
# COMP2004 Distributed Systems

Soumyabrata DEV

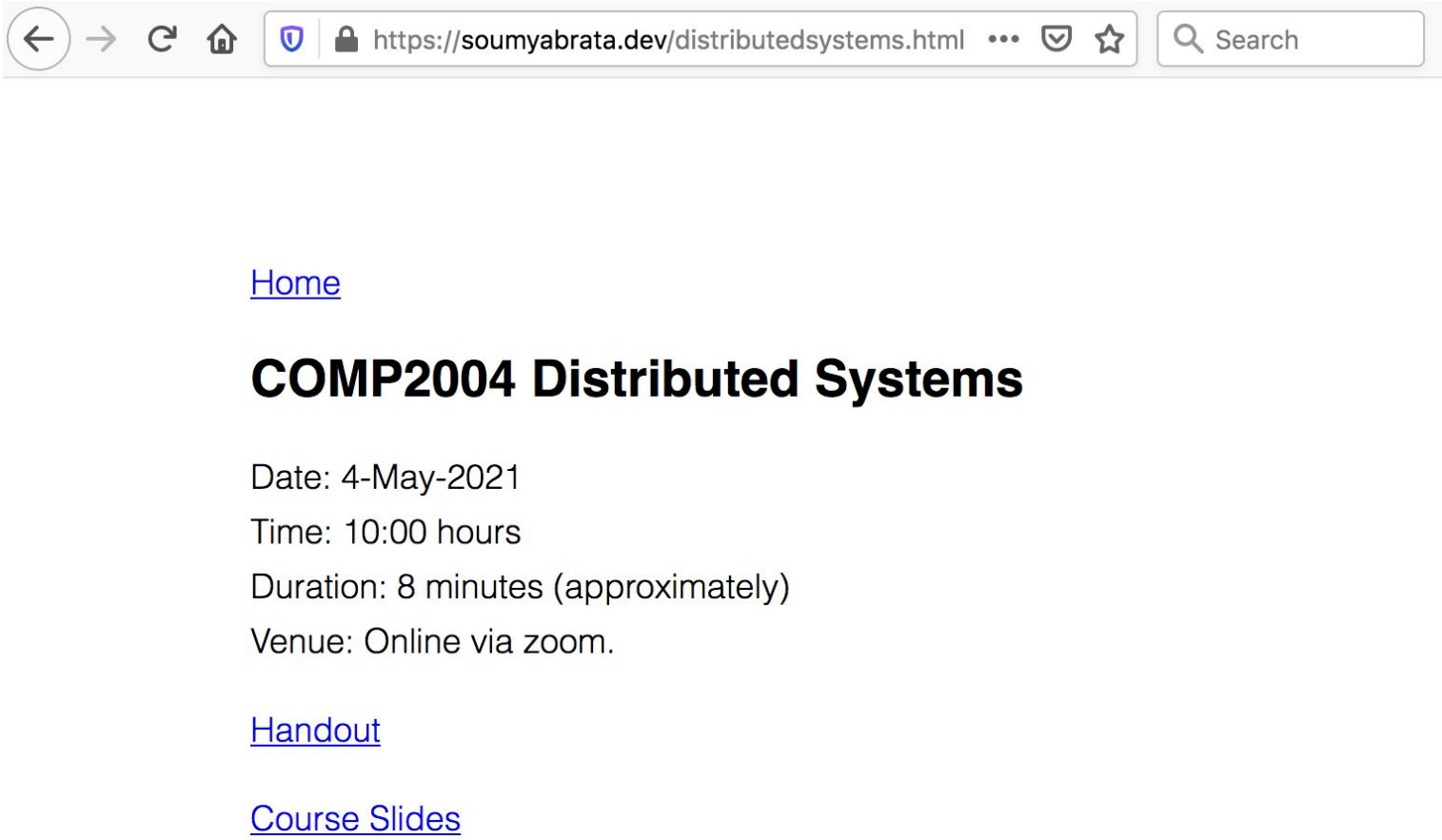
School of Computer Science
University College Dublin
https://soumyabrata.dev/

University College Dublin (UCD)
4-May-2021

# Course Overview



Course website is available here:
https://soumyabrata.dev/distributedsystems.html

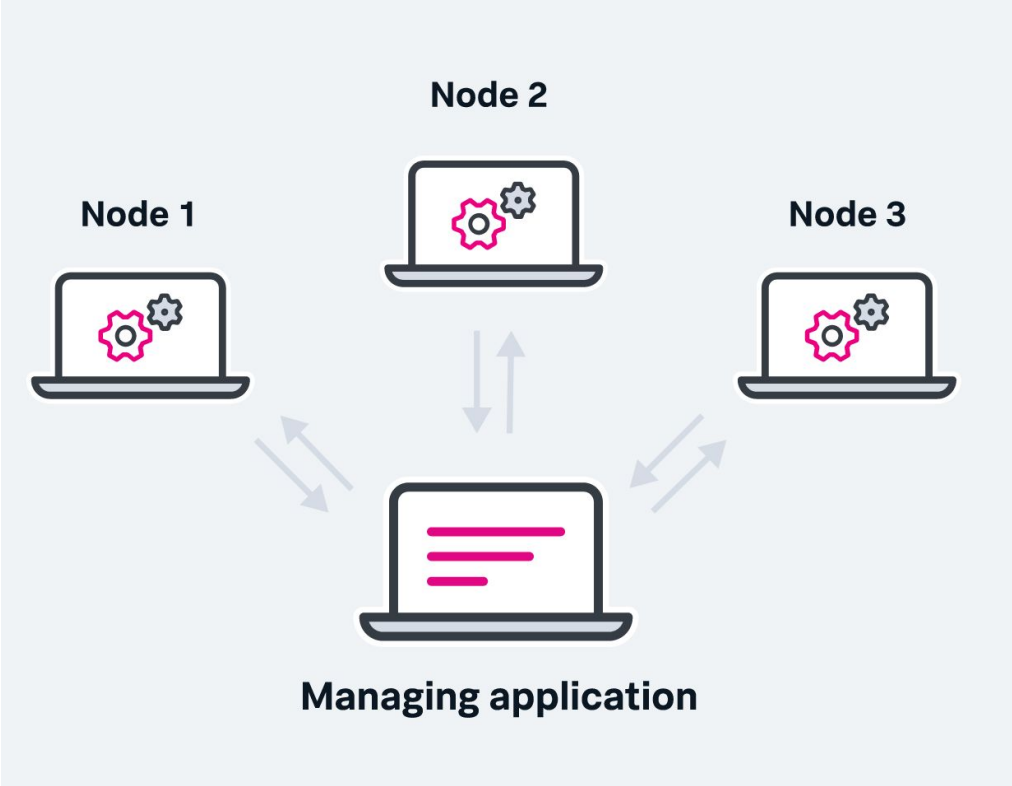


# (1/3) Distributed system

# Distributed System



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



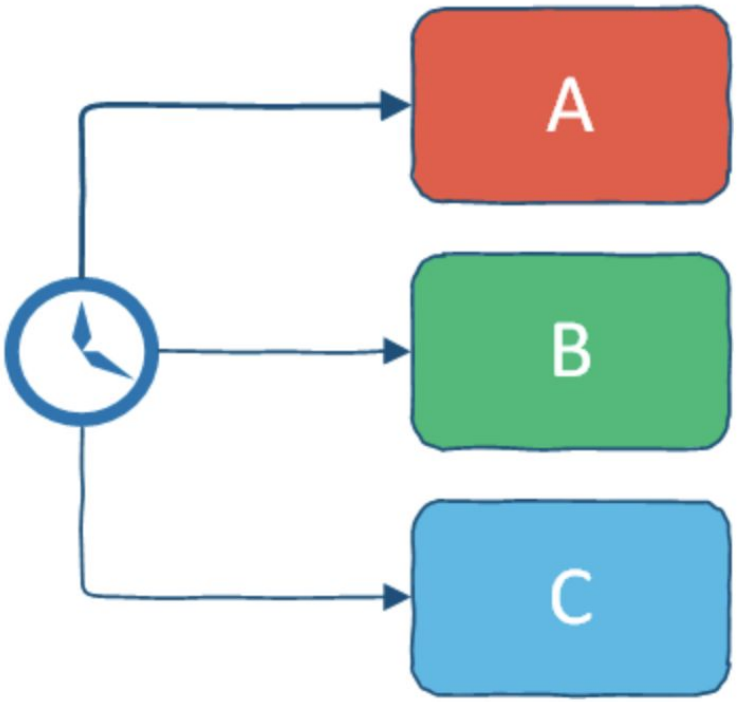

[1]Image archived from https://www.splunk.com/en_us/data-insider/what-are-distributed-systems.html

# What is time?

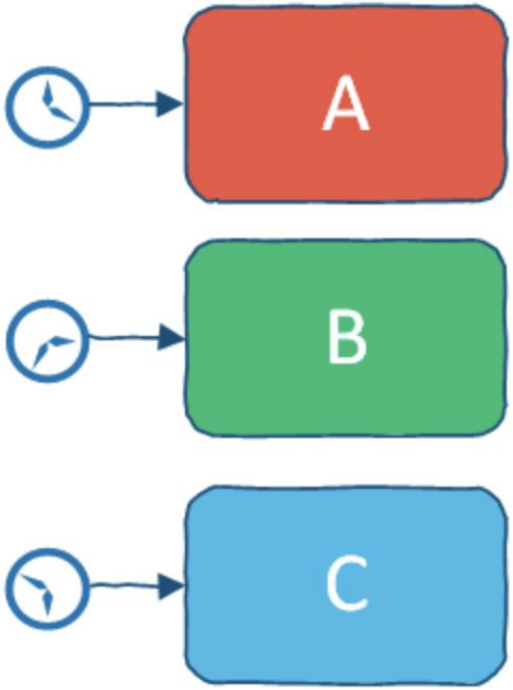


Time is a source of order - it allows us to define the order of operations.

**Global clock of total order**




**Local clock with skew and drift**



# (2/3) Lamport’s logical clock

# Happened-before



Lamport's "happened-before" notation

$a \rightarrow b$ event a happened before event b

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

Transitive:

if $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$

# 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



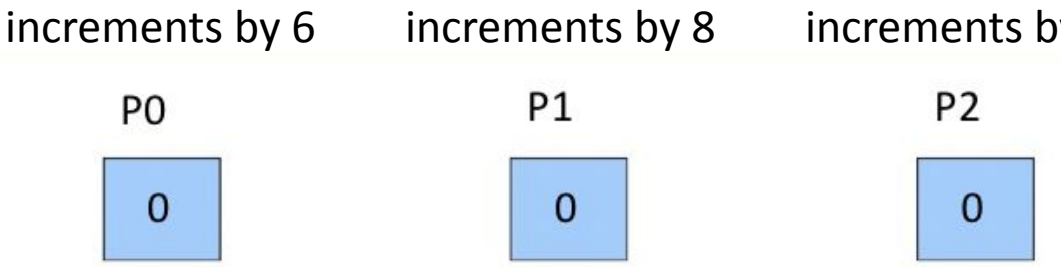
Lamport's clocks are defined as:

1. Events within a process is assigned an unique ID.
2. When sending a message from *A*, we set *clock(A) = clock (A) + 1*, then pass it as a part of the message.
3. On receipt of message in *B*, we set *clock(B) = max[clock(B), clock(A)] + 1*.

# (3/3) Illustration of Lamport's logical clock

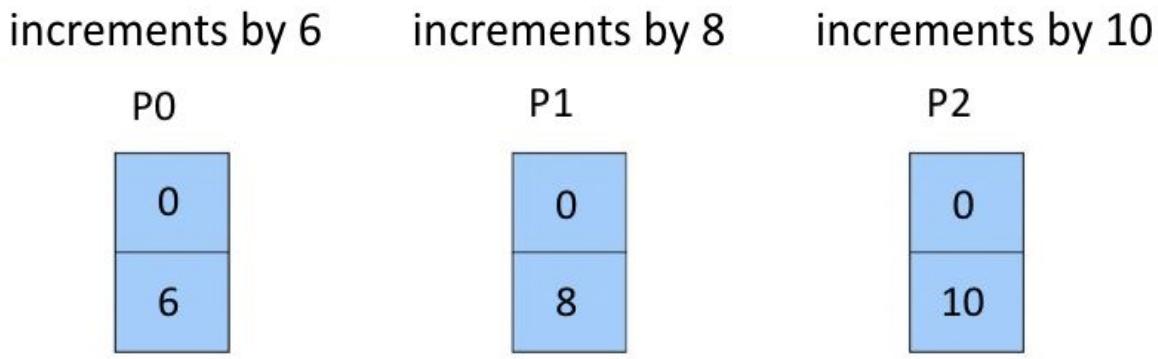
# Example





# Example

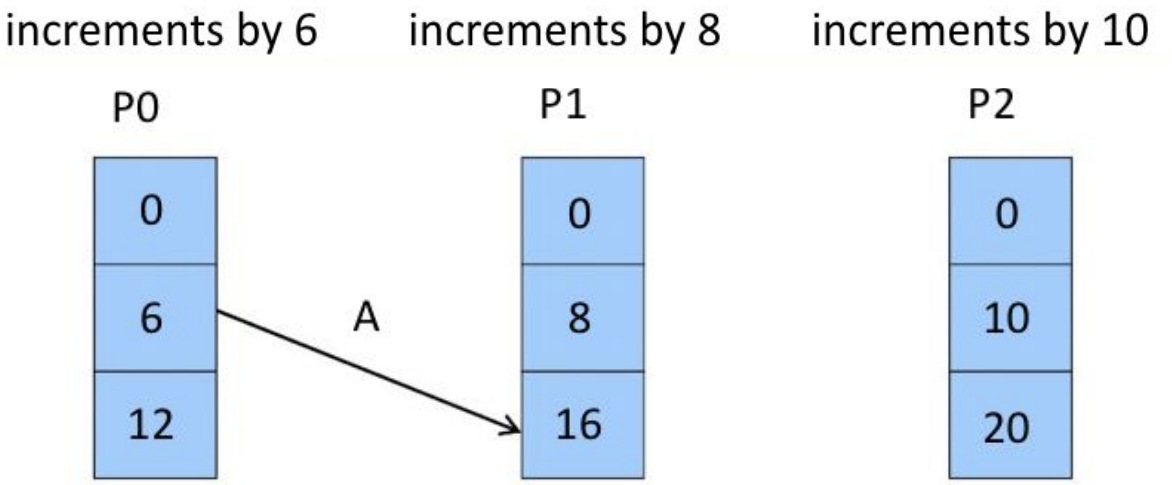





P0 sends message A to P1

# Example

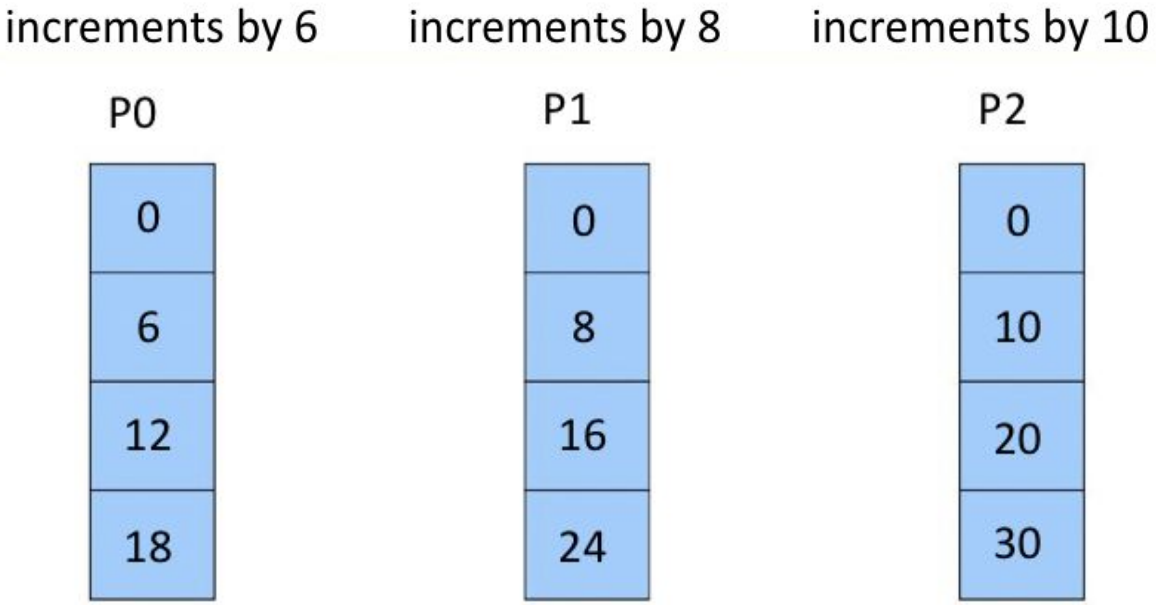





P1 receives message A (everything is OK since 6 < 16)

# Example

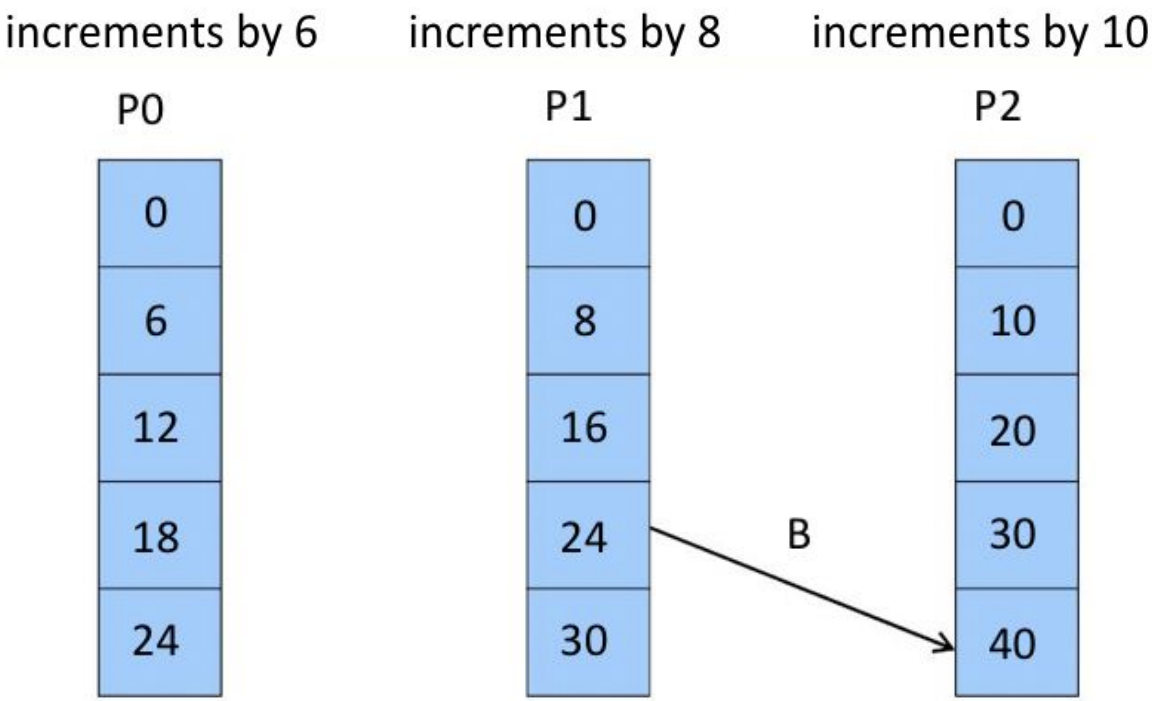





P1 sends message B to P2

# Example

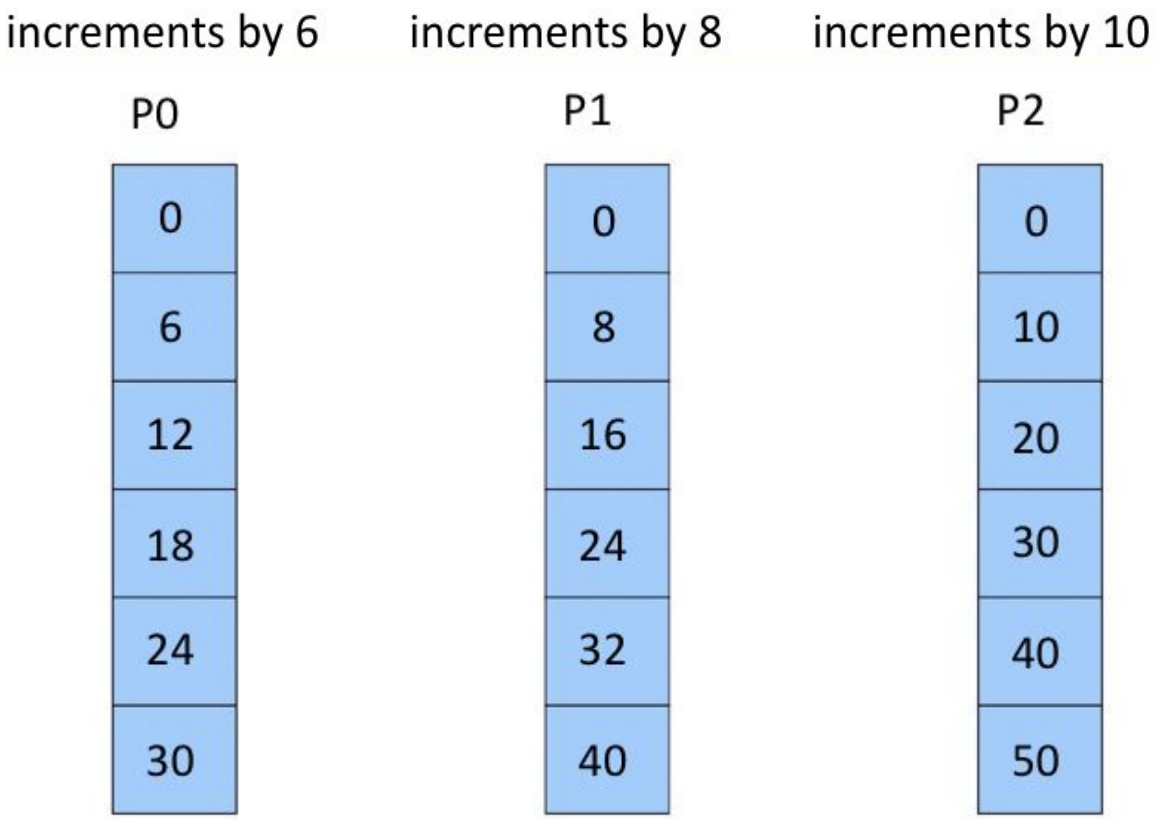





P2 receives message B (everything is OK since 24 < 40)

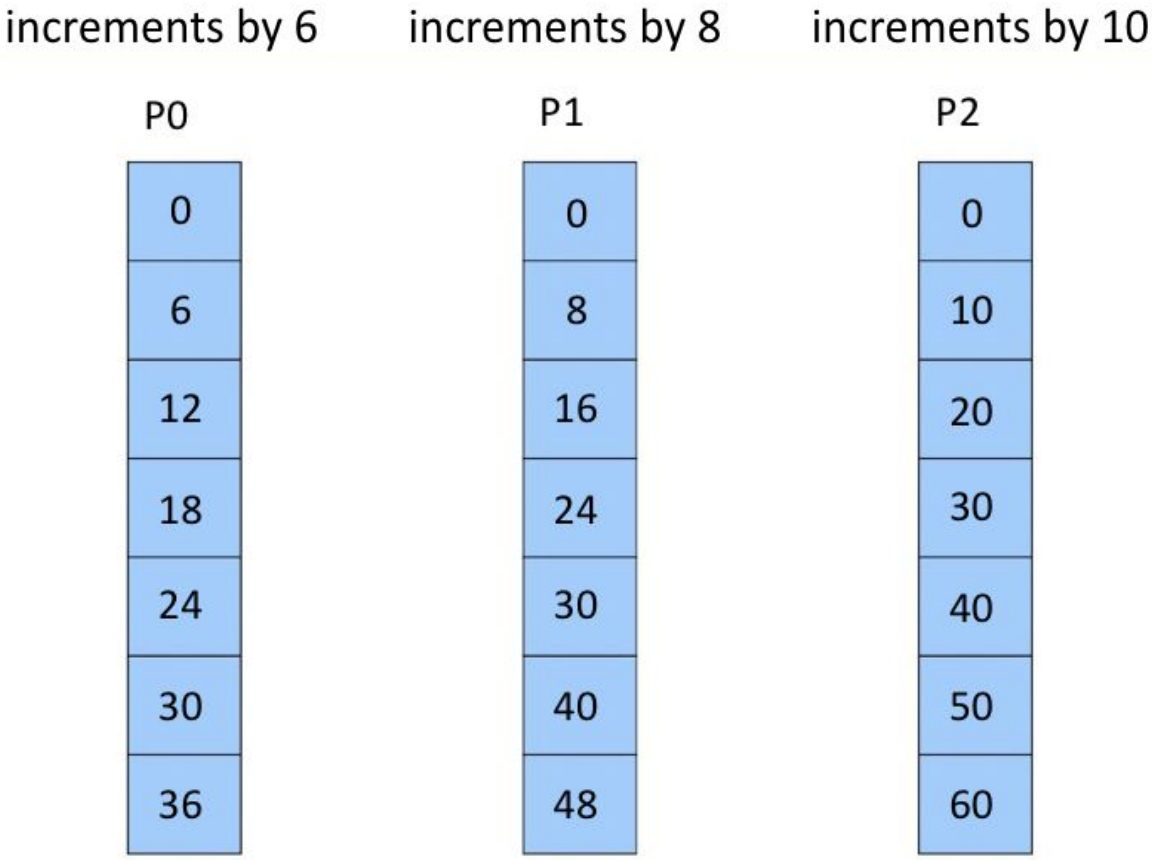
# Example




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

# Example

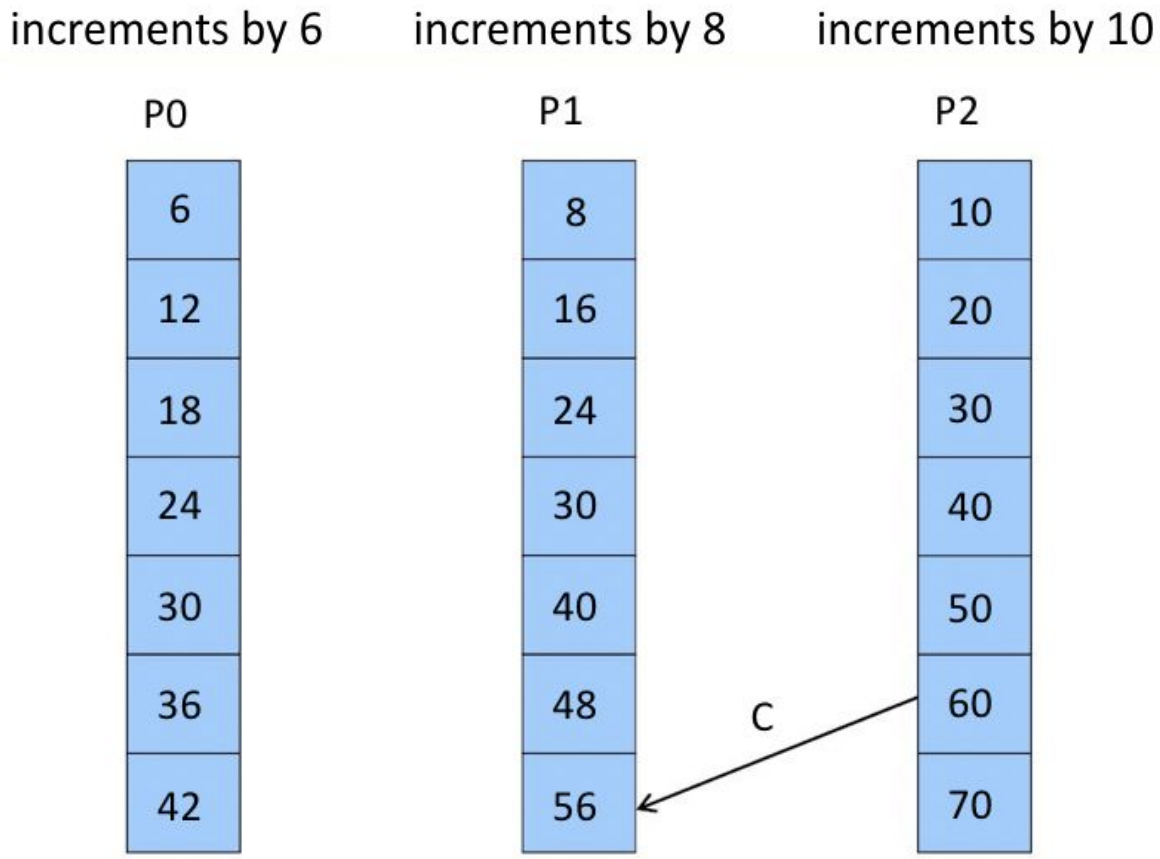





P2 sends message C to P1

# Example

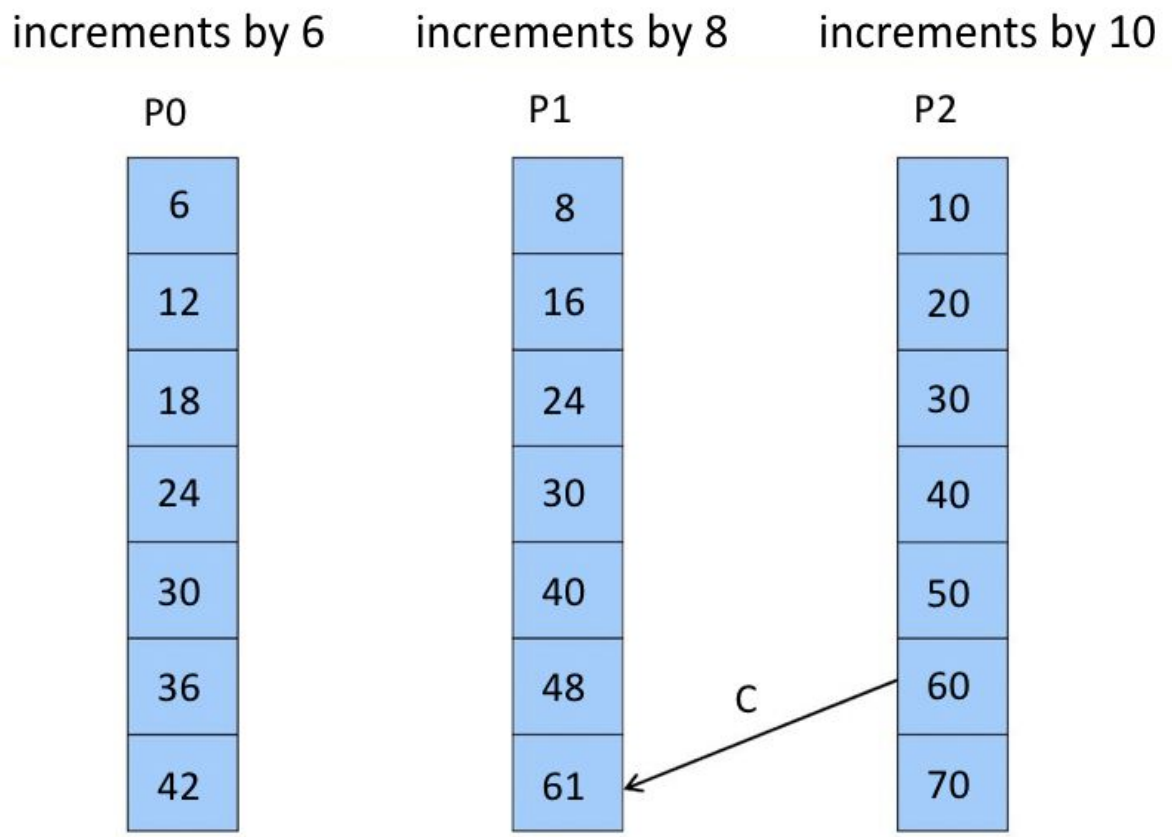





P1 receives message C (Ouch! The message was sent at time 60 but received at time 56)

# Example

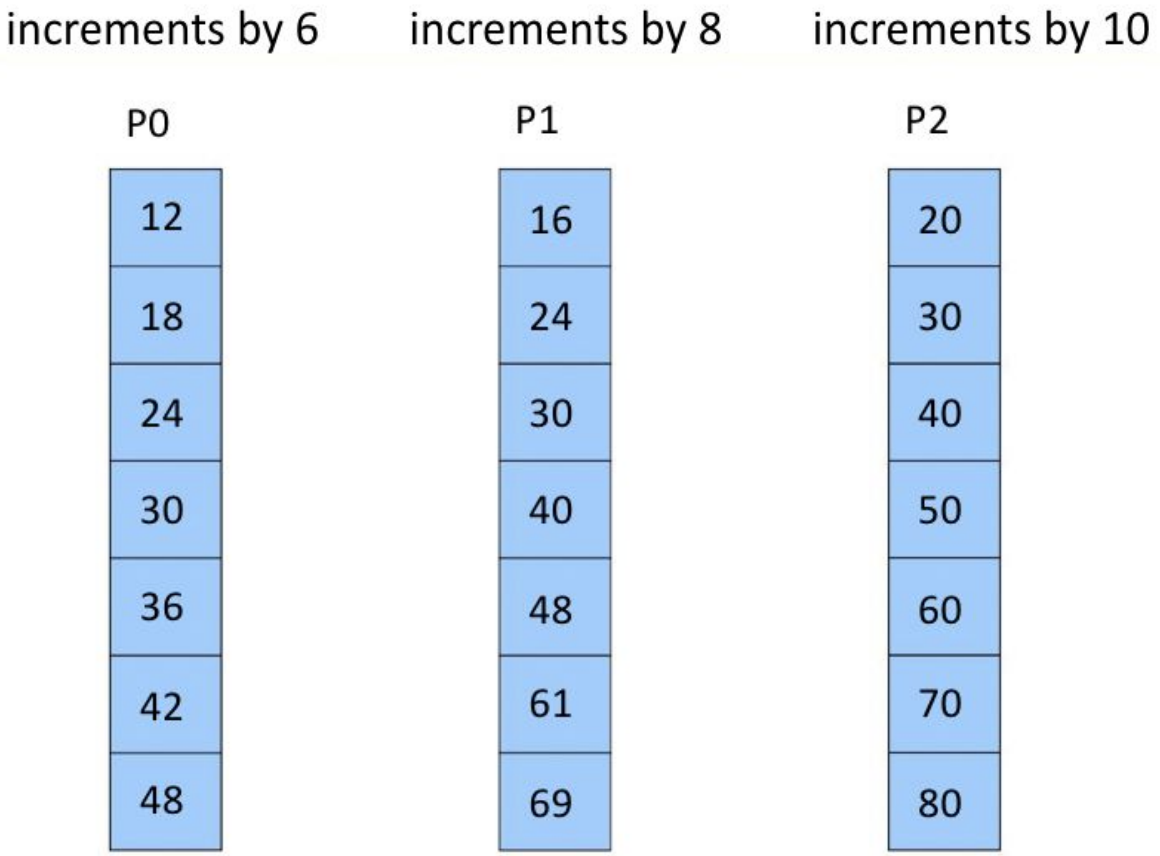





Logical Time at P1 updated to be 1 greater than the time C was sent at.

# Example

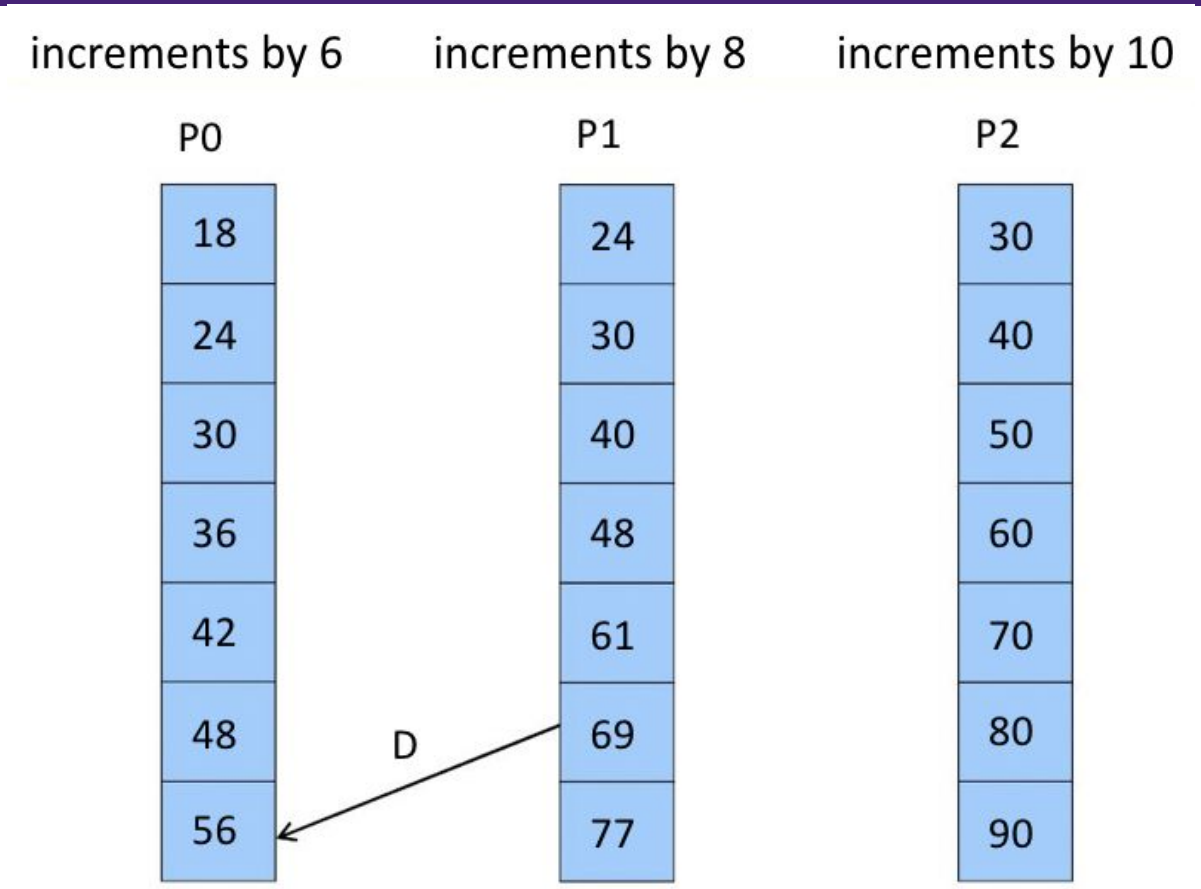





P1 sends message D to P0

# Example

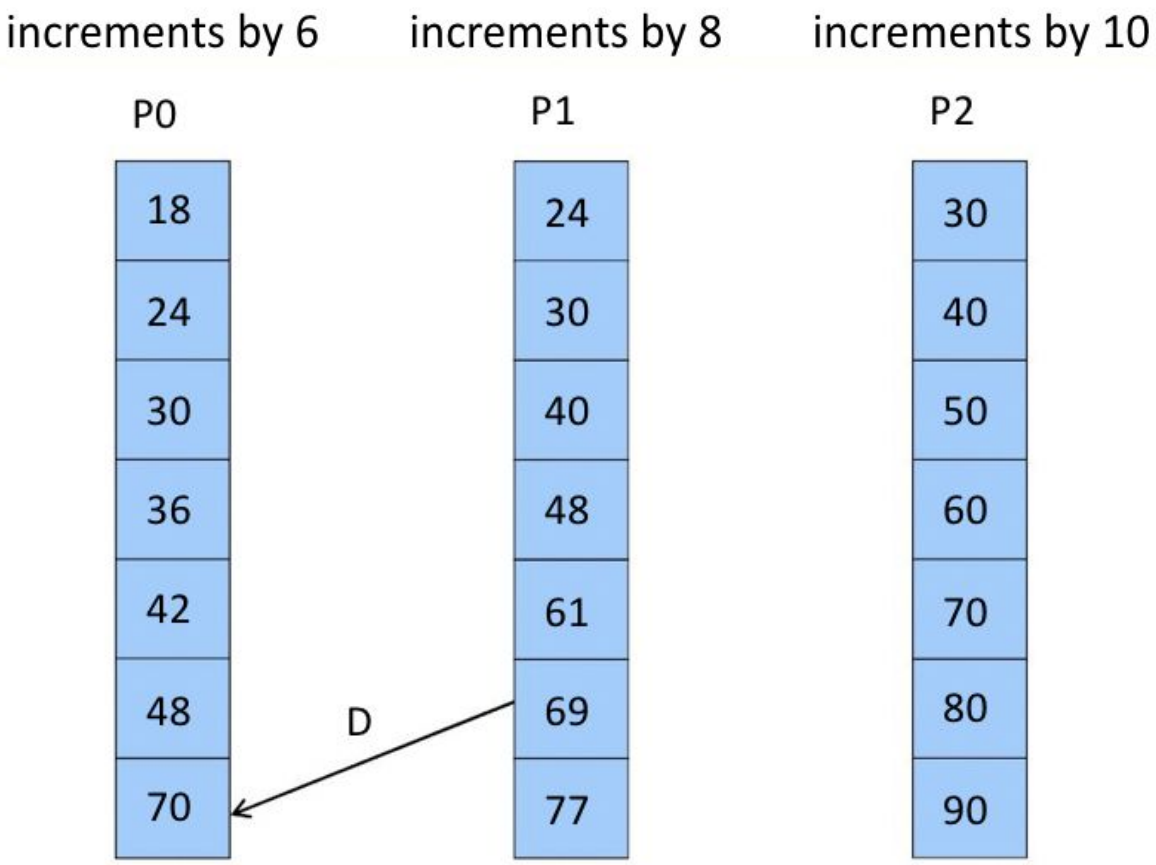





P0 receives message D (Ouch! 56 < 69)

# Example

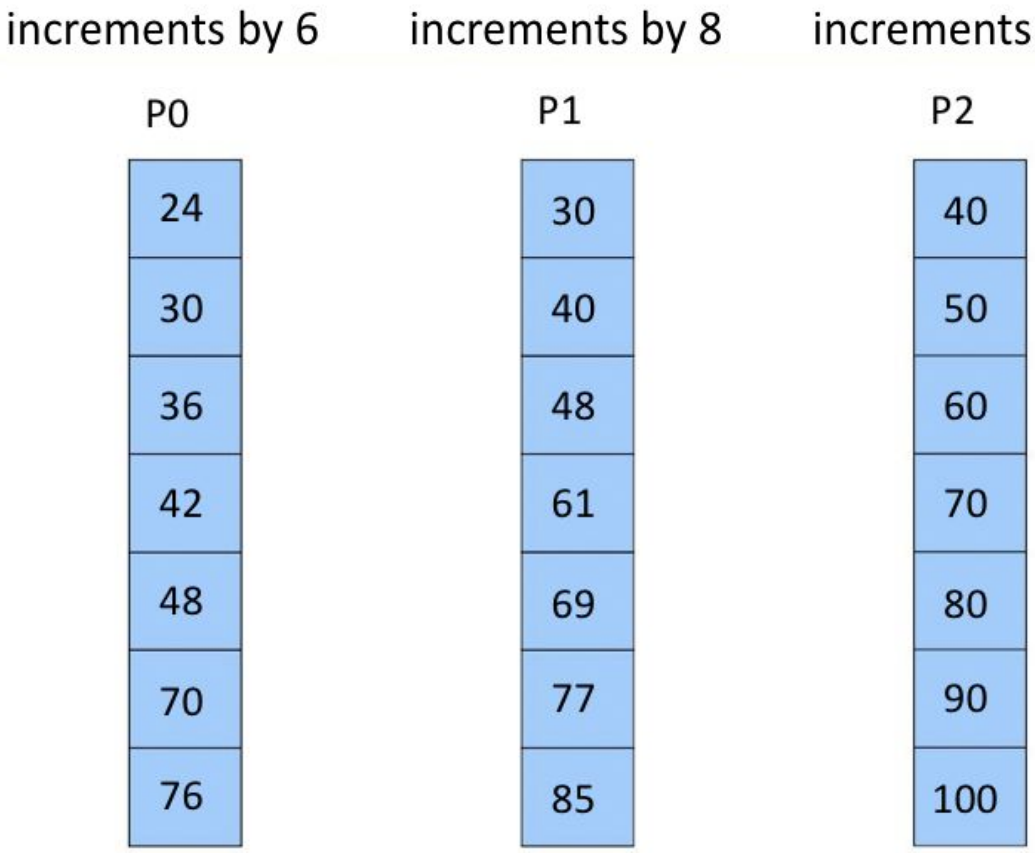





Logical Time at P0 updated

# Example






End of Run

# Limitation of Lamport's algorithm



- Lamport clocks can guarantee that if $a \longrightarrow b$ then *clock(a) < clock(b)*.
  - However it can't guarantee, that if *clock(a) < clock(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