# Tema 5: Sincronización en SSDD

En un sistema distribuido no es trivial sincronizar los relojes, esto sucede porque cada nodo que lo forma no tienen la misma velocidad, por ello al final no es tan importante sincronizar los relojes sino ordenar los eventos en el tiempo.

# EXAMEN! Tipos

## Relojes Físicos

Es un dispositivo electrónico que está basado en un cristal de cuarzo y dependiendo de la velocidad a la que oscila se pueden generar interrupciones de forma que somos capaces de estimar el tiempo que ha pasado entre ellos, este tiempo lo utilizamos para marcar eventos en la modificación de ficheros por ejemplo. No existen 2 relojes iguales.

Cuando la hora local del nodo se retrasa de la referencia UTC que tenemos, podemos optar por avanzar el reloj, esta modificación no es grave.

Sin embargo si el reloj se adelanta en referencia a UTC no podemos simplemente retrasar la hora ya que se pueden perder eventos o desincronizarlos, entonces lo que se suele hacer es ralentizar el reloj hasta alcanzar la sincronización reloj = UTC.

### Algoritmo de Cristian (Centralizado)

Suponemos que tenemos un servidor de tiempo al cual el cliente consulta para saber el tiempo actual y pone la hora para saber el tiempo actual, el problema con esto es el tiempo de transmisión y computo de este mensaje que también es importante, para ello se estima el RTT del envió y recepción y se calcula la hora en el cliente de forma que es igual  al $t_{servidor} + \frac{RTT}{2}$

### Algoritmo de Berkeley (Centralizado)

Se basa en elegir un nodo coordinador el cual envía al resto de nodos su hora, los nodos responden enviándole su deriva (el desfase que tienen en comparación de la hora del coordinador), por último el coordinador

haciendo un cálculo (donde él también puede adelantar su hora) envía el tiempo que tienen que retrasar o adelantar a cada nodo.

diapo14

***Ventaja***: *Es más tolerante a fallos*.

### Algoritmo Descentralizado

Todos los equipos se sincronizan entre ellos sin un coordinador, se hace un broadcast de cada uno al resto de nodos, cuando todos tienen todos los mensajes (o se vence un timer) se calcula el promedio y se ajusta el reloj local adelantando o retrasando.

***Ventaja***: *Es más tolerante a fallos*.

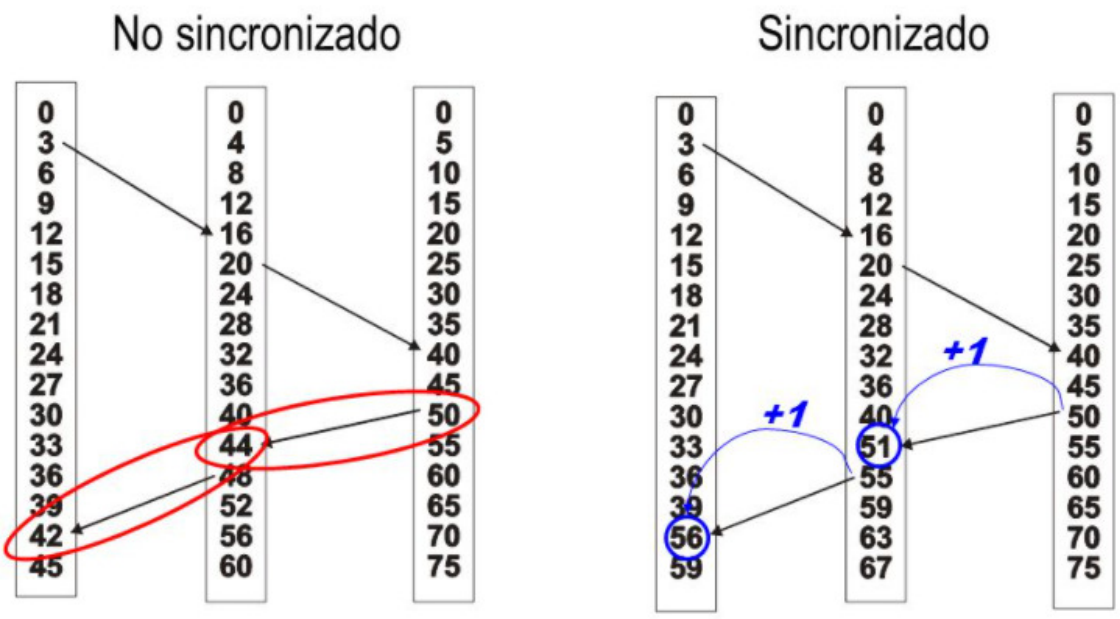
***Desventaja***: *Hay más saturación de la red*.

## Relojes Lógicos

Es básicamente un contador **creciente** que se utiliza para ordenar los eventos que suceden en el sistema distribuido (de forma que el evento A ocurre antes que B). **Es más importante saber qué evento ocurre antes que otro.**

### Relojes lógicos de Lamport

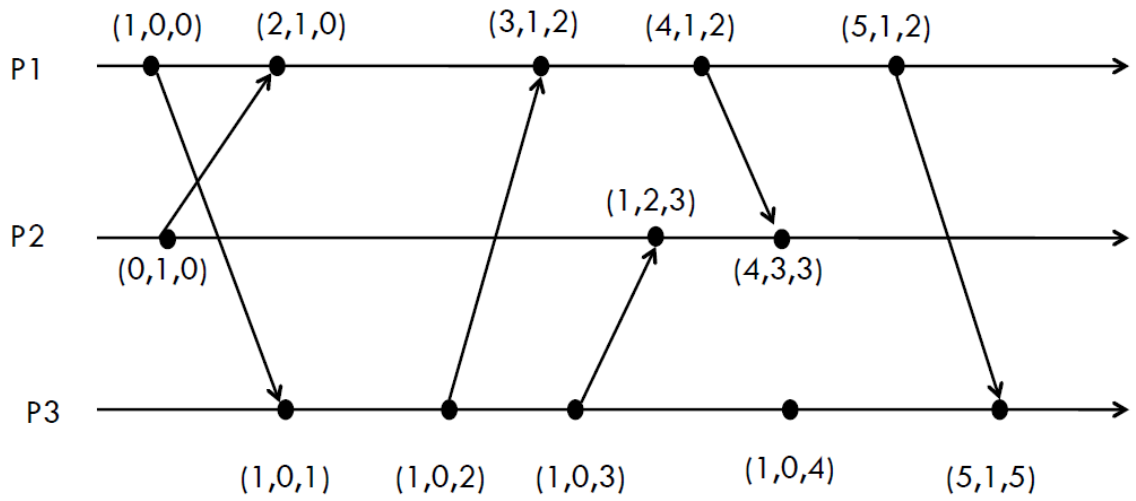
Funciona de tal forma que si enviamos un mensaje con un tiempo de emisor mayor que el tiempo que tiene el receptor entonces el receptor anula su reloj lógico y establece un tiempo superior al de el emisor. Esto se realiza para evitar inconsistencias.




Aún así no podemos saber si el evento realmente sucedió antes que otro, sabemos que sucedió en un reloj con un tiempo anterior o posterior pero como cada nodo tiene su propio contador, este puede ir muy lento pero sacar antes o muy rápido al proceso para enviarlo después o al revés. Para aliviar esto se crearon los ***relojes vectoriales***.

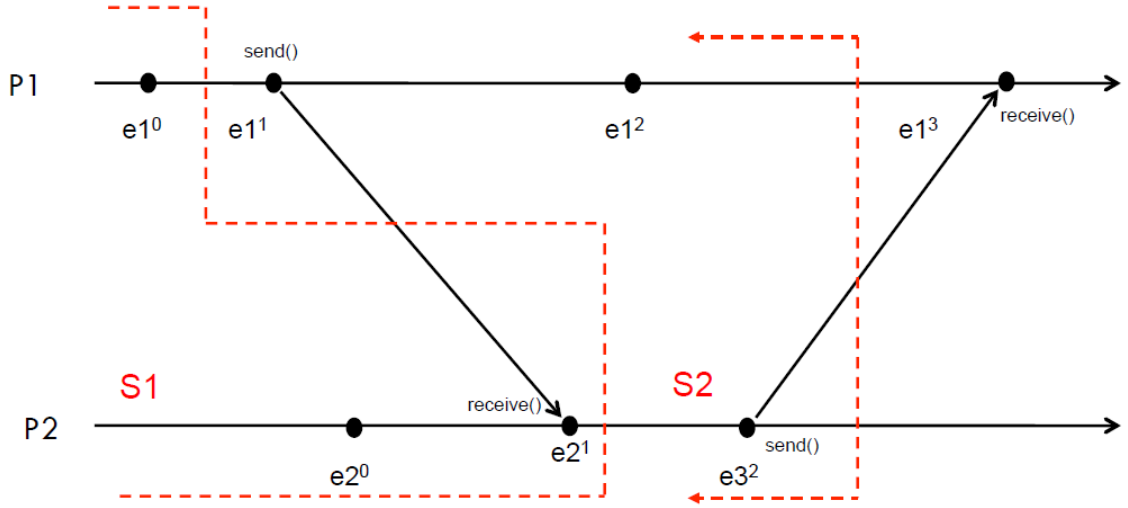
## Relojes Vectoriales

Son relojes que nos permiten determinar en un vector el proceso que está realizado y su orden de tal forma que tenemos por ejemplo $(x, y, z)$ donde $x$ pertenecería al primer proceso, $y$ al segundo y $z$ al tercero y cada proceso tendría este vector empezando con un 1 en su posición, es decir para el proceso 1 empezaríamos con el valor $(1, 0, 0)$, para el 2 $(0, 1, 0)$ y el 3 $(0, 0, 1)$; cada vez que enviamos un mensaje se va sumando 1 a cada envío y concatenando el valor de nuestro emisor.




## Estados

Se basa en la idea que un envió no puede estar en un estado posterior al recibimiento del mensaje (No podemos enviar un paquete a alguien mañana y recibirlo hoy), pero si que se puede enviar en un estado X y recibirlo en el siguiente estado X+1.




**S1** es un estado **inconsistente**
**S2** es un estado **consistente**