SIMON FRASER UNIVERSITY

School of Computing Science

CMPT 371 - Data Communications and Networking

Name: _____Instructor Solution Key_____ **Student Number**: **_____**

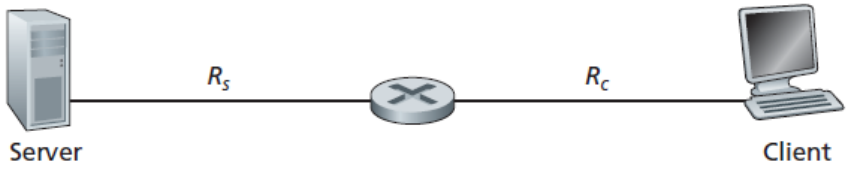
Instructor: Bob Gill, P.Eng, FEC Maximum Marks: 100 Date: Oct 23 2017

Max Time: 120 minutes

1. This is a closed book exam.
2. Put your name and student number on the exam books NOW!
3. **You have 120 minutes** to complete the exam. **Be a smart exam taker** - if you get stuck on one problem go on to another problem. Also, don't waste your time giving irrelevant (or not requested) details.
4. The total number of points for each question is given in parenthesis and there are 100 points in total.
5. Show all your work. Partial credit is possible for an answer, but only if you show the intermediate steps in obtaining the answer.
6. You may write at the back of the page if you run out of space on front.

Good Luck!!!!!

1. [12] Consider Figure below and assume that we know the bottleneck link along the path from the server to the client is the first link with rate $R_s \frac{bits}{sec}$. Suppose we send a pair of packets (say A and B) back to back from the server to the client, and there is no other traffic on this path. Assume each packet of size L bits, and both links have the same propagation delay d_{prop}. Note that R_c refers to client data rate.
   a. [2] What is the packet inter-arrival time at the destination? That is, how much time elapses from when the last bit of the first packet arrives until the last bit of the second packet arrives?
   b. [5] Now assume that the second link is the bottleneck link (i.e., $R_c < R_s$). Is it possible that the second packet queues at the input queue of the second link? Explain.
   c. [5] Now suppose that the server sends the second packet T seconds after sending the first packet. How large must T be to ensure no queuing before the second link? Explain.



Solution:

Let's 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 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,

$L/R_s + L/R_s + d_{prop} < L/R_s + d_{prop} + L/R_c$

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.

c) If we send the second packet T seconds later, we will ensure that 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, the minimum value of T is $L/R_c - L/R_s$.

2. [8] Calculate the total time required to transfer a 1,000-KB file in the following cases, assuming an RTT of 100 msec, a packet size of 1-KB data, and an initial 2×RTT of "handshaking" before data is sent.
    a. [4] The bandwidth is 1.5 Mbps, and data packets can be sent continuously.
    b. [2] The bandwidth is 1.5 Mbps, but after we finish sending each data packet we must wait one RTT before sending the next.
    c. [2] The bandwidth is "infinite," meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.

We will count the transfer as completed when the last data bit arrives at its destination. An alternative interpretation would be to count until the last ACK arrives back at the sender, in which case the time would be half an RTT (50ms) longer.

a. Time Req. = 2 initial RTT's (200ms) + 1000KB/1.5Mbps (transmit) + RTT/2 (propagation)
≈ 0.25 + 8Mbit/1.5Mbps = 0.25 + 5.33 sec = 5.58 sec.
If we pay more careful attention to when a mega is 10^6 versus 2^{20}, we get 8,192,000 bits/1,500,000bits/sec = 5.46 sec, for a total delay of 5.71 sec.

b. To the above we add the time for 999 RTTs (the number of RTTs between when packet 1 arrives and packet 1000 arrives), for a total of 5.71+99.9 = 105.61.

c. This is 49.5 RTTs, plus the initial 2, for 5.15 seconds.

3. [5 – 1 each] True or false?
    a. A user requests a Web page that consists of some text and three images. For this page, the client will send one request message and receive four response messages.
    b. Two distinct Web pages (for example, www.mit.edu/research.html and www.mit.edu/students.html) can be sent over the same persistent connection.
    c. With non-persistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.
    d. The Date: header in the HTTP response message indicates when the object in the response was last modified.
    e. HTTP response messages never have an empty message body.

F
T
F
F
F

4. [16] 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.

    a. [8] Would parallel downloads via parallel instances of ***non-persistent*** HTTP make sense in this case?

    b. [8] Now consider ***persistent*** HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer with calculations.

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

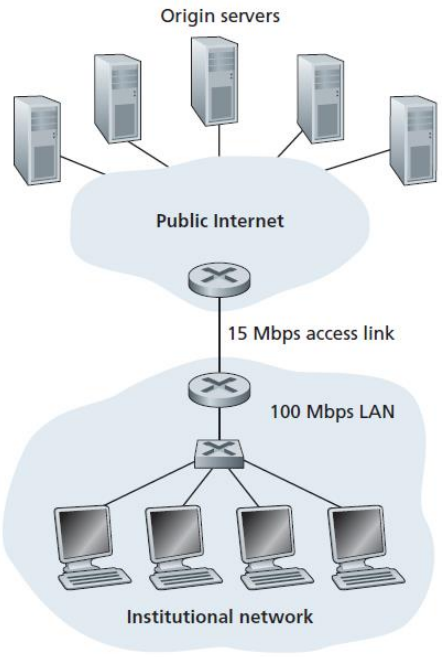
Now consider a ***persistent HTTP connection***. The total time needed is given by:

(200/150+*T*p + 200/150 +*T*p + 200/150+*T*p + 100,000/150+ *T*p)
+ 10*(200/150+*T*p + 100,000/150+ *T*p) =7351 + 24**T*p (seconds)

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

Thus, we see that persistent HTTP is not significantly faster (less than 1 percent) than the non-persistent case with parallel download.

5. [12 – 6each] Consider diagram shown, for which there is an institutional network connected to the Internet. Suppose that the average object size is 850,000 bits and that the average request rate from the institution's browsers to the origin servers is 16 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is three seconds on average. Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access delay, use $\Delta/(1-\Delta\beta)$, where Δ is the average time required to send an object over the access link and β is the arrival rate of objects to the access link.
    a. Find the total average response time.
    b. Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.4. Find the total response time.



Solution:

a) The time to transmit an object of size L over a link or rate R is L/R. The average time is the average size of the object divided by R:

Δ = (850,000 bits)/(15,000,000 bits/sec) = .0567 sec

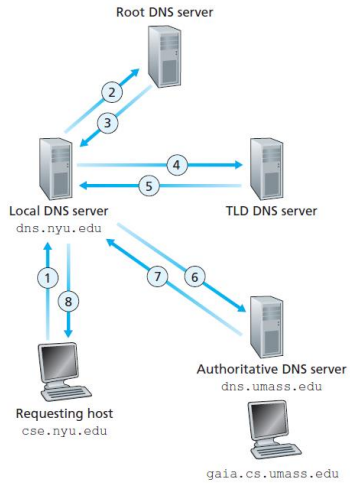
The traffic intensity on the link is given by $\beta\Delta$=(16 requests/sec)(.0567 sec/request) = 0.907. Thus, the average access delay is (.0567 sec)/(1 - .907) $\approx$ .6 seconds. The total average response time is therefore .6 sec + 3 sec = 3.6 sec.

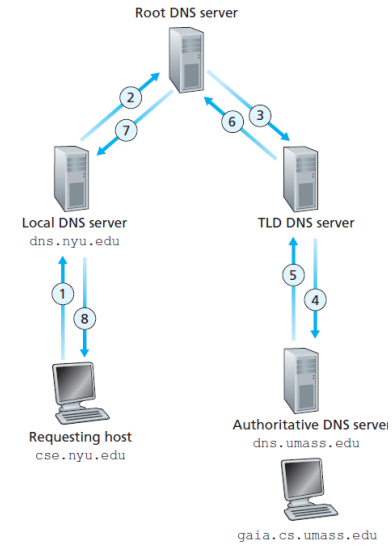
b) The traffic intensity on the access link is reduced by 60% since the 60% of the requests are satisfied within the institutional network. Thus the average access delay is (.0567 sec)/[1 – (.4)(.907)] = .089 seconds. The response time is approximately zero if the request is satisfied by the cache (which happens with probability .6); the average response time is .089 sec + 3 sec = 3.089 sec for cache misses (which happens 40% of the time). So the average response time is (.6)(0 sec) + (.4)(3.089 sec) = 1.24 seconds. Thus the average response time is reduced from 3.6 sec to 1.24 sec.

6. [18] Assume that the RTT between a client and the local DNS server is TTl, while the RTT between the local DNS server and other DNS servers is RTTr. 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)

7. [14] UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, and 01110100.
   a. [6] What is the 1s complement of the sum of these 8-bit bytes? (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work.
   b. [2] Why is it that UDP takes the 1's complement of the sum; that is, why not just use the sum?
   c. [2] With the 1s complement scheme, how does the receiver detect errors?
   d. [2] Is it possible that a 1-bit error will go undetected?
   e. [2] How about a 2-bit error?

Solution:

a. Note, wrap around if overflow.

$$\begin{array}{r} 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0 \\ \hline 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \end{array} \text{ and then } \begin{array}{r} 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \\ +\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0 \\ \hline 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \end{array}$$

One's complement = 1 1 0 1 0 0 0 1.

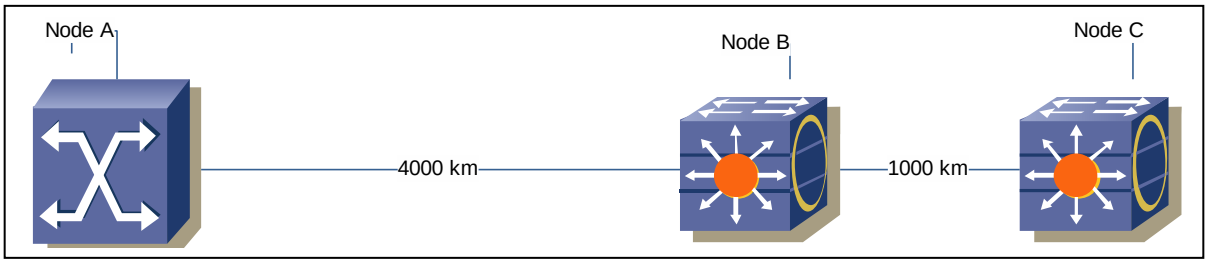
b. To detect errors, the receiver adds the four words (the three original words and the checksum).

c. If the sum contains a zero, the receiver knows there has been an error.

d. All one-bit errors will be detected.

e. Two-bit errors can be undetected (e.g., if the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1).

8. [15] In the diagram below, frames are generated at Node A and sent to node C through node B.



Determine the minimum data rate required between nodes B and C so that the buffers of node B are not flooded, based on the following criteria.

- The data rate between A and B is 100 kbps
- The propagation delay is 5μs/km for both lines
- These are full duplex lines between the nodes.
- All data frames are 1000 bits long; ACK frames are separate frames of negligible length.
- Between A and B, sliding window protocol with size of 3 is used.
- Between B and C, stop-and-wait is used.
- There are no errors.
- Note that in order not to flood the buffers of B, the average number of frames entering and leaving B must be the same over a long interval.

Transmission from $A \rightarrow B$:

- Propagation time = $4000 \times 5\ \mu\text{sec} = 20$ msec
- Transmission time per frame = $\frac{1000}{100 \times 10^3} = 10$ msec

Transmission from $B \rightarrow C$:

- Propagation time = 1000 5 μsec = 5 msec
- Transmission time per frame = x = 1000/R
- R = data rate between B and C (unknown)

➢ A can transmit three frames to B and then must wait for the acknowledgment of the first frame before transmitting additional frames. The first frame takes 10 msec to transmit; the last bit of the first frame arrives at B 20 msec after it was transmitted, and therefore 30 msec after the frame transmission began. It will take an additional 20 msec for B's acknowledgment to return to A. Thus, A can transmit 3 frames in 50 msec.

➢ B can transmit one frame to C at a time. It takes 5 + x msec for the frame to be received at C and an additional 5 msec for C's acknowledgment to return to A. Thus, B can transmit one frame every 10 + x msec, or 3 frames every 30 + 3x msec.

➢ Thus: 30 + 3x = 50;

➢ Therefore x = 6.66 msec and

➢ R = 1000/x = 150 kbps