ASSN4 Internet text/voice/video chatting program

20150987 컴퓨터공학과 김성태

20150842 컴퓨터공학과 이다영

## Introduction

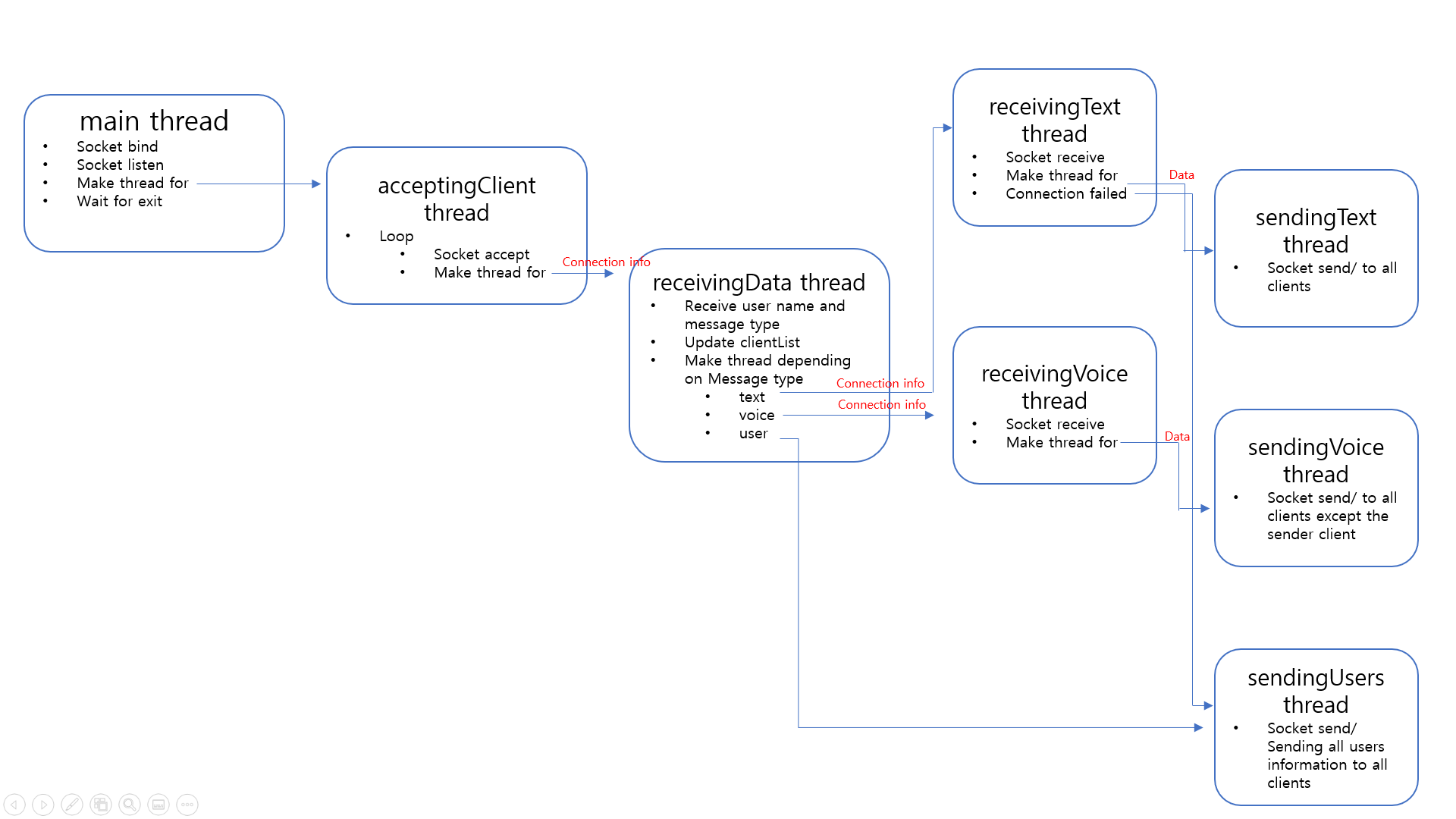
The main purpose of Assignment4 is building an Internet text/voice/video chatting program using sockets. Our program consists of server and client. For server side, it receives data from clients and sends the data to the other clients. For client side, it sends data to sever and receives data, which other clients sent to the sever, from sever.

## Design

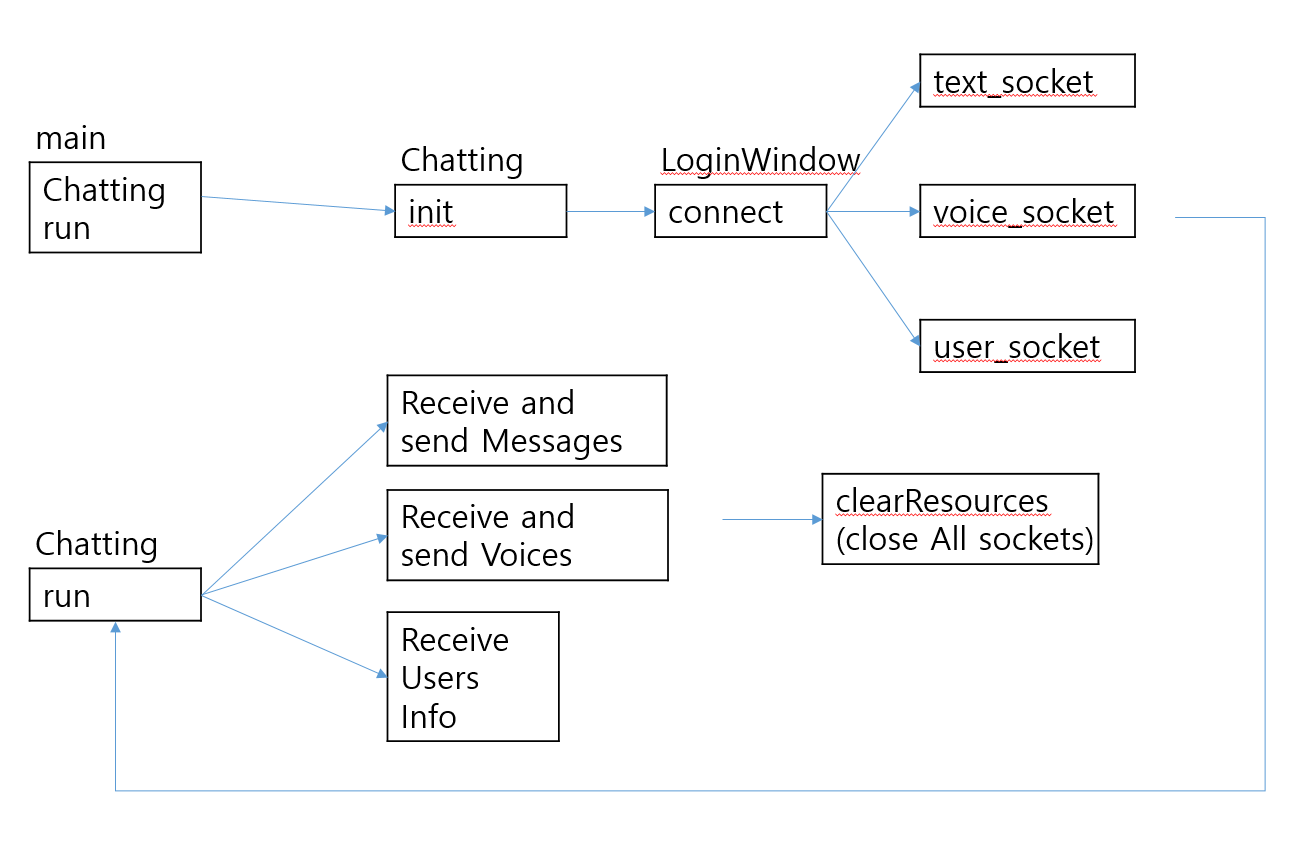
### Server

* The followings are functions, which should be supported.

1. Socket bind/listen
2. Socket accept/Register clients
3. Receive text/voice/video message from clients ( socket receive)
4. Send text/voice/video message to clients. ( socket send)



### Client



## Implementation

### Server

1. Main Thread

It creates socket, binds and listens. Then it makes acceptingClient thread for making new connection with client. After that, it waits until server is shutdown with keyboardInterrupt.

1. accecptingClient Thread

It accepts clients and when the client is accepted, it creates new thread for receivingData.

1. receivingData thread

It receives the data from the connection, which is made in acceptingThread. First received data was regarded as connection\_type+username. Therefore, it checks the connection is for text, voice or user. After specifying the connection type, it gets username by parsing the message. Then, it adds the connection info with user\_name and connection type in clientList. At the last, it creates receivingText or receivingVoice thread, depending on the connection type.

1. receivingText thread

It receives text Data from connection. When the text data is received, it creates new thread(sendingText) for sending the received text data to all clients. Also, it deals with the connection status. When the connection with client is broken, socket.receive() will raise connection Error so it catches the error and de-registers the client from clientList.

1. receivingVoice thread

It receives voice Data from connection. When the voice data is received, it creates new thread(sendingVoice) for sending the received voice data to all clients except the sender.

1. sendingText thread

It sends Text data to all registered client in clientList with socket.send().

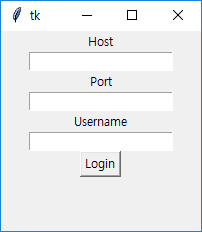
1. sendingVoice thread

It sends Voice data to all registered clients in clientList except the sender with socket.send().

1. sendingUsers thread

It sends all clients information to all registered clients in clientList.

### Client

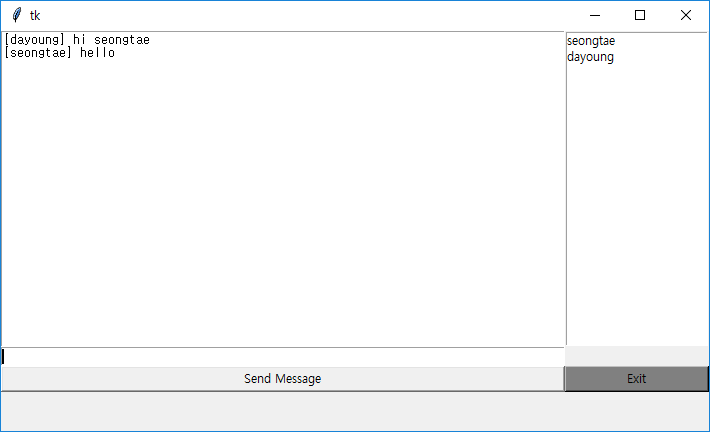
1. Login Window

We implemented GUI for this assignment. At first, the client should connect the server for communicating with other clients. Above picture shows a connect window. For connecting to server, the client should give information of Host, Port and Username. The Login button binds to loginClick function. After click Login button, Login Window will be destroyed.

1. Connect

As user clicks the Login button, the client tries to connect with server by text, voice, and user socket.

1. Chatting Window



Chatting window consists of ChatField, Textfield, Sned Message button, Exit button, and usersPanel. ChatField shows text messages received from the server. TextField is the area where user writes a text message to send. Send Message Button binds to sendMessageClick function. Exit button binds to exitClick function which clears all resources and terminates main thread. UsersPanel shows a list of users participating in this chatting.

1. Chatting init

When Chatting object is declared, Connect object is constructed. After connection, pyaudio module starts to receive and send voice.

1. Main Thread

In run function, there is implementation of Chatting Window GUI and starting new thread for communicating with the server.

1. sendMessageClick

Since send message button binds to sendMessageClick, this function is called only if send message button is clicked. When the button is clicked, it gets sendData form the textField and sends the data through the text\_socket.

1. sendingVoice

This function is about sending voice through the voice\_socket. The voice that we made becomes data by the send\_stream.read fuction. This function is always executed after connection.

1. receivingMsg

This function is about receiving text message through the text\_socket. It is always executed after connection. When new text message is received, the recvData is displayed to ChatField.

1. receivingVoice

This function is about receiving voice through the voic\_socket. Whenever voice is received, the voice is out through the receive\_stream which is pyaudio module.

1. receivingUsers

This function is about receiving users’ information participating in this chatting room. When user information is received, the userPanel is reset and print the usernames.

1. clearReasources

This function is called whenever disconnection occurs. When disconnection occurs, this function makes audio stream stop and sockets closed.

## Result

### Environment Settings for program

* Python3
* Window 10

### Run the program

* Run the server
  + python server.py
* Run the client
  + python client.py
  + Set host, port(8080), and username
  + Type what you want to send

## Discussion

* At assignment 2, we implemented our program with console but we felt that it is not user friendly. Therefore, we changed our program to use GUI, which is supported by python tkinter.
* We divided text and voice socket. At first try, we merged text and voice socket but we had hard time to distinguish which is text message and voice message. Also, since voice message was sent continuously, it looks that there is a little network delay. Therefore, we selected to open two sockets, one for text and one for socket. By dividing sockets, we don’t need to figure out what is text and voice message.
* We separated main thread and connecting client thread. At first try, we integrated two functions in one thread, which do “bind”, “listen” and “accept”. However, in Window OS, when main thread is blocking with the function “accept”, it doesn’t respond with KeyboardInterrupt Signal. Therefore, in order to kill server with keyboardInterrupt, we divided main thread, which do “bind” and “listen”, and connecting\_client\_thread, which do “accept”.

## Conclusion

We have completely implemented an Internet text and voice chatting program. Since the program should use audio transferring, it was a fine practice to learn how to deal with voice chatting, which is slightly different with text chatting. We also experienced that we had to use lock for synchronous programming.

## Reference

<https://github.com/laucer/Python-chat-with-GUI/blob/master/client.py>

* We refers this site for GUI components.