Homework 2

Due: 11:59 PM on Tuesday, February 9, 2021.

Please answer in your own words and show any and all work.

1. (15 points) Identify the OSI layer responsible for each of the following functionalities:
   1. Providing node-to-node communications with reliable service.

**Answer: Data Link Layer** is concerned with reliable transmission connection between two adjacent nodes. Since the Network layer is responsible for the reliable delivery of packets from source to destination, before reaching a destination, a packet moves from multiple nodes. Moving packet one node to another Network layer relies on the Link layer. “At each node, the network layer passes the datagram down to the link layer, which delivers the datagram to the next node along the route. At this next node, the link-layer passes the datagram up to the network layer” [1]. Service provided by link-layer at the node depends on specific link-layer protocol employer over that link. Examples of link layers are Wi-Fi, Ethernet, and more.

* 1. Determining the best path to route packets.

Answer: **Network Layer** is responsible for identifying the best path across the network. It has a message(datagram) from the transport layer, which contains the destination address to the network layer. After that, the Network layer provides the servicing of delivering the datagram to the destination host. The network layer is for OSI and Internet; both contain IP protocol, which will identify the fields in the datagram and how the end system and router behave on these fields. It also consists of a router protocol that took care of route packets/datagrams to reach the destination.

* 1. Providing end-to-end communications with reliable service.

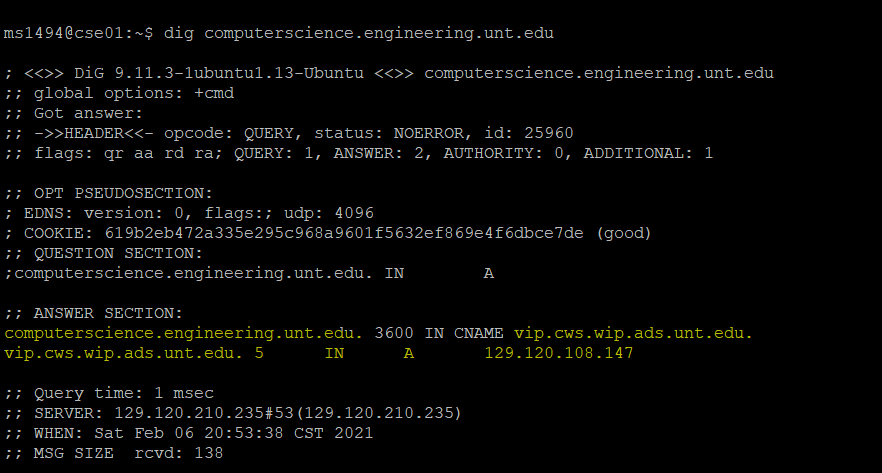
Answer: **Transport Layer** is responsible for providing end-to-end reliable communication service. Just Like Internet protocol OSI Transport layer too have two transport protocols: TCP and UDP. “TCP stands for transmission control protocol and UDP”[1] for User Datagram protocol. The TCP is known as stateful protocol since it is responsible for connection-oriented service, whereas UDP is a connectionless or stateless protocol usually used in simpler messaging transmission. Suppose we required reliable service between hosts or end-system so the best way is to use TCP because it makes sure all the packets are delivered to the destination and make a proper connection before starting the transmission of datagrams.

1. (10 points) A good tool for exploring DNS is dig, short for Domain Information Groper. dig performs a DNS lookup and prints information about the request and the response received. If you run dig, you may see results that differ from other students here.
   1. After connecting to one of our CSE machines, use the dig command to find the IP address for computerscience.engineering.unt.edu. What is the IP address and canonical real name of this host? Show the results of your query.

Answer:

IP address: 129.120.108.147

Canonical name: vip.cws.wip.ads.unt.edu



* 1. Based on the results from the dig command, what is the expiration time for the computerscience host? Be sure to include the proper units.

Answer: 3600 sec

1. (7 points) Suppose that you have two browser applications open and active at the same time and that both applications are accessing the same Server to retrieve HTTP documents at the same time. How does the Server know how to tell the difference between the two applications (to send the correct document(s) to the correct application)?

Answer:

Each browser will send one HTTP request to Server for retrieving the response page for their query. Each request connection between Server and client end system is established using a TCP transport protocol, ensuring reliable end-to-end connection without data loss. After a connection is created, the client and server application processes access through their socket interface. A process’s socket is just like a door for the process. At the client end, this is a door between a client process and TCP connection. On the server-side, it is a door between the server process and TCP connection. Client and Server send and receive HTTP request and response messages through their socket interface. So, if clients want to communicate for any request to Server, it first establishes a connection with the server door and their door (socket interface). Once the connection is established client sent the message to its socket interface, which is later transferred to the transport layer. It is the transport layer's responsibility to deliver it to the destination socket. As in our case, two different application are accessing the same Server the two connection is established via a TCP transport protocol, and two sockets are connected to the server socket. Since TCP is a connection-oriented protocol, it will first select the connection for application one with socket 1 and then for a second application, have a different socket number, let’s say 2. So once the message is out of the client socket, at receiving end, these processes are identified by two crucial pieces of information i) IP address of the system (which is the same in our case) ii) identity of receiving process port number(different). These two information make sure that the messages are delivered to the correct requester(application). In this way, TCP makes sure that the application receives the correct HTTP response.

Process

Socketket

Client

Process

Server

Socket

Socket

Process



Server sends these requested files to client without storing their information. Therefore, HTTP is said stateless protocol.

1. (20 points) Suppose that we want to distribute a file with a size of Gbits to clients/peers. The Server supports an upload rate of Mbps while each client/peer has a download rate of Mbps and an upload rate of , where . Show your calculations.
   1. Calculate the minimum distribution time for a client-server distribution using the two values of given above.

Answer:

F = 20 Gb

N = 100

di = 2Mbps

1Gb = 1000Mb therefore F = 20000 Mb

dmin= di = 2Mbps

Lets calculate distribution for client-server which is Dcs.

Dcs >= max{NF/us,,F/dmin} [3]

So let's put the values in the equation.

Dcs >= Max {100 \*20000Mb/30Mbps , 20000Mb/2Mbps}

>= Max{2000000/30 s, 10000 s}

>= Max{66666.67 s , 10000s}

66666.67 sec

|  |
| --- |
| **So the client to server distribution time for u= 300Kbps**  **= 66666.67 seconds**  **So the client to server distribution time for u= 700Kbps**  **= 66666.67 seconds** |

* 1. Calculate the minimum distribution time for a peer-to-peer distribution using the two values of given above.

Answer:

For peer-to-peer, the calculation is a little different:

F = 20 Gb

N = 100

di = 2Mbps

1Gb = 1000Mb therefore F = 20000 Mb

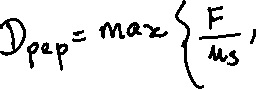
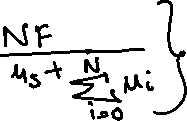
dmin= di = 2Mbps

u1= 300 Kbps =300/ 1000 Mbps = 0.3 Mbps

u2 = 700 Kbps = 700/1000Mbps = 0.7 Mbps

So, distribution time for peer-to-peer is DP2P.

Formula from the slide:



So, for ui = 0.3 Mbps

F/us = 20 \*1000Mb / 30Mbps = 666.67 sec

F/dmin = 20\*1000Mb / 2Mbps = 10000 sec

NF/(us +N\*ui) = 100 \*20\*1000 Mb/ (30 +100\*0.3) = 2000000/60 = 33,333.33 sec

So if we check DP2P = max{666.67,10000,33333.33}

= 33333.33 sec

Now Lets calculate if u = 700Kbps = .7 Mbps

NF/(us +N\*ui) = 100 \*20\*1000 Mb/ (30 +100\*0.7) =20000sec

Rest two values are the same as above

So, if we check DP2P = max{666.67,10000,20000}

So, the DP2P with 700Kbps = 20000 sec

|  |  |
| --- | --- |
| **ui** | **N= 100** |
| **300Kbps** | **33333.33 sec** |
| **700Kbps** | **20000 sec** |



1. (7 points) TCP/IP over Ethernet supports basic frames with a total size of up to 1518 bytes (including both the message payload and headers). Suppose that an application protocol wants to send an *L*-byte message across the network to its peer over TCP/IP. The TCP segment adds 20 bytes of header to the payload, while IP packet adds an additional 20 bytes of header to the segment. If the Ethernet frame adds 18 bytes of header to the packet, calculate the size of *L* the application needs so that exactly 90% of the transmitted bits in the physical layer carry the message payload (i.e., the data itself, not the header). Show your calculations.

**Answer**:

Frames supported over Ethernet = 1518 bytes

L bytes message needs to bot shared/

This L Bytes message overheads are:

TCP: 20 bytes of header

IP: 20 bytes of header

Ethernet: total 18 bytes of header.

Therefore, if the application needs to transmit 90% bits in physical with message payload can be calculated as:

Total Overhead: 20+20+18 = 58

Length of message =L

L/ L+overhead = 90/100

L = .9 (L+ 58)

L = .9 L + 52.2

L- .9L = 52.2

0.1L = 52.2

L = 522 Bytes

|  |
| --- |
| **Therefore, the Size of L = 522 Bytes** |

1. (7 points) Identify at least one reason why DNS uses UDP instead of TCP for its query and response messages. Justify your answer.

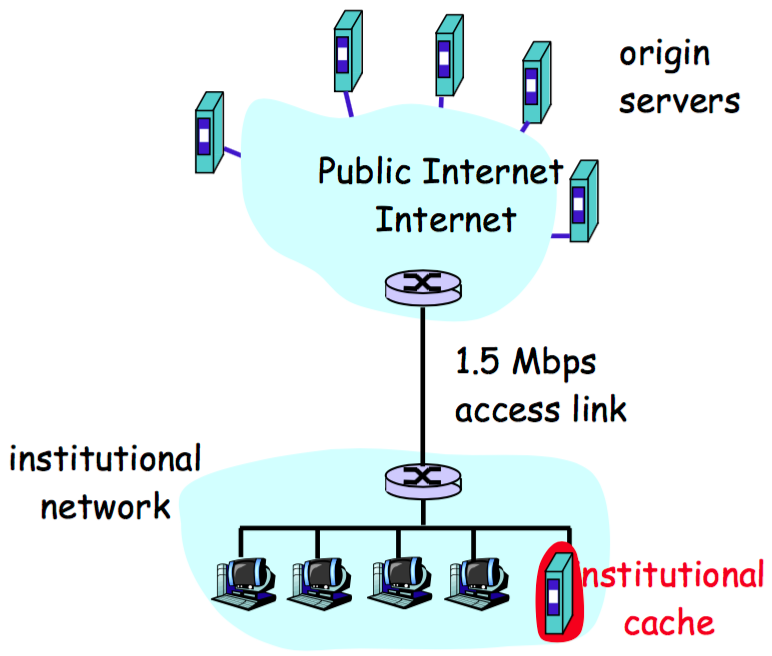
**Answer**: DNS is a directory where hostnames are linked with their IP address. DNS stands for Domain Name Server since it serves to keep the directory of each host corresponding to their IP address. From the book, we can say that “The DNS is (1) a distributed database implemented in a hierarchy of DNS servers, and (2) an application-layer protocol that allows hosts to query the distributed database.” [1].

As we know that the DNS uses UDP instead of TCP for its query and response. Few are some points below which explain why:

* UDP is much faster: Since UDP does not perform a three-way handshake process to start the data transmission, the data transmission in UDP is much faster. The DNS request is very small. It can quickly fit in the UDP segment and pass to the Network layer faster than TCP/IP. On the other hand, TCP first establishes a connection using the handshake process, which throttles in case of congestion and makes the whole data transmission process as long delay sometimes. Since UDP has no connection delay, it becomes one of the primary reasons why DNS uses UDP over TCP.
* No connection state: UDP supports more clients than TCP/IP due to its no connection state. No connection state implies that unlike TCP, UDP does not “receive and send buffers, Sequence and Acknowledge Number Parameters and congestion-control parameter. Protection, in this case, can be provided at the application layer. For this reason, a server devoted to a particular application can typically support many more active clients when the application runs over UDP rather than TCP”[1].

For DNS, we are trying to achieve faster network and protection, which we can achieve through UDP based on the above two points.

1. (20 points) Consider the following simplified network diagram where there is an institutional network connected to the Internet:



Suppose that the average object size is 60,000 bits and that the average request rate from the institution’s browsers to the origin server is 23 requests per second. Also, suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is two seconds on average. Model the total average response time as the sum of the average access delay and the average Internet delay. For the average access delay, use , where is the average time required to send an object over the access link and is the arrival rate of objects to the access link. You can assume that the HTTP request messages are negligibly small and thus create no realized traffic on the network or the access link. Show your calculations.

* 1. Find the total average response time.

**Answer:**

Object Size = 60,000 bits

Request rate = 23 request/sec

The time to transmit an object of size L over a link rate R is L/R. The average size of an object divided by R

As we know, is the average time required to send an object over the access link.

= 60,000bits/1.5 \*106 bps

= 0.04 sec

The traffic intensity on the link given by

= 23 request/sec \* 0.04sec/request

= 0.92 sec

So the average access delay can be calculated using .

= 0.04/ (1- 0.92)

= 0.5

Since it is clear in question total average response time is : average access delay + the average Internet delay

Total average response = 0.5sec + 2 sec

= 2.5sec

|  |
| --- |
| **So the total average response is 2.5 sec** |

* 1. Now, suppose a cache is installed in the institutional LAN (see figure). Suppose the miss rate is 0.4. Find the total response time.

**Answer:** The traffic intensity on access link is now reduced by 60% since 60% of the request are satisfied within the institutional network.

So, the probability of requests satisfied by the access link is 0.4.

The average access delay =

= 0.04(as calculated above) / [1 – (0.4) (0.92)]

= 0.063 sec

Average response time = avg access link delay + internet delay

= 0.063 sec + 2sec

= 2.063 sec

The average response time is nearly zero if the request is satisfied by the cache.

So, the total response time is = average response time by access link + average response time for cache.

Cache satisfied request probability = 0.6

Access link satisfied probability = 0.4

|  |
| --- |
| **Total Response Time = 0.6 \* 0 + 0.4 \* 2.063**  **= 0.825 sec** |

1. (7 points) Suppose that you join BitTorrent as a new peer without possessing any chunks. Unfortunately, you cannot become a top-4 uploader for any of your peers since you do not have anything to upload. Describe how you will be able to get your first chunk. Be specific.

**Answer:**

BitTorrent is a popular peer-to-peer protocol for file distribution. All the peers participating in file distribution in this is called a torrent.

BitTorrent infrastructure has a node name tracker. When any new peer joins torrent, it registers itself with a tracker and periodically informs the tracker that it is continuing with torrent. This is how torrent keeps track of active peers. When any user joins a torrent, after registering itself with the tracker, the tracker randomly selects a subset of peers from the participating peers and send IP addresses of these subsets of peer to recently join the user. Now users can establish concurrent connections with these lists of peers. So, all the peers with which users able to connect are now neighbor peers.

Initially, users do not have chunks of data, but their neighbors have a chunk that they share with other peers. “Each peer will have a subset of chunks from the file, with different peers having different subsets” [1]. Occasionally a user will ask each neighboring peer over the TCP connection for the list of chunks they have. So, if the user has L different neighbors, the user obtains L chunks. With this information, a user issues a request for the chunk, which it does not have. For instance, the user will have a subset of chunks and know their neighbor's chunk. With all this information, she needs to make two important decision

which chunks should the user request first among their neighbors?

And second, to which of their neighbors they should send requested chunks?

For making decisions about which chunk to request BitTorrent, use the rarest first technique, make sure the chunk that is least among their neighbor will be requested. This how the rarest chunk is distributed quickly. For the second part, sending the requested chunk to their peer user prioritizes their peer based on the highest supply data rate. The user will identify four peers and continuously measures the rate at which they receive bits, and determines the four peers feeding at the highest bit rate. Then they reciprocate by sending chunks to these same four peers. In every 10 seconds, the user recalculates the rates and possibly modifies the set of four peers. These four peers are said to be unchoked. Significantly, every 30 seconds, a user also picks one additional neighbor at random and sends it chunks. Let us call the randomly chosen peer A. In BitTorrent, A is said to be optimistically unchoked. If the connected peers are satisfied by trading, they will put each other in their top four lists and continue trading until one of them finds a better partner. This will also effects in capable uploading for peers with compatible rates for each other. The random neighbor selection isl also helpful for new peers to get chunks to have something to trade. Neighboring peers aprat from these five peers (four “top” peers and one probing peer) are “choked,” which means they do not receive any chunks from a user. [1]

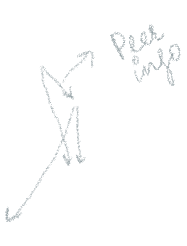
1. (7 points) Suppose that you have a circular Distributed Hash Table (DHT) with an ID space {1, 2, 3, …, 14, 15} and seven peers 1, 3, 4, 5, 8, 12, and 14. If peer 5 leaves the DHT, how does peer 3 update its successor information? After peer 5 leaves the DHT, which peer is now peer 3’s immediate successor? After peer 5 leaves the DHT, which peer is now peer 3’s second successor?

Answer:

ID Space = {1, 2, 3, …, 14, 15}

Peers are 1, 3, 4, 5, 8, 12, and 14.



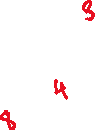
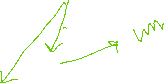
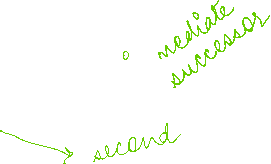




In circular DTH, each peer keeps the information of their two immediate successors, and it pings them periodically to check if the successor is alive or not.

When peer 5 leaves DTH, below are steps that peer 3 uses to update its second successor information since there is no change in its immediate successor.

So if peer 5 leaves the DTH. There will be no response from peer 5 to its predecessor when its predecessor ping to check its successor's aliveness. Peer 4 identified peer 5 departed the DTH, it makes peer 8 as its immediate successor and will ask peer 8 who is his successor (which is peer 12) and make it as second direct successor. When peer 5 departed, peer 3 check aliveness of its immediate successor peer 4 which still alive so, peer 4 will be its immediate successor and then its ask peer 4 who is his immediate successor now, since 8 is his immediate successor, 8 will become the second successor for 3.



So we can say when peer 5 leaves the DTH, then there will be no change in 3’s immediate successor which is 4, but there will be a change in its second successor, which has now been updated to 8.

Resources:

[1] ; TextBook: Computer Networking Top Down Approach 7th edition

[2]: Wikipedia

[3]Professor Slides