Nedap University 8

Module 2

Network Systems

Final assignment

Author: Anniek Bisschop

Date: 20-4-2023

Design

The application includes a client package and a server package. In addition to these packages, I have a protocol class that is used by both the server and the client. I would like to add a separate clientTui for user-friendliness. Both packages have a Main class from which the client and server can be started.

Server: I have chosen to keep the server as "dumb" as possible so that it only needs to be started.

At the start, I first made a drawing to see how I could shape the application. This helped with the design. At the beginning, I kept in mind that I wanted to add a protocol class later on. I did this in a later stage because I had a better overview of which functions were shared by both the server and the client.

Because I felt that the first connection via the datagram packets was established fairly quickly, I wanted to try implementing a Go-back-N protocol. After spending a day on this, I decided to fall back on the simpler stop-and-wait without sliding window because I couldn't set it up properly. I found it more important at this stage to choose something that actually worked.

Protocol

I used UDP with stop-and-wait ARQ protocol. Since UDP itself has a header with port and destination address, I decided to create a separate header for the stop and wait. This header has 4 bytes for sequence number and 4 bytes for ack. This is done with the aim of possible future implementations, so that something else can be added if necessary.

If a function does not receive the correct sequence number, it will continue trying to send that packet up to 100 times. In addition, there is a timeout function that takes into account that this sending process can also take too long. In that case, there is a timeout so that the server does not remain unreachable forever.

Protocol class

It defines methods to create headers for packets, send and receive acknowledgments, and receive data packets. It also has a method to compute a hash value for a given byte array.

The createResponsePacket() method has two overloads, one that takes a String message and one that takes a byte array. Both methods create a packet with a header and append the message or byte array data to it. The sendAck() method creates and sends an acknowledgment packet with a new sequence number, while the receiveAck() method waits for an acknowledgment packet and returns it.

Finally, the checkFileHash() method takes a File object and a hash value and returns true if the hash of the file's contents matches the expected hash. This method can be used to check the integrity of a file that has been transmitted over the network.

Testing

Because of a shortage of time to finish the project, I wasn't able to write a unit test. I also found it difficult to figure out how to test it properly between the client and server. So I choose to test more thing manually.

Many tests were done manually by setting timeouts or disabling certain functions so that packets couldn't be received. This helped me discover, for example, that certain retransmits were not being done, but only a System out was written. Additionally, the hashing function was very helpful. By making it a boolean, I could quickly see if files were corrupted or not. Furthermore, I sent various types and sizes of files frequently

Areas for improvement

I would have liked to write unit tests to be able to test more effectively.   
The hashing function currently only returns a boolean and if it is "false", it only indicates that the file may be corrupt, nothing else happens. I would have liked to further develop all functions.   
The code that has been delivered contains some duplicate code and can be a lot cleaner at certain points, such as upload and download.   
I would have liked to put these in separate functions. I also do not handle some errors very gracefully.   
As mentioned earlier, I would have liked an extra client TUI and perhaps even further divided the code into smaller pieces.   
The naming for the upload functions and download functions could have been better because they can be confusing. For example, I have a downloadFromServer() function that handles the download request from the client. This could also be interpreted differently.  
I would like to have more consistency in the code.  
I have some unused params in some functions but I don’t want to remove them at this time because it will may cause errors.  
Not all functions have a fully implemented stop-and-wait

What I am satisfied with

Firstly, persistence in testing, searching, and debugging.   
Although it sometimes took a lot of time, I often managed to find and solve the problem myself.   
Thinking about how I can convert things in my head to code.   
Unlike the previous project, I started writing code faster and tried more things.   
I have made frequent use of git.  
Javadoc in protocol class.  
Using a overloaded method.