Infraestructura Computacional

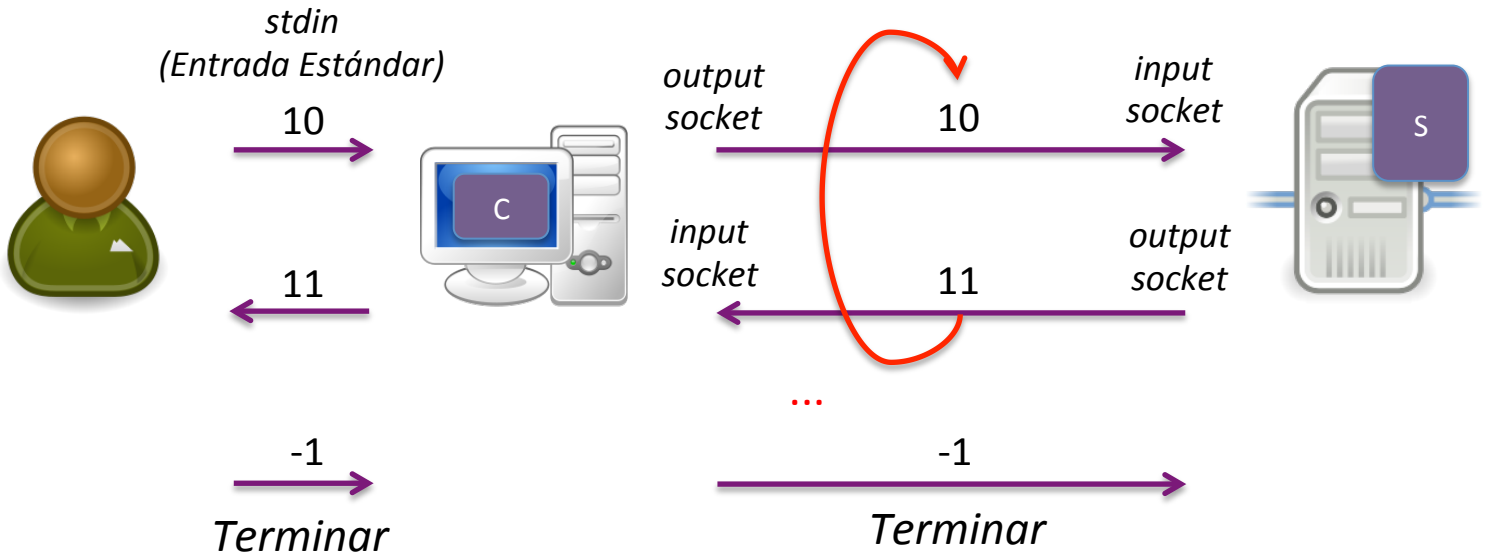
Concurrencia en Servidores Java

Arquitectura Cliente Servidor

- Servidor
  - Siempre en el mismo equipo
  - Dirección IP fija
- Cliente
  - Inicia la comunicación
  - Dirección IP puede ser dinámica

Ejemplo

- Cliente
  - Inicia la comunicación
  - Recibe números del usuarios vía la entrada estándar (System.in) y los envía al servidor
  - Recibe la respuesta del servidor y la despliega en la pantalla (System.out)
  - Termina la conexión si recibe -1
- Servidor
  - Espera conexiones de los clientes
  - Puede atender múltiples solicitudes sobre la misma conexión
  - Para cada solicitud recibe un número n, n>0, y retorna n+1
  - Termina la conexión con un cliente si recibe -1
  - Continúa con el siguiente cliente



Servidor Iterativo

```
public static void main(...)
{
    ...
    ss = new ServerSocket (PUERTO);
    while (true){
        Socket sc = ss.accept();

        // Flujos para lectura y escritura
        PrintWriter salida =
            new PrintWriter(sc.getOutputStream(),true);
        BufferedReader entrada =
            new BufferedReader(newInputStreamReader(sc.getInputStream()));

        atender();
        entrada.close();
        salida.close();
        sc.close();
    }
}
```

Creación del socket

Espera conexiones de clientes. Cuando recibe una conexión crea una instancia (sc) que identifica a un cliente particular.

Atiende los pedidos

Cierra los flujos y el socket con el cliente

Cliente

```
sc = new Socket (HOST,PUERTO);

// Flujos para lectura y escritura
PrintWriter escritor = new PrintWriter(sc.getOutputStream(), true);
BufferedReader lector = new BufferedReader(new InputStreamReader(sc.getInputStream()));
BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
String fromServer;
String fromUser;

while (ejecutar) {
    System.out.print("Numero:");
    fromUser = stdIn.readLine();
    if (fromUser != null && !fromUser.equals("-1")) {
        System.out.println("Cliente: " + fromUser);
        escritor.println(fromUser);
        if ((fromServer = lector.readLine()) != null) {
            System.out.println("Servidor: " + fromServer);
        }
    } else {
        ejecutar = false;
    }
}

out.close();
in.close();
stdIn.close();
sc.close();
```

Creación del socket

String para almacenar información que viene del servidor

String para almacenar información que viene del usuario

Cliente

```
sc = new Socket (HOST,PUERTO);

// Flujos para lectura y escritura
PrintWriter escritor = new PrintWriter(sc.getOutputStream(), true);
BufferedReader lector = new BufferedReader(new InputStreamReader(sc.getInputStream()));
BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
String fromServer;
String fromUser;

while (ejecutar) {
    System.out.print("Numero:");
    fromUser = stdIn.readLine();
    if (fromUser != null && !fromUser.equals("-1")) {
        System.out.println("Cliente: " + fromUser);
        escritor.println(fromUser);
        if ((fromServer = lector.readLine()) != null) {
            System.out.println("Servidor: " + fromServer);
        }
    } else {
        ejecutar = false;
    }
}

out.close();
in.close();
stdIn.close();
sc.close();
```

Recibe pedido del usuario

Envía pedido al servidor

Recibe respuesta del servidor

Presenta la respuesta al usuario

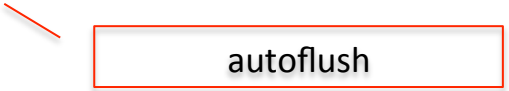
Ejecución

- El servidor y el cliente son procesos independientes
  - Se deben iniciar de forma separada
- El servidor debe iniciarse primero, siempre
  - Si un cliente intenta establecer conexión con un servidor que no está disponible, la conexión falla

Errores Frecuentes

- autoflush
  - Escribirá los datos en buffer si los métodos println, printf o format son invocados.
  - Si no se habilita, el lector quedará esperando el mensaje.

```
new PrintWriter(sc.getOutputStream(), true);
```



Errores Frecuentes

- accept
  - La instrucción accept crea una instancia del socket inicial que contiene los datos de un cliente particular.
  - El socket para intercambiar información con el cliente es sc (no ss).

```
ss = new ServerSocket (PUERTO);
while (true){
    Socket sc = ss.accept();
```

socket para comunicación con el cliente

Errores Frecuentes

- protocolo
  - Debe ser consistente
  - Si tanto el servidor como el cliente empiezan esperando un mensaje ambos quedan en espera

```
fromClient = entrada.readLine();
```

esperando mensaje del cliente

```
fromServer = lector.readLine();
```

esperando mensaje del servidor

Referencias

- http://docs.oracle.com/javase/tutorial/networking/sockets/index.html