CII2J4 JARINGAN KOMPUTER

LAPORAN TUGAS BESAR

KELAS : IF-45-10

DOSEN : FFT

ANGGOTA KELOMPOK

Endri Rizki Nugraha (1391210293)

Rio Ferdinan Putra Pratama (1301213367)

Zaidan Ibnusabil Iryanto (1301213501)

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2023

Latar Belakang

Jaringan komputer merupakan sebuah sistem yang terdiri dari beberapa perangkat seperti komputer, server, router, switch, dan perangkat jaringan lainnya yang saling terhubung satu sama lain melalui kabel atau jaringan nirkabel. Latar belakang jaringan komputer berkaitan dengan perkembangan teknologi informasi yang memungkinkan komputer untuk terhubung satu sama lain dan berkomunikasi secara efektif.

Sejarah jaringan komputer dimulai pada tahun 1960-an ketika Department of Defense Amerika Serikat mengembangkan jaringan komputer yang disebut ARPANET. Tujuan dari ARPANET adalah untuk memungkinkan berbagai institusi dan lembaga pemerintah di seluruh dunia untuk berbagi informasi dan sumber daya komputasi.

Pada tahun 1980-an, protokol TCP/IP dikembangkan dan diadopsi sebagai standar untuk jaringan komputer. Hal ini memungkinkan komputer yang berbeda-beda jenis dan platform untuk terhubung dan berkomunikasi secara efektif.

Dalam beberapa tahun terakhir, perkembangan teknologi jaringan komputer telah mengalami kemajuan yang pesat. Terdapat beberapa jenis jaringan seperti LAN (Local Area Network), WAN (Wide Area Network), MAN (Metropolitan Area Network), dan WLAN (Wireless Local Area Network) yang memungkinkan perangkat untuk terhubung satu sama lain di berbagai lokasi dengan cara yang berbeda.

Dengan adanya jaringan komputer, banyak organisasi dan perusahaan dapat melakukan pekerjaan secara efisien dan efektif. Jaringan komputer juga memungkinkan orang untuk terhubung satu sama lain di seluruh dunia dan berkomunikasi dengan mudah dan cepat. Oleh karena itu, pemahaman tentang latar belakang jaringan komputer sangat penting dalam memahami bagaimana teknologi informasi berkembang dan berperan dalam kehidupan sehari-hari.

Batasan masalah

1. Implementasi pembuatan TCP socket dan mengaitkannya ke alamat dan port tertentu
2. Program Web Server dapat menerima dan memparsing HTTP request yang dikirimkan oleh browser
3. Web Server dapat mencari dan mengambil file dari sistem yang diminta oleh client
4. Web Server dapat membuat HTTP response message yang terdiri dari header dan konten file yang diminta
5. Web server dapat mengirimkan response message yang sudah dibuat ke browser (client) dan dapat ditampilkan dengan benar di sisi client
6. Jika file yang diminta oleh client tidak tersedia, web server dapat mengirimkan pesan “404 Not Found” dan dapat ditampilkan dengan benar di sisi client

Sistem yang dibangun

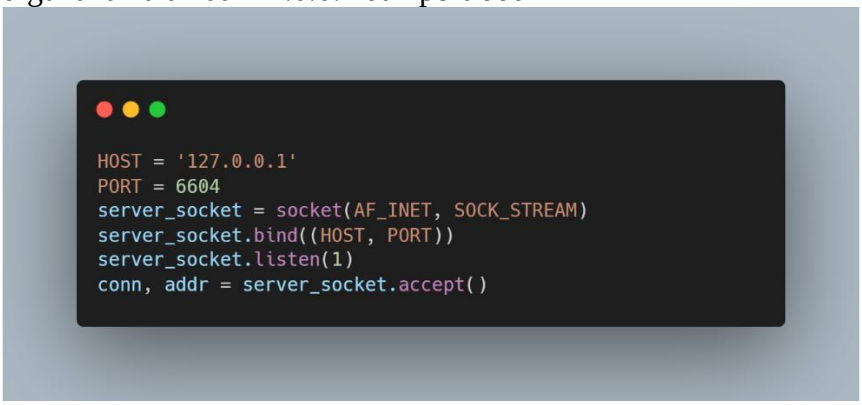
Hasil Program

```python
from socket import *
import os

#Alamat Server
HOST = '127.0.0.1'

#Port Server
PORT = 6604

#base directory folder
DIRECTORY = os.path.dirname(os.path.abspath(__file__))

#fungsi untuk membaca isi file dalam bentuk binary
def read_file(filepath) :
    file = open(filepath, 'rb')
    content = file.read()
    return content

#http response
def http_response(request):
    response = '' #inisisasi response
    filename = request.split()[1][1:]   #mengambil filename dari request
dengan metode split
    filepath = os.path.join(DIRECTORY, filename)    #menentukan filepath
dengan menggabungkan base directory dengan filename menggunakan metode join
    if os.path.isfile(filepath):    #pengkondisian jika file ditemukan
        content = read_file(filepath)
        response = f"HTTP/1.1 200 OK\r\nContent-Length:
{len(content)}\r\n\r\n".encode('utf-8')
        response += content
    else:   #pengkondisian jika file tidak ditemukan
        response = "HTTP/1.1 404 Not Found\r\nContent-Type: text/html;
charset=utf-8\r\n\r\n".encode('utf-8')
        response += "<h1>404 Not Found</h1>".encode('utf-8')
    return response


#Buat TCP Scoket
server_socket = socket(AF_INET, SOCK_STREAM)

#bind alamat dan port tertentu
server_socket.bind((HOST, PORT))

#Tunggu Koneksi masuk
server_socket.listen(1)

#Pesan konfirmasi bahwa server telah berjalan
print(f"Server berjalan di {HOST} port {PORT} (http://{HOST}:{PORT})......")

while True :
    #Terima koneksi server
    conn, addr = server_socket.accept()

    #Baca data yang dikirimkan client
    request = conn.recv(1024).decode()

    #memanggil fungsi http response untuk menampilkan http respons
    response = http_response(request)

    #print method path dan protocol yang terdapat pada variabel request
    method, path, protocol = request.split('\n')[0].split()
    print("FILE BERHASIL DIBUKA !!")
    print('Method:', method)
    filepath = os.path.join(DIRECTORY, request.split()[1][1:])
    print('Path:', filepath)
    print('Protocol:', protocol)

    conn.sendall(response)  #kirim response ke client
    conn.close()    #tutup koneksi
```

Analisis

Untuk membuat Web Server yang dapat menerima dan membuka file dibutuhkan beberapa langkah sebagai berikut :

1. Membuat Socket TCP dan menghubungkannya ke alamat dan port tertentu disini digunakan alamat 127.0.0.1 dan port 6604

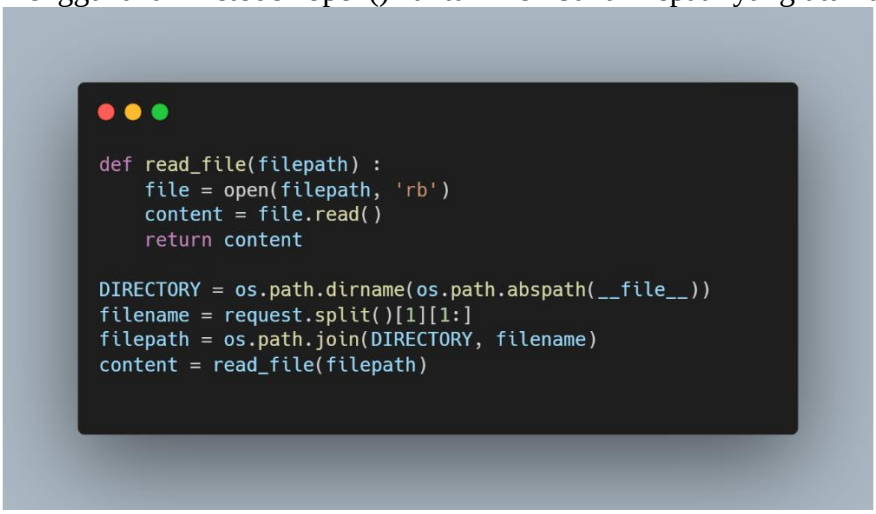


```
HOST = '127.0.0.1'
PORT = 6604
server_socket = socket(AF_INET, SOCK_STREAM)
server_socket.bind((HOST, PORT))
server_socket.listen(1)
conn, addr = server_socket.accept()
```

2. Menerima dan parsing request yang dikirimkan oleh browser dengan metode ".accept()" untuk menerima dan "conn.recv(1024).decode()" untuk parsing HTTP request

```
conn, addr = server_socket.accept()
request = conn.recv(1024).decode()
response = http_response(request)
```

3. Mencari dan mengambil file dari sistem dengan metode "os.path.dirname(os.path.abspath(__file__))" untuk menentukan direktori file .py saat ini, "request.split()[1][1:]" untuk mengambil nama file dari header HTTP, "os.path.join(DIRECTORY, filename)" untuk menggabungkan direktori file .py dan nama file yang akan dibuka sehingga menjadi suatu filepath yang utuh, lalu menggunakan metode "open()" untuk membuka filepath yang utuh tersebut.



```
def read_file(filepath) :
    file = open(filepath, 'rb')
    content = file.read()
    return content

DIRECTORY = os.path.dirname(os.path.abspath(__file__))
filename = request.split()[1][1:]
filepath = os.path.join(DIRECTORY, filename)
content = read_file(filepath)
```

4. Membuat pengkondisian jika file tersebut ada dan benar, maka Web server akan mengirimkan HTTP response message “HTTP/1.1 200 OK” dan konten file yang diminta.

```
if os.path.isfile(filepath):    #pengkondisian jika file ditemukan
        content = read_file(filepath)
        response = f"HTTP/1.1 200 OK\r\nContent-Length:
{len(content)}\r\n\r\n".encode('utf-8')
        response += content
```

5. Web server akan mengirimkan response message yang sudah dibuat ke browser(client) .

```
    request = conn.recv(1024).decode()
    response = http_response(request)
    method, path, protocol = request.split('\n')[0].split()
    print("FILE BERHASIL DIBUKA !!")
    print('Method:', method)
    filepath = os.path.join(DIRECTORY, request.split()[1][1:])
    print('Path:', filepath)
    print('Protocol:', protocol)
```

6. Membuat pengkondisian, jika file yang diminta oleh client tidak tersedia, web server dapat mengirimkan pesan “404 Not Found” dan dapat ditampilkan dengan benar di sisi client.

```
else:   #pengkondisian jika file tidak ditemukan
        response = "HTTP/1.1 404 Not Found\r\nContent-Type: text/html;
charset=utf-8\r\n\r\n".encode('utf-8')
        response += "<h1>404 Not Found</h1>".encode('utf-8')
```

Kesimpulan

Dari analisis yang sudah dibuat maka dibutuhkan beberapa langkah untuk membuat Web Server dengan spesifikasi yang tertera untuk tugas besa jaringan komputer kali ini, langkah langkah tersebut adalah