I have exported my Chatbot program into an executable JAR file located within the root of my project directory. To run my program just run the Chatbot.jar file. I have made a GUI for this program that will be displayed in two windows. The first window is the client view where you can set the client ID, error percentage, and trace options. The second window is the client view where you can set only the trace options.

Once the Server and Client have been set you can start chatting. Messages are typed at the bottom of the client window and sent using the “Send Message” button. The server will check the message for errors, send a NAK message when an error is detected, then pass the message to the rule engine to get a reply. Once the server receives a message with a word containing “bye” it send an end transmission message back to the client which then displays the communication stats.

My Chatbot has 5 rules and an additional “bye” check to terminate transmission. The rules are as follows:

1. If the message contains HI or HELLO the server will respond with, “Hi, XYZ! How are you?”
2. If the message is over 30 characters in length the server will respond with, “XYZ you really like to talk don’t you?”
3. If the message contains a “?” character the client will respond with, “XYZ, no one has ever taken an interest in a robot like me before. Thank you!”
4. If the message contains the words “computer” or “program” the server will respond with, “XYZ I am more than just a computer program you know.”
5. If none of the other checks come up with a response the default message will be, “I really cannot understand what you are saying. Sorry.”

I tested my program 5 times with error rates of 0, 50, 60, 75, and 90 percent. Each time I sent 10 canned messages, with the final message being “goodbye.” At 0 and 60 percent the number of request directly matched the theoretical number of requests that was calculated. At 50 percent we actually had more client requests then the theoretical number, and our maximum amount of retransmissions was five. Finally, for 75 and 90 percent error rates, we had significantly less client requests than expected from our theoretical predictions; 6 less and 29 less respectively.

Many of the other programming projects I have been assigned have focused on certain data structures or specific programming methods. This project was unique and educational in that it had a specific task that needed to be completed and we were able to fully use our previous knowledge to accomplish this task. Additionally, this project forced us to work with another person’s code (namely Professor Lin). This was a learning experience in having to trust the interface, the implementation, and usage of each class.

I also learned how networking is implemented on the application level. Aside from the project requirements I researched more about the DatagramSocket class. I learned about its uses, limitations, and how it can be implemented to send packets between applications.

The main obstacle I ran into during the creation of this chat program was the actual construction of the byte array. At first I tried concatenating everything into a string and then converting that to a byte array. I got this to work but the message length and checksum are not limited to 1 character (0-9) numbers. So I had to think of another solution and ended up having to assign the byte array elements by using for loops and index numbers.

Finally, I thought this project was very good in getting us to think about networking on the application level. It allowed us to be creative and utilize the programming skills learned in other courses. The only suggestion I have would be to clarify requirements 6 and 8. Requirement 6 specifies that the probability of 1 error is 75 percent and 2 errors is 25 percent. However, in requirement 8 for testing we never actually test anything with 2 errors.

As I was creating this program I did many debugging along the way. As I finished each of the requirements or finished coding each of the procedures I debugged to make sure the user was not able to crash the program by entering in excessively long strings, incorrect number, ect. As of right now there are no known bugs.