I have read and understood the course academic integrity policy.

Chapter 1:

1) **Circuit/packet switching** – Suppose users share an 8 Mbps link. Also suppose each user requires a constant bitrate of 2 Mbps when transmitting, but each user transmits only 25% of the time.

a) When circuit switching is used, how many users can be supported? **(4 points)**

Num = total link / personal link

Num = 8 / 2 =4

Answer is 4 users

b) For the remainder of the question, suppose packet switching is used. Will there be any queuing delay before the link if 4 or fewer users transmit simultaneously? **(4 points)**

No, the total bandwidth is not less than the sum of all user bandwidth.

c) Find the probability that a given user is transmitting. **(4 points)**

The probability of given user is 0.25, according to "each user transmits only 25% of the time."

d) Suppose there are 5 users. Find the probability that at any given time, all 5 users are transmitting simultaneously. **(2.5 points)** Find the fraction of time during which the queue grows. **(2.5 points)**

Probability = $0.25^5 = 9.765625*10^{-4}$

Only when five user transmit data simultaneously, the queue will grow,

The answer is $9.765625*10^{-4}$.

2) **Transmission delay and propagation delay** – Consider two hosts, A and B, directly connected by a link of 2500 Km, propagation speed $2.5*10^8$ m/s, and transmission rate 8 Mbps. Assume zero queuing delay.

a) How long does it take to move a packet of length 1000 bytes from one A to B? **(4 points)**

Propagation delay = 2.5 * 10^6 m / 2.5*10^8(m/s) = 0.01 s

Transmission delay = 1000*8 / (8 *10^6)= $1*10^{-3}$ s

Total time= Propagation delay + Transmission delay = $1.1*10^{-2}$ s

b) Assume the transmission time of the packet is d_{trans} and the propagation delay d_{prop}. Suppose that host A begins to transmit a packet at time $t=0$. *At time* $t= d_{trans}$ where is the last bit of the packet? **(3 points)**

last bit just transmitted from A,   still is in the link

c) Suppose d_{prop} is greater than d_{trans}. At time $t = d_{trans}$, where is the first bit of the packet? **(3 points)**

The first bit is still in the link.

d) Suppose d_{prop} is less than d_{trans}. At time $t = d_{trans}$, where is the first bit of the packet? **(3 points)**

The first bit is arrived at receiver B end, probably stored in cache.

e) You want to urgently deliver 40 terabytes of data from Buffalo to San Francisco. You have available a 100 Mbps dedicated link for the data transfer. Would you transmit the data over this link or use FedEx overnight delivery instead? Explain (write down any assumptions). **(4 points)**

The distance between two city is 2659 miles which is 4254 km according to Google map. So, the delay（we just assume there is a wire connect two city directly, the real delay may be longer） will be:

Propagation delay = 4245 * 10^3 m / 2.5*10^8(m/s) = 0.01698 s

Transmission delay = 40*10^12 *8/ (100 *10^6)= 3.2*10^6 s =888.88 hours

Total time= Propagation delay + Transmission delay ＞ 888 hours

The total time on using internet to transmit data is much longer than overnight delivery, so, the FedEX could be a better choice for urgent data.

(but, consider FedEX has a high rate of losing package, maybe higher than losing packet on internet transmission, this is not a very definitive answer)

3) **Bandwidth-delay product** – Suppose two hosts, A and B, are separated by 30,000 Km and are connected by a direct link of R=3 Mbps. Suppose the propagation speed over the link is $2.5*10^8$ m/s.

a) Calculate the bandwidth-delay product R* d_{prop}. **(3 points)**

d_{prop} = 3 * 10^7 m / 2.5*10^8(m/s) = 0.12 s

R* d_{prop} = 3*0.12 = 0.36 Mb

b) Consider sending a file of 1,200,000 bits from host A to host B. Suppose the file is sent continuously as a large message. What is the maximum number of bits that will be in the link at any time? **(3 points)**

d_{prop} = 3 * 10^7 m / 2.5*10^8(m/s) = 0.12 s

the bandwithth-delay product is less than the file size, so The maximum number of bits is equal to R* d_{prop}

MaxNum = R* d_{prop} = 3*0.12 = 0.36 Mb

c) What is the width (in meters) of a bit in the link? **(3 points)**

Width =len /(R* d_{prop})= (3*10^7)m / (3.6*10^5 b) = 83.33 m/b

Ans is 83.33 meter one bit.

d) Repeat (a), (b) and (c) but now with a link of 3 Gbps. **(9 points)**

a):

d_{prop} = 3 * 10^7 m / 2.5*10^8(m/s) = 0.12 s

R* d_{prop} = 3*0.12 = 0.36 Gb

b):

d_{prop} = 3 * 10^7 m / 2.5*10^8(m/s) = 0.12 s

the bandwithth-delay product is greater than the file size, so The maximum number of bits is equal to file size

MaxNum = Size of file = 1.2Mb

c):

Width = len /(R* d_{prop})= (3*10^7)m / (3.6*10^8 b) = 0.083 m/b

Ans is 0.083 meter

4) **Store-and-forward** – Consider a packet of length L which begins at Host A and travels over three links to Host B. These three links are connected by two store-and-forward packet switches. Let d_i, s_i, R_i denote the length, propagation speed, and the transmission rate of link i, for i = 1, 2, 3. The packet switch delays each packet by d_{proc}.

a) Assuming no queuing delays, in terms of d_i, s_i, R_i (i = 1, 2, 3), and L, what is the total end-to-end delay for the packet? **(5 points)**

Because of there is no queuing delays, the answer is:

$$d_{total} = 2 * d_{proc} + \sum_{i=1}^{i=3} L/R_i + d_i/s_i$$

b) Now suppose $R_1 = R_2 = R_3 = R$ and $d_{proc} = 0$. Further suppose the packet switches do not store-and-forward but instead immediately transmit each bit they receive before waiting for the entire packet to arrive. What is the end-to-end delay? **(5 points)**

$$d_{total} = 3 * \frac{L}{R} + \sum_{i=1}^{i=3} d_i/s_i$$

Chapter 2:

5) **HTTP performance** – Here, we consider the performance of HTTP, comparing non-persistent HTTP with persistent HTTP. Suppose the page your browser wants to download is 100 Kbits long, and contains 5 embedded images, each of which is 100 Kbits long. The page and the 5 images are stored on the same server, which has 300 msec roundtrip time (RTT) from your browser. We will abstract the network path between your browser and the Web server as a 100 Mbps link. You can assume that the time it takes to transmit a GET message into the link is zero, but you should account for the time it takes to transmit the base file and the embedded objects into the "link". **This means that the server-to-client "link" has both a 150 msec one-way propagation delay, as well as a transmission delay associated with it, similar to the examples in class.** In your answers, be sure to account for the time needed to set up a TCP connection (1 RTT).

a) Assuming non-persistent HTTP (and assuming no parallel connections are open between the browser and the server), how long is the response time – the time from when the user requests the URL to the time when the page and its embedded images are displayed? Be sure to describe the various components that contribute to this delay. **(7 points)**

Response time = 2RTT+ Transmission time for page + 5*(2RTT + Transmission time for image)

Response time = 2* 0.3s + 100 k / 1*10^5 kbps + 10* 0.3s + 5*(100 k / 1*10^5 kbps)

= 3.6+ 6*10^-3

= 3.606 s

b) Again, assume non-persistent HTTP, but now assume that the browser can open as many parallel TCP connections to the server as it wants. What is the response time in this case? **(7 points)**

Response time = 2RTT+ Transmission time for page + 2RTT + 5* Transmission time for image

Response time = 2* 0.3s + 100 k / 1*10^5 kbps + 2* 0.3s + 5*(100 k / 1*10^5 kbps)

= 1.2+ 6*10^-3

= 1.206 s

c) Now assume persistent HTTP (HTTP 1.1). What is the response time assuming no pipelining? **(7 points)**

Response time = 2RTT+ Transmission time for page + 5RTT + 5* Transmission time for image

Response time = 2* 0.3s + 100 k / 1*10^5 kbps + 5* 0.3s + 5*(100 k / 1*10^5 kbps)

= 2.1+ 6*10^-3

= 2.106 s

d) Now assume persistent HTTP with pipelining is used. What is the response time? **(7 points)**

Response time = 2RTT+ Transmission time for page + RTT + 5* Transmission time for image

Response time = 2* 0.3s + 100 k / 1*10^5 kbps + 0.3s + 5*(100 k / 1*10^5 kbps)

= 0.9+ 6*10^-3

= 0.906 s

6) **BitTorrent** – Suppose Bob joins a BitTorrent torrent, but he does not want to upload any data to other peers (also called free-riding).

a) Can Bob receive a complete copy of the file that is shared by the swarm? Why or why not? **(5 points)**

Yes, but slow. Because, BitTorrent is using tit-for-tat structure which is a cooperative model, user should not only down load data, but also upload thing. If Bob blocks his upload transmission, he's down lowed rate will be limited, but not banished, "optimistically unchoke" will finally help him get all chunk.

b) Bob claims that he can 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? **(5 points)**

Bob can use different computer with different IP address join one same torrent. He can use each IP address set a connection with other peer in swarm. He can now down load data in paralleled, because each computer(peer) can download different part of one data file. Although he's download rate is still be limited by tit-for-tat, by parallel downloading, the total rate of download will be faster. But it needs special algorithm to assemble data segment again.

Bonus problem (Difficult) – 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., ACKs, hand-shaking, HTTP GET requests) are 200 bits long. Assume that N parallel connections each gets $1/N$ of the link bandwidth. Now consider the HTTP protocol and assume each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender.

a) Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? **(10 points)**

b) Now consider persistent HTTP without pipelining. Do you expect significant gains over the non-persistent case? Justify and explain your answer. **(10 points)**

State any assumptions you have to make.

a) It doesn't make sense.

we assume propagation delay can be ignored, because distance is too short

For the non-parallel of non-persistent situation

RTT = 2*200b/(150bps)

Response time= 11*(2*RTT+ Transmission time)

=11*(2*2*200b/(150bps)+1*10^5b/(150bps)

= 7392s

For the parallel of non-persistent situation

RTT of Base file is not changed.

RTT of rest file will be RTT*10

Total response time = 2*RTT_base + 2*RTT_res + response time of base + response time of rest

=2*2*200b/(150bps)+10*2*2*200b/(150bps)+1*10^5b/(150bps) + 10 * 1*10^5b/(150bps)

=11*(2*2*200b/(150bps)+1*10^5b/(150bps)

=7392s

In calculation the total time is same, so it doesn't make sense. Even use parallel of non-persistent HTTP will take more process of establishing and tearing down connection.

b):

in this case, using pipelining will save ten times RTT process. It takes 27s. but the total response time is 7392. This saving can not be consider as significant gains.

However, if the file size is small or the transmission rate is higher, this saving will be significant.