Problem 1

Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT or RTT_1, RTT_2, ..., RTT_n. Further suppose that the Web page associated with the link has a small amount of HTML text. Let RTT_0 denote the RTT between the local host and the server containing the HTML file. Assume zero transmission time. Suppose the HTML file references 11 very small objects on the same server. Neglect transmission times, 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?

DNS resolve time $t_{DNS} = \mathrm{RTT}_1 + \mathrm{RTT}_2 + \cdots + \mathrm{RTT}_n$. Once the DNS is resolved, setting up the TCP connection needs RTT_0 time and another RTT_0 to request and receive the small object. The total response time is $2 \cdot \mathrm{RTT}_0$.
Since the objects are very small, we assume that the content retrieve time is negligible. Therefore:

(a) Non-persistent HTTP with no parallel TCP connection $t_{HTTP} = 11 \cdot 2\mathrm{RTT}_0$.

Total time $t = \mathrm{RTT}_1 + \mathrm{RTT}_2 + \cdots + \mathrm{RTT}_n + 24\mathrm{RTT}_0$.

(b) With 5 parallel connections, 11 small objects need 3 batches to transmit, and each batch needs 2 RTT_0.
Total time $t = \mathrm{RTT}_1 + \mathrm{RTT}_2 + \cdots + \mathrm{RTT}_n + 8\mathrm{RTT}_0$.

(c) With persistent HTTP:

In non-pipelining mode, there will be 1 RTT_0 per object. Thus, 11 small objects need 11 RTT_0 to transmit, the total time $t = \mathrm{RTT}_1 + \mathrm{RTT}_2 + \cdots + \mathrm{RTT}_n + 13\mathrm{RTT}_0$.

(d) In pipelining mode, 1 RTT_0 is enough for all the reference objects in one round. Total time $t = \mathrm{RTT}_1 + \mathrm{RTT}_2 + \cdots + \mathrm{RTT}_n + 3\mathrm{RTT}_0$.

Problem 2

How does the web server (e.g., eBay) identify users when you do the Internet shopping? Briefly explain how it works.

eBay Web server uses cookie to identify users and their transaction histories. eBay Web server creates an entry in its back-end database that is indexed by a unique identification number when the user first browses the eBay website. This identification number is sent back in the HTTP response Set-cookie: header, and it will be saved to user's cookie file. Each time the user requests an eBay web page in the future, a cookie header line including the identification number is sent along the HTTP request. In this manner, the eBay server is able to identify user and track their shopping carts.

Problem 3

A Web browser running on the client host is requesting a webpage from the server. We make the following assumptions:

- TCP window is large once the TCP handshake is complete (i.e. ignore flow control). TCP header size is h bits, and the maximum payload size is p bits.
- The bandwidth is b bps, and the propagation delay is d seconds.
- Ignore DNS related delays, and ignore the payload in three-way handshake packets, ACK packets, and HTTP request packets. In other words, those packets consist of header only.
- The client requests a webpage consisting of an HTML file that indexes 5 binary files on the same server. Each of the file is $2p$ bits long. In other words, each of the file can be sent in exactly 2 TCP packets. Piggybacking is used whenever possible.
- Each HTTP request is sent in one TCP packet.

Please answer the following questions:

(a) Suppose pipelining of HTTP requests is allowed and no parallel TCP connections are used. Calculate the minimal time it takes the browser to receive all the files.

(b) Suppose the non-persistent, non-pipelining mode with parallel TCP connections is used, repeat the calculation.

(c) Which mode gives the smaller latency? Briefly justify your answer.

Problem 3 continued on next page...

(a) Three-way handshake (SYN, SYNACK, ACK + request) takes:

$$t_{handshake} = 3 \times (\frac{h}{b} + d);$$

Sending requested HTML file ($2 \cdot (h + p)$ bits):

$$t_{html} = \frac{2 \cdot (h + p)}{b} + d;$$

Client sends HTTP requests (header only) in a pipelined fashion:

$$t_{req} = \frac{h}{b} + d;$$

Server respond with 5 objects which takes 10 packets (header + payload):

$$t_{resp} = \frac{10 \cdot (h + p)}{b} + d;$$

Altogether:

$$t = t_{handshake} + t_{html} + t_{req} + t_{resp} = 16\frac{h}{b} + 12\frac{p}{b} + 6d.$$

(b) Client sets up 5 parallel TCP connections to request object (note that the physical bandwidth is shared equally among five TCP connections):

$$t'_{req} = 3 \times (\frac{h}{b/5} + d);$$

Altogether:

$$t' = t_{handshake} + t_{html} + t'_{req} + t_{resp} = 30\frac{h}{b} + 12\frac{p}{b} + 8d.$$

(c) Persistent HTTP request is better (since $t < t'$).

Problem 4

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 uses a line containing only a period to mark the end of a message body.
HTTP uses "Content-Length header field" to indicate the length of a message body.
No, HTTP cannot use the method used by SMTP, because HTTP message could be binary data, whereas in SMTP, the message body must be in 7-bit ASCII format.

Problem 5

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) Yes, we can use dig to query that Web site in the local DNS server. For example, dig www.cnn.com will return the query time for finding cnn.com. If www.cnn.com was just accessed a couple of seconds ago, an entry for www.cnn.com is cached in the local DNS cache, so the query time is 0 msec. Otherwise, the query time is large.

(b) We can examine the DNS cache periodically and take snapshots. By calculating the frequency of web server resolve messages, we can know that whichever appears most frequently in the DNS caches is the most popular server, because if more users are interested in a Web server, then DNS requests for that server are more frequently sent by users and will appear more frequently in the DNS cache.