Name: Tanjip Sumait Nokibn

1|ⓐ The layer responsible for hop-to-hop delivery is the Data Link layer. The main difference between hop-to-hop and host-to-host delivery lies in their scope and addresses they use. Data link layer uses physical address known as MAC addresses. which change at every hop. In contrast host-to-host delivery which is managed by network layer uses logical IP addresses, which remain the same from the start to end of journey.

ⓑ LAN delay = 50 ms, Access delay = 44 ms. Internet delay = 3000 ms
40% original server, 20% proxy one, 20% proxy two

Total delay = (0.4)×(3050 + 44) + 0.2 (50) + 0.4 (50)

= 1267.6 ms

2|ⓐ The reason is most likely that the cookies from my morning session were deleted or expired. This could have happened if I closed my browser and the browser itself set the cookies with a very short expiration time.

ⓑ For checking email on a mobile phone, IMAP is a much better protocol than POP3. This is because IMAP is designed to synchronize your emails and their status across multiple devices. When ~~I~~ read, delete or move an email on my phone, those actions are reflected on the server and will show up the same way when I check my email later on my laptop. POP3, on the other hand, is often set to 'download-and-delete' messages from the server. If I read an email on my phone, it might

be removed from the server and wouldn't be available on my computer.

(d)

(i) As the local DNS server had no info in its cache so it needs to contact the root, TLD and authoritative servers. So, total 3 RTTs. Then an extra RTT for the local server to send the answer back to my PC

$$\text{total RTT} = 26 \times 4 = 104 \text{ ms}$$

(ii) As non-persistent HTTP: each object requires 2 RTTs.

RTT client to server = 39 ms

RTT = 78 ms

For 30 objects = $2 \times 30 \times 78 = 4680$ ms

After including the DNS RTT = 4680 + 104 = 4784 ms

Also including server prep time = 4784 + 30×4 = 4904 ms

(iii) Total object download time = 30 × 215 ms = 6450 ms

So Total time to load the page = 4904 + 6450 = 11354 ms

3(a) This process is part of terminating a TCP connection, specifically a scenario known as a "half-close". This usually occurs when one side of the connection has finished sending data but the other side still has more data to send. In this case, the client sends a FIN flag to say it is done sending. The server acknowledges this with an ACK, but by not sending its own FIN back yet, it keeps its side of connection open to continue sending its remaining data to the client.

(b) When I open three tabs to the same website, the destination port for all tabs will be the same well-known port. However, the source port for each tab will be a unique dynamic port number assigned by my computer. All tabs cannot have the same source port number when connecting to the same server because the combination of source IP, source port, destination IP and destination port must be unique for each connection. If two tabs used the same source port to the same web server, the OS and the server would have no way to tell which data packet belongs to which tab, causing confusion and errors.

(c) (i) SN for server = 9856, DS1 = 951, DS2 = 478

So,

sequence number = 9856 + 1 + 951 + 478 = 11286

acknowledgment number = 1546 + 1 + 222 + 369 = 2138

(ii) For the rwnd of client,

In the scenario as client has not yet received Data segment 3. It will not recieve any of the segments after 3. so the segment 4 & 5 will be still in the buffer. So rwnd size after 5th data segment is = 1684 − 99 − 201 = 1384.

(iii) As the client not recieved data segment it will be waiting for the data- B 3.

So,

sequence number = 1546 + 1 + 222 + 369 = 2138

Acknowledgement number = 9856 + 1 + 951 + 478

= 11286

11(b) The DNS system helps to make streaming faster by redirecting nearby CDN server first sends and a request to resolve the websites name to an IP address. This request travels through the DNS hierarchy and eventually reaches the authoritative DNS server, The server looks at the IP of the local DNS it returns the IP address of a CDN cache server instead of central server, which reduces buffering and load times.

(c) The difference comes down to how the upload burden is shared. In a client server model a single server has to upload the entire file to every single user. As more users join, the servers fixed upload capacity is split among more and more people, causing the minimum distribution time to increase quickly and linearly. In a P2P model, every user who downloads a piece of the file also starts uploading it to others. So, as more users join, the total upload capacity of the whole system actually increases. While the time might still go up with more users, it increases at a much slower rate because the work is distributed among all the peers.