UDP Socket

UDP menyediakan layanan transfer yang tidak dapat diandalkan dengan mengirimkan paket datagrams per kelompok antar server dan client tanpa pemeriksaan/connectionless Keuntungan UDP :

- Simple, tidak menyediakan utility untuk memastikan koneksi.
- Fleksibel, dapat mengirim pesan ke beberapa penerima pesan meskipun dengan adanya operasi lain yang sedang berjalan.
- Efisien, paket yang dikirim untuk koneksi data hanya sedikit.
- Cepat, data dapat langsung dikirim tanpa pemeriksaan koneksi.
- Kemampuan Broadcast, dapat mengirim ke beberapa penerima sekaligus dalam sekali kirim.

Kelemahan UDP :

- Banyak data yang hilang.
- Banyak kesalahan pengiriman.

A. Implementasi UDP Socket

Silahkan buat sebuah file dengan nama UDPServer.java lalu masukkan source code seperti berikut.

SocketServer.java

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class SocketServer {

    private final DatagramSocket server;
    private final DatagramPacket message;

    public SocketServer(DatagramSocket server, DatagramPacket
message) {
        this.server = server;
        this.message = message;
    }

    public void start() throws IOException {
        while (true) {
            System.out.println("Server started....");
            server.receive(message);
            threatMessage();
        }
    }

    public void stop() {
        server.close();
    }

    private void threatMessage() throws IOException {
```

```
                String messageReceived = new
        String(message.getData()).trim();
                System.out.println("Message Received: " + messageReceived);

                if (messageReceived.equals("hello")) {

                    InetAddress address = message.getAddress();
                    int port = message.getPort();

                    String sendMessage = "ok";
                    byte[] byteSendMessage = sendMessage.getBytes();

                    DatagramPacket answerPacket = new
        DatagramPacket(byteSendMessage, byteSendMessage.length, address,
        port);

                    server.send(answerPacket);
                }
            }
}
```

Pada source code diatas dapat dilihat bahwa server akan menunggu data yang dikirim dari client, data yang dikirim adalah packet dalam bentuk datagram sehingga packet tersebut tidak memiliki penomoran pada saat pengiriman berlangsung.
Untuk menjalankan server, buat sebuah file bernama SocketServerDemo agar server dapat dijalankan sebelum client.

SocketServerDemo.java

```
        import java.io.IOException;
        import java.net.DatagramPacket;
        import java.net.DatagramSocket;
        import java.net.SocketException;
        import java.util.logging.Level;
        import java.util.logging.Logger;

        public class SocketServerDemo {

            public static Logger logger =
        Logger.getLogger(SocketServerDemo.class.getName());

            public static void main(String[] args) {
                int port = 5100;

                try {
                    DatagramSocket socket = new DatagramSocket(port);
                    DatagramPacket packet = new DatagramPacket(new byte[256],
        256);
                    SocketServer server = new SocketServer(socket, packet);
                    server.start();
                } catch (SocketException e) {
                    logger.log(Level.WARNING, e.toString());
                } catch (IOException e) {
                    logger.log(Level.WARNING, e.toString());
```

```
            }
        }
}
```

Pada kode program di sisi server menggunakan port 5100. Ketika client mengirimkan pesan "hello" kepada server, server akan merespon dengan pesan "ok".
Untuk kebutuhan testing, kita juga akan membuat sebuah client udp, silahkan buat sebuah file dengan nama SocketClientDemo.java kemudian masukkan codingan berikut.

SocketClientDemo.java

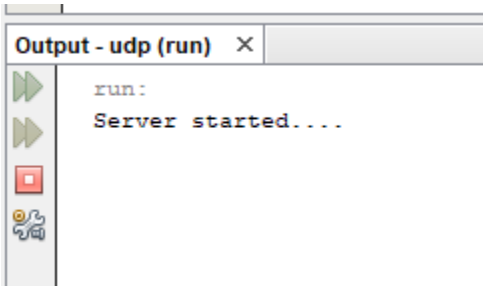
```
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.nio.charset.StandardCharsets;
import java.util.logging.Level;
import java.util.logging.Logger;

public class SocketClientDemo {

    public static Logger logger =
Logger.getLogger(SocketClientDemo.class.getName());

    public static void main(String args[]) {
        int port = 5100;

        String messages = "hello";
        String s;

        DatagramSocket datagramSocket = null;

        InputStream is = new
ByteArrayInputStream(messages.getBytes(StandardCharsets.UTF_8));
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(is));

        try {
            datagramSocket = new DatagramSocket();
            InetAddress inetAddress =
InetAddress.getByName("127.0.0.1");

            while (true) {
                s = bufferedReader.readLine();
                byte[] b = s.getBytes();

                DatagramPacket datagramPacket = new DatagramPacket(b,
                        b.length, inetAddress, port);

                datagramSocket.send(datagramPacket);
```

```
                    byte[] buffer = new byte[65536];
                    DatagramPacket packet = new DatagramPacket(buffer,
        buffer.length);
                    datagramSocket.receive(packet);

                    byte[] data = packet.getData();
                    s = new String(data, 0, packet.getLength());

                    logger.log(Level.INFO, "{0} : {1} : {2}", new
        Object[]{packet.getAddress().getHostName(), packet.getPort(), s});
                }

            } catch (IOException e) {
                logger.log(Level.WARNING, e.toString());
            }
        }

}
```

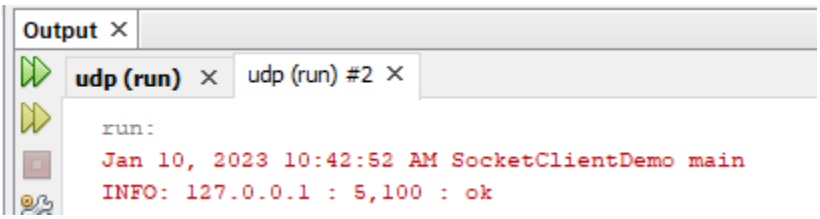
Pastikan client menggunakan port yang sama dengan server, yakni 5100. Client mengirimkan message “hello” kepada server. Source code diatas berfungsi sebagai client, dimana pada saat client dijalankan, anda dapat mengirim pesan ke server lalu server akan mengembalikan pesan anda kembali. Jika terdapat banyak client, maka pesan tersebut tidak akan syncronize dengan client yang lain akan tetapi pesan tersebut hanya syncronize dengan server.

B. Uji Coba UDP Socket

Dari kedua file diatas, client perlu menjalankan server terlebih dahulu. Pastikan server telah terhubung dan siap untuk menerima message dari client.



Setelah server berhasil dijalankan, maka jalankan kode program SocketClientDemo yang akan mengirimkan message “hello” kepada server.



Setelah sisi client dijalankan, maka akan tampil informasi apabila server telah merespon message dari client yakni dengan respon “ok”. Sedangkan pada sisi server akan menerima message berupa string “hello” dari client.

Setelah server menerima message dari client, server akan tetap berjalan sehingga bila client dijalankan kembali maka server akan merespon kembali seperti sebelumnya.