03_01_Synchronization

1. Background

1.1. Neccissity

Computers may need a notion of time

- for **assigning timestamps** to events
- for **comparing timestamps** to order events

1.2. Achievement

In asynchronous distributed systems, synchronization has to be **achieved through message**

2. Happened-before Relation (Causality Relation)

2.1. Happened-before Relation (Causality Relation)

The **happened-before relation** (or **causality relation** $\longrightarrow$) on E is the the smallest relation satisfying

- **local order**: if a,b in E_i, and a occurs before b, then a $\longrightarrow$ b
- **message exchange**: if a in E_i is the event of sending a message m and b in P_j is the event of receiving m, then a $\longrightarrow$b
- **transivity:** if a $\longrightarrow$ b and b $\longrightarrow$ c, then a $\longrightarrow$ c

2.2. Concurrency Relation:

if neither $a \longrightarrow b$ nor $b \longrightarrow a$ holds, then a and b are **concurrent** (notation: **a||b**)

3. Logical Clocks

3.1. Requirements

Timestamps have to obey the HB relation

Consistent

A logical clock is a function C: E $\longrightarrow$ S which is **consistent** with the HB relation:

if a b, then C(a) < C(b

$$\text{if } a \rightarrow b, \text{then } C(a) < C(b)$$

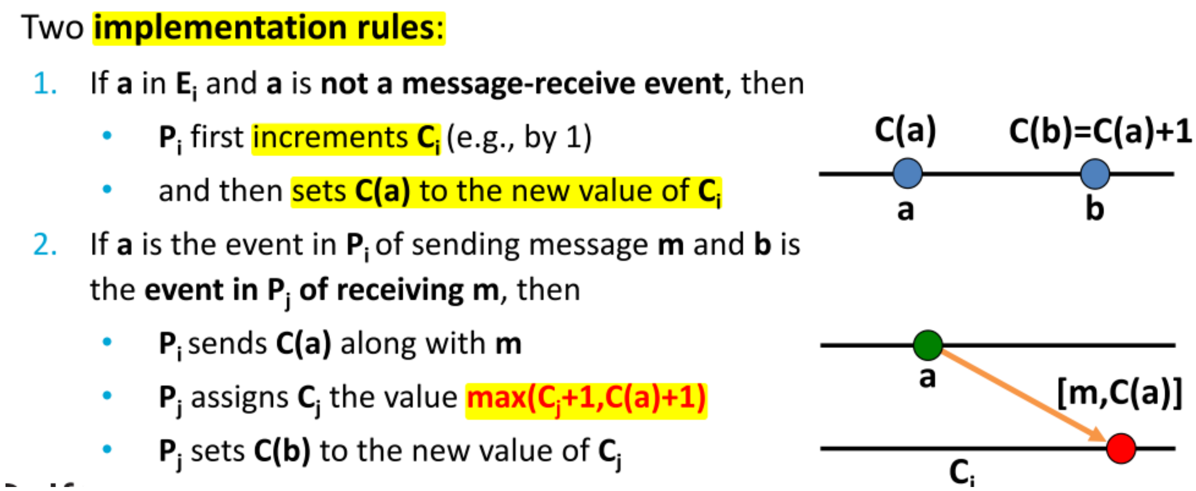
Characterizes

A logical clock **characterizes** the HB relation if:

$$C(a) < C(b) \text{ iff } a \rightarrow b$$

3.2. Scalar Clocks (Lamport Clocks)

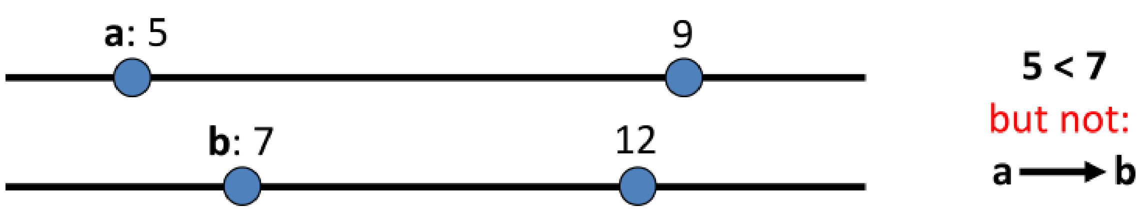
- Assume an **integer counter** C_i in process P_i
- Initialization of the C_i is irrelevant



Problem

A scalar clock **cannot characterize** the HB relation:

- take a set of processes with **only internal events** (no messages)
- any pair of events in any two different processes are **concurrent**, but will in general not have equal clock values



3.3. Vector Clocks

Vector Clocks

- **Comparison** of two vectors v, w of length k:

$$
\begin{array}{lll}
v = w & \text{iff } v[i] = w[i], & i = 1, 2, \ldots, k \\
v \leq w & \text{iff } v[i] \leq w[i], & i = 1, 2, \ldots, k \\
v < w & \text{iff } v[i] \leq w[i], & i = 1, 2, \ldots, k, \text{ and } v \neq w \\
v \geq w & \text{iff } v[i] \geq w[i], & i = 1, 2, \ldots, k \\
v > w & \text{iff } v[i] \geq w[i], & i = 1, 2, \ldots, k, \text{ and } v \neq w
\end{array}
$$

- Component-wise **maximum**

$$\max(v, w)[i] = \max(v[i], w[i])$$

- **Unit vector** in dimension i:

$$e_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$$

Prepare

Assume a **vector of integers** V_i in process P_i

Initialization: $V_i = (0, 0, \ldots, 0)$

Implementation Rules

1. If **a** in **E_i** and **a** **is not a message-receive event**, then
   - **P_i** first increments **V_i[i]** by 1 /* count local events */
   - and then sets **V(a)** to the new value of **V_i**

2. If **a** is the event in **P_i** of sending message **m** and
   b is **the event in P_j of receiving m**, then /* next event in P_j */
   - **P_i** sends **V(a)** along with **m**
   - **P_j** assigns **V_j** the value **max(V_j+e_j,V(a))**
   - **P_j** sets **V(b)** to the new value of **V_j**

Meaning

- In $V_i[i]$, P_i simply numbers its local events
- The meaning of $V_i[j]$ with $i \neq j$ is: the number of the last event in P_j that P_i **"knows about,"** either directly or indirectly
- Two events a and b are **concurrent** if there two indices i and j such that
  - V(a)[i] > V(b)[i]
  - V(a)[j] < V(b)[j]
  - Above means timestamps of a and b are incomparable: at event a there is more "knowledge" about process P_i and at event b there is more "knowledge" about P_j

Characterize Analysis

- if $a \to b$, then $V(a) < V(b)$
- if $V(a) < V(b)$:
  - suppose a occurs in P_i and b in P_j
  - because $V(b)[i] \geq V(a)[i]$, at event b there is **at least the same amount of knowledge** about P_i as at event a
  - this can only be caused by a chain of events from a to b, so $a \to b$

3.4. Characterizing the HB relation

Theorem: (See Slides)

For a **k-dimensional vector clock** to characterize the HB relation in a system with n processes, we need $k \geq n$, that is, **size of vector clock is at least equal to the number of processes**