Laboratorio 1: Sistemas Distribuidos

Profesor: Jorge Díaz
Ayudantes de Lab: Iñaki Oyarzun M. & Javiera Cárdenas J.

Agosto 2023

1 Objetivos del laboratorio

- Aprender acerca de la comunicación en sistemas distribuidos.
- Familiarizarse y hacer uso de la comunicación síncrona y asíncrona por medio de **gRPC y RabbitMQ**
- Profundizar el uso de **Golang**

2 Introducción

Un sistema distribuido es un conjunto de computadores que trabajan de manera conjunta para cumplir algún objetivo. Para esto, es importante que estos se comuniquen entre sí mediante el intercambio de mensajes.
Este intercambio de mensajes puede llevarse a cabo de diferentes maneras. Por ejemplo, puede existir una comunicación síncrona, la cual se asume que los mensajes llegarán en un lapso de tiempo o bien asíncrona, en donde no se asume esto.

Para llevar esto a la práctica, se propone un problema en el cual debe implementarse una estructura de intercambio de mensajes síncronos y asíncronos para construir un sistema distribuido que logre resolver dicha situación. Esto por medio de **gRPC (síncrono)** y **RabbitMQ (asíncrono)**. En la siguiente sección, podrán hallar documentación sobre estas tecnologías en los enlaces propuestos.

3 Tecnologías

- El lenguaje de programación a utilizar es **Go**
- Para las comunicaciones se utilizarán **RabbitMQ y gRPC**

4 Sobre las tecnologías

A continuación dejamos a disposición algunos links de interés que serán de utilidad para la instalación y realización de este laboratorio (hacer click para acceder)

- Vídeo de explicación, instalación y uso de gRPC
- Quickstart de gRPC
- Playlist de tutoriales sobre RabbitMQ
- Documentación de RabbitMQ

5 Laboratorio

5.1 Contexto

Counter-Strike: Global Offensive (CS:GO) es un videojuego de disparos en primera persona (FPS) desarrollado por Valve y Hidden Path Entertainment. Es la cuarta entrega principal de la serie Counter-Strike y fue lanzado en 2012. El juego se centra en dos equipos, los terroristas y los antiterroristas, que compiten en rondas para lograr objetivos específicos según el modo de juego.

Los jugadores eligen un bando y se enfrentan en mapas diseñados estratégicamente, donde los terroristas pueden plantar una bomba en un sitio de bomba o retener a rehenes, mientras que los antiterroristas deben prevenir la explosión de la bomba o rescatar a los rehenes. El juego también incluye modos de juego como Deathmatch, Carrera de armas y otros.

CS:GO se ha convertido en uno de los juegos de disparos más populares y competitivos del mundo, con una escena competitiva activa y torneos de alto nivel. Los jugadores pueden adquirir y personalizar armas, mejorar sus habilidades y trabajar en su coordinación en equipo para tener éxito. Además, el juego cuenta con un sistema de economía en el que los jugadores deben administrar su dinero para comprar armas y equipo en rondas futuras.

El juego ha recibido actualizaciones regulares a lo largo de los años, con nuevas armas, mapas y ajustes de equilibrio para mantener la frescura y la emoción.

En una de sus últimas noticias, Valve a dado a conocer el lanzamiento de una actualización completa del sistema, dando a conocer cambios sustanciales a la fórmula de desarrollo del juego, llevando a cabo un cambio que comienza desde el motor gráfico y terminando en modificaciones a aspectos clave en el desarrollo de la parte estratégica de cada aspecto del videojuego. Teniendo granadas de humo mas realistas, mejoras en el comportamiento de las colisiones y los modelos 3D de cada objeto, entre otros. Siendo llamado "Counter Strike 2". Este cambio no ha dejado ajeno a ningún fan de la saga, saturando completamente el sistema de acceso a la beta del juego. Es por ello que, como especialistas en sistemas distribuidos, han sido llamados a prestar ayuda a la empresa de `GabeN` para desarrollar un sistema que pueda brindar por etapas una cantidad de llaves y proporcionar el acceso a los diferentes servidores regionales existentes por medio de conexiones síncronas y asíncronas entre servidores.

5.2 Explicación

Para el desarrollo de este laboratorio, se identificarán 2 entidades que serán parte del proceso completo:

- **Central de Valve:** Será la encargada de generar la cantidad de llaves de acceso a la beta de Counter, notificar a los servidores regionales y procesar la cantidad de usuarios por región que deseen acceder.

    1. Deberá tener un archivo txt llamado "**parametros_de_inicio**" que indicará el valor menor y mayor para generar una cantidad predefinida de llaves al azar en el formato **Valor1-Valor2**, es decir, si el archivo contiene la siguiente linea "**50-500**" la central deberá generar una cantidad al azar de llaves de entre 50 y 500, para luego notificar de manera **síncrona** a los servidores regionales. Dejando un registro de la hora y la cantidad de llaves generadas.
    2. Deberá tener **una cola asíncrona** para obtener las solicitudes entrantes por cada uno de los servidores regionales.

3. Una vez recibido un mensaje de registro por parte de uno de los servidores regionales se deberá obtener y registrar el servidor y la cantidad de usuarios que logran acceder a la beta, teniendo en cuenta la cantidad de accesos disponibles. Registrando cuantos usuarios solicitaron acceso, cuantos lograron ser registrados y cuantos no fueron registrados.
4. Una vez procesada la solicitud del servidor, la central deberá notificar al servidor regional cuantos usuarios no pudieron acceder a la beta (en caso de haber sido registrados todos los usuarios, deberá enviar el valor 0).
5. La central deberá tener un contador que permita indicar la cantidad de iteraciones de generación de llaves a realizar, siendo indicado por input del usuario mediante la segunda linea dentro del archivo mencionado en el paso 1 "**parametros_de_inicio**". Con el comodín siendo el -1. Es decir, si se ingresa una cantidad de 3, se deberán realizar 3 iteraciones de generación de llaves de acceso para luego finalizar el programa, y si se ingresa el valor -1, el programa deberá ejecutar sin límite el proceso de generación de llaves de acceso.

- **Servidores regionales:** Serán los encargados de administrar a los usuarios interesados en acceder a la beta para registrarlos de acuerdo con la cantidad de betas disponibles enviando la cantidad de usuarios que se van a registrar a la cola **asíncrona**, los servidores regionales son Asia, Europa, Oceanía, América.

  1. Deberá tener un archivo txt llamado "**parametros_de_inicio**" que almacenará la cantidad de usuarios interesados inicialmente en el acceso a la beta.
  2. Una vez recibida la información de la cantidad de betas disponibles por parte de la central de manera **síncrona**, el servidor regional deberá generar un valor entre la cantidad de usuarios interesados que contiene en su archivo "**parametros_de_inicio**" dividida en 2 menos el 20% de ese valor y la misma cantidad dividida en 2 más el 20% del valor. Es decir, si dentro del archivo "**parametros_de_inicio**" el servidor regional tiene una cantidad de 300 interesados, deberá generar un número al azar entre 120 y 180.
  3. El servidor notificará a la cola asíncrona del servidor central dicha cantidad, indicando qué servidor envió el valor.
  4. El servidor regional deberá esperar una notificación desde la central que le indique cuantos usuarios no pudieron ser registrados, para poder restarlos respecto al valor enviado en una siguiente iteración del proceso. Volviendo con esto al paso 1.

6 Restricciones

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

7 Consideraciones

- **Prints por pantalla:** Para ver el desarrollo y a la vez como apoyo para el debugging dentro del laboratorio, se solicitará que se realicen los siguientes prints por pantalla como mínimo:
  - **(Central):** Mostrar por pantalla la generación en la cual se encuentra el proceso junto el máximo de estas.
    (Ej: `Generación X/Y` o `Generación 1/3` o `Generación 3/infinito`)

  - **(Central):** Mostrar por pantalla cada solicitud tomada de la cola asíncrona.
    (Ej: `Mensaje asíncrono de servidor X leído`)
  - **(Central):** Mostrar por pantalla cuando se inscribe una determinada cantidad de cupos a la beta.
    (Ej: `Se inscribieron X cupos de servidor Y`)

  - **(Servidor Regional):** La cantidad de personar interesadas en inscribir la beta.
    (Ej: `Hay X personas interesadas en acceder a la beta]`)
  - **(Servidor Regional):** La cantidad de personas que lograron inscribirse.
    (Ej: `Ss inscribieron X personas`)
  - **(Servidor Regional):** La cantidad de personas que quedaron pendientes para obtener un cupo.
    (Ej: `Quedan X personas en espera de cupo`)

- Tanto la central, la cola Rabbit y servidores regionales conocen sus direcciones ip y puerto.
- Cada Servidor Regional deberá estar dentro de un container de docker de forma independiente, de igual manera, la central y la cola rabbit deberán estar en containers independientes.
- Se realizará una ayudantía para poder resolver dudas y explicar la tarea. Será notificado por aula.
- Consultas sobre la tarea se deben realizar en Aula o enviar un correo a **Javiera** o **Iñaki** con el asunto **Consulta grupo XX - Lab 1**
- Las librerías de **Golang** permitidas son:
  - time
  - strconv
  - strings
  - math
  - net
  - context
  - fmt
  - log
  - os
  - os/signal
  - sync
- **Sobre Presentación:** Cada grupo deberá presentar su código en el Laboratorio de Redes mostrando el desempeño de este, utilizando las maquinas virtuales de la Universidad que le serán asignadas a cada grupo.
- **Distribución de máquinas virtuales:** Las máquinas virtuales a utilizar, serán 4 en total, donde la distribución que se deberá cumplir es:
  - Cada servidor regional debe estar en una máquina independiente.
  - La central y la cola rabbit no pueden estar dentro de una misma máquina.
- **Ranking:** Tras realizada la presentación del Laboratorio se publicará un ranking presentando el desempeño del código realizado por cada equipo (Siendo indicado de manera presencial el procedimiento respectivo).

8 Informe

Tras realizada la presentación del Laboratorio deben realizar un informe que responda las siguientes preguntas a partir de su código, el desempeño de este y el ranking.

- Indique el tiempo que utilizo en la presentación del Laboratorio.
- Tras observar el ranking. ¿A qué atribuye su tiempo comparado con el resto?
- Si se aumentara el tamaño de la cola RabbitMQ. ¿Qué le ocurriría al proceso en general?
- Si se agregara una central que ejecute las mismas funciones que la actualmente implementada. ¿Esto beneficiaría o complicaría al proceso?
- En el caso de que exista una cantidad de solicitudes pendientes mayor a la cantidad de llaves que se han generado para la beta. ¿Qué ocurre?
- Luego de lo observado en la presentación. ¿Como podría optimizar su código para que este disminuya su tiempo?

9 Reglas de Entrega:

- El laboratorio se entrega en **grupos de 3 personas**. Que debe inscrito previamente a traves de una encuesta dada por aula.
- La fecha de entrega es el día **Miércoles 13 de Septiembre a las 23:59 hrs.**
- El laboratorio se presentara en el Laboratorio de Redes, utilizando las maquinas virtuales de la Universidad por lo que su entrega debe estar lo mas limpia posible, sin Warnings ni errores.
- Se aplicará un descuento de **5 puntos** al total de la nota por cada Warning, Error o Problema de Ejecución.
- Debe dejar un **MAKEFILE** o similar en cada entrega asignada a su grupo para la ejecución de cada entidad. Este debe manejarse de la siguiente forma:
  - `make docker-central`: Iniciará el código hecho en Docker para el servidor central.
  - `make docker-regional`: Iniciará el código hecho en Docker para los servidores regionales.
  - `make docker-rabbit`: Iniciará el código hecho en Docker para la cola Rabbit.
- Debe dejar un **README** en la entrega 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 **10 puntos**.
- Copias serán evaluadas con **nota 0** y serán notificadas a los profesores y autoridades pertinentes.

10 Consultas:

Para hacer las consultas, recomendamos hacerlas por medio del foro del ramo en Aula. De esta forma los demás grupos pueden beneficiarse en base a la pregunta. **Se responderán consultas hasta 48 hrs. antes de la fecha y hora de entrega.**