Question 4

As part of this problem, you will implement a simple messaging the system that is implemented in Python. The system has two basic components: a *chat client* that will be used to send messages and *topic server* that will be used to map a user identifier to an IP address. The chat clients will talk with the *chat server* using TCP with the topic server. The code for the chat client will be include in `chat.py` and may run either in "direct mode" or "topic mode". In the direct mode, the client will directly talk with another client. The command line arguments are the following:

- Argument 1: The first argument is "–direct" to indicate that the client will run in "direct mode"
- Argument 2: The second number is number indicating the port number the chat client will be waiting for incoming connections. If the number is zero, the program will act as a client and not open the listening port.
- Argument 3: The argument is supplied only the second argument is zero. If you are a client, the argument is a string that include the IP address and port number for the client that you will be talking to. For example, the string *192.168.1.5:2000* indicates that the IP address is 192.168.1.5 and the port number is 2000.

As an example, you may run two clients in direct mode by running the following commands.

- In one terminal window, you can start the chat program as a server by running it as follows:
  `python chat.py -direct 10000`
- In another terminal window, you can connected to the chat program as a client using the following arguments:
  `python chat.py -direct 0 127.0.0.1:10000`

In indirect mode, the client will talk with multiple other clients that are interested in the same topic. Note that a topic is just a user-provided identifier. In this mode, the command line arguments are the following.

- Argument 1: The first argument is "-topic" to indicate that the client will run in "topic mode"
- Argument 2: The second number is a string that include the IP address and port number for the topic server. The string is formatted as above.
- Argument 3: The topic that the client is interested in.

As an example, you may run two chat clients in topic mode by running the following commands

- First, start the topic sever listening on a given port.
  `python topicserver.py 20000`
- In another terminal window, you can connected to the chat program as a client using the following arguments:
  `python chat.py -topic 127.0.0.1:20000 lame-conversation`
- In another terminal window, you can connect another client by issuing the same command.

The operation of the chat client has two phases: registration/lookup and chat.

- **Registration:** The client will register with the directory server that the user is using the IP address of the host on which the chat client runs
- **Chat:** The client will send a chat message to the topic server and receive message from the topic server when other clients send their data.

In both modes, the client will read input from the keyboard. Each provided line will be sent in a single chat message whose format is detailed below. When a chat client receives data from a peer chat client or the topic server, it should display the received message. The display should have the following format using the following statement: `print("%s:%s n" % (topic, message)`. You should use `select` to multiplex the reading from the keyboard and the reception of bytes over TCP. The clients will exchange messages that conform to the following format:

In both direct and topic modes, the messages exchanged by the clients will be valid JSON using the following format:

```
{
        "source" : {
                "ip": "192.168.1.1",
                "port" : "2000"
        },
        "destination" : {
                "ip": "192.168.1.5",
                "port" : "2032"
        },
        "message" : {
          "topic" : "lame-conversation",
          "text" : "This is such a lame message"
        }
}
```

The service will include as `topicserver.py` and will maintain the mapping between topic identifiers and a list of sockets from all the clients currently connected in a hash table. The registration message will include information about the topics that the chat client is interested and have the following structure.

```
{
        "source" : {
                "ip": "192.168.1.1",
                "port" : "2000"
        },
        "topics" : ["lame-conversation", "other-conversation"]
}
```

The topic will handle a message from a client as follows. It will first check if the client is registered for the topic included in the message. If this is the case, the message will be broadcast to all the clients that are currently connected to the server. Otherwise, the message will be discarded.
You can obtain partial credit if you get part of the problem correctly implemented:

- Correctly implement the direct mode where the chat clients interact with the topic service (50% of the points).
- Solving the remainder of the problem will earn you the remainder (50% of the points).