Notice

Some figures are taken from third-party slide sets. In this module, figures are taken from the Tanenbaum/van Steen slide set:

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5
No Clock is Perfect

Clock time, C

UTC, t

Fast clock \( \frac{dC}{dt} > 1 \)
Perfect clock \( \frac{dC}{dt} = 1 \)
Slow clock \( \frac{dC}{dt} < 1 \)
Synchronizing Clocks

\[ dT_{\text{req}} \quad dT_{\text{res}} \]
Implications

Time according to local clock

Computer on which compiler runs

2144  2145  2146  2147

output.o created

Computer on which editor runs

2142  2143  2144  2145

output.c created
Event Ordering

- total order needed?
  - independent events

- partial order sufficient?
  - *causal* ordering
  - *happened-before* relationship
Happened Before

- if event $a$ occurs before $b$ in the same process, then $a \rightarrow b$

- if $a$ is sent event and $b$ is corresponding receive event, then $a \rightarrow b$

- transitivity: if $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$

- if not($a \rightarrow b$ or $b \rightarrow a$), then concurrent
Lamport Clock

- Clock: counter $C_i$ for process $P_i$

1. before each event: $C_i = C_i + 1$
2. attach $C_i$ to each message $m$ as $ts(m)$
3. upon receipt of $m$: $C_i = \max\{ C_i, ts(m) \}$
Lamport Clock

(a)
Lamport Clock

(b)
Vector Clock

- Lamport clock captures potential causality
  - might impose too strict ordering
  - independent events still appear ordered

- Clock: vector $V_i$ for process $P_i$
  - $V_i[j]$: number of preceding events at process $j$
  - $V_i[i]$: Lamport clock at process $i$
Vector Clock

1. before each event: \[ V_i[i] = V_i[i] + 1 \]

2. attach \( V_i \) to each message \( m \) as \( ts(m) \)

3. upon receipt of \( m \): \[ V_i[k] = \max\{ V_i[k], ts(m)[k] \} \]
   for each \( k \)

- overhead...
Vector Clock

Vector Clock detects potential causality only
Lamport Clock mandates stricter ordering
Causally And Totally Ordered Communication System

- controversy during 1990s
  - distributed system middleware
  - CATOCS expensive, no transactions
  - might not fit application requirements
- current situation
  - key/value stores vs. transactional DB systems
  - Paxos-type systems for high-level agreement
  - causal ordering used where applicable