*CSE 473 – Introduction to Computer Networks* *John DeHart*

# Midterm Exam Review Solution

*10/12/2015*

1. (5 points). A user in Chicago, connected to the internet via a 100 Mb/s (b=bits) connection retrieves a 250 KB (B=bytes) web page from a server in London, where the page references three images of 500 KB each. Assume that the one way propagation delay is 75 ms and that the user's access link is the bandwidth bottleneck for this connection.

   Approximately how long does it take for the page (including images) to appear on the user's screen, assuming non-persistent HTTP using a single connection at a time (for this part, you should ignore queueing delay and transmission delays at other links in the network)?

   *With non-persistent HTTP, there is one TCP connection for the page and one for each one of the images. Each connection incurs a delay of 2*RTT plus transmission time. Hence the total time until the page+images shows up on the user's screen is calculated as follows:*

   *RTT = 2 * 75ms = 150ms*

   *Transmission Time = (Amount of data)/(data rate) =*

   *((250 Kbytes * 8 bits/byte) + (3 * 500 Kbytes * 8 bits/byte))/(100 x 10Mbits/sec)=*

   *( 2 Mbits + 3 * 4 Mbits) / (100 Mbits/s)*

   *(#RTT)*(RTT) + (Transmission Time)*

   *2*(3+1) * 150 ms + (14 Mb)/(100 Mb/s) =*

   *8 * 150ms + (14 * $10^6$ bits)/100 * $10^6$ b/sec =*

   *1200 ms + 0.14 sec =*

   *1200 ms + 140 ms = 1.34 seconds*

   How long does it take if the connection uses persistent HTTP (single connection)?

   *With persistent HTTP, there is only one TCP connection. The TCP connection handshake takes one RTT, this is followed by one more RTT to request the page, one more RTT to request the images, plus the transmission time for the page+images. Hence the total time for the page+images to show up on the user's screen is equal to, 3 RTTs plus transmission time which is:*

   *3*(2*75 ms) + 140 ms = 450 ms + 140 ms = 590 ms*

   Suppose that user's access router has a 4 MB buffer (B=byte) on the link from the router to the user. How much delay does this buffer add during periods when the buffer is full?

   *W=E[N] x E[L] / c, E[N] x E[L] = 4MB = 32*$10^6$ bits, c = 100 Mb/s = $10^8$ b/s*

   *So it adds: 32*$10^6$ bits/$10^8$ bits/sec=320 ms to the delay.*

2. (10 points). Consider a circular DHT with 27 nodes numbered 0, 1,...,26, where the nodes cache key-values pairs for 60 seconds before discarding them. Assume that each node *i* handles keys with hash values in the range 100*i* to 100*i*+99. Also, assume that node *i* has routes to nodes *i*+1, *i*+3 and *i*+9 (where the additions are modulo 27). What is the maximum number of hops required to get from one server to another in this DHT?

*Each step moves you closer to the target by 1, 3 or 9 hops. One can readily see that the maximum number of hops is realized when the key is at the maximum possible distance, i.e., 26 nodes. Given this and because 26=9+9+3+3+1+1, the maximum number of hops is 6.*

Suppose that some client sends a *get* request to node 20 with a key string of "candy corn", and that *hash*("candy corn")=1853. If the request is handled by exactly 4 nodes in the DHT (including 20), which nodes are they?

*The key hashes in the range that belongs to node 18. So node 20 will forward the request to the node that takes it closest to node 18, i.e., using 9 hop route that takes it to node 20+9=2 (mod 27). Node 2 goes through the same process and uses a 9-hop route to reach node 2+9=11. Node 11 cannot use a 9-hop route since it would take it too far, and must therefore use a 3-hop route that takes it to node 11+3=14, which since this is the 4th node is where the query ends (node 14 must have previously cached the entry for hash("candy corn")). So, the four nodes are 20, 2, 11 and 14.*

Suppose node 13 has an entry for "candy corn" in its cache but no other nodes do. Now assume that every other node receives a *get* request for "candy corn". How many of these requests does 13 handle? (Ignore any new cache entries that might be created in the process of handling these requests.)

*Node 13 will handle the request if it is on the shortest path between the node where the request arrived and node 18. This holds whenever the request first arrived at node 13, 10, 4, 1, 22 or 19. So, node 13 will handle 5 requests from other nodes, plus one more if you include the request sent to 13.*

Repeat your answer to the last question assuming that node 2 has the cached entry for "candy corn". (That is, how many of the requests would 2 handle in this case?)

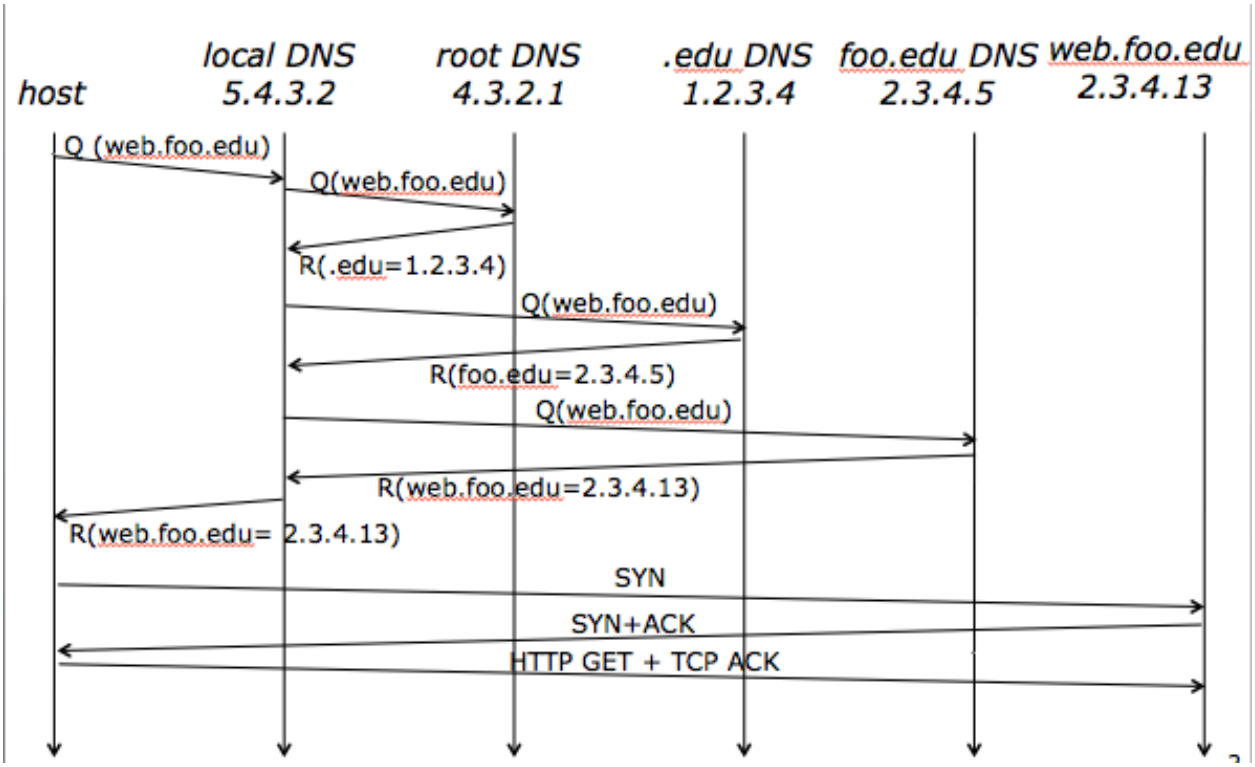
*Node 2 would only handle requests that arrive at node 2 or node 20. So 1 request from other nodes, plus the one sent to 2.*

*If you need to, list each node and the node it will send to to try to get to the node responsible for 1853:*

| | | | |
|---|---|---|---|
| *0 → 9* | *7 → 16* | *14 → 17* | *21 → 3* |
| ***1 → 10*** | *8 → 17* | *15 → 18* | ***22 → 4*** |
| *2 → 11* | *9 → 18* | *16 → 17* | *23 → 5* |
| *3 → 12* | ***10 → 13*** | *17 → 18* | *24 → 6* |
| ***4 → 13*** | *11 → 14* | *18 →* | *25 → 7* |
| *5 → 14* | *12 → 15* | ***19 → 1*** | *26 → 8* |
| *6 → 15* | ***13 → 16*** | *20 → 2* | |

*Bold for nodes leading to 13, underline for nodes leading to 2*

3. (10 points) The diagram below shows a DNS query from a host *A* to its local DNS server. The IP addresses of all hosts are shown in the diagram. The label "*Q(web.foo.edu)*" specifies the query string. Complete the diagram showing all packets sent to resolve the name and continuing through the opening of a TCP connection to the web site and the first GET request. All arrows that represent DNS queries should have a label of the form "*Q(a.b.edu)*" and replies should have a label of the form "*R(b.edu=2.3.7.11)*". TCP connection packets should be labeled with the appropriate flags and HTTP packets with the request type. Assume that the local DNS server performs recursive processing and has nothing in its cache, while the others perform iterative processing. You may assume that all queries and responses are for *A* records.




List all the mappings in the local DNS server's cache after the query has been processed.

*Remember, "recursive processing" means the local server is processing a recursive query and will send out iterative queries. All other servers will get and send iterative queries.*

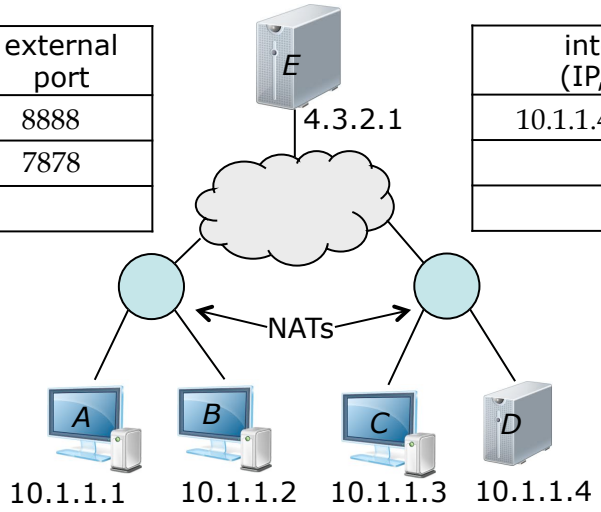
*.edu=>1.2.3.4, foo.edu=>2.3.4.5, web.foo.edu=>2.3.4.13*

List the mappings in the local DNS server's cache if the *.edu* server did recursive processing rather than iterative.

*.edu=>1.2.3.4, web.foo.edu=>2.3.4.13*

4. (15 points). The diagram below shows two residential networks with routers that implement NAT and a remote server with a public internet address

| internal (IP,Port) | external port |
|---|---|
| 10.1.1.1, 5555 | 8888 |
| 10.1.1.2, 6565 | 7878 |
| | |




| internal (IP,Port) | external port |
|---|---|
| 10.1.1.4, 1212 | 5454 |
| | |
| | |

The packet header diagrams at right are for a packet from a host in the left-hand network, going to the server. The first shows the header when the packet arrives at the router, the second shows it when the packet leaves the router. Add an entry to the left-hand NAT table that is consistent with these two packet headers. What is the public IP address of the left-hand router?

| src adr | dest adr | src port | dest port |
|---|---|---|---|
| 10.1.1.1 | 4.3.2.1 | 5555 | 3333 |
| 3.7.5.7 | 4.3.2.1 | 8888 | 3333 |

*Public IP address of left-hand NAT router is 3.7.5.7*

The three header diagrams at right are for a packet from a host in the right-hand network, going to a host in the left-hand network. Fill in the blank fields. Add entries to the two NAT tables that are consistent with this sequence of packet headers. What is the public IP address of the right-hand router?

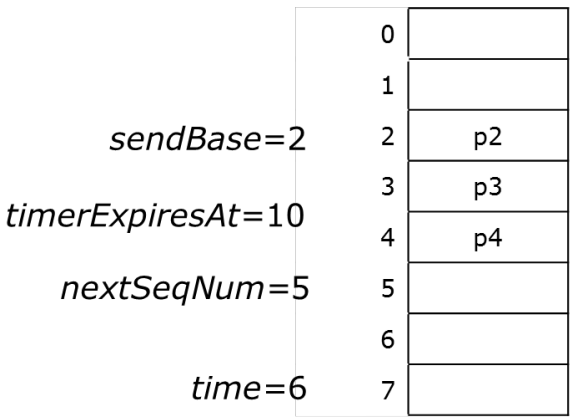
| src adr | dest adr | src port | dest port |
|---|---|---|---|
| 10.1.1.4 | 3.7.5.7 | 1212 | 7878 |
| 5.3.5.2 | 3.7.5.7 | 5454 | 7878 |
| 5.3.5.2 | 10.1.1.2 | 5454 | 6565 |

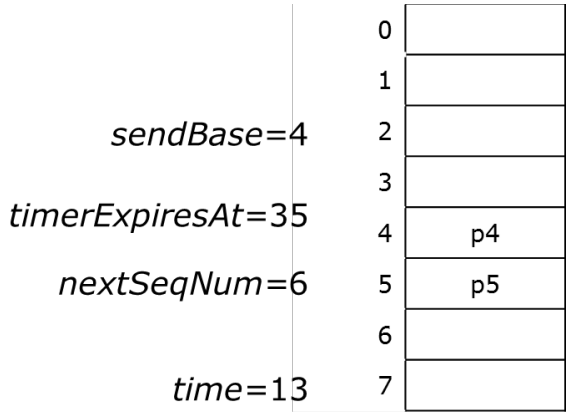
*Public IP address of right-hand NAT router is 5.3.5.2*

In the diagrams at right, fill in the header fields that would be used by a response to the last packet, (the response goes from the left hand network to the right).

| src adr | dest adr | src port | dest port |
|---|---|---|---|
| 10.1.1.2 | 5.3.5.2 | 6565 | 5454 |
| 3.7.5.7 | 5.3.5.2 | 7878 | 5454 |
| 3.7.5.7 | 10.1.1.4 | 7878 | 1212 |

5. (10 points) The diagram at left below shows the state of the sending side of a communication session using a go-back-*N* protocol with cumulative acknowledgements, a window size of 4 and a retransmission timeout of 25. The array represents the "re-send" buffer and entry in the buffer represents a packet with its sequence number. As usual for go-back-*N*, we are using a single shared timer. Its expiration time is shown in the diagram. Assume that the channel may lose packets but never re-orders them.




Suppose that at time 7, the application passes us a new payload to be sent, and at time 12 we receive an ack with sequence number 3.

List all the packets sent by the sender between times 6 and 13, including repeats (if any).

*p5, p2, p3, p4, p5*

Show the state of the sender at time 13, in the right-hand diagram.

*See above*

At time 13, how many additional packets can be sent before an ack is received?

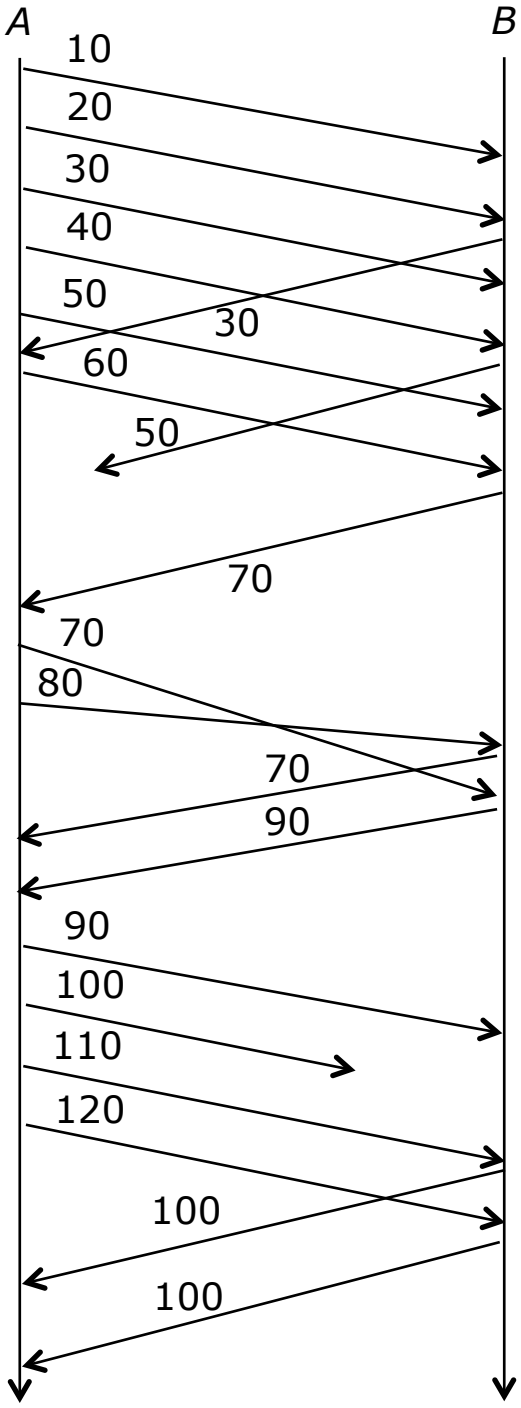
*2*

6. (10 points) Consider a TCP sender that sends 10 bytes per packet. Suppose that the sender transmits six packets one right after another (including the one shown at right). Show the packets and the corresponding acks that TCP would return in a typical situation. Label the data packets with the sequence number and the ack packets with the ack number.

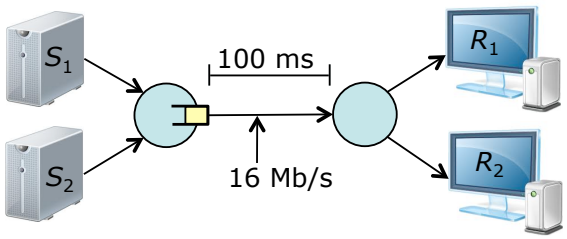
   If the second of the acks is lost, will the sender retransmit any packets? Why or why not?

   *No, because the next cumulative ack lets the sender know that the corresponding packet was delivered.*

   Continuing the example, suppose that the next two packets sent arrive out of order and that the sender transmits four more packets after these and that the second packet in this group of four is lost. Extend the diagram to show these packets and the resulting acknowledgements. You need not show the retransmission of the lost packet.



7. (10 points) The diagram at right shows two TCP senders at left and the corresponding receivers at right. Both senders use TCP *Tahoe*. Assume that the MSS is 1 KB, that the one-way propagation delay for both connections is 100 ms and that the link joining the two routers has a bandwidth of 16 Mb/s. Let $cwnd_1$ and $cwnd_2$ be the values of the senders' congestion windows. What is the smallest value of $cwnd_1+cwnd_2$ for which the link joining the two routers stays busy all the time?




*16 Mb/s is 2 MB/s or 400 KB per RTT. So $cwnd_1+cwnd_2$ =400 KB is the smallest amount that keeps the link busy.*

Assume that the link buffer overflows whenever $cwnd_1+cwnd_2 \geq 600$ KB and that at time 0, $cwnd_1$=500 KB and $cwnd_2$=100 KB. Approximately, what are the values of $cwnd_1$ and $cwnd_2$ one RTT later? Assume that all losses are detected by triple duplicate ACKs.

*They are both set to 1kB, i.e., one MSS, since with TCP Tahoe the connection goes back to slow-start after a loss.*

At this point, what are the values of $ssthresh_1$ and $ssthresh_2$?

*$ssthresh_1$=250 KB and $ssthresh_2$=50 KB*

After 20 more RTTs, approximately what are the values of $cwnd_1$ and $cwnd_2$?

*After 8 RTTs $cwnd_1$ is about 250 KB, so after 20, it is about 262 KB. After 6 RTTs $cwnd_2$ is about 50 KB, so after 20, it is about 64 KB.*

Approximately, how many more RTTs before $cwnd_1+cwnd_2 \geq 600$ KB again?

*262+64=326, so it will take (600–326)/2 = 274/2 = 137 RTTs before $cwnd_1+cwnd_2 \geq 600$ KB again.*