Universidad Nacional De Asunción

Facultad Politécnica

Ingeniería en Informática

Sistemas Distribuidos

Tarea - Unidad 5

Profesores:

- Fernando Mancía.
- Moyses Molinas.

Materia: Sistemas Distribuidos.

Alumnos:

- Marcos Raúl Flores Duarte.
- Fabrizio Amadeo Rejala Galeano.

Sección: TQ.

2024

Ejercicio 1: Existen dos enfoques principales para la sincronización de relojes en sistemas distribuidos: sincronización interna y sincronización externa.

1- Explica cómo funcionan ambos tipos de sincronización.

S**incronización externa** ocurre cuando los sistemas distribuidos ajustan sus relojes usando una fuente de tiempo externa, como servidores que obtienen la hora de relojes atómicos o GPS. Conectándose a servidores de UTC

Sincronización interna: los sistemas no dependen de una fuente externa. Un método clásico es el algoritmo de Berkeley, donde un nodo en la red actúa como coordinador, recoge las horas de los demás nodos, calcula un promedio y ajusta los relojes para que queden sincronizados entre ellos. Es útil en redes locales donde no se tiene acceso a una fuente externa de tiempo.

2. Comparación de sincronización externa vs. sincronización interna

Característica	Sincronización Externa (Algoritmo de Cristian)	Sincronización Interna (Algoritmo de Berkeley)
Precisión	Alta, típicamente en el rango de milisegundos al depender de GPS	Menor precisión ya que se basa en el promedio de los tiempos locales de los nodos
Tolerancia a fallos	Si el servidor de tiempo externo falla, la red puede desincronizarse. A pesar de esto, Cristian propone la utilización de múltiples servidores para mitigar esto.	Más tolerante a fallos en la red, el fallo de un nodo no afecta a todo el sistema ya que los restantes pueden seguir sincronizados.
Dependencia de componentes externos	Depende de servidores externos para obtener la referencia de tiempo.	No depende de externos, los relojes se sincronizan localmente
Escalabilidad	Altamente escalable. Está diseñado para funcionar en redes distribuidas globalmente.	Menos escalable. Si aumentan más nodos, hay más mensajes para recoger la hora de los nodos, lo que aumenta el tráfico de la red y en redes grandes la latencia puede ser bastante alta para lograr la sincronización

3. ¿En qué tipo de aplicaciones o escenarios sería más conveniente utilizar sincronización externa?

La sincronización externa es más conveniente en sistemas distribuidos que requieren alta precisión y tienen acceso a una infraestructura confiable para obtener tiempo exacto. Ejemplos incluyen:

- Sistemas financieros: Donde el registro preciso de transacciones es crítico para evitar errores o disputas.
- Redes globales: Como en la Internet de las Cosas (IoT), donde los dispositivos en distintas ubicaciones geográficas necesitan estar sincronizados con alta precisión.
- Sistemas de telecomunicaciones Sincronización Interna:y satélites: Que dependen de una sincronización exacta para coordinar comunicaciones entre nodos y usuarios.

4. ¿Cuándo preferirías usar sincronización interna?

La sincronización interna es más útil en redes locales o sistemas cerrados donde no se requiere una precisión extremadamente alta ni se tiene acceso a una fuente externa de tiempo. Es preferible en los siguientes escenarios:

- Sistemas de redes locales (LAN): Donde los nodos están cerca unos de otros y no es necesario depender de un servidor externo, como en los centros de datos.
- Sistemas tolerantes a fallos: En sistemas donde es importante evitar la dependencia de componentes externos, la sincronización interna es preferible.

5- Analiza el impacto de la sincronización en sistemas críticos (por ejemplo, transacciones bancarias, control de tráfico aéreo, redes de sensores)

La sincronización en sistemas críticos es fundamental para asegurar la consistencia de datos y la correcta coordinación entre procesos, especialmente en áreas como transacciones bancarias, control de tráfico aéreo y redes de sensores.

En **transacciones bancarias**, la sincronización evita problemas como el doble gasto o la inconsistencia en los saldos, garantizando que las operaciones se completen de manera segura.

En el **control de tráfico aéreo**, es esencial para la seguridad, ya que permite coordinar en tiempo real la posición de los aviones. Una falla en la sincronización podría generar graves riesgos de colisión.

En las **redes de sensores**, la sincronización garantiza la correcta recolección de datos, evita duplicaciones, y optimiza el uso de energía, especialmente al sincronizar los relojes de los nodos.

Ejercicio 2: El método de Cristian y el algoritmo de Berkeley son dos técnicas populares para la sincronización de relojes en sistemas distribuidos.

1- Explica el método de Cristian y el algoritmo de Berkeley.

Método de Cristian: Este método se basa en un servidor de tiempo central confiable. Los clientes envían una solicitud al servidor, que responde con su tiempo actual. El cliente ajusta su reloj local considerando el tiempo de ida y vuelta del mensaje. Es simple, pero depende de la precisión del servidor y de las fluctuaciones de red.

Algoritmo de Berkeley: No depende de una fuente de tiempo externa. En este caso, un nodo (denominado maestro) consulta a otros nodos (esclavos) sobre sus tiempos actuales, calcula un promedio, y ajusta todos los relojes (incluyendo el suyo) para que coincidan. Es útil en sistemas donde no se puede confiar en un único servidor de tiempo.

2- Compara ambos métodos en cuanto a:

- **Precisión en la sincronización**:
  - El **método de Cristian** es más preciso en redes de baja latencia y si el servidor de tiempo es muy exacto.
  - El **algoritmo de Berkeley** tiene una precisión menor, ya que depende del promedio de los relojes locales, lo que puede resultar en desajustes si hay mucha dispersión.
- **Dependencia de una fuente externa de tiempo**:
  - **Cristian** requiere un servidor de tiempo externo.
  - **Berkeley** no depende de una fuente externa, lo que le da más autonomía.
- **Robustez ante fallos**:
  - En **Cristian**, si el servidor de tiempo falla, el sistema no puede sincronizarse.
  - En **Berkeley**, el fallo de un nodo se maneja mejor, ya que el maestro puede seguir consultando a los demás nodos.
- **Desempeño en redes con alta latencia**:
  - **Cristian** es más sensible a la latencia de red, ya que depende de medir el tiempo de ida y vuelta de los mensajes.
  - **Berkeley** maneja mejor la latencia, ya que el ajuste es basado en el promedio de tiempos, aunque puede perder precisión si la red es muy inestable.

3- Discute las ventajas y desventajas de utilizar el método de Cristian en aplicaciones de sistemas distribuidos que dependen de sincronización precisa, como bases de datos distribuidas o servicios financieros.

- **Ventajas**:
    - Simple y eficiente si se cuenta con un servidor de tiempo preciso.
    - Ideal para redes con baja latencia y acceso confiable a una fuente de tiempo externa.
- **Desventajas**:
    - Vulnerable a fallos del servidor de tiempo.
    - Puede ser impreciso en redes con alta latencia o fluctuaciones, ya que el tiempo de ida y vuelta del mensaje puede variar.
    - No es la mejor opción si no se puede confiar en una única fuente de tiempo.

En aplicaciones críticas como **bases de datos distribuidas** o **servicios financieros**, la precisión es crucial. Si el servidor de tiempo no es confiable o la red tiene alta latencia, podrían surgir inconsistencias en los datos, errores en las transacciones o problemas de coherencia.

4- ¿Cuándo sería preferible utilizar el algoritmo de Berkeley en lugar del método de Cristian? Da 3 ejemplos concretos de sistemas donde uno de estos métodos sea más apropiado que el otro.

El **algoritmo de Berkeley** es preferible en sistemas donde no se dispone de una fuente de tiempo externa confiable, o cuando es necesario mantener un nivel de tolerancia a fallos entre los nodos.

Ejemplos:

1. **Redes de sensores distribuidos**: No siempre se puede contar con una conexión a una fuente externa de tiempo, y los relojes de los sensores pueden ajustarse mediante el promedio de sus tiempos locales.
2. **Sistemas industriales en fábricas**: En áreas donde la red externa es débil o inconsistente, el uso del algoritmo de Berkeley permite mantener una sincronización interna sin depender de servidores de tiempo externos.
3. **Aplicaciones militares en entornos remotos**: En zonas sin acceso a internet o con poca conectividad, los dispositivos pueden sincronizarse utilizando el promedio de sus relojes locales sin necesidad de una fuente externa de tiempo.

Ejercicio 3: El protocolo NTP (Network Time Protocol) se utiliza para sincronizar los relojes de los dispositivos en una red distribuida y organiza las fuentes de tiempo en una jerarquía de estratos.

1- Define la jerarquía de estratos en NTP, explicando la función de cada uno de los siguientes niveles:

• Estrato 0: No son dispositivos conectados directamente a la red. Son dispositivos de referencia de tiempo, como relojes atómicos, GPS o radios de referencia. Proveen la hora más precisa.

• Estrato 1: Son servidores de tiempo conectados directamente a dispositivos de estrato 0. Estos servidores reciben el tiempo de referencia y lo distribuyen a dispositivos en estratos inferiores. Son los servidores de tiempo más precisos disponibles en una red.

• Estrato 2: Son servidores que reciben el tiempo de servidores de estrato 1. Tienen menos precisión que los servidores de estrato 1 debido a la pequeña latencia introducida en la sincronización.

• Estrato 3 (y superiores): Son servidores que reciben el tiempo de servidores de estratos superiores. A medida que el número de estratos aumenta, la precisión disminuye ligeramente debido a la acumulación de errores en la sincronización y el retardo de red.

2- ¿Qué diferencias existen entre los dispositivos de estrato 0 y los de estrato 1?

Estrato 0: No están conectados directamente a la red, sino que son dispositivos de referencia (relojes atómicos, GPS) que proporcionan una fuente de tiempo extremadamente precisa, están conectados a los dispositivos de estrato 1.

Estrato 1: Son servidores de red que reciben la hora de los dispositivos de estrato 0 y distribuyen esta información a otros dispositivos de la red. Los servidores de estrato 1 actúan como intermediarios entre las fuentes de tiempo de referencia (estrato 0) y los demás nodos de la red.

3- ¿Cómo influye el número de estratos en la precisión y confiabilidad de la sincronización de tiempo?

A medida que aumenta el número de estratos, la precisión de la sincronización tiende a disminuir. Esto se debe a:

- **Retraso acumulado**: Cada estrato introduce un pequeño retraso debido a la latencia de red y al procesamiento de los paquetes de sincronización.
- **Errores acumulativos**: Cada servidor en estratos inferiores sincroniza su reloj basándose en el reloj de un servidor de nivel superior, lo que puede introducir pequeños errores en cada paso.

4- ¿Cómo se maneja la tolerancia a fallos en los diferentes estratos? Analiza cómo NTP maneja la elección de servidores de tiempo alternativos en caso de fallos o desconexiones.

Si un servidor de tiempo falla o se desconecta, los clientes NTP pueden seleccionar servidores alternativos. El protocolo permite a los clientes de red sincronizarse con múltiples servidores de tiempo, creando una **redundancia** que mejora la confiabilidad.

En caso de fallo:

- Los clientes NTP seleccionan otro servidor de tiempo disponible que esté conectado y tenga una buena calidad de sincronización.
- Los servidores alternativos son evaluados según varios criterios, como la latencia de red, la estabilidad del reloj y la proximidad jerárquica.

Ejercicio 4: Investiga el funcionamiento de NTP en una aplicación real, como Google Spanner o sistemas financieros distribuidos y responde las siguientes preguntas:

1- ¿Cómo gestionan estos sistemas la sincronización de tiempo usando servidores NTP?

En aplicaciones como **Google Spanner** o **sistemas financieros distribuidos**, la sincronización de tiempo precisa es fundamental para mantener la coherencia de datos y el orden correcto de las operaciones. Estos sistemas utilizan **NTP** para sincronizar los relojes de los nodos distribuidos en diferentes ubicaciones geográficas. Para evitar problemas de latencia y minimizar los desfases, emplean múltiples servidores NTP distribuidos geográficamente, lo que les permite ajustar sus relojes de forma precisa y confiable.

En Google Spanner, por ejemplo, se utiliza un mecanismo adicional llamado **TrueTime** que se apoya en NTP para garantizar que las transacciones distribuidas ocurran en un orden coherente a nivel global.

2- ¿Qué estrato utilizan y por qué?

Estos sistemas suelen usar **servidores NTP de estrato 1** o **estrato 2**, porque buscan un equilibrio entre precisión y disponibilidad. Los servidores de **estrato 1** son preferidos cuando es posible, ya que ofrecen la máxima precisión al estar directamente conectados a relojes de referencia (estrato 0), como relojes GPS o relojes atómicos. Sin embargo, en algunos casos, se utilizan servidores de **estrato 2** cuando los de estrato 1 no son suficientes o están muy alejados geográficamente, dado que aún proporcionan una sincronización precisa con un retraso mínimo.

3- Analiza por qué la precisión temporal es crucial en este tipo de sistemas y qué problemas podrían ocurrir si hay desfases en la sincronización

En aplicaciones como **Google Spanner** o en **sistemas financieros distribuidos**, la precisión temporal es crucial por varias razones:

- **Coherencia de datos**: En sistemas distribuidos, los eventos deben ser procesados en un orden coherente. Si el tiempo no está bien sincronizado entre los nodos, una transacción en una parte del sistema podría parecer que ocurrió antes de una operación relacionada en otro nodo, causando errores de coherencia.
- **Timestamps**: Muchos sistemas usan marcas de tiempo (timestamps) para ordenar las operaciones. Si los relojes están desfasados, las transacciones podrían procesarse en el orden incorrecto, causando inconsistencias o errores en la base de datos.

- **Seguridad**: En sistemas financieros, los desfases de tiempo pueden ser explotados para manipular el orden de las transacciones, lo que podría tener graves implicaciones de seguridad.

Problemas por desfases:

- **Inconsistencias** en los datos almacenados, lo que puede derivar en decisiones de negocio incorrectas.
- **Errores en el procesamiento de transacciones**, que en sistemas financieros puede generar graves pérdidas o fraudes.
- **Fallos en la replicación de datos**, causando diferencias entre los nodos de un sistema distribuido.