Network Programming 2010

Pertemuan-4

Pemrograman Socket

Husni

husni@if.trunojoyo.ac.id

Husni.trunojoyo.ac.id

Komputasi.wordpress.com

Outline

- Paket java.net & Kelas InetAddress
- Komunikasi Socket
- Socket Client/Server TCP

Package java.net

- Berisi kelas-kelas untuk pemrograman jaringan
- Memungkinkan programmer membuat program yang dapat mengakses server di Internet
- Memungkinkan programmer membuat server Internet (jaringan) sendiri

Kelas InetAddress

- Ada dalam paket java.net
- Menangani alamat internet, baik nama host maupun IP address
- Metode getByName() mengembalikan Alamat Internet dari nama host sebagai obyek InetAddress
- Harus men-throw pengecualian *UnknownHostException.*

```
import java.net.*;
import java.util.*;
public class IPFinder {
    public static void main(String[] args) {
        String host;
        Scanner input = new Scanner(System.in);
        System.out.print("\n\nEnter host name: ");
        host = input.next();
        try {
          InetAddress address = InetAddress.getByName(host);
          System.out.println("IP address: “+ address.toString());
        }
        catch (UnknownHostException uhEx) {
                System.out.println("Could not find " + host);
        }
    }
}
```

IP Address Lokal

- Bagaimana mendapatkan IP Address mesin yang sedang digunakan?
- Gunakan metode getLocalHost();

Contoh: getLocalHost()

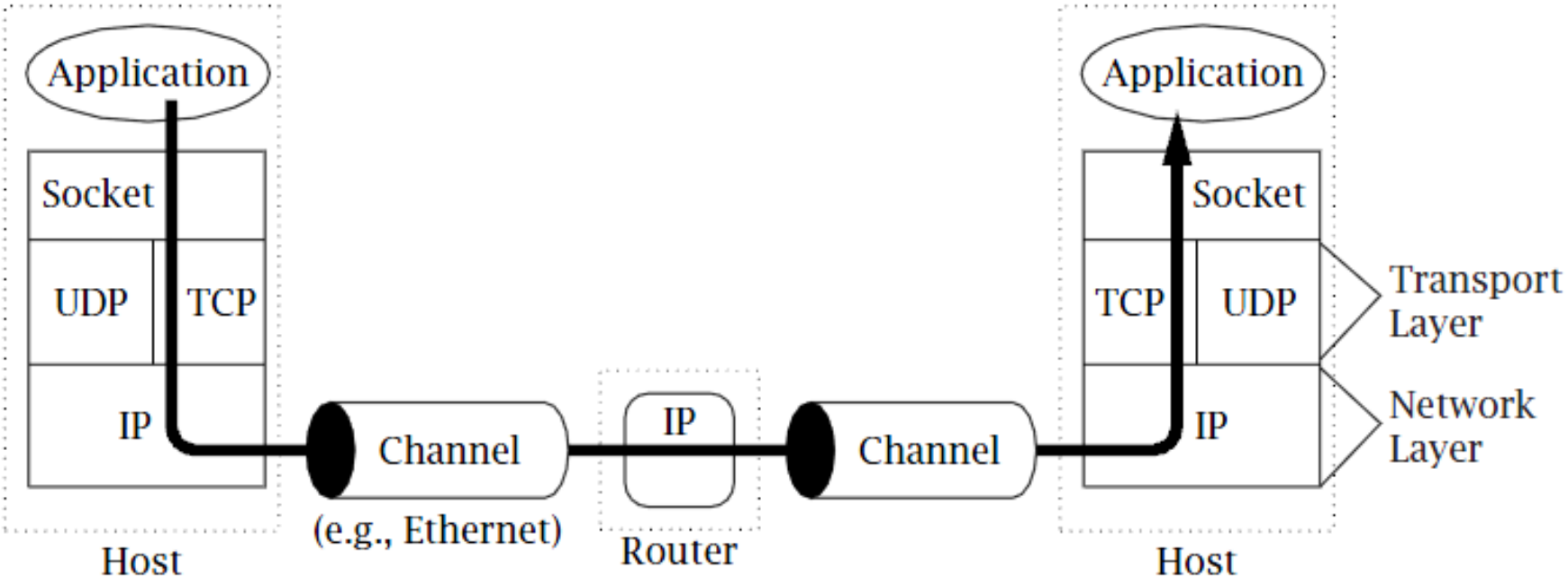
```
import java.net.*;

public class MyLocalIPAddress {
    public static void main(String[] args) {
        try {
            InetAddress address = InetAddress.getLocalHost();
            System.out.println(address);
        }
        catch (UnknownHostException uhEx) {
            System.out.println("Could not find local address!");
        }
    }
}
```

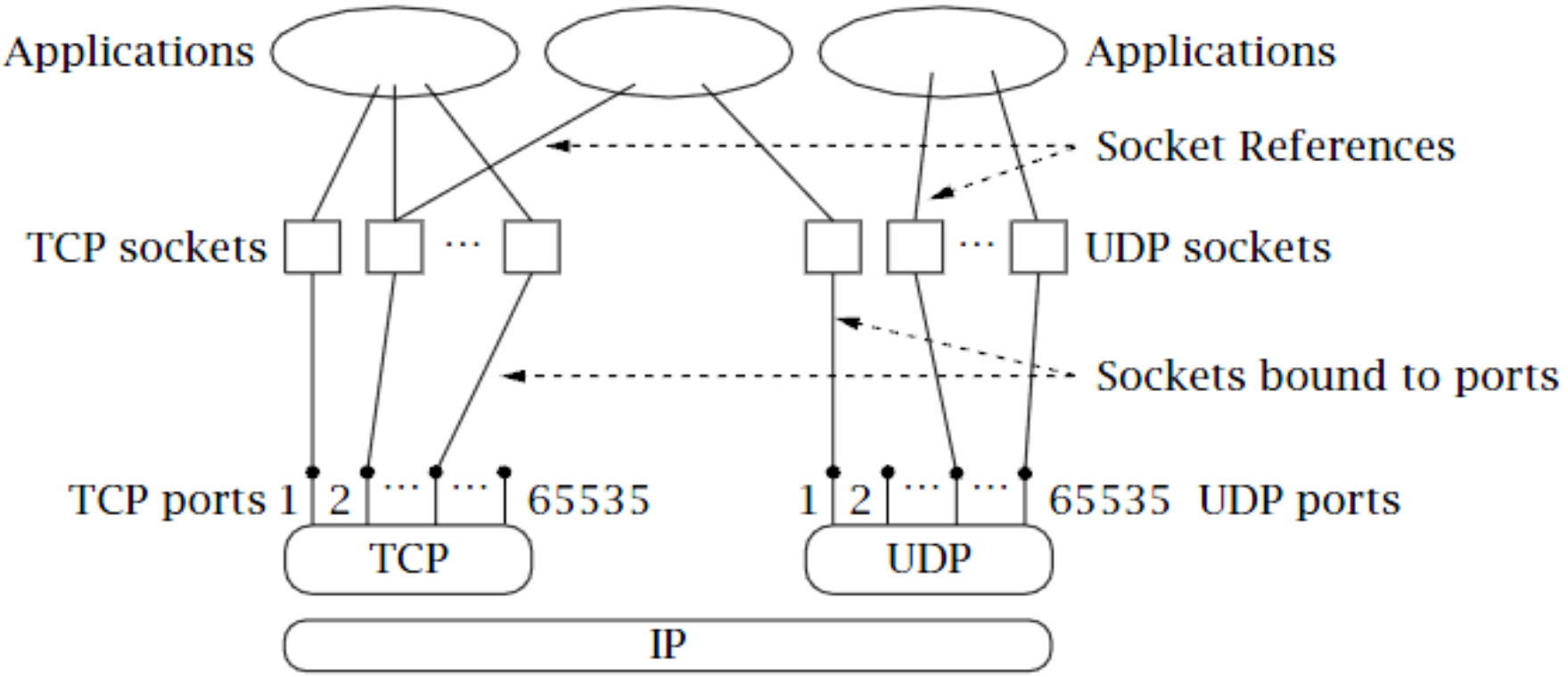
Socket

- Program (proses) berbeda dapat saling berkomunikasi melalui socket
- Java mengimplementasikan socket TCP/IP dan Datagram (UDP)
- Proses-proses yang berkomunikasi biasanya membentuk relasi client/server
- Langkah-langkah membangun program client/server di atas TCP dan UDP hampir sama!

Jaringan TCP/IP



Socket, Protokol & Port



Socket TCP

- Link komunikasi yang dibuat via socket TCP bersifat **connection-orientated**
- Koneksi antara client & server tetap terbuka selama terjadi dialog dan dihancurkan hanya saat salah satu pihak minta mengakhiri komunikasi.
- Pembuatan server dan client memerlukan langkah-langkah hampir sama!

Membangun Client TCP

1. Bangun koneksi ke server. Buat obyek Socket. Tentukan IP Address atau Hostname mesin server dan nomor port dimana aplikasi server berjalan.

   Socket link = new Socket(IP_Address,No_Port);

   Socket link = new Socket(InetAddress.getLocalHost(),1234);

2. Set up stream input dan output. Panggil metode **getInputStream()** & **getOutputStream()** dari obyek Socket.

   Scanner input = new Scanner(link.getInputStream());

   PrintWriter output =new PrintWriter(link.getOutputStream(),true);

3. Kirim dan terima data. Obyek Scanner menerima data dari server (input), obyek PrintWriter mengirimkan data ke server (output).

   output.println(message);

   response = input.nextLine();

4. Tutup koneksi. Panggil metode **close()** dari obyek Socket

   link.close();

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

public class TCPEchoClient {
    private static InetAddress host;
    private static final int PORT = 1234;

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

```
private static void accessServer() {
    Socket link = null;                              //Langkah 1.
    try {
        link = new Socket(host,PORT);                //langkah 1.
        Scanner input =
        new Scanner(link.getInputStream());          //Langkah 2.
        PrintWriter output =
        new PrintWriter(
        link.getOutputStream(),true);                //Langkah 2.

        //Set up stream untuk input dari keyboard
        Scanner userEntry = new Scanner(System.in);
        String message, response;

        do {
                System.out.print("Masukkan pesan: ");
                message = userEntry.nextLine();
```

```
                output.println(message);      //Langkah 3.
                response = input.nextLine(); //Langkah 3.
                System.out.println("\nSERVER> "+response);
        }while (!message.equals("***CLOSE***"));
    }
    catch(IOException ioEx) { ioEx.printStackTrace(); }
    finally {
        try {
            System.out.println("\n* Closing connection... *");
            link.close();                          //Langkah 4.
        }
        catch(IOException ioEx) {
            System.out.println("Unable to disconnect!");
            System.exit(1);
        }
     }
  } //akhir metode accessServer
}  //akhir kelas TCPEchoClient
```

Membangun Server TCP

1. **Buat obyek ServerSocket**. Ikat obyek tersebut pada port tertentu (1024 – 65535)

   ServerSocket servSock = new ServerSocket(1234);
2. **Buat server dalam status menunggu**. Gunakan metode accept() dari obyek ServerSocket, mengembalikan obyek Socket saat suatu koneksi terbangun

   Socket link = servSock.accept();
3. **Set up stream input dan output**. Buat obyek Scanner & PrintWriter

   Scanner input = new Scanner(link.getInputStream());

   PrintWriter output = new PrintWriter(link.getOutputStream(),true);
4. **Kirim dan terima data**. Gunakan metode println() dari PrintWriter & nextLine() dari Scanner.

   output.println("Awaiting data...");

   String input = input.nextLine();
5. **Tutup koneksi** (setelah dialog selesai). Panggil metode close() dari obyek Socket

   link.close();

```
//Server yang meng-echoe kembali pesan dari client (TCP)

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

public class TCPEchoServer {
    private static ServerSocket servSock;
    private static final int PORT = 1234;
    public static void main(String[] args) {
        System.out.println("Opening port...\n");

        try {
            servSock = new ServerSocket(PORT);        //Langkah 1.
        }
        catch(IOException ioEx) {
            System.out.println("Unable to attach to port!");
            System.exit(1);
        }
        do {
            handleClient();
        } while (true);
    }      // akhir dari metode main()
```

```
private static void handleClient() {
    Socket link = null;                                    //Langkah 2.
    try {
       link = servSock.accept();                           // Langkah 2.
       Scanner input = new Scanner(link.getInputStream());//Langka 3
       PrintWriter output = new
    PrintWriter(link.getOutputStream(),true);

       int numMessages = 0;
       String message = input.nextLine();                  // Langkah 4.

       while (!message.equals("***CLOSE***")) {
           System.out.println("Message received.");
           numMessages++;
           output.println("Message " + numMessages + ": " +
    message);  message = input.nextLine();
       }
       output.println(numMessages + " messages received.");
   } // akhir dari try
```

```
        catch(IOException ioEx) {
                ioEx.printStackTrace();
        }
        finally {
            try {
                System.out.println("\n* Closing connection... *");
                link.close(); // Langkah 5.
            }
            catch(IOException ioEx) {
                System.out.println("Unable to disconnect!");
                System.exit(1);
            }
        } //akhir dari finally
    } // akhir dari handleClient()
} // akhir dari kelas TCPEchoServer
```

Tugas (Personal)

- Tulis dan Jalankan. Jika GAGAL, betulkan!
- Berikan komentar tepat di atas tiap baris!
- Upload kode, komentar dan penjelasan ke blog Anda!

```
public class TugasKeren {
   public static void main(String[] args) {
      try {
         InetAddress address = InetAddress.getByName("10.1.1.1");
         boolean reachable = address.isReachable(10000);

         System.out.println("Is host reachable? " + reachable);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

```
import java.net;

public class TugasOK {

   public static void main(String[] args) {
      try {
         InetAddress address = InetAddress.getByName("IP_ADDRESS");
         NetworkInterface ni = NetworkInterface.getByInetAddress(address);
         if (ni != null) {
            byte[] mac = ni.getHardwareAddress();
            if (mac != null) {
               for (int i = 0; i < mac.length; i++) {
                  System.out.format("%02X%s", mac[i],
                        (i < mac.length - 1) ? "-" : "");
               }
            } else {
               System.out.println("Address is not accessible.");
            }
         } else {
            System.out.println("Network Interface is not found.");
         }
      } catch (UnknownHostException e) {
         e.printStackTrace();
      } catch (SocketException e) {
         e.printStackTrace();
      }
   }
}
```

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;

public class TugasHebat {
   public static void main(String[] args) throws Exception {
      String host = "localhost";
      InetAddress inetAddress = InetAddress.getByName(host);

      String hostName = inetAddress.getHostName();
      for (int port = 0; port <= 65535; port++) {
         try {
            Socket socket = new Socket(hostName, port);
            String text = hostName + " is listening on port " + port;
            System.out.println(text);
            socket.close();
         } catch (IOException e) {
            String s = hostName + " is not listening on port " + port;
            System.out.println(s);
         }
      }
   }
}
```