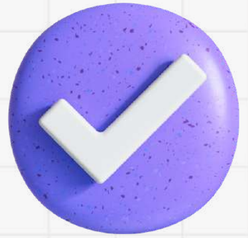
Jaringan Komputer

TUGAS Besar



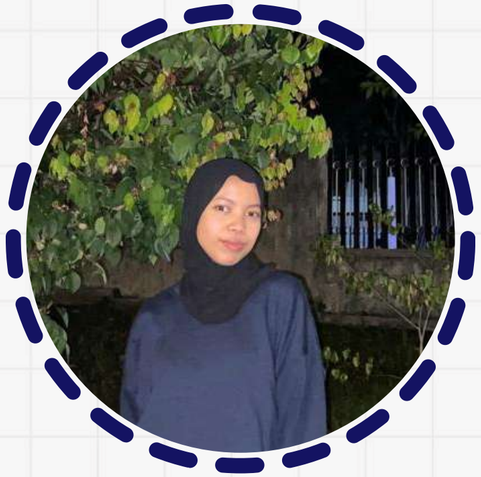
Membuat Web Server Berbasis TCP dengan menerapkan Socket Programming

IF - 46-08 // UNIVERSITAS TELKOM

Anggota Kelompok

Anak Agung Sagung Putri W

1301223079



Adinda Laras Sri Rahtami

1301223253

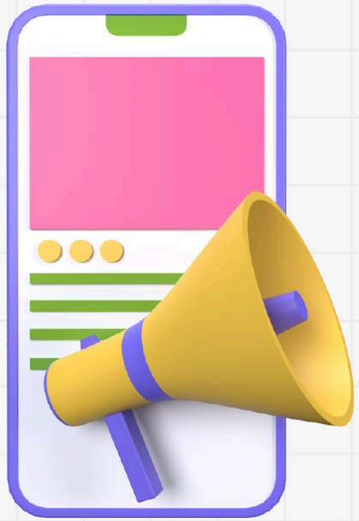
Latar Belakang

- Perkembangan teknologi informasi meningkatkan kebutuhan akan akses cepat dan efisien terhadap informasi.
- Pengembangan web server yang efektif dan efisien penting untuk memastikan permintaan pengguna dilayani dengan cepat dan tepat.
- Keberadaan web server adalah keharusan dalam era internet saat ini, menyediakan akses ke berbagai informasi dan layanan.
- Web server mampu menangani beberapa request HTTP secara simultan, yang penting untuk menjaga performa dan stabilitas, terutama saat menghadapi traffic tinggi.

Rumusan Masalah

Bagaimana merancang dan mengimplementasikan web server yang dapat menangani satu permintaan HTTP pada satu waktu dengan efektif?

Bagaimana cara mengimplementasikan multithreading pada web server agar bisa melayani banyak permintaan HTTP sekaligus?

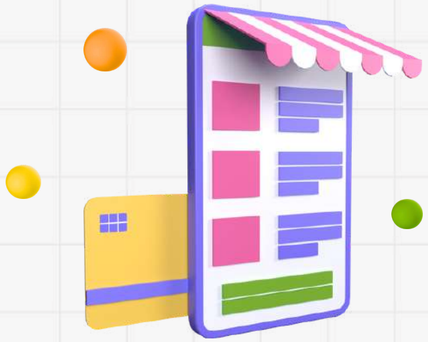


Tujuan

Mengembangkan web server yang mampu menangani satu permintaan HTTP pada satu waktu.

Membuat web server multithread yang mampu melayani beberapa permintaan secara simultan

DASAR TEORI



TCP

Standar komunikasi data dalam jaringan komputer yang digunakan untuk pertukaran data antar perangkat.

Socket Programming

Metode pemrograman yang menggunakan soket, yang berfungsi sebagai terowongan untuk komunikasi dua arah

HTTP

Protokol jaringan pada lapisan aplikasi yang dikembangkan untuk memfasilitasi transfer data antar komputer

HASIL PROGRAM & ANALISIS

Kami membuat 2 file yaitu server.py dan client.py

Server.py

1

```python
import socket
import threading
import os
import time
```

Menunjukkan penggunaan modul socket, threading, os, dan time

2

```python
def handle_client(client_socket, addr):
    try:
        start_time = time.time()

        request = client_socket.recv(1024).decode('utf-8')
        print(f"Received request:\n{request}")

        headers = request.split('\n')
        filename = headers[0].split()[1]

```

Kode ini untuk menangani komunikasi dengan klien yang terhubung ke server. Menerima permintaan klien, lalu mengekstrak nama file yang diminta dari header

Server.py

3

```
try:
            f = open(filename[1:])
            response_body = f.read()

            response_header = 'HTTP/1.1 200 OK\r\n\r\n'
            client_socket.send(response_header.encode('utf-8'))
            for i in range(0, len(response_body)):
                client_socket.send(response_body[i].encode('utf-8'))
            client_socket.send('\r\n'.encode('utf-8'))

            end_time = time.time()
            print(f"Request from {addr} processed in {end_time -
start_time:.5f} seconds.")
```

Kode ini menangani pengiriman respon ke klien Terdapat kondisi untuk menangani kemungkinan kesalahan. Jika file berhasil dibuka, kode merespon dengan kode status 200 (OK)

4

```
except FileNotFoundError:
            response_header = 'HTTP/1.1 404 Not Found\r\n\r\n'

            response_body =
'<html>\r\n<body>\r\n    <h1>404 Not Found</h1>\r\n</body>\r\n</html>'

            client_socket.send(response_header.encode('utf-8'))

            for i in range(0, len(response_body)):
                client_socket.send(response_body[i].encode('utf-8'))
            client_socket.send('\r\n'.encode('utf-8'))

            end_time = time.time()
            print(f"Request from {addr}
failed: File not found. Processed in {end_time - start_time:.5f}
seconds.")
        finally:
            client_socket.close()
```

Kode ini menangani ketika file yang diminta tidak ditemukan. Ketika file yang diminta tidak ditemukan, maka akan mengirimkan respons 404 Not Found.

Server.py

5

```python
serverAddress = '127.0.0.1'
serverPort = 8000
serverSocket =
 socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.bind((serverAddress, serverPort))
serverSocket.listen(1)
print("Server is listening on port 8000...")
```

Kode ini mengatur persiapan server untuk menerima koneksi klien baru.

6

```python
while True:
    client_socket, addr = serverSocket.accept()
    print(f"Accepted connection from {addr}")

    client_handler = threading.Thread(target=
handle_client, args=(client_socket, addr))
    client_handler.start()

```

Kode ini berfungsi untuk menambahkan loop utama server yang menangani koneksi masuk dan membuat thread baru setiap koneksi klien

Client.py

```python
import socket
import sys

```

Potongan codingan di atas menunjukkan penggunaan modul socket dan sys.

```python
def send_request(host, port, filename):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        client_socket.connect((host, port))

        request_line = f"GET {filename} HTTP/1.1\r\n"
        headers = f"Host: {host}\r\nConnection: keep-alive\r\n\r\n"
        request = request_line + headers

        client_socket.sendall(request.encode('utf-8'))

        response = b""
```

Kode ini bertujuan untuk mengirimkan permintaan HTTP GET ke server tertentu dan menangani respons dari server.

Client.py

```
while True:
            chunk = client_socket.recv(4096)
            if not chunk:
                break
            response += chunk

        print(response.decode('utf-8'))
    finally:
        client_socket.close()

```

Kode ini menerima respons dari server, Loop dilakukan hingga tidak ada data di server, Dan server diubah menjadi string

```

if len(sys.argv) != 4:
    print(
"Usage: python client.py <server host> <server port> <file name>"
)
    sys.exit(1)
```

Kode tersebut memeriksa apakah jumlah argumen yang diberikan sesuai dengan yang diharapkan (yaitu 4 argumen)

Client.py

```
1  host = sys.argv[1]
2  port = int(sys.argv[2])
3  filename = "/" + sys.argv[3]
4
5  send_request(host, port, filename)
6
```

Kode ini mengambil argumen pertama dari baris perintah saat dijalankan dan menyimpannya dalam variabel host. Lalu memanggil fungsi untuk mengirimkan permintaan HTTP ke server

Hasil


```
Accepted connection from ('127.0.0.1', 62498)
Received request:
GET /Tubesjarkomadingepe HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive


Request from ('127.0.0.1', 62498) failed: File not found. Processed in 0.00040 seconds.
```

```
PS D:\KULIAH\SEMESTER 4\JARKOM\TUBES\Tubesjarkomadingepe> python client.py 127.0.0.1 8000 ac
umalaka.html
HTTP/1.1 404 Not Found

<html>
<body>
    <h1>404 Not Found</h1>
</body>
</html>
```

Hasil

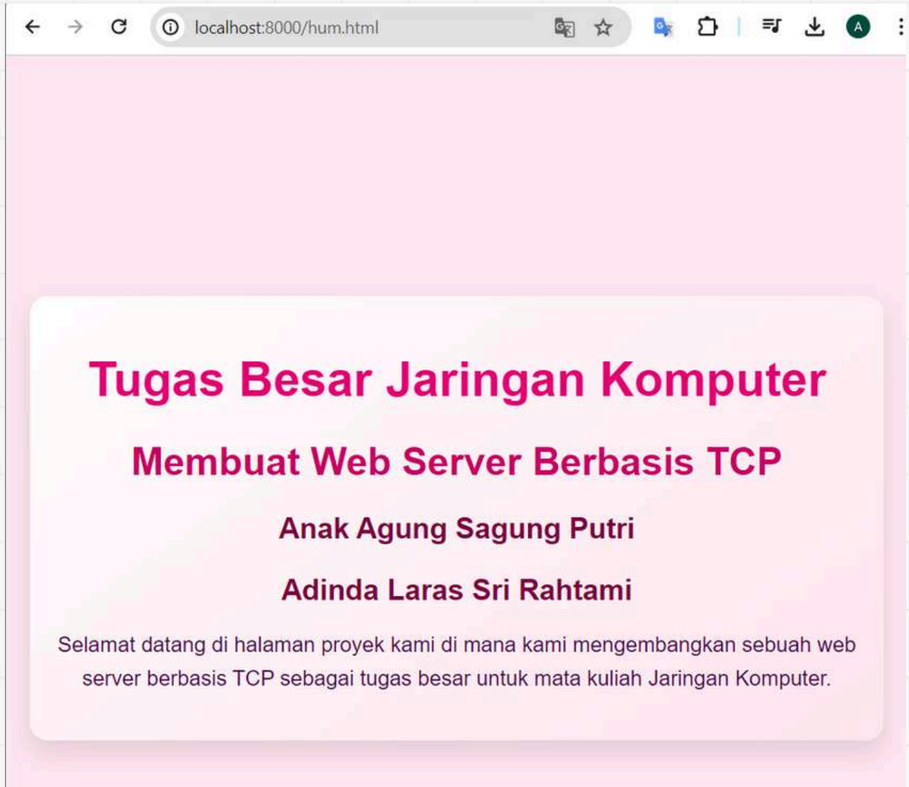
```
History restored

PS D:\KULIAH\SEMESTER 4\JARKOM\TUBES\Tubesjarkomadingepe> python server.py 127.0.0.1 8000 hum.h
tml
Server is listening on port 8000...
Accepted connection from ('127.0.0.1', 62375)
Received request:
GET /hum.html HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive


Request from ('127.0.0.1', 62375) processed in 0.02638 seconds.
```

```
PS D:\KULIAH\SEMESTER 4\JARKOM\TUBES\Tubesjarkomadingepe> python client.py 127.0.0.1 8000 hum.
html
HTTP/1.1 200 OK

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jarkom - Tugas Besar</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #ffe6f0;
            color: #333;
            margin: 0;
            padding: 20px;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .container {
            max-width: 800px;
            padding: 20px;
            background: linear-gradient(145deg, #fff, #ffe6f0);
            box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
            border-radius: 15px;
            text-align: center;
            transition: transform 0.3s ease;
        }
        .container:hover {
            transform: scale(1.05);
        }
        h1 {
```



Kesimpulan

Pada tugas besar jaringan komputer ini kami berhasil membuat Web Server Berbasis TCP dengan menerapkan Socket Programming.
Keseluruhan kode untuk tugas besar ini memungkinkan pembuatan web server berbasis TCP yang dapat menangani permintaan HTTP secara efektif dan melayani beberapa permintaan secara bersamaan dengan menggunakan multithreading.

TERIMA KASIH