Michelle Emamdie
Homework 2
CNT 4007C

Homework 2

Problem 1

The transport layer will need to use UDP so that DNS can be used on the application layer side. Additionally, the transport layer will need to establish a TCP connection with the server once the DNS connects through UDP and then a GET request will retrieve the document.

Problem 2

Section 2.6.1 in the book, P2P File Distribution discusses the scalability of different Peer-To-Peer architectures and also provides examples of different implementations of P2P file distribution systems like BitTorrent. The book also compares P2P to a typical client-server architecture in great detail.

In comparison to a client-server architecture, P2P has a lower distribution time and can self-scale better. The distribution time in a client-server architecture increases linearly with the number of peers or in this case, clients, that the server must distribute the files to. This makes the client-server architectures less efficient in this sense. Since the server should send a file to each client, this puts a strain on the server and the amount of bandwidth that the server requires to maintain normal functionality. Conversely, in P2P systems, instead of only having one server distributing to many clients, there are multiple peers that have pieces of files that can be redistributed to other peers. This shows that there is no centerpiece of the system in P2P as there is in client-server architectures. This reduces the strain on any one part of the system and allows the same distribution of bits throughout.

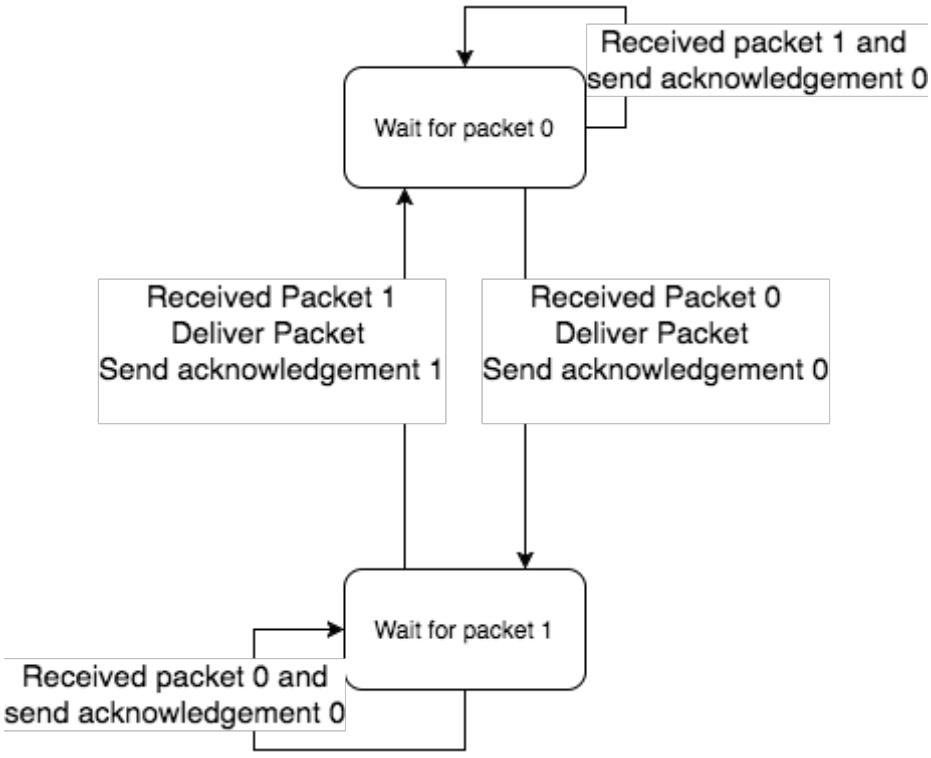
In regards to the distribution time of a client-server architecture, the book explains that the server has to transmit one copy of the file to each peer. So, the calculation for the distribution time of a client-server architecture would be the number of peers multiplied by the number of bits in the file, divided by the server's upload rate. In the P2P architecture, there are many peers distributing each file and they are able to work together to distribute a single file. While the server must introduce the file to one of the peers, distribution time once inside of a P2P community is quicker. Each bit is uploaded at least once so that the link can be accessed but beyond that, the distribution time is the file size, divided by the upload rate. When comparing the distribution rates of client-server architecture to P2P architectures, minimum upload rates are taken into consideration for a standardization.

Next, the book discusses BitTorrent and examines the successes they have had. BitTorrent is a popular P2P file sharing application where peers participate in the distribution of a file, otherwise known as a torrent. Peers join a torrent when they have no file chunks in an attempt to acquire the full torrent file. As time passes, they get pieces of the file while simultaneously uploading to another peer, the chunks that they have already acquired. This is called seeding. When a peer has downloaded a torrent, so that other peers can easily access the torrent. Since

torrents can have a nearly unlimited amount of people join, this allows for the load to be split across all the peers involved. Each peer provides a different piece of the file and neighboring peers are able to share pieces they have with their neighbors. There is a technique called "rarest first" which finds the piece of the file that the fewest peers have. That piece is shared first, and then the rest of the file. This allows for rapid redistribution of the rare parts in an attempt to make them more profound in that torrent's community. Then, once the piece is spread more evenly, the next rarest chunk will be sent first and will become the rarest.

To successfully operate, BitTorrent uses a trading algorithm that allows peers to change the priority of seeding for their neighbors. They are allowed to change the rate at which they can supply data for their peers so that the holder's internet is not slowed down. The rates are constantly being checked to make sure that the receiver is indeed receiving the piece of the file at the rate at which the holder specifies. In another effort to alleviate strain on each individual peer, the peers change about every 30 seconds to randomize the pool of peers. The existing peers get prioritized as "unchocked" while the randomized peers are "optimistically unchoked". Chocked peers do not receive any of the file chunks from the main peer.

Problem 3



This is the FSM I created to show what the receiver side of RDT 3.0 looks like. The first thing we must know is that RDT 3.0 can only send one packet at a time. So, the system starts out by sending packet 0. Once packet 0 is sent and the lower box acknowledges this, then the lower box sends packet 1 to the upper box. What happens is the state only changes when a packet is

received. This diagram just shows what happens when there are two packets but can be expanded to a circular form if there are multiple packets being sent.

Problem 4

If the probability of a segment being damaged is for example, 0.25. Then, the probability it fails is 1-p. So, we would need to send **1/(1-p)** segments to guarantee the segment is transmitted properly.