Network Working Group
Request for Comments: 325
N.I.C. # 9632

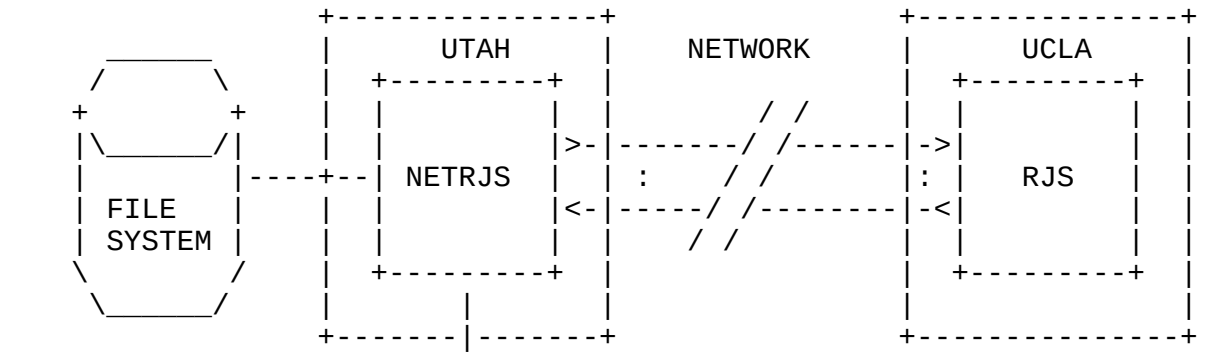
G. HICKS
UTAH

APRIL 6, 1972

Network Remote Job Entry Program - NETRJS

Since October 1971 we, at the University of Utah, have had very large compute bound jobs running continuously. These jobs did reduce response time on our PDP-10 for the other Tenex users.

Since February we have been submitting jobs to the UCLA 360/91. Our normal mode of operation is diagrammed below.



To use NETRJS the user creates a job file under the Tenex system. He then requests NETRJS to send that file to the UCLA RJS (Remote Job Service System). Using NETRJS, the user is able to monitor the progress of his job. When RJS notifies the user that his job output is ready to be picked up, the user can request NETRJS to retrieve his output ("printed" or "punched") to a local file.

WHERE TO GET THE SOURCE PROGRAM

A copy of the source program is available by contacting:

Gregory P. Hicks
Computer Science Department
Merrill Engineering Building 3160
University of Utah
Salt Lake City, Utah 84112
(801) 531-8224

UCLA REMOTE JOB SERVICE

To use the UCLA 360/91 via RJS it is necessary to:

1) Have a valid account at UCLA
2) Have an assigned RJS terminal id.

These can both be obtained from:

Bob Braden
UCLA
Math Sciences Building
3531 Boelter Hall
Los Angeles, Calif. 90024
(213) 825-7518

The remainder of this paper describes the program in greater detail and the steps necessary to run a users program at UCLA.

DESCRIPTION OF THE PROBLEM

There are some jobs that are compute bound for such a long time that they seriously affect response time for interactive users. These are jobs that run from ten hours upward. Another computer was needed to handle these jobs. The UCLA-CCN 360/91 was suggested. The 360/91 is primarily a batch processing type of system where, as a matter of course, it is tuned to jobs that typically run for hours. UCLA does have software to allow jobs to be submitted via remote terminals, either through direct communication lines or through the ARPANET.

HOW AND WHY THE PROGRAM WAS WRITTEN

UCLA’s software allowed the remote terminals to have unlimited connect time (i.e. The time the terminal was actually connected to the 360/91) at no charge to the user. The software at UCLA required that each terminal be allotted 2k (2048 S/360 bytes) core for each connection that is open at any one time. Now, since each terminal could have a virtual card reader, virtual line printer and a virtual card punch, this means that one terminal could occupy 10k of core at UCLA. This - according to the UCLA systems people - would put a heavy load on the system if all the ports were occupied at once. So to alleviate this situation - as a matter of design decision - it was decided to have the operator request to have the various connections opened. The operator could not have more than three connections at any one time. He could have the two operator connections and one of the following open at once:

A) Virtual line printer
B) Virtual card reader
C) Virtual card punch

This would cause the operator to be more active than an operator that had a program that waited for his output to be sent to him automatically. However because of the reduced load on the remote (UCLA) system, the turn around time was probably faster than if all remote RJS users had all the connections open.

DATA TABLES NEEDED BY NETRJS

The NETRJS has no information "built into" it about who has a valid terminal id at UCLA. This information is contained on the disk in a file called PWD.SAV. There is a program that creates and updates this information for NETRJS. NETRJS is therefore site independent. It will work from any TENEX system that is able to use the ARPANET.

THE COMMAND INTERPRETER

NETRJS borrowed R. S. Tomlinson’s TELNET command interpreter and replaced his commands with those needed to run a program at UCLA. As in TELNET, the command interpreter does recognize commands partially typed in. If it does not have enough of the command to recognize it will let the operator know about it.

OPERATION COMMANDS TO NETRJS

The commands available are:

1) SEND.FILE.NAMED
2) RETRIEVE.OUTPUT.FROM
   Options available here are:
   a) PRINTER
   b) PUNCH
      Under punch the options are:
      i) Retrieving an object deck or
      ii) Retrieving an ascii file
3) TIME.NOW.IS
4) LOGOUT
5) RESTART
6) DISCONNECT
7) QUIT
8) SYSTAT
9) JOBSTAT
10) ?
11) ^Q

Now to explain what the various commands do.

SEND.FILE.NAMED - asks the operator for the name of his program on the disk, converts it to card images and sends the file to the Remote Job Service at UCLA. When the file has been accepted by UCLA the operator will get a confirming message telling him how many cards were read and the name of his job. At this point the operator may signoff from RJS and return at a later time to get his output.

RETRIEVE.OUTPUT.FROM - asks the operator for the name of the virtual device the output is available on. The operator may specify either the printer or the punch.

TIME.NOW.IS - outputs the time for the users information.

RESTART - will produce a very virgin NETRJS. This should be used only as a last resort since it does "reset-the-world".

LOGOUT - will do just that. It will log the user out from his local and his remote job. It does require a confirming carriage return or it will do nothing.

DISCONNECT - will log the user out from the remote job and will disconnect (break all connections between) him and the remote computer.

QUIT - this is the only recommended way that the NETRJS program be terminated. The program may be continued with no harm done.

JOBSTAT - will cause RJS to show the status of any jobs that are still active, and that have been submitted by the remote terminal.

SYSTAT - will cause RJS to tell the operator what remote terminals are using the RJS system at the present time.

? - will do several things. When in the command level, it caused NETRJS to tell the operator what it expects next. When nothing has been typed it will respond with all the top level commands. When something is entered, it will respond with all the commands that begin with those particular letters. As in TELNET, it will see nothing that is illegal. When sending or receiving a file the ? will tell NETRJS to type out it’s progress so far. This message is typed at the end of the transaction that it is currently processing. For that reason, it may be a few minutes before the message is typed out.

^Q - this command is a very useful abort facility when used in the following fashion: ^Q ^C will terminate NETRJS. This command is not interpreted by the command interpreter. It is looked for by the sending (console) portion of the program. In any case the program may not be continued. How it works... This command simply re-enables the ^C in the exec and stops the console from doing anything (eg: sending messages to UCLA, finding out the system status at UCLA, etc...) The recommended sequence for this command (IF IT MUST BE USED) is: ^Q^C. For this reason: The program is still processing but the operator cannot communicate with it.

HANDY COMMANDS TO THE RJS AT UCLA

Some of the more useful commands available to the RJS user are:

SIGNOFF - this will inform RJS that the user wants to terminate the session. If there are no output streams active the signoff will be accepted. If there are output streams active the RJS will wait until such time as they have completed.

RESTART - (may be abbreviated RST) will restart the specified device/job. Devices available are (at this time): PRINTER1 AND PUNCH1. The user may specify his jobname. This will restart his job (for output) from the beginning. The format of the restart command is:

RESTART (device or jobname[,JOB])

There will be a confirming message that specifies the action taken.

STATUS SYSTEM - this will tell the operator what remote terminals are using the RJS System now. In addition, status system includes the status of all jobs currently in the system that have been submitted from the user’s terminal. The only abbreviation allowed is for system (SYS).

STATUS JOB - will tell the operator what output if any is waiting to be returned to his terminal. It will also tell him if there are any jobs being executed. This command should be done each time the user signs onto RJS. Abbreviation allowed: J for JOB.

THE NETRJS CONTROL CHARACTERS

The escape character for NETRJS is the control character ^S. This was specified so that remote sites (and users!) could use the program and still retain their sanity and that of telnet. This will always

return you to the command level of NETRJS. This is good if you think that you've made a mistake (eg... when writing the program, etc...) and you want to abort a send. In other words, do not do ^S and think that you can continue where you left off in a send or retrieval. It won't work at this time. In a later implementation, it may, with the provision of stopping (and then continuing, if you wish) the printer, punch, or reader.

EVERYTHING UNDERLINED SHOULD BE TYPED IN BY USER

SAMPLE SESSION USING NETRJS

```
UTAH TENEX 1.28.03, JANUARY 31, 1972 EXPC 1.33.3
[1] (USER) HICKS
           -----
(PASSWORD)
           -----
(ACCOUNT #) 500
            ---
 JOB 6 ON TTY21 1-APR-72 10:12
 RUN NETRJS
 ----------

<<UCLA91 IS UP.>>
NRJ8761 NETWORK REMOTE JOB SERVICE READY
RJS7501 TERMINAL NETUTAH1 HAS SIGNED ONTO RJS
RJS6601 NO ALERTS OUTSTANDING
(^S)
----
<<?
  -
RETRIEVE.OUTPUT.FROM
SEND.FILE.NAMED
DISCONNECT

QUIT
RESTART
TIME.NOW.IS
LOGOUT
DDT
UCLA91
SYSTAT
  SEND.FILE.NAMED TEST.F4;1
  ---             ---------
?
-
5 RECORDS TRANSFERRED.
```

```
FILE TEST.F4;1 HAS BEEN SENT.
12 RECORDS TRANSFERRED.
MORE FILES TO BE SENT? (Y OR N) Y
                                -
INPUT FILE: TEST.DAT;1
?
-
40 RECORDS TRANSFERRED.

FILE TEST.DAT;1 HAS BEEN SENT.
100 RECORDS TRANSFERRED.
MORE FILES TO BE SENT? (Y OR N) N
                                -
<< >>
  -
RJS534I JOB NETUTAH1 ACCEPTED BY RJS - 0000112 CARDS READ
<<SYSTAT >>
  --
RJS802I TERMINAL NETUTAH1 HAS 1 SPL JOB(S)
RJS800I TERMINAL GSM ACTIVE AN LINE1
RJS909I PUNCH REROUTE = ENGR
RJS800I TERMINAL NETILL ACTIVE ON LINE8
RJS800I TERMINAL NETUTAH1 ACTIVE ON LINE11

STATUS JOB
----------
RJS810I TERMINAL NETUTAH1 HAS THE FOLLOWING JOBS ON RJS
RJS812I NETUTAH1 SPL(A) 001
RJS481I PRINT OUTPUT FOR JOB NETUTAH1 NOW AVAILABLE, PRTY=030, IMMED
RJS481I PUNCH OUTPUT FOR JOB NETUTAH1 NOW AVAILABLE, PRTY=060, IMMED

<<RETRIEVE.OUTPUT.FROM PRINTER
  ---                  -------
GOING TO FILE NAMED TEST.PRT [NEW FILE]
                     --------          -
RJS783I TERMINAL STATUS CHANGED
?
-
99 RECORDS TRANSFERRED.
STATUS JOB
------ ---
RJS809I TERMINAL NETUTAH1 HAS THE FOLLOWING JOBS IN RJS
RJS800I NETUTAH1 PRT(A)   060
?
-
169 RECORDS TRANSFERRED.

YOUR OUTPUT IS HERE.
```

```
217 RECORDS AND 16119 BYTES TRANSFERRED.>>

<<RETRIEVE.OUTPUT.FROM PUNCH
  ---                  -----
WILL THIS BE AN OBJECT DECK (O)
OR AN ASCII FILE (A)?? ASCII
                       -
GOING TO FILE NAMED TEST.PUJ [NEW FILE]
                    --------           -
RJS783I TERMINAL STATUS CHANGED


YOUR OUTPUT IS HERE.
17 RECORDS AND 1222 BYTES TRANSFERRED.>>
SIGNOFF
-------
RJS751I TERMINAL NETUTAH1 HAS REQUESTED SIGNOFF FROM RJS
NRJ898I SIGNOFF ACCEPTED
@
```

WHEN THE OPERATOR'S CONSOLE IS ACTIVE

The NETRJS prompt character is the "<<". This indicates that the program is ready to accept commands. When the ">>" is printed the console is in the remote mode. Anything that is typed at this point goes directly to RJS. When sending or retrieving files the operator may communicate with RJS if he wishes even though the ">>" has not yet been output. The operator's console is in the remote mode at all times except when NETRJS is at the command level.

EDITING ON THE NETWORK

There are two control characters that allow you to make mistakes and then to correct them. One is control-x. This has the action of notifying RJS to ignore your last line. This is echoed by "___". The other is control-h. This will cause UCLA to ignore the character immediately preceding it. This is echoed by "_" and there will be no echo of the deleted character.

TABS AND OTHER NEAT GOODIES

Most people writing programs under TENEX like to use the tab stops provided on the teletype. So we needed a nice number to set them to. When running NETRJS the user should be aware that the tab stops are set every 8 columns. In other words, a line that starts with a tab will find the first character in (card) column 8. The program accepts almost any type of <eol> indicator eg: TENEX eol and cr lf.

(This last in any order, by the way.) It also deletes blank lines and since the user may wish to use a program written in a language such as Fortran to generate data, the program also ignores null (00) characters. When retrieving a file of data, for an <eol> the program uses <cr> and <lf>. This is to keep the output compatible with text-editors such as QED, DED, SOS and etc.

FILE FORMAT

When sending a file to UCLA there is a line limit of 80 characters. This is a design restriction in that jobs going to the OS/360 be no longer than 80 characters/line. (Remember, IBM is still card oriented...) The operator will get a nasty (well... semi-nasty anyway) message if NETRJS finds a line that is longer than that. For an output file, there is no limit to the length of a line other than that imposed by the programming language used up to 255 characters. When the user retrieves a punch file he may find that there are a few extra lines thrown in... This is done by UCLA to allow the user to identify his deck when it is punched on the card punch.