

# UNIVERSITAS GUNADARMA



## MANUAL BOOK

“Aplikasi Pengirim Pesan via TCP dan UDP”

Nama	: Rifqi Aditya Prabowo
NPM	: 51421314
Kelas	: 4IA10
Fakultas	: Teknologi Industri
Jurusan	: Teknik Informatika
Dosen	: Hendri Dwi

Ditulis Guna Melengkapi Sebagian Syarat

Mata Kuliah Pemrograman Jaringan

Universitas Gunadarma

2025

## DAFTAR ISI

DAFTAR ISI.....	ii
BAB I PENDAHULUAN.....	1
1.1    Latar Belakang.....	1
1.2    Tujuan Penelitian.....	2
BAB II TINJAUAN PUSTAKA .....	3
2.1    Jaringan Komputer .....	3
2.2    Model TCP/IP (Transmission Control Protocol/Internet Protocol).....	3
2.3    Protokol TCP dan UDP .....	4
<b>2.3.1 Transmission Control Protocol (TCP)</b> .....	4
<b>2.3.2 User Datagram Protocol (UDP)</b> .....	4
<b>2.3.3 Kekurangan Client Server</b> .....	4
2.4    Pemrograman Socket.....	5
2.5    Antarmuka Pengguna Grafis (GUI).....	5
2.6    Python.....	6
2.7    Visual Studio Code.....	6
BAB III ANALISIS DAN PEMBAHASAN .....	7
3.1    Gambaran Umum .....	7
3.2    Arsitektur Sistem .....	7
3.3    Perancangan Sistem.....	8
<b>3.3.1 Perancangan Antarmuka</b> .....	8
<b>3.3.2 Desain Alur Komunikasi</b> .....	8
3.4    Implementasi Sistem.....	8
BAB IV PENUTUP .....	12
4.1    Kesimpulan.....	12
4.2    Saran.....	12
DAFTAR PUSTAKA .....	14

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Dalam era digital yang semakin berkembang, komunikasi antarperangkat melalui jaringan komputer menjadi aspek yang sangat penting dalam berbagai sistem informasi. Salah satu fondasi utama dalam komunikasi jaringan adalah penggunaan protokol TCP (Transmission Control Protocol) dan UDP (User Datagram Protocol). Keduanya merupakan protokol komunikasi yang bekerja pada lapisan transport dalam model TCP/IP, namun memiliki karakteristik dan keunggulan masing-masing.

TCP dikenal sebagai protokol yang andal karena menyediakan koneksi yang terjamin dan memiliki mekanisme kontrol kesalahan, sedangkan UDP menawarkan komunikasi yang lebih cepat tanpa menjamin keandalan, sehingga cocok untuk kebutuhan real-time seperti video streaming atau gaming. Pemahaman mendalam terhadap kedua protokol ini menjadi penting bagi pengembang sistem yang ingin merancang aplikasi berbasis jaringan yang optimal.

Untuk memperdalam pemahaman tersebut, dikembangkanlah sebuah aplikasi sederhana yang mengimplementasikan komunikasi client-server menggunakan protokol TCP dan UDP. Aplikasi ini dilengkapi dengan antarmuka pengguna (GUI) pada sisi client untuk mempermudah interaksi, serta fitur pencatatan (logging) pada sisi server untuk merekam setiap pesan yang diterima. Melalui aplikasi ini, pengguna dapat mengamati secara langsung perbedaan antara komunikasi berbasis koneksi dan tanpa koneksi, serta memahami bagaimana data dikirim dan diterima dalam jaringan komputer.

Pengembangan aplikasi ini tidak hanya bermanfaat dalam konteks akademik sebagai media pembelajaran, tetapi juga dapat dijadikan dasar dalam pengembangan sistem komunikasi jaringan yang lebih kompleks di masa

mendatang.

## **1.2 Tujuan Penelitian**

Penelitian ini dilakukan sebagai bagian dari tugas mata kuliah Pemrograman Jaringan. Tujuan dari penelitian ini adalah untuk membangun sebuah aplikasi komunikasi client-server yang memanfaatkan protokol TCP dan UDP sebagai sarana untuk memahami perbedaan karakteristik kedua protokol tersebut dalam proses pengiriman data melalui jaringan komputer. Aplikasi ini dirancang dengan dilengkapi antarmuka pengguna grafis (GUI) pada sisi client guna mempermudah pengguna dalam memilih protokol dan mengirimkan pesan ke server secara interaktif. Selain itu, pada sisi server juga diterapkan fitur pencatatan (logging) terhadap setiap pesan yang diterima dari client melalui kedua jenis protokol tersebut, sehingga proses komunikasi dapat terdokumentasi dengan baik. Melalui pengembangan aplikasi ini, diharapkan pengguna atau mahasiswa dapat memperoleh media simulasi yang sederhana dan efektif untuk mempelajari konsep dasar pemrograman jaringan, khususnya yang berkaitan dengan protokol transport dan pemrograman socket.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Jaringan Komputer**

Jaringan komputer adalah suatu sistem yang terdiri dari dua atau lebih komputer yang terhubung satu sama lain untuk berbagi sumber daya, seperti printer, file, atau koneksi internet. Komunikasi dalam jaringan komputer memungkinkan pertukaran data secara efisien dan cepat, baik dalam ruang lingkup lokal (LAN) maupun global (WAN). Dalam era modern saat ini, jaringan komputer menjadi komponen penting dalam hampir semua sektor, termasuk pendidikan, pemerintahan, bisnis, dan industri. Dengan jaringan komputer, proses kolaborasi, pertukaran informasi, dan pengelolaan data dapat dilakukan dengan lebih terstruktur dan efektif (Tanenbaum & Wetherall, 2011).

#### **2.2 Model TCP/IP (Transmission Control Protocol/Internet Protocol)**

Model TCP/IP (Transmission Control Protocol/Internet Protocol) merupakan arsitektur dasar yang digunakan untuk menjelaskan cara kerja jaringan komputer modern. Model ini terdiri dari empat lapisan, yaitu Application Layer, Transport Layer, Internet Layer, dan Network Access Layer. Setiap lapisan memiliki fungsi spesifik dan saling berinteraksi untuk mendukung proses komunikasi data antar perangkat. Model TCP/IP telah menjadi standar utama dalam pengembangan aplikasi jaringan karena bersifat terbuka, fleksibel, dan dapat diimplementasikan pada berbagai platform dan sistem operasi (Kurose & Ross, 2017).

Pada lapisan Transport, terdapat dua protokol utama yang digunakan secara luas, yaitu TCP dan UDP. Kedua protokol ini memiliki peran penting dalam pengelolaan transmisi data antar host. Pemilihan protokol yang tepat bergantung pada kebutuhan aplikasi, apakah membutuhkan kecepatan tinggi, keandalan, atau keduanya.

## **2.3 Protokol TCP dan UDP**

### **2.3.1 Transmission Control Protocol (TCP)**

TCP adalah protokol berbasis koneksi (connection-oriented) yang memastikan data yang dikirim dari pengirim diterima secara utuh oleh penerima. TCP menggunakan mekanisme three-way handshake untuk membentuk koneksi awal antara dua perangkat, serta menyediakan fitur seperti pengurutan paket, kontrol aliran, dan deteksi kesalahan. Dengan adanya fitur tersebut, TCP cocok digunakan untuk aplikasi yang memerlukan keandalan tinggi seperti pengiriman file, email, dan aplikasi berbasis web (Forouzan, 2012).

### **2.3.2 User Datagram Protocol (UDP)**

Berbeda dengan TCP, UDP merupakan protokol tanpa koneksi (connectionless) yang tidak menjamin pengiriman data secara berurutan dan lengkap. Meskipun demikian, UDP memiliki keunggulan dalam hal kecepatan dan efisiensi, karena tidak memerlukan overhead tambahan seperti pada TCP. Protokol ini sering digunakan untuk aplikasi real-time seperti video conferencing, voice over IP (VoIP), dan game online, di mana kecepatan pengiriman lebih penting daripada keandalan data (Comer, 2018).

Perbandingan antara TCP dan UDP sering kali dijadikan dasar dalam menentukan strategi komunikasi data dalam pengembangan aplikasi jaringan. Oleh karena itu, pemahaman terhadap kelebihan dan kekurangan masing-masing protokol sangat penting bagi para pengembang sistem.

### **2.3.3 Kekurangan Client Server**

Kekurangan dari client server yaitu diantaranya :

1. Diperlukan administrator jaringan yang profesional untuk mengelola, memelihara, dan mengatasi masalah pada jaringan.
2. Server harus memiliki spesifikasi yang kuat untuk menangani beban kerja dan jumlah pengguna yang banyak.
3. Software khusus diperlukan untuk mempermudah manajemen dan konfigurasi jaringan.

4. Biaya untuk memelihara dan mengoperasikan jaringan client-server bisa lebih tinggi dibandingkan dengan model jaringan lain.
5. Jika server mengalami gangguan, seluruh data dan sumber daya di server

## **2.4 Pemrograman Socket**

Socket adalah antarmuka yang digunakan dalam pemrograman jaringan untuk menjembatani komunikasi antara dua perangkat yang terhubung dalam jaringan. Dengan menggunakan socket, pengembang dapat membangun aplikasi client-server yang mampu mengirim dan menerima data melalui jaringan menggunakan protokol TCP maupun UDP.

Dalam pemrograman socket, terdapat dua jenis socket utama: TCP socket dan UDP socket. TCP socket bekerja dengan membentuk koneksi terlebih dahulu sebelum melakukan pengiriman data, sedangkan UDP socket dapat langsung mengirimkan data tanpa koneksi sebelumnya. Implementasi socket biasanya dilakukan dengan menggunakan bahasa pemrograman seperti Python, Java, atau C/C++, yang telah menyediakan pustaka atau modul socket secara bawaan (Donahoo & Calvert, 2009).

Socket menjadi dasar dari banyak aplikasi jaringan yang ada saat ini, mulai dari web browser, email client, hingga sistem chatting. Dengan memahami cara kerja socket, pengembang dapat lebih mudah mengontrol bagaimana data dikirim, diterima, dan ditangani oleh sistem.

## **2.5 Antarmuka Pengguna Grafis (GUI)**

Antarmuka Pengguna Grafis atau GUI (Graphical User Interface) merupakan elemen penting dalam pengembangan aplikasi modern. GUI memungkinkan pengguna berinteraksi dengan aplikasi melalui elemen visual seperti tombol, menu, teks, dan jendela, sehingga lebih mudah digunakan dibandingkan antarmuka berbasis teks (command-line interface).

Dalam konteks aplikasi jaringan, penggunaan GUI pada sisi client dapat meningkatkan pengalaman pengguna serta menyederhanakan proses komunikasi

data, seperti memilih protokol, mengetik pesan, dan melihat hasil respon dari server. Dengan demikian, aplikasi menjadi lebih interaktif dan tidak hanya terbatas pada pengguna yang memahami perintah terminal (Shneiderman et al., 2016).

## **2.6 Python**

Python adalah bahasa pemrograman tingkat tinggi yang bersifat dinamis, mudah dibaca, dan mudah dipelajari. Python pertama kali dikembangkan oleh Guido van Rossum pada tahun 1991 dengan tujuan menciptakan bahasa yang sederhana namun tetap memiliki kemampuan yang kuat untuk pengembangan perangkat lunak (Van Rossum & Drake, 2009).

Python memiliki sintaks yang bersih dan ekspresif, yang membuatnya cocok untuk pemula maupun pengembang berpengalaman. Bahasa ini mendukung berbagai paradigma pemrograman, termasuk pemrograman berorientasi objek, prosedural, dan fungsional.

## **2.7 Visual Studio Code**

Visual Studio Code merupakan aplikasi penyunting kode sumber gratis dari Microsoft, mendukung sebagian besar bahasa pemrograman, termasuk Node.js, JavaScript, dan TypeScript. Dengan beragam fitur canggih, Visual Studio Code menonjol di antara software editor sejenis, meraih popularitas dalam berbagai sektor pengembangan, mulai dari pembuatan aplikasi mobile (Android dan iOS), pengembangan situs web, hingga implementasi machine learning. Fitur kunci seperti penyorotan sintaksis, penyelesaian kode, kutipan kode, refaktorisasi kode, pengawakutuan, dan integrasi dengan Git, membuatnya menjadi pilihan utama bagi pengembang. Visual Studio Code juga menawarkan ekstensi dan ekosistem yang luas, memastikan kompatibilitas tinggi dengan berbagai bahasa pemrograman dan runtime environment seperti Python, PHP, .NET, dan Java, memberikan fleksibilitas dan produktivitas dalam pengembangan perangkat lunak.



## **BAB III**

### **ANALISIS DAN PEMBAHASAN**

#### **3.1 Gambaran Umum**

Sistem yang dikembangkan merupakan aplikasi komunikasi berbasis jaringan yang memungkinkan pengguna (client) mengirimkan pesan ke server melalui dua pilihan protokol transport, yaitu TCP dan UDP. Sistem ini terdiri dari dua komponen utama, yaitu aplikasi client dengan antarmuka grafis (GUI) dan aplikasi server berbasis command-line interface (CLI). Server akan menerima pesan dari client, mencatatnya ke dalam log, dan mengirimkan balasan sebagai tanda bahwa pesan telah diterima.

Tujuan utama dari sistem ini adalah memberikan simulasi praktis dalam memahami perbedaan karakteristik dan perilaku antara komunikasi TCP dan UDP dalam pemrograman jaringan.

#### **3.2 Arsitektur Sistem**

Sistem ini menggunakan arsitektur client-server sederhana. Berikut adalah deskripsi masing-masing komponennya:

- Client: Aplikasi dengan GUI yang memungkinkan pengguna mengetik pesan, memilih jenis protokol (TCP atau UDP), dan mengirim pesan ke server.
- Server: Aplikasi yang berjalan terus-menerus dan dapat menerima pesan dari banyak client melalui dua port berbeda (untuk TCP dan UDP), serta membalas pesan tersebut.

### **3.3 Perancangan Sistem**

#### **3.3.1 Perancangan Antarmuka**

Antarmuka aplikasi client dirancang agar mudah digunakan oleh pengguna. Elemen-elemen utama yang ditampilkan meliputi:

1. Kolom input pesan
2. Pilihan protokol (radio button TCP/UDP)
3. Tombol kirim
4. Area log percakapan

GUI dirancang menggunakan pustaka tkinter pada Python karena sederhana dan mendukung pengembangan cepat untuk desktop.

#### **3.3.2 Desain Alur Komunikasi**

### **3.4 Implementasi Sistem**

Implementasi sistem dilakukan menggunakan bahasa pemrograman Python karena memiliki pustaka standar yang mendukung komunikasi jaringan melalui protokol TCP dan UDP, serta pustaka antarmuka grafis sederhana seperti tkinter. Sistem ini terdiri dari dua komponen utama, yaitu aplikasi server dan aplikasi client, yang masing-masing memiliki peran dan arsitektur khusus.

Aplikasi server dikembangkan dalam dua bagian terpisah, yaitu server TCP dan server UDP. Server TCP menggunakan socket bertipe `SOCK_STREAM`, yang menyediakan koneksi yang andal dan menjamin integritas data melalui proses three-way handshake. Dalam implementasinya, server TCP melakukan binding ke alamat IP lokal dan port tertentu, kemudian menunggu koneksi dari client. Setelah koneksi terbentuk, server menerima pesan yang dikirim oleh client, mencetak isi pesan ke layar, dan membalas pesan sebagai bentuk konfirmasi.

```

gui_client.py U  tcp_server.py U X  udp_server.py U
tcp_udp-client-server > tcp_server.py > ...
1  import socket
2  import threading
3
4  HOST = '127.0.0.1'
5  PORT = 65432
6
7  def handle_client(conn, addr):
8      print(f"[TCP Server] Connected by {addr}")
9      with conn:
10         while True:
11             data = conn.recv(1024)
12             if not data:
13                 break
14             message = data.decode()
15             print(f"[TCP Server] Received from {addr}: {message}")
16             conn.sendall(b"[TCP Server] Pesan diterima")
17
18             # Simpan ke log file
19             with open("log_tcp.txt", "a") as log_file:
20                 log_file.write(f"{addr} - {message}\n")
21
22  with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
23      s.bind((HOST, PORT))
24      s.listen()
25      print(f"[TCP Server] Listening on {HOST}:{PORT}")
26
27      while True:
28          conn, addr = s.accept()
29          thread = threading.Thread(target=handle_client, args=(conn, addr))
30          thread.start()
31

```

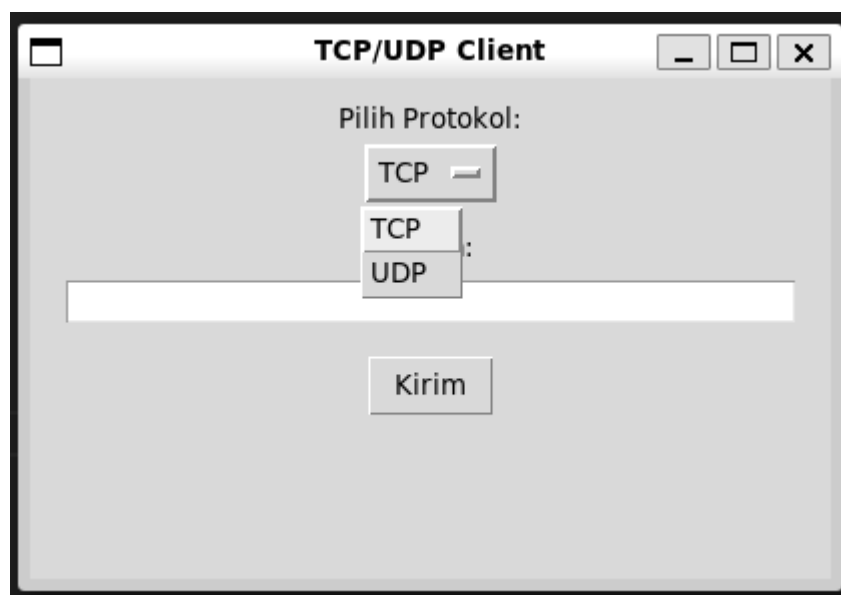
Sebaliknya, server UDP menggunakan socket bertipe SOCK\_DGRAM, yang tidak memerlukan koneksi permanen. Server ini tetap melakukan binding ke port tertentu, namun tidak memelihara status koneksi dengan client. Server akan langsung menerima datagram dari client, memproses isi pesan, dan mengirimkan balasan ke alamat pengirim. Meskipun komunikasi UDP lebih cepat, ia tidak menjamin bahwa data akan sampai secara utuh atau berurutan.

```

tcp_server.py  client.py  udp_server.py X  gui_client.py
tcp_udp_chat > udp_server.py > ...
1  import socket
2
3  HOST = '127.0.0.1'
4  PORT = 65433
5
6  with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
7      s.bind((HOST, PORT))
8      print(f"[UDP Server] Listening on {HOST}:{PORT}")
9
10     while True:
11         data, addr = s.recvfrom(1024)
12         message = data.decode()
13         print(f"[UDP Server] Received from {addr}: {message}")
14
15         # Simpan ke log file
16         with open("log_udp.txt", "a") as log_file:
17             log_file.write(f"{addr} - {message}\n")

```

Untuk sisi client, implementasi dilakukan dengan menggunakan tkinter untuk membangun antarmuka pengguna (GUI). Antarmuka ini terdiri dari kolom teks untuk input pesan, dua radio button untuk memilih protokol komunikasi (TCP atau UDP), tombol kirim, dan area log untuk menampilkan hasil komunikasi dengan server. Ketika pengguna menekan tombol kirim, aplikasi akan memeriksa protokol yang dipilih, kemudian membuat koneksi ke server sesuai dengan jenis protokol tersebut dan mengirimkan pesan.



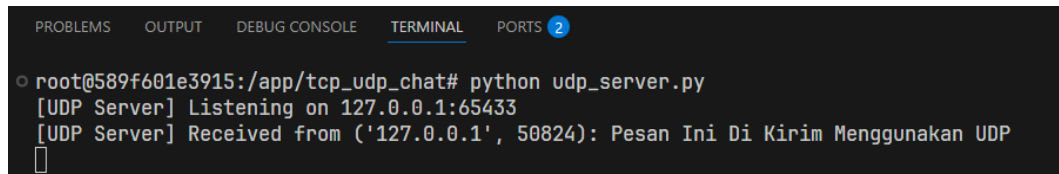
Jika protokol yang dipilih adalah TCP, client akan membuat socket bertipe stream, terhubung ke server melalui port TCP, dan mengirimkan data. Setelah itu, client menunggu balasan dari server, dan menampilkan hasilnya di area log.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS 2
root@589f601e3915:/app/tcp_udp_chat# python tcp_server.py
[TCP Server] Listening on 127.0.0.1:65432
[TCP Server] Connected by ('127.0.0.1', 57402)
[TCP Server] Received from ('127.0.0.1', 57402): Pesan Ini Di Kirim Menggunakan TCP

```

Sebaliknya, jika protokol UDP yang dipilih, client akan mengirimkan datagram langsung ke server melalui port UDP, kemudian menerima balasan tanpa perlu membentuk koneksi terlebih dahulu.

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active and underlined), and 'PORTS' with a blue circle containing the number '2'. The terminal shows the following text: a prompt 'root@589f601e3915:/app/tcp\_udp\_chat#' followed by the command 'python udp\_server.py'. Below this, it says '[UDP Server] Listening on 127.0.0.1:65433'. Then, it shows a received message: '[UDP Server] Received from ('127.0.0.1', 50824): Pesan Ini Di Kirim Menggunakan UDP'. A cursor is visible on the line following the message.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2
root@589f601e3915:/app/tcp_udp_chat# python udp_server.py
[UDP Server] Listening on 127.0.0.1:65433
[UDP Server] Received from ('127.0.0.1', 50824): Pesan Ini Di Kirim Menggunakan UDP
█
```

Untuk memudahkan pengembangan dan pengujian, struktur direktori proyek dibuat sederhana, terdiri dari tiga berkas utama: `gui_client.py`, `server_tcp.py`, dan `server_udp.py`. Berkas `client.py` menangani seluruh antarmuka pengguna dan logika pemilihan protokol. Sementara `server_tcp.py` dan `server_udp.py` masing-masing berfungsi sebagai server untuk protokol TCP dan UDP.

Secara keseluruhan, implementasi sistem ini bertujuan memberikan pemahaman mendalam kepada pengembang mengenai perbedaan teknis dan konseptual antara komunikasi menggunakan TCP dan UDP, serta penerapan nyata dari pemrograman socket dalam jaringan komputer.

## **BAB IV**

### **PENUTUP**

#### **4.1 Kesimpulan**

Berdasarkan hasil perancangan dan implementasi yang telah dilakukan, dapat disimpulkan bahwa sistem komunikasi client-server menggunakan protokol TCP dan UDP berhasil dibangun dengan baik dan sesuai dengan tujuan awal pengembangan. Sistem ini mampu menunjukkan perbedaan mendasar antara kedua jenis protokol tersebut secara nyata. Protokol TCP memberikan keandalan dalam komunikasi karena menjamin bahwa data yang dikirim akan diterima secara utuh dan berurutan. Hal ini disebabkan oleh adanya mekanisme pembentukan koneksi dan verifikasi paket yang menyertainya. Sementara itu, protokol UDP memungkinkan pengiriman data yang lebih cepat karena tidak memerlukan proses koneksi, namun dengan konsekuensi bahwa data yang diterima tidak selalu dijamin keutuhannya.

Selain itu, implementasi antarmuka pengguna (GUI) pada sisi client menggunakan pustaka tkinter terbukti efektif dalam memudahkan pengguna dalam melakukan pengiriman pesan, memilih jenis protokol, serta melihat hasil komunikasi secara langsung. Dengan adanya GUI, interaksi pengguna menjadi lebih intuitif dan respons sistem dapat langsung diamati, baik ketika menggunakan TCP maupun UDP. Secara keseluruhan, aplikasi ini memberikan gambaran praktis tentang bagaimana komunikasi jaringan bekerja dalam dunia nyata, sekaligus meningkatkan pemahaman mahasiswa terhadap konsep dasar pemrograman jaringan.

#### **4.2 Saran**

Sebagai tindak lanjut dari sistem yang telah dibangun, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan selanjutnya. Pertama, komunikasi antara client dan server dapat ditingkatkan menjadi dua arah secara

real-time menggunakan konsep multithreading, sehingga client dan server dapat saling bertukar pesan secara bersamaan tanpa harus menunggu proses sebelumnya selesai. Kedua, aspek keamanan komunikasi dapat ditambahkan melalui penerapan metode enkripsi pesan agar data yang dikirim tidak mudah disadap atau dimanipulasi oleh pihak yang tidak berwenang.

Selanjutnya, sistem dapat diperluas agar mampu berjalan tidak hanya di jaringan lokal, tetapi juga pada jaringan publik seperti internet, dengan memberikan konfigurasi dinamis untuk alamat IP dan port. Penambahan fitur log aktivitas baik di sisi server maupun client, yang disimpan ke dalam file atau database, juga dapat memberikan manfaat dalam pemantauan dan analisis penggunaan sistem. Terakhir, untuk meningkatkan kenyamanan pengguna, antarmuka GUI dapat dikembangkan lebih lanjut menggunakan pustaka lain seperti PyQt atau Kivy yang menyediakan tampilan lebih modern dan responsif.

## DAFTAR PUSTAKA

- Comer, D. E. (2018). *Internetworking with TCP/IP Volume One* (6th ed.). Pearson.
- Donahoo, M. J., & Calvert, K. L. (2009). *TCP/IP Sockets in Python: Practical Guide for Programmers* (2nd ed.). Morgan Kaufmann.
- Forouzan, B. A. (2012). *Data Communications and Networking* (5th ed.). McGraw-Hill.
- Kurose, J. F., & Ross, K. W. (2017). *Computer Networking: A Top-Down Approach* (7th ed.). Pearson.
- Prasetya, D., & Riyadi, E. (2020). Simulasi client server dengan Java Socket. *Jurnal Teknologi dan Sistem Komputer*, 8(2), 115–120.  
<https://doi.org/10.14710/jtsiskom.8.2.2020.115-120>
- Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., & Elmqvist, N. (2016). *Designing the User Interface* (6th ed.). Pearson.
- Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer Networks* (5th ed.). Pearson.
- Yulianto, R., Handayani, T., & Saputra, R. (2021). Pengembangan aplikasi jaringan berbasis GUI untuk pembelajaran TCP/UDP. *Jurnal Ilmiah Teknik Informatika*, 14(1), 23–29.