CS 755 – System and Network Architectures and Implementation

Module 7 – Ordering

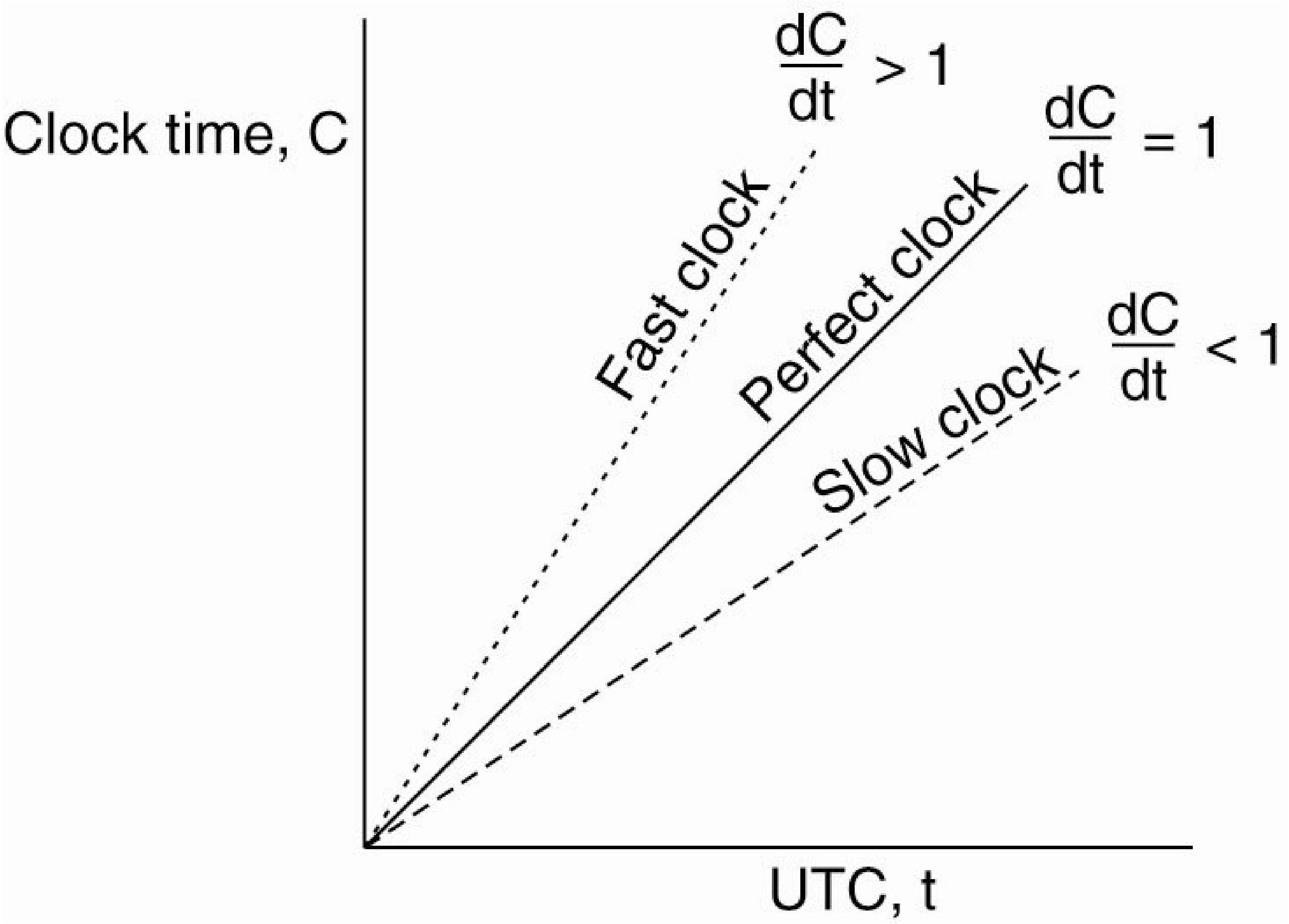
Martin Karsten

mkarsten@uwaterloo.ca

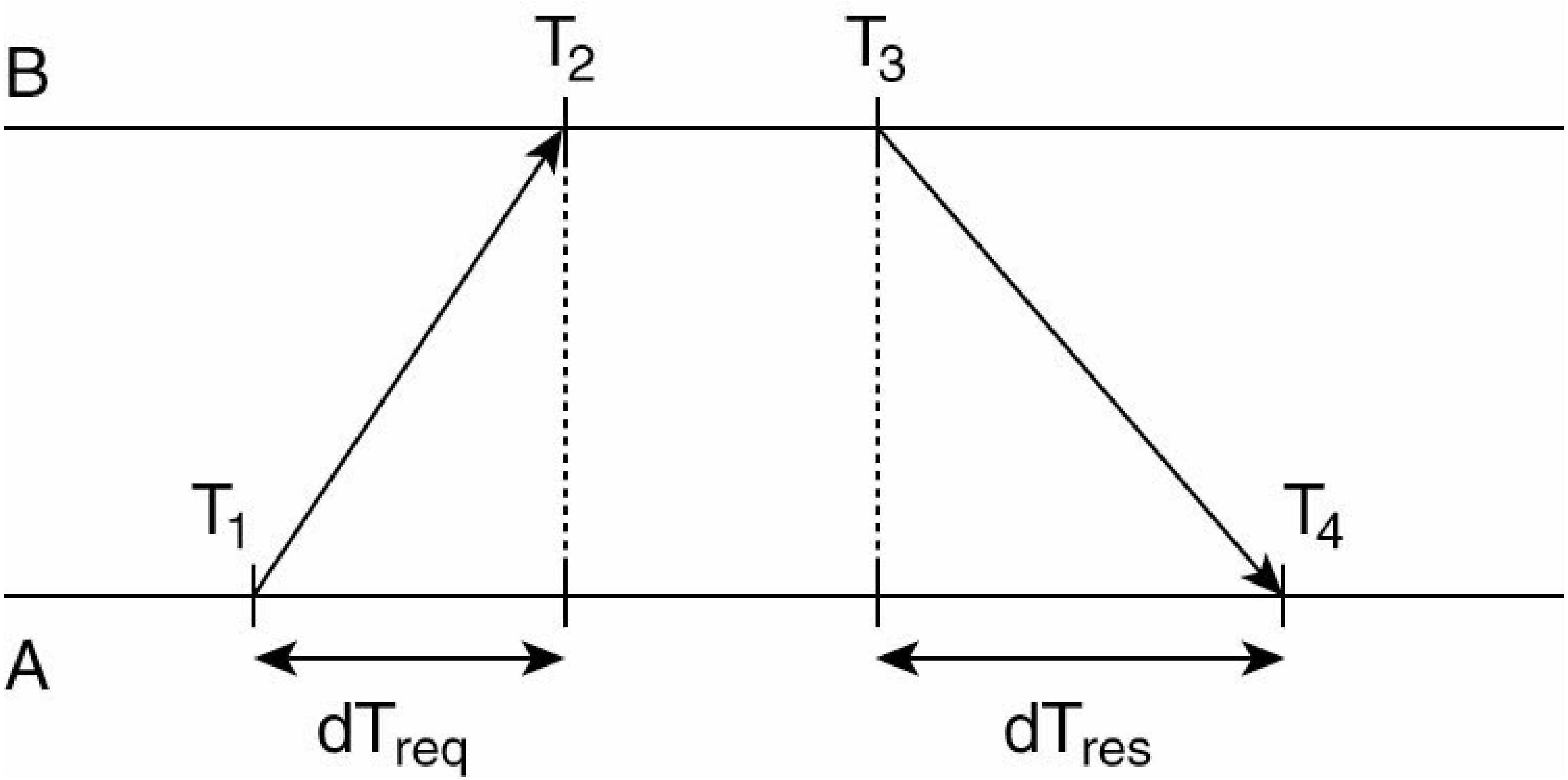
Notice

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

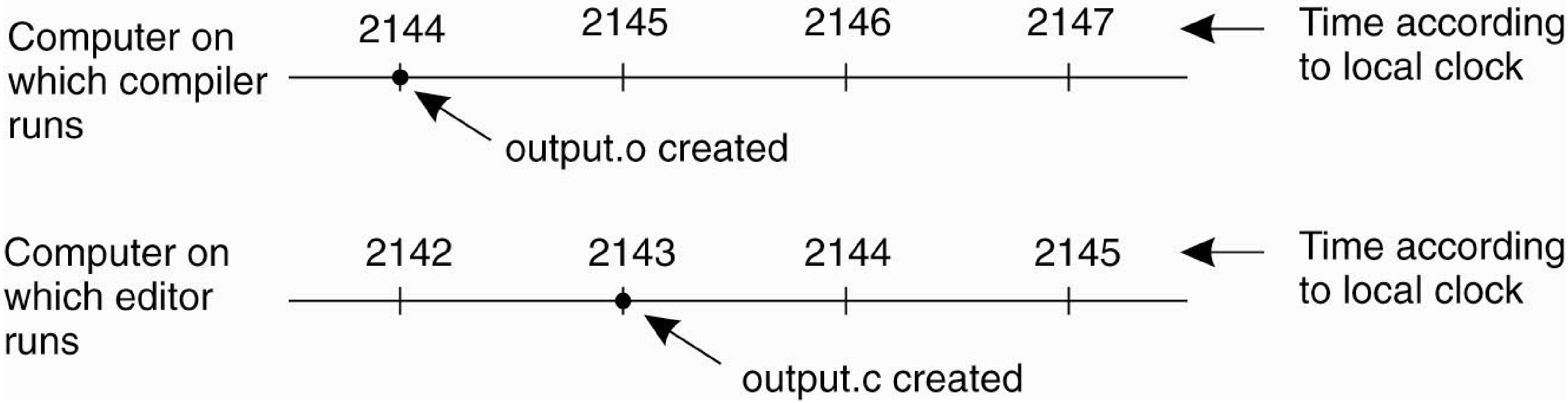
No Clock is Perfect



Synchronizing Clocks



Implications


Computer on which compiler runs
2144
2145
2146
2147
Time according to local clock
output.o created
Computer on which editor runs
2142
2143
2144
2145
Time according to local clock
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 $a \rightarrow c$

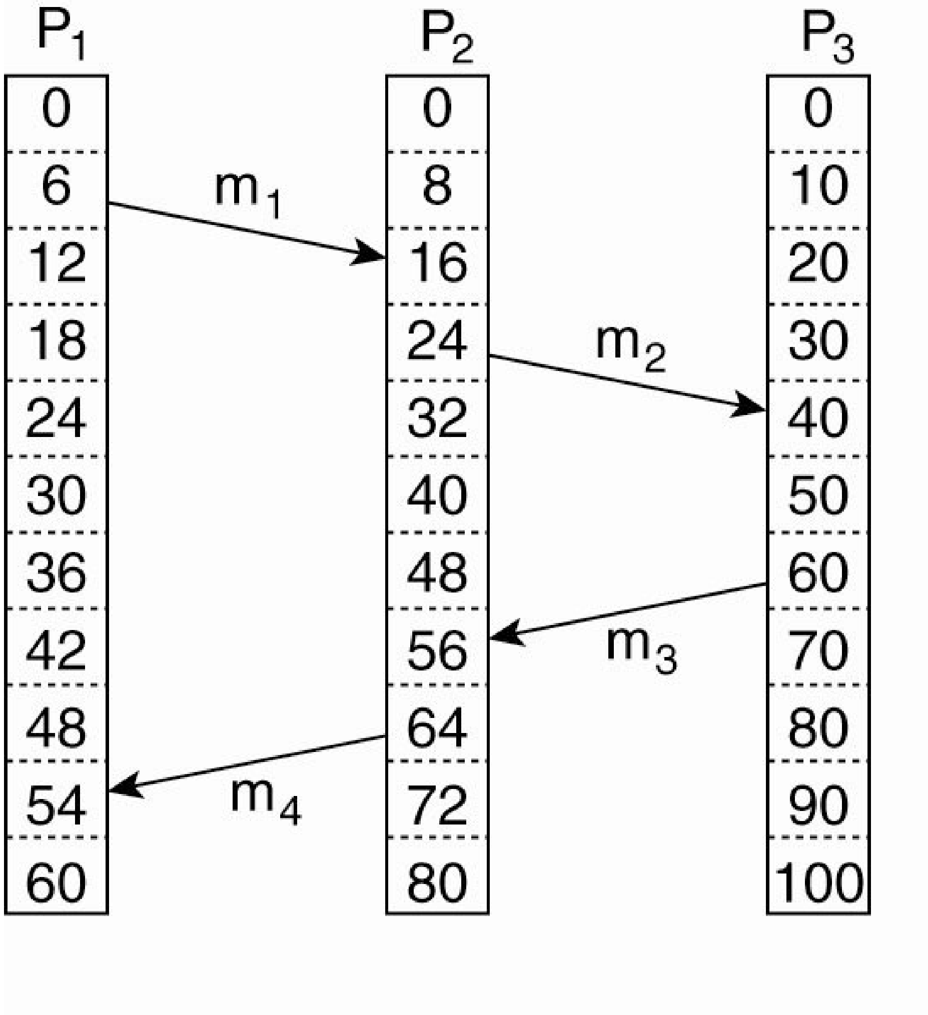
- if not($a \rightarrow b$ or $b \rightarrow a$), then *concurrent*

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: $C_i = \max\{ C_i, ts(m) \}$

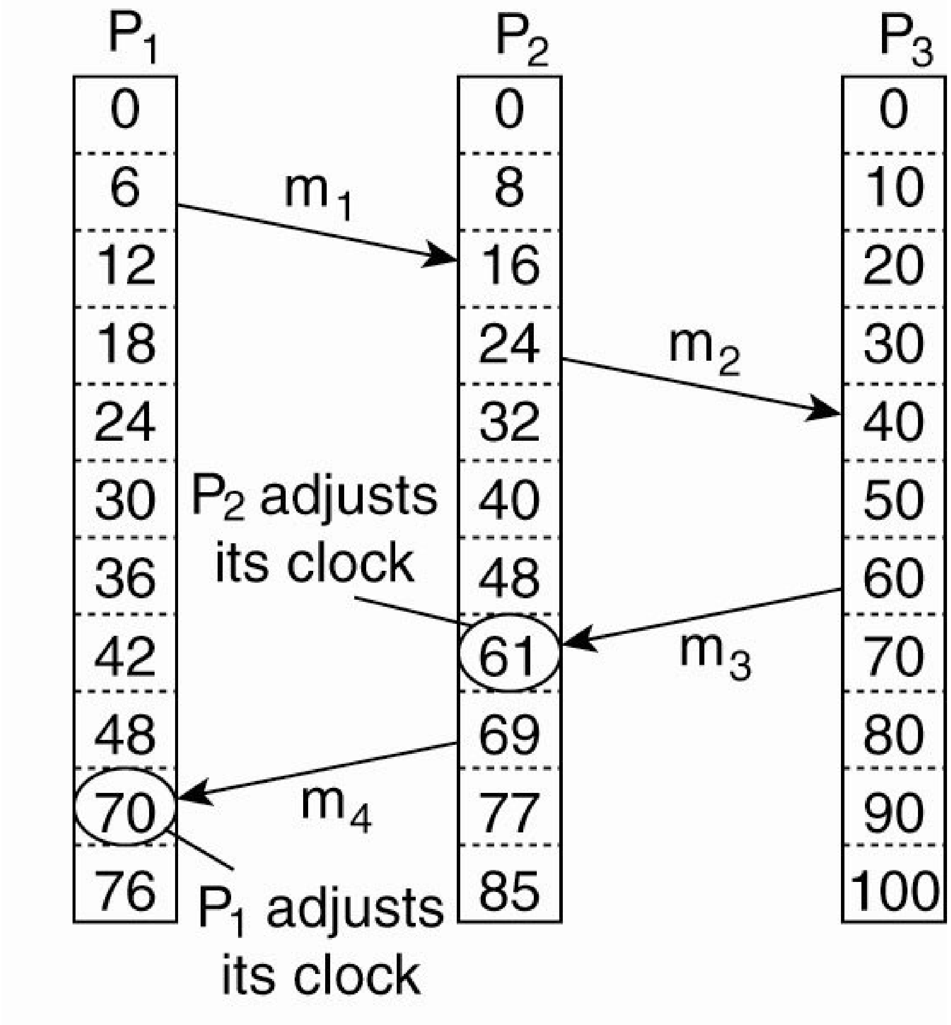
Lamport Clock



P_1	P_2	P_3
0	0	0
6	8	10
12	16	20
18	24	30
24	32	40
30	40	50
36	48	60
42	56	70
48	64	80
54	72	90
60	80	100

(a)

Lamport Clock



P1	P2	P3
0	0	0
6	8	10
12	16	20
18	24	30
24	32	40
30	40	50
36	48	60
42	61	70
48	69	80
70	77	90
76	85	100

(b)

Vector Clock

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

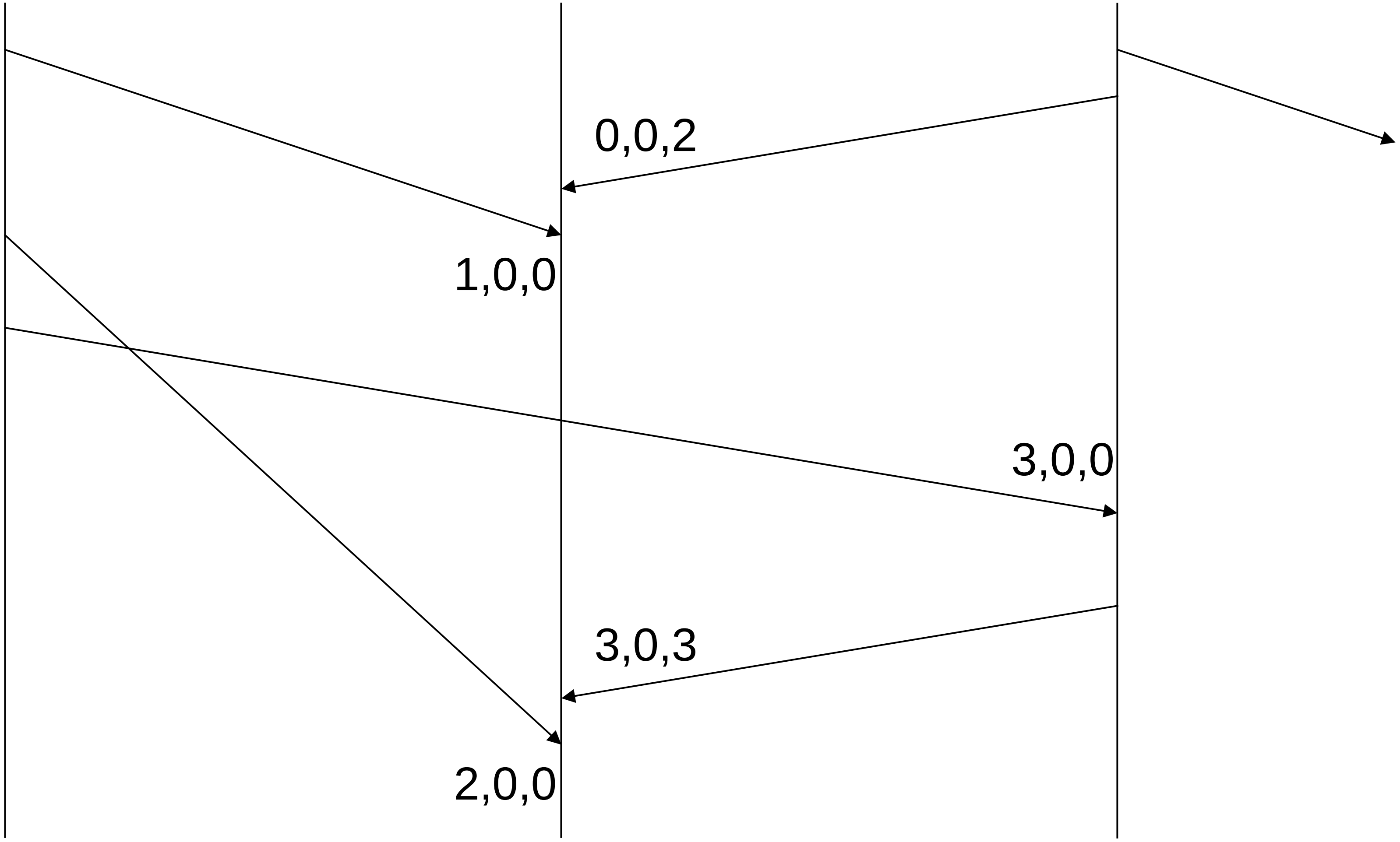
- Clock: vector V_i for process P_i
  - $V_i[j]$: number of preceedings events at process j
  - $V_i[i]$: Lamport clock at process i

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

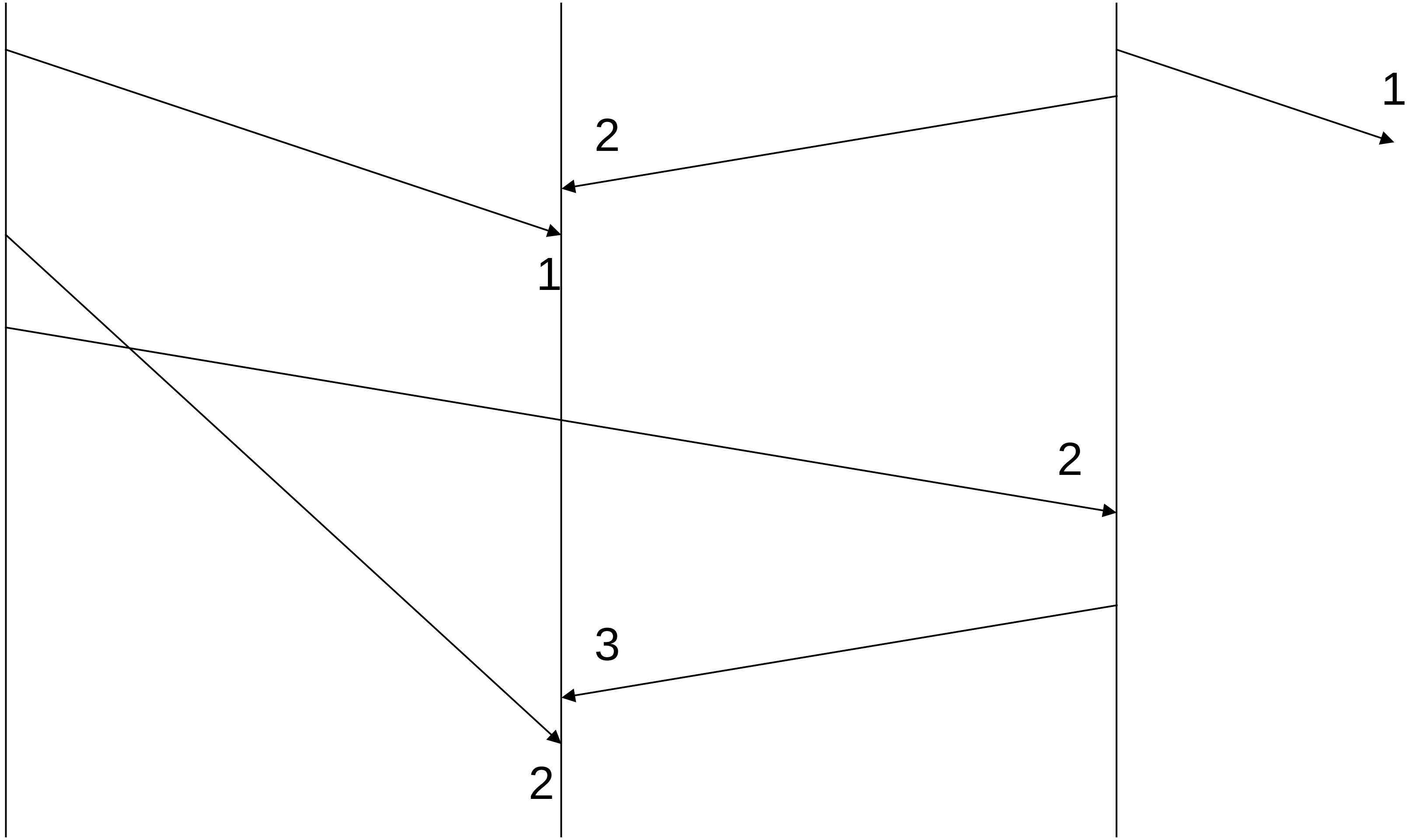
- overhead...

Vector Clock



Vector Clock detects potential causality only

Lamport Clock



Lamport Clock mandates stricter ordering

Causally And Totally Odered 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