Network System Manual

Eric Schmidt

May 1979
(Revised March 1980)

Introduction

This documentation should be read by people responsible for maintaining the network (and the systems it runs on). It is divided into the following sections:

Maintaining the Network
Setting up the Network
Future Plans
For Berkeley
Bugs

Besides the commands described in the net introduction, there are a number of network-internal commands and statistics files.

Maintaining the Network

1. Check the network:
   a) See if the network daemons are running with the command

      ```
      % ps ax | grep net
      ```

      If not running, see below.

   b) Check the network queue to see how long commands have been waiting to be sent.

2. To restart the network daemons, try
   a) See if they are running, as above.

   b) If so, there should be two net daemon processes per machine connection— “kill −9” the child named “netdaemon” and the parent “netstart” will start a new one.

   c) If there are no “netstart’s” or “netdaemon’s”, executing the shell script

      ```
      % /usr/net/bin/start
      ```

      will truncate the log files and start up all the daemons on your machine.

   d) To have two “netdaemons” pointing to the same machine is to invite disaster. What happens is that a few small requests get through, and then the error rate goes up by a factor of a hundred. The first thing to do when you see this is to check the number of net daemons.

   (All this must be done as super-user).

3. There are files /usr/spool/berknet/plogfile? with a log for each directly-connected machine. Example:

   ```
   % tail /usr/spool/berknet/plogfiley
   ```

   will tell you in a cryptic form what the network has done on the Cory machine. This is a good file to inspect to see if transmissions are failing, etc.

   Basically, “sends” begin “ˆS” and end “ˆT”. If a send fails for some reason, “ˆF” is printed instead of “ˆT”. “ˆR” is printed when a receive begins. “RCV” is entered when the command is received and executed. "^P" indicates a pass through.

The file /usr/spool/berknet/netstat?, one per directly-connected machine, have various statistics about the usage of the network, and the system load.

4. The overloading on various machines is causing high error rates. If these rates persist, the network can overload to the point where the queues are immense and nothing gets through. The only thing that can be done at this point is to remove the files (using *netrm* as super-user) and adjust the delay times in the 'initfile'.

5. If free space is a scarce commodity, truncate logfile and plogfile?, and check /usr/spool/berknet/send? and /usr/spool/berknet/rcv. If there are any files there which are quite old, use your judgement to remove them.

6. Net news should be provided periodically (usually in '/usr/help' or '/usr/news').

7. The network queues may have too many entries and "break" the simple rendezvous protocol I use. The easiest way to fix this is to rename the send directory that is too large, make a new, empty one, and move a small group of files in at a time from the rename'd directory to the new one. In this way the daemon thinks the sending directory is small and it less likely to have problems.

8. The netdaemons take a number of command line options:

```
netdaemon −m mach [−d] [−or] [−os] [−ou num]
```

where −m is required and specifies the remote machine. If −d is set, debug mode is turned on. If −or is specified, the daemon will only recieve, not send, requests ("or" stands for only receive). Likewise, −os is "only send". Finally, "−ou num", where num is a user-id, as returned by getuid(), will force the daemon to only send queue files owned by "root" (pass-throughs), owned by "network" (mail, e.g.) and by user-id num.

Setting up the Network

1. Hardware

For another machine to join the network, there must be some hardware link, such as tty lines, special character-oriented hardware, or DMA lines between the two machines. The software does not require the link to be reliable or fast. The best way to start is with slow-speed TTY lines (say 1200 baud) which demonstrate the network's usefulness at low cost. The highest transfer speed on a TTY link is about one-half the link speed (at best), because of processing time, the $3 \rightarrow 4$ character expansion from 8 bits to 6, and the responses.

2. Software

To run the network code, you must have a UNIX® running Version 6 or 7. Version 7 machines should have all the right software already. Version 6 systems must have the *make* command, the "Version 7" C compiler that came out a few years after initial Version 6 (about 1978), and the *alarm()* system call. This compiler, for example, supports multi-level include files.

There is a directory "/usr/src/cmd/berknet" with all the network source files and a "makefile". The file "READ_ME" has information about the different conditional compilation option available, and table entries which must be made in the '.c' files.

Assuming the options have been specified in the makefile, the command

```
% make all
```

will make all the necessary files. Then the command

```
% make install DESTDIR=
```

will install the user commands and service programs. The directories are specified as options in the makefile. Finally,

```
% make clean
```

removes all the '.o' and executable files.

There are also other little-used programs, made by "make othernet". Included are programs to send and receive packets and files, and a program to simulate TTY lines using pipes. It should not be necessary to run these.

The documentation is in /usr/man/man1 and in /usr/doc/berknet.

3. Directories and Files.

The central directories are '/usr/net', which has subdirectories 'bin' and 'network', and '/usr/spool/berknet', which has subdirectories 'rcv' and 'send?', where the '?' represents the one-letter codes of the directly-connected machines. For various reasons, the support programs *(netdaemon, netstart, mmail, mwrite,* etc.) must be in '/usr/net/bin'. The user programs may be anywhere but the pathnames in "Paths.h" must be reset correctly.

The logfiles are 'logfile' and 'plogfile?', one for each directly-connected machine. If not present in '/usr/spool/berknet', they are not created.

The file '/usr/net/bin/start' should start up all the net daemons on the current machine. This file is normally executed by '/etc/rc'. The file '/usr/net/initfile' has a format similar to '.netrc' but is read by the net daemons when they are started. It has the network device names, speed and various tuning parameters. The complete list is in the source file 'netrc.c'. It is generally possible to change almost anything about the network through the 'initfile' and restarting the daemon.

The program '/usr/net/bin/netstart' is a simple program to start up a net daemon, and if it should abort for any reason, restart it.

There must be an account 'network', which executes all responses and the free commands. Its login directory should be '/usr/net/network' and login shell should be 'nsh' in that directory. The list of free commands can be changed in 'nsh.c'.

At Berkeley, we follow the convention that the TTY special files are named '/dev/net-X', where 'X' is the remote machine name.

The UNIX mail program should be modified to recognize remote names and to fork a "/usr/net/bin/sendberkmail" command. Since many people will not or cannot add the options the network would like mail to have, there is a program "v6mail.c" that implements all the options the network wants.

4. Adding a new machine.

Tables in 'config.h' must be updated. Machine descriptions (V6, etc.) must be added to 'mach.h'. Path names must be specified in 'Paths.h'. Free commands must be checked in 'nsh.c'. A procedure "gothru" in 'sub.c' must be updated. All the files etc. described in #3 above must be present.

Future Plans

It is important to understand the scope of this network; what it is and what it is not. Since it is "batched", there are a lot of things it cannot do. Our experience is that remote file copying, mailing and printing between UNIX machines are adequate for our immediate needs.

In the future, we will concentrate on improving the hardware and speeding up the network, rather than major user-interface changes.

This is a list of the things that have been planned for the future for the network.

1. Use Bill Joy's retrofit library to simulate the version 7 system calls. This would reduce the dependence on conditional compilation for V6 code.
2. The file length restriction is a major inconvenience. One way to allow large files would be to send large files (over 100,000 chars) only when there are no smaller ones. This would be

non-preemptive, but might be workable. Another way would be to have two hardware links, and two sets of daemons, one for large files and one for small ones.

3. Bob Fabry has suggested generalizing the machine name to be user-definable as a login/machine pair, to make it easier for people with multiple accounts on multiple machines.
4. It is possible to share binaries between all the similar machine configurations (e.g. the Comp. Center machines). This involves “patching” the local machine in the binary.
5. Ed Gould suggested that the notion of “default” machine was too restrictive— that an appropriate default for, say, “netlpr” was a nearby machine with a quality printer, whereas the default for “net” should be the logical most useful machine.
6. Security — I have just recently bullet-proofed the network so ‘root’ commands are very restricted. However, the presence of passwords in the ‘.netrc’ files poses a hazard to other machines when one machine is broken into. As long as the root password is not in a file, the root is safe. I am fairly convinced there is no way using encryption to handle the ‘.netrc’ files. The introductory documentation is very explicit about the threat these passwords pose.
7. Certain other more exotic requests are unlikely to get done until things change:
   a) Having the same user-id’s across machines.
   b) Adding an option to “net” to wait until a response has been received.
   c) There should be a net status command which would give things like load averages, the number of users, etc.
   d) The notion of a local queue is not general enough— *netq* should print out relevant queues on other machines.
   e) Files on intermediate machines can’t be *netrm*’ed.

For Berkeley

1. A driver for the network links to avoid character processing would make 9600-baud practical. On the Computer Center machines, this could be accomplished using a high speed link through the Bussiplexor.
2. We need a *netrcs* command to use the rcs facilities from remote machines.

Bugs

1. Extra files beginning with ‘df...’ are created in the ‘send?’ directories, with no control files (‘cf...’). They should be removed periodically. *netrm* will remove them.
2. In general, some requests can block the queue until removed. Shorter requests will get through, and longer ones will not. Again, their net queue files should be removed.
3. The network rendezvous protocol seems to occasionally get in a state where a specific file is continually retransmitted and never seems to get through. This happens when both the host system and the network queues are overloaded, and thus is very unpleasant to debug.
4. The network daemons occasionally core dump. They should not.