IRC Client Development Report

This project is developed using Python v2.7 and PyQt 4 on a Windows 7 OS.

The development was started by copying the template code in the requirements document. At first all threads were saved in a different python document. The two files “incomingParser.py” and “outgoingParser.py” were discarded later after seeing that it was more efficient to develop and use the methods defined in a single python script file. Currently only valid file is “chatClientMain.py”; which contains all the required code inside.

Since I did not have a lot of know-how about Qt, I used the interface code as it is and did not change anything related to that part of the code except the modifications to display the user list.

I first started off with the parsers, which are required to translate commands and displayed messages between client and server. At first I created very basic parsing to make sure command passing was successful between the server and the client. Initial form of the parser lacked some requirements such as “timestamp”, “newline at the end of messages” and “SOK, YOK, TOK, MOK responses to the server questions".

To connect to the server and give the basic version of the IRC client a go, to test the progress so far I then moved on to implementing the section where connection parameters were elaborated. I decided to use the **format** **“IP:Port”** as input parameter from the command line since it is the more intuative way to write an IP-Port pair.

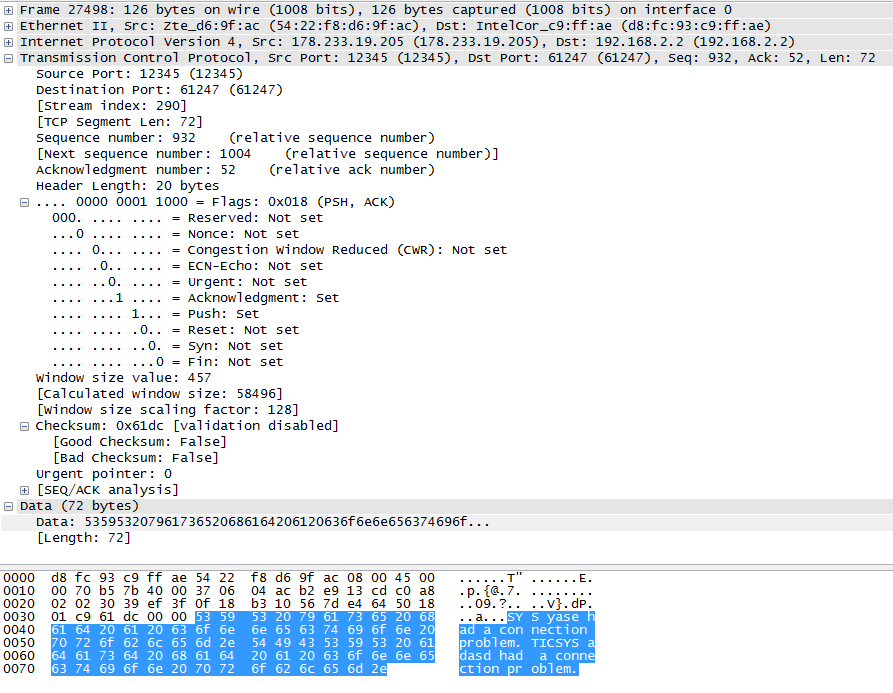
I fixed the bugs I discovered in the system first, then I improved the code to better correspond to the requirements by adding timestamp and newline to the end of messages. I also added the responses to server questions.

The last thing I implemented was the user list, since I tried to do it but ended up breaking the whole Qt code. Experimenting with Qt I finally found a way to pass the user list from ReadThread to the Qt interface.

Server Bugs

1. Server cannot handle the responses to ites own requests. When “SOK”, “YOK”, “MOK” or “TOC” responses are sent to the server, server tries to interpret them as requests and responds with “ERR”.
2. Server sometimes cannot handle sending multiple messages. This is most appearent in “SYS” messages. It sometimes sends multiple messages instead of a single one. Appearently server needs some sort of buffer and lock mechanism. Example can bes seen in the following screenshots:





Issues in the Template

1. It may be bad implementation at my part but the exception block in the template:

try:

self.csoc.send(queue\_message)

except socket.error:

self.csoc.close()

break

causes the connection to end prematurely everytime after a message is sent. So I discarded that section altogether.