TELNET vs. telnet

- TELNET is a protocol that provides "a general, bi-directional, eight-bit byte oriented communications facility".
- telnet is a program that supports the TELNET protocol over TCP.

- RLOGIN is a protocol
- rlogin is a program that supports the RLOGIN protocol

TELNET and RLOGIN

- Both are used to remote login across network
- TELNET is standard application provided by almost every TCP/IP implementation
  - Can work between hosts running different OSs
  - Can negotiate options to determine features provided by each end
- RLOGIN is from BSD and was designed to work between Unix systems only

Local login

Only ONE TCP connection !!!
RLOGIN Protocol - Start up

- Specified in RFC 1282
- Use a single TCP connection
- Protocol between client and server after connection is set up
  - Client writes 4 strings to server
    - a byte of 0
    - login name of user on client, terminated by a byte of 0
    - login name of user on server, terminated by a byte of 0
    - name of user's terminal type, a slash, terminal speed, terminated by a byte of 0
      (\0John\0John\0ibmpc3/9600\0)
  - Server responds with a byte of 0
  - Optionally server can ask user for a password
  - Server sends a request to client asking for terminal’s window size

Interactive input

Client

- keystroke
- ACK = 1 WIN = xxx

Server

- echo
- ACK = 1 WIN = xxx

Flow control

- Done by client
- User type Control-S, Control-Q, asking SERVER to stop sending data, or start sending data
  - Control-S \rightarrow stop writing to the network
  - Control-Q \rightarrow start writing to the network

Server to Client Commands

- Need to mark command bytes because only one TCP connection is used
- Use TCP’s urgent mode to mark command bytes
- To send a command to client, server enters urgent mode and makes command byte the last byte of urgent data
  - Four command bytes
    - 0x02: client flushes output
    - 0x10: client stops performing flow control
    - 0x20: client resumes flow control
    - 0x80: client sends current window size to server immediately and notifies server if changing window size
- Out-of-band signaling

TCP Segment Format

<table>
<thead>
<tr>
<th>0</th>
<th>15</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>source port number</td>
<td>destination port number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acknowledgment number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reserved</td>
<td>window size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP checksum</td>
<td>urgent pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>option (if any)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data (if any)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Client to Server Command

- When client receives command 0x80 from server, client needs to send window size to server
- This command needs to be marked
  - 2 bytes of 0xff
  - 2 flag bytes of ASCII char ‘s’
  - 4 16-bit values: #rows, #chars per row, #pixels in X direction, #pixels in Y direction
- In-band signaling
RLOGIN Client Escape

- Talk to client directly instead of sending the input to server
- Start with a tilde ‘~’
- Followed by one of four characters
  - ‘~’: terminate client
  - EOF (Ctrl-D): terminate client
  - Job control suspend char (Ctrl-Z): suspend client
  - Job control delayed suspend char (Ctrl-Y): suspend only client input but keep server output

TELNET Protocol

- Specified in RFC 854
- TCP connection
  - data and control over the same connection.
- Network Virtual Terminal
  - Designed to work between any host and any terminal
  - Negotiated options

TELNET Client and Server

Network Virtual Terminal

- intermediate representation of a generic terminal.
- provides a standard language for communication of terminal control functions.
  - Client maps user’s terminal type to NVT
  - Server maps NVT into its terminal type
- Use NVT ASCII, the 7-bit US variant of ASCII character set, but also allow 8-bit data transmission
Negotiated Options

- All NVTs support a minimal set of capabilities.
- Some terminals have more capabilities than the minimal set.
- The 2 endpoints negotiate a set of mutually acceptable options (character set, echo mode, etc).

The protocol for requesting optional features is well defined and includes rules for eliminating possible negotiation “loops”.

The set of options is not part of the TELNET protocol, so that new terminal features can be incorporated without changing the TELNET protocol.

Option examples

- Four operation modes
  - Half-duplex
  - Character at a time
  - Line at a time
  - Linemode

- character set (EBCDIC vs. ASCII)

Command Structure

- In-band signaling for both directions
  - All TELNET commands and data flow through the same TCP connection.

- Commands start with a special character called the Interpret as Command (IAC) escape character.
  - The IAC code is 255 (0xff).
- Next byte is command byte
  - `<IAC COMMAND XX>`

- If a 0xff is sent as data - it must be followed by another 0xff.

Looking for Commands

- Each receiver must look at each byte that arrives and look for IAC.

- If IAC is found and the next byte is IAC - a single byte is presented to the application/terminal (a 0xff).

- If IAC is followed by any other code - the TELNET layer interprets this as a command.

Command Codes

- IP 243 0xf3
- AO 244 0xf4
- AYT 245 0xf5
- EC 246 0xf6
- EL 247 0xf7
- WILL 251 0xfb
- WON’T 252 0xfc
- DO 253 0xfd
- DON’T 254 0xfe
- IAC 255 0xff
TELNET Option Negotiation

- Either side can send request to other side
  - `<IAC COMMANDID OPTIONID>`

- Four different requests for any given option
  - **WILL**: sender wants to enable option
    - **DO**: receiver says OK
    - **DONT**: receiver says NO
  - **DO**: sender wants receiver to enable option
    - **WILL**: receiver says OK
    - **WONT**: NO
  - **WONT**: sender wants to disable option
    - **DONT**: receiver must say OK
  - **DONT**: sender wants receiver to disable option
    - **WONT** receiver must say OK

- Four different requests for any given option
  - **WILL**: sender wants to enable option
  - **DO**: receiver says OK
  - **DONT**: receiver says NO
  - **DO**: sender wants receiver to enable option
  - **WILL**: receiver says OK
  - **WONT**: NO
  - **DONT**: receiver must say OK
  - **WONT**: sender wants to disable option
  - **DONT**: receiver must say OK
  - **DONT**: sender wants receiver to disable option
  - **WONT** receiver must say OK

- Four different requests for any given option
  - **WILL**: sender wants to enable option
  - **DO**: receiver says OK
  - **DONT**: receiver says NO
  - **DO**: sender wants receiver to enable option
  - **WILL**: receiver says OK
  - **WONT**: NO
  - **DONT**: receiver must say OK
  - **WONT**: sender wants to disable option
  - **DONT**: receiver must say OK
  - **DONT**: sender wants receiver to disable option
  - **WONT** receiver must say OK

TELNET Options

<table>
<thead>
<tr>
<th>Option ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>echo</td>
</tr>
<tr>
<td>3</td>
<td>suppress go ahead</td>
</tr>
<tr>
<td>5</td>
<td>status</td>
</tr>
<tr>
<td>6</td>
<td>timing mark</td>
</tr>
<tr>
<td>24</td>
<td>terminal type</td>
</tr>
<tr>
<td>31</td>
<td>window size</td>
</tr>
<tr>
<td>32</td>
<td>terminal speed</td>
</tr>
<tr>
<td>33</td>
<td>remote flow control</td>
</tr>
<tr>
<td>34</td>
<td>linemode</td>
</tr>
<tr>
<td>36</td>
<td>environment variables</td>
</tr>
</tbody>
</table>

TELNET Suboption Negotiation

- Used for options that require more information than just “enable” or “disable”
- For example,
  - client: `<IAC, WILL, 24>
  - server: `<IAC, DO, 24>
  - server: `<IAC, SB, 24, 1, IAC, SE>
  - client: `<IAC, SB, 24, 0, 'I', 'B', 'M', 'P', 'C', IAC, SE>

Control Functions

- TELNET includes support for a series of control functions commonly supported by servers.
- This provides a uniform mechanism for communication of (the supported) control functions.

Control Functions

- **Interrupt Process (IP)**
  - suspend/abort process.
- **Abort Output (AO)**
  - process can complete, but send no more output to user’s terminal.
- **Are You There (AYT)**
  - check to see if system is still running.

More Control Functions

- **Erase Character (EC)**
  - delete last character sent
  - typically used to edit keyboard input.
- **Erase Line (EL)**
  - delete all input in current line.
TELNET Client Escape

- Talk to client directly instead of sending the input to server
- Normal client escape char is Ctrl-]
- Can type commands to change settings or print information

TELNET vs. TCP

- Not all TCP servers talk TELNET (most don't)
- You can use the telnet program to play with these servers, but the fancy commands won't do anything.
  - type ^], then "help" for a list of fancy TELNET stuff you can do in telnet.

Playing with TELNET

- You can use the telnet program to play with the TELNET protocol.
- telnet is a generic TCP client.
  - Sends whatever you type to the TCP socket.
  - Prints whatever comes back through the TCP socket.
  - Useful for testing TCP servers (ASCII based protocols).

Some TCP Servers you can play with

- Many Unix systems have these servers running (by default):
  - echo port 7
  - discard port 9
  - daytime port 13
  - chargen port 19

Security

- Does not encrypt any data sent over the connection
  - Including password
- No authentication – man in the middle attack
- Daemons have vulnerabilities

Secure Shell (SSH)

- Operate over TCP/IP connection or other reliable but insecure transport layers to support secure remote login, secure file transfer and other secure services
- Can automatically encrypt, authenticate, and compress transmitted data
- Three major components
  - Transport Layer Protocol to provide server authentication, confidentiality, and integrity
  - User Authentication Protocol to provide authentication of client-side user to server
  - Connection Protocol to multiplex encrypted channels to logical channels
FTP Client and Server

FTP Commands
- Commands are sent in NVT ASCII with a pair of CR, LF at end of each line
- Commands are 3 or 4 bytes of ASCII chars, some require optional arguments

Access Control Commands
- USER username specify user
- PASS password specify password
- CWD change directory
- CDUP change directory to parent
- QUIT logout

Control Connection
- The control connection is the “well known” service.
- The control connection uses the TELNET protocol.
- Commands and replies are all line oriented text (default is ASCII).

FTP Commands
- PORT publish local data port
- PASV server should listen
- TYPE type establish data representation
- MODE establish transfer mode
- STRU establish file structure

File Transfer Protocol (FTP)
- Internet standard for file transfer
- Designed to work between different systems but support limited number of file types and structures
- Use two TCP connections
  - Control connection
    - Client active open to TCP port 21 of server
    - Stay up all the time of communication
    - IP type of service (TOS) should be “minimize delay”
  - Data connection
    - Created each time a file is transferred
    - IP TOS should be “maximize throughput”
Service Commands

- RETR filename retrieve file
- STOR filename send file
- STOU send file and save as unique
- APPE send file and append
- ABOR abort prev. service command
- PWD print working directory
- LIST transfer list of files over data link

FTP Replies

- All replies are sent over control connection.
- Replies are a single line containing
  - 3 digit status code (sent as 3 numeric chars).
  - text message.
- The FTP spec. includes support for multiline text replies.

FTP Reply Status Code

First digit of status code indicates type of reply:
- ‘1’: Positive Preliminary Reply (got it, but wait).
- ‘2’: Positive Completion Reply (success).
- ‘3’: Positive Intermediate Reply (waiting for more information).
- ‘4’: Transient Negative Completion (error - try again).
- ‘5’: Permanent Negative Reply (error - can’t do).

FTP Data Representation

- Four dimensions determine how file is transferred and stored
  - File type: ASCII file, EBCDIC file, binary file, local file
  - Format control: nonprint, telnet format control, Fortran carriage control
  - Structure: file structure, record structure, page structure
  - Transmission mode: stream mode, block mode, compressed mode
- Typical implementation restricts to ASCII or binary, nonprint, file structure, stream mode

FTP Connection Management

- Three uses for data connection
  - Sending file from client to server
  - Sending file from server to client
  - Sending a listing of files or directories from server to client
- Client chooses an ephemeral port, sends it to server across control connection, and passive open for data connection
- Server receives this port number from control connection, and active open to that port on client
Anonymous FTP

- A server can allow anyone to login and use FTP to transfer files
- To use anonymous FTP, login with “anonymous” as username and an email address as password
- Some anonymous FTP server denies client from a host that does not have valid hostname

Assignment & Next time

- Reading:
  - TI 26, 27 **
  - RFC 854 TELEN
  - RFC 959 FTP

- Next Lecture:
  - UDP programming