Applications: Telnet, rlogin, ftp, Sun RPC, nfs, finger, whois, X

Remote login

- remote login to host from host to host \( \Rightarrow \) telnet, rlogin
- rlogin: mostly Unix systems
- rlogin: simpler (no option negotiation)
- both use client/server (rlogind, telnetd)
- one TCP connection
- low volume, short packets, asymmetric
Remote login: components

Telnet/login client

Terminal driver

TCP

IP

subnet

kernel

Telnet/login server

TCP

IP

subnet

pseudo-terminal driver

login shell

Multiple connections

How two clients connect to the login server:

D1

S.login D1.1022

S

D2

S.login D2.1023
rlogin protocol

- password sent as cleartext (snooping!) ⇒ Kerberos
- .rhosts file (host list) bypasses login check ⇒ security risk!
- echoing done by server
- everything typed sent to server, everything received displayed

rlogin server-client interaction

- flow control must be done by client (pipe!) ⇒ XON/XOFF ASCII (control-s/q)
- client interrupt (control-c): stop display locally
- server → client commands via TCP urgent mode
  - flush output (server sends after interrupt)
  - client stops performing flow control
  - client resumes performing flow control
  - please send window size
- client → server commands: window size changes in-band via escape sequence 0xffff
telnet: remote login

- one of the oldest Internet applications (1969)
- *network virtual terminal*: dumb terminal, 7-bit ASCII
- common to FTP, SMTP, finger, whois: CRLF for end-of-line
- *in-band* signaling via IAC (0xff): “interpret as command”
- can do line mode (good for slow connections) or character-by-character mode

telnet: option negotiation

- start with NVT, then either side can propose changes
- Negotiation:
  - WILL sender wants to enable option itself
  - DO sender wants receiver to enable option
  - WONT sender wants to disable option itself
  - DONT sender wants receiver to disable option
- always needs to honor request to disable option
- Typical exchanges:
  - WILL DO sender wants own option, receiver agrees
  - WILL DONT sender wants own option, receives refuses
  - DO WILL sender wants receiver option, receiver agrees
  - DO WONT sender wants receiver option, receiver refuses
  - WONT DONT sender wants to disable, receiver agrees
  - DONT WONT sender wants to disable, receiver agrees
**telnet options codes**

1    echo
2    suppress go ahead
6    timing mark
24   terminal type
31   window size
32   terminal speed
33   remote flow control
34   linemode
36   environment variables

**telnet example**

telnet> toggle options
Will show option processing.
telnet> open tao
Trying 192.35.149.93 ...
Connected to tao.
Escape character is '^]'.
SENT do SUPPRESS GO AHEAD
SENT will TERMINAL TYPE (don’t reply)
RCVD do TERMINAL TYPE (don’t reply)
RCVD will SUPPRESS GO AHEAD (don’t reply)

UNIX(r) System V Release 4.0 (tao)

RCVD will ECHO (reply)
SENT do ECHO (don’t reply)
RCVD do ECHO (reply)
SENT wont ECHO (don’t reply)
RCVD dont ECHO (don’t reply)
ftp: file transfer protocol

- file transfer ↔ file access (NFS)
- copies complete files
- file management (directory, renaming, deleting, …)
- two TCP connections: control (port 21) + data
- ➤ no need for escape characters
- control stays open through ftp session ➤ low throughput, delay
- data connection opened for each file ➤ high throughput

ftp: data representation

File type: ASCII (NVT ASCII), EBCDIC, image (=binary), ≠ 8 bits/byte

Format control (text): nonprint, telnet format, Fortran carriage control

Structure: file, record, page

Transmission mode: stream, block, run-length compressed
ftp: commands

Commands sent as NVT ASCII (4 characters - why?).

- **ABOR** abort previous FTP command, transfer
- **LIST filelist** list files or directories
- **PASS password** password
- **PORT a_1, a_2, a_3, a_4, p_1, p_2** client IP address
- **QUIT** logoff from server
- **RETR file** retrieve (get) a file
- **STOR file** store (put) a file
- **SYST** return system type
- **TYPE type** specify file type: A=ASCII, I=Image
- **USER username** username on server

ftp replies

Reply codes: 3-digit number, optional message
Same idea found in a number of protocols: SMTP, HTTP, ....

- **1yz** positive preliminary reply
- **2yz** positive completion
- **3yz** Positive intermediate reply
- **4yz** Transient negative reply - retry later
- **5yz** Permanent negative reply - don’t retry
- **x0z** syntax errors
- **x1z** information
- **x2z** connection
- **x3z** authentication
- **x4z** unspecified
- **x5z** filesystem status
**ftp: sample error codes**

125  data connection already open; transfer starting.
150  pending BINARY mode data connection for file \(N\) bytes
200  Command OK
226  Transfer complete.
331  Username OK, password required.
425  Can’t open data connection.
452  Error writing file.
500  Syntax error.

**ftp: opening data connection**

1. initiated by ftp client
2. choose ephemeral port option for client; passive open (listen/accept)
3. client sends own address and port to server
4. server does active open
5. new port avoids TIME-WAIT between connections
Anonymous ftp

- pre-web “browsing”
- commonly used for downloading free software, papers
- same as ftp, but user is ftp or anonymous
- use email address as password (or just user@)
- some servers require valid address-to-host mapping for logging

ftp: example

ftp> debug 255
Debugging on (debug=255).
ftp> open gaia.cs.umass.edu
Connected to gaia.cs.umass.edu.
220 gaia.cs.umass.edu FTP server (Version wu-2.4(8) Tue Jul 26
Name (gaia.cs.umass.edu:hgs): hgschulz
----> USER hgschulz
331 Password required for hgschulz.
Password:
----> PASS xxxxxxx
230 User hgschulz logged in.
----> SYST
215 UNIX Type: L8
Remote system type is UNIX.
----> TYPE I
200 Type set to I.
Using binary mode to transfer files.
ftp> ls
----> PORT 192,35,149,52,175,88
200 PORT command successful.
---TYPE A
200 Type set to A.
---LIST
150 Opening ASCII mode data connection for /bin/ls.
total 1012
-rw------- 1 hgschulz dcc 275 Apr 17 1995 .Xauthority ...
226 Transfer complete.
---TYPE I
200 Type set to I.
ftp> get outgoing
---PORT 192,35,149,52,175,107
200 PORT command successful.
---RETR outgoing
ftp> quit
---QUIT
221 Goodbye.

---nfs: network file system

- transparent file access ➔ part of file system tree ➔ application doesn’t know whether file is local or remote
- mostly used in LANs
- client (workstation) ↔ server (disk storage)
- uses Sun RPC with UDP (mostly) or TCP (rarely; for WAN)
**Sun RPC**

Look like function calls to programmer, but...

1. function invokes *client stub* procedure  
2. client stub packages arguments into packets  
3. *server stub* receives message and calls function  
4. on function return, server stub sends result  
5. client stub returns results to program

Advantages:

- no network programming
- retransmission handled by RPC package
- data translation (no htonl(), ...) ➠ XDR: (un)signed integers, booleans, floating point, fixed/variable-length arrays, structures

### RPC request

<table>
<thead>
<tr>
<th>field</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>(length field for TCP)</td>
<td>4</td>
</tr>
<tr>
<td>transaction ID (XID)</td>
<td>4</td>
</tr>
<tr>
<td>call (0)</td>
<td>4</td>
</tr>
<tr>
<td>RPC version (2)</td>
<td>4</td>
</tr>
<tr>
<td>program number</td>
<td>4</td>
</tr>
<tr>
<td>version number</td>
<td>4</td>
</tr>
<tr>
<td>procedure number</td>
<td>4</td>
</tr>
<tr>
<td>credentials</td>
<td>&lt; 400</td>
</tr>
<tr>
<td>verifier</td>
<td>&lt; 400</td>
</tr>
<tr>
<td>parameters</td>
<td></td>
</tr>
</tbody>
</table>
### RPC reply

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction ID (XID) of request</td>
<td>4</td>
</tr>
<tr>
<td>reply (1)</td>
<td>4</td>
</tr>
<tr>
<td>status (0=accepted)</td>
<td>4</td>
</tr>
<tr>
<td>verifier</td>
<td>&lt; 400</td>
</tr>
<tr>
<td>accept status (0=success)</td>
<td>4</td>
</tr>
<tr>
<td>procedure results</td>
<td></td>
</tr>
</tbody>
</table>

### Portmapper

- RPC servers use ephemeral ports
- *portmapper* server registers RPC programs via RPC
- always resides at port 111
- client obtains port numbers of desired program via RPC
Portmapper: example

rpcinfo -p

<table>
<thead>
<tr>
<th>program</th>
<th>vers</th>
<th>proto</th>
<th>port</th>
<th>service</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>4</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>3</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>2</td>
<td>tcp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>4</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>3</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100000</td>
<td>2</td>
<td>udp</td>
<td>111</td>
<td>rpcbind</td>
</tr>
<tr>
<td>100007</td>
<td>3</td>
<td>udp</td>
<td>32773</td>
<td>ypbind</td>
</tr>
<tr>
<td>100003</td>
<td>2</td>
<td>udp</td>
<td>2049</td>
<td>nfs [fixed port!]</td>
</tr>
<tr>
<td>100005</td>
<td>1</td>
<td>udp</td>
<td>32828</td>
<td>mountd</td>
</tr>
<tr>
<td>100005</td>
<td>2</td>
<td>udp</td>
<td>32828</td>
<td>mountd</td>
</tr>
<tr>
<td>100005</td>
<td>1</td>
<td>tcp</td>
<td>32793</td>
<td>mountd</td>
</tr>
<tr>
<td>100005</td>
<td>2</td>
<td>tcp</td>
<td>32793</td>
<td>mountd</td>
</tr>
</tbody>
</table>

NFS

- usually multithreaded (why?)
- stateless
- opaque file handle: created by server; contains local file system info
- mounts server file system at some local location:

  mount -t nfs host:/usr /nfs/host/usr

- UDP: retransmit with exponential backoff, potentially forever
- application not aware of server crashes
NFS Commands

GETATTR  file attributes (directory listing)
SETATTR  set attributes
STATFS   status of filesystem (df)
LOOKUP   given name, return handle
READ     read from file at offset
WRITE    write to a file at offset
CREATE   create a file
REMOVE   remove a file
RENAME   rename a file
LINK     make a hard link to file
SYMLINK  create a symbolic link
READLINK reads symbolic link
MKDIR    make a directory
RMDIR    remove a directory
REaddir  read a directory (ls)

Most commands are idempotent (can be repeated) needed for server crash, UDP packet loss.

Finger

- one-line query (user), returns result (.plan, .project, /etc/passwd name, ...), server closes
- potential security risk (reveals personal info)
- empty line: get all users
- /W user => verbose
- can be recursive: user@host
- used for vending machines

finger @gaia.cs.umass.edu
Login   Name          TTY  Idle When    Office
yajnik  Maya Yajnik  p0   4 Tue 09:39
casetti Claudio Casetti p1  Tue 10:16
zhzhang Zhi-Li Zhang  p2  30 Tue 10:32  GRC A203 413 545-3179
yamamoto Miki yamamoto p3  52 Tue 10:37


whois

- “protocol” like finger, but information returned differs

```bash
whois -h rs.internic.net .gmd-fokus
GMD-FOKUS (NET-GMD-FOKUS-B)
   Hardenbergplatz 2
   D-1000 Berlin 12
   GERMANY

   Netname: GMD-FOKUS-B
   Netnumber: 192.35.152.0

   Coordinator:
      Wasserroth, Stephan (SW111) stephan.wasserroth@GMD.DE
      +49 30 25499 253

   Record last updated on 17-Apr-91.
```

The X window system

- allows remote display/execution
- client: program that wants to draw
- server: screen, keyboard, mouse; serves several local or remote clients
- uses TCP or local Unix IPC
- draw ops, mouse, keyboard events ➔ network packets
- several layers (Xlib, Motif, …)