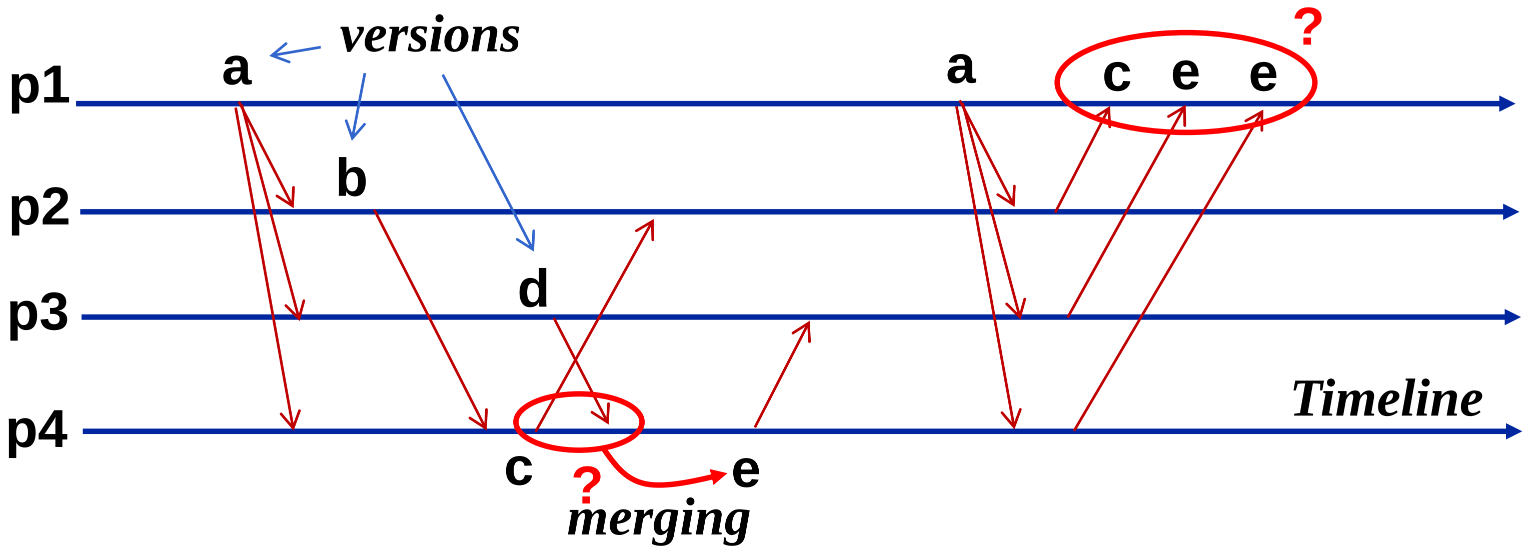
Distributed Systems
MIEEC, Fall 2018

Logical clocks

Luis Almeida / Pedro Souto
DEEC – **University of Porto**, Portugal

Temporal order

- Many applications do not require actual time
  - **Order** is enough
    - e.g., causality, versions control, distributed storage...
- Enforcing order in a distributed system is not trivial...
  - e.g., due to concurrent paths



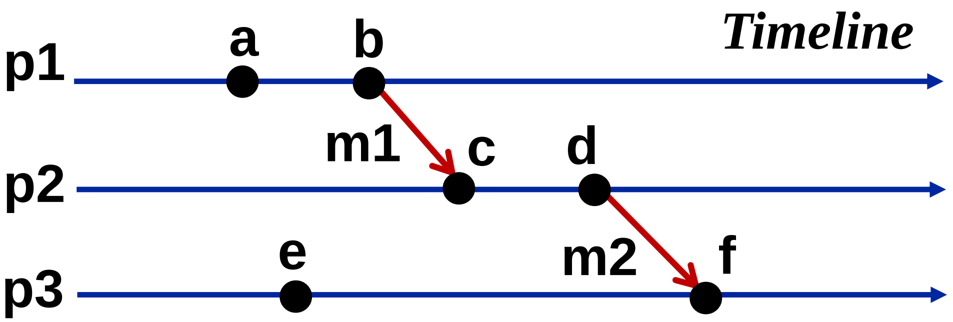
Precedence between events

- Certain events have **clear precedence** relationship
  - **Happened-before** relationship (→) proposed by L. Lamport
    - **HB1** (***events in the same process*** *always precede each other*):
      If e and e' are events occurring in one process in such order then $e \rightarrow e'$
    - **HB2** (***transmission of messages*** *must always precede reception*):
      If e is the transmission of a message and e' its reception then $e \rightarrow e'$
    - **HB3:** (***transitivity***)
      if $e \rightarrow e'$ and $e' \rightarrow e''$ then $e \rightarrow e''$
  - The HB relationship is **partial** since there may exist pairs of events to which it does not apply → **Concurrent events**

$$\neg (a \rightarrow e) \wedge \neg (e \rightarrow a) \Leftrightarrow a \,||\, e$$

Precedence between events

- **The HB relationship captures the control flow**



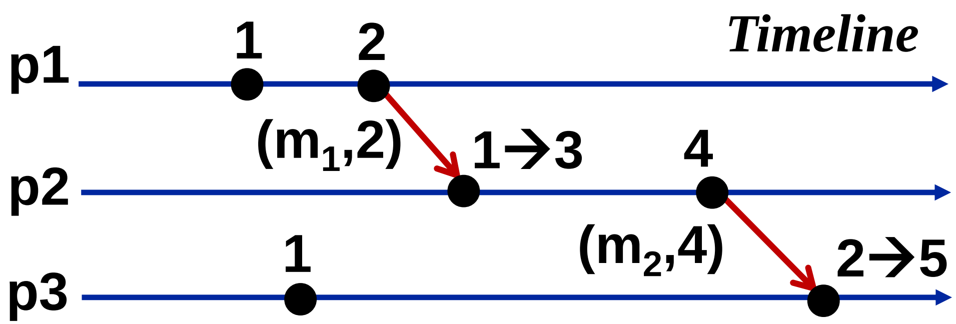
$a \rightarrow b$; $b \rightarrow c$; $c \rightarrow d$; $d \rightarrow f$; $e \rightarrow f$

$e \parallel (a,b,c,d)$

Lamport clocks and timestamps

- Beyond the HB relationship **Lamport** also proposed:
  - A **logical clock** per process that counts its relevant events
    - ***The Lamport clock***
  - The association of **timestamps** of such clock to **events**
    - ***The Lamport timestamps - L(e)***
- Rules for a **Lamport clock** L_i *of process* p_i
  - **LC1** (increments on each local event):
    - $\boldsymbol{L_i = L_i + 1}$ (just before executing an event)
  - **LC2** (synchronizes on message reception):
    - Every message *m* carries the timestamp of the sender $\boldsymbol{(m\ ,\ t = L_i)}$
    - The receiver p_j adjusts its clock to enforce HB: $\boldsymbol{L_j = \max(L_j, t) + 1}$

Lamport clocks and timestamps



- The relationship between **HB** and **Lamport timestamps**

$e \rightarrow e' \Rightarrow L(e) < L(e')$

L(e) < L(e') ???

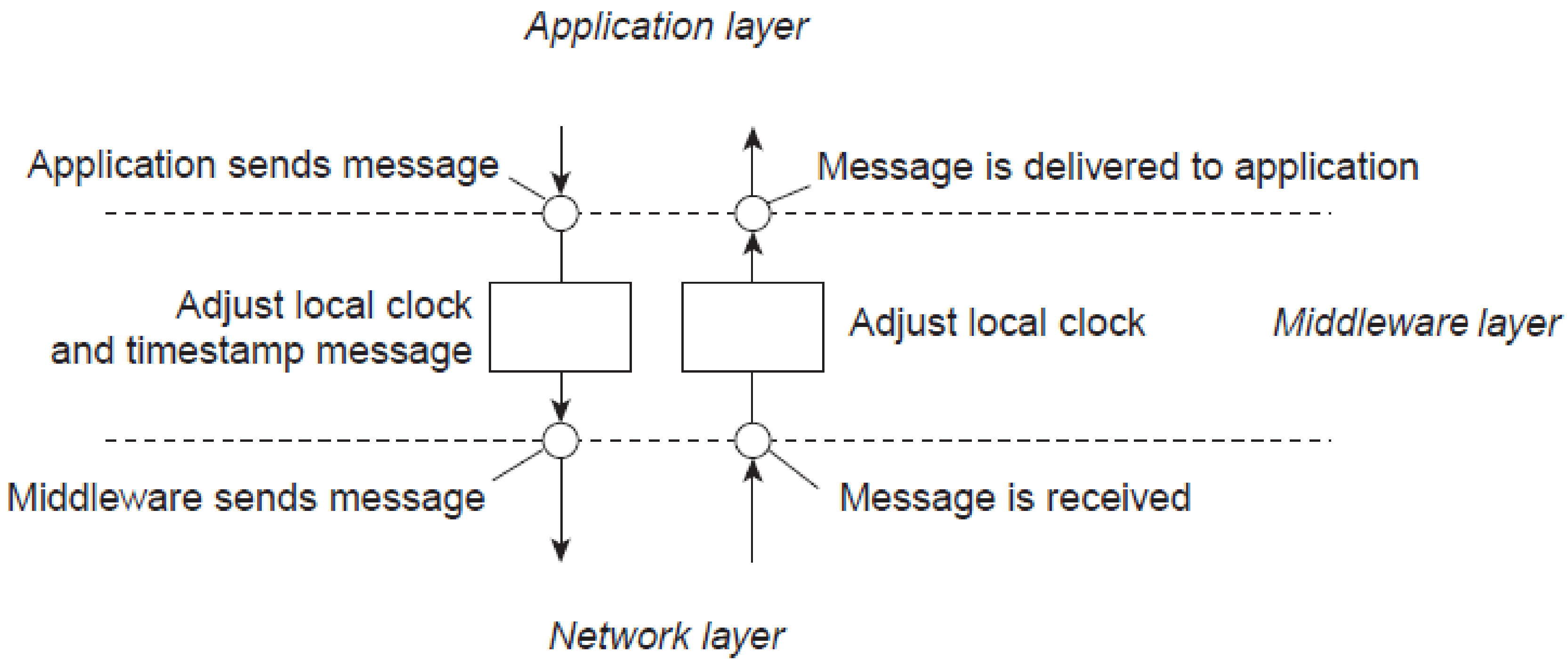
Cannot distinguish **precedence** from **concurrency**!

$L(e) \geq L(e') \Rightarrow \neg (e \rightarrow e')$

Distinguishes **not precedence** only!

Implementing Lamport clocks

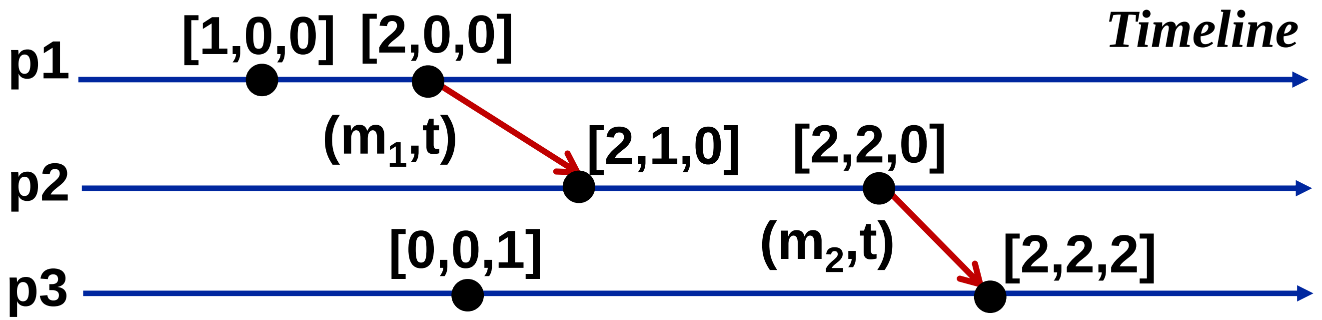
- **Middleware** layer
  - Separates message **reception** from **delivery**
  - Adjusts local clock



Vector clocks

- Vector with ***vision*** of **all clocks** in all processes
  - Independently proposed by Mattern and Fidge in 1988
- Process p_i has vector V_i
  - **VC1**: position *i* counts local events
    - $\mathbf{V_i}$ [i] **=** $\mathbf{V_i}$ [i] **+ 1** (just before executing an event)
  - **VC2:** timestamps are the whole vector
    - **t =** $\mathbf{V_i}$ (and sent with every message)
  - **VC3:** position j updated when receiving a message from p_j
    - $\mathbf{V_i}$ **[j] = max (**$\mathbf{V_i}$ **[j] , t [j])**
- Comparing vectors
  - $\mathbf{V < V'}$ *iff* $\forall_j\ \mathbf{V[j] \leq V'[j]} \ \wedge\ \exists_i\ \mathbf{V[i] < V'[i]}$

Vector clocks



- The relationship between **HB** and **Vector clocks**

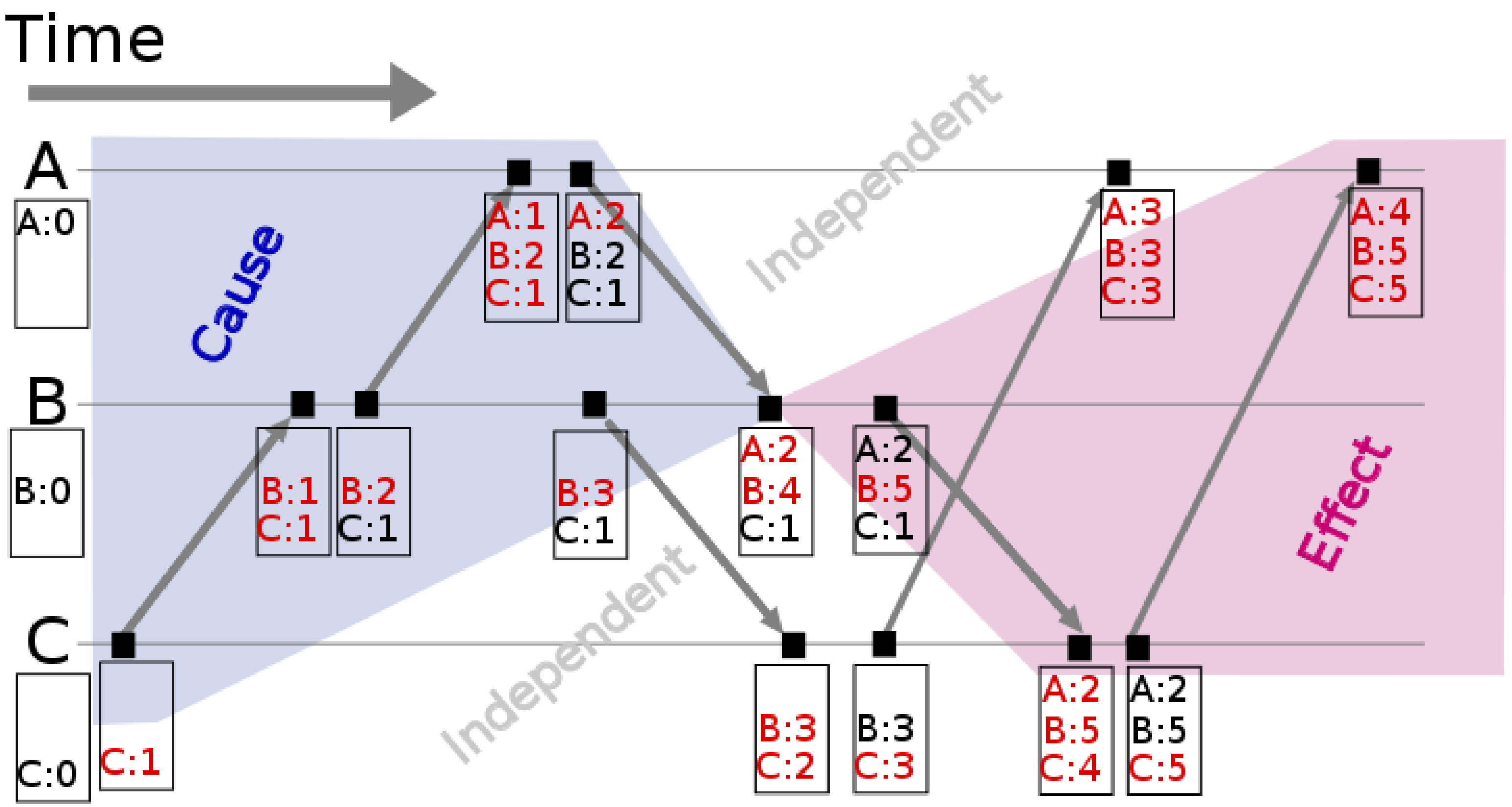
$$e \rightarrow e' \Leftrightarrow V(e) < V(e')$$

Distinguishes **precedence** from **concurrency**!

$$\neg\,(V(e) < V(e')) \wedge \neg(V(e') < V(e)) \Leftrightarrow e \,||\, e'$$

Concurrent events

Vector clocks



(from wikipedia)

Further readings

- Tanenbaum and van Stenn, Distributed Systems
  - Section 6.2: Logical clocks
- The wikipedia has interesting high level info
  - For a quick reference
- Very interesting sequence of post in the Basho Blog
  - http://basho.com/posts/technical/vector-clocks-revisited/
- Very interesting presentation on Version Vectors
  - https://www.youtube.com/watch?v=3SWSw3mKApM