LAPORAN PRAKTIKUM
TRANSFER FILE SEDERHANA
MENGGUNAKAN 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	: 03 /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

DAFTAR ISI

BAB I
PENDAHULUAN

A. Dasar teori

1. Socket Programming

   Socket adalah antarmuka komunikasi antar perangkat dalam jaringan. Dalam konteks Python, modul socket digunakan untuk membuat aplikasi yang memungkinkan komunikasi antara client dan server melalui jaringan berbasis TCP/IP.

2. TCP (Transmission Control Protocol)

   TCP adalah protokol yang andal untuk komunikasi data karena menjamin pengiriman data secara utuh dan berurutan. Protokol ini cocok digunakan untuk mentransfer file karena integritas data sangat penting.

3. File Transfer Protocol (FTP) Sederhana

   Praktikum ini meniru prinsip kerja FTP, di mana client mengirim file ke server. Meskipun sederhana, prinsip dasarnya mencakup pembukaan koneksi, pengiriman nama file, pembacaan byte data file, dan penyimpanan file oleh server.

4. Konsep Buffer dan Stream Data

   File dikirim dalam bentuk byte stream melalui buffer. Buffer berfungsi sebagai ukuran potongan data yang dikirim secara bertahap untuk menghindari overload jaringan dan memudahkan kontrol proses pengiriman.

5. I/O dan Penanganan File di Python

   Modul os, sys, dan fungsi open() digunakan untuk membaca file yang akan dikirim dan menyimpannya di sisi server. Ini merupakan dasar manajemen file dalam sistem berbasis Python.

B. Tujuan Praktikum

- Menerapkan konsep pemrograman socket untuk mentransfer file dari client ke server.

- Membangun program server dan client untuk melakukan pengiriman file menggunakan TCP.
- Menguji keandalan komunikasi jaringan dan integritas data selama proses transfer file.
- Memahami penggunaan buffer dan kontrol progres pengiriman data.

C. 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**

Membuat file server.py

```
#!/usr/bin/env python3
import socket
import os

UPLOAD_DIR = "uploads"
os.makedirs(UPLOAD_DIR, exist_ok=True)

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('0.0.0.0', 5001))
server.listen(5)

print("[*] Menunggu koneksi...")

while True:
    client_socket, addr = server.accept()
    print(f"[+] Terhubung dengan {addr}")

    try:
        filename = client_socket.recv(1024).decode()
        print(f"[INFO] Menerima file: {filename}")
        save_path = os.path.join(UPLOAD_DIR, f"received_{filename}")

        with open(save_path, "wb") as f:
            while True:
                bytes_read = client_socket.recv(4096)
                if not bytes_read:
```

```
                break
            f.write(bytes_read)

        print(f"[INFO] File {filename} berhasil disimpan di {save_path}")
        client_socket.send("[Server] File berhasil diterima.".encode())
    except Exception as e:
        print(f"[ERROR] {e}")
        client_socket.send(f"[Server] Terjadi kesalahan: {e}".encode())
    finally:
        client_socket.close()
```

2. Membuat Program client

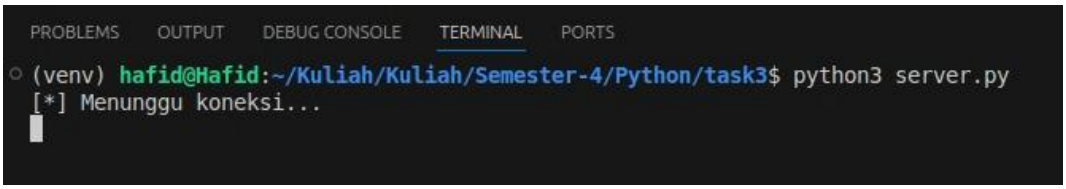
Membuat file client.py

```
#!/usr/bin/env python3
import socket
import os
import sys
import time

def print_progress_bar(sent, total, bar_length=40):
    percent = sent / total
    bar = '' * int(bar_length * percent) + '-' * (bar_length - int(bar_length *
percent))
    print(f"\rMengirim: |{bar}| {percent:.0%}", end='')

server_ip = input("Masukkan IP server: ")
server_port = 5001
filename = input("Masukkan nama file yang ingin dikirim: ")

if not os.path.exists(filename):
    print("File tidak ditemukan!")
```

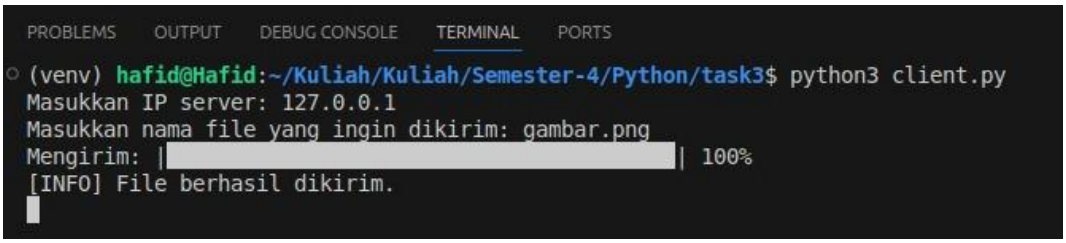
```
    sys.exit(1)

filesize = os.path.getsize(filename)
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((server_ip, server_port))

client.send(filename.encode())

sent_bytes = 0
buffer_size = 4096

with open(filename, "rb") as f:
    while True:
        bytes_read = f.read(buffer_size)
        if not bytes_read:
            break
        client.sendall(bytes_read)
        sent_bytes += len(bytes_read)
        print_progress_bar(sent_bytes, filesize)
        time.sleep(0.01)  # biar tampilan smooth

print("\n[INFO] File berhasil dikirim.")

# Terima konfirmasi dari server
response = client.recv(1024).decode()
print(f"Server: {response}")

client.close()
```

3. **Output program**

   a. **Server dijalankan**



   b. **Client dijalankkan**



4. **Penjelasan dan Analisa**

   a. **Program Server (server.py)**

   Server dijalankan dengan mendengarkan koneksi dari client pada IP 0.0.0.0 dan port 5001. Saat client terhubung, server menerima nama file, kemudian membuka file baru dan menulis data byte yang dikirim oleh client ke dalam direktori uploads.

   Poin penting:

   - Folder uploads otomatis dibuat untuk menyimpan file hasil kiriman.
   - File disimpan dengan nama received_namafile.
   - Menggunakan loop untuk menerima data secara bertahap (4096 byte).
   - Memberikan umpan balik ke client setelah file berhasil disimpan.

   b. **Program Client (client.py)**

   Client meminta input IP server dan nama file yang ingin dikirim. Jika file ditemukan, program akan membaca isinya dan mengirimkannya dalam potongan (chunk) berukuran 4096 byte. Saat pengiriman, client menampilkan progress bar sebagai indikator.

Poin penting:

- Memverifikasi keberadaan file terlebih dahulu.
- Menggunakan time.sleep(0.01) untuk membuat progress bar tampak lebih smooth.
- Menerima balasan dari server setelah proses pengiriman selesai.

c. Analisa Output

- **Saat server aktif, ia mencetak status menunggu koneksi.**
- **Saat client mengirim file, nama file diterima dan ditampilkan di sisi server.**
- **Progress bar di sisi client memperlihatkan status pengiriman data secara real time.**
- **Server mencetak status berhasil menerima file.**
- **Client menerima pesan konfirmasi dari server.**

Proses ini menunjukkan komunikasi client-server berhasil, pengiriman file lancar, dan tidak ada data yang hilang selama transmisi. Fungsi recv() dan sendall() berhasil menjalankan tugas masing-masing secara efektif.

BAB III
PENUTUP

A. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa socket TCP dapat digunakan untuk membangun sistem pengiriman file sederhana antara client dan server. Dengan pendekatan ini, mahasiswa dapat memahami dasar komunikasi jaringan, pengelolaan file, serta konsep buffer dalam pengiriman data. Program berhasil menunjukkan proses pengiriman file yang stabil, efisien, dan terkontrol.