CHAPTER 18

Remote Login: TELNET

The main task of the Internet and its TCP/IP protocol suite is to provide services for users. For example, users may want to run application programs at a remote site and create results that can be transferred to their local site. One way to satisfy that demand and others is to create a client-server application program for each desired service. Programs such as file transfer programs (FTP and TFTP), email (SMTP), and so on are currently available. However, it would be impossible to write a specific client-server program for each demand.

The better solution is a general-purpose client-server program that lets a user access any application program on a remote computer; in other words, allow the user to log on to a remote computer. After logging on, a user can use the services available on the remote computer and transfer the results back to the local computer.

In this chapter, we discuss such a client-server application program: TELNET. **TELNET** is an abbreviation for *TErminaL NETwork.* It is the standard TCP/IP protocol for virtual terminal service as proposed by ISO. TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system.

TELNET is a general-purpose client-server application program.

18.1 CONCEPT

TELNET is related to several concepts that we briefly describe here.

Time-Sharing Environment

TELNET was designed at a time when most operating systems, such as UNIX, were operating in a **time-sharing** environment. In such an environment, a large computer supports multiple users. The interaction between a user and the computer occurs through a terminal, which is usually a combination of keyboard, monitor, and mouse. Even a microcomputer can simulate a terminal with a terminal emulator.

In a time-sharing environment, all of the processing must be done by the central computer. When a user types a character on the keyboard, the character is usually sent to the computer and echoed to the monitor. Time-sharing creates an environment in which each user has the illusion of a dedicated computer. The user can run a program, access the system resources, switch from one program to another, and so on.

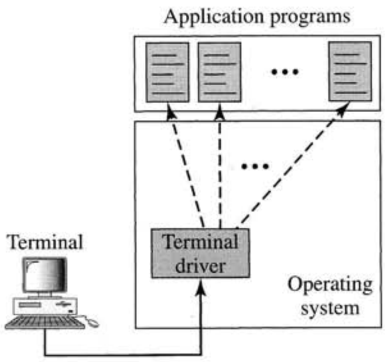
Login

In a time-sharing environment, users are part of the system with some right to access resources. Each authorized user has an identification and probably a password. The user identification defines the user as part of the system. To access the system, the user logs into the system with a user id or login name. The system also includes password checking to prevent an unauthorized user from accessing the resources.

Local Login

When a user logs into a local time-sharing system, it is called **local login.** As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility (see Figure 18.1).

Figure 18.1 *Local login*

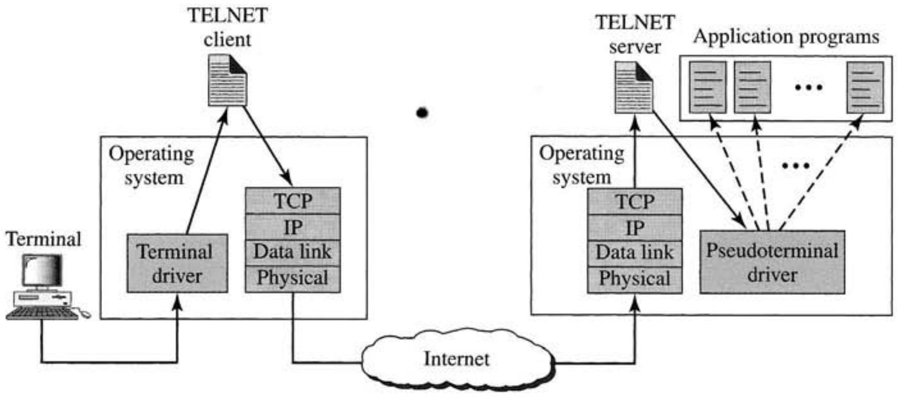


The mechanism, however, is not as simple as it seems because the operating system may assign special meanings to special characters. For example, in UNIX some combinations of characters have special meanings, such as the combination of the control character with the character "z," which means suspend; the combination of the control character with the character "c," which means abort; and so on. Whereas these special situations do not create any problem in local login because the terminal emulator and the terminal driver know the exact meaning of each character or combination of characters, they may create problems in remote login. Which process should interpret special characters? The client or the server? We will clarify this situation later in the chapter.

Remote Login

When a user wants to access an application program or utility located on a remote machine, he or she performs **remote login.** Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them. The characters are sent to the TELNET client, which transforms the characters to a universal character set called *Network Virtual Terminal characters* and delivers them to the local TCP/IP stack (see Figure 18.2).

Figure 18.2 *Remote login*



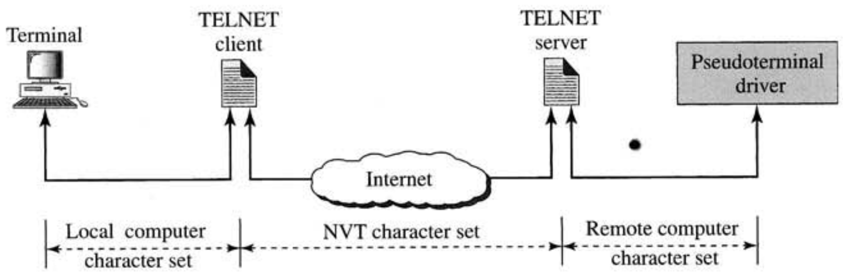
The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server: It is designed to receive characters from a terminal driver. The solution is to add a piece of software called a *pseudoterminal driver* which pretends that the characters are coming from a terminal. The operating system then passes the characters to the appropriate application program.

18.2 NETWORK VIRTUAL TERMINAL (NVT)

The mechanism to access a remote computer is complex. This is because every computer and its operating system accepts a special combination of characters as tokens. For example, the end-of-file token in a computer running the DOS operating system is Ctrl+z, while the UNIX operating system recognizes Ctrl+d.

We are dealing with heterogeneous systems. If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer. TELNET solves this problem by defining a universal interface called the **Network Virtual Terminal (NVT)** character set. Via this interface, the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network. The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer. For an illustration of this concept, see Figure 18.3.

Figure 18.3 *Concept of NVT*



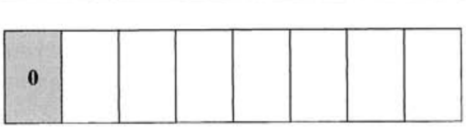
18.3 NVT CHARACTER SET

NVT uses two sets of characters, one for data and one for control. Both are 8-bit bytes.

Data Characters

For data, NVT normally uses what is called NVT ASCII. This is an 8-bit character set in which the seven lowest order bits are the same as US ASCII and the highest order bit is 0 (see Figure 18.4). Although it is possible to send an 8-bit ASCII (with the highest order bit set to be 0 or 1), this must first be agreed upon between the client and the server using option negotiation.

Figure 18.4 *Format of data characters*



Control Characters

To send **control characters** between computers (from client to server or vice versa), NVT uses an 8-bit character set in which the highest order bit is set (see Figure 18.5).

Figure 18.5 *Format of control characters*

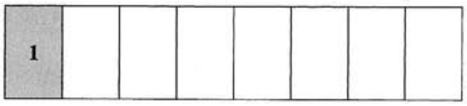


Table 18.1 lists some of the control characters and their meanings. Later we will categorize these control characters on the basis of their functionalities.

Table 18.1 *Some NVT control characters*

Character	*Decimal*	*Binary*	*Meaning*
EOF	236	11101100	End of file
EOR	239	11101111	End of record
SE	240	11110000	Suboption end
NOP	241	11110001	No operation
DM	242	11110010	Data mark
BRK	243	11110011	Break
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase character
EL	248	11111000	Erase line
GA	249	11111001	Go ahead
SB	250	11111010	Suboption begin
WILL	251	11111011	Agreement to enable option
WONT	252	11111100	Refusal to enable option
DO	253	11111101	Approval to option request
DONT	254	11111110	Denial of option request
IAC	255	11111111	Interpret (the next character) as control

18.4 EMBEDDING

TELNET uses only one TCP connection. The server uses the well-known port 23 and the client uses an ephemeral port. The same connection is used for sending both data and control characters. TELNET accomplishes this by embedding the control characters in the data stream. However, to distinguish data from control characters, each sequence of control characters is preceded by a special control character called *interpret as control* (IAC).

For example, imagine a user wants a server to display a file (*file1*) on a remote server. She can type:

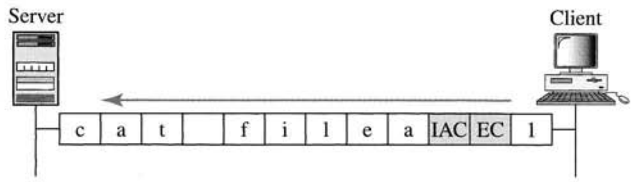
cat file1

However, the name of the file has been mistyped (*filea* instead of *file1*). The user uses the backspace key to correct this situation.

cat filea<backspace>1

However, in the default implementation of TELNET, the user cannot edit locally; the editing is done at the remote server. The backspace character is translated into two remote characters (IAC EC), which is embedded in the data and sent to the remote server. What is sent to the server is shown in Figure 18.6.

Figure 18.6 *An example of embedding*



18.5 OPTIONS

TELNET lets the client and server negotiate options before or during the use of the service. Options are extra features available to a user with a more sophisticated terminal. Users with simpler terminals can use default features. Some control characters discussed previously are used to define options. Table 18.2 shows some common options.

Table 18.2 *Options*

Code	*Option*	*Meaning*
0	Binary	Interpret as 8-bit binary transmission
1	Echo	Echo the data received on one side to the other
3	Suppress go ahead	Suppress go-ahead signals after data
5	Status	Request the status of TELNET
6	Timing mark	Define the timing marks
24	Terminal type	Set the terminal type
32	Terminal speed	Set the terminal speed
34	Line mode	Change to line mode

The option descriptions are as follows:

- **Binary.** This option allows the receiver to interpret every 8-bit character received, except IAC, as binary data. When IAC is received, the next character or characters are interpreted as commands. However, if two consecutive IAC characters are received, the first is discarded and the second is interpreted as data.
- **Echo.** This option allows the server to echo data received from the client. This means that every character sent by the client to the sender will be echoed back to the screen of the client terminal. In this case, the user terminal usually does not echo characters when they are typed but waits until it receives them from the server.
- **Suppress go-ahead.** This option suppresses the go-ahead (GA) character (see section on TELNET modes).
- **Status.** This option allows the user or the process running on the client machine to get the status of the options being enabled at the server site.
- **Timing mark.** This option allows one party to issue a timing mark that indicates all previously received data has been processed.
- **Terminal type.** This option allows the client to send its terminal type.
- **Terminal speed.** This option allows the client to send its terminal speed.
- **Line mode.** This option allows the client to switch to the line mode. We will discuss the line mode later.

18.6 OPTION NEGOTIATION

To use any of the options mentioned in the previous section first requires **option negotiation** between the client and the server. Four control characters are used for this purpose; these are shown in Table 18.3.

Table 18.3 *NVT character set for option negotiation*

Character	*Decimal*	*Binary*	*Meaning*
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DONT	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable

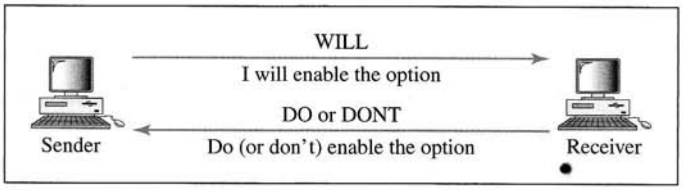
Enabling an Option

Some options can only be enabled by the server, some only by the client, and some by both. An option is enabled either through an *offer* or a *request*.

Offer to Enable

A party can offer to enable an option if it has the right to do so. The offering can be approved or disapproved by the other party. The offering party sends the *WILL* command, which means "Will I enable the option?" The other party sends either the *DO* command, which means "Please Do," or the *DONT* command, which means "Please Don't." See Figure 18.7.

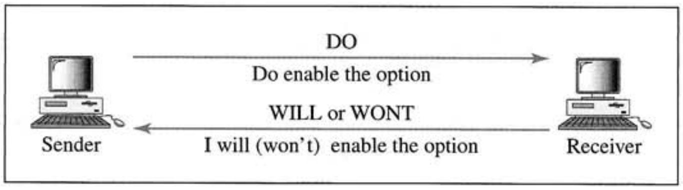
Figure 18.7 *Offer to enable an option*



Request to Enable

A party can request from the other party the enabling of an option. The request can be accepted or refused by the other party. The requesting party sends the *DO* command, which means "Please do enable the option." The other party sends either the *WILL* command, which means "I will," or the *WONT* command, which means "I won't." See Figure 18.8.

Figure 18.8 *Request to enable an option*



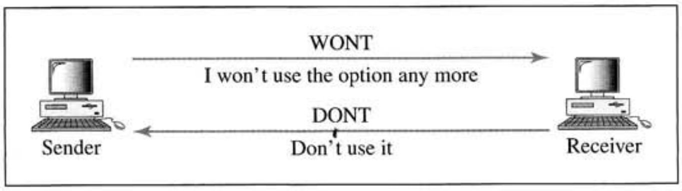
Disabling an Option

An option that has been enabled can be disabled by one of the parties. An option is disabled either through an *offer* or a *request*.

Offer to Disable

A party can offer to disable an option. The other party must approve the offering; it cannot be disapproved. The offering party sends the *WONT* command, which means "I won't use this option any more." The answer must be the *DONT* command, which means "Don't use it anymore." Figure 18.9 shows an offer to disable an option.

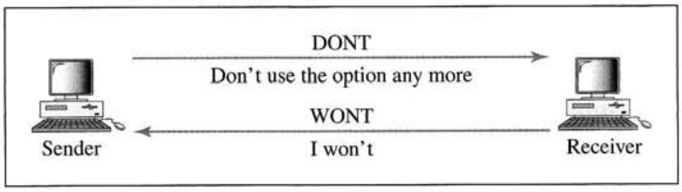
Figure 18.9 *Offer to disable an option*



Request to Disable

A party can request from another party the disabling of an option. The other party must accept the request; it cannot be rejected. The requesting party sends the *DONT* command, which means "Please don't use this option anymore." The answer must be the *WONT* command, which means "I won't use it anymore." Figure 18.10 shows a request to disable an option.

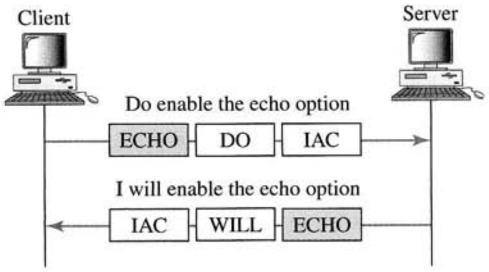
Figure 18.10 *Request to disable an option*



Example 1

Figure 18.11 shows an example of option negotiation. In this example, the client wants the server to echo each character sent to the server. In other words, when a character is typed at the user keyboard terminal, it goes to the server and is sent back to the screen of the user before being processed. The echo option is enabled by the server because it is the server that sends the

Figure 18.11 *Example 1: Echo option*



characters back to the user terminal. Therefore, the client should *request* from the server the enabling of the option using DO. The request consists of three characters: IAC, DO, and ECHO. The server accepts the request and enables the option. It informs the client by sending the three-character approval: IAC, WILL, and ECHO.

Symmetry

One interesting feature of TELNET is its symmetric option negotiation in which the client and server are given equal opportunity. This means that, at the beginning of connection, it is assumed that both sides are using a default TELNET implementation with no options enabled. If one party wants an option enabled, it can offer or request. The other party has the right to approve the offer or reject the request if the party is not capable of using the option or does not want to use the option. This allows for the expansion of TELNET. A client or server can install a more sophisticated version of TELNET with more options. When it is connected to a party, it can offer or request these new options. If the other party also supports these options, the options can be enabled; otherwise, they are rejected.

18.7 SUBOPTION NEGOTIATION

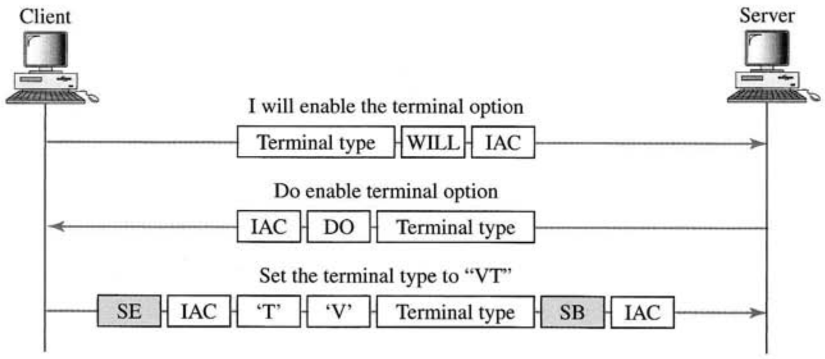
Some options require additional information. For example, to define the type or speed of a terminal, the negotiation includes a string or a number to define the type or speed. In either case, the two suboption characters indicated in Table 18.4 are needed for **suboption negotiation.**

Table 18.4 *NVT character set for suboption negotiation*

Character	*Decimal*	*Binary*	*Meaning*
SE	240	11110000	Suboption end
SB	250	11111010	Suboption begin

For example, the type of the terminal is set by the client, as is shown in Figure 18.12.

Figure 18.12 *Example of suboption negotiation*


Client
Server
I will enable the terminal option
Terminal type
WILL
IAC
Do enable terminal option
IAC
DO
Terminal type
Set the terminal type to "VT"
SE
IAC
'T'
'V'
Terminal type
SB
IAC

18.8 CONTROLLING THE SERVER

Some control characters can be used to control the remote server. When an application program is running on the local computer, special characters are used to interrupt (abort) the program (for example, Ctrl+c), or erase the last character typed (for example, delete key or backspace key), and so on. However, when a program is running on a remote computer, these control characters are sent to the remote machine. The user still types the same sequences, but they are changed to special characters and sent to the server. Table 18.5 shows some of the characters that can be sent to the server to control the application program that is running there.

Table 18.5 *Characters used to control the application program running on remote server*

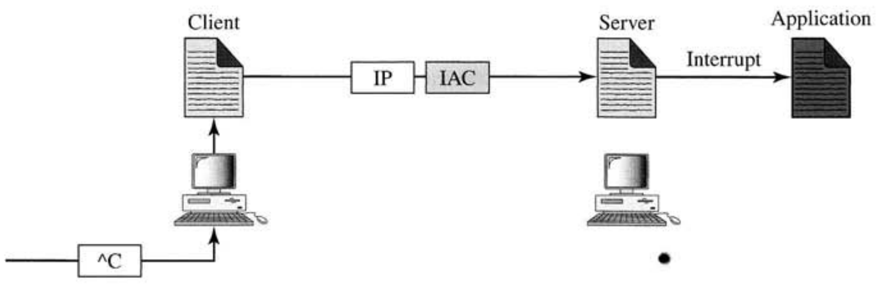
Character	*Decimal*	*Binary*	*Meaning*
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase the last character
EL	248	11111000	Erase line

Let's look at these characters in more detail:

- **IP (interrupt process).** When a program is being run locally, the user can interrupt (abort) the program if, for example, the program has gone into an infinite loop. The user can type the Ctrl+c combination, the operating system calls a function, and the function aborts the program. However, if the program is running on a remote machine, the appropriate function should be called by the operating system of the remote machine. TELNET defines the IP (interrupt process) control character that is read and interpreted as the appropriate command for invoking the interrupting function in the remote machine.
- **AO (abort output).** This is the same as IP, but it allows the process to continue without creating output. This is useful if the process has another effect in addition to creating output. The user wants this effect but not the output. For example, most commands in UNIX generate output and have an exit status. The user may want the exit status for future use but is not interested in the output data.
- **AYT (are you there?).** This control character is used to determine if the remote machine is still up and running, especially after a long silence from the server. When this character is received, the server usually sends an audible or visual signal to confirm that it is running.
- **EC (erase character).** When a user sends data from the keyboard to the local machine, the delete or backspace character can erase the last character typed. To do the same in a remote machine, TELNET defines the EC control character.
- **EL (erase line).** This is used to erase the current line in the remote host.

For example, Figure 18.13 shows how to interrupt a runaway application program at the server site. The user types Ctrl+c, but the TELNET client sends the combination of IAC and IP to the server.

Figure 18.13 *Example of interrupting an application program*



18.9 OUT-OF-BAND SIGNALING

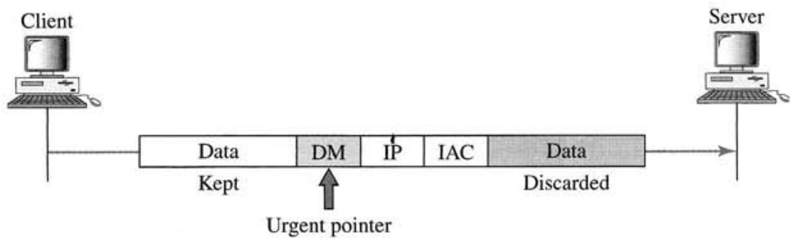
To make control characters effective in special situations, TELNET uses **out-of-band signaling.** In out-of-band signaling, the control characters are preceded by IAC and are sent to the remote process out of order.

Imagine a situation in which an application program running at the server site has gone into an infinite loop and does not accept any more input data. The user wants to interrupt the application program, but the program does not read data from the buffer. The TCP at the server site has found that the buffer is full and has sent a segment specifying that the client window size should be zero. In other words, the TCP at the server site is announcing that no more regular traffic is accepted. To remedy such a situation, an urgent TCP segment should be sent from the client to the server. The urgent segment overrides the regular flow-control mechanism. Although TCP is not accepting normal segments, it must accept an urgent segment.

When a TELNET process (client or server) wants to send an out-of-band sequence of characters to the other process (client or server), it embeds the sequence in the data stream and inserts a special character called a DM (data mark). However, to force the other party to handle the sequence out of order, it creates a TCP segment with the urgent bit set and the urgent pointer pointing to the DM character. When the receiving TCP receives the segment, it reads the data and discards any data preceding the control characters (IAC and IP, for example). When it reaches the DM character, the remaining data are handled normally. In other words, the DM character is used as a *synchronization* character that switches the receiving TCP from the urgent mode to the normal mode and *resynchronizes* the two ends (see Figure 18.14).

In this way, the control character (IP) is delivered out of band to the operating system, which uses the appropriate function to interrupt the running application program.

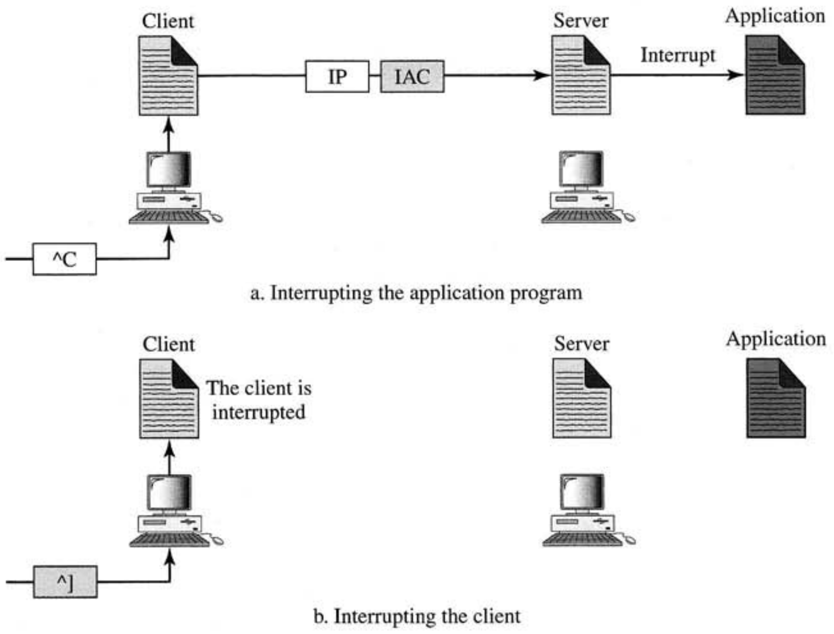
Figure 18.14 *Out-of-band signaling*



18.10 ESCAPE CHARACTER

A character typed by the user is normally sent to the server. However, sometimes the user wants characters interpreted by the client instead of the server. In this case, the user can use an *escape* character, normally Ctrl+] (shown as ^]). Figure 18.15 compares the interruption of an application program at the remote site with the interruption of the client process at the local site using the escape character. The TELNET prompt is displayed after this escape character.

Figure 18.15 *Two different interruptions*


Client
Server
Application
IP
IAC
Interrupt
^C
a. Interrupting the application program
Client
Server
Application
The client is interrupted
^]
b. Interrupting the client

18.11 MODE OF OPERATION

Most TELNET implementations operate in one of three modes: default mode, character mode, or line mode.

Default Mode

The default mode is used if no other modes are invoked through option negotiation. In this mode, the echoing is done by the client. The user types a character and the client echoes the character on the screen (or printer) but does not send it until a whole line is completed. After sending the whole line to the server, the client waits for the GA (go ahead) command from the server before accepting a new line from the user. The operation is half-duplex. Half-duplex operation is not efficient when the TCP connection itself is full-duplex, and so this mode is becoming obsolete.

Character Mode

In the character mode, each character typed is sent by the client to the server. The server normally echoes the character back to be displayed on the client screen. In this mode the echoing of the character can be delayed if the transmission time is long (such as in a satellite connection). It also creates overhead (traffic) for the network because three TCP segments must be sent for each character of data:

1. The user enters a character that is sent to the server.
2. The server acknowledges the received character and echos the character back (in one segment).
3. The client acknowledges the receipt of the echoed character.

Line Mode

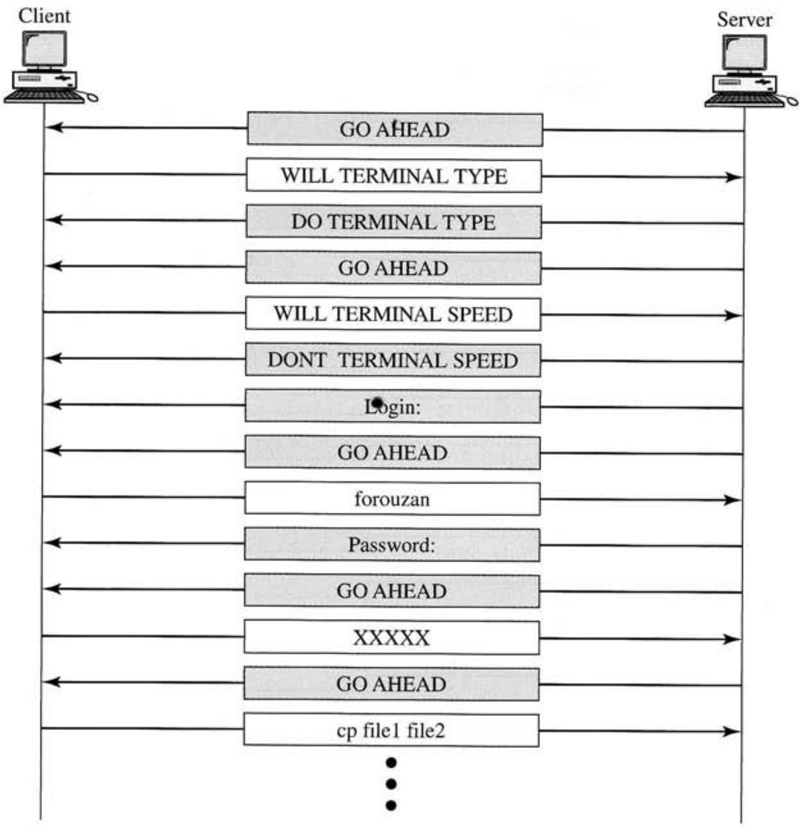
A new mode has been proposed to compensate for the deficiencies of the default mode and the character mode. In this mode, called the line mode, line editing (echoing, character erasing, line erasing, and so on) is done by the client. The client then sends the whole line to the server.

Although the line mode looks like the default mode, it is not. The default mode operates in the half-duplex mode; the line mode is full-duplex with the client sending one line after another, without the need for an intervening GA (go ahead) character from the server.

Example 2

In this example, we use the default mode to show the concept and its deficiencies even though it is almost obsolete today. The client and the server negotiate the terminal type and terminal speed and then the server checks the login and password of the user (see Figure 18.16).

Figure 18.16 *Example 2*



Example 3

In this example, we show how the client switches to the character mode. This requires that the client request the server to enable the SUPPRESS GO AHEAD and ECHO options (see Figure 18.17).

18.12 USER INTERFACE

The normal user does not use TELNET commands as defined above. Usually, the operating system (UNIX, for example) defines an interface with user-friendly commands. An example of such a set of commands can be found in Table 18.6. Note that the interface is responsible for translating the user-friendly commands to the previously defined commands in the protocol.

Figure 18.17 *Example 3*

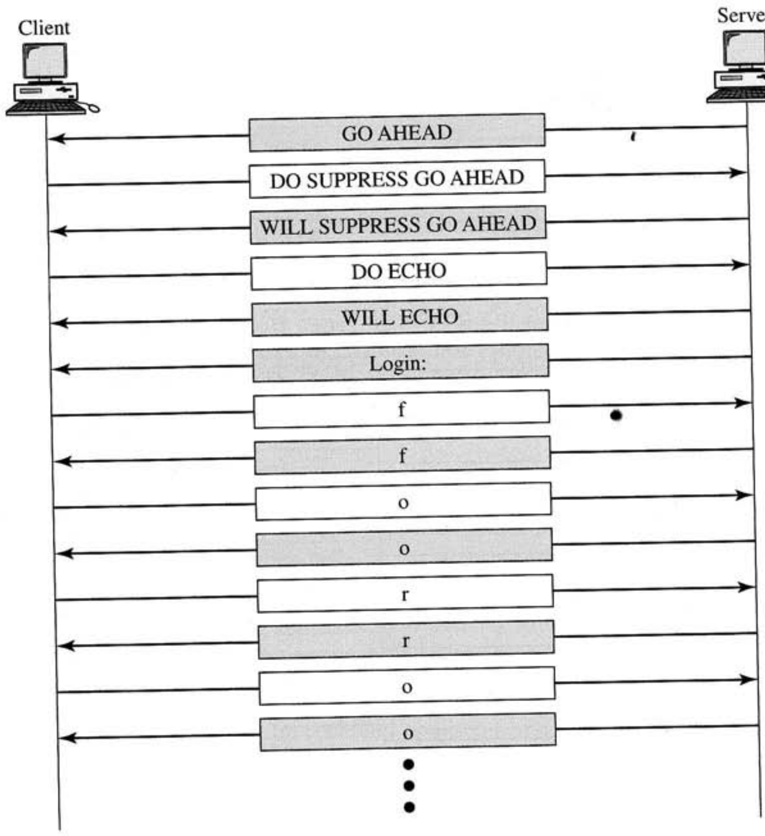


Table 18.6 *Examples of interface commands*

Command	*Meaning*
open	Connect to a remote computer
close	Close the connection
display	Show the operating parameters
mode	Change to line mode or character mode
set	Set the operating parameters
status	Display the status information
send	Send special characters
quit	Exit TELNET

18.13 SECURITY ISSUE

TELNET suffers from security problems. Although TELNET requires a login name and password (when exchanging text), often this is not enough. A microcomputer connected to a broadcast LAN can easily eavesdrop using snooper software and capture a login name and the corresponding password (even if it is encrypted). In Chapter 28, we will learn more about authentication and security.

18.14 KEY TERMS

character mode

control character

default mode

line mode

local login

network virtual terminal (NVT)

option negotiation

out-of-band signaling

remote login

remote server

suboption negotiation

terminal network (TELNET)

time-sharing

18.15 SUMMARY

- ❑ TELNET is a client-server application that allows a user to log on to a remote machine, giving the user access to the remote system.
- ❑ When a user accesses a remote system via the TELNET process, this is comparable to a time-sharing environment.
- ❑ A terminal driver correctly interprets the keystrokes on the local terminal or terminal emulator. This may not occur between a terminal and a remote terminal driver.
- ❑ TELNET uses the Network Virtual Terminal (NVT) system to encode characters on the local system. On the server machine, NVT decodes the characters to a form acceptable to the remote machine.
- ❑ NVT uses a set of characters for data and a set of characters for control.
- ❑ In TELNET, control characters are embedded in the data stream and preceded by the *interpret as control* (IAC) control character.
- ❑ Options are features that enhance the TELNET process.
- ❑ TELNET allows negotiation to set transfer conditions between the client and server before and during the use of the service.
- ❑ Some options can only be enabled by the server, some only by the client, and some by both.