TP3 - Blockchain Rústica

Integrantes

- Daneri, Alejandro
- Lafroce, Matias

Introducción

El presente trabajo práctico tiene como objetivo implementar una funcionalidad de *blockchain* simplificada.

Para la realización de este trabajo, optamos por realizarlo utilizando los siguientes algoritmos

- **Bully** (Algoritmo de elección de líder)
- **Algoritmo Centralizado** (Algoritmo de exclusión mutua)

A su vez se ha utilizado el protocolo TCP para la comunicación entre nodos de la red.

Descubrimiento de nodos activos en la red

Se ha implementado un algoritmo en el cual un nodo que entra a la red puede detectar los demas nodos que se encuentran actualmente en la red de la siguiente manera:

- En primer lugar, el nodo recorrerá todos los puertos disponibles que haya en la red y por cada puerto que se encuentre ocupado se creara un Peer el cual modelara al nodo remoto con el cual se esta conectando. Todas las interacciones posteriores con cualquiera de los nodos remotos se hará por medio de un PeerHandler que se detallara en profundidad más adelante.
- Luego realizar el barrido de puertos se quedara escuchando en un puerto que este disponible a la espera de mensajes de conexión nueva de nuevos nodos que se conecten. Cuando aparece un nuevo nodo se procederá a la creación de otro Peer de la misma manera que se menciono anteriormente

Algoritmo de elección de líder - Bully

Para la parte de la elección de un nodo *líder* o *coordinador*, decidimos implementarlo utilizando un algoritmo **Bully**, en donde la elección del coordinador se basa en un criterio simple de que el nodo con el numero de proceso mayor en la red es quien será el *coordinador*.

Dicho algoritmo se ejecuta cuando un nodo detecta que el coordinador actual "esta caído". Si un nodo, al enviarle algún mensaje al líder o coordinador, no recibe respuesta en un tiempo predeterminado, entonces considerará al mismo como desconectado y se procederá a la elección del líder nuevamente.

A su vez cada nodo nuevo que ingresa a la red recibe el lider actual cuando se conecta

Implementación

1. El nodo que comienza la elección envía, solamente a los Peers con numero de proceso mayor que el propio, un mensaje del tipo `LeaderMessage::LeaderElectionRequest` por la red, indicando el comienzo del proceso de elección de líder.
2. Cuando cada nodo reciba un mensaje de este tipo deberá responder al emisor con un mensaje `LeaderMessage::OkMessage`, y repitiendo el paso anterior con sus superiores.
3. Si un determinado nodo no obtiene respuesta alguna de los nodos a los que se contacto en el paso (1), entonces, sera el nuevo líder, por lo que deberá enviar un mensaje `LeaderMessage::CoordinatorMessage` hacia todos los nodos de la red indicando que es el nuevo líder.

Cuando un proceso de líder se encuentre en curso, no podrán seguirse procesando los eventos que necesiten la figura de líder en su procesamiento. Una vez la elección de líder haya finalizado, se podrá seguir con el flujo normal del programa.

Algoritmo de Exclusión Mutua - Algoritmo Centralizado

Implementación

El lock centralizado se realiza utilizando el nodo coordinador. Cuando un nodo quiere realizar una escritura en la blockchain, primero envía un mensaje de lock al nodo coordinador. Si el pedido fue exitoso, el coordinador le envía `lock_acquired` al nodo cliente. Luego este puede enviar el mensaje de escritura con la transacción a guardar. Si otro nodo intenta pedir el lock este será encolado durante `LOCK_TIMEOUT` segundos. Si se ejecutó un *release* (por el nodo que adquirió el lock previamente), se desencola el nodo que pidió el lock y se le envía el mensaje de `lock_acquired`. En contrapuesta, si el nodo encolado hizo "time out", se le devuelve al cliente que falló la adquisición del lock, teniendo que reintentar el pedido del lock. De esta forma evitamos un "busy wait" distribuido.

Debido a que el dispatcher de mensajes trabaja en un hilo propio, el modelo del lock también corre en un hilo separado del dispatcher, de forma que no bloquee mensajes entrantes de los nodos conectados.

Diagrama de implementacion

Modo de uso

Para iniciar el proceso y conectar un nuevo nodo a la red bastará con ejecutar

```
cargo run
```

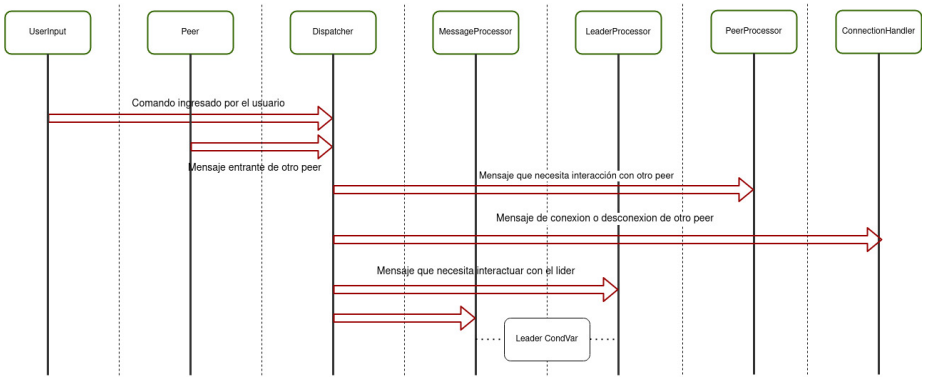

Figure 1: Threads

Una vez que se haya iniciado el proceso, el mismo podrá recibir los siguiente comandos:

```
wb insert <nombre> <nota> : Solicita agregar la nota de <nombre> con valor <nota> a la bloc
```

```
wb remove <nombre> : Solicita eliminar la nota de <nombre> de la blockchain.
```

```
rb : Solicita el estado de la blockchain actual.
```

Estructuras y Traits utilizados

Blockchain

TransactionData

Es una estructura que se encarga de modelar el dato que va a contener una transaccion de la blockchain. Esta compuesto por el nombre del estudiante y la nota.

Transaction

Esta estrucutura modela el tipo de transaccion que va a realizarse en la blockhain. Puede ser una insercion de una nota o una eliminacion

Block

En nuestra implementacion modelamos a cada bloque de la blockchain con una sola transaccion. A su vez el bloque contiene el hash del este bloque como el hash del bloque que lo precede en la cadena

Blockchain

Es la blockchain propia la cual esta compuesta de un vector de bloques

Client

Es el encargado de la creacion de los canales y handlers que van a interactuar entre si para la ejecucion del programa, los mismo van a ser desarrollados en profundidad mas adelante.

Centralized Lock

Modela el lock centralizado que va a tener en su poder el lider de la red

Peer

Modela un nodo remota que se conecta a la red

Client Event

Contiene la jerarquia de mensajes que son enviados entre los handlers para comunicarse entre si

Handlers

Connection Handler

Es el encargado de manejar los mensajes de conexiones entrantes y de aviso a los demas al conectarse a la red

Input Handler

Se encarga de manejar la comunicacion entre los mensajes que ingresan por consola y el cliente

Leader Handler

Se encarga de manejar los mensajes que son intercambiados en todo el proceso de eleccion de lider. TIene comunicacion con el Peer Handler para poder enviar los mensajes finalmente a los nodos remotos

Peer Handler

Es el encargo de mapear los mensajes que recibe por los canales de comunicacion hacia el nodo remoto via conexion TCP (o viceversa)

Conclusión

Al hacer este trabajo pudimos desarrollar ciertos conceptos importantes, como lo es el algoritmo de elección de líder y el algoritmo de exclusión mutua. Por otra parte, nos sentimos bastante a gusto de la manera que llevamos a cabo la

implementación del trabajo practico haciendo uso de canales, esta abstracción nos sirvió para facilitar el modelado de las entidades que entran en juego a lo largo del programa y como ellas interactúan entre si, sin estar teniendo en cuenta tanto problemas de concurrencia que seguramente nos traería si solamente hubiéramos usado herramientas como Mutexs