Project Akhir Sistem Terdistribusi

Peer to Peer Messaging

NAMA	: Cholif Bima Ardiansyah
NIM	: L0123040
HARI, TANGGAL	: Rabu, 24 Desember 2025
WAKTU	: 20:56
DOSEN	: Fajar Muslim S.T., M.T.

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET SURAKARTA

2025

BAB I

PENDAHULUAN

1. Latar Belakang

Dalam paradigma komputasi modern, ketergantungan pada server pusat seringkali menjadi titik lemah (single point of failure). Jika server mati, seluruh komunikasi terputus. Oleh karena itu, arsitektur Peer-to-Peer (P2P) menawarkan solusi alternatif yang menarik, di mana setiap node (komputer) memiliki kedudukan yang setara dan dapat bertindak sebagai client maupun server secara bersamaan.

Project ini, yang diberi nama "Pesan P2P", dikembangkan untuk mengimplementasikan konsep tersebut. Aplikasi ini memungkinkan pengguna untuk berkomunikasi secara langsung (direct message) maupun broadcast tanpa melalui server perantara. Pengembangan dilakukan menggunakan bahasa pemrograman Python karena dukungan pustaka jaringannya yang kuat namun tetap mudah dipahami.

2. Rumusan Masalah

Tujuan utama dari pengembangan aplikasi ini adalah:

1. Membangun aplikasi *chatting* desentralisasi yang tahan terhadap kegagalan satu titik.
2. Mengimplementasikan protokol komunikasi kustom berbasis JSON di atas TCP Socket.
3. Membuat antarmuka pengguna (GUI) yang modern dan responsif menggunakan library customtkinter.

BAB II

LANDASAN TEORI

1. Arsitektur Peer-to-Peer (P2P)

Berbeda dengan arsitektur *Client-Server* tradisional di mana klien bergantung sepenuhnya pada server pusat, arsitektur *Peer-to-Peer* (P2P) mendistribusikan beban kerja ke seluruh *node* dalam jaringan. Dalam sistem P2P, setiap komputer (*peer*) memiliki kedudukan yang setara (*equipotent*), berfungsi sebagai penyedia layanan (*server*) sekaligus pengguna layanan (*client*). Keunggulan utama model ini adalah desentralisasi, skalabilitas, dan ketahanan terhadap kegagalan satu titik (*robustness*).

2. Socket Programming dan TCP/IP

Socket adalah titik akhir (*endpoint*) dalam komunikasi jaringan dua arah antar proses. Aplikasi ini dibangun menggunakan protokol **TCP (Transmission Control Protocol)** di atas **IP (Internet Protocol)**.

- **TCP:** Dipilih karena sifatnya yang *connection-oriented* dan *reliable*. TCP menjamin bahwa pesan yang dikirim akan diterima secara utuh dan berurutan oleh penerima, yang sangat krusial untuk aplikasi percakapan (chat).
- **Implementasi Python:** Pustaka socket standar Python digunakan untuk menangani operasi tingkat rendah seperti *binding* alamat, *listening* port, dan *accepting* koneksi.

3. Multithreading

Aplikasi komunikasi *real-time* menuntut kemampuan untuk melakukan beberapa tugas sekaligus. Jika aplikasi hanya menggunakan satu alur eksekusi (*single-thread*), antarmuka pengguna (GUI) akan membeku (*freeze*) saat aplikasi sedang menunggu pesan masuk. Oleh karena itu, teknik *Multithreading* diterapkan untuk memisahkan proses:

1. *Main Thread:* Menangani interaksi GUI dan input pengguna.
2. *Worker Thread:* Berjalan di latar belakang untuk terus-menerus mendengarkan data masuk dari jaringan tanpa mengganggu tampilan utama.

BAB III

ANALISIS DAN PERANCANGAN

1. Identifikasi Kebutuhan Fungsional

Aplikasi ini memiliki fitur-fitur utama sebagai berikut:

1. Peer Identity: Pengguna dapat menginisialisasi *node* mereka dengan menentukan Nama (*Username*) dan *Port* unik.
2. Peer Discovery & Connection: Pengguna dapat terhubung ke *peer* lain dengan memasukkan IP Address dan Port target. Sistem secara otomatis melakukan *handshake* untuk bertukar informasi identitas.
3. Messaging:
   Private Chat: Mengirim pesan ke satu *peer* spesifik.
4. Live Status: Indikator status koneksi dan deteksi otomatis jika ada *peer* yang bergabung (*Join*) atau keluar (*Leave*).

2. Arsitektur Sistem

Aplikasi ini menggunakan arsitektur **Unstructured P2P Mesh**.

- Komunikasi Data: Menggunakan protokol TCP/IP melalui library socket Python.
- Format Data: Pertukaran data dibungkus dalam format JSON. Setiap paket data memiliki struktur baku yang didefinisikan dalam class Message, meliputi:
  - type: Tipe pesan (MESSAGE, JOIN, PING, dll).
  - sender_id: Identitas unik pengirim.
  - data: Isi pesan atau payload.
- Multitasking: Menggunakan threading untuk memisahkan proses UI (tampilan) dengan proses mendengarkan pesan masuk (*listening*), sehingga aplikasi tidak *freeze* saat menunggu data.

BAB IV

IMPLEMENTASI KODE

1. Struktur Kode

Project ini dipecah menjadi beberapa modul modular agar mudah dikelola:

- main.py: Menangani antarmuka grafis (GUI) dan interaksi pengguna.
- peer.py: Otak utama logika P2P, mengatur state aplikasi dan daftar teman.
- network.py: Menangani detail teknis koneksi TCP (*low-level socket operations*).
- message.py: Protokol pembungkusan pesan.
- utils.py: Inisialisasi identitas peer.

2. Implementasi Protokol Pesan (message.py)

Agar komunikasi terstandarisasi, saya membuat kelas Message dan MessageType. Enum MessageType mendefinisikan jenis aksi yang bisa dilakukan:

```
class MessageType(Enum):
    MESSAGE = "MESSAGE"      # Chat biasa
    JOIN = "JOIN"            # Handshake awal
    LEAVE = "LEAVE"          # Notifikasi keluar
    PEERS = "PEERS"          # Pertukaran daftar kontak
    PING = "PING"            # Cek koneksi (Heartbeat)
```

Setiap pesan dikonversi menjadi *bytes* JSON sebelum dikirim melalui jaringan untuk memastikan interoperabilitas.

3. Manajemen Koneksi (peer.py & network.py)

Salah satu tantangan terbesar adalah menangani koneksi dua arah. Saya mengimplementasikan P2PServer untuk menerima koneksi masuk dan P2PClient untuk memulai koneksi keluar.

Logic **Handshake** diimplementasikan pada fungsi connect_to_peer dan _handle_join:

1. Saat User A melakukan *connect* ke User B.
2. Aplikasi mengirim pesan bertipe JOIN berisi data diri A.
3. User B menerima, menyimpan A di daftar known_peers, lalu membalas dengan pesan PEERS yang berisi daftar teman yang dia kenal.
4. Ini memastikan kedua belah pihak saling menyimpan kontak ("Saling Follow").

```
def connect_to_peer(self, host: str, port: int) -> bool:
    handler = P2PClient.connect(host, port)
    if not handler: return False

    # Kirim pesan JOIN agar dikenali
    join_msg = Message(
        msg_type=MessageType.JOIN,
        sender_id=self.peer_id,
        sender_name=self.name,
        data={"host": self.local_ip, "port": self.port}
    )
    handler.send(join_msg)
    return True
```

4. Antarmuka Pengguna (GUI)

Untuk tampilan, saya menggunakan customtkinter dengan tema "Dark" agar terlihat modern. GUI terdiri dari *Sidebar* (kiri) untuk daftar user aktif, dan *Chat Area* (kanan) untuk percakapan.

Fitur menarik yang saya tambahkan adalah indikator warna:

- **Merah:** Belum terhubung.
- **Hijau:** Terhubung dan Online.

BAB V

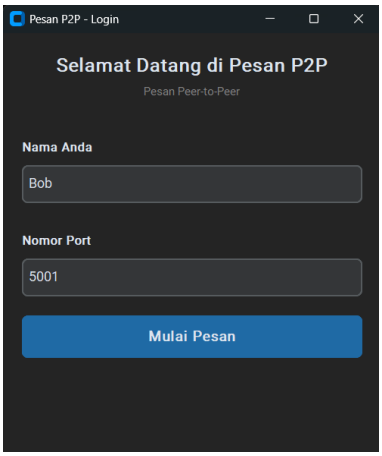
PENGUJIAN DAN ANALISIS HASIL

Pengujian dilakukan menggunakan metode *Localhost Simulation* dengan menjalankan tiga *instance* terminal yang berbeda port (5000 dan 5001) dan IP sama dengan contoh kasus peer Alice dengan peer Bob pada satu mesin.

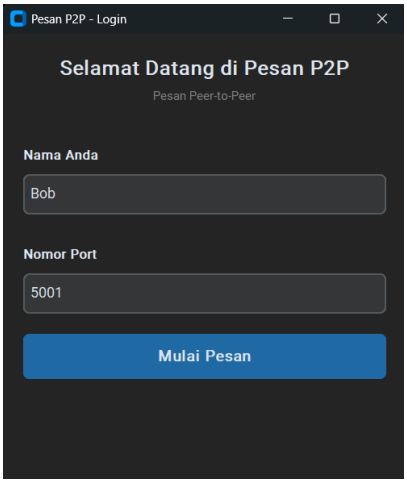
1. Skenario Pengujian 1: Koneksi Antar Peer

Tujuan: Memastikan Peer Alice bisa terhubung ke Peer Bob. Prosedur:

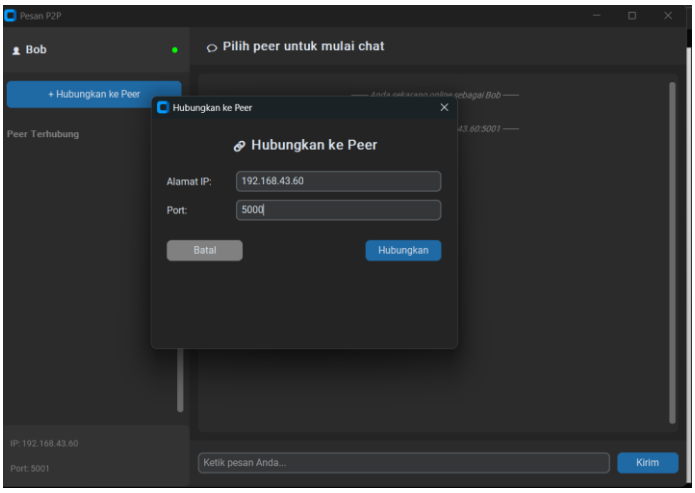
1. Menjalankan aplikasi dengan nama Alice di port 5000.



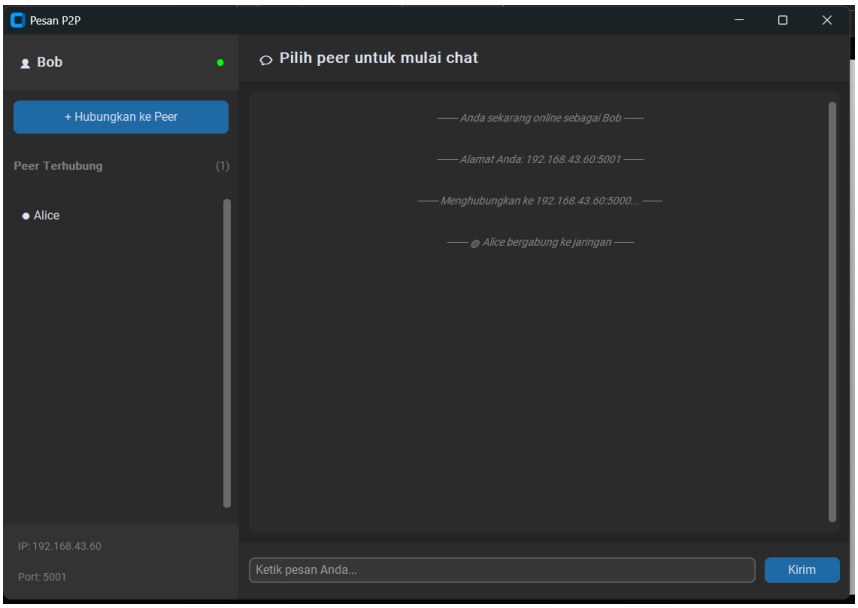
2. Menjalankan aplikasi dengan nama Bob di port 5001.



3. Pada aplikasi Bob, menekan tombol "+ Connect to Peer" dan memasukkan IP/Port Alice.



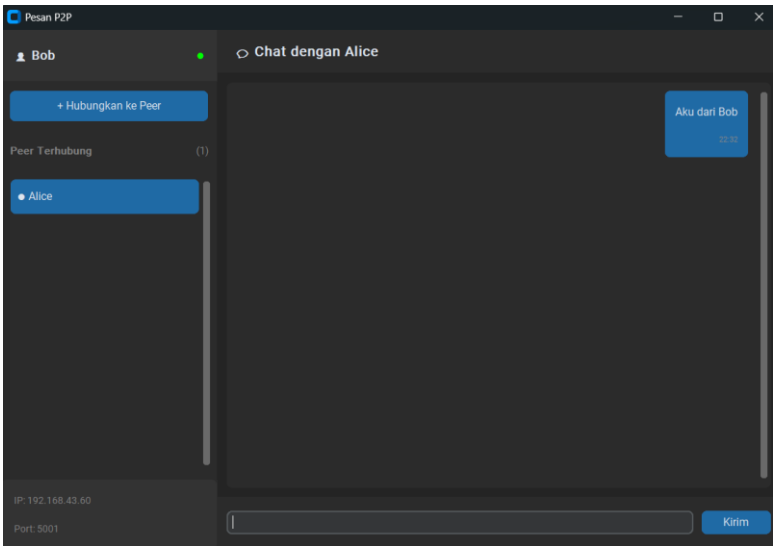
4. Hasil**:** Aplikasi Alice langsung muncul di daftar "Connected Peers" milik B, dan secara otomatis Bob juga muncul di daftar milik Alice berkat mekanisme pesan JOIN dan PEERS yang sudah diimplementasikan.



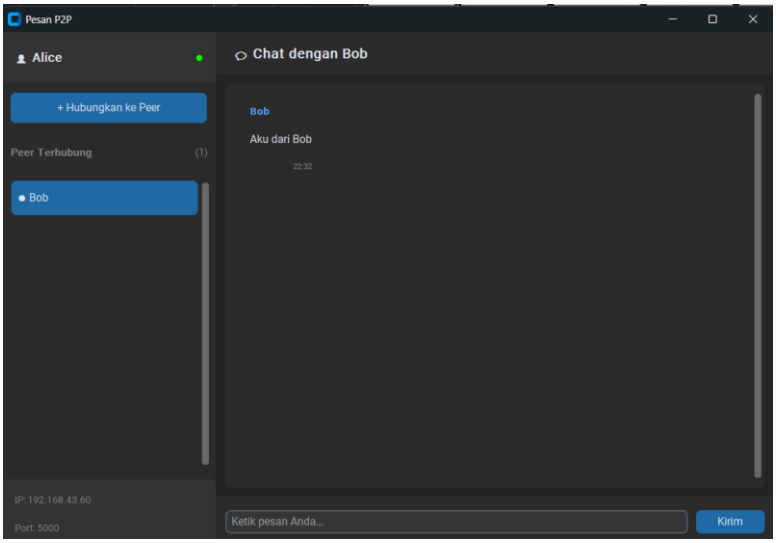
2. Skenario Pengujian 2: Pengiriman Pesan Real-time

Tujuan: Memastikan pesan terkirim tanpa delay. Prosedur:

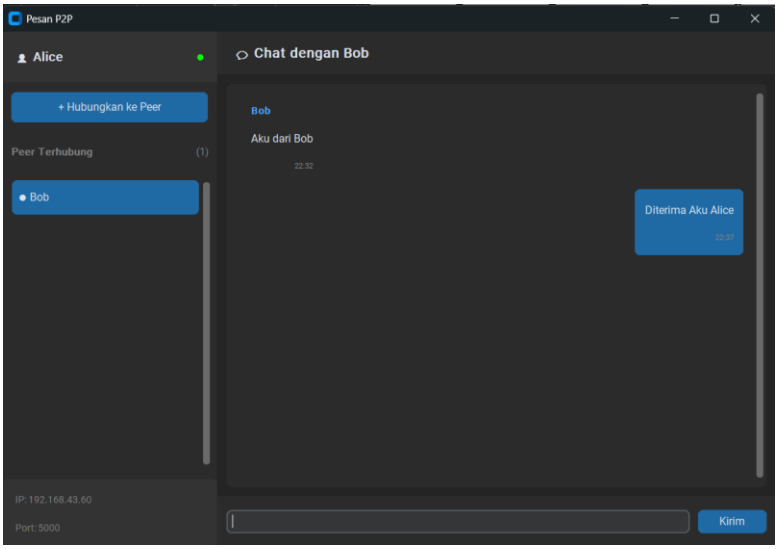
1. Peer Bob mengetik "Aku dari Bob" dan menekan Enter.



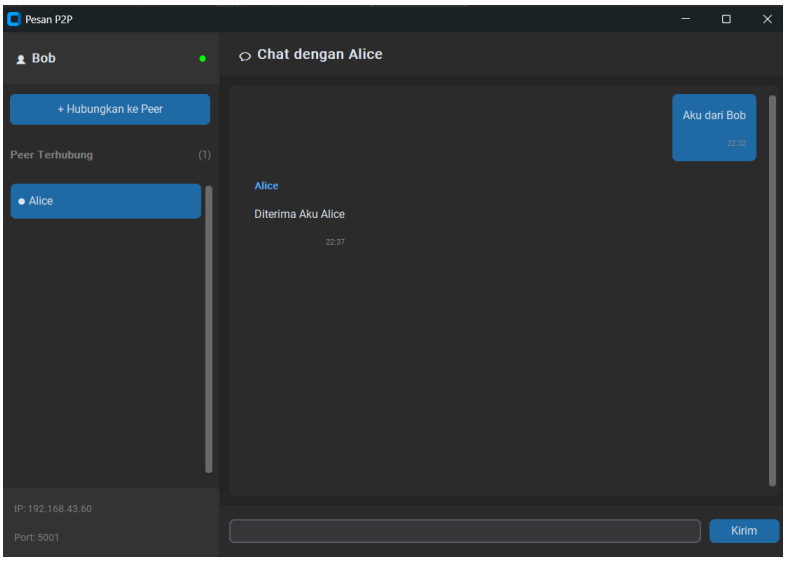
2. Mengamati pesan yang dikirimkan peer Bob dari layar peer Alice.



3. Peer Alice membalas pesan dari peer Bob “Diterima, Aku Alice”.



4. Peer Bob melihat pesan balasan dari peer Alice.



BAB VI

KESIMPULAN

1. Kesimpulan

Berdasarkan hasil implementasi dan pengujian, dapat disimpulkan bahwa:

1. Aplikasi **Pesan P2P** berhasil dibangun menggunakan Python murni tanpa bergantung pada database atau server eksternal.
2. Arsitektur desentralisasi berjalan dengan baik; komunikasi terjadi langsung antar *socket* klien.
3. Penggunaan *multithreading* sangat krusial agar GUI tidak macet saat menunggu pesan masuk dari jaringan.
4. Fitur *handshake* (tukar menukar identitas saat koneksi awal) berhasil memecahkan masalah asimetri koneksi, sehingga kedua belah pihak saling mengetahui status *online* masing-masing.

2. Daftar Pustaka

[1] M. Van Steen and A.S. Tannenbaum, *Distributed System*, 4th ed., distributed-system.net, 2023.

[2] Dokumentasi Python 3.12 socket & threading.

[3] Dokumentasi customtkinter library.

[4] Source Code Project Repository: https://github.com/Cholifbima/p2p-messaging

[5] Link yt penjelasan aplikasi: https://www.youtube.com/watch?v=asnVaZT8I10