Informe Sistema Distribuido

Franco Hernández Piloto y Hivan Cañizares Diaz

January 2025

1 Sincronización de Relojes

Para abordar los problemas relacionados con la sincronización del tiempo en nuestro conjunto de servidores FTP, implementaremos el algoritmo de Berkeley. Este método es especialmente útil para sistemas distribuidos que requieren sincronización interna de relojes físicos.

1.1 Algoritmo de Berkeley

El algoritmo de Berkeley fue diseñado para entornos que carecen de fuentes de tiempo UTC. Es un enfoque centralizado que permite a los relojes del sistema mantenerse sincronizados entre sí.

1.1.1 Características principales

- Un servidor se elige como maestro entre todos los nodos del entorno.
- Los demás nodos actúan como esclavos.
- El maestro solicita periódicamente la hora a los esclavos.
- Se calcula la diferencia promedio entre todas las horas, incluyendo la del maestro.
- El maestro envía esta diferencia promedio a todos los esclavos.
- Todos los nodos, incluido el maestro, actualizan sus horas.

1.1.2 Funcionamiento detallado

1. El coordinador (maestro) solicita el tiempo de los esclavos, enviando su propio tiempo $T_{maestro}$.
2. Cada esclavo calcula y devuelve la diferencia entre el tiempo del maestro y su propio tiempo: $DT_n = T_{maestro} - T_{esclavo_n}$.
3. El coordinador recibe todas las diferencias de tiempo DT_n, incluyendo su propia diferencia (que es 0).

4. Se calcula el promedio de todas las diferencias de tiempo para obtener la Diferencia de Tiempo Promedio (DTP).
5. El coordinador envía la DTP a todos los esclavos.
6. Todos los nodos actualizan sus relojes según la fórmula:

$$RELOJ_{nuevo} = [DT_n + (DTP)] + RELOJ_{actual}$$

1.1.3 Consideraciones importantes

- Es crucial respetar las unidades de tiempo en todas las operaciones.
- El algoritmo busca un consenso entre los nodos, no necesariamente la sincronización con el tiempo real.
- Si un servidor tiene un reloj muy desajustado, su diferencia se desestima para evitar propagar el error.
- Cualquier nodo puede ser elegido como maestro, permitiendo la recuperación en caso de fallo del maestro actual.

1.1.4 Inconvenientes

- Si el maestro falla, la sincronización se pierde hasta que se elija un nuevo maestro.
- Los retrasos en la transmisión de mensajes pueden introducir errores en la sincronización.
- La naturaleza centralizada del algoritmo requiere un mecanismo para elegir un nuevo maestro en caso de fallos, preferiblemente uno que cometa la menor cantidad de errores.

1.1.5 Implementación en nuestro sistema FTP

En nuestro proyecto de servidores FTP, la implementación del algoritmo de Berkeley nos permitirá mantener una consistencia temporal entre todos los servidores. Y el maestro irá rotando de servidor en servidor inmunizando el caso de si el servidor maestro es eliminado.Seria como una lista circular donde pueden entrar o irse nodos y el turno de cada nodo de ser maestro ira cambiando consecutivamente.

1.2 Consideraciones importantes al recibir mas de una peticion practicamente al mismo tiempo

Cada vez que se reciba una peticion, se tendra un margen de 0,5s o 1s para guardar todas las peticiones que lleguen en ese momento que significara que llegaron "al mimso tiempo", luego se ordenarán de forma que segun la fecha del sistema ,gracias al algoritmo de tiempo ,hayan llegado.

2 Estrategia de Distribución y Replicación de Información

Para garantizar la disponibilidad y la resistencia a fallos en nuestro sistema distribuido de servidores FTP, implementaremos una estrategia de replicación inspirada en la topología de anillo de Chord, pero con modificaciones significativas para aumentar la redundancia y la tolerancia a fallos.

2.1 Estructura Básica

- Los servidores se organizan en una topología de anillo lógico.
- Cada pieza de información se replica en tres nodos distintos.

2.2 Estrategia de Replicación

1. **Replicación en Vecinos**: Cada pieza de información se almacena ,segun el mapeo del hash,en el nodo a su isquierda y el nodo a su derecha.
2. **Replicación Adicional**: Además, se guarda una tercera copia en el vecino del nodo izquierdo (considerando una representación ,de mapeo en hash, en sentido horario del anillo).

2.3 Ventajas del Sistema

- **Tolerancia a Fallos Mejorada**: El sistema puede soportar la caída simultánea de hasta dos nodos sin pérdida de información.
- **Redundancia Óptima**: Cada pieza de información tiene exactamente tres copias, equilibrando la redundancia y el uso eficiente del almacenamiento.
- **Recuperación Rápida**: En caso de fallo de un nodo, la información sigue siendo accesible a través de las copias en los nodos vecinos.

2.4 Funcionamiento

Consideremos un anillo con nodos A, B, C, D, E, F en sentido horario:

- La información del nodo A se almacena en A, B, y F.(imagina que la informacion esta en la arista entre A y B)
- La información del nodo B se almacena en B, C, y A.(imagina que la informacion esta en la arista entre B y C)
- Y así sucesivamente para cada nodo.

2.5 Escenarios de Fallo

- **Fallo de Un Nodo**: Si falla un nodo, sus dos vecinos inmediatos aún tienen copias de la información.
- **Fallo de Dos Nodos Adyacentes**: Incluso si fallan dos nodos adyacentes, la tercera copia en el vecino del vecino izquierdo asegura que no se pierda información.

2.6 Limitaciones

- El sistema no puede garantizar la integridad de los datos si fallan tres nodos adyacentes simultáneamente.
- Se requiere un mayor uso de almacenamiento debido a la triple replicación.