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.

The web server for online shopping websites identify users through the use of cookies included with the HTTP Protocol.The use of cookies is required because without them, base HTTP protocol is stateless and cannot tell if the client has visited it before. The steps for this are as follows (in this example, I'm using Amazon.com as the web server) :

(1) The first time connecting, the client sends a typical http request message to the online server.

(2) if this is the first time that the client is accessing the website, then the Amazon server creates a unique ID for the user. This ID is stored in the server's database, and is sent back as a cookie as part of the server's http response to the client. (for example: the usual http response + "set-cookie: 1333", 1333 being an arbitrary ID)

(3) From this point onwards, as long as the browser does not clear its cookies, the client's browser stores the cookie for future use. Thus, for all future http requests from the client to the server, the cookie value is also sent. (ie The next time the user visits the amazon website - or whichever website is storing the cookies- they send over the cookie value as part of any HTTP requests)
usual http request + "cookie:1333"

(4) If the cookie is passed to the web server, then the server looks up the ID of the cookie in its database. If there is an entry, then the server restores the user's session from where they last left off. This is what is meant by cookie specific actions, as the cookie is used as a reference to the user and their online session. This is used for actions such as reloading a user's cart from their last session, providing recommendations based on what they have previously viewed, or restoring any customizations specified by the user, among other uses.

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?

(a) Non-persistent HTTP with no parallel TCP connections: Add the RTT's for visiting each server, Plus 1 RTT to set up the TCP Connection, 1 RTT for the base HTML file, and 2 RTT for each of the 9 objects (1 to set up TCP connection each time since it is nonpersisent, and 1 for the actual data transmission)
$RTT_1 + RTT_2 + ...RTT_n + 2RRT_0 + 9(2RTT_0)$
$= 20RTT_0 + RTT_1 + RTT_2 + ... + RTT_n$

(b) Non-persistent HTTP with the browser configured for 5 parallel connections? $RTT_1 + RTT_2 + ...RTT_n + 2RRT_0 + 2(2RTT_0)$
$= 6RTT_0 + RTT_1 + RTT_2 + ...RTT_n$

(c) Persistent HTTP with no parallel TCP connections? $RTT_1 + RTT_2 + ...RTT_n + 2RRT_0 + 9(RTT_0)$
$= 11RTT_0 + RTT_1 + RTT_2 + ... + RTT_n$

(d) Persistent HTTP with the browser configured for arbitrarily many parallel connections? $RTT_1 + RTT_2 + ...RTT_n + 2RTT_0 + RTT_0$
$= 3RTT_0 + RTT_1 + RTT_2 + ... + RTT_n$

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?

SMTP marks the end of the message body with a period and a blank like ("CRLF.CRLF") . In contrast, HTTP messages indicates the length of messages in the message header field. The HTTP protocol cannot use the SMTP method to end a message body because the HTTP protocol only looks at the information in the header field to determine the length of a message and where it should end. Also, the SMTP Protocol only allows 7-bit ASCII for the body of the message, while HTTP data messages allow for binary data. It would not matter where the location of the SMTP end message symbol is in an HTTP message; depending on the length specified, the message could still end before or after this appears.

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.

(a) You can see if an external Web site was accessed by comparing the DNS response time for this website to the DNS response to another website that was not recently accessed. If there is a significant difference, then you can determine that one website was accessed more recently, and therefore was part of the local DNS cache. Commands you could use for this would be dig or nslookup. If the query time returned is 0 or close to 0, then you know that a user in the network recently accessed the website. If the time to query is longer, than you know that the site is not in the cache, therefore no one in the network has accessed it recently.

(b) As a system administrator, you can determine the web servers that are most popular among your department by periodically checking the DNS caches over time and see which entries appear most often, since the more popular web sites will have more requests by the users in the department. This means that these popular servers will be part of the DNS caches that you have access to more often, so you will see them more often as you check the caches.

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.

Client-Server distribution

u\N	10	100
300Kbps	7500s	50000s
2Mbps	7500s	50000s

For the Client- Server chart, each entry gets a minimum time to distribute the file using the Equation: $D_{c-s} \geq \max\{\frac{NF}{u_s}, \frac{F}{d_{min}}\}$ where N = # of clients, F = File size = 15 Gbits = 15000 Mbits, U_s = server upload rate, and d_min = download rate = 2 Mbps. For this table, the u in each row of the table doesn't matter.

For the all 4 cases, the expression $\frac{F}{d_{min}}$ does not change values: $= \frac{1.5*10^4 Megabits}{2Mbps} = 7.5 * 10^3$ seconds. For the other expression in the max comparison, we have two cases for this table:

N = 10: $\frac{NF}{u_s} = \frac{10*(1.5*10^4 Megabits)}{30Mbps} = 5 * 10^3$ seconds

N=100: $\frac{NF}{u_s} = \frac{100*1.5*10^4 Megabits}{30Mbps} = 5 * 10^4$ seconds.

Therefore, the largest term for N=10 cases is 7500, while for N=100 it is 50000.

P2P distribution

u\N	10	100
300Kbps	7500s	25000s
2Mbps	7500s	7500s

For the P2P distribution chart, the minimum file distribution times are from the expression: $D_{P2P} \geq \max\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s+\sum u_i}\}$ where N = # of clients, F = File size = 15 Gbits = 15000 Mbits, U_s = server upload rate, d_min = download rate = 2 Mbps, and u_i = peer upload capacity. This u_i is what is adjusted between the rows in the table (300 Kbps and 2Mbps).

For the all 4 cases, the expression $\frac{F}{d_{min}}$ does not change values: $= \frac{1.5*10^4 Megabits}{2Mbps} = 7.5 * 10^3$ seconds.

Also, all 4 cases have the same value for $\frac{F}{u_s} = \frac{1.5*10^4 Mb}{30Mbps} = 500$ seconds.

The difference in all 4 boxes is in the last expression:

N=10, $u_i = 300Kbps = 0.3Mbps$: $\frac{10*(1.5*10^4 Mb)}{30Mbps+10*(0.3Mbps)} = 4545.45$ seconds.

N=100, $u_i = 300Kbps = 0.3Mbps$: $\frac{100*(1.5*10^4 Mb)}{30Mbps+100*(0.3Mbps)} = 25000$ seconds.

N=10, $u_i = 2Mbps$: $\frac{10*(1.5*10^4 Mb)}{30Mbps+10*(2Mbps)} = 3000$ seconds.

N=100, $u_i = 2Mbps$: $\frac{100*(1.5*10^4 Mb)}{30Mbps+100*(2Mbps)} = 6521.74$ seconds.

Therefore: For both cases in the table where N=10, the largest term is 7500. For the case with N=100, u=2 Mbps, the largest term is also the 7500. The difference comes in the case with N=100 and u=300 Kbps, where the largest term is 25000.