

FUNDAMENTOS DE REDES (2017-2018)
DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS
UNIVERSIDAD DE GRANADA

Definición e implementación de un protocolo de aplicación

Simón López Vico
Ana María Peña Arnedo
Alberto Jesús Durán López

22 de noviembre de 2017

Índice

0. Descripción de la aplicación, funcionalidad y actores que intervienen	3
1. Diagrama de estados del servidor	3
2. Mensajes que intervienen	4
2.1. Tabla de estados del servidor	4
2.2. Tabla de estados del cliente	4
3. Descripción del programa	5
3.1. Servidor	5
3.2. Cliente	5
4. Evaluación de la aplicación	6
5. Observaciones	8

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

Esta práctica consiste en la implementación de un protocolo de aplicación que consta de un servidor y dos clientes. El juego se basa en el famoso "Pilla-Pilla" donde un jugador 'X' tendrá que alcanzar a otro 'Y'. Para ello, ambos se conectarán con un nombre de usuario correcto almacenado en el servidor. Por tanto, los movimientos que ambos hagan se enviarán a través del servidor hacia el otro cliente.

1. Diagrama de estados del servidor

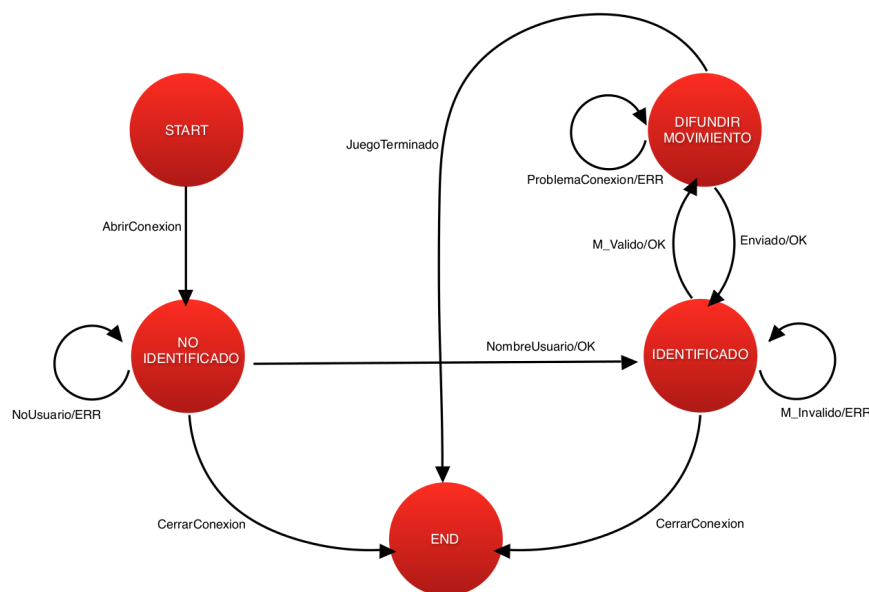


Figura 1: Diagrama de estados del servidor

Empezamos en el estado *START*. El cliente abre la conexión hacia el servidor. Por tanto, pasará al estado *IDENTIFICADO* si introduce un usuario correcto ó se quedará en el estado *NO IDENTIFICADO* hasta que se autentique correctamente. Una vez conectados ambos clientes, podrán realizar movimientos y pasar al estado *DIFUNDIR MOVIMIENTO* un número indefinido de veces o pasar al estado *END* si el juego se ha acabado. Cabe destacar que cada estado posee su propia comprobación de errores.

Para esta aplicación usamos *sockets TCP* ya que nos ofrecen un servicio de transmisión orientado a conexión en el que se garantiza que los datos llegan en el orden en el que fueron enviados, sin errores ni repetición, de manera que nos proporciona control de flujo extremo a extremo y control de errores de forma transparente

2. Mensajes que intervienen

2.1. Tabla de estados del servidor

MENSAJES QUE INTERVIENEN EN EL SERVIDOR

CÓDIGO	CUERPO	DESCRIPCIÓN
2001	ERROR+"No existe el usuario"	Servidor no reconoce el usuario
2002	ERROR+"Movimiento no válido"	Servidor no acepta el movimiento
2003	ERROR+"Problema Conexión"	Servidor no puede difundir movimiento
2011	DifundirMovimiento	Servidor reenvía posiciones
2012	CerrarConexion	Avisa a clientes de que el juego finaliza

Figura 2: Mensajes que intervienen en el servidor

2.2. Tabla de estados del cliente

MENSAJES QUE INTERVIENEN EN EL CLIENTE

CÓDIGO	CUERPO	DESCRIPCIÓN
1001	AbrirConexion	Cliente intenta conectarse
1002	NombreUsuario	Introduce su nombre de usuario
1003	Movimiento	Posición modificada se envía al servidor

Figura 3: Mensajes que intervienen en el cliente

3. Descripción del programa

3.1. Servidor

La clase servidor contendrá las posiciones de cada uno de los jugadores y los semáforos necesarios para garantizar la exclusión mutua, así como los nombres de usuario admitidos por el servidor. En el constructor del servidor se asignará una posición aleatoria a cada uno de los jugadores.

Esta clase estará compuesta por dos subclases, servidor uno y servidor dos, que heredan de la clase thread. Cada una de estas subclases contendrá un método 'run' similar. En dicho método el servidor estará esperando a que el cliente envíe su nombre de usuario. Tras recibirlo, comprobará que es correcto comparándolo con una lista de nombres aceptados por el servidor. Si no lo está, el servidor seguirá esperando hasta que el cliente envíe un nombre de usuario válido.

Si el nombre de usuario ya es válido, el servidor enviará la letra que corresponde a cada cliente, de manera que al primer cliente que se conecta se le asigna el carácter 'X' y al otro cliente se le asigna el carácter 'O'. Después, imprimirá por pantalla que el cliente se ha conectado al servidor e inmediatamente después le enviará su posición en el mapa.

A continuación, el servidor entrará en un bucle en el que comprobará, en cada iteración, que la distancia entre ambos jugadores es menor que la raíz de 2, es decir, que un jugador se encuentra en el 8-entorno del otro. Dentro de dicho bucle el servidor estará esperando a que el cliente le envíe su posición. Al recibirla, actualizará la variable que contiene la posición de dicho jugador y enviará a este cliente la posición del otro jugador.

Cuando el servidor salga de este bucle, enviará al cliente el mensaje 'FIN', lo que será interpretado por el cliente como que ha terminado el juego. Se imprimirá por pantalla el mensaje de que el cliente se ha desconectado y se cerrará el socket servicio.

3.2. Cliente

La clase cliente contendrá las posiciones del jugador controlado por el cliente en cuestión, así como las posiciones del contrincante. Al igual que en el servidor, en esta clase tendremos semáforos que garantizan la exclusión mutua, así como una hebra para conectarnos con el servidor. En el constructor del cliente se inicializarán cada uno de los semáforos y la hebra, y se lanzará para que empiece a ejecutarse.

El cliente contendrá una subclase que hereda de la clase thread. Dicha clase contendrá el nombre de usuario del jugador, y el método 'run' para lanzar la hebra.

El programa cliente se ejecutará acompañado de un argumento que constituirá el nombre de usuario correspondiente al cliente que desea conectarse al servidor. Una vez ejecutado, se enviará al servidor dicho nombre. El servidor, como hemos comentado, le responderá o bien con el mensaje *FAILED* o bien con el carácter correspondiente asignado al jugador. Si el mensaje es *FAILED*, se aborta el programa.

Al recibir el carácter correspondiente al jugador, se podría la posición asignada por el servidor al cliente. Seguidamente se imprimirá por pantalla el mapa incluyendo el símbolo del jugador en una determinada posición. En este momento, comienza el juego.

Se realizará un bucle del que se saldrá cuando el mensaje recibido desde el servidor sea *FIN*. En este bucle, se introducirá un carácter para efectuar un movimiento. Inmediatamente después, se enviará la nueva posición del jugador al servidor, y el cliente le pedirá al servidor la posición de su adversario.

Finalmente, al salir del bucle se imprimirá por pantalla *FIN DEL JUEGO* y se cerrará el socket de servicio.

4. Evaluación de la aplicación

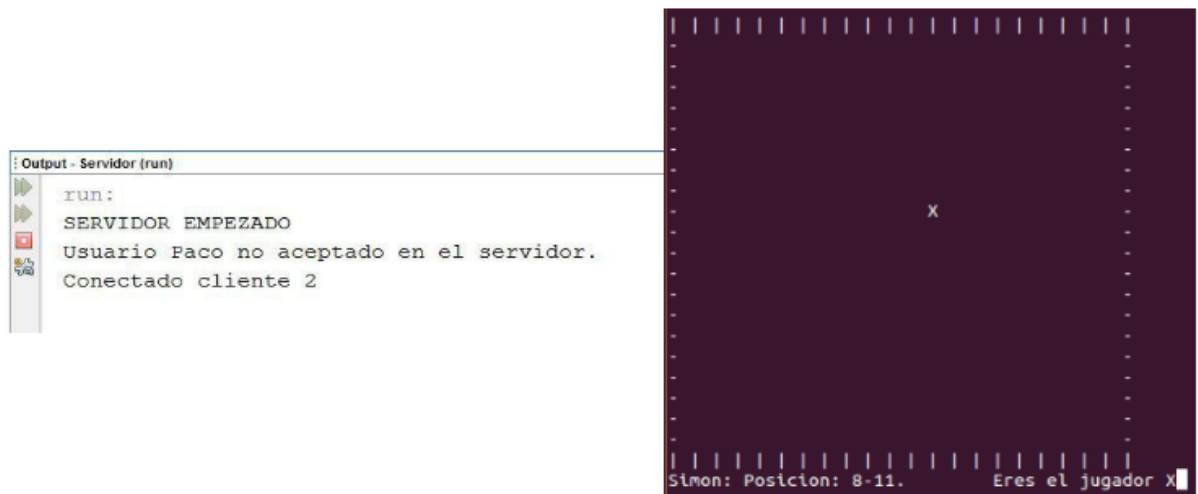


Figura 4: Autenticación errónea y correcta (asignación de letra)

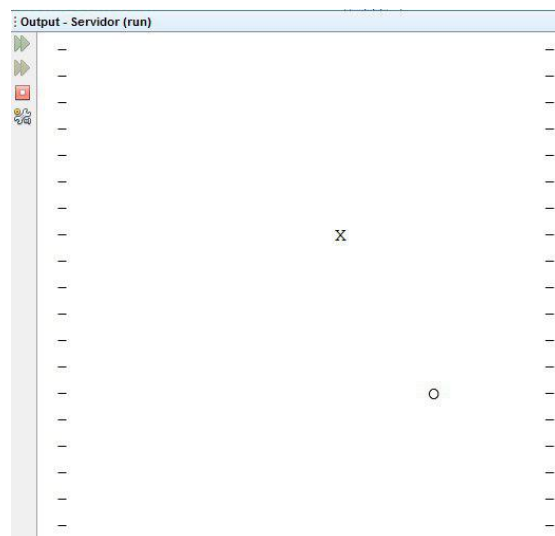


Figura 5: Vista del servidor cuando se conectan los dos clientes

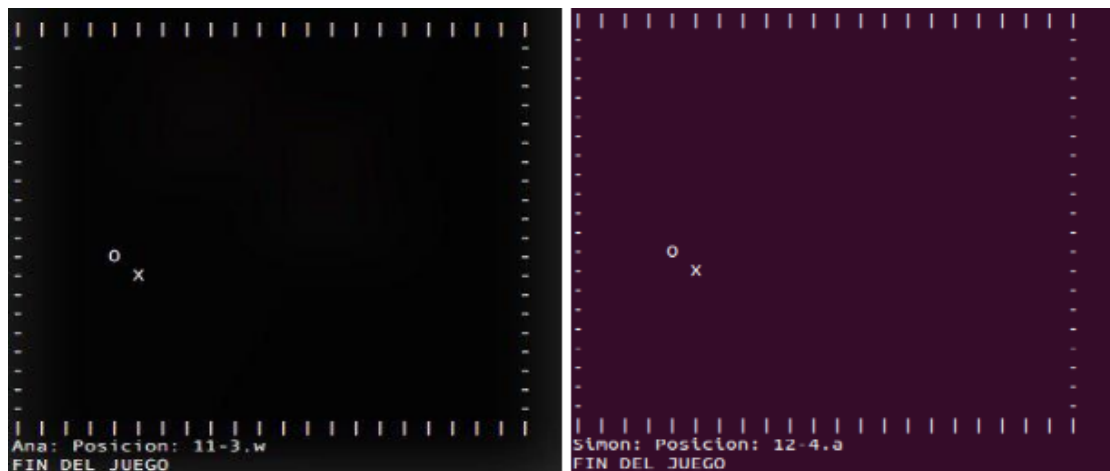


Figura 6: Fin del juego para ambos clientes

	Time	Source	Destination	Protocol	Length	Info
	81 15.194565	192.168.43.179	192.168.43.82	TCP	66	51535 → 2053
	82 15.283519	192.168.43.26	192.168.43.82	TCP	60	57573 → 2052
	83 15.302754	192.168.43.26	192.168.43.82	TCP	54	57573 → 2052
	84 15.580774	192.168.43.179	192.168.43.82	TCP	71	51535 → 2053
	85 15.591687	192.168.43.179	192.168.43.82	TCP	66	51535 → 2053
	86 15.620837	192.168.43.26	192.168.43.82	TCP	60	57573 → 2052
	87 15.650874	192.168.43.26	192.168.43.82	TCP	54	57573 → 2052
	88 15.947229	192.168.43.179	192.168.43.82	TCP	71	51535 → 2053
	89 15.954826	192.168.43.26	192.168.43.82	TCP	60	57573 → 2052
	90 15.956633	192.168.43.179	192.168.43.82	TCP	66	51535 → 2053
	91 15.978634	192.168.43.26	192.168.43.82	TCP	54	57573 → 2052
	92 16.289894	192.168.43.179	192.168.43.82	TCP	70	51535 → 2053
	93 16.303131	192.168.43.26	192.168.43.82	TCP	60	57573 → 2052
	94 16.303389	192.168.43.179	192.168.43.82	TCP	66	51535 → 2053
	95 16.320959	192.168.43.26	192.168.43.82	TCP	54	57573 → 2052
	96 16.613085	192.168.43.179	192.168.43.82	TCP	70	51535 → 2053
	97 16.620235	192.168.43.26	192.168.43.82	TCP	60	57573 → 2052
	98 16.626238	192.168.43.179	192.168.43.82	TCP	66	51535 → 2053
	99 16.644045	192.168.43.26	192.168.43.82	TCP	54	57573 → 2052
	100 16.930017	192.168.43.179	192.168.43.82	TCP	70	51535 → 2053

Figura 7: Captura de Wireshark. Comunicación a través de los puertos 2052 y 2053

5. Observaciones