COMP2004 Distributed Systems

Soumyabrata DEV

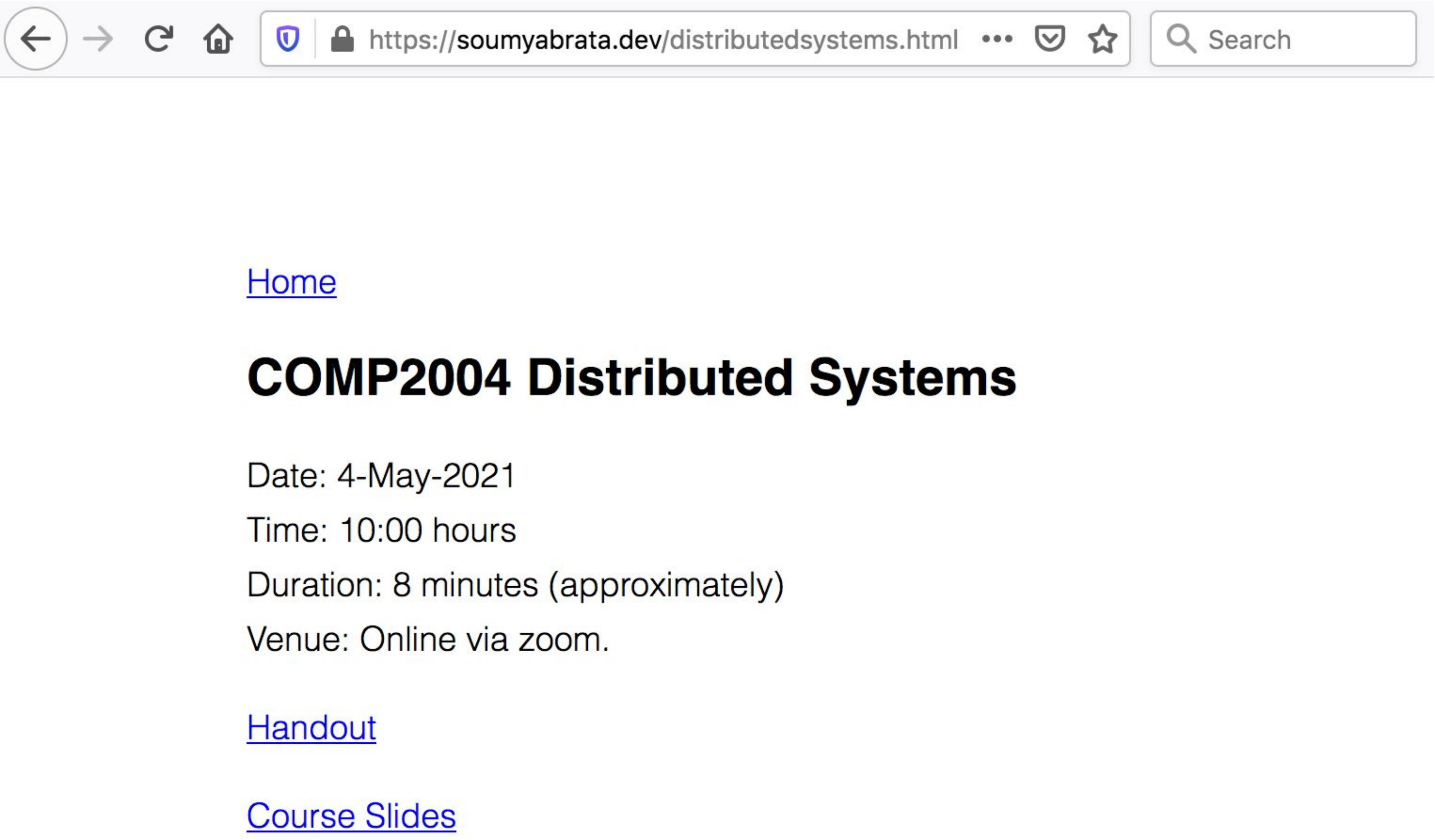
School of Computer Science
University College Dublin
https://soumyabrata.dev/

University College Dublin (UCD)
4-May-2021

Course Overview

Course website is available here:

https://soumyabrata.dev/distributedsystems.html



Home

COMP2004 Distributed Systems

Date: 4-May-2021

Time: 10:00 hours

Duration: 8 minutes (approximately)

Venue: Online via zoom.

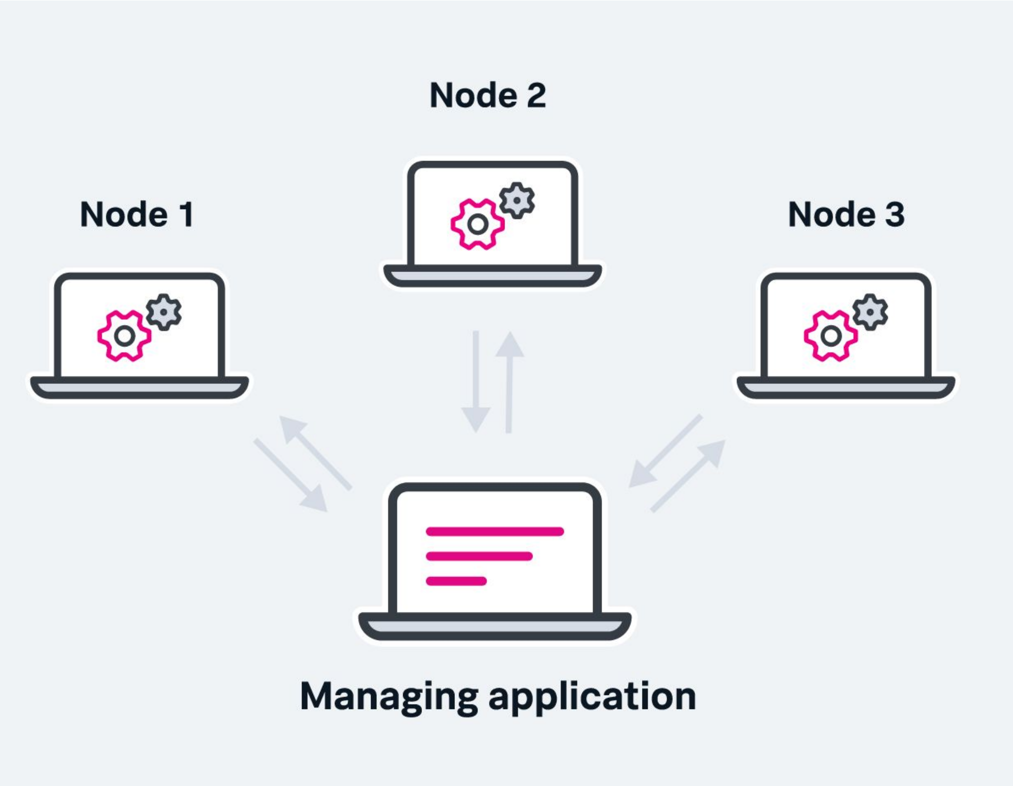
Handout

Course Slides

(1/3) Distributed system

Distributed System

A distributed system is a computing environment in which various components are spread across multiple computers on a network[1].

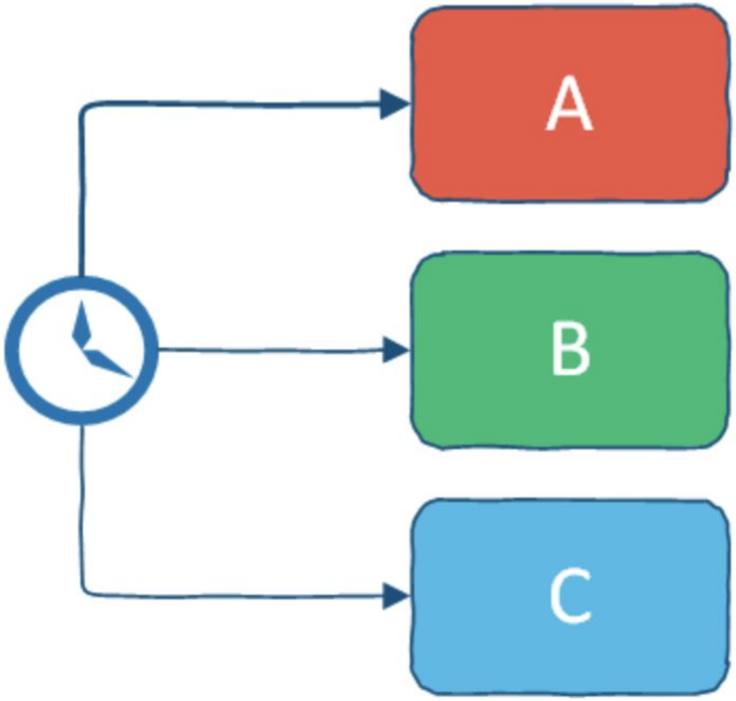


[1] Image archived from https://www.splunk.com/en_us/data-insider/what-are-distributed-systems.html

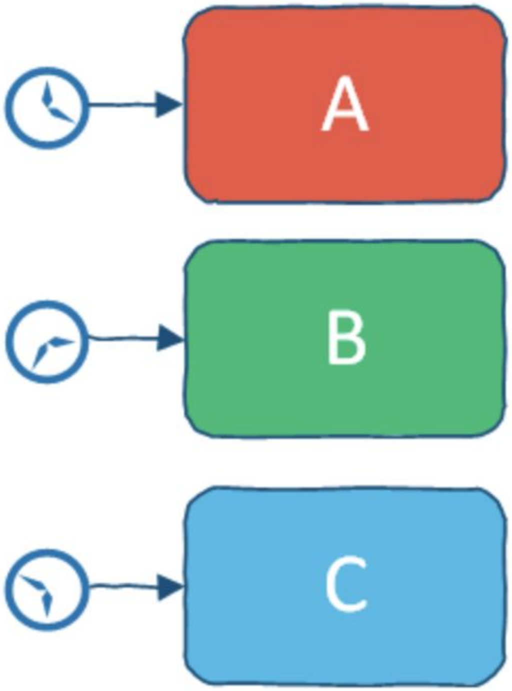
What is time?

Time is a source of order - it allows us to define the order of operations.

Global clock of total order



Local clock with skew and drift



(2/3) Lamport’s logical clock

Happened-before

Lamport’s “happened-before” notation

a → b event a happened before event b

eg. a: message being sent, b: message receipt

Transitive:

if a → b and b → c, then a → c

Logical clocks

Assign “clock” value to each event

- if $a \longrightarrow b$ then $clock(a) < clock(b)$
- since time cannot run backwards

If *a* and *b* occur on different processes that do not exchange messages, then neither $a \longrightarrow b$ nor $b \longrightarrow a$ are true

- These events are called concurrent

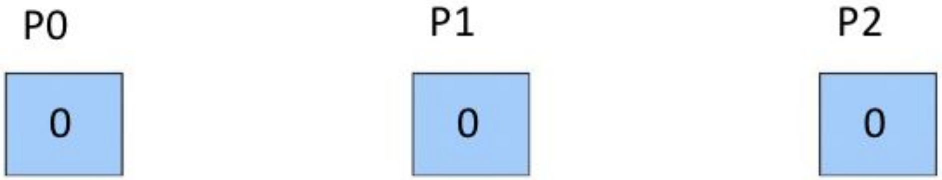
Lamport’s logical clock

Lamport’s clocks are defined as:

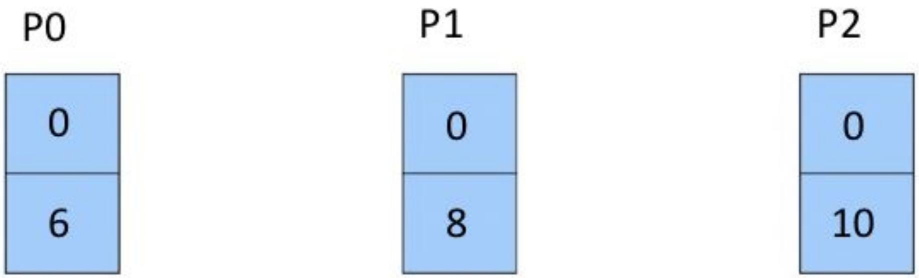
1. Events within a process is assigned an unique ID. It is a counter that increments from 1.
2. When sending a message from *A*, we set $\mathit{clock}(A) = \mathit{clock}\ (A) + 1$, then pass it as a part of the message.
3. On receipt of message in *B*, we set $\mathit{clock}(B) = \max(\mathit{clock}(B)+1, \mathit{clock}(A))$

(3/3) Illustration of Lamport’s logical clock

Example



Example



P0

0

6

P1

0

8

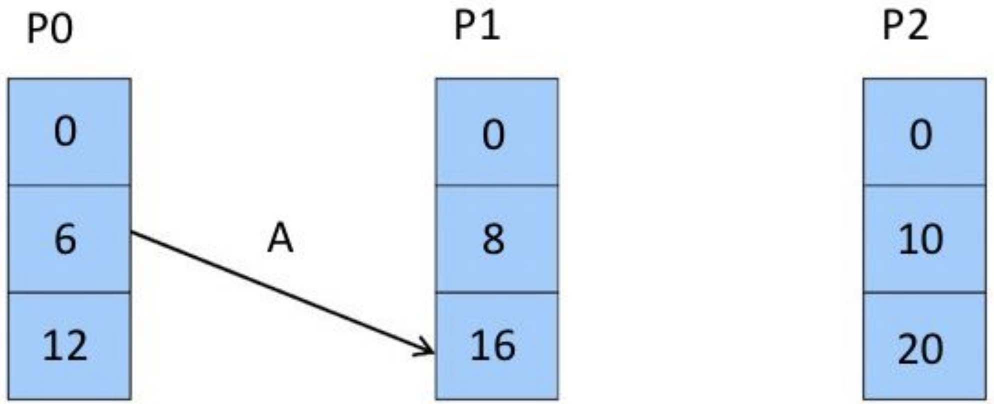
P2

0

10

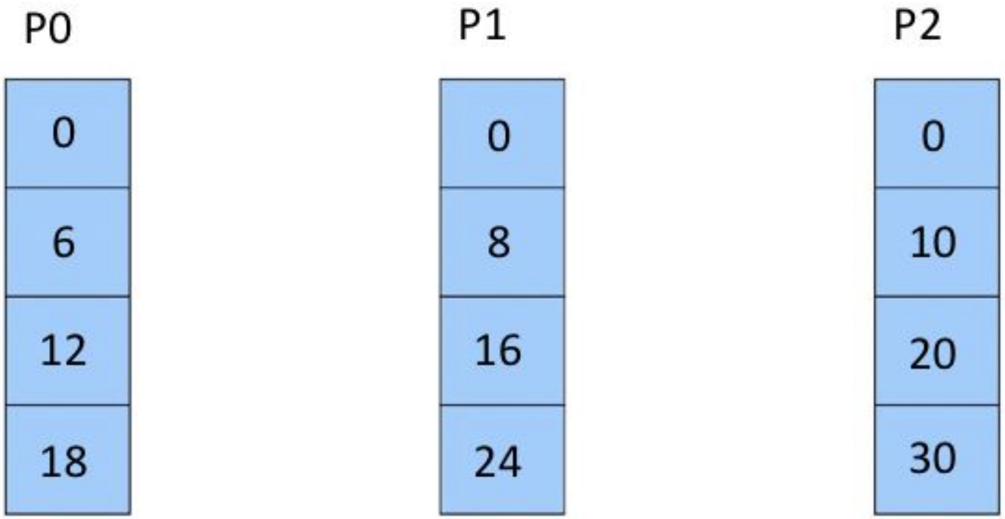
P0 sends message A to P1

Example



P1 receives message A (everything is OK since 6 < 16)

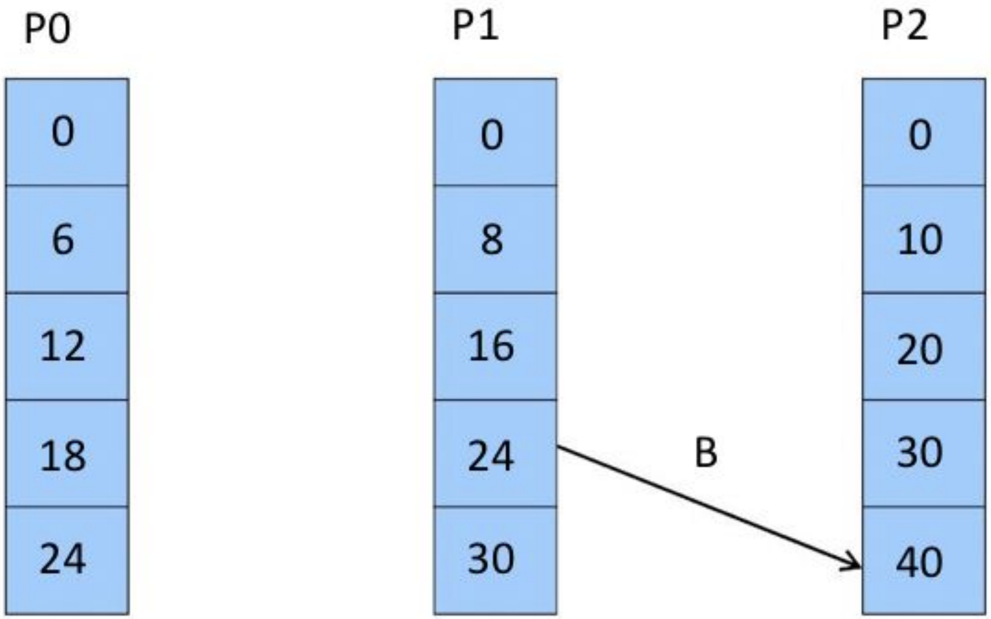
Example



P0	P1	P2
0	0	0
6	8	10
12	16	20
18	24	30

P1 sends message B to P2

Example

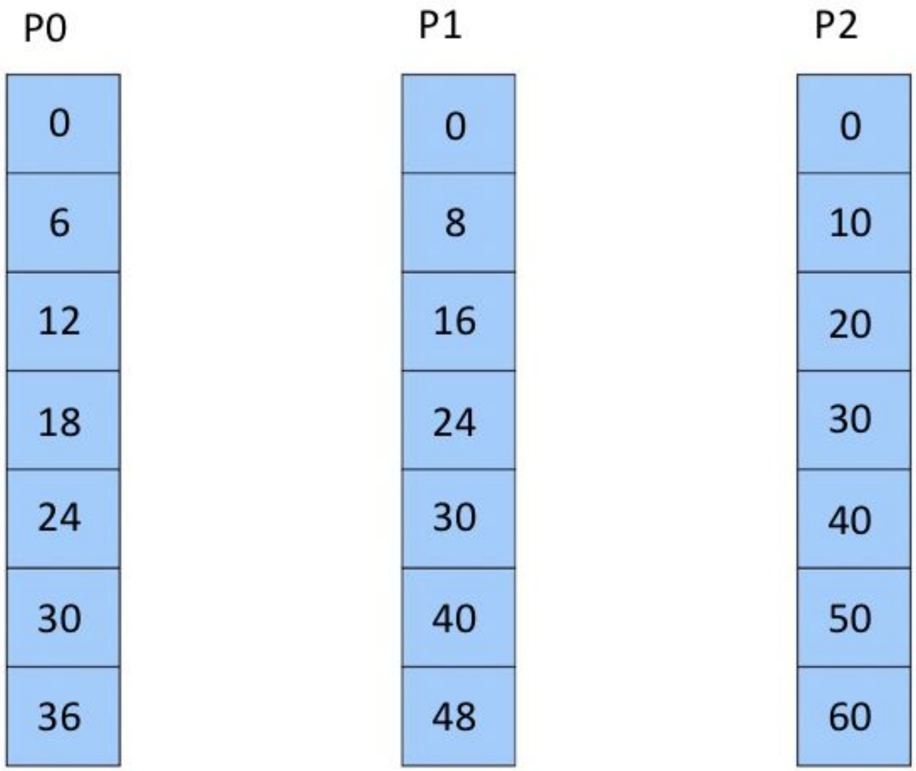


P2 receives message B (everything is OK since 24 < 40)

Example

P0	P1	P2
0	0	0
6	8	10
12	16	20
18	24	30
24	32	40
30	40	50

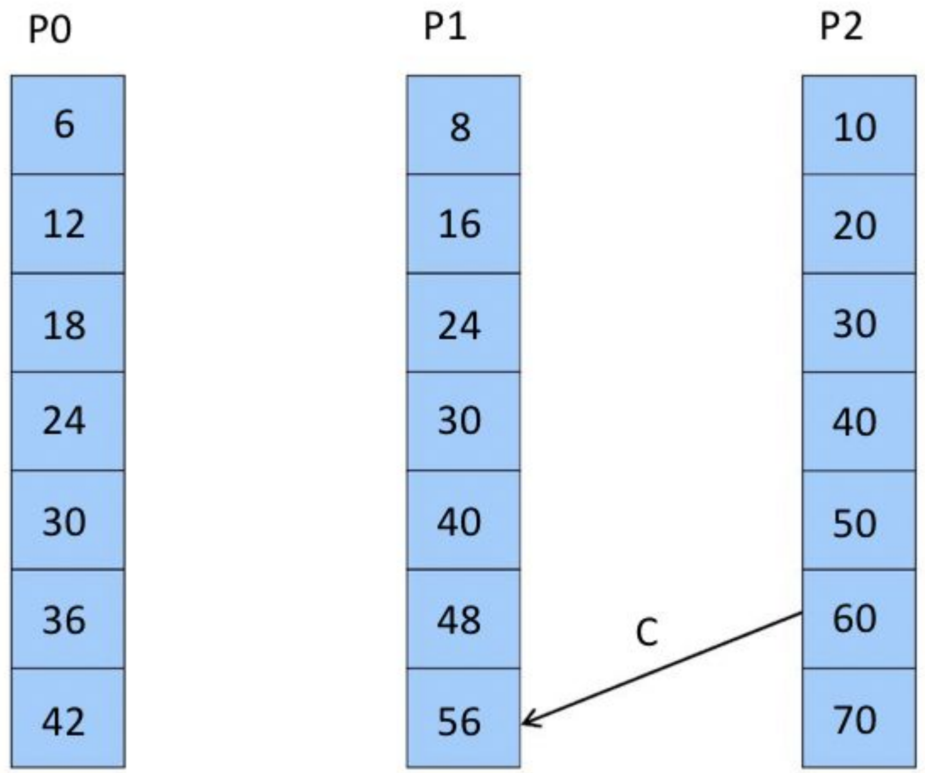
Example



P0	P1	P2
0	0	0
6	8	10
12	16	20
18	24	30
24	30	40
30	40	50
36	48	60

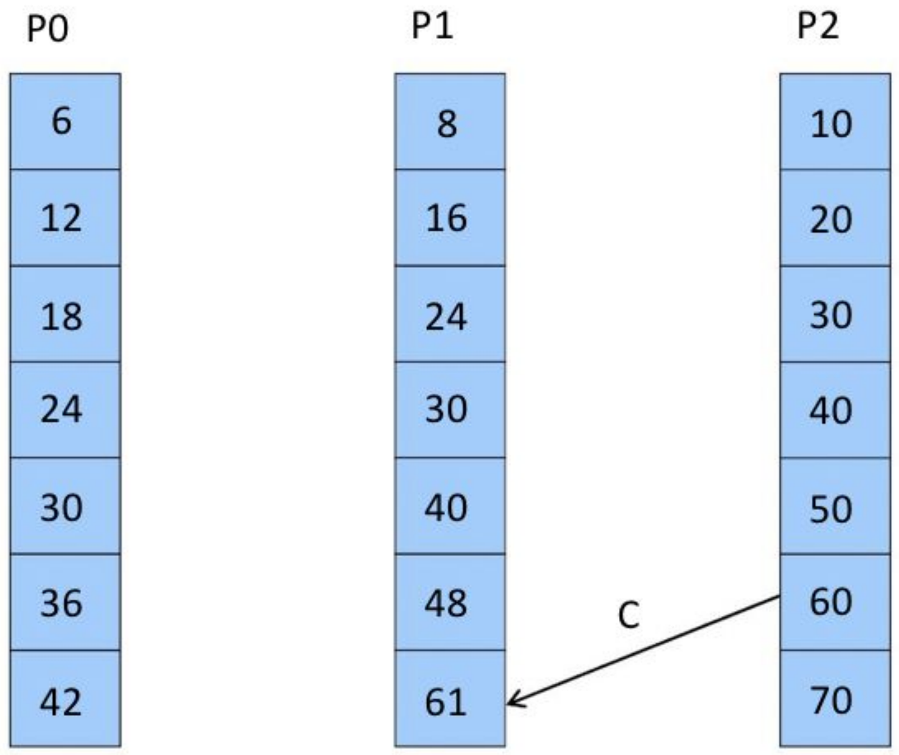
P2 sends message C to P1

Example



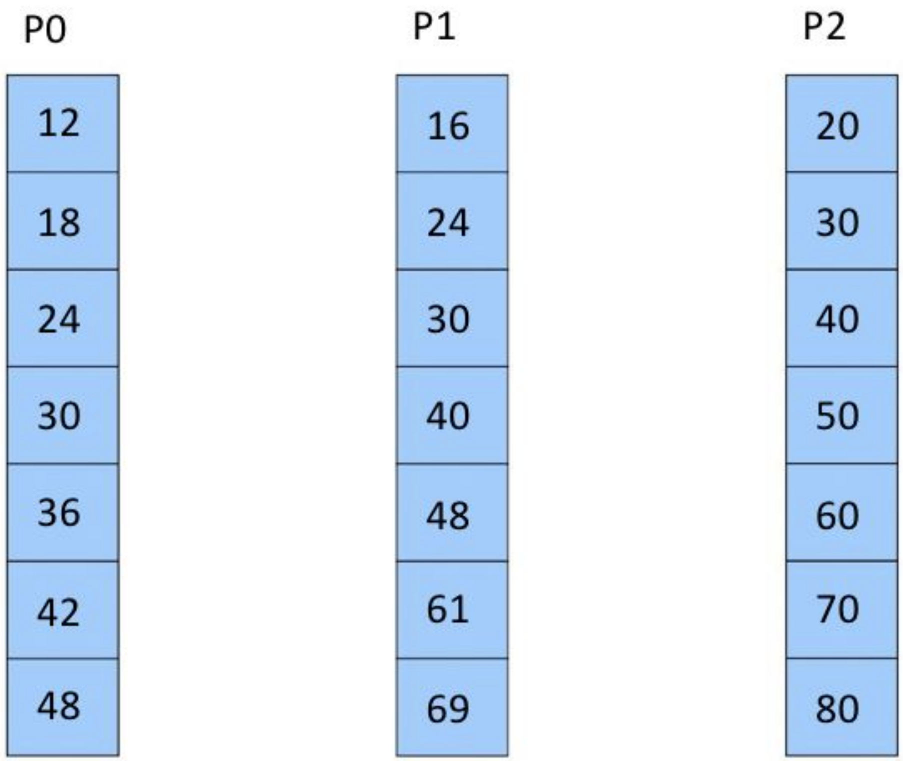
P1 receives message C (Ouch! The message was sent at time 60 but received at time 56)

Example



Logical Time at P1 updated to be 1 greater than the time C was sent at.

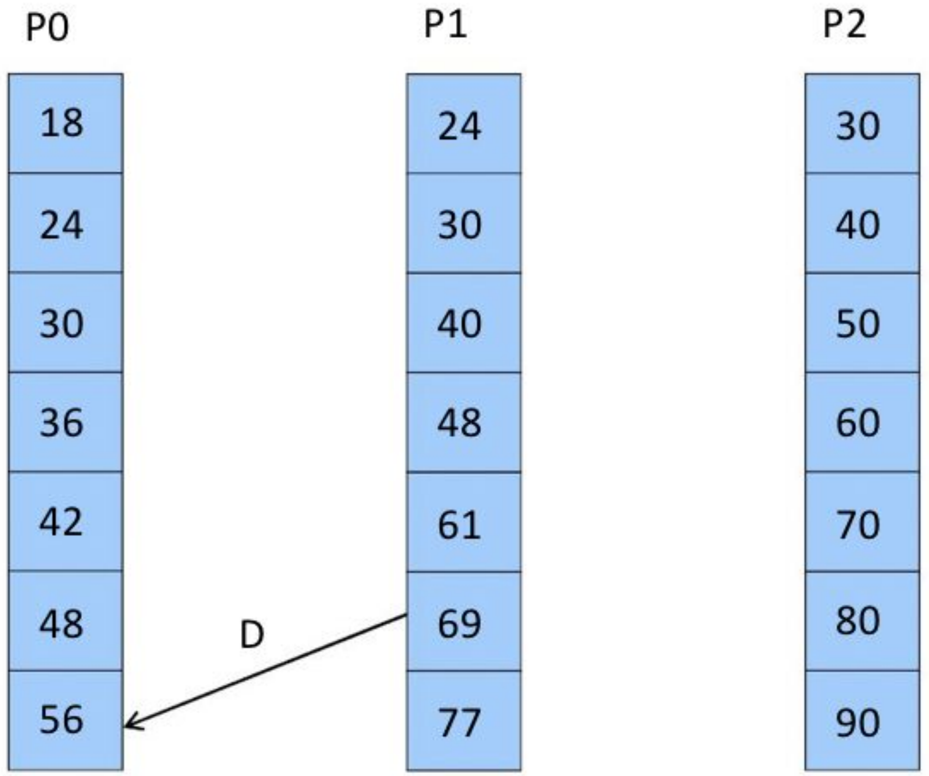
Example



P0	P1	P2
12	16	20
18	24	30
24	30	40
30	40	50
36	48	60
42	61	70
48	69	80

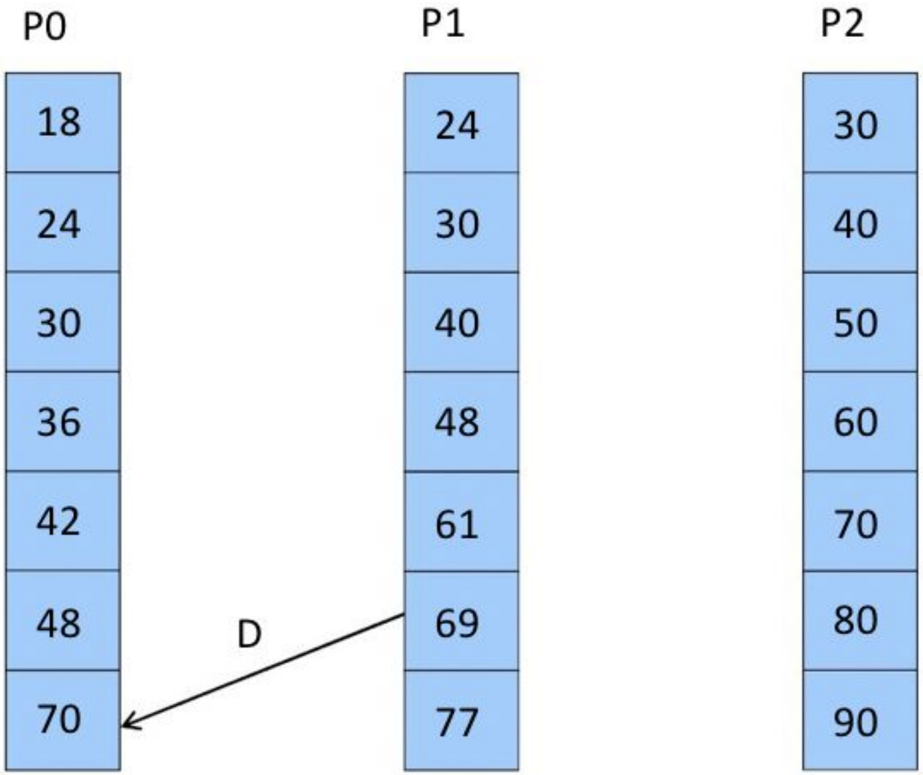
P1 sends message D to P0

Example



P0 receives message D (Ouch! 56 < 69)

Example



P0	P1	P2
18	24	30
24	30	40
30	40	50
36	48	60
42	61	70
48	69	80
70	77	90

Logical Time at P0 updated

Example

P0	P1	P2
24	30	40
30	40	50
36	48	60
42	61	70
48	69	80
70	77	90
76	85	100

End of Run

Limitation of Lamport’s algorithm

- Lamport clocks can guarantee that if a ⟶ b then C(a) < C(b).

- However it can't guarantee, that if C(a) < C(b) then event a happened before b.

- In order to overcome this limitation, we use vector clock to represent the causality of events.

Thank You