Prof. Athina Markopoulou
Spring 2016

Introduction to Computer Networks
EECS148/CompSci 132

Midterm Exam, Solution Sketch

Tuesday, May 3rd, 2016

All paper-based materials (books, notes, printouts, hand-written notes) can be used.
All other electronics (laptops, phones, kindle, calculators) should be turned off.

Student's Last Name:

Student's First Name:

UCINETID:

Seat:

Problem #	Points	Out of Total
1		30
2		40
3		30
Total		100

Academic Honesty Policy:
I agree to abide by the UCI Academic Senate Policy on Academic Honesty (Appendix VIII.B), which specifies that students have responsibility for:

1. *Refraining from cheating and plagiarism.*
2. *Refusing to aid or abet any form of academic dishonesty.*
3. *Notifying professors and/or appropriate administrative officials about observed incidents of academic misconduct. The anonymity of a student reporting an incident of academic dishonesty will be protected.*

Student's Signature:

GOOD LUCK!

1. *(30 Points)* **Packet Switching, Stop-and-Wait, Delay and Throughput.**

Host A wants to send to Host B a file of $L = 10,000$ bits. There are two links and one router (R) on the path from A to B. Packet switching is used for all transmissions in both directions. Each of the two links (A-R, R-B) has the same transmission bandwidth $W = 1Mbps$ and the same propagation delay $\tau = 5ms$, in both directions. Host A breaks the file of L bits into packets of equal size, each consisting of data ($D = 1000$ bits) and a negligible header (of $H << D$ bits) added to it. Furthermore, the end-points A and B use the stop-and-wait protocol to reliably transmit the file. (Host B sends back an acknowledgment packet ACK, of size H, bits for each packet received. Host A waits for a timeout $T = 30ms$ to receive the ACK; if the timeout expires before receiving the ACK, A retransmits the same packet; if it receives an ACK before the timeout, A transmits the next packet.) Assume zero processing delay at the nodes, infinite supply of packets at host A, no delays in sending the ACKs. No other traffic flows on that path.

(a) *(2 Points)* What is the transmission delay of a single packet?

(b) *(15 Points)* First, assume that no packet gets lost or corrupted. What is the message delay (i.e., from the time that A starts transmitting until the time B receives the entire file)? What is the throughput as seen by A (i.e. the useful bits/sec)?

(c) *(10 Points)* Then, consider that the first packet transmitted is lost on the first link (A-R), and no other packets get lost or corrupted. Show the sequence of packets sent until the *first* packet is correctly received at B.

(d) *(3 Points)* If you could choose the Timeout T to be used in this stop-and-wait protocol, what value would you choose?

Solution:

Number of packets $N = \frac{L}{D} = \frac{10000}{1000} = 10$. Each packets has size $P = D + H \simeq 1000$ bits (since the header size is negligible).

(a) Transmission of a single packet: $d_{tr} = \frac{P}{W} = \frac{1000bits}{1.10^6 bps} = 10^{-3}sec = 1ms$.

(b) Please see the corresponding figure. For one packet to be received at the destination, it takes time: $2\tau + 2d_{tr} = 12ms$. For one packet to make it to the destination and for the ACK to be received back at the source it takes time: $2\tau+2d_{tr}+2\tau = 4\tau+2d_{tr} = 4*5ms+2*1ms = 22ms$. We are now ready to answer the questions.

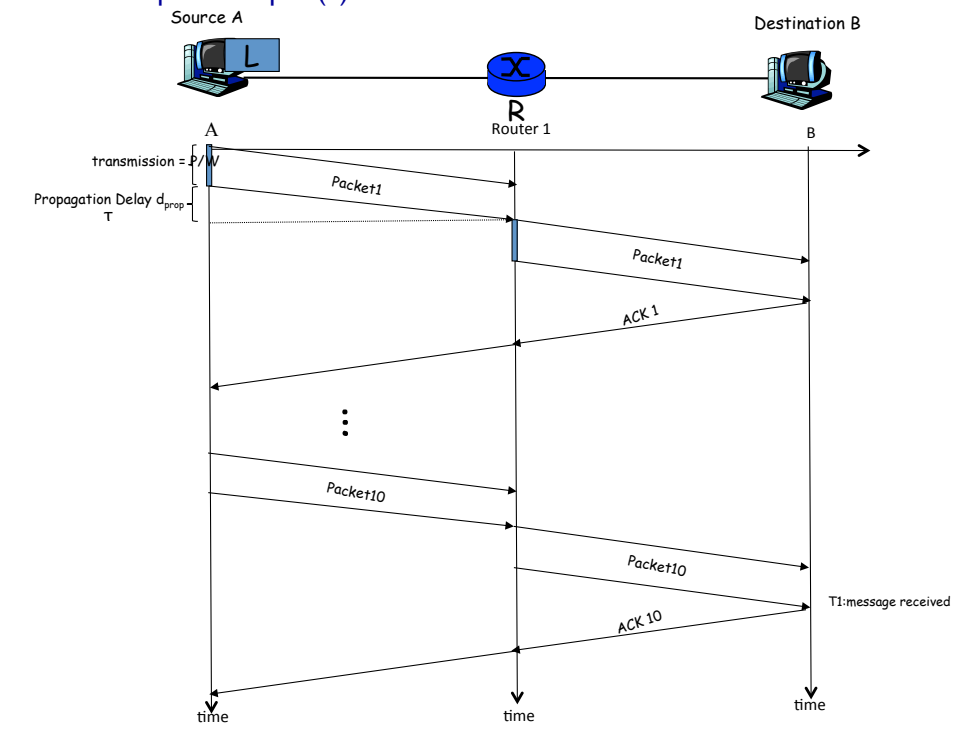
First, the message delay requires all first $N-1$ packets to be received and acknowledged, and the 10th packet to be received:

$$Message\ delay = (N-1)*(4\tau+2d_{tr})+2\tau+2d_{tr} = 38\tau+20d_{tr} = 9*22ms+12ms = 210ms$$

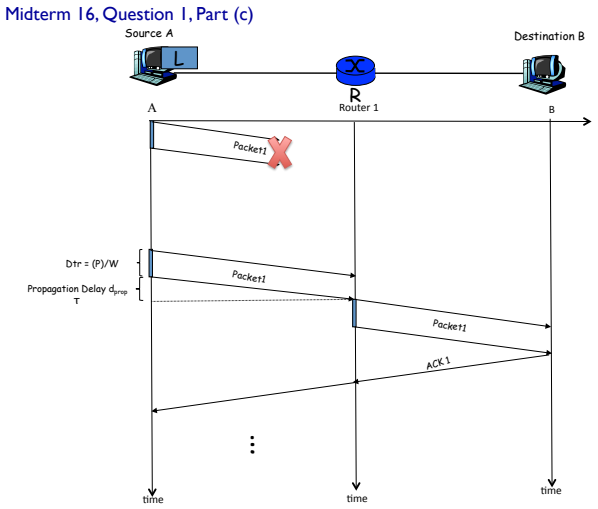
Second, the throughput calculation can be made for each packet or for all 10 packets in a message. For example, it takes time $\frac{L}{D}(4\tau + 2d_{tr})$ to receive and acknowledge all N packets. During this time, only the transmission of the message (L bits in total) is useful. Therefore:

$$Throughput - \frac{L}{\frac{L}{D}*(4\tau+2d_{tr})} = \frac{D}{4\tau+2d_{tr}} = \frac{1000}{22ms} = 45.454bps$$

(c) See figure for the messages exchanged from the time that the first packet starts being transmitted at A until it is correctly received by B. The total time (not required in the answer) is: $d_{tr} + Timout + 2d_{tr} + 2\tau = 2*5ms + 30ms + 3*1ms = 43ms$



(d) The timeout value should be at least as large as the time from starting transmitting a packet until the ACK is received; otherwise the timeout will expire prematurely before the ACL can reach the source.: $T \geq 4\tau + 2d_{tr} = 22ms$



2. *(40 Points)* **Application Layer**

(a) *(20 Points)* **HTTP.** Below is the HTTP Response sent from the server in response to an HTTP GET request.

HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008 12:39:45 GMT <cr><lf> Server: Apache/2.0.52 (Fedora)<cr><lf>Last-Modified: Sat, 10 Dec 2005 18:27:46 GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf> Accept-Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf> Keep-Alive: timeout=max=100 <cr><lf> Connection: Keep-Alive<cr><lf>Content-Type: text/html; charset= ISO-8859-1 $< cr >< lf >< cr >< lf >$ <!doctype html public "-//w3c//dtd html 4.0 transitional//en"><lf><html><lf><head><lf> <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"><lf> <meta name="GENERATOR" content= "Mozilla/4.79 [en] (Windows NT 5.0; U) Netscape"><lf> <title> CMPSCI 453 / 591 / NTU-ST550A Spring 2005 homepage</title><lf></head> <lf><much more document text following here (not shown)>

Answer the following questions. Quickly justify your answer referencing the above HTTP Response.

i. Was the server able to successfully find the document or not? What time was the document reply provided?

ii. When was the document last modified?

iii. How many bytes are there in the document being returned?

iv. What are the first 5 bytes of the document being returned? Did the server agree to a persistent connection?

Answer: The relevant parts of the HTTP Response are underlined above.

i) *YES.* The status code of 200 and the phrase OK indicate that the server was able to locate the document successfully. The reply was provided on Tuesday, 07 Mar 2008 12:39:45 Greenwich Mean Time.

ii) This is provided after "Last Modified". The document index.html was last modified on Saturday 10 Dec 2005 18:27:46 GMT.

iii) This is provided after "Content-Length": there are 3874 bytes in the document being returned.

iv) The HTTP header ends after

<cr><lf><cr><lf>

Whatever follows that is the HTTP body of the response. The first five bytes of the returned document are the 5 first characters: "¡!doc".

The server also agreed to a persistent connection, as indicated by the "Connection: Keep-Alive" field

(b) *(20 Points)* **DNS and HTTP.**

i. *(10 Points)* DNS includes the following resource records (RR) related to Stanford:

- `(saga.stanford.edu, 171.64.74.48, A)`
- `(stanford.edu, 171.64.13.26, NS)`
- `(stanford.edu, mx2.stanford.edu, MX)`
- `(www.stanford.edu,www-lb.stanford.edu, CNAME)`

Explain what each record means and where it is stored in the DNS hierarchy. (E.g. at which authoritative server, TLD server, or elsewhere?)

Asnwer

- A record: authoritative mapping between name and IP. This record is stored at authoritative name server at Stanford.
- NS record: A nameserver for the domain stanford.edu is 171.64.13.26. This record is stored at TLD responsible for .edu.
- MX record: A mail server for stanford.edu is mx2.stanford.edu. This record is stored at authoritative server at Stanford.
- CNAME record: `www.stanford.edu` is an alias name for host www-lb.stanford.edu. This record is stored at Stanford name server.

In addition, any of these records can be cached in other DENS servers in the DNS hierarchy.

ii. *(10 Points)* You open your browser and you click on `http://www.stanford.edu/`. Assume that the Stanford webpage consists of an HTML file and 4 images. Let RTT_{cs} be the RTT between your laptop and the Stanford webserver. The HTML file is small, thus its transmission delay is negligible. However, the images are large: (an HTTP response for) each image fits in exactly two TCP segments, each with the maximum segment size (MSS Bytes) and transmission delay $\frac{1}{4} \cdot RTT_{cs}$. The transmission delays of all other messages are negligible.

Assume you are the first person at UCI who visits the website `www.stanford.edu`. For simplicity, let RTT_{DNS} be the delay for any DNS-query and response made. State any other assumptions you make regarding DNS. Additional simplifying assumptions: no packet is lost; ignore the TCP window effect; ignore any processing delays; all TCP segments have either negligible or maximum (MSS) size, as specified above.

Consider that your browser uses **non-persistent HTTP with up to 5 parallel connections.** You are interested in the delay from when you enter the URL until the webpage is displayed on your browser. **(i)** List (or draw) all messages exchanged including application layer messages (e.g., DNS, HTTP) and transport layer segments (e.g., UDP, TCP) and **(ii)** compute the delay.

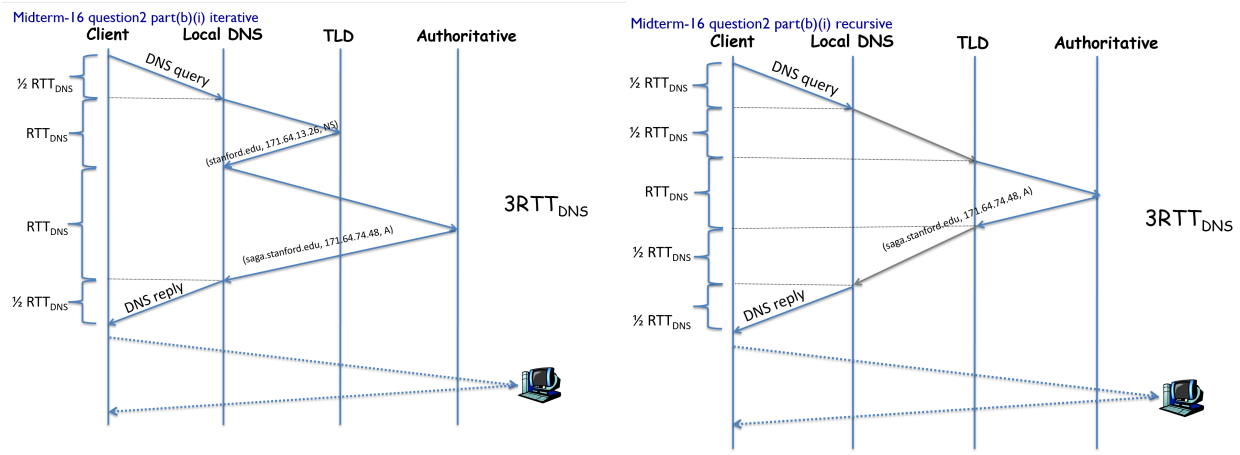
Answer:

(i) DNS. First thing the browser has to do is to find the IP of `www.stanford.edu`. It contacts its local DNS server (e.g., *ns4.service.uci.edu*). Since this is the first request out of UCI for Stanford, the local DNS server does *not* have the answer cached. Since, typically, local servers have the addresses of the TLD servers cached,

the local DNS server contacts the TLD server (for .edu).[1] Depending on whether the queries are iterative or recursive, things will evolve as in subfigures (a) or (b) of Figure 1, respectively. The total time required for DNS is:

$$T_{DNS} = 3RTT_{DNS}$$

We also note that all DNS messages are carried over the UDP transport protocol.



(a) iterative DNS query

(b) recursive DNS query

Figure 1: Resolving `www.stanford.edu` to its IP address. The Client contacts the local DNS server, who is assumed to have cached the TLD servers. Depending on whether iterative or recursive queries are used, things will evolve as depicted.

(ii) Obtaining the base html. Once the browser has the IP address, it opens a TCP connection, and sends an HTTP request for the base object; this is "Connection 1", shown in the leftmost part of Figure 2. The based html is small, so its transmission delay is negligible. Since this is a non-persistent HTTP connection, the TCP connection starts closing down right after receiving the base object (for the details see your book and slides). This takes one RTT_{CS} to set up the TCP connection (SYN, SYNACK), and another RTT_{CS} to send the HTTP request for the base html (the same packet has the last part of the 2-way TCP handshake: the TCP SYN flag set), and response, so in total:

$$T_1 - T_0 = 2RTT_{CS}$$

(iii) Obtaining the images. Since this HTTP session can open up to 5 parallel connections, it can start 4 more TCP connections (in parallel with the first one, currently in the process of closing) and use them to request one object per connection. These are connections 2-5 on the right side of the figure. The objects have non-negligible size: each object fits in 2 TCP segments[2] of transmission time $\frac{RTT}{4}$

[1] If the local DNS server does not have the TLD server cached, then it will contact the root instead, and this answer is correct as well. The total time for DNS will be the same, since every DNS query-response pair is assumed to take the same RTT_{DNS}.

[2] We ignore the effect of the TCP congestion for now, and we assume that the sending window is large enough to allow to send both segments back-to-back.

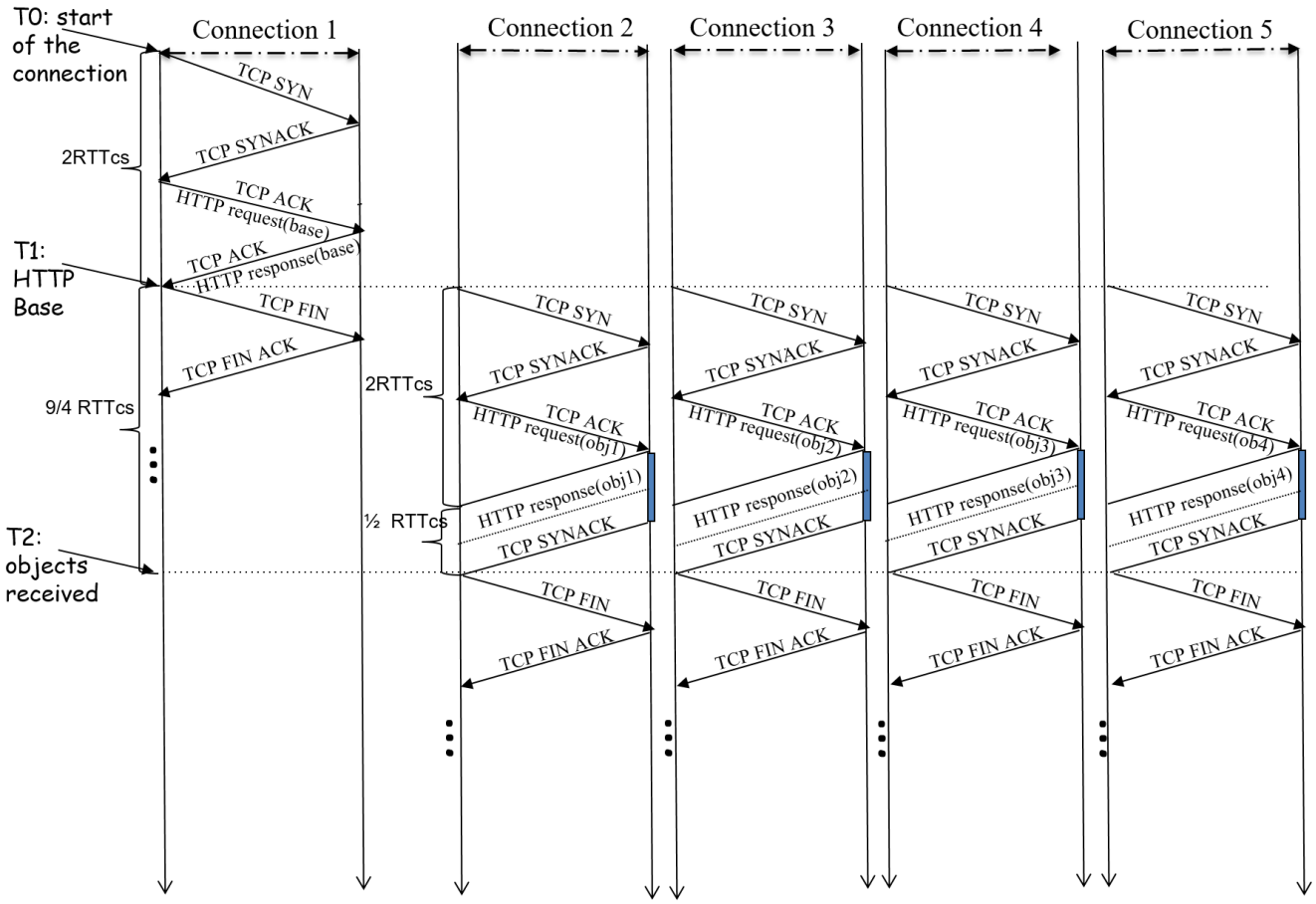


Figure 2: HTTP Request and Response for base html and 4 Parallel Connections for the referenced objects.

each. Each packet has a function simultaneously at the TCP and HTTP layer, and both are indicated on each packet on Figure 2. As soon as the images get obtained (time T_2), the connections start closing. Therefore:

$$T_2 - T_1 = RTT_{CS} + RTT_{CS} + \frac{2}{4}RTT_{CS} = \frac{5}{2}RTT_{CS}$$

At time T_2, the browser can display the whole page and the images. So the time from when the user enters the URL until the browsers displays is in total:

$$T_{DNS} + T_2 - T_0 = 3RTT_{DNS} + 2RTT_{CS} + \frac{5}{2}RTT_{CS} = 3RTT_{DNS} + \frac{9}{2}RTT_{CS}$$

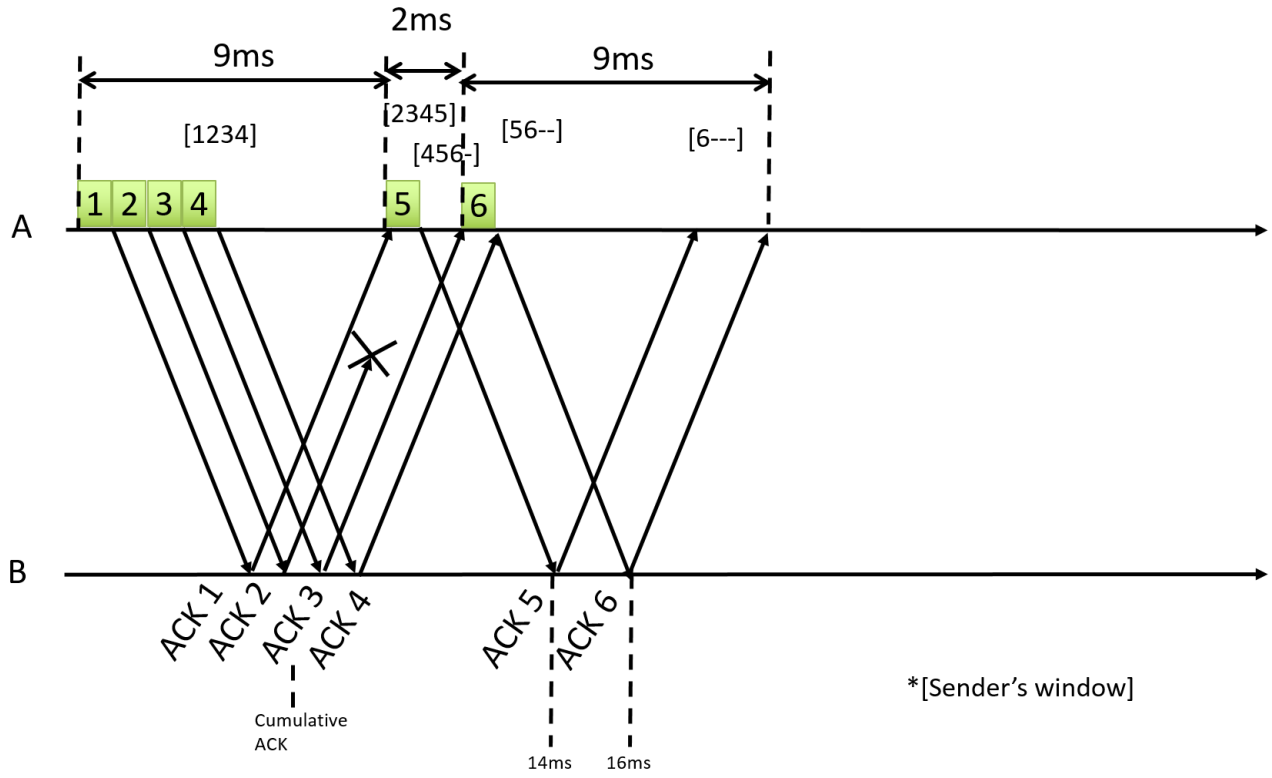
3. *(30 Points)* **Reliability: Go-back-N vs. Selective Repeat.**

Suppose that host A sends $N = 6$ data packets to host B using one of the idealized pipelined protocols we learnt in class (a) Go-Back-N (GBN) (b) Selective Repeat (SR). All protocols have (sender and/or receiver if applicable) window size equal to 4 packets and Timeout value equal to 10 ms (assume that the timeout starts *after* the packet is sent). The one-way propagation delay between host A and B, as well as from B to A, is 4 ms, the transmission time of each data packet is 1ms, and the transmission time of each ACK packet is 0 ms. Suppose that ACK No. 2 is lost on the way from host B to host A and no other packet gets lost. The beginning of the space-time diagrams are provided below; the sequence numbers refer to packets.

- Fill out the rest of the diagrams until all 6 packets are sent and acknowledged. Show all packets (transmissions, retransmissions and acknowledgements), their sequence numbers and the times they were sent/received.
- For each scenario, write down the total time, i.e., when the all 6 packets are successfully acknowledged at the server.

Answer: The diagrams below show show the messages exchanged, their timing, and the sender/receiver window.

(a) *(15 Points)* **GBN with ACK 2 lost.**
Total time = 20 ms



(b) *(15 Points)* **Selective Repeat with ACK 2 lost.**
Total time = 30 ms