CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

TIME AND ORDERING

Lecture E

VECTOR CLOCKS

Vector Timestamps

- Used in key-value stores like Riak
- Each process uses a vector of integer clocks
- Suppose there are N processes in the group 1…N
- Each vector has N elements
- Process *i maintains vector* $\mathbf{V_i[1…N]}$
- *j*th element of vector clock at process *i*, $V_i[j]$, is *i*'s knowledge of latest events at process *j*

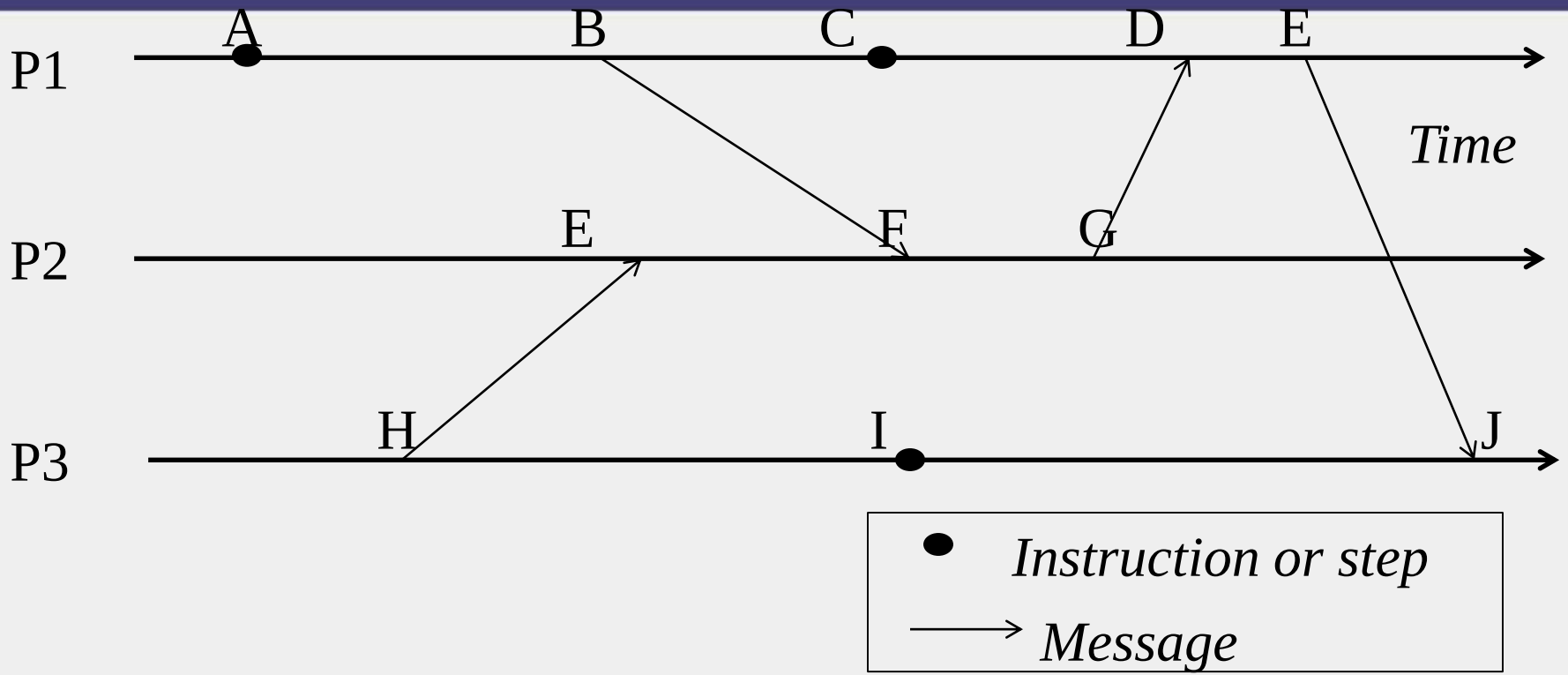
Assigning Vector Timestamps

- Incrementing vector clocks
  1. On an instruction or send event at process *i*, it increments only its *i*th element of its vector clock
  2. Each message carries the send-event's vector timestamp $V_{message}[1 \ldots N]$
  3. On receiving a message at process *i*:

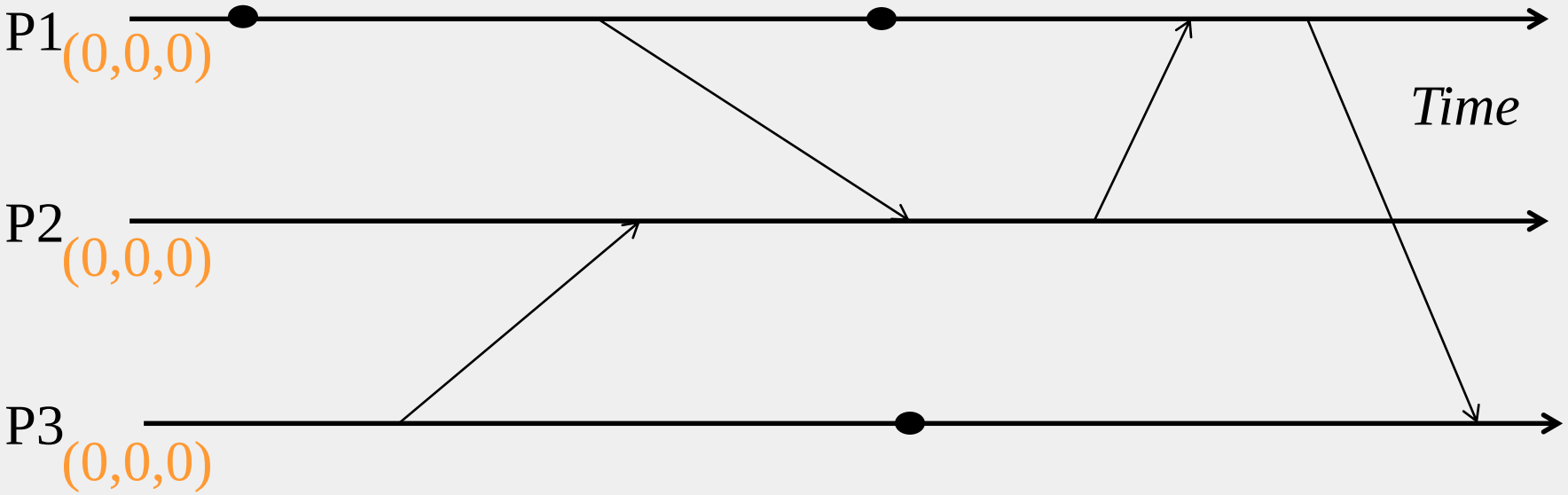
$V_i[i] = V_i[i] + 1$

$V_i[j] = \max(V_{message}[j], V_i[j])$ for $j \neq i$

Example

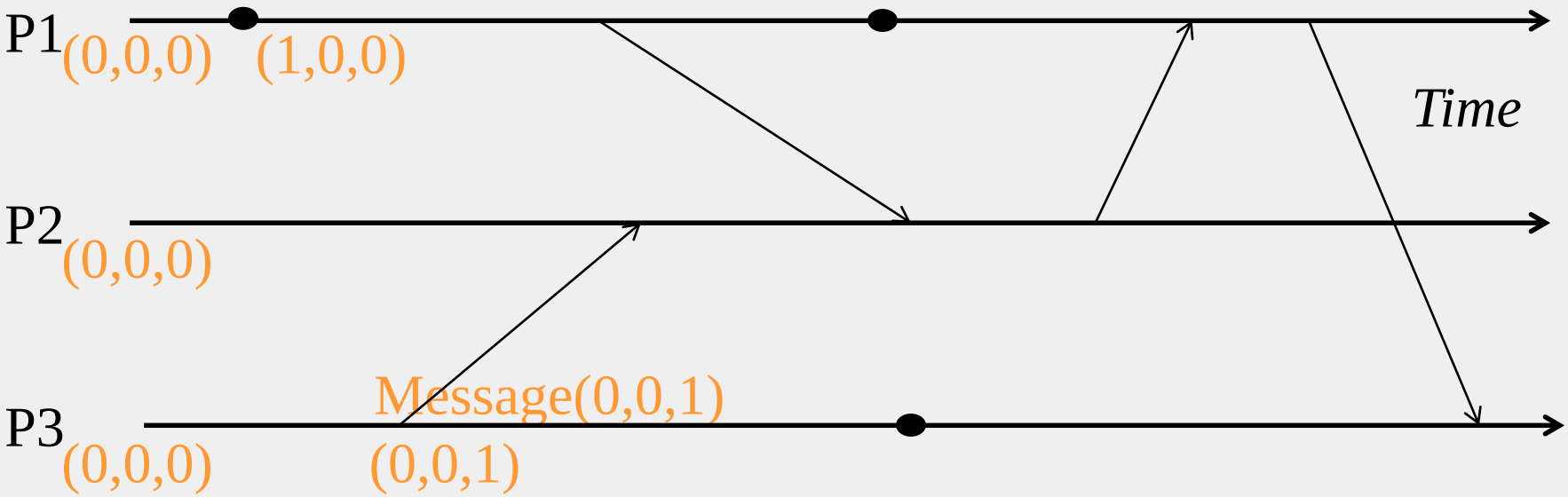


Vector Timestamps

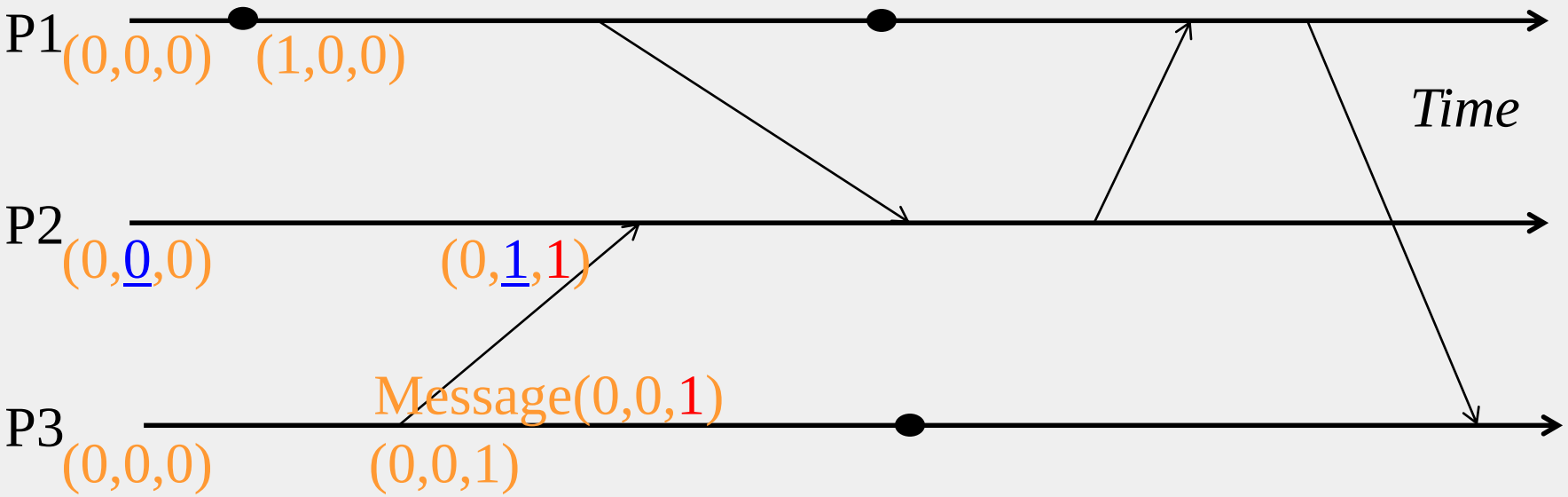


Initial counters (clocks)

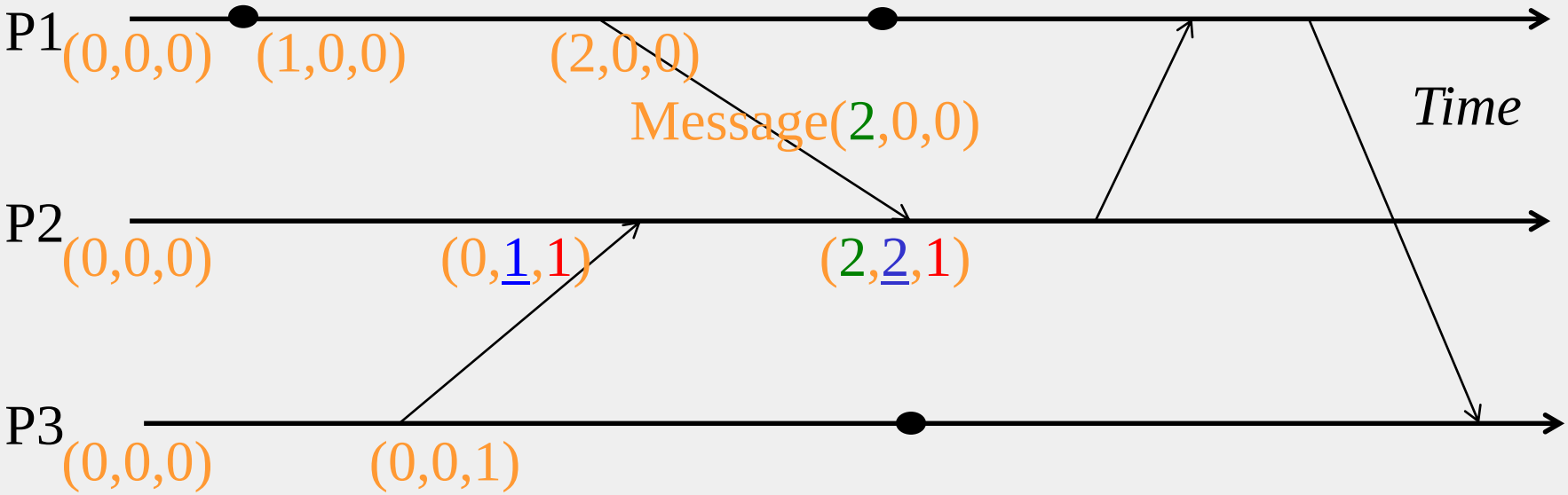
Vector Timestamps



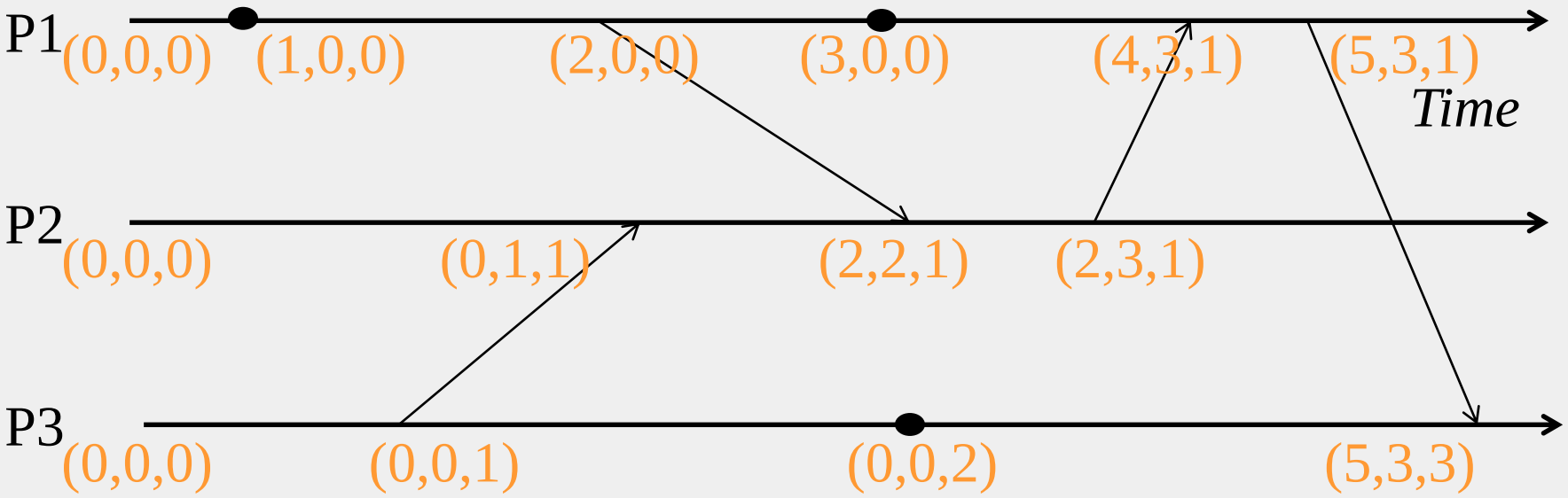
Vector Timestamps



Vector Timestamps



Vector Timestamps



Causally-Related …

- $VT_1 = VT_2$,

  iff (if and only if)

  $VT_1[i] = VT_2[i]$, for all $i = 1, \ldots, N$

- $VT_1 \leq VT_2$,

  iff $VT_1[i] \leq VT_2[i]$, for all $i = 1, \ldots, N$

- Two events are causally related *iff*

  $VT_1 < VT_2$, i.e.,

  iff $VT_1 \leq VT_2$ &

  there exists j such that

  $1 \leq j \leq N$ & $VT_1[j] < VT_2[j]$

… or Not Causally-Related

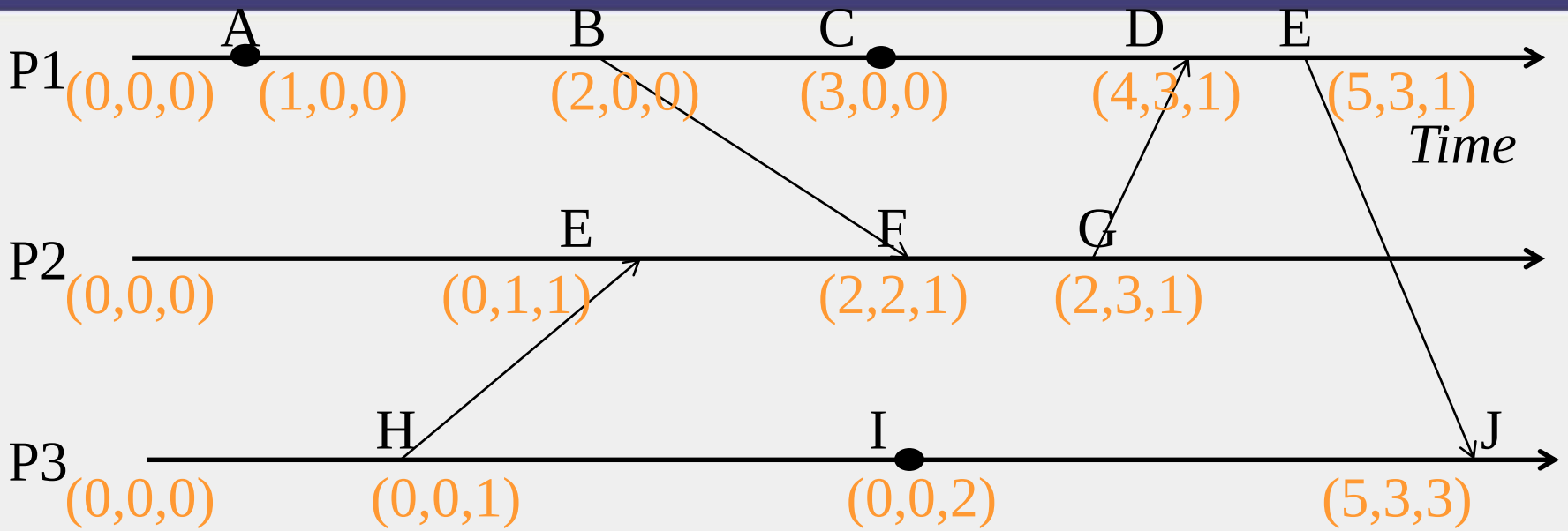
- Two events VT_1 and VT_2 are concurrent

  iff

  NOT ($VT_1 \leq VT_2$) AND NOT ($VT_2 \leq VT_1$)

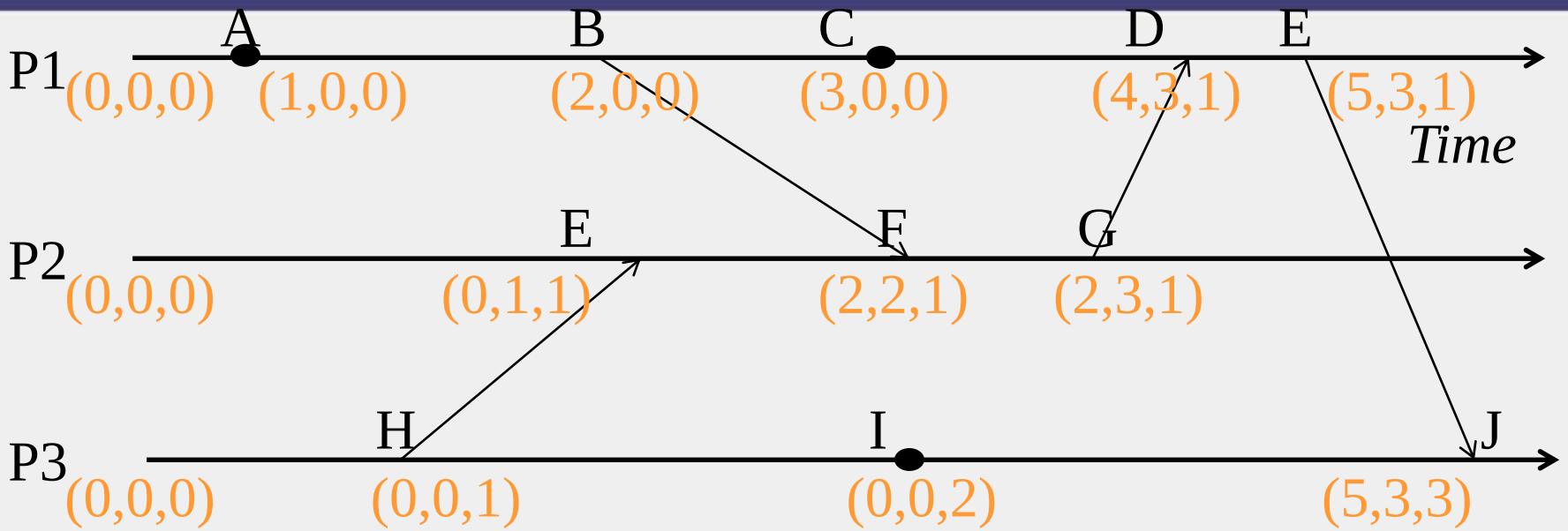
  We'll denote this as $VT_2 \,|||\, VT_1$

Obeying Causality



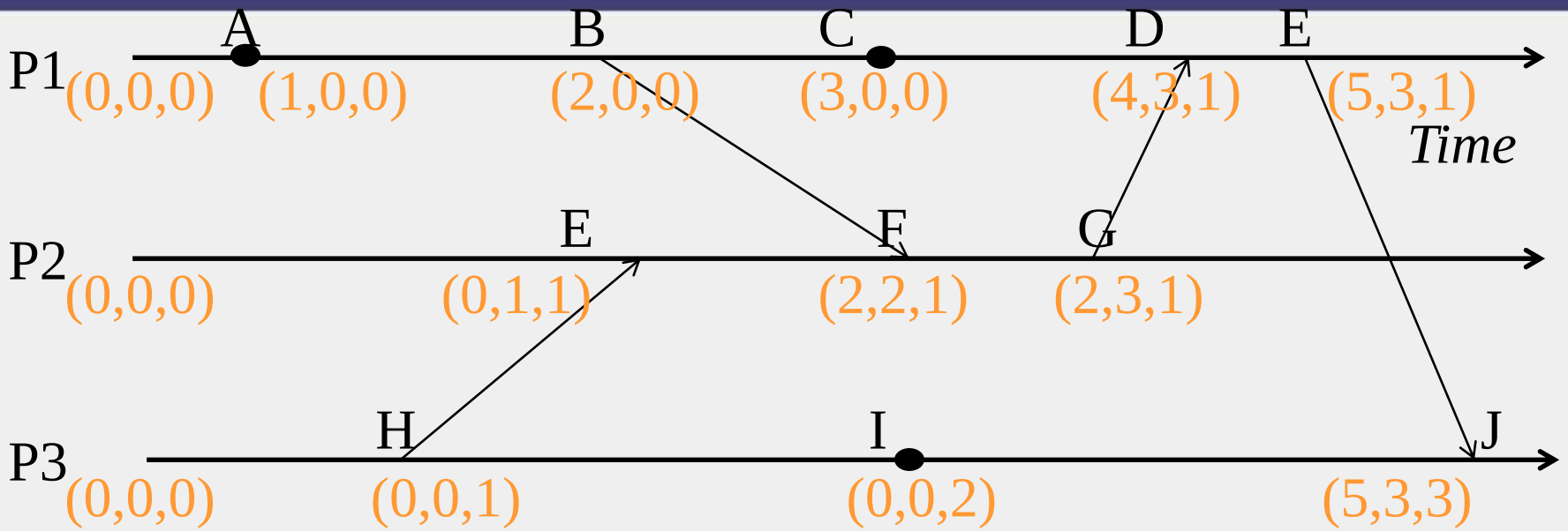
- A → B :: (1,0,0) < (2,0,0)
- B → F :: (2,0,0) < (2,2,1)
- A → F :: (1,0,0) < (2,2,1)

Obeying Causality (2)



- H → G :: (0,0,1) < (2,3,1)
- F → J :: (2,2,1) < (5,3,3)
- H → J :: (0,0,1) < (5,3,3)
- C → J :: (3,0,0) < (5,3,3)

Identifying Concurrent Events



- C & F :: (3,0,0) ||| (2,2,1)
- H & C :: (0,0,1) ||| (3,0,0)
- (C, F) and (H, C) are pairs of *concurrent* events

Logical Timestamps: Summary

- **Lamport timestamps**
  - Integer clocks assigned to events
  - Obey causality
  - Cannot distinguish concurrent events
- **Vector timestamps**
  - Obey causality
  - By using more space, can also identify concurrent events

Time and Ordering: Summary

- **Clocks are unsynchronized in an asynchronous distributed system**
- **But need to order events, across processes!**
- **Time synchronization**
  - Cristian's algorithm
  - NTP
  - Berkeley algorithm
  - But error a function of round-trip-time
- **Can avoid time sync altogether by instead assigning logical timestamps to events**