Laboratorio 1: Sistemas Distribuidos

Profesores: Jorge Díaz & Sebastián Godinez
Ayudantes de Lab: Iñaki Oyarzun M. & Sebastián Martínez C.

Agosto 2022

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)

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

5 Laboratorio

5.1 Contexto

Horzine, una corporación de origen Británica, líder en biotecnología y en la investigación sobre el mejoramiento de la humanidad, ha sido seleccionada para llevar a cabo el desarrollo de un proyecto secreto para el gobierno Británico de desarrollar un "Súper soldado".

La organización opta por dejar esta tarea en manos del reconocido Doctor Clamely. Quién decide llevar a cabo técnicas de clonación y modificación de ADN a partir de los restos de su hijo fallecido en diferentes ubicaciones secretas a lo largo del planeta, el doctor logra crear de esta forma el primer mutante (llamado Clot), el cual tiene unas leves mejoras de sus capacidades. La organización no conforme con estos avances, decide empujar mas allá el límite implantando armas en las extremidades del espécimen y mejorando aún más sus capacidades (llamado Flesh-Pound), según era de esperarse, debido al gran poder de estas abominaciones, la organización pierde el control provocando los primeros brotes de monstruos capaces de arrasar con todo.

Horzine en un intento de recuperar el control, contrata a un equipo de mercenarios, los cuales enviará desde su central a contener cada uno de estos estallidos. Es por ello que, como mano derecha del Doctor (desaparecido después del desastre), conociendo cada una de las ubicaciones de sus laboratorios, se les solicita implementar un sistema que permita recibir las señales de auxilio, para poder enviar y establecer contacto con los mercenarios en las ubicaciones. Dependerá de su equipo determinar la contención de este apocalipsis o sucumbir al fin del mundo...

5.2 Explicación

Para ayudar a la organización deberá comunicar 2 tipos de sistemas. La central (servidor) con los laboratorios (host):

- **Central:** Será la encargada de recibir las solicitudes de ayuda por parte de los laboratorios y de coordinar los equipos de mercenarios.
    1. Deberá tener **una cola asíncrona** para obtener las solicitudes entrantes por cada uno de los laboratorios.
    2. Una vez recibido un mensaje de auxilio por un laboratorio, la central deberá enviar a uno de los 2 equipos disponibles para contener el desastre (en forma de un mensaje síncrono).
    3. Una vez enviado el equipo de mercenarios, la central estará en contacto con el laboratorio de manera **síncrona** en todo momento, para saber su estado. Enviando mensajes **cada 5 segundos** al laboratorio.
    4. Durante los contactos realizados, la central deberá contabilizar las consultas hechas al laboratorio hasta que avise estar listo (situación contenida). Para luego registrarlas en un archivo txt (llamado **SOLICITUDES**) en el formato (NombreLab;CantidadDeConsultas)
    5. Una vez que los mercenarios notifiquen haber estado listos. Estos vuelven a la central estando nuevamente disponibles para manejar otra solicitud recibida.
    6. Para el cierre del programa, (ya sea por un comando o por medio del cierre control + C), la central deberá conectarse a los Laboratorios de manera sincrónica para indicar el término de la ejecución. (Es decir, Central notifica a los laboratorios con un mensaje para terminar de ejecutarse, los laboratorios confirman la información y se terminan sus programas, la central recibe las confirmaciones y se cierra, terminando la ejecución del sistema completo)

- **Laboratorios:** Estarán atentos ante los estallidos para notificar en seguida a la central y solicitar ayuda. Son 4 (Laboratorio Renca (la lleva) - Chile ; Laboratorio Pohang - Korea ; Laboratorio Kampala - Uganda ; Laboratorio Pripiat - Rusia)

  1. Cada 5 segundos, con una probabilidad de **0.8** se verá la ocurrencia de un estallido (0.8 ocurre estallido / 0.2 no ocurre estallido).
  2. En caso de un estallido, se genera y envía una solicitud a la central, estando a la espera de la llegada de la ayuda. (en ese tiempo solo se espera y no se envía otra solicitud).
  3. Una vez llegada la ayuda de la central, se pasa a un estado de contención, en dónde cada vez que se le consulte el estado de la contención, con una probabilidad de **0.6** se determina si ha sido resuelta o no (0.6 si fue resuelta / 0.4 no resuelta).
  4. En caso de ser resuelta la contención, se notifica el estado a la central y el equipo retorna, cerrando la conexión. En caso contrario se sigue conteniendo la situación.
  5. Una vez retorna el equipo, se vuelve al estado número 1. Repitiendo el ciclo.

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 cada solicitud tomada de la cola asíncrona.
    (Ej: `Mensaje asíncrono de laboratorio X leído`)
  - **(Central):** Mostrar por pantalla cuando se envía escuadrón.
    (Ej: `Se envía Escuadra X a Laboratorio Y`)
  - **(Central):** Mostrar por pantalla la consulta realizada al Laboratorio para saber el estado.
    (Ej: `Status Escuadra X: [Respuesta Laboratorio]`)
  - **(Central):** Cuando el escuadrón retorne a la central.
    (Ej: `Retorno a Central Escuadra X, Conexión Laboratorio Y Cerrada`)

  - **(Laboratorio):** Cuando se calcula la probabilidad de estallido.
    (Ej: `Analizando estado Laboratorio:  [ ESTALLIDO / OK ]`)
  - **(Laboratorio):** Cuando se envía el mensaje de emergencia y se espera la ayuda.
    (Ej: `SOS Enviado a Central.  Esperando respuesta...`)
  - **(Laboratorio):** Cuando llega el escuadrón.
    (Ej: `Llega Escuadrón X, conteniendo estallido...`)
  - **(Laboratorio):** Cuando se calcula resolución (o no) del estallido.
    (Ej: `Revisando estado Escuadrón:  [LISTO / NO LISTO]`)
  - **(Laboratorio):** En caso de resolución del estallido, notificar retorno de Escuadrón.
    (Ej: `Estallido contenido, Escuadrón X Retornando...`)

- Se le entregarán 4 máquinas virtuales a cada equipo para poder desarrollar este entregable.
- Tanto la central y los laboratorios conocen sus direcciones ip y puerto.
- Los laboratorios deben estar en máquinas diferentes (es decir, ninguna máquina debe tener 2 laboratorios)
- La central debe estar en una sola máquina junto a uno de los laboratorios.
- 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 **Sebastián** o **Iñaki** con el asunto **Consulta grupo XX - Lab 1**
- Las librerías de **Golang** permitidas son:
  - time
  - strconv
  - strings
  - math
  - net
  - context
  - fmt
- Un hint para la fase de testing:
  ```
  sudo firewall-cmd --zone=public --add-port=[PUERTO]/tcp --permanent
  sudo firewall-cmd --reload
  ```
- **Sobre Github:** Cada grupo tendrá acceso a un repositorio privado el cual le permitirá actualizar y entregar el código de su laboratorio. Para ello, el día de la entrega, deberá dejar en la rama principal (main) todos sus archivos finales.
- **Bono docker:** Existirá una bonificación de **10 ptos.** para aquellos grupos que puedan hacer uso de Docker para encapsular su código y ejecutarlo en cada una de las máquinas virtuales provistas (dejando evidencia de aquello igualmente en el repositorio) **OJO:** Solo se entregará dicho bono para aquellos que realicen la implementación para los laboratorios y la central.
- **Término de la ejecución:** Puede ser implementado a libre elección (indicando como se realizó en el readme), ya sea por medio de la detección de la combinación ctrl+C o por medio de un comando.

8 Reglas de Entrega:

- El laboratorio se entrega en **grupos de 2 personas**, existirá un grupo de 3 integrantes, el cual deberá ser notificado previamente a Sebastián (sebastian.martinezca@sansano.usm.cl). El primer grupo en notificar por correo será seleccionado. Dando aviso luego en aula para anunciar que ya fué tomado el cupo.
- La fecha de entrega es el día **Martes 13 de Septiembre a las 23:59 hrs.**
- La tarea se revisará en las máquinas virtuales, por lo que, los archivos necesarios para la correcta ejecución de esta, deben estar en ellas. Recuerde que el código debe estar identado, comentado, 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 máquina virtual asignada a su grupo para la ejecución de cada entidad. Este debe manejarse de la siguiente forma:
  - `make central`: Iniciará el código para la central
  - `make laboratorio`: Iniciará el código para el laboratorio
  - `make docker-central`: **(Opcional)** Iniciará el codigo hecho en Docker para la central.
  - `make docker-laboratorio`: **(Opcional)** Iniciará el codigo hecho en Docker para el laboratorio.
- 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 autoridades pertinentes.

9 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.**