Fall 2022 CSEE4119 HW2 answer key

CSEE 4119 Fall 2022
Homework 2

Assigned: 10/6/2022
Due: 10/19/2022, 11:59 PM
Corrections:
2022/11/06: fixed incorrect answer for Q1c. Previous answer incorrectly counted the time for the client to transmit the requests for the last 7 objects, but the server can be sending the responses in parallel. It will take it longer to transmit each image than it will take the client to transmit the next request, and so we are gated on the time to transmit the images (the next request will always be waiting as soon as the server finishes sending an image).
2022/10/14: rephrased Q4, A4 & B4 to make them more clear. Previous phrasing may have lead you to an incorrect answer.
2022/10/13: fixed some misnumbering of questions
2022/10/06: No changes yet
Uploading course materials, including questions/answers from this homework, to sites such as CourseHero, Chegg or Github is academic misconduct at Columbia (see pg 10).

Q1 HTTP

Consider a 3×10^7-meter link connecting a client to a server, over which a sender can transmit at a rate of 1,000,000 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control messages (e.g., ACK, SYN) and/or requests are 2000 bits long (from HW1, you know that these data packets are longer than is typical on the Internet, but we are making a simplification here). To set up a TCP connection, a three-way handshake is a needed method that requires both the client and server to exchange SYN and ACK packets before actual data communication. (Don't worry about this if you don't know it, we will cover this later in class. Just assume that, before sending any data, the client sends the server a control packet, the server replies with a second control packet, and the client replies to that with a third control packet that also includes the request.) If the client/server use N parallel connections, assume each gets 1/N of the link bandwidth. Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long (so each can be perfectly fit into a data packet), and that the initial downloaded page (100 Kbits) contains references to 8 referenced objects. Assuming the signal propagation speed is 3×10^8 m/s, the RTT between client and server only depends on propagation delay, and connections can transmit at their share of the link bandwidth (without worrying about flow control or congestion control). Answer question a - e.
General Comment: please do not round your final answer! For example, as d_trans for 1 control packet is 0.002s, so rounding to the nearest tenth or hundredth does not provide enough precision to know you got the correct answer!

1. [3 points] How long will it take for the client to get the whole 9 objects from the server if using non-persistent HTTP with a single TCP connection? Please show your calculation and the final answer.

RTT is the time for a small packet to travel from client to server and back, since it only depends on propagation delay here, we can calculate by:

RTT = $2\times\frac{3\times10^{7}m}{3\times10^{8}m/s}$ = 0.2 s

The time to transmit a control packet is: $t_c = \frac{2000bits}{1000000bits/s} = 0.002\,(s)$

The time to transmit a single object is : $t_o = \frac{100000bits}{1000000bits/s} = 0.1\,(s)$

The time to set up a TCP connection and transmit a single object is thus:

t_s =2RTT + $3\times t_c$ + t_o= 0.506 (s)

We need to obtain the downloaded page and the 8 referenced objects:

T = 9 $\times t_s$ =9$\times$0.506 = 4.554 (s)

+1 point for RTT and +1 point for t_s

+1 point for final answer

2. [3 points] How long will it take for the client to get the whole 9 objects from the server if using non-persistent HTTP with enough parallel TCP connections to allow all requests in parallel? Please show your calculation and the final answer.

RTT = 0.2 (s)

The time to obtain the downloaded page is still t_s, after that we will open 8 parallel TCP connections to get the 8 referenced objects, and each TCP connection will have $\frac{1}{8}$the bandwidth which is $\frac{1{,}000{,}000bits/s}{8} = 125Kbits/s$

T = t_s + (2RTT +$8\times 3\times t_c + \frac{100{,}000bits}{125{,}000bits/s}$)

= 0.506 + (2*0.2 + 8*3*0.002 + 0.8) = 1.754 (s)

+1 point for $t_s$and +1 point for time to transmit the 8 referenced objects

+1 point for final answer

Common Mistakes

- Divided bandwidth 9 ways, instead of 8
- Didn't send HTML and do 3-way handshake first with full bandwidth

3. [3 points] How long will it take for the client to get the whole 9 objects from the server if using persistent HTTP and a single TCP connection? Please show your calculation and the final answer.

Edit: fixed answer:

1. Use the values for RTT, t_c, and t_o in the answer key for 1a. Let's call the one way delay OWD, and the time at the end of each of my numbered steps t_n.
2. The client finishes sending the SYN at time 0.002s (t_c). So t_2=0.002s.
3. The server receives the SYN at time t_2+OWD = 0.102s.
4. The server finishes sending the SYN-ACK at t_3+t_c=0.104s.
5. The client receives the SYN-ACK at t_4+OWD = 0.204s

6. The client finishes sending the SYN-ACK-ACK and the request for the HTML at time t_5+t_c= 0.206s.
7. The server receives the request at time t_6+OWD= 0.306s and begins sending the HTML.
8. The server finishes sending the HTML at time t_7+t_o=0.406s.
9. The client receives the HTML at time t_8+OWD=0.506s.
10. The client begins requesting the images and finishes the first request at t_9+t_c=0.508s.
11. The client then requests the other images, finishing requesting all of them at t_9+8*t_c=0.522s
12. The server receives the first image request at time t_10+OWD=0.608s.
13. It will take it longer to transmit each image than it will take the client to transmit the next request, and so we are gated on the time to transmit the images (the next request will always be waiting as soon as the server finishes sending an image). The server will finish transmitting the images at time t_12+8*t_o=1.408.
14. The last image will finish arriving at the client at time t_13+OWD=1.50+1 point for transmitting time of a single object

+1 point for the correct formula for T
+1 point for final answer
Common mistakes

- ~~Missed the 8 control packets send with each object~~

15. Did not use pipelining (e.g., had 8 additional RTT's)8.

-

4. [1 points] In this question, what is the percent speedup for persistent HTTP over the non-persistent case? Overall, are these gains significant?

Percent speedup = (1.522-1.754)/1.754 = -13.2269%
13.2269% speedup. Not significant (only sped up 0.2s)
Calculating speedup as 1.754/1.522 = 115% is also acceptable.
+.5 correct speedup
+.5 gains are not significant
Common mistakes

- Although the % speedup may be arguably "significant", the absolute speedup was only 0.2s which is not very much.
- Some students compared the non-persistent case with a single TCP connection (the slower dase) with the persistent case. While in the prompt we did not specify which case to use, we clarified this in an Ed post, and you should always make the comparison with the faster/more efficient option.

5. [2 points] Suppose the link length is now 3×10^{10}meter. Is there any significant gains using persistent HTTP with a single TCP connection? Prove your answer either by calculation or explanation.

RTT is now $2\times\frac{3\times10^{10}m}{3\times10^{8}m/s}$ = 100s
Yes +1 point
By arguing that in this case 2RTT >> RTT +1 point

6. [3 points] How long will it take for the client to get the whole 9 objects from the server if using HTTP/2 Server Push, which allows the server to send resources to the client before the client requests them? You may imagine that we can push all resources referenced in an HTML page when it receives the request for the HTML page. Please show your calculation and the final answer.

The referenced objects will arrive the same time as HTML arrives.

T = $t_s + 8 \times t_{obj} = 0.506 + 8 * 0.1 = 1.306\,(s)$

If they have the basic idea and explanations right, + up to 2 points (+1 or +2 according to their explanation) If final answer is correct, +1 point

HTTP/2 Prioritization:

The sequencing of how a Webpage's resources are loaded can have a significant impact on how long it takes for the user to see the content and interact with the page, i.e. the user experience. With HTTP/2, the browser can provide detailed instructions to the server for how the resources should be delivered, allowing for prioritization of certain resources. For the following 2 parts, we no longer should assume one packet per object (as in previous parts). Instead, assume that objects can be infinitesimally subdivided and interleaved.Suppose the site you're visiting is an e-commerce web page and of the 8 objects referenced in the HTML file:

- 1 is a CSS file acting as a style sheet; this tells the browser how to style the page content, and the browser cannot display any content until after it has loaded the stylesheet (so it knows how to style the content it is going to display).
- 2 are external scripts (JavaScript), 1 is synchronous (runs critical logic, so nothing can render until it is done executing), and 1 is asynchronous (runs non-critical logic like analytics). Each JS script takes 0.3 seconds to execute, and they can be executed in parallel / in the background.
- 5 are images, 3 of which are visible upon visiting the site.

For simplicity, we can still assume that each downloaded object is 100 Kbits long.

7. [2 Points] Without prioritization, the HTTP/2 default is to load everything in parallel, splitting the bandwidth evenly among everything. How long would it take for the images to appear on the user's screen? (assume we're still using Server Push). Please show your calculation and the final answer.

Same time as f) but need to run JS for 0.3s after getting all the objects

T = $t_s + 8 \times t_{obj} + 0.3 = 0.506 + 8 * 0.1 + 0.3 = 1.606\,(s)$

+0.5 correct formula for time it takes for all objects to load

+0.5 added 0.3s for JS to execute

+1 for correct final answer OR incorrect answer from f) + 0.3

Common mistakes

- Did not use server push as stated in the prompt

8. [3 Points] What is the per-object-type prioritization policy that allows for the optimal user experience (we can define it as the shortest waiting time before the visible images render on the user's screen)? You can consider sync and async JS, and invisible and visible images to be different object types. You can assume that the client will fetch objects of a given priority one at a time before moving to the next priority. Specify the policy by listing object types in priority order, e.g., an answer of "images > Async JS = HTML > CSS = Sync JS" means that images have the highest priority, CSS and Sync JS are equal and the lowest priority, and CS and HTML are equal and in the middle. Using the optimal prioritization policy how long would it take for all the

visible images to render on the user's screen? Please show your calculation and the final answer.

- HTML>Sync JS / CSS > Visible Images > Invisible Images/ Async JS.
  - Explanation: Load HTML, Sync JS, CSS to 100% completion in that order. Then load visible images either sequentially or in parallel (both are correct, with the difference being the images will appear all at once at the end of the loading period instead of one-by-one). Then load invisible images and Async JS either sequentially or in parallel.
  - The idea is to finish loading the blocking JS as early as possible so it has time to execute and unblock the rendering of the visible images. If we load the JS first, then it'll finish running by the time the CSS and visible images are done loading

T = $t_s + 5 * t_{obj}$ = 0.506+5*0.1 = 1.006 (s)

+1 for correct priority order
+1 for correct formula for T
+1 for correct final answer
Common mistakes

- Forget or wrong HTML order in priority (we need to load HTML first)
- For some reason, some students added an additional d_prop, this is not needed

HTTP/2 RFC:

Obtain the HTTP/2 specification (RFC 7540). Answer the following questions:

9. [1 point] What is the length of a HTTP Frame header?

72 bits
+1 correct
Common mistakes

- Said "unsigned 24 bit integer", which is the "Length" field in the frame header, but not the actual length of the HTTP frame header itself.

10. [1 point] What section of the RFC did you find your answer in?

Section 4.1
+1 correct

11. [1 point] Flow Control: How does a receiver specify how many frames it is prepared to receive?

Flow control is based on WINDOW_UPDATE frames. Receivers advertise how many octets they are prepared to receive on a stream and for the entire connection. This is a credit-based scheme. (Section 5.2.1)
+1 correct

12. [1 point] Prioritization: A client can assign a priority for a new stream by including prioritization information in the HEADERS frame that opens the stream. Does specifying a prioritization guarantee that concurrent streams will be processed in a particular order?

No, Explicitly setting the priority for a stream is input to a prioritization process. It does not guarantee any particular processing or transmission order for the stream relative to any other stream. An endpoint cannot force a peer to process concurrent streams in a particular order using priority. (Section 5.3)
+1 correct

Q2 DNS and CDN

1. [4 points] Why is UDP, not TCP, the default transport protocol for most DNS implementations? Give exactly two different reasons.

Any of the two answers below will suffice.
The DNS query and DNS response messages are short and they can fit in one UDP segment.
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).
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.
Not having to keep connections is an important advantage in terms of reducing load on nameservers, since there are potentially many users for each of the name server.
No need to perform congestion control for such a small and short communication.
Each of the reasons is worth 2 points. The students are asked to provide exactly 2 answers, so there are 4 points total. If the student provides more than 2 answers, the TA should grade the first 2 answers. If the student gives a different answer compared to the list, the TAs should award points accordingly based on their best discretion.

2. [5 points] Use nslookup to find out the A, AAAA, CNAME records for www.columbia.edu, and NS and MX records for columbia.edu. For each type of the record, show the exact command that you used, and the resolution result that you get.

```
nslookup -type=A www.columbia.edu
Server:         127.0.0.53
Address:        127.0.0.53#53
Non-authoritative answer:
www.columbia.edu        canonical name = www.a.columbia.edu.
www.a.columbia.edu      canonical name = source.failover.cc.columbia.edu.
Name:   source.failover.cc.columbia.edu
Address: 128.59.105.24
nslookup -type=AAAA www.columbia.edu
Server:         127.0.0.53
Address:        127.0.0.53#53
Non-authoritative answer:
www.columbia.edu        canonical name = www.a.columbia.edu.
www.a.columbia.edu      canonical name = source.failover.cc.columbia.edu.
nslookup -type=CNAME www.columbia.edu
Server:         127.0.0.53
Address:        127.0.0.53#53
Non-authoritative answer:
www.columbia.edu        canonical name = www.a.columbia.edu.
nslookup -type=NS columbia.edu
Server:         127.0.0.53
Address:        127.0.0.53#53
Non-authoritative answer:
columbia.edu    nameserver = dns2.itd.umich.edu.
columbia.edu    nameserver = auth1.dns.cogentco.com.
columbia.edu    nameserver = ext-ns1.columbia.edu.
columbia.edu    nameserver = auth4.dns.cogentco.com.
columbia.edu    nameserver = ns1.lse.ac.uk.
columbia.edu    nameserver = auth2.dns.cogentco.com.
columbia.edu    nameserver = auth5.dns.cogentco.com.
Authoritative answers can be found from:
nslookup -type=MX columbia.edu
Server:         127.0.0.53
```

Address: 127.0.0.53#53
Non-authoritative answer:
columbia.edu mail exchanger = 10 mxa-00364e01.gslb.pphosted.com.
columbia.edu mail exchanger = 10 mxb-00364e01.gslb.pphosted.com.
Authoritative answers can be found from:
1 point each for the correct answer for each query type, so 5 points in total.
The students must give correct answer for both the command and the result to receive the point.

3. [1 point] Based on your result in (b), do you think www.columbia.edu has support for IPv6, and why?

It does not support IPv6, as www.columbia.edu (and the domains that it CNAMEd to) does not have a AAAA record.
0.5 point for the correct answer (yes/no).
0.5 point for the correct explanation (no AAAA record).

4. [2 points] You learned about recursive DNS resolvers in the class. When performing your experiments in (b), which recursive DNS resolver did you use?

Note: Put down the public IP address of your recursive resolver. If the recursive resolver is your router, or a service that is running on your local computer, try to locate where it is sending requests to. Answers in private or link-local IP blocks, such as 192.168.0.0/16, and 127.0.0.0/8, are not acceptable.
Acceptable answers should be similar of the following:

1. The students configured their own DNS resolver, or they looked into the DHCP configuration that their computer or router is using. In this case, the IP of the resolver will be the client-facing IP. Example answers: 8.8.8.8, 128.59.1.3 (Columbia's recursive DNS resolver)
2. The student went to a website such as dnsleaktest.com, which let the users issue DNS queries to unique domain names and record the IP of the recursive DNS resolver. In this case, the IP of the resolver will be the authoritative-name-server-facing IP. Example answers: 172.253.210.11 (one of the IP address that Google Public DNS uses), 128.59.1.22 (the address that Columbia's recursive DNS resolver use to send DNS queries to authoritative name servers)

2 point for correct answer
0 point for incorrect answer

5. [2 points] Now, go to CDNPerf, and measure the latency of reaching www.columbia.edu from sites across the world. Attach the screenshot to your assignment PDF. Based on the results, do you think the users visiting www.columbia.edu from the US east coast would be visiting the same physical servers as users from the US west coast? If not, how does www.columbia.edu direct users from different regions to different physical servers? Show your evidence and reasoning.

Users visiting www.columbia.edu from the east coast are visiting the same physical servers as users from the US west coast.
Reasoning: CDNPerf shows that all the users of www.columbia.edu are visiting the same IP address, and there are drastic differences in terms of latency depending on the geographical location, with users in the east coast having the lowest latency.
1 point for pointing out www.columbia.edu operates on a single server
1 point for correct reasoning

6. [2 points] Using CDNPerf, measure the latency of reaching www.akamai.com from sites across the world. Attach the screenshot

to your assignment PDF. Based on the results, do you think the users visiting www.akamai.com from the US east coast would be visiting the same physical servers as users from the US west coast? If not, how does www.akamai.com direct users from different regions to different physical servers? Show your evidence and reasoning.

Users visiting www.akamai.com from the east coast are not visiting the same physical servers as users from the US west coast. www.akamai.com redirects users from different regions and networks to different users by using DNS redirection. More specifically, the authoritative name server for www.akamai.com will return different resolution results based on the source IP of the DNS queries, thereby directing users in different regions and networks to different physical servers.

Reasoning: CDNPerf shows that all the users of www.akamai.com are visiting different IP addresses, and enjoy similarly low latency. For example, in my test, vantage points in New York are reporting a RTT of 1.75 ms, and vantage points in Seattle are reporting a RTT of 0.63ms. Since light in optical fiber takes 38 ms to travel from New York to Seattle and then back, vantage points in these two cities cannot be visiting the same physical server. Since users in different locations are visiting servers with different IP addresses, it is using IP Unicast + DNS based redirection.

0.5 point for pointing out www.akamai.com is using different physical servers for users in different regions.

0.5 point for explaining why www.akamai.com must be using different physical servers for users in different regions.

1 point for explaining www.akamai.com uses Unicast + DNS redirection to redirect users.

7. [2 points] Using CDNPerf, measure the latency of reaching www.cloudflare.com from sites across the world. Attach the screenshot to your assignment PDF. Based on the results, do you think the users visiting www.cloudflare.com from the US east coast would be visiting the same physical servers as users from the US west coast? If not, how does www.cloudflare.com direct users from different regions to different physical servers? Show your evidence and reasoning.

Users visiting www.cloudflare.com from the east coast are not visiting the same physical servers as users from the US west coast.

www.cloudflare.com redirect users through IP Anycast. More specifically, www.cloudflare.com announces the same IP prefix at multiple physical locations, and let BGP routing rules to decide how users reach its network.

Reasoning: CDNPerf shows that all the users of www.cloudflare.com are visiting the same IP addresses (due to IP anycast), but the reported latencies are very low so that they violate the speed of light constraints. For example, in my test, vantage points in Bend, Oregon are reporting a RTT of 9.4 ms, and vantage points in Philadelphia, Pennsylvania are reporting a RTT of 3.72ms. Since light in optical fiber takes 37 ms to travel from Bend, Oregon to Philadelphia, Pennsylvania and then back, vantage points in these two cities cannot be visiting the same physical server. Multiple locations are visiting the same IP address even though according to the latencies they could not be using the same physical servers, so Cloudflare must be using Anycast.

0.5 point for pointing out www.cloudflare.com is using different physical servers for users in different regions.

0.5 point for explaining why www.cloudflare.com must be using different physical servers for users in different regions.

1 point for explaining www.cloudflare.com uses Anycast to redirect users.

8. [6 points] List exactly one advantage and one disadvantage for each of the content delivery approaches you discovered in (e) - (g). (By content delivery approaches, we mean your answers to those earlier questions.) There might be more than one advantage/disadvantage for the content delivery approaches you discovered, so please only list the most significant advantage/disadvantage you can think of.

(e): Use one single server to deliver content for users in different geographical areas. Pros: easy to understand, deploy, and maintain. Works well if many of your users are in the same geographic region, like Columbia.
Cons: worst performance because the server can be really far away from the user; poor reliability, because the site will fail if the only server fails. Limited total throughput because there is only one server.
(f) CDNs using DNS based server redirection.
Pros: CDNs inherently allows content to be served near users, which allows for lower latency and more throughput. CDNs also allow for robustness because if one server is down, there might be other servers still available.
CDNs using DNS redirection allows the CDN to control which group/geographical area of the user should go to which PoP to assign server with the lowest RTT or perform load balancing, although the granularity is limited as we can only redirect users on a per DNS recursive resolver basis.
CDNs using DNS redirection also allow off-net server deployment.
Cons: DNS records and TTL may not be respected. The user might be far away from their recursive resolver, which will lead to less than optimal server assignment. Longer failover time compared to Anycast due to DNS TTL.
Worse protection for DDOS attacks compared to anycast CDNs. The DNS system can be complex and a single point of failure.
(g) Anycast based CDNs.
Pros: CDNs inherently allows content to be served near users, which allows for lower latency and more throughput. CDNs also allow for robustness because if one server is down, there might be other servers still available.
Anycast is also very simple to configure.
Anycast has lower fail over time compared to DNS based redirection. Anycast also provides better DDOS protection because attackers around the world cannot select the same server.
Cons: the path/server that the users actually use is at the mercy of other networks/BGP. Harder to control or predict which path/site that the users will take, adding difficulty to load balancing. The routing results might be less than optimal.
Hard to perform off-net deployment.
For each of the (d)(e)(f), award 1 point if the student correctly points out at one of the pros and cons, so there are 2 * 3 = 6 points total. If the student's answer differs from the solution, award points based on the TA's discretion. If the student submits more than one one advantage or disadvantage for any approach, only consider the first response. If the student did not answer the specific advantages and disadvantages of DNS redirection based CDNs vs Anycast CDNs, but talked about the advantages and disadvantages of distributed CDNs in general, award 2 * 1 = 2 points.

9. [2 points] Does www.nytimes.com use CDN? If so, what CDN company does it use to deliver its front page (the HTML document) to its users? Show your evidence and reasoning.

Hint: you might want to perform a DNS resolution on www.nytimes.com and check the CNAME and IP addresses involved. .
DNS resolution shows that www.nytimes.com will CNAME to nytimes.map.fastly.net. Therefore, www.nytimes.com is using Fastly to deliver its front page HTML document.

As long as the student names Fastly as the CDN vendor and provides the correct explanation, award full points.
+0.5 point for pointing out www.nytimes.com uses a CDN
+0.5 point for correctly explain why www.nytimes.com uses a CDN
+1 point for pointing out Fastly is delivering www.nytimes.com front page.

10. [8 points] Major Internet companies (think about Google, Netflix) deploy web servers in their own Autonomous Systems (ASes, which you can think of as another word for networks), and these web servers will use IP addresses that belong to their own ASes. Some major Internet companies also deploy servers in other networks/Autonomous Systems close to their users -- the *Enter Deep* strategy from the book and lecture -- to reduce the distance between servers and their users and thereby providing better service,. For example, Netflix has a program called Netflix Open Connect, where it deploys its video servers into ISP networks that host human users. We refer to the web servers that are deployed in networks other than the company's own network as off-net servers. Now, one of the TAs of this course watched some videos on Netflix a few days ago, and he noticed that his computer loaded video chunks from ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net. Answer the following questions:
    1. [1 points] What is the IP address of ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net? Explain the steps that you take to find out the IP address of this domain.

    The student can use nslookup or dig or any other DNS utility to find the IP address of this domain. For dig, the command is dig ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net. For nslookup, the command is nslookup ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net. The IP address I got is 199.109.94.18, but it might change over time.
    1 point if the student provides a legitimate method to convert the domain to IP address, and also includes the IP address that he/she got. 0 points if the student did not provide a reasonable explanation for the IP address, even if the IP address might still be correct.

    2. [1 point] What is the value of the PTR record that DNS associates with the IP address that you find in (i)? Explain the steps that you take to find out the reverse DNS name.

    The student can use nslookup or dig or any other DNS utility to find the reverse DNS name of the IP that they got. For dig, the command is dig -x 199.109.94.18. For nslookup, the command is nslookup -PTR 199.109.94.18. The IP address I got is netflix-c001.lga001.nysernet.net., but it might change over time.
    1 point if the student provides a legitimate method to convert the IP to reverse DNS name, and also includes the reverse DNS name that he/she got. 0 points if the student did not provide a reasonable explanation for the reverse DNS name, even if the reverse DNS name might still be correct.

    3. [2 points] Is the DNS name found in the PTR record the same as the ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net? Why?

    No. The DNS record for ipv4-c001-lga001-nysernet-isp.1.oca.nflxvideo.net is provided by Netflix. The DNS record for 18.94.109.199.in-addr.arpa. is provided by NYSERNet. The RFCs of the DNS also do not require the two domain names to be the same.
    1 point for correct answer (same/different), 1 point for reasonable explanation.

4. [2 points] Based on the information that you have, do you think this particular Netflix video server is deployed on Netflix's network, or one of the ISP networks that host Netflix users? Why?

This server is hosted at NYSERNet, not Netflix's networks/AS. Reasoning: the student can look up IP to AS database and thus discover that this IP address belongs to NYSERNet, not Netflix. The student can also realize that the reverse DNS record points to .nysernet.net, and the authoritative name server of this domain also is a NYSERNet server.

1 point for correct answer (yes/no), 1 point for reasonable explanation.

Q3 CDNs and video

CDNs, Video Streaming and P2P [29 points, parts a-l]

The 1.5-hour video file on YouTube has two versions: low-quality(70MB) and high-quality(2GB). Assume for now that each version is encoded as a single file (rather than chopped into chunks, which we will get to below). Label your answer to each part clearly.

1. [2 points] What is the bitrate of the low-quality video? (show calculation)

Low-quality: 70MB * (8Mb/MB) / (1.5hr * 3600s/hr) = 0.1037 Mbps = 106.2 kbps or 103.7kbps or 103704bps

Grading: (+1 for 106.2 kbps or 103.7kbps, +1.5 Partially correct calculation due to conversion error with correct explanation,+2 if with calculation, +0 for wrong answer)

Common Error: not converting it to bits or wrong calculation.

2. [2 points] What is the bitrate of the high-quality video? (show calculation)

High-quality: 2GB * (8Gb/GB) / (1.5hr * 3600s/hr) = 0.002963 Gbps = 3.03 Mbps or 2.96Mbps or 2962963bps

Grading: (+1 for 3.03Mbps or 2.96Mbps, +1.5 Partially correct calculation due to conversion error with correct explanation, +2 if with calculation, +0 for wrong answer)

Common Error: not converting it to bits or wrong calculation.

3. [2 points] If current access rate is 256 KB/s, which version will DASH choose, and why?

Access bit rate: 256KB/s = 2048kbps = 2Mbps or 2.05Mbps. Dash will choose low-quality version.

Grading: (+0.5 if choose low-quality without calculation or with wrong calculation, +2 if with calculation towards correct answer either 2Mbps or 2.05Mbps, +0 otherwise)

Common Error: choosing low-quality without calculation

4. [3 points] A user is watching a YouTube video on Chrome browser. Which application-layer protocols are involved in this process?

HTTP(S) and DNS.

Grading: (+1.5 points for each, if provide with other correct answer, like RTSP, DHCP, no mark reduction. If provided with wrong protocols, like UDP, -1.5 for each wrong protocol, but will not get points below 0 for this question)

Common Error: only mentioning HTTP

5. [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.

Another acceptable answer:

An advantage is that it allows the services to use standard browsers and standard caches.
+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
Common Error: only mentions reliability. Or advantages of TCP over UDP

6. [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.)
Common Error: only mentions reducing handshake time.

We will now consider how long it will take YouTube to serve users under different architectures. There are N users watching the same video of size F at the same time, download rates are all d, client upload rates are all u. These users all connect to same server at the starting stage. The uploading rate of this server is u_s. What is the minimum distribution time (to serve all clients) under the following scenarios? Show your math and justify each term in your answer

7. [2 points] In traditional server-client architecture?

$D_{CS} = max[?][?][?]\{NF/u_s, F/d\}$

Explanation:
NF/u_s is the time for uploading N files from the server.
F/d is the time for one client to download a file.
+2 for correct answer and detailed explanation
+1.5 for correct answer and vague but correct explanation
+1 for correct answer and not-incorrect but very poor explanation
+1 Partially correct formula but a correct explanation.
+0.5 for incorrect answer but a reasonable explanation if the network were designed differently, or for correct answer with no or incorrect explanation.
+0 otherwise.
Common Error: correct answer with no explanation.

8. [2 points] If YouTube moves to a P2P model, in which clients can exchange pieces of the video with each other in parallel or download from the server?

$D_{P2P} = max[?][?][?]\{F/u_s, F/d, NF/(u_s+Nu)\}$

Explanation:
F/u_s is the time for uploading the first seed file from server.
F/d is the time for one client to download a file.
$NF/(u_s+Nu)$ is the time for N files to be uploaded.
Grading:

+2 for correct answer and detailed explanation
+1.5 for correct answer and vague but correct explanation
+1 for correct answer and not-incorrect but very poor explanation
+1 Partially correct formula but a correct explanation.
+0.5 for incorrect answer but a reasonable explanation if the network were designed differently, or for correct answer with no or incorrect explanation.
+0 otherwise.
Common Error: correct answer with no explanation.

9. [2 points] If Youtube uses a CDN with M servers, where $N = kM$ and k is a positive integer? Each server can serve multiple users at the same time, but the total upload rate for each server cannot exceed u_s.

M servers: $D_{MServer} = max[?][?][?]\{kF/u_s, F/d\}$.
Suppose $u_s \leq kd$, each server serves k users, each at rate of u_s/k, each user can only download at rate u_s/k. The distribution time is kF/u_s.
Suppose $u_s \geq kd$, each user download at rate d, each server serves k users, each at rate of d. The distribution time is F/d.
+2 for correct answer and detailed explanation
+1.5 for correct answer and vague but correct explanation
+1 for correct answer and not-incorrect but very poor explanation
+1 Partially correct formula but a correct explanation.
+0.5 for incorrect answer but a reasonable explanation if the network were designed differently, or for correct answer with no or incorrect explanation.
+0 otherwise.
Common Error: correct answer with no explanation.

10. [2 points] Under what high-level conditions will the minimum download time in (h) be less than that in (i) and vice-versa? (You don't need to specify precise mathematical conditions, but explain your answers.)
(h) could be less than (i) if aggregate user upload speeds provide faster file distribution than the server upload speed. (i) could be less than (h) if aggregate user upload speed is much smaller than server upload speed, making the limiting factor in (h) $NF/(u_s+Nu)$
+2 points for correct answer (being lenient with mathematical precision, just identifying the key factors are the server upload speed vs the user upload speeds)
Common error: Not mentioning anything about upload speed. Or commenting on increase in number of users.

11. [3 points] Identify two major reasons why Youtube uses the model in (i) rather than (h).
Possible reasons Youtube uses the CDN approach include, but are not limited to (lots of reasonable answers): ISPs would block youtube traffic clogging up their networks in the P2P model, users might see worse download times in the P2P model if user upload speeds are insufficient, it would be more difficult to implement optimized ABR algorithms in a P2P network, a centralized architecture gives Youtube more direct control over satisfying user requests/handling user traffic.
+1.5 per valid reason
+.5 for each not-wrong but likely insignificant reason

Now 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.

12. [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**)
    - Using chunks allows easy caching as HTTP object. Using chunks is reliable thanks to TCP as it needs to receive every packet in a chunk before playing it. So it can provide a solid quality of video for 30 seconds. Whereas for CABR, it is hard to implement cache when receiver doesn't know the exact number of packets to expect, since it is not guaranteed.
    - Using chunks needs simpler algorithm and lighter overhead on both client and server sides. While CABR needs more complicated decoding.
    - Using chunks is more compatible with middleboxes. Some middleboxes block UDP connections, plus many optimize TCP and/or HTTP connections (for example, caches and proxies), and so they could function over chunked video. Middle boxes include firewall and NAT. The middle boxes are doing the their work at chunk-wise(same work for all packets in a chunk) when using chunks. In CABR they do their work at packet-wise, which increases the overhead of middle boxes.

(other answers may be possible)
Grading:

+4 with reasonable answer and explicit explanations (mentioning both why this is an advantage of chunks and what is the difference to CABR),
+3 with correct answer and vague explanations (only mentioning the reason of being an advantage without comparison with CABR),
+2 with correct answer but with vague explanations,
+1 if only providing an advantage,
+0 otherwise
Common Error: listing out avantages without comparison with CABR.

13. [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**)
    - CABR is more adaptive to change. The time scale is smaller, the scheme changes per second. Chunks changes the requested quality every 30 seconds.
    - CABR deals better with loss, so user can have a more smooth experience with a poor access rate. When losing one packet, chucks needs to wait until this lost packet is retransmitted and received.

(other answers may be possible)
Grading:
+4 with reasonable answer and explicit explanations (mentioning both why this is a disadvantage of chunks and what is the difference to CABR),
+3 with correct answer and vague explanations (only mentioning the reason of being a disadvantage without comparison with CABR),
+2 with correct answer but with vague explanations,
+1 if only providing an advantage,
+0 otherwise
Common Error: listing out avantages without comparison with CABR.

Q4 Socket Programming

Download the Python programs TCPClient, UDPClient, TCPServer and UDPServer from Courseworks (slides -> socket_programming_examples) to your local machine. You can use the virtual machine for project 1 if you do not have a local Python development environment. Uncomment the line `clientSocket.send(message.encode())` in TCPClient.py.
In the following questions, if a “premature” error occurs and prevents you from completing everything specified in the problem, explain the error and what causes it.

To run a Python program, type `python3 <python-file-name>.py` in the terminal.

A. TCPClient.py & TCPServer.py

1. [2 points] Run TCPServer, then run TCPClient, then input a sentence on the client side. Explain what happens.

We will obtain the expected behavior. The client’s input sentence will get transmitted to the server, and it gets a response back from the server.
+2 Correct answer and explanation
+0 Incorrect answer or no explanation

2. [2 points] Run TCPClient, input a sentence, then run TCPServer. Explain what happens.

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.
+2 Correct answer and explanation

+0 Incorrect answer or no explanation

3. [2 points] Run TCPClient, then run TCPServer and then input a sentence on the client. Explain what happens.

Same as answer to part A2.

+2 Correct answer and explanation

+0 Incorrect answer or no explanation

4. [2 points] What happens if TCPServer.py is running on port M, and TCPClient.py attempts to connect to TCPServer.py via port N?

This will result in failure on the client side, as it can't established a connection with the server.

+2 Correct answer and explanation

+0 Incorrect answer or no explanation

B. UDPClient.py & UDPServer.py

1. [2 points] Run UDPServer, then run UDPClient, then input a sentence on the client side. Explain what happens.

We achieve the expected behavior. The client's message gets transmitted to the server, and it gets a response back from the server.

2. [2 points] Run UDPClient, input a sentence, then run UDPServer. Explain what happens.

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.

+2 Correct answer and explanation

+0 Incorrect answer or no explanation

3. [2 points] Run UDPClient, then run UDPServer and then input a sentence on the client. Explain what happens.

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.

+2 Correct answer and explanation

+0 Incorrect answer or no explanation

4. [2 points] What happens if UDPServer.py is running on port N, and UDPClient.py attempts to connect to UDPServer.py via port M?

The client will hang after sending the message as it doesn't receive a response back. (it's enough if students mention 'the server cannot receive the message').

+2 Correct answer and explanation

+0 Incorrect answer or no explanation

C. [6 points] Number of sockets

1. [1 point] Run TCPServer.py. How many sockets have we established? Why?

We've established 1 socket. This is the "listening socket."

+1 Correct answer and explanation

+0 Incorrect answer or no explanation

2. [1 point] With TCPServer.py still running, run TCPClient.py. What was the maximum number of sockets that were established on the *server* side just prior to TCPClient.py terminates?

2 sockets – one "listening socket" and another "connected socket."

+1 Correct answer

+0 Incorrect answer

3. [1 point] Consider the case where our TCPServer.py now can accept concurrent connections. How many sockets are there on the server side if our TCP server simultaneously handles n TCP client connections? Why?
N + 1 sockets. N "connected sockets" and 1 "listening socket."
+1 Correct answer and explanation
+0 Incorrect answer or no explanation
4. [1 point] Run UDPServer.py. How many sockets do we have? Why?
1 socket – a UDP server only needs one socket to send/receive messages.
+1 Correct answer and explanation
+0 Incorrect answer or no explanation
5. [1 point] With UDPServer.py still running, run UDPClient.py. What was the maximum number of sockets that were established on the *server* side just prior to UDPClient.py terminates?
1 socket
+1 Correct answer
+0 Incorrect answer
6. [1 point] Consider the case where our UDPServer.py now can accept concurrent clients. How many sockets does our UDP server need if it simultaneously handles n UDP client connections? Why?
1 socket. UDP is connectionless so one socket should be enough to handle all the requests.
+1 Correct answer and explanation
+0 Incorrect answer or no explanation

D. [6 points] Now we will add new functionalities to our simple Python programs. For each of the questions below, please briefly describe how you will implement each feature. Please be specific in describing what you will add, modify, or delete. Code/pseudocode is welcome but not required.

1. [2 points] Consider the case where our TCPServer.py receives 10 connections at a time t = 0. In the current implementation, the server will process each connection sequentially (i.e. first accept connection 1, then capitalize message 1; then accept connection 2, then capitalize message 2; etc.). How can we incorporate concurrency to our implementation, so that TCPServer.py will be able to keep accepting connections while capitalizing the messages *at the same time*?
Hint: you might find this blog post helpful:
https://towardsdatascience.com/multithreading-multiprocessing-python-180d0975ab29
Answer 1: use multiple processes (i.e. fork/exec)
Every time when we receive a new connection request, right after we make a connected socket, we can fork (and exec). The parent process then continues to listen for new connections (close the connected socket and keeps only the listening socket), while the child process can start to process the request via the connected socket (close the listening socket and keeps only the connected socket).
Answer 2: use multiple threads
Every time when we receive a new connection request, we will initiate a new thread to handle the request. In other words, if we have 10 connections, we will eventually end up with 11 threads – 1 master thread that keeps accepting connections, and 10 threads that handle connections.
+2 Correct answer and explanation
+0 Incorrect answer or no explanation
2. [2 points] Let's say that instead of passing simple text messages between the (TCP/UDP)Client.py and (TCP/UDP)Server.py, we would like to pass an instance of the "Notes" class (the "Notes" class has 3 attributes – creation date, content, and author). Describe the sending and receiving processes on both the client and server side that would enable passing these structured objects over our sockets.

Before sending the Notes class into a socket, we will serialize it using a Python library such as pickle; for everything that we retrieve from the socket, we will deserialize it using the same Python library.

.

Another answer: instead of serialization, we can also send each attribute of the class separately in a sequence. In the Notes class example, we can send/receive three successive messages of creation date, content, and finally author.

+2 Correct answer and explanation

+0 Incorrect answer or no explanation

3. [2 points] How can we implement client connection request retries in the case where a connection to the server fails (i.e. after running (TCP/UDP) server.py, the program either throws an exception or hangs)?

We can set a timeout and use try/catch statements.

- In the case where the client hangs, a timeout will catch the error.
- In the case where an exception occurs on the client side, the try/catch statement will catch the error.

+2 Correct answer and explanation

+0 Incorrect answer or no explanation