

TUGAS BESAR JARINGAN KOMPUTER

Fathan Arya Maulana / 103012300083

Dzaky Alfaris / 103012300391

M. Rifqi Dzaky Azhad / 103012330009

PENDAHULUAN

Latar Belakang

Dalam era digital saat ini, komunikasi antar perangkat dalam jaringan komputer menjadi sangat penting, terutama melalui protokol HTTP yang digunakan pada layanan web. Untuk memahami cara kerja dasar dari web server dan klien, mahasiswa perlu mempelajari dan mengimplementasikan komunikasi berbasis socket programming dengan protokol TCP.

Rumusan Masalah

Membangun web server berbasis TCP yang dapat menerima dan memproses request HTTP dari klien.

Mengimplementasikan server yang mampu menangani banyak permintaan secara simultan dengan menggunakan multithreading.

Tujuan

Mengembangkan web server sederhana berbasis TCP yang dapat menangani request HTTP. Mengimplementasikan multithreading pada server untuk melayani beberapa klien secara bersamaan. Meningkatkan pemahaman tentang komunikasi jaringan dan pemrograman socket.

PENDEKATAN

Pendekatan yang digunakan dalam tugas ini dimulai dengan membangun server web sederhana berbasis TCP menggunakan socket programming. Server dikembangkan untuk menerima dan memproses permintaan HTTP metode GET, mengambil file dari direktori lokal, dan mengirimkan respons sesuai format HTTP.

Setelah server dasar berfungsi, pendekatan berikutnya adalah mengimplementasikan fitur multithreading. Setiap koneksi dari klien akan dilayani oleh thread terpisah agar server dapat menangani beberapa request secara bersamaan. Pengujian dilakukan melalui browser dan klien mandiri yang dibuat menggunakan bahasa pemrograman Python.

SERVER SINGLE

```
● ● ●  
1 from socket import *  
2  
3 SERVER_HOST = '127.0.0.1'  
4 SERVER_PORT = 4321  
5  
6 serverSocket = socket(AF_INET, SOCK_STREAM)  
7 serverSocket.bind((SERVER_HOST, SERVER_PORT))  
8 serverSocket.listen(1)
```

- Code server single diawali dengan import library socket dengan seluruh method dan atributnya.
- Dilanjutkan dengan inisialisasi SERVER_HOST dan SERVER_PORT.
- serverSocket dibentuk dan melakukan bind dan listen dengan nilai 1.

```
● ● ●  
1 def handle_client(connectionSocket, addr):  
2     print("Terhubung dengan ", addr)  
3  
4     try:  
5         while True:  
6             message = connectionSocket.recv(1024).decode().strip()  
7             if message == 'exit':  
8                 print("Koneksi client tertutup")  
9                 break  
10  
11            try:  
12                filename = message.split()[1][1:]  
13                with open(filename, encoding='utf-8') as f:  
14                    content = f.read()  
15                    response = "HTTP/1.1 200 OK\r\n\r\n" + content  
16  
17                except FileNotFoundError:  
18                    response = "HTTP/1.1 404 Not Found\r\n\r\n"  
19  
20                connectionSocket.send(response.encode())  
21  
22            except Exception as e:  
23                print(f"Error: {e}")  
24  
25        finally:  
            connectionSocket.close()  
            print("Koneksi tertutup")
```

- Mendefinisikan prosedur 'handle_client' dengan parameter socket dan address user
- Prosedur ini melakukan loop untuk menerima pesan dari client dan mengembalikan file yang direquest,
- Terdapat penanganan error 404 apabila tidak ada file yang diminta.

```
● ● ●  
1 while True:  
2     connectionSocket, addr = serverSocket.accept()  
3     handle_client(connectionSocket, addr)
```

- Akhir code adalah loop untuk menerima connectionSocket dan alamat client dan menjalankan 'handle_client'.

SERVER MULTI

```
● ● ●  
1 from socket import *  
2  
3 SERVER_HOST = '127.0.0.1'  
4 SERVER_PORT = 4321  
5  
6 serverSocket = socket(AF_INET, SOCK_STREAM)  
7 serverSocket.bind((SERVER_HOST, SERVER_PORT))  
8 serverSocket.listen(1)
```

```
● ● ●  
1 def handle_client(connectionSocket, addr):  
2     print("Terhubung dengan ", addr)  
3  
4     try:  
5         while True:  
6             message = connectionSocket.recv(1024).decode().strip()  
7             if message == 'exit':  
8                 print("Koneksi client tertutup")  
9                 break  
10            try:  
11                filename = message.split()[1][1:]  
12                with open(filename, encoding='utf-8') as f:  
13                    content = f.read()  
14                    response = "HTTP/1.1 200 OK\r\n\r\n" + content  
15                except FileNotFoundError:  
16                    response = "HTTP/1.1 404 Not Found\r\n\r\n"  
17                connectionSocket.send(response.encode())  
18            except Exception as e:  
19                print(f"Error: {e}")  
20            finally:  
21                connectionSocket.close()  
22                print("Koneksi tertutup")
```

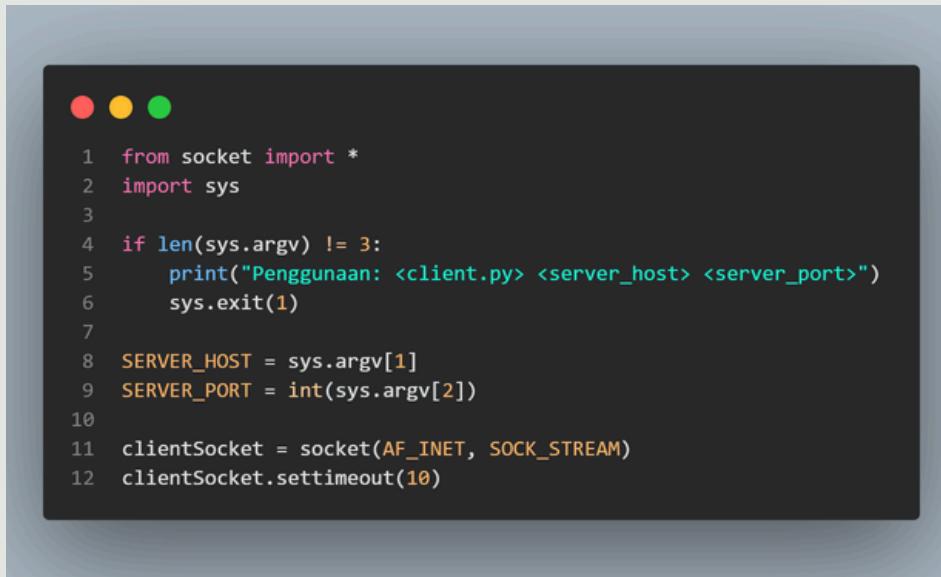
```
● ● ●  
1 while True:  
2     connectionSocket, addr = serverSocket.accept()  
3     thread = threading.Thread(target=handle_client, args=(connectionSocket, addr))  
4     thread.start()
```

- Import library socket dan threading.
- Dilakukan inisialisasi SERVER_HOST dan SERVER_PORT.
- Objek serverSocket dibentuk menggunakan socket(AF_INET, SOCK_STREAM).
- Dilakukan proses bind dan listen dengan nilai 5.

- Dilanjutkan dengan mendefinisikan prosedur 'handle_client' dengan parameter socket dan address user.
- Prosedur ini mirip dengan server single, yaitu dengan melakukan loop untuk menerima pesan dari client dan mengembalikan file yang direquest, terdapat penanganan error 404 apabila tidak ada file yang diminta.

- Bagian akhir kode menggunakan loop while True untuk menerima koneksi klien.
- Setiap koneksi diproses dalam thread baru dengan memanggil fungsi handle_client.

CLIENT



```
● ● ●
1 from socket import *
2 import sys
3
4 if len(sys.argv) != 3:
5     print("Penggunaan: <client.py> <server_host> <server_port>")
6     sys.exit(1)
7
8 SERVER_HOST = sys.argv[1]
9 SERVER_PORT = int(sys.argv[2])
10
11 clientSocket = socket(AF_INET, SOCK_STREAM)
12 clientSocket.settimeout(10)
```

- Import seluruh fungsi dari modul socket dan sys.
- Program memeriksa apakah argumen yang dimasukkan dari command line berjumlah dua (alamat host dan port server).
- Nilai SERVER_HOST dan SERVER_PORT diambil dari argumen tersebut. Selanjutnya,
- Socket TCP dibuat dengan socket(AF_INET, SOCK_STREAM)
- diberi timeout selama 10 detik untuk mencegah program menunggu respons terlalu lama dari server.



```
● ● ●
1 try:
2     clientSocket.connect((SERVER_HOST, SERVER_PORT))
3     print("Terhubung dengan Server")
4
5     while True:
6         filename = input("Isi nama file (atau 'exit'): ")
7
8         if not filename:
9             continue
10        if filename == "exit":
11            break
12
13        request = f"GET /{filename} HTTP/1.0\r\nHost: {SERVER_HOST}\r\n\r\n"
14        clientSocket.send(request.encode())
15
16        response = ""
17        while True:
18            try:
19                chunk = clientSocket.recv(4096).decode()
20                if not chunk:
21                    break
22                response += chunk
23            except timeout:
24                break
25
26        print(response)
27
28 except Exception as e:
29     print(f"Error: {e}")
30
31 finally:
32     clientSocket.close()
```

- Menghubungkan klien ke server menggunakan connect(). Jika berhasil, pengguna dapat memasukkan nama file yang ingin diminta.
- Permintaan dikirim dalam format HTTP GET.
- Data diterima secara bertahap (chunk) menggunakan recv(), sampai tidak ada data lagi.
- Jika terjadi kesalahan selama koneksi, program akan menangkap dan mencetak pesan error.

RUNNING

Server single

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python server_single.py  
Terhubung dengan ('127.0.0.1', 56601)
```

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python client.py 127.0.0.1 4321  
Terhubung dengan Server  
Isi nama file (atau 'exit'): 1.html  
HTTP/1.1 200 OK  
  
<div class="container">  
    <h1 class="flash-success">SELAMAT WEB SERVER ANDA BISA JALAN 🎉😊</h1>  
    <p>- ST0</p>  
    <div class="emoji">☺</div>  
</div>  
;  
</script>  
</body>  
</html>  
Isi nama file (atau 'exit'): █
```

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python client.py 127.0.0.1 4321  
Terhubung dengan Server  
Isi nama file (atau 'exit'): 1.html  
HTTP/1.1 200 OK  
  
<div class="container">  
    <h1 class="flash-success">SELAMAT WEB SERVER ANDA BISA JALAN 🎉😊</h1>  
    <p>- ST0</p>  
    <div class="emoji">☺</div>  
    </div>  
    ;  
</script>  
</body>  
</html>  
Isi nama file (atau 'exit'): █
```

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER (JARKOM)\[2] Tugas\Tugas Besar\Jaringan-Komputer-LAB_Tugas-Besar>python client.py 127.0.0.1 4321  
Terhubung dengan Server  
Isi nama file (atau 'exit'): 1.html  
Isi nama file (atau 'exit'): █
```

Server multi

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python server_multi.py  
Terhubung dengan ('127.0.0.1', 52940)  
Terhubung dengan ('127.0.0.1', 52941)
```

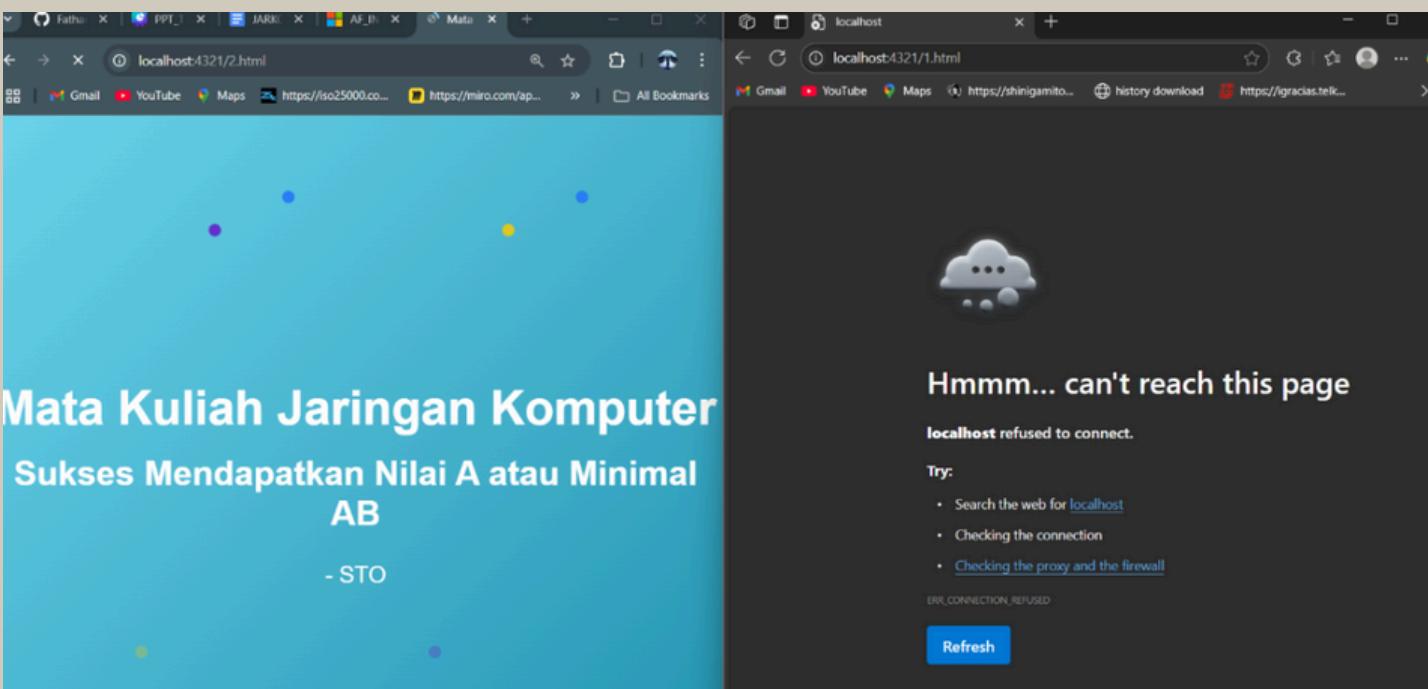
```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.0.0.1 1234  
Terhubung dengan Server  
Isi nama file (atau 'exit'): 1.html  
HTTP/1.1 200 OK  
  
<!DOCTYPE html>
```

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.0.0.1 1234  
Terhubung dengan Server  
Isi nama file (atau 'exit'): 1.html  
HTTP/1.1 200 OK  
  
<!DOCTYPE html>  
<html lang="id">
```

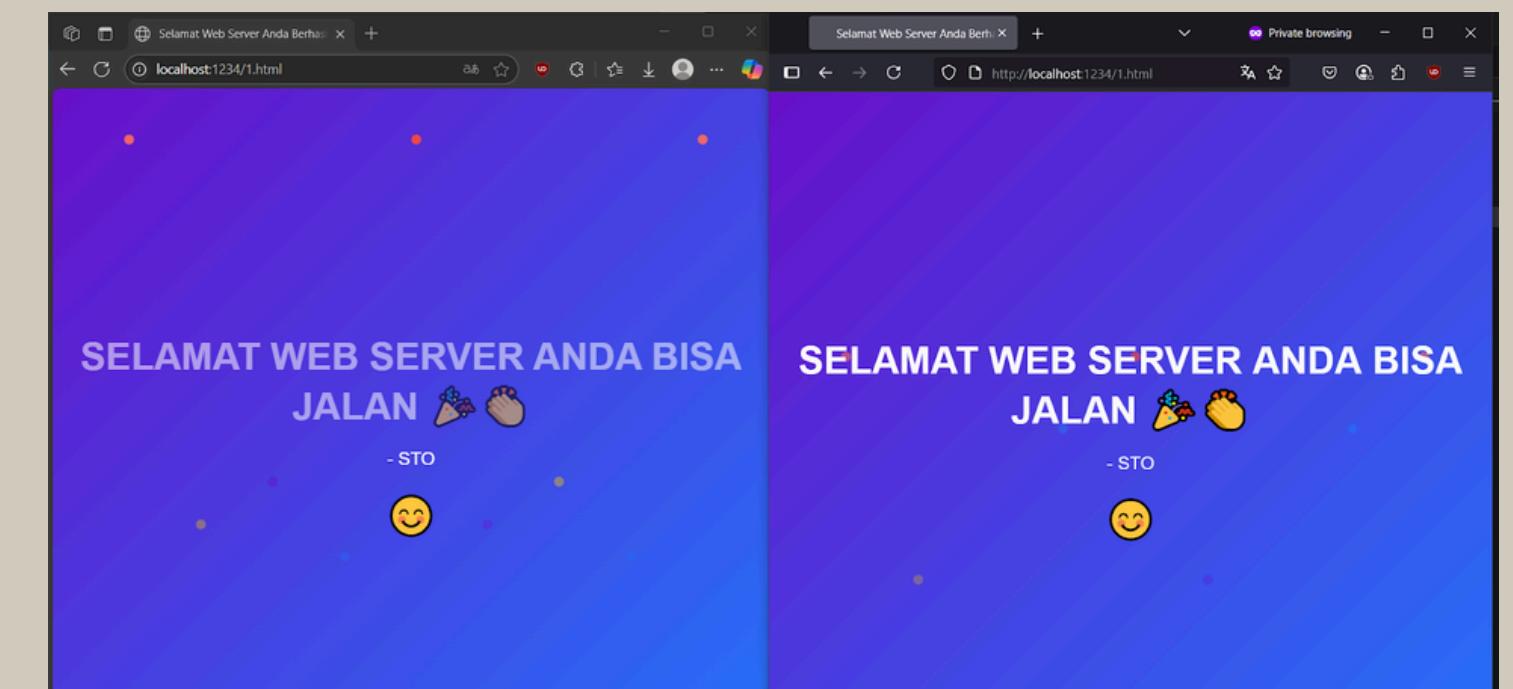
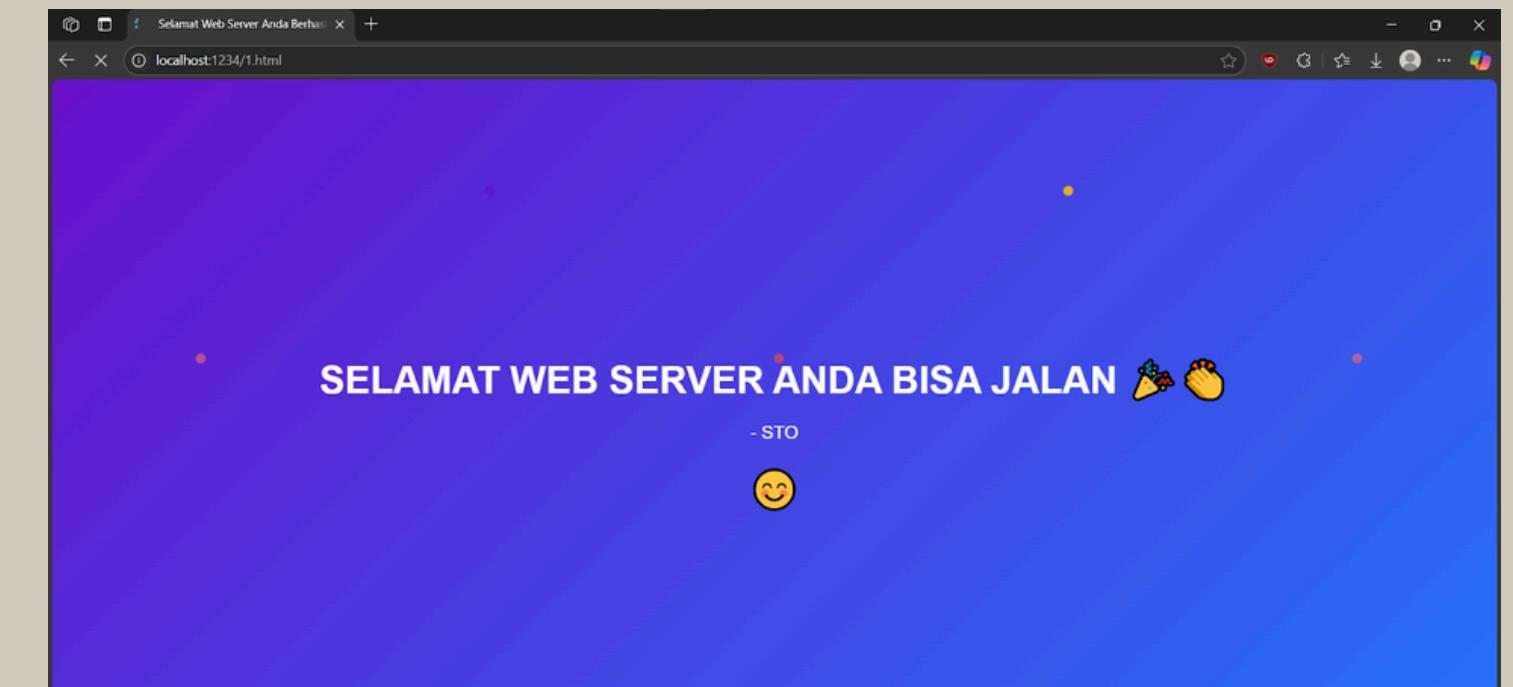
```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.0.0.1 1234  
Terhubung dengan Server  
Isi nama file (atau 'exit'): 1.html  
HTTP/1.1 200 OK  
  
<!DOCTYPE html>
```

RUNNING

Server single

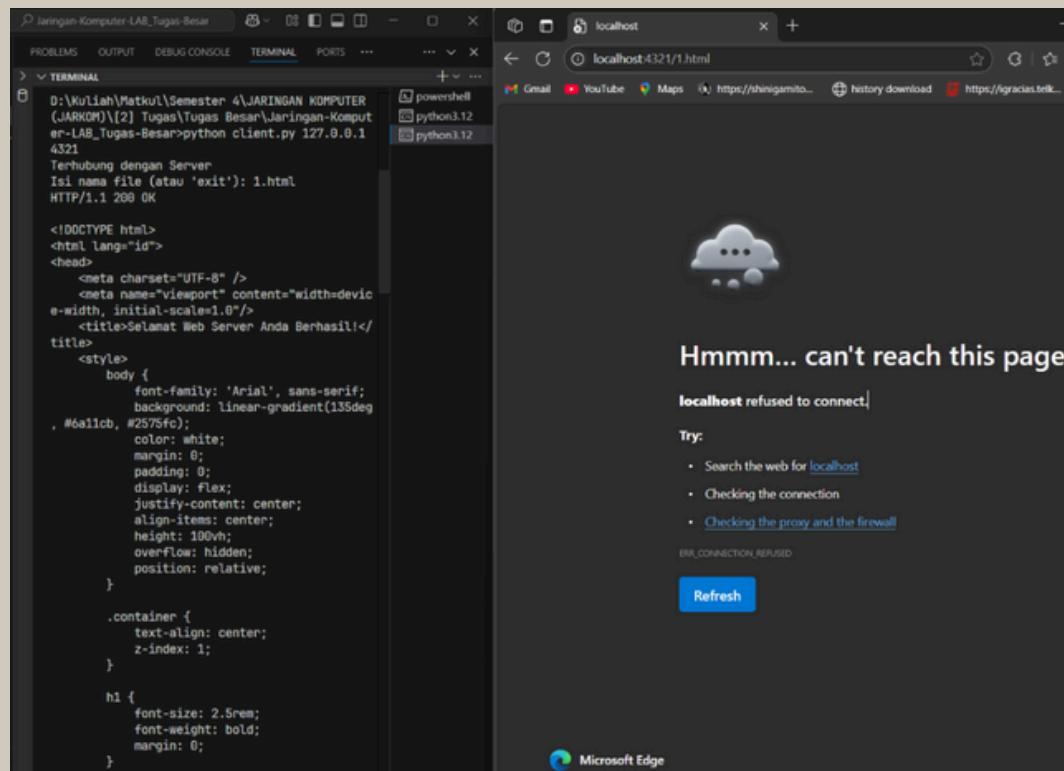


Server multi



RUNNING

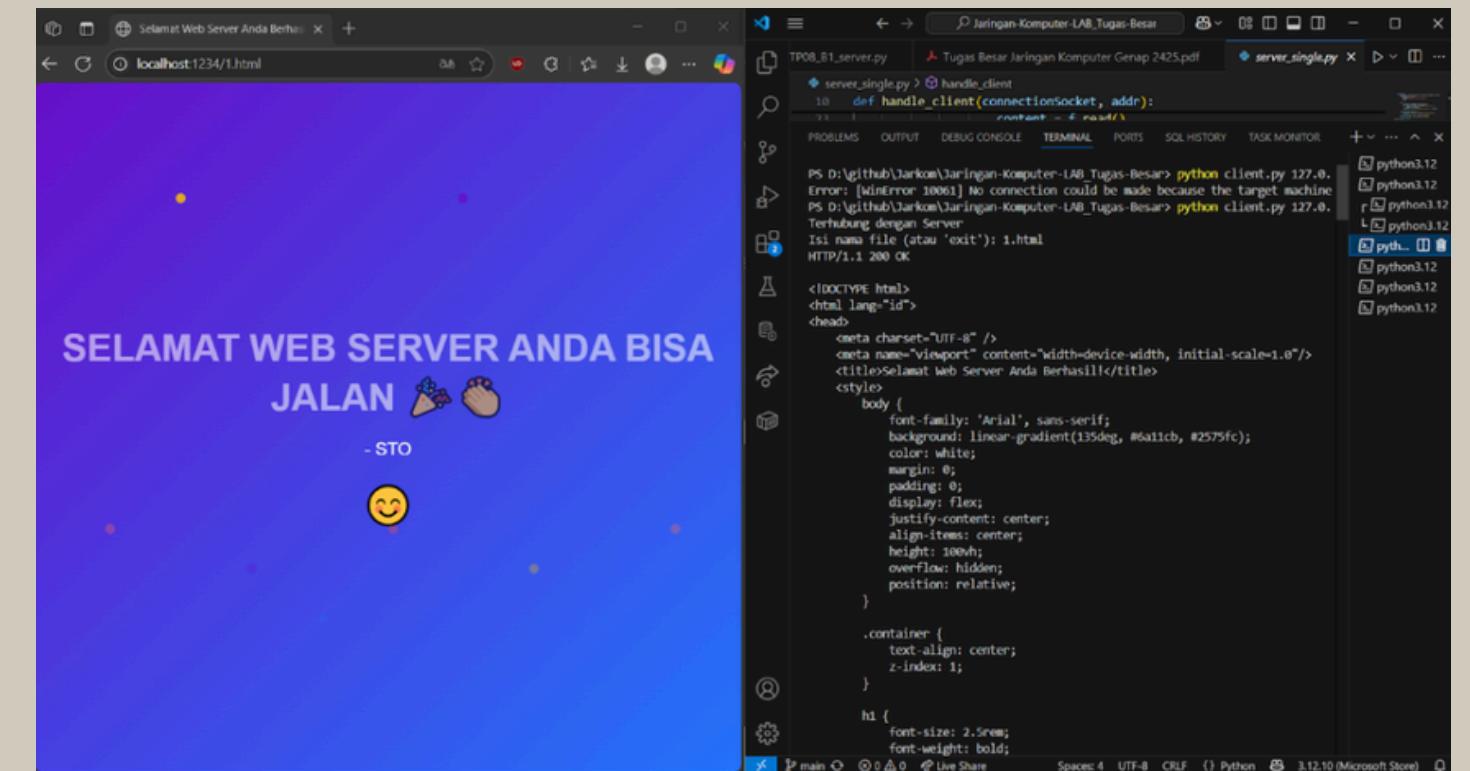
Server single



A screenshot of a Microsoft Edge browser window. The address bar shows 'localhost'. The page content is a 404 error page with a dark background and white text. It features a small cloud icon and the message 'Hmmm... can't reach this page'. Below this, it says 'localhost refused to connect' and provides some troubleshooting steps. A 'Refresh' button is at the bottom.

```
D:\Kuliah\Matkul\Semester 4\JARINGAN KOMPUTER>
Terhubung dengan Server
Isi nama file (atau 'exit'): 4.html
HTTP/1.1 404 Not Found
```

Server multi



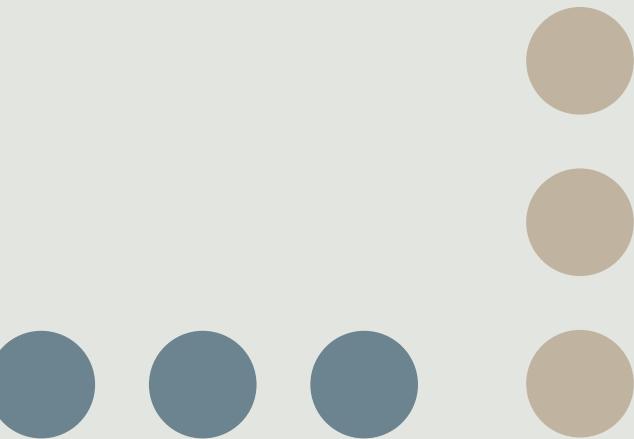
A screenshot of a Microsoft Edge browser window. The address bar shows 'localhost:1234/1.html'. The page content is a success message with a purple gradient background, featuring the text 'SELAMAT WEB SERVER ANDA BISA JALAN' and some small icons. The developer tools sidebar on the left shows the source code for 'server_single.py'.

```
PS D:\github\Jarkom\Jaringan-Komputer-LAB_Tugas-Besar> python client.py 127.0.
0.1 1234
Terhubung dengan Server
0.1 1234
Terhubung dengan Server
Terhubung dengan Server
Isi nama file (atau 'exit'): 10.html
HTTP/1.1 404 Not Found

Isi nama file (atau 'exit'): □
```

KESIMPULAN

Server single hanya mampu menangani satu klien dalam satu waktu, sedangkan server multi (threading) dapat melayani banyak klien secara bersamaan. Laporan ini mengeksplorasi perbedaan server dengan dan tanpa threading, serta mengetahui bahwa socket programming menggunakan protokol TCP dapat memenuhi kebutuhan GET, baik melalui Command Prompt maupun browser, serta mampu menangani permintaan file yang tidak ditemukan dengan mengembalikan response error 404 secara tepat.



Thank You

REFERENSI

Informatics Lab, School of Computing, Telkom University (2025). Modul Praktikum Jaringan Komputer S1 Informatika.

Kurose, J. F., & Ross, K. W. (2022). Computer networking: A top-down approach (8th ed.). Pearson Education.