Problem 1

How does the web server (e.g., Amazon) identify users when you do the Internet shopping? Please briefly explain how it works with HTTP protocol step by step.

Write your solution to Problem 1 in this box

Largely, there are 2 methods web servers use to identify users:

① HTTP Authentications: Forms on websites are especially created to have users input their username and password. On successful login, the user can be identified and special actions can be performed. However, the problem is that this state is not saved, so the user will be required to re-authenticate/sign in every time.

(sends their first HTTP request)

② Cookies: When a user accesses/visits a website for the first time, the web server creates and issues a unique identifier called a 'cookie' for that particular user. This cookie is saved on the backend database of the server and is also sent to the user's web browser where it is saved in a user's local memory within a cookie file that the browser has access to.
When subsequent requests are sent from the client to the server, the browser attaches the cookie in the header of the request. The server can grab this from the request header, identifying that the request has come from the same browser, usually keeping the user logged in and allowing specific actions to be performed if needed.

Problem 2

Suppose within your Web browser you click on a link to obtain a Web page from a web server S. The web page is a HTML file and the HTML file further contains references to 9 small JPEG files on the same server S. However, S's IP address is not cached in your local host, so a DNS lookup is required. Suppose that n DNS servers are visited by your browser before you get S's IP address. Let RTT_1, RTT_2, ..., RTT_n denote the RTTs (round-trip time) of visiting each of the n DNS server and RTT_0 denote the RTT between the local host and S. If we ignore file transmission time for DNS responses, HTML file, and JPEG files, how much time elapses from when the client clicks on the link until the client receives all objects with:

(a) Non-persistent HTTP with no parallel TCP connections?

(b) Non-persistent HTTP with the browser configured for 5 parallel connections?

(c) Persistent HTTP with no parallel TCP connections?

(d) Persistent HTTP with the browser configured for arbitrarily many parallel connections?

Write your solution to Problem 2 in this box

Total amount of time to get IP addresses is $RTT_1 + RTT_2 + \dots + RTT_n$

1 RTT_0 sets up the TCP connection, and a second RTT_0 is needed to receive and request the 9 small JPEG files.

$\therefore T = 2RTT_0 + \sum_{i=1}^{n} RTT_i$

(a) $T = \underbrace{2*9RTT_0}_{\text{For 9 JPEG objects}} + \underbrace{2RTT_0}_{\text{Bootstrap HTTP connection \& grab HTML page}} + \underbrace{\sum_{i=1}^{n} RTT_i}_{\text{to get IP address}} = \boxed{20RTT_0 + \sum_{i=1}^{n} RTT_i}$

(b) 5 parallel connections → Need to do this twice for 9 JPEG images

$\therefore T = \underbrace{2*2RTT_0}_{\text{For parallel connections}} + \underbrace{2RTT_0}_{\text{Bootstrap HTTP connection \& grab HTML page}} + \underbrace{\sum_{i=1}^{n} RTT_i}_{\text{to get IP address}} = \boxed{6RTT_0 + \sum_{i=1}^{n} RTT_i}$

(c) $T = \underbrace{9RTT_0}_{\text{for 9 JPEG objects}} + \underbrace{2RTT_0}_{\text{Bootstrap HTTP connection \& grab HTML page}} + \underbrace{\sum_{i=1}^{n} RTT_i}_{\text{to get IP address}} = \boxed{11RTT_0 + \sum_{i=1}^{n} RTT_i}$

(d) $T = \underbrace{RTT_0}_{\text{To get all 9 JPEG images}} + \underbrace{2RTT_0}_{\text{Bootstrap HTTP connection \& grab HTML page}} + \underbrace{\sum_{i=1}^{n} RTT_i}_{\text{to get IP address}} = \boxed{3RTT_0 + \sum_{i=1}^{n} RTT_i}$

Problem 3

How does SMTP marks the end of a message body? How about HTTP? Can HTTP use the same method as SMTP to mark the end of the message body?

Write your solution to Problem 3 in this box

SMTP EOM signal consists of a CRLF followed by a terminating dot/full-stop followed by another CRLF (Typically, CRLF indicates a new line, i.e., a carriage return immediately followed by a line Feed)

HTTP requests mark the EOM body by specifying the content length in the "content-length" attribute field in the request header.

No, HTTP cannot use the same method. Apart from the different EOM body formats, SMTP requires each message, including the body of each message, to be in 7-bit ASCII format. If the message contains characters that are not 7-bit ASCII or contain binary data, then the message has to be encoded into 7-bit ASCII. HTTP data does not impose this restriction.

Problem 4

Suppose your department has a local DNS server for all computers in the department.

(a) Suppose you are an ordinary user (i.e., not a network/system administrator). Can you determine if an external Web site was likely accessed from a computer in your department a couple of seconds ago? Explain.

(b) Now suppose you are a system administrator and can access the caches in the local DNS servers of your department. Can you propose a way to roughly determine the Web servers (outside your department) that are most popular among the users in your department? Explain.

Write your solution to Problem 4 in this box

(a) This can be done using a DNS resolver. If we no idea what websites have been accessed recently, we can run the linux command 'ipconfig/displaydns' to see which records are saved in the cache of the DNS resolver. Alternatively, we can make queries using the dig tool on most linux/unix machines. The dig service performs a DNS lookup, returning query and access times, server IP, when and a few other parameters. If the resulting access time is fairly small, then we can conclude that the website was accessed recently.

(b) We can use the basic idea behind a cache, that is, that it typically stores data that is frequently/recently accessed. Therefore, if we take frequent snapshots of the DNS server's cache over a period of time and create a mapping of server names and their frequency, we can determine the external web servers that are most popular in our department.

Problem 5

Consider distributing a file of $F = 15$ Gbits to N peers. The server has an upload rate of $u_s = 30$ Mbps, and each peer has a download rate of $d_1 = 2$ Mbps and an upload rate of u. For N = 10 and 100, and u = 300 Kbps and 2 Mbps, prepare a chart giving the minimum distribution time for each of the combinations of N and u for both client-server distribution and P2P distribution (i.e., there are 8 distribution time values in total). In this problem, we assume $1K = 1 \times 10^3$, $1M = 1 \times 10^6$, $1G = 1 \times 10^9$.

Please fulfill your answers into the following two charts and briefly explain how you get them.

Write your solution to Problem 5 in this box

Client-Server distribution

u\N	10	100
300Kbps	7,500 seconds	50,000 seconds
2Mbps	7,500 seconds	50,000 seconds

P2P distribution

u\N	10	100
300Kbps	7,500 seconds	25,000 seconds
2Mbps	7,500 seconds	7,500 seconds

Client-server:

(1) $N=10$, $d = 2$ Mbps

Time to distribute from servers: $\frac{FN}{U_S} = \frac{15\times10^9\times10}{30\times10^6} = 5\times10^3$s

Time for clients to download a copy: $\frac{F}{d_{min}} = \frac{15\times10^9}{2\times10^6} = \frac{7.5\times10^3}{2} = 7.5\times10^3$s

$D_{C-S} \geq \max\{NF/U_S, F/d_{min}\} \longrightarrow \boxed{7.5\times10^3 s}$

P2P

(1) $N=10$, $u = 300$ Kbps

Time to send one copy: $F/U_S = \frac{15\times10^9}{30\times10^6} = 500$s

Min client download time: $F/d_{min} = \frac{15\times10^9}{2\times10^6} = 7500$s

Max upload rate: $U_S + \sum_{i=1}^{10} U_i = 30\times10^6 + 10(300\times10^3) = 30\times10^6 + 3\times10^6 = 33\times10^6$s

$NF/(U_S+\sum U_i) = \frac{15\times10^9\times10^4}{11\times33\times10^6} = \frac{5\times10^4}{11} = 4545$s

$\max\{F/U_S, F/d_{min}, NF/(U_S+\sum U_i)\} = \boxed{7500s}$