SFWR 4C03: Computer Networks and Computer Security	Feb 23-26 2004
Lecturer: Kartik Krishnan	Lectures 19-21

Remote login (Telnet):

Telnet permits a user to connect to an account on a remote machine. A client program running on the user's machine communicates using the Telnet protocol with a server program running on the remote machine.

The Telnet application:

The user (say Kartik) has an account on both the local and remote machines. For example, Kartik on *farkas.mcmaster.ca* types *telnet optlab.cas.mcmaster.ca* at his user prompt. Here, *farkas.mcmaster.ca* is the client and *optlab.cas.mcmaster.ca* is the server. The Telnet client would perform a *methotrexate()* call to determine the IP address of *optlab.cas.mcmaster.ca*. Then the client would create a socket to communicate with the Telnet server. The server prompts Kartik for a login identifier - the name of the user's account on the remote server followed by a password. The Telnet users interact with the remote machine in the same way they would interact with their local machine. The client relays Kartik's keystrokes to the remote server, and the remote server displays them on its pseudo terminal (which is actually the display screen on the client machine).

The Telnet protocol:

The Telnet client program performs two important functions - interacting with the user terminal on the local host and exchanging messages with the Telnet server. The client connects to port 23 on the remote machine, which is the port number reserved for Telnet servers. The TCP connection persists for the duration of the login session. The client and the server maintain the connection, even when the user interrupts the transfer of data, for example by hitting *cntl-C*.

Since Telnet is designed to work over two hosts on different platforms, the protocol assumes that the two hosts run a Network Virtual Terminal (NVT). The TCP connection is set up across these two NVT terminals. The NVT is a very simple character device with a keyboard and a printer - data typed by the user on the keyboard is translated by the client software into NVT format and sent via its NVT terminal to the server, and data received in NVT format from the server is translated by the client into the local machine format and output to the printer (see Figure 25.2 of Comer).

The NVT terminals on the two hosts exchange data in the 7-bit U.S. variant of the ASCII format, with each character sent as an octet with the first bit set to

0. Some control information, such as end-of-line indication, is transmitted as the 2 character sequence CR (carriage return) followed by an LF (linefeed). Each Telnet control message starts with the special octet (Interpret as Command (IAC) octet of all 1s) to ensure that the recipient interprets the subsequent octets as a command. Otherwise, each octet is interpreted as data (e.g., a user keystroke). Sending control messages on the same connection as the data is referred to as *inband signaling*. The initial control messages between the client and the server are used to exchange information about their capabilities (Telnet option negotiation). For example, the client may indicate the type and speed of its terminal, and whether data is to be sent one character or one line at a time. More details on Telnet option negotiation are available in Section 25.8 of Comer. After the capabilities exchange, the server instructs the client to send a login identifier and password. Once the authentication completes, the user interacts directly with the remote machine. The client application relays user keystrokes to the server, and the server relays the output back to the client, using inband signaling, with the interpretation that commands follow the IAC octet of all ones.

Telnet cannot rely on the conventional data stream alone to carry such control sequences between client and server. To see this, consider the previous interaction of Kartik with the remote server *optlab.cas.mcmaster.ca*. Kartik is writing into a file on this remote machine, and suppose the TCP connection set up on remote server (during the Telnet session) is misbehaving. As a result, none of Kartik's subsequent keystrokes are read, and echoed on the local screen. Now, Kartik wants to terminate writing into this file. He does this by typing *cntl-C* on his keyboard, which relays the control sequence *IAC IP* to the remote machine. Suppose also that receiving hosts' TCP buffers are full, and it cannot receive any further information from the sender (it discloses this over the TCP connection by advertising a window size of zero). So to ensure that Kartik's *cntl-C* command gets through, Telnet uses an *out of band* signal. So, in this case, the Telnet client sets a TCP segment with the URG flag bit set to 1. This TCP segment bypasses flow control and reaches the server immediately. The server discards all data until it finds the control sequence *cntl-C* following the IAC octet, and Kartik's session with the file on the remote machine is terminated. However, the Telnet session on the server is not terminated, and the server is back to its normal operation.

File Transfer Protocol (FTP):

FTP allows a user to copy files to and from a remote machine. The client program also sends commands to the server program to coordinate the copying of files between the two machines on behalf of the user.

The FTP application:

The FTP client connects to the remote machine which prompts the user to enter a login identifier and a password. However, some users may not have their own accounts on the remote machine. To grant access to a broad set of users, many FTP servers have a special account (e.g. *anonymous*) that does not require password information. Instead, the user logs in using *guest* or his email address as password. The FTP server coordinates access to a collection of files in various directories. In case of anonymous FTP, the server typically

has a special directory, with one or more subdirectories, that can be accessed by the client. The user logged into the FTP server can traverse through the directories of files on the remote machine, and send or receive files. This is typically done via the command-line interface. The interface may also allow the client to send or receive multiple files with a single command. Recent FTP client applications provide a menu-based graphical user interface. For instance, a Web browser allows users to specify the desired file as an URL (e.g., ftp://ftp.optlab.cas.mcmaster.ca/midterm.pdf). In this case, the web browser connects to the FTP server as an anonymous user and sends a sequence of FTP commands to fetch the requested file.

The FTP protocol:

FTP differs from other applications such as Telnet since it uses separate TCP connections for control and data. Recall that in Telnet both control information and data are sent over the same TCP connection using in-band signaling. The two TCP connections in FTP are:

1. The control connection is established in the normal client-server fashion. In this case, the server does a *passive open* (is listening) on port 21 for FTP, and waits for the client connection. The client does an *active open* (the 2nd handshake in a TCP connection) to establish the control connection. The client uses an ephemeral port number (above 1023) for the control connection. This control connection stays up for the entire time that the client communicates with this server. This connection is used for commands from the client to the server and for the server's replies. The IP type of service for the control connection should be to minimize delay in passing these commands over the TCP connection.

2. A data connection is created each time a file is transferred between the client and the server. The IP type of service for the data connection should be to maximize throughput since this connection is file transfer, and we want to send this entire file over a high bandwidth line.

The specification of FTP includes more than 30 different commands, which are transmitted over the control connection in NVT ASCII format. The commands are not case-sensitive and may have arguments; each command ends with a two character sequence of a carriage return (CR) followed by a line feed (LF). It must be emphasized here that these commands are different from the commands typed by the user at the interface provided by the client. Transferring a single file for instance requires only a single user-level command (e.g., *put* or *get*), but this single command triggers the client to send a set of FTP commands to the server. The FTP server responds to each command with a three-digit reply code (for the FTP client) and an optional text message (for the user).

The control connection persists over a sequence of FTP commands, as the client and the server continue their dialogue. The typical interaction starts with a command that identifies the user on the server machine followed by another command to send the user password. The arguments for these commands are gleaned from the user's input (his account name and password). The server uses this information to verify whether the user has an authorized account on the remote machine, and in the case of anonymous FTP decides on the set of

directories to which the anonymous guest has access. The next set of commands depend on the user request to send, receive, or view the files in a present directory.

The actual file (data) transfer uses a separate TCP connection established by the host sending the data. For instance if the user wants to retrieve the file *midterm.pdf* from the remote server, the server initiates the creation of the TCP data connection. In case, the user wants to put a file into the remote machine, it is the client who initiates the creation of the TCP connection. The data connection is usually established on port 20 on the server machine. In the former case (when the file is to be retrieved from the server), the server does not know the destination port for the FTP client. So before sending the command to retrieve the file, the client instructs its operating system to allocate a port number (above 1023) for such a transaction. This information is given to the server via the control connection. The data connection is created (using the usual TCP 3 way handshake), and the server writes the contents of the file, and closes the connection. The client reads the bytes from its socket upto the end of file (EOF) character. Also, unlike Telnet, FTP does not require the data transfer to 7 bit ASCII characters (NVT format); it actually permits a wider range of data types including binary files. The client requests the form of data transfer using the control connection. In practice, each data transfer requires a separate TCP connection. In contrast, the control connection can persist across multiple data transfers. An example anonymous FTP session is shown in Section 26.10 of Comer. Use this session to distinguish between the control and data TCP connections. Also, can you identify our various discussions on the FTP protocol in this session?.

Simple Mail Transfer Protocol (SMTP):

SMTP is used to send an email message from a local mail server to a remote mail server. In addition, SMTP may be used from the user's mail agent to the local mail server, although alternatives like POP3 are also available.

The electronic mail application:

Delivering an email message from one user to another involves several components. A user invokes an *user agent* to send and receive email messages. Examples of these include *elm*, *pine* etc. The agent also supports a variety of other features such as mail folders, email aliases, and an editor for composing messages. In addition to providing an interface for the user, the agent also coordinates the interaction with a local mail server, also called the *message transfer agent*. The message transfer agent maintains each user's mailbox and coordinates the exchange of messages with message transfer agents in other locations. For example, Kartik's email address is *kartik@optlab.cas.mcmaster.ca*, so there is a mailbox called *kartik* on the local mail server for Kartik's messages. The separation of functionality between the user agent and message transfer agent is valuable - the user agent provides rich features for a single user, and the message transfer agent provides reliable service for multiple users.

An important point regarding email is that it is not interactive unlike FTP or Telnet. In sending an email, the *user agent* does not know if and when the message reaches the recipient's mail server, or if this recipient is online (connected to his mail server) at this time. Thus, the messages composed by the

user via his user agent are kept in a *spool* area, and the local mail server (mail transfer agent) inspects this spool area every 30 minutes collecting email messages deposited here. Similarly, at the receiving end, the mail for the recipient is kept in the mailbox on the receiving mail server, if he/she is currently off-line.

The typical email set up is summarized in Figure 1. An email message

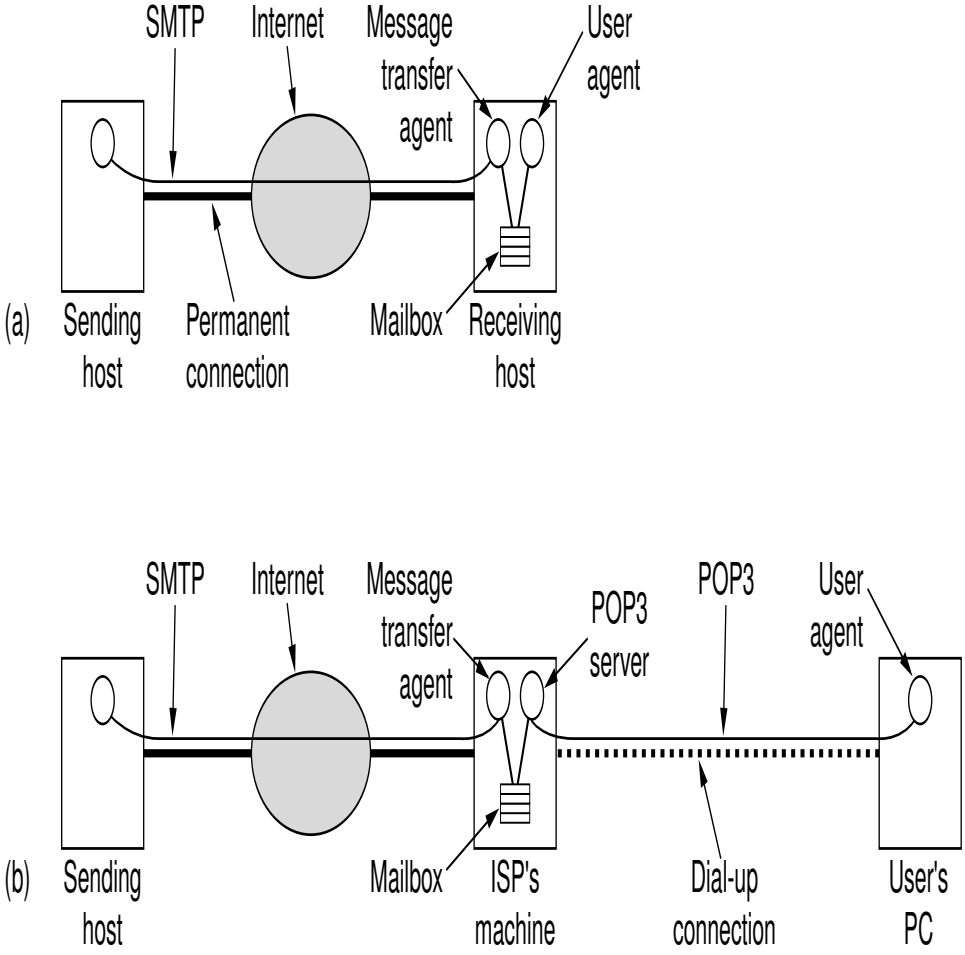

Figure 1: (a) Email when the receiver has a permanent Internet connection. (b) Reading email when the user has a dial-up connection to an ISP.

consists of an envelope, header and a body. A typical email message is shown in Figure 2. The body represents the text sent by the user. Each header field starts on a separate line, and consists of a single string, followed by a colon and another string (e.g., Date: Sun 29th Feb 2004 11.29:32 GMT). The envelope includes the email addresses of the intended recipients. A listing of various fields in the envelope and header appear in Figures 3 and 4 respectively.

Some of the fields in the header and envelope such as *To* and *Subject* depend on the user input. Others, such as *Date* and *Message-Id* are set by the user agent or the local mail server. The header, body and envelope information are sent across the TCP connection between the two mail servers in the U.S. 7 bit ASCII format. Each line ends with a two character sequence - a carriage return

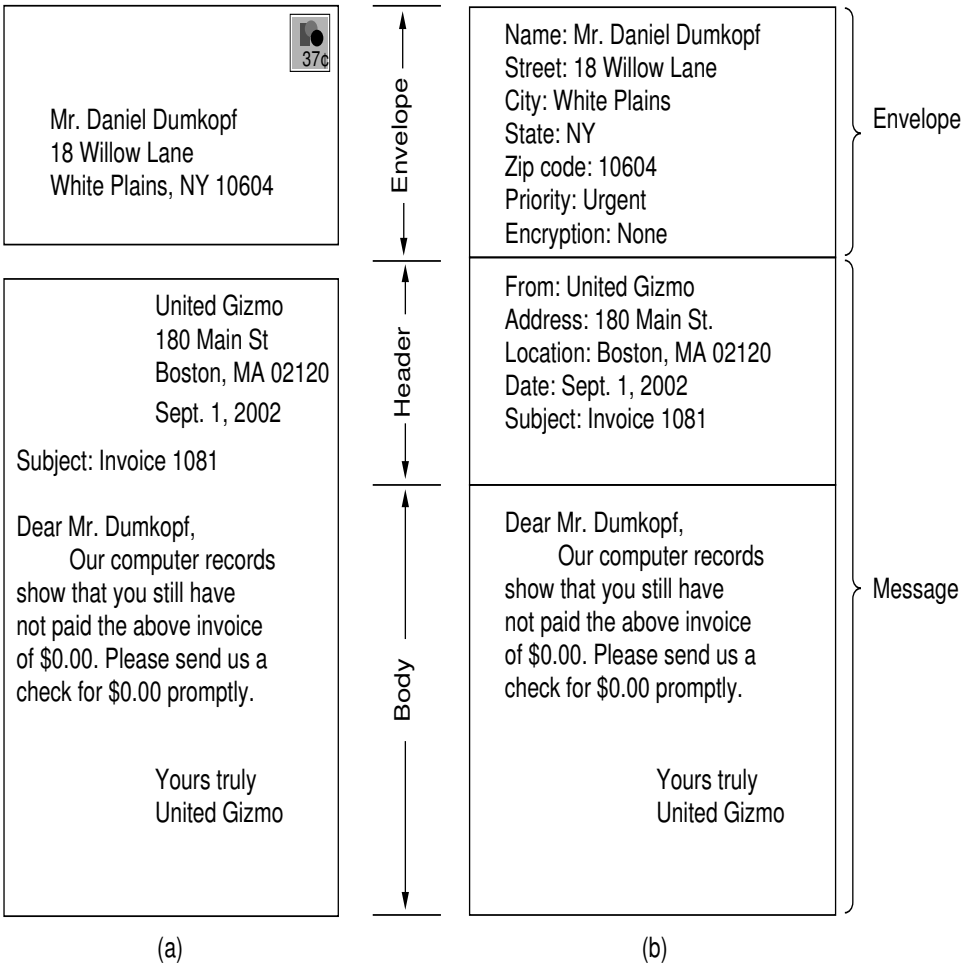

Figure 2: (a) Paper mail. (b) Electronic mail.

Header	Meaning
To:	E-mail address(es) of primary recipient(s)
Cc:	E-mail address(es) of secondary recipient(s)
Bcc:	E-mail address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	E-mail address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Figure 3: Fields in the envelope.

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	E-mail address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

Figure 4: Fields in the header.

(CR) followed by a line feed (LF). Initially, email messages contained only text data. Later, the Multipurpose Internet Mail Extensions (MIME) provided a standard way to convert other types of data into text format for inclusion in email messages. MIME includes additional headers that indicate the size and the encoding of the contents of the message. Some of MIME headers are mentioned in Figure 5. We also discuss some of the MIME types in Figure 6.

Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

Figure 5: Headers added by MIME.

The SMTP protocol:

The sending mail server establishes a TCP connection on port 25 of the receiving mail server. The combined header and body are transmitted from one mail server to another using sequence of commands. The mail servers do not distinguish between the header and the body of the email messages although they may prepend additional header fields to the message, such as *Received*. This allows the recipient to trace the sequence of mail servers involved in transferring the message. As in FTP, the sender issues a sequence of commands, one at a time, and receives replies consisting of a three digit reply code and a textual

Type	Subtype	Description
Text	Plain	Unformatted text
	Enriched	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Application	Octet-stream	An uninterpreted byte sequence
	Postscript	A printable document in PostScript
Message	Rfc822	A MIME RFC 822 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 822 message

Figure 6: Various MIME types.

string. A typical exchange involves separate commands to

1. Identify the local mail server.
2. Identify the sender of the email message.
3. Identify each recipient of the email message.
4. Send the actual email message.

However, unlike FTP, SMTP uses a single TCP connection for both the command reply exchanges and the transfer of the email message.

In addition to transferring the message between the mail servers, delivering an email message requires two additional steps involving the user agent - transferring the message to the local mail server, and the reception of the message from the remote mail server. These transfers may also involve SMTP, although other protocols can also be used. The reception of email messages from the mail server typically uses other protocols such as Post Office Protocol (POP3) (especially when the receiver has a dial-up connection to an Internet Service Provider (ISP)), and the Internet Access Protocol (IMAP). The POP3 protocol downloads all the messages in the user's mailbox on the mail server to his machine, so that he can read his messages offline once the transfer is complete. IMAP on the other allows the user to read messages directly (online) from his mailbox on the mail server. IMAP is used for instance in *Hotmail* and *Webmail*.

The World Wide Web (HTTP):

I will return to a discussion of the world wide web (HTTP), which is perhaps the most important application and amounts for more than 95% of the web traffic, after I have discussed Computer Security. This will be covered in detail in a subsequent lecture.

Recommended Reading

1. Chapter 25 of Comer [1], Chapter 26 of Stevens [2] for a discussion on Telnet.
2. Chapter 26 of Comer, and Chapter 27 of Stevens [2] for a discussion on FTP.
3. Chapter 27 of Comer, Chapter 28 of Stevens, and Section 7.2 of Tanenbaum [3] for a discussion on SMTP and electronic mail. The various figures in this lecture are taken from this chapter of Tanenbaum.

References

[1] D.E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architectures*, 4th edition, Prentice Hall, NJ, 2000.

[2] W. Richard Stevens, *TCP/IP Illustrated, Volume I: The Protocols*, Addison Wesley Professional Computing Series, 1994.

[3] A.S. Tanenbaum, *Computer Networks*, 4th edition, Prentice Hall, 2003.