Midterm Exam

CS 494/594 Internetworking Protocols (Winter 2022) Portland State University
Instructor: Dr. Nirupama Bulusu

SUBMISSION DEADLINE: Thursday, February3rd, 11:59 pm PST.

Name:____**Parth Parashar**_____

Instructions:

1. Your answers can either be typed or handwritten. Please upload your completed midterm exam as a PDF file to Canvas. Please save a local copy of your exam.
2. There are 5 questions in this exam. Answer all the questions.
3. Show all your work, even if you are only able to partially answer a question.
4. Proctoring: Please post questions regarding any clarifications you might need on the exam by email/Slack before Wednesday February 2nd at noon. I may not be able to get to all questions posted after then.

If you are facing any unique challenges due to coronavirus and require accommodations, please do not hesitate to email me and discuss your situation.

1. (20 points) Peer-to-Peer File Distribution

Alyssa Hacker launches AlyTorrent, a peer-to-peer file distribution company. Alyssa chooses a peer-to-peer architecture, in which there is a swarm of 2^n peers and, during a considered time period, called an epoch, no peers join or leave the swarm. It takes a peer 1 unit of time to upload or download a piece, during which time it can only do one or the other. Initially one peer has the whole file and the others have nothing.

(i) If the swarms target file consists of only 1 piece, what is the minimum time necessary for all the peers to obtain the file? Consider only the upload/download time and ignore everything else.
Answer: - Minimum time necessary for all peers to obtain the file = n time units
This is because each peer takes 1 unit of time to upload or download a piece. Thus, during each time unit, each peer with the piece can transmit it to one peer without the piece. This means that number of peers with the pieces doubles with each unit time.

Time Unit	Peers having that file
0	2^0 = 1
1	2^1 = 2
2	2^2 = 4
n	2^n

(ii) Let *x* be your answer to the preceding question. If the swarms target file instead consisted of 4 pieces, what is the minimum time necessary for all peers to obtain the target file?
Answer: - Let us assume that we have 4 pieces of file named W,X,Y,Z of the target file.
Therefore, the swarms will be divided into 4 groups namely X1,X2,X3,X4.
X1→ Possess the complete file.
This means that the time taken for peer-1 to copy file piece X to X2 is 1.
This means, total time = 3
Time taken for swapping = 6
Following the similar trend, we can say that,
Final Time = (1+1+1+6+n-2)
Final Time = n+7

(iii) Suppose each epoch consists of *n* time units. If the swarms target file consists of only 1 piece, and 2^n more peers join the swarm after the first epoch, what is the minimum time necessary for all the peers in AlyTorrent, old and newly joined, to have the file?

Answer: - Let us consider that there are 2^2 = 4 peers involved in the transaction.

After an epoch of 1 unit time, the number of peers who possess the file are 2.

Now, at this moment, the number of peers added in the swarm is 2^2=4.

Total peers needing target file = 6

Total peers having the file = 2

Now at 2 time unit, 2 more peers will acquire the file.

AT 3 time unit, all 8 peers would acquire the file.

Thus, in total, it would take n+1 time unit for all peers to acquire the file if the number of peers increase by the same 2^n number.

2. *(20 points) Distributed Hash Table*

After a year of running her peer-to-peer file distribution company, Alyssa notices that file lookup through her central server is becoming extremely slow. Alyssa reads a research paper on distributed hash tables and decides to emulate the functionality of a BitTorrent tracker using a distributed hash table. After learning how knowledgeable you are about DHTs, she hires you as a consultant to design and implement a DHT tracker for her. How would you design the DHT?

Answer: -Since Alyssa's company is working on peer-to-peer network architecture, therefore, it can be said that the system has a centralized server. This centralized server is being used to store indexes of all the files that are being uploaded or downloaded by the users using it.

When a user requests for a file, the system obtains the IP addresses of the peers who have that file.
Now, when there are millions of users using it, then the number of requests are large, and therefore, the load on the server will also be very high.

DHT Design: -
DHT is the short form for Distributed Hash Table which is used to decentralize the file locating process. In DHT, each peer is responsible for subset of other peers in DHT and operation is similar to bit Torrent protocol in which multiple computers (nodes) act as a peer and they organize themselves into a loop network.

(a) Who would be the peers?

Answer: -People participating in the download or upload of a file are peers. A peer who possess a chunk of file or the complete file can upload the file over the network so that others can download it using the network. These peers are called seeders.

In contrast, the peers who only download the file over the network are called leechers. A peer can store information about its successor, predecessor and the shortest route. Whenever a request is made, the peers either answer it or passes it to the peer who can answer it.

Peers perform two operations in DHT as mentioned below: -

1) Peer can query the database with key and database returns values that match the key.
2) Peers would also be able to insert the key value pair into the table.

(b) What would be the keys, and the value stored for each key?

Answer: - Now since we know that peers can perform two operations as mentioned below: -

1) Peer can query the database with key and database returns value that match the key

2) Peers would also be able to insert the key value pair into the table.

Therefore, we can say that,

Each peer is identified using an identifier which is in the range of [0, 2^n– 1] and each key should also be an integer of that range. This will help us identify which part of database is stored at which peer.

(c) How would you initialize and bootstrap the DHT?

In bootstrapping the process from centralized to decentralized system, the new peer joining the network must know at least one peer.
This is because the new peer, after assigning the ID, will get to know its successors and predecessors from the network.
It then announces its arrival in the group and updates the group node list accordingly.

Example of the above process is: -

Consider that existing network has nodes as [30,40,60,65,90,100].
Now if peer 50 wants to join the network, and if it knows only peer 30 in the network. Then will ask peer 30 that who's responsible for key 50 and it will find out that key 50 is handled by 60 as its successor. Then it will announce its arrival by populating itself in network and updating the node list. Similarly, when node wants to leave the network or failure happens at node then bucket list gets updated with the peers present in the network.
By this method, multiple peers can simultaneously participate in the file sharing process as required. This increases the scalability and performance of the network. Here we saw that responsibility of maintaining network is not only on centralized server but on all the nodes in the network which is the main principle behind peer-to-peer decentralized DHT network.

3. (20 points) DNS

Alyssa Hacker thinks that DNS works by having each DNS server forward a query it can't answer to its parent until the query lands at a server that can answer it. Ben Bitdiddle disagrees with her and shows her how DNS actually works. Describe how Alyssa's version of DNS differs from the way DNS actually works. Given DNS$_{Alyssa}$, a DNS system that works as Alyssa described it, and DNS$_{Ben}$, an actual DNS system, which is better: DNS$_{Ben}$ or DNS$_{Alyssa}$? Justify your answer.

Answer: -
DNS*Alyssa* : This is a case of recursive query type. It works in the following manner: -

1) User sends the query requesting the IP address of particular server to local DNS server.

2) Then if local DNS server is completely de-cached, it will contact the root DNS server
3) Root DNS server then sends the request to the TDS to get the list of all the IP addresses.
4) TDS contacts the authoritative DNS server to resolve the query. This is because authoritative DNS server has hostname to IP address mapping.
5) Then authoritative DNS server responds back with the corresponding IP address for the requested server to TDS and from TDS to the root server.
6) Finally root server sends the requested IP address to local DNS and it gets forwarded to requesting host.
7) Each time when requesting host needs any IP address, the query has to go through every DNS server.

This is not an efficient method for name resolution because it has to deal with the burden of name resolution on the root DNS server.

DNS*Ben :* This is a case of the iterative query type. It works in the following manner: -

1) The requesting host sends a query to resolve the hostname and IP address to the local DNS server
2) Every Parent DNS server sends the response to the local DNS server by informing whom to contact further for resolution of the query.
3) In first iteration, local DNS server sends the query to the root name server, which replies with the address of the top-level domain server for the hostname.
4) Local DNS server then goes to TDS for further resolution
5) TDS gives it authoritative domain server.
6) Local DNS gets IP address mapped to the requested hostname from the authoritative domain server by sending the query to authoritative DNS server.
7) IP address will then get forwarded to the requested host.

This is a better approach than Alyssa's approach because all procedure is done through the local DNS server of the host server. This is particularly effective when there are a large number of requesting hosts. This takes off the burden on the root DNS server.

4. (20 points) Web and HTTP

You click on a link to Ben Bitdiddle's Web page that references 10 very small objects. The objects are located on the same server. Ignoring transmission times, how much time elapses (in Round Trip Time (RTT)) for:

(a) Non-persistent HTTP with 2 parallel TCP connections?

Answer: -

In a non-persistent HTTP, Multiple requests and responses are sent over separate TCP connections rather than maintaining the same TCP connection for multiple requests and responses.

Then, a client will request to initiate a connection for every request to the server. The server will accept the request and send the user appropriate response for the given request.

Once the response is sent, the server will terminate the TCP Connection.

Each TCP connection transports exactly one request message and one response message.

So, in this case, we will initiate 11 TCP connections. One to get the references of 10 objects from the server and the other 10 to get those 10 referenced objects.

As Ben will have 2 parallel TCP connections open at a time, so 2 images can be sent Simultaneously.

The RTT will be = 1 x 2 (One RTT to establish connection and other to get file references) + 5 x 2 (One RTT to establish connection and the second RTT to receive the object)

Thus, in total it would take 12 RTT

(b) Persistent HTTP with at most 2 pipelined requests?

Answer: - In Persistent HTTP, the server leaves the TCP connection open after sending the response. So, all the subsequent requests and responses between the same client and server can be sent over the same connection.

Thus, the entire base HTML file and 10 small objects will be delivered over a single persistent TCP connection.

The TCP connection in Persistent HTTP closes when it isn't used for a certain time (usually called the timeout interval).

Pipelining is the process of sending requests back-to-back without waiting for replies to pending requests.

In the case, it would take two RTT to initiate TCP for two pipeline connections each and both would take 1 RTT each to retrieve files. Thus, in total it would take 2 + 2 + 1 + 1 = 6 RTT.

c) Which of the two choices (a) or (b) would you recommend to Ben? Can you suggest any improvements by which Ben can cut download time by half?
Answer: - From the two answers above, choice (b) takes considerably less time as it reduces the RTT by a huge margin.
Thus, it is recommended to use the persistent HTTP.
In my opinion, only a single pipelined connection enough to retrieve all the referenced files. This is because having two connections increases the RTT. If only one pipelined connection is used, it would take 2 RTT to open a TCP connection and one RTT to receive all the referenced images. Thus, in total, it would take only 3 RTT and would reduce the total time by half.

5. (20 points) Content Distribution Networks and Web Caching

To minimize the time it takes a user to download a webpage, web content is often replicated at multiple locations. This problem explores two basic approaches for replicating web content: Web Caching and CDNs. CDN-based approaches rely on a content distribution network to serve content from multiple locations; there are (at least) three ways to implement the CDN distribution model. Thus, we have four different options at our disposal:

a) Web Caching: A local network routes all HTTP requests through a Web Cache. The cache first checks if the requested page is already cached, in which case the proxy returns the cached page to the user. If the page is not cached, the proxy requests the page from the site, caches the result, and returns it to the user.

b) CDN using DNS: When the authoritative server for the site receives a DNS request, it returns one of many IP addresses, each of which corresponds to a CDN server hosting the web content.

c) CDN using IP Anycast: DNS requests for the site always return the same, single IP address. Multiple CDN servers (at various different locations) that host the web content advertise this same IP address.

d) CDN based on rewriting media URLs: All requests for the base page of a site are served from a single server hosted by that site. The site's server re-writes the URLs for media content (e.g., images) to be URLs corresponding to CDN servers, so that media content is served from the CDN.

Haobo Yu's new web startup has gone viral, and he wants to purchase services from a CDN in order to minimize the load on your servers. List one of the options, or 'None 'if appropriate.

1. Which approach allows the finest-grained control over which users are directed to which servers? In particular, which approach allows a site to dictate exactly how many users are sent to each server, and to instantaneously re-balance load between servers?

Answer: - In approach (c), we have the most fine-grained control over which users are directed to which servers. With CDN using IP anycast, multiple machines share the same IP address. So, when a request is sent, routers direct it to machine on the network which is closest to the client.
Also, in case a network is incapacitated with heavy traffic, an Anycast network can respond to the outage similar to a load balancer and instantaneously re-balance the load between servers.

2. Haobo delegated this decision to his business partner Gita, who chooses one of the CDN-based approaches. After purchasing services from the CDN, traffic to Haobo's servers decreases substantially, but he continues to see requests for the content on his website (including both base pages and media) for a few hours. After a full day, traffic to his server dies out completely. Which approach did Gita choose?

Answer: - Because, all the base content of the site would be hosted on the main server and all the media related content would be served using CDN, it is recommended that Git should choose option (d). This is based on rewriting the media URL(s). This would allow us to reduce the costs on the usage of CDN while simultaneously use it efficiently during peak hours to provide smooth service to the incoming traffic.

3. Which approach can guarantee that customers will be served from the CDN server closest to the customer?

Answer: - It is evident from the question that Approach (c) CDN using IP Anycast can guarantee that customers will be served from the CDN server closest to the customer.

The traffic that goes to an anycast node is routed to the nearest node thus reducing latency between the client and node itself. Even if one of the servers fail this approach can just reroute to the next nearest server.

4. Which approach leads to additional load on Haobo's DNS server?

   Answer: - When compared to the other approaches, approach (b) CDN using DNS puts more load on Hoabo's DNS server. This is because it has to store and return multiple CDN IP addresses whereas in approach (c) CDN using anycast, the same IP address is returned by Haobo's DNS server.