Laboratorio: Sistemas Distribuidos

Profesora: Erika Rosas Olivos - Juan Calderón Maureira
Ayudantes: Jorge Díaz M. jorge.diazma@sansano.usm.cl

Octubre 2021

1. Objetivos

- Aprender acerca de la comunicación en sistemas distribuidos.
- Familiarizarse con tecnologías de comunicación:
  - Remote Procedures en Go (gRPC).
  - Asynchronous Messaging (RabbitMQ).
- Implementar y coordinar diferentes componentes de un sistema distribuido.
- Familiarizarse con conceptos de replicación y consistencia de datos.

2. Introducción

Un sistema distribuido es un conjunto de computadores que trabajan de manera conjunta para cumplir algún objetivo. Para realizar esto, es esencial que estos se comuniquen entre sí mediante el intercambio de mensajes. La comunicación entre estos computadores puede ser desarrollada de distintas maneras, por ejemplo, puede existir una comunicación síncrona, en la cual se asume que los mensajes llegarán en un determinado lapso de tiempo, o bien una asíncrona, en la que no se toma este supuesto. Para poner esto en práctica, se propone un sistema en el cual deban hacer uso del intercambio de mensajes para cumplir las tareas propuestas.

Además de lo anterior, en la medida en la que los componentes de los sistemas distribuidos aumentan en complejidad se hace presente un nuevo problema, la coordinación. Servicios que comparten recursos o que se distribuyen tareas, además de poder comunicarse, deben ser capaces de ponerse de acuerdo para trabajar en conjunto.

Para realizar este laboratorio, se utilizarán tecnologías de comunicación como RPC y RabbitMQ. En la siguiente sección pueden encontrar documentación acerca de estas tecnologías, siguiendo los enlaces dispuestos.

3. Tecnologías

- El lenguaje de programación a utilizar es **Go**
- Para la comunicación síncrona se utilizará **gRPC**
- Para la comunicación asíncrona se utilizará **RabbitMQ**

4. Laboratorio

El laboratorio consiste implementar un sistema distribuido inspirado en la serie del momento Squid Game.

4.1. El Juego del Calamar

El juego del calamar consiste es una evento donde participan **16 jugadores**, donde el o los participantes que logren ganar las 3 etapas se llevarán, en conjunto, una gran suma de wones.

Los jugadores, deberán comunicarse de manera síncrona con un proceso/entidad llamado **El Líder**. El Líder será el encargado de recibir las peticiones del los jugadores como, por ejemplo, participar del juego del calamar o recibir sus jugadas en cada etapa del juego.

Junto con lo anterior, el Líder deberá informar de forma asíncrona a un proceso llamado **Pozo** cuando un jugador sea eliminado, agregando 100 millones de wones por cada uno. Además, el Líder debe informar a un proceso llamado **NameNode** sobre todas las jugadas que hagan todos los jugadores. Este NameNode deberá decidir como registrar las jugadas de todos los jugadores participantes comunicándose distribuyendo estos datos entre 3 procesos llamados **DataNode**. Estos Procesos DataNode deberán crear un archivo por cada jugador asignado en esa ronda y registrar todos sus acciones.

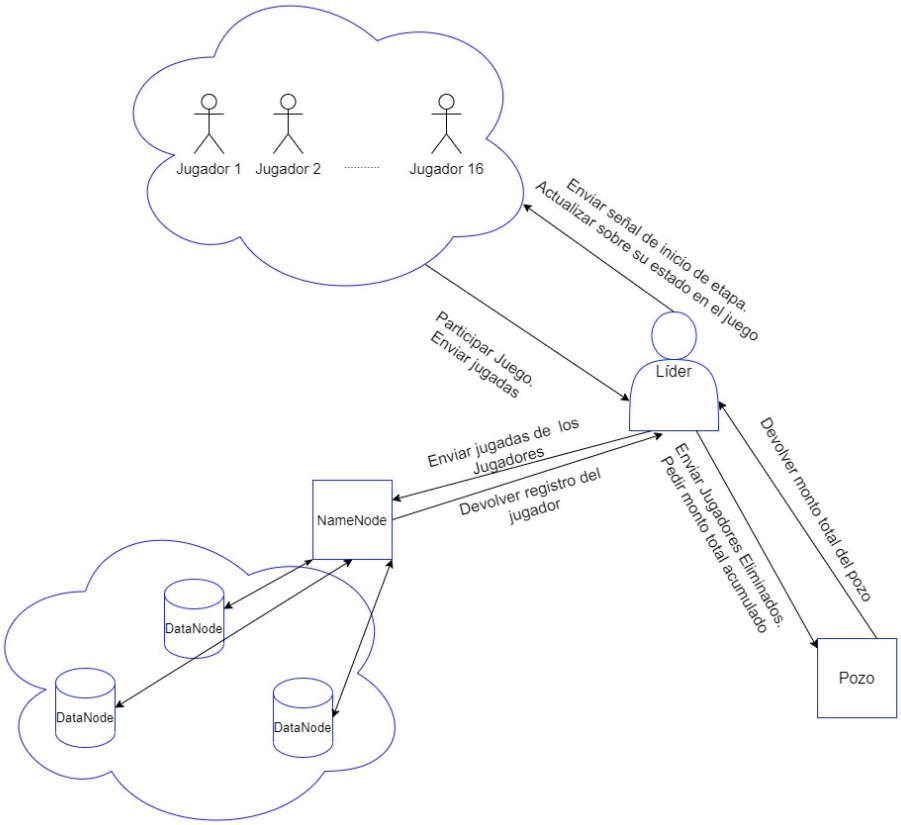

Figura 1: Relaciones del Sistema

4.2. Especificaciones por proceso

A continuación se especificará las funcionalidades de cada uno de los procesos.

4.2.1. Jugadores

Los jugadores serán los procesos que participarán del Juego del calamar. Harán de *clientes* del sistema. Un jugador debe ser una persona y deberá poder jugar mediante una interfaz por consola. El resto de los participantes pueden ser bot, en otras palabras, lo mismo que el jugador normal pero automatizado. Los jugadores deben ser capaces de:

- Enviar la petición para unirse al juego del calamar.
- Ser capaces de jugar cualquiera de las 3 etapas, en otras palabras, tener el código implementado para jugar esas rondas (cada etapa es un subjuego diferente que se explicará en las próximas secciones de este documento).
- Enviar sus jugadas en cada ronda de cada etapa.
- En caso de ser eliminado, debe finalizar el proceso.
- Pedir ver el monto acumulado en el Pozo.

Los procesos de los jugadores pueden estar corriendo en diferentes máquinas y deben ser capaces de conectar con el Líder. Toda la comunicación con el Líder debe realizarse mediante gRPC.

4.2.2. El Líder

Es el encargado de la logística del juego. Es quien coordina y orquesta todo el sistema. Debe esperar a que estén conectados todos los jugadores inicialmente para comenzar el juego. El Líder debe ser capaz de :

- Disponer de una interfaz por consola.
- Mostrar por consola cada vez que muere un jugador.
- Mostrar por consola los jugadores vivos al término de cada etapa.
- Mostrar por consola el o los ganadores del Juego del Calamar.
- Mediante la interfaz, dar comienzo a la siguiente etapa del Squid Game.
- Mediante la interfaz, de preguntar sobre todas las jugadas en todas las rondas de un determinado jugador.
- Debe analizar si un jugador es eliminado, informándole al jugador y al pozo.
- Debe enviarle todas las jugadas de todos los jugadores al NameNode.

Este proceso debe estar corriendo solamente en una de las máquinas virtuales. Debe comunicar de manera asíncrona, mediante RabbitMQ, al Pozo cuando un jugador es eliminado, pero debe poder preguntar de manera síncrona el monto acumulado en el Pozo usando gRPC. Debe comunicarse de manera síncrona con el NameNode utilizando gRPC.

4.2.3. Pozo

Es el encargado de mantener el conteo del monto acumulado del juego. Debe crear un archivo txt donde se registre cada uno de los jugadores eliminados y el monto acumulado actual.
El Pozo debe ser capaz de :

- Registrar cada uno de los jugadores eliminados en el archivo txt de la siguiente forma:
  - Jugador_numero Ronda_numero monto_acumulado_actual
  - ...

    - Jugador_1 Ronda_1 600000000

- Responder a las peticiones sobre el monto actual acumulado.

Este proceso debe estar corriendo solamente en una de las máquinas virtuales. Debe procesar de manera asíncrona, mediante RabbitMQ, el registro de los jugadores eliminados, pero de manera síncrona responder a la petición de monto acumulado.

4.2.4. NameNode

Es el encargado de distribuir el registro de las jugadas de los jugadores en cada ronda.
El NameNode debe ser capaz de :

- Debe distribuir, de manera aleatoria entre los 3 Datanode, la tarea de registrar las jugadas de los jugadores en cada ronda.
- Registrar en un archivo txt donde están almacenados las jugadas de cada jugador en cada ronda de la siguiente manera:
    - Jugador_numero Ronda_numero ip_datanode
    - ...
    - Jugador_1 Ronda_1 10.0.1.10

Este proceso debe estar corriendo solamente en una de las máquinas virtuales. Debe procesar de manera síncrona, el registro de las jugadas de los jugadores y la petición de ver las jugadas almacenadas de los jugadores.

4.2.5. DataNode

Son los procesos encargados de almacenar el registro de las jugadas de los jugadores
Los DataNode debe ser capaz de :

- Debe registrar, en archivos txt, las jugadas de cada jugador en cada ronda de la siguiente manera:
    - archivo : jugador_numero__ronda_numero.txt
    - ◦ jugada
    - archivo : jugador_1__ronda_1.txt
    - ◦ 5
      ◦ 6
      ◦ 5
      ◦ 5

Este proceso debe estar corriendo en tres máquinas diferentes, pero no en la máquina donde se este ejecutando el proceso NameNode. Debe procesar de manera síncrona, el registro de las jugadas de los jugadores y la petición de obtener el registro las jugadas de los jugadores.

4.3. Etapas

4.3.1. Etapa 1: Luz Roja, Luz Verde

El juego consiste en que cada jugador deberá elegir números entre el 1 y el 10 en diferentes rondas hasta sumar 21. Además de esto, en cada ronda el Líder deberá elegir al azar un número entre el 6 y el 10.

En cada ronda se debe verificar que jugadores eligieron un número igual o mayor al del Líder. Quienes lo hayan hecho quedan eliminados del juego.

El juego no puede durar más de 4 rondas. Todos los jugadores que no logren llegar a 21 antes de las 4 rondas serán eliminados.

4.3.2. Etapa 2: Tirar la Cuerda

Este juego consiste en que los jugadores participantes se dividirán en dos equipos de tamaño par. En caso de que el número de jugadores sea impar, se deberá escoger un jugador al azar y eliminarlo del juego.

En este juego, los jugadores de cada equipo deben elegir un número al azar entre 1 y 4 y sumar sus números. Junto con esto, el Líder también deberá elegir un número entre 1 y 4.

El equipo que logré que la suma de sus valores tenga la misma *paridad* que el número elegido por el Líder pasará a la siguiente ronda y el otro equipo será eliminado. Si los dos equipos obtienen la misma paridad que el Líder pasan a la siguiente ronda. Si ningún equipo obtiene la misma paridad que el Líder, este debe elegir un equipo al azar y eliminarlo, permitiendo pasar de etapa al equipo restante.

4.3.3. Etapa 3: Todo o Nada

Este juego consiste en que los jugadores participantes competirán en parejas. Sí el número de jugadores es impar (diferente a 1), se debe eliminar un jugador al azar. Los jugadores deben elegir un número entre 1 y 10, y el líder debe elegir un número al azar entre 1 y 10. El jugador cuyo valor absoluto de la resta entre el valor del Líder y de él sea el menor gana la partida y el otro es eliminado. Si ambos jugadores eligen el mismo número, ambos ganan.
Los jugadores que salgan ganadores de esta etapa son considerados los ganadores del Squid Game.

5. Comunicación y Máquinas Virtuales

Para la ejecución del laboratorio se utilizarán 4 máquinas virtuales. El proceso **Líder, NameNode y Pozo** deben estar ejecutándose en máquinas diferentes cada uno.

Para la comunicación entre los jugadores y el Líder, entre el **Líder y el Namenode**, y el **NameNode y los Datanode**, deben utilizar llamadas a procedimientos remotos, para esto deben emplear gRPC como la tecnología que se haga cargo de esta parte.

Por otro lado, para la comunicación entre el **Pozo y el Líder** sobre la notificación de un jugador eliminado, debe ser de manera asíncrona utilizando como tecnología base RabbitMQ. Sin embargo, las peticiones sobre la cantidad de dinero acumulado debe ser utilizando gRPC.

6. Restricciones

Todo uso de librerías externas que no se han mencionado en el enunciado debe ser consultado con el ayudante.

7. Consideraciones

- Si en algún momento de cualquier etapa el juego se queda con 1 participante, se detiene el juego y se considera el ganador del Juego del Calamar.
- Debido a la cantidad de ayudantes este semestre, las dudas serán **exclusivamente** canalizadas por Aula en foros creados para el laboratorio.
- Cabe mencionar que el próximo laboratorio será una continuación de este enunciado.

8. Reglas de Entrega

- La tarea se realiza en los grupos ya publicados.
- La fecha de entrega es el día 2 de noviembre a las 23:55.

- La tarea se revisará en las máquinas virtuales, por lo que los archivos necesarios para la correcta ejecución de esta debe estar en ellas. Recuerde que el código debe estar identado, comentado, sin warnings y sin errores. Se aplicará un descuento de 5 puntos al total de la nota por cada Warning, Error o Problema de Ejecución. Además de esto, debe subir un archivo comprimido `.zip` o `.tar.gz`, y debe indicar el número de su grupo siguiendo el patrón: L2-Grupo[N°Grupo].zip, Ejemplo: L2-Grupo01.zip
- Debe dejar un MAKEFILE o similar en cada máquina virtual asignada a su grupo para la ejecución de cada entidad.
- Debe dejar un README en cada máquina virtual asignada a su grupo con nombre y rol de cada integrante, además de la información necesaria para ejecutar los archivos.
- No se aceptan entregas que no puedan ser ejecutadas desde una consola de comandos. Incumplimiento de esta regla significa nota 0.
- Cada hora o fracción de atraso se penalizará con un descuento de 5 puntos.
- Copias serán evaluadas con nota 0 y serán notificadas a los profesores y las autoridades pertinentes.

9. Consultas

Para hacer sus consultas recomendamos hacerlas en el foro del ramo en Aula. Sin embargo, en caso de algo personal o urgente, el correo es: `jorge.diazma@sansano.usm.cl`. Cabe destacar que se responderán consultas vía Aula y/o correo electrónico hasta 24 hrs antes de la fecha y hora de entrega (en este caso, hasta el 1 de noviembre a las 23:55 hrs).