characteristics similar to ours. We realize that there are other terminal hardware-software disciplines which might find it difficult to conform to all the requirements stated here and, therefore, the final Telnet protocol will differ from the one stated in this NWG/RFC. One conclusion that we may all have to come to is that connection with the network may force us toward a more standard way of handling terminals and their character streams in our monitors and terminal control hardware. In the meantime, we hope that this paper and others on the same subject that may be in process, coupled with a survey of hardware-software requirements at each site by a NWG subgroup, can result in an initial standard network Telnet protocol being agreed upon quickly, as it is important to get users onto the network as soon as possible so that interactive network usage can indicate further directions for network protocol evolution. Next we outline some design problems, then propose some conventions to solve these problems for access to systems such as the NLS(NIC) and indicate some problems needing further study. The proposed conventions for access to the NLS(NIC) are summarized in Appendix A.

2 Some Design Problems

2A. Basic Assumption

The function of the Telnet process is to make a terminal at a user site appear over the network as logically equivalent to a terminal "directly" connected to the server site. There are a number of implications of this basic function.

i) The user should be able to cause generation of all codes which a server system terminal can generate. With respect to the Network Information Center and some other sites it would seem a reasonable requirement to have keying conventions so that the user can generate all 128 ASCII character codes as input to the network. Other sites with different character codes may require a Telnet process to provide those codes to the network.

ii) The user should be able to escape back to his local system or escape from the server process to the server system.

iii) The Telnets of line-at-a-time systems should be able to work with character-at-a-time systems and line-at-a-time systems and Telnets of character-at-a-time systems should be able to work with line-at-a-time and character-at-a-time systems.

2B Echo Control

We use the term echo control rather than the terms half duplex or full duplex because the Telnet connection is in reality full duplex with respect to network transmissions. Three terminal cases need to be considered.

Case 1 - Character-at-a-time serving site echoed

Case 2 - Character-at-a-time user site echoed

Case 3 - Line-at-a-time user site echoed

Some serving sites may be able to operate with all three cases and some convention is required to set the mode. Strictly speaking, what characters are echoed for what keys struck is of no concern to the serving site, although one would like to try to minimize differences in typescript as it appears to the user.

2C Format Control Characters

The format control characters of horizontal tab (HT), vertical tab (VT), form feed (FF), line feed (LF), and carriage return (CR), need to be handled in a consistent way for Cases 2 and 3 above. With Case 1 above, the situation is simplified.

2D Network Message Boundaries

The NCP to NCP protocol was specified with the goal of having the network message boundaries being invisible to the user processes. It would be good if this goal could be maintained, but it may be difficult with some line-at-a-time systems.

2E An Implementation Convention

ConVentions to solve the above problems are most simply established if we assume that the character stream received from a Telnet process by the server site is entered into that point in the server monitor where character input from "direct1y" connected terminals is entered and output from the server process is entered into the monitor point where normal character output is entered. The server NCP receives its input at the point where normal monitor character output is obtained. In other words, the server process would obtain its input from the server monitor character buffers and send its output to these buffers rather than obtainIng input directly from NCP buffers or outputting to NCP buffers.

The Telnet process, on the other hand, would obtain and send character streams directly from or to its local NCP.

Other situations exist where the user processes at both ends communicate directly with the NCP. Therefore, we would recommend that both modes of connection (user process-monitor-NCP, or user process-NCP) be available for communication between the NCP and a user process. These modes would be set under program control by the user process. The initial network convention during the login procedure and until changed by the server process would be to obtain characters from and send characters to the monitor. The server NCP communicates with the monitor also. The scheme is illustrated in Figure 1.

The motivation for such flexibility may be clearer from the discussion below.

3 Proposed Telnet Conventions

3A The server site is to assume initially that echoing is performed by a user site process until explict1y commanded otherwise. If the user site can send character-at-a-time, then after connection and login have been established, tne user could switch Lo server-site-echo by command to the server site and then command (invisible to the server site) his local Telnet to change its echo mode also.

3B The server process is to assume it will receive the same character set which terminals "directly" connected to it can generate. (We recommend at least 128 character ASCII.) The user’s Telnet may have to recognize two-character sequences to enable generation of both upper and lower-case codes and the control codes. We recommend that the user be able to set either upper or lower case as the default case for single case terminals and be able to specify a case shift character. The user should also be able to specify a character to indicate that the next character struck is to be converted to the appropriate control character code. This latter convention enables control codes directly generated at the terminal to be recognized by the user's system thus enabling escape to the user system. Creating a convention allowing all control codes to enter the network and allowing output of the network to feed into the server monitor before entering the server process, gives a simple mechanism for generating an escape to many existing systems. (The problem is more complicated than this for some systems and we discuss it further below.)

3C We recommend that network standards be established for the meaning of local echoes of HT, VT, and FF or a convention to be established for sending the meaning of these characters to the server process. The NLS(NIC), for example, needs to keep track of the position of the print head and in the absence of such conventions will convert these character codes to spaces and line feeds. This means that the appearance of the page on output may differ from the appearance on input. It would be helpful to the user if his page on output could be formatted as it appeared on input.

3D LF characters would be handled as if they were generated by hitting the line feed key on a terminal "directly" connected to the server system.

3E The carriage return (CR) character can be the source of considerable difficulty. For example, on input, different systems and the same system at different times, can echo and transmit different codes to the terminal and the user process. Some monitor systems echo nothing, just a CR, or a CRLF. Some systems transmit a CR, CRLF, or end of line code (EOL) to the user process. The user process may control the echo or add to it. Given the combinations which can exist at each end of the network connection and with respect to each other, confusion can exist unless we assume the definition of 2A and the implementation convention of 2E. These assumptions imply that when a CR is struck, a CR gets sent over the network. If the user monitor system or terminal control hardware converts a CR to a CRLF or EOL, then the Telnet program must convert it back to a CR. When the CR reaches the server monitor it will handle it properly for the server process.

When echoing is handled by the server system, the proper code or codes will be echoed. The user Telnet on receiving a CRLF can pad it with the proper nulls to handle carriage movement timing for a particular terminal.

When echoing is handled by the user system it would be ideal if the user's Telnet or system used the same echo convention as the server system would. This means that either the Telnet must have a table of echo conventions for the various systems to which it can connect, or that it can obtain this information from the server system or process, or vice versa.

For an initial Telnet protocol this is probably not necessary. The user system can default and echo a CRLF on each CR received. This default should be satisfactory for all the situations we are familiar with and for the NIC.

3F For communication from character- and line-at-a-time systems, the Telnet process may need to recognize a character (user assignable) which we call end of stream (EOS). This character is to have the function defined in the following discussion. The important point is to distinguish end-of-stream as a network function and end-of-line as a user or server system function. Consider line-at-a-time systems first. We have not had much experience with line-at-a-time systems, so what follows will need further study and clarification. As we understand it, line-at-a-time systems recognize a character such as CR or a break signal as the code to wake up the user process and cause transmission to it of the line of text. From the point of view of NLS(NIC) it is important that the user be able to enter lines of text each terminated by a CR where appropriate and at other times to be able to enter text not terminated with a CR. (A statement for NLS(NIC) is a string of text of "arbitrary" length and need not have CRs in it; on output the line is folded for the user at his (user definable) page boundary.)

As an example of what is required, consider the case where the user’s system recognizes CR as end-of-line. In this case the Telnet would be awakened when a CR is received. We would recommend that in this case the CR code be literally entered into the Telnet output buffer. If a CR is preceded by an EOS character, then the CR should not be placed in the Telnet output buffer. Transmission through the network can take place either when an EOS is received or automatically when the Telnet output buffer fills. Transmission to character-at-a-time systems from line-at-a-time systems could require the awkward striking of three keys to get one character through the network.

Now consider transmission for a character-at-a-time system to a server line-at-a-time system. A similar problem to the one to be described also exists between line-at-a-time systems. Given the definition of an EOS character different from CR, a line can be buffered up until the EOS is received and then sent without the EOS. How is the serving system to know that a line has been sent? One way would be for the serving NCP to recognize message boundaries. This convention would violate a design goal. Another way would be for the user Telnet to request its NCI to send an INS command. The sending of INS type of control commands might introduce race conditions in the network and should be investigated before their use with a Telnet process is established. Since some of the line-at-a-time systems, we need some way to be compatible with this hardware using software control signals. We leave this problem for further NWG subgroup study.

3G We now come back to the problem of interrupting or escaping in the remote server system. In systems which do not lock out the input keyboard when output is going on, the mechanisms and conventions outlined above would seem adequate unless a special break signal is the escape signal. This latter case requires more study. In systems which allow no input while output is occurring, one may have to live with the consequences of such a terminal discipline and be prepared to wait until output stops before an escape code can be sent. If the keyboard is locked and an escape break signal can be sent to the user's system, it can prevent output from going to the terminal, but must be prepared to continue receiving it from the server site until the user can inform his Telnet process to send an interrupt or escape signal to the server site. Again this is a problem for further study.

The Online System of the Network Information Center operates on a character-at-a-time monitor system and the conventions established in this paper are adequate for access to it. These conventions are summarized in Appendix A.

APPENDIX A

NETWORK CONNECTION PROTOCOL TO SRI-NETWORK INFORMATION CENTER

1 Initial Connection Protocol

Connection establishment to NIC is identical to that presented in Section II of NWG/RFC 80 NIC (5608,); it is reproduced here:

Telnet contacts NIC

NIC <- user site

- RTS <us> <l> <p>
  - NIC logger is socket 1

user site <- NIC

- STR <l> <us> CLS <l> <us>
  - if accepted
- CLS <l> <us>
  - if rejected

assuming NIC accepts

user site <- NIC

- STR <ss+l> <us>
- RTS <ss> <us+l> <q>
  - NIC receives text thru local socket ss from remote socket us+l via link q

assuming user site accepts

- NIC <- user site
  - STR <us> <ss+l>
  - RTS <us+l> <ss> <r>
    - NIC sends text to remote socket us thru local socket ss+l via link r

```
                  .
                  .
                  .

      user site <- NIC

           ALL <q> <space>

                  .
                  .
                  .

      NIC <- user site

           ALL <r> <space>
```

2 Connection Breaking Protocol

A CLS trade is made between the NCPs for each of the two connections as per Document #1 NIC (5143,).

> We may decide to put a time-out into the NIC connections such that no interaction for some (as yet unspecified) "reasonable" length of time will result in a CLS-out of the connections being initiated by NIC.

3 Third Level Protocol

The first 8 bits received by NIC thru socket ss should be the message data type designating that an 8-bit ASCII stream follows, as per NWG/RFC #63, NIC (4963,).

I.e., the first 8 bits are 00000001

The first 8 bits received by Telnet thru socket us will also indicate a message data type of l. Each network message should have an integral multiple of 8 bits. If a network standard is established different from the suggestion of NWG/RFC #63, NIC (4963,), then we would change this protocol to conform.

NIC will have NCP-generated interrupts disabled, i.e.,

INR will be ignored

INS will not be sent to the remote host

4 NLS(NIC) Character Conventions of Interest to Telnet

Echoing can either be under control of NIS(NIC) or under control of the user site. When we refer to echoing below, we mean under control of NLS(NIC). When echoing is handled by the user site we would expect the user to set the NLS(NIC) output conventions to conform to the echoing conventions at his site. NLS(NIC) assumes echoing is handled by the user site unless explicitly commanded otherwise.

Format affecting control characters

horizontal tab

spaces to next (user definable) stop on both echoing and output.

if during literal input, enters file as ASCII ’11.

form feed

carriage return and (user definable) appropriate number of line feeds on echo and output.

If during literal input, enters file as ASCII ’14

vertical tab

carriage return and (user definable) appropriate number of line feeds on echo and output

if during literal input, enters file as ASCII ’13

carriage return

carriage return followed by line feed on echo and output

if during literal input, enters file as EOL (see below)

line feed

line feed on echo and output

enters file as ASCII ’12 on literal input

EOL (end of line)

presently ASCII code ’37

carriage return followed by line feed on echo and output

if during literal input, enters file as ASCII ’37

If the user’s system automatically appends a LF to a CR before sending it to Telnet or converts CR to some EOL code not ASCII ’37, we would expect Telnet to send NLS(NIC) just a CR or ASCII ’37. If we receive CRLF, then on output we will send CRLFLF.

5 NLS(NIC) Interrupt Attention Convention

A (user definable) ASCII code in the text input stream is used to abort the executing process and return control to the main NLS(NIC) command processor.

This code is presently DEL (ASCII ’177).

Escape to the NIC monitor: No escape is required as all operations needed for use of the NIC can be performed within NLS(NIC).

Character Set: We strongly recommend that the Telnet process be able to generate by some set of keying conventions all 128 ASCII codes. Use of NLS(NIC) will probably feel most comfortable from a device with upper and lower case graphics, although we can provide service to single case devices. We can provide a useful service if the full ASCII set cannot be sent, but would like to minimize the special cases we have to handle. Sites which cannot provide the full ASCII set should contact us.

```
         +----+                      |
         |    |        Server        |
         |    |        Program       |
         |    |                      |
         +----+                      |
          ^ |                        |
          | v                        |
         +----+        Terminal      |
         |    |        control       |
         |    |        software      |    SERVER
         |    |        and           |     SITE
         +----+        possibly      |
          ^ |          hardware      |
          | v                        |
         +----+                       |
         |    |                       |
         |    |        NCP            |
         |    |                       |
         +----+                       |
          ^ |                         |
          | v                         |

          . .
          . .                                            Figure 1 -
          . .
          . .                                        Telnet Connection

          ^ |
          | v
         +----+                       |
         |    |                       |
         |    |        NCP            |
         |    |                       |
         +----+                       |
          ^ |                         |
          | v                         |
         +----+                       |
         |    |                       |
         |    |        Telnet         |
         |    |                       |
         +----+                       |
          ^ |                         |    USER
          | v                         |    SITE
         +----+        Terminal       |
         |    |        control        |
         |    |        hardware-      |
         |    |        software       |
         +----+                       |
          ^ |                         |
          | v                         |
         +----+                       |
         |    |        User           |
          \   |        terminal       |
           \--+                       |
```

New York University

Courant Institute of Mathematical Sciences
AEC Computing and Applied Mathematics Center
251 Mercer Street
New York, N.Y. 10012

ARPA Network Information Center
Stanford Research Institute
Menlo Park, California
RFC #551

Yeshiah Feinroth - NYU
Robert Fink - LBL

Several Atomic Energy Commission installations are planning to enter the network in the (hopefully) near future. These sites include Argonne National Laboratory (IBM 360/195), Lawrence Berkeley Labs, (CDC 7600), and New York University (CDC 6600). Our applications make early implementation of an RJE facility imperative, and although we are resigned to the necessity of implementing FTP, we would like to avoid RJE protocol at least in the first go-around. We would like to be able to use FTP to transfer a file, have it queued for execution, and return output and status information.

To this end we propose to implement local conventions within the site dependent PATHNAME parameter of the FTP. Specifically, the following commands are specified:

STOR	RJE.JOB<rest of pathname>	queue this file for execution
(STOR/RETR)	RJE.PR < >	transfer remote job print file
(STOR/RETR)	RJE.PU < >	" " " punch "
(STOR/RETR)	RJE.MT < >	" " " magtape "
RETR	RJE.STAT < >	retrieve status of remote job

The job execution parameters are not part of the protocol, but must be specified in the standard site dependent control cards which are transmitted with the file. These parameters also determine the output disposition, and the output can be retrieved by the user via RETR, or (optionally) automatically by server initiation via STOR.* The RETR RJE.STAT causes the server to create a file with the status information and transfer it to the user. The FTP user/acct/pass logon is used only to validate the data transfer, not the job’s right to execute, and to identify and distribute the output.

We are concerned that we may have overlooked some problems which are obvious to more knowledgeable people and invite (and request) comments.

* note that in this case the RJE server is an FTP user.

[This RFC was put into machine readable form for entry]
[into the online RFC archives by Tony Hansen 08/08]