Pengembangan Aplikasi Komunikasi Real-time Menggunakan Model Client-Server dan Protokol TCP

Linna Oktaviana Sari[1] Anang Arif Hibaturrahman[2] Muhammad Rizki Dalfi[3] Qorri Aquina Adisty[4] Rialdi[5] Siti Juhisa[6]
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Riau, Kota Pekanbaru, Provinsi Riau, Indonesia[1,2,3,4,5]
Email: linnaoasari@lecturer.unri.ac.id[1] anang.arif6365@student.unri.ac.id[2] muhammad.rizki3405@student.unri.ac.id[3] qorri.aquina1517@student.unri.ac.id[4] rialdi6366@student.unri.ac.id[5] siti.juhisa0677@student.unri.ac.id[6]

Abstrak
Penelitian ini membahas pengembangan aplikasi komunikasi real-time dengan menggunakan model client-server dan protokol TCP. Aplikasi ini bertujuan untuk memfasilitasi pertukaran pesan instan antara klien yang terhubung dan server pusat dalam lingkungan jaringan terdistribusi. Penelitian melibatkan implementasi, optimasi, dan evaluasi kinerja aplikasi dalam hal responsivitas dan keandalan. Melalui analisis menyeluruh terhadap proses komunikasi, termasuk pemanfaatan bandwidth dan pertimbangan latensi, penelitian ini memberikan wawasan terhadap aspek praktis dari penerapan sistem komunikasi real-time. Hasil penelitian menunjukkan efektivitas dan efisiensi aplikasi yang dikembangkan, memberikan pemahaman mendalam mengenai potensi model client-server dan protokol TCP dalam menciptakan pengalaman komunikasi real-time yang lancar dan dapat diandalkan. Penelitian ini memberikan kontribusi pada pemahaman lebih luas tentang sistem komunikasi berjaringan dan memberikan dasar untuk pengembangan di masa depan dalam ranah aplikasi real-time.
Kata Kunci: Aplikasi Komunikasi Real-Time, Model Client-Server

PENDAHULUAN

Dalam era di mana konektivitas dan komunikasi instan menjadi landasan utama perkembangan teknologi informasi, pengembangan aplikasi komunikasi real-time telah menjadi fokus penelitian yang signifikan. Aplikasi semacam itu memungkinkan pertukaran pesan secara cepat dan efisien antara pengguna yang terhubung melalui jaringan. Salah satu model yang umum digunakan untuk membangun aplikasi semacam itu adalah model client-server, yang memungkinkan distribusi tugas antara perangkat pengguna (client) dan server pusat. Penelitian ini difokuskan pada pengembangan aplikasi komunikasi real-time menggunakan model client-server dan protokol TCP (Transmission Control Protocol). Protokol TCP, sebagai bagian integral dari suite protokol Internet, telah menjadi standar de facto dalam menyediakan koneksi yang andal dan terjamin di lingkungan jaringan. Model client-server, dengan pemisahan peran antara klien dan server, memberikan fleksibilitas dan skabilitas yang dibutuhkan untuk memenuhi kebutuhan aplikasi komunikasi real-time yang kompleks.

Latar belakang pengembangan aplikasi semacam itu melibatkan tantangan dalam menangani aspek-aspek kritis seperti latensi, keamanan, dan ketersediaan. Oleh karena itu, penelitian ini bertujuan untuk mengatasi aspek-aspek tersebut melalui implementasi yang cermat, optimasi, dan evaluasi kinerja aplikasi komunikasi real-time. Analisis mendalam tentang bagaimana model client-server dan protokol TCP dapat secara efektif mendukung komunikasi real-time menjadi pusat dari upaya ini. Melalui penelusuran literatur yang cermat dan eksperimen yang terkontrol, penelitian ini diharapkan memberikan wawasan yang berharga terhadap kemungkinan penerapan aplikasi komunikasi real-time dalam konteks

model client-server dan protokol TCP. Hasil penelitian ini diharapkan dapat memberikan kontribusi pada pemahaman lebih lanjut tentang desain aplikasi komunikasi yang responsif dan andal dalam lingkungan jaringan yang terdistribusi.

Landasan Teori

Komunikasi Real-time

Komunikasi real-time adalah paradigma komunikasi di mana pertukaran informasi terjadi dalam waktu yang sangat singkat atau hampir instan. Ini penting untuk aplikasi yang menuntut respon cepat, seperti sistem pesan instan, video konferensi, dan sistem kendali real-time. Komunikasi real-time memiliki beberapa karakteristik utama:

1. Latensi Rendah: Komunikasi real-time meminimalkan waktu antara pengiriman dan penerimaan informasi.
2. Respons Cepat: Aplikasi yang menggunakan komunikasi real-time memberikan respons dengan cepat terhadap permintaan atau perubahan kondisi.
3. Keandalan Tinggi: Data harus dikirim dan diterima dengan keandalan tinggi untuk memastikan integritas informasi. Implementasi komunikasi real-time memerlukan desain sistem yang efisien, manajemen bandwidth yang baik, dan pengelolaan sumber daya yang cermat.

Model Client-Server

Model client-server adalah arsitektur distribusi yang melibatkan dua komponen utama: client (klien) dan server. Dalam model ini, klien membuat permintaan kepada server, dan server memberikan respons atau layanan. Beberapa karakteristik model client-server meliputi:

1. Pemisahan Tugas: Klien dan server memiliki peran yang terpisah, dengan klien menginisiasi permintaan dan server menyediakan layanan atau informasi.
2. Skabilitas: Model ini memungkinkan penambahan klien tanpa mempengaruhi kinerja server, sehingga mendukung skabilitas yang baik.
3. Manajemen Sumber Daya: Server bertanggung jawab atas manajemen sumber daya dan pemrosesan data, sementara klien bertanggung jawab atas antarmuka pengguna. Model ini sangat umum dalam pengembangan aplikasi jaringan, termasuk aplikasi web, basis data terpusat, dan aplikasi komunikasi.

Protokol TCP (Transmission Control Protocol)

Protokol TCP adalah protokol transport yang terdapat pada lapisan transport dalam model referensi OSI. Protokol ini menyediakan koneksi yang andal dan terjamin melalui beberapa mekanisme, termasuk pemecahan paket, pengendalian aliran, dan pengelolaan koneksi. Beberapa fitur utama protokol TCP mencakup:

1. Koneksi Terestablish: Protokol TCP membuka, mempertahankan, dan menutup koneksi dengan menggunakan tiga langkah tangan (three-way handshake).
2. Pemecahan Paket: Data dikirim dalam bentuk paket-paket kecil untuk memastikan pengiriman yang teratur dan dapat diandalkan.
3. Pengendalian Aliran: Protokol ini mengatur laju pengiriman data untuk mencegah kelebihan beban atau kehilangan data.
4. TCP digunakan secara luas di Internet untuk aplikasi yang membutuhkan koneksi andal, seperti transfer file, surat elektronik, dan aplikasi web.
5. Dengan memahami landasan teori ini secara mendalam, pengembang dapat merancang aplikasi komunikasi real-time yang efisien, handal, dan responsif menggunakan model client-server dan protokol TCP.

Implementasi Pemrograman

Dalam penyusunan aplikasi komunikasi real-time dengan menggunakan model client-server dan protokol TCP melalui Command Prompt, langkah-langkah implementasi pemrograman menjadi esensi utama. Bab ini memperkenalkan serangkaian tindakan teknis yang diambil untuk mengimplementasikan konsep teoretis ke dalam suatu solusi praktis yang dapat dijalankan melalui Command Prompt. Implementasi ini berfokus pada kejelasan interaksi melalui antarmuka teks, memungkinkan pengguna berkomunikasi secara langsung dengan aplikasi melalui perintah-perintah di Command Prompt. Setiap langkah ditekankan pada pembentukan dasar aplikasi yang sederhana namun efektif, dengan memperhatikan aspek-aspek penting seperti manajemen koneksi, pengiriman pesan real-time, dan keandalan komunikasi melalui protokol TCP. Langkah-langkah implementasi diarahkan pada pembuatan program client dan server yang dapat berkomunikasi secara real-time melalui Command Prompt. Fokus Pengaplikasian bagaimana protokol TCP dapat diintegrasikan ke dalam struktur program, memastikan pengiriman dan penerimaan data yang andal dalam waktu yang sangat singkat. Dibutuhkan 2 file berikut

```
// file tcpserver.java

import java.io.*;

import java.net.*;

import java.util.Scanner;


public class tcpserver {

  public static void main(String[] args) throws Exception {

    ServerSocket serverSocket = new ServerSocket(4444);

    System.out.println("Server started. Listening on port 4444...");


    while (true) {

      Socket clientSocket = serverSocket.accept();

      System.out.println("Client connected from: " + clientSocket.getInetAddress());


      Thread clientThread = new Thread(new ClientHandler(clientSocket));

      clientThread.start();

    }

  }

}


class ClientHandler implements Runnable {

  private Socket clientSocket;

  private BufferedReader br;

  private BufferedWriter bw;
```

```
public ClientHandler(Socket clientSocket) {
  this.clientSocket = clientSocket;
  try {
    br = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
    bw = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));
  } catch (IOException e) {
    e.printStackTrace();
  }
}

@Override
public void run() {
  try {
    Scanner scanner = new Scanner(System.in);
    String message;

    while (true) {
      message = br.readLine();
      if (message == null || message.equalsIgnoreCase("bye")) {
        break;
      }
      System.out.println("Client: " + message);

      // Response to the client
      String response = "Server received your message: " + message;
      bw.write(response);
      bw.newLine();
      bw.flush();

      System.out.print("Server: ");
      message = scanner.nextLine();
      bw.write(message);
      bw.newLine();
      bw.flush();
    }
```

```
      clientSocket.close();
      br.close();
      bw.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}

//file tcpclient.java
import java.io.*;
import java.net.*;

public class tcpclient {
  public static void main(String[] args) throws Exception {
    Socket clientSocket = new Socket("localhost", 4444);
    System.out.println("Connected to server. Start chatting!");

    Thread serverListener = new Thread(new ServerListener(clientSocket));
    serverListener.start();

    try {
      InputStreamReader isr = new InputStreamReader(System.in);
      BufferedReader br = new BufferedReader(isr);
      OutputStreamWriter osw = new OutputStreamWriter(clientSocket.getOutputStream());
      BufferedWriter bw = new BufferedWriter(osw);

      String message;
      while (true) {
        System.out.print("Client: ");
        message = br.readLine();
        bw.write(message);
        bw.newLine();
        bw.flush();
        if (message.equalsIgnoreCase("bye")) {
          break;
```

```
        }
      }

      clientSocket.close();
      isr.close();
      osw.close();
      br.close();
      bw.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}

class ServerListener implements Runnable {
  private Socket clientSocket;
  private BufferedReader br;

  public ServerListener(Socket clientSocket) {
    this.clientSocket = clientSocket;
    try {
      br = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  @Override
  public void run() {
    try {
      String message;

      while (true) {
        message = br.readLine();
        if (message == null || message.equalsIgnoreCase("bye")) {
          break;
```

```
        }
        System.out.println("Server: " + message);
      }

      br.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

File server harus dicompile terlebih dahulu, kemudian file client akan dicompile setelah server. Yang masing masing file harus dijalankan pada prompt yang berbeda sehingga akan menghasilkan output berupa komunikasi dua arah seperti konsep aplikasi chat.

KESIMPULAN

Beberapa hal yang dapat disimpulkan dari tulisan ini: Keberhasilan Implementasi Model Client-Server: Implementasi model client-server pada aplikasi komunikasi real-time melalui Command Prompt telah berhasil membentuk arsitektur yang memungkinkan klien dan server berinteraksi secara efisien. Pemisahan peran antara klien dan server dapat dijalankan dengan lancar, memungkinkan pengguna untuk mengirim dan menerima pesan real-time melalui antarmuka teks. Keandalan Komunikasi dengan Protokol TCP: Integrasi protokol TCP dalam implementasi aplikasi memastikan keandalan dan konsistensi komunikasi. Protokol TCP memungkinkan pemecahan paket, pengelolaan koneksi yang stabil, dan pengendalian aliran data, sehingga memastikan pesan-pesan dapat dikirim dan diterima secara teratur tanpa kehilangan data. Responsivitas dan Keterbacaan Melalui Command Prompt: Antarmuka Command Prompt memberikan responsivitas yang tinggi dan keterbacaan yang baik bagi pengguna. Pesan real-time dapat dengan cepat dimasukkan dan ditanggapi melalui perintah-perintah sederhana, menciptakan pengalaman komunikasi yang efisien dan langsung.

DAFTAR PUSTAKA

Forouzan, B. A. (2013). TCP/IP Protocol Suite (4th ed.). McGraw-Hill.
Kerrisk, M. (2010). The Linux Programming Interface. No Starch Press.
Stevens, W. R., Fenner, B., & Rudoff, A. M. (2004). UNIX Network Programming, Volume 1: The Sockets Networking API (3rd ed.). Addison-Wesley.
Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer Networks (5th ed.). Pearson.