UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA

FACULTAD DE INFORMÁTICA

Año académico 2017/18

GRADO EN INGENIERÍA INFORMÁTICA

REDES DE COMUNICACIONES (1904)

Ginés Miguel Cánovas Méndez

DNI: 23333196H

Daniel Ibáñez Molero

DNI: 48658416F

GRUPO 2 **SUBGRUPO 2**

CONVOCATORIA DE JULIO

Profesor: Oscar Cánovas Reverte

IMPLEMTENTACIÓN NANOGAMES

INDICE:

1. INTRODUCCION:

En este documento vamos a abordar lo último relacionado con nuestra implementación de nuestro servidor de juegos. Hablaremos finalmente de si nuestra estructura de mensajes y los autómatas de los mismos clientes y servidores diseñados en la anterior práctica han sido modificados o si hemos conseguido mantenerlos igual que estaban. Una vez finalizado dicho apartado iremos a los detalles de nuestra implementación del juego en la cual, hablaremos un poco sobre cómo han sido nuestros mensajes que recibe o mandan tanto nuestro cliente o servidor y las diferentes funcionalidades dentro del mismo. Finalmente daremos una conclusión sobre cómo nos ha parecido el juego y lo que hemos aprendido al implementarlo.

2. DISEÑO IMPLEMENTADO

En este apartado abordaremos lo elementos utilizados para la implementación de nuestro juego en base a las correcciones que nos indicó nuestro profesor.

2.1 FORMATO DE MENSAJES

El formato de mensajes que nuestro profesor nos indicó era usando los campos como bytes. Este mensaje estaba compuesto de unos campos los cuales podían estar compuestos por 1 o más bytes. Estos campos guardaban el código de operación cual fuese y los demás bytes se encargaban de guardar la información que debía ser usada en cada código.

El formato de mensaje es el siguiente:

1 byte(código)	N bytes (información)	N bytes (información)	Etc…

A continuación vamos a mostrar todos los mensajes que hemos realizado en nuestro juego:

Token, EnterGame:

1 byte (código token)	N bytes(generado por el token)

Un primer campo tendremos siempre nuestro código y en la segunda parte se almacena un long en N bytes.

Nick, Rules, Status, gameMessage:

1 byte (código)	1 byte (longitud)	N bytes (String con información)

Como podemos observar cada mensaje tendrá unos campos y un tamaño dependiendo de la información que queramos transmitir. Estos mensajes serán explicados más adelante.

ControlMessage, Nick ok:

1 byte (código de comprobación)

Nos manda un mensaje de control en caso de que se acierte o falle los mensajes que procesamos y únicamente necesitamos el código de comprobación.

ListMessage:

1 byte código	1 byte Longitud total de cadenas	Longitud del String 1 byte	Bytes del String

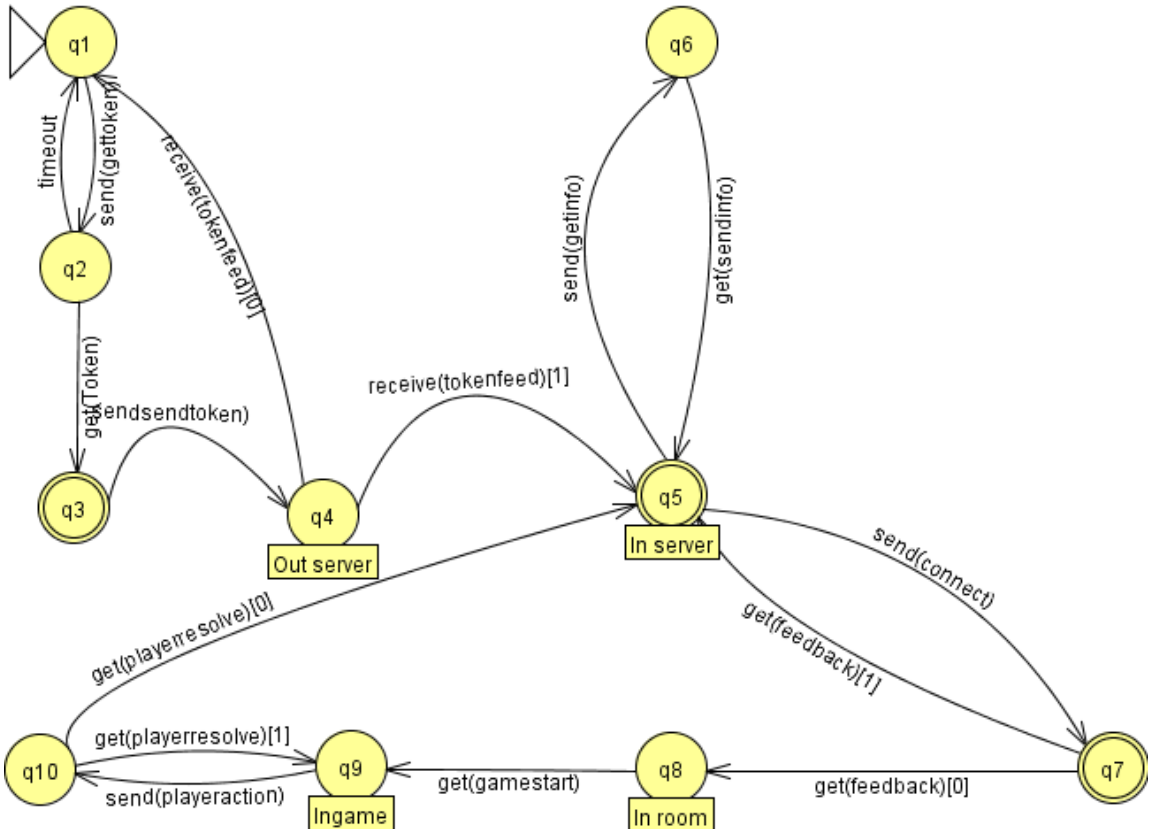
Puede haber más campos dependiendo de la longitud total de cadenas que contengamos. Este tipo de mensaje será usado para mostrar las salas.

En la parte de implementación hablaremos un poco de los mensajes y como han sido modificados y codificados.

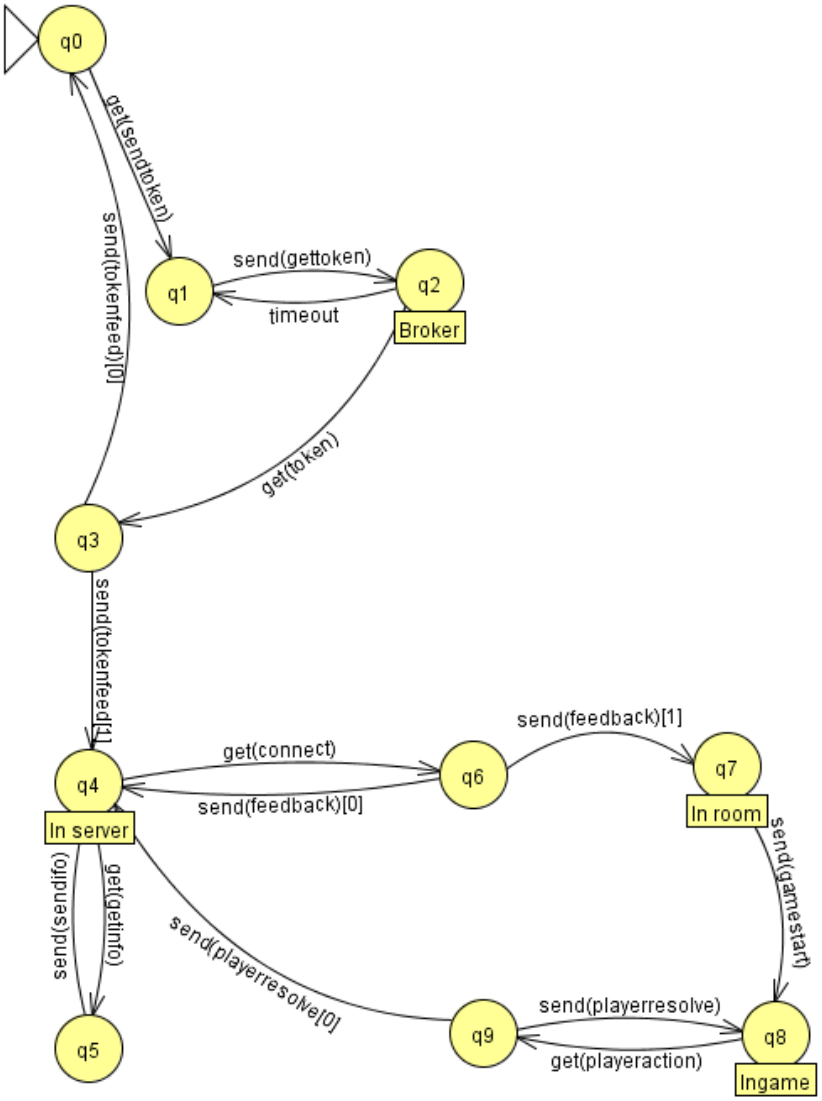
.

2.2AUTOMATAS DE CLIENTE Y SERVIDOR

A continuación mostraremos los autómatas que han sido relacionados con nuestro cliente y servidor, el cual a nuestro parecer no ha sido modificado puesto que cumple con lo que están implementado los estados.



Este primer autómata se trata de cómo funciona nuestro cliente.



Este segundo autómata se trata de cómo funciona nuestro servidor.

3. DETALLES ASPECTOS EN IMPLEMENTACION

A continuación vamos a mencionar los detalles de diferentes partes dentro de nuestro juego para ver cómo han sido implementados y cómo funcionan dentro del mismo.

3.1FORMATO DE LOS MENSAJES

En esta sección hablaremos un poco de cómo y porqué están implementados los diferentes mensajes mencionados anteriormente. Como sabemos dichos mensajes son heredados de nuestra clase **NGMessage** la cual se encargará de mostrarnos y gestionar todos los mensajes que le llegan a nuestro servidor y el mismo direccionarlos a nuestros clientes. Los mensajes que hemos implementado en nuestro juego son los siguientes:

NGTokenMessage:

Este mensaje recibe un token que es generado automáticamente por el usuario que quiere conectarse y consigue acceder a nuestro servidor. Este mensaje como hemos representado arriba contará con un byte para su código de recibir o mandad confirmación y los bytes de nuestro token que es un long.

NGNickMessage:

Este mensaje recibe una cadena la cual tendrá el nombre del usuario que quiere conectarse y jugar con los juegos que tenemos en nuestro servidor. Este mensaje es mandado por nuestro cliente y es recibido por nuestro servidor el cual intentará darle una forma u otra.

NGControlMessage y NGNickMessageOk:

Estos dos mensajes se encargar de mandar información sobre si un mensaje recibido ha sido correcto o no. Son procesados por nuestro servidor hacia los diferentes jugadores.

NGListMessage:

Este mensaje se encargar de mostrar por pantalla las salas que tenemos dentro de nuestro servidor. Es un mensaje que manda el servidor para que una vez tecleado el comando roomlist nos llegue la información de nuestras salas.

NGEnterMessage:

Este mensaje se encargar de mandar al servidor que un jugador quiere conectarse a una sala. Dicho mensaje llevará la información de nuestro código y además un entero donde nos indicará que sala es a la que quiere enterar.

NGRulesMessage, NGStatusMessage, NGGameMessage:

Estos mensajes que quedan son los implementados por nuestro servidor y lo que harán será mandar información acerca de nuestro juego como son las reglas, el status de nuestra sala y los mensajes del juego.

3.2 MECANISMO DE GESTIÓN SALAS

El mecanismo que usamos nosotros para gestionar nuestras salas el siguiente:

Siempre que un servidor se conecta automáticamente nuestro juego genera dos salas de nuestro juego implementado por defecto. Esas salas se irán llenando conforme vayan entrando jugadores. Una vez llenas las salas por defecto el servidor generará automáticamente una nueva sala con la misma información inicial que tenían las otras.

Cuando un jugador intenta salir de la sala la sala echa por defecto a los dos jugadores y se resetea a sí misma para vaciarla de jugadores, puntuación…

Finalmente hemos pensado en mantener las salas en el servidor puesto que cuando se generan es porque el tráfico en el servidor ha sido alto y podría volver a darse el mismo o incluso más.

3.3 IMPLEMENTACIÓN LÓGICA DEL JUEGO

En este último apartado hablaremos un poco de cómo va a funcionar nuestro juego. Nuestro juego va a comenzar una vez dos jugadores entren en la partida. Esos jugadores empezarán a jugar y nunca el juego los echará puesto que, lo que tendrán al lado del contador de vidas es un contador que nos mostrará las partidas que han ido ganando durante el juego.

Si un jugador quiere salir tecleará el comando exit y el otro como no sabe si ha salido contestará. Cuando comprueba que el otro jugador ha salido el socket se cierra y automáticamente manda un status diciendo que el jugador ha abandonado la partida y tenemos que echarte.

Finalmente el juego irá recibiendo respuestas y actualizando sus status y sus challenges dependiendo de lo que mandaran los oponentes.

4 CONCLUSIONES

A modo de conclusión destacar que gracias al juego comprendemos bien el paso de mensajes que se han ido explicando tanto en las clases de teoría y prácticas. Para nosotros haber realizado este proyecto nos ha ayudado a ver cómo funciona aunque por supuesto con mayor complejidad los diferentes juegos online que existen. Para finalizar decir que el proyecto ha sido largo y algunas veces costoso pero gracias a las explicaciones de los profesores hemos conseguido terminarlo.