# Tiempo, Estado y Relojes

## 30221 - Sistemas Distribuidos

**Rafael Tolosana Calasanz**

Dpto. Informática e Ing. de Sistemas

## Lectura Recomendada

- G. Colouris, J. Dollimore, T. Kindberg and G. Blair. Distributed systems: Concepts and Design. 5th Edition. Addison-Wesley. May, 2011. ISBN: 978-0132143011. **Chapter 14**
- Raynal, M. (2013). Distributed Algorithms for Message-Passing Systems: **Chapters 6, 10, 14**
- Tanenbaum Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall. **Chapter 6**

# Motivación

# Motivación

## Ejecución Secuencial

- Sucesión de **estados**
- ¿Qué es un estado?

# Motivación

## Ejecución Secuencial

- Sucesión de **estados**
- ¿Qué es un estado?

## Ejecución Distribuida

- Sucesión de **estados**
- ¿Qué es un estado?
- Vamos a considerar que un SD es un conjunto de procesos secuenciales $P_1 \ldots P_n$ que se comunican mediante el paso de mensajes asíncrono

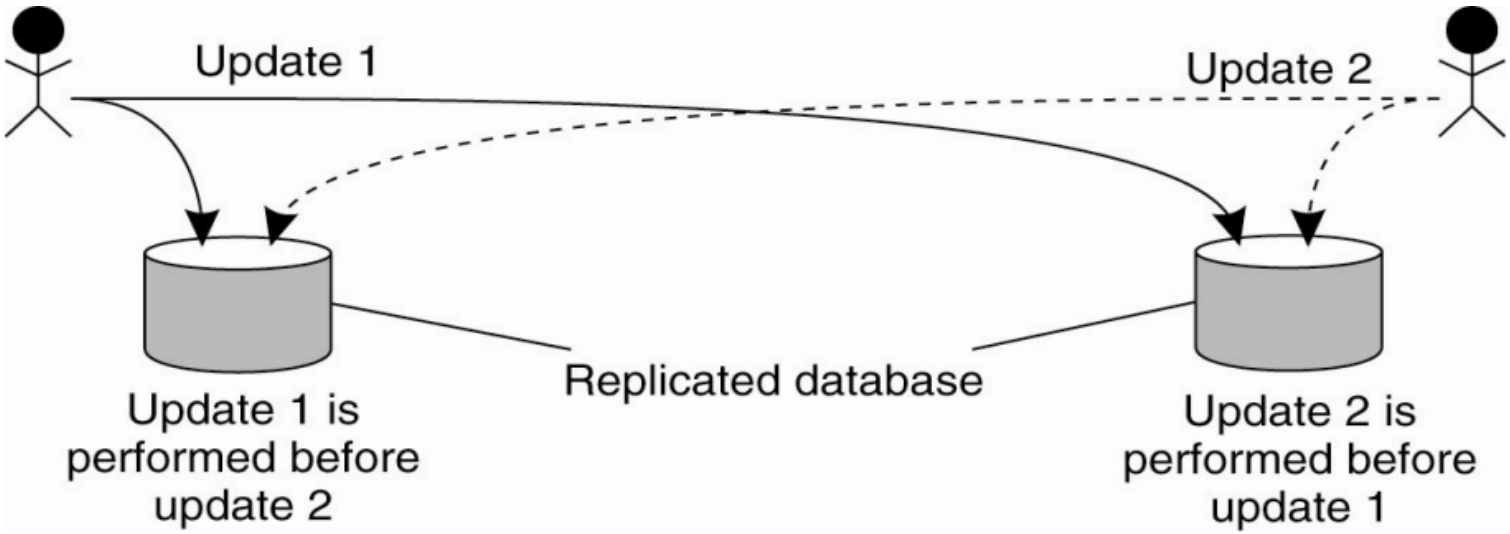
# Motivación

Con el intercambio de mensajes, los procesos de un SD pueden **cambiar de estado**:

- Saber el orden de ocurrencia de los eventos es útil
- Sincronización de procesos
- Determinar orden (causalidad)

# Motivación

## Database Replicas



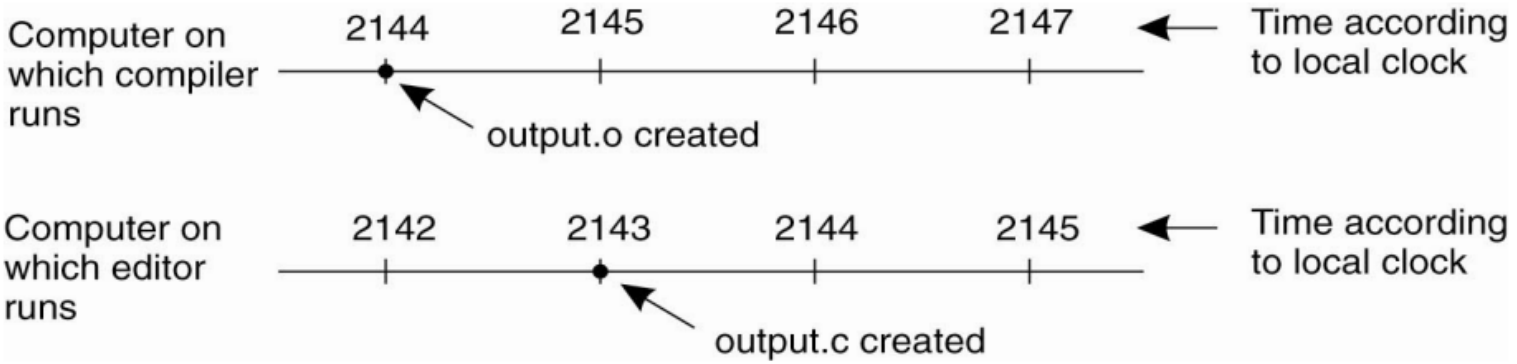

Updating a replicated database and leaving it in an inconsistent state.

[1]

[1] Tanenbaum Van Steen, Distributed Systems: 7 Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

# Motivación

## Edición & Compilación

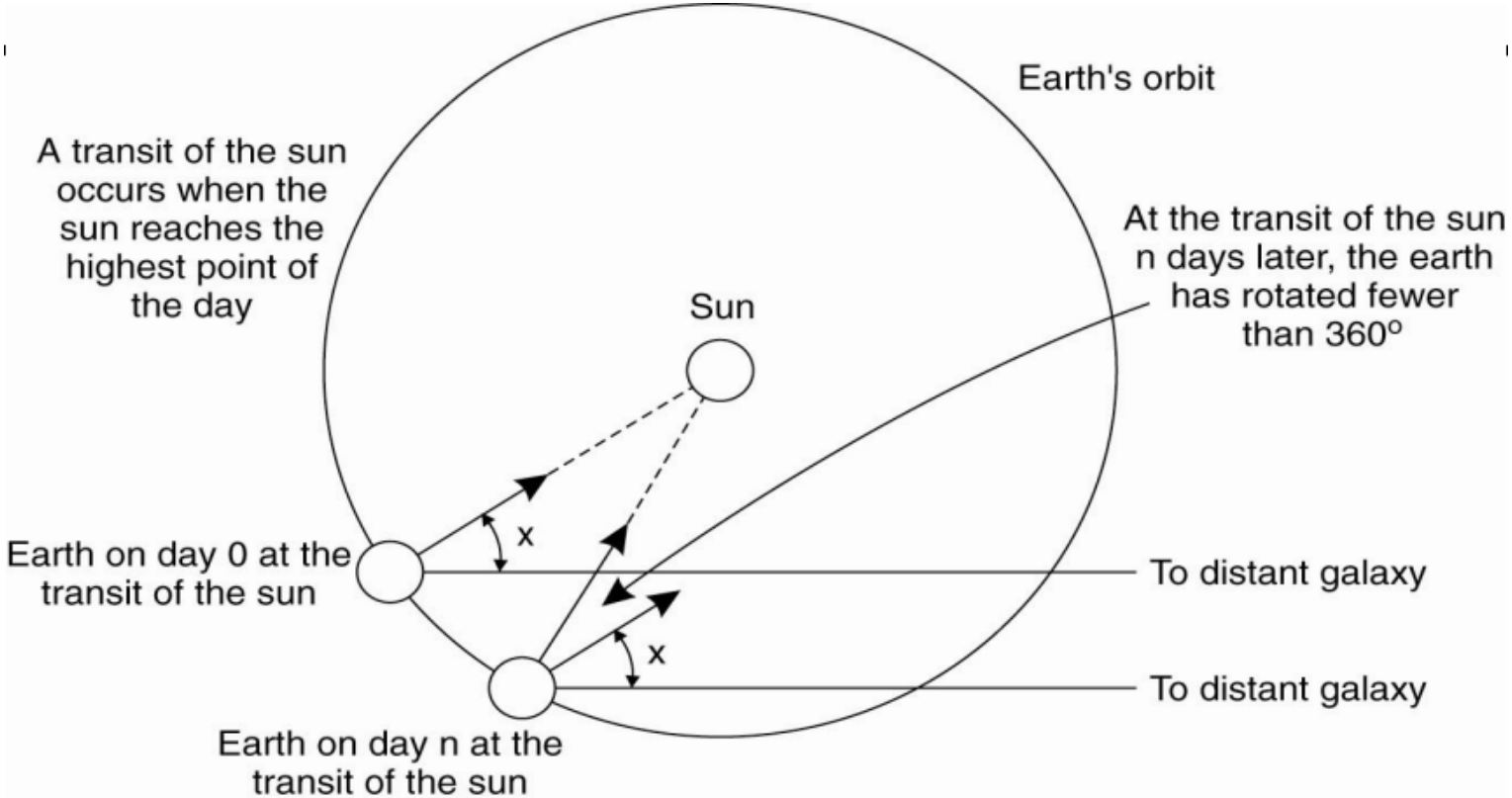



[2]

[2] Tanenbaum Van Steen, Distributed Systems: 7 Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

# Tecnologías para Ordenar Eventos

## Relojes Físicos




Computation of the mean solar day. [3]

[3] Tanenbaum Van Steen, Distributed Systems: 7 Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

# Tecnologías para Ordenar Eventos

## Relojes Lógicos

Propuestos por Leslie Lamport en 1978

# Relojes Lógicos

# Relojes Lógicos

**Si consideramos un evento** *e* **a**:

- La ejecución de una *sentencia interna* de un proceso
- El *envío* de un mensaje
- La *recepción* de un mensaje

# Relojes Lógicos

**Si consideramos un evento** $e$ **a**:

- La ejecución de una *sentencia interna* de un proceso
- El *envío* de un mensaje
- La *recepción* de un mensaje

**Definición**: Dado un proceso $P_i$, *la historia* $\hat{h}_i$ del proceso $P_i$, puede verse como una secuencia de eventos:

- $\hat{h}_i = e_i^1, e_i^2, \ldots, e_i^x, e_i^{x+1} \ldots$

# Relojes Lógicos

**Si consideramos un evento** $e$ **a**:

- La ejecución de una *sentencia interna* de un proceso
- El *envío* de un mensaje
- La *recepción* de un mensaje

**Definición**: Dado un proceso $P_i$, *la historia* $\hat{h}_i$ del proceso $P_i$, puede verse como una secuencia de eventos:

- $\hat{h}_i = e_i^1, e_i^2, \ldots, e_i^x, e_i^{x+1} \ldots$

**Definición**: Dado un conjunto de procesos $p_i$ que forman un SD, cada uno con su historia $\hat{h}_i$, *la historia* $H$ del SD es:

- $H = \cup_{i=1}^{n} \hat{h}_i$

# Relojes Lógicos

**Sea** $H = \cup_{i=1}^{n} h_i$ el conjunto de todos los eventos producidos por un SD. Podemos establecer una **relación de causalidad** entre eventos $e_1 \xrightarrow{ev} e_2$

- **Orden de envío mensaje**: Si $P_i$ envía un mensaje a $P_j$, el evento de enviar el mensaje se produce antes que el de recibirlo.
- **Orden de proceso**: Los eventos internos de un proceso $P_i$, se pueden ordenar, puesto que es un proceso secuencial y tiene un reloj interno.
- **Propiedad transitiva**:
  - $e_1 \xrightarrow{ev} e_2 \wedge e_2 \xrightarrow{ev} e_3 \implies e_1 \xrightarrow{ev} e_3$

# Relojes Lógicos

La **relación** $\xrightarrow{ev}$ se conoce como **happens-before** y fue introducida por *Leslie Lamport* en 1978:

- Da lugar a los **relojes lógicos escalares** de Lamport
- La relación happens-before establece **un orden parcial** en el conjunto de eventos de un SD
- **Sin** tener en cuenta el **tiempo físico**

# Relojes Lógicos

**El objetivo** es **asociar fecha lógica – evento** en un SD

- Sean $e_1$, $e_2$ dos eventos en un SD
  - interno
  - recepción
  - envío
- Sean date($e_1$), date($e_2$) las fechas asociadas a los evento $e_1$, $e_2$
- Entonces **se cumple que**:
  - $\forall e_1, e_2 : e_1 \xrightarrow{ev} e_2 \implies date(e_1) < date(e_2)$

# Relojes Lógicos

El dominio temporal más simple que puede considerarse y que preserva la causalidad es una secuencia creciente de enteros:

- Una fecha es un entero
- Por tanto, cada proceso $P_i$ gestiona una variable local de tipo entero que denominamos reloj lógico (y que inicializamos a 0)
- Operación, intuitivamente:
  - **Justo antes de producir el siguiente evento, se incrementa el reloj.**
  - **El valor del reloj lógico es la fecha del evento**

# Relojes Lógicos

## Reglas de Incremento

**when producing an internal event** $e$ **do**
(1) $clock_i \leftarrow clock_i + 1$. % date of the internal event
(2) Produce event $e$.

**when sending** MSG$(m)$ **to** $p_j$ **do**
(3) $clock_i \leftarrow clock_i + 1$; % date of the send event
(4) send MSG$(m, clock_i)$ to $p_j$.

**when** MSG$(m, h)$ **is received from** $p_j$ **do**
(5) $clock_i \leftarrow \max(clock_i, h)$;
(6) $clock_i \leftarrow clock_i + 1$. % date of the receive event. [4]

[4] Raynal, M. (2013) Distributed Algorithms for Message-Passing Systems

# Relojes Lógicos

**Definición**: Dados dos eventos $e_1$ y $e_2$, se dice que son *concurrentes (o independientes)* si no se puede establecer una relación *happens-before* entre ellos:

- $e_1 \parallel e_2 \stackrel{def}{=} \neg(e_1 \xrightarrow{ev} e_2) \wedge \neg(e_2 \xrightarrow{ev} e_1)$

# Relojes Lógicos

**Propiedad 1:** Sean $e_1$ y $e_2$ dos eventos de un SD:

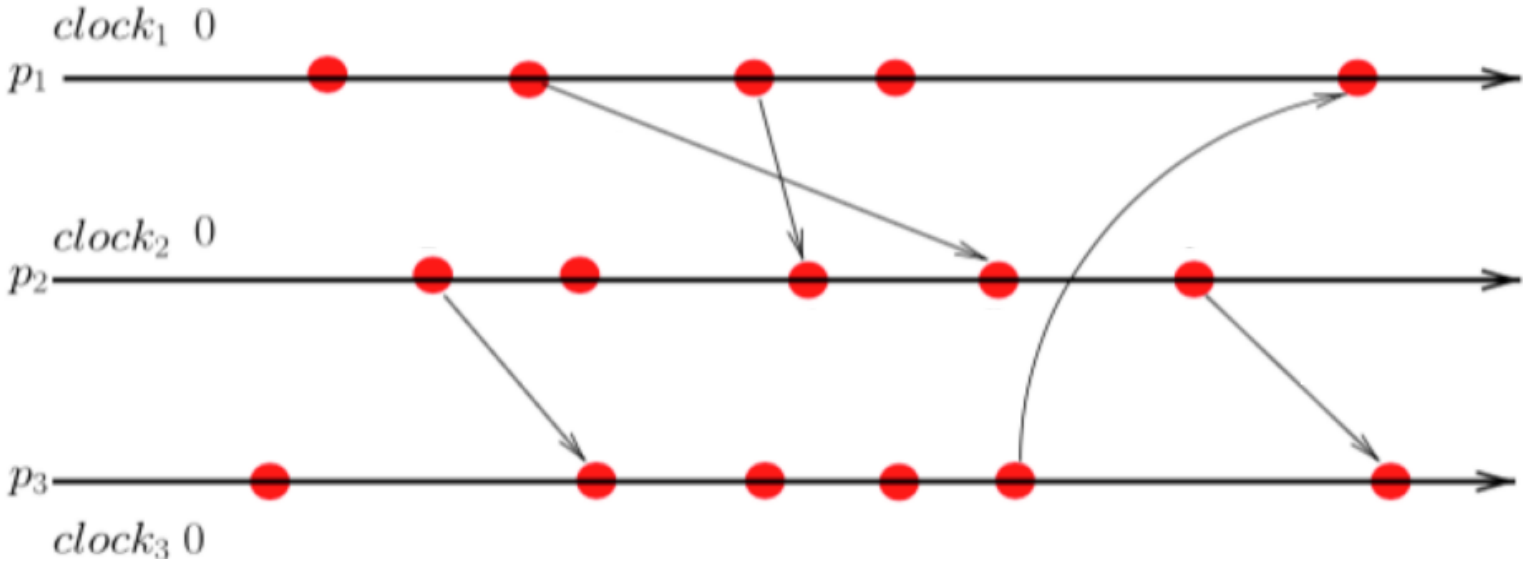
- $date(e_1) \leq date(e_2) \implies \neg(e_2 \xrightarrow{ev} e_1)$

**Propiedad 2:** Sean $e_1$ y $e_2$ dos eventos de un SD:

- $date(e_1) = date(e_2) \implies e_1 \parallel e_2$

# Relojes Lógicos

## Ejercicio

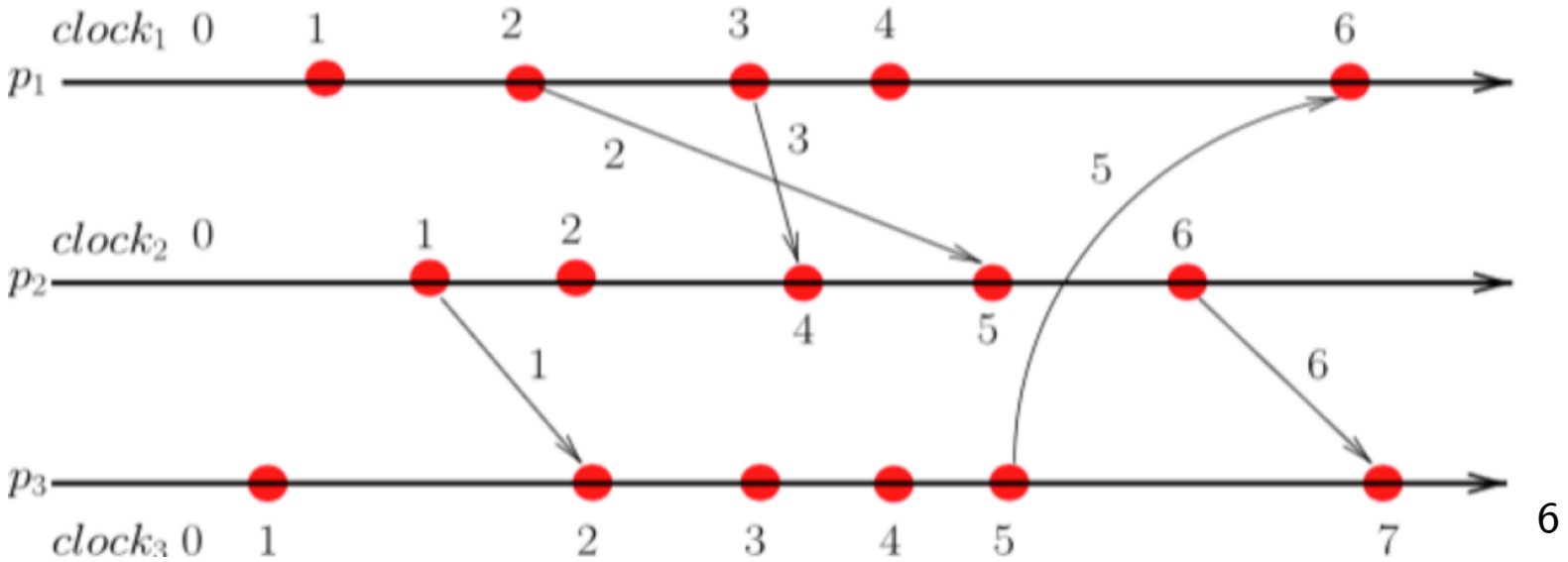



5

[5] Raynal, M. (2013) Distributed Algorithms for Message-Passing Systems

# Relojes Lógicos

## Ejercicio

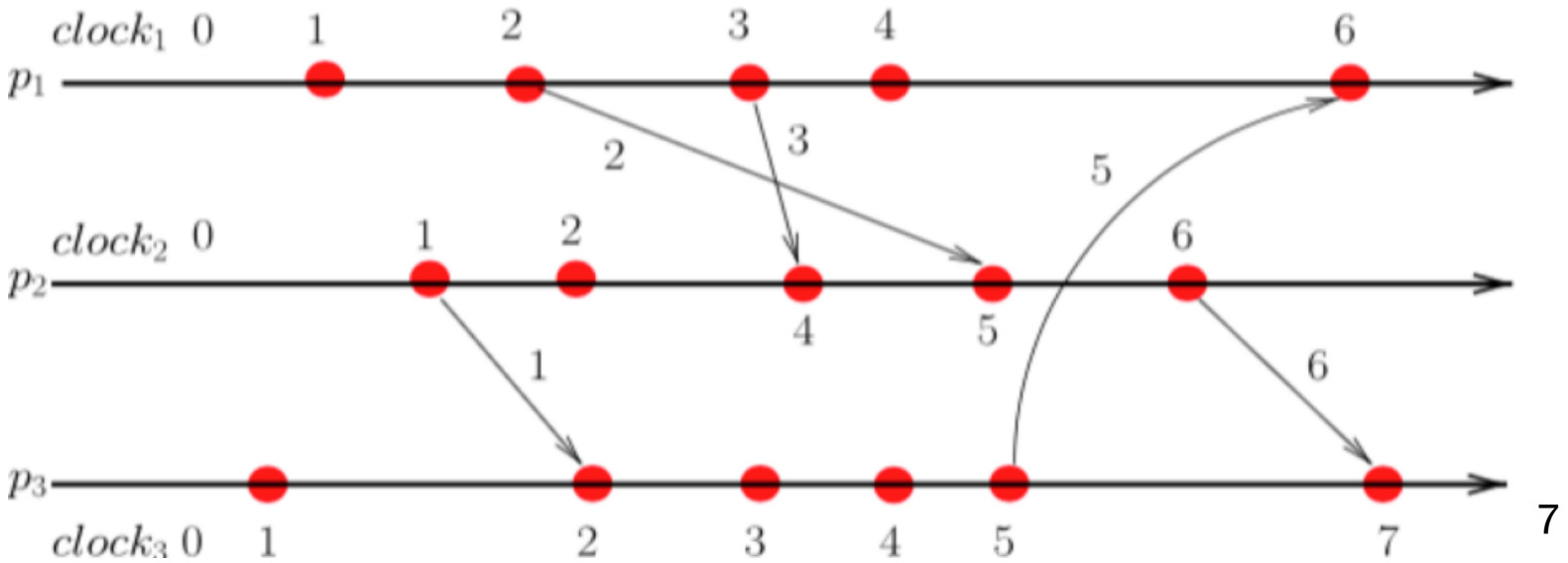



6

[6] Raynal, M. (2013) Distributed Algorithms for Message-Passing Systems

# Relojes Lógicos

## Ejercicio: Encontrar todos los pares de eventos concurrentes




[7] Raynal, M. (2013) Distributed Algorithms for Message-Passing Systems

# Del Orden Parcial al Orden Total

**Con los relojes lógicos puede suceder**

- $date(e_1) < date(e_2) \wedge \neg(e_1 \xrightarrow{ev} e_2)$

**Se puede conseguir un orden total**

- Sobre la relación de orden parcial se va a construir una relación de orden total

# Del Orden Parcial al Orden Total

**Con los relojes lógicos puede suceder**

- $date(e_1) < date(e_2) \wedge \neg(e_1 \xrightarrow{ev} e_2)$

**Se puede conseguir un orden total**

- Sobre la relación de orden parcial se va a construir una relación de orden total

- Vamos a asumir:
  - $date(e_1) < date(e_2) \implies e_1 \xrightarrow{ev} e_2$
  - $date(e_1) = date(e_2) \implies ?$

# Del Orden Parcial al Orden Total

**Relación de Orden Total (relación lexicográfica)**
**Definición**: El *timestamp* de un evento $e$ es un par $(date(e), pid)$, de manera que:

- $date(e)$ es la fecha (valor del reloj) en que $e$ se generó
- pid es el process id del proceso que generó $e$

**Definición**: Sean $e_i$, $e_j$ dos eventos de un SD cuyos timestamps son $(date(e_i), i)$ y $(date(e_j), j)$, respectivamente. La relación lexicográfica de orden total que preserva la relación happens-before se define como:

- $e_i \xrightarrow{to\_ev} e_j \stackrel{def}{=} (date(e_i) < date(e_j)) \vee (date(e_i) = date(e_j)) \wedge (i < j))$

# Resumen

# Resumen

## Concepto de Reloj Lógico

- Eventos: internos, envío, recepción
- Reloj lógico: fecha (entero)
- Asociación evento - fecha y reglas de incremento
- Generamos una relación de orden parcial
- **Al comparar fechas, no hay certeza de si los eventos están relacionados**
- **Podemos convertirla en una relación de orden total**

# Tiempo, Estado y Relojes

## 30221 - Sistemas Distribuidos

**Rafael Tolosana Calasanz**

**Dpto. Informática e Ing. de Sistemas**