دانشکده برق و کامپیوتر

پردیس فنی

دانشگاه تهران

پروژه نهایی شبکه های کامپیوتری

دکتر شاه منصوری

فاطمه نائینیان

شماره دانشجویی: 810198479

1402

میخواهیم یک chatroom با کمک Socket Programming پیاده سازی کنیم. برای اینکار لازم است دو مجموعه کد client و server را پیاده سازی کنیم.

ابتدا client را توضیح میدهم. یک client لازم است تا از کاربر اطلاعات دریافت کند و به کاربر اطلاعات بدهد پس دو تابع برای دریافت و ارسال اطلاعات از کاربر به سرور نیاز است. این دو تابع به شکل زیر پیاده سازی می شود.

```
import threading
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 50000))

MAX_CLIENTS = 99
BUFFER_SIZE = 4096

def Receive():
    while True:
        try:
            message = client.recv(BUFFER_SIZE).decode('utf-8')
            print(message)
        except Exception as e:
            print('Error!')
            print(e)
            client.close()
            break

def Send():
    while True:
        try:
            message = f'{input("")}'
            client.send(message.encode('utf-8'))
        except:
            print('Error!')
            break

receive_thread = threading.Thread(target=Receive)
receive_thread.start()
send_thread = threading.Thread(target=Send)
send_thread.start()
receive_thread.join()
send_thread.join()
```

اما سرور وظیفه بیشتری دارد. سرور علاوه بر دریافت و ارسال اطلاعات به client باید اطلاعات را پردازش و هندل کند. 8 دستور گفته شده در متن پروژه در سرور پردازش میشود و پایگاه داده ها نیز در سرور است.

$$
\begin{aligned}
&1.\ create\ [group-name]\\
&2.\ join\ [group-name]\\
&3.\ leave\ [group-name]\\
&4.\ groups\\
&5.\ users\\
&6.\ public\ [group-name]\ [message]\\
&7.\ private\ [username]\ [message]\\
&8.\ exit
\end{aligned}
$$

ابتدا یک تابع مینویسیم تا به ازای هر client جدید یک ارتباط TCP برقرار کند. برای این کار تابع زیر را مینویسیم.

```
def receive():
    while True:
        client_socket, address = server_socket.accept()
        print(f'connected with {str(address)}')
        client_socket.send('enter a username: '.encode('utf-8'))
        username = client_socket.recv(1024).decode()
        client_socket.send('connected! \n'.encode('utf-8'))
        user = User(client_socket, username)
        USERS.append(user)
        group = GROUPS[0]
        group.users.append(user)
        client_socket.send(f'you joined the group General'.encode('utf-8'))
        publicMessage(user,group.name, f'{username} joined the group')
        thread = threading.Thread(target=handle, args=(user,))
        thread.start()
```

با کمک این تابع کاربر را به لیست کاربران اضافه میکنیم و در گروه general عضو کرده و تابع handle را برای ان فعال میکنیم تا دستورات را پردازش کند.

این تابع به صورت زیر است.

```
def handle(user):
    while True:
        try:
            message = user.client_socket.recv(1024)
            words = message.decode().split(" ")

            if words[0] == 'create':
                create_group(words[1], user)

            elif words[0] == 'join':
                join_group(words[1], user)

            elif words[0] == 'leave':
                leave_group(words[1], user)

            elif words[0] == 'groups':
                group_names = '\n'.join([group.name for group in GROUPS])
                user.client_socket.sendall(f'groups list:
\n{group_names}'.encode())

            elif words[0] == 'users':
                user_names = '\n'.join([user.username for user in USERS])
                user.client_socket.sendall(f'users list:
\n{user_names}'.encode())

            elif words[0] == 'public':
                publicMessage(user, words[1], ' '.join(words[2:]))

            elif words[0] == 'private':
                privateMessage(user, words[1], ' '.join(words[2:]))

            elif words[0] == 'exit':
                exit_server(user)

            else:
                user.client_socket.sendall('incorrect message'.encode())

        except Exception as e:
            print(e)
            break
```

تابع create_group برای ساخت یک گروه جدید استفاده می شود که به صورت زیر است:

```
def create_group(group_name, user):
    not_found = True
    for group in GROUPS:
        if group.name == group_name:
            user.client_socket.sendall(f'Group already exists'.encode())
            not_found = False
    if not_found:
        GROUPS.append(Group(group_name))
        user.client_socket.sendall(f'New group chat with the name {group_name}
has created successfully'.encode())
```

با کمک تابع leave_group از گروه خارج شده و به کاربران گروه نیز اطلاع می دهد.

```
def leave_group(group_name, user, exit_mode):
    not_found = True
    for group in GROUPS:
        if group.name == group_name:
            not_found = False
            if user in group.users:
                publicMessage(user, group.name , f'{user.username} left the group
{group.name}!')
                group.users.remove(user)
                if not exit_mode:
                    user.client_socket.sendall(f'You left the group
{group.name}'.encode())
            else:
                user.client_socket.sendall(f'You are not a member of the group
{group.name}'.encode())
    if not_found:
        user.client_socket.sendall(f'Group Not not_found!'.encode())
```

تابع join_group برای پیوستن به یک گروه است که کاربران نیز اطلاع داده می شود.

```
def join_group(group_name, user):
    not_found = True
    for group in GROUPS:
        if group.name == group_name:
            not_found = False
            if user not in group.users:
                group.users.append(user)
                user.client_socket.sendall(f'You joined group {group_name}
successfully'.encode())
                publicMessage(user, group_name , f'{user.username} joined the
group {group_name}')
            else:
                user.client_socket.sendall(f'You already joined the group
{group_name}'.encode())
    if not_found:
        user.client_socket.sendall(f'Group Not not_found!'.encode())
```

هنگام exit ، client ارتباط TCP خود را میبندد که به صورت زیر است و از همه گروه ها خارج می شود.

```
def exit_server(user):
    for group in GROUPS:
        if user in group.users:
            leave_group(group.name, user, True)
    username = user.username
    USERS.remove(user)
    print(f'user {username} disconnected...')
    user.client_socket.close()
```

برای پیام public به همه اعضای گروه پیام ارسال می شود.

```
def public_message(user, group_name, message):
    not_found = True
    for group in GROUPS:
        if group.name == group_name:
            not_found = False
            if user in group.users:
                for client in group.users:
                    if client.client_socket is not user.client_socket:
```

```
                        client.client_socket.sendall(f'[public group {group_name}
from {user.username}] {message}'.encode())
            else:
                user.client_socket.sendall(f'You are not a member of the group
{group_name}'.encode())
    if not_found:
        user.client_socket.sendall(f'Group Not not_found'.encode())
```

در پیام private فقط به یک client دیگر پیام ارسال می شود:

```
def private_message(user, client_name, message):
    not_found = True
    for client in USERS:
        if client.username == client_name:
            not_found = False
            client.client_socket.sendall(f'[private from {user.username}]
{message}'.encode())
    if not_found:
        user.client_socket.sendall(f'Client Not not_found'.encode())
```