

## Modul Pemrograman Jaringan Python

### Topik Bahasan

- Server UDP
- Client UDP

### 1. Tujuan

Membangun server game sederhana yang mendukung komunikasi berbasis UDP. Game ini akan memungkinkan beberapa pemain untuk terhubung dan berinteraksi dengan server, seperti bergerak di dalam dunia game dan saling berinteraksi.

### 2. Fitur Utama

1. **Koneksi Klien:** Mendukung beberapa klien yang terhubung ke server.
2. **Koordinasi Pemain:** Menerima dan menyebarkan informasi posisi pemain.
3. **Update Dunia Game:** Menyebarkan update posisi pemain ke semua klien secara real-time.
4. **Handling Koneksi:** Mengelola koneksi klien dan mengatasi pemutusan koneksi.

### 3. Desain Aplikasi

- **Server Game:**
  - Menerima posisi pemain dari klien.
  - Mengupdate posisi pemain dan menyebarkannya ke semua klien.
  - Mengelola daftar klien yang terhubung.
- **Klien Game:**
  - Mengirim informasi posisi ke server.
  - Menerima update posisi dari server.
  - Mengirim input pemain ke server.

## 4. Struktur Kode

### 4.1 Server Game

- Server game akan menerima informasi posisi dari klien dan menyebarkannya ke semua klien yang terhubung.

```
import socket
import threading
import time
import random

# Konfigurasi klien
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345
ADDR = (SERVER_HOST, SERVER_PORT)

def receive_updates(client_socket):
    while True:
        try:
            message, _ = client_socket.recvfrom(1024)
            print(f"\nUpdate dari server: {message.decode()}")
        except:
            break

def send_position(client_socket):
    while True:
        # Simulasikan posisi acak
        position = f"{random.randint(0, 100)},{random.randint(0, 100)}"
        client_socket.sendto(position.encode(), ADDR)
        time.sleep(1) # Kirim update setiap detik

def start_client():
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as client_socket:
        # Thread untuk menerima update
        thread = threading.Thread(target=receive_updates, args=(client_socket,))
        thread.start()

        # Thread untuk mengirim posisi
        send_position(client_socket)

if __name__ == "__main__":
    start_client()
```

Ln: 3 Col: 11

Hasil running server\_udp.py

```
program modul 5-6 — Python server_udp.py — 80x9
Last login: Sun Aug 25 19:07:59 on console
[user@usernoMacBook-Pro ~ % cd Desktop/program\ modul\ 5-6/
[user@usernoMacBook-Pro program modul 5-6 % ls
client_udp.py  server_udp.py
[user@usernoMacBook-Pro program modul 5-6 % python3 server_udp.py
Server berjalan. Tekan Enter untuk keluar.Server game berjalan di 127.0.0.1:12345
5
```

## 4.2 Klien Game

Klien game akan mengirim posisi pemain ke server dan menerima update posisi dari server.

```
client_udp.py
import socket
import threading
import time
import random

# Konfigurasi klien
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345
ADDR = (SERVER_HOST, SERVER_PORT)

def receive_updates(client_socket):
    while True:
        try:
            message, _ = client_socket.recvfrom(1024)
            print(f"\nUpdate dari server: {message.decode()}")
        except:
            break

def send_position(client_socket):
    while True:
        # Simulasikan posisi acak
        position = f"{random.randint(0, 100)},{random.randint(0, 100)}"
        client_socket.sendto(position.encode(), ADDR)
        time.sleep(1) # Kirim update setiap detik

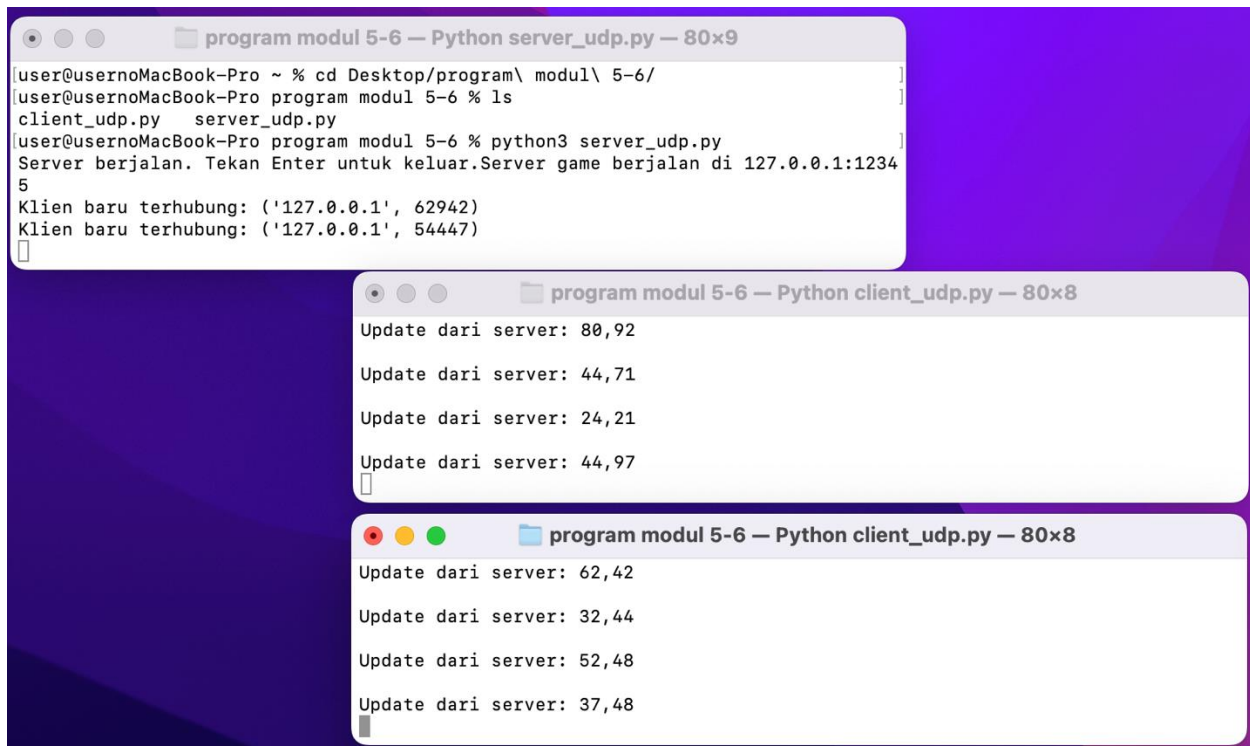
def start_client():
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as client_socket:
        # Thread untuk menerima update
        thread = threading.Thread(target=receive_updates, args=(client_socket,))
        thread.start()

        # Thread untuk mengirim posisi
        send_position(client_socket)

if __name__ == "__main__":
    start_client()

Ln: 3 Col: 11
```

## Hasil running client\_udp.py



The image shows three terminal windows on a macOS desktop. The top window, titled 'program modul 5-6 — Python server\_udp.py — 80x9', shows the server script being executed. It displays the directory listing, the execution of the server script, and two successful client connections from 127.0.0.1. The middle window, titled 'program modul 5-6 — Python client\_udp.py — 80x8', shows a client script receiving four updates from the server with timestamps like 80,92 and 44,71. The bottom window, also titled 'program modul 5-6 — Python client\_udp.py — 80x8', shows another client script receiving four updates from the server with timestamps like 62,42 and 32,44.

```
program modul 5-6 — Python server_udp.py — 80x9
user@usernoMacBook-Pro ~ % cd Desktop/program\ modul\ 5-6/
user@usernoMacBook-Pro program modul 5-6 % ls
client_udp.py  server_udp.py
user@usernoMacBook-Pro program modul 5-6 % python3 server_udp.py
Server berjalan. Tekan Enter untuk keluar.Server game berjalan di 127.0.0.1:1234
5
Klien baru terhubung: ('127.0.0.1', 62942)
Klien baru terhubung: ('127.0.0.1', 54447)

```

```
program modul 5-6 — Python client_udp.py — 80x8
Update dari server: 80,92
Update dari server: 44,71
Update dari server: 24,21
Update dari server: 44,97

```

```
program modul 5-6 — Python client_udp.py — 80x8
Update dari server: 62,42
Update dari server: 32,44
Update dari server: 52,48
Update dari server: 37,48

```

## 5. Penjelasan

- **Server:**
  - **handle\_client():** Fungsi ini menerima pesan dari klien, yang berisi posisi pemain atau informasi lainnya, dan menyebarkan pesan tersebut ke semua klien yang terhubung.
  - **broadcast(message, sender\_address):** Fungsi ini mengirimkan pesan ke semua klien kecuali pengirim pesan.
  - **Multithreading:** Server menggunakan thread untuk menjalankan fungsi `handle_client`, memungkinkan server untuk menangani beberapa klien secara bersamaan.
- **Klien:**
  - **receive\_updates(client\_socket):** Fungsi ini berjalan di thread terpisah untuk menerima update posisi dari server.
  - **send\_position(client\_socket):** Fungsi ini mengirimkan posisi acak ke server setiap detik untuk mensimulasikan pergerakan pemain.
  - **start\_client():** Fungsi ini memulai thread untuk menerima update dari server dan mengirimkan posisi ke server.

## Pengujian

1. **Jalankan Server:** Mulai server dengan menjalankan skrip server di terminal.
2. **Jalankan Klien:** Mulai beberapa instansi klien di terminal yang berbeda. Klien akan mengirimkan posisi mereka secara acak dan menerima update dari server.
3. **Uji Fungsionalitas:**
  - Periksa apakah setiap klien mengirimkan posisi mereka ke server dan menerima update posisi dari pemain lain.
  - Verifikasi bahwa pesan dari klien yang terhubung diterima dan disebarkan ke semua klien lainnya.

Selain aplikasi chat atau server game sederhana, UDP dengan multithreading dapat diterapkan dalam berbagai skenario lain yang memerlukan komunikasi jaringan yang cepat dan efisien. Berikut adalah beberapa contoh penerapan UDP dengan multithreading di luar chat atau game:

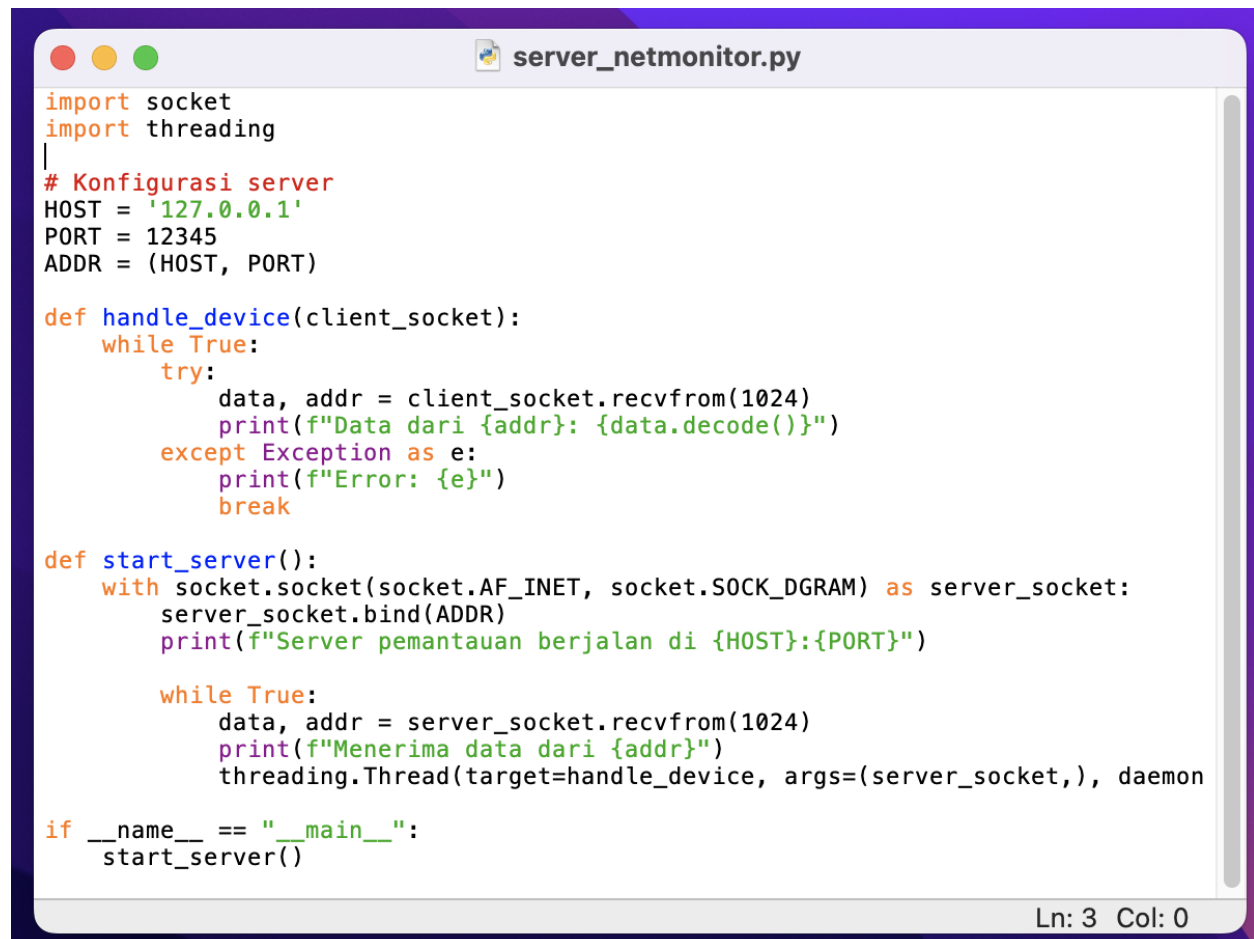
## 1. Sistem Pemantauan Jaringan

Sistem pemantauan jaringan dapat menggunakan UDP untuk mengumpulkan data dari berbagai perangkat atau node dalam jaringan. Multithreading memungkinkan pengumpulan dan pemrosesan data dari beberapa perangkat secara bersamaan.

### Contoh: Server Pemantauan Jaringan

#### Server:

- Menerima data dari berbagai perangkat secara simultan.
- Memproses dan menyimpan data yang diterima untuk analisis lebih lanjut.
- Menyediakan antarmuka untuk melihat status jaringan secara real-time.



```
import socket
import threading

# Konfigurasi server
HOST = '127.0.0.1'
PORT = 12345
ADDR = (HOST, PORT)

def handle_device(client_socket):
    while True:
        try:
            data, addr = client_socket.recvfrom(1024)
            print(f"Data dari {addr}: {data.decode()}")
        except Exception as e:
            print(f"Error: {e}")
            break

def start_server():
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:
        server_socket.bind(ADDR)
        print(f"Server pemantauan berjalan di {HOST}:{PORT}")

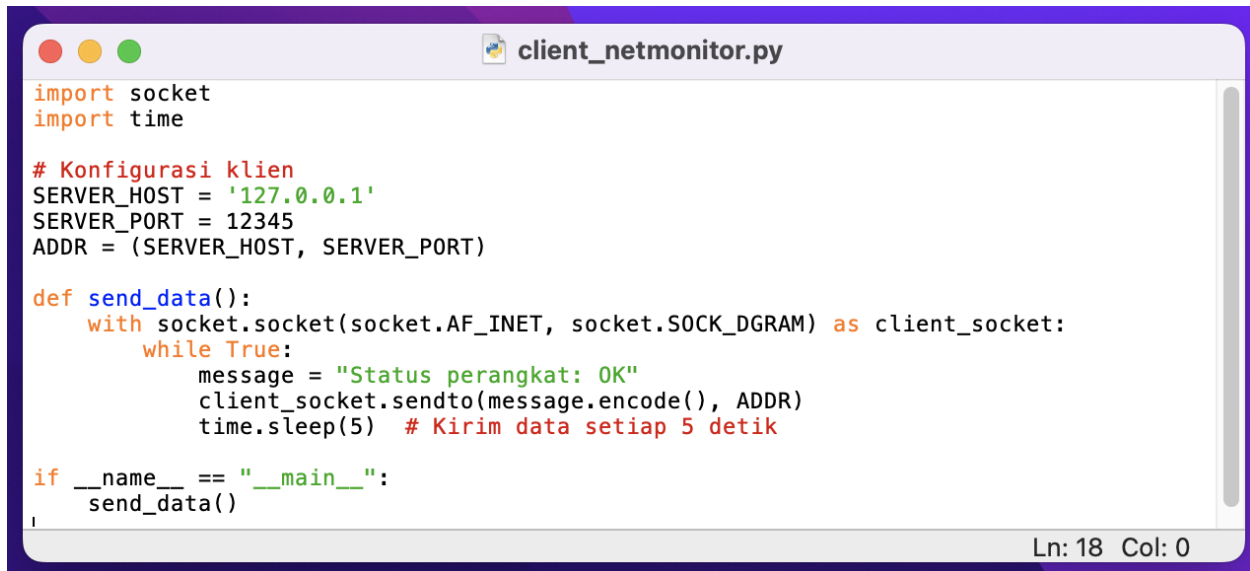
        while True:
            data, addr = server_socket.recvfrom(1024)
            print(f"Menerima data dari {addr}")
            threading.Thread(target=handle_device, args=(server_socket,)).daemon

if __name__ == "__main__":
    start_server()

Ln: 3 Col: 0
```

## Klien:

- Mengirimkan data secara berkala ke server pemantauan.



```
client_netmonitor.py

import socket
import time

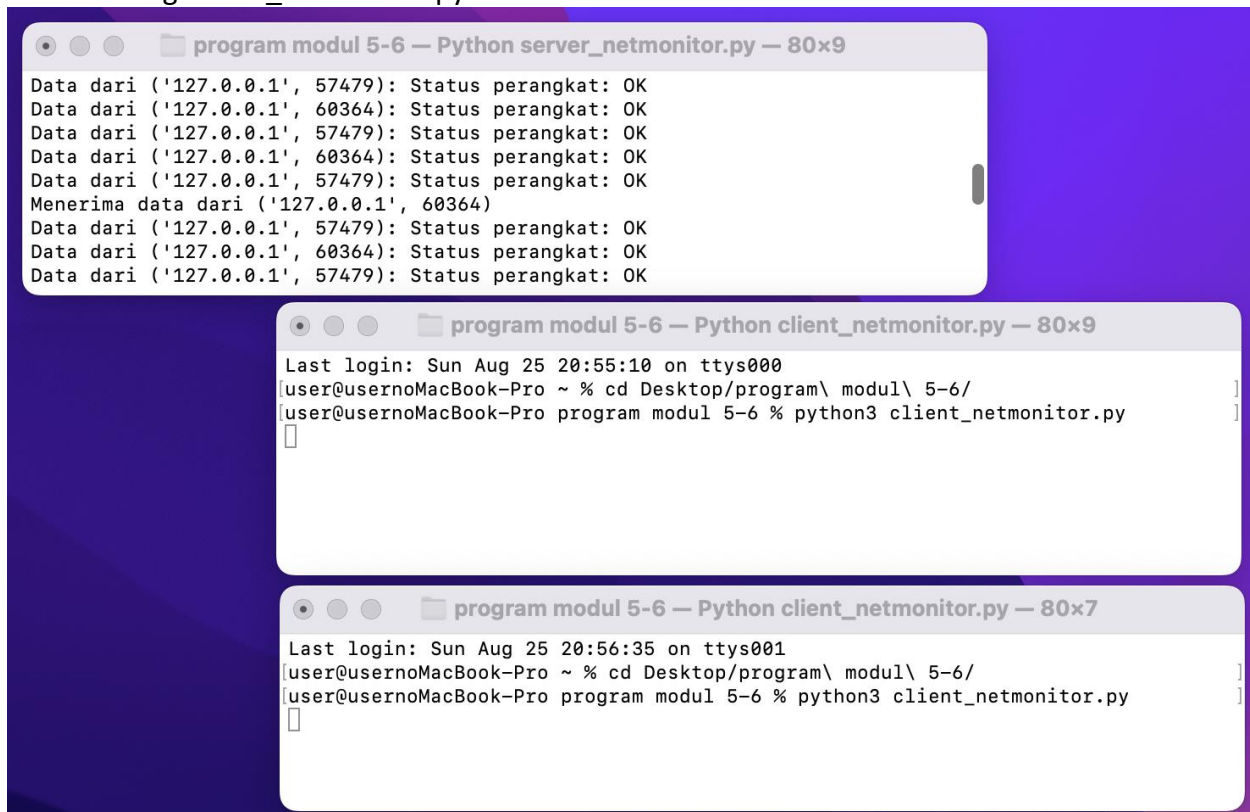
# Konfigurasi klien
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345
ADDR = (SERVER_HOST, SERVER_PORT)

def send_data():
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as client_socket:
        while True:
            message = "Status perangkat: OK"
            client_socket.sendto(message.encode(), ADDR)
            time.sleep(5) # Kirim data setiap 5 detik

if __name__ == "__main__":
    send_data()

Ln: 18 Col: 0
```

## Hasil running server\_netmonitor.py



```
program modul 5-6 — Python server_netmonitor.py — 80x9
Data dari ('127.0.0.1', 57479): Status perangkat: OK
Data dari ('127.0.0.1', 60364): Status perangkat: OK
Data dari ('127.0.0.1', 57479): Status perangkat: OK
Data dari ('127.0.0.1', 60364): Status perangkat: OK
Data dari ('127.0.0.1', 57479): Status perangkat: OK
Menerima data dari ('127.0.0.1', 60364)
Data dari ('127.0.0.1', 57479): Status perangkat: OK
Data dari ('127.0.0.1', 60364): Status perangkat: OK
Data dari ('127.0.0.1', 57479): Status perangkat: OK

program modul 5-6 — Python client_netmonitor.py — 80x9
Last login: Sun Aug 25 20:55:10 on ttys000
[user@usernoMacBook-Pro ~ % cd Desktop/program\ modul\ 5-6/
[user@usernoMacBook-Pro program modul 5-6 % python3 client_netmonitor.py
]

program modul 5-6 — Python client_netmonitor.py — 80x7
Last login: Sun Aug 25 20:56:35 on ttys001
[user@usernoMacBook-Pro ~ % cd Desktop/program\ modul\ 5-6/
[user@usernoMacBook-Pro program modul 5-6 % python3 client_netmonitor.py
]
```

## Tugas

- Buatlah penerapan udp multithreading pada studi kasus lain