Logical Clock (Lamport Clock)

Li intialized to 0 at process Pi

- Compute event at process Pi:
  - Increment Li by 1
  - New value of Li is the timestamp of the compute event
- Send event at process Pi: Consider e = send(m)
  - Increment Li by 1
  - New value of Li is the timestamp of send event e
  - Piggyback the timestamp of e with message m
- Receive event at process Pi: Suppose (m,t) where m is a message, and t is the piggybacked timestamp, is received at event e at Pi
  - Update Li as Li := max(Li, t)+1

Vector Logical Clocks

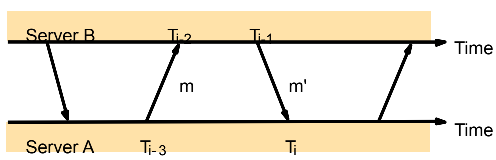
- ❖ With Lamport Logical Timestamp
  - e → f ⇒ timestamp(e) < timestamp (f), but
  - timestamp(e) < timestamp (f) ⇒ {e → f} OR {e and f concurrent}
- ❖ Vector Logical time addresses this issue:
  - ❑ Each process maintains a vector clock, length = number of processes
  - ❑ At each event, process i increments i^{th} element of vector V_i
    - ➔ The new V_i is the timestamp of the event
  - ❑ A message carries the Send event's vector timestamp
  - ❑ For a receive(message) event at process k … let $V_{message}$ denote vector timestamp received with the message

$$V_k[j] = \begin{cases} Max(V_k[j],\ V_{message}[j]), & \text{if } j \text{ is not } k \\ V_k[j] + 1 & j = k \end{cases}$$

Comparing Vector Timestamps

- ❖ $VT_1 = VT_2$,
  - *iff* $VT_1[i] = VT_2[i]$, for all i = 1, … , n
- ❖ $VT_1 \leq VT_2$,
  - *iff* $VT_1[i] \leq VT_2[i]$, for all i = 1, … , n
- ❖ $VT_1 < VT_2$,
  - *iff* $VT_1 \leq VT_2$ &
  - $\exists$ j ($1 \leq j \leq n$ & $VT_1[j] < VT_2[j]$)
- ❖ VT_1 is concurrent with VT_2
  - *iff* (not $VT_1 \leq VT_2$ AND not $VT_2 \leq VT_1$)

Theoretical Base for NTP



- *t* and *t'*: actual transmission times for *m* and *m' (unknown)*
- *o*: true offset of clock at *B* relative to clock at *A*
- o_i: estimate of actual offset between the two clocks
- d_i: estimate of accuracy of o_i ; total transmission times for *m* and *m'*; $d_i = t + t'$

$$T_{i-2} = T_{i-3} + t + o$$
$$T_i = T_{i-1} + t' - o$$

This leads to

$$d_i = t + t' = T_{i-2} - T_{i-3} + T_i - T_{i-1}$$
$$o = o_i + (t' - t)/2, \quad \text{where}$$
$$o_i = (T_{i-2} - T_{i-3} + T_{i-1} - T_i)/2.$$

It can then be shown that

$$o_i - d_i/2 \leq o \leq o_i + d_i/2.$$

Causal Ordering using vector timestamps

Algorithm for group member p_i $(i = 1, 2 \ldots, N)$

On initialization
  $V_i^g[j] := 0\ (j = 1, 2 \ldots, N);$ — The number of group-g messages from process j that have been seen at process i so far

To CO-multicast message m to group g
  $V_i^g[i] := V_i^g[i] + 1;$
  B-multicast$(g, \langle V_i^g, m \rangle);$

On B-deliver$(\langle V_j^g, m \rangle)$ *from* p_j, *with* $g = group(m)$
  place $\langle V_j^g, m \rangle$ in hold-back queue;
  wait until $V_j^g[j] = V_i^g[j] + 1$ and $V_j^g[k] \leq V_i^g[k]\ (k \neq j);$
  CO-deliver m; // after removing it from the hold-back queue
  $V_i^g[j] := V_i^g[j] + 1;$

Linearizability

An execution is linearizable if there exists a permutation that is

valid,

per-process order-preserving, and

real-time order-preserving