Instituto Tecnológico Autónomo de México



Sistemas Distribuidos

Proyecto Alpha: Pégale a monstruo

Miguel Ángel Reyes Badilla ¿?????

Mónica Elizabeth Alba González 160502

Fecha de entrega: 15 de marzo de 2019

Índice

Introducción

El presente documento describe nuestra solución al proyecto de la materia de sistemas distribuidos: Pégale al monstruo. El cual consta de dos partes esenciales: el servidor, quien provee las posiciones de los monstruos, y el cliente, quien se encarga de “golpear” al monstruo y responderle enviar al servidor su respuesta. La comunicación que se da entre ambos se establece de tres maneras distintas. La primera es mediante RMI y se enfoca únicamente al registro de jugadores. La segunda es mediante multicast (UDP) con la que el servidor envía a todos los usuarios la posición del nuevo monstruo o el ganador de una ronda. Finalmente, la respuesta del jugador se realiza mediante TCP, mandando su nombre, id y la posición correcta. A continuación, explicaremos detalladamente la estructura general de nuestra solución.

Del lado del servidor: La primera acción que se realiza al ejecutarse el programa es la creación del estado del juego, en donde se almacenan los jugadores con sus respectivos datos, el ganador actual, el número de ronda actual y el número de golpes que tiene que hacer un usuario para poder ganar. Posteriormente, el servidor levanta el servicio RMI que contiene los métodos necesarios para que los usuarios (jugadores) se puedan dar de alta en el juego. Además, crea un hilo aparte para poder recibir los mensajes TCP de los usuarios y otro hilo para el envío de mensajes multicast, los cuales se envían automáticamente después de 1.5 segundos. Una vez que el servidor haya recibido una respuesta TCP, verifica que sea el primer mensaje que haya recibido en esa ronda, de ser así al jugador se le suman 5 puntos. Cuando un jugador haya llegado a los 25 puntos acumulados, se manda un mensaje multicast indicando el nombre del ganador, y se continúa con la siguiente ronda.

Del lado del usuario: Al ingresar el usuario su nombre, se registra en el juego a través del RMI, si ya hay un usuario ya registrado y se encuentra activo, tendrá que elegir otro nombre, si no, puede empezar (o regresar) a jugar. Después, lanza un hilo para poder escuchar las posiciones de los monstruos, y crea una conexión TCP únicamente cuando haya acertado en a posición del monstruo; esta se destruye una vez que dicha respuesta haya sido enviada al servidor. Un jugador se puede salir en cualquier momento del juego, al presionar el botón de salir, se cierran sus conexiones y se cambia su estado de “activo” a “inactivo” para que, si desea volver a jugar, no pierda su puntaje.

Diseño de la implementación

A continuación se mostrará el diagrama que describe la relación entre las clases de nuestra solución.

Servidor

Levanta RMI

Escucha mensajes TCP

Envía mensajes multicast

Cliente

Busca RMI

Escucha mensajes multicast

Envía respuesta TCP

RMI

Resultados de estresamiento

El proyecto fue sometido a varias pruebas de estresamiento para determinar cuál era su alcance, para las cuales seguimos los siguientes criterios:

* Un número pequeño de excepciones es aceptable (máximo el 5% de las llamadas totales).
* Para agilizar las pruebas de estrés, el tiempo de espera al enviar los mensajes se puede modificar cuando se simulan pocos jugadores.
* Realizar 10 veces el experimento con el número determinado de jugadores.
* Enviar 50 monstruos en cada prueba.
* El tiempo medido se realiza desde que se envía un mensaje multicast, hasta que se registran los cambios de los puntajes de los jugadores (después de recibir el tcp de respuesta).

Las pruebas se realizaron en dos computadoras distintas:

* Asus: Procesador i7-8550U a 1.80GHz, RAM de 8 GB, Sistema Operativo Windows 10 de 64 bits, 930GB de memoria.
* AQUÍ

En la primera computadora se realizaron las pruebas con 50, 100, 150, 200, 250, 300, 350 y 360 jugadores, con 10 repeticiones cada uno. Para los primeros 3 casos consideramos medio segundo entre cada envió de monstruo por parte del servidor. Para 200 jugadores, consideramos un segundo entre cada envío; y todos los demás se quedaron fijos con 1.5 segundos.

Se puede notar que la desviación estándar de casi todos los casos es muy alta, esto se debe a que cuando son varios jugadores los que responden casi a la vez, el tiempo de recorrido del mensaje tarda considerablemente más, debido al congestionamiento de los canales de comunicación.

Finalmente, la prueba se concluyó en 360 clientes, ya que los errores, aunque eran una porción muy pequeña respecto al total de mensajes enviados, eran más comunes que en los otros casos. Sin embargo, si se le hubiera aumentado el tiempo de envío de mensajes multicast, hubiéramos podido mejorar solamente un poco este resultado. Este ajuste ya no lo hicimos debido a que 1.5 segundos ya sería un lapso de tiempo considerable al momento de jugar.

===Poner grafiquitas por casos, señalando a parte el promedio, el promedio de los errores y la desviación ===