EECE 7374 – Fall 2023
Homework #1 Solution

Chapter 1:

1) **Circuit switching**
   a) 4 users, each getting a dedicated 2 Mbps.
   b) If 4 users transmit simultaneously, the input rate is 8 Mbps. Since the link is also 8 Mbps, there will be no queuing delay.
   c) Since a user is transmitting 25% of the time, the probability that a user is transmitting at any instant is 0.25.
   d) Assuming that all 5 users are transmitting independently, the probability that all of them are transmitting simultaneously is $(0.25)^5$. Since the queue only grows when all 5 users are transmitting, the fraction of time during which the queue grows is $(0.25)^5$.

2) **Transmission delay and propagation delay**
   a) The delay is the sum of the transmission delay and the propagation delay.
   Transmission delay d_{tran} = (1000*8 bits)/(8000000 b/sec) = 1 msec.
   Propagation delay d_{prop} = $(2500*10^3$ m)/($2.5*10^8$ m/sec) = 10 msec.
   Total delay d_{tot} = 11 msec.
   b) The bit is just leaving Host A.
   c) The first bit is in the link and has not reached Host B.
   d) The first bit has reached Host B.
   e) 40 terabytes = 40 * 10^{12} * 8 bits. So, using the dedicated link will take 40 * 10^{12} * 8 / (100 *10^6) =3200000 seconds = 37 days. But with FedEx overnight delivery, you can guarantee the data arrives in one day.

3) **Bandwidth-delay product**
   a) R* d_{prop} = $3*10^6$ b/sec * ($30000*10^3$ m)/($2.5*10^8$ m/sec) = 360,000 bits.
   b) 360,000 bits.
   c) The width of a bit = length of link / bandwidth-delay product = ($30000*10^3$ m)/(360,000 bits) = 83.333 m.
   d) R* d_{prop} = $3*10^9$ b/sec * ($30000*10^3$ m)/($2.5*10^8$ m/sec) = 360,000,000 bits.
   min(bandwidth delay product, message size) = 1,200,000 bits.
   Width of a bit = 0.0833 m.

4) **Store-and-forward**
   a) Host A requires L/R_1 to transmit the packet onto the first link; the packet propagates over the first link in d_1/s_1; the packet switch adds a processing delay of d_{proc}; after receiving the entire packet, the packet switch connecting the first and the second link requires L/R_2 to transmit the packet onto the second link; the packet propagates over the second link in d_2/s_2. Similarly, we can find the delay caused by the second switch and the third link: L/R_3, d_{proc}, and d_3/s_3. Adding these five delays gives:
   $$d_{end\text{-}to\text{-}end} = L/R_1 + L/R_2 + L/R_3 + d_1/s_1 + d_2/s_2 + d_3/s_3 + d_{proc} + d_{proc}$$
   b) Because bits are immediately transmitted, the packet switch does not introduce any

transmission delay. Thus,

$$d_{end\text{-}end} = L/R + d_1/s_1 + d_2/s_2 + d_3/s_3$$

Chapter 2:

5) **HTTP performance**

First, it takes 1 msec to send 100 Kbits over a 100 Mbps link.

a) The delays associated with this scenario are:
   - 300 msec (RTT) to set up the TCP connection that will be used to request the base file
   - 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client. Total 301 msec.
   - 300 msec (RTT) to set up a TCP connection that will be used to request the first image.
   - 150 msec (one way delay) to send the GET message for the first image, and have the message propagate to the server, plus 1 msec to transmit the first image, plus 150 msec for the image to propagate back to the client. Total 301 msec.

   The last two steps are repeated 4 more times for the 4 remaining images. The total response time is 6*601 msec = 3.606 sec.
b) Here we have:
   - 300 msec (RTT) to set up the TCP connection that will be used to request the base file.
   - 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client. Total 301 msec.
   - The client now sets up 5 parallel TCP connections. 300 msec (RTT) is needed to set up the 5 TCP connections since they are set up in parallel.
   - 150 msec (one way delay) to send the GET messages in parallel for the 5 images and have the GET messages propagate to the server. It will take the server 5 msec to transmit the 5 images, plus 150 msec for the last image to propagate back to the client (total 305 msec).

   Total response time = (300 + 301 + 300 + 305) msec = 1.206 sec.
c) Here we have:
   - 300 msec (RTT) to set up the TCP connection that will be used to request the base file and the 5 images.
   - 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client. Total 301 msec.
   - 150 msec (one way delay) to send the GET message for the first image, and have the message propagate to the server, plus 1 msec to transmit the first image, plus 150 msec for the image to propagate back to the client. Total 301 msec.

   The last step is repeated 4 more times for the 4 remaining images. The total response time is (300 + 6*301) msec = 2.106 sec.
d) Here we have:

- 300 msec (RTT) to set up the TCP connection that will be used to request the base file and the 5 images.
- 150 msec (one way delay) to send the GET message for the base file, and have the message propagate to the server, plus 1 msec to transmit the base file, plus 150 msec for the base file to propagate back to the client. Total 301 msec.
- 150 msec (one way delay) to send the GET messages in parallel for the 5 images and have the GET messages propagate to the server. It will take the server 5 msec to transmit the 5 images, plus 150 msec for the last image to propagate back to the client (total 305 msec).

Total response time = (300 + 301 + 305) msec = 906 msec.

6) **BitTorrent**

a) Yes, as long as there are enough peers staying in the swarm for a long enough time. Bob can always receive data through optimistic unchoking by other peers.

b) He can run a client on each machine and let each client do "free-riding", and combine those collected chunks from different machines into a single file. He can even write a small scheduling program to let different machines only asking for different chunks of the file.

Bonus problem

Here 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 non-persistent HTTP with no parallel connections. The total time needed is given by:

$11*(200/150 + T\text{p} + 200/150 + T\text{p} + 200/150 + T\text{p} + 100{,}000/150 + T\text{p})$
$= (7377 + 44*\text{Tp})$ seconds

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

$(200/150 + T\text{p} + 200/150 + T\text{p} + 200/150 + T\text{p} + 100{,}000/150 + T\text{p})$
$+ (200/(150/10) + T\text{p} + 200/(150/10) + T\text{p} + 200/(150/10) + T\text{p} + 100{,}000/(150/10)+ T\text{p})$
$= (7377 + 8*T\text{p})$ seconds

Finally, consider a persistent HTTP connection without pipelining. The total time needed is given by:

$(200/150 + T\text{p} + 200/150 + T\text{p} + 200/150 + T\text{p} + 100{,}000/150 + T\text{p})$
$+ 10*(200/150 + T\text{p} + 100{,}000/150 + T\text{p})$
$= (7351 + 24*T\text{p})$ seconds

a) Assuming the speed of light is $3*10^8$ m/sec, then $Tp = 10/(3*10^8) = 0.03$ usec. Tp is therefore negligible compared with transmission delay and hence, parallel downloads do not help significantly here – it saves only $44*Tp - 8*Tp = 36*Tp = 1.08$ usec.

b) We also see that persistent HTTP is not significantly faster (less than 1 percent) than the non-persistent case with parallel download.