Resumen Relojes Lamport y Vectorial

Concurrencia de datos mediante relojes lógicos

Jesus Humberto Abundis Patiño

Relojes Lamport

- Cuentan número de eventos en cada máquina
- Mensaje enviado y recibido cuenta como evento
- Relación pasó-antes entre eventos

a —> b significa “a” pasó antes que “b”

Pares de eventos con esta relación

- Todos los eventos que ocurren en un sólo sistema; el evento con la estampa de tiempo menor ocurrió primero.

- Eventos que ocurrieron en distintos equipos tienen una relación pasó antes si están conectados mediante el envío y recepción de mensajes.

  Si un proceso P1 envía un mensaje a P2, entonces todos los eventos en P1 que ocurrieron antes de dicho envío tbn ocurrieron antes que los eventos en P2 que surjan a partir de que este recibe el msj

Actualización del Reloj

- Los procesos empiezan con una hora 0
- El proceso que envíe un msj aumentará su reloj en una unidad.
- El procesos que reciba el msj actualizará su hora “t_actual” mediante

$$t\_actual \leftarrow \max(t\_actual, t\_recibido) + 1$$

Garantiza que si “a” ocurrió antes que “b”,
el valor del reloj lógico de “a” es menor al de “b”

Relaciones pasó-antes

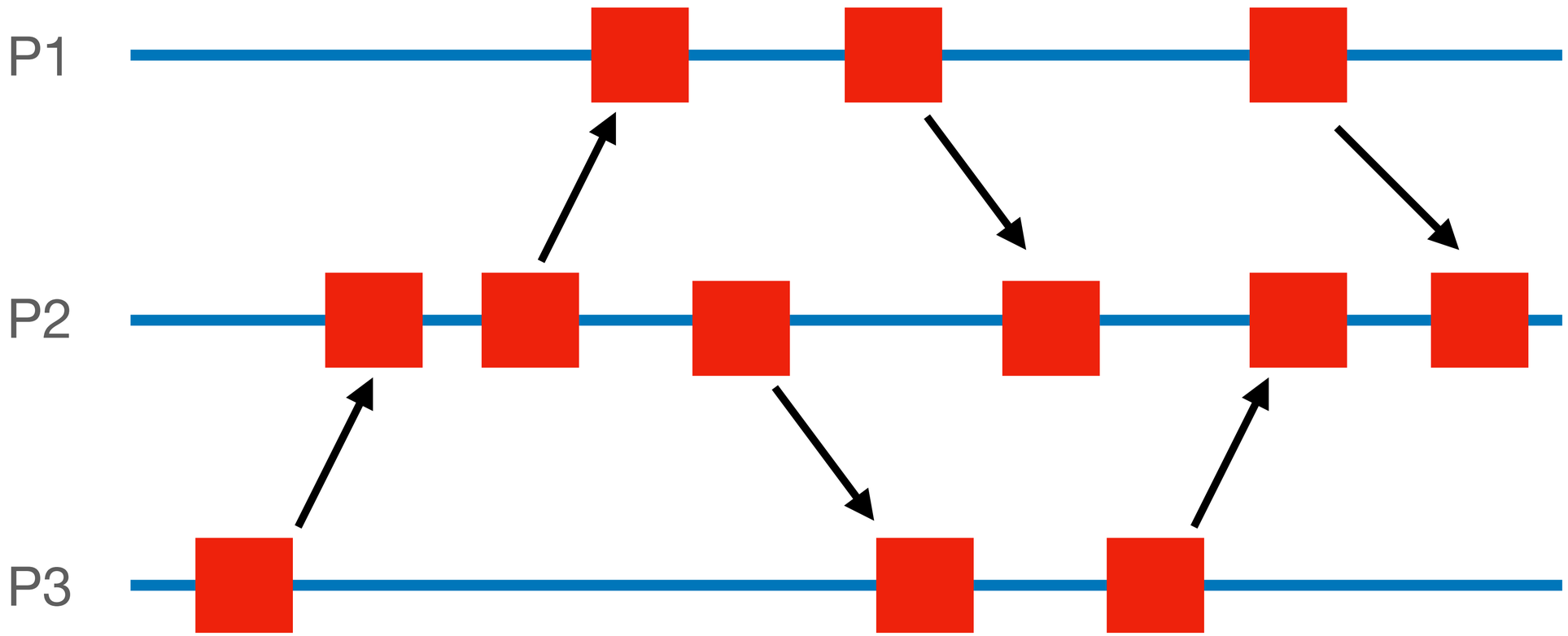
- Conexión a través del mismo proceso
- Siguiendo un msj a otro proceso

Concurrentes

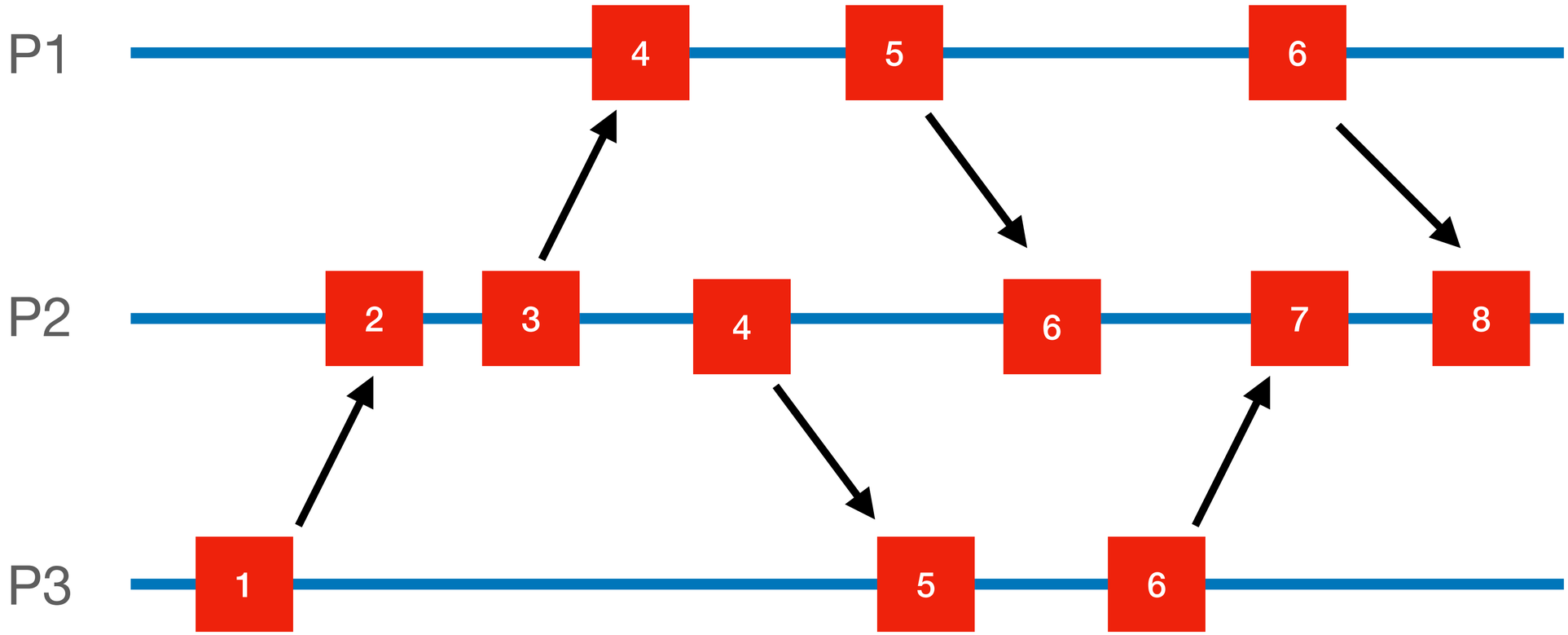
Eventos sin relación pasó-antes se llaman concurrentes

- El sistema no es afectado por el orden de operaciones
- No ocurren necesariamente al mismo tiempo

Ejemplo Lamport



Ejemplo Lamport



Orden entre los eventos

⇨ Relación pasó antes entre:

(P3; 1) —> (P1; 4)

⇨ Eventos concurrentes

(P3; 6) con (P1; 6)

(P1; 5) con (P2; 4)

Observe que eventos no ocurren al mismo tiempo

Algoritmo (logrando orden del sistema)

- Proceso envía request a todos los equipo agregando al mensaje su reloj Lamport
- Receptores colocan msj en un queue ordenado por valor de reloj y Envían msj de recibido a emisor y demás equipos
- Los equipos actualizan su reloj
- Los equipos envían el msj al principio del queue (eliminándolo de este) si se ha notificado la recepción de dicho msj por otros equipos

Cada operación require n^2 mensajes (n computadoras)

Continuación

Valores de Reloj x < y $\Longrightarrow$ t_x <= t_y

Dijimos antes que si “a” ocurre antes que “b”, entonces el reloj lógico de “a” es menor al de “b”, pero nunca dijimos que si la hora lógica de “a” era menor a la lógica de “b” implicaría que “a” ocurrió antes que “b”

Relojes Vectoriales

⇨ Son una extensión a los relojes Lamport dónde cada máquina guarda su reloj y el de las otras en un vector, e.g., (5, 4, 2) en el segundo equipo para una red con tres equipos

⇨ Estos si capturan la causalidad
(dados dos eventos, ¿cuál ocurrió primero?)

El evento “a” ocurrió antes que “b” si el vector correspondiente a “a” cumple con que todas sus componentes son menores o iguales a las de “b”; de otra manera “a” y “b” son concurrentes.

Relojes Vectoriales

Los vectores con la hora son tuplas ordenadas

Considere tres procesos, denotados por P1, P2 y P3, tenemos que cada uno contará con un reloj vectorial de tres componentes, las cuales mantienen un orden; la primer componente siempre se referirá a P1, la segunda a P2 y la tercera a P3.

De esta manera podemos tener que P2 tenga un reloj vectorial (5, 4, 2), lo cual significa que la hora en P2 es 4 y que tiene conocimiento que la hora en P1 es 5 y la hora en P3 es 2; este conocimiento se actualiza cada que recibe un msj.

Actualización del vector

- Todos los equipos empiezan con un vector inicializado con ceros
- Cuando un equipo se prepara a enviar un msj incrementa su reloj en una unidad y posteriormente envía el msj con la hora actualizada, e.g., si P3 tiene la hora (2, 7, 4), la actualiza a (2, 7, 5) y envía el request junto a la hora actualizada
- El equipo que recibe el msj incrementa su contador propio de acuerdo a las reglas de Lamport (t_local <— max(t_local, t_msj) + 1), e.g., P2 tiene la hora (3, 5, 2) y recibe un msj de P3 el cual contiene la hora (2, 7, 5), entonces su hora es: max(5, 7) + 1 = 8, de tal manera que P2 actualiza la hora (3, 8, 2)

Actualización del vector

⇨ Posteriormente, el equipo que recibe procede a actualizar las demás entradas tomando el máximo entre pares correspondiente del vector del msj que recibió y su vector propio, e.g., las componentes 1 y 3 de P2 son 3 y 2, mientras que las correspondientes al msj son 2 y 5; los máximos son 3 y 5 de tal manera que la hora final en P2 es (3, 8, 5)

Ejemplo vectorial 1

Consideremos tres procesos que cuentan con relojes:

P1

(1, 4, 2)

P2

(3, 1, 1)

P3

(5, 2, 4)

Si P1 le envía un msj a P2, digamos la llamada de una función (RPC) que se encuentra en un servidor corriendo el proceso P2, los nuevos relojes serán:

P1

(2, 4, 2)

P2

(3, 5, 2)

Solución a ejemplo vectorial 1

- El proceso P1 incrementa su reloj en una unidad antes de enviar el msj

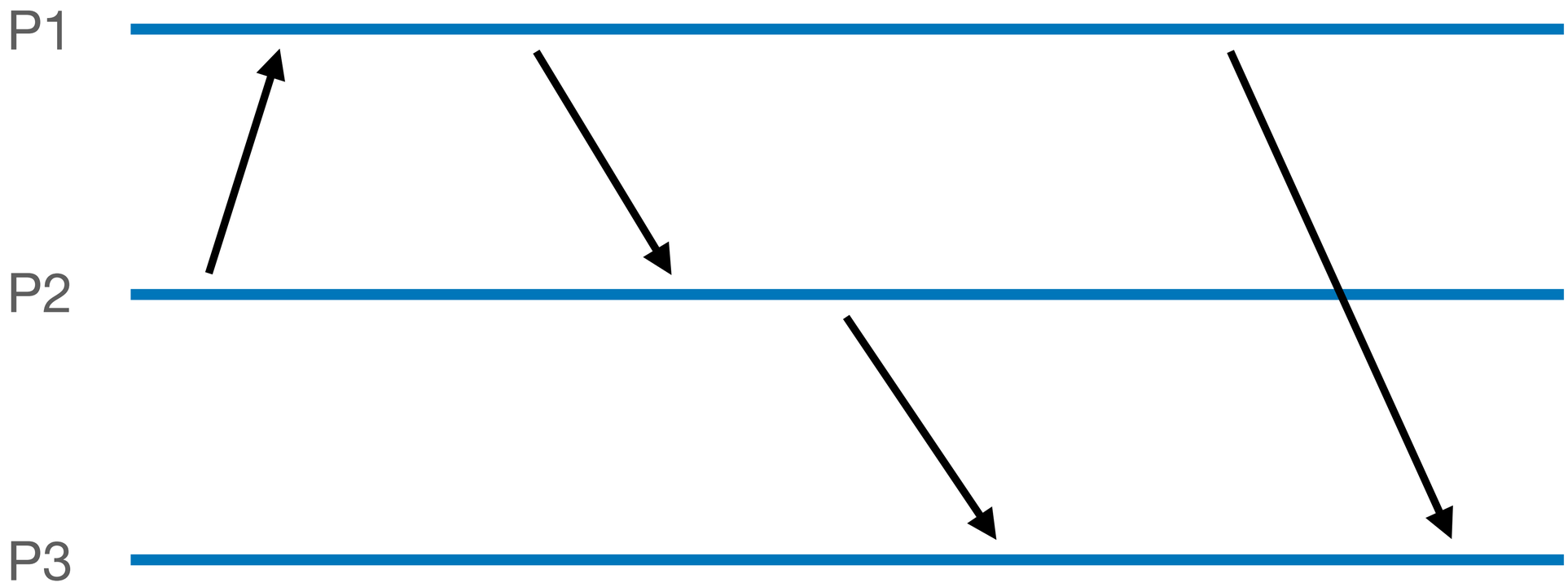
$$(1, 4, 2) \Longrightarrow (2, 4, 2)$$

- El proceso P2 aplica la regla Lamport a su reloj observando su hora (1) y la hora recibida (4) (estas horas corresponden a la segunda entrada de los vectores): t_actual <— max(t_actual, t_recibido) + 1 = 5

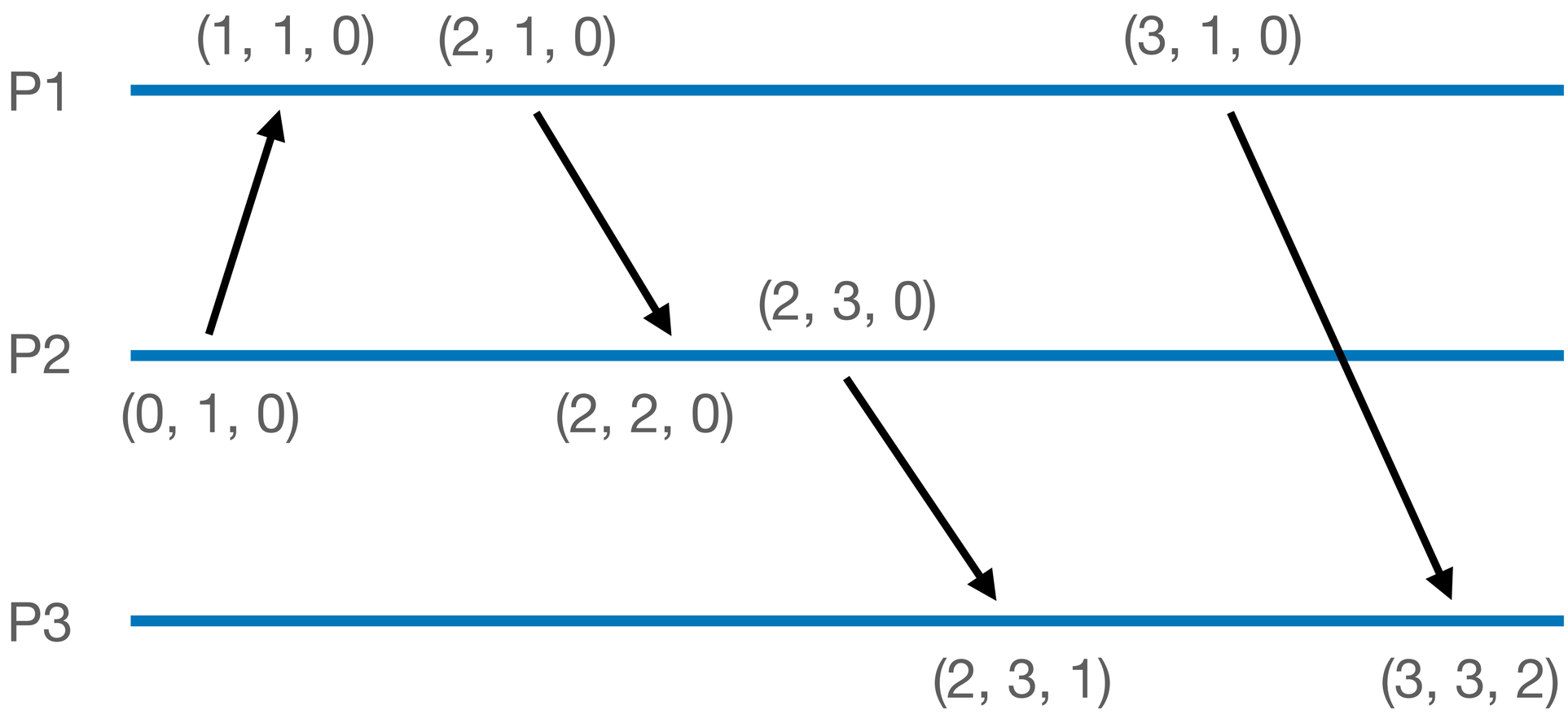
- El resto de los elementos del vector de P2 se actualizan tomando el máximo en las entradas restantes, i.e., el máximo en las entradas 1 y 3 entre el reloj local y el reloj del msj recibido;

Entradas 1 y 3 en P1: 2 y 2
Entradas 1 y 3 en P2: 3 y 1 $\Longrightarrow$ Maximos 3 y 2 $\Longrightarrow$ (3, 5, 2)

Ejemplo Vectorial 2



Ejemplo Vectorial 2



Orden entre los eventos

(P1; 2 ,1, 0) ocurrió antes que (P3; 2, 3, 1):

2 <= 2, 1<= 3, 0 <= 1

(P2; 2 ,3, 0) concurrente con (P1; 3, 1, 0):

$$(2,3,0) \not\leq (3,1,0) \quad \text{y} \quad (3,1,0) \not\leq (2,3,0)$$