Sistemas Distribuidos

Resumen

Dorian Wozniak

Índice

Capítulo 1

COORDINACIÓN BÁSICA

Un objetivo esencial de los sistemas concurrentes es coordinar la ejecución de procesos interdependiente que solo interactuan mediante paso de mensajes. Para ello ese requiere gestionar el tiempo mediante la correcta ordenación del tiempo, tal que cumplan con el **principio de causalidad**.

La generación de eventos cumple dos propiedades básicas:

- **Vivacidad** (liveness): La ejecución de procesos no se bloquea indefinidamente.
- **Corrección** (safety): La ejecución de procesos es correcta según la especificación.

1.1 Modelos temporales

Hay tres modelos temporales básicos de algoritmos distribuidos:

- **Modelo asíncrono**
  - Indeterminismo temporal: No hay límite de tiempo para entrega de mensajes y ejecución de acciones.
  - Los fallos influyen notablemente en el indeterminismo.
  - Se puede gestionar el paso del tiempo mediante entregas de mensajes, sin relojes físicos.
  - No todos los problemas se pueden resolver de forma asíncrona.
- **Modelo síncrono**
  - Determinismo en límites de comportamiento: El tiempo de retraso de mensajes y ejecución de acciones está acotado.
  - Los relojes físicos de cada nodo estan sincronizados dentro de un límite máximo de sincronización.
  - Es dificil diseñar sistemas distribuidos que cumplan con los límites de tiempo con alta probabilidad.
- **Modelo parcialmente síncrono**: Sistema generalmente asíncrono que circunstancialmente se vuelve síncrono.

Los algoritmos asíncronos son válidos para sistemas síncronos, pero no viceversa.

1.2 Relojes

Los **relojes** son la herramienta base de coordinación y sincronización.

Hay dos tipos de relojes básicos. Normalmente se utiliza una mezcla de ambos:

1.2.1 Relojes físicos

Los relojes físicos son utilizados para modelos síncronos y parcialmente síncronos.

- Avanzan con frecuencia diferente entre ellos. Se trata de mantenerlos dentro de un límite respecto a un reloj global.
- Dos modelos de sincronización de relojes físicos:
  - Interno (entre relojes del sistema)
  - Externo (con un reloj fuera del sistema, ej. UTC)
- La sincronización se modeliza:
  - **r Deriva** (*drift*): Diferencia de frecuencia de dos relojes
  - **d Desviación** (*skew*): Máxima deriva permitida

- **R**: **Intervalo de resincronización**: Relación entre deriva y desviación

1.2.2 Reloj lógicos

Los relojes lógicos son usados para modelos asíncronos.

Representan una relación de causalidad entre eventos de envío y recepción de mensajes entre procesos secuenciales.

Relación de orden local: En un sistema de N procesos, es una relación binaria tal que si un proceso P_i observa e antes de e', $e \rightarrow_i e'$

1.2.2.1 Relojes de Lamport (escalares)

Los relojes de Lamport definen relaciones de orden global. Así, si a localmente precede a b en un proceso, a precede a b globalmente ($a \rightarrow b$).

- Para todo mensaje m, su envío precede a su recepción
- La relación es casi de orden parcial salvo por la no reflexividad
- En $a \rightarrow b$, a afecta causalmente a b
- Los eventos no ordenados causalmente son **eventos concurrentes**

Condiciones que deben satisfacer

- C_i es el reloj local del proceso P_i
- Si $a \rightarrow b$ en el proceso P_i, $C_i(a) \leq C_i(b)$
- Sea a un envío de mensaje m desde P_i, y b la recepción del mensaje m eb P_j, $C_i(a) < C_j(b)$

Reglas de implementación

- Antes de estampillar un **evento local** del proces P_i, hacer $C_i = C_i + 1$
- Siempre que un mensaje m **se envía** de P_i a P_j:
  - P_j ejecuta $C_i = C_i + 1$ y envía el nuevo C_i con m
  - P_j recibe C_i con m y ejecuta $C_j = max(C_j, C_i) + 1$. El evento recibir(m) se almacena con el nuevo C_j

1.2.2.2 Relojes vectoriales

Se diferencia de los relojes escalares en que no solo mantiene el valor de su reloj sino de todos los relojes que (cree) que tienen el resto de los procesos en un vector. Los procesos ahora comparten el vector de relojes, no solo un reloj.

- **Reloj vectorial de un proceso i**: $V_i[1..N], N = \text{n}^\text{o}$ procesos
- El valor del reloj que cree que un proceso i tiene de un proceso $j \neq i$ es $V_i[j]$
- El vector empieza en 0 en todos sus campos

Reglas de actualización

- Antes de estampillan un **evento local** de P_i, $V_i[i] = V_i[i] + 1$
- Siempre que un mensaje m se envía de P_i a P_j:
  - En P_i suma 1 a su reloj y envía el vector junto al mensaje a P_j, y estampilla
  - En P_j, suma 1 a su reloj y al recibir un mensaje de P_i, antes de estampillar, mezcla ambos vectores:
    - $V_j[j] = V_j[j]$ (su reloj se mantiene igual)
    - $V_j[k] = max(V_j[k], V_i[k]) \forall k \neq j$ (actualiza relojes retrasados respecto a P_i)

Características

- Para todo $i, j, $V_i[i] \geq V_j[i]$ (El reloj de un proceso más actualizado es la del propio proceso)
- $a \rightarrow b$ solo si $V[a] \leq V[b]$
- Notación
  - $V = V'$ si $V[i] = V'[i] \forall i \in 1..N$ (Dos vectores son iguales sólo si sus componentes son iguales)
  - $V \geq V'$ si $V[i] \geq V'[i] \forall i \in 1..N$ (Un vector es mayor o igual que otro si todas sus componentes son mayores o iguales al otro)