İÇİMİ DEMİRAĞ
191180033
BM402
ÖDEV RAPORU

TCP PROTOKLÜYLE ANLIK MESAJLAŞMA UYGULAMASI

İÇİNDEKİLER

Sayfa

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

1. GİRİŞ

TCP; paketlerin gönderilmesi, teslim edilmesi ve hataların düzeltilmesi gibi işlemleri gerçekleştiren internet üzerindeki veri iletişiminin güvenilirliğini artırmak için kullanılan bir iletişim protokolüdür. TCP ile kullanıcıların mesaj kayıtlarını tutacak, geçmiş mesajlara erişim sağlayacak, kullanıcıların bağlantı listesi oluşturmasına izin verecek, geçmiş mesajlar üzerinde anahtar kelime araması yapılabilmesini sağlayacak ve kullanıcılar bu bağlantı listesi üzerinde gruplandırmalar yapabilecek bir uygulama yapılabilir.

Bu kapsamda ödevin devamında TCP ile yapılan anlık mesajlaşma uygulamasının server ve client kodlarından ve çıktılarından bahsedilmektedir.

2. ÖDEV RAPORU

2.1 TCP Server

Ek-1'de TCP server için yazılan kodlara yer verilmiştir. Script içerisindeki komutları genel olarak açıklamak gerekirse [1];

- socket(): İstemci veya sunucu tarafında bir soket nesnesi oluşturmak için kullanılan bir fonksiyondur.
- .bind(): Bir soketi belirli bir IP adresi ve port numarasıyla bağlamak için kullanılan bir fonksiyondur.
- .listen(): Soketi bağlı olduğu adrese dinlemek için kullanılan bir fonksiyondur. Bağlantı istekleri dinlenir ve accept() fonksiyonuyla kabul edilir.
- .accept(): Bir istemci bağlantısı kabul edilir ve yeni bir soket nesnesi oluşturulur. Bu nesne, istemci ile iletişim kurmak için kullanılır.
- .send(): Bir soket nesnesinden veri göndermek için kullanılır.
- .recv(): Bir soket nesnesinden veri almak için kullanılır.
- .close(): Bir soket nesnesinin bağlantısını sonlandırmak için kullanılır.

Bu scripti çalıştırdığımızda ve birkaç client bağlandığında ise bizi şekil 2.1'deki gibi bir çıktı karşılayacaktır.

```
icimidemirag@Icimi-MacBook-Air tcp_python % python3 server.py
Running the server on 127.0.0.1 12345
Successfully connected to client 127.0.0.1 50243
Successfully connected to client 127.0.0.1 50251
Successfully connected to client 127.0.0.1 50252
```

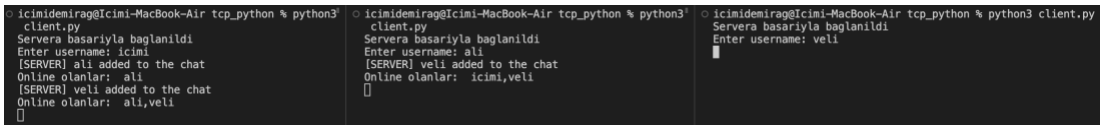
Şekil 2.1 TCP server

2.2 TCP Client

Ek-2'de TCP client için yazılan kodlara yer verilmiştir. Script içerisindeki komutları genel olarak server tarafıyla aynıdır sadece .bind() yerine kullanılan .connect()'i açıklamak gerekirse [1];

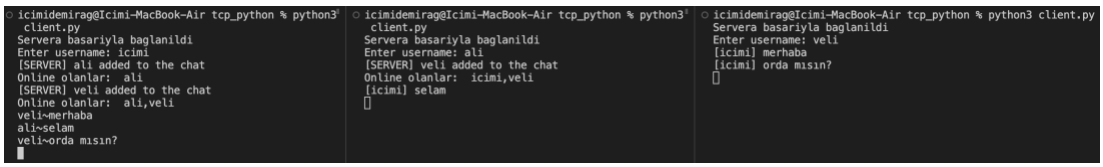
- .connect(): Bir soket nesnesi belirli bir IP adresi ve port numarasına bağlanmak için kullanılır.

Yazdığım scripti çalıştırınca öncelikle şekil 2.2'de de görülen sunucuya başarıyla bağlanıldı yazısı gelecektir ve hemen ardından kullanıcı adı istenecektir. Ayrıca diğer client'lar da katıldıkça online olanların bilgisi güncellenecek ve server diğer kullanıcılara birinin chat'e katıldığına dair bilgilendirme mesajı yollayacaktır.



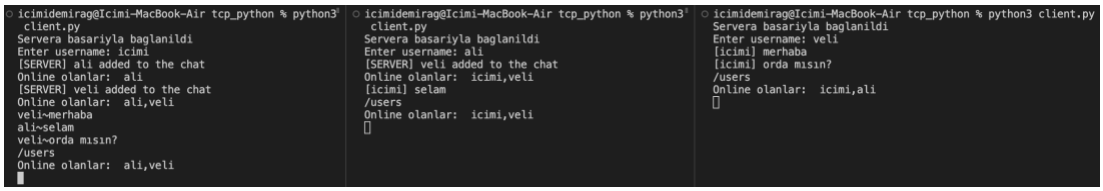
Şekil 2.2 TCP client sunucu bağlantısı

Benim yaptığım uygulamada direkt chat ekranına girildiğinden birbiriyle konuşmak isteyen kullanıcılar, şekil 2.3'te de görüldüğü gibi konuşmak istediği kullanıcının adını yazıp hemen ardından "~" işaretini koyup mesajlarını iletebilmektedirler.



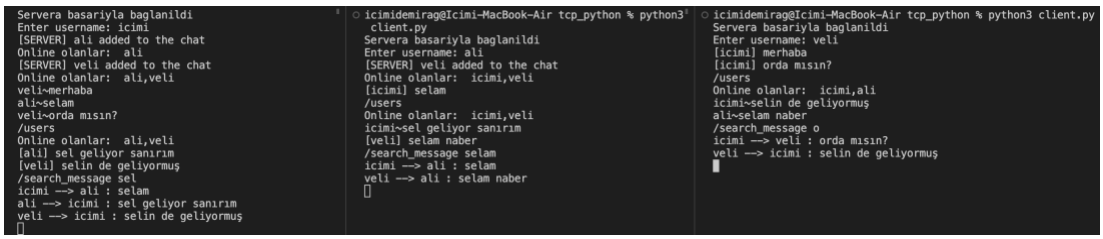
Şekil 2.3 Birebir chat

Ayrıca kullanıcıların bağlantı listesine "/users" komutu ile ulaşılabilir, şekil 2.4'te tüm clientlarda bu komut yazıldığındaki görüntü görünmektedir. Bu sayede sohbetin ilerleyen zamanlarında online olanların kim olduğu unutulursa bu komutla o an kimlere mesaj atılabileceğini görebilmektedir.



Şekil 2.4 Kullanıcıların bağlantı listesi

Kullanıcıların direkt sohbet odasına girmesi, gelen mesajların kullanıcı bazında kaydının tutulmasına ve geçmiş mesaj bilgilerinin görüntülenmesine kolaylık sağlamaktadır. Bu sayede tüm mesajlar tek bir ortam üzerinde görünür olacaktır fakat çok fazla mesaj olması durumunda eski mesajlara erişmek için "/search_message" komutu eklenmiştir. Şekil 2.5'te görüldüğü gibi herhangi bir cümle, kelime ya da harf aranılabilmektedir.



Şekil 2.5 search_message komutu

Bunun yanı sıra sadece bir kişi ile aralarında geçen konuşmaları görmek için ise şekil 2.6'da da görüldüğü üzere "/search_person" komutu da bulunmaktadır.

```
Servera basariyla baglanildi
Enter username: veli
[icimi] merhaba
[icimi] orda mısın?
/users
Online olanlar:  icimi,ali
icimi~selin de geliyormuş
ali~selam naber
/search_message o
icimi --> veli : orda mısın?
veli --> icimi : selin de geliyormuş
[ali] uzun zamandır görüşemiyoruz
ali~evet bence de
/search_person ali
veli --> ali : selam naber
ali --> veli : uzun zamandır görüşemiyoruz
veli --> ali : evet bence de
```

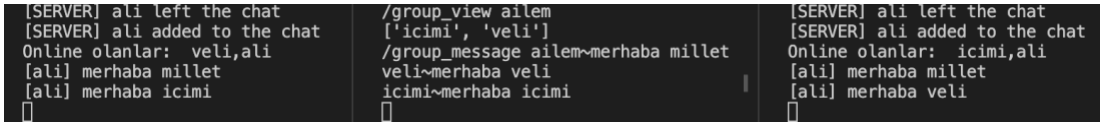
Şekil 2.6 search_person komutu

Liste üzerinde gruplandırma yapmak için "/groups_view", "/group_view", "/group_name_add", "/group_name_remove", "group_person_add" ve "/group_person_remove" komutları bulunmaktadır. Şekil 2.7'de bu komutların her birinin örnekleri gösterilmektedir.

```
/groups_view
[]
/group_name_add ailem
/groups_view
['ailem']
/group_person_add ailem ali
/group_person_add ailem veli
/group_view ailem
['ali', 'veli']
/group_name_add diğer
/groups_view
['ailem', 'diğer']
/group_name_remove ailem
/groups_view
['diğer']
/group_person_add diğer ali
/group_view diğer
['ali']
/group_person_remove diğer ali
/group_view diğer
[]
```

Şekil 2.7 Gruplandırma işlemleri

Bu yapılan gruplandırmalar sayesinde grup mesajı da göndermek mümkündür. Şekil 2.8’de “/group_message” komutuyla yapılan grup mesajı örneği görülmektedir.



Şekil 2.8 Grup chat

Ayrıca şekil 2.8’de birisinin uygulamadan çıktığında server’ın diğerlerine bildirdiği de görülmektedir.

3. SONUÇ

Anlık mesajlaşma uygulaması geliştirdiğim bu ödevde, TCP protokolüyle bireysel chat, group chat, eski mesajlarda arama, kişiyle mesaj geçmişi, kişileri gruplandırma gibi bir dizi işlevi kodladım. Kodladığım bu program sayesinde bir anlık bir mesajlaşma uygulamasında ihtiyaç duyulan bazı gereksinimleri TCP protokolüyle daha güvenilir bir şekilde yapılması sağlanmıştır. Sonuç olarak, TCP protokolü ve bu protokol ile kullanıcıların anlık birbirleriyle konuşabildiği bir chat uygulamasının işlevleri hakkında pratikte bilgi edinmiş ve daha iyi kavramış oldum.

KAYNAKLAR

1. Nathan Jennings. “Socket Programming in Python (Guide)”. (2023).

EKLER

EK-1. TCP server kodu

```
import socket
import threading
import json

HOST = '127.0.0.1'
PORT = 12345
active_clients = []

def listen_for_messages(client, from_username):
    while True:
        try:
            message = client.recv(2048).decode('utf-8')
            temp = json.loads(message)
            if temp['command'] == "users":
                message_dict = {"command":"users", "users":[x[0] for x in active_clients]}
                client.send(json.dumps(message_dict).encode('utf-8'))
                continue
            if message != '':
                to_username = temp['message'].split("~")[0]
                content = temp['message'].split("~")[1]

                final_msg = {"command":"message", "message":from_username + '~' + content}
                for user in active_clients:
                    if user[0]==to_username:
                        send_message_to_client(user[1], json.dumps(final_msg))
                        break
            else:
                print(f"The message send from client {from_username} is empty")
        except Exception as e:
                print(f"Error: {e}")
                client.close()
                active_clients.remove((from_username, client))
                message_dict = {"command":"left", "message":"[SERVER] " + f"{from_username} left the
chat"}
                send_messages_to_all(json.dumps(message_dict))
                break
```

```
def send_message_to_client(client, message):
    client.send(message.encode('utf-8'))

def send_messages_to_all(message):
    for user in active_clients:
        send_message_to_client(user[1], message)

def client_handler(client):
    while True:
        message = client.recv(2048).decode('utf-8')
        temp = json.loads(message)

        if temp['command'] == "add_client":
            prompt_message = {"command":"message", "message":"SERVER~" + f"{temp['message']}
added to the chat"}
            send_messages_to_all(json.dumps(prompt_message))
            active_clients.append((temp['message'], client)) #username, client
            break
        else:
            print("Username is empty")

    threading.Thread(target=listen_for_messages, args=(client, temp['message'], )).start()

def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        server.bind((HOST, PORT))
        print(f"Running the server on {HOST} {PORT}")
    except:
        print(f"Unable to bind to host {HOST} and port {PORT}")

    server.listen()

    while True:
        client, address = server.accept()
        print(f"Successfully connected to client {address[0]} {address[1]}")
```

```
        threading.Thread(target=client_handler, args=(client,)).start()

if __name__ == '__main__':
    main()
```

EK-2. TCP client kodu

```
import socket
import threading
import json
import time

HOST = '127.0.0.1'
PORT = 12345
username = ""
users= []
messages = []
groups = {}

def listen_for_messages_from_server(client):
    global users
    global username

    while True:
        message = client.recv(2048).decode('utf-8')
        temp = json.loads(message)

        if temp['command'] == "users":
            users = filter(lambda c: c != username, temp["users"])
            print("Online olanlar: ",','.join(map(str, users)))
        elif temp['command'] == "left":
            print(temp['message'])
            user_temp = temp['message'].split(" ")[1]
            for group_name in groups:
                if(user_temp in groups[group_name]):
                    groups[group_name].remove(user_temp)

        elif temp['command'] == "message":
            name = temp['message'].split("~")[0]
            content = temp['message'].split("~")[1]

            if(name == "SERVER"):
                temp = {"command":"users"}
```

```
            client.send(json.dumps(temp).encode('utf-8'))
        else:
            messages.append(((name,username),content))

        print(f"[{name}] {content}")
    else:
        print("Message reveived from client is empty")

def send_message_to_server(client):
  while True:
    message = input("")

    if (message.startswith("/search_message")):
        content = message.split("/search_message ")[1]
        for ((fr,to),cnt) in messages:
            if(content in cnt):
                print(f"{fr} --> {to} : {cnt}")
        continue
    elif (message.startswith("/search_person")):
        person = message.split("/search_person ")[1]
        for ((fr,to),cnt) in messages:
            if((person == fr) or (person == to)):
                print(f"{fr} --> {to} : {cnt}")
        continue
    elif (message.startswith("/groups_view")):
        print(list(groups.keys()))
        continue
    elif (message.startswith("/group_view")):
        content = message.split("/group_view ")[1]
        print(groups[content])
        continue
    elif (message.startswith("/group_name_add")):
        group_name = message.split("/group_name_add ")[1]
        groups[group_name]=[]
        continue
    elif (message.startswith("/group_name_remove")):
        group_name = message.split("/group_name_remove ")[1]
```

```
            groups.pop(group_name)
            continue
        elif (message.startswith("/group_person_add")):
            group_name = message.split(" ")[1]
            person = message.split(" ")[2]
            groups[group_name].append(person)
            continue
        elif (message.startswith("/group_person_remove")):
            group_name = message.split(" ")[1]
            person = message.split(" ")[2]
            groups[group_name].remove(person)
            continue
        elif (message.startswith("/group_message")):
            group_name = message.split(" ")[1].split("~")[0]
            content = message.split("~")[1]
            group_temp = groups[group_name]
            for person in group_temp:
                temp = {"command":"message", "message":person+"~"+content}
                client.send(json.dumps(temp).encode('utf-8'))
                time.sleep(0.1)
                messages.append(((username, person),content))
            continue
        elif(message.startswith("/")):
            temp = {"command":message.split("/")[1]}
        else:
            temp = {"command":"message", "message":message}
            name = temp['message'].split("~")[0]
            content = temp['message'].split("~")[1]
            messages.append(((username, name),content))
        if message != '':
            client.send(json.dumps(temp).encode('utf-8'))
        else:
            print("Empty message")

def communicate_to_server(client):
    global username
    username = input("Enter username: ")
```

```
    if username != '':
        temp = {"command":"add_client", "message":username}
        client.send(json.dumps(temp).encode('utf-8'))
    else:
        print("Username cannot be empty")
        exit(0)

    threading.Thread(target=listen_for_messages_from_server, args=(client,)).start()

    send_message_to_server(client)

def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        client.connect((HOST, PORT))
        print("Servera basariyla baglanildi")
    except:
        print("Servera baglanilamiyor")

    communicate_to_server(client)

if __name__ == '__main__':
    main()
```