Fundamentos de Redes (2017)
Grado en Ingeniería Informática
Universidad de Granada

Creación de un protocolo de Aplicación

Carlos de la Torre

19 de noviembre de 2017

Índice

1. Resumen/Abstract

Creación y definición de un protocolo de aplicación basado en el paradigma cliente-servidor, aunque las implementaciones que se han realizado son solamente una aproximación a dicho paradigma después de conseguir que las conexiones se realicen correctamente se pueden incluir las diferentes capas de seguridad e integridad a dicho protocolo.

Creation and definition of an application protocol based on the client-server paradigm, although the implementations that have been made are only an approximation to this paradigm after getting the connections done correctly, the different layers of security and integrity can be included. said protocol.

2. Descripción de la aplicación, funcionalidad y actores que intervienen

La aplicación pretender ser un servidor y cliente de compra de entradas de cine, para ello se llegara a una serie de asunciones de comunicación entre ambos intervinientes.

Básicamente la aplicaciones tendrá un servidor con una pequeña base de datos que contendrá las películas en las que podemos comprar entradas, el cliente que "habla.[el] mismo idioma que el servidor es que permitirá realizar las reservas de la entradas, mostrando al usuario una lista que previamente habrá recibido del servidor.

2.1. Servidor

Esta parte del protocolo se encargara de realizar la conversión entre la base de datos, (en nuestro caso esta base de datos es un array con la lista de películas) y una cadena de caracteres para poder enviar dicha cadena a través de un socket TCP utilizando la misma forma que se han usado en los anteriores ejercicios.

Esta parte también se encarga de recibir los diferentes mensajes del cliente y dar de baja la película por la que se realiza la compra.

A la hora de realizar el código se ha separado las diferentes acciones en clases, así de esta forma se puede implementar mas fácilmente que el servidor pueda atender a diferentes clientes al mismo tiempo.

Tal y como se muestra a continuación la parte importante del protocolo esta en la función *respuesta*

```
1         if (peticion == 9) {
2             for (int pelicula = 0; pelicula < this.lista.size(); pelicula++) {
3                 resultado = resultado + this.lista.get(pelicula) + ";";
4             }
5         } else {
6             resultado = "La pelicula elegida es: " + this.lista.get(peticion) + ";";
7             System.out.print("Ya no hay entradas para: " + this.lista.get(peticion) + "\n
      ");
8             this.lista.remove(peticion);
9         }
10        return resultado;
11    }
12 }
```

Listing 1: Función de respuesta del Servidor

2.2. Cliente

Esta parte del protocolo se encarga de mostrar la información al usuario, básicamente el cliente se encarga de recibir los datos en una cadena completa de datos pero separada por *;* y luego toma esta cadena y la presenta en pantalla como si fuera una lista.

Después de esto permite al usuario seleccionar cual es la película de la que quiere comprar la entrada y después manda el mensaje al servidor para que se encargue de realizar las operaciones pertinentes.

Como parte extra se podría implementar que el servidor devolviera una respuesta al cliente diciendo que se ha realizado correctamente la compra.

En el siguiente código se puede ver que el cliente primero recibe la lista del servidor y despues enviá su respuesta.

```
1         try {
2             // Creamos un socket que se conecte a "hist" y "port":
3             socketServicio = new Socket(host, port);
4             // esto lo usamos para enviar datos al servidor en modo texto
5             PrintWriter outPrinter = new PrintWriter(socketServicio.getOutputStream(),
      true);
6             // esto lo usamos para leer datos desde el servidor en modo texto
7             BufferedReader inReader = new BufferedReader(new InputStreamReader(
      socketServicio.getInputStream()));
```

```
            // Leemos la respuesta del servidor con las peliculas
            buferRecepcion = inReader.readLine();
            // Si queremos enviar una cadena de caracteres por un OutputStream
            // hay que pasarla primero a un array de bytes:
            buferEnvio = eleccion(buferRecepcion);
            // Enviamos el array por el outputStream;
            outPrinter.print(buferEnvio);
            // Aunque le indiquemos a TCP que queremos enviar varios arrays de bytes
            // sólo los enviará efectivamente cuando considere que tiene suficientes
            // datos que enviar... Podemos usar "flush()" para obligar a TCP a que no
            // espere para hacer el envío:
            outPrinter.flush();
            // Una vez terminado el servicio, cerramos el socket (automáticamente se
            // cierran el inpuStream  y el outputStream)
            outPrinter.close();
            inReader.close();

            // Excepciones:
        } catch (UnknownHostException e) {
            System.err.println("Error: Nombre de host no encontrado.");
        } catch (IOException e) {
            System.err.println("Error de entrada/salida al abrir el socket.");
        }
    }

    private static String eleccion(String datos) {
        // cadena que se va a enviar
        String envio;
        // Esto es para la entrada de teclado
        Scanner teclado = new Scanner(System.in);
        //lista de peliculas
        String[] peliculas = datos.split(";");
        // pintaos la lista
        for (int i = 0; i < peliculas.length; i++) {
            System.out.print(i+".- "+peliculas[i]+"\n");
        }
        System.out.print("El 9 repite la lista\n");
        // elejimos la pelicula
        System.out.print("Por favor elija que pelicula quiere ver\n");
        envio = teclado.nextLine();

        return envio;
    }
```

Listing 2: Codigo cliente

3. Diagrama de estados

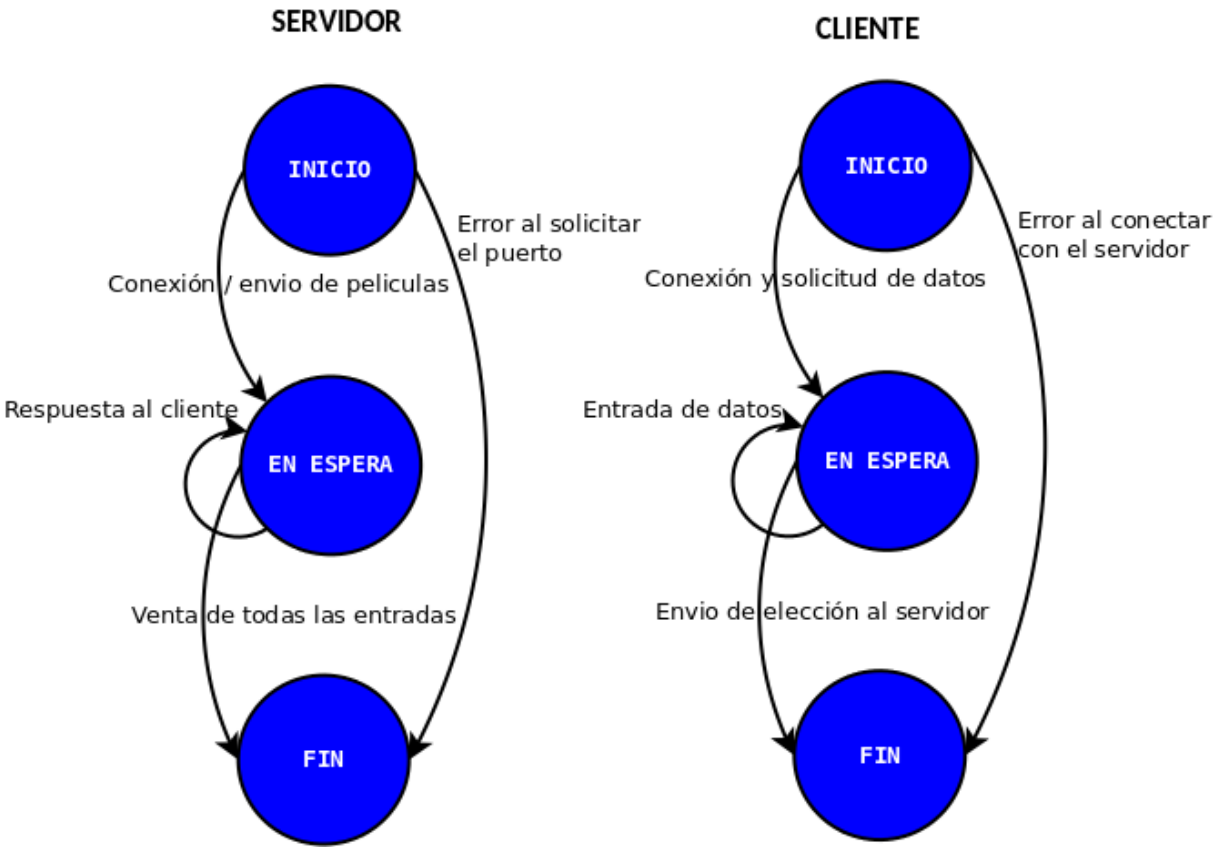

Figura 3.1: Diagrama de estados de los actores

4. Mensajes que intervienen

Los mensajes del servidor son:

Código	Cuerpo	Descripción
101	OK + respuesta(9)	Se realiza el envío de la lista de películas.
102	OK + "La película elegida es:" + elección del usuario.	Se contesta al cliente cual ha sido la compra realizada
103	Error + Error al escuchar en el puerto.	Este mensaje se da cuando el puerto de escucha del servidor esta ocupado.
104	Error + Error al obtener los flujo de entrada/salida.	Se produce cuando el se produce la conexión con el servidor pero no hay flujo de datos.

Los mensajes del cliente son:

Código	Cuerpo	Descripción
201	OK + Lista de peliculas.	Cuando se realiza la conexión se hace la lectura de la lista de peliculas
202	OK + Elección del usuario	Cuando el usuario elige la pelicula que quiere comprar
203	Error + Nombre de host no encontrado.	Error que se produce cuando no se ha podido encontrar el servidor
204	Error + Error de entrada/salida al abrir el socket.	Error que se produce cuando el cliente esta conectado pero no puede enviar datos

5. Conclusiones

Esta claro que para crear un protocolo de aplicación hay que definir muchísimo mejor la API de mensajes con las que se vana comunicar el cliente y el servidor, pero para tener una primera aproximación esta aplicación de compra de entradas simuladas es suficiente para darse cuenta de todos los pormenores que hay al crear un protocolo de aplicación complejo.