Universidad Nacional Autónoma de México
Facultad de Ciencias
Práctica 4 ReadMe
Cervantes Duarte Jose Fernando 422100827
Morales Chaparro Gael Antonio 320076972
Rivera Lara Sandra Valeria 320039823
7 de noviembre de 2024

1 Descripción de Broadcast

Para la implementación del reloj de Lamport usamos como base el algoritmo de Broadcast, y principalmente nos basamos en el siguiente algoritmo:

Algorithm 1 Broadcast con relojes

if $\text{id}_{\text{nodo}} = 0$ **then**
    mensaje $\leftarrow$ datos
    esperar(1 - 5)
    for k en vecinos **do**
        reloj $\leftarrow$ reloj + 1
        eventos.agregar([reloj, 'E'(evento), datos, id_{nodo}, k])
        enviar(datos, reloj, id_{nodo})
    end for
end if
while verdadero **do**
    esperar(1 - 5)
    (datos, reloj_remoto, j) $\leftarrow$ canal_entrada
    mensaje $\leftarrow$ datos
    reloj $\leftarrow$ max(reloj, reloj_remoto) + 1
    eventos.agregar([reloj, 'R'(evento_recibido), datos, j, id_{nodo}])
    esperar(1 - 5)
    for k en vecinos **do**
        if $k \neq j$ **then**
            reloj $\leftarrow$ reloj + 1
            eventos.agregar([reloj, 'E', datos, id_{nodo}, k])
            enviar(datos, reloj, id_{nodo})
        end if
    end for
end while

El método `broadcast` implementa el algoritmo de broadcast utilizando el reloj de Lamport, y consiste en los siguientes pasos:

1.1 Nodo Distinguido (Inicio del Broadcast)

1. Si el nodo es distinguido (con `id_nodo = 0`), inicia el proceso de difusión asignando el mensaje inicial `data` a su variable de instancia `mensaje`.

2. Antes de enviar el mensaje, espera un tiempo aleatorio utilizando la función `env.timeout(randint(1, 5))`, que simula el costo del procesamiento o latencia en un sistema distribuido.

3. El nodo distinguido incrementa su `reloj` de Lamport en +1 antes de enviar el mensaje a cada vecino, asegurando que el evento de envío tenga una marca temporal actualizada.

4. Cada evento de envío se registra en la lista `eventos` con la siguiente estructura:

$$[\text{reloj}, \text{'E'}, \text{data}, \text{id\_nodo}, \text{vecino}]$$

donde:

- `reloj` es la marca temporal actual,
- 'E' indica un evento de envío,
- `data` es el contenido del mensaje,
- `id_nodo` es el nodo emisor, y
- `vecino` es el nodo receptor.

1.2 Recepción y Retransmisión de Mensajes

1. Cada nodo entra en un bucle infinito donde espera la llegada de un mensaje en su `canal_entrada`.
2. Al recibir un mensaje, el nodo extrae el contenido del mensaje, el reloj remoto y el identificador del nodo emisor (`j`).
3. El reloj de Lamport del nodo se actualiza utilizando la fórmula:

$$\text{reloj} = \max(\text{reloj\_local}, \text{reloj\_remoto}) + 1$$

Esto asegura que la marca temporal del evento de recepción es consistente en el sistema distribuido.

4. El evento de recepción se registra en la lista `eventos` con la estructura:

$$[\text{reloj}, \text{'R'}, \text{data}, \text{j}, \text{id\_nodo}]$$

donde:

- `reloj` es la marca temporal del evento,
- 'R' indica un evento de recepción,
- `data` es el contenido del mensaje,
- `j` es el nodo emisor del mensaje, y
- `id_nodo` es el nodo receptor.

5. El nodo espera un tiempo aleatorio antes de retransmitir el mensaje a todos sus vecinos excepto al nodo que le envió el mensaje, excluyendo a `j` de la lista de vecinos.
6. Para cada vecino restante, incrementa su reloj de Lamport y registra el evento de retransmisión en la lista de eventos con la misma estructura que el evento de envío inicial.
7. Finalmente, el mensaje es enviado a los vecinos filtrados utilizando el método `canal_salida.envia`.

2 Actualización del Reloj de Lamport

Cada evento (envío o recepción) incrementa el reloj de Lamport, manteniendo la consistencia temporal en el sistema distribuido. La estructura de eventos en `eventos` facilita la auditoría del orden de eventos en la red y permite analizar la causalidad entre eventos.