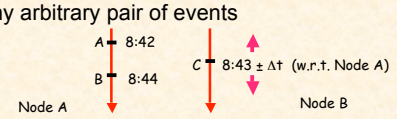
Logical Time

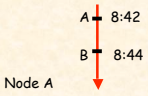
Logical Time

- We wish to order events in a distributed system
- We cannot **perfectly** synchronise clocks across a distributed system
- In general, we **cannot** use physical time to order any arbitrary pair of events



Logical Time

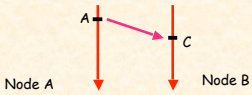
- However, two observations are useful
  - If two events occurred at the same process, then they occurred in the order in which they are observed



A occurs before B

Logical Time

- Whenever a message is sent between two processes, the event of sending the message occurs before the event of receiving the message!



A occurs before C

- This is the basis of Lamport's 'happened-before' relationship
  - causal ordering
  - Potential causal ordering

Causal Ordering

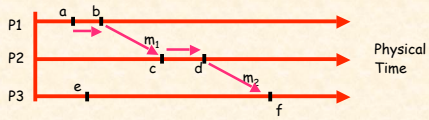
$x \xrightarrow{p} y$ events x and y occurred at process p in order x,y

$x \rightarrow y$ event x 'happened-before' y

$x \rightarrow y$ defined:

- Rule1: if ∃ process p: $x \xrightarrow{p} y$, then $x \rightarrow y$
- Rule 2: For any message m, send(m) → rcv(m)
  where send(m) is the event of sending the message and rcv(m) is the event of receiving the message
- Rule 3: If x,y and z are events such that $x \rightarrow y$, $y \rightarrow z$ then $x \rightarrow z$

Causal Ordering



We can infer the following…

a→b	Rule 1	b→c	Rule 2
c→d		d→f	
a→f	Rule 3		
b→d			
etc.			

Rule 3 allows us to define causal orderings across processes!

Causal Ordering

- Notes:
  - $a \not\rightarrow e$ and $e \not\rightarrow a$ because we cannot determine an ordering for a and e as there is no chain of messages relating them
  - a || e
  - Events a,b,c, .. F may or may not be causally related
  - Our scheme only captures potential causality

Logical Clocks

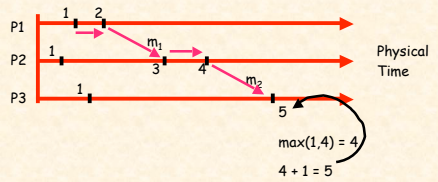
- Simple mechanism which captures causal ordering via numerical values
- Each process P keeps its own logical clock, Cp, which it uses to timestamp events
- Processes update their logical clocks and transmit the values of clocks as follows
  - Cp incremented before each event is issues at process p

    Cp := Cp + 1

Logical Clocks

- When process P sends a message m, it piggybacks the value t = Cp on m

  On receiving (m,t), a process q computes Cq := max(Cq,t) and then increments clock before timestamping the event rcv(m)

  i.e. $t_{rcv(m)} = Cq + 1$
- $a \rightarrow b \Rightarrow C(a) < C(b)$
  - where C(x) is timestamp at event x

Logical Clocks