# Problem 1

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

Write your solution to Problem 1 in this box

Web servers identify users for certain activities such as internet shopping, by using cookies. A cookie is a line that identifies a particular user. When a client connects to a web server for the first time, the server will create a cookie and store it in its backend database and respond to the request with a 'set cookie'
line in the response. Then, the clientside browser will store the cookie. Next time the client connects to the same webserver, it will send request messages along with the its cookie. The webserver will respond with the appropriate cookie-specific action. For example, if you add items to your shopping cart, and you close the web browser, and you open it for another session at the same website, then you will still see the items that you added last time, as the webserver knows who you are based on the cookie associated with your browser.

# Problem 2

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 (round-trip time) of $RTT_1$, $RTT_2$, ..., $RTT_n$. 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 ten very small objects on the same server. 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

a) RTT1 + RTT2 + ... + RTTn + 1 RTT0 to establish connection + 1 RTT0 to get root html file + 1 RTT0 to reestablish connection + 1 RTT0 to fetch first object + 1 RTT0 to reestablish connection + 1 RTT0 to fetch second object ...
= RTT1 + RTT2 + RTTn + 22 RTT0s

b) RTT1 + RTT2 + ... + RTTn + 1 RTT0 to establish connection + 1 RTT0 to get root html file + 1 RTT0 to reestablish connection + 1 RTT0 to get the first 5 objects + 1 RTT0 to reestablish connection + 1 RTT0 to get the second 5 objects
= RTT1 + RTT2 + ... + RTTn + 6 RTT0s

c) RTT1 + RTT2 + ... + RTTn + 1 RTT0 to establish connection + 1 RTT0 to get root html file + 10 RTT0's to get each object
= RTT1 + RTT2 + ... + RTTn + 12 RTT0s

d) RTT1 + RTT2 + ... + RTTn + 1 RTT0 to establish connection + 1 RTT0 to get root html file + 1 RTT to get all 10 objects
= RTT1 + RTT2 + ... + RTTn + 3 RTT0s

# Problem 3

Suppose Bob joins a BitTorrent torrent, but he does not want to upload any data to any other peers (so called free-riding).

(a) Bob claims that he can receive a complete copy of the file that is shared by the swarm. Is Bob's claim possible? Why or why not?

(b) Bob further claims that he can further make his "free-riding" more efficient by using a collection of multiple computers (with distinct IP addresses) in the computer lab in his department. How can he do that?

Write your solution to Problem 3 in this box

a) Free-riding is possible if the members of the swarm have different parts of the file that make up the whole file. If any part of the whole file is missing, then no one can finish downloading the file, unless a seeder connects to the swarm. A seeder is someone who already has the complete file and is uploading to other peers in the swarm. Therefore, as long as the different parts of the file are present in the peers connected to the swarm, Bob can successfully download the complete file.

b) Bob can make his free-riding more efficient by using multiple systems. He can configure each system to download a different part of the file. When all systems are finished downloading, he will have the complete file.

# 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?

Write your solution to Problem 4 in this box

SMTP marks the end of a message body with the client sending a single period on a new line.

HTTP marks the end of a message body with a part of its header that contains the size of the message being sent, in bytes. This part is called 'Content-Length'.

HTTP cannot use the same method as SMTP to mark the end of a message. SMTP has a restriction that requires the message to be encoded with only 7 bit ASCII characters, while HTTP has no restrictions. Therefore, a period on a new line could possibly still be part of the message for HTTP. The only way to be sure is to counter the number of characters that are in the message, or the number of bytes.

# 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.

Write your solution to Problem 5 in this box

a) Yes, you can determine if an external website was accessed from a computer in your department very recently. If an external website is accessed recently, then the website's IP address to hostname translation will be stored in the local DNS server. Thus, the lookup time for the translation will be quick. If it was not accessed recently, then the translation will not be found in the local DNS server, and so the local DNS must inquire the root DNS server and other TLD and authoritative servers to find the IP address to hostname translation. We can see this information using the 'dig' command.

b) Yes, we can find out which webservers are most popular among our department. We must keep track of which IP address to hostname translations are stored in the local DNS server over a period of time. The most frequently appearing translations are the most popular webservers in the department.