Praktikum Minggu ke-14

Socket Programming

A. TUJUAN PEMBELAJARAN

1. Siswa memahami konsep aplikasi client server di jaringan.
2. Mahasiswa memahami konsep pemprograman socket dasar.
3. Mahasiswa mampu membangun program socket sederhana dg single thread

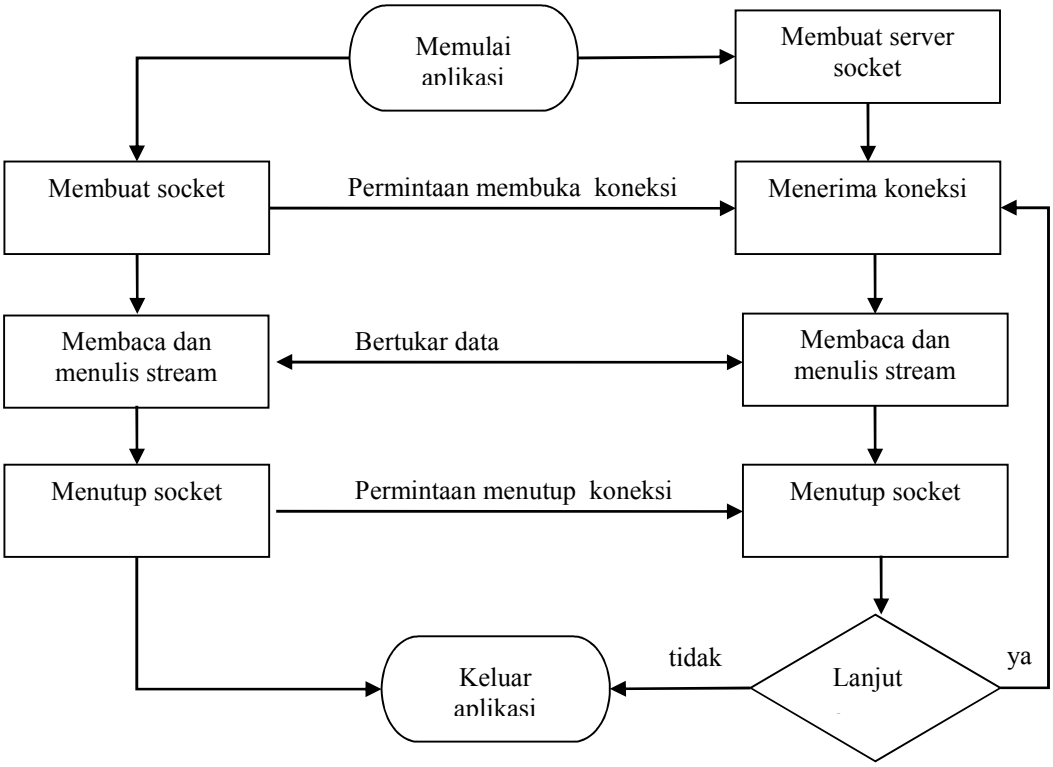
B. DASAR TEORI

Model yang umum diterapkan dalam jaringan komputer adalah model client/server. Konsepnya sederhana, sebuah aplikasi client melakukan permintaan untuk suatu layanan (service) informasi atau mengirim sebuah perintah ke suatu aplikasi server. Aplikasi server akan menerima permintaan dari client, kemudian memproses berdasarkan permintaan tersebut. Dari hasil pemrosesan yang sudah dilakukan, aplikasi server akan mengembalikan hasil pemrosesan tersebut ke aplikasi client.

Sehingga pada prinsipnya aplikasi server dalam status menunggu (*listen*) permintaan dari aplikasi client, sedangkan client mencoba membuat koneksi (*connect*) ke server.

Aplikasi client melakukan koneksi ke server melalui sebuah alamat *socket.* Alamat socket adalah kombinasi dari alamat IP dan nomor port. Contoh alamat socket adalah 192.168.1.30:80, dimana nomer 80 adalah nomer portnya. Jika alamat IP diibaratkan sebuah nomer telepon, maka nomer port adalah nomer ekstensinya.

Suatu proses yang hendak berkomunikasi dengan proses lain lewat mekanisme socket haruslah mengikatkan dirinya dengan salah satu port. Tahap proses ini disebut dengan *binding* .



Gambar 10.1 menunjukkan bagan interkoneksi server dan client.

Model pemrograman client-server menggunakan dari dua macam koneksi pada layer 4/ layer transport OSI, yaitu connection oriented dan connectionless oriented. Untuk connection oriented menggunakan protokol TCP (Transmission Control Protocol). Sedangkan connectionless oriented menggunakan UDP (User Datagram Protocol).

Pada praktikum ini, kita akan mencoba membangun sebuah aplikasi cliet-server sederhana dengan menggunakan bahasa pemrograman Java. Pada J2SE telah disediakan paket java.net yang berisi kelaskelasdan interface yang menyediakan API (Application Programming Interface) level rendah (Socket, ServerSocket,DatagramSocket) dan level tinggi (URL, URLConnection). Berikut adalah contoh dari kelas Socket berikut method dan eksepsi error yang tersedia (Untuk lebih jelasnya anda bias membacanya di Java documentation) :

- Socket(String host, int port, InetAddress localAddr, intlocalPort);

  - membuat sebuah socket dan mengkoneksikannya ke port yang dituju pada alamat IP yang disebutkan pada parameter address atau nama host. Selain itu juga akan dilakukan bind socket ke alamat lokal dan port lokal. (Hal ini dilakukan jika koneksi antara client dan server membutuhkan nomor port yang sudah ditentukan.

- Methods :
  - getInetAddress() : untuk mendapatkan nama host yang dituju dan alamat IPnya
  - getPort() : untuk mendapatkan nomor remote host
  - getLocalPort(): untuk mendapatkan nomor port localhost
  - getLocalAddress(): untuk mendapatkan alamat local dimana socket digunakan
  - getInputStream(): mengembalikan objek input stream dari socket
  - getOutputStream(): mengembalikan objek output stream ke socket
  - setSoTimeout(int timeout) dan getSoTimeOut() : Kedua method tersebut digunakan untuk memberi (set) dan mengambil (get) nilai opsi Socket untuk time out block (dalam milidetik) reading dari socket (SO_TIMEOUT). Jika dalam waktu timeout tidak mendapat suatu nilai maka, akan dilemparkan ke exception java.net.SocketTimeoutException. Nilai default timeoutnya adalah 0, yang berarti tanpa batas.
- Exceptions :
  - SocketException : Kelas ini merupakan kelas yang diturunkan dari kelas IOException. Kelas exception ini dipanggil atau dipicu ketika ada kegagalan dalam pemakaian socket, sebagai contoh adalah kegagalan dalam protokol TCP. Salah satu penyebabnya yang mungkin terjadi adalah ketika port yang akan digunakan sudah digunakan sebelumnya pada localhost. Penyebab yang lain adalah user tidak dapat melakukan bind ke port yang dituju. Misalnya saja, Anda ingin menggunakan port 80 untuk aplikasi Anda, namun ternyata pada komputer Anda tersebut sudah berjalan HTTP Server yang juga menggunakan port 80. Bila hal ini terjadi,

maka JVM akan melemparkan kegagalan yang ada ke kelas exception SocketException.

- BindException : Exception ini akan dipanggil ketika ada port lokal yang akan digunakan sudah terpakai oleh yang lain, atau ada kegagalan dalam permintaan untuk menggunakan alamat.
- ConnectException : Exception ini akan dipanggil ketika sebuah koneksi ditolak oleh host yang dituju, oleh karena tidak ada proses yang siap menerima data pada port yang dituju.
- NoRouteToHostException: Koneksi yang akan dibangun tidak dapat dipenuhi oleh karena melebihi waktu timeout yang tersedia atau host yang dituju tidak dapat dicapai (unreachable).

C. LISTING PROGRAM

1a. Program Socket Server sederhana

Source : http://www.tutorialspoint.com/java/java_networking.htm

```
// File Name GreetingServer.java

import java.net.*;
import java.io.*;

public class GreetingServer extends Thread
{
   private ServerSocket serverSocket;

   public GreetingServer(int port) throws IOException
   {
      serverSocket = new ServerSocket(port);
      serverSocket.setSoTimeout(10000);
   }
```

```
public void run()
{
   while(true)
   {
      try
      {
         System.out.println("Waiting for client on port " +
         serverSocket.getLocalPort() + "...");
         Socket server = serverSocket.accept();
         System.out.println("Just connected to "
               + server.getRemoteSocketAddress());
         DataInputStream in =
               new DataInputStream(server.getInputStream());
         System.out.println(in.readUTF());
         DataOutputStream out =
              new DataOutputStream(server.getOutputStream());
         out.writeUTF("Thank you for connecting to "
           + server.getLocalSocketAddress() + "\nGoodbye!");
         server.close();
      }catch(SocketTimeoutException s)
      {
         System.out.println("Socket timed out!");
         break;
      }catch(IOException e)
      {
         e.printStackTrace();
         break;
      }
   }
}
public static void main(String [] args)
{
```

```
      int port = Integer.parseInt(args[0]);
      try
      {
         Thread t = new GreetingServer(port);
         t.start();
      }catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

1b. Program Socket Client sederhana

```
// File Name GreetingClient.java

import java.net.*;
import java.io.*;

public class GreetingClient
{
   public static void main(String [] args)
   {
      String serverName = args[0];
      int port = Integer.parseInt(args[1]);
      try
      {
         System.out.println("Connecting to " + serverName +
                  " on port " + port);
         Socket client = new Socket(serverName, port);
         System.out.println("Just connected to "
                  + client.getRemoteSocketAddress());
         OutputStream outToServer = client.getOutputStream();
```

```
         DataOutputStream out = new DataOutputStream(outToServer);
         out.writeUTF("Hello from "
                     + client.getLocalSocketAddress());
         InputStream inFromServer = client.getInputStream();
         DataInputStream in =
                        new DataInputStream(inFromServer);
         System.out.println("Server says " + in.readUTF());
         client.close();
      }catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

Pertama-tama, *compile* kedua program (client dan server) kemudian jalankan program server lebih dahulu dengan perintah sbb. :

```
$ java GreetingServer 6066
Waiting for client on port 6066...
```

Kemudian jalankan program client dengan perintah berikut ini :

```
$ java GreetingClient localhost 6066
Connecting to localhost on port 6066
Just connected to localhost/127.0.0.1:6066
Server says Thank you for connecting to /127.0.0.1:6066
Goodbye!
```

2. Program Aplikasi HTTP Server sederhana

```
1. ///A Simple Web Server (WebServer.java)
2.
3. import java.io.BufferedReader;
4. import java.io.InputStreamReader;
```

```
5.  import java.io.PrintWriter;
6.  import java.net.ServerSocket;
7.  import java.net.Socket;
8.
9.  public class webServer {
10.  /**
11.   * WebServer constructor.
12.   */
13.  protected void start() {
14.    ServerSocket s;
15.    int port = 8888;
16.    System.out.println("Webserver starting up on port " +port);
17.    System.out.println("(press ctrl-c to exit)");
18.    try {
19.      // tahap bind(),membuat socket
20.      s = new ServerSocket(port);
21.    } catch (Exception e) {
22.      System.out.println("Error: " + e);
23.      return;
24.    }
25.
26.    System.out.println("Waiting for connection");
27.    for (;;) {
28.      try {
29.        // tahap listen(), menunggu koneksi
30.        Socket remote = s.accept();
31.         // tahap accept()
32.        System.out.println("Connection, sending data.");
33.        BufferedReader in = new BufferedReader(new InputStreamReader(
34.            remote.getInputStream()));
35.        PrintWriter out = new PrintWriter(remote.getOutputStream());
36.        // membaca request
37.        String str = ".";
38.        while (!str.equals(""))
39.          str = in.readLine();
40.        // Mengirim response dan mengirim HTTP headers
41.        out.println("HTTP/1.0 200 OK");
42.        out.println("Content-Type: text/html");
43.        out.println("Server: Bot");
44.        // Batas pengiriman header ditandai dengan baris kosong
45.        out.println("");
46.        // Mengirim halam HTML
47.        out.println("<H1>Selamat datang di web server percobaan</H1>");
48.        out.println("<blink>Selamat datang di web server percobaan</blink>");
49.        out.flush();
50.        remote.close();
51.      } catch (Exception e) {
```

```
52.         System.out.println("Error: " + e);
53.       }
54.     }
55.   }
56.
57.   /**
58.    * Menjalankan server
59.    *
60.    * @param args
61.    *            Command line parameters are not used.
62.    */
63.   public static void main(String args[]) {
64.     webServer ws = new webServer();
65.     ws.start();
66.   }
67. }
```

END

3a. Program Aplikasi Chat Server dengan menggunakan TCP

```
1.  import java.io.BufferedReader;
2.  import java.io.IOException;
3.  import java.io.InputStreamReader;
4.  import java.io.PrintWriter;
5.  import java.net.ServerSocket;
6.  import java.net.Socket;
7.
8.  public class TCPEchoServer {
9.
10.    private static ServerSocket servSock;
11.    private static final int PORT = 1234;
12.
13.    public static void main(String args[]) {
14.        System.out.println("Membuka socket.....\n");
15.        try {
16.            servSock = new ServerSocket(PORT);
17.        } catch (IOException e) {
18.            System.out.println("Gagal membuka port !!!");
19.            System.exit(1);
20.        }
21.        do {
22.            run();
23.        } while (true);
```

```
24.     }
25.
26.     private static void run() {
27.         Socket link = null;
28.         try {
29.
30.             link = servSock.accept();
31.             BufferedReader in = new BufferedReader(new
    InputStreamReader(link.getInputStream()));
32.             PrintWriter out = new PrintWriter(link.getOutputStream(), true);
33.             int numMessages = 0;
34.             String message = in.readLine();
35.             while (!message.equals("close")) {
36.
37.                 System.out.println("Pesan diterima : [" +message.toString() + "] dalam
    " + message.length() +  " bytes");
38.                 //System.out.println("Message received");
39.                 numMessages++;
40.                 out.println("Isi Pesan " + numMessages + ":" + message);
41.                 message = in.readLine();
42.
43.             }
44.             out.println(numMessages + " buah pesan telah diterima.");
45.         } catch (IOException e) {
46.         } finally {
47.             try {
48.                 System.out.println("*********Menutup koneksi ****");
49.                 link.close();
50.             } catch (IOException e) {
51.                 System.out.println("Tidak dapat memustukan koneksi");
52.                 System.exit(1);
53.             }
54.         }
55.     }
56. }
```

3b. Program Aplikasi Chat Client dengan menggunakan TCP

```
1.  import java.io.BufferedReader;
2.  import java.io.IOException;
3.  import java.io.InputStreamReader;
4.  import java.io.PrintWriter;
5.  import java.net.InetAddress;
6.  import java.net.Socket;
7.  import java.net.UnknownHostException;
8.
```

```
9.  public class TCPEchoClient {
10.
11.    private static String strHost;
12.    private static InetAddress host;
13.    private static final int PORT = 1234;
14.
15.    public static void main(String args[]) {
16.       try {
17.        // host = InetAddress.getLocalHost();
18.        strHost = "10.252.44.177" ; // <- Masukan sesuai dengan tujuan
19.       host = InetAddress.getByName(strHost);
20.       } catch (UnknownHostException e) {
21.          System.out.println("Alamat tidak ditemukan");
22.          System.exit(1);
23.       }
24.       run();
25.    }
26.
27.    private static void run() {
28.       Socket link = null;
29.       try {
30.          link = new Socket(host, PORT);
31.          BufferedReader in = new BufferedReader(new
   InputStreamReader(link.getInputStream()));
32.          PrintWriter out = new PrintWriter(link.getOutputStream(), true);
33.
34.          BufferedReader userEntry = new BufferedReader(new
   InputStreamReader(System.in));
35.          String message, response;
36.          do {
37.             System.out.print("Masukkan pesan : ");
38.             message = userEntry.readLine();
39.             out.println(message);
40.             response = in.readLine();
41.             System.out.println("SERVER " + response);
42.          } while (!message.equals("close"));
43.       } catch (IOException e) {
44.          e.printStackTrace();
45.       } finally {
46.          try {
47.             System.out.println("Menutup Koneksi.");
48.             link.close();
49.          } catch (IOException e) {
50.             System.out.println("Tidak dapat memutuskan koneksi!");
51.             System.exit(1);
52.          }
53.       }
```

```
54.     }
55. }
```

D. PERCOBAAN

1. Tulis kembali program C1 dalam editor yang anda sukai.
2. Jalankan program “netstat -ap TCP”. Catat alamat IP sumber , alamat IP tujuan dan nomor port yang sedang dalam keadaan menunggu (listen()).
3. Jalankan program C1 ! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
4. Jalankan web browser dan arahkan URL ke alamat IP computer anda ditambah dengan nomor port dari web server percobaan!
5. Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
6. Buat flowchart untuk menggambarkan interaksi program C1 dan browser anda!
7. Tulis kembali program C2,C3 dalam editor yang anda sukai.
8. Jalankan program Aplikasi Server C2! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
9. Jalankan program C3! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2.
10. Kirim 3 pesan yang berbeda. Pesan terakhir adalah “close”. Catat hasil pada aplikasi server dan aplikasi client.
11. Jalankan perintah pada no 11! Amati perubahan port dari aplikasi client yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 2. Mengapa terjadi perubahan pada port aplikasi client?
12. Buat flowchart untuk menggambarkan interaksi program C2 dan program C3!

4a. Program UDP server sederhana

Source : https://systembash.com/a-simple-java-udp-server-and-udp-client/

```
import java.io.*;
import java.net.*;

class UDPServer
{
   public static void main(String args[]) throws Exception
      {
         DatagramSocket serverSocket = new DatagramSocket(9876);
            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];
            while(true)
               {
                  DatagramPacket receivePacket = new DatagramPacket(receiveDa
ta, receiveData.length);
                  serverSocket.receive(receivePacket);
                  String sentence = new String( receivePacket.getData());
                  System.out.println("RECEIVED: " + sentence);
                  InetAddress IPAddress = receivePacket.getAddress();
                  int port = receivePacket.getPort();
                  String capitalizedSentence = sentence.toUpperCase();
                  sendData = capitalizedSentence.getBytes();
                  DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress,
port);
                  serverSocket.send(sendPacket);
               }
      }
}
```

4b. Program UDP client sederhana

Source : https://systembash.com/a-simple-java-udp-server-and-udp-client/

```
import java.io.*;
import java.net.*;

class UDPClient
{
```

```
   public static void main(String args[]) throws Exception
   {
      BufferedReader inFromUser =
         new BufferedReader(new InputStreamReader(System.in));
      DatagramSocket clientSocket = new DatagramSocket();
      InetAddress IPAddress = InetAddress.getByName("localhost");
      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];
      String sentence = inFromUser.readLine();
      sendData = sentence.getBytes();
      DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 9876);
      clientSocket.send(sendPacket);
      DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
      clientSocket.receive(receivePacket);
      String modifiedSentence = new String(receivePacket.getData());
      System.out.println("FROM SERVER:" + modifiedSentence);
      clientSocket.close();
   }
}
```

Socket Programming Tingkat Lanjut
(kerjakan di rumah)

A. DASAR TEORI

Pada praktikum sebelumnya anda telah membuat program aplikasi client-server. Kelemahan dari program sebelumya adalah tiap server hanya bisa melayani satu koneksi. Padahal dalam aplikasi sebenarnya setiap aplikasi server mempunyai kemampuan menerima koneksi lebih dari satu.

Pemrograman Java menawarkan dua buah solusi , yaitu dengan menggunakn thread dan menggunakan RMI (Remote Method Invocation). Pada praktikum ini, kita akan berfokus pada solusi dengan menggunakan multithread. Seperti yang sudah anda pelajari pada mata kuliah sistem operasi, pada proses single thread proses akan menjalakan bagian program secara terurut, dan memakai resources secara bergantian. Berbeda dengan multi thread, dimana thread saling berbagi bagian program dan resources dengan thread lain yang mengacu pada proses yang sama.

Pada praktikum ini kita akan mempelajari strategi penggunaan multithread pada pemrograman client-server. Pada sisi aplikasi server terdapat penambahan 1 buah class bernama clientHandler yang digunakan untuk melayani koneksi ke client. Main class dari aplikasi server memanggil thread yang berisi class tersebut apabila ada aplikasi client yang meminta koneksi.

B. LISTING PROGRAM

3. Program Utama(Main Program) Aplikasi Server dengan Multithread

```
1. import java.io.IOException;
2. import java.net.ServerSocket;
3.
4. public class TCPEchoServerThread {
5.
```

```
6.      private static ServerSocket servSock;
7.      private static final int PORT = 12345;
8.
9.      public TCPEchoServerThread() {
10.     }
11.
12.     public void start() {
13.        try {
14.           servSock = new ServerSocket(PORT);
15.
16.           while (true) {
17.              Thread clientThread = new Thread(new clientHandler(servSock.accept()));
18.              clientThread.start();
19.           }
20.        } catch (IOException e) {
21.           e.printStackTrace();
22.        } finally {
23.           try {
24.              System.out.println("Menutup koneksi....");
25.              servSock.close();
26.           } catch (IOException e) {
27.              System.out.println("Tidak dapat memustukan koneksi");
28.              e.printStackTrace();
29.              System.exit(1);
30.           }
31.        }
32.    }
33.
34.    public static void main(String[] args) {
35.        TCPEchoServerThread es = new TCPEchoServerThread();
36.        System.out.println("Server telah berjalan di komputer ini pada port " +PORT);
37.        es.start();
38.    }
39. }
```

4. Program Aplikasi Server Untuk Melayani Koneksi Client

```
1.  import java.io.BufferedReader;
2.  import java.io.IOException;
3.  import java.io.InputStreamReader;
4.  import java.io.PrintWriter;
5.  import java.net.*;
6.
7.  class clientHandler implements Runnable {
8.
9.     private static int numConnections;
```

```
10.    private int connectionId = 0;
11.    Socket link;
12.
13.    public clientHandler(Socket s) {
14.       connectionId = numConnections++;
15.       System.out.println("Melayani koneksi ke-"+ connectionId);
16.       link = s;
17.    }
18.
19.    public void run() {
20.       PrintWriter out = null;
21.       BufferedReader in = null;
22.       int numMessages = 0;
23.
24.       try {
25.          out = new PrintWriter(link.getOutputStream(), true);
26.          in = new BufferedReader(new InputStreamReader(link.getInputStream()));
27.          String message=in.readLine();
28.          while (!message.equals("close") ) {
29.           System.out.println("Pesan diterima : [" +message.toString() + "] dari client " +
     connectionId +" dalam " + message.length() +  " bytes");
30.             numMessages++;
31.               out.println("Isi Pesan " + numMessages + ":" + message);
32.               message = in.readLine();
33.          }
34.       } catch (Exception e) {
35.          e.printStackTrace();
36.       } finally {
37.          out.close();
38.          try {
39.             in.close();
40.             link.close();
41.             System.out.println("Menutup koneksi, #" + connectionId);
42.          } catch (IOException e) {
43.             e.printStackTrace();
44.          }
45.       }
46.    }
47. }
```

D. PERCOBAAN

1. Tulis kembali program C1,C2 dalam editor yang anda sukai.
2. Jalankan program Aplikasi Server C1!

3. Jalankan program “netstat -ap TCP”. Catat alamat IP sumber , alamat IP tujuan dan nomor port yang sedang dalam keadaan menunggu (listen()).
4. Jalankan program client! Amati perubahan yang sedang terjadi pada komputer anda dengan menggunakan perintah pada nomor 3.
5. Jalankan program sebanyak 2 client lagi! Kirim 3 pesan yang berbeda. Pesan terakhir adalah “close”. Catat hasil pada aplikasi server dan aplikasi client.
6. Buat flowchart untuk menggambarkan interaksi program C1,C2 dan aplikasi client!