

Índice hora 2

Hora 1

1 API de sockets BSD

2 Sockets TCP

2.1 Cliente TCP

2.2 Servidor TCP

2.3 Detalles de sockets TCP

Hora 2

3 Sockets UDP

3.1 Cliente UDP

3.2 Servidor UDP

3.3 Detalles de sockets UDP

4 Otras funcionalidades del API BSD

5 Excepciones

Hora 3

6 Streams

7 Servidores concurrentes

7.1 Sockets no bloqueantes

7.2 Selectores

7.3 Threads

Objetivos

- Aprender a programar aplicaciones cliente/servidor UDP
- Revisar otras funcionalidades del API de sockets BSD básico
- Notar la posibilidad de situaciones de error: excepciones

3 Sockets UDP

- Para la aplicación, UDP permite transferir datagramas de manera no fiable (se pueden perder), sin garantía de orden, entre una máquina y cualquier otro conjunto de máquinas.
 - No hay conexión como tal
 - No hay distinción de funcionamiento cliente/servidor como tal, aunque se suele mantener la denominación llamando cliente al extremo que manda el primer paquete y servidor el extremo que se queda a la escucha en un puerto conocido.
 - El emisor explícitamente indica la dirección IP destino y puerto destino para cada paquete
 - El receptor debe extraer la dirección IP origen y puerto origen del emisor a partir del paquete recibido

Sockets UDP

- Clases Java relacionadas con sockets UDP:
 - `java.net.DatagramSocket`: socket general y socket cliente/servidor
 - `java.net.DatagramPacket`: datagrama UDP

Sockets UDP

Servidor (ejecutando en `hostid`)

crea socket,
 port=`x`, para
 recibir solicitudes:
`serverSocket =`
`DatagramSocket()`

Lee el datagrama desde
`serverSocket`

Escribe la respuesta en
`serverSocket`
 Especificando la dirección
 IP del cliente y el
 número de puerto

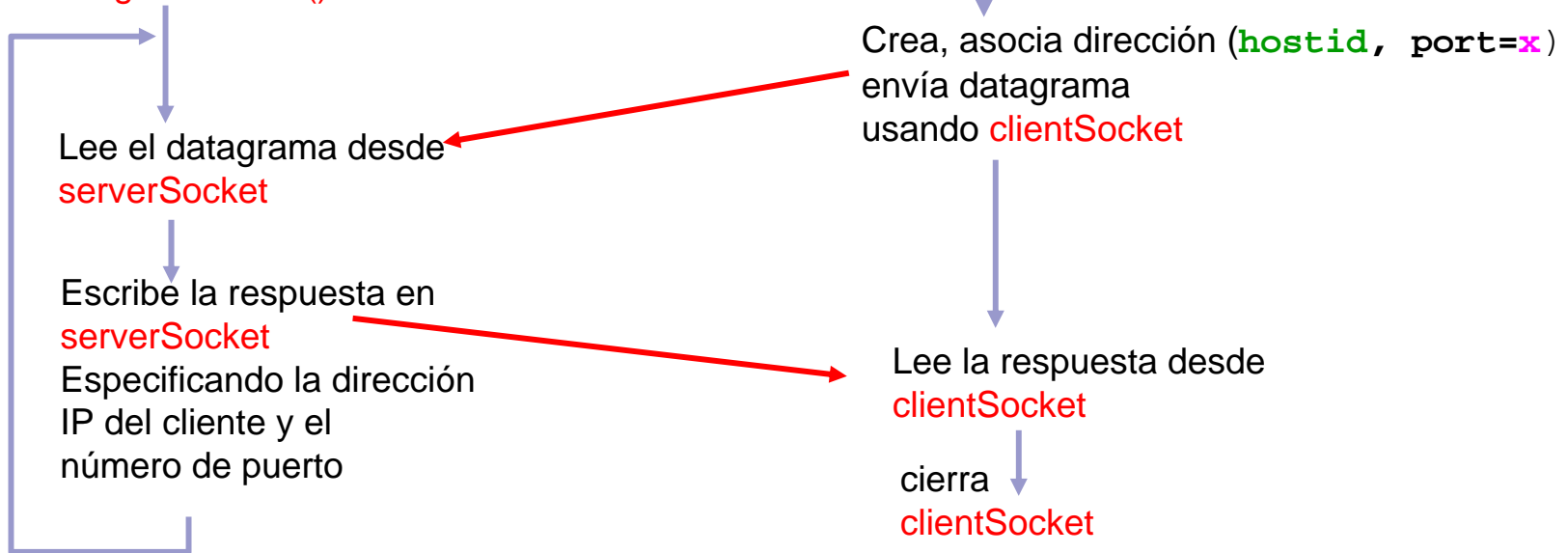
Cliente

crea socket,
`clientSocket =`
`DatagramSocket()`

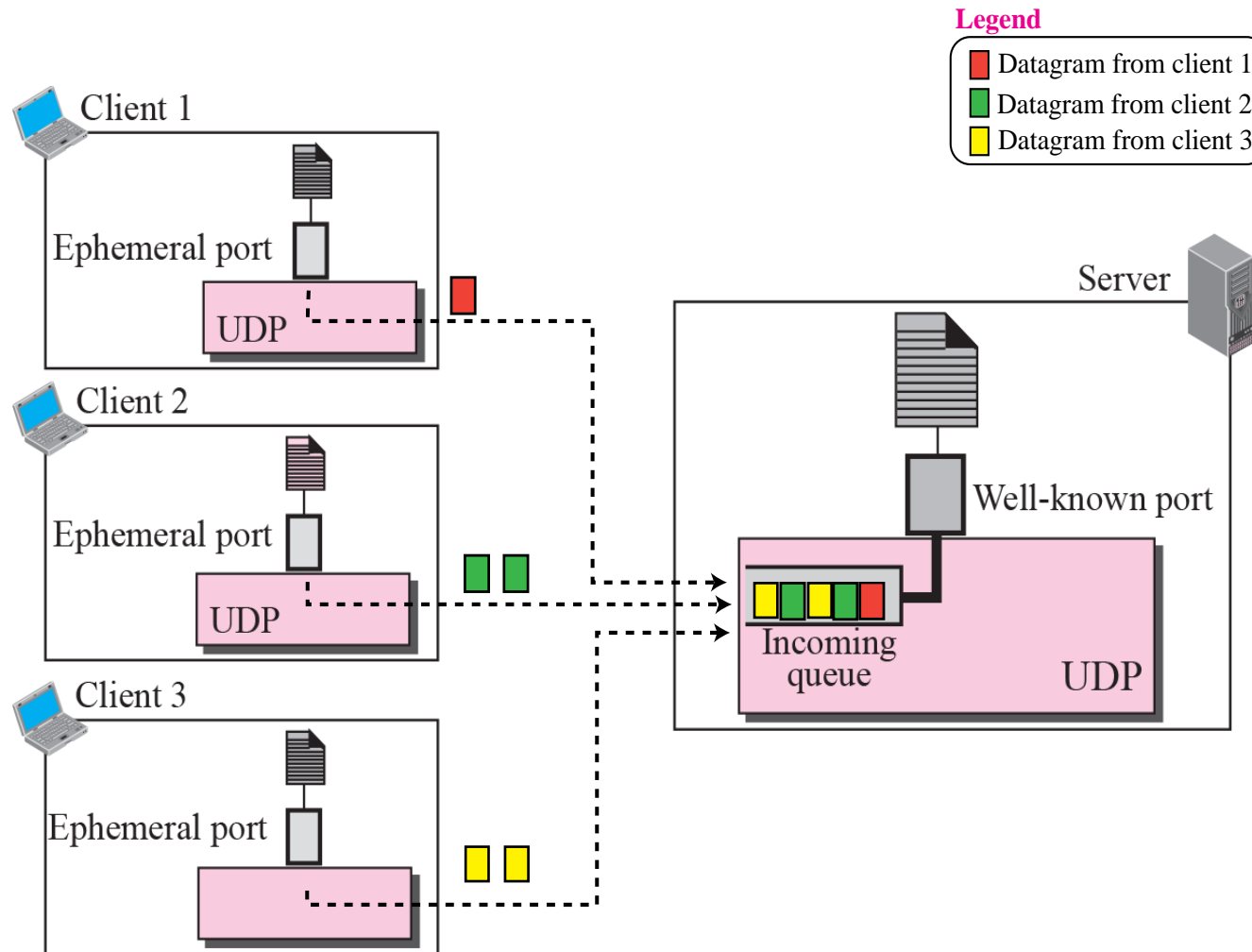
Creación, asocia dirección (`hostid`, `port=x`)
 envía datagrama
 usando `clientSocket`

Lee la respuesta desde
`clientSocket`

cierra
`clientSocket`

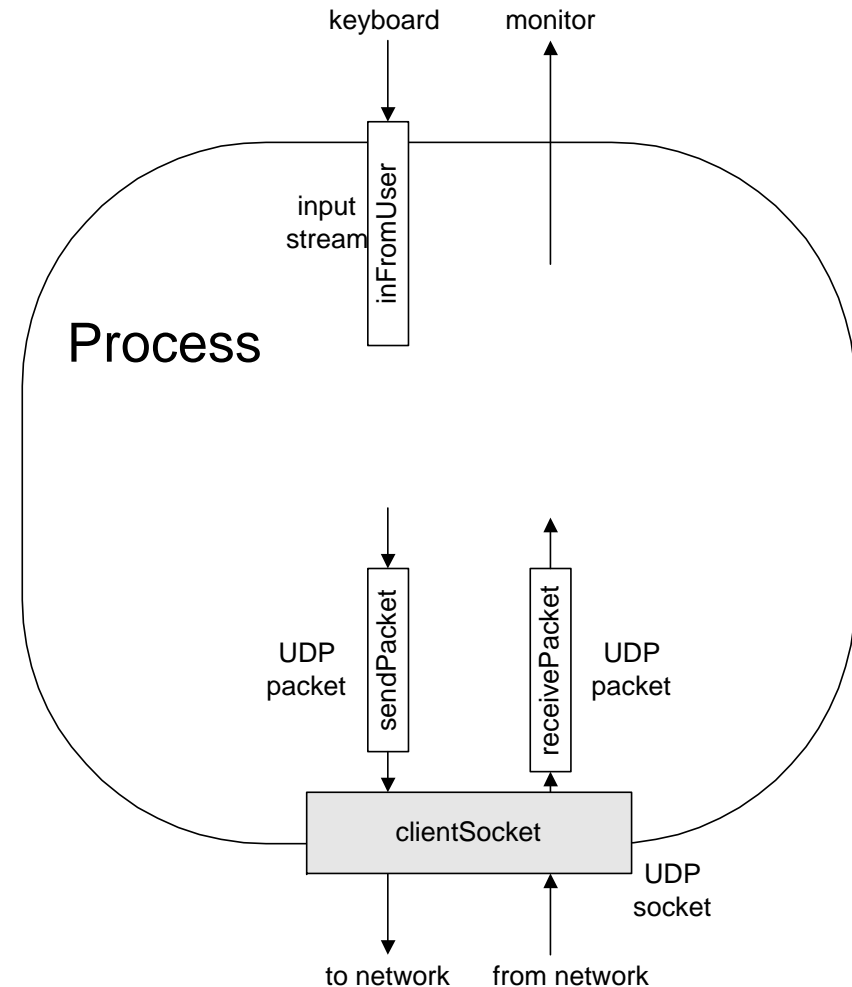


Sockets UDP



Ejemplo de aplicación cliente-servidor UDP

1. El cliente lee una línea desde el dispositivo de entrada estándar (stream inFromUser) y lo envía al servidor
2. El servidor lee la línea desde un socket
3. El servidor convierte la línea a mayúsculas, y la devuelve al cliente
4. El cliente lee la línea modificada que lee desde el socket y la imprime en pantalla



3.1 Cliente UDP

```
import java.io.*;
import java.net.*;
class UDPClient {
    public static void main(String args[]) throws Exception
    {
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        //Crea input stream (de texto) asociado al teclado
        BufferedReader inFromUser = new BufferedReader(new
        InputStreamReader(System.in));
        //Crea socket cliente de tipo datagrama UDP
        DatagramSocket clientSocket = new DatagramSocket();
        //Convierte el nombre DNS del servidor en una dirección IP
        InetAddress IPAddress = InetAddress.getByName("server.com");
        //Lee una línea de teclado
        String sentence = inFromUser.readLine();
        //Copia la línea al array de bytes
        sendData = sentence.getBytes();
    }
}
```


Cliente UDP (continuación)

//Crea un datagrama con el array de bytes a enviar, su longitud, y la IP y puerto del servidor

```
DatagramPacket sendPacket = new DatagramPacket(sendData,  
sendData.length, IPAddress, 9876);
```

//Envía el datagrama

```
clientSocket.send(sendPacket);
```

//Crea un datagrama vacío a llenar con lo que se reciba

```
DatagramPacket receivePacket = new  
DatagramPacket(receiveData, receiveData.length);
```

//Recibe el datagrama

```
clientSocket.receive(receivePacket);
```

//Lee la línea devuelta por el servidor

```
String modifiedSentence = new String(receivePacket.getData());
```

```
System.out.println("FROM SERVER:" + modifiedSentence);
```

//Cierra el socket

```
clientSocket.close();
```

```
}
```

```
}
```

3.2 Servidor UDP

```
import java.io.*;
import java.net.*;
class UDPServer {
    public static void main(String args[]) throws Exception
    {
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];

        //Crea socket servidor de tipo datagrama UDP escuchando en un puerto
        DatagramSocket serverSocket = new DatagramSocket(9876);
        //Bucle infinito esperando datagramas de clientes
        while(true)
        {
            //Prepara contenedor para recibir el datagrama
            DatagramPacket receivePacket = new
            DatagramPacket(receiveData, receiveData.length);
            //Recibe el datagrama (de cualquier cliente)
            serverSocket.receive(receivePacket);
        }
    }
}
```

Servidor UDP (continuación)

```
//Convierte a string el contenido del datagrama
String sentence = new String(receivePacket.getData());
//Saca dirección IP origen y puerto origen del datagrama
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
//Convierte a mayúsculas
String capitalizedSentence = sentence.toUpperCase();
//Convierte a cadena de bytes
sendData = capitalizedSentence.getBytes();
//Crea el datagrama a enviar al cliente
DatagramPacket sendPacket = new
DatagramPacket(sendData, sendData.length, IPAddress, port);
//Envia el datagrama
serverSocket.send(sendPacket);
    }
}
}
```

3.3 Detalles de sockets UDP

- Creación del socket (java.net.DatagramSocket)
 - Constructores:
 - DatagramSocket()
 - Normalmente se utiliza este en los clientes
 - DatagramSocket(int puerto)
 - DatagramSocket(int puerto, InetAddress dirIP)
 - Intercambio de datos
 - Enviar:
 - void send(DatagramPacket)
 - Recibir:
 - receive(DatagramPacket)
 - Establecer timeout para recibir.
 - setSoTimeout(int milisegundos)
 - Cierre del socket:
 - void close()

Detalles de sockets UDP

- Datagramas (java.net.DatagramPacket):
 - Constructores (para enviar):
 - DatagramPacket(byte[] buf, int length, InetAddress addr, int port)
 - DatagramPacket(byte[] buf, int offset, int length, InetAddress addr, int port)
 - buf : array con los datos que se van a enviar
 - offset: desplazamiento dentro del array de datos a enviar
 - length: número de bytes a enviar
 - addr: dirección IP del destino
 - port: número de puerto del destino
 - Constructores (para recibir):
 - DatagramPacket(byte[] buf, int length)
 - DatagramPacket(byte[] buf, int offset, int length)
 - buf : array para almacenar los datos que se van a recibir
 - offset: desplazamiento dentro del array de datos a recibir
 - length: número máximo de bytes que se van a recibir

Detalles de sockets UDP

- Datagramas (java.net.DatagramPacket):
 - Métodos
 - InetAddress getAddress()
 - Devuelve la dirección almacenada en el paquete.
 - int getPort()
 - Devuelve el puerto almacenado en el paquete.
 - byte[] getData()
 - Devuelve los datos almacenados en el paquete.
 - int getLength()
 - Devuelve el tamaño de los datos almacenados en el paquete.
 - void setAddress(InetAddress dest)
 - Establece la dirección de destino.
 - void setPort(int puerto)
 - Establece el puerto destino.
 - void setData(byte[] buffer)
 - Establece los datos a enviar.
 - void setLength(int n)
 - Establece el tamaño máximo del paquete.

4 Otras funcionalidades del API BSD

- Clase java.net.InetAddress:
 - InetAddress almacena la dirección IP
 - InetAddress getByName(String host)
 - Obtiene la dirección IP de la máquina a partir de su nombre de DNS
 - Recibe el nombre de la máquina o su dirección IP como cadena.
 - InetAddress getByAddress(byte[] dirIP)
 - Obtiene la dirección IP en formato InetAddress a partir de la IP en formato de número de 4 bytes
 - InetAddress[] getAllByName(String host)
 - Obtiene todas las direcciones IP de una máquina de su nombre de DNS.
 - Recibe el nombre de la máquina o su dirección IP como cadena.
 - InetAddress getLocalHost():
 - Obtiene la dirección IP de la máquina en la que se está ejecutando.
 - String getHostName(), String getHostAddress()
 - Devuelve el nombre de la máquina o IP en string correspondiente a esta IP.
 - Convertir InetAddress a String:
 - String toString()

Otras funcionalidades del API BSD

- Clase java.net.Socket:
 - InetAddress getInetAddress()
 - Devuelve la dirección remota del socket
 - InetAddress getLocalAddress()
 - Devuelve la dirección local del socket
 - int getPort()
 - Devuelve el puerto remoto
 - int getLocalPort()
 - Devuelve el puerto local
 - void close()
 - Cierra el socket

5 Excepciones

- Hasta ahora hemos ignorado las excepciones, pero los métodos revisados pueden generar múltiples situaciones de error:
 - **BindException**: ocurre cuando se intenta construir el socket sobre un puerto en uso o si no se tienen privilegios para hacerlo.
 - **ConnectException**: ocurre cuando el host remoto rechaza la conexión.
 - **NoRouteToHostException**: ocurre cuando ha expirado el tiempo de establecimiento de conexión.
 - **SocketException**: ocurre cuando hay un error en el socket.
 - **UnknownHostException**: ocurre si no encuentra la máquina con la cual se desea establecer la conexión.

Excepciones

- Ejemplo: obtención del nombre y la dirección IP de la máquina cuyo nombre de DNS se ha pasado por línea de comandos

```
import java.net.*;
class QuienEs {
    public static void main (String[] args) {
        if (args.length == 1) {
            try {
                InetAddress dir = InetAddress.getByName(args[0]);
                System.out.println("IP: " + dir.getHostAddress());
                System.out.println("Nombre: " + dir.getHostName());
            }
            catch (UnknownHostException e) {
                System.err.println("host desconocido");
            }
        }
    }
}
```

Resumen

- Clases básicas para el desarrollo de aplicaciones UDP
 - `java.net.DatagramSocket`: socket general y socket cliente/servidor
 - `DatagramSocket()`
 - Socket cliente
 - `DatagramSocket(int puerto)`
 - Socket servidor, se queda a la escucha en dicho puerto
 - `send(DatagramPacket)`
 - `receive(DatagramPacket)`
 - `java.net.DatagramPacket`: datagrama UDP
 - `DatagramPacket(byte[] buf, int length)`
- El API de sockets BSD incorpora toda clase de métodos para el mapeo entre nombres de DNS y direcciones IP
- Como otros métodos Java pueden generar excepciones, los relacionados con `java.net.*` también

Referencias

- [Forouzan]
 - Capítulo 2, “Application Layer”, sección 2.7
- Manual en línea Java 1.6,
<http://docs.oracle.com/javase/6/docs/api/overview-summary.html>
- "All About Datagrams" (Tutorial Oracle Java),
<http://docs.oracle.com/javase/tutorial/networking/datagrams/index.html>
- “Socket Programming in Java: a tutorial,”
<http://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html>