Computer Network Assn#2

20170302 CSED 김대희

20170354 CSED 백승훈

**1. 프로그램 구조**

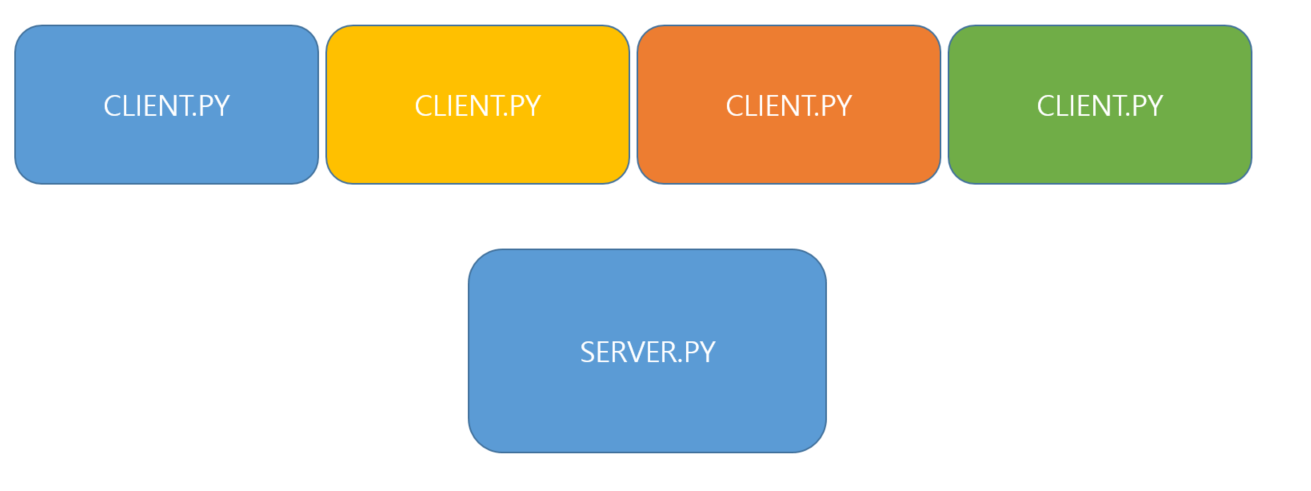


Figure 프로그램 구조

이번 과제에서 구현한 채팅 프로그램은 서버와 클라이언트를 각각 실행하여 서버의 IP주소로 접속하여 간단한 텍스트를 주고받을 수 있는 프로그램이다. 프로그램의 구조는 위와 같다. Server.py를 컴파일하여 해당 IP address만 알고있으면 client.py와 server.py에서 같은 값의 port를 가지고 있어서 server로 접속하여 텍스트를 주고 받을 수 있다. 간단한 GUI를 구현하여 채팅창에서 이를 확인할 수 있다.

**2.소스코드**

2.1server.py

|  |
| --- |
| //server.py  from socket import AF\_INET, socket, SOCK\_STREAM  from threading import Thread  def accept\_incoming\_connections():  """Sets up handling for incoming clients."""  while True:  client, client\_address = SERVER.accept()  print("%s:%s has connected." % client\_address)  client.send(bytes("Greetings from the cave! Now type your name and press enter!", "utf8"))  addresses[client] = client\_address  Thread(target=handle\_client, args=(client,)).start()   * 위 함수는 서버에서 클라이언트의 connection을 준비하는 부분이다. 실제로 client가 server의 IP address를 통해서 접속하게 되면은 chatting에 사용하게 될 username을 생성하여 client socket을 start시켜주게 하는 부분이다.   def handle\_client(client): # Takes client socket as argument.  """Handles a single client connection."""  name = client.recv(BUFSIZ).decode("utf8")  welcome = 'Welcome %s! If you ever want to quit, type {quit} to exit.' % name  client.send(bytes(welcome, "utf8"))  msg = "%s has joined the chat!" % name  broadcast(bytes(msg, "utf8"))  clients[client] = name   * Client socket을 인자로 받아들여서 해당 client를 server에서 관리하는 부분이다. 맨 처음 텍스트 채팅방에 들어왔을 경우에 환영의 메시지와 함께 채팅을 종료하는 방법을 알려주는 모습이다. Broadcast를 통해서 chattingroom에 text를 위치 시키도록 한다.   while True:  msg = client.recv(BUFSIZ)  if msg != bytes("{quit}", "utf8"):  broadcast(msg, name+": ")  else:  client.send(bytes("{quit}", "utf8"))  client.close()  del clients[client]  broadcast(bytes("%s has left the chat." % name, "utf8"))  break   * {quit}을 이용하여 client가 종료할 수 있도록 해주는 부분이다. Quit을 하게되면 client socket을 close하여 해당 chattingroom에서 나가게 한다.   def broadcast(msg, prefix=""): # prefix is for name identification.  """Broadcasts a message to all the clients."""  for sock in clients:  sock.send(bytes(prefix, "utf8")+msg)   * boadcast 함수를 정의하여 socket의 text를 표시할 수 있도록 해준다.   clients = {}  addresses = {}  HOST = ''  PORT = 33000  BUFSIZ = 1024  ADDR = (HOST, PORT)  SERVER = socket(AF\_INET, SOCK\_STREAM)  SERVER.bind(ADDR)   * client와 host의 port를 일치시켜주어 해당 port로 chatting room으로 입장시켜준다. 만약에 ADDR에서 PORT를 바꿔준다면 다른 chattingroom으로 들어가게 된다. (밑에 실행기 부분 생략) |

2.2client.py

|  |
| --- |
| #!/usr/bin/env python3  from socket import AF\_INET, socket, SOCK\_STREAM  from threading import Thread  import tkinter   * 원격 통신을 하기 위해서, socket관련 모듈들과 thread 관련 모듈들을 import하고 간단한 gui 환경을 구축하기 위하여 tkinter 모듈을 import했다.   def receive():  """Handles receiving of messages."""  while True:  try:  msg = client\_socket.recv(BUFSIZ).decode("utf8")  msg\_list.insert(tkinter.END, msg)  msg\_list.see(tkinter.END)  except OSError:  break   * receive는 상대방 측에서 incode된 메시지를 decode하고, gui 창에서 메시지를 볼 수 있도록 해준다.   def send(event=None): # event is passed by binders.  msg = my\_msg.get()  my\_msg.set("") # Clears input field.  client\_socket.send(bytes(msg, "utf8"))  if msg == "{quit}":  client\_socket.close()  top.quit()   * gui 창에 텍스트 박스 안에 있는 내용을 incode하여 상대방 측으로 보내는 역할을 한다. 만약 {quit}가 메시지 내용이면, 채팅 프로그램을 종료한다.   def on\_closing(event=None):  """This function is to be called when the window is closed."""  my\_msg.set("{quit}")  send()   * 창을 닫으면, {quit}이라는 내용을 send하는 것처럼 반영되어 send에서 프로그램을 종료하도록 한다.   top = tkinter.Tk()  top.title("Chat On!")   * tkinter 모듈을 실행하고, 실행창의 이름을 “chat on”으로 설정해준다.   messages\_frame = tkinter.Frame(top)  my\_msg = tkinter.StringVar() # For the messages to be sent.  my\_msg.set("")  scrollbar = tkinter.Scrollbar(messages\_frame) # To see through previous messages.  # this will contain the messages.  msg\_list = tkinter.Listbox(messages\_frame, height=30, width=100, yscrollcommand=scrollbar.set)  scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)  msg\_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)  msg\_list.pack()  messages\_frame.pack()   * 메시지 입력을 받기 위한 창과 이전의 기록들을 확인할 수 있도록 gui를 설정해준다.   entry\_field = tkinter.Entry(top, textvariable=my\_msg)  entry\_field.bind("<Return>", send)  entry\_field.pack()  send\_button = tkinter.Button(top, text="Send", command=send)  send\_button.pack()   * send 버튼을 눌러서 메시지 전송을 하면 send를 호출하여 메시지를 전송한다.   top.protocol("WM\_DELETE\_WINDOW", on\_closing)   * gui 창이 닫히면, “WM\_DELETE\_WINDOW” 시그널을 감지하여 on\_closing을 호출한다.   #Socket part  HOST = input('Enter host: ') # Enter host of the server without inverted commas  PORT = 33000  BUFSIZ = 1024  ADDR = (HOST, PORT)   * 호스트의 IP 주소를 시작하면 적게 되어있다. Port는 33000으로 고정해두었다.   client\_socket = socket(AF\_INET, SOCK\_STREAM)  client\_socket.connect(ADDR)   * 호스트의 IP:33000으로 connect 시도한다.   receive\_thread = Thread(target=receive)  receive\_thread.start()   * thread를 생성한다.   tkinter.mainloop() # for start of GUI Interface |