# Laporan Pembuatan Program Web Server Sederhana Berbasis TCP Socket Programming


Disusun Oleh:
Kelompok Sebelas - IF 4505

1. Muhammad Fatih Yumna Lajuwirdi Lirrahmana - 1301213389
2. R. Adicondro Yusuf Hendratmo-1301213152

**Program Studi S1 Informatika**
**Fakultas Informatika**
**Universitas Telkom**
**2023**

# BAB I
# PENDAHULUAN

## A. LATAR BELAKANG

TCP (Transmission Control Protocol) Socket Programming adalah teknologi yang memungkinkan komunikasi antara dua perangkat melalui jaringan menggunakan protokol TCP. TCP merupakan salah satu protokol yang paling umum digunakan di internet untuk memastikan pengiriman data yang andal, terurut, dan tidak duplikat.

Dalam TCP Socket Programming, program komputer dibagi menjadi dua bagian, yaitu client dan server. Client bertanggung jawab untuk meminta data dari server, sedangkan server bertanggung jawab untuk memberikan data yang diminta oleh client. Kedua program tersebut saling berkomunikasi menggunakan alamat IP dan port.

Penerapan TCP Socket Programming sangat luas dan dapat digunakan untuk berbagai macam aplikasi, seperti transfer file, live streaming, game online, dan sebagainya. Namun, dalam implementasinya, terdapat beberapa tantangan seperti pengelolaan koneksi, pemrosesan data yang diterima, dan pengiriman respons ke client.

Meskipun demikian, TCP Socket Programming tetap menjadi teknologi yang populer dan penting dalam dunia komputer dan jaringan. Dengan memahami dasar-dasar TCP Socket Programming, seseorang dapat mengembangkan aplikasi jaringan yang dapat diandalkan dan efektif.

## B. BATASAN MASALAH

1. Terbatas pada penggunaan protokol HTTP saja. Kode ini hanya dapat menangani permintaan HTTP dan mengirimkan respon HTTP. Hal ini membatasi penggunaannya pada server web sederhana yang hanya mampu menangani permintaan HTTP.
2. Terbatas pada satu koneksi client saja. Kode ini hanya dapat menangani satu koneksi client pada satu waktu. Jika lebih dari satu koneksi client terhubung ke server secara bersamaan, server akan tidak responsif terhadap koneksi selanjutnya sampai koneksi saat ini ditutup.
3. Tidak ada manajemen proses dan performa yang baik. Kode ini tidak memperhatikan manajemen proses dan performa secara mendalam, seperti manajemen memori, manajemen sumber daya, dan lain-lain. Karena itu, kode ini tidak cocok untuk digunakan dalam lingkungan produksi atau skala besar.
4. Tidak ada perlindungan keamanan yang memadai. Kode ini tidak memiliki fitur keamanan yang memadai, seperti verifikasi pengguna, autentikasi, dan enkripsi. Karena itu, kode ini rentan terhadap serangan dan tidak cocok untuk digunakan dalam lingkungan yang membutuhkan keamanan tinggi.

## C. SISTEM YANG DIBANGUN

Sistem yang dibangun oleh Program Web Server Sederhana Berbasis TCP Socket Programming adalah sebagai berikut:

- Implementasi pembuatan TCP Socket dan mengaitkan ke alamat port tertentu

- Program web server dapat menerima dan memparsing HTTP request yang diminta oleh client
- Web Server dapat mencari dan mengambil file (dari file system) yang diminta oleh client
- Web server dapat membuat HTTP response message yang terdiri dari header dan konten file yang diminta
- Web server dapat mengirimkan response message yang sudah dibuat ke browser (client) dan dapat ditampilkan dengan benar di sisi client
- Jika file yang diminta oleh client tidak tersedia, web server dapat mengirimkan pesan “404 Not Found” dan dapat ditampilkan dengan benar di sisi client.

# BAB II
# HASIL PROGRAM DAN ANALISIS

Hasil program dan analisis

```
import socket #import modul socket
import os #import modul os

# membuat socket TCP (1)
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #membuat objek socket TCP

# mengaitkan socket ke alamat dan port tertentu
server_address = ('localhost', 8080) #membuat objek server_address
server_socket.bind(server_address) #mengaitkan server_socket ke server_address

# mendengarkan koneksi masuk
server_socket.listen(1) #mendengarkan koneksi masuk dengan parameter backlog 1
print('Server is listening on port 8080...') #menampilkan pesan server sedang mendengarkan koneksi
masuk

while True: #looping forever
   # menerima koneksi masuk
   client_socket, client_address = server_socket.accept()#menerima koneksi masuk dan menyimpannya
ke client_socket dan client_address
   print(f'Connection from {client_address} has been established!')#menampilkan pesan koneksi telah
berhasil dibuat

   # menerima data dari client
   data = client_socket.recv(1024).decode('utf-8')#menerima data dari client dan menyimpannya ke
data
   print(f'Received data from client: {data}')#menampilkan pesan data telah diterima

   # memparsing HTTP request(2)
   request_method = data.split(' ')[0] #mengambil request method
   request_file = data.split(' ')[1]#mengambil request file

   # mencari file yang diminta oleh client(3)
   file_path = os.getcwd() + request_file #menggabungkan current working directory dengan request
file
   if os.path.isfile(file_path): #jika file ditemukan
       # membuka file dan membacanya
       with open(file_path, 'rb') as file:#membuka file dengan mode read binary
           file_content = file.read()#membaca file dan menyimpannya ke file_content
       # membuat HTTP response message(4)
       response = 'HTTP/1.1 200 OK\n\n' + file_content.decode('utf-8')
   elif request_file == "/500":
```

```
        # membuat HTTP response message untuk error 500
        response = 'HTTP/1.1 500 Internal Server Error\n\nError: Server Error'
    elif request_file == "/300":
        # membuat HTTP response message untuk error 300
        response = 'HTTP/1.1 300 Multiple Choices\n\nError: Multiple Choices'
    else:
        # jika file tidak ditemukan, mengirimkan pesan "404 Not Found"(6)
        response = 'HTTP/1.1 404 Not Found\n\n404 Not Found'

    # mengirimkan response message ke client(5)
    client_socket.sendall(response.encode('utf-8'))

    # menutup koneksi dengan client
    client_socket.close()
```

# BAB III
# KESIMPULAN