INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Sistemas Distribuidos
Profesor José Octavio Gutiérrez García

Evaluación de desempeño del juego distribuido *Pégale al monstruo*

Rodrigo Gil Hernández 182473
Yosshua Eli Cisneros Villasana 179889
Carlos Alberto Delgado Elizondo 181866

27 de marzo de 2022

Estresamiento del juego concurrente
Pégale al monstruo

Introducción

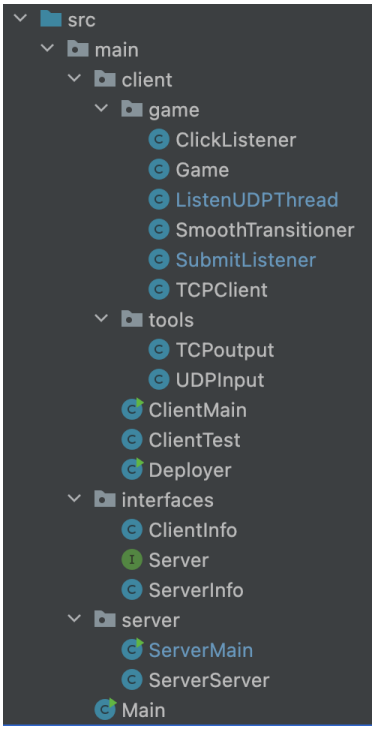
Se desarrolló el juego *Pégale al Monstruo*, que es una implementación distribuida del juego *Whac-A-Mole* y lo estresamos para ver cómo se comporta al variar el número de clientes.

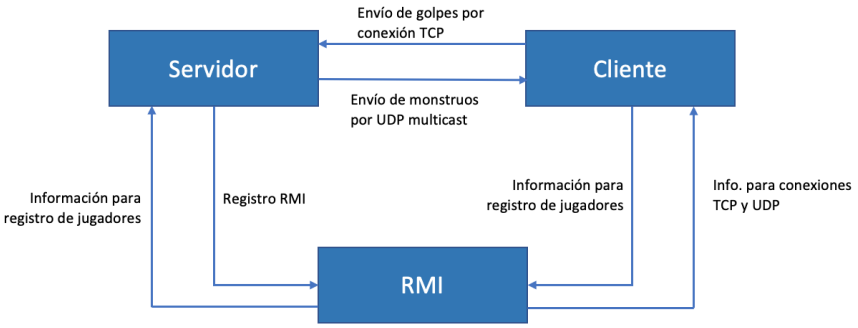
Se trata de un juego donde se despliega un monstruo al que un conjunto de jugadores le tiene que pegar. El primero en pegarle gana un punto y gana quien acumule una cantidad de puntos fijada al inicio de la partida.

El juego se ejecuta de forma distribuida. Los jugadores pueden conectarse al mismo juego, introduciendo la dirección IPv4 del servidor.

El diagrama de clases se puede observar en el diagrama de la izquierda.

Hay tres grandes entidades: el servidor, el cliente y el servicio RMI cuya interacción se describe en el diagrama. Las clases ClientTest, TCPoutput, UDPInput, Deployer se crearon para estresar al sistema.





Se estresa el juego para ver qué tan bien se comporta el juego al variar el número de clientes. Esto es necesario en caso de que quisiéramos liberar el juego y que veamos cuál es el máximo de jugadores para el que proveemos un buen servicio.

Descripción del experimento

Para estresar el servidor se hará que el cliente envíe el monstruo que le acaba de llegar por multicast UDP por TCP al servidor. Se trata de **n** cantidad de clientes compitiendo por quien alcanza primero los **15 ó 20 puntos**. Evaluaremos 5 métricas

- tiempo de respuesta promedio al dar golpe (por lado del cliente)
- desviación estándar promedio
- número de mensajes enviados promedio
- tiempo de respuesta instancia RMI promedio
- tiempo de respuesta instancia RMI desviación estándar

- **Tiempo de respuesta promedio al dar golpe (por lado del cliente)**: Mide el tiempo del lado del cliente desde que se envía el mensaje TCP hasta que regresa
- **Tiempo de respuesta promedio al dar golpe**:Mide la desviación estándar del tiempo del lado del cliente desde que se envía el mensaje TCP hasta que regresa.
- **Tiempo de respuesta promedio al dar golpe**: Mide el número promedio de mensajes enviados por cliente, este usualmente es el mismo ya que se disparan cuando reciben dónde está el monstruo por UDP multicast.
- **Tiempo de respuesta instancia RMI promedio:** Mide el tiempo desde que se inicia la instanciación del RMI hasta que se usa el método del RMI (solo existe un método RMI que le da la información al cliente).
- **Tiempo de respuesta instancia RMI desviación estándar:** Mide la desviación estándar del tiempo desde que se inicia la instanciación del RMI hasta que se usa el método del RMI

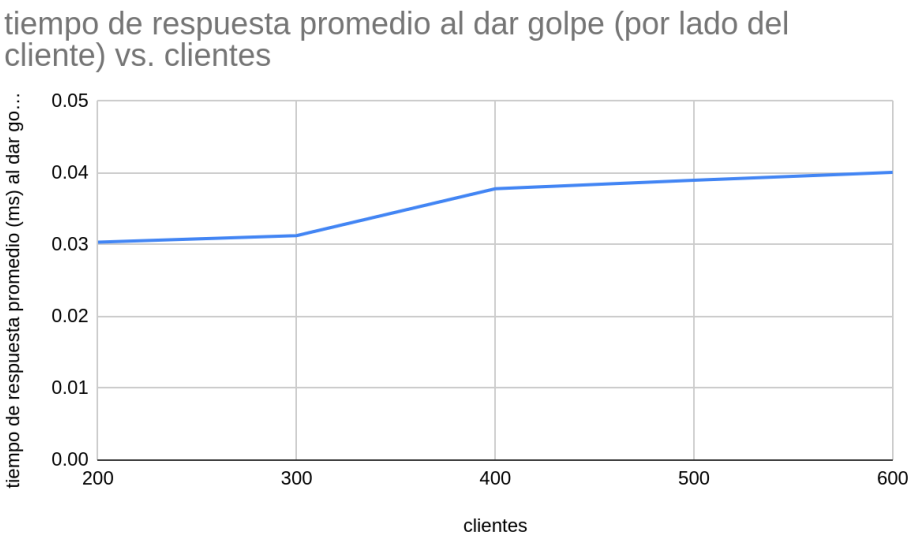
El experimento se desarrolló en dos máquinas distintas:

- **PC - 1:** WINDOWS 10 AMD Ryzen 7 1700X Eight-Core 4.0 GHZ - 16GB RAM 3200 MHZ
- **PC - 2:** WINDOWS 10 AMD Ryzen 5 3550H Eight-Core 2.1 GHZ - 16GB RAM 2400 MHZ

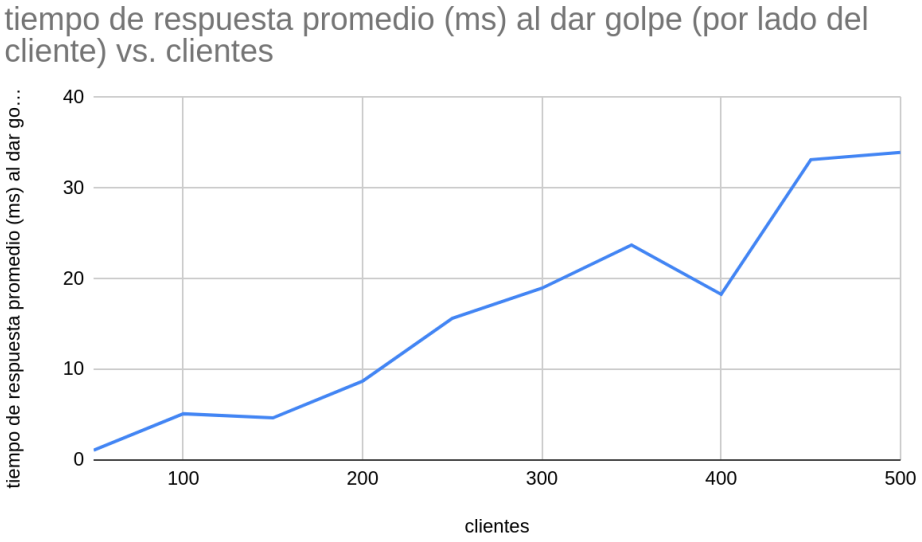
Resultados y conclusiones

Existen ciertas tendencias que se pueden deducir de las gráficas de los resultados; por ejemplo, en ambos casos, el **tiempo de respuesta promedio al dar golpe** sube a medida que aumenta el número de clientes

PC1:



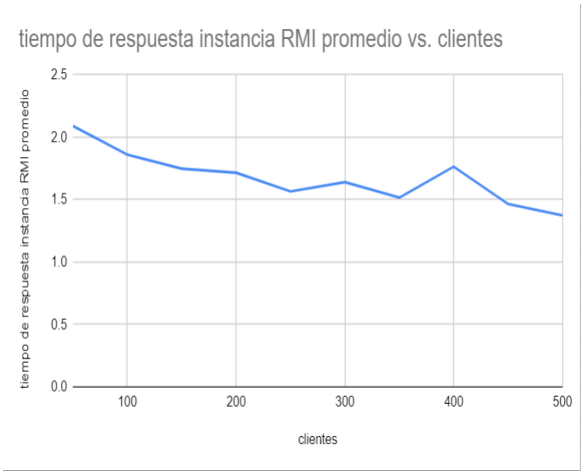
PC2:



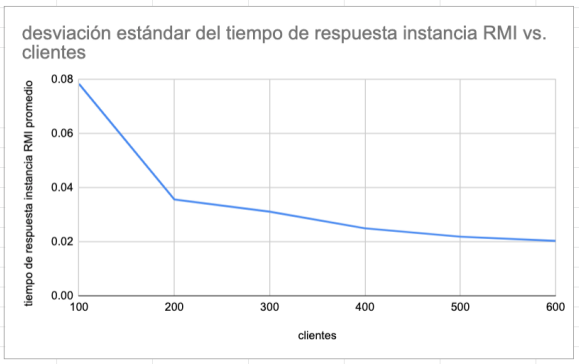
La desviación estándar de este tiempo igualmente sube a medida que aumenta la cantidad de clientes.

PC1 **PC2**

Un comportamiento inesperado es el tiempo de conexión del servicio RMI, cuyo promedio baja a medida que aumenta la cantidad de clientes.



PC1 **PC2**



PC1 **PC2**

Esto es debido a que el primer cliente RMI es el que más tarde en conectarse. Esto puede deberse a que RMI trabaja por TCP [2] Y todas las conexiones del tester se corren desde el mismo equipo, por lo que solo se necesita un canal TCP. En las

pruebas realizadas, la primera conexión tarda alrededor de 13 ms, mientras que las demás tardan 1 o 2 ms. De hecho, las gráficas de las desviación estándar apoyan el hecho de que solo se oscile entre estos dos números, pues la desviación está dentro del orden de las centésimas de segundo.

Finalmente, paramos las pruebas hasta 500-600 clientes ya que después de esto los clientes empezaron a tener errores en donde el servidor rechaza sus mensajes (connection reset by peer).

Dentro de ambos experimentos no se detectaron fallas en la recepción de mensajes ni en el registro de jugadores. Como en ambos tuvieron 100% de éxito, decidimos no incluir las gráficas.

Anexos y referencias

[1] Hoja de cálculo con datos de los experimentos https://docs.google.com/spreadsheets/d/13bNy9W54PkpZ7tka6-dFDheXYSpQ3ZquxPs7LscVFKI/edit?usp=sharing

[2] Oracle. (Sin fecha). Java Remote Method Invocation: 10 - RMI Wire Protocol. Consultado en: https://docs.oracle.com/javase/9/docs/specs/rmi/protocol.html