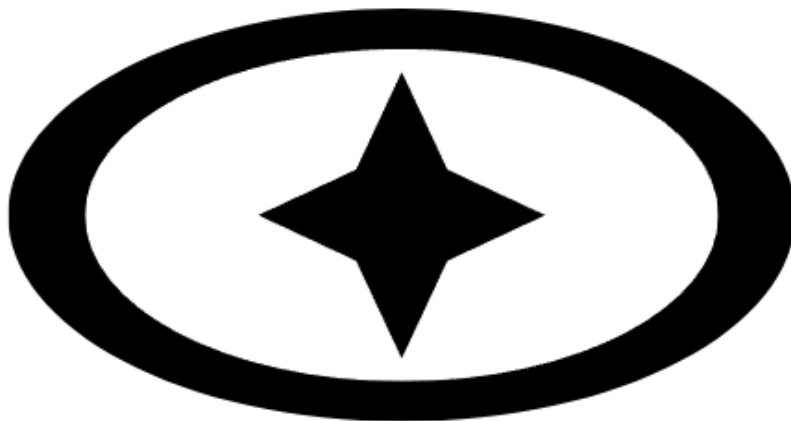


Tugas UDP Socket Programming

II2100

Jaringan Komputer



Disusun oleh:

Kelompok MataKiri

Joan Melkior Silaen (18223102)

Vandega Arozan Musholine (18223010)

Institut Teknologi Bandung

2024

Daftar Isi

Daftar Isi.....	1
I. Penyelesaian Spesifikasi.....	2
II. Cara Kerja Program.....	2
Komunikasi Berbasis Socket.....	2
Protokol UDP.....	2
Arsitektur Client-Server.....	3
Implementasi Multi-threaded pada Server.....	3
III. Tata Cara Pengujian.....	3
IV. Lingkungan Pengujian.....	3
V. Tangkapan Layar.....	4
VI. Source Code.....	6
Server.....	6
Client.....	8
VII. Pembagian Tugas.....	10
DAFTAR PUSTAKA.....	11

I. Penyelesaian Spesifikasi

No	Spesifikasi	Selesai
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 1.1. Tabel Penyelesaian Spesifikasi

II. Cara Kerja Program

Komunikasi Berbasis Socket

Socket merupakan endpoint dalam komunikasi jaringan antara dua perangkat. Pada program yang sudah dibuat, melalui socket yang telah dipersiapkan pada port yang ditentukan, server akan menunggu koneksi dari client. Saat client terhubung, *socket* dapat mengirim dan menerima data antara dua pihak secara kontinu sampai koneksi ditutup.

Protokol UDP

Meskipun mode komunikasi UDP belum tentu memiliki urutan atau integritas data yang sama dengan TCP, namun UDP diaplikasikan pada mode komunikasi yang lebih cepat. Mode *connectionless* yang ada tidak membutuhkan koneksi permanen, sehingga efisiensinya lebih tinggi untuk situasi yang memerlukan kecepatan, seperti streaming video atau aplikasi real-time yang tidak begitu terdampak oleh sedikit kehilangan data. UDP mengirim pesan dalam bentuk *datagrams* tanpa membutuhkan *handshaking* dengan server.

Arsitektur Client-Server

Menggunakan alamat IP dan port yang ditentukan, client menginisiasi koneksi ke server. Setelah terhubung, client mampu mengirim data ke server (seperti message). Server menjadi pusat pemrosesan, mendapat permintaan dari client dan memberikan respons. Program ini mempraktikkan cara client mampu mengirim permintaan untuk berbagai tindakan dan cara server memberikan respons sesuai dengan data yang diterima.

Implementasi Multi-threaded pada Server

Server menggunakan pendekatan *multi-threading* agar dapat mendukung adanya beberapa client dalam satu waktu yang sama, di mana masing-masing client yang terhubung ditangani oleh thread yang berbeda. Hal ini mampu membuat server menerima dan merespons pesan dari beberapa client dalam satu waktu yang sama, menjamin ketersediaan koneksi yang baru.

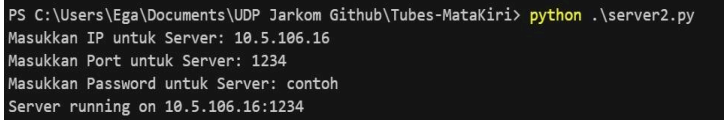
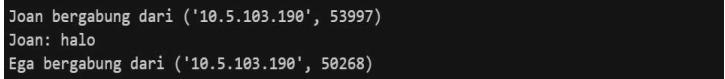


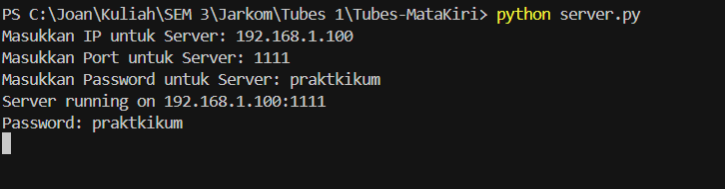
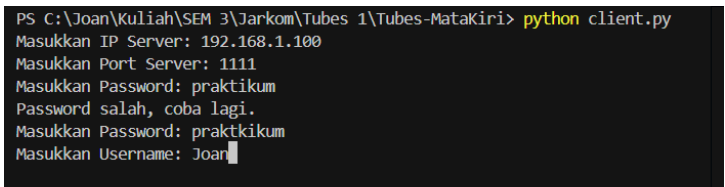
III. Tata Cara Pengujian

1. Siapkan dua device (dalam percobaan ini berupa laptop)
2. Run server file pada salah satu laptop (dalam percobaan ini, run dilakukan dalam terminal VSCode)
3. Dalam terminal, masukkan IP address pada server sesuai keinginan
4. Dalam terminal, masukkan port pada server sesuai keinginan
5. Dalam terminal, atur password yang ingin dijadikan sebagai kata sandi bagi client nantinya
6. Run file client pada laptop lainnya (dalam percobaan ini, run dilakukan dalam terminal VSCode)
7. Masukkan IP address sesuai dengan yang ada pada server
8. Masukkan port sesuai dengan yang ada pada server
9. Masukkan password sesuai dengan yang ada pada server
10. Masukkan username sesuai keinginan
11. Masukkan pesan yang ingin dikirim
12. Apabila ingin menambah jumlah klien, lakukan run file client dalam terminal baru dengan mengulangi langkah 7–11, dengan catatan username client baru berbeda dengan username yang sudah ada
13. Apabila ingin meninggalkan chatroom, maka ketik 'exit'.

IV. Lingkungan Pengujian

Pengujian memiliki tujuan untuk memastikan stabilitas komunikasi di lingkungan yang memungkinkan proses multithreading dan socket berbasis IP. Dalam pengujian yang kami lakukan, lokasi pengujian berada di lingkungan kampus yang memiliki fasilitas Wi-Fi. Pengujian dilakukan menggunakan dua laptop yang berbeda. Pada setiap laptop, digunakan aplikasi VSCode untuk melakukan run file. Proses pengujian melingkupi pengiriman pesan antara client dan client, client dan server, serta pengecekan kecepatan dan efisiensi pengiriman data protokol UDP.

V. Tangkapan Layar

No	Foto	Keterangan
1	 <pre>PS C:\Users\Ega\Documents\UDP Jarkom Github\Tubes-MataKiri> python .\server2.py Masukkan IP untuk Server: 10.5.106.16 Masukkan Port untuk Server: 1234 Masukkan Password untuk Server: contoh Server running on 10.5.106.16:1234</pre>	Contoh run pada server, dengan masukan IP, port, dan password sesuai keinginan, lalu mencetak "server running on IP:port"
2	 <pre>Joan bergabung dari ('10.5.103.190', 53997) Joan: halo Ega bergabung dari ('10.5.103.190', 50268)</pre>	Notifikasi muncul pada server jika client berhasil terhubung dengan server
3	 <pre>Ega: halo juga Joan: apa kabar Ega: baik</pre>	Server dapat menampilkan percakapan antar-client
4	 <pre>Ega telah keluar dari chatroom. Joan telah keluar dari chatroom.</pre>	Server dapat menampilkan notifikasi apabila ada client yang keluar dari chatroom
5	 <pre>PS C:\Joan\Kuliah\SEM 3\Jarkom\Tubes 1\Tubes-MataKiri> python server.py Masukkan IP untuk Server: 192.168.1.100 Masukkan Port untuk Server: 1111 Masukkan Password untuk Server: praktikum Server running on 192.168.1.100:1111 Password: praktikum</pre>	Contoh run pada file client, dengan masukan IP, port, dan password sesuai dengan server
6	 <pre>PS C:\Joan\Kuliah\SEM 3\Jarkom\Tubes 1\Tubes-MataKiri> python client.py Masukkan IP Server: 192.168.1.100 Masukkan Port Server: 1111 Masukkan Password: praktikum Password salah, coba lagi. Masukkan Password: praktikum Masukkan Username: Joan</pre>	Contoh handler jika password salah

7	<pre> Anda bergabung sebagai Joan > Joan bergabung ke chatroom. > █ </pre>	Contoh jika client berhasil terhubung dengan server
8	<div> <pre> Anda bergabung sebagai Joan Joan bergabung ke chatroom. > > Ega bergabung ke chatroom. > █ </pre> </div> <div> <pre> Anda bergabung sebagai Ega Ega bergabung ke chatroom. > > █ </pre> </div>	Point of view kedua client ketika berhasil terhubung dengan server
9	<div> <pre> Anda bergabung sebagai Joan Joan bergabung ke chatroom. > > Ega bergabung ke chatroom. > Ega: halo > Joan: halo juga > Ega: apa kabar > Joan: baik > █ </pre> </div> <div> <pre> Anda bergabung sebagai Ega Ega bergabung ke chatroom. > Ega: halo > Joan: halo juga > Ega: apa kabar > Joan: baik > █ </pre> </div>	Contoh tampilan ketika dua client saling mengirim pesan
10	<div> <pre> Anda bergabung sebagai Joan Joan bergabung ke chatroom. > > Ega bergabung ke chatroom. > Ega: halo > Joan: halo juga > Ega: apa kabar > Joan: baik Anda telah keluar dari chatroom. Terputus dari server. PS C:\Joan\Kuliah\SEM 3\Jarkom\Tubes 1\Tubes-MataKiri> █ </pre> </div> <div> <pre> Anda bergabung sebagai Ega Ega bergabung ke chatroom. > Ega: halo > Joan: halo juga > Ega: apa kabar > Joan: baik > Joan telah keluar dari chatroom. > █ </pre> </div>	Tampilan ketika client meninggalkan chatroom

Tabel 5.2. Tabel Tangkapan Layar Pengujian Program

VI. Source Code

Server

```

import socket
import threading
import queue

SERVER_IP = input("Masukkan IP untuk Server: ") # Input IP dari admin server
SERVER_PORT = int(input("Masukkan Port untuk Server: ")) # Input port dari admin server
PASSWORD = input("Masukkan Password untuk Server: ") # Input password dari admin server

clients = []
messages = queue.Queue()
usernames = {}

try:
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind((SERVER_IP, SERVER_PORT))
    print(f"Server running on {SERVER_IP}:{SERVER_PORT}")

```

```

except socket.error as e:
    if e.errno == 98: # Port already in use
        print(f"Port {SERVER_PORT} sudah digunakan")
        SERVER_PORT = int(input("Masukkan Port lain: "))
        server_socket.bind((SERVER_IP, SERVER_PORT))
    else:
        print(f"Error: {e}")
        exit(1)

def broadcast():
    while True:
        while not messages.empty():
            message, addr = messages.get()
            for client in clients:
                try:
                    server_socket.sendto(message, client)
                except socket.error as e:
                    # Handle disconnection errors gracefully
                    if client in clients:
                        clients.remove(client)

def handle_client():
    while True:
        try:
            data, address = server_socket.recvfrom(1024)
            message = data.decode()

            if address not in clients:
                if message.startswith("PASSWORD "):
                    if message.split()[1] == PASSWORD:
                        server_socket.sendto(b"PASSWORD OK", address)
                    else:
                        server_socket.sendto(b"PASSWORD INCORRECT", address)
                elif message.startswith("USERNAME "):
                    username = message.split()[1]
                    if username in usernames.values():
                        server_socket.sendto(b"USERNAME TAKEN", address)
                    else:
                        usernames[address] = username
                        clients.append(address)
                        messages.put((f"{username} bergabung ke
chatroom.".encode(), address))
                        server_socket.sendto(b"USERNAME OK", address)
                        print(f"{username} bergabung dari {address}")

```

```

        else:
            if message.lower().strip() == "exit":
                username = usernames.pop(address, "User")
                exit_message = f"{username} telah keluar dari chatroom."
                messages.put((exit_message.encode(), address))
                clients.remove(address)
                print(exit_message)
            else:
                print(f"{message}")
                messages.put((message.encode(), address))
    except socket.error as e:
        if e.errno == 10054:
            continue
        print(f"Error: {e}")

if __name__ == "__main__":
    print(f"Password: {PASSWORD}")

    broadcast_thread = threading.Thread(target=broadcast)
    broadcast_thread.daemon = True
    broadcast_thread.start()

    handle_thread = threading.Thread(target=handle_client)
    handle_thread.daemon = True
    handle_thread.start()

    try:
        while True:
            pass
    except KeyboardInterrupt:
        print("\n\nServer shutting down...")
        server_socket.close()

```

Client

```

import socket
import threading
import sys

SERVER_IP = input("Masukkan IP Server: ")
SERVER_PORT = int(input("Masukkan Port Server: "))

```



```

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

def clear_screen():
    # Untuk membersihkan tampilan terminal
    sys.stdout.write("\033[H\033[J")
    sys.stdout.flush()

def clear_last_line():
    # Untuk menghapus baris terakhir setelah input message
    sys.stdout.write("\033[F\033[K")
    sys.stdout.flush()

def receive_messages():
    while True:
        try:
            message, _ = client_socket.recvfrom(1024)
            print(message.decode() + "\n> ", end="")
        except:
            print("Terputus dari server.")
            break

# Loop untuk verifikasi password
while True:
    password = input("Masukkan Password: ")
    client_socket.sendto(f"PASSWORD {password}".encode(), (SERVER_IP,
SERVER_PORT))
    password_response, _ = client_socket.recvfrom(1024)

    if password_response.decode() == "PASSWORD OK":
        break
    else:
        print("Password salah, coba lagi.")

# Loop untuk verifikasi username
while True:
    username = input("Masukkan Username: ")
    client_socket.sendto(f"USERNAME {username}".encode(), (SERVER_IP,
SERVER_PORT))
    username_response, _ = client_socket.recvfrom(1024)

    if username_response.decode() == "USERNAME OK":
        break
    elif username_response.decode() == "USERNAME TAKEN":

```

```

        print("Username sudah digunakan. Coba username lain.")
    else:
        print("Terjadi kesalahan, coba lagi.")

# Setelah berhasil login, bersihkan layar dan tampilkan pesan selamat datang
clear_screen()
print(f"Anda bergabung sebagai {username}")

# Mulai thread untuk menerima pesan dari server
receive_thread = threading.Thread(target=receive_messages)
receive_thread.daemon = True
receive_thread.start()

# Loop untuk mengirim pesan
while True:
    message = input("> ")
    clear_last_line()
    if message.lower() == "exit":
        client_socket.sendto("exit".encode(), (SERVER_IP, SERVER_PORT))
        print("Anda telah keluar dari chatroom.")
        break
    client_socket.sendto(f"{username}: {message}".encode(), (SERVER_IP,
SERVER_PORT))


client_socket.close()

```

VII. Pembagian Tugas

No	Nama	Tugas
1	Joan Melkior Silaen	<ul style="list-style-type: none"> - Menyusun server.py - Menyusun client.py - Membuat dan menyusun repository - Menyusun dokumen bagian daftar isi, bagian V dan VI
2	Vandega Arozan Musholine	<ul style="list-style-type: none"> - Menyusun server.py - Menyusun client.py - Menyusun dokumen bagian cover, I, II, III, IV, V, dan VII

Tabel 7.3. Tabel Pembagian Tugas

Video:  Demonstrasi UDP Jarkom
Github : [25_Tugas Socket Programming](#)

DAFTAR PUSTAKA

Kurose, J. & Ross, K. (2020). *Computer Networking : A Top-Down Approach*. New Jersey : Pearson

Elgheriani, N. S. & Dkhila, S. S. (2023). *Client/Server Chatting Program (Using TCP/UDP Datagrams)*. Diakses dari <https://www.minarjournal.com/dergi/client-server-chatting-program-using-tcp-udp-datagrams20230927032955.pdf>.