Q1. What was the biggest challenge that you encountered in this assignment?

The hardest part of this assignment was making sure that classes that needed to be thread safe were in fact thread safe. When your critical classes are not thread safe, really weird errors occur. For instance, I found that I had multiple threads reading different fragments of the same packet because I didn't ensure that only one thread could read from one client at one time, so my client send a packet with one hash, and my thread would read a packet with a different hash. Thread safety is something that should be considered whenever you make a new class because once you have 10 classes to manage, it gets difficult to track where each thread is.

Q2. If you had an opportunity to redesign your implementation, how would you go about doing this and why?

Given the opportunity to redesign my implementation, I would probably start with all of my thread safe actions. At the moment, I'm not exactly sure where they all are. That being said, I would like to go through and group them together if possible, and possibly consolidate them if I can. As it stands I believe my implementation is over-liberal in the use of the synchronized keyword, so if I can get rid of some of those chunks, then I'll be able to increase the concurrency of the code. Also, I would like to change my package structure for this program. I feel this too can be consolidated and still provide this projects with the benefits of using multiple packages.

Q3. How well did your program cope with increases in the number of clients? Did the throughput increase, decrease, or stay steady? What do you think is the primary reason for this?

As the number of clients increases, the throughput increases until it hits a ceiling, assuming the number of thread remains constant and assuming that the network medium stays the same. If the program is run with a thread pool size of 10, and each packet is 8KB, no matter how many threads it has, it can only read the 8KB so fast. As long as there is a message to be read for each thread, then the system has reached its maximum through put and adding clients will not affect that at all. This is because throughput is reliant on the server reading and handling packets, and that relies on the thread pool, so if all of the threads are always active, except for time it takes for the thread pool manager to assign it a new task, the server has reached it's maximum throughput. The way to push the ceiling is tho increase the number of threads, not clients.

Q4. Consider the case where the server is required to send each client the number of messages it has received so far. It sends this messages at fixed intervals of 3 seconds. However, since each client has joined the system at different times, the times at which these messages are sent by the server would be different. For example, if client A joins the system at time T0 it will receive these messages at {T0 +3, T0 +6, T0 + 9, …} and if client B joins the system at time T1 it will receive these messages at {T1 + 3, T1 + 6, T1 + 9, …}
How will you change your design so that you can achieve this?

As it stands, I have a timing mechanism set up in my server to display its output every minute, so all I have to do is attach a timestamp to each client when it registers with the server, then every time I print, I'll have to check to see if I need to send this new message to any clients. I know this design will force the system to take a performance hit, but this check will only happen every minute. If the node is sent a message, then its timestamp is updated to the current time. As of this time, that's the simplest way to add this feature to the implementation I have.

Q5. Consider the CDN that you designed in the previous assignment. This CDN must support 10,000 clients and the requirement is also that the maximum number of hops (a link in the CDN corresponds to a hop) that a packet traverses is not more than 4. Assume that you are upgrading your CDN routers using the knowledge that you have accrued; however, you are still restricted to a maximum of 10 threads in your thread-pool and 100 concurrent connections.
Describe how you will configure your CDN to cope with the scenario of managing 10,000 clients. How many router nodes will you have? What is the topology that you will use to organize these nodes?

When the CDN is set up, it will create 10 servers, when the nodes register with the discovery node, they are randomly assigned a server node to connect to, such that each server will have an equal number of nodes connected to it. These servers will serve as the routers in the CDN and they will form the MST to have the messages get to the right destination with the smallest number of hops. In this scenario, the only functionality that I would have to give the CDN is to distribute the clients to the servers appropriately. This reduces the likelihood of a duplicate packet from ever reaching any node within the system because it will have to go through its server first. This networks basically becomes a small world network because you have clusters that are connected though a small number of links.