

Laporan Tugas Besar II 2120 Jaringan Komputer Tahun Akademik 2024/2025

Tugas UDP Socket Programming

Kelompok -_-



Disusun Oleh:

Stevan Einer Bonagabe-18223028

Jacob Reinhard Marudut Siagian-18223026

I. Daftar Isi

Cover	0
I. Daftar Isi	1
II. Daftar Tabel	2
III. Daftar Gambar	3
IV. Tabel Penyelesaian Spesifikasi	5
Tabel 2.1 Keterangan Penyelesaian Spesifikasi Utama	5
Tabel 2.2 Keterangan Penyelesaian Spesifikasi Tambahan (opsional)	5
V. Penjelasan Terkait Cara Kerja Program	7
4.1. Cara kerja program secara umum(spesifikasi program utama)	7
VI. Penjelasan Terkait Tata Cara dan Lingkungan Pengujian Program	15
VII. Foto atau Tangkapan Layar	17
VIII. Source Code Program Client dan Server	22
IX. Tautan ke GitHub atau Google Drive	25
X. Tabel Pembagian Tugas atau Kontribusi Anggota	26
Tabel 9.1 Pembagian Tugas setiap anggota	20
XI. Daftar Pustaka	27

II. Daftar Tabel

<u>Tabel 2.1.....</u>	<u>4</u>
<u>Tabel 2.2.....</u>	<u>4</u>
<u>Tabel 9.1.....</u>	<u>19</u>

III. Daftar Gambar

<u>Gambar7.1.....</u>	<u>11.</u>
<u>Gambar7.2.....</u>	<u>11.</u>
<u>Gambar7.3.....</u>	<u>12</u>
<u>Gambar7.4.....</u>	<u>12</u>
<u>Gambar7.5.....</u>	<u>13.</u>
<u>Gambar 7.6.....</u>	<u>13</u>
<u>Gambar7.7.....</u>	<u>14</u>
<u>Gambar7.8.....</u>	<u>14</u>
<u>Gambar7.9.....</u>	<u>14</u>
<u>Gambar7.10.....</u>	<u>14</u>
<u>Gambar7.11.....</u>	<u>15.</u>
<u>Gambar7.12.....</u>	<u>15</u>
<u>Gambar7.13.....</u>	<u>15</u>
<u>Gambar8.1.....</u>	<u>17</u>
<u>Gambar8.2.....</u>	<u>18</u>

IV. Tabel Penyelesaian Spesifikasi

Tabel 2.1 Keterangan Penyelesaian Spesifikasi Utama

No	Spesifikasi	Keterangan
1	Server mampu menerima pesan yang dikirim client dan mencetaknya ke layar.	✓
2	Server mampu meneruskan pesan satu client ke client lain.	✓
3	Client mampu mengirimkan pesan ke server dengan IP dan port yang ditentukan pengguna.	✓
4	Client mampu menerima pesan dari client lain (yang diteruskan oleh server), dan mencetaknya ke layar.	✓
5	Client harus memasukkan <i>password</i> untuk dapat bergabung ke chatroom.	✓
6	Client memiliki username yang unik.	✓

Tabel 2.2 Keterangan Penyelesaian Spesifikasi Tambahan (opsional)

No	Spesifikasi	Keterangan
1	Aplikasi mengimplementasikan TCP over UDP. Note: asisten sangat menyarankan untuk mengerjakan spesifikasi ini, karena akan memberikan pemahaman kuat terkait TCP.	✓
2	Seluruh pesan dienkripsi menggunakan algoritma kriptografi klasik simetris, misal cipher Vigenere atau Caesar.	✗
3	Seluruh pesan dienkripsi menggunakan algoritma kriptografi modern simetris, misal cipher RC4.	✗
4	Seluruh pesan dienkripsi menggunakan algoritma kriptografi modern asimetris, misal cipher RSA, atau kombinasi algoritma kriptografi modern asimetris dan modern simetris.	✗
5	Seluruh pesan dienkripsi menggunakan algoritma Double-Ratchet atau MLS.	✗
6	Aplikasi memiliki GUI.	✗
7	Aplikasi mampu digunakan untuk mengirimkan dan menerima pesan bertipe file biner.	✗

8	Aplikasi mampu menunjukkan apabila integritas pesan telah rusak, baik dengan memanfaatkan <i>checksum</i> ataupun <i>digital signature</i> .	✗
9	Aplikasi mampu menyimpan pesan-pesan lampau meskipun telah ditutup; mekanisme dan tempat penyimpanan bebas, baik di <i>client</i> maupun di <i>server</i> .	✗
10	Aplikasi mampu meng otentikasi pengguna.	✗

V. Penjelasan Terkait Cara Kerja Program

4.1. Cara kerja program secara umum(spesifikasi program utama)

- **Deskripsi Umum:**

- Server

```
def handle_client(message, client_address):  
    """Process client messages and handle ACKs, broadcasting  
    to other clients."""  
    global clients  
  
    decoded_message = message.decode()  
  
    if decoded_message.startswith("SIGNUP_TAG:"):   
        name = decoded_message.split(":")[1]  
        if name in clients.values():  
            server_socket.sendto("Username already taken,  
please use another.".encode(), client_address)  
        else:  
            clients[client_address] = name  
            print(f"{name} joined from {client_address}")  
            for client in clients:  
                server_socket.sendto(f"{name} joined the  
chat.".encode(), client)  
            return  
  
    elif decoded_message.startswith("LOGOUT_TAG:"):   
        name = decoded_message.split(":")[1]  
        if client_address in clients:  
            del clients[client_address]  
            print(f"{name} left from {client_address}")  
            for client in clients:  
                server_socket.sendto(f"{name} has  
left.".encode(), client)
```

```

    return

    # Handle regular message with sequence number
    if "|" in decoded_message:
        seq_number, client_msg = decoded_message.split("|", 1)
        seq_number = int(seq_number)

        # Send ACK for the received message to the sender
        ack_message = f"ACK|{seq_number}"
        server_socket.sendto(ack_message.encode(),
                             client_address)

        # Broadcast the message to all other clients
        for client in clients:
            if client != client_address:
                server_socket.sendto(client_msg.encode(),
                                     client)

```

Fungsi ini menangani interaksi antara server dan setiap klien. Saat klien terhubung, fungsi ini menerima data dari klien, memeriksa apakah itu pesan bergabung, pesan logout, atau pesan biasa dengan urutan. Jika klien mengirimkan pesan bergabung, server akan memeriksa apakah nama pengguna sudah ada di dalam daftar klien. Jika nama tersedia, klien ditambahkan ke dalam daftar, dan pesan bergabung akan dikirim ke semua klien. Jika klien mengirimkan pesan biasa, server akan mengirimkan ACK (tanda pengakuan) untuk memastikan klien mengetahui bahwa pesan telah diterima. Pesan kemudian ditambahkan ke antrian untuk disiarkan ke klien lain. Jika klien mengirimkan pesan logout, server akan menghapus klien dari daftar dan mengirimkan pesan keluar ke semua klien yang tersisa.

```

def receive():

    """Receive messages from clients and add them to the
    queue."""

    print("Server is running...")

    while True:

```



```

        message, client_address = server_socket.recvfrom(1024)

        messages.put((message, client_address))

        handle_client(message, client_address)

# Start the server thread

threading.Thread(target=receive, daemon=True).start()

# Keep the server running

try:

    while True:

        pass

except KeyboardInterrupt:

    print("\nServer shutting down.")

```

Fungsi receive adalah komponen inti dari server dalam mengelola komunikasi. Fungsi ini bertanggung jawab untuk:

- Menerima dan mengidentifikasi pesan dari klien.
- Mengelola pendaftaran dan penghapusan klien.
- Mengirimkan ACK untuk menjamin pengiriman pesan (dalam sistem berbasis UDP).
- Mengatur aliran pesan ke antrian untuk disiarkan ke klien lain.

Dengan adanya fungsi ini, server dapat mengelola banyak klien sekaligus, menangani pesan secara efisien, dan memastikan bahwa setiap pesan klien diakui dan diteruskan ke klien lain sesuai kebutuhan.

- Client

```
while attempts < max_attempts:

    user_password = input("Enter password: ")

    if user_password == password:

        username = input("Enter your username: ")

client_socket.sendto(f"SIGNUP_TAG:{username}".encode(),
(SERVER_IP, SERVER_PORT))

    # Wait for server's response on username availability

    try:

        response, _ = client_socket.recvfrom(1024)

        if response.decode() == "Username already taken,
please use another.":

            print("Username already taken, please choose
another.")

        else:

            print("Connected to chat server!")

            break

    except socket.timeout:

        print("Could not connect to server: timeout")

        exit()

    else:

        attempts += 1
```

```

        print(f"Incorrect password. Attempts remaining:
{max_attempts - attempts}")

if attempts == max_attempts:

    print("Maximum attempts reached. Exiting...")

    exit()

```

Kode ini mengatur autentikasi pengguna dengan memeriksa kata sandi dan memastikan bahwa nama pengguna belum digunakan di server. Jika semua langkah berhasil, klien akan terhubung ke server dan siap bergabung dalam obrolan. Namun, jika pengguna salah memasukkan kata sandi hingga batas maksimum atau server tidak merespons dalam waktu tertentu, klien akan menutup program. Fungsi utama dari kode ini adalah untuk memastikan hanya pengguna yang diizinkan dan nama pengguna yang unik yang bisa bergabung ke server.

```

def receive_messages():

    """Receive messages from the server."""

    while True:

        try:

            data, _ = client_socket.recvfrom(1024)

            message = data.decode()

            # Check if the message is an ACK

            if message.startswith("ACK|"):

                ack_sequence = int(message.split("|")[1])

                if ack_sequence == sequence_number:

                    print("[Server] Message acknowledged.")

                    ack_received.set()

```

```

        else:

            print(message)    # Display broadcasted messages
                                from other clients

        except socket.timeout:

            Continue

```

Fungsi `receive_message()` memiliki dua tugas utama:

1. Menangani pesan broadcast dari klien lain yang diteruskan oleh server, sehingga klien dapat melihat pesan tersebut.
2. Memproses pesan ACK dari server untuk memastikan bahwa pesan yang dikirim telah diterima oleh server dan menghindari pengiriman ulang pesan.

Dengan demikian, fungsi ini membantu klien menerima pesan secara real-time dan mengonfirmasi bahwa pesan yang dikirim telah diterima oleh server.

```

def send_messages():

    """Send messages to the server, with retransmission on
    timeout."""

    global sequence_number

    sequence_number = 0

    while True:

        message_content = input("Your message: ")

        full_message = f"{sequence_number}|{username}:"
        {message_content}"

        # Use an event to wait for ACK

    global ack_received

```

```

ack_received = threading.Event()

# Attempt retransmission until acknowledged
while not ack_received.is_set():
    client_socket.sendto(full_message.encode(),
(SERVER_IP, SERVER_PORT))

    if not ack_received.wait(RETRANSMISSION_TIMEOUT):
        print(f"[TIMEOUT] Resending message
'{message_content}'...")

    sequence_number += 1

```

Fungsi `send_messages` memungkinkan klien untuk:

1. Mengirim pesan dengan nomor urutan unik ke server.
2. Menunggu ACK dari server untuk memastikan pesan diterima.
3. Mengirim ulang pesan jika server tidak memberikan ACK dalam waktu yang ditentukan (retransmission).

Fungsi ini membantu memastikan keandalan komunikasi antara klien dan server, bahkan jika ada potensi kehilangan paket atau masalah jaringan.

- **Referensi:**

- Penjelasan mengenai socket programming:
<https://youtu.be/3QiPPX-KeSc?si=JD5pyp5yLfecQQ2j>
- Cara memeriksa port yang terbuka/tertutup melalui program:
<https://youtu.be/icE6PR19C0Y?si=3wW11NlvVrs6MqPs>

VI. Penjelasan Terkait Tata Cara dan Lingkungan Pengujian Program

1. Tata Cara Pengujian:

1. Pertama-tama, download program untuk client dan server (disarankan jangan mengubah nama file setelah di download)
2. Lalu buka “settings” di laptop anda dan cari informasi IPv4 address mengenai wi-fi/ethernet yang sedang anda gunakan
3. Lalu, jalankan program server pada VS-Code atau command prompt
Jika menggunakan VS-Code, tekan tombol berikut ini disebelah kanan atas.



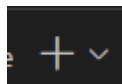
Jika menggunakan command prompt:

1. Bukalah command prompt pada alamat anda menyimpan file server dan client
2. Ketiklah command “python *namafileservr.py*”
4. Ketika anda menjalankan program server untuk pertama kali, akan diminta permission dan anda harus menekan “yes” untuk dapat menjalankan program
5. Masukkan informasi IPv4 address wi-fi/ethernet anda saat diminta oleh program dan pastikan bahwa format IP address valid seperti yang ditampilkan pertama kali ketika program berjalan.

*DISARANKAN menggunakan port 135 karena Port ini umumnya terbuka untuk mendukung komunikasi di dalam jaringan lokal, terutama di lingkungan yang menggunakan layanan berbasis Microsoft

6. Jika sudah berhasil memasukkan IP address dan port, server akan berjalan.
7. Setelah server berjalan, jalankan program client pada VS-Code atau command prompt.

Jika menggunakan VS-Code:



1. Tekanlah tombol berikut ini di bagian atas kanan terminal VS-Code
2. Lalu ketiklah “python *namafileClient.py*”

Jika menggunakan command prompt:

1. Bukalah command prompt pada alamat anda menyimpan file server dan client
2. Ketiklah command “python *namafileClient.py*”
8. Setelah itu, akan muncul UI kami dan anda diminta memasukkan password dan username anda. (password= JARKOM YES). Jika password salah, anda hanya memiliki sisa kesempatan 2 kali. Selain itu, anda tidak dapat menggunakan username yang sudah dipakai orang lain.
9. Jika anda berhasil login, anda akan bergabung dengan group chat.
10. Untuk menuliskan pesan, anda dapat menuliskannya pada kolom ini (gambar) dan klik tombol “send” untuk mengirim pesan. Pesan yang sudah terkirim, akan muncul pada jendela diatas (gambar)
11. Jika anda ingin logout, anda dapat menekan tombol “logout” dan program akan terhenti.

● **Lingkungan Pengujian:**

1. Laptop/Komputer harus memiliki minimal 8GB RAM, tetapi diatas 8GB akan lebih ideal, terutama jika server dan beberapa klien dijalankan di mesin yang sama karena ini dapat menggunakan memori tambahan.
2. Laptop/Komputer harus terhubung dengan wi-fi/ethernet yang stabil dengan latensi rendah untuk memastikan komunikasi yang lancar antara server dan klien, terutama saat menguji banyak klien sekaligus.
3. Laptop/Komputer harus dapat menjalankan program python (bahasa pemrograman Python versi ≥ 3.10)
4. Laptop/Komputer harus memiliki Prosesor yang cukup baik (misalnya Intel i5 atau yang lebih tinggi) direkomendasikan untuk memastikan operasi yang lebih lancar, terutama saat menangani banyak thread untuk komunikasi server-klien.
5. Laptop/Komputer harus memastikan firewall atau antivirus mengizinkan skrip Python untuk berkomunikasi melalui jaringan, karena kadang-kadang koneksi socket bisa terblokir. (Akan diminta permission saat pertama kali menjalankan server)

VII. Foto atau Tangkapan Layar

- **Antarmuka Program:**

```
1. Enter server IP: 192.168.137.1
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: adwmao
Connected to chat server!
Your message: JNEDSFIP joined the chat.
JNEDSFIP: KOPEWFJ
apala
[Server] Message acknowledged.
Your message: JNEDSFIP: SDFVJNIOPV
JNEDSFIP: python -u "c:\Users\LENOVO\Downloads\2_client
```

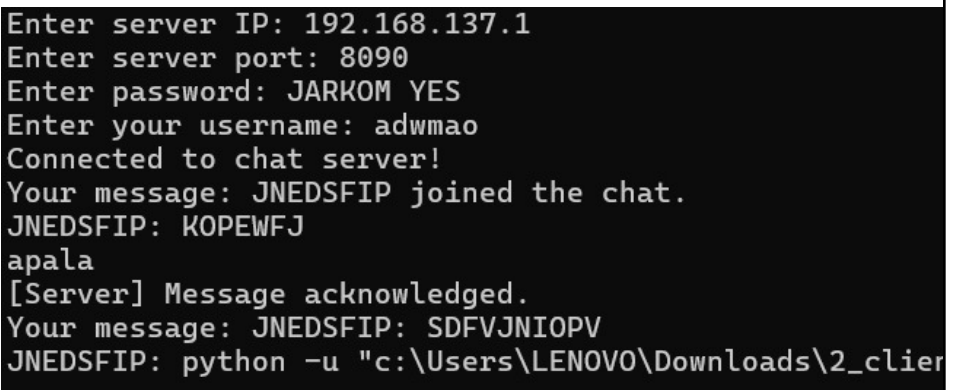
Gambar 7.1 Spesifikasi 1 dan 2 dari sisi Client(kiri jendela “FEAD”, kanan jendela “HIOFAS”)

Penjelasan: Gambar 7.1 menunjukkan percakapan dalam sebuah chat room. Seorang pengguna dengan nama "HIOFASD" mengirimkan pesan "asd" dan pengguna lain dengan nama "FEAD" bergabung dengan chat tersebut. Dan mengirimkan “hai”. Pesan “hai” dari “FEAD” tersampaikan pada jendela “HIOFASD” Ini sesuai dengan spesifikasi 1 dan 2 bahwa server mampu menerima pesan dari client dan meneruskannya ke client lain.

```
2. Enter server IP: 192.168.137.1
Enter server port: 8090
Server is running...
adwmao joined from ('192.168.137.1', 50551)
JNEDSFIP joined from ('192.168.137.107', 627
```

Gambar 7.2 Spesifikasi 1 dan 2 dari sisi server

Penjelasan: Gambar 7.2 menunjukkan sebuah terminal di mana ada pesan-pesan yang dikirim dan diterima oleh server. Ini sesuai dengan spesifikasi 1 dan 2 bahwa server dapat menerima dan meneruskan pesan.

3. 

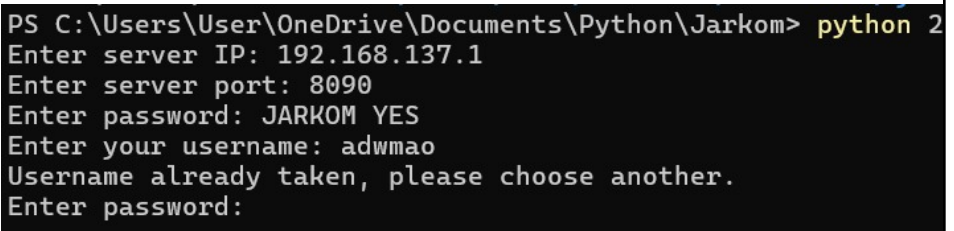
```

Enter server IP: 192.168.137.1
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: adwmao
Connected to chat server!
Your message: JNEDSFIP joined the chat.
JNEDSFIP: KOPEWFJ
apala
[Server] Message acknowledged.
Your message: JNEDSFIP: SDFVJNIOPV
JNEDSFIP: python -u "c:\Users\LENOVO\Downloads\2_client

```

Gambar 7.3 Spesifikasi 3 dan 5

Penjelasan: Gambar 7.3 menunjukkan sebuah antarmuka pengguna di mana pengguna dapat memasukkan alamat IP server, nomor port, password, dan username. Ini sesuai dengan spesifikasi 3, 5, dan 6 bahwa client dapat mengirimkan pesan ke server dengan IP dan port yang ditentukan dan harus memasukkan password

4. 

```

PS C:\Users\User\OneDrive\Documents\Python\Jarkom> python 2
Enter server IP: 192.168.137.1
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: adwmao
Username already taken, please choose another.
Enter password:

```

Gambar 7.4 Spesifikasi 6

Penjelasan: Gambar 7.4 menunjukkan sebuah pesan kesalahan yang menyatakan bahwa "Nama sudah dipakai, gunakan yang lain!" Ini sesuai dengan spesifikasi 6 bahwa client harus memiliki username yang unik.

- **Penggunaan atau Pengujian Program:**

```
1. Enter server IP: 192.168.137.1
Enter server port: 8090
Server is running...
adwmao joined from ('192.168.137.1', 50551)
JNEDSFIP joined from ('192.168.137.107', 62'
```

Gambar 7.5 Proses Pengujian Server

```
2. PS C:\Users\User\OneDrive\Documents\Python\Jarkom> python 2_client.py
Enter server IP: 90 bj
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: BEN
Traceback (most recent call last):
  File "C:\Users\User\OneDrive\Documents\Python\Jarkom\2_client.py", line 22, in <
    client_socket.sendto(f"SIGNUP_TAG:{username}".encode(), (SERVER_IP, SERVER_POF
socket.gaierror: [Errno 11001] getaddrinfo failed
PS C:\Users\User\OneDrive\Documents\Python\Jarkom>
```

Gambar 7.6 Tindakan ketika IP number tidak valid

```
3. Enter server IP: 192.168.137.1
Enter server port: 8090
Server is running...
adwmao joined from ('192.168.137.1', 50551)
JNEDSFIP joined from ('192.168.137.107', 62'
```

Gambar 7.7 Tindakan ketika server sudah berjalan

Penjelasan Hasil Pengujian Server: Dalam pengujian ini, kami menguji kode server dan berjalan lancar.

```

Enter server IP: 192.168.137.1
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: adwmao
Connected to chat server!
Your message: JNEDSFIP joined the chat.
JNEDSFIP: KOPEWFJ
apala
[Server] Message acknowledged.
Your message: JNEDSFIP: SDFVJNIOPV
JNEDSFIP: python -u "c:\Users\LENOVO\Downloads\2_client (1).py"

```

Gambar 7.8 Proses Pengujian Log-in user

```

Enter server IP: 192.168.137.1
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: adwmao
Connected to chat server!
Your message: JNEDSFIP joined the chat.
JNEDSFIP: KOPEWFJ
apala
[Server] Message acknowledged.
Your message: JNEDSFIP: SDFVJNIOPV
JNEDSFIP: python -u "c:\Users\LENOVO\Downloads\2_client (1).py"

```

Gambar 7.9 Tindakan ketika password salah

```

PS C:\Users\User\OneDrive\Documents\Python\Jarkom> python 2_client.py
Enter server IP: 192.168.137.1
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: adwmao
Username already taken, please choose another.
Enter password:

```

Gambar 7.10 Tindakan ketika nama tidak unik

Penjelasan Hasil Pengujian Server: Dalam pengujian ini, kami menguji kode client bagian login dan berjalan lancar. Tindakan ketika password salah dan nama tidak unik dapat terlaksanakan dengan baik.

```

Enter server IP: 192.168.137.1
Enter server port: 8090
Enter password: JARKOM YES
Enter your username: adwmao
Connected to chat server!
Your message: JNEDSFIP joined the chat.
JNEDSFIP: KOPEWFJ
apala
[Server] Message acknowledged.
Your message: JNEDSFIP: SDFVJNIOPV
JNEDSFIP: python -u "c:\Users\LENOVO\Downloads\2_client (1).py"

```

Gambar 7.11 UI ketika user berhasil masuk ke group chat

```

Your message:
Logging out...Exception in thread Thread-2 (send_messages):
Traceback (most recent call last):

  File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11.3.11.2544.0_x64__qbz5n2kfra8p0\Lib\threading.py", line 1045, in _bootstrap_inner
    Exception in thread Disconnected.Thread-1 (receive_messages)
PS C:\Users\LENOVO\OneDrive - Institut Teknologi Bandung\Desktop\Semester 3\C latihan program\coba\jarkom>

```

Gambar 7.12 UI ketika salah satu user berhasil logout

```

Your message:
Logging out...Exception in thread Thread-2 (send_messages):
Traceback (most recent call last):

  File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11.3.11.2544.0_x64__qbz5n2kfra8p0\Lib\threading.py", line 1045, in _bootstrap_inner
    Exception in thread Disconnected.Thread-1 (receive_messages)
PS C:\Users\LENOVO\OneDrive - Institut Teknologi Bandung\Desktop\Semester 3\C latihan program\coba\jarkom>

```

Gambar 7.13 Error/kekurangan dari program

Penjelasan Hasil Pengujian Server: Dalam pengujian ini, kami menguji kode client bagian groupchat dan logout. Kedua fitur tersebut berjalan dengan baik. Tetapi, ada satu kekurangan, yaitu ketika kami mau login setelah logout(user A), interface pada client (user A) not responding dan interface group chat tidak muncul..

Untuk penyebab error ini, kemungkinan karena thread “receive” pada sisi client yang tidak berhenti sepenuhnya saat keluar, yang membuat thread ini berjalan ganda setelah login ulang. Tetapi, saya telah memastikan thread berhenti saat logout, sehingga tidak

akan ada kebocoran thread yang menyebabkan freeze saat login lagi dengan membuat Flag dan Pengecekan Flag “is_logged_in” untuk memastikan loop dalam “receive” berhenti sepenuhnya saat logout. Namun, hal ini tidak menyelesaikan error yang ada.

VIII. Source Code Program Client dan Server

- Client Code:

```
1  import socket
2  import threading
3
4  # Client configuration
5  SERVER_IP = input("Enter server IP: ") or '127.0.0.1'
6  SERVER_PORT = int(input("Enter server port: ") or 9090)
7  RETRANSMISSION_TIMEOUT = 2 # seconds
8
9  # Set up UDP socket
10 client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
11 client_socket.settimeout(RETRANSMISSION_TIMEOUT)
12
13 # Login credentials
14 password = "JARKOM YES"
15 attempts = 0
16 max_attempts = 3
17
18 while attempts < max_attempts:
19     user_password = input("Enter password: ")
20     if user_password == password:
21         username = input("Enter your username: ")
22         client_socket.sendto(f"SIGNUP_TAG:{username}".encode(), (SERVER_IP, SERVER_PORT))
23
24         # Wait for server's response on username availability
25         try:
26             response, _ = client_socket.recvfrom(1024)
27             if response.decode() == "Username already taken, please use another.":
28                 print("Username already taken, please choose another.")
29             else:
30                 print("Connected to chat server!")
31                 break
32         except socket.timeout:
33             print("Could not connect to server: timeout")
34             exit()
35     else:
36         attempts += 1
```

```

37         print(f"Incorrect password. Attempts remaining: {max_attempts - attempts}")
38
39     if attempts == max_attempts:
40         print("Maximum attempts reached. Exiting...")
41         exit()
42
43     def receive_messages():
44         """Receive messages from the server."""
45         while True:
46             try:
47                 data, _ = client_socket.recvfrom(1024)
48                 message = data.decode()
49
50                 # Check if the message is an ACK
51                 if message.startswith("ACK|"):
52                     ack_sequence = int(message.split("|")[1])
53                     if ack_sequence == sequence_number:
54                         print("[Server] Message acknowledged.")
55                         ack_received.set()
56                 else:
57                     print(message) # Display broadcasted messages from other clients
58
59             except socket.timeout:
60                 continue
61
62     def send_messages():
63         """Send messages to the server, with retransmission on timeout."""
64         global sequence_number
65         sequence_number = 0
66         while True:
67             message_content = input("Your message: ")
68             full_message = f"{sequence_number}|{username}: {message_content}"
69
70             # Use an event to wait for ACK
71             global ack_received
72             ack_received = threading.Event()

```

```

73
74             # Attempt retransmission until acknowledged
75             while not ack_received.is_set():
76                 client_socket.sendto(full_message.encode(), (SERVER_IP, SERVER_PORT))
77                 if not ack_received.wait(RETRANSMISSION_TIMEOUT):
78                     print(f"[TIMEOUT] Resending message '{message_content}'...")
79                 sequence_number += 1
80
81     # Start receiving and sending threads
82     threading.Thread(target=receive_messages, daemon=True).start()
83     threading.Thread(target=send_messages, daemon=True).start()
84
85     # Keep client running
86     try:
87         while True:
88             pass
89     except KeyboardInterrupt:
90         print("\nLogging out...")
91         client_socket.sendto(f"LOGOUT_TAG:{username}".encode(), (SERVER_IP, SERVER_PORT))
92         client_socket.close()
93         print("Disconnected.")

```

Gambar 8.1 Client Code

- **Server Code:**

```
1 import socket
2 import threading
3 import queue
4
5 # Configuration for the server
6 SERVER_IP = input("Enter server IP: ") or '127.0.0.1'
7 SERVER_PORT = int(input("Enter server port: ") or 9090)
8
9 # Set up the UDP socket
10 server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
11 server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
12 server_socket.bind((SERVER_IP, SERVER_PORT))
13
14 messages = queue.Queue()
15 clients = {}
16
17 def handle_client(message, client_address):
18     """Process client messages and handle ACKs, broadcasting to other clients."""
19     global clients
20
21     decoded_message = message.decode()
22
23     if decoded_message.startswith("SIGNUP_TAG:"):
24         name = decoded_message.split(":")[1]
25         if name in clients.values():
26             server_socket.sendto("Username already taken, please use another.".encode(), client_address)
27         else:
28             clients[client_address] = name
29             print(f"{name} joined from {client_address}")
30             for client in clients:
31                 server_socket.sendto(f"{name} joined the chat.".encode(), client)
32             return
33
34     elif decoded_message.startswith("LOGOUT_TAG:"):
35         name = decoded_message.split(":")[1]
36         if client_address in clients:
37             del clients[client_address]
```



```

38         print(f"{name} left from {client_address}")
39         for client in clients:
40             server_socket.sendto(f"{name} has left.".encode(), client)
41     return
42
43     # Handle regular message with sequence number
44     if "|" in decoded_message:
45         seq_number, client_msg = decoded_message.split("|", 1)
46         seq_number = int(seq_number)
47
48         # Send ACK for the received message to the sender
49         ack_message = f"ACK|{seq_number}"
50         server_socket.sendto(ack_message.encode(), client_address)
51
52         # Broadcast the message to all other clients
53         for client in clients:
54             if client != client_address:
55                 server_socket.sendto(client_msg.encode(), client)
56
57 def receive():
58     """Receive messages from clients and add them to the queue."""
59     print("Server is running...")
60     while True:
61         message, client_address = server_socket.recvfrom(1024)
62         messages.put((message, client_address))
63         handle_client(message, client_address)
64
65 # Start the server thread
66 threading.Thread(target=receive, daemon=True).start()
67
68 # Keep the server running
69 try:
70     while True:
71         pass

```

```


67
68 # Keep the server running
69 try:
70     while True:
71         pass
72 except KeyboardInterrupt:
73     print("\nServer shutting down.")

```

Gambar 8.2 Server Code

IX. Tautan ke GitHub atau Google Drive

Berikut merupakan Tautan Github program dan tautan video demo kami:

- **Tautan Kode Program :** <https://github.com/Vannier19/jarkom.git>
- **Tautan Video:**  Video_Demo_Tugas_Jarkom

X. Tabel Pembagian Tugas atau Kontribusi Anggota

Tabel 9.1 Pembagian Tugas setiap anggota

No	Spesifikasi	Pembagian
1	Server mampu menerima pesan yang dikirim client dan mencetaknya ke layar.	Server: 18223028
2	Server mampu meneruskan pesan satu client ke client lain.	Server: 18223028
3	Client mampu mengirimkan pesan ke server dengan IP dan port yang ditentukan pengguna.	Client: 18223026 Pencarian port: 18223028
4	Client mampu menerima pesan dari client lain (yang diteruskan oleh server), dan mencetaknya ke layar.	Client: 18223026
5	Client harus memasukkan <i>password</i> untuk dapat bergabung ke chatroom.	Client: 18223028
6	Client memiliki username yang unik.	Client: 18223028
7	Aplikasi Memiliki GUI	18223028
8	Aplikasi mengimplementasikan TCP over UDP. Note: asisten sangat menyarankan untuk mengerjakan spesifikasi ini, karena akan memberikan pemahaman kuat terkait TCP.	18223026
9	Design/struktur keseluruhan laporan	18223028
10	Sampul atau cover yang setidaknya memuat nama kelompok, nama lengkap serta NIM setiap anggotanya, dan gambar atau foto bebas yang boleh menggantikan logo ITB.	18223028
11	Daftar isi, gambar, dan tabel.	18223028
12	Tabel penyelesaian spesifikasi; silahkan <i>copy paste</i> tabel-tabel spesifikasi di atas, lalu ganti kolom nilai dengan kolom "selesai" yang berisi <i>checklist</i> . Tambahkan juga kolom keterangan bila perlu.	18223026
13	Penjelasan terkait cara kerja program yang merujuk setidaknya ke satu pustaka atau literatur (boleh <i>slide</i> kuliah). Semakin lengkap, semakin bagus.	18223026
14	Penjelasan terkait tata cara dan lingkungan pengujian program.	18223028

15	Foto atau tangkapan layar yang menunjukkan antarmuka dan contoh penggunaan atau pengujian program; hubungkan gambar-gambar dengan spesifikasi terkait. Seluruh gambar harus dilengkapi penjelasan. Jelaskan juga hasil pengujian bila perlu, misal jika terjadi kerusakan pada pesan sebagaimana dijabarkan sebelumnya.	18223028
16	<i>Source code</i> program <i>client</i> dan <i>server</i> .	18223028
17	Tautan ke GitHub atau <i>Google Drive</i> yang berisi kode program, serta tautan video apabila mengerjakan. Pastikan tautan bersifat publik.	18223026
18	Tabel yang menunjukkan pembagian tugas atau kontribusi masing-masing anggota. NOTE: Dilarang membagi tugas menjadi program-laporan. Seluruh anggota kelompok harus memiliki andil dalam seluruh deliverables.	18223028
19	Daftar pustaka.	18223026

XI. Daftar Pustaka

- YouTube. "Building a Python UDP Chat Server." Diakses 11 Oktober 2024.
<https://www.youtube.com/watch?v=3QiPPX-KeSc>.
- YouTube. "UDP vs TCP – What is the Difference?" Diakses 13 Oktober 2024.
<https://www.youtube.com/watch?v=szm3camsf8k>.
- YouTube. "Understanding UDP Socket Programming." Diakses 13 Oktober 2024.
https://youtu.be/CC58_XY8yLQ?si=oQEW8wjtq9oYQngg.
- YouTube. "Python Socket Programming with UDP." Diakses 13 Oktober 2024.
https://youtu.be/esLgiMLbRkI?si=J1TdH96_oRTRLccX.
- YouTube. "Introduction to Networking Protocols: UDP." Diakses 13 Oktober 2024.
https://youtu.be/3qlhbez-RPI?si=qT_GM-aE7pJCfyv2.
- YouTube. "How UDP Works – Networking Concepts." Diakses 15 Oktober 2024.
<https://youtu.be/sopNW98CRag?si=K7gRYZWn6qNIwNkY>.
- YouTube. "Programming with UDP Sockets in Python." Diakses 16 Oktober 2024.
<https://youtu.be/ibf5cx221hk?si=9hpTTR-uTcFZSnkT>.
- YouTube. "How to Implement UDP in Python – Networking Basics." Diakses 16 Oktober 2024. <https://youtu.be/MLvBToAhiI4?si=84rVT6KhZp8IXA0c>.
- YouTube. "UDP Protocol Explained – Detailed Overview." Diakses 17 Oktober 2024.
<https://youtu.be/zWaTOJqStsU?si=qXzQJU04FkeU8Gtx>.
- YouTube. "Creating a UDP Server in Python – Step-by-Step Guide." Diakses 23 Oktober 2024. https://youtu.be/U_Q1vqaJi34?si=CKcJj6O6E52Vu1mP.