Solutions to Homework #2, EE450, Fall 2010

Chapter 1, # 19, 20 and 22

Problem 19

Throughput = $min\{R_s, R_c, R/M\}$

Problem 20

If only use one path, the max throughput is given by
$\max\{\min\{R_1^1, R_2^1, \ldots, R_N^1\}, \min\{R_1^2, R_2^2, \ldots, R_N^2\}, \ldots, \min\{R_1^M, R_2^M, \ldots, R_N^M\}\}$.

If use all paths, the max throughput is given by $\sum_{k=1}^{M} \min\{R_1^k, R_2^k, \ldots, R_N^k\}$.

Problem 22

Lets call the first packet A and call the second packet B.
a). If the bottleneck link is the first link, then, packet B is queued at the first link waiting for the transmission of packet A. So, the packet inter-arrival time at the destination is simply L/R_s.

b). If the second link is the bottleneck link and since both packets are sent back to back, it must be true that the second packet arrives at the input queue of the second link before the second link finishes the transmission of the first packet. That is,
$L/R_s + L/R_s + d_{prop} < L/R_s + d_{prop} + L/R_c$ (1)
The left hand side of the above inequality represents the time needed by the second packet to *arrive at* the input queue of the second link (the second link has not started transmitting the second packet yet).
The right hand side represents the time needed by the first packet to finish its transmission onto the second link.

We know that (1) is possible as $R_c < R_s$. And it is clear that (1) shows that the second packet must have queuing delay at the input queue of the second link.
If we send the second packet T seconds later, then we can ensure the there is no queuing delay for the second packet at the second link if we have,
$L/R_s + L/R_s + d_{prop} + T >= L/R_s + d_{prop} + L/R_c$
Thus, T must be $L/R_c - L/R_s$.

Chapter 2, #7, 8 and 10

Problem 7

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

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

Problem 8

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)

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

Problem 10

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.

Chapter 5, #33

Problem 33

The time required to fill $L \cdot 8$ bits is

$$\frac{L \cdot 8}{128 \times 10^3} \sec = \frac{L}{16} m\sec.$$

b) For $L = 1{,}500$, the packetization delay is

$$\frac{1500}{16} m\sec = 93.75 m\sec.$$

For $L = 50$, the packetization delay is

$$\frac{50}{16} m\sec = 3.125 m\sec.$$

c)

$$\text{Store-and-forward delay} = \frac{L \cdot 8 + 40}{R}$$

For $L = 1{,}500$, the delay is

$$\frac{1500 \cdot 8 + 40}{622 \times 10^{6}} \sec \approx 19.4 \mu \sec$$

For $L = 50$, store-and-forward delay $< 1 \mu \sec$.

d) Store-and-forward delay is small for both cases for typical link speeds. However, packetization delay for $L = 1500$ is too large for real-time voice applications.