COMP 445 – Winter 2021

Data Communications & Computer Networks

Due Date: March 1st 2021

Lab Assignment 2: Network Socket Programming

In this lab, you will be exposed to Request for Comments (RFC) and learn the basics of using sockets to create an IRC client app and an IRC server.

This lab assignment will be split into two parts. There is the lab part which will act as a tutorial to build up your knowledge and the second part is the implementation of a simplified IRC client and server.

During the lab part, there will be a series of questions that you will need to answer and include in your report. These questions are there to guide you and help fill in the knowledge you will need to properly implement the IRC client and server.

Objectives

- Gain experience understanding a network protocol by reading an RFC
- Capture, filter and analyze network traffic using Wireshark
- Apply transport protocol concepts to connect apps across a network using low level socket interface
- Study and implement IRC application protocol in both client and server configuration at both communication ends

Background

The IRC (Internet Relay Chat) is an application layer protocol for use with text based conferencing which saw the light of day in the early 90's. The protocol was officially introduced in RFC 1459 and was subsequently revised in RFCs 2810 (IRC Architecture), 2811 (IRC Channel Management), 2812 (IRC Client Protocol), 2813 (IRC Server Protocol) and 7194 (IRC TLS Port). IRC clients are programs that end-users run to connect to IRC servers. Through these clients, they can send private messages to users or to channels. IRC typically uses TCP as its transport layer.

Introduction

This lab will introduce you to transport protocol connectivity using socket programming, application protocol basics taking IRC protocol as an example and network traffic analysis using Wireshark. You will learn how to program a client/server network application and analyze the live traffic as it traverses the routes from point to point. you will also learn how to study a protocol from scratch by reading RFCs and use your socket programming knowledge you acquired to transform it into a functional network application. This lab is divided into three parts as follows:

1. Part 1 will investigate the first RFC related to IRC (RFC 1459). You will learn how to read RFCs and similar technical documents for the sake of understanding the message structure anatomy and functional behaviour of any network protocol..
2. Part 2 will demonstrate what IRC packets look like in transit. This will be accomplished using one of the most famous network analysis software, *Wireshark,* that is used to capture and analyze traffic in live or simulated networks.
3. Part 3 will focus on basic socket programming concepts that you need in order to implement the simplified IRC client and server.

Request for Comments Deep Dive

In this part, you will start by learning more about the protocol that will be implemented in the assignment part of the lab. For this, you will need to read up on the RFC 1459, entitled "Internet Relay Chat Protocol". While going through the document, answer the following questions and note them in your report. If you are having difficulty reading the message format in Backus-Naur Form (BNF), take a look at RFC5234 which gives information relating to the syntax.

Questions (2pt each):

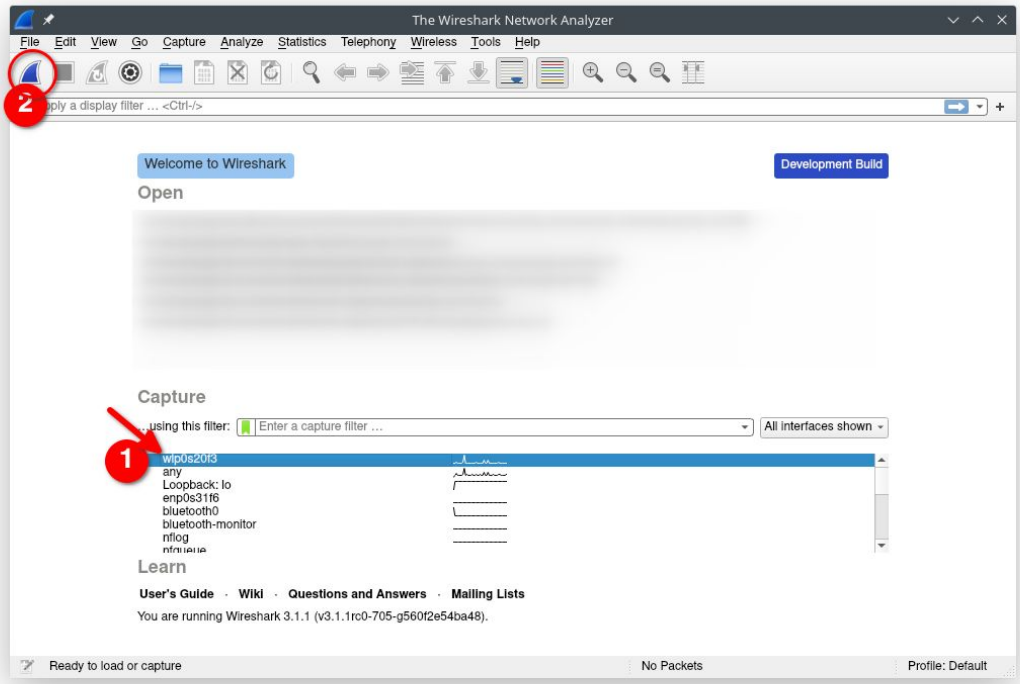
1. What uniquely distinguishes clients on a server?
2. What information must all servers have relating to users?
3. How many bits are used in the protocol to define a character?
4. What is the binary representation of the character that is used to separate commands and parameters?
5. What is/are the character(s) that mark(s) the termination of an IRC message?
6. What's the maximum length of a message?
7. What is the only valid prefix that a client can put in a message?
8. What does a server do when it receives a numeric reply from a client?
9. What is the content of the reply that a server generates when it receives a NICK message from a client that causes a nickname collision?
10. Name one security issue with the protocol? Refer to the relevant section in the RFC 1459 or its updates.

Packet Analysis of IRC Connection

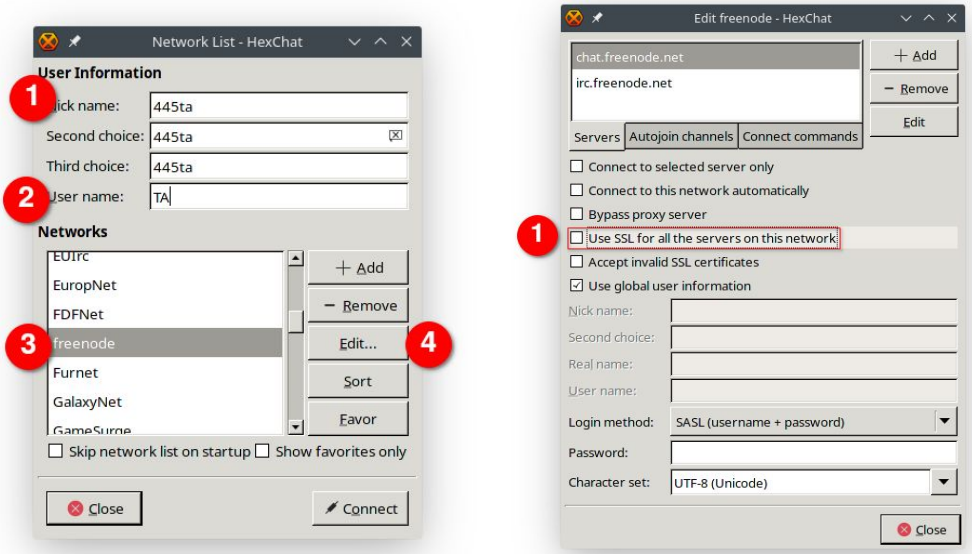
In this second part, we will use Wireshark to capture network traffic while using an existing IRC client. However, emphasis on the content of RFC 1459 should be taken into account. Newer features of IRC might be encountered while analyzing the traffic and should not skew the implementation of your assignment.

Steps:

1. Download and install Wireshark.
2. Download and install HexChat (IRC Client) (for Windows users, use the Windows 7 Installer; for Linux install from the source code)
3. Start capture of traffic



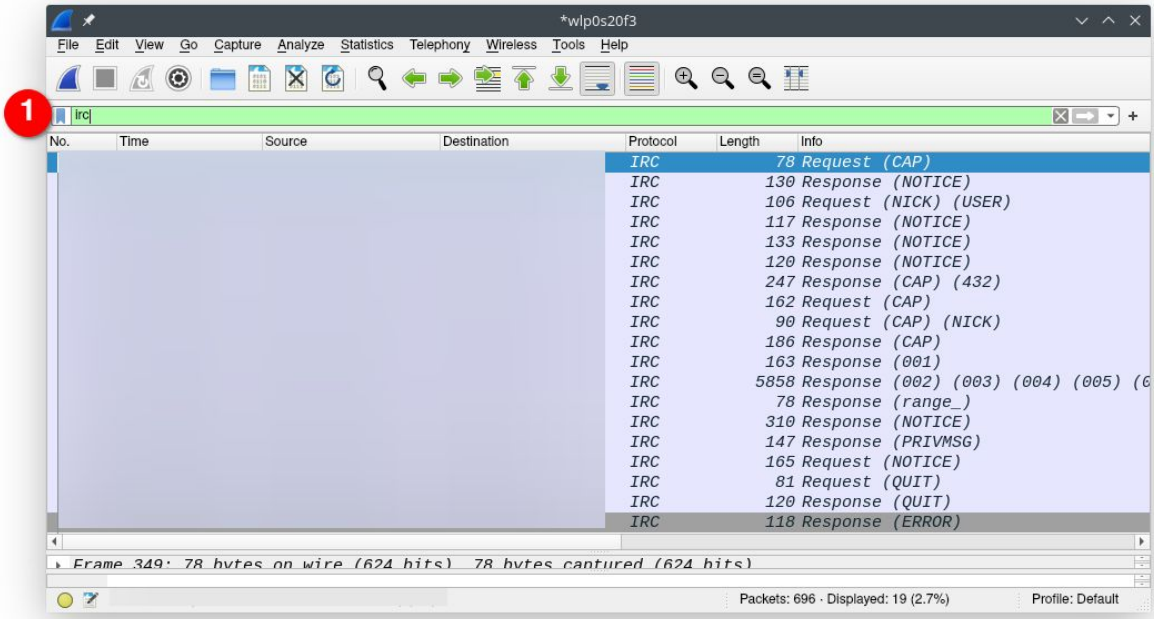
   a. Open Wireshark
   b. Select the network interface you want to capture
4. Open HexChat and connect to freenode.net (this will require the registration of a username, you can use a temporary email)



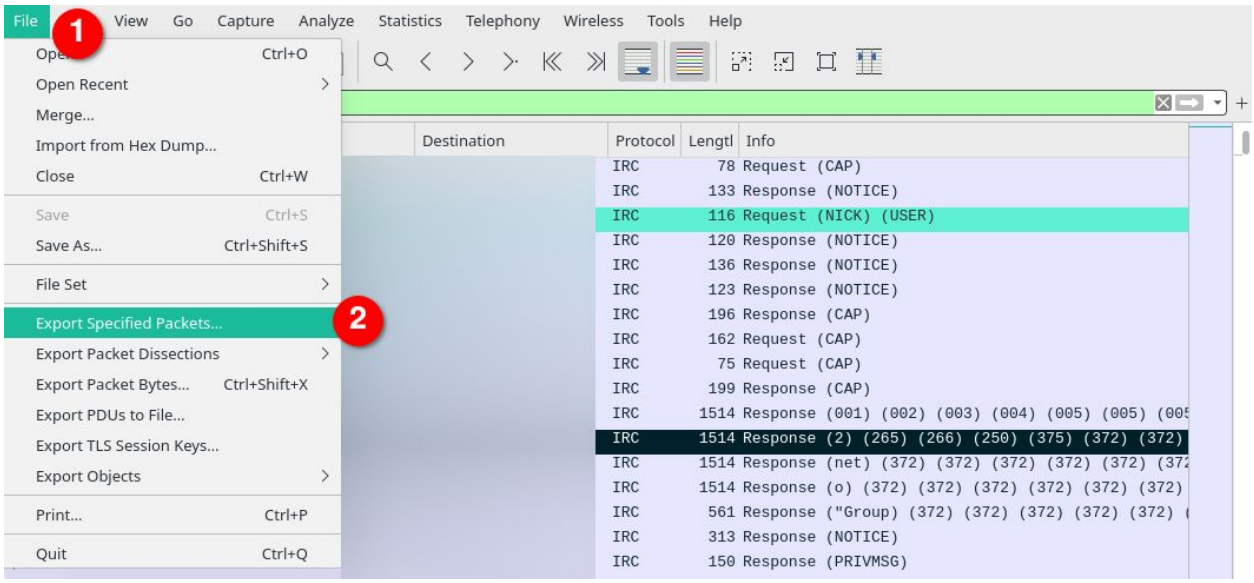
5.

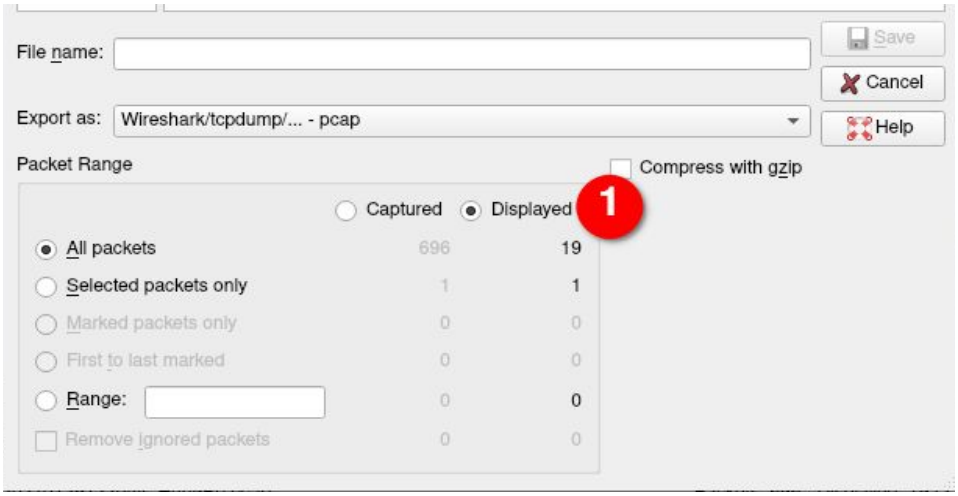
a. Select your nick and username
b. In the networks box, select the server.
c. To be able to view your traffic, make sure to disable the option "Use SSL for all servers on this network" by editing the connection.(this will disable encryption; not recommended outside of this lab)
d. Press "Connect"

6. To only view IRC related traffic in Wireshark, add the filter 'irc'



7. To export only the visible traffic, go to *File* -> *Export Specified Packets* and make sure *Displayed* is selected.





<u>To answer the following questions, you need to select the 'pcapng' file that is associated with your team version number.</u>

Questions (3pt each):

1. Find the destination port that the client connected to on freenode.net
2. Find the nickname that the session is using on the server.
3. Find the period of time that the user is connected to the server
4. Find the last message that the user sent to the channel
5. Find the connection password

Simple Client & Server

The following section will be an introduction to socket programming. To get started with the assignment, you will need to set up your development and testing environment based on Python. It is recommended to use any version higher than 3.6 (included). It is suggested that this section be split between members of the team. Furthermore, the skills you develop using socket programming can greatly benefit you in your upcoming career.

Make sure that all members of your team have a similar environment so as to minimize issues down the road. We urge you to take advantage of version control systems such as Git and make use of branches. For more information, consult Git Cheat Sheet - GitHub Education. If you use a hosting website, make sure the visibility of the **repository is set to private**. Another suggestion is to use a virtual environment while developing your code. This will allow you to specify which versions of packages are included in your project. Doing so will remove ambiguity while working as a team and minimize chances of the grader not being able to run your code.

In order to get familiar with socket programming, it is suggested to build a client that sends user input to a server, which in turn echoes it back. A tutorial to do so can be found here.

Bonus (optional) After understanding the basics, the next step is to configure your server to be able to service different clients in a non-blocking fashion. One way to enable said feature is to set the input socket on the server to a non-blocking mode and using the select library to handle incoming sockets that are ready to be read.

Assignment Part

Requirements

As a reminder, you **may only use the socket library** to connect your applications across the network. The use of ready made IRC libraries or higher level libraries for network communication will render the assignment as INCOMPLETE and will result in penalties proportional to the level of abstraction and functionality done by the library that you are supposed to implement and did not.

Based on RFC 1459, implement a client and server that follow the specifications. In order to lighten the load of the assignment, certain assumptions are made and mentioned below.

Assumptions for Simplified IRC

1. Clients need to be able to register their connection with the server
2. Servers do not handle connection passwords (and the PASS command)
3. Upon successful connection registration, clients are automatically added to the #global channel
4. No channel management actions need to be considered (joining new channels, modifying them, …)
5. There does not need to be any operators on the server.
6. For messages to be visible to other clients, they need to be sent to #global
7. Private messages to users do not need to be considered
8. When the client quits the server, proper handling of the sockets needs to occur on both ends

Provided the above assumptions, you should **implement the minimal set of commands and replies** that allow for the assumptions presented above to properly function.

IRC Client

The client application is split between the front end presentation of messages and the backend. The backend is the implementation of the IRC client protocol which reads and writes to the IRC server.

Given that this is a networking class, a front end terminal user interface template is provided with the assignment. Thus, the remainder of the work to be done on the client is focused on the backend. The content of which can be broken down to parsing user input, crafting packets in accordance with the protocol and managing the socket connection to the server.

The command line interface (CLI) for the client must allow for the specification of the target server and port with which to connect to. Keep in mind that connecting to the server should be delayed until the user defines their nickname and username. An example of the interface can be:

```
→  Documents python irc_client.py --help
usage: irc_client.py [-h] [--server SERVER] [--port PORT]

optional arguments:
  -h, --help       show this help message and exit
  --server SERVER  Target server to initiate a connection to.
  --port PORT      Target port to use.
```

In the client front end, it should be clear from whom a message originates from. If it is a reply from the server indicating an issue, it should be expressed to the user and if necessary dealt with.

Server

The server should have a CLI option to bind its port. An example of what the interface may look like is:

```
→  Documents python irc_server.py --help
usage: irc_server.py [-h] [--port PORT]

optional arguments:
  -h, --help   show this help message and exit
  --port PORT  Target port to use.
```

In addition to what is mentioned above, the server be aware of clients that are connected and subscribed to the #global channel. Include logging of the server's state as it will help you debug its implementation and will facilitate grading.

Included Files

Apart from this handout, additional files have been included. Their use will be described below:

- **irc_code**
  - **view.py** - Provides the View object which is a Terminal User Interface (TUI) for the client, uses ncurses library
  - **irc_client.py** - Template for the IRC client backend, demonstrates interaction with the View object
  - **patterns.py** - Contains classes used for Publisher/Subscriber Pattern that is used for the View and IRCClient
  - **banner.txt** - Contains the banner that gets printed upon opening the client
- **packet_analysis_version_<n>.pcapng** - File to use to answer the questions from Questions Packet Analysis.

It is suggested to use the provided code as a starting point. Keep in mind that you will have to modify sections of it and add components to fulfill the requirements and assumptions made previously.

To start the client, run with "**python irc_client.py"** and to exit the client type out the command "**/quit**". You are free to change the method of closure.

Deliverables

You need to create a PDF and zip file that should include the following:

1. **a2_code_<id1>_<id2>.zip**
   a. README.md describing usage of your client and server
   b. requirements.txt file containing all the packages needed to run your programs
   c. Source Code for client
   d. Source Code for server
2. **Lab Report** needs to include the answers to the questions in the Lab Section (PDF format)

When submitting to Moodle, make sure to upload the zip and the PDF lab report as two separate files.

Grading Scheme

- Report (50pts)
  - RFC Questions (20 pts)
  - Packet Analysis (15pts)
  - Design description (15 pts; max 250 words)
- Assignment (50pts)
  - Use of Observer pattern (10pts)
  - IRC Client (20 pts)
    - Establishing and closing socket communication with a server
    - Crafting and parsing of application layer packets
  - IRC server (20pts)
    - Establishing and closing socket connections with multiple clients
    - Proper initialization of IRC session
    - Propagation of user messages
    - Appropriate server replies
- Bonus (10pts)
  - IRC Server listens to incoming connections in a non-blocking manner (10pts)

Demo

After submission, a demo will be required that will testify your understanding of different lab components and genuineness of the report submitted. Every group member will be asked questions and grades assigned could be different for the same group members. Plagiarism is STRICTLY prohibited and so is any support from online repositories, forums, etc. failure to adhere to this will result in ZERO for the assignment and report to the school for academic misconduct.

During the demonstration of your client and server, some of the scenarios that you may encounter are connecting multiple users to the server, showing that the server responds with error messages where appropriate, testing edge cases of the protocol (within the bounds of the requirements). Please note that this is not an exhaustive list of actions that you may need to perform.