JARINGAN KOMPUTER – TUGAS PENDAHULUAN MODUL 7
SOCKET PROGRAMMING

Nama : Muhammad Hamzah Haifan Ma'ruf

NIM : 2311102091

Kelas : S1IF-11-07

Kode Dosen : AIZ

JAWABAN MODUL

A. Teori

1. Apa yang dimaksud dengan socket dalam konteks network programming? Jelaskan peran socket sebagai antarmuka komunikasi antara aplikasi dan jaringan. Bagaimana socket memungkinkan komunikasi antara dua perangkat di jaringan?

   Socket adalah sebuah antarmuka (interface) yang digunakan oleh aplikasi untuk berkomunikasi melalui jaringan. Dalam konteks network programming, socket berfungsi sebagai endpoint (titik akhir) dari sebuah koneksi dua arah antara dua perangkat.

   Peran Socket:

   - Sebagai antarmuka komunikasi antara aplikasi dan jaringan (TCP/IP stack).
   - Mengelola proses pengiriman dan penerimaan data antara dua perangkat (komputer, server, dll.).

   Bagaimana socket memungkinkan komunikasi?

   - Socket dibentuk dengan kombinasi IP Address + Port Number.
   - Aplikasi menggunakan socket untuk membuat koneksi, mengirim data, dan menerima data melalui protokol tertentu (seperti TCP atau UDP).
   - Contoh: Browser (klien) membuka koneksi socket ke server web di port 80 untuk menerima halaman web.

2. Jelaskan perbedaan antara TCP (Transmission Control Protocol) dan UDP (User Datagram Protocol) dalam konteks socket programming. Berikan contoh kasus penggunaan masing-masing protokol dalam aplikasi nyata.

Fitur	TCP	UPP

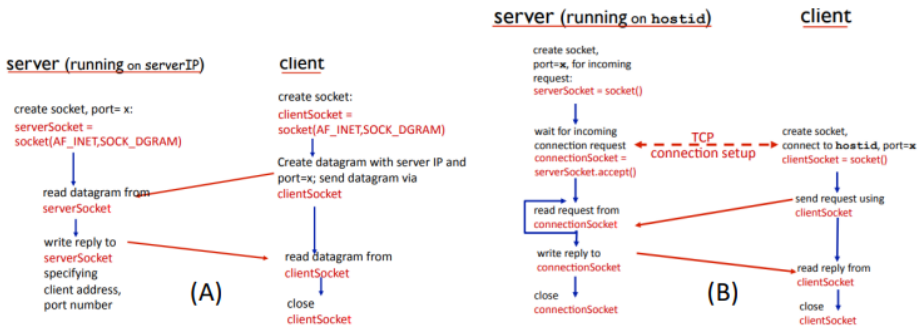
Jenis Protokol	Connection-oriented	Connectionless
Keandalan	Lebih andal (ada verifikasi dan pengurutan data)	Kurang andal (tidak menjamin urutan atau keberhasilan kirim)
Kecepatan	Lebih lambat karena overhead pengelolaan koneksi	Lebih cepat
Contoh Fungsi	`socket.SOCK_STREAM`	`socket.SOCK_DGRAM`

Contoh Kasus Penggunaan:

TCP: Aplikasi web, email, transfer file (FTP), karena butuh integritas data dan koneksi yang stabil.

UDP: Aplikasi streaming video/audio, VoIP, game online, karena butuh kecepatan dan toleransi terhadap kehilangan data.

3. Perhatikan gambar dibawah ini:



Jelaskan apa maksud dari kedua gambar diatas, serta sebutkan apa yang membuat kedua gambar tersebut berbeda!

Gambar di atas menunjukkan dua model komunikasi dalam network programming menggunakan socket, yaitu komunikasi menggunakan protokol UDP (User Datagram Protocol) dan TCP (Transmission Control Protocol). Pada gambar (A), komunikasi dilakukan dengan menggunakan socket bertipe datagram (`SOCK_DGRAM`) yang mengacu pada protokol UDP. Protokol ini bersifat

connectionless, artinya tidak memerlukan proses penyambungan koneksi terlebih dahulu antara server dan client. Dalam alurnya, server membuat socket UDP dan menunggu datagram dari client. Setelah menerima datagram, server membalas pesan tersebut ke alamat dan port client. Di sisi client, socket UDP dibuat, kemudian client mengirim datagram ke IP dan port server, menerima balasan, lalu menutup koneksi.

Sementara itu, pada gambar (B), komunikasi dilakukan dengan socket bertipe stream (`SOCK_STREAM`) yang menggunakan protokol TCP. Protokol TCP bersifat connection-oriented, artinya sebelum pertukaran data dimulai, diperlukan proses penyambungan koneksi melalui mekanisme handshake. Server membuat socket TCP, kemudian menunggu koneksi masuk menggunakan fungsi `accept()`, dan komunikasi dilakukan melalui socket hasil koneksi tersebut. Di sisi client, socket TCP dibuat dan menghubungkan diri ke server menggunakan `connect()`, setelah itu client mengirim permintaan, menerima balasan, lalu menutup koneksi.

Perbedaan utama antara kedua gambar ini terletak pada jenis protokol yang digunakan. Gambar (A) menggunakan UDP yang lebih cepat karena tidak menjamin keandalan dan tidak memerlukan koneksi, sehingga cocok untuk aplikasi seperti streaming, game online, dan VoIP. Sebaliknya, gambar (B) menggunakan TCP yang lebih andal karena menjamin urutan dan integritas data, meskipun lebih lambat, dan cocok digunakan dalam aplikasi seperti web server, email, dan transfer file.

4. Jelaskan struktur alamat yang digunakan dalam socket programming. Apa yang dimaksud dengan IP Address dan Port Number, serta bagaimana keduanya bekerja bersama untuk menentukan tujuan komunikasi?

   Socket menggunakan alamat berikut:

   - IP Address (Alamat IP): Alamat unik perangkat dalam jaringan, contohnya 192.168.1.10.
   - Port Number (Nomor Port): Angka untuk mengidentifikasi aplikasi atau layanan tertentu pada perangkat, contohnya 80 untuk HTTP.

   Kombinasi IP dan Port:

   - Bersama-sama, IP dan Port digunakan untuk menentukan tujuan komunikasi secara spesifik, misalnya: 192.168.1.10:80 → server web di perangkat dengan IP tersebut.

5. Dalam socket programming akan ada kasus yang akan memerlukan Thread yang sudah anda pelajari pada Mata Kuliah Sistem Operasi. Sehingga jelaskan apa yang dimaksud dengan Threading dan Multi-Threading! Selain itu pada code dibawah ini mana yang termasuk Threading dan Multi-Threading? Berikan penjelasannya!

   Threading: Eksekusi satu unit kerja (thread) secara paralel dengan proses utama.

   Multi-Threading: Menjalankan beberapa thread sekaligus, memungkinkan program menangani beberapa tugas secara bersamaan — cocok untuk server yang menangani banyak klien.

   - Code 1

```
import threading
def task():
      print("Task is running")
thread = threading.Thread(target=task)
thread.start()
thread.join()
```

   Ini adalah Threading tunggal, hanya satu thread yang dibuat untuk menjalankan task.

   - Code 2

```
import threading
def task(name):
      print(f"Task {name} is running")
threads = []
for i in range(5):
      thread = threading.Thread(target=task, args=(i,))
      threads.append(thread)
      thread.start()
for thread in threads:
      thread.join()
```

   Ini adalah Multi-Threading karena lima thread dibuat dan dijalankan secara paralel untuk menjalankan task.

6. Dalam network programming, sering kali terjadi error seperti timeout, connection refused, atau address already in use. Jelaskan penyebab umum dari masing-masing error berikut dan strategi umum untuk menanganinya:

    a. Timeout

    Penyebab: Server tidak merespons dalam waktu yang ditentukan.

    Penanganan:

    - Tambahkan penanganan timeout dengan try-except.
    - Atur waktu timeout yang cukup (socket.settimeout()).
    - Cek koneksi jaringan dan ketersediaan server.

    b. Connection Refused

    Penyebab:

    - Server tidak berjalan atau port tidak terbuka.
    - Aplikasi server belum bind ke port yang sesuai.

    Penanganan:

    - Pastikan server aktif dan mendengarkan pada port.
    - Periksa firewall atau security group (jika cloud).

    c. Address Already in Use

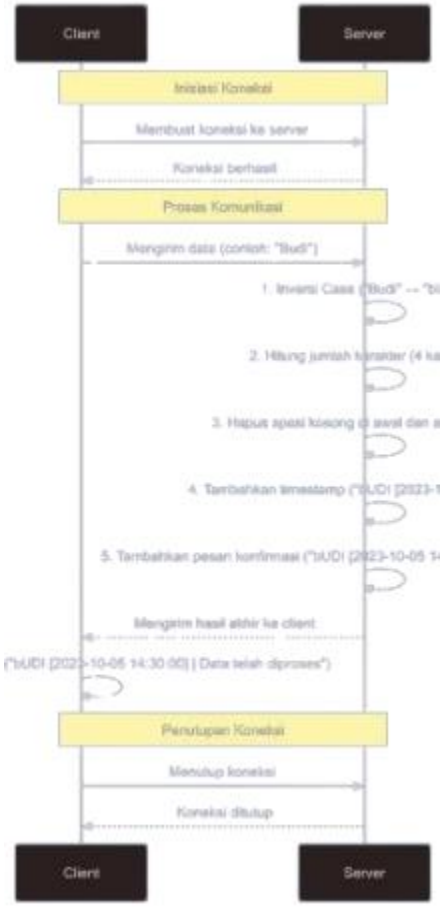
    Penyebab: Port sudah digunakan oleh proses lain atau socket belum dilepaskan dengan benar.

    Penanganan:

    - Gunakan socket.SO_REUSEADDR agar bisa reuse address.
    - Pastikan tidak ada proses lain yang sedang bind ke port tersebut.

Berikut merupakan algoritma sederhana untuk socket programming yang akan digunakan pada poin B dan poin C:

Program ini adalah program client-server sederhana. Client mengirimkan data ke server, server melakukan 5 proses berbeda terhadap data tersebut, lalu mengembalikan hasilnya ke client untuk ditampilkan.



Langkah-Langkah Algoritma

1. Inisialisasi Server:

- Server membuat socket untuk mendengarkan koneksi masuk dari client.
- Server mengikat socket ke alamat IP dan port tertentu.
- Server mulai mendengarkan koneksi masuk dari client

2. Inisialisasi Client:

- Client membuat socket untuk terhubung ke server.

- Client menghubungkan socket ke alamat IP dan port server.

3. Proses Komunikasi:

- Client:

a. Membaca sebaris karakter (data) dari input pengguna (keyboard).
b. Mengirimkan data tersebut ke server melalui socket.

- Server:

a. Menerima data dari client melalui socket.
b. Melakukan 5 proses berikut terhadap data:

1) Inversi Case: Jika karakter adalah huruf kecil, ubah menjadi kapital; jika
2) kapital, ubah menjadi kecil. Karakter non-huruf tetap tidak berubah.
3) Menghitung jumlah karakter dalam data.
4) Menghapus semua spasi kosong di awal dan akhir data.
5) Menambahkan timestamp (waktu saat ini) ke data.
6) Menambahkan pesan konfirmasi "Data telah diproses".

c. Mengirimkan data yang telah dimodifikasi kembali ke client.

- Client:

a. Menerima data yang telah dimodifikasi dari server.
b. Menampilkan data tersebut di layar.

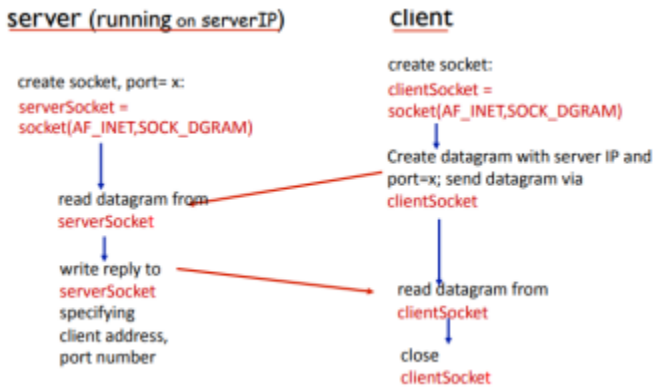
4. Penutupan Koneksi:

- Setelah selesai, baik client maupun server menutup socket untuk mengakhiri komunikasi.

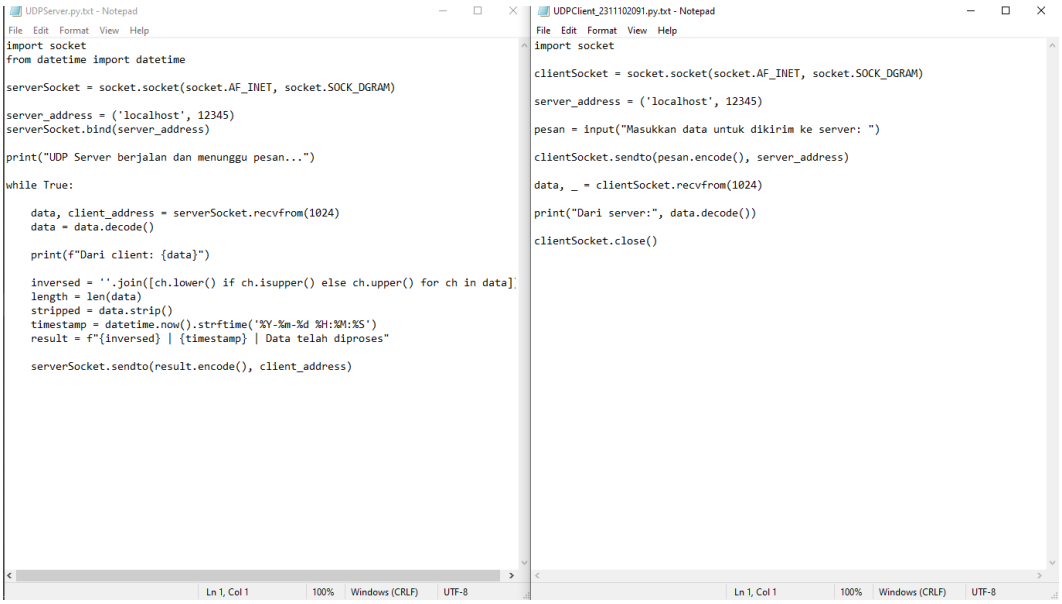
B. Socket Programming with UDP

Berikan screenshot dari kode dan hasil dari progam yang akan dibuat berikut ini:

1. Buatlah program client dan server dengan UDP menggunakan bahasa pemrograman Python sesuai dengan algoritma diatas! Untuk mempermudah pemahaan berkaitan algoritma diatas berikut skema program nanti harus berjalan



Berikan nama setiap file dengan UDPClient_NIM.py dan UDPServer.py!



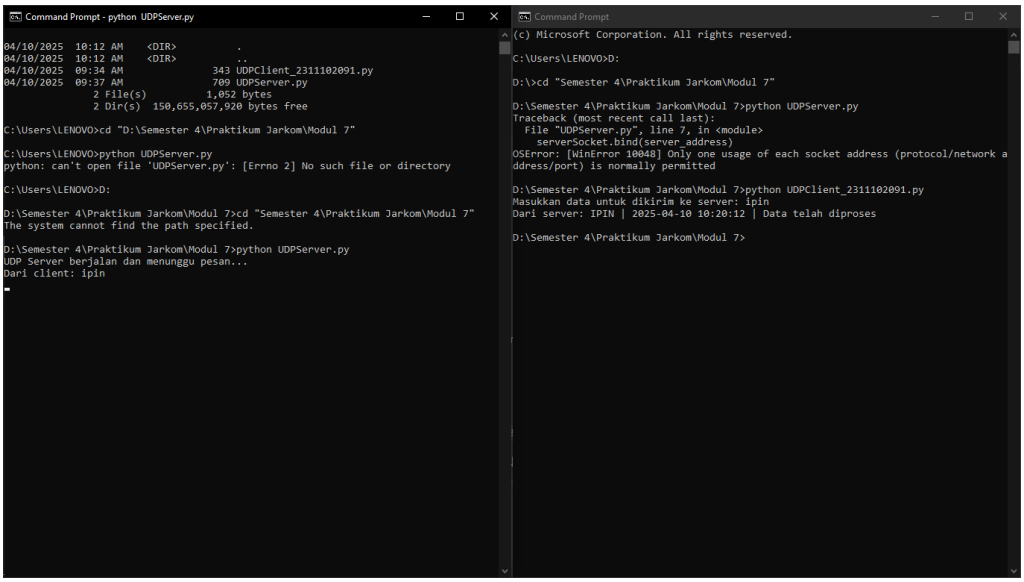
UDPServer.py.txt - Notepad

```
import socket
from datetime import datetime

serverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('localhost', 12345)
serverSocket.bind(server_address)

print("UDP Server berjalan dan menunggu pesan...")

while True:

    data, client_address = serverSocket.recvfrom(1024)
    data = data.decode()

    print(f"Dari client: {data}")

    inversed = ''.join([ch.lower() if ch.isupper() else ch.upper() for ch in data])
    length = len(data)
    stripped = data.strip()
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    result = f"{inversed} | {timestamp} | Data telah diproses"

    serverSocket.sendto(result.encode(), client_address)
```

UDPClient_2311102091.py.txt - Notepad

```
import socket

clientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('localhost', 12345)

pesan = input("Masukkan data untuk dikirim ke server: ")

clientSocket.sendto(pesan.encode(), server_address)

data, _ = clientSocket.recvfrom(1024)

print("Dari server:", data.decode())

clientSocket.close()
```

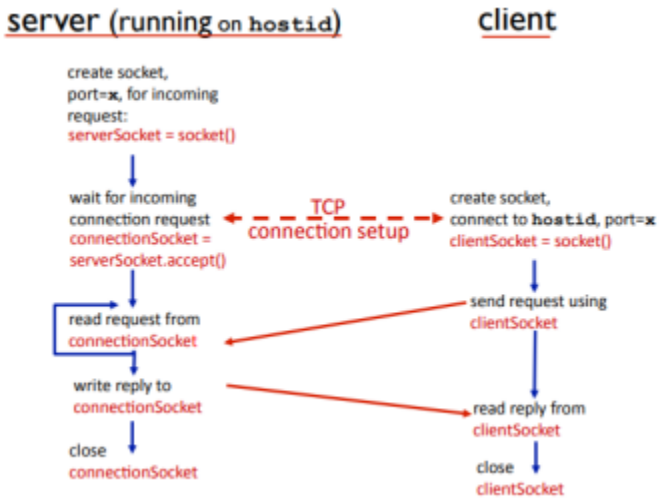
2. Eksekusi code yang sudah dibuat sehingga dapat menunjukkan hubungan antara client-server (WAJIB GUNAKAN CMD/TERMINAL)! Jika tidak muncul hubungan tersebut berarti ada masalah dengan code yang anda buat, silahkan diperbaiki!



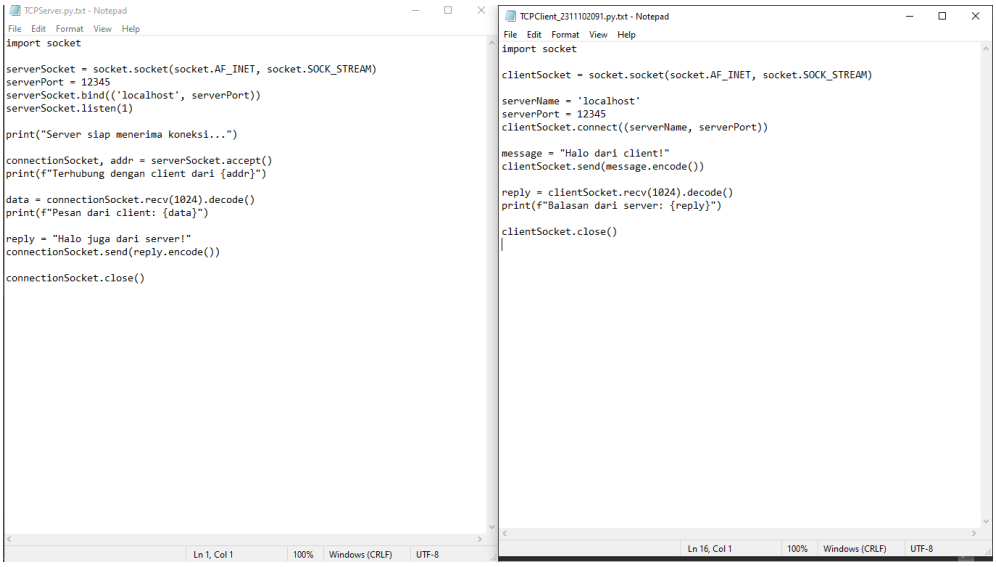
C. Socket Programming with TCP

Berikan screenshot dari kode dan hasil dari progam yang akan dibuat berikut ini:

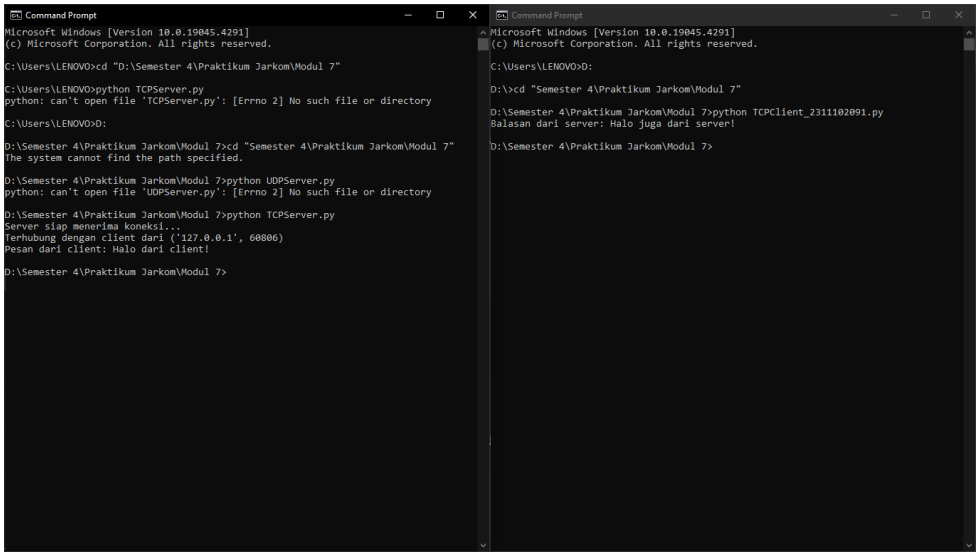
1. Buatlah program client dan server dengan TCP menggunakan bahasa pemrograman Python sesuai dengan algoritma diatas! Untuk mempermudah pemahaan berkaitan algoritma diatas berikut skema program nanti harus berjalan!



Berikan nama setiap file dengan TCPClient_NIM.py dan TCPServer.py!



2. Eksekusi code yang sudah dibuat sehingga dapat menunjukkan hubungan antara client-server (WAJIB GUNAKAN CMD/TERMINAL)! Jika tidak muncul hubungan tersebut berarti ada masalah dengan code yang anda buat, silahkan diperbaiki!



D. Analisa Socket Program

Pada tahap ini akan melakukan analisa pada kode yang berikan serta menjawab pertanyaan-pertanyaan yang diberikan:

```
import socket

def resolve_dns(domain_name):

 dns_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

 dns_server = "8.8.8.8"

 dns_port = 53


 dns_query = b"\x00\x01\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00"
+ \

 domain_name.encode('utf-8') + b"\x00\x00\x01\x00\x01"


 try:

 print(f"Mengirim permintaan DNS untuk {domain_name}...")

 dns_socket.sendto(dns_query, (dns_server, dns_port))

 dns_socket.settimeout(5) # Timeout 5 detik

response, _ = dns_socket.recvfrom(1024)

 print("Respon DNS diterima.")


 ip_address = socket.inet_ntoa(response[-4:])

 print(f"Alamat IP untuk {domain_name}: {ip_address}")

 return ip_address

 except socket.timeout:

 print("Timeout: Tidak ada respon dari server DNS.")

 return None

 finally:
```

```
 dns_socket.close()

def fetch_web_page(ip_address):

 http_request = f"GET / HTTP/1.1\r\nHost:
{ip_address}\r\nConnection:

close\r\n\r\n"



 tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

 try:

 print(f"Menghubungkan ke server web di {ip_address}:80...")

 tcp_socket.connect((ip_address, 80))

 tcp_socket.sendall(http_request.encode('utf-8'))



 response = b""

 while True:

 data = tcp_socket.recv(4096)

 if not data:

 break

 response += data



 print("Respons HTTP diterima.")

 return response.decode('utf-8')

 except Exception as e:

 print(f"Error: {e}")

 return None
```

```
    finally:
    tcp_socket.close()
# Main program
if __name__ == "__main__":
    domain_name = "example.com"

    ip_address = resolve_dns(domain_name)
    if ip_address:
    print("\nMengunduh halaman web...")
    web_content = fetch_web_page(ip_address)
    if web_content:
    print("\nKonten Halaman Web:")
    print(web_content[:500])
    else:
    print("Gagal mengunduh halaman web.")
    else:
    print("Gagal mendapatkan alamat IP dari DNS.")
```

1. Apa maksud kode diatas?

   Kode di atas merupakan program Python sederhana yang berfungsi untuk menyelesaikan nama domain (contohnya example.com) menjadi alamat IP menggunakan protokol UDP, kemudian menggunakan alamat IP tersebut untuk mengambil halaman utama sebuah situs web melalui protokol HTTP dengan koneksi TCP. Program ini dibagi menjadi dua bagian utama: fungsi resolve_dns() untuk menangani permintaan DNS dan fungsi fetch_web_page() untuk mengunduh halaman web dari alamat IP yang telah diperoleh. Dengan kata lain,

kode ini menunjukkan proses dari pemetaan nama domain hingga pengambilan data web secara lengkap.

2. Protokol apa saya yang akan digunakan pada kode diatas?

   Program ini menggunakan dua jenis protokol komunikasi jaringan, yaitu UDP (User Datagram Protocol) dan TCP (Transmission Control Protocol). Protokol UDP digunakan pada bagian pemetaan domain melalui DNS (Domain Name System), sedangkan protokol TCP digunakan saat mengambil halaman web melalui permintaan HTTP. Kedua protokol ini digunakan secara terpisah namun berurutan, sesuai dengan kebutuhan komunikasi pada tiap tahap.

3. Bagaimana program ini menggunakan protokol TCP untuk mengunduh halaman web melalui HTTP? Jelaskan langkah-langkah yang dilakukan oleh program untuk membuat koneksi TCP dan mengirim permintaan HTTP. Mengapa protokol TCP lebih cocok daripada UDP untuk HTTP?

   Dalam bagian fetch_web_page(), program menggunakan TCP untuk membuat koneksi ke alamat IP target pada port 80 yang merupakan port default untuk layanan HTTP. Setelah koneksi berhasil, program mengirimkan permintaan HTTP dalam bentuk string GET, lalu membaca data balasan dari server secara bertahap sampai semua data diterima. Karena TCP menjamin keandalan, urutan, dan keutuhan data, protokol ini sangat cocok digunakan untuk komunikasi HTTP yang membutuhkan transmisi data yang lengkap dan tidak boleh rusak atau hilang.

4. Jelaskan bagaimana program ini menggunakan protokol UDP untuk menyelesaikan nama domain (example.com) menjadi alamat IP. Mengapa protokol UDP dipilih untuk komunikasi DNS? Apa keuntungan dan kerugian penggunaan UDP dalam konteks ini?

   Pada bagian resolve_dns(), program menggunakan socket UDP untuk mengirimkan query DNS ke server DNS publik Google (8.8.8.8) di port 53. Permintaan DNS dibuat secara manual dalam bentuk biner, kemudian dikirim tanpa membuat koneksi (karakteristik UDP). Server DNS akan membalas dengan data yang berisi alamat IP dari domain yang diminta. UDP dipilih untuk DNS karena lebih ringan dan cepat, serta cukup andal untuk permintaan kecil. Namun, kelemahannya adalah tidak menjamin pengiriman atau keutuhan data. Jika terjadi gangguan, paket bisa hilang dan tidak dikirim ulang secara otomatis.

5. Berikan kemungkinan output dari kode tersebut

Kemungkinan output dari program ini bisa berupa alamat IP hasil pemetaan domain, pesan berhasil menghubungkan server web, serta konten HTML dari halaman web yang diunduh. Sebagai contoh, untuk domain example.com, program dapat menampilkan IP seperti 93.184.216.34 dan sebagian isi halaman web seperti tag <html>, <head>, dan <title>. Jika terjadi masalah, output bisa berupa pesan error seperti "Timeout: Tidak ada respon dari server DNS" atau "Gagal mengunduh halaman web", tergantung titik kegagalan yang terjadi selama proses eksekusi program.