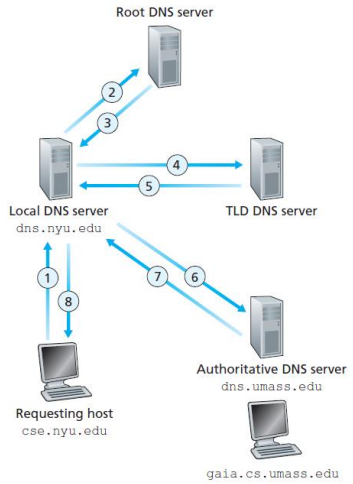
ENSC-371 – Summer 2017 Assignment#2: Due Date 13 June 2017

Note: Provide Detailed solutions and not just final answers. Final Answers only will get a mark of zero.

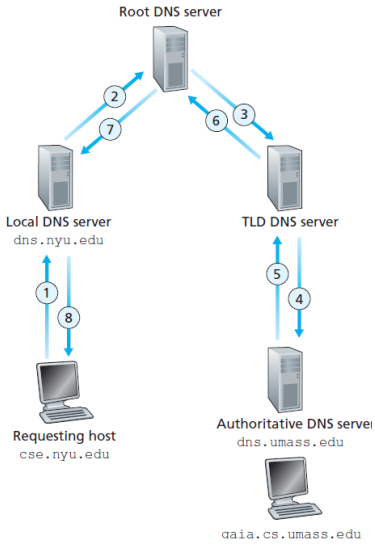
Problem-1:



Assume that the RTT between a client and the local DNS server is TT_l, while the RTT between the local DNS server and other DNS servers is RTT_r. Assume that no DNS server performs caching.

a. What is the total response time for the scenario illustrated in top figure on the right?
b. What is the total response time for the scenario illustrated in bottom figure on the right?
c. Assume now that the DNS record for the requested name is cached at the local DNS server. What is the total response time for the two scenarios?

Now suppose the HTML file references eight very small objects on the same server. Neglecting transmission times, how much time elapses with the following conditions:



d. Non-persistent HTTP with no parallel TCP connections?
e. Non-persistent HTTP with the browser configured for 5 parallel connections?
f. Persistent HTTP?

Solution:

1.

a) 2 TT_l + 3 RTT_r

(1 TT_l for connection, 1 TT_l + 3 RTT_r for request)

b) 2 TT_l + 3 RTT_r

(1 TT_l for connection, 1 TT_l + 3 RTT_r for request)

c) 2 TT_l for both figures

(1 TT_l for connection, 1 TT_l for request)

d) 18 TT_l + 3 RTT_r

(18 TT_l for connection + request for HTML file + 8 references, 3 RTT_r for the non-local DNS servers)

d) 6 TT_l + 3 RTT_r

(1 TT_l for connection, 1 TT_l for HTML request, 2 TT_l for 5+3 parallel connections, 2 TT_l for 8 object requests, 3 RTT_r for the non-local DNS servers)

e) 3 TT_l + 3 RTT_r

(1 TT_l for connection, 1 TT_l for HTML request, 1 TT_l for 8 object requests, 3 RTT_r for the non-local DNS servers)

Note: Provide Detailed solutions and not just final answers. Final Answers only will get a mark of zero.

Problem-2:

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 $d_i = 2$ Mbps and an upload rate of u. For $N = 10$, 100, and 1,000 and $u = 300$ Kbps, 700 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.

Solution:

- For calculating the minimum distribution time for client-server distribution, we use the following formula: $D_{cs} = max\{NF/u_s, F/d_{min}\}$

- Similarly, for calculating the minimum distribution time for P2P distribution, we use the following formula: $D_{P2P} = max\{F/u_s, F/d_{min}, NF/(u_s + \sum_{i=1}^{N} u_i)\}$

  Where, F = 15 Gbits = 15 * 1024 Mbits; u_s = 30 Mbps; $d_{min} = d_i$ = 2 Mbps

  Note, 300Kbps = 300/1024 Mbps.

  Client Server

		N		
		10	**100**	**1000**
	300 Kbps	7680	51200	512000
u	**700 Kbps**	7680	51200	512000
	2 Mbps	7680	51200	512000

  Peer to Peer

		N		
		10	**100**	**1000**
	300 Kbps	7680	25904	47559
u	**700 Kbps**	7680	15616	21525
	2 Mbps	7680	7680	7680

Note: Provide Detailed solutions and not just final answers. Final Answers only will get a mark of zero.

Problem-3:

Part-a:

Can you configure your browser to open multiple simultaneous connections to a Web site? What are the advantages and disadvantages of having a large number of simultaneous TCP connections?

Part-b:

We have seen that Internet TCP sockets treat the data being sent as a byte stream but UDP sockets recognize message boundaries. What are one advantage and one disadvantage of byte-oriented API versus having the API explicitly recognize and preserve application-defined message boundaries?

Solution:

Part-a:

Yes, you can configure many browsers to open multiple simultaneous connections to a Web site. The advantage is that you will you potentially download the file faster. The disadvantage is that you may be hogging the bandwidth, thereby significantly slowing down the downloads of other users who are sharing the same physical links.

Part-b:

For an application such as remote login (telnet and ssh), a byte-stream oriented protocol is very natural since there is no notion of message boundaries in the application. When a user types a character, we simply drop the character into the TCP connection.

In other applications, we may be sending a series of messages that have inherent boundaries between them. For example, when one SMTP mail server sends another SMTP mail server several email messages back to back. Since TCP does not have a mechanism to indicate the boundaries, the application must add the indications itself, so that receiving side of the application can distinguish one message from the next. If each message were instead put into a distinct UDP segment, the receiving end would be able to distinguish the various messages without any indications added by the sending side of the application.

Note: Provide Detailed solutions and not just final answers. Final Answers only will get a mark of zero.

Problem-4:

Suppose Bob (not your instructor) joins a Bit-Torrent 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?

Solution:

a. Yes. His first claim is possible, as long as there are enough peers staying in the swarm for a long enough time. Bob can always receive data through optimistic un-choking by other peers.

b. His second claim is also true. He can run a client on each host, let each client "free-ride," and combine the collected chunks from the different hosts into a single file. He can even write a small scheduling program to make the different hosts ask for different chunks of the file. This is actually a kind of Sybil attack in P2P networks.

Note: Provide Detailed solutions and not just final answers. Final Answers only will get a mark of zero.

Problem-5:

Suppose you installed (*don't actually install – You can if you want but not required*) and compiled Python programs TCPClient and UDPClient on one host and TCPServer and UDPServer on another host.

a. Suppose you run TCPClient before you run TCPServer. What happens? and Why?
b. Suppose you run UDPClient before you run UDPServer. What happens? and Why?
c. What happens if you use different port numbers for the client and server sides?

Solution:

a) If you run TCPClient first, then the client will attempt to make a TCP connection with a non-existent server process. A TCP connection will not be made.

b) UDPClient doesn't establish a TCP connection with the server. Thus, everything should work fine if you first run UDPClient, then run UDPServer, and then type some input into the keyboard.

c) If you use different port numbers, then the client will attempt to establish a TCP connection with the wrong process or a non-existent process. Errors will occur.