TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

SISTEMAS DISTRIBUTIVOS

COORDINACIÓN Y ELECCIONES EN ENTORNOS DISTRIBUIDOS

COORDINACIÓN Y ELECCIONES EN ENTORNOS DISTRIBUIDOS

En los sistemas distribuidos, múltiples procesos autónomos, ubicados en diferentes nodos, deben tomar decisiones conjuntas, cooperar para alcanzar objetivos comunes, y responder de manera coherente a los cambios del entorno. Esta necesidad da origen a uno de los retos fundamentales en este tipo de arquitecturas: la coordinación (Muñoz Escoí, 2013).

La coordinación en entornos distribuidos hace referencia al conjunto de mecanismos y estrategias que permiten que procesos independientes trabajen de manera organizada, eficiente y coherente, especialmente cuando deben acceder a recursos compartidos, sincronizar acciones o tomar decisiones globales (Pérez Rojas, 2006).

Debido a la ausencia de una memoria compartida, a la latencia de red y a la posibilidad de fallos, lograr dicha coordinación requiere de algoritmos distribuidos que operen correctamente incluso bajo condiciones de incertidumbre.

Coordinación distribuida: concepto y objetivos

El objetivo principal de la coordinación distribuida es permitir que grupos de procesos alcancen un estado común o ejecuten acciones sincronizadas, sin necesidad de una entidad central.

Entre las tareas más comunes de coordinación se encuentran:

- Distribuir cargas de trabajo.
- Sincronizar accesos a recursos.
- Elegir líderes para representar al grupo.
- Consensuar decisiones globales.
- Detectar y responder a fallos en nodos específicos.

Para lograr estas tareas, los sistemas utilizan estructuras de control como elecciones, quórums, relojes sincronizados y mensajes de consenso.

El problema de la elección en sistemas distribuidos

Una de las tareas fundamentales en coordinación distribuida es la elección de un coordinador o líder. Este líder es un proceso que, temporalmente, asume la responsabilidad de tomar decisiones críticas en nombre del grupo, como asignar recursos, validar operaciones o sincronizar procesos (Pérez Rojas, 2006).

La elección de un coordinador se convierte en una necesidad cuando:

- El coordinador anterior ha fallado.
- Se requiere una figura de autoridad para gestionar la comunicación.
- Se inicia un nuevo ciclo de operación o una fase de recuperación.

Un algoritmo de elección debe cumplir con:

- **Unicidad:** al final del proceso, debe haber exactamente un coordinador.
- **Finalización:** la elección debe completarse en un tiempo finito.
- **Validez:** el coordinador elegido debe ser uno de los procesos activos.
- **Equidad:** cada proceso debe tener una oportunidad razonable de ser elegido.

Principales algoritmos de elección

1. Algoritmo del anillo (Ring Algorithm)

Este algoritmo asume que los procesos están organizados lógicamente en un anillo circular y cada uno conoce a su sucesor.

Funcionamiento:

1. Cuando un proceso detecta la ausencia del coordinador, inicia una elección.
2. Envía un mensaje de elección a su sucesor con su ID.
3. Cada proceso agrega su propio ID al mensaje y lo pasa al siguiente.
4. Cuando el mensaje retorna al iniciador, este determina el ID más alto y lo proclama coordinador.
5. Se envía un mensaje de “coordinador elegido” por el anillo.

Ventajas:

- Garantiza unicidad.
- No depende del conocimiento completo de la red.

Desventajas:

- Ineficiente en anillos grandes.
- Retrasos acumulativos.

2. Algoritmo de Bully (Bully Algorithm)

Este algoritmo asume que cada proceso tiene un identificador único y que todos conocen a los demás.

Funcionamiento:

1. Un proceso detecta la caída del coordinador y lanza una elección.
2. Envía mensajes a todos los procesos con ID superior al suyo.
3. Si nadie responde, se autoproclama coordinador.
4. Si alguien responde, el proceso se retira y se espera el resultado.
5. El nuevo coordinador notifica a todos.

Ventajas:

- Rápido cuando el proceso con ID más alto está activo.
- Menor número de mensajes que el algoritmo del anillo.

Desventajas:

- Genera una “avalancha” de mensajes si muchos procesos inician elecciones simultáneamente.
- Requiere conocimiento completo de los demás procesos.

3. Algoritmos basados en quórum

Utilizan subconjuntos de nodos (quórums) para tomar decisiones de forma descentralizada. No hay un líder permanente, sino que las decisiones se toman cuando una mayoría responde (Pérez Rojas, 2006).

Aplicaciones típicas:

- Control de acceso a recursos distribuidos.
- Bases de datos replicadas.
- Consenso distribuido.

Importancia del consenso en la coordinación

En muchos escenarios, coordinar implica alcanzar consenso, es decir, que todos los procesos acuerden un valor común.

Esto es especialmente crítico en:

- Sistemas de archivos distribuidos.
- Redes blockchain.
- Protocolos de replicación.

Ejemplos de algoritmos de consenso:

- Paxos: garantiza seguridad bajo condiciones adversas, pero es complejo de implementar.
- Raft: más comprensible que Paxos, mantiene un líder y sincroniza registros replicados.
- Viewstamped Replication: usado en entornos donde se permite una visión temporalmente inconsistente mientras se asegura eventual coherencia.

Problemas y desafíos en la coordinación distribuida

- Fallos de procesos: la caída de nodos debe detectarse con rapidez y sin falsos positivos.
- División de red (network partitioning): puede generar múltiples coordinadores simultáneos.

- Rendimiento: elecciones y consensos generan sobrecarga de mensajes.
- Sincronización de estado: el nuevo coordinador debe tener una visión precisa y actualizada del sistema.

  - Ejemplo. Supóngase una red de cinco procesos distribuidos (P1 a P5), donde el coordinador (P5) falla.

- P2 detecta la ausencia y lanza una elección.
- Envía mensajes a P3, P4 y P5 (más altos).
- P4 responde, toma el control y notifica a todos.
- Se restablece el estado del sistema con P4 como nuevo líder.

  - Este ejemplo demuestra cómo un algoritmo como Bully mantiene la coordinación en presencia de fallos.