New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Course Title: Data Communication & Networks **Course Number:** CSCI-GA.2662-001
Instructor: Jean-Claude Franchitti **Session:** 2

Assignment #2 Solutions

1. **Question 1: Textbook chapter 2 problem 7:**

   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 of $RTT_1, \ldots, RTT_n$. Further suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Let RTT_0 denote the RTT between the local host and the server containing the object. Assuming zero transmission time of the object, how much time elapses from when the client clicks on the link until the client receives the object?

   Answer:

   The total amount of time to get the IP address is

   $$RTT_1 + RTT_2 + \cdots + RTT_n .$$

   Once the IP address is known, RTT_O elapses to set up the TCP connection and another RTT_O elapses to request and receive the small object. The total response time is

   $$2RTT_o + RTT_1 + RTT_2 + \cdots + RTT_n$$

2. **Question 2: Textbook chapter 2 problem 8:**

   Referring to Problem P7, suppose the HTML file references eight very small objects on the same server. Neglecting transmission times, how much time elapses 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?**

Answer:

a.

$$RTT_1 + \cdots + RTT_n + 2RTT_o + 8 \cdot 2RTT_o$$
$$= 18RTT_o + RTT_1 + \cdots + RTT_n .$$

b.

$$RTT_1 + \cdots + RTT_n + 2RTT_o + 2 \cdot 2RTT_o$$
$$= 6RTT_o + RTT_1 + \cdots + RTT_n$$

c.

$$RTT_1 + \cdots + RTT_n + 2RTT_o + RTT_o$$
$$= 3RTT_o + RTT_1 + \cdots + RTT_n .$$

3. Question 3: Textbook chapter 2 problem 10:

Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that *N* parallel connections each get *1/N* of the link bandwidth.

Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Answer:

Note that each downloaded object can be completely put into one data packet. Let Tp denote the one-way propagation delay between the client and the server.

First consider parallel downloads via non-persistent connections. Parallel download would allow 10 connections share the 150 bits/sec bandwidth, thus each gets just 15 bits/sec. Thus, the total time needed to receive all objects is given by:

(200/150+Tp + 200/150 +Tp + 200/150+Tp + 100,000/150+ Tp)
+ (200/(150/10)+Tp + 200/(150/10) +Tp + 200/(150/10)+Tp + 100,000/(150/10)+ Tp)
= 7377 + 8*Tp (seconds)

Then consider persistent HTTP connection. The total time needed is give by:
(200/150+Tp + 200/150 +Tp + 200/150+Tp + 100,000/150+ Tp)
+ 10*(200/150+Tp + 100,000/150+ Tp)
=7351 + 24*Tp (seconds)

Assume the speed of light is $300*10^6$ m/sec, then Tp=$10/(300*10^6)$=0.03 microsec. Tp is negligible compared with transmission delay.

Thus, we see that the persistent HTTP does not have significant gain (less than 1 percent) over the non-persistent case with parallel download.

4. **<u>Question 4</u>: Textbook chapter 2 problem 19:**

   In this problem, we use the useful *dig* tool available on Unix and Linux hosts to explore the hierarchy of DNS servers. Recall that in Figure 2.21, a DNS server higher in the DNS hierarchy delegates a DNS query to a DNS server lower in the hierarchy, by sending back to the DNS client the name of that lower-level DNS server. First read the man page for *dig*, and then answer the following questions.

   a. **Starting with a root DNS server (from one of the root servers [a-m].root-servers.net), initiate a sequence of queries for the IP address for your department's Web server by using *dig*. Show the list of names of DNS servers in the delegation chain in answering your query.**

   b. **Repeat part a) for several popular Web sites, such as google.com, yahoo.com, or amazon.com**

<u>Answer:</u>

a.

The following delegation chain is used for gaia.cs.umass.edu
a.root-servers.net
E.GTLD-SERVERS.NET
ns1.umass.edu(authoritative)

First command: dig +norecurse @a.root-servers.net any gaia.cs.umass.edu

;; AUTHORITY SECTION:

edu.	172800	IN	NS	E.GTLD-SERVERS.NET.
edu.	172800	IN	NS	A.GTLD-SERVERS.NET.
edu.	172800	IN	NS	G3.NSTLD.COM.
edu.	172800	IN	NS	D.GTLD-SERVERS.NET.
edu.	172800	IN	NS	H3.NSTLD.COM.
edu.	172800	IN	NS	L3.NSTLD.COM.
edu.	172800	IN	NS	M3.NSTLD.COM.
edu.	172800	IN	NS	C.GTLD-SERVERS.NET.

Among all returned edu DNS servers, we send a query to the first one.
dig +norecurse @E.GTLD-SERVERS.NET any gaia.cs.umass.edu

umass.edu.	172800	IN	NS	ns1.umass.edu.
umass.edu.	172800	IN	NS	ns2.umass.edu.
umass.edu.	172800	IN	NS	ns3.umass.edu.

Among all three returned authoritative DNS servers, we send a query to the first one.
dig +norecurse @ns1.umass.edu any gaia.cs.umass.edu

gaia.cs.umass.edu. 21600 IN A 128.119.245.12

b. The answer for google.com could be:

a.root-servers.net
E.GTLD-SERVERS.NET
ns1.google.com(authoritative)

5. **<u>Question 5</u>: Textbook chapter 2 problem 23:**

Consider distributing a file of F bits to N peers using a client-server architecture. Assume a fluid model where the server can simultaneously transmit to multiple peers, transmitting to each peer at different rates, as long as the combined rate does not exceed u_s.

a. **Suppose that $u_s/N \leq d_{min}$. Specify a distribution scheme that has a distribution time of NF/u_s.**

b. **Suppose that $u_s/N \geq d_{min}$. Specify a distribution scheme that has a distribution time of F/d_{min}.**

c. **Conclude that the minimum distribution time is in general given by max{NF/u_s, F/d_{min}}.**

<u>Answer:</u>

a. Consider a distribution scheme in which the server sends the file to each client, in parallel, at a rate of a rate of u_s/N. Note that this rate is less than each of the client's download rate, since by assumption $u_s/N \leq d_{\min}$. Thus each client can also receive at rate u_s/N. Since each client receives at rate u_s/N, the time for each client to receive the entire file is F/(u_s/N) = NF/ u_s. Since all the clients receive the file in NF/ u_s, the overall distribution time is also NF/ u_s.

b. Consider a distribution scheme in which the server sends the file to each client, in parallel, at a rate of $d_{\min}$. Note that the aggregate rate, $N\, d_{\min}$, is less than the server's link rate u_s, since by assumption $u_s/N \geq d_{\min}$. Since each client receives at rate $d_{\min}$, the time for each client to receive the entire file is F/ $d_{\min}$. Since all the clients receive the file in this time, the overall distribution time is also F/ $d_{\min}$.

c. From Section 2.6 we know that

$D_{CS} \geq \max\ \{NF/u_s, F/d_{\min}\}$ (Equation 1)

Suppose that $u_s/N \leq d_{\min}$. Then from Equation 1 we have $D_{CS} \geq NF/u_s$. But from (a) we have $D_{CS} \leq NF/u_s$. Combining these two gives:

$D_{CS} = NF/u_s$ when $u_s/N \leq d_{\min}$. (Equation 2)

We can similarly show that:

$D_{CS} = F/d_{\min}$ when $u_s/N \geq d_{\min}$ (Equation 3).

Combining Equation 2 and Equation 3 gives the desired result.

6. <u>Question 6</u>: Textbook chapter 2 problem 29:

Because an integer in [0, 2^n - 1] can be expressed as an *n*-bit binary number in a DHT, each key can be expressed as $k = (k_0, k_1,...,k_{n-1})$, and each peer identifier can be expressed $p = (p_0, p_1,...,p_{n-1})$. Let's now define the XOR distance between a key *k* and peer *p* as

$$d(k,p) = \sum_{j=0}^{n-1} |k_j - p_j| 2^j$$

Describe how this metric can be used to assign (key, value) pairs to peers. (To learn about how to build an efficient DHT using this natural metric, see [Maymounkov 2002] in which the Kademlia DHT is described.)