Programación Distribuida y Tiempo Real

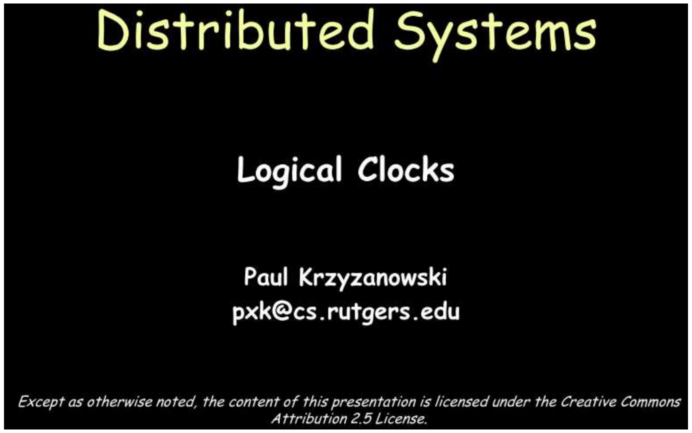
Sincronización – Relojes Lógicos

Relojes Lógicos

- Se deja de lado la asociación reloj-fecha y hora
  - No se corrije *drift*, ni *offset*, ni *skew*
- No se conserva la noción de proporcionalidad de tiempo

Relojes Lógicos

- Se deja de lado la asociación reloj-fecha y hora
  - No se corrije *drift*, ni *offset*, ni *skew*
- No se conserva la noción de proporcionalidad de tiempo
- Se focaliza la relación de orden de la sucesión de eventos
- En la base de todo el desarrollo: relación "antes de"
- CS 417: Distributed Systems, Paul Krzyzanowski
  https://www.cs.rutgers.edu/~pxk/417/index.html



Relojes Lógicos

Logical clocks

- Assign sequence numbers to messages
  - All cooperating processes can agree on order of events
  - vs. **physical clocks**: time of day
- Assume no central time source
  - Each system maintains its own local clock
  - No total ordering of events
    - No concept of *happened-when*

Relojes Lógicos

Happened-before

Lamport's "happened-before" notation

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

e.g.: a: message being sent, b: message receipt

Transitive:

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

Relojes Lógicos

Assign "clock" value to each event

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

- En principio, un contador de eventos locales
- Ante la ocurrencia de un evento
    - Incrementar el valor del contador
    - "Tiempo" de ocurrencia del evento: valor del contador
- En cada computadora es sencillo
    - Contador + "estampilla" para los eventos
    - Seguro se respeta la relación "antes de"

Relojes Lógicos

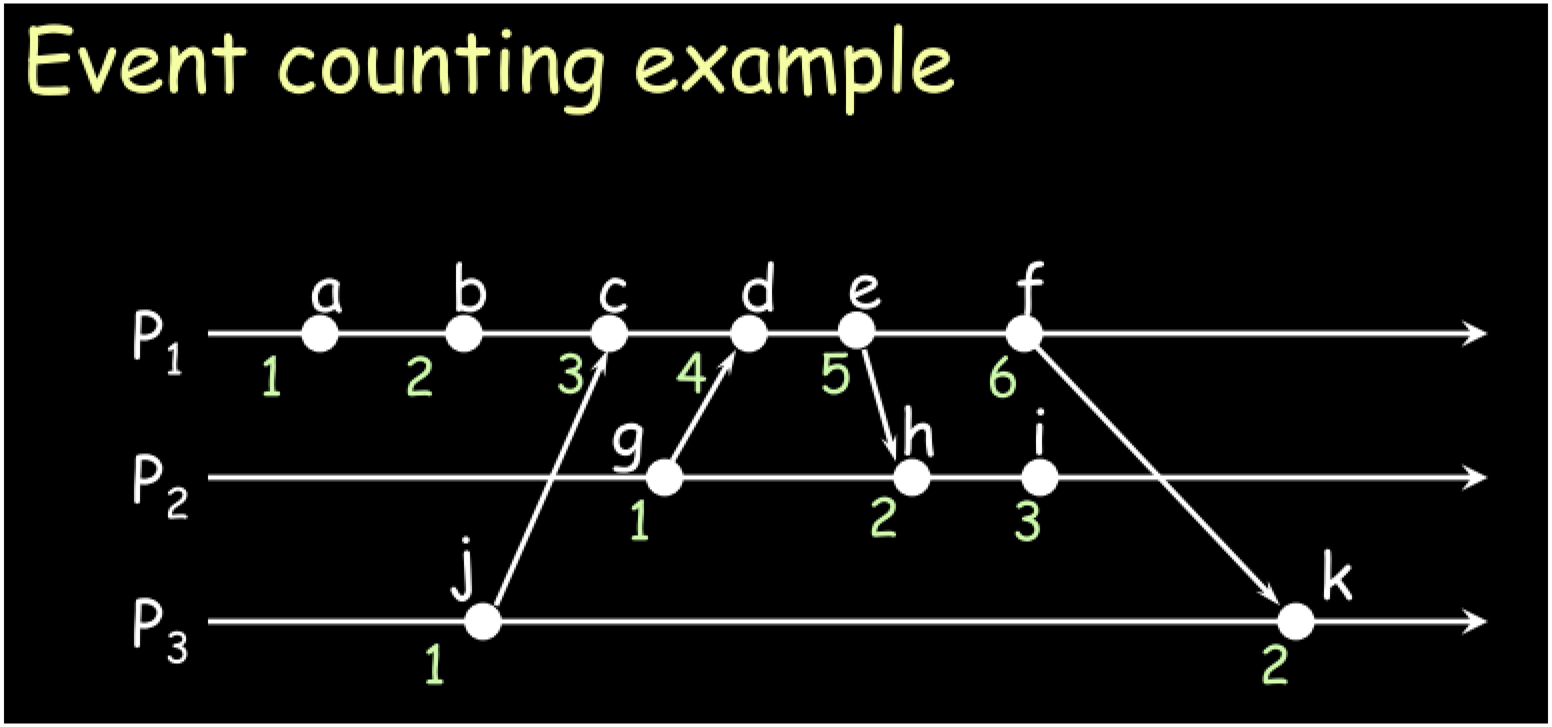
Assign "clock" value to each event

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

- En principio, un contador de eventos locales
- Ante la ocurrencia de un evento
  - Incrementar el valor del contador
  - "Tiempo" de ocurrencia del evento: valor del contador
- En cada computadora es sencillo
  - Contador + "estampilla" para los eventos
  - Seguro se respeta la relación "antes de"

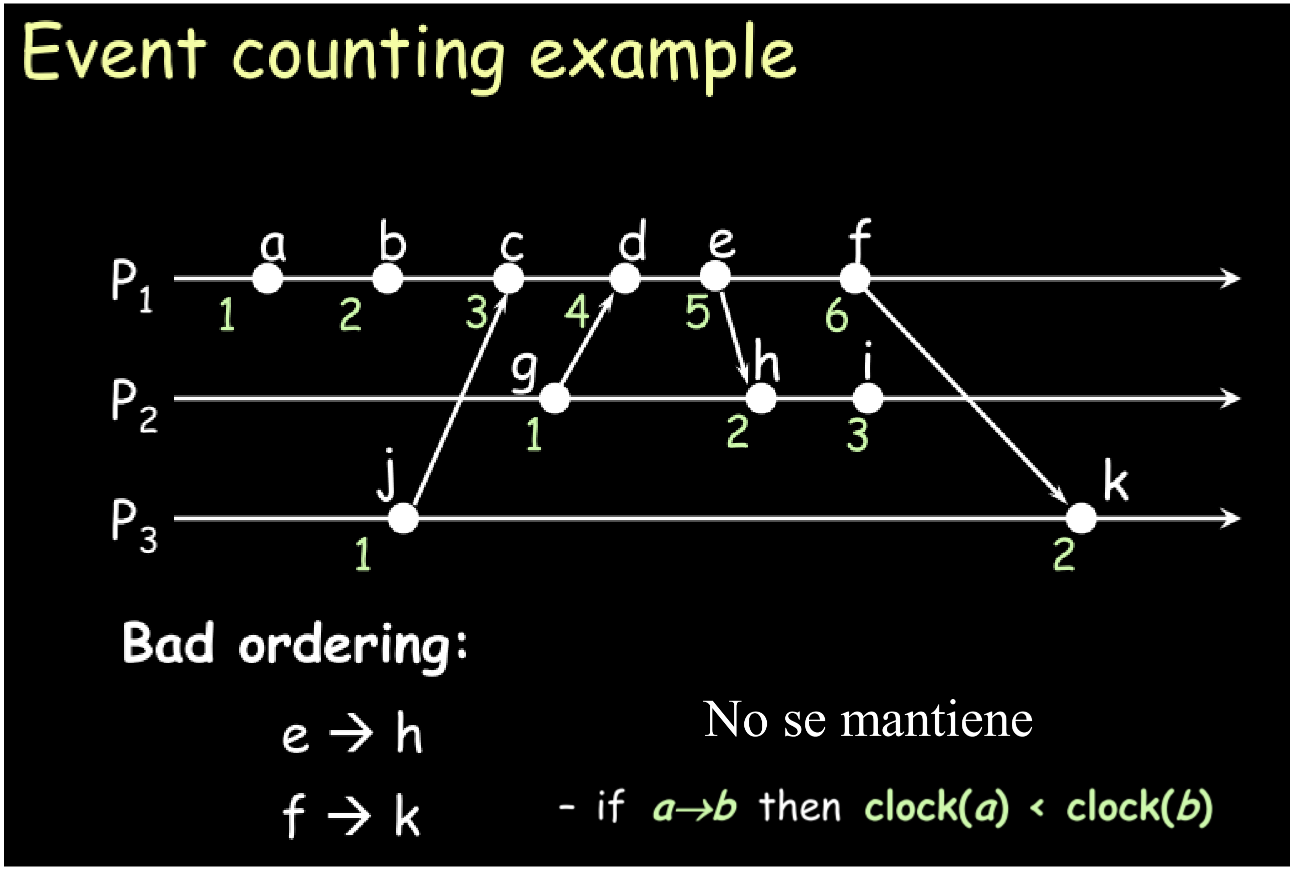
Relojes Lógicos

- Si solo contamos…



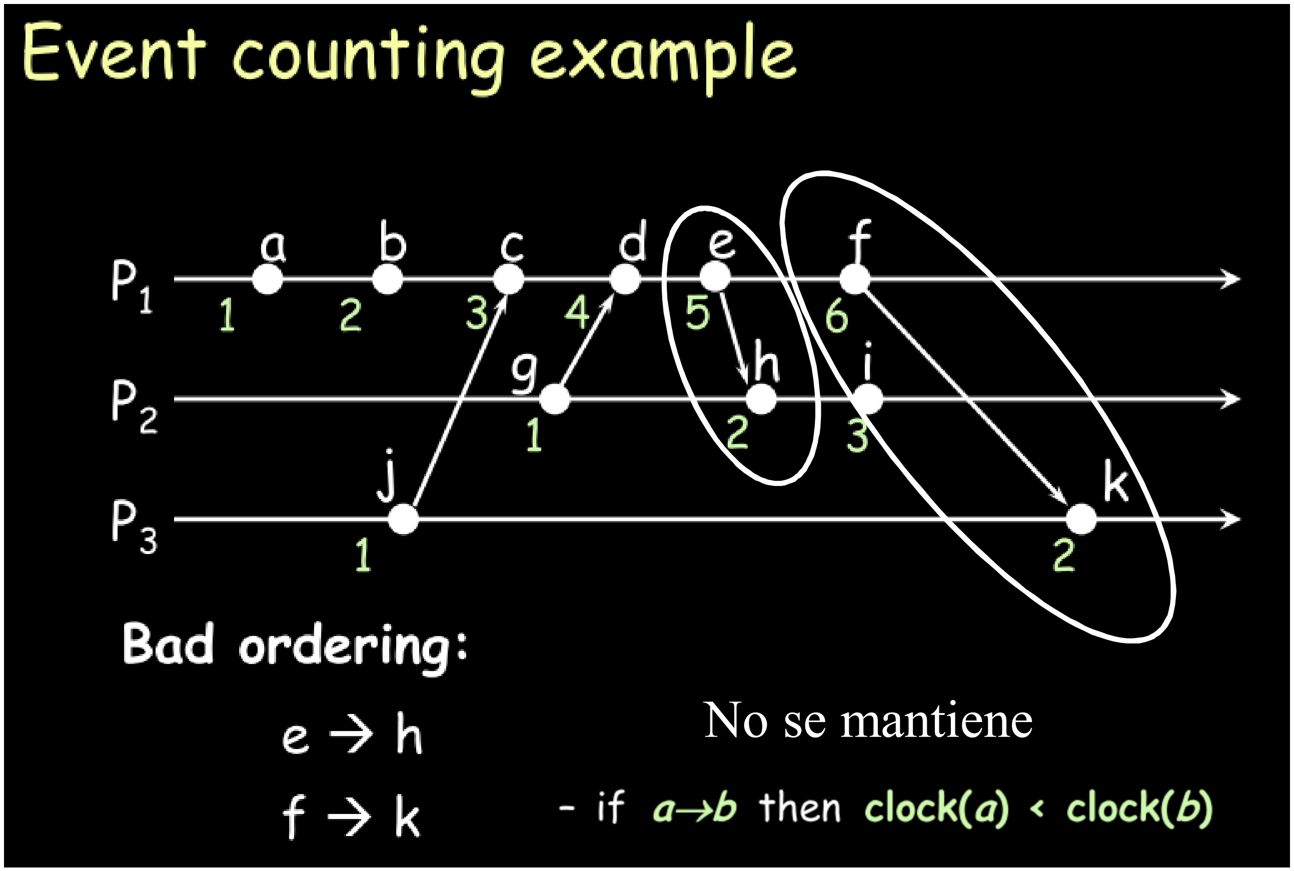
Relojes Lógicos

- Si solo contamos…



Relojes Lógicos

- Si solo contamos…



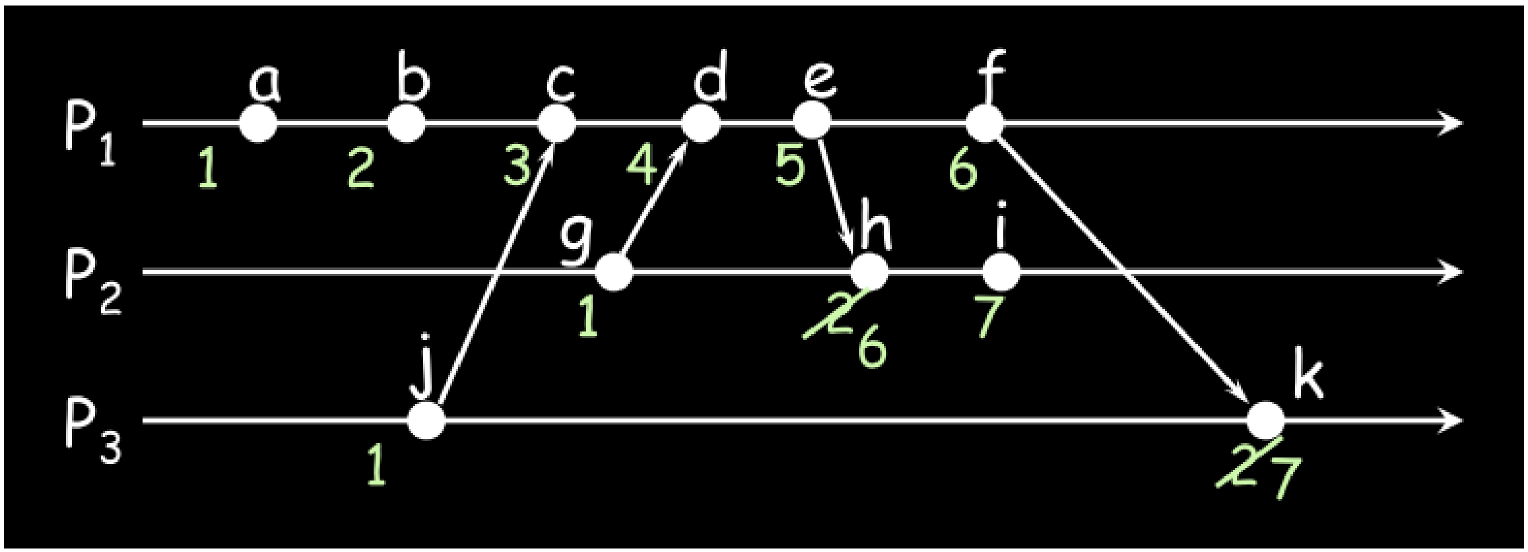
Relojes Lógicos

Lamport's algorithm

- Each message carries a timestamp of the sender's clock
- When a message arrives:
  - if receiver's clock < message timestamp
    set system clock to (message timestamp + 1)
  - else do nothing
- Clock must be advanced between any two events in the same process

Relojes Lógicos

Lamport's algorithm



- Localmente se avanza
- Mensaje recibido: verificar contador de envío
- Monotónicamente creciente en todos los procesos

Relojes Lógicos

Lamport's algorithm

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

- These events are **concurrent**

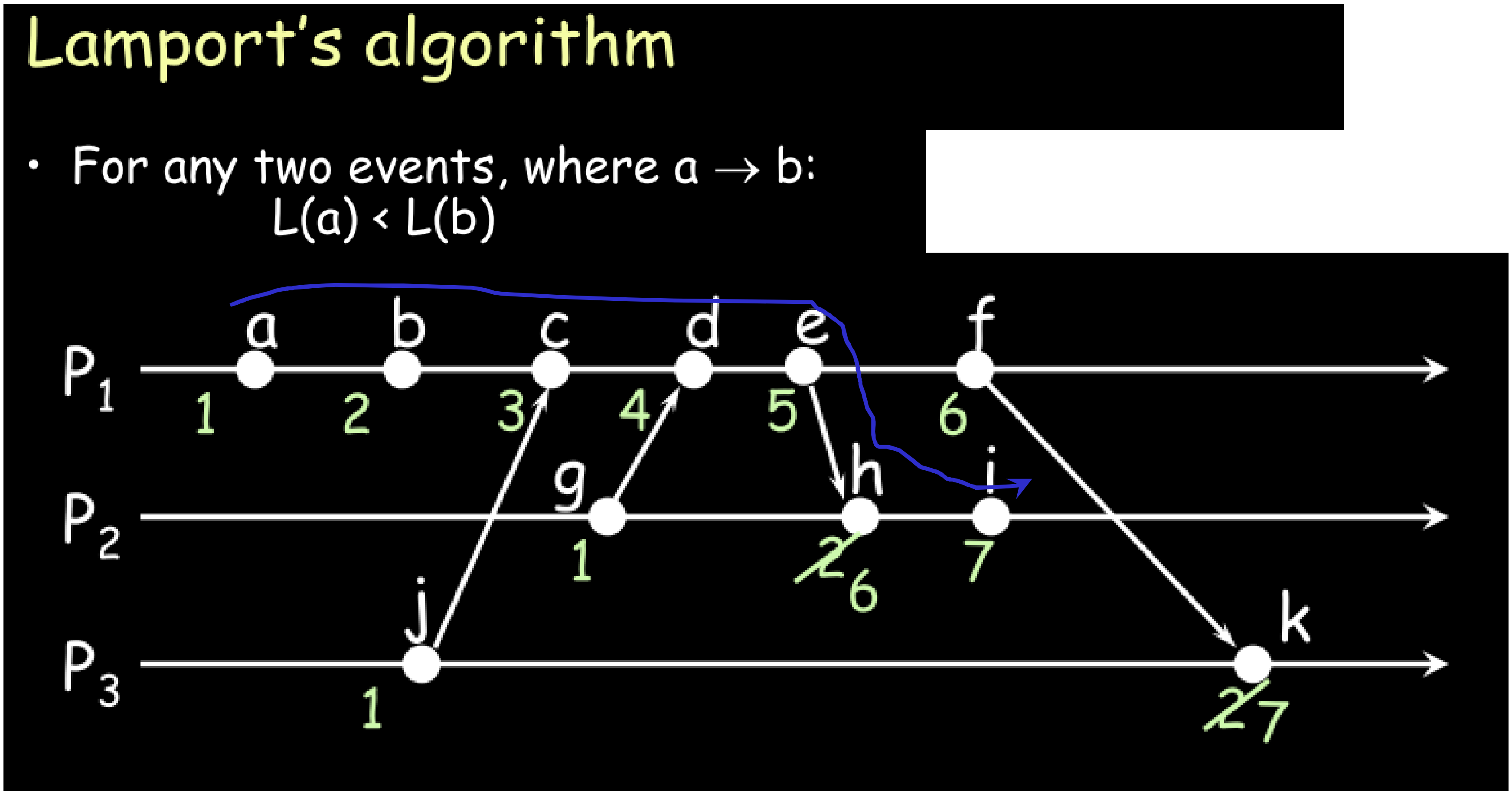
Algorithm allows us to maintain time ordering among related events

- *Partial ordering*

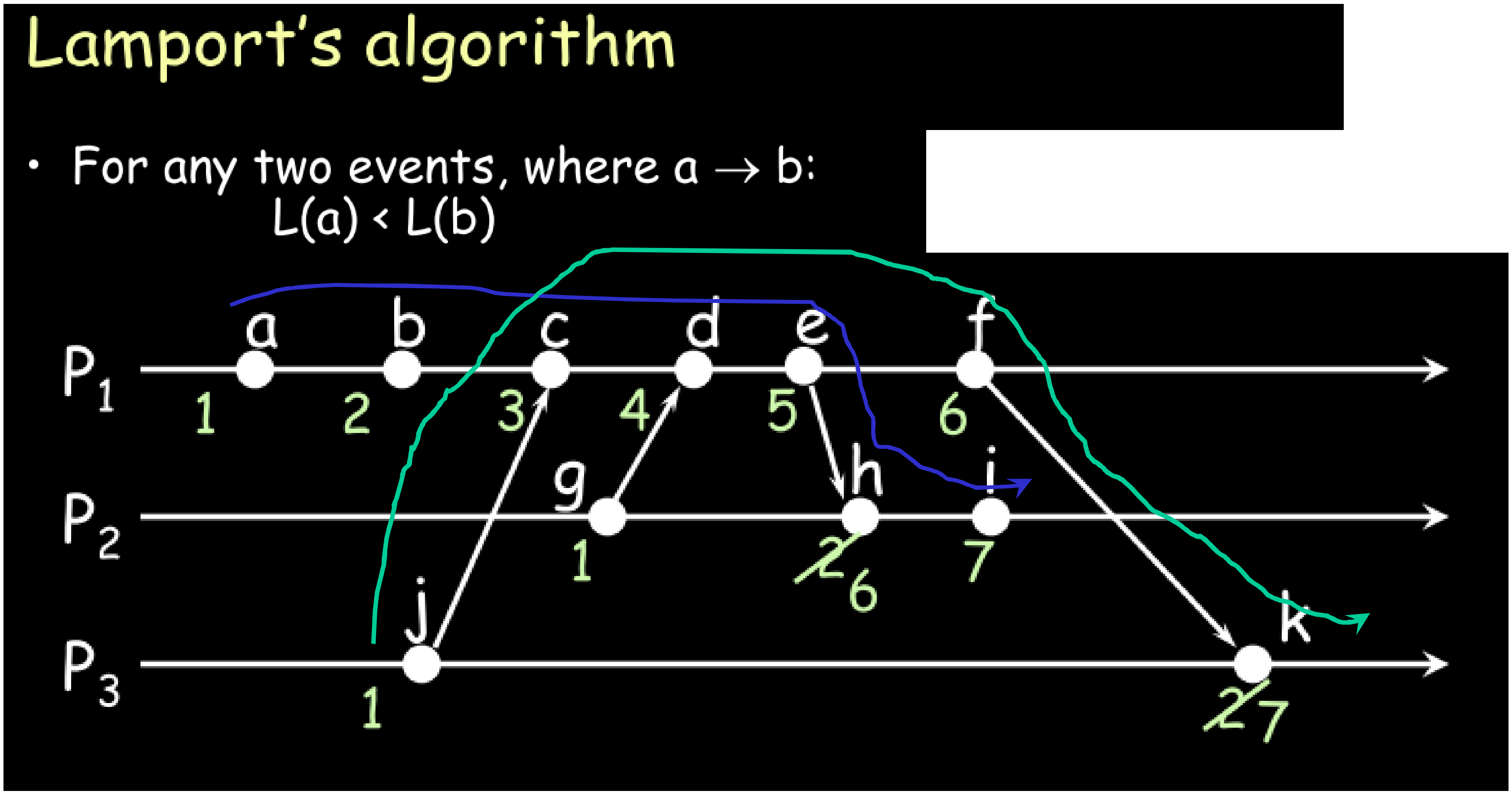
- Each event has a **Lamport timestamp** attached to it
- For any two events, where $a \rightarrow b$:

  $L(a) < L(b)$

Relojes Lógicos



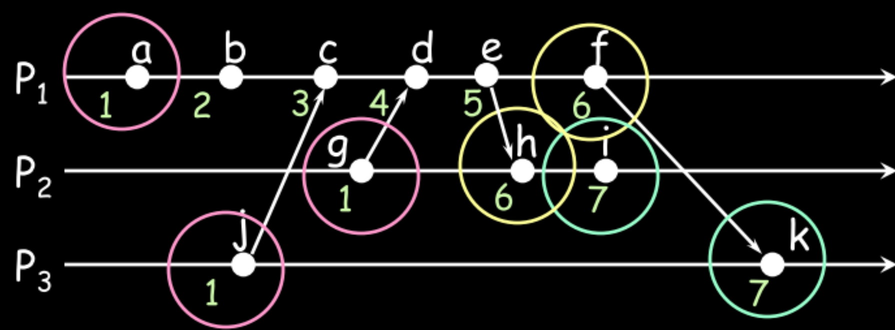
Relojes Lógicos



Relojes Lógicos

Lamport's algorithm

Problem: Identical timestamps



a→b, b→c, …: local events sequenced

i→c, f→d , d→g, … : Lamport imposes a *send*→*receive* relationship

Concurrent events (e.g., a & i) *may* have the same timestamp … or not

Relojes Lógicos

Lamport's algorithm

Unique timestamps (total ordering)

We can force each timestamp to be unique

- Define global logical timestamp (T_i, i)
  - T_i represents local Lamport timestamp
  - i represents process number (globally unique)
    - E.g. (host address, process ID)
- Compare timestamps:

  $(T_i, i) < (T_j, j)$

  if and only if

  $T_i < T_j$ or

  $T_i = T_j$ and $i < j$

Does not relate to event ordering

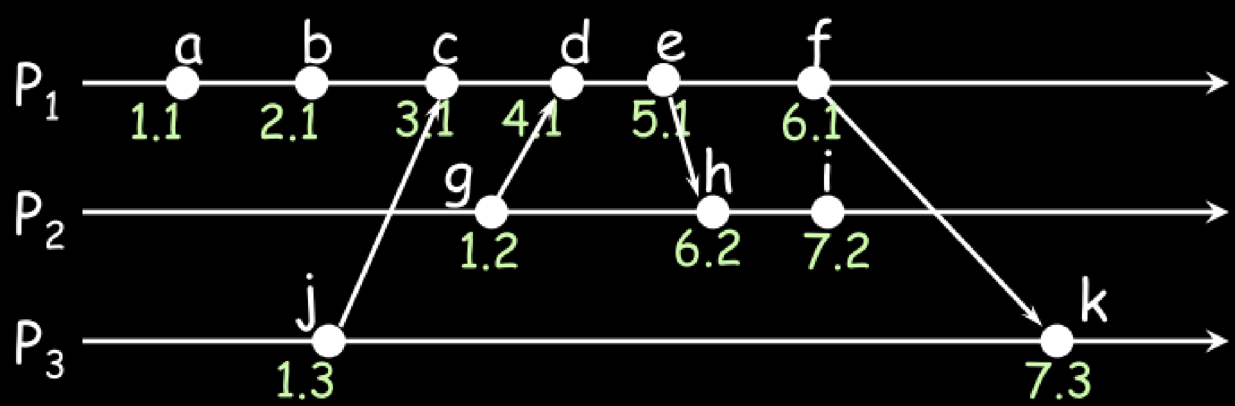
Relojes Lógicos

Lamport's algorithm

Unique timestamps (total ordering)

Does not relate to event ordering

Unique (totally ordered) timestamps



Relojes Lógicos

Lamport's algorithm

- For any two events, where a $\rightarrow$ b:
  $L(a) < L(b)$

Problem: Detecting causal relations

If $L(e) < L(e')$
- Cannot conclude that $e \rightarrow e'$

Looking at Lamport timestamps
- Cannot conclude which events are causally related

Relojes Lógicos

Vector clocks

Rules:

1. Vector initialized to 0 at each process

   $V_i[j] = 0$ for $i, j = 1, \ldots, N$

2. Process increments its element of the vector in local vector before timestamping event:

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

3. Message is sent from process P_i with V_i attached to it

4. When P_j receives message, compares vectors element by element and sets local vector to higher of two values

   $V_j[i] = \max(V_i[i], V_j[i])$ for $i=1, \ldots, N$

Relojes Lógicos

Comparing vector timestamps

<u>Define</u>

$V = V'$ iff $V[i] = V'[i]$ for $i = 1 \ldots N$

$V \leq V'$ iff $V[i] \leq V'[i]$ for $i = 1 \ldots N$

For any two events e, e'

if $e \rightarrow e'$ then $V(e) < V(e')$

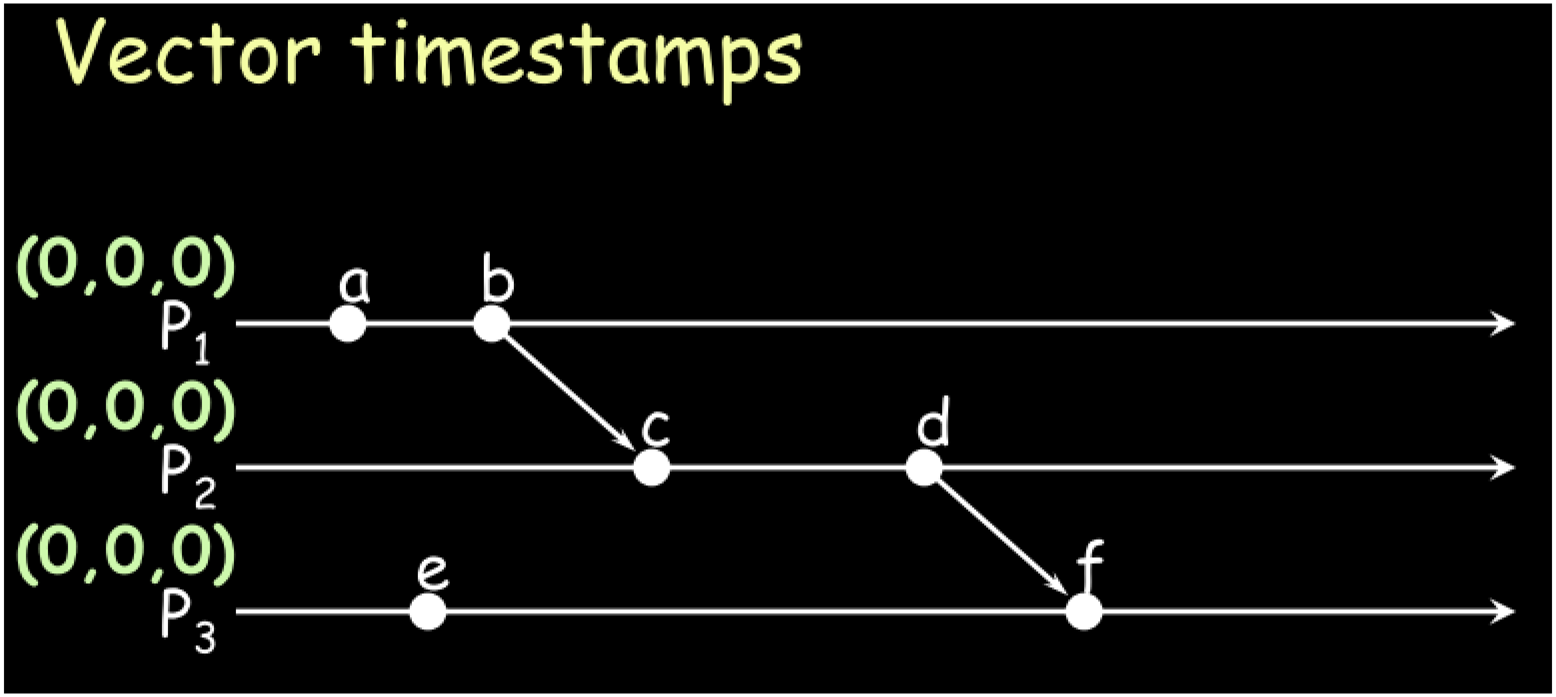
- Just like Lamport's algorithm

if $V(e) < V(e')$ then $e \rightarrow e'$

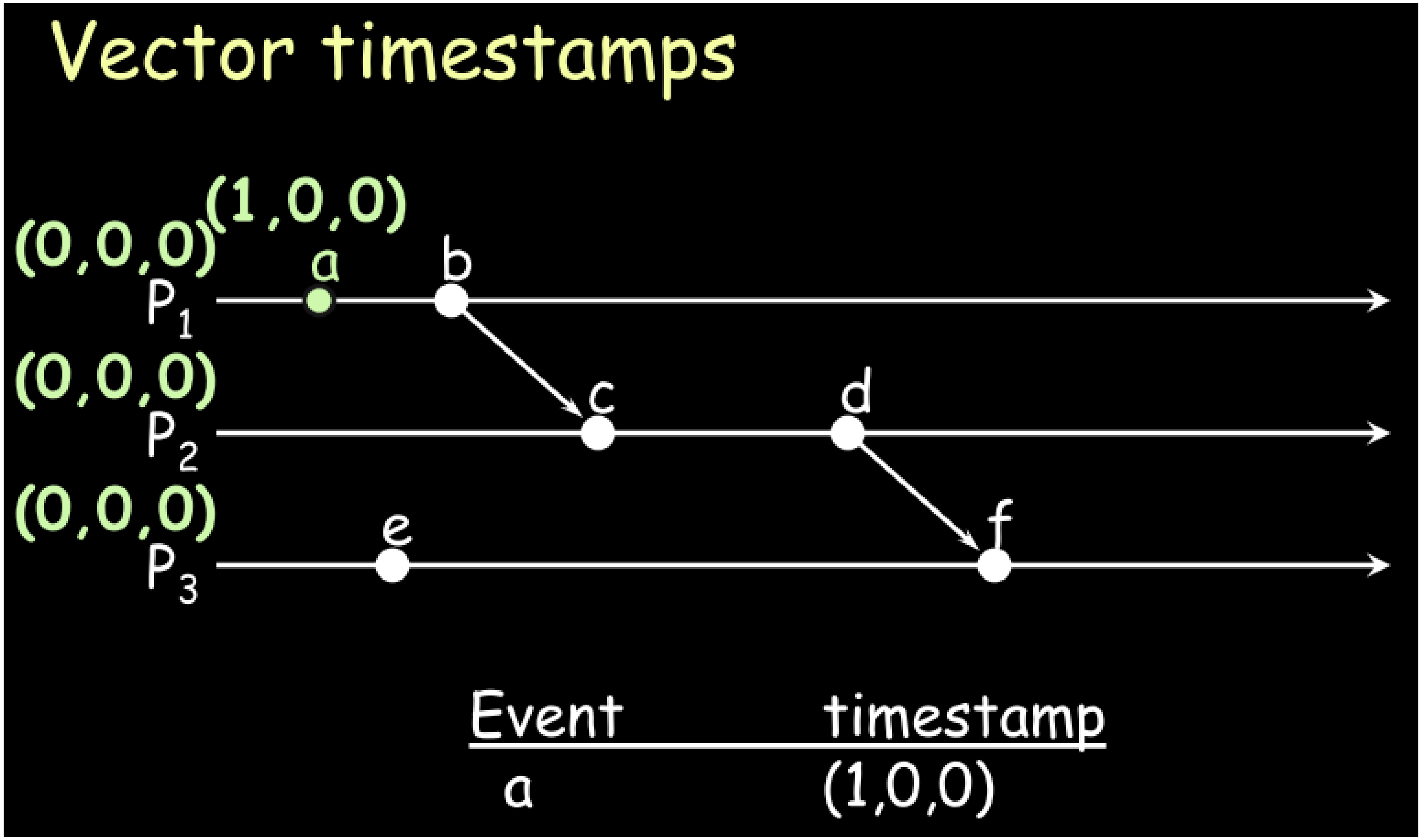
Two events are **concurrent** if **neither**

$V(e) \leq V(e')$ nor $V(e') \leq V(e)$

Relojes Lógicos

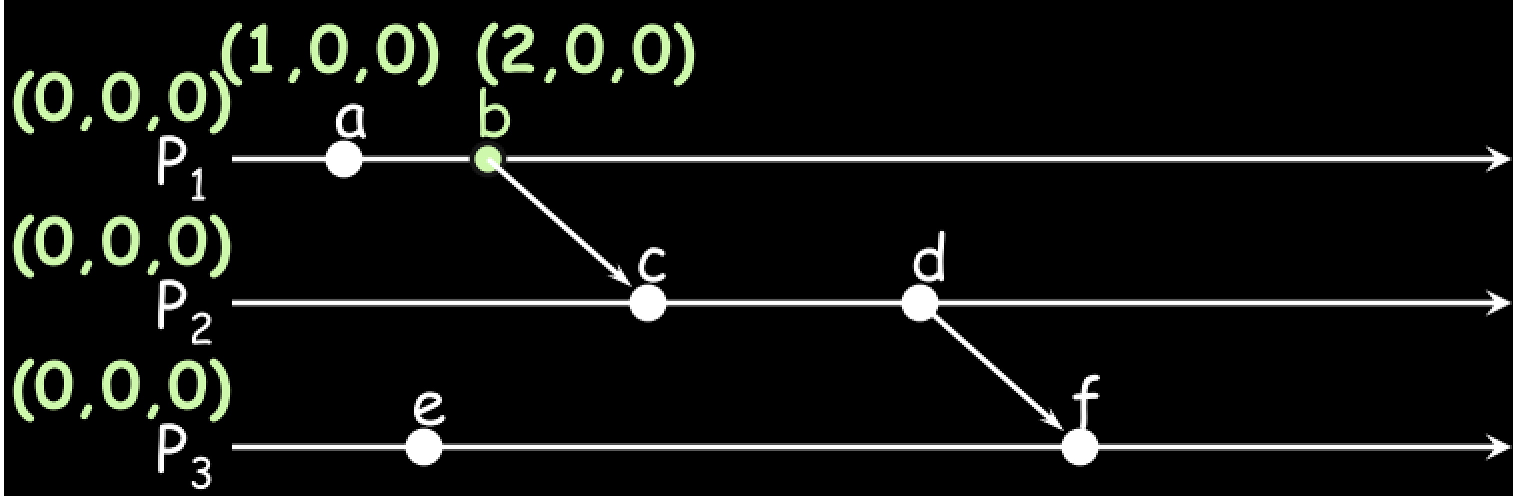


Relojes Lógicos



Relojes Lógicos

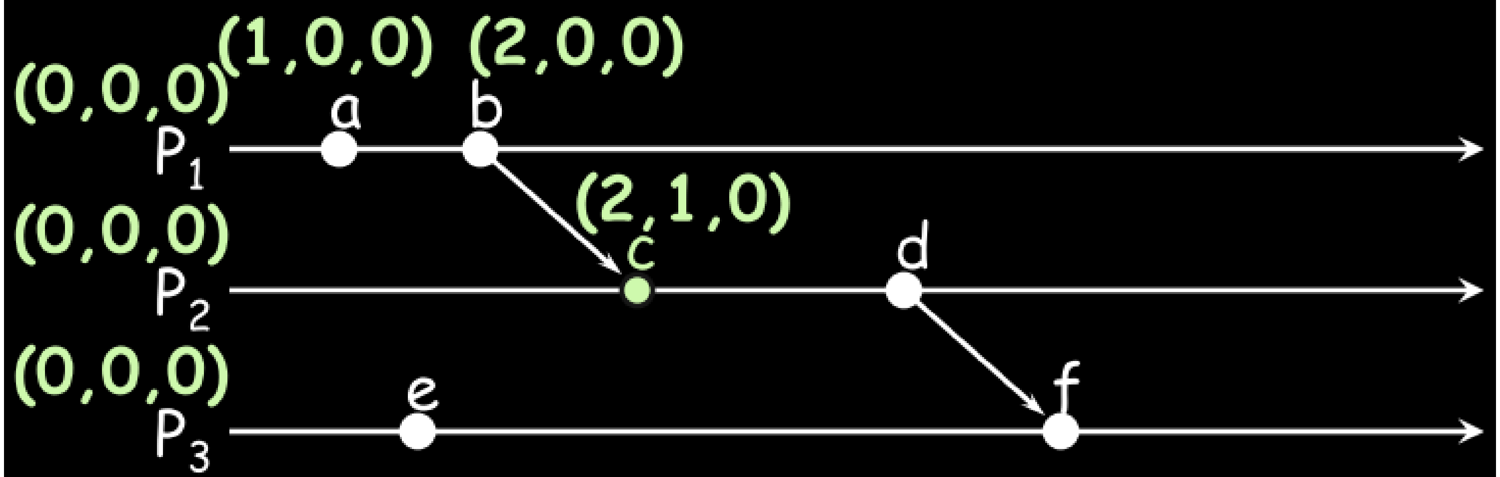
Vector timestamps



Event	timestamp
a	(1,0,0)
b	(2,0,0)

Relojes Lógicos

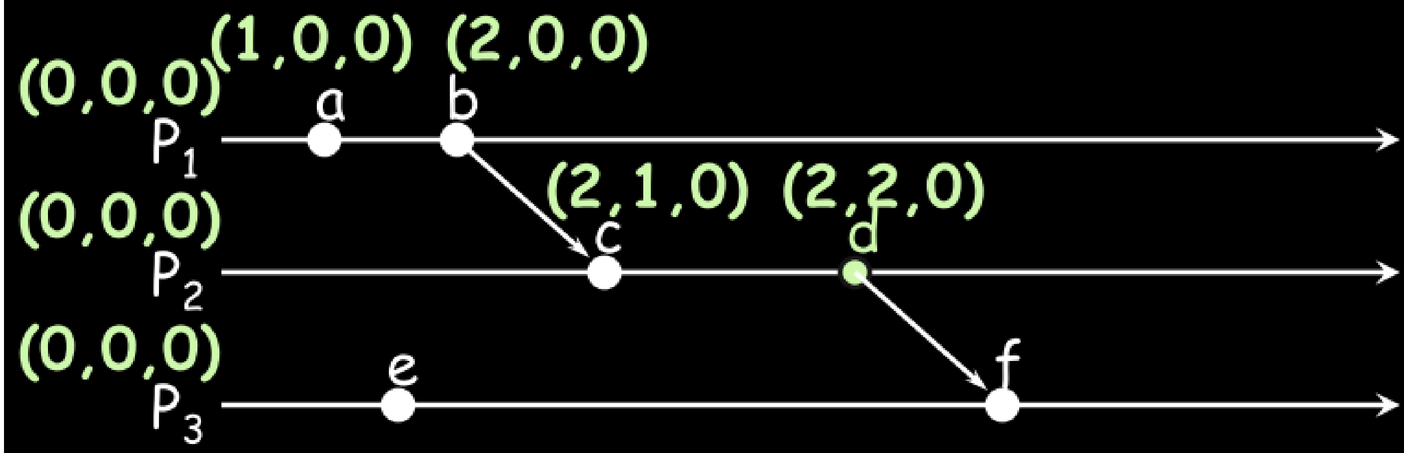
Vector timestamps



Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)

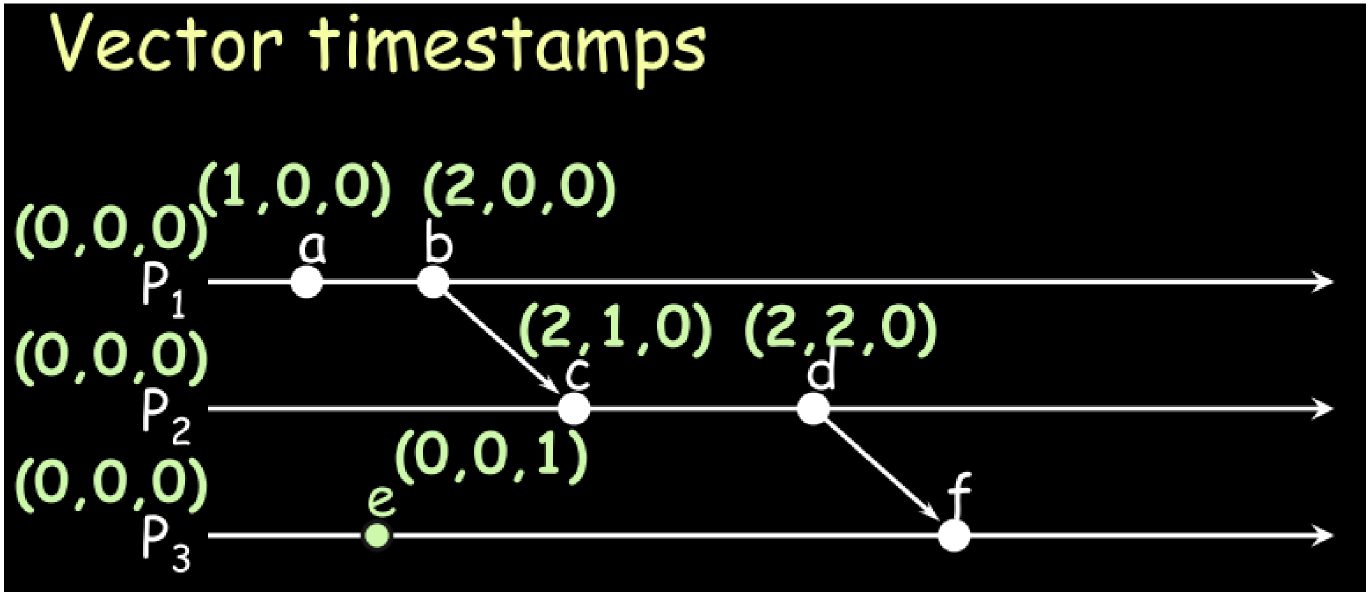
Relojes Lógicos

Vector timestamps



Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)

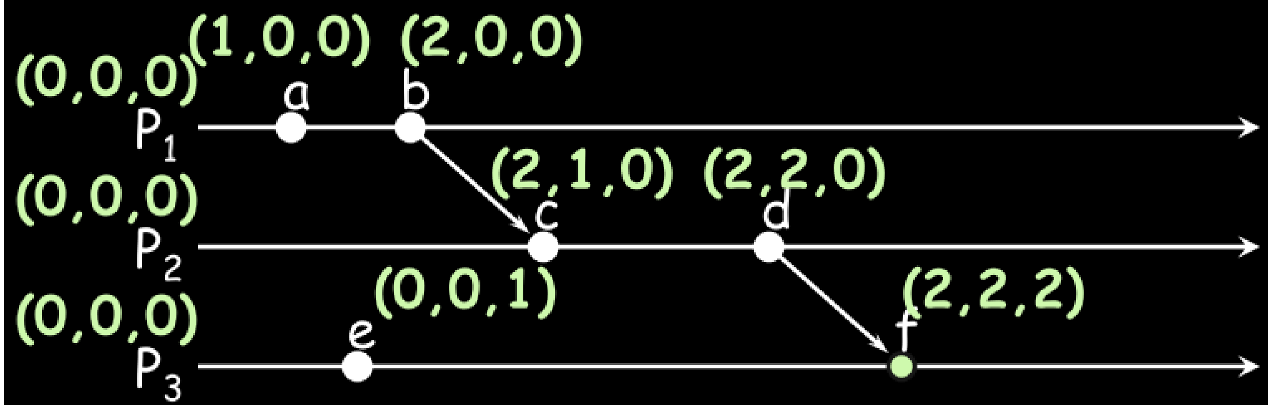
Relojes Lógicos



Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)

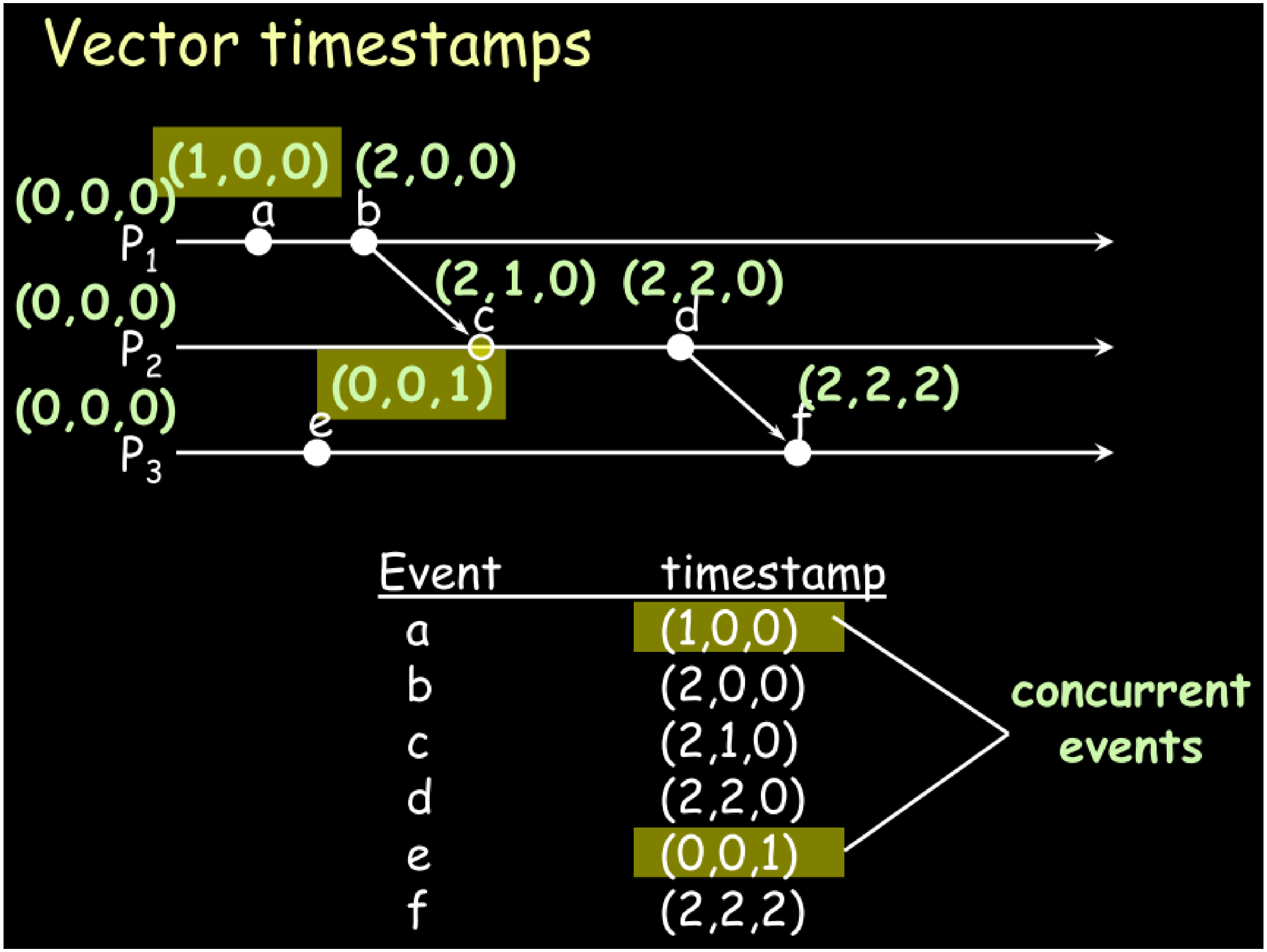
Relojes Lógicos

Vector timestamps

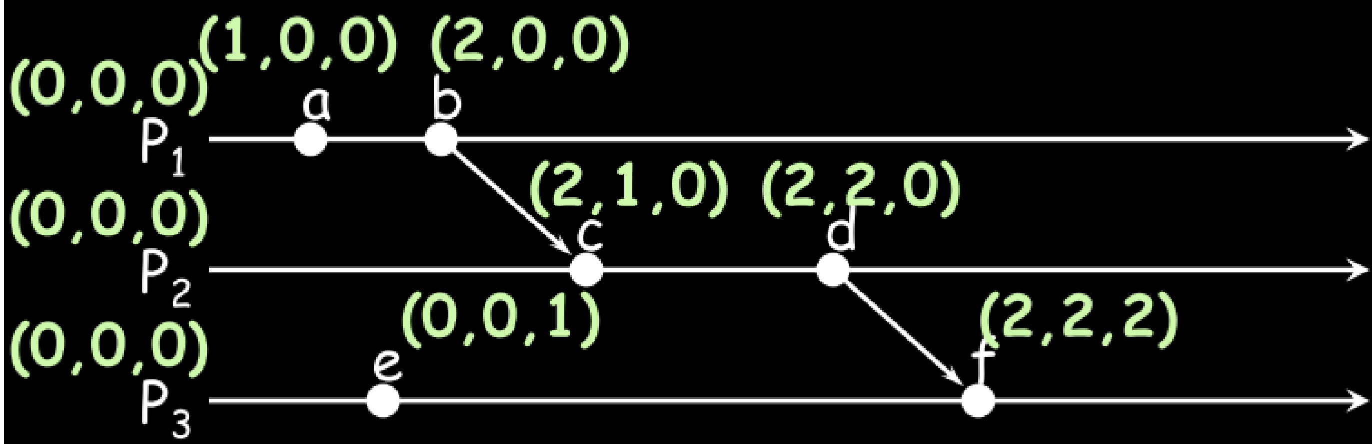


Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

Relojes Lógicos



Relojes Lógicos

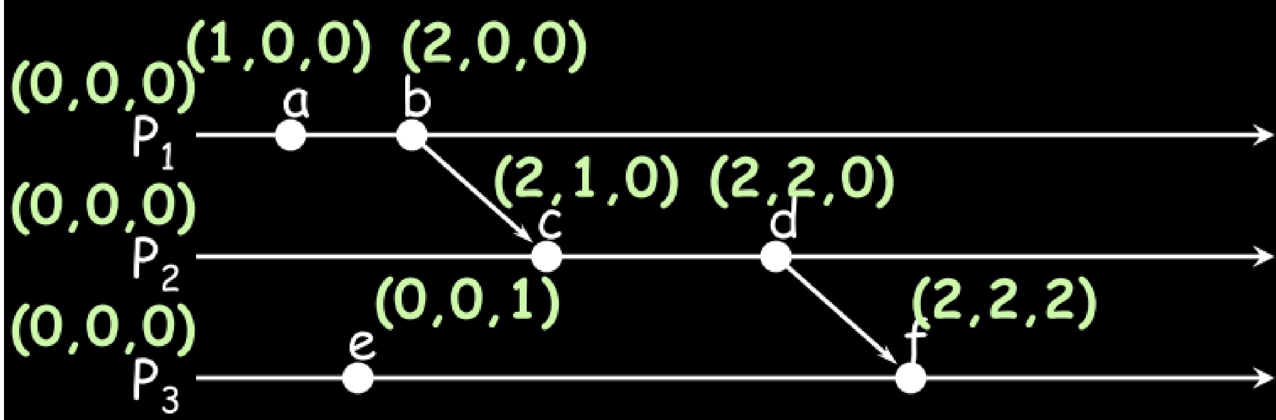


Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

concurrent events

Relojes Lógicos

Vector timestamps

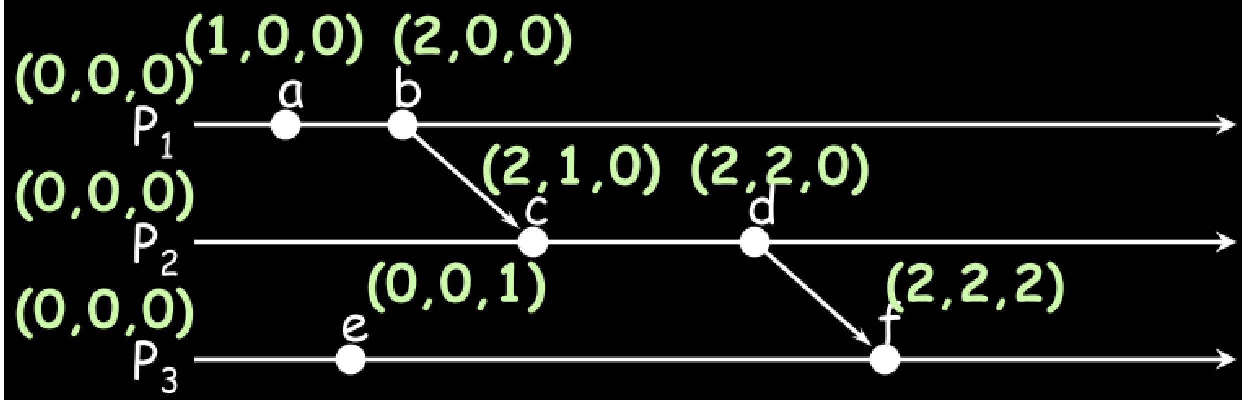


Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

concurrent events

Relojes Lógicos

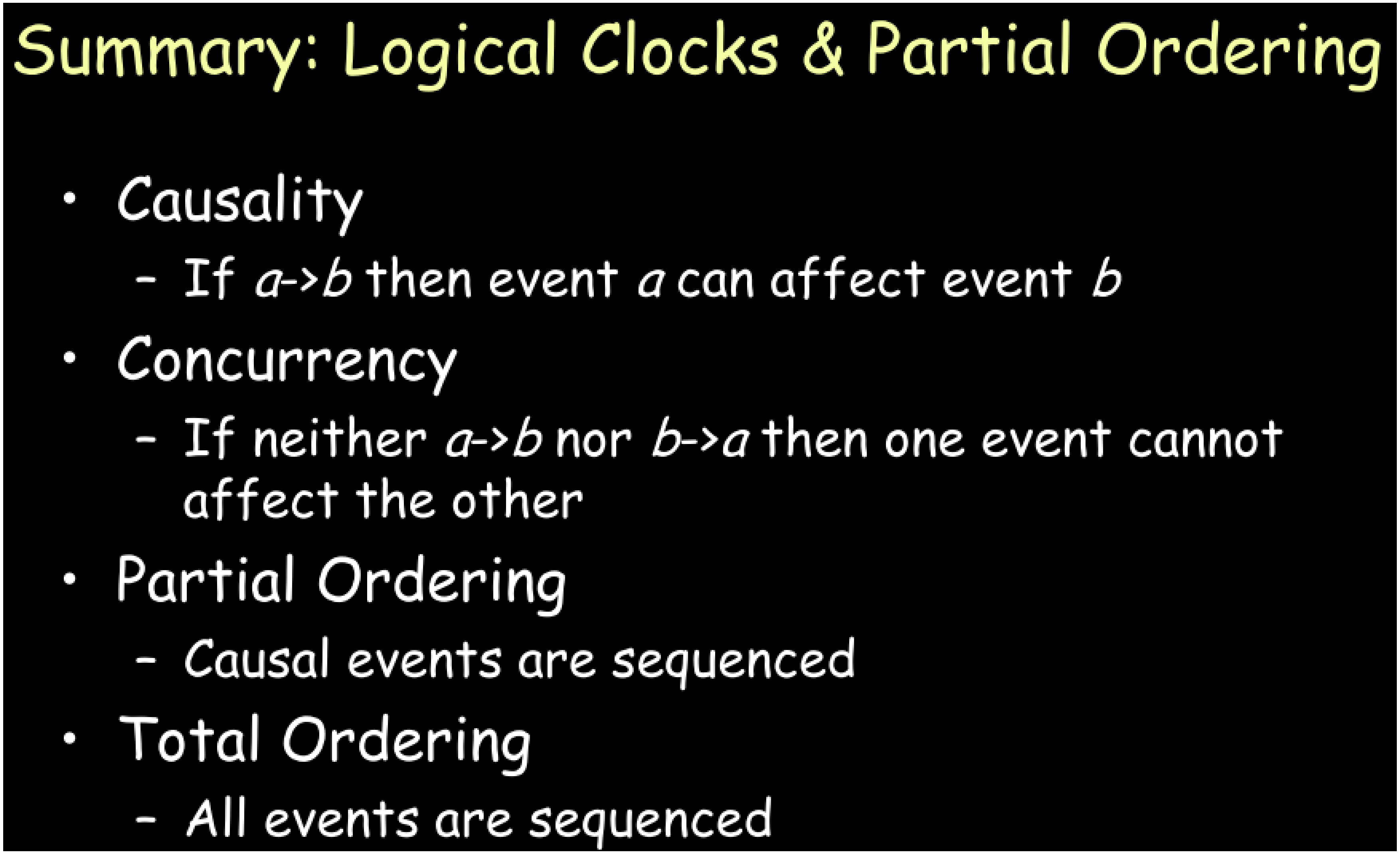
Vector timestamps



Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

concurrent events

Relojes Lógicos



Dudas/Consultas

- Plataforma Ideas