**LAPORAN PRAKTIKUM**

**Chat Sederhana Berbasis Socket di Python**

****

|  |  |
| --- | --- |
| Nama | : L Hafidl Alkhair |
| Nim | : 2023903430060 |
| Kelas | : TRKJ-2C |
| Jurusan | : Teknologi Informasi dan Komputer |
| Progam Studi | : Teknologi Rekayasa Komputer dan Jaringan |
| Dosem Pengampu | : Afla Nevrisa S.Kom, M.Kom |

****

**PROGRAM STUDI TEKNOLOGI REKAYASA KOMPUTER JARINGAN**

**JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER**

**POLITEKNIK NEGERI LHOKSEUMAWE 2025**

# LEMBAR PENGESAHAN

|  |  |
| --- | --- |
| No. Praktikum | : 02 /TIK/TRKJ-2C/ Pemrograman Jaringan |
| Judul | : Laporan Praktikum |
| Nama | : L Hafidl Alkhair |
| Nim | : 202390343060 |
| Kelas | : TRKJ-2C |
| Jurusan | : Teknologi Informasi Dann Komputer |
| Program Studi | : Teknologi Rekayasa Komputer Jaringan |
| Tanggal Praktikum | : 20 Mei 2025 |
| Tanggal Penyerahan | : 1 Juni 2025 |

|  |
| --- |
| Buketrata, 1 Juni 2025 |
| Dosen Pembimbing, |
| **Afla Nevrisa S.Kom, M.Kom**  NIP. 199211172022032007 |

BAB I

PENDAHULUAN

1. Dasar teori
2. Socket Programming

Socket adalah antarmuka yang memungkinkan komunikasi antara dua node dalam jaringan. Dalam pemrograman Python, socket digunakan untuk membuat aplikasi jaringan seperti server dan client. Socket menyediakan dua endpoint: socket server dan socket client, yang saling bertukar data.

1. TCP/IP

Transmission Control Protocol (TCP) adalah protokol komunikasi yang andal dan banyak digunakan di internet. Protokol ini menjamin bahwa data yang dikirim akan sampai ke tujuan secara utuh dan berurutan. Program chat ini menggunakan socket TCP untuk memastikan pesan antar client terkirim dengan benar.

1. Multithreading

Multithreading memungkinkan aplikasi untuk menjalankan beberapa bagian kode secara bersamaan. Dalam aplikasi server, threading digunakan untuk menangani banyak client secara paralel tanpa membuat server menunggu satu client selesai.

1. Client-Server Architecture

Arsitektur ini memisahkan sistem menjadi dua komponen utama: client (pengguna) dan server (penyedia layanan). Server akan terus mendengarkan permintaan dari client dan menanggapinya sesuai logika aplikasi.

1. Tujuan Praktikum

* Menerapkan konsep dasar pemrograman socket di Python.
* Membuat program server dan client yang dapat berkomunikasi menggunakan socket.
* Mengimplementasikan multithreading pada sisi server untuk menangani banyak client secara bersamaan.
* Memahami prinsip komunikasi jaringan berbasis TCP/IP.

1. Alat dan bahan

* Laptop/PC
* Python 3.x
* Text editor (seperti VS Code, Sublime Text, atau lainnya)
* Terminal atau Command Prompt
* Jaringan lokal (opsional)

BAB II

PRAKTIKUM

1. **Membuat Program Server**

Buat file server.py dengan kode berikut:

#!/usr/bin/env python3

import socket

import threading

# Konfigurasi Server

IP = '127.0.0.1'

PORT = 3606

MAX\_CLIENTS = 5

# Daftar client yang terhubung

clients = []

# Fungsi broadcast pesan ke semua client kecuali pengirim

def broadcast(message, sender\_socket):

for client in clients:

if client != sender\_socket:

try:

client.send(message)

except:

client.close()

if client in clients:

clients.remove(client)

# Fungsi handle untuk masing-masing client

def handle\_client(client\_socket, addr):

print(f"[+] {addr} terhubung.")

broadcast(f"[INFO] {addr} bergabung ke server.\n".encode('utf-8'), client\_socket)

while True:

try:

message = client\_socket.recv(1024)

if not message:

break

print(f"[{addr}] {message.decode('utf-8')}")

broadcast(f"[{addr}] {message.decode('utf-8')}".encode('utf-8'), client\_socket)

except:

break

# Jika client keluar

print(f"[-] {addr} putus koneksi.")

broadcast(f"[INFO] {addr} keluar dari server.\n".encode('utf-8'), client\_socket)

if client\_socket in clients:

clients.remove(client\_socket)

client\_socket.close()

def main():

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind((IP, PORT))

server.listen()

print(f"[\*] Server berjalan di {IP}:{PORT}")

while True:

client\_socket, addr = server.accept()

if len(clients) < MAX\_CLIENTS:

clients.append(client\_socket)

thread = threading.Thread(target=handle\_client, args=(client\_socket, addr))

thread.start()

print(f"[Aktif] Jumlah koneksi: {len(clients)}")

else:

print(f"[!] Menolak {addr}: server penuh.")

client\_socket.send("Server penuh. Silakan coba lagi nanti.\n".encode('utf-8'))

client\_socket.close()

if \_name\_ == "\_main\_":

main()

1. **Membuat Program client**

Buat file client.py dengan kode berikut:

#!/usr/bin/env python3

import socket

import threading

# Konfigurasi Client

SERVER\_IP = '127.0.0.1'

SERVER\_PORT = 3606

def receive\_messages(client):

while True:

try:

message = client.recv(1024).decode('utf-8')

if not message:

break

print(message)

except:

print("[!] Terputus dari server.")

client.close()

break

def main():

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

client.connect((SERVER\_IP, SERVER\_PORT))

except:

print("[!] Gagal terhubung ke server.")

return

print("[\*] Terhubung ke server. Ketik 'exit' untuk keluar.")

# Thread untuk menerima pesan

thread = threading.Thread(target=receive\_messages, args=(client,))

thread.start()

# Kirim pesan

while True:

message = input()

if message.lower() == 'exit':

break

try:

client.send(message.encode('utf-8'))

except:

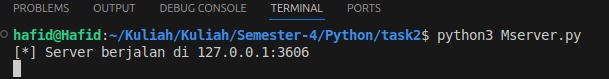
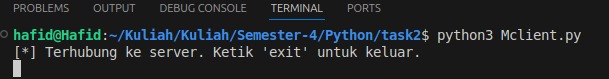
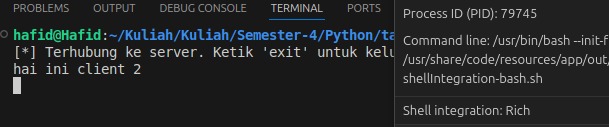
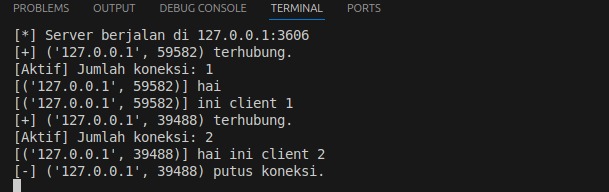
print("[!] Koneksi putus.")

break

client.close()

if \_name\_ == "\_main\_":

main()

1. **Output program**
2. Server dijalnkan
3. Client 1 dijalankan
4.  Client mengirimkan pesan
5. Log pada server
6. **Penjelasan dan Analisa**

Server (server.py)

Program server dibuat untuk menerima koneksi dari banyak client. Setiap kali ada client terhubung, server membuat thread baru untuk menangani komunikasi dengan client tersebut. Server juga melakukan broadcast ke seluruh client untuk menyampaikan pesan yang diterima dari salah satu client.

Fitur utama:

* Menggunakan TCP socket.
* Menyimpan semua koneksi client dalam list clients.
* Setiap pesan yang diterima dari satu client akan dikirim ke client lain melalui fungsi broadcast.
* Menangani koneksi maksimum 5 client secara simultan.

Client (client.py)

Client menghubungkan diri ke server dan dapat mengirimkan serta menerima pesan. Pesan dari server akan ditampilkan di terminal.

Fitur utama:

* Terhubung ke server melalui IP dan PORT yang ditentukan.
* Menggunakan thread untuk menerima pesan secara paralel dari input pengguna.
* Dapat keluar dari chat dengan mengetik exit.

Analisa Output:

* Saat server dijalankan, ia mendengarkan pada 127.0.0.1:3606.
* Saat client terhubung, muncul pesan pada server bahwa client terhubung.
* Pesan yang diketik oleh satu client akan muncul di terminal client lain, menunjukkan bahwa broadcast bekerja.
* Jika client keluar, server dan client lain menerima informasi tersebut.

Program ini berhasil menunjukkan komunikasi dua arah (atau lebih) antar client melalui perantara server, sesuai dengan konsep dasar chat room.

BAB III

PENUTUP

1. Kesimpulan

Melalui praktikum ini, mahasiswa mampu memahami dan mengimplementasikan komunikasi antar komputer menggunakan socket di Python. Program chat sederhana ini menunjukkan bagaimana server dapat menangani banyak client dengan multithreading, dan bagaimana pesan dapat ditransmisikan secara efisien melalui jaringan lokal menggunakan protokol TCP.