Assignment 2 for CNT 4700 Computer Networks: Internet Architecture & Application Layer
Leeson Chen - October 7th

Instructions: Be precise and to the point. Many questions require answers using a sentence or two). Some questions will ask you to elaborate, use visual aids or graphs, or show traces/code. Use your own words and phrases, do not copy from any other source.

Q1. DNS-related: Consider a scenario of a user browsing the web from the machine storm.cise.ufl.edu, accessing an article in a website at URL:
https://www.nytimes.com/2019/09/26/technology/ai-computer-expense.html
The user performs three accesses: - using http, - using https and - using port 8080.
A. Show the sequence of DNS servers queried to resolve the URLs (assume no caching)
The first query begins at the top-level, root DNS server. From there, because the New York Times has a dot com website, it queries to the second-level .com DNS server. The .com server queries to the specific .com, which is nytimes.com. Finally, within the nytimes.com website, it finds the specific path of the remaining URL.
root -> .com -> nytimes.com -> rest of URL

B. Write the complete URL for each of the three accesses [you may have to edit the URL]
http://www.nytimes.com/2019/09/26/technology/ai-computer-expense.html
(just http is less secure)
https://www.nytimes.com/2019/09/26/technology/ai-computer-expense.html
(https is more secure)
`ssh -L 22222: https://www.nytimes.com/2019/09/26/technology/ai-computer-expense.html:80`

C. Write the four tuple [src address, src port, dst address, dst port] for each of the accesses
From using Wireshark:
http: [192.168.0.102, 443, 64.233.177.121, 50153]
https: [192.168.0.102, 443, 151.101.205.164, 50073]
8080: [192.168.0.102, 443, 151.101.205.164, 8080]

Q2. Discuss how the following technologies (or their variations) help in improving the performance of content distribution networks (CDNs):
A. HTTP
HTTP stands for hyper text transfer protocol, the network's application layer protocol. It is a client server model, that is stateless and comes in persistent and non persistent forms. HTTP, and its secure variant HTTPS, are considered the standard protocols for accessing web content, including anything from news articles to forums to CDNs such as YouTube and Netflix. HTTP is the main protocol through which CDN video content gets streamed to users.

B. DNS

In a CDN, the content (video) is often stored in a different server than the one the user initially visits (see the slide example of net cinema website vs king CDN server). We will use two example URLs, movies.com and storage.com, where the videos are stored. The user would visit movies.com, but his local DNS server resolves to movies.com's authoritative DNS. This authoritative DNS returns the URL for storage.com, and the path to his movie. Then storage.com has an authoritative DNS, returning the IP of the server with his video. Finally, The user can directly request video from storage.com/path and stream his movie via HTTP.
We can see from this example that the content is often stored on servers other than the authoritative servers where user requests resolve. Multiple CDN servers will often contain the same copy of content.

Q3. Elaborate on the data 'push' vs 'pull' in the context of
A. http vs SMTP
HTTP "pulls" data, meaning that it has a constant steady connection wherein data is being downloaded or updated. Conversely, SMTP pushes data. The email client doesn't need a constant connection to the internet, as that would be costly and wasteful. Instead, it only needs to be notified of new emails, and download the text of that new email when it appears. From the mail server to the client, the server pushes new data and otherwise leaves the client alone.

B. peer-to-peer network hierarchy communication (e.g., super nodes, group leaders)
A P2P network with hierarchy communication would mean that among all the different peer clients, there are occasional super nodes which serve as directories, helping to connect user to user and establishing the P2P connection. Because these super nodes / group leaders must act as directories, they need to know a comprehensive list of all the users, or otherwise be able to refer to other super nodes which would complete the list. If new users join the network, their directory information must be added, which would serve as a push in this scenario. That information has to push to all relevant super nodes. A pull for this example would mean that a user is asking a super node where to find another user. The super node would process this request and return to the asker with the information to connect to their target.

C. Proxy and web caching
In the context of proxy servers and web caching, copies of frequently-requested data is stored at intermediate middlemen servers so that the main server is not congested and overwhelmed with clients from all around the world trying to access the same data. The copies have to be pushed to those proxy servers, and updated regularly in the case of data that may fall out of date. For example, the morning news should be pushed to all its proxy servers at least once a morning. The pull in this case would refer to clients accessing the data copies, and receiving the data via http.

D. CDNs
The CDN example is a little similar to the previous, wherein CDNs may frequently use proxy servers due to the high volume and large size of their data. A CDN may push copies of videos to proxy servers to avoid becoming overwhelmed at the main server. Here, pulling the data means users watching the videos by requesting the data from the CDN servers. Pushing may

not only refer to proxies, but also where users or the studio adds new content, such as new YouTube videos or Netflix shows. Those new videos would need to be pushed to the servers as well.

Q4. Someone suggested to use a local file called hosts.txt on each machine instead of DNS. Discuss the advantages (at least 2) and disadvantages (at least 2) of such suggestion.

The advantages of a hosts text file for each user would be that DNS lookup and name / IP resolution becomes moot. Instead of requiring the somewhat complicating process of querying a root DNS server then the following servers to access a website, the user could just get the IP from their text file. This would have the benefits of being time-saving, as well as lowering network congestion and usage for all the DNS queries. However, there are many obvious downsides of this approach. For starters, this text file would need to be massive in order to encompass all the different websites in existence. Secondly, websites disappear and new ones go up all the time. This text file would very quickly become out-of-date, not to mention out of sync with the millions of other text files around the world.

Q5. What is 'saw-tooth' behavior in TCP, and what is causing it?

Saw-tooth behavior in TCP is a product of a TCP congestion control method called additive increase multiplicative decrease (AIMD). In this approach, the sender increases its transmission rate (congestion window size / cwnd) by 1 segment every round trip time. The congestion window size gets larger and larger by additive increments, creating an up-right diagonal line on a graph of window size over time. The multiplicative decrease happens when TCP detects a loss, and the congestion window immediately cuts in half, causing a very sudden decrease and a straight-down vertical line on the cwnd vs time graph. These diagonal lines and vertical lines cause a pattern that looks like saw tooth blades.

Q6. A TCP flow and a UDP flow walk into a network, sharing a bottleneck link .... Complete the story, detailing the packet rate dynamics if the link gets congested, and comment on the end result.

Honestly, I'd love to make a cheesy entertaining story out of this question, but I'm really not sure how seriously I'm supposed to take it, so I'm going to err on the safe side and just answer it like a normal question. If both a TCP and a UDP connection are sharing a bottleneck link, the UDP connection will "overpower" the TCP because UDP lacks any built in congestion control. The TCP link will lower its usage as more and more UDP traffic appears. In this scenario we do not know if the connections are using Go Back N or selective repeat or stop and wait, or if it is the client server or P2P or hybrid architecture. We also don't know if the user is using HTTP, SMTP, or a different protocol, if they're streaming video from a CDN, etc.

Q7. TCP is supposedly fair, dividing the bandwidth between competing TCP flows. You want to transfer a huge file fast, suggest a way of doing so using TCP to get over the fairness delays, and approximate your new bandwidth share.

The TCP fairness goal is such that for K number of sessions over a TCP connection, with the whole link having a bottleneck link of R bandwidth, each session has an average rate of R/K. This essentially means that the TCP bandwidth is evenly divided amongst all sessions. However, this can be circumvented by opening multiple sessions for your single purpose, which tricks TCP into giving your application more connections than it should get per session. In this example, if you were to open up five additional sessions on top of ten existing ones, the new bandwidth allotment would be (R/(K+5))*5, which simplifies into you getting 5/15 connections or one third of the bandwidth.

Q8. Given that most data-link layers perform error checking (and correction to some extent) why do we need checksum in UDP (and TCP)?
In the diagram of the TCP segment structure and UDP segment structure, both segments contain a checksum of 16 bits, half the length of the 32 bit segment. The checksums are used to detect loss or corruption of data, such as bit flips. By checking the data integrity with checksums, it adds an additional, if not optional, layer of security to the security already present in the data link layer. Security must be built into an application, and redundant security is still a higher standard than less security. The data link layer is not infallible either.

Q9. Comment on response to packet-loss vs ack-loss in
A. Go-back-N
In the Go-back-N (GBN) approach, packets can be thought of as traveling in groups. The sliding window only allows for a certain amount of un-ACKed packets to be sent at a time. Once those packets have been sent, if the oldest packet hasn't received an ACK, new packets are not sent. On the receiver side, ACKs are sent for cumulative groups. The ACK is specifically for the highest sequence packet that was sent in order. E.g., if packets 3, 4, 5, and 6 were all sent cumulatively, the ACK 6 indicates all were received correctly.
In this approach, the loss of a packet has the same effect as a lost ACK. The lost packet means all further packets sent before the window stops will be considered out of order. The sender then times out, and will have to retransmit the packets within that window. If an ACK is lost, it has the same outcome: the oldest packet times out, and the whole window gets resent.

B. selective repeat
In the selective repeat approach, the sliding window keeps traveling forward, but ACKs are sent individually instead of cumulatively, and need not be in order. There are individual ACKs for each packet as well as individual timers. A lost packet will eventually time out and be resent. A lost packet also times out and gets resent. The sender's window will not slide forward any more past the oldest un-ACKed packet.