Module
4
Standards

Lesson 14 Application & Presentation Layers

LESSON OBJECTIVE

General

This lesson is designed to give the reader an overview of the standards followed in the topmost layers of the OSI reference model viz. Application layer and Presentation layer.

Specific

On completion of this lesson, the learner shall be able to

1. Understand the working of Telnet protocol.
2. Identify the applications of FTP.
3. Outline the procedure of HTTP.

4.7.1 INTRODUCTION

This lesson continues our exploration of networking standards by examining high level internet services and the application and presentation layer standards that support them. These services determine how users pursue the internet. We will see that these high level services provide increased communication functionality and allow users and programs to interact with automated services on remote machines and with remote users. We shall now look at TELNET, FTP and HTTP in detail.

4.7.2 TELNET PROTOCOL

The TCP/IP protocol suite includes a simple remote terminal protocol called TELNET that allows users to log in to a computer across the internet. TELNET establishes a TCP connection and then passes keystrokes from the user’s keyboard directly to the remote computer as if they had been typed on a keyboard attached to the machine. TELNET also carries output from the remote machine back to the user’s screen. The service is called transparent because it gives the appearance that the user’s keyboard and display attach directly to the remote machine. TELNET is however not as sophisticated as some remote terminal protocols.

TELNET client software allows the user to specify the remote machine either by specifying its domain name or IP address. TELNET offers three basic services. First it defines a *network virtual terminal* that provides a standard interface to remote systems. Client programs do not have to understand the details of all possible remote systems; they are built to use the standard interface. Second TELNET includes a mechanism that allows the client and server to negotiate options, and it provides a set of standard options. Finally TELNET treats both ends of the connection symmetrically. In particular, TELNET does not force client input to come from a keyboard, nor does it force the client to display output on a screen. Thus TELNET allows an arbitrary program to become client.

As shown in the figure above, when a user invokes TELNET, an application program on the user's machine becomes the client. The client establishes a TCP connection over which they will communicate. Once this is done, the client accepts keystrokes from the user's keyboard and sends them to the server, while it concurrently accepts characters that the server sends back and displays them on the user's screen. The server must accept a TCP connection from the client, and then relay data between the TCP connection and the local operating system. In practice the server is more complex and it handles multiple client connections.

To make TELNET interoperate between as many systems as possible, it must accommodate the details of heterogeneous computers and operating systems. To accommodate heterogeneity, TELNET defines how data and command sequences are sent across the internet. The definition is known as the *network virtual terminal (NVT)*. The client software translates keystrokes and command sequences from the user's terminal into NVT format and sends them to the server. Server software translates incoming data and commands from NVT format into the format the remote system requires.

4.7.3 FILE TRANSFER PROTOCOL

Many network systems provide computers with the ability to access files on remote machines. Standard file transfer protocols existed for the ARPANET before TCP/IP became operational. These early versions of file transfer software evolved into a current standard named File Transfer Protocol. FTP server implementations allow concurrent access by multiple clients. Clients use TCP to connect to a server. A single master server process awaits connections and creates a slave process to handle each connection. Unlike most servers, however, the slave process does not perform all the necessary computations. Instead, the slave accepts and handles the control connection from the clients but uses an additional process to handle a separate data transfer connection. The control

Connection carries commands that tell the server which file to transfer. The data transfer connection, which also uses TCP, carries all data transfers. Usually client and server create a separate process to handle the data transfer.

As the figure shows, the client control process connects to the server control process using one TCP connection while the associated data transfer processes use their own TCP connection. In general, the control processes and the control connection remain alive as long as the FTP session is active. However, FTP establishes a new data transfer connection for each file transfer.

4.7.4 HTTP

During the early history of the internet FP data transfers accounted for approximately one-third of internet traffic. However, by 1995, web traffic overtook FTP to become the largest consumer of internet backbone bandwidth. By 2000, web traffic had completely overshadowed other applications. In fact, for many users, the internet and the web are indistinguishable.

The protocol used for communication between a browser and a web server or between intermediate machines and web servers is known as hypertext transfer protocol (HTTP). It has the following set of characteristics:

Application level: HTTP operates at the application level. It assumes a reliable connection oriented transfer protocol such as TCP but does not provide reliability or retransmission itself.

Request/response: once a transport session has been established, the browser must send an HTTP request to which the other side responds.

Stateless: each HTTP request is self-content. The server does not keep a history of pervious request or sessions.

Bidirectional transfer: HTTP allows transfer from a server to a browser and also vice-versa.

Support for caching: to improve response time, a browser caches a copy of each web page it retrieves. If user requests a page again, HTTP allows the browser to interrogate the server to determine whether the contents of the page have changed since the copy was cached.

Support for intermediaries: HTTP allows a machine along the path between a browser and a server to act as a proxy server that caches web pages and answers a browser’s request from its cache.

HTTP GET REQUEST

Generally, the browser contacts a web server directly to obtain a page.

The browser begins with a URL, extracts a host name section, uses DNS to map the name into an equivalent IP address and uses the IP address to form a TCP connection to the server. Once the TCP connection is in place, the browser and web server use HTTP to communicate; the browser sends a request to retrieve a specific web page and the server responds by sending a copy of the page.

Objective Questions

14.01

Subjective Questions

14.11

Level 2 Questions

14.21