John David Vidad
2019–10432
CS 145 LAB 3

CS 145 Project Parameter-Adaptive Reliable UDP-based Protocol

1 Requirements

- **Level Declaration:** Level 3

2 Documentation

2.1 Code Explanation

The code is divided into 5 major parts: the possible changing of parameters, the getting of transaction ID, the estimation of payload size, the adjustment of the estimated of payload size, and the sending of the remaining payload

2.1.1 Changing of Parameters

With the assumption of valid inputs, we traverse the arguements by pair after the python file arguement.

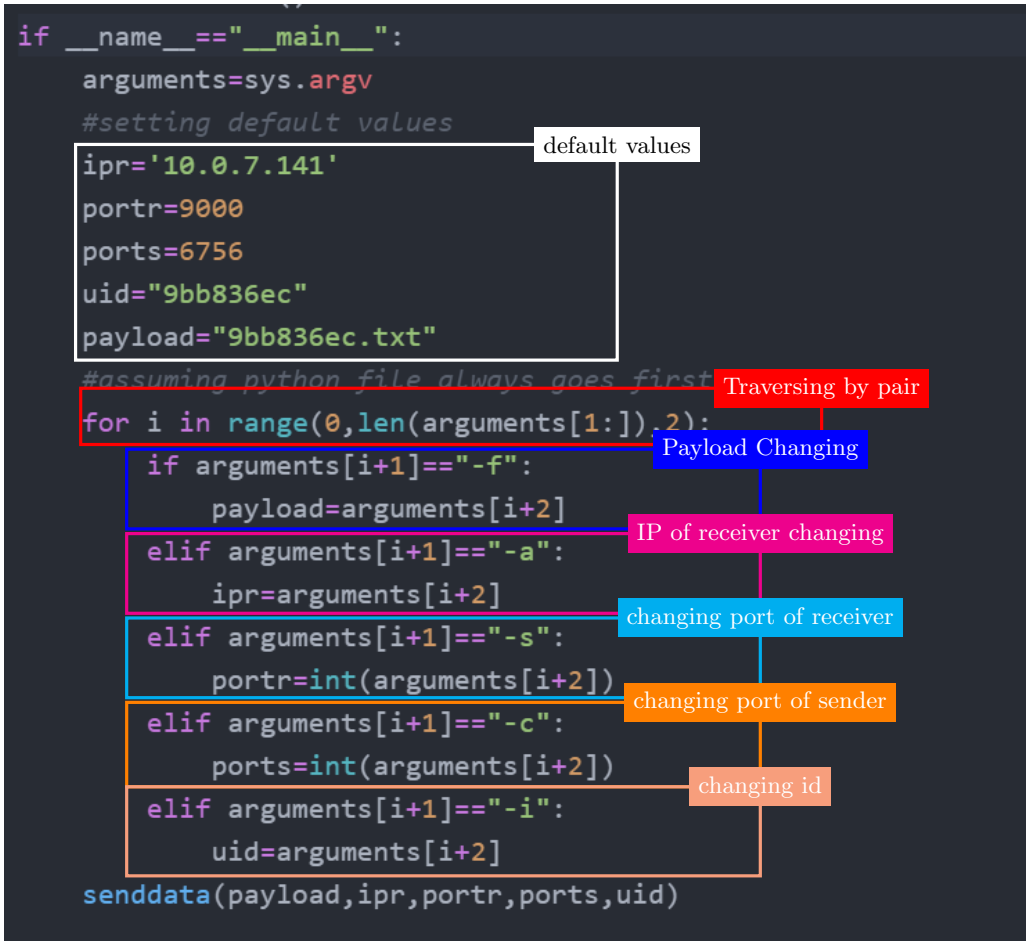

Figure 1: parameter changing snippet

Running a python file will have at least 1 arguement and via the argv of the sys module we are able to access it as list of strings with the first index being the python file. To show that this is working, we simply created a show input wherein it shows all the variables like ip of the receiver, the port numbers, the id, and he period. below is a Screenshot shot of testing and showing that the changing of parameters work.

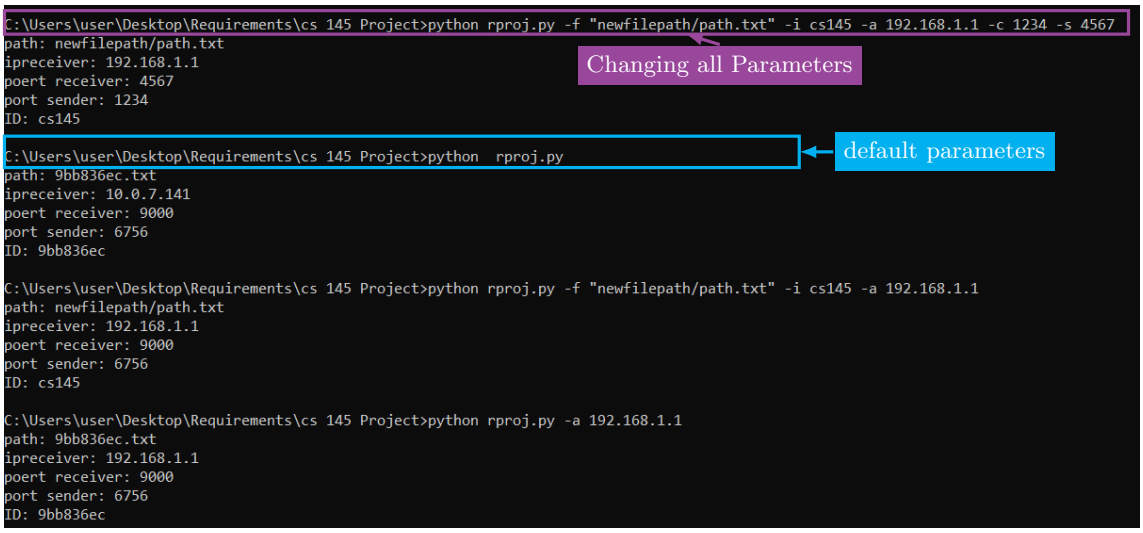

Figure 2: parameter changing test

Each Test case either changes the default values on indication or stay with the default, for a clearer view, all figures will be viewable via a google drive link provided in the Links section with names in the figure number.

For the first test, we changed everything with the following values, path: “newfilepath/path.txt”, IP 192.168.1.1, port receiver 4567, port sender 1234, id=cs145 and have the following changes as shown to be what it is inteded to do.

For the second test, we ought to see the default values thus, we did not change any of the parameters. The output is as expected, basing from the default values from figure 1.

2.1.2 Initializing of transaction

For this code section, there are three things that will happen: Initializing sender port, loading payload, and connecting for transaction ID.

```
12      client= socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
13      client.bind(("",port_sender))
14      client.settimeout(30)
```

Figure 3: Port Initalization

The initalization happens by creating a socket.socket for the UDP. after creating a socket variable, we bind the port fo sender parameter to the socket, so that it will listen to that packet. To counter busy servers, we set an arbitrary timeout of 30s, basing on the payloads we get, they don’t normally take 15 seconds, but just to be sure, we set a timeout of 30.

```
18      try:
19          datafile=open(path,"r")
20          data=datafile.readline()
21          datafile.close()
22      except FileNotFoundError:
23          print("File Does not Exist")
24          return
```

Figure 4: Loading fo data from path to memory

After the initialization of the port, we load the whole payload to the memory. We have a catcher for a FileNotFoundError to catch a file not existing error and it will end the program if such happens, after telling the user it does not exist. In line 19, we open the file with a read only type of access ('r' tag) since we only need to read it. Assuming no multiple lines of data exists (this is based on testing), on line 20, we read the first line of the file, which is where the payload is contained. AFter reading it, we save it to a variable named data. In line 21, we close the file.

3 Links

- **Google Drive Link for pictures:**
- **GitHub Link:** https://github.com/anagramiccoder/cs145proj
- **Video Link:** https://github.com/anagramiccoder/cs145proj