Informática II
Práctica 2: MasterMind en red y concurrente.

GSyC
Departamento de Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación

Diciembre de 2022

1. Introducción

Partiendo de la base de la práctica **MasterMind en Red de la práctica 1** desarrollada en esta asignatura, vamos a comenzar la implementación del juego en modo multijugador.

Los jugadores, podrán crear y unirse a partidas creadas por otros jugadores, de forma que se deberá gestionar la información de las mismas en el servidor. A través de esta implementación, dos jugadores podrán jugar una misma partida de forma que el jugador1 (jugador que crea la partida) espera a que otro jugador (jugador2) elija participar de ella. El jugador que cree la partida será el encargado de seleccionar el número de turnos que se jugarán.

En esta práctica, cada juego tendrá dos rondas, una primera ronda donde el **jugador2** creará la combinación ganadora y el **jugador1** deberá encontrarla. Cuando se finalice la primera ronda, se volverá a jugar con el mismo número de turnos seleccionado, siendo el **jugador1** el que cree la combinación ganadora en este caso y el **jugador2** el que deberá encontrarlo.

Se podrán crear y participar en tantas partidas como se desee en un único servidor de juego. Todas las partidas creadas serán almacenadas en un diccionario que permitirá almacenar de forma global todas las partidas en un servidor.

La estructura compartida de diccionario será accesible por todos los hilos en ejecución del programa, por lo que se deberá usar como una variable **global** que almacene cada una de las instancias del MasterMind en juego.

Además de las partidas almacenadas en una estructura compartida, se deberá guardar la información de los participantes en una **lista enlazada**, que guarde IP, puerto y el id de la partida.

2. Requisitos Generales

Los requisitos que debe cumplir la práctica son los siguientes:

- Se deberán mantener las mismas características que se implementaron en la versión anterior.
- El servidor debe ser capaz de ejecutar múltiples partidas de dos jugadores.
- Utilizará un diccionario para guardar cada instancia de MasterMind y una lista enlazada para almacenar la información los jugadores.
- Todas las partidas estarán definidas por un identificador (**número de partida**) que se utilizará como clave en el diccionario. Por lo general, ese identificador se podrá usar una **variable global** que marque el número de partidas definidas en el servidor.
- Deberá utilizar el paradigma de programación orientada a objetos siempre que se pueda.
- El desarrollo de la práctica no tendrá una única solución sino que cada solución desarrollada puede ser válida.
- La práctica podrá ser realizada por grupos de hasta 3 personas. En el código fuente deberá aparecer la autoría de los alumnos que participan en el desarrollo de la práctica.

3. Requisitos específicos del cliente

3.1. Argumentos

Se utilizará la misma estructura que se ha usado en la práctica anterior.

3.2. Lógica de juego del cliente

El cliente deberá seguir siendo ligero en los diferentes cálculos que va a ejecutar, de forma que no deberá realizar ninguna opción y su única función es enviar las combinaciones que el usuario introduzca por teclado y recibir las órdenes que le dicte el servidor conforme a cada uno de los movimientos realizados.

De esta forma, el principal uso del cliente será el tratamiento de cadenas de caracteres. En el apartado 5 se mostrarán los nuevos mensajes del protocolo que deberá ser capaz de interpretar el cliente. Se podrán reutilizar los mensajes que envíe el servidor para las respuestas del cliente siempre que se considere oportuno.

4. Requisitos específicos del servidor

4.1. Argumentos

Se utilizará la misma estructura que se ha usado en la práctica anterior.

4.2. Fases del juego

El juego será en todo momento controlado por el servidor, y para su correcto funcionamiento tendremos diferentes fases donde se desarrollarán las diferentes configuraciones de la partida.

- **Fase unión:** En esta fase el jugador se presentará al servidor y le mandará el nombre que ha elegido. El servidor deberá enviarle un menú inicial que permitirá elegir si quiere crear una nueva partida o unirse a alguna de las partidas ya definidas. En este segundo caso, una vez que el jugador le envíe la opción seleccionada, el servidor deberá enviarle una lista de partidas que no tengan los dos jugadores activos. (Apunte: Podéis usar el identificador de partida del diccionario como selector de esa partida a la que unirse).
- **Fase inicialización:** En esta fase se define el número de turnos que se van a jugar antes de la finalización de la partida. Será obligación del jugador que crea la partida definir el número de turnos y ese valor se mantendrá hasta la finalización del juego entre los dos jugadores. Se deberá mantener que el cliente deberá enviar al servidor el número de turnos. En este caso, el servidor deberá responder positiva o negativamente dependiendo de si es un valor correcto o no.
- **Fase de configuración de la palabra secreta:** El jugador que no vaya a adivinar la combinación (jugador1 en la primera ronda, jugador2 en la segunda ronda) deberá enviarle al servidor la palabra secreta que permita comenzar el juego. Se puede usar una palabra reservada para denotar al servidor que genere aleatoriamente una combinación secreta. Aunque se puede usar cualquier palabra, siempre que se notifique en los mensajes de control, se ha definido inicialmente la palabra clave **nocombiCode**.
- **Fase juego:** Se realizarán dos rondas (una para cada jugador) en las cuales se deberá encontrar la combinación secreta (de la misma forma que se ha hecho hasta ahora). La información de la partida no sólo le deberá llegar al jugador que está buscando la combinación, sino que deberá ser enviada al jugador que no está . El modo de funcionamiento será el mismo que se tenía hasta ahora, se pedirá una combinación en el cliente y será enviada al servidor que validará si es correcta o no. Esto se repetirá hasta que el jugador se quede sin turnos o que el jugador gane. Por cada turno que se juegue el servidor deberá enviar de vuelta a los dos clientes (jugador1 y 2) información de los acierto así como de los semiaciertos que ha tenido el jugador en esa ronda. Una vez finalizada la primera ronda, el segundo jugador será el que pase a tener el control del juego y a resolverlo. Tras la segunda ronda, ganará el jugador que menos turnos haya tardado en encontrar la solución.

5. Mensajes del protocolo

En esta sección se describen los nuevos mensajes necesarios para implementar las nuevas funcionalidades. Dado que la comunicación entre el cliente y el servidor será a través del intercambio de cadenas de caracteres convertidas a información binaria, hemos de definir los mensajes de control. Todos los mensajes desarrollados en la práctica anterior deberán seguir siendo enviados.

5.1. Fase de unión:

Son los mensajes que se van a intercambiar entre el cliente y el servidor a la hora de conectarse la primera vez y decidir si va a crear una partida o a conectarse a alguna ya creada.

- ## Mensaje: Saludo

Tipo	Nombre

Donde:

- **Tipo** será el tipo de mensaje de unión al servidor enviado por un cliente. Ejemplo: HELLO
- **nombre** será el nombre del jugador que se une al servidor obtenido de los parámetros de la aplicación.

- ## Mensaje: Menú

Tipo	InformacionMenu/Opción

Donde:

- **Tipo** será el tipo de mensaje de gestión del menú del jugador. Ejemplo: MENU
- **InformacionMenu/opción** contendrá la información relacionada con el menú que el servidor envíe al cliente. Este mismo mensaje Menú puede ser usado por el cliente para notificarle al servidor la opción que se ha elegido. Ejemplo: MENU#2

- ## Mensaje: Partidas en curso

Tipo	mensaje/Partidas

Donde:

- **Tipo** será el tipo de mensaje de envío de partidas abiertas. Ejemplo: GAMES
- **mensaje** este mensaje será enviado por el servidor al cliente y contendrá la información relacionada con las partidas abiertas que no tengan un jugador. Ejemplo: GAMES#1 - Partida de David.2 - Partida de Jesús.

- ## Mensaje: Selección de Partida

Tipo	Partida

Donde:

- **Tipo** será el tipo de mensaje de selección de la partida. Ejemplo: GAME
- **Partidas** este mensaje será enviado por el cliente al servidor al cliente notificando cuál de las partidas abiertas es a la que se unirá. Ejemplo: GAME#1.

- ## Mensaje: Mostrar Mensaje y esperar nueva información

Este mensaje será usado para notificar alguna información al cliente y que deba esperar otro nuevo mensaje por parte del servidor.

Tipo	mensaje

Donde:

- **Tipo** será el tipo de mensaje de selección de la partida. Ejemplo: WAIT
- **mensaje** este mensaje será enviado al cliente para mostrar cierto mensaje de espera. Ejemplo: WAIT#Configurando la partida en el otro jugador.

5.2. Fase de inicialización:

En esta fase el creador de la partida deberá enviar el número máximo de turnos con el que se jugará.

Tipo	mensaje/turnos

Donde:

- **Tipo** será el tipo de mensaje de unión al servidor. Ejemplo: TURN

- **mensaje/turnos** Este mensaje será enviado desde el servidor al cliente y contendrá la cadena de texto que pedirá al usuario introducir el número máximo de turnos. Ejemplo: TURN#Introduce el número de turnos. También puede ser usado el mismo tipo de mensaje para recibir del cliente el número de turnos elegido. Ejemplo: TURN#15

5.3. Fase de configuración de la palabra secreta:

En esta fase el jugador que no vaya a buscar la palabra deberá seleccionar la combinación secreta.

Tipo	mensaje/turnos

Donde:

- **Tipo** será el tipo de mensaje de gestión de la clave secreta. Ejemplo: KEYS
- **mensaje/turnos** Este mensaje será enviado desde el servidor al cliente y contendrá el mensaje que le pedirá al cliente que introduzca la combinación ganadora. Ejemplo: KEYS#Introduce combinación ganadora o nocombiCode. También puede ser usado este tipo de mensajes entre el cliente y el servidor para notificar la combinación ganadora. Ejemplo: KEYS#bbbb

5.4. Fase de juego:

Durante el juego se intercambiarán ciertos mensajes en donde el servidor enviará información vital al cliente para mostrar al jugador, o el cliente deberá enviarle información al servidor sobre la tirada actual.

- ## Mensaje: Nueva Partida y Fin de Visualización

  Ambos mensajes serán usados para finalizar la partida actual (una vez que no haya turnos) y comenzar la segunda ronda.

Tipo	mensaje

  Donde:

  - **Tipo** será el tipo de mensaje. Ejemplos: NEWGAME o ENDVIEW. En el primer caso será enviado por el servidor al jugador1 (partida en curso) para finalizar la partida y comenzar otra nueva en la que ese jugador pasará a tener el rol de jugador en modo visionado. El segundo caso es para la finalización del modo visionado (jugador2) y crear una nueva partida donde ahora será este jugador el que tenga que encontrar la palabra clave
  - **mensaje** Mensaje que deberá mostrar el cliente. Ejemplo: NEWGAME#Volviendo a conectar jugadores.
  - **Nota**: Puede ser necesario en alguno de estos mensajes enviar más información por parte del servidor a los clientes o viceversa.

Tipo	mensaje/turnos

  Donde:

  - **Tipo** será el tipo de mensaje de gestión de la clave secreta. Ejemplo: KEYS
  - **mensaje/turnos** Este mensaje será enviado desde el servidor al cliente y contendrá el mensaje que le pedirá al cliente que introduzca la combinación ganadora. Ejemplo: KEYS#Introduce combinación ganadora o nocombiCode. También puede ser usado este tipo de mensajes entre el cliente y el servidor para notificar la combinación ganadora. Ejemplo: KEYS#bbbb

- ## Mensaje: Fin de la partida

Tipo	mensaje

  Donde:

  - **Tipo** será el tipo de mensaje de finalización de la partida. Ejemplo: OVER
  - **nombre** será el mensaje enviado a ambos clientes para finalizar el juego. Ejemplo: OVER# Ha ganado David en 3 movimientos.

6. Consejos

- La clase MasterMind deberá adaptarse de forma que ahora entre los diferentes atributos que almacena guarde una lista de los **sockets** de los jugadores que están usando esa instancia del juego así como una lista con los **nombres de los jugadores**. Es importante tener en cuenta el orden en ambas listas para controlar quién es el jugador1 y quién el jugador2 en ambos turnos. Puede que sea necesario implementar algún método más para gestionar la información que almacena esta clase por ejemplo para saber el turno en el que se está actualmente en la partida.
- Os aconsejamos a usar el MasterMind en esta segunda práctica como un objeto completo (ya que lo tenéis que almacenar en una lista) y no implementado en el hilo del servidor.
- **Es muy importante** que al almacenar el socket que se ha creado al aceptar la conexión en el servidor (constructor de la clase hilo del servidor) uséis la clase **newSocket** que os proporcionamos para evitar tener problemas a la hora de enviar y recibir información. Ejemplo: `self.__client_socket = newSocket(client_socket)`
- Para el correcto funcionamiento de los hilos y evitar el bloqueo entre ellos es importante saber a qué cliente se le debe enviar la información. Para ello es necesario usar variables globales que serán definidas en el script del **hilo del servidor**. Estas variables locales deberán almacenar el contador de número de partidas abiertas actualmente (Identificador de partida), un diccionario que almacene todas las instancias de las partidas y una lista enlazada que almacene todos los participantes (esto último es obligatorio).
- Cualquier mensaje que se haya definido en esta memoria puede ser modificado y adaptado a las necesidades de cada programa.
- Es **muy importante** pensar en cómo ambos hilos lanzados en el servidor deben colaborar entre sí enviando mensajes a ambos clientes. Por ejemplo: cuando se une a la partida el segundo jugador, el el hilo del servidor al que se notifica que se ha unido el jugador es el que está vinculado con ese jugador (jugador2), de forma que será este hilo el que deberá notificar al jugador1 que se entra en la fase de inicialización y que deberá enviar a su hilo los turnos.

7. Entrega

La entrega de esta práctica se hará a través del aula virtual, con límite: **18 de enero de 2023 a las 23:55h**.