1. To segment or not to segment [25 points, parts a-i]

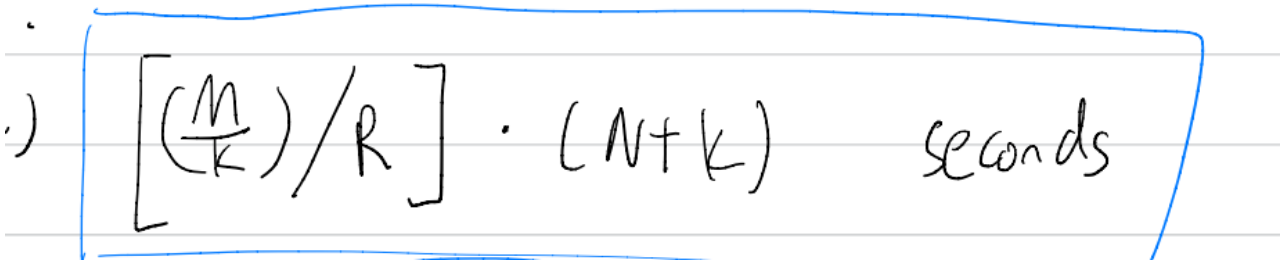
Normally large messages are broken down into smaller packets, a process called segmentation. In this question, we will compare segmenting to not segmenting (sending the message as one large packet). Suppose Bob wants to send a message that is M bits long to Alice.
With segmenting, the message is sent as k packets. Without segmenting, it is sent as a single packet of M bits. There are total N routers between Alice and Bob, and each link has bandwidth R bps. Ignore propagation delay and the time to break a message up and reassemble. Until mentioned, ignore headers and loss.

a. [2 points] With segmentation, what is the time to deliver the full message?

$(\frac{N}{k} + 1) \times \frac{M}{R}$

2 points for answer

1 point if misses "+1"



总路程=num of packet + number of router

b. [2 points] Without segmenting, what is the time to deliver the full message?

$(N + 1) \times \frac{M}{R}$

2 points for answer

1 point if misses "+1"

c. [3 points] Suppose now each datagram delivered has an h(bits) header, both with and without segmenting. Without segmenting, the whole message is one datagram. With segmenting, each packet is one datagram. In which circumstances do the two approaches have the same end-to-end delay?

Transmission delay for message switching: $(N + 1) \times \frac{h + M}{R}$

Transmission delay for packet switching: $(\frac{N}{k} + 1) \times \frac{kh + M}{R}$

$(N + 1) \times \frac{h + M}{R} = (\frac{N}{k} + 1) \times \frac{kh + M}{R}$

$h = \frac{NM}{k}$

1 point for correct message switching

1 point for correct packet switching

1 point for final answer

d. [1 point] Based on your answer for question a)~c), if there are many routers along the path, will delivery be faster with or without segmentation?

With segmentation

e. [2 points] Which mechanism (or behavior) is the root cause of the difference we see in d)? Explain in a few sentences.

Lower transmission delay allows for better pipelining. Because the switch uses **store-and-forward**, for message switching, the switch cannot begin to transmit the file until it receives it fully over the 1st link, resulting in a long delay before the switch can begin transmitting even the beginning of the file.
But for packet switching, after the first packet arrives at the switch, both of the links could be busy transmitting segments of the files at the same time, thus decreasing the delay.
2 points if store-and-forward is mentioned, or has a closely related expression (such as the router has to wait for the entire packet to arrive before forwarding it)
1 point if says“parallel” or “parallelism”.(It’s a bad expression, but it shows you know the routers are working together in the example above.) “Pipelining” or “sequential” are better expressions but still not the root cause. We want your answer to explain why pipelining (or so called “parallelism”) is not possible when the file is not segmented.

Next, we will explore segmentation in a lossy link. Consider transferring a file of length L=1.2 MB from Host A to Host B. The path from A to B is a single link with transmission rate R=2Mbps on which packets can be dropped randomly (not due to congestion). If a packet loss happens, the sender can sense the loss at the end of the transmission period for the lost packet and retransmit immediately. We will assume each packet (regardless of size) has a 2% chance of being dropped on each attempt. We will compare segmentation into packets of length 1500 B (assume no headers need to be added), vs no segmentation (single transfer of 1.2 MB).

f. [3 points] With segmentation, what is the expected time for the whole file to be

transmitted? You don't need to give the exact result, but please write down a formula from which the expectation can be calculated.

# of packets = 1.2 MB / 1500 B = 800 packets
1500 B / 2 Mbps = 6 ms to transmit one packet one time
Expected # of transmissions of a packet = E[n] =
$\sum_{n=1}^{\infty} n(0.98)(0.02)^{(n-1)} = 1/.98 = 1.0204081632...$
Expected time to transmit one packet = 6 ms * E[n] = 6ms * 1/.98 = 6.12244898 ms
The chance of each packet experiencing loss is independent, so total delay is the # of packets times the time to transmit one packet:
800 packets * 6ms * 1/.98 = 4.8 seconds/.98 = 4.89795918 seconds

1 point for correct # of packets and time for transmitting 1 packet (each 0.5 point)
2 points for total delay (can also be 800 * t * (1 + 0.02 + 0.02^2 +...)) (partial credit: 1 point for getting correct first or second try, 1 point for getting the full summation correct) Can receive full credits giving the correct formula and no need to show the exact result.

One common mistake is having wrong transmit time for one packet (note that 1Byte = 8 bits). In this case, will lose 0.5 points in f and 0.5 points in g.

g. [3 point] If segmentation is not used, will the expected transmission time be the same as with segmentation? (Same/Different) Explain your answer (either give a brief argument or a calculation).

Same
1.2 MB / 2 Mbps = 4.8 seconds to transmit the message one time
Expected # of transmission = 1/.98 still
Expected time to transmit the message = 4.8/.98 = 4.89795918 seconds

1 point for correct result
2 points reason/calculation, will give full points if there is correct formula to calculate the transmission time or the argument clearly explains why the expected time will not change with or without transmission.

h. [6 points] Users don't have a straightforward way to assess the "expected" delay of their network. Instead, let's assume that a user does not complain at all if the transmission can be completed in less than 200% of the “ideal” delay (file size/transmission rate), and becomes very unhappy if it is 200% or more. For this question, we can refer to 200% as *unacceptable delay.* What is the probability of unacceptable delay with and without segmentation (using the same parameters as in the previous questions)? In this question (but not the previous one), we can assume that the 1st retransmission (the 2nd attempt) for any packet is guaranteed to be successful.
Please give your calculations for:

i) Segmented, delay at least 200%

Since we are assuming every retransmission works, this only occurs if every packet is retransmitted.
So 0.02^800, a very very small number.

1 point correct unacceptable delay
1 point correct situation (retransmit all packets), if the situation is correct, will also get the point for unacceptable delay
1 point correct answer

ii) Not segmented, delay at least 200%

If it is not segmented, then it takes 4.8 seconds to transmit the message one time.
If it has to be retransmitted, then it will take 200%.
So this is the chance that it does not succeed on the first try. P(fail at first transmission) = 0.02

1 point correct unacceptable delay
1 point correct situation (retransmit a packet), if the situation is correct, will also get the point for unacceptable delay
1 point correct answer

i. [3 points] Based on your answer for expected delay and unacceptable delay, discuss how segmentation affects the performance of file transmission. Specifically, what explains your results? Explain in a few sentences.

At the same per-packet drop rate, segmented transmission achieves the same expected delay, while delay is more likely to fall into the “acceptable” margin, and less likely to fall into the “unacceptable” margin. (Its performance becomes even better if packet drop rate grows with packet size.) The reason for that is the use of packet-based retransmission mechanism. Segmentation brings faster recovery from loss, since we don’t retransmit the whole file when there’s a packet loss.

2. How long is your packet? [10 points, parts a-b]

Suppose that two switches are connected by a fiber optic link. The first switch begins

transmitting the largest IP packets allowed by Ethernet, sending packets back-to-back at the OC-48 rate (you will have to look up what these values are). Just as the first switch finishes sending the payload of the second packet, the second switch receives the first bit of the first IP packet. The speed of light in fiber is 2/3 the speed of light in a vacuum.

a. [8 points] How far apart are the two switches?

Answer:

a. This means that 2 packets fit on the wire at a time, so the switches are the length that a bit can travel in the time that it takes to transmit two packets.
OC-48 R=2488.32 Mbps
Ethernet MTU L = 1500B
transmission delay D = 1500B * 8(b/B) / (2488.32 * 10^6 (b/s))
distance = 2* D(s) * 3*10^8 (m/s) * 2/3 =
2*2/3 * 3*10^8 (m/s)*(1500B * 8(b/B) / (2488.32 * 10^6 (b/s)))=2*964.51 m = 1929.02 meters
Note: For Ethernet MTU, 1518Byte (1500 bytes + Ethernet headers=1518 bytes) can also be used, which will lead to a slightly different answer. Answers with reasonable round up in the result/calculation steps, or used more exact value for OC-48 or speed of light will also be accepted.
+1 for OC-48 rate
+1 for Ethernet MTU size
(for the above two parts, only +0.5 each if the answer is implicit)

+1 for speed of light

+2 for formula of transmission delay of a single packet
(only +1 for wrong units (e.g. B to b; G is 10^9 here, not 2^30))

+2 for formula of distance
(only +1 for forgetting to use 2 packets)
(only +1 for using speed of light in the vacuum)

+1 for correct final answer
(2km is not reasonable round up)

b. [2 points] Suppose the link is swapped to OC-24. Does the "length" of a packet on the wire become shorter or longer? Why?

3.

b. Longer. OC-48 is 2 times higher transmission than OC-24, so with OC-24 it takes more time to send a packet (higher transmission delay). The propagation speed is the same, but the "front" of the packet has more time to propagate so will travel farther.
Note: students can also use some equations to explain the relationship.
+1 Answer longer
+2 Answer with solid explanation
(only +1 if the answer regards “length” as the transmission delay)
(only +1.5 if the explanation is correct but not solid enough) (need to talk a bit about the propagation part (e.g. rate))

It gets easier [20 points, parts a-h]

HD video is 1080p=1920x1080 pixels. Netflix recommends 5.0 Mbps for HD quality. Suppose Ethan sets up his phone as a hotspot so that everyone in class can share his Verizon 4G connection (at the speed he reported in class) to stream video.

a. [2 points] If FDM is used to divide the connection into circuits, assuming no traffic other than Netflix, how many students can watch HD video? Show your work.

78.52 Mbps / (5.0 Mbps/user) = 15 users
Explanation: to stream at HD quality, each user needs 5Mbps bandwidth.
+1 correct equation
+1 correct answer
-.5 no units

Netflix videos are (or at least were) encoded in 4 second chunks (see this paper). Netflix recommends 5.0 Mbps for HD, but it actually decides the encoding bitrate (bits per second) for different video resolutions (pixels per image) based on properties of the video. For example, BoJack Horseman is simply drawn (and hilarious and heartbreaking), so it begins streaming 1080p video at 1.5 Mbps.

b. [2 points] Continue to assume that circuits were provisioned for 5.0 Mbps (which more complex HD videos might need), but the class is watching BoJack, with chunks encoded at 1.5 Mbps (assume that includes all headers and overhead). What is the download time for a 4 second chunk of video? Show the math.

((4 seconds) * (1.5 Mbps)) / (5.0 Mbps) = 1.2 seconds
Explanation: a 4-sec video chunk is (4 s)*(1.5 Mb/s)=6.0Mb, therefore needing (6Mb)/(5.0Mb/s)=1.2 sec to download.
+1 correct equation
+1 correct answer
-.5 no units

c. [2 points] What fraction of time would the connection be idle for a provisioned 5.0 Mbps circuit when streaming BoJack (assuming the only traffic is the chunk downloads)?

Explanation: From b), downloading a 4-sec video chunk requires 1.2 seconds, so for 4-1.2=2.8 sec the connection is idle. That is to say, 2.8 / 4 seconds = 0.7 fraction of time the connection is idle.
+1 correct equation
+.5 if incorrect equation, but instead uses 1 - 1.5 Mbps / 5.0 Mbps to get answer
+1 correct answer
-.5 no units
(4 seconds - 1.2 seconds) / 4 seconds = 0.7

d. [4 points] Suppose we switch to using packet switching. Use the idle time calculated in the last question (for streaming 1.5 Mbps on a 5.0 Mbps circuit). How many students can watch BoJack at once, with every student downloading chunks at 5.0 Mbps at a random time within a few minute window, while keeping the chance of a collision below 1%? Show your work.

31 users:
100*(1- sum [(31 choose n) .3^n .7^(31-n)] from n=0 to 15) = 0.954044%

$$100\left(1-\sum_{n=0}^{15}\binom{31}{n}\times 0.3^{n}\times 0.7^{31-n}\right)$$

32 users:
100*(1- sum [(32 choose n) .3^n .7^(32-n)] from n=0 to 15) = 1.38399%

$$100\left(1-\sum_{n=0}^{15}\binom{32}{n}\times 0.3^{n}\times 0.7^{32-n}\right)$$

Need to show both for full credit, because you need to establish that 31 can fit and 32 can't fit.
+1 Correct formula
+1 Derivation of formula or justification based on knowledge of common distributions
Either
+0.5 Show that 31 users can keep the probability of collision below 0.1
+0.5 Show that 32 users can NOT keep the probability of collision below 0.1
Or
+1 Explain that 31 is the maximum integer to satisfy the inequality.
+1 Correct answer

e. [2 points] The network operator notices that everyone is only using 1.5 Mbps, so she decides to switch to provisioning circuits at 1.5 Mbps. With that change, how many students can watch simultaneously? Show your work.

78.52 Mbps / (1.5 Mbps/user) = 52 users
+1 Correct formula
+1 Correct answer
-0.5 for lack of units.

f. [4 points] Suppose that many students try to watch via packet switching, with downloads

at 5.0 Mbps. What is the chance there will be a collision? Show your work.

Explanation: Like in part (d), the probability of a given user downloading is 0.3. The network can still support a maximum of 15 people downloading at the same time, so we need to find the chance of collision with 52 people periodically attempting to download at the same time. We can adapt the same formula derived in part (d).

+1 correct formula
+1 explanation (okay if explained in part (d))
+1 correct use of formula
+1 numerical answer
-0.5 for lack of units.
50.3924%:
100*(1- sum [(52 choose n) .3^n .7^(52-n)] from n=0 to 15)

$$100\left(1-\sum_{n=0}^{15}\binom{52}{n}\times 0.3^{n}\times 0.7^{52-n}\right)$$

g. [2 points] Consider the various answers you've found. Which download rate (1.5 Mbps vs 5 Mbps) and network design (circuit vs packet switching) is best when everyone is watching BoJack Horseman? Explain your answer in a few sentences.

Circuit switching at 1.5 Mbps is the best. Since we know everybody will be downloading at 1.5 Mbps, we don't need to worry about any users having an insufficient download speed. Trying to do packet switching at 1.5 Mbps would not allow any additional users without guaranteeing collisions, so it offers no benefit. Packet switching at 5 Mbps has a high probability of collision for the same number of users as calculated in (f). And circuit switching at 5 Mbps would support fewer users and waste bandwidth.

+1 Answer for circuit switching at 1.5 Mbps
+1 Logical justification for answer, even if answer is incorrect
Circuit switching at 1.5 Mbps. (Could try to do 5 Mbps with TDM.) We know exactly the demand, so we can provision it.

h. [2 points] Now suppose different students are watching different videos. Which approach should we use? Why is the same/different than in the last question? Explain in a few sentences.

Packet switching may be better, because we do not know the exact bitrate needed by various videos, so a fixed allocation could provide too much or too little, and it would be hard to do the time assignments if different users may start at any time.
Since some users might watch a video encoded at more than 1.5 Mbps, having a 1.5 Mbps connection would prevent those users from watching videos without constant buffering, so limiting to 1.5 Mbps connections is off the table. Circuit switching at 5 Mbps would have a lot of wasted bandwidth in the case in which a substantial portion of the users are watching BoJack (or a similarly low-bitrate encoding) anyway. Since we don't know what the usage will be, packet switching is the better option since it can support different needed data rates without wasting bandwidth.

+1 Answer for packet switching at 5 Mbps
+1 Logical justification for answer, even if answer is incorrect

1. Dive into the deepest site [16 points, parts a - j]

You are using a workstation in the Columbia network and want to browse content from Borjomi ("the deepest site" for those who have time). You want to browse https://www.thedeepestsite.com

a) [1 point] You enter the URL on the search bar of your browser. Your browser has an empty cache. What server should the workstation contact to get the IP address corresponding to https://www.thedeepestsite.com ?

The local name server of the Columbia network (or other answer that makes sense, e.g. public dns) (1 point)
(no point if say root/TLD/authoritative name server) (0.5 point if only provide specific IP or name, e.g. dns.columbia.edu, and no point if the specific IP or name is incorrect, e.g. dns.thedeepestsite.com)

b) [1 points] When contacting the server from your previous answer, what application protocol is used?

DNS (0.5 point if the wrong full name is written)

c) [1 points] When contacting the server, what transport protocol is used?

UDP (0.5 point if the wrong full name is written) （因为 DNS request small， 所以 udp 足够）

d) [1 points] When contacting the server, what destination port is used?

53 （书上写的，DNS 固定用 UDP 的 port53）

e) [2 point] What is one reason why that transport protocol has been used instead of the other main one we've seen in class?

f) [2 point] What is another reason?

(e and f) Some acceptable reasons:

- The DNS query and DNS response messages are short and they can fit in one UDP segment. 1 point if only say “short”. 1 point if only say “fit in”
- By using UDP, we avoid the connection setup overhead of TCP, which involves a 3-way handshake that requires an extra round-trip time (RTT). 1 point if only mention no 3-way handshake, but no explanation on the specific advantage of it

- For DNS, decreasing request latency to improve performance is considered an acceptable trade-off for the added complexity of having to deal with packet loss at the application layer. 1 point if only mention low latency
- Not having to keep connections is an important advantage in terms of reducing load on nameservers. 1 point if not provide the specific advantage of connectionless. 1 point if mention reducing load, but with trivial/unclear reason
- No need to perform congestion control for such a small and short communication.1 point if only say congestion control costs more (delay etc.)
- No point if only say low overhead or equivalent, since it’s too general
- No point if say shorter header since it’s trivial
- No point if say UDP suffices since TCP is also sufficient

(2 points per reason).

g) [2 points] If the server you mention in a) performs caching, you should receive an immediate response. Does this application protocol use conditional requests to check for possible IP updates? If so, why are conditional requests a good solution (what benefit do they provide)? If not, mention the correct mechanism for caching and explain why it is better than conditional requests.

No, DNS uses Time To Live as the main caching mechanism. If the local DNS server made a conditional call every time it had a request, since the amount of information request (= an IP address) is small, making a request just to make a "conditional check" (like conditional GET with proxies) would take as much time as if the local DNS server hadn't the IP address in cache and was requesting the IP address. Hence DNS caching would have no sense.

(0.5 for saying no) (no point for this question if say yes or the answer is unclear)
(0.5 for mentioning TTL or similar description)
(1 point for explaining that conditional GET wouldn't make any sense in this context,
0.5 point if only say conditional GET can increase the delay compared with TTL etc.
0.5 point if say the cache doesn't change often
No point if say the cache doesn't change at all)

h) [3 points] If no results are cached, what would be the chain of servers that are contacted (not necessarily by your client) to get the IP address corresponding to https://www.thedeepestsite.com?

(local nameserver), root nameserver, .com TLD nameserver, authoritative nameserver of thedeepestsite (1 point each)

i) [2 points] You land on a page that looks a lot like thedeepestsite.com, except that you are immediately prompted to enter your credit card information to access the page. That is fishy and very unusual. Further investigation with Columbia's IT department shows that a phishing IP address was returned by the server you mentioned in question a). Name one attack that might have caused this to happen.

Some acceptable reasons:

- DNS server impersonation,
- DNS server cache poisoning of the local nameserver (acceptable too: poisoning of another nameserver).

(2 points for either of the above, or another reasonable answer)
(1 point if only say redirection attacks, since we have many kinds of redirection attacks)
(DDoS doesn't apply since you can get the webpage)

j) [1 point] What makes DNS vulnerable to the attack that you mentioned?

Absence of authentication.
Absence of encryption is accepted for man-in-the-middle attacks
0.5 point if only say there is security issue, need to give the specific issue.
0.5 point if only explain what the attackers can do

2. Socket Programming [16 points, part a - g]

Download the Python programs TCPClient, UDPClient, TCPServer and UDPServer (from Piazza) on your local machine.

a) [2 points] Run TCPClient, input a sentence and then run TCPServer. What happens? Why? (In this and the following cases, if an error occurs that prevents you from completing the steps, explain the error and what caused it)

a) TCP is connection oriented and a connection needs to be established between the server and the client before any communication can take place. Because the server wasn't running when the client tried to connect to the server no connection was established, the message won't send and the code will throw a connection refused error.

b) [2 points] Run TCPClient, then run TCPServer and then input a sentence on the client. What happens? Why?

b) Same as answer to part a.

c) [2 points] Repeat part i using UDP. What happens? Why?

c) UDP is connectionless and because of that when you run UDPClient first, it doesn't throw an error as it doesn't try to establish a connection with the server. After you input the sentence, it sends it to the port the server is supposed to be running on, but it isn't running at the time. Because of that the client never receives a response back and hangs
The above behavior is for Linux/Mac. On windows, it will get ConnectionResetError on socket.recvfrom() in c) and f).Both will receive full credits.

d) [2 points] Repeat part ii using UDP. What happens? Why?

d) Because UDP is connectionless it doesn't matter if you run server or client first. As long as both are running at the time of sending the message, the message gets delivered to the server successfully and the client gets a response back.

e) [2 points] For TCP, what happens if you run the server on port N and the client tries to send a message to port M? Why?

e) For both TCP and UDP this will result in failure. For TCP, it hasn't established a connection with the server.

f) [2 points] For UDP, what happens if you run the server on port N and the client tries to send a message to port M? Why?

f) For both TCP and UDP this will result in failure. For UDP, the client will hang after sending the message as it doesn't receive a response back. (above is the answer using our UDPClient example, it's enough if mention 'the server cannot receive the message')

For e) and f), the answers can be out of the specific TCPServer and TCPClient or the result of running our TCPServer and TCPClient examples. Partial credit: For f), will get 1 point if pointing out "the client can send the message" without mentioning "the server cannot receive the message".

g) [4 points] Answer the following questions on sockets:

i. [2 points] Why does a TCP server need more than one socket?

A TCP server needs a welcoming socket to establish connections and uses another socket for sending and receiving messages.

ii. [1 point] How many sockets does a TCP server need for n simultaneous connections, each coming from different users?

n + 1

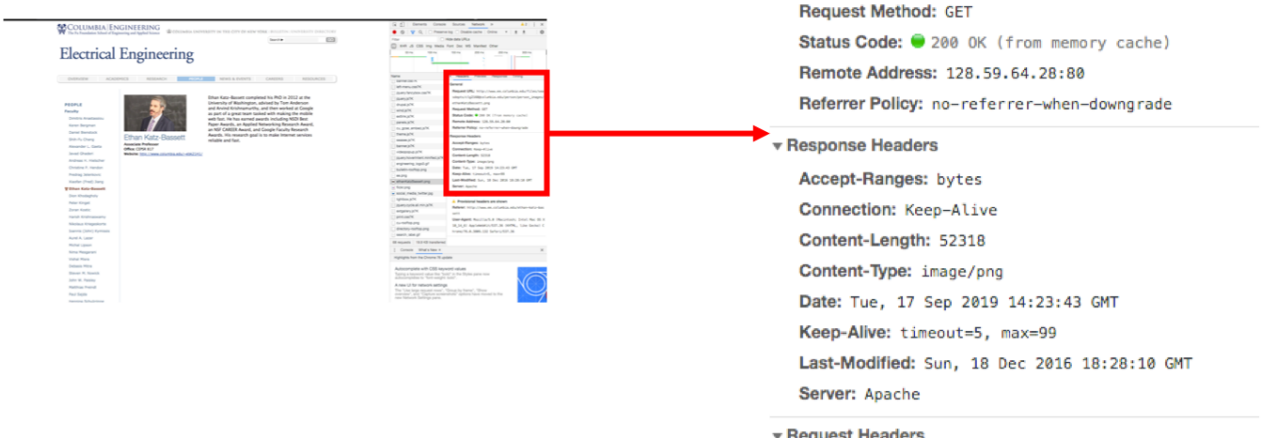
iii. [1 point] Why is only one socket enough for a UDP server?

3.

A UDP server only needs one socket to send and receive messages.

Using Browser Developer Tools to Understand the Web [16 points, parts a - h]

One good way to inspect networks is to use developer tools provided by browsers. Below is a screenshot of Google Chrome's developer tool capturing HTTP messages. Let's look at one specific request asking for Prof. Katz-Bassett's photo. Answer questions a - e.



a) [1 point] What is the host name of this request?

Hostname: www.ee.columbia.edu
1 point for answer

b) [1 point] What is the path name of this request?

Path name:
/files/seasdepts/clg2168@columbia.edu/person/person_images/ethanKatzBassett.png
1 point for answer

c) [2 points] Does the browser request a persistent connection or non-persistent connection? How do you know it?

Persistent connection. Because in response Connection: Keep-Alive.
1 point for answer, 1 point for explanation

d) [1 point] If someone pings ee.columbia.edu, what IP address is going to respond?

128.59.64.28
1 point for answer

e) [2 points] Explain what "Keep-Alive: timeout=5, max=99" means in response header.

"timeout=5" says the server will allow an idle connection to remain open before it is closed in 5 seconds.
"max=99" says the server is willing to receive 99 requests on the current keep-alive connection, and after that this connection will be closed.

(ref: https://tools.ietf.org/id/draft-thomson-hybi-http-timeout-01.html#rfc.section.2)
1 point for each answer
0 point for missing "idle"
0 point for "receive 99 resources", because a "resource" is not clearly defined, and the number of requests may not equal the number of resources

Now it's your turn to play with developer tools. A good tutorial on how to get started using Google Chrome: https://developers.google.com/web/tools/chrome-devtools/network. If you are using other browsers (e.g. FireFox, Safari, Edge, IE, etc.) you can also easily find tutorials pretty much alike.

(**Important**) Different browsers may have different behaviors, so please use **the latest version of Google Chrome** to complete this problem. **Clear your browser cache** first to make sure you get expected results. Open the homepage of Columbia University (www.columbia.edu) in your browser. Answer question f - g.

f) [1 point] What is the server's IP address?

128.59.105.24

1 point for answer

g) [2 points] Which request gets the base HTML file for the page? (answer by giving the request URL). How do you know that is the correct request?

https://www.columbia.edu/

The browser first needs to get the **HTML file**, analyze it and then requests other files needed to render this page. Thus it is the **very first request** sent to the server. We can inspect the response of this request to verify this.

h) [6 points] Clear your browser cache first to make sure you get expected results. Try visiting the following address now: http://columbia.edu. Give the status codes for all requests until the base HTML document is loaded, and a short explanation for each of these code

Status Codes:

i) 301- Moved Permanently - redirect from columbia.edu to www.columbia.edu.
ii) 302- Found - change to a secure connection (http -> https).
iii) 200 - OK - successful GET request for the base HTML.

1 for code, 1 for explanation x 3.

-1 for mentioning other codes loading beyond the base HTML page.

4. Video Streaming and Content Delivery [36 points, parts a - n]

a) [2 points] Why do many popular stored-video streaming services use TCP or other transports with reliable delivery?

In stored-video streaming, video can be buffered for better user experience. With guaranteed delivery and congestion control, TCP meets this need better.

+2 for mentioning buffering or preloading

+1 if only mentions reliability. This is insufficient because it doesn't explain why stored-video streaming specifically favors TCP. It could have used UDP just like live-streaming.

+1 if mentions advantage of TCP over UDP such as in-order/guaranteed delivery

b) [2 points] Increasingly, YouTube and other streaming services use QUIC, which is UDP-based but adds TCP features like reliable delivery, connection-oriented, and congestion control. What is a primary advantage of providing TCP-like features on top of UDP, rather than using TCP directly? (Note: We expect that you will have to look up QUIC to learn about it for the answer. Please feel free to do so, but remember to cite your sources appropriately)

It can be evolved in the application, rather than requiring upgrades to the kernel, which is much harder.
Multiplexing prevents head-of-line blocking.

+2 for either reason above
+2 for other reasonable explanations
+1 for acceptable explanations that are not major concerns (such as reducing handshake time, which is a minor advantage.)

c) [1 point] Consider a video stored in the server has a length of 60s. This video is divided into 30 video chunks, and each chunk has a size of 0.5MB, 1MB, or 2MB, depending on the selected resolution. No other compression methods are used. A user wants to stream this video on a browser. How does the browser learn how this video is divided, if it has no cache records of this website?

It needs to fetch a manifest file in the beginning.

d) [2 points] If DASH is used, what is an advantage of dividing the video into a fixed number of chunks (30), rather than chunks of a fixed size (say, 60 1s chunks at the highest resolution, 30 2s chunks at the medium resolution, and 15 4s chunks at the lowest resolution)?

When switching among different resolutions, it avoids issues with synchronization.
It is more responsive to throughput changes, since it has lower measurement intervals in general.
Less buffering required in bad network conditions.

+2 for either reason above
+1 for “avoid extra calculation” or “avoid extra encoding”

e) [2 points] If this webpage only has an HTML page plus the video, (in 30 chunks), starting from entering the website’s URL, how many HTTP GET requests are needed to fully load this video?

32. HTML file + manifest file + 30*video chunks.

+2 for 32
+1 if one missing

Suppose YouTube wants a viewer to be able to switch between high and low quality as conditions change. YouTube considers 2 options:

- **[option 1: chunks, like we discussed in class]**
  - break the video up into 30 second chunks, and player can only switch bitrates at chunk boundaries
  - Based on recent performance and/or amount of video it has buffered, player selects a chunk by picking the URL for it from a manifest and fetching it using HTTP
  - player cannot begin playing a chunk until the full 30 second file arrives;
- **[option 2: continuous ABR, not currently used by YouTube, but let's pretend]**
  - YouTube develops a new application protocol called *CABR* that encodes the video into *N* packets per second
  - if the player receives at least $M < N$ packets per second, it can play low quality. It does not matter which *M* it receives, but it has to wait until it receives at least *M* to play that second of video
  - if it receives all *N* packets per second, it can play high quality
  - if it receives more than *M* but less than *N*, the quality will be proportionally in the middle
  - Instead of sending one HTTP request for each chunk, the client sends a single *CABR* request for the video, using UDP.
  - Based on recent performance, the YouTube server picks how many packets to send (somewhere between *M* and *N*), then sends them using UDP.
  - Whenever it gets a packet, the client responds listing the packets it has received in the last second. (We haven't covered transport yet, but this is equivalent to the TCP overhead of **option 1**, so neither approach has an advantage in this aspect)
  - Depending on congestion and loss, the client may receive fewer packets than the server sent, but the server will only retransmit if the client receives fewer than *M* packets.

f) [4 points] What is a key advantage to using **chunks** versus **continuous ABR**? (1-2 sentences explaining what is the advantage and why it helps **chunks** versus **continuous ABR**)

Reliable delivery over TCP guarantees quality of video streaming in good network conditions. This method will not suffer from loss of packets (over UDP) with possible no retransmission (last bullet point in continuous ABR) meaning that quality may be poor even if network conditions are great.

g) [4 points] What is a key disadvantage to using **chunks** over **continuous ABR**? (1-2 sentences explaining what is the disadvantage and why it hurts **chunks** versus **continuous ABR**)

Bitrate can only change at chunk boundaries making it slower to react to changing network conditions;
if network conditions are relatively bad it may spend a lot of time buffering before being able to play the next chunk. (bad experience for customer)

+4 for other reasonable explanation
+2 if disadvantage proposed but doesn't hold a strong ground

h) [3 points] Ethan used the Chrome Developer tools while watching Netflix. He saw content fetched from multiple hosts, including:
- `www.netflix.com` (the home page HTML)
- `ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net` (chunks of video)
- `ae.nflximg.net` (icons and other small images)
Conduct research on how to use a network administration command-line tool called **nslookup** (https://en.wikipedia.org/wiki/Nslookup). Then use what you learn to obtain the A records of each of the three domains that Ethan found from Columbia's default local DNS server (i.e., when connected to a network at Columbia). Please include both command and query result in your answer.

i) [2 points] **nslookup** can also be used to do a reverse DNS lookup, i.e. find out the hostname of an IP address. What is the hostname of 128.59.64.28? Please include both command and query result in your answer.

j) [2 points] Use **nslookup** to do reverse DNS lookups on the IP addresses you found in A records for the three Netflix hosts that Ethan discovered (if nslookup returned multiple IP addresses in A records for a host, choose the first). For the first IP address, does **nslookup** return `www.netflix.com?` If not, explain why it returns what it does. Please include both command and query result in your answer.

```
$ nslookup 34.197.6.113

Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
113.6.197.34.in-addr.arpa       name = ec2-34-197-6-113.compute-1.amazonaws.com.
```

No. The web server is on Amazon Cloud. Or (partial credit) Netflix is using DNS redirection to send you to a specific server, so when you look up that server you get the specific hostname.

+1 Correct Answer
+1 Correct Explanation

Note that it should NOT be a lookup of 128.59.1.4 or other DNS resolver.

k) [2 points] For the second IP address, does **nslookup** return `ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net`? If not, explain why it returns what it does. Please include both command and query result in your answer.

l) [2 points] For the third IP address, does **nslookup** return `ae.nflximg.net`? If not, explain why it returns what it does. Please include both command and query result in your answer.

https://www.cdnperf.com/tools/cdn-latency-benchmark is a web interface that allows you to run ping tests on a hostname or IP address from various global locations in parallel. Run a ping test on `fp-afd.azurefd.net` in United States (under 'country' dropdown menu, select United States). `fp-afd.azurefd.net` is hosted on Azure Front Door, Microsoft's CDN that hosts `www.bing.com` and other services.

m) [4 points]
   a) Paste the screenshot of your ping result (map with latency measurements). What is the IP address of `fp-afd.azurefd.net` when ping is issued from New York and Seattle respectively? (If the webpage gives you results from multiple New York (or Seattle) locations, pick the one with the lowest latency out of all the New York (Seattle) results. If it doesn't give you any locations from that city, pick the one from the location nearest to New York (or Seattle) on the map.)
   b) Are pings executed from Seattle and New York probing the same physical server? Why or why not?
   c) What about Microsoft's deployment helps lower user latency?

n) [4 points] Which city is `fp-afd.azurefd.net`'s New York IP address located in? Which city is `fp-afd.azurefd.net`'s Seattle IP address located in? How do you get your answers?

Note: You can only use tools we present in this question(nslookup, cdnperf's ping, etc) to explain your answer.

1. Connectionless Transport: UDP [12 points]

a) [4 points] The book/lecture discussed how to calculate the UDP checksum by showing a simple example of summing three 16-bit words, without explaining exactly what words are used in actual UDP in practice. So in reality, what is the actual UDP checksum? Please explain how to derive the checksum value in the UDP header. (You should figure out what is needed when calculating checksum.)
(Hint: You can refer to RFC 768 to check out necessary details on the checksum that aren't in the book)

"Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header , and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets." [RFC 768]

For every 16 bit (word/row) take the 1's complement, add them, then take the 1's complement of the result

IPv4 Pseudo Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source IPv4 Address																															
4	32	Destination IPv4 Address																															
8	64	Zeroes								Protocol								UDP Length															
12	96	Source Port																Destination Port															
16	128	Length																Checksum															
20	160+	Data																															

You can refer to https://tools.ietf.org/html/rfc768 or https://en.wikipedia.org/wiki/User_Datagram_Protocol to see more details.

+1 If the pseudo header of IP (or the exact fields) is mentioned

+1 If UDP header (or the exact fields) is mentioned
+1 if the data/variable payload/data octets is mentioned
(give both points above if the student mentions UDP segments (not UDP data), because UDP segments contains UDP header and data)
+1 if explains add these 16 bit values together, then take the 1's complement of the result

b) [4 points] Suppose a source with IP address 104.17.119.38 wants to send a UDP segment to destination 172.217.11.36. Here UDP runs over IPv4. The UDP datagram transmitted is displayed as follows:

Source port: 4119	Destination port: 6713
Length: 47726	Checksum: ?
Payload	

To simplify calculation, assume values in the payload are all zeros. Fill in the checksum in this UDP header. Please show how you calculate the checksum step by step.

(Life is easier in hexadecimal)

4119 = 0x1017, 6713 = 0x1A39, 47726 = 0xBA6E, 104.17 = 0x6811, 119.38 = 0x7726, 172.217 = 0xACD9, 11.36 = 0x0B24
besides, UDP protocol number is 17=0x11

Sum source and destination port number: 0x1017 + 0x1A39 = 0x2A50;
Add length: 0x2A50 + 0xBA6E = 0xE4BE;
Add payload: 0xE4BE + 0x0000 + … (a bunch of zeros) = 0xE4BE;
Add IP addresses: 0xE4BE + 0x6811 + 0x7726 = 0x1C3F5 --> wrap around --> 0xC3F6
0xC3F6 + 0xACD9 + 0x0B24= 0x17BF3 --> wrap around --> 0x7BF4;
Add zeros and protocol number: 0x7BF4 + 0x0011 = 0x7C05;
Add UDP length: 0x7C05 + 0xBA6E = 0x13673 --> wrap around --> 0x3674;
0x3674 = 0011 0110 0111 0100 --> 1's complement --> 1100 1001 1000 1011 = 0xC98B.

The checksum is 1100 1001 1000 1011.

(solution in binary)

```
        0001 0000 0001 0111 (source port)
+       0001 1010 0011 1001 (destination port)
_____________________________________
        0010 1010 0101 0000
+       1011 1010 0110 1110 (length)
_____________________________________
```

```
        1110 0100 1011 1110
+       0000 0000 0000 0000 00... (payload, all zeros)
-----------------------------------------
        1110 0100 1011 1110
+       0110 1000 0001 0001 (104.17)
-----------------------------------------
      1 0100 1100 1100 1111
        0100 1100 1101 0000 (wrap around)
+       0111 0111 0010 0110 (119.38)
-----------------------------------------
        1100 0011 1111 0110
+       1010 1100 1101 1001 (172.217)
-----------------------------------------
      1 0111 0000 1100 1111
        0111 0000 1101 0000 (wrap around)
+       0000 1011 0010 0100 (11.36)
-----------------------------------------
        0111 1011 1111 0100
+       0000 0000 0001 0001 (zeros and protocol number)
-----------------------------------------
        0111 1100 0000 0101
+       1011 1010 0110 1110 (UDP length)
-----------------------------------------
      1 0011 0110 0111 0011
        0011 0110 0111 0100 (wrap around)
```

0011 0110 0111 0100 --> 1's complement --> 1100 1001 1000 1011
The checksum is 1100 1001 1000 1011.
+1 if fails to consider IP pseudo-header, only sums fields in UDP segment and the result is incorrect
+1 if not consider UDP header
+2 if gets correct checksum only for UDP segment, 0x1B41 = 0001 1011 0100 0001
+2 if sums UDP segment and IP pseudo-header, but the final result is incorrect.
+4 if sums UDP segment and IP pseudo-header, and finally gets the correct answer.
+2 if only gives the correct result without the process
+2 if only gives the process

c) [2 points] With the 1's complement scheme, how does the receiver detect errors?

The receiver should first add all the fields in UDP segment and IP pseudo-header together. If the sum contains a 0 bit, the receiver will know there has been an error.
+1 mentions adding all the fields together

+1 error occurs when any non-one digits appear in the sum

d) [2 points] The UDP checksum provides for error detection. Can the checksum fail to catch any errors? If so, please provide a concrete example (i.e., which fields are changed to what values), such that, if the receiver erroneously receives this UDP segment instead of the correct one presented in b), the checksum mechanism does not detect the errors, and explain why the errors are undetected. If not, explain why it catches all errors.

2.

For example, two bit errors that cancel each other out will not be detected. If the last digit of the source port is converted to a 0 and the last digit of the source IP is converted to a 1, the sum remains the same. In this case, checksum cannot detect the two-bit error.
+1 any tuple that has the same checksum with b) (other examples with values not in this question are also acceptable)
+1 correctly explains why the errors are undetected according to their given tuples

Connection-Oriented Transport: TCP [8 points]

a) [2 points] Host X is sending two TCP segments to Host Y over a TCP connection. The first segment has sequence number 80, and there is 1500 bytes data in the first segment.

i) If Host Y receives the first segment normally, what will be the ACK in the acknowledgement that Host Y sends to Host X?

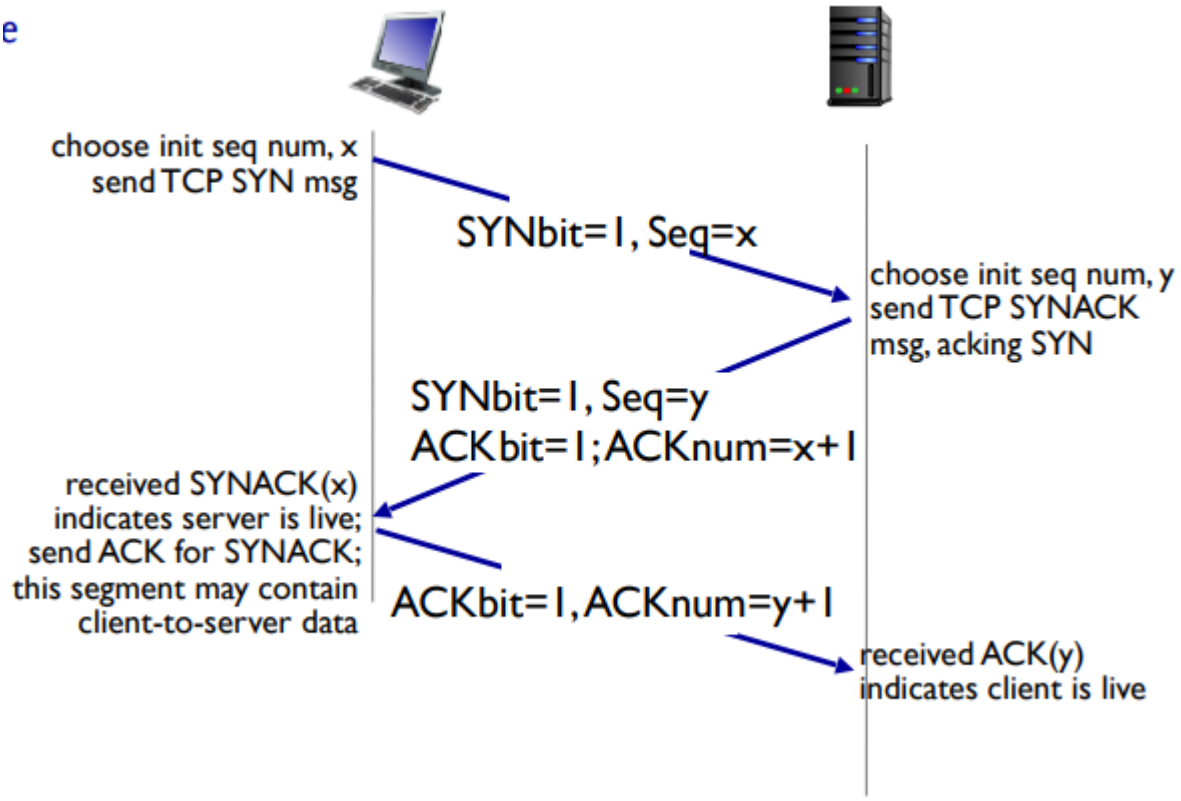
ii) If the first segment is lost but the second segment (also with 1500 bytes data) arrives at Host Y, what will be the ACK in the acknowledgement that Host Y sends to Host X?

ACK: 1580
ACK: 80
+1 for each correct ACK number

b) [3 points] Which fields in the TCP header are used (for connection management) in each handshake packet in order to establish a connection?

i) SYN, sequence number
ii) SYN, ACK, sequence number, acknowledgement number
iii) ACK, sequence number, acknowledgement number

e



（不确定答案和 ppt 哪一个对）

c) [1 points] Which behavior of a TCP sender is closer to Go-Back-N?

- Sender only keeps one timer.
- Cumulative ACKs

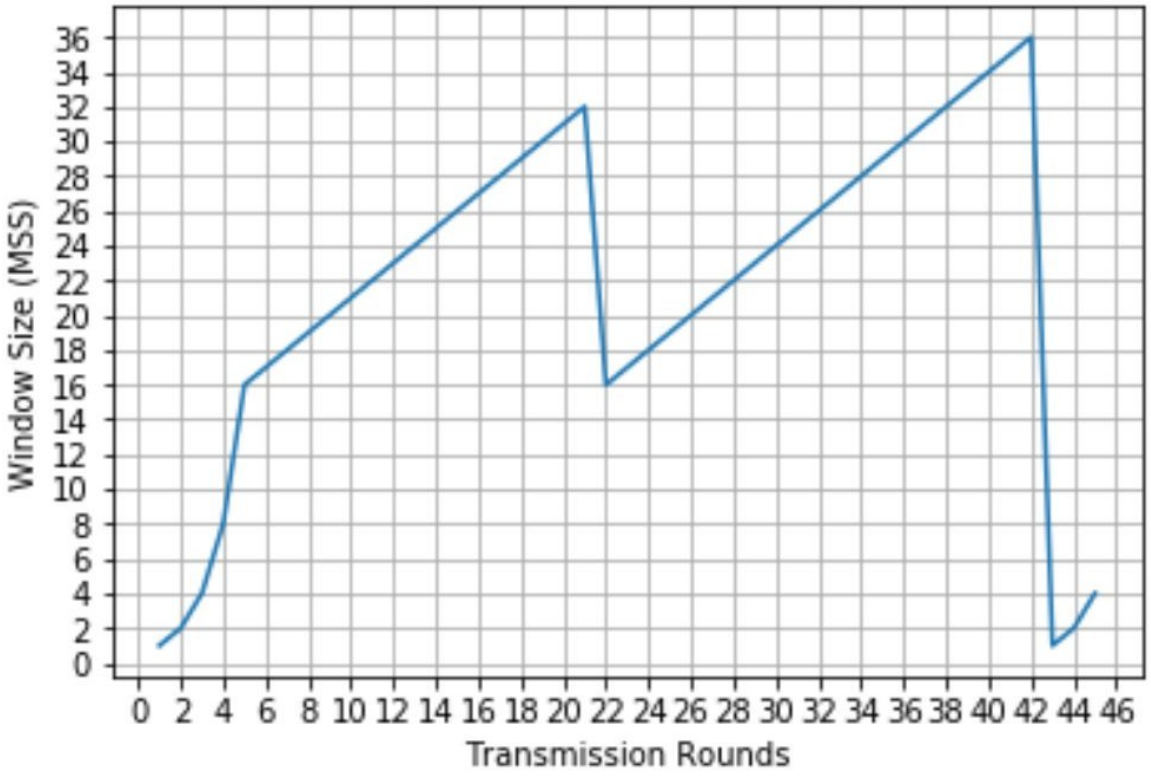
d) [2 points] Which behaviors of TCP is closer to Selective Repeat? Please answer 1 from the sender side and 1 from the receiver side.

3.

- Sender: Sender doesn't need to retransmit every unacked packet; fast retransmit
- Receiver: Out-of-order packets are buffered.

TCP congestion control [13 points]

Given below is a MSS vs Transmission round graph for a subset of a TCP Reno connection- the connection is active during intervals before and after the shown time period, so the connection establishment and termination is not shown. Answer the following based on the graph.



a) [2 point] Identify time intervals where TCP is operating in congestion-avoidance mode. Specify each interval by giving (starting round, ending round).

(5,21) [(6,22) also accepted] and (22 to 42)[(23,42) also accepted]
+1 for each window

b) [1 point] What is the approximate *threshold(ssthresh)* of the congestion window size value at the start of the first transmission round?

16 MSS

c) [2 points] Describe what event(s) trigger what happens at Transmission Round **21**. Does this event definitely indicate that one (or more) of the sender's packets was lost?

Indicates a case of three duplicate ACKS, where not all of the sent packets were received. Does not definitely indicate a loss.
+1 for mentioning duplicate ACKs
+1 for No.

d) [1 point] Name the event which occurs at Transmission Round **42**.

Timeout
+1 for Timeout.

e) [3 points] Does this event definitely indicate that one (or more) of the sender's packets was lost? If not, please describe the other possible explanations.

No, it does not definitely indicate a packet loss. The timeout may have been due to an unusually high RTT for a packet sent in the previous window or a lost ACK from the receiver.
+1 for stating No.
+2 if valid reason stated

f) [4 points] If the RTT is 100 ms, how long (after round **42**) does it take for the current congestion window size to reach that of the ssthresh immediately before round **41**? Assume no duplicate ACKs or packet loss incurs after round **42**.

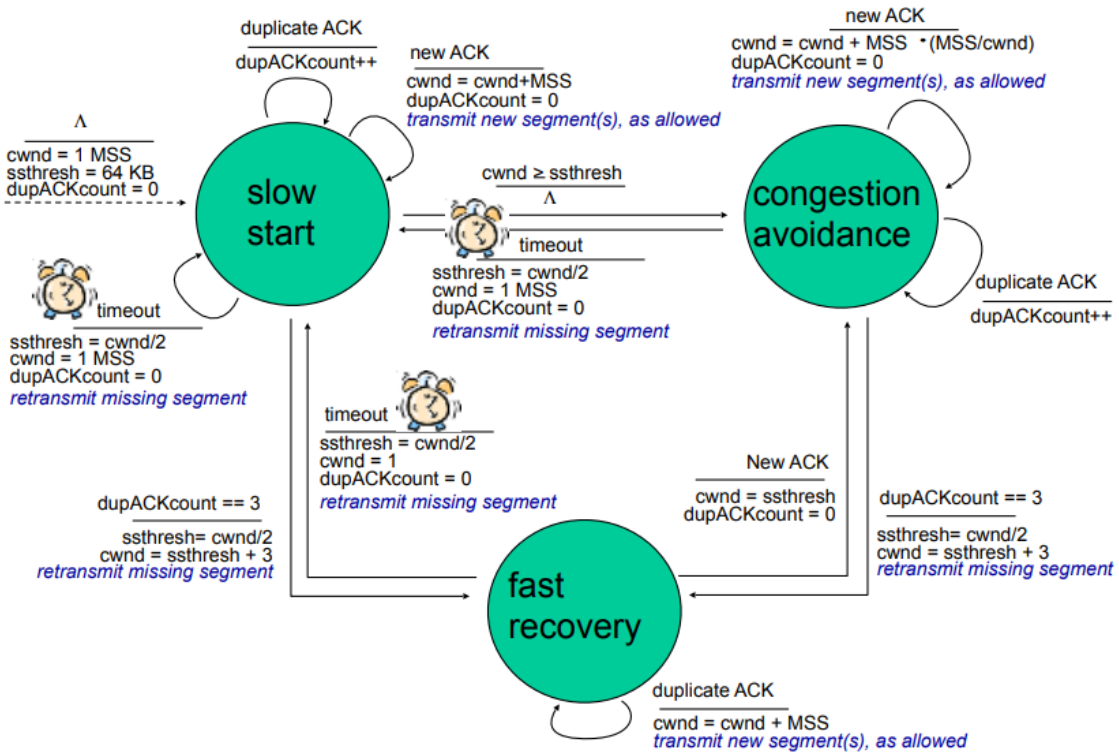
Old threshold value = 16 MSS - identified by looking at the size of the window immediately after receiving a dupAck at Round 19.

Size of window before loss = 36 MSS
New threshold = Size@loss/2 = 18 MSS

After round 42 (not included) Grows exponentially till 16 MSS = 4 RTTs.
Thus, total time taken = 4*100ms = 400ms

+1 for identifying new threshold.
+1 for identifying old threshold
+1 to calculate size for exponential growth
+1 to calculate time.

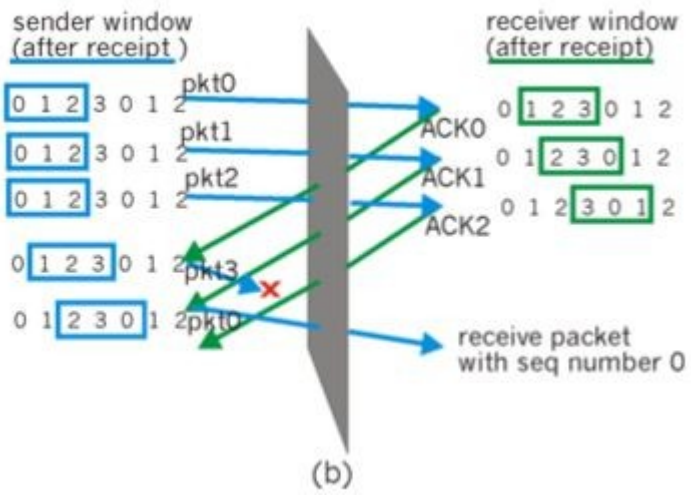
4.

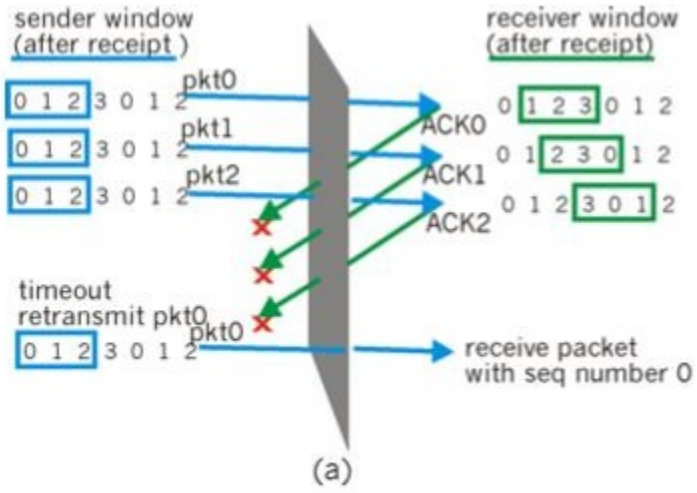


Principles of Transport and Reliable Data Transfer [22 points]

a) Consider SR and GBN protocols. Assume that the sequence number space for each protocol is of size n. Relative to this sequence space of size n, what is the largest allowable sender window that can be used safely while allowing the protocol to operate properly and to exchange messages without ambiguity, in the presence of loss and retransmissions for:
   a) [5 points] SR
   b) [5 points] GBN

Note: The example below shows a sender window of size 3, but your answer should be given in terms of the sequence space of size n. Explain each answer in a few sentences.




sender window
(after receipt)
0 1 2 3 0 1 2
pkt0
0 1 2 3 0 1 2
pkt1
0 1 2 3 0 1 2
pkt2
timeout
retransmit pkt0
0 1 2 3 0 1 2
pkt0
receiver window
(after receipt)
0 1 2 3 0 1 2
ACK0
0 1 2 3 0 1 2
ACK1
0 1 2 3 0 1 2
ACK2
receive packet
with seq number 0

(a)

b)

To avoid both scenarios, we want to avoid a situation where the leading (rightmost) edge of the receiver's window wraps around the sequence number space and overlaps with the trailing (leftmost) edge of the sender's window, as that will cause the receiver to read an old packet as a new one (since they have the same sequence number).

SR: The issue here is that when the sender resends packet 0 after failing to resend it previously, the receiver's window contains sequence # 0 for the next iteration (the n+1th packet), so it reads packet 0 as a new packet instead of the old packet. So, in order to prevent this from happening, the maximum window size must be **n/2** so that at most, both windows will cover up to n sequence numbers, so there will be no overlap with the next iteration of n sequence numbers.
+2 for correct answer (n/2)
+3 for explanation of how overlapping sequence numbers affect SR

Can get full credit if the answer and reasoning are correct. The reason doesn't have to be the same as above, but it needs to explain how overlaps can happen with window size larger than n/2. Will take 2 points off if the reason is not specific enough. (e.g. if your reason of a (i) can be directly used in a (ii), then it's likely to be not specific enough)

GBN: The situation is slightly different for GBN. Because GBN cares about the order in which the packets arrive (i.e. if packet *expectedseqnum* has not been received, it disregards all packets with sequence numbers > *expectedseqnum*), we only need to protect ourselves against the case where the first packet in the sender's window and the receiver's *expectedseqnum* have the same sequence number. This will only happen if the window is of size ≥ n, so the largest allowable window will be **n - 1**.
+2 for correct answer (n-1)
+3 for explanation of how overlapping sequence numbers affect GBN

Can get full credit if the answer and reasoning are correct. The reason doesn't have to be the same as above, but it needs to explain how overlaps can happen with window size larger than n - 1. Will take 2 points off if the reason is not specific enough. (e.g. if your reason of a (i) can be directly used in a (ii), then it's likely to be not specific enough)

Answer True or False and provide a brief explanation for your answer.

i) [3 points] With the SR protocol, it is possible for the sender to send a packet that falls outside of the receiver's window.

True. Take the following case:

1. Sender sends packets 0, 1 to receiver at t = 0s.
2. Receiver receives packets 0, 1 at t = 1s and sends ACKs 0, 1 to sender. It advances its window to [2, 3].
3. ACKs do not arrive at the sender, so it resends packets 0, 1.
4. Receiver receives packets 0, 1, which are outside of its window [2, 3].

+1 for correct answer

+2 for correct explanation (partial credit:+1 mostly correct, but lack contexts about why proposed situation happens)

ii) [3 points] With the SR protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

True. Take the following case:

1. Sender sends packets 0, 1 to receiver at t = 0s.
2. Receiver receives packets 0, 1 at t = 1s and sends ACKs 0, 1 to sender.
3. ACKs are delayed in arriving, so sender times out and resends packets 0, 1 to receiver at t = 3s.
4. Receiver receives packets 0, 1 again at t = 4s and resends ACKs 0, 1 to sender.
5. Sender finally receives the delayed ACKs 0, 1 and advances its window to [2, 3].
6. Sender then receives the new ACKs 0, 1 that were resent by receiver, which are outside of its window.

+1 for correct answer

+2 for correct explanation

iii) [3 points] With the GBN protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

True. Identical example as in SR.

+1 for correct answer

+2 for correct explanation

iv) [3 points] With the GBN protocol, it is possible for the sender to have a window that is further advanced than the receiver's *expectedseqnum*. Assume that the maximum window size is what you calculated in part (c).

False. The sender in GBN must receive an ACK from the receiver before advancing its window, so it can only remain on par with or behind the receiver.

+1 for correct answer

+2 for correct explanation

5. Alternatives to loss-based TCP congestion control [30 points]

a) [4 points] The congestion control we have talked about in class are loss-based, relying only on indications of lost packets as the signal to slow down. [It may be easier to answer this question after considering the followup questions below on alternate

approaches]

i) What is the main reason that loss can be too aggressive of a signal, and why is it too aggressive? [By too aggressive, we mean that there was loss, but the sender should not have slowed down.] For example, you may consider common scenarios in which loss-based congestion control performs poorly. Give one example.

Scenarios where slowing down won't reduce chance of loss are to be quoted.

1. "In shallow buffers, packet loss happens before congestion. Thus, loss-based congestion control can result in abysmal throughput because it overreacts, halving the sending rate upon packet loss, even if the packet loss comes from transient traffic bursts (this kind of packet loss can be quite frequent even when the link is mostly idle)"-quoted from https://cloud.google.com/blog/products/gcp/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster
2. Congestion loss can also happen when the sender is not driving congestion- there might be a third party throttling the network with other data.
3. Overcompensates for random/spurious loss of packets (eg: radio interference).

+1 for reason (e.g. stating loss before congestion)
+1 for example

There can be some other examples, but the examples have to be where packet **loss** actually occurs. Examples where loss doesn't really happen are wrong in this question. For example, a common incorrect example is the case where RTT gets long (but loss doesn't occur), then TCP thinks loss happens and behaves aggressively . Will take one point off for such examples even if other reasonings are correct.

ii) What is the main reason that loss can be too conservative of a signal, and why is it too conservative? [By too conservative, we mean that there was no loss, but the sender should have slowed down.] For example, you may consider common scenarios in which loss-based congestion control performs poorly. Give one example.

Scenarios where latency is high despite no loss are to be considered.

1. “In deep buffers, congestion happens before packet loss. Loss-based congestion control causes the infamous 'bufferbloat' problem, by repeatedly filling the deep buffers in many last-mile links and causing seconds of needless queuing delay.” quoted from https://cloud.google.com/blog/products/gcp/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster

   +1 for reason (stating high latency despite no loss)
   +1 for example

We will now consider alternate approaches.

TCP Vegas is a TCP congestion control protocol developed by Arizona State University in 1994. It operates based on packet delays as a measure of network performance instead of packet loss- i.e., it uses RTTs (roughly, from when the sender sends a packet until the packet is ACKed) to estimate the level of congestion of a network, reducing its sending rate if RTT increases, unlike Reno which uses packet loss as its sole signal to reduce.

b) [2 points] Why can RTT be used to detect if a network is congested? In other words, why does an RTT increase suggest congestion?

When the network is congested, the buffers on the intermediate routers tend to fill up. This causes a queueing delay, which in turn results in an increased RTT. Thus, by monitoring RTT, we can estimate if the network is congested or not.

+1 for congestion causes queueing delay
+1 for queueing delay increases RTT

c) [4 points] Compare TCP Vegas with Reno.
   i) What is a key advantage of Vegas over Reno?
   ii) What is a key disadvantage of Vegas compared to Reno?

Advantage: Earlier congestion detection- might be able to avoid loss of even a single packet by accurately estimating RTT.

Vegas never throttles the connection like Reno does (by 2, or down to 1 MSS)- in a network that has a very low congestion threshold, Vegas can theoretically maintain a higher average throughput than Reno.

Disadvantages:
The available bandwidth is not used to its full potential- filling of buffers can cause a higher RTT which results in slowing of sending rate despite no congestion in the network.

Will not perform well if used together with other protocols.

Vegas entirely depends on accuracy of RTT estimations if the BaseRTT is set too low or too high, Vegas will perform poorly.

Rerouting- Vegas will assume a higher RTT as congestion when the route of the packet changes- till it catches up, throughput is low.

+2 for one advantage
+2 for one disadvantage

d) [2 points] Will connections using Vegas and connections using Reno compete fairly with each other?

Fairness: Since Vegas detects congestion earlier than Reno, Vegas will impose a bandwidth restriction prior to that of Reno. Therefore, connections using Reno will occupy more bandwidth than those with Vegas. Thus, the resultant combinatory network cannot be termed “fair”.

+1 for stating coupled network is unfair
+1 for reason (mention Bandwidth, or other reasons)

Google’s QUIC and TCP implementations (sometimes) use BBR(https://groups.google.com/forum/#!forum/bbr-dev) congestion control algorithm. Read the reference (or others online) and answer the following questions. Be sure to cite your sources.

e) [4 points] In steady stage, as we discussed in class, TCP Reno decides what rate to send based on additive-increase, multiplicative-decrease of the congestion window, increasing until loss and then decreasing. In steady-state, what two main factors does BBR use to decide the rate to send at? (You should consider the simpler v1 of BBR from 2016/2017, which avoids complexities that have been added since).

BBR uses bottleneck bandwidth and RTT to decide its sending rate.

+2 for mentioning bandwidth
+2 for mentioning RTT

f) [4 points] How does this BBR approach avoid the disadvantages of loss-based congestion control from your earlier answer?

Maximum throughput and minimum delay correlates positively with bottleneck bandwidth.
By using maximum throughput as a factor in deciding the sending rate, BBR tries to identify (and is often successful) the optimum usage of bandwidth by throttling rates to that point by slowly increasing the data rate until max RTT is reached. This probing behaviour results in maintaining a steady state sending rate close to the bottleneck bandwidth. (Less conservative than Reno)
On the other hand, It will not interpret spurious loss as a signal to slow down if it is still possible to maintain the same throughput. (Less aggressive than Reno)

+1 for stating bottleneck bandwidth
+1 for stating minimizing delay
+1 for how BBR avoids back-off because of spurious loss unlike Reno (Reno is too aggressive)

+1 for how BBR avoids being too conservative- probing operations, and maintain steady state bandwidth

g) [4 points] How does BBR avoid disadvantages of delay-based schemes like TCP Vegas? Answer the question in terms of fairness in conjunction with other schemes.

i)

Delay based mechanisms throttle the connections before actual packet loss. For example, when coupled with Reno, Vegas would reduce the sending rate earlier than Reno, which in turn will reduce its throughput. At the same time, since the traffic has gone down, Reno increases its sending rate. Eventually, Reno dominates the connection. Overall, delay based mechanisms are too conservative when compared to loss based connections.

BBR overcomes this by projecting the optimal data rate and RTT in advance, and can thus push for a larger window than Vegas. This ensures that the bandwidth is close to optimal at all times. Thus, it is able to compete with highly aggressive schemes like Reno, leading to a fairer network distribution.

+1 for stating Loss based dominates over delay based schemes
+1 for reasoning the above-comparing the disadvantages of Vegas over Reno using a specific situation.
+2 for how BBR solves

[4 points total] For our purposes, a middlebox is an intermediate node between the endpoints of a flow that understands TCP. ECN introduces incompatibility issues with some middleboxes that do not understand ECN. For each of the items below, what (if any) negative effect can the behavior have on the TCP connection? How (if at all) will the sender and receiver react?

i) The middlebox always clears the ECE bit after the initial handshake.

The sender does not receive explicit congestion indications and cannot adjust the transmission rate before a loss happens. Relying on packet loss only as congestion indicator reduces performance (increased round trip times, lower throughput after loss happens).
+2 for mentioning the erasure of ECE bit makes that the sender can not take advantage of ECE

ii) The middlebox always clears the CWR bit after the initial handshake.

The receiver does not learn about the sender having reduced its congestion window and will therefore continue to enable the ECE bit. This results in the sender further reducing its congestion window, reducing throughput.
+1 for mentioning receiver doesn't know the sender's congestion window is reduced

+1 stating sender keeps reducing congestion window.

j) [2 points] Due to the possibility of middlebox interference, ECN is not enabled by default by many operating systems. However, protocols like DCTCP, a TCP variant designed to run in a data center, rely on ECN. What is a possible reason for ECN being more suitable in a data center environment compared to the whole Internet?

Data center operators control the whole infrastructure and can therefore guarantee that all middleboxes (if used at all) are compatible with ECN.

DCTCP when run along with TCP (or other non AQM connections) will be starved of bandwidth- most connections over the internet use TCP as a default setup. (Ref: https://ieeexplore.ieee.org/document/7063495)

+2 for one reason

6. TCP: On Elephants and Mice [11 points]

An elephant flow is an extremely large (several MBytes) TCP flow between a sender and a receiver. In contrast, mice flows are small (tens of KB, say) TCP connections. Even though most flows are short mice, the small number of elephant flows often occupy a disproportionate share of the total bandwidth on a link.

a. [1 point] Give one example of an application/application level protocol/Internet use case that generates an elephant flow.

Video OR large file ftp OR other large transfers
+1 for mentioning a large transfer
No credit for simply saying 'something large'

b. [1 point] Give one example of an application/application level protocol/Internet use case that generates a mouse flow.

Loading HTML OR email OR other small transfers
+1 for mentioning a small transfer
Partial credit for things that might be small transfers, but with no clear justification (video games)
No credit for simply saying 'web browsing' since Youtube is web browsing & tons of pages have large video ads. Anything suggesting the website or content in it was small, however, was given (at least some) credit.
No credit for saying HTTP, since any file can be fetched via HTTP

We will now consider how TCP treats mice relative to elephants. When we discussed fairness in class, we assumed that both flows were large (elephants). Now we will consider whether TCP is fair when elephants and mice compete.

c. [3 points] Give one way in which TCP (inadvertently?) favours elephants over mice. Explain in terms of specific aspects of how TCP works. [Hint: you might consider one sender sending an elephant flow and another sender sending a series of mice flows (in different connections) that combine to the same size as the elephant flow.]

d. [3 points] give a second way in which TCP favours elephants over mice. Explain in terms of specific aspects of how TCP works.

e. [3 points] give a third way in which TCP favours elephants over mice. Explain in terms of specific aspects of how TCP works.

1. Initial handshake contributes larger fraction to overall time cost for short flows, so they pay a larger startup overhead
2. Loss recovery can be more costly for short flows because of lack of established ACK clocking
3. Loss recovery can be more costly for short flows because of limited retransmission triggers, e.g. possibly not enough data for fast retransmits or forward retransmits
4. Loss recovery can be more costly for short flows because no RTT sampling and therefore starting with large initial RTO value
5. Initial slow start phase contributes larger fraction to overall time cost (need to grow congestion window)
6. parallel connections deal better with larger files.
7. TCP flows with large window sizes fill up router buffers, causing mice flows to have (on average) larger queueing delays than elephants. Related idea is that TCP is only fair in steady state, whereas mice flows only exist in the transient (unfair) state.

(c~e): providing a correct answer with vague explanation +1.5, if with correct explanation in terms of how TCP works +3
Mentioning header overhead is not a good answer, consider that a single flow with 1.4k-byte data is also a mouse. Only +1

CSEE 4119 Sample Midterm

Short Answer Questions (28 points)
Go-Back-N and Selective Repeat
1. (8 points) consider Go-Back-N and Selective Repeat. (1-2 sentences each)
a. What is an advantage of Go-Back-N over Selective Repeat?

Multiple correct answers (2pts)

- Only one timer
- Receiver does not need to buffer out-of-order segments
- For fixed sequence number space, GBN can transmit more efficiently with larger window size because it allows more unacked packets on the wire
- SR requires 2 * window_size <= sequence number size, GBN doesn't.

Partial credit (1pt)

- Simpler (without describing one of the ways above in which it is simpler)
- Simpler to implement (without more details including one of the correct answers)
- Less space in header of packet without explaining why.
- Cumulative ACK so fewer ACKs needed to re-ack successfully received packets.
- In GBN, the receiver doesn't need a sliding window (or only need a sliding window of size 1)

Wrong (0pt)

- In GBN, packets are guaranteed to receive in-order. (both GBN and SR deliver segments in order to the application)

b. What is an advantage of Selective Repeat over Go-Back-N?

Correct answers (2pts)

- Buffers out of order segments, so fewer retransmissions required

- (ok to call this more efficient or more resource efficient, as long as it says it is because it doesn't have to discard out of order segments

Partial answers (1pt)

- More efficient/less transmission without mentioning how SR deals with out of order packets using a receiver-side buffer.
- Only explains SR retransmits when necessary but doesn't mention this is done by having the receiver-side buffer or by not discarding the out-of order packets.

c. Are there ways in which TCP is more like Go-Back-N than like Selective Repeat? Answer Yes or No. If Yes, include a brief description of one way.

Correct answer (2pts)
Yes, mention at least one of the following key ideas:

- Smallest transmitted but unacked seq #: TCP sender need only maintain the smallest seq # of a transmitted but unacknowledged byte (and the seq # of the next byte to be sent)
- Receiver resends the smallest missing sequence number (duplicative ack for the oldest missing sequence).
- Both uses cumulative ACKs.

Partial answer (1pt)
Yes. Without explaining why.
Yes, but reason with in-order delivery: Selective repeat also has in order delivery.
Yes, TCP only maintains 1 timer.

Incorrect answer (0pt)
Answering no, regardless of reasoning.

d. Are there ways in which TCP is less like Go-Back-N than like Selective Repeat? Answer Yes or No. If Yes, include a brief description of one way.

Correct answer (2pts)
Yes, mention at least one of the following key ideas:

- Selective retransmission (skipping retransmitting segments that have already been acknowledged by receiver, and only transmit one sequence)
- Selective acknowledgement (allowing TCP to ack out-of-order segments selectively with a buffer rather than cumulatively acknowledging the last received. Notice that TCP RFCs do not impose any rules to enforce how out-of-order packets are handled on the receiver side, but it's indeed the approach taken in practice for higher efficiency. Please refer to the textbook (ed7. Page 239) for a detailed explanation.)
- Receiver buffers out of order packets.

Partial answer (1pt)
Yes, without explaining or fail to explain why.

Incorrect answer (0pt)
Answering no, regardless of reasoning.

Conditional Get

2. (6 points) Answer the following questions in short sentences:

a. What protocol includes a Conditional Get?

HTTP

+1 for HTTP

b. What is the purpose of a Conditional Get?

The Conditional Get is a mechanism that allows a cache to verify its objects are up to date. +2

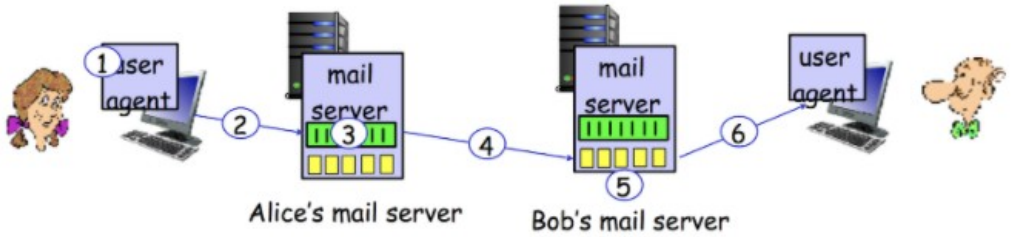
+2 for mentioning verifying object up to date, or if object has been modified.

c. How does Conditional Get work? Describe the pattern of communication between the entities involved.

After the user proxy sends a GET request with an IF-Modified-Since date to server. If the object on the server has not been modified since the specified time, the server sends Not-Modified message with empty body; otherwise sends the updated object to the proxy.
+1 for mentioning user proxy send GET with IF-Modified-Since;
+1 for mentioning when modified server sends the object
+1 for mentioning when not modified, server sends Not-modified.

Email Protocols

3. (6 points) suppose Alice is sending an email to Bob. Answer the following questions:



a. What is a protocol commonly used in step 2?

SMTP
+2 for SMTP, +1 for TCP or DNS. If multiple protocols are written, both protocols must be correct to get credit.

b. What is a protocol commonly used in step 4?

SMTP
+2 for SMTP, +1 for TCP. If multiple protocols are written, both protocols must be correct to get credit.

c. What is a protocol commonly used in step 6?

POP3, IMAP, or HTTP
+2 for any of the above, +1 for TCP. If multiple protocols are written, both protocols must be correct to get credit.

Transport Protocols

4. (8 points) consider clients A and B, communicating with server 5 using either UDP or TCP (as specified below). Answer True or False for the following statements. If it's false, please provide a brief explanation with 1 or 2 sentences. Missing or wrong explanation won't receive any credit.

a. If using UDP, A and B must explicitly (i.e. call the bind function to) bind to a port. But if using TCP, they don't have to do that.

Answer: False. In both protocols, clients don't need to bind a port explicitly to communicate with the server.
+2; wrong explanation for one protocol -1;

b. S must explicitly bind to a port regardless of whether it is using TCP or UDP

Answer: False. In TCP, server can also use the same port like 80 or 8080 to serve multiple clients.

+2; wrong explanation for one protocol -1;

c. If using UDP, S can use the same port to communicate with A and B. But if using TCP, S can't use the same port to communicate to both clients.

Answer: False. In TCP, server can also use the same port like 80 or 8080 to serve multiple clients.

+2; wrong explanation for one protocol -1;

d. S can use the same socket to communicate with A and B, whether it is using UDP or TCP

Answer: False. In TCP, server would open a new socket for each client.

+2; wrong explanation for one protocol -1;

DNS (12 points)

Host A, B, and C connect to the Internet for the first time and must perform DNS lookups to find IP addresses for websites they want to access. Make the following assumptions in this problem:

- All hosts know the IP address of their local DNS server, and all DNS servers know the hostname and IP address of the next servers in the hierarchy.
- All three hosts are connected to the same ISP and use the same local DNS server.
- None of the hosts or servers have any additional information cached at the beginning. However, as queries are made, all results are cached and remain indefinitely.
- Hosts make recursive queries to their local DNS server, which makes iterative queries to other DNS servers in the network.
- DNS resolvers will respond to request with both a hostname and its corresponding IP address (both NS-type and A-type). The emphasis in this question is on the hierarchy of the DNS system rather than the exact structure of the messages sent.

(a) [4 points] Host A, connects to the Internet for the first time and tries to connect to www.columbia.edu and must perform a DNS lookup to find Columbia's IP address. List the different queries made, in order, between different hosts and DNS servers until Host A gets the IP address www.columbia.edu.

1) Host A queries local DNS server
2) Local DNS server queries root DNS server; root responds with name/IP for .edu TLD server
3) Local DNS server queries .edu TLD server; gets response with name/IP for Columbia's authoritative server
4) Local DNS server queries Columbia's server; gets response with name/IP for www.columbia.edu
5) Local DNS returns name/IP to Host A

For all parts, full credit given if student considers each query and response as two separate steps, as long as the order is still correct.

2 points: Correctly understood hierarchy and process flow
-1 point for confusing recursive and iterative process or not specifying sender
, -2 points for no indication of hierarchy

2 points: Correct identification of specific servers: root, TLD (.edu), authoritative (Columbia)
-1 point per missing step (up to maximum of -2)
-0.5 if TLD/.edu server described as something else (e.g. "regional server")
Full credit if other servers misnamed as long as function is clear (e.g. "global" instead of "root")

(b) [4 points] After, Host B then tries to access www.nyu.edu and performs a DNS lookup in the same fashion. List the different queries made for this process.

(i) Host B queries local DNS server
(ii) Local DNS server queries .edu TLD server (which was saved in its cache); gets response with IP for NYU's authoritative server
(iii) Local DNS server queries NYU's server; gets response with IP for www.nyu.edu
(iv) Local DNS returns IP to Host B

2 points: Identify saved step because of caching
+2 points if correctly skips root server
+1 point if some other step was skipped instead. Partial credit only given if a step was removed compared to student's answer in part (a).
2 points: Correct process flow chain.
-1 point for each missing step.
-1 for recursive instead of iterative flow

(c) [4 points] Then, Host C tries to access www.soIumbia.edu for the first time. List the queries made in this DNS lookup process.

(1) Host C queries local DNS server
(2) Local DNS server responds immediately with IP address from its cache

4 points total:
-2 for skipping local DNS server (i.e. student assumes Host C will have address already)
-1 for each unnecessary step

Utilization (8 points)
Suppose two hosts (source and destination) are directly connected using the stop-and wait protocol. The transmission rate is R. The propagation speed is V. The distance between the two hosts is D. The size of one data packet is F. The size of one ACK is F_{ack}. Assume that the packet processing time is negligible.
a. (4 points) Assume the sender starts a timer as soon as it finishes transmitting a packet. What is the minimum safe timeout interval t_{out} that avoids spurious timeout?

Propagation time:

$$t_{prop} = D/V$$

Time to transfer one ACK packet:

$$t_{I(ACK)} = F_{ack}/R$$

Ideal timeout interval counts from the time when last bit of packet is sent to the time when ACK of this packet is received at sender:

$$t_{out} = 2t_{prop} + t_{I(ACK)} = 2D/V + F_{ack}/R$$

If write $2D/V + F_{ack}/R$ for the final answer, +4;
If write $2D/V + F_{ack}/R + F/R$ for the final answer, +3;
If final answer is incorrect, but write propagation time = D/V and time to transfer one ACK= F_{ack}/R, +1 for each;
Otherwize +0.

(4 points) Derive the sender's utilization of its connection. Assume that packets (both data and ACK) never fail (no packet loss or corruption).

Time for one packet transmission $t_I = F/R$
Utilization:

$$utilization = t_I/(t_I + t_{out}) = F/R/(F/R + 2D/V + F_{ack}/R)$$

If write $F/R/(F/R + 2D/V + F_{ack}/R)$ for final answer, +4;
If final answer is incorrect but write time to transmit one packet is F/R, and $utilization = t_I/(t_I + t_{out}) = t_{activelyTrans}/(t_{activelyTrans} + t_{idlelyWait})$, +1 for each;
Otherwise +0.

File Segmentation (14 points, parts a-h)
Considering transferring a file of length L=1 MB from Host A to Host B. The path from A to B has two links, of the same transmission rate R=2 Mbps, and packet-switching is used.
(Ignore propagation and processing delays in this problem.)
a. [2 points] How long will it take to transfer this file from Host A to Host B, if sent as a single large packet?

a. Only one file, so there is no queuing delays.
10^6 * 8 bit / 2*10^6 bps * 2 (links)= 8 s

b. [2 points] If we divide the file into 1000 packets, how long does it take to move the first packet from Host A to Host B?

b. The queuing delay is 0 for the first transmitted packet.
10^6 * 8 / 1000 = 8000 bit/each packet
8000 / 2Mbps * 2(links)= 8ms

c. [2 points] Following b), the file is divided into 1000 packets, and suppose we generate these 1000 packets in one time. What's the average queuing delay experienced by the 1000 packets, considering all queueing they experience along the path? (Hint: Since 1000 packets are generated in one time, you can imagine 1000 packets arrive simultaneously at the first link.)

c. The queuing delay for the first packet is 0, 4ms for the second packet, and generally, (n-1)*4 ms for the nth transmitted packet. Thus the average delay for 1000 packets is
(4 + 8 + … + 999*4) / 1000 ms = 999*500*4 ms = 1998 s
(there is no queuing delay in the second link)

d. [2 points] Following c), the file is divided into 1000 packets and generated in one time, what's the transmission delay in total for all the 1000 packets?

d. It's the same as a), 8s

e. [2 points] Following b), the file is divided into 1000 packets, but for this time, the next packet won't be transmitted until the previous packet completely arrives at the switch between two hosts. How long will it take to move all the packets from Host A to Host B?

e. No queuing delay under this case.
The first packet will use 4ms*2 = 8ms to reach Host B. Then every 4ms a new packet will be received. 8ms + 4ms*999 = 4004 ms = 4.004 s

f. [2 points] Compare the result obtained in a) and e), please explain why the time cost differs.

f. The queuing delay in both cases is 0, and we only need to consider the transmission delay.
Because the switch uses store-and-forward packet switching method, for a), the switch cannot begin to transmit the file until it receives it fully over the 1st link, resulting in a long delay before the switch can begin transmitting even the beginning of the file.
But for e), after the first packet arrives at the switch, both of the links could be busy transmitting segments of the files at the same time, thus decreasing the delay.

g. [1 points] Name one advantages of file segmentation in the transmission. ("By segmentation," we mean the process of breaking up into packets, like in e).)

g. Reduce delay

h. [1 points] Please name another advantage of file segmentation.

h. Loss/bit tolerant: If one packet is lost or some bits are not correctly transmitted during the transmission, we needn't transfer the whole file.
If other answers make sense, it's all right.