PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2523 - Sistemas Distribuidos - 2/2020

Tarea 2

Entrega: 9 Noviembre 2020, 20:00 hrs.

Objetivo

El objetivo de esta tarea es utilizar el lenguaje `Go`/`Golang`, en un caso práctico de *sincronización de relojes*. En esta entrega deben implementar dos métodos de sincronización de relojes: lógicos y vectoriales en `Go`. En esta tarea, debe implementar su propia versión del código de relojes lógicos y vectoriales. No está permitido utilizar bibliotecas, librerías o módulos que implementen esta funcionalidad.

Golang

Los siguientes recursos son útiles para comenzar con `Go`.

1. Tutoriales básicos: Get started with Go: Hello, World tutorial
2. Aprender Go: A Tour of Go

Relojes lógicos y relojes vectoriales

Los relojes lógicos y los relojes vectoriales proveen un mecanismo para establecer un orden parcial de ocurrencia de eventos entre distintos procesos. Los relojes lógicos mantienen un contador local en cada proceso, el cual se va actualizando a medida que se reciben mensajes. Por otro lado, el reloj vectorial mantiene un vector del largo de la cantidad de procesos en el sistema, donde se refleja lo que cada proceso conoce de los eventos que han ocurrido en los demás procesos.
Usando relojes lógicos es posible implementar comunicación dentro de un grupo de procesos, usando *totally-ordered multicast*, Por otro lado, mediante los relojes vectoriales se puede implementar *multicast*, de manera que al recibir un mensaje, éste sólo sea procesado una vez que se hayan recibido todos aquellos que pueden tener precedencia causal con él. Esto se conoce como ***causal-ordered multicast***.Una explicación del funcionamiento de estos algoritmos de multicast se puede encontrar en la sección 6.2 del libro de Van Steen y Tanenbuam.

Tareas a resolver

En esta tarea debe construir un mecanismo de relojes lógicos y un mecanismo de relojes vectoriales que permitan implementar los algoritmos de ***multicast***. Debe entregar dos programas: uno para cada mecanismo.

El código debe:

- Leer un archivo de configuración con las referencias de cada proceso del sistema (sección Input)
- Leer un archivo de instrucciones (sección Input)
- Utilizar relojes lógicos/vectoriales según el caso a resolver.
- Ejecutar *multicast* entre los participantes

Input y ejecución

El *input* se entrega en un archivo de texto que será igual para todos los participantes. La ejecución será de la siguiente forma:

```
./program.go config_file.txt id instruction_file.txt
```

`config_file.txt` es un archivo de configuración con al menos tres líneas, donde cada línea contiene la IP del *host* y el puerto en que escuchará, de la siguiente manera:

```
Host Port
```

El siguiente ejemplo de archivo de configuración considera un grupo de 3 procesos ejecutando en la misma máquina.

```
127.0.0.1 2000
127.0.0.1 2005
127.0.0.1 3000
```

`id` es un entero que identifica al *host* y corresponde a un número entre 0 (inclusive) y la cantidad de líneas (no inclusive) del archivo de configuración. El proceso toma la configuración de la línea correspondiente a `id`. Cada proceso se iniciara con un `id` distinto. Por ejemplo, con el archivo anterior si `id` es `2`, entonces ese proceso usará el *host* local en el puerto 3000.
El archivo `instruction_file.txt` es un archivo de instrucciones que puede ser distinto para cada proceso, y donde cada línea detalla la interacción entre los participantes. Cada línea puede tener una de tres formas:

- C_i `M` C_j: indica que el proceso C_i envía un mensaje al proceso C_j
- C_i `M` C_j C_k C_l . . .: indica que el proceso Ci hará *multicast* enviando un mensaje a los procesos C_j, C_k, C_l, etc. Si hay N procesos en el archivo, habrá a los más $N - 1$ procesos receptores del *multicast*.
- C_i `A` T: indica que el proceso C_i adelantará su reloj lógico en un valor T. Esto quiere decir que, si estamos en el caso de un reloj lógico, donde C_1 tiene un reloj 2 y recibe una instrucciń C_1 A 4, su vector debe quedar con valor 6. Para el caso del reloj vectorial, si C_1 tiene un reloj vectorial $(0, 0, 0)$ y recibe una instrucción C_1 A 2, su vector debe quedar de la forma $(0, 2, 0)$.

Si un cliente se encuentra procesando el archivo de instrucciones y recibe un mensaje, debe bloquear la lectura del archivo hasta terminar de recibir el mensaje.

Output

Output Reloj Lógico

Cada proceso debe mostrar en pantalla los eventos que van ocurriendo. En cada línea un proceso debe indicar el valor actual de su reloj lógico, mostrar que va a enviar un mensaje y a qué destinatario. Cuando recibe un mensaje debe indicar el valor actual de su reloj lógico, de quién lo recibió, e indicar el nuevo valor de su reloj lógico.
Una vez terminada la ejecución, cada proceso debe mostrar el valor de su reloj lógico.

Output Reloj Vectorial

Cada proceso debe mostrar en pantalla los eventos que van ocurriendo. En cada línea, un proceso debe indicar el valor actual de su reloj vectorial, mostrar que va a enviar un mensaje y a qué destinatario. Cuando recibe un mensaje, debe indicar el valor actual de su reloj vectorial, de quién lo recibió, e indicar el nuevo valor de su reloj vectorial.
Una vez terminada la ejecución, cada proceso debe mostrar el valor de su reloj lógico, mediante una línea con valor por cada proceso. Por ejemplo:

```
1 2 0
```

Informe

Debe describir el funcionamiento e implementación de cada reloj y *multicast*, así como indicar las fuentes de referencia que hayan utilizado. Los aspectos formales del informe incluyen incluir el nombre y entregarlo en formato PDF.

Desglose de puntaje

Código (12 puntos)

- Funcionamiento de **reloj lógico**: 2 puntos.
- Funcionamiento de *totally-ordered multicast*: 4 puntos.
- Funcionamiento de **reloj vectorial**: 2 puntos.
- Funcionamiento de **causal-ordered multicast**: 4 puntos.

Si el código no compila, no se puede evaluar esta parte de la tarea.

Informe (7 puntos)

- Formalidades: 1 punto por el informe en PDF, incluir nombre y cumplir el formato de entrega.
- Descripción de **reloj lógico**: 1 punto.
- Descripción de *totally-ordered multicast*: 2 puntos.
- Descripción de **reloj vectorial**: 1 punto.
- Descripción de **causal-ordered multicast**: 2 puntos.

Formato de entrega

El informe debe ser entregado en formato PDF junto al código en un .zip o .rar en canvas. El nombre del archivo comprimido debe ser `NúmeroDeAlumno_IIC2523-2_Tarea_2`.

Integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1,1 en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.