Chapter 2. Application Layer

Table of Contents

1. Context

In the previous chapter, we gave an introduction to the Internet protocols. We discussed two popular protocol suites, namely the ISO OSI reference model and the TCP/IP protocol suite. In this chapter, we will discuss the conceptual and implementation facets of software that operate at the application layer of the TCP/IP protocol suite. The Internet is implemented by the TCP/IP protocol stack. Thus, sometimes the Internet is referred to as the TCP/IP network. In this chapter and chapters that follow, we will use Internet and TCP/IP network interchangeably.

2. Introduction

After this chapter, if you want to know more, read Chapter 2 of James F. Kurose and Keith W. Ross, "Networking: A top-down approach", (6th edn.).

This chapter gives a broad overview of the Internet's application software. It looks at electronic mail, file transfer, peer-to-peer (P2P), domain name system (DNS) and the World Wide Web in terms of their operation and protocols used for communication.

Electronic mail (email) is the electronic version of ordinary mail. Email can be used for one-to-one communication, in the same way as conventional mail is, but its electronic nature makes other patterns of communication, including one-to-many and many-to-one, equally feasible.

File transfer applications allow users to share anything on the network that can be stored in a file, including information, images and software.

Peer-to-peer applications make intermittently connected hosts, called peers, communicate directly with each other without passing through the intermediary computer. A popular peer-to-peer application is BitTorrent.

The World Wide Web (WWW) holds a distributed collection of information in interlinked files held by particular computers known as Web servers. This information can be accessed by any device running a Web browser like Google Chrome.

3. Objectives

At the end of this module, you should be able to:

At the end of this module, you should be able to:

- understand in some detail the technical operation of the Internet's major applications; and
- explain the operations of the following protocols:
    1. HyperText Transfer Protocol (HTTP);
    2. Simple Mail Transfer Protocol (SMTP);
    3. File Transfer Protocol (FTP);
    4. Domain Name System (DNS);
    5. Peer-to-Peer (P2P).

4. Network application software

Network applications are **distributed** in the sense that their complete functionality depends on the co-operation of many separate devices that can be in different locations. With email, for example, there is a sending device at which the contents of the email are composed and sent, and a receiving computer at which the email is received and its contents can be read. On the World Wide Web, there is a client computer that requests a Web page and a server from which it is to be obtained. In this way, networked applications are quite different from stand-alone applications, in that the computations needed to be carried out are distributed between different end systems.

Therefore, developing network applications involves writing software that runs on multiple end systems. These applications can be written in any programming language, such as C++, C or Java.

Essentially, you do not need to write software that runs on network-core devices, such as routers or switches. Network-core devices do not operate at the application layer but at lower layers of the TCP/IP protocol stack.

Network applications communicate by exchanging **messages** across the network. One application initiates the communication by sending a message to another application on another device. In this chapter, an application that initiates the communication will be referred to as a **client** application. The application that waits to be contacted to begin a communication will be referred to as a **server** application.

5. Process communication

An application is also called a **process**. A process is basically a running application on one end system. In this chapter, we will use application and process interchangeably.

When a process wants to send a message, it does so by using the services of the transport layer. The software interface between the transport layer and the application layer is called a **socket**. From the programmer standpoint, a socket is a software construct which enables applications to view networking as if it were file I/O (input/output) — a program can read from a socket or write to a socket as simply as reading from a file or writing to a file.

For processes running on different computers to communicate, an addressing mechanism must be used. The process on a host is identified by a **port number**. A port number is a number between 0 and 65536 assigned to the running process by the operating system. A host on the Internet can run more than one network application. Each application on one host is assigned a unique port number for identification. Popular Internet applications have been assigned standard port numbers. For example, HTTP uses port 80 and FTP uses port 21. Figure 2.1 illustrates how two Internet applications identify each other for communication.

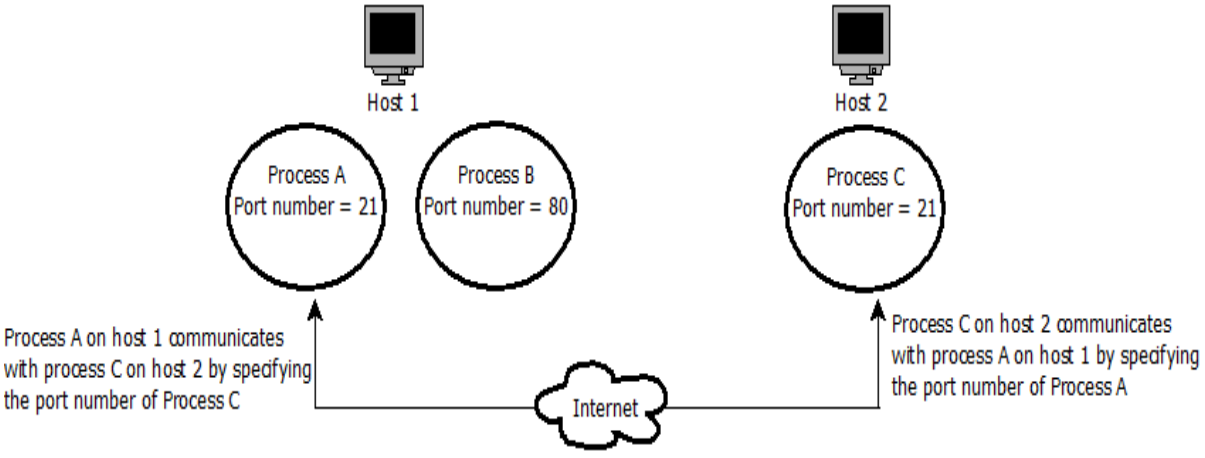

Figure 2.1: PROCESS IDENTIFICATION ON THE INTERNET

6. Transport Layer services provided by the Internet

The Internet or TCP/IP network provides two transport-layer protocols that can be used by the application layer, namely **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)**. Each protocol provides services to applications in the application layer. The table below lists some of the services offered by both protocols. More details of these protocols will be discussed in the next chapter.

TCP Services	UDP Services
Connection-oriented services: the client and server processes make a connection before application messages can be exchanged.	**Connectionless services:** the client and server processes exchange messages without creating a connection.
Reliable data transfer service: guarantees delivery of application messages without error and in proper order.	**Unreliable data transfer service:** no guarantee that messages will reach the receiving application. Furthermore, messages may arrive out of order.

When you want to write an Internet application, you must explicitly choose whether to use TCP or UDP. How would you make a choice? Most likely, you would study the services provided by both protocols, and then pick the protocol with the services that best match your application's needs. For example, the Web requires no data loss when transferring Web pages between the server and client application, hence it uses the TCP protocol. Conversely, Internet games can tolerate data loss. Therefore, Internet games can be designed to use UDP.

7. Application Layer Protocols

Application-layer protocols define how applications running on different computing devices exchange messages. Application-layer protocols define the following:

- Types of messages exchanged between applications
- The syntax and semantics of fields in the messages
- Rules for governing how messages are exchanged between applications running on different devices

In this chapter, we will discuss the following application-layer protocols:

1. HyperText Transfer Protocol (HTTP), used on the Web
2. Simple Mail Transfer Protocol (SMTP), used by electronic mail
3. File Transfer Protocol (FTP), for transfer of files between applications
4. Domain Name System (DNS), which provides a directory service for the Internet
5. Peer-to-Peer (P2P), used when peer processes want to communicate without passing through a dedicated server.

8. The web and HTTP

The predecessor of the Internet was the ARPANET. When it was established, it was expected that its main purpose would be to allow the sharing of the large and expensive computers that were attached to it. Although it was used in this way, its users, essentially the research community in the USA, soon determined that it was of primary use as an email network. Subsequently, with the widening of the user community, the most popular usage of the Internet has become the World Wide Web. The major application of the Internet has changed from Telnet to email to the World Wide Web.

In this section, we will examine the operation of the World Wide Web protocol called **HyperText Transfer Protocol (HTTP)** in some detail.

8.1. Web Terminology

Before we delve in the operations of the HTTP protocol, we should introduce some Web terminology.

Web server and browser

In the Web, there are two distinct applications that communicate with each other: the browser program running in the user's host (desktop, laptop, tablet, smartphone, and so on) and the Web server program running in the Web server host. A Web browser is the client program that requests services from the Web server. Popular Web browsers include Mozilla Firefox, Google Chrome and Internet Explorer. Popular Web servers include Apache and Microsoft Internet Information Server. The Web server hosts resources in the form of Web pages.

Web page

A Web page is a collection of one or more files in a particular format. A file format can be text, JPEG image or video clip. The most common file format is HTML.

HTML

HTML files are text files created using the Hypertext Mark-up Language (HTML), which marks up the text and other content of a document in such a way as to describe the appearance when displayed in a Web browser. HTML marks have opening and closing tags. The following simple example illustrates the form of an HTML document:

<html>

<head>

<TITLE> Simple Example</TITLE>

</head>

<body>

<H1>Heading</H1>

<P>A short illustrative paragraph that includes a single link to </p>

</body>

</html>

Most Web pages consist of a base HTML file linked to other HTML files or other files such as image files, which can be on the same server or on a different one. In this way, the contents of the World Wide Web are stored on the numerous Web servers scattered across the world, where those contents are linked together to form a single 'web'.

URL

Web pages on Web servers are addressed by Uniform Resource Locators (URLs). A URL has the form:

http://<host name>/<file path name> or https://<host name>/<file path name>

- http or https: http denotes the HTTP protocol. https denotes a secure http protocol
- <host name>: a human-friendly rendering of the Internet address of the host holding the required Web page

- <file path name>: denotes the path name for the file on that host

A typical request could be:

https://www.uct.ac.za/index.html

8.2. Overview of HTTP protocol

HTTP defines how the Web clients request Web pages from Web servers, and how the Web servers send the Web pages to the Web clients. The HTTP specification is defined in [RFC 1945] and [RFC 2616].

The generic client-server interaction proceeds through four stages. They are listed below, and illustrated in Figure 2.2.

1. Send request: The request, as we have seen, takes a form such as: https://www.uct.ac.za/courses/index.html

   which is taken as a request to the computer named www.uct.ac.za for the file /courses/index.html'.

2. Receive request: The named computer (in this case, a Web server) receives the request and locates the file in its file store.

3. Send response: The Web server sends the requested file to the requesting computer. The file is of a particular type and assumes a well-known form.

4. Receive response: The client computer receives the file. In this case, having received it, one of the functions of the browser running on the client computer is to cause it to be displayed.

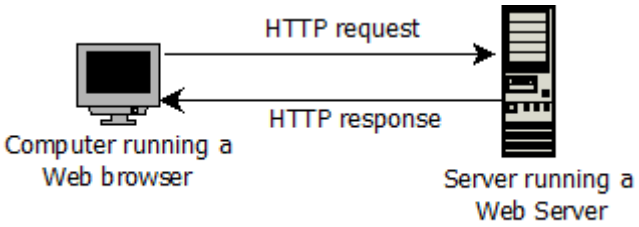

FIGURE 2.2: HTTP CLIENT-SERVER INTERACTION

HTTP uses the TCP protocol from the transport layer for client-server interaction because reliability is of greater concern in Web applications. TCP guarantees delivery of application messages without error and in proper order. The Web client initiates the TCP connection with the server. Once the interaction is complete, the TCP connection is closed. The HTTP server does not keep any information about the client. Once the TCP connection closes, the server forgets about the client. Because a server does not maintain the state of a client, HTTP is said to be a **stateless protocol**.

8.3. HTTP message format

There are two types of HTTP messages: request messages and response messages.

8.3.1. Request message

The request message is sent by the Web client to the Web server to request for a Web page. The request message is made up of three parts, namely the request line, header lines and the message body.

1. **Request line**: This is the start line that begins all HTTP request messages. It has three fields:

    a) Method: The method can take values GET, POST, HEAD, PUT and DELETE. The Web client uses the GET method when it wants to request for an object.
    b) URL: The URL field holds the name of the object requested by the Web client.
    c) Version: Specifies the version of HTTP used.

2. **Header lines**: Header lines allow the Web client to pass additional information about the request to the server. Examples of header lines include User-Agent and Accept.

3. **Message body**: When the Web client uses the POST method to send a request to the server, the entity body holds the entered values in the form of the HTML document.

8.3.2. Response message

The response message is sent by the Web server in response to the request from the Web client. Like requests, responses are made up of three parts, namely the status line, header lines and the message body.

1. **Status line**: This is the start line of all HTTP response messages. It has three fields:
    - Version field**:** Like in the request message, the version field holds information about the version number of the HTTP used.
    - Status code and phrase fields: The status code and associated phrase indicates the result of the request. Common status codes and phrases include:
        - 200 OK: The request is successful.
        - 301 Moved Permanently: The requested resource has been moved permanently.
        - 400 Bad Request: The request could not be understood.
        - 404 Not Found: Requested resource not found.
2. **Header lines**: Like in the request message, header lines allow the Web server to pass additional information to the Web client about the response.
3. **Message body**: The entity body contains the requested object.

8.4. Cookies

As mentioned earlier, the HTTP server does not retain any information about the client once the TCP connection is terminated. However, some Web applications need to keep the state of a client. For example, if you are doing online shopping, the server must keep track of all the products you have put in your online basket. HTTP servers use cookies to keep the state of the client.

Cookies have four components:

1. **Website database**: If the Web server wants to use cookies after receiving a request from the Web client, it generates a unique identification number and stores it in its database.
2. **HTTP response-message cookie header line**: The cookie header line in the response message holds the unique identification number generated by the server.

3. **Cookie file**: A cookie file is kept on the user's end system and is managed by the user's browser. When the browser receives an HTTP response message with a cookie header line set, the browser extracts the unique identification number and appends a line in the cookie file. An entry in the cookie file includes the hostname and the unique identification number.
4. **HTTP request-message cookie header line**: Before the Web client sends a request to the Web server, it checks the cookie file for any entry that matches the Web server's name. If the match is found, the associated unique identification number is set in the HTTP request-message cookie header line.

Most popular Web applications like Facebook, Twitter, Amazon and Google use cookies for Web client identification.

8.5. Web caching

A Web cache is a server that satisfies HTTP requests on behalf of a Web server. A Web cache stores copies of Web pages from the Web server. A Web client can be configured to send HTTP requests to the Web cache. If the requested Web page is found in the cache, it is returned to the requesting client. If it is not found, the Web cache requests for the Web page from the origin Web server. In this sense, a Web cache acts as both a client and server. This is illustrated in Figure 2.3.

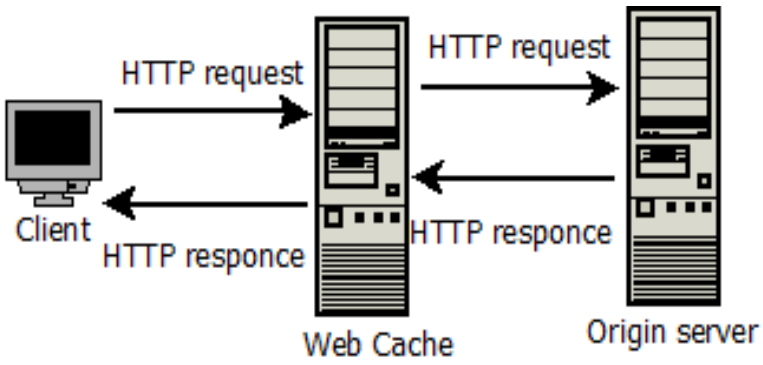

FIGURE 2.3: WEB CACHE IN HTTP

One of the benefits of deploying a Web cache in a network is that it reduces response time for a client request. For example, University of Cape Town can deploy a Web cache in its network and point all devices to it. In this way, the university can substantially improve client response time, especially if the link between the university and the Internet is slow.

9. Electronic Mail on the Internet

Electronic mail (email) is the electronic version of ordinary mail. Email can be used for one-to-one communication, in the same way as conventional mail usually is, but its electronic nature makes other patterns of communication, including one-to-many and many-to-one, equally feasible. These other patterns of communication lead to uses for email that are not practicable or achievable with conventional mail.

Figure 2.4 illustrates the components and protocols that make up the Internet email.

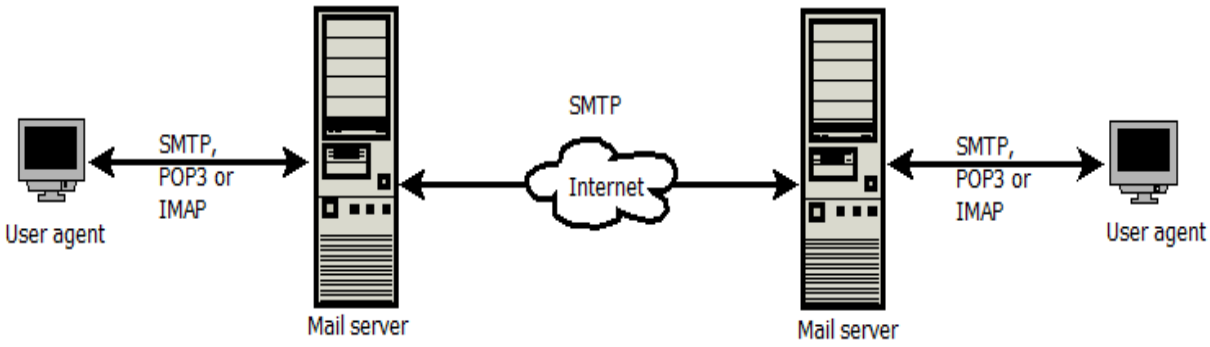

FIGURE 2.4: INTERNET E-MAIL SYSTEM

- **Mail server**: Mail servers hold incoming emails. Each recipient has a mailbox in the mail server. The mailbox manages and maintains emails that have been sent to a recipient.
- **User agent**: A user agent is software like Microsoft Outlook, that allows users to compose and send emails.
- **SMTP**: Simple Mail Transfer Protocol (SMTP) is the principal protocol for Internet email. SMTP defines how mail servers exchange emails. It also defines how user agents send emails to the mail server. SMTP uses TCP protocol for communication.
- **POP3**: Post Office Protocol – Version 3 (POP3) is a protocol used by the user agent to pull an email from the mail server. The protocol uses TCP to authenticate a user with the mail server, retrieve the email and store it locally.
- **IMAP**: Internet Mail Access Protocol (IMAP) is also used by the user agent to pull an email from the mail server. Unlike POP3, IMAP allows users to keep emails in the mail server. This allows users to access the same email from different devices.

We now demonstrate how a sender sends an email to the recipient using the components mentioned above.

1. The message sender uses the user agent to compose the message. The agent uses SMTP for transmission to the local mail server. A typical message has the following:
    - From: Specifies the sender's address. Addresses take the form 'mailbox@domain_name' - for example, mit@cs.uct.ac.za.
    - To: Specifies the receiver's email address.
    - Subject: is optional. If present, it specifies the subject of the email.
    - Message body: is also optional. If present, it holds the message.
2. The mail server converts the domain name to the address of the destination email server and puts the message on the network.
3. The network routes the messages to the destination mail server.
4. The destination mail server assembles the message and places it in the recipient's mailbox.
5. The recipient's user agent pulls the email using POP3 or IMAP.

10. File Transfer Protocol

The file transfer protocol (FTP) defines how applications on two end systems share files. Typically, the exchange of files involves a server and a client application. The server application, also called the FTP server, stores files. The client application, also called the FTP client, connects to the server using a TCP connection and uploads and downloads possible files.

FTP and HTTP are both file transfer protocols and share many characteristics. One notable difference is the number of TCP connections used to transfer files. HTTP uses one TCP connection between the server and client to transfer a Web page. FTP, on the other hand, uses two TCP connections between the client and the server, a **TCP control connection** and a **TCP data connection**, as illustrated in Figure 2.5. The control connection is used to transfer FTP commands, and the data connection is used to transfer files. The FTP client initiates a TCP control connection to the server and uses the connection to send the username and password for authentication. Once authenticated, the client can now send commands to change the remote directory and transfer one or more files stored in the local file system into the remote file system, or vice versa. Once the server receives the file transfer command over the TCP control connection, it initiates a TCP data connection to the client side. The data connection is only used to transfer one file and then closes. If the user wants to send or receive another file during the same session, another TCP data connection must be initiated. Thus, during the FTP session between the client and the server, one TCP control connection is maintained but a new data connection is created for each file transferred.

As discussed earlier, HTTP is called a stateless protocol because the Web server does not retain any information about a client after the TCP connection is closed. On the other hand, the FTP server maintains the state about the client. The FTP server associates a TCP control connection to a user account and keeps track of the client's current directory.

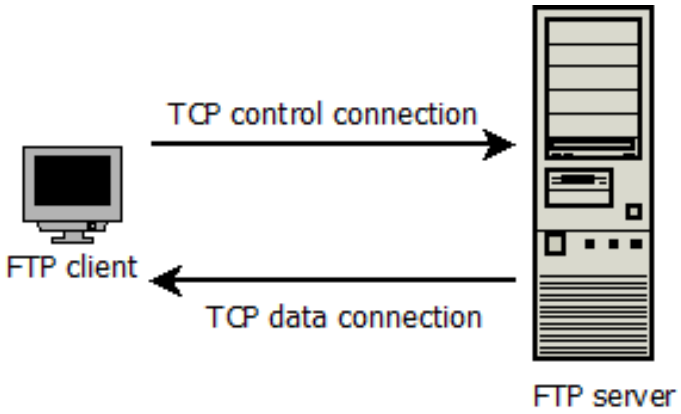

FIGURE 2.5: TCP CONNECTION BETWEEN THE FTP CLIENT AND SERVER

The client communicates with the server by sending commands using the TCP control connection. Some common client commands include:

- **USER username**: Used to send the username
- **PASS password**: Used to send the password
- **RETR filename**: Used to request for a file from the server
- **STOR filename**: Used to send a file to the FTP server

Each command sent by the client is followed by a reply from the server. Some typical replies, along with their possible messages, are as follows:

- 331 Username OK, password required
- 125 Data connection already open; transfer starting

- 425 Can’t open data connection
- 452 Error writing file

11. Domain Name System (DNS)

As discussed earlier, hosts in a network have unique addresses for identification. On the Internet, hosts are identified by **hostnames** and **IP addresses**. A hostname is a mnemonic name like www.uct.ac.za. The IP address is usually written as four decimal values (each representing a byte or octet) separated by a period, such as 192.168.3.1. The domain name system (DNS) translates hostnames to IP addresses, fulfilling a critical role in the Internet infrastructure. DNS is made up of two components: DNS servers and DNS clients.

DNS servers implement a distributed database that contains the mapping of the hostnames to IP addresses. There are millions of IP addresses on the Internet. To deal with the issue of scale, the domain name system uses many servers organised in a hierarchy and distributed in different parts of the world. Each server in the hierarchy holds a portion of the database. Figure 2.6 below shows a portion of a DNS hierarchy. As can be seen, the top of the hierarchy contains root DNS servers followed by top-level domain (TLD) DNS servers. The branches of the hierarchy contain authoritative DNS servers.

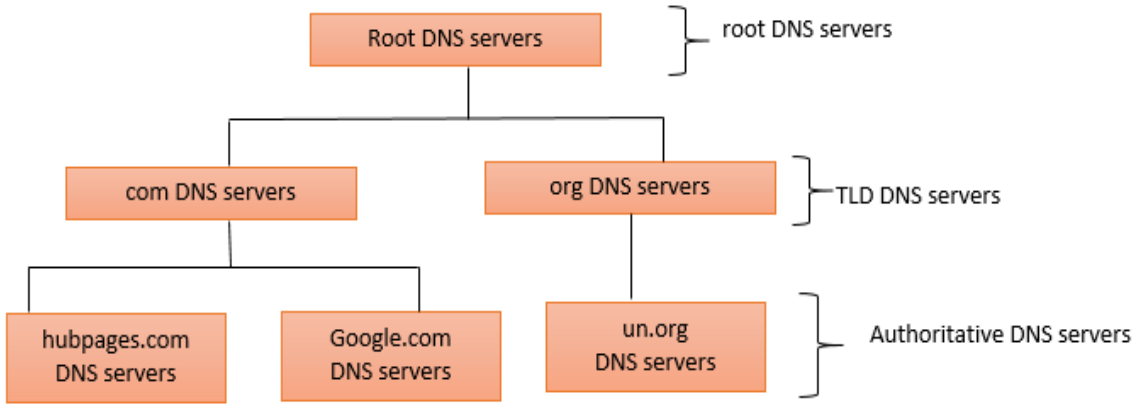

FIGURE 2.6: PORTION OF THE DNS HIERARCHY

DNS clients are application-layer programs that allow hosts to query DNS servers. They are installed on host devices and use UDP protocol from the transport layer to send requests to the DNS servers. To illustrate the interaction between the DNS client and the DNS server hierarchy, let’s look at an example below.

Let’s assume the client wants to determine the IP address for the hostname www.hubpages.com. The following events will take place.

1. The client first will contact the local name server.
2. The local name server contacts one of the root servers, which returns IP addresses for TLD servers for the top-level domain com.
3. The local name server then contacts one of these TLD servers, which returns the IP address of an authoritative server for hubpages.com.

4. The local name server contacts one of the authoritative servers for hubpages.com, which returns the IP address.

5. Finally, the local name server sends the IP address of hubpages.com back to the client.

12. Peer-to-Peer (P2P)

Most of the Internet applications we have covered so far like HTTP, email and FTP involve a client and the server application. Clients send requests to the server application over the network. The server application processes the request and sends the response to the client. Peer-to-peer (P2P) applications works differently. In a P2P application, intermittently connected hosts called 'peers' communicate with each other with minimal or no reliance on a dedicated server. In this section, we will discuss one type of application that is suited to use P2P design, called **file distribution** application.

File distribution applications are designed to distribute large files to different hosts on a network by the server. A file can be an operating system, movie file or a new software. In traditional client-server applications, all the hosts connect to one server and download a file. This places a huge burden on the server in terms of computation and bandwidth. In P2P file distribution, hosts connect both to the server and to each other. When a host downloads a file from the server, it can redistribute any portion of the file it has received to any other peers, thereby assisting the server in the distribution process. An example of a file distribution application on the Internet is the BitTorrent system. BitTorrent was originally developed by Bram Cohen [Chao 2011]. In the remaining part of this section, we will give a brief description of how the BitTorrent system works.

12.1. BitTorrent protocol

BitTorrent protocol is one of the popular P2P protocols for file distribution. The protocol allows a number of peers to share a single file. A file is distributed among peers as chunks, with a typical chunk size of 256 Kbytes. The collection of all peers participating in the distribution of a particular file is called a **torrent**. A peer in a torrent can either upload or download chunks of a file. Once a peer has acquired the entire file, it may leave the torrent or remain in the torrent and continue to upload chunks to other peers. Also, any peer may leave the torrent at any time with only a subset of chunks, and later rejoin the torrent. A torrent is managed by an infrastructure node called a **tracker**. Whenever a new peer joins a torrent, it registers itself with the tracker. This way, a tracker has knowledge of all the peers in a torrent.

13. Activities

Activity 1

Develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will:

1. create a connection socket when contacted by a client (browser);
2. receive the HTTP request from this connection;
3. parse the request to determine the specific file being requested;
4. get the requested file from the server's file system;
5. create an HTTP response message consisting of the requested file preceded by header lines;

6. send the response over the TCP connection to the requesting browser.

If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.

Activity 2

Create a simple mail client in Python that sends email to any recipient. Your client will need to establish a TCP connection with a mail server (e.g. a Google mail server), dialogue with the mail server using SMTP, send an email message to a recipient (e.g. your friend) via the mail server, and finally close the TCP connection with the mail server.

Activity 3

In this activity, you will develop a Web proxy. When your proxy receives an HTTP request for an object from a browser, it generates a new HTTP request for the same object and sends it to the origin server. When the proxy receives the corresponding HTTP response with the object from the origin server, it creates a new HTTP response, including the object, and sends it to the client. This proxy will be multi-threaded, so that it will be able to handle multiple requests at the same time.

14. Review Questions

You can find answers for these review questions at the end of the chapter.

Review question 1

What is a protocol? Match these protocols and applications:

Protocols: SMTP, FTP and HTTP.

Applications: file transfer, World Wide Web and email.

Review question 2

What is the difference between IP address and port number?

Review question 3

For a communication session between a pair of processes, which process is the client and which is the server?

Review question 4

For a P2P file-sharing application, do you agree with the statement, "There is no notion of client and server sides of a communication session"? Why or why not?

Review question 5

What information is used by a process running on one host to identify a process running on another host?

Review question 6

Suppose you wanted to do a transaction from a remote host as fast as possible. Would you use UDP or TCP? Why?

Review question 7

Why do HTTP, FTP, SMTP and POP3 run on top of TCP rather than on UDP?

Review question 8

Consider an e-commerce site that wants to keep a purchase record for each of its customers. Describe how this can be done with cookies.

Review question 9

Describe how Web caching can reduce the delay in receiving a requested object. Will Web caching reduce the delay for all objects requested by a user or for only some of the objects? Why?

Review question 10

Suppose Alice, with a Web-based email account (such as Hotmail or Gmail), sends a message to Bob, who accesses his mail from his mail server using POP3. Discuss how the message gets from Alice's host to Bob's host. Be sure to list the series of application-layer protocols that are used to move the message between the two hosts.

15. Answers to review questions

Review question 1

A protocol is a set of rules governing the communications that can take place in a given situation. (It can also be defined as the set of messages that needs to be exchanged to achieve some goal.)

SMTP is the Internet's basic email protocol, FTP is its file transfer protocol, and HTTP the protocol used by the World Wide Web.

Review question 2

IP addresses uniquely identify hosts on the Internet. Port numbers identify network applications on a host.

Review question 3

The process which initiates the communication is the client; the process that waits to be contacted is the server.

Review question 4

No. In a P2P file-sharing application, the peer that is receiving a file is typically the client and the peer that is sending the file is typically the server.

Review question 5

The IP address of the destination host and the port number of the socket in the destination process.

Review question 6

You would use UDP. With UDP, the transaction can be completed in one roundtrip time (RTT) - the client sends the transaction request into a UDP socket, and the server sends the reply back to the client's UDP socket. With TCP, a minimum of two RTTs are needed - one to set up the TCP connection, and another for the client to send the request, and for the server to send back the reply.

Review question 7

The applications associated with those protocols require that all application data be received in the correct order and without gaps. TCP provides this service whereas UDP does not.

Review question 8

When the user first visits the site, the server creates a unique identification number, creates an entry in its back-end database, and returns this identification number as a cookie number. This cookie number is stored on the user's host and is managed by the browser. During each subsequent visit (and purchase), the browser sends the cookie number back to the site. Thus the site knows when this user (more precisely, this browser) is visiting the site.

Review question 9

Web caching can bring the desired content 'closer' to the user, possibly to the same LAN to which the user's host is connected. Web caching can reduce the delay for all objects, even objects that are not cached, since caching reduces the traffic on links.

Review question 10

The message is first sent from Alice's host to her mail server over HTTP. Alice's mail server then sends the message to Bob's mail server over SMTP. Bob then transfers the message from his mail server to his host over POP3.