Socket pada UDP

Husni
husni@if.trunojoyo.ac.id
Husni.trunojoyo.ac.id
Komputasi.wordpress.com

UDP

- Bersifat *connectionless*
- Tidak ada koneksi yang antara client & server yang dipelihara selama dialog
- Paket datagram langsung dikirim saat diperlukan
- Lebih cepat daripada transmisi paket TCP
- Unreliable

Server UDP

- Tidak membuat obyek socket untuk setiap client
- Tidak membuat obyek dari kelas ServerSocket, tetapi DatagramSocket
- Obyek DatagramPacket dibuat dan dikirim pada kedua ujung, bukan string sederhana.

Pembuatan Server UDP

1. Buat obyek DatagramSocket

   DatagramSocket datagramSocket = new DatagramSocket(1234);

2. Buat buffer untuk datagram yang masuk

   byte[] buffer = new byte[256];

3. Buat obyek DatagramPacket untuk datagram yang masuk. Parameter untuk *constructor* adalah array byte dari buffer dan ukuran array tersebut.

   DatagramPacket inPacket = new DatagramPacket(buffer, buffer.length);

Pembuatan Server UDP

4. Terima datagram yang masuk

   datagramSocket.receive(inPacket);

5. Terima address dan port pengirim dari paket

   InetAddress clientAddress = inPacket.getAddress();

   int clientPort = inPacket.getPort();

6. Ambil (retrieve) data dari buffer. Gunakan constructor kelas String dengan parameter array paket masuk, posisi awal dan jumlah byte yang akan diambil.

   String message = new String(inPacket.getData(), 0,inPacket.getLength());

Pembuatan Server UDP

7. Buat datagram respon. Buat obyek Datagram-Packet dengn *constructor* array byte yang berisi pesan respon, ukuran respon, alamat dan port dari client

   DatagramPacket outPacket = new DatagramPacket(response.getBytes(), response.length(),clientAddress, clientPort);

8. Kirim datagram respon

   datagramSocket.send(outPacket);

9. Tutup socket datagram

   datagramSocket.close();

```
import java.io.*;
import java.net.*;
public class UDPEchoServer {
    private static final int PORT = 1234;
    private static DatagramSocket datagramSocket;
    private static DatagramPacket inPacket, outPacket;
    private static byte[] buffer;

    public static void main(String[] args) {
        System.out.println("Opening port...\n");
        try {
            datagramSocket = new DatagramSocket(PORT); //Langkah 1.
        } catch(SocketException sockEx) {
            System.out.println("Unable to attach to port!");
            System.exit(1);
        }
        handleClient();
    }
```

```
private static void handleClient() {
    try {
        String messageIn,messageOut;
        int numMessages = 0;
        do {
            buffer = new byte[256]; //Langkah2.
            inPacket = new DatagramPacket(
                buffer, buffer.length); // Langkah 3.
            datagramSocket.receive(inPacket);// Langkah 4.
            InetAddress clientAddress = inPacket.getAddress();
            int clientPort = inPacket.getPort(); // Langkah 5.
            messageIn = new String(inPacket.getData(),
            0,inPacket.getLength()); // Langkah 6.
```

```
            System.out.println("Message received.");
            numMessages++;
            messageOut = "Message " + numMessages + ": " + messageIn;
            outPacket = new DatagramPacket(messageOut.getBytes(),
            messageOut.length(),clientAddress, clientPort);  //Langkah 7.
            datagramSocket.send(outPacket); //Langkah8.
        } while (true);
    }
    catch(IOException ioEx) { ioEx.printStackTrace(); }
    finally {  //jika terjadi exception thrown, tutup koneksi.
        System.out.println("\n* Closing connection... *");
        datagramSocket.close(); //Langkah9.
    }
  }
}
```

Pembuatan Client UDP

1. Buat obyek DatagramSocket. Tanpa port.

   DatagramSocket datagramSocket = new DatagramSocket();

2. Buat obyek datagram keluar. Sama dengan langkah 7 pada server UDP

   DatagramPacket outPacket = new DatagramPacket(message.getBytes(), message.length(), host, PORT);

Pembuatan Client UDP

3. Kirim pesan datagram

   datagramSocket.send(outPacket);

4. Buat buffer untuk datagram yang masuk

   byte[] buffer = new byte[256];

5. Buatobyek DatagramPacket untuk datagram yang masuk

   DatagramPacket inPacket = new DatagramPacket(buffer, buffer.length);

Pembuatan Client UDP

6. Terima datagram yanag masuk

   datagramSocket.receive(inPacket);

7. Ambil data masuk dibuffer

   String response = new
   String(inPacket.getData(), 0,
   inPacket.getLength());

8. Tutup DatagramSocket

   datagramSocket.close();

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

public class UDPEchoClient {
    private static InetAddress host;
    private static final int PORT = 1234;
    private static DatagramSocket datagramSocket;
    private static DatagramPacket inPacket, outPacket;
    private static byte[] buffer;
```

```
public static void main(String[] args) {
    try {
        host = InetAddress.getLocalHost();
    }
    catch(UnknownHostException uhEx) {
        System.out.println("Host ID not found!");
        System.exit(1);
    }
    accessServer();
}
```

```
private static void accessServer() {
    try {
        //Langkah1...
        datagramSocket = new DatagramSocket();
        //Set up stream untuk masukan dari keyboard
        Scanner userEntry = new Scanner(System.in);
        String message="", response="";
        do  {
            System.out.print("Enter message: ");
            message = userEntry.nextLine();
            if (!message.equals("***CLOSE***")) {
                outPacket = new DatagramPacket( message.getBytes(),
                        message.length(), host,PORT); //Langkah 2.
                //Langkah 3...
                datagramSocket.send(outPacket);
                buffer = new byte[256]; // Langkah 4.
                inPacket = new DatagramPacket( buffer, buffer.length);// Langkah 5
```

```
            //Langkah 6...
            datagramSocket.receive(inPacket);
            response = new String(inPacket.getData(),
                0, inPacket.getLength()); //Langkah 7.
            System.out.println("\nSERVER> "+response);
        }
    } while (!message.equals("***CLOSE***"));
  }
  catch(IOException ioEx) { ioEx.printStackTrace(); }
  finally {
    System.out.println("\n* Closing connection... *");
    datagramSocket.close(); //Langkah8.
  }
 }
}
```

Tugas

- Lanjutkan tugas program RandomFileAccess. Dikumpulkan hardcopy berisi source code, penjelasan dan capture hasil ujicoba. Deadline: UTS
- Pelajari tentang konsep dan pemrograman Thread (Sistem Operasi dan Java)
- Buat diagram alir dari proses (*process states*) dan diagram alir thread (*thread states*). Apa bedanya? Dikumpulkan hardcopy dan upload diblog. Deadline: 26 Oktober 2010