Lamport and Vector Clocks Revisited

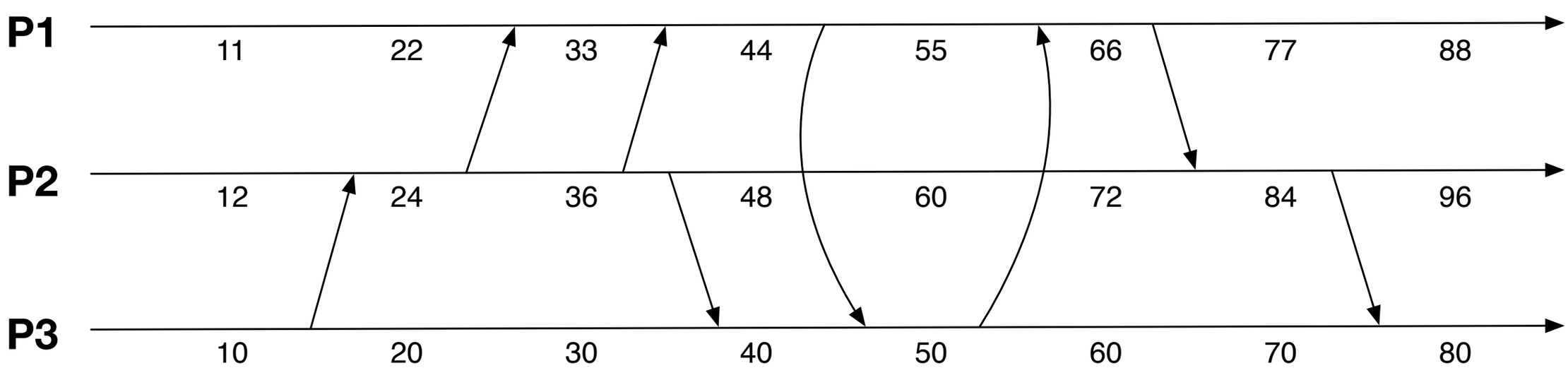
Thomas Schwarz, SJ

Lamport Clocks

Lamport Clock

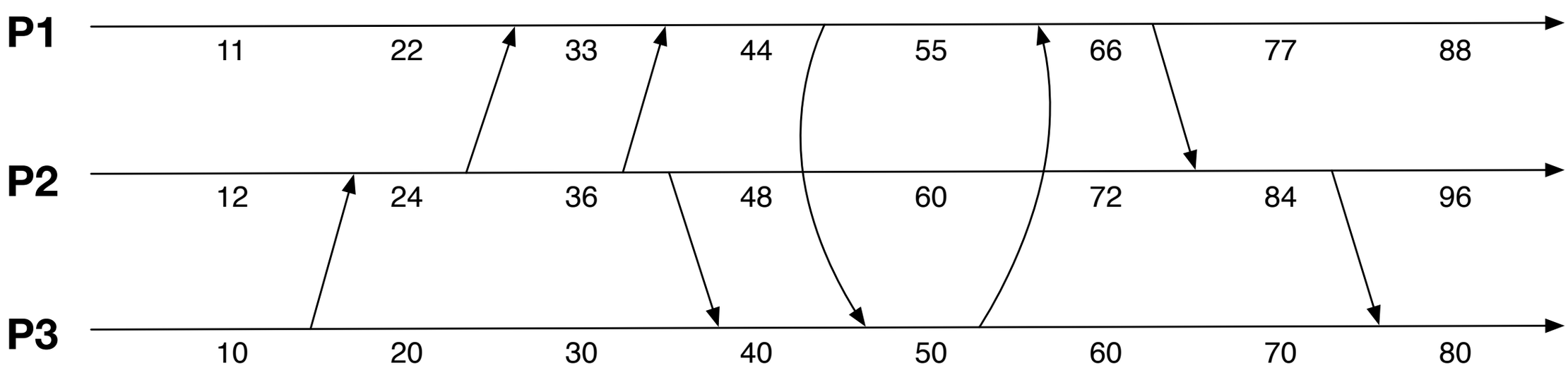
- Logical clock:
  - When a message is received with timestamp τ_m:
    - Set clock c to $\max(c, \tau_m)$
    - Then increment by 1

Lamport Example



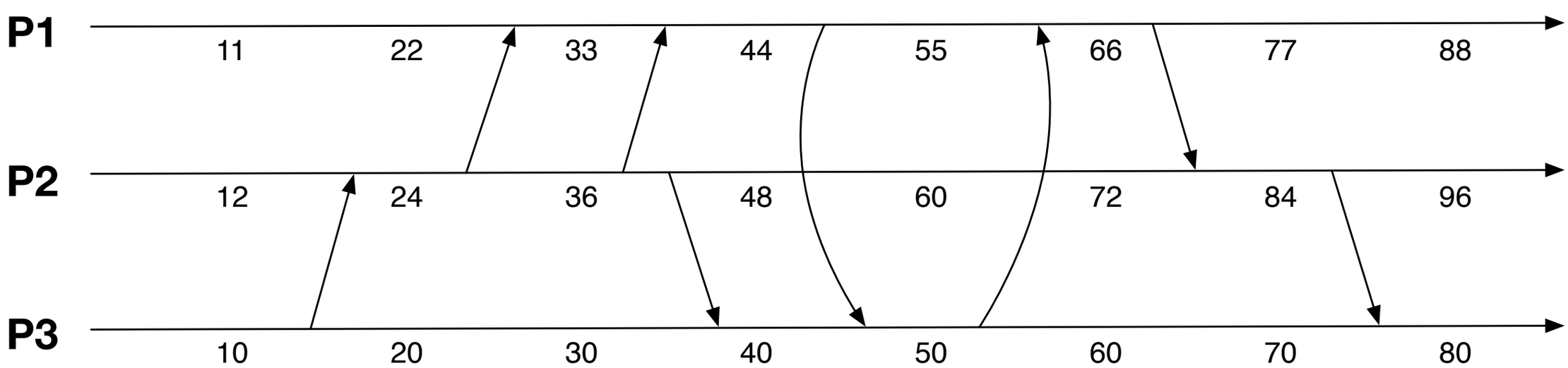
Message 1: P3–> P2: 14 —> 21

Lamport Example



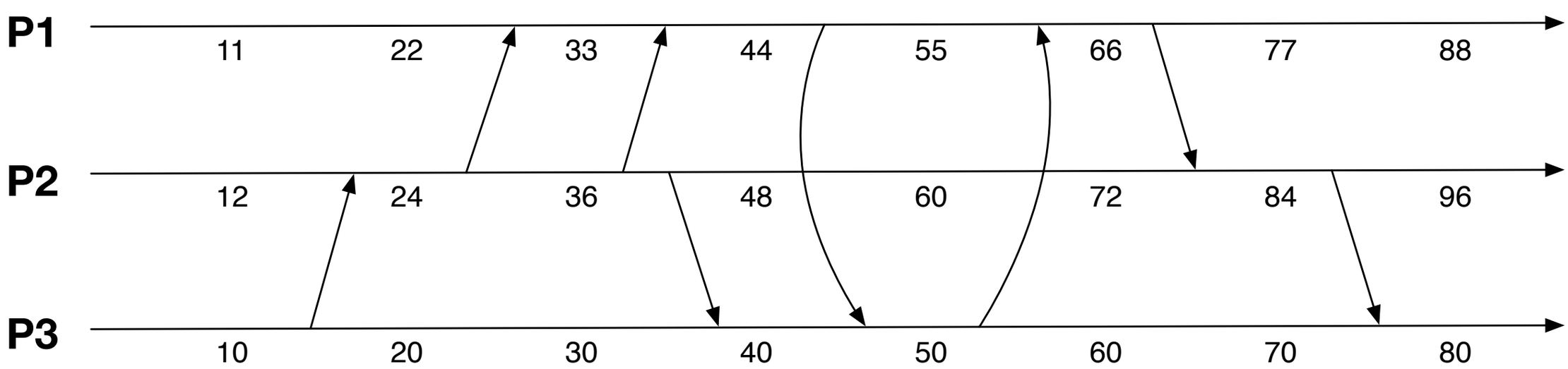
Message 2: P2–> P1: 26 —> 29

Lamport Example



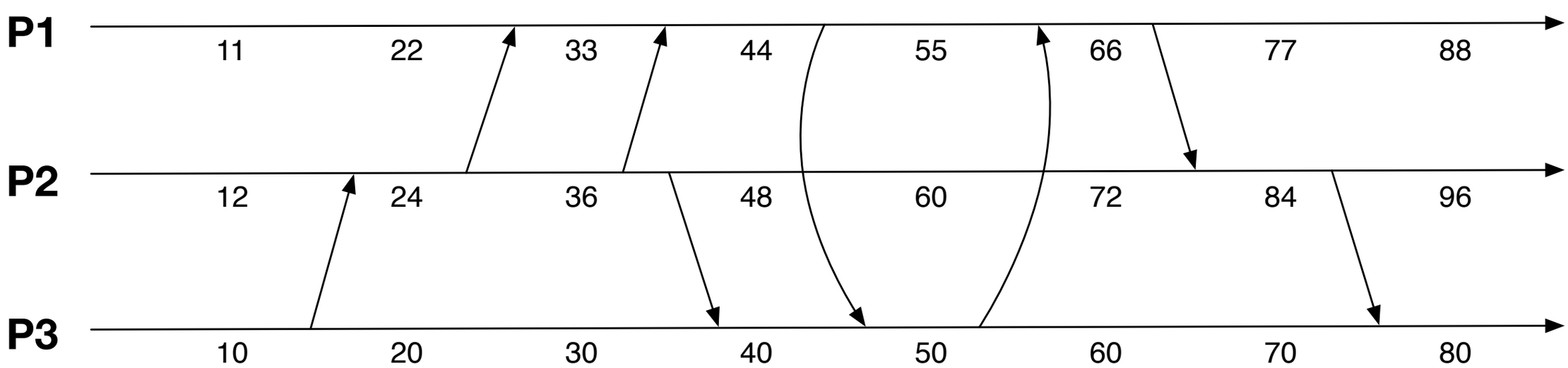
Message 3: P2–> P1: 38 —> 37

Lamport Example



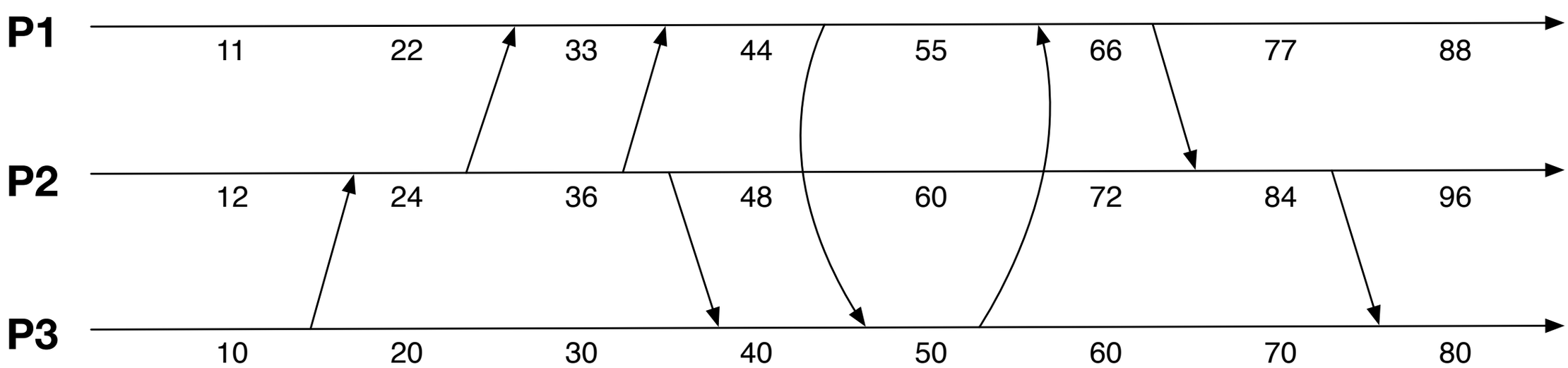
Message 4: P1–> P3: 49 —> 46

Lamport Example



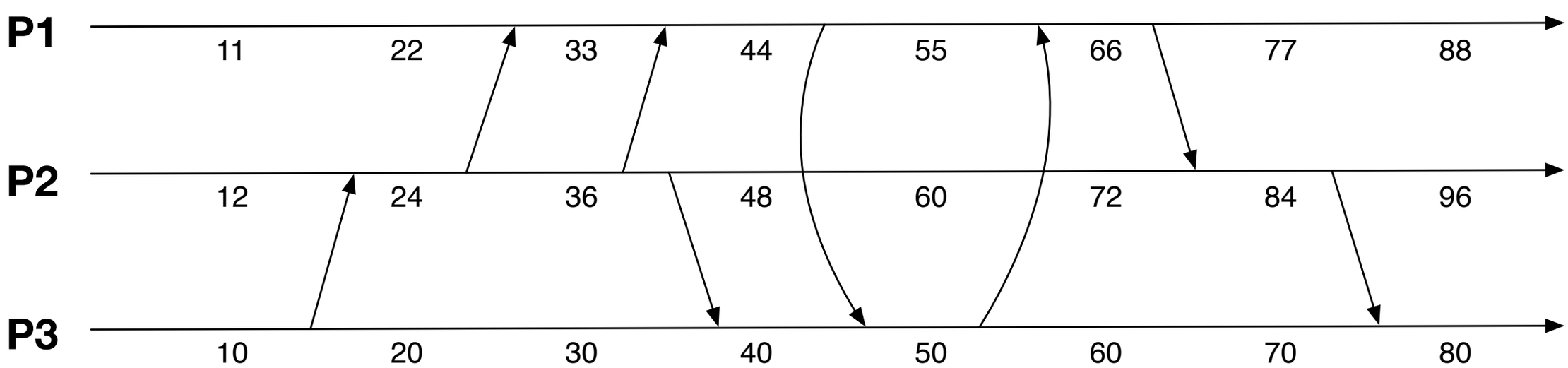
Message 5: P3–> P1: 55 —> 60

Lamport Example



Message 6: P1–> P2: 69 —> 75

Lamport Example



Message 7: P2–> P3: 87 —> 75

Vector Clocks

Vector Clocks

- A Vector Clock (VC) is a vector of integers,
  - One entry for each process in the entire distributed system
    - E.g. $P_1 : (0,0,0);\ P_2 : (0,0,0);\ P_3 : (0,0,0)$
- For each local event on process i, increment local entry
  - E.g. Local event at P_1:
    - $P_1 : (1,0,0)\ \ P_2 : (0,0,0)\ \ \ P_3 : (0,0,0)$

Vector Clocks

- A Vector Clock (VC) is a vector of integers
  - Sending a message P_i to P_j
    - At P_i : Increment local clock
      - Tag message with the clock at P_i
      - E.g.: Sending μ from P_1 to P_2:
        - $P_1 : (2,0,0)$
        - Message tag is $\mu : (2,0,0)$

Vector Clocks

- A Vector Clock (VC) is a vector of integers
  - Sending a message P_i to P_j
    - At P_j : Clock Vector becomes
      - $\text{clock}_j = [\max(\text{clock}_j[\nu], \mu[\nu])$ for $\nu \in \{1,\ldots,N\}]$
      - Then increment local clock:
        - $\text{clock}_j[j]+=1$

Vector Clocks and Multicasting

- The sender of a multicast message:
  - Increments its part of its vector clock
  - Tags the multicast message with its new vector clock

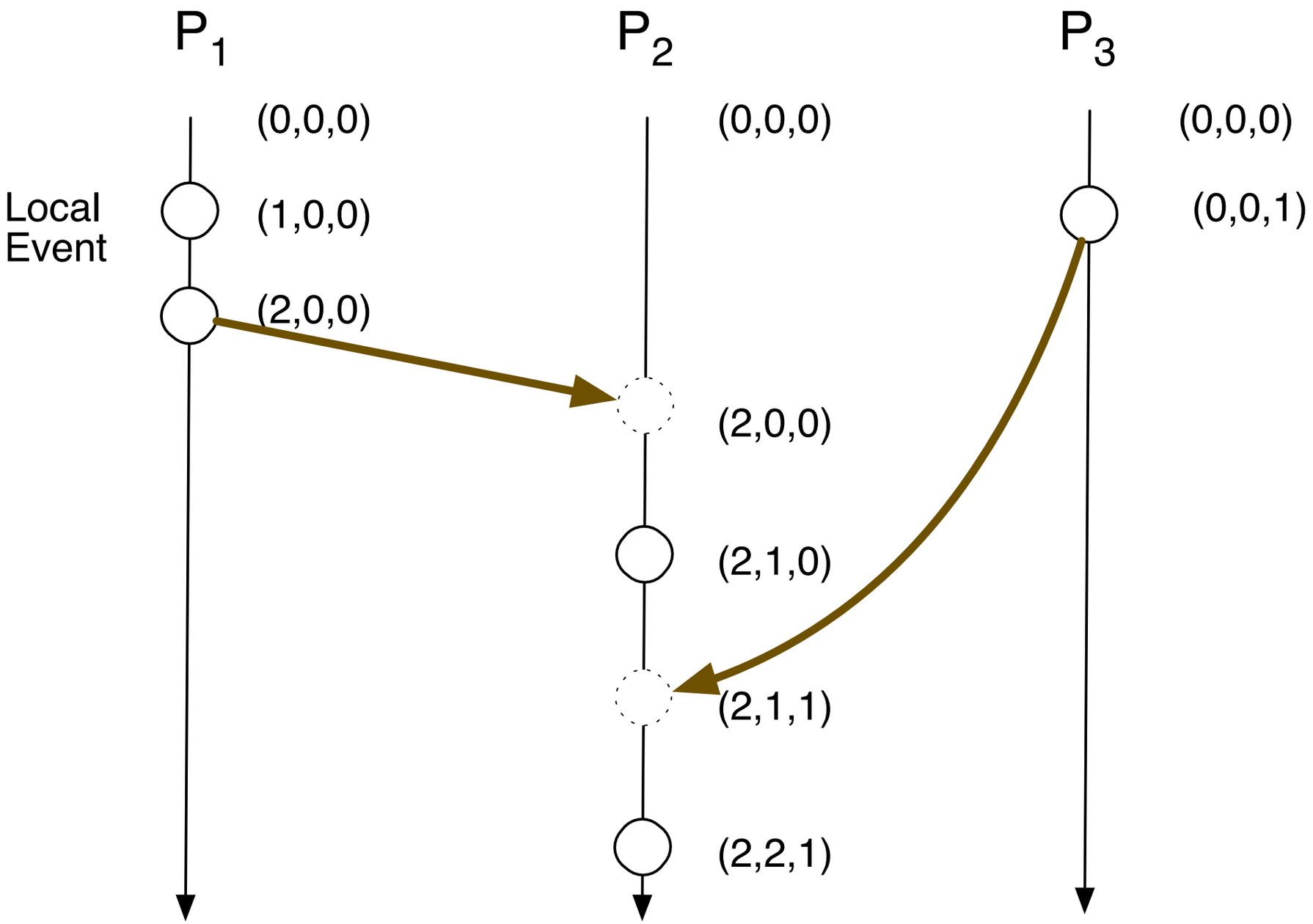
Vector Clocks and Multicasting

- Only use the vector-clock capture sending and receiving messages
- The receiver P_j of a multicast message from P_i:
  - Compares the tag μ with its vector clock
  - Delays message until the following is true
    1. $\mu[i] = \text{clock}_j[i] + 1$
       - This is the next message we are expecting
    2. $\mu[k] \leq \text{clock}_j[k]$ for all $k \neq j$

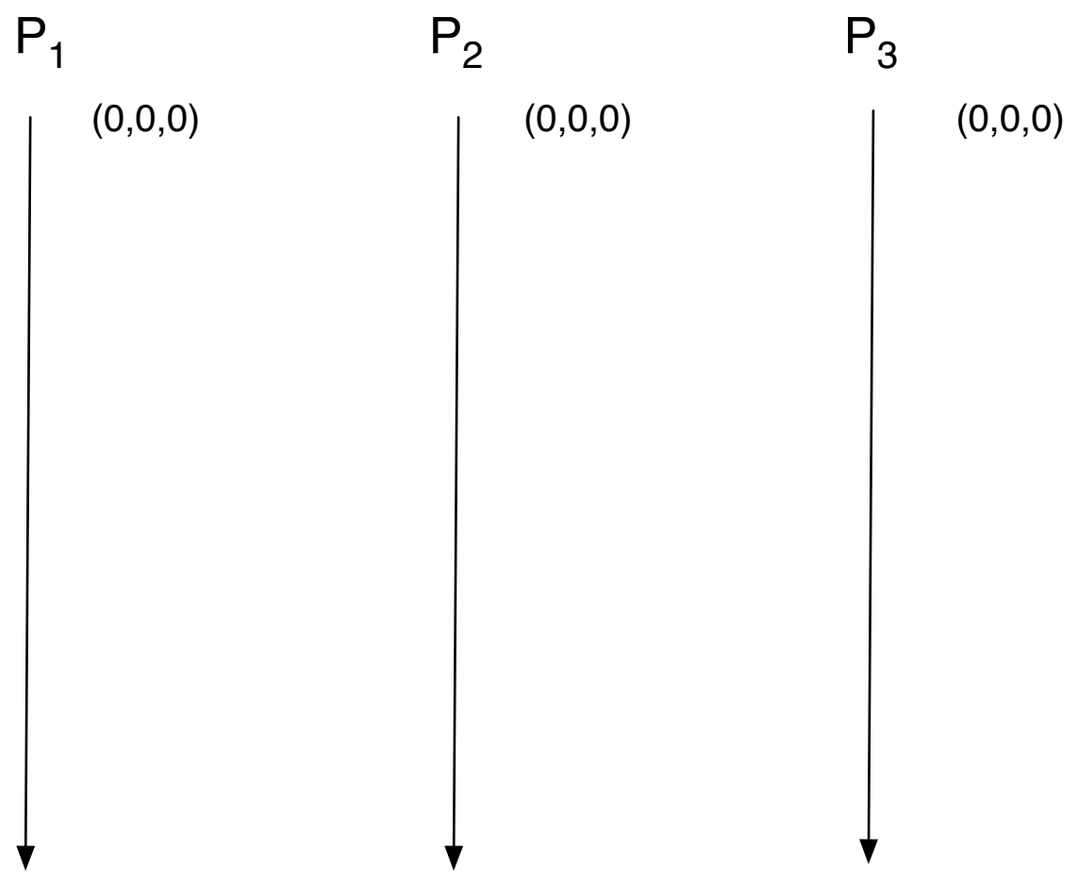
Vector Clocks and Multicasting

- This guarantees that all multicasts are delivered in the same order at all processes.

Example

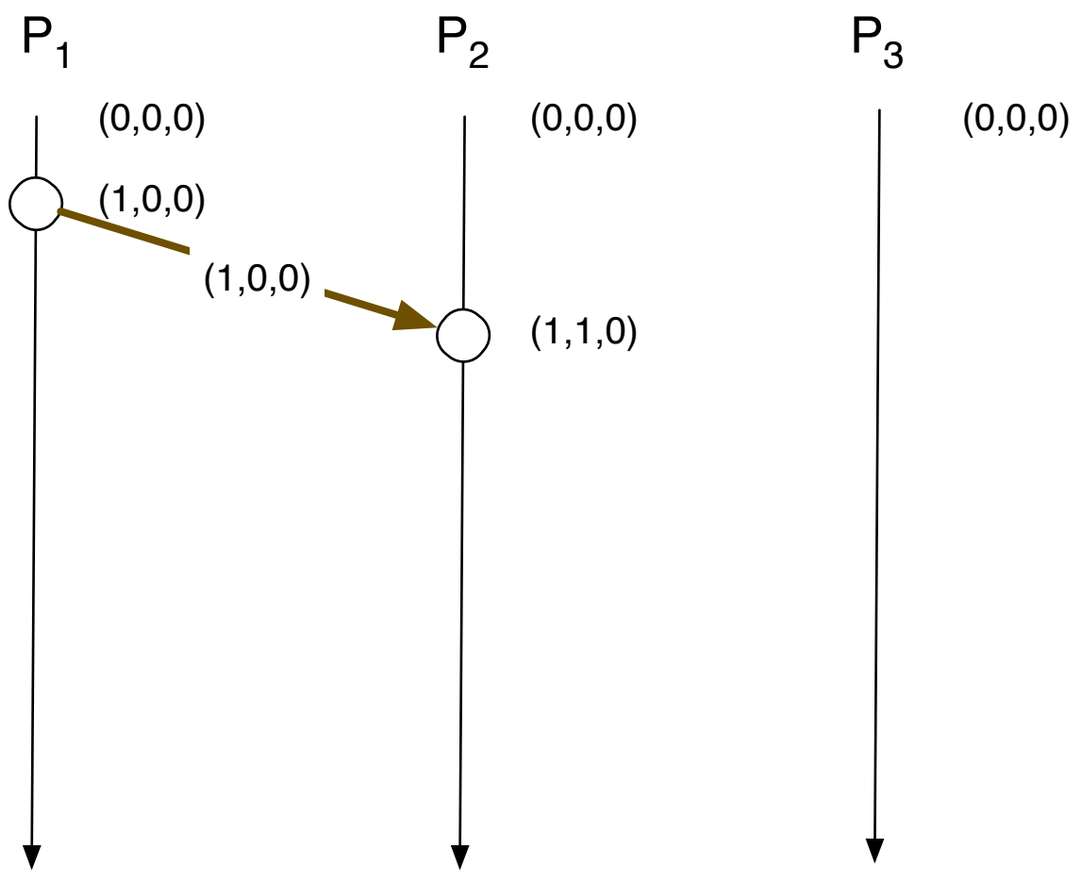


Another Example



P_1 sends to P_2

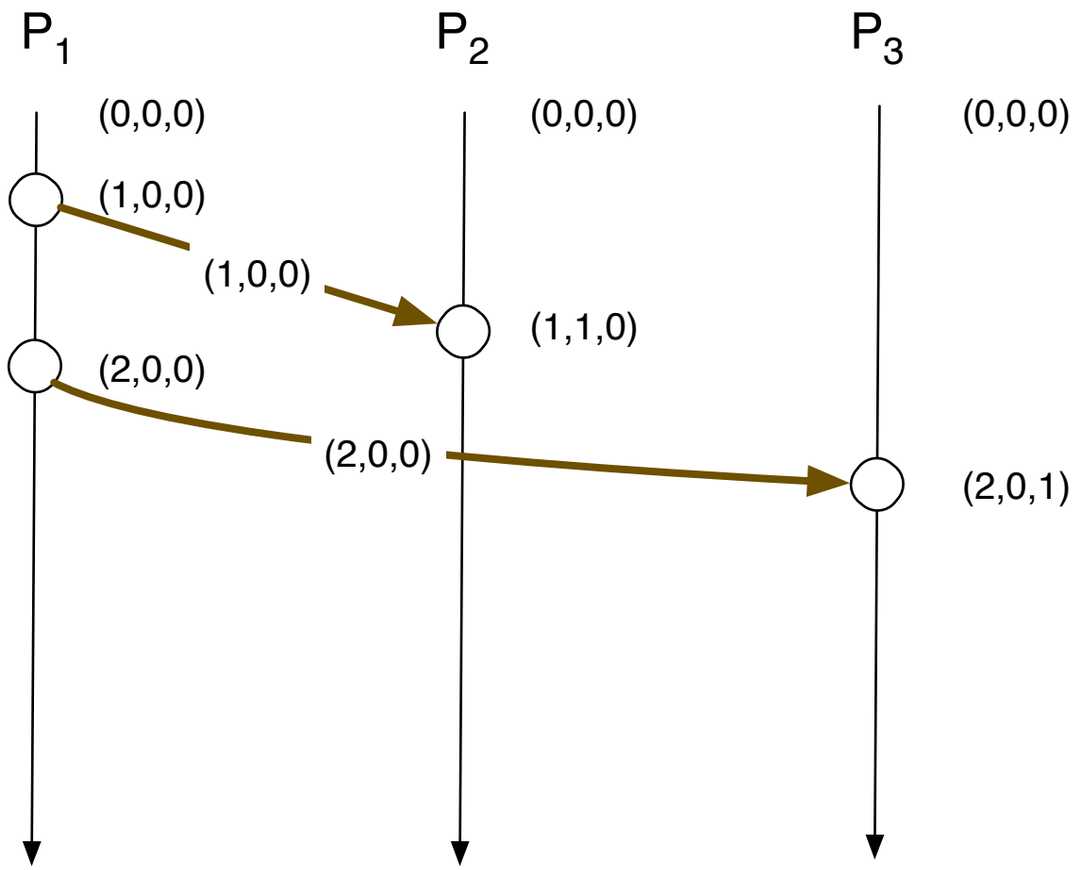
Another Example



P_1 **sent to** P_2

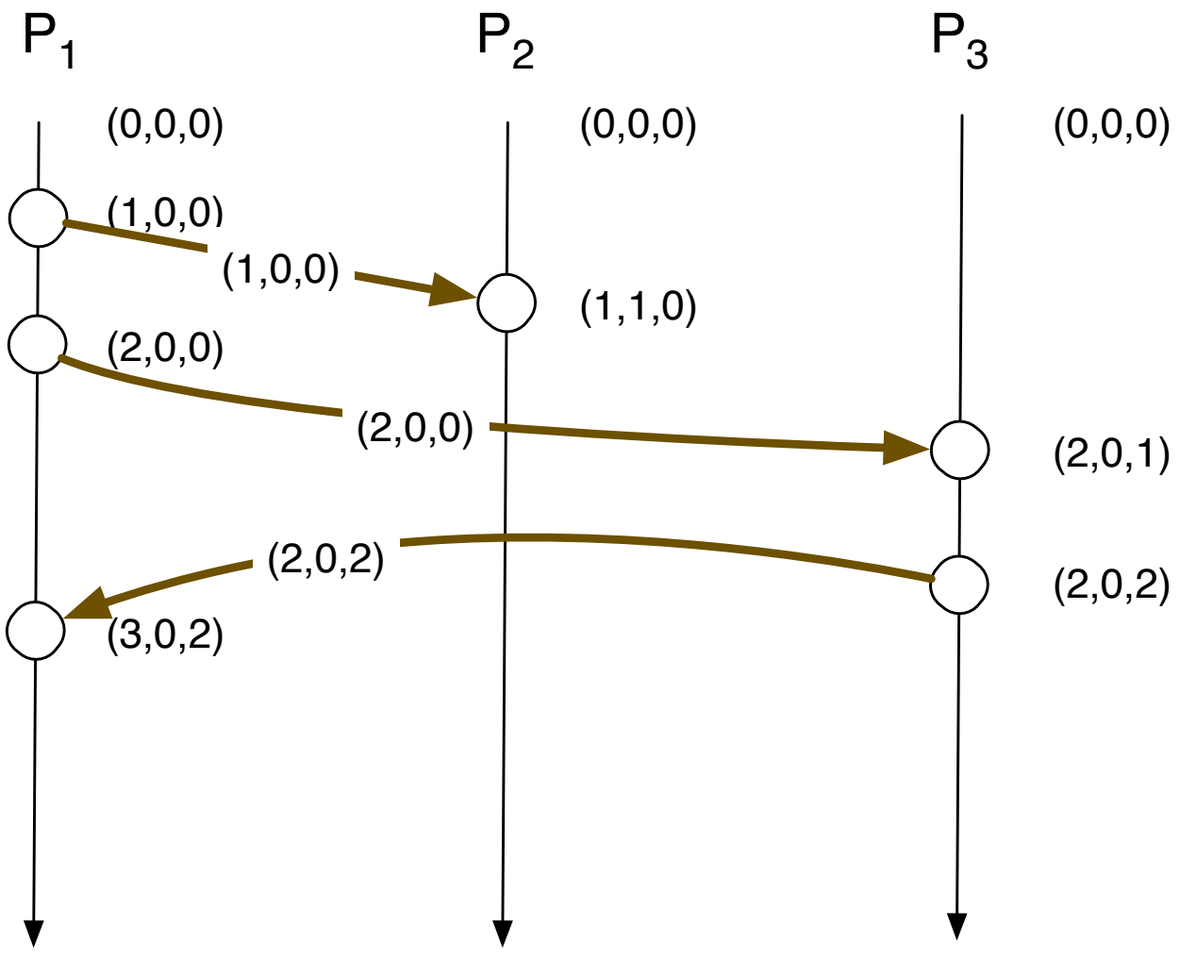
Now: P_1 **sends to** P_3

Example



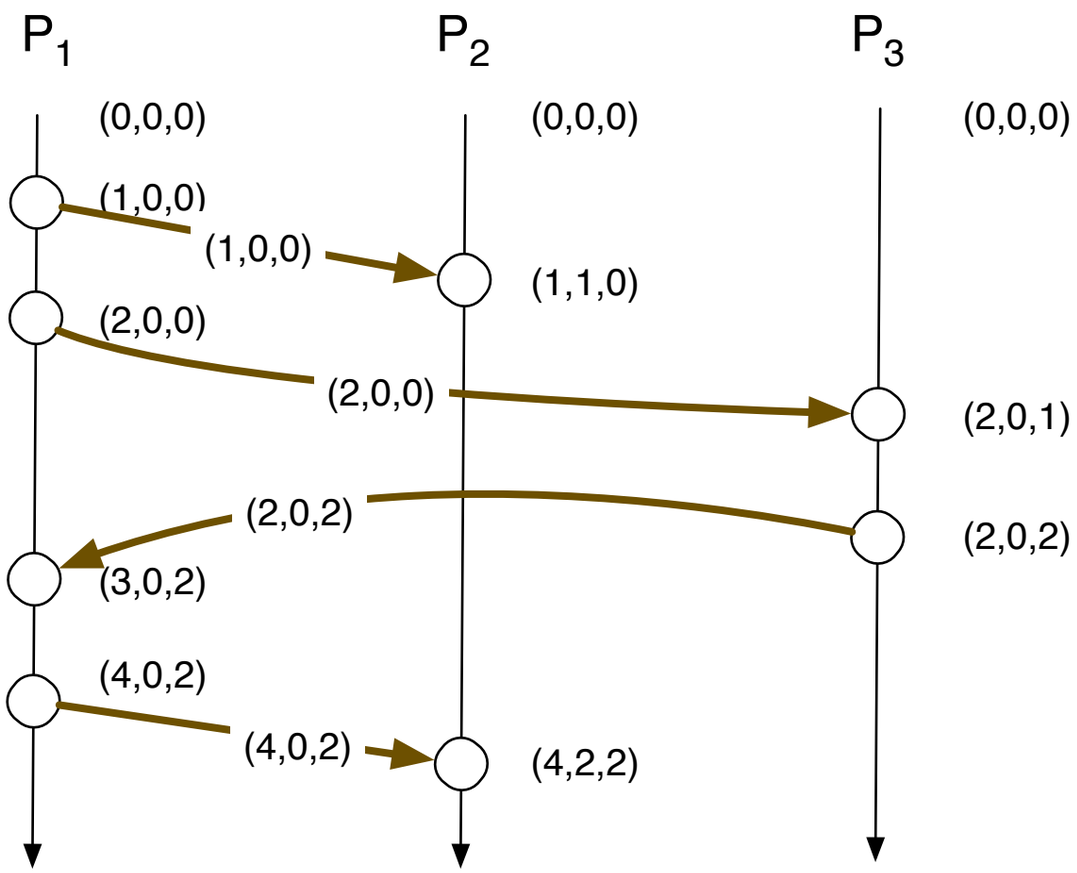
P_3 **sends to** P_0

Example



Now P_1 sends to P_2

Example

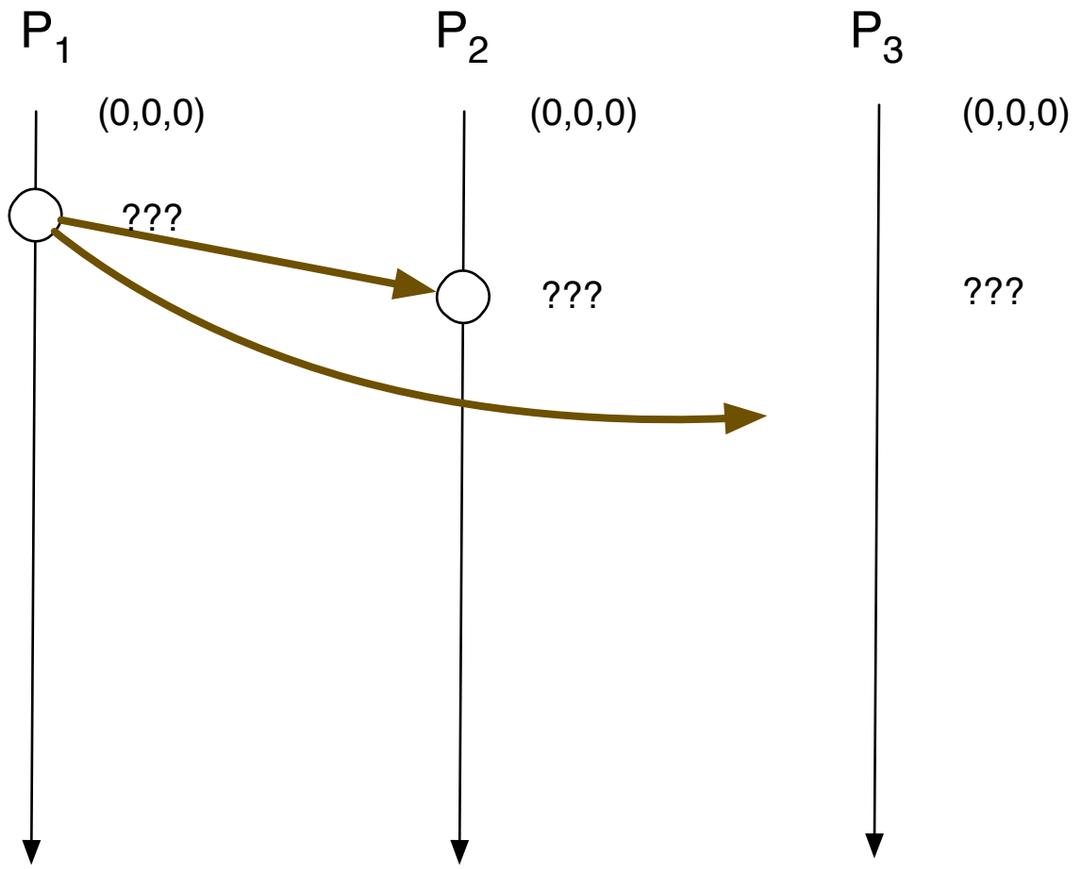

P1
P2
P3
(0,0,0)
(0,0,0)
(0,0,0)
(1,0,0)
(1,0,0)
(1,1,0)
(2,0,0)
(2,0,0)
(2,0,1)
(2,0,2)
(2,0,2)
(3,0,2)
(4,0,2)
(4,0,2)
(4,2,2)

Quiz

- Multicast messages:
  - At the sender, we only increment once!

Quiz

- Question 1:
- P_1 multicasts, P_2 receives the multicast, P_3 does not



Quiz

- Question 2, 3, 4:
  - P_2 multicasts, and the multicast message is received at both P_1 and P_3