Network Programming

Vittayasak Rujivorakul

Lecture 14

Network Programming

Vittayasak Rujivorakul

Lecture 14

การระบุหาเครื่อง

- ในระบบเครือข่าย การค้นหาเครื่องแต่ละเครื่องใช้ชื่อซึ่งจะต้อง **unique** ในระบบ
- สำหรับ TCP/IP protocol การ ระบุชื่อนั้นกำหนดโดยใช้ **IP** (Internet Protocol) **address** ซึ่งมีอยู่ในรูป
  - ชื่อในรูป Domain Name เช่น www.logic.co.th
  - ชื่อในรูปของตัวเลข เช่น 202.44.32.17
- ในปัจจุบัน IP address ภายในใช้ 32 bit representation
- สามารถหาค่าได้จาก InetAddress.getByName ( ) method ใน java.net

ตัวอย่าง Hostname

```
Import java.net. * ;
public class Hostname   {
    public static void main (Strint args [ ] )   {
          try   {
              if  (args.length  < 1)   {
                    System.out.println ("Your local host is : " +
                                        InetAddress.getByname (null) ) ;
              }
              else System.out.println (InetAddress.getByName (args [0] ) ) ;
          }
              catch (UnknownHostException e)    {
              Syatem.err.println (e) ;
          }
    }
```


Client/Server

- **"client.server"** : สถาปัตยกรรม s/w ซึ่งแบ่งออกเป็น 2 programs :
  - ส่วนหนึ่งเป็นโปรแกรมขอใช้บริการ เรียกว่า **client**
  - อีกส่วนหนึ่งนั้นจะให้บริการ เรียกว่า **server**
- **การเชื่อมโยง client กับ server** ในระบบเครือข่าย โดย **TCP**

  (Transmission Control Protocol)
  - เป็นการเชื่องโยงแบบ Connection-Oriented (client และ server ต้องติดต่อเชื่อมโยงกันตลอดเวลาของการสื่อสาร) โดยอาศัย
    - **Socket** ซึ่งเป็น s/w abstraction ที่ทำหน้าที่เป็นจุดเชื่อมโยงระหว่าง 2 เครื่อง การระบุ **socket ต้องอาศัย Port**
    - **Port** ระบุช่องทางของ service ที่มีในเครื่องนั้น มีค่าระหว่าง **0 - 65535**

Server

- **การทำงานโปรแกรม Server**
  - เปิดตัวรับการติดต่อ **(listener)** โดยใช้ ServerSocket
  - รับการติดต่อจาก client โดยใช้ **accept ( ) method** ซึ่งจะสร้าง Socket ไว้ใช้ในการติดต่อสำหรับ client นั้น
  - ทำการติดต่อเชื่อมโยงโดยอาศัย **InputStream และ OutputStream** เพื่ออ่านเขียน
  - ปิดการติดต่อ โดยการ**ปิด Socket**
  - **ปิด I/O stream** ที่ใช้ในการติดต่อ
  - กลับไปรอรับการติดต่อจาก client อื่นต่อไป

ตัวอย่าง SimpleServer

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

public class SimpleServer {
  public static void main(String args[ ]) {
    ServerSocket s = null;
    Socket s1;
    String sendMsg = "Hello";
    OutputStream s1out;
    DataOutputStream dos;
    try {
          // register service on port 5432
          s = new ServerSocket(5432);
    } catch (IOException e) { }
    while(true) {
        try {
          s1 = s.accept( );
          s1out = s1.getOutputStream( );
          dos = new DataOutputStream(s1out);
          dos.writeUTF(sendMsg);
          s1out.close( );
          s1.close( );
        }catch(IOException e) { }
    }
  }
}
```


Client

- **การทำงานโปรแกรม Client**
  - **สร้าง Socket** เพื่อเปิดการติดต่อไปยัง Server : **Socket (hostname, port)**
  - ทำการติดต่อเชื่อมโยงโดยอาศัย **InputStream และ OutputStream** เพื่ออ่านเขียน
  - ปิดการติดต่อ โดยการ**ปิด Socket**
  - **ปิด I/O Stream** ที่ใช้ในการติดต่อ

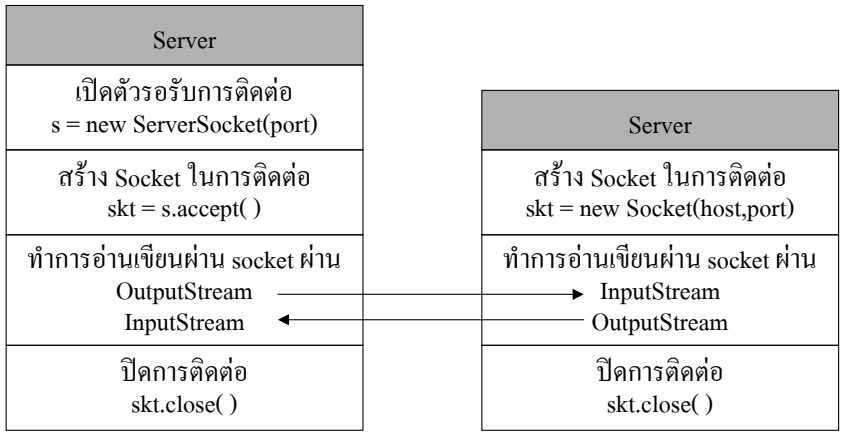
    จบการติดต่อ (จบการทำงาน)

ตัวอย่าง SimpleClient

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

public class SimpleClient {
  public static void main(String args[]){
    Socket s1;
    int c;
    InputStream s1In;
    DataInputStream dis;
    try {
      s1 = new Socket("localhost",5432);
      s1In = s1.getInputStream( );
      dis = new DataInputStream(s1In);
      String st = new String(dis.readUTF( ));
      System.out.println(st);
      s1In.close( );
      s1.close( );
    }catch(UnknownHostException e){
    }catch(IOException e) { }
  }
}
```

C/S Connection Model



UDP (User Datagram Protocol)

- เป็น **connectionless** protocol และถือเป็น unreliable protocol แต่ให้ความเร็วในการรับ - ส่ง **ข้อมูลอาจสูญหาย**และไม่ถูก recover ได้
- การทำงานทั้ง client และ server อาศัย
  - **DatagramSocket** เพื่อทำการรับ-ส่ง packets
  - ใช้ **DatagramPacket** ในการบรรจุข้อมูลลง packets

DatagramSocket & DatagramPacket Constructors

- **DatagramSocket**
  - DatagramSocket ( ) ใช้เพื่อติดต่อกับ localhost ที่ port ใด ๆ ที่มีบริการเปิดอยู่
  - DatagramSocket (int port) ใช้เพื่อติดต่อกับ localhost ที่ port ที่ระบุ
  - DatagramSocket (int port, InetAddress addr) ใช้เพื่อติดต่อกับ host ที่ระบุที่ port ที่กำหนด
- **DatagramPacket**
  - DatagramPacket (byte [ ] recvBuf, int readLen) ใช้เพื่อรับ packet
  - DatagramPacket (byte [ ] sendBuf, int sendLen, InetAddress addr, int port) ใช้เพื่อส่ง packet ไปยัง host และ port นั้น

UDP Server

- **การทำงานโปรแกรม UDP Server**
  - เปิด DatagramSocket เพื่อรับการติดต่อ
  - รับการติดต่อจาก client โดยรอ receive(packet) method
  - ทำการติดต่อเชื่อมโยงกับ client โดยตอบกลับไปยัง client ตาม address และ port ที่ส่ง packet มาโดยอาศัย DatagramPacket

UdpServer

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

public class UdpServer{
  public byte[] getTime( ) {
     Date d = new Date( );
     return d.toString( ).getBytes( );
  }
  public void init( ) throws IOException{
     DatagramSocket socket;
     DatagramPacket inPacket;
     DatagramPacket outPacket;
     InetAddress cAddress;
     int clientPort;
     byte[] msg = new byte[10];
     byte[] time;
     socket = new DatagramSocket(8000);
     while (true) {
     inPacket = new DatagramPacket(msg, 10);
     socket.receive(inPacket);
     cAddress = inPacket.getAddress( );
     clientPort = inPacket.getPort( );
     time =getTime( );
     outPacket = new DatagramPacket
     (time,time.length,cAddress, clientPort);
              socket.send(outPacket);
    }
  }
  public static  void main(String args[ ] ) throws
     Exception {
     UdpServer u = new UdpServer( );
     u.init( );
  }
}
```


UDP Client

- การทำงานโปรแกรม UDP Client
  - สร้าง DatagramSocket
  - สร้าง DatagramPacket โดยระบุข้อความ, host address และ port
  - ส่ง packet เพื่อเริ่มการติดต่อ
  - ทำการอ่านเขียนผ่าน DatagramPacket

UdpClient

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

public class UdpClient {
  public void init( ) throws IOException {
     DatagramSocket socket;
     DatagramPacket outPacket;
     DatagramPacket inPacket;
     InetAddress sAddress;
     byte[ ] b = new byte[1000];
     String inMsg;
     socket = new DatagramSocket( );
     sAddress = InetAddress.getLocalHost( );
     outPacket = new DatagramPacket(b, 1,
     sAddress, 8000);
     socket.send(outPacket);
     inPacket = new DatagramPacket(b, b.length);
     socket.receive(inPacket);
     inMsg = new String(inPacket.getData( ), 0,
       inPacket.getLength( ));
     System.out.println(inMsg);
     socket.close( );
  }
  public static void main(String args[ ]) throws
     IOException {
      UdpClient udpClient = new UdpClient( );
      udpClient.init( );
  }
}
```