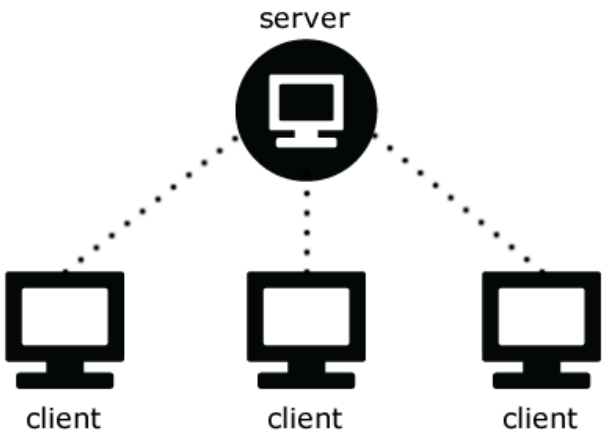
# I. Pengenalan Aplikasi Client-Server Sederhana (Multithread)




Aplikasi Client-Server Sederhana (Multithread) adalah jenis aplikasi yang terdiri dari dua komponen utama: server dan beberapa klien. Server bertindak sebagai entitas pusat yang menerima permintaan dan mengirimkan respon kepada klien. Klien, di sisi lain, adalah pengguna yang terhubung ke server untuk berinteraksi dengan aplikasi.

Dalam implementasi multithread, aplikasi ini menggunakan multiple threads (benang) untuk menangani koneksi dari berbagai klien secara bersamaan. Setiap koneksi klien ditangani dalam thread terpisah, yang memungkinkan server untuk secara efisien melayani banyak klien secara simultan.

# II. Cara Kerja Aplikasi Client-Server Sederhana (Multithread)

Berikut adalah penjelasan singkat tentang bagaimana aplikasi Client-Server Sederhana (Multithread) bekerja:

1. Server Inisialisasi:
- Server dijalankan dan siap menerima koneksi dari klien.
- Inisialisasi sumber daya yang diperlukan oleh server, seperti socket atau port yang akan digunakan untuk berkomunikasi.
2. Menerima Koneksi:
- Server mulai mendengarkan permintaan koneksi masuk dari klien.
- Setiap kali ada permintaan koneksi masuk, server menerima koneksi tersebut dan membuat thread baru untuk menangani koneksi tersebut.
3. Thread Klien:
- Setiap thread klien bertanggung jawab untuk menangani komunikasi dengan satu klien tertentu.
- Thread klien menerima permintaan dari klien, memprosesnya, dan mengirimkan balasan kembali ke klien.

- Thread-thread klien berjalan secara independen, memungkinkan server untuk melayani banyak klien secara bersamaan.

4. Komunikasi antara Klien dan Server:
- Klien mengirim permintaan ke server melalui koneksi yang telah dibuat.
- Server menerima permintaan tersebut dan menjalankan tugas yang diperlukan berdasarkan permintaan tersebut.
- Server mengirimkan balasan kembali ke klien melalui koneksi yang sama.

5. Sinkronisasi:
- Dalam beberapa situasi, thread-thread klien perlu melakukan sinkronisasi untuk menghindari kondisi perlombaan (race condition) atau konflik akses ke sumber daya bersama.
- Teknik sinkronisasi seperti mutex, semafor, atau penguncian lainnya digunakan untuk mengatur akses ke sumber daya bersama agar tidak terjadi konflik.

6. Penutupan Koneksi:
- Setelah selesai melakukan komunikasi, koneksi antara klien dan server harus ditutup secara eksplisit.
- Setiap thread klien dapat diberhentikan atau dibebaskan untuk digunakan kembali setelah koneksi ditutup.

Dalam aplikasi Client-Server Sederhana (Multithread), setiap koneksi klien ditangani dalam thread terpisah, yang memungkinkan server untuk secara simultan melayani banyak klien. Ini meningkatkan skalabilitas dan efisiensi aplikasi, memungkinkan lebih banyak pengguna untuk terhubung dan berinteraksi dengan server secara bersamaan.

## III. Pembuatan Aplikasi Aplikasi Client-Server Sederhana (Multithread)

Berikut adalah contoh sederhana program Aplikasi Client-Server Sederhana (Multithread) menggunakan Python:

Server:

```
import socket
import threading


def handle_client(client_socket, client_address):
    print(f"[NEW CONNECTION] Client {client_address} connected.")

    while True:
        # Menerima pesan dari client
        message = client_socket.recv(1024).decode('utf-8')
```

```
        if message == 'exit':
            break

        # Menampilkan pesan dari client
        print(f"[CLIENT {client_address}] {message}")

        # Mengirim balasan ke client
        response = input("Your response: ")
        client_socket.send(response.encode('utf-8'))

    # Menutup koneksi dengan client
    client_socket.close()
    print(f"[CONNECTION CLOSED] Client {client_address} disconnected.")


def start_server():
    # Konfigurasi host dan port
    host = '127.0.0.1'
    port = 8000

    # Membuat socket server
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)
    print("[SERVER STARTED] Waiting for connections...")

    while True:
        # Menerima koneksi dari client
        client_socket, client_address = server_socket.accept()

        # Menangani koneksi client dalam thread baru
        client_thread = threading.Thread(
            target=handle_client, args=(client_socket, client_address))
        client_thread.start()


start_server()
```

Client :

```
import socket


def start_client():
    # Konfigurasi host dan port server
    host = '127.0.0.1'
```

```
    port = 8000

    # Membuat socket client
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))

    while True:
        # Mengirim pesan ke server
        message = input("Your message: ")
        client_socket.send(message.encode('utf-8'))

        if message == 'exit':
            break

        # Menerima balasan dari server
        response = client_socket.recv(1024).decode('utf-8')
        print(f"[SERVER RESPONSE] {response}")

    # Menutup koneksi dengan server
    client_socket.close()


start_client()
```

## IV. Tugas dan Latihan

Membuat laporan percobaan praktikum dan beri Analisa Hasil Percobaan tadi yang sudah dibuat Aplikasi Client-Server Sederhana (Multithread)