Reinaldo Daniswara
604840665

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?

Write your solution to Problem 1 in this box

1. Total amount of time to get IP address is

$$RTT_1 + RTT_2 + RTT_3 + \dots RTT_n$$

Then, RTT_0 set up the TCP connections, and another RTT_0 is to receive and request small objects. Thus,

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

a. Non-persistent HTTP with no parallel TCP connections?

$$2*11\,RTT_0 + 2\,RTT_0 + \sum_{i=1}^{n} RTT_i = 24\,RTT_0 + \sum_{i=1}^{n} RTT_i$$

Need 11 RTT_0 for 11 smal objects, and 2 RTT_0 to set up the HTTP page plus the IP address

b. Non-persistent HTTP with browser configure for 5 parallel connections

$$2*3\,RTT_0 + 2\,RTT_0 + \sum_{i=1}^{n} RTT_i = 8\,RTT_0 + \sum_{i=1}^{n} RTT_i$$

Need 2 RTT_0 for parallel connections, 2 RTT_0 to set up the HTTP page plus the IP address

c. Persistent HTTP with no parallel TCP

$$RTT_0 + 2\,RTT_0 + \sum_{i=1}^{n} RTT_i = 3\,RTT_0 + \sum_{i=1}^{n} RTT_i$$

Since it's persistent HTTP, Need 1 RTT_0 for 11 small objects requested simultaneously, 2 RTT_0 to set up the HTTP page plus the IP address

d. Persistent with many parallel connections

$$1\,RTT_0 + 2\,RTT_0 + \sum_{i=1}^{n} RTT_i = 3\,RTT_0 + \sum_{i=1}^{n} RTT_i$$

Since it's persistent HTTP, Need 1 RTT_0 for many parallel connections, 2 RTT_0 to set up the HTTP page plus the IP address

Problem 2

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 2 in this box

2. Web servers identify users when you do the internet users by HTTP authentications or by cookies. Even though, HTTP authentications is less comfortable than cookies, some e-commerce still has it. The details on those will discuss below.
Via HTTP authentications: Authentication basically just a prompt where user inputs the user ID and the password. When a client request a message with no special header lines, the server will respond with an empty body and 401 authorization request code. Where users need to input their ID and password. Once it's granted, then user can access it. But, users need to enter and reenter the ID and password for a couple time.
Via cookies: When the first time a user access the e-commerce (e-bay site), site creates unique ID, and entry in backend database for that particular users or commonly called as a cookie number. This cookie file kept in the user's host, and managed by user's browser. When users try to revisit the website, the client will include a cookie header, which is the identification number for that server. As a result, users do not have to re-enter the ID and password for several times. In addition, cookies can save some user preferences in order to give more convenient shopping experience.

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...

Write your solution to Problem 3 in this box

A. Since there are 5 TCP packets, it means 5h bits, each HTML file is 2p bits, so it means that it has 10p bits. The header files contains 3h bits. Thus, total bit now is 8h + 10p. All other 4 files will face transmission delay, and all this time also add the propagation delay (d seconds). Thus, the total time is
((8h + 10p) / b) + (4 * (h/b)) + d seconds = ((12h + 10p) / b) + d seconds

B. Since, in here, it has 3h bits per connection, then it means
((15h + 10p) / b) + d seconds.

C. Based on the calculation above, if we neglect the propagation delay, which is about the same, I would assume that the pipelining gives a smaller latency, as ((12h + 10p)/b) < ((15h + 10p)/b). The reason is because in non-persistent, there is an extra time to set up the connection.

Problem 4

What is the difference between MAIL FROM: in SMTP and From: in the mail message itself?

Write your solution to Problem 4 in this box

4. The MAIL FROM in SMTP is just a verification to allow the sender of the message to be recognized by the server. On the other hand, FROM in the mail itself is just a header of the message which allows the reader of the message knows who is the sender of the mail. In other words, MAIL FROM in SMTP is contains the name of the email and the server address, while FROM in the message itself more like a human readable name, for example it translate reidanis@cs.net into rei danis.

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

5

a. We can use DNS resolver to determine if an external web site was likely accessed from a computer in your department a couple of seconds ago. There is 2 process to found an external website that was accessed. If we don't know exactly what website that was accessed, open a terminal and use command "ipconfig/displaydns". It will shows the DNS records that saved in the cache of DNS resolver. Another process is when you know what kind of website that was accessed, we can use command "dig". For example, "dig my.ucla.edu". If the query time listed is 0 then the domain name was just accessed. In contrast, if it is not zero, means it is not available in DNS resolver, or it did not just accessed.

b. The concept is similar with a regular cache. If it is accessed frequently, it means that it's there. Thus, if we have access to the caches in local DNS servers, then we will be able to know which web server that is requested more by the client. By taking a note, or taking a snapshot frequently of the DNS caches in local DNS servers, we will know that which web servers that are most popular in my department.