BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 4

DCA2201

COMPUTER NETWORKING

Unit 11

Application Layer Protocols

Table of Contents

1. INTRODUCTION

In the previous unit, we discussed transport layer resource allocation and quality of service. If the network takes an active role in allocating resources properly, then congestion may be avoided. In this unit, we will discuss various application layer protocols. All user applications are found in the application layer. The layers below the application layer are there to provide transport services. In this unit, we will discuss one real application, electronic mail and one application layer protocol, HTTP.

We will start this unit with a discussion on electronic mail. Then, we will discuss different e-mail protocols such as SMTP, MIME and POP3. We will also have a discussion on HTTP.

1.1 Objectives:

After studying this unit, you should be able to:

- *Describe SMTP*
- *Explain MIME*
- *Describe Post Office Protocol*
- *Explain HTTP*

2. ELECTRONIC MAIL

Electronic mail, or more commonly email, has been used all over the world for the last three decades. Even if other forms of network communication, such as instant messaging and voice-over-IP calls have expanded greatly in use over the past decade, email remains the workhorse of Internet communication.

The first email systems simply consisted of file transfer protocols, with the convention that the first line of each message (i.e., file) contained the recipient's address. Many features were added later, such as the ability to send one message to a list of recipients. Multimedia capabilities became important in the 1990s to send messages with images and other non-text material. In the following section, we will discuss three mail transfer protocols such as SMTP, MIME and POP.

2.1 Simple Mail Transfer Protocol (SMTP)

The basic Internet mail protocols provide mail and message exchange between TCP/IP hosts, but generally require that data be represented as 7-bit ASCII text. Because this can be restrictive, facilities have been added for the transmission of data that cannot be represented in this manner. Simple Mail Transfer Protocol (SMTP) is a standard for the exchange of mail between two computers, which specified the protocol used to send mail between TCP/IP hosts.

Working Mechanism of SMTP: SMTP is based on end-to-end delivery: An SMTP client contacts the destination host's SMTP server directly, on well-known port 25, to deliver the mail. It keeps the mail item being transmitted until it has been successfully copied to the recipient's SMTP. This is different from the store-and-forward principle that is common in many mailing systems, where the mail item can pass through a number of intermediate hosts in the same network on its way to the destination and where successful transmission from the sender only indicates that the mail item has reached the first intermediate hop.

In various implementations, it is possible to exchange mail between the TCP/IP SMTP mailing system and the locally used mailing systems. These applications are called *mail gateways or mail bridges.* Sending mail through a mail gateway can alter the end-to-end delivery specification, because SMTP only guarantees delivery to the mail-gateway host, not

to the real destination host located beyond the TCP/IP network. When a mail gateway is used, the SMTP end-to-end transmission is in the following three categories such as: host-to-gateway, gateway-to-host, or gateway-to-gateway. **SMTP messages**

In SMTP, each message has: *a header, or envelope, the structure of which is strictly defined by RFC 2822.* The mail header is terminated by a null line (that is, a line with nothing preceding the <CRLF> sequence). Everything after the null (or blank) line is the message body, *which is a sequence of lines containing ASCII characters* (that is, characters with a value less than 128 decimal).

RFC 2821 defines a client/server protocol. As usual, the client SMTP (referred to as the *sending* SMTP) is the entity that initiates the session, and the server (referred to as the *receiving* SMTP) is the one that responds to the session request. Because the client SMTP frequently can also act as a server for a user mailing program, it is often simpler to refer to the client as the sender SMTP and to the server as the receiver SMTP.

The SMTP destination address

SMTP destination address is also known as the mailbox address, the general form of the destination address is *local-part@domain-name.* This can take several forms as shown in table 11.1.

Table 11.1: SMTP destination address

Form	**Description**
user@host	For a direct destination on the same TCP/IP network.
user%remote-host@ gateway-host	For a user on a non-SMTP destination remote-host, through the mail gateway-host.
@host-a,@host-b:user@ host-c	For a relayed message. This contains explicit routing information. The message is first delivered to host-a, who re-sends (relay) the message to host-b. Host-b then forwards the message to the real destination host-c. Note that the message is stored on each of the intermediate hosts, so we do not have an end-to-end delivery in this case.

Mail header format

Typically, a user does not have to worry about the message header because it is managed by SMTP itself. The header is a list of specifications in the form of *keyword: value.* Table 11.2 shows common SMTP header fields.

Table 11.2: Common SMTP header fields

Keyword	Value
To	Primary recipients of the message
cc	Secondary(carbon-copy) recipients of the message
From	The sender of the message
Reply-to	The mailbox to which responses are to be sent. This field is added by the sender.
Return-path	The address and route by which the sender can be reached. This field is added by the final transport system that delivers the mail
Subject	A summary of the message being sent. This is specified by the sender

A sample header might appear as follows:

From: myEmail@mydiv.redbookscorp.com

To: "Your Email" <yourEmail@yourdiv.redbookscorp.com>

cc: "Your Boss" <yourBoss@yourdiv.redbookscorp.com>

Reply-To: myEmail@mydiv.redbookscorp.com

Subject: This is a sample SMTP header

Mail exchange

The SMTP design is based on the model of communication shown in Figure 11.1. As a result of a user mail request, the sender SMTP establishes a two-way connection with the receiver SMTP. The receiver SMTP can be either the ultimate destination or an intermediate (mail gateway). The sender SMTP will generate commands that are replied to by the receiver SMTP.

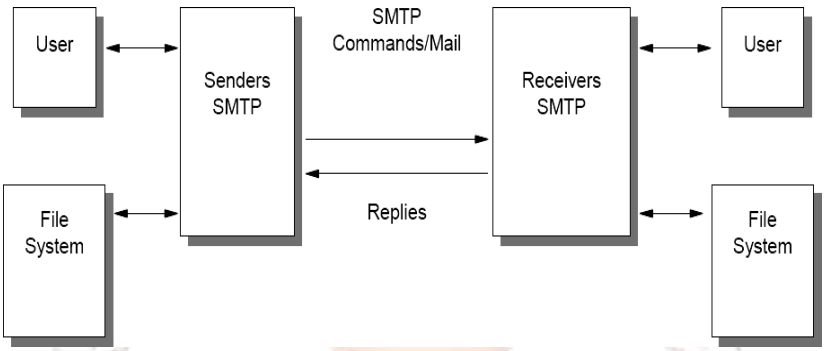

Fig 11.1: The SMTP model

SMTP mail transaction flow

Although mail commands and replies are rigidly defined, the exchange can easily be followed in Figure 11.2. All exchanged commands, replies, and data are text lines delimited by a <CRLF>. All replies have a numeric code at the beginning of the line.

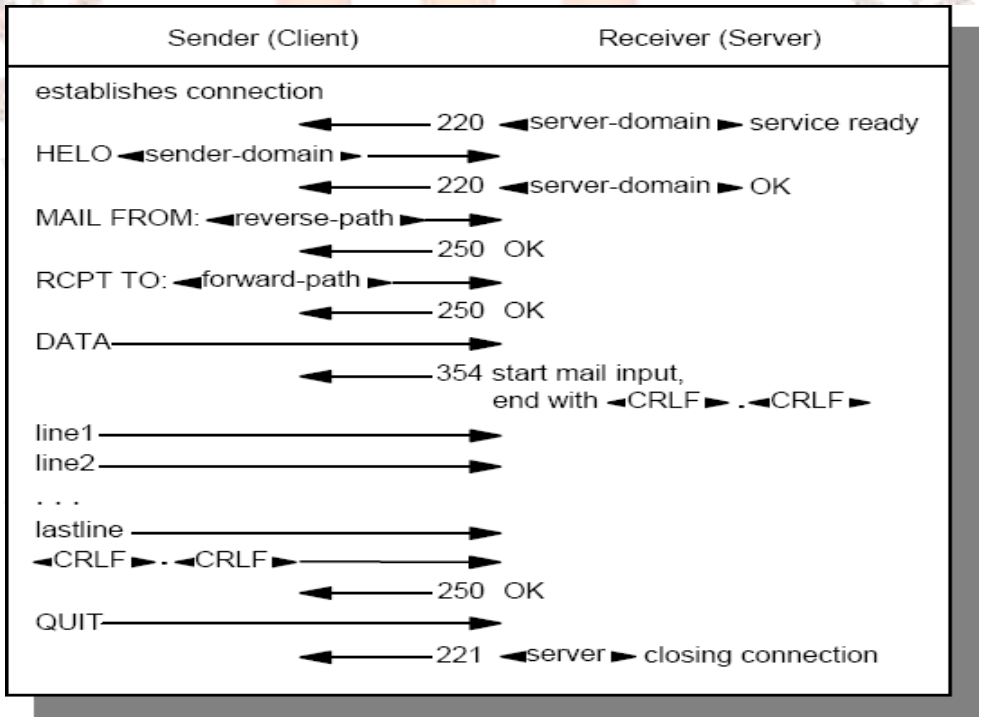

Fig 11.2: SMTP: Normal SMTP data flow – One mail message to one destination mailbox

The steps of this flow are:

1. The sender SMTP establishes a TCP connection with the destination SMTP and then waits for the server to send a 220 Service ready message or a 421 Service not available message when the destination is temporarily unable to proceed.

2. HELO (HELO is an abbreviation for hello) is sent, to which the receiver will identify itself by sending back its domain name. The sender-SMTP can use this to verify that it contacted the right destination SMTP. The sender SMTP can substitute an EHLO command in place of the HELO command. A receiver SMTP that does not support service extensions will respond with a 500 Syntax Error, command unrecognized message. The sender SMTP then retries with HELO, or if it cannot transmit the message without one or more service extensions, it sends a QUIT message. If a receiver-SMTP supports service extensions, it responds with a multiline 250 OK message, which includes a list of service extensions that it supports.
3. The sender now initiates the start of a mail transaction by sending a MAIL command to the receiver. This command contains the reverse-path that can be used to report errors.
4. The second step of the actual mail exchange consists of providing the server SMTP with the destinations for the message. There can be more than one recipient. This is done by sending one or more RCPTTO :< forward-path> commands. Each of them will receive a reply OK if the destination is known to the server, or a No if such user here is not.
5. When all RCPT commands are sent, the sender issues a DATA command to notify the receiver that the message contents will follow. The server replies with 354 Start mail input, end with <CRLF>.<CRLF>.
6. The client now sends the data line by line, ending with the 5-character sequence <CRLF>.<CRLF> line, upon which the receiver will acknowledge with a 250 OK, or an appropriate error message if anything went wrong.
7. At this juncture, the client now has several possible actions: If the client has no more messages to send, it can end the connection with a QUIT command, which will be answered with a 221 Service closing transmission channel reply. If the sender has no more messages to send, but is ready to receive messages (if any) from the other side, it can issue the TURN command. The two SMTPs now switch their role of sender/receiver, and the sender (previously the receiver) can now send messages by starting with step 3. If the sender has another message to send, it returns to step 3 and sends a new MAIL command.

2.2 Multipurpose Internet Mail Extensions (MIME)

Electronic mail is probably the most widely used TCP/IP application. However, SMTP is limited to 7-bit ASCII text, with a maximum line length of 1000 characters. This results in a number of limitations like, SMTP cannot transmit executable files or other binary objects. SMTP cannot transmit text data that includes national language characters, because these are represented by code points with a value of 128 (decimal) or higher in all character sets based on ASCII. SMTP servers might reject mail messages over a certain size. Because, any given server can have permanent or transient limits, or both, on the maximum amount of mail data it can accept from a client at any given time. SMTP gateways that translate from ASCII to EBCDIC and vice versa do not use a consistent set of code page mappings, resulting in translation problems. Some SMTP implementations or other mail transport agents (MTAs) in the Internet do not adhere completely to the SMTP standards defined in RFC 2821.

MIME (Multipurpose Internet Mail Extensions) is a communication protocol which allows the transmission of data in many forms such as audio, video, images and ASCII text handled in the e-mail protocol (SMTP). MIME allows mail messages to contain:

- Multiple objects in a single message
- Text having unlimited line length or overall length
- Character sets other than ASCII, allowing non-English language messages
- Multi-font messages
- Binary or application specific files
- Images, Audio, Video and multi-media messages.

MIME header

A MIME-compliant message must contain a header field. The general syntax for MIME header fields is the same as that for RFC 2822, having the format:

Keyword: Value

Therefore, the following field is valid (parenthetical phrases are treated as comments and ignored):

MIME-Version: 1.0 (this is a comment)

Table 11.3 below lists and defines the five MIME headers.

Table 11.3: MIME Headers

Keyword	Value
MIME-Version	As noted earlier, it must have the value 1.0
Content-Type	This describes how the object within the body is to be interpreted. The default value is text/plain. The default character set parameter for this is the charset=us-ascii, which indicates unformatted 7-bit ASCII text data.
Content-Transfer-Encoding	This describes how the object within the body was encoded in order that it be included in the message using a mail safe form.
Content-Description	A plain-text description of the object within the body, which is useful when the object is not human-readable
Content-ID	A world-unique value identifying the content of this part of this message

A secure version of MIME, S/MIME (Secure/Multipurpose Internet Mail Extensions), is defined to support encryption of email messages. Based on the MIME standard, S/MIME provides cryptographic security services such as authentication, message integrity, non-repudiation, privacy and data security for e-mail applications.

S/MIME can be used by traditional mail user agents to add cryptographic security services to mail that is sent and to interpret cryptographic security services in the mail that is received. However, S/MIME is not restricted to mail, it can be used with any transport mechanism that transports MIME data, such as HTTP. S/MIME takes advantage of the object-based features of MIME and allows secure messages to be exchanged in mixed-transport systems.

S/MIME can also be used in automated message transfer agents that use cryptographic security services that do not require any human intervention, such as the signing of software generated documents and the encryption of FAX messages sent over the internet.

2.3 Post Office Protocol (POP)

The Post Office Protocol, version 3 (POP3), is a standard protocol with standard number 53. The Post Office Protocol is an electronic mail protocol with both client (sender/receiver) and

server (storage) functions. POP3 supports basic functions (download and delete) for electronic mail retrieval.

Connection states

POP3 commands consist of a keyword and possibly one or more arguments following the keyword. Keywords are three or four characters long, and are separated from the arguments by a space. Each argument can be up to 40 characters long.

The server must send a response to each command issued by the client. This response can be up to 512 characters, and must begin with a status indicator signifying if the reply is positive or negative. These indicators are *+OK* or *-ERR*, and must be sent in uppercase.

As noted previously, POP3 interactions exist in three states. Commands can be issued from the authorization and transaction states, but not from the update state. With the exception of the QUIT command (which can be executed in both the authorization and transaction state), each command can only be executed in one of the states.

After a POP3 client establishes a TCP connection to the server (using well-known port 110), the interaction between the client and server passes through three distinct states:

1. *Authorization state:* First, the POP3 server sends a greeting message to the client. Following this, the session then enters the authentication state. During this state, the client must authenticate itself to the server. This can be done using one of three methods:
   - USER/PASS: The combined use of a user ID and password
   - APOP: Used to specify a name and an MD5 digest
   - AUTH: Used to specify a mechanism (such as TLS) by which both authentication and data protection can be provided
2. *Transaction state:* If the server successfully authenticates the client, the session enters the transaction state in which the client can access the mailbox. In this state, different commands include:
   - STAT: Retrieve the number of messages and total size of the messages.
   - LIST [msg#]: If no *msg* number is provided, retrieve information about each message present in the mailbox. If a msg number is specified, the server returns information for that message.

- RETR msg: Retrieve message number msg.
- DELE msg: Delete message number msg.
- NOOP: Do nothing. The server returns a positive response.
- RSET: Cancel any previous delete commands.
- QUIT: Update the mailbox (delete any messages requested previously) and then end the TCP connection.

3. *Update state:* after the client sends the QUIT command, the session enters the update state. During this state, the server provides all of the changes requested by the client's commands and then closes the connection. If the connection is closed for any reason, before a QUIT command is issued, none of the client's commands will take effect.

SELF-ASSESSMENT QUESTIONS – 1

1. _____protocol is based on *end-to-end delivery*.
2. SMTP destination address is also known as _____.
3. MIME is an acronym for _____.
4. Which protocol allows the transmission of data in many forms such as audio, video, images etc.?
   a) SMTP
   b) POP3
   c) MIME
   d) HTTP
5. The general syntax for MIME header fields has the format: _____
6. The _____ is an electronic mail protocol with both client and server functions.
7. First, the POP3 server sends a greeting message to the client. Following this, the session then enters the _____state.
8. After the client sends the QUIT command, the session enters the _____state.

3. HTTP

The *HyperText Transfer Protocol (HTTP)* is the Web's application-layer protocol. HTTP is implemented in two programs: a client program and a server program. The client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages. HTTP defines the structure of these messages and how the client and server exchange the messages.

A webpage consists of objects (objects are files like HTML file, JPEG image etc). Web browser (such as internet explorer and Firefox) implement the client side of HTTP and web servers (such as Apache and Microsoft Internet Information Server) implement the server side of HTTP. HTTP defines how web clients request Web pages from Web servers and how servers transfer web pages to clients.

3.1 Non-persistent and Persistent Connections

When client-server interaction is taking place over TCP, the application developer needs to make an important decision whether each request/response pair be sent over a *separate* TCP connection, or should all of the requests and their corresponding responses be sent over the same TCP connection. In the former approach, the application is said to use *non-persistent connections;* and in the latter approach, application has to use *persistent connections.*

HTTP with non-persistent connections

In case of non-persistent connections, the steps of transferring a web page from server to client are given below:

1. The HTTP client process initiates a TCP connection to the server on port number 80, which is the default port number for HTTP. Associated with the TCP connection, there will be a socket at the client and a socket at the server.
2. The HTTP client sends an HTTP request message to the server via its socket. The request message includes the path name.
3. The HTTP server process receives the request message via its socket, retrieves the object from its storage, encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.
4. The HTTP server process tells TCP to close the TCP connection.

5. The HTTP client receives the response message. The TCP connection terminates. The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file.

The steps above illustrate the use of non-persistent connections, where each TCP connection is closed after the server sends the object. The connection does not persist for other objects. Note that each TCP connection transports exactly one request message and one response message.

The amount of time that elapses from when a client requests the base HTML file until the entire file is received by the client is known as *round-trip time (RTT).* RTT is the time it takes for a small packet to travel from client to server and then back to the client. The RTT includes packet-propagation delays, packet queuing delays in intermediate routers and switches, and packet-processing delays.

HTTP with persistent connections

Non-persistent connections have some shortcomings. First, a brand-new connection must be established and maintained for each requested object. For each of these connections, TCP buffers must be allocated and TCP variables must be kept in both the client and server. This can place a significant burden on the web server, which may be serving requests from hundreds of different clients simultaneously. Second, each object suffers a delivery delay of two RTTs. One RTT to establish the TCP connection and one RTT to request and receive an object with persistent connections, the server leaves the TCP connection open after sending a response. Subsequent requests and responses between the same client and server can be sent over the same connection. In particular, an entire Web page can be sent over a single persistent TCP connection. Moreover, multiple web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection. These requests for objects can be made back-to-back, without waiting for replies to pending requests. Typically, the HTTP server closes a connection when it isn't used for a certain time. When the server receives the back-to-back requests, it sends the objects back-to-back. The default mode of HTTP uses persistent connections with pipelining.

3.2 HTTP Message Format

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

HTTP request message

A typical HTTP request message is given below:

GET /somedir/page.html HTTP/1.1

Host: www.smu.edu

Connection: close

User-agent: Mozilla/5.0

Accept-language: Eg

We can see that this message is written in ASCII text. This request message has five lines. The request message can have more than five lines or as few as one line. The first line of an HTTP request message is called the *request line*; the subsequent lines are called the *header lines*. The request line has three fields: the method field, the URL field, and the HTTP version field. The method field can take on several different values, including GET, POST, HEAD, PUT, and DELETE. The majority of HTTP request messages use the GET method. The GET method is used when the browser requests an object, with the requested object identified in the URL field. In the example, URL field is /somedir/page.html and browser implements the version HTTP/1.1. Header line *Host*: www.smu.edu specifies the host on which the object resides. By including the Connection: *close* header line, the browser is telling the server that it doesn't want to bother with persistent connections; it wants the server to close the connection after sending the requested object. The *User-agent: header line* specifies the user agent, that is, the browser type that is making the request to the server. Here the user agent is Mozilla/5.0, a Firefox browser. This header line is useful because the server can actually send different versions of the same object to different types of user agents. Finally, the *Accept-language:* header indicates that the user prefers to receive an English version of the object, if such an object exists on the server; otherwise, the server should send its default version. Figure 11.3 shows the general format of a request message.

In figure 11.3, after the header lines, there is an entity body. The entity body is empty with the GET method, but is used with the POST method. An HTTP client often uses the POST method when the user fills out a form. With a POST message, the user is still requesting a web page from the server, but the specific contents of the web page depend on what the user

entered into the form fields. If the value of the method field is POST, then the entity body contains what the user entered into the form fields.

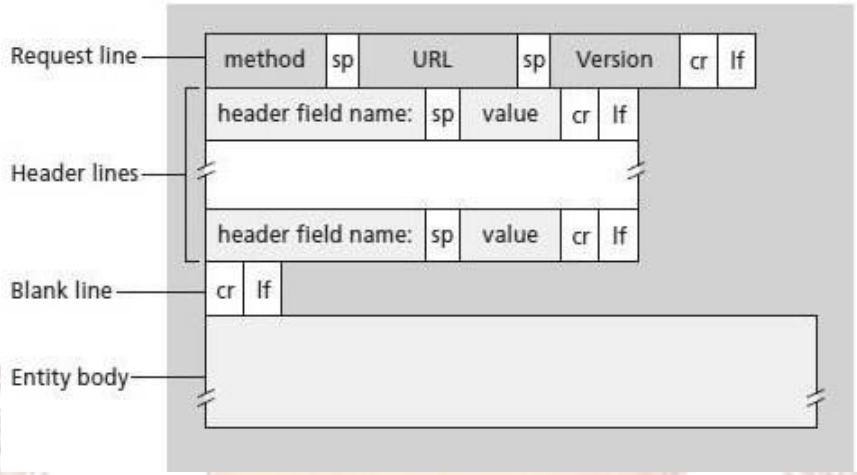

Fig 11.3: General format of an HTTP request message

HTTP response message

A typical HTTP response message is given below. This response message could be the response to the example request message just discussed.

HTTP/1.1 200 OK

Connection: close

Date: Tue, 13 Jan 2015 16:15:04 GMT

Server: Apache/2.2.3 (CentOS)

Last-modified: Tue, 13 Jan 2015 16:11:05 GMT

Content-length: 5823

Content-type: text/html

(Body)

In this response message, it has three sections. An initial *status line,* six *header lines,* and then the *entity body*. Entity body contains the requested object itself. The status line has three fields: the protocol version field, a status code, and a corresponding status message. In this example, the status line indicates that the server is using HTTP/1.1 and that everything is OK. The server uses the *Connection: close header line* to tell the client that it is going to close the TCP connection after sending the message. The *Date: header line* indicates the time and date when the HTTP response was created and sent by the server. Note that this is not the

time when the object was created or last modified; it is the time when the server retrieves the object from its file system, inserts the object into the response message, and sends the response message. The *Server: header line* indicates that the message was generated by an Apache Web server; it is analogous to the User-agent: header line in the HTTP request message. The *Last-Modified: header line* indicates the time and date when the object was created or last modified. The *Content-Length: header line* indicates the number of bytes in the object being sent. The *Content-Type*: *header line* indicates that the object in the entity body is HTML text. Figure 11.4 shows the general format of a response message. In that the status code and associated phrase indicate the result of the request. Some common status codes and associated phrases include:

- 200 OK: Request succeeded and the information is returned in the response.
- 301 Moved Permanently: Requested object has been permanently moved.
- 400 Bad Request: This is a generic error code indicating that the request could not be understood by the server.
- 404 Not Found: The requested document does not exist on this server.
- 505 HTTP Version Not Supported: The requested HTTP protocol version is not supported by the server.

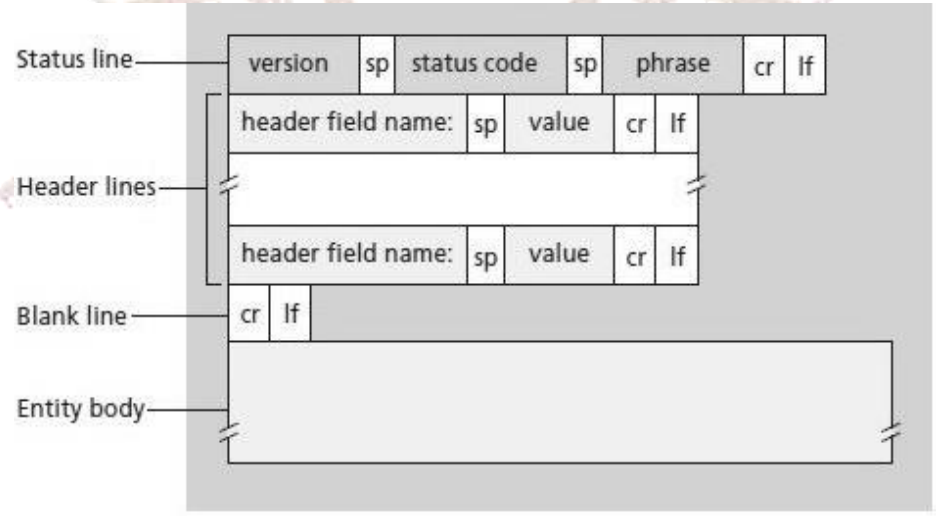

Fig 11.4: General format of an HTTP response message

3.3 Cookies

A cookie, also known as an HTTP cookie is a small piece of data sent from a website and stored in a user's web browser while the user is browsing that website. HTTP server is stateless, web servers can handle thousands of simultaneous TCP connections. However, it is often desirable for a web site to identify users, either because the server wishes to restrict user access or because it wants to serve content as a function of the user identity. For these purposes, HTTP uses cookies. Cookies allow sites to keep track of users.

Cookie technology has four components: (1) a cookie header line in the HTTP response message; (2) a cookie header line in the HTTP request message; (3) a cookie file kept on the user's end system and managed by the user's browser; and (4) a back-end database at the web site. Cookie technology is shown in figure 11.5.

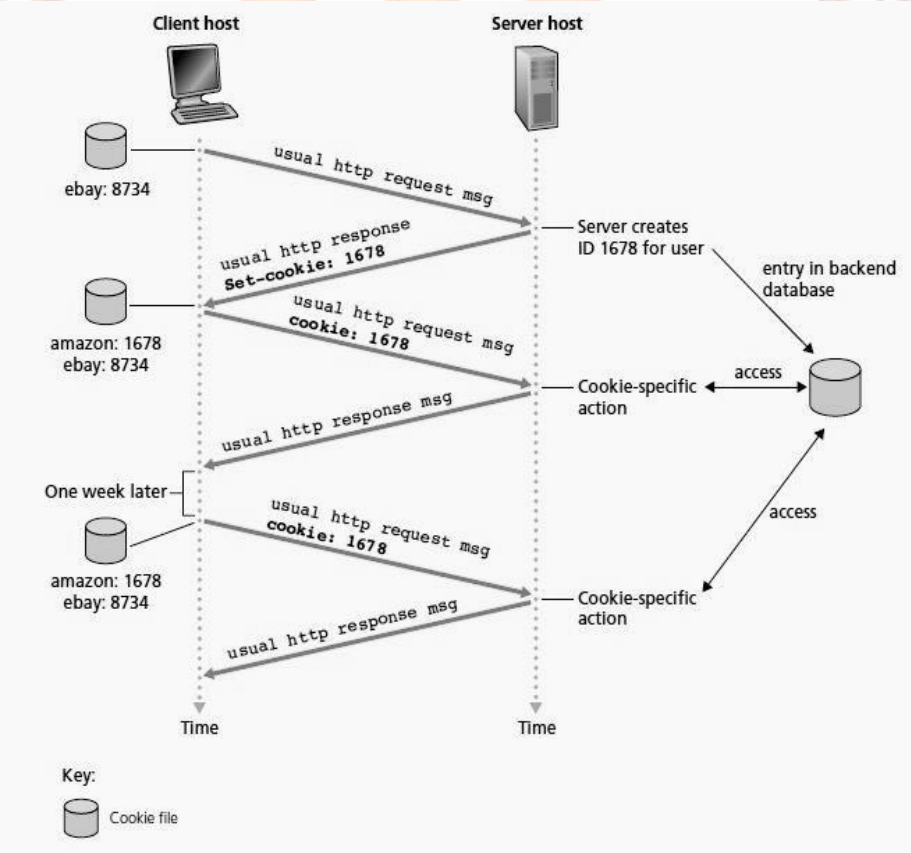

Fig 11.5: Keeping user state with cookies

Consider an example using figure 11.5. Suppose a client accesses the web page Amazon.com for the first time. Suppose that in the past he had already visited eBay site. When the request

comes into the Amazon Web server, the server creates a unique identification number and creates an entry in its back-end database that is indexed by the identification number. The Amazon Web server then responds to this client's browser, including in the HTTP response a *Set-cookie: header,* which contains the identification number. For example, the header line might be:

Set-cookie: 1678

When client's browser receives the HTTP response message, it sees the *Setcookie: header.* The browser then appends a line to the special cookie file that it manages. This line includes the hostname of the server and the identification number in the *Set-cookie: header*. Note that the cookie file already has an entry for eBay, since this client has visited that site in the past. As he continues to browse the Amazon site, each time he requests a web page, his browser consults his cookie file, extracts his identification number for this site, and puts a cookie header line that includes the identification number in the HTTP request. In this manner, the Amazon server is able to track this client's activity at the Amazon site. Although the Amazon web site does not necessarily know client's name, it knows exactly which pages user 1678 visited, in which order, and at what times.

From this discussion we see that cookies can be used to identify a user. The first time a user visits a site, the user can provide a user identification. During the subsequent sessions, the browser passes a cookie header to the server, thereby identifying the user to the server. Cookies can thus be used to create a user session layer on top of stateless HTTP.

3.4 Web Caching

A web cache, also called a ***proxy server***, is a network entity that satisfies HTTP requests on the behalf of an original web server. The Web cache has its own disk storage and keeps copies of recently requested objects in this storage. A user's browser can be configured so that all of the user's HTTP requests are first directed to the Web cache. Once a browser is configured, each browser request for an object is first directed to the web cache as shown in figure 11.6.

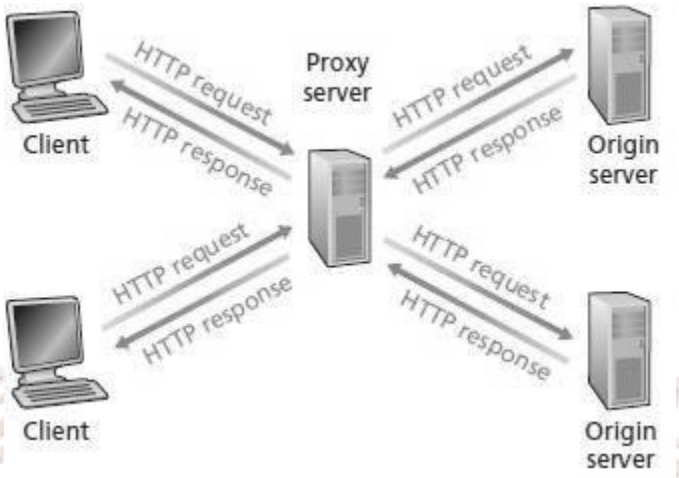

Fig 11.6: Clients requesting objects through a web cache

Consider the example, suppose a browser is requesting an object http://www.smu.edu/courses.html. The following are the different steps, which happens after the request.

1. The browser establishes a TCP connection to the Web cache and sends an HTTP request for the object to the Web cache.
2. The Web cache checks to see if it has a copy of the object stored locally. If it does, the Web cache returns the object within an HTTP response message to the client browser.
3. If the Web cache does not have the object, the Web cache opens a TCP connection to the origin server, that is, to www.smu.edu. The web cache then sends an HTTP request for the object into the cache-to-server TCP connection. After receiving this request, the origin server sends the object within an HTTP response to the web cache.
4. When the Web cache receives the object, it stores a copy in its local storage and sends a copy, within an HTTP response message, to the client browser.

A cache is both a server and a client at the same time. When it receives requests from and sends responses to a browser, it is a server. When it sends requests to and receives responses from an origin server, it is a client.

Web caching has seen deployment in the Internet for two reasons. First, a web cache can substantially reduce the response time for a client request. Second, web caches can substantially reduce traffic on an institution's access link to the Internet. By reducing traffic, the institution does not have to upgrade bandwidth as quickly, thereby reducing costs.

Furthermore, Web caches can substantially reduce web traffic on the Internet as a whole, thereby improving performance for all applications.

SELF-ASSESSMENT QUESTIONS – 2

10. HTTP is the acronym for _____.
11. Which of the following is an example of webserver?
    a) Firefox
    b) Internet Explorer
    c) Apache
12. The connection in which each request/response pair is sent over a *separate* TCP connection is known as _____.
13. The first line of an HTTP request message is called _____ and the subsequent lines are called _____.
14. allow sites to keep track of users.
15. A web cache, also called a _____ is a network entity that satisfies HTTP requests on the behalf of an original web server.

4. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- Simple Mail Transfer Protocol (SMTP) is a standard for the exchange of mail between two computers.
- SMTP is based on *end-to-end delivery:* An SMTP client contacts the destination host's SMTP server directly, on well-known port 25, to deliver the mail.
- *MIME (Multipurpose Internet Mail Extensions)* is a communication protocol which allows the transmission of data in many forms such as audio, video, images and ASCII text handled in the e-mail protocol.
- The Post Office Protocol is an electronic mail protocol with both client (sender/receiver) and server (storage) functions. POP3 supports basic functions (download and delete) for electronic mail retrieval.
- The *HyperText Transfer Protocol (HTTP)* is the Web's application-layer protocol.
- HTTP is implemented in two programs: a client program and a server program.
- Each request/response pair is sent over a *separate* TCP connection. This is known as a non-persistent connection.
- All of the requests and their corresponding responses be sent over the same TCP connection. This is known as persistent connection.
- Cookies allow sites to keep track of users.
- Cookie technology has four components: (1) a cookie header line in the HTTP response message; (2) a cookie header line in the HTTP request message; (3) a cookie file kept on the user's end system and managed by the user's browser; and (4) a back-end database at the web site.

5. TERMINAL QUESTIONS

1. Describe SMTP.
2. Explain MIME.
3. Write short note on Post Office Protocol, Version 3.
4. Explain HTTP request and response messages.
5. Explain the use of Cookies in web.
6. Describe web caching.

6. ANSWERS

Self-Assessment Questions

1. SMTP
2. Mailbox address
3. Multipurpose Internet Mail extensions (MIME)
4. (c) MIME
5. Keyword:value
6. Post Office Protocol, version 3 (POP3)
7. Post Office Protocol
8. Authentication state
9. Update state
10. HyperText Transfer Protocol (HTTP)
11. (c) Apache
12. Non-persistent connections
13. Request line, header lines
14. Cookies
15. Proxy server

Terminal Questions

1. Simple Mail Transfer Protocol (SMTP) is a standard for the exchange of mail between two computers (STD 10/RFC 821), which specified the protocol used to send mail between TCP/IP hosts. (Refer section 2.1 for more details).

2. *MIME (Multipurpose Internet Mail Extensions)* is a communication protocol which allows the transmission of data in many forms such as audio, video, images and ASCII text handled in the e-mail protocol (SMTP). (Refer section 2.2 for more details).
3. The Post Office Protocol, version 3 (POP3), is a standard protocol with STD number 53. The Post Office Protocol is an electronic mail protocol with both client (sender/receiver) and server (storage) functions. (Refer section 2.3 for more details).
4. The request message can have more than five lines or as few as one line. *The first line* of an HTTP request message is called the *request line;* the subsequent lines are called the *header lines*. In the response message, it has three sections. An initial *status line,* six *header lines,* and then the *entity body.* (Refer section 3.2 for more details).
5. HTTP server is stateless, web servers can handle thousands of simultaneous TCP connections. However, it is often desirable for a web site to identify users, either because the server wishes to restrict user access or because it wants to serve content as a function of the user identity. For these purposes, HTTP uses cookies. Cookies allow sites to keep track of users. (Refer section 3.3 for more details).
6. A web cache, also called a proxy server, is a network entity that satisfies HTTP requests on the behalf of an original web server. The Web cache has its own disk storage and keeps copies of recently requested objects in this storage. (Refer section 3.4 for more details).

References:

- Andrew S Tanenbaum, David J. Wetherall, *"Computer Networks,"* Fifth edition.
- Larry L. Peterson, Bruce S. Davie, *"Computer Networks – a Systems Approach,"* Fifth edition.
- James F. Kurose, Keith W.Ross, *"Computer Networking – A top-down approach,"* Sixth edition.
- Behrouz A. Forouzan, Sophia Chung Fegan, *"Data Communication and Networking,"* Fourth edition.
- William Stallings, *"Computer Networking with Internet Protocols and Technology,"* Third edition.