# IIC2523
# Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 05)

# Coordinación y sincronización en un Sistema Distribuido

# Temas de la clase

1. Coordinación en un Sistema Distribuido
    a. Necesidades
    b. Tipos de algoritmos
2. Sincronización de relojes
    a. Introducción
    b. Algoritmos de sincronización.
        i. Uso de Reloj Físicos (Cristian, Berkeley, NTP)
        ii. Uso de Reloj Lógicos (Lamport)

# Coordinación en un Sistema Distribuido

¿Por qué es necesario?

Tipos de Coordinación/algoritmos

# Coordinación en un Sistema Distribuido

- Se refiere a cómo los procesos cooperan entre sí.
- Mecanismos para **gestionar las interacciones y dependencias** entre actividades en un sistema distribuido para **asegurar** que una o más **tareas se logre exitosamente**.

# Coordinación en un Sistema Distribuido - Tipos

- Estos mecanismos se pueden catalogar según el objetivo que buscan asegurar.
- En esta unidad estudiaremos 4 tipos:
  - **Sincronización de relojes**: asegurar que los nodos tengan una noción común del tiempo.
  - **Algoritmos de Consenso**: asegurar que los nodos estén de acuerdo con una operación.
  - **Elección de líder**: asegurar que todos los nodos estén de acuerdo con un rol "superior" (o de coordinación) de otro nodo.
  - **Exclusión Mutua**: asegurar el acceso único para un recurso.

# Coordinación en un Sistema Distribuido - Tipos

- Estos mecanismos se pueden catalogar según el objetivo que buscan asegurar.
- En esta unidad estudiaremos 4 tipos:
  - **Sincronización de relojes**: asegurar que los nodos tengan una noción común del tiempo.
  - **Algoritmos de Consenso**: asegurar que los nodos estén de acuerdo con una operación.
  - **Elección de líder**: asegurar que todos los nodos estén de acuerdo con un rol "superior" (o de coordinación) de otro nodo.
  - **Exclusión Mutua**: asegurar el acceso único para un recurso.
- Existen algoritmos que ocupan más de un mecanismo para asegurar su tarea.
  - Algoritmo ***Raft*** tiene una fase de elección de líder para luego pasar a una fase de consenso de información.
  - Algoritmo de **Berkeley** primero elige un líder para luego sincronizar tiempos.

# Sincronización de relojes

Introducción

Algoritmos de sincronización

# Sincronización de relojes

- Los sistemas distribuidos, por su naturaleza, operan con relojes físicos individuales que pueden diferir con el tiempo debido a variaciones en el *hardware* y otros factores.
  - Es posible que un evento que ocurrió después de otro tenga asignada una hora anterior, lo que puede tener efectos drásticos.

# Sincronización de relojes

- Los sistemas distribuidos, por su naturaleza, operan con relojes físicos individuales que pueden diferir con el tiempo debido a variaciones en el *hardware* y otros factores.
  - Es posible que un evento que ocurrió después de otro tenga asignada una hora anterior, lo que puede tener efectos drásticos.
- La sincronización busca mitigar estas diferencias para asegurar un orden temporal y consistente de los eventos.
  - Si el nodo está atrasado, adelanta su reloj.
  - No obstante, si un nodo está adelantado, este no suele retroceder sus relojes para evitar problemas de consistencia. Solo hacen más lento su reloj para alcanzar a los demás.

# Sincronización de relojes

- Los sistemas distribuidos, por su naturaleza, operan con relojes físicos individuales que pueden diferir con el tiempo debido a variaciones en el *hardware* y otros factores.
  - Es posible que un evento que ocurrió después de otro tenga asignada una hora anterior, lo que puede tener efectos drásticos.
- La sincronización busca mitigar estas diferencias para asegurar un orden temporal y consistente de los eventos.
  - Si el nodo está atrasado, adelanta su reloj.
  - No obstante, si un nodo está adelantado, este no suele retroceder sus relojes para evitar problemas de consistencia. Solo hacen más lento su reloj para alcanzar a los demás.

**¿Qué problema se les ocurre si se retrasa el reloj? 🤔**

# Sincronización de relojes

La sincronización se puede realizar en el reloj físico o en un reloj lógico.

# Sincronización de relojes - Reloj Físico

La sincronización se puede realizar en el **reloj físico** o en un reloj lógico.

- Los dispositivos electrónicos que cuentan las oscilaciones de un cristal a una frecuencia definida, almacenando el resultado en un contador.
  - Los relojes de *hardware* basados en **cuarzo** tienen una tasa de deriva (*drift rate*) de aproximadamente $10^{-6}$ segundos por segundo, o 31.5 segundos por año.
  - Los relojes **atómicos**, basados en transiciones de átomos de cesio, ofrecen una precisión aún mayor.

# Sincronización de relojes - Reloj Físico

La sincronización se puede realizar en el **reloj físico** o en un reloj lógico.

- Los dispositivos electrónicos que cuentan las oscilaciones de un cristal a una frecuencia definida, almacenando el resultado en un contador.
  - Los relojes de *hardware* basados en **cuarzo** tienen una tasa de deriva (*drift rate*) de aproximadamente $10^{-6}$ segundos por segundo, o 31.5 segundos por año.
  - Los relojes **atómicos**, basados en transiciones de átomos de cesio, ofrecen una precisión aún mayor.
- Cada cierto tiempo, es necesario actualizar el contador entre computadores para evitar la descrincronización.

# Sincronización de relojes - Reloj Lógico

La sincronización se puede realizar en el reloj físico o en un **reloj lógico**.

- Se utilizan cuando no es necesaria una cuenta precisa del tiempo real absoluto, solo un mecanismo que ayude a ordenar los eventos.

# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport

- Permiten establecer un orden parcial a los distintos sucesos o eventos.
  - Determinar el orden causal de eventos.
  - Identificar eventos concurrentes.
- Con eventos concurrentes nos referimos a que no sabemos si un evento ocurrió antes, después o al mismo tiempo que otro evento.

# Sincronización de relojes - Algoritmos (Reloj Lógico)

**Relojes Lógicos de Lamport**

- Es un contador de *software* que aumenta monótonamente.
- Cada proceso $P_i$ mantiene su propio reloj lógico que denotaremos $C(P_i)$.
- El objetivo es inferir el orden en que se presentan los mensajes sin necesidad de recurrir a relojes físicos perfectamente sincronizados.

# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport

- Lamport definió la relación "sucedió-antes" (denotada por "→"). Esta relación se define mediante las siguientes propiedades:
  - **Regla 1:** Si dos eventos (a y z) ocurren en el proceso $P_i$ y "a" ocurre antes de "z", entonces a → z.
  - **Regla 2:** Para cualquier mensaje `m`, el evento de envío `send(m)` ocurre antes que el evento de recepción `receive(m)`
    - Es decir, `send(m)` → `receive(m)`.

# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport

- Para garantizar estas reglas, los contadores cumplen lo siguiente.
  - Sea $C(P_z)$ el contador del proceso $P_z$
  - **Regla 1:**
    - Cada vez que ocurre un evento en $P_z$ $C(P_z) \mathrel{+}= 1$
    - El evento se registra con el valor del reloj $C(P_z)$.
  - **Regla 2:**
    - Si $P_i$ manda mensaje a $P_z$ se debe adjuntar $C(P_i)$
    - Se actualiza $C(P_z)$ $C(P_z) = \max(C(P_i), C(P_z)) + 1$
    - El evento se registra con el valor del reloj $C(P_z)$.

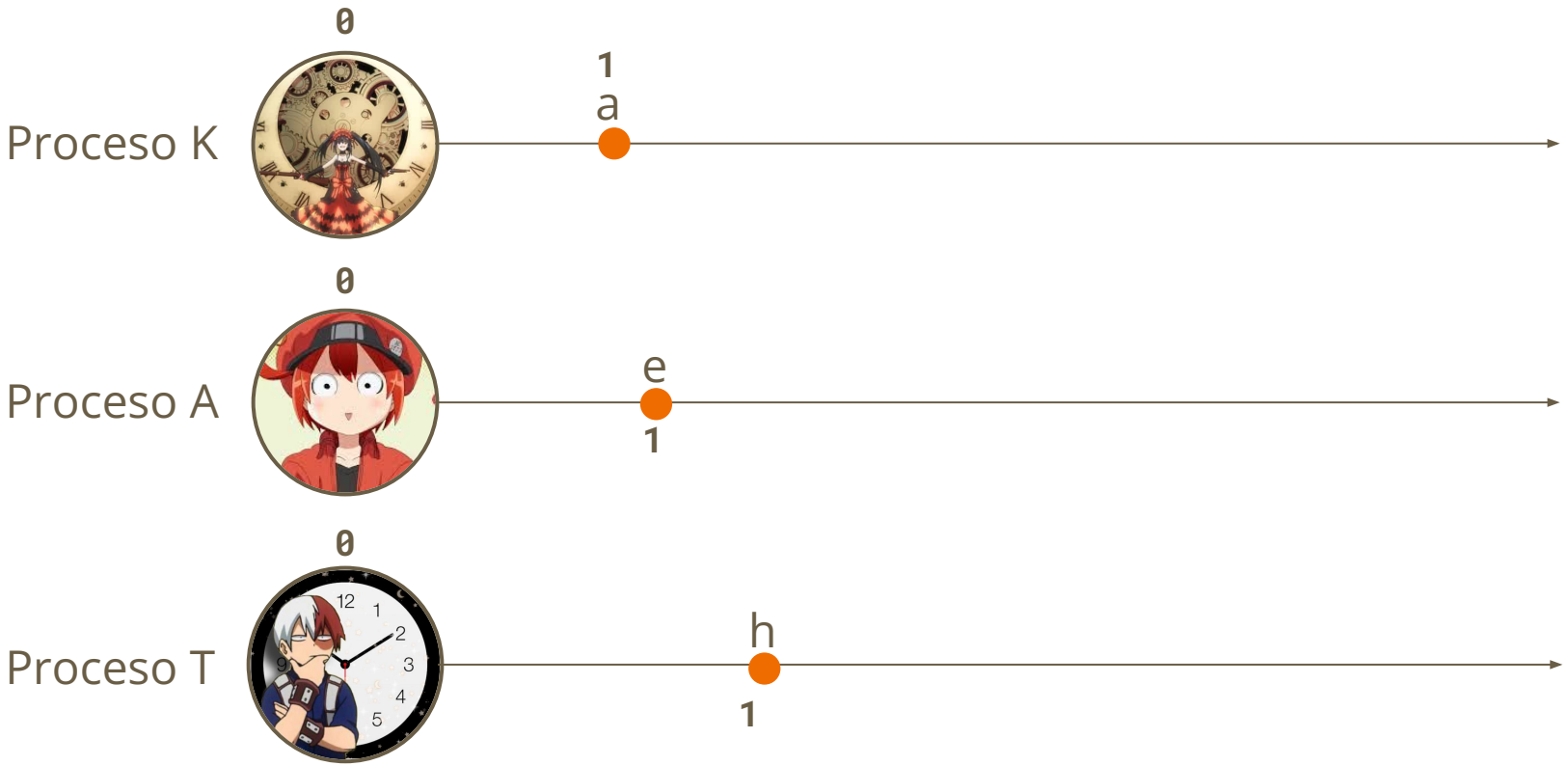
# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport

- Para garantizar estas reglas, los contadores cumplen lo siguiente.
  - Sea $C(P_z)$ el contador del proceso $P_z$
  - **Regla 1:**
    - Cada vez que ocurre un evento en $P_z$ `C(Pz) += 1`
  - **Regla 2:**
    - Si $P_i$ manda mensaje a $P_z$ se debe adjuntar $C(P_i)$
    - Se actualiza $C(P_z)$ `C(Pz) = max(C(Pi), C(Pz)) + 1`
- Con la implementación anterior, si a $\rightarrow$ b entonces el contador al momento de ocurrir **a** será menor al momento de ocurrir **b**.

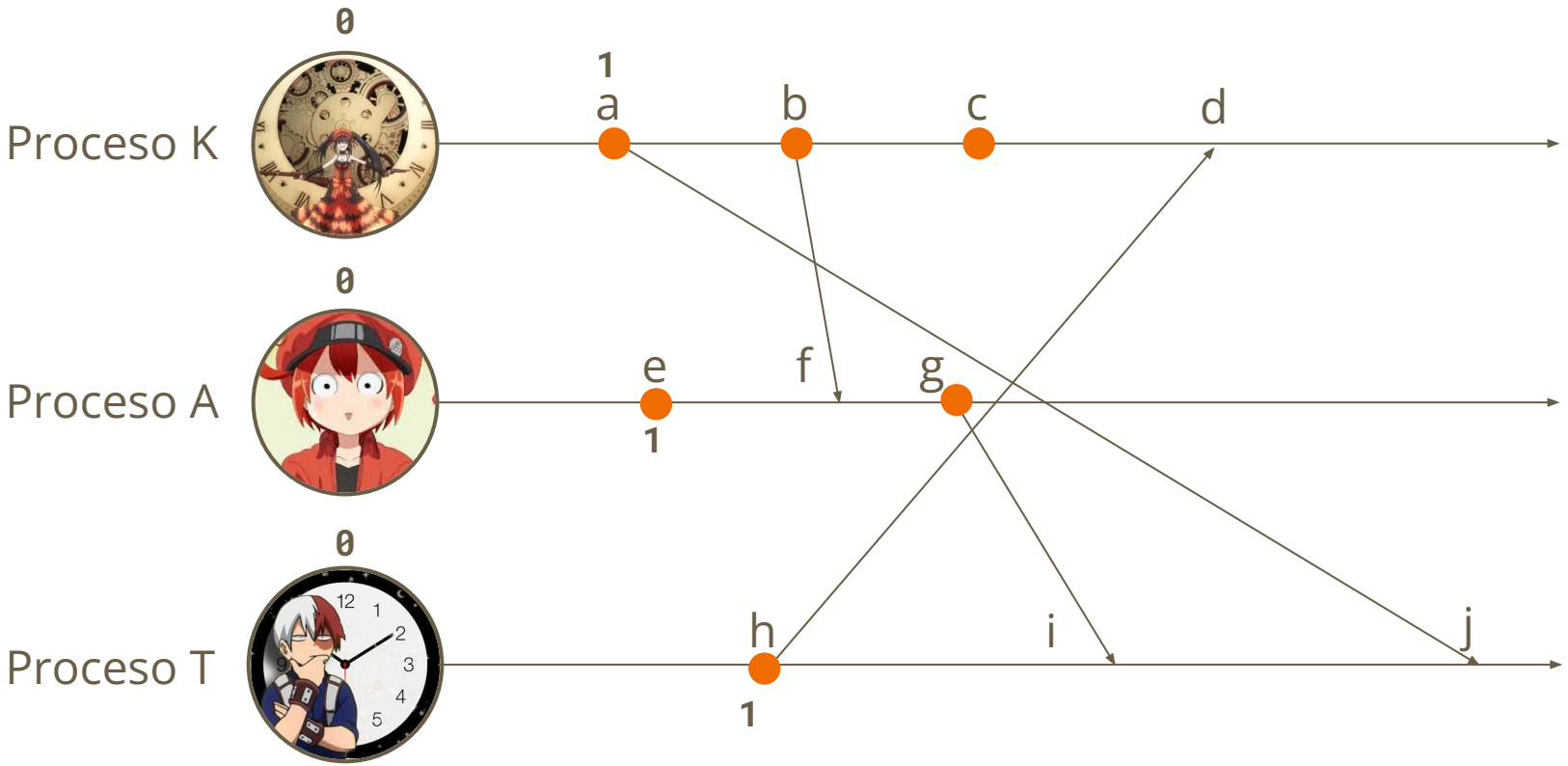
# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport



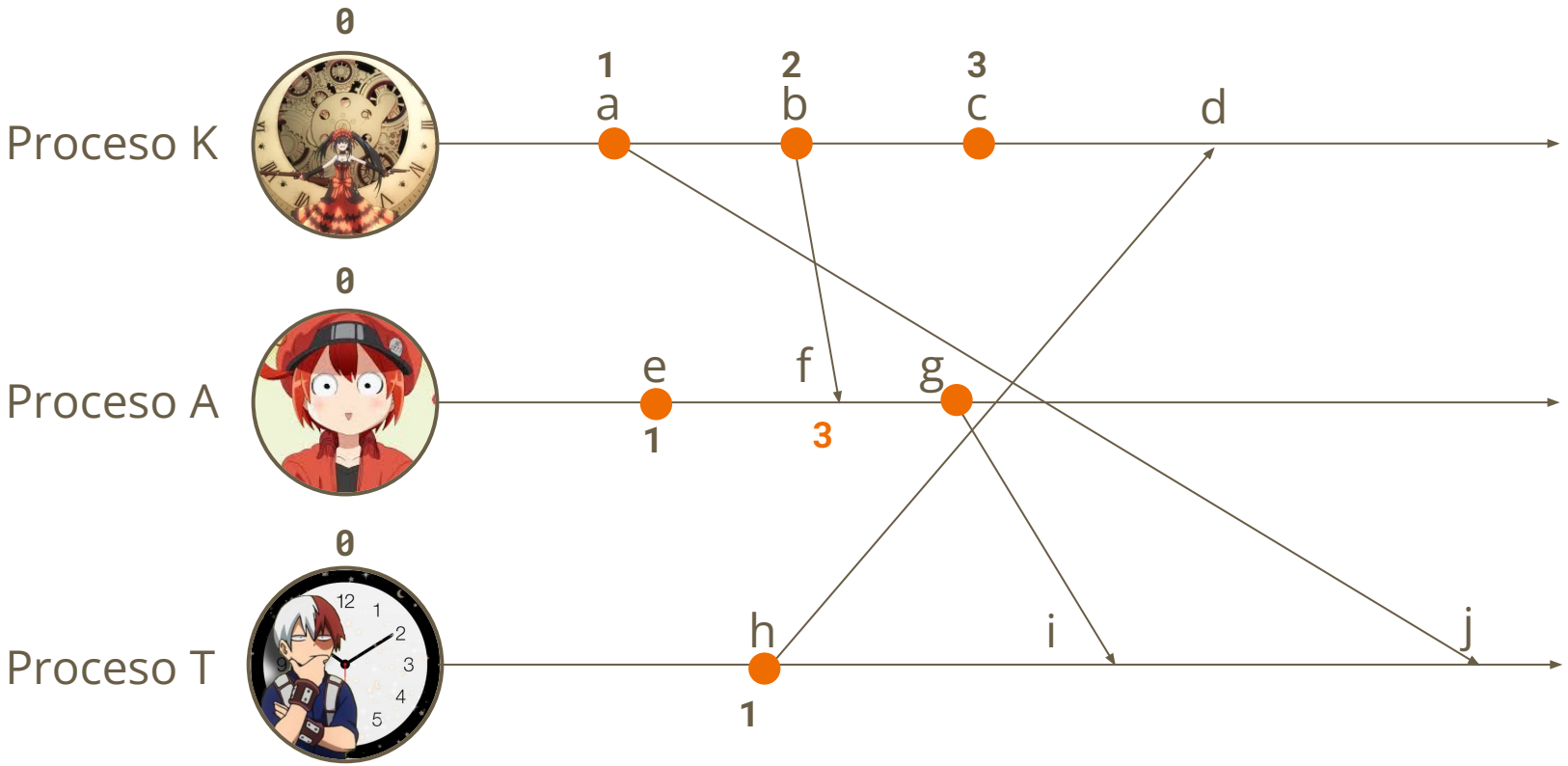
# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport



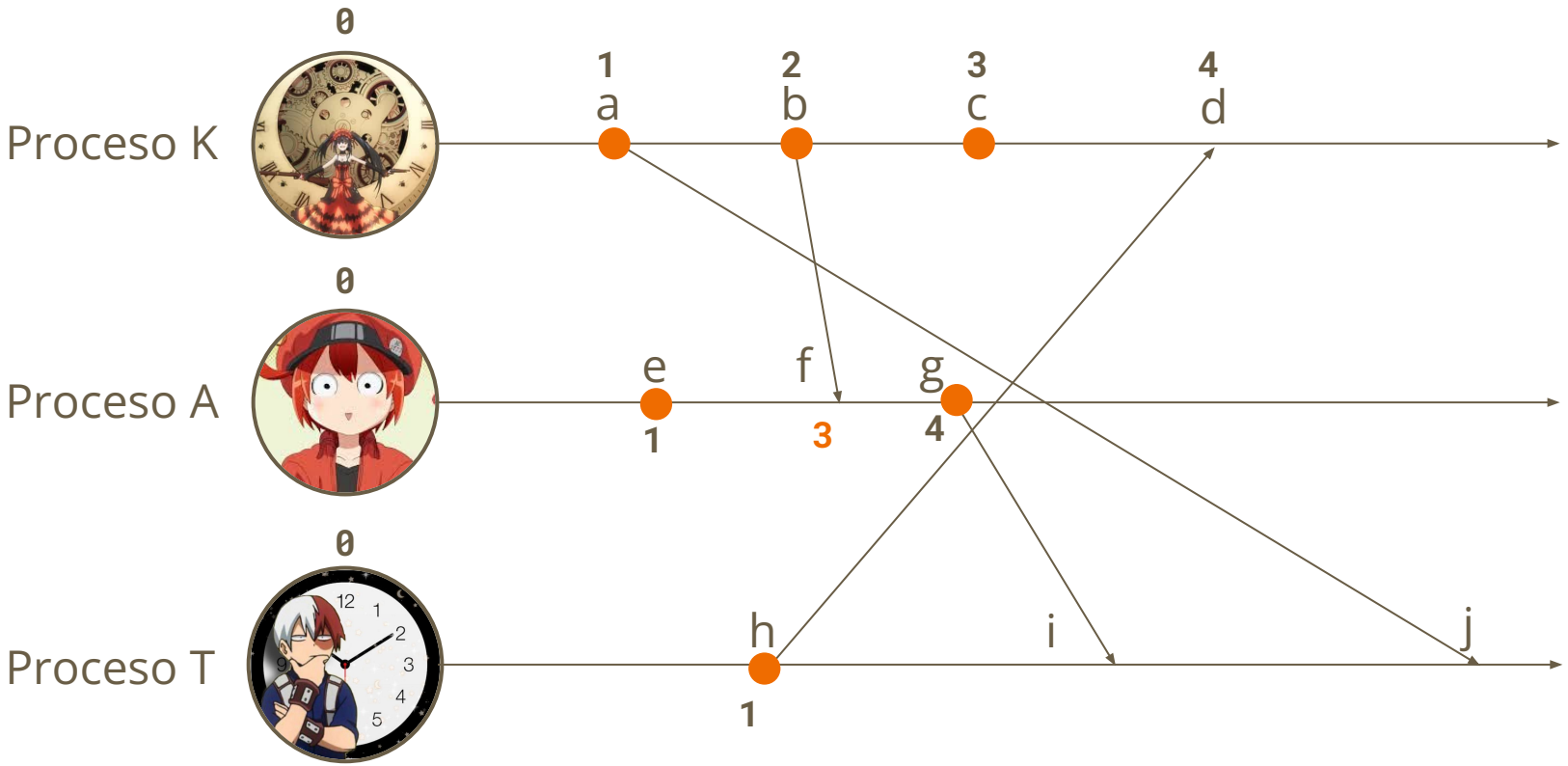
# Sincronización de relojes - Algoritmos (Reloj Lógico)

**Relojes Lógicos de Lamport**



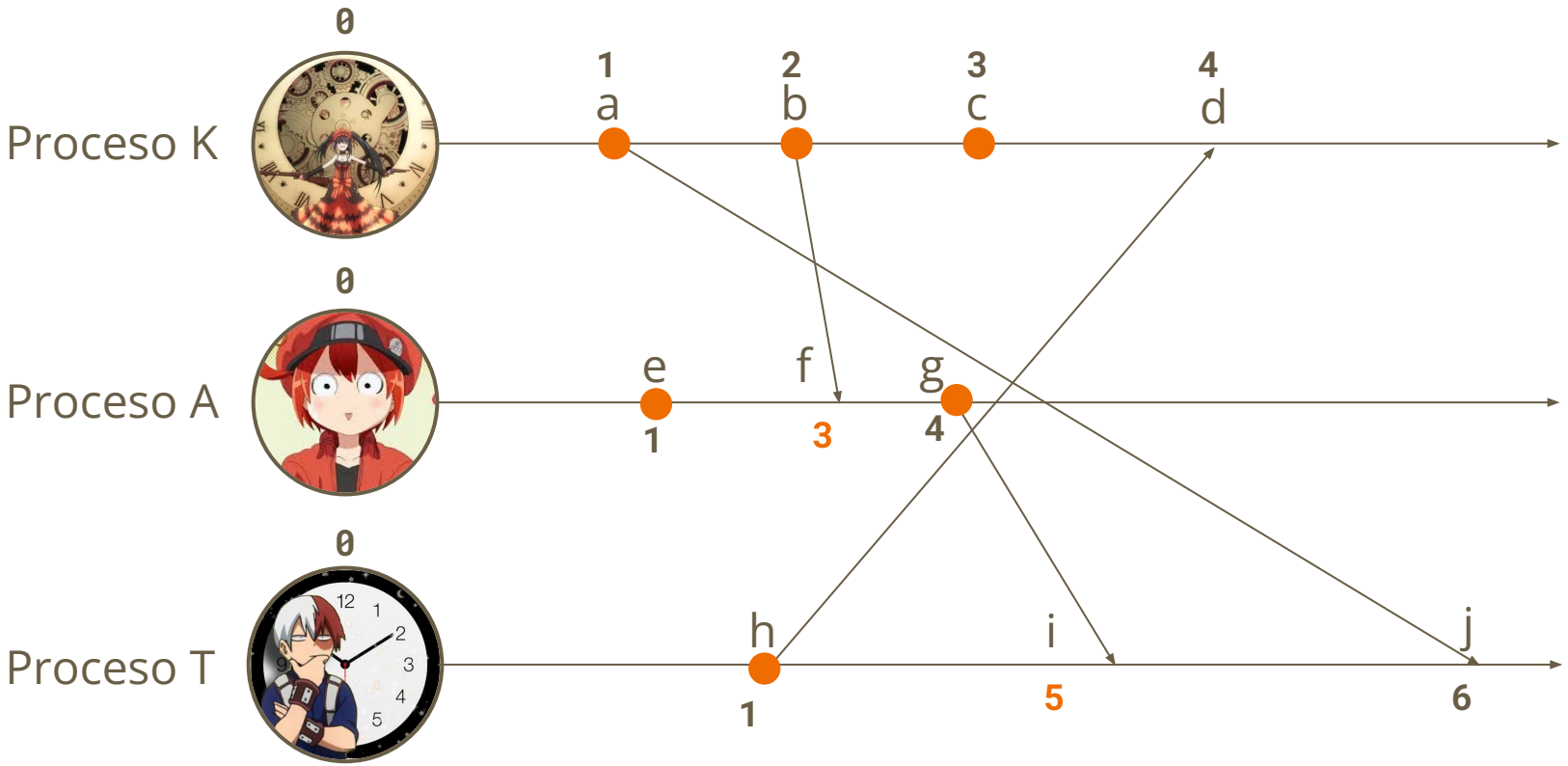
# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport



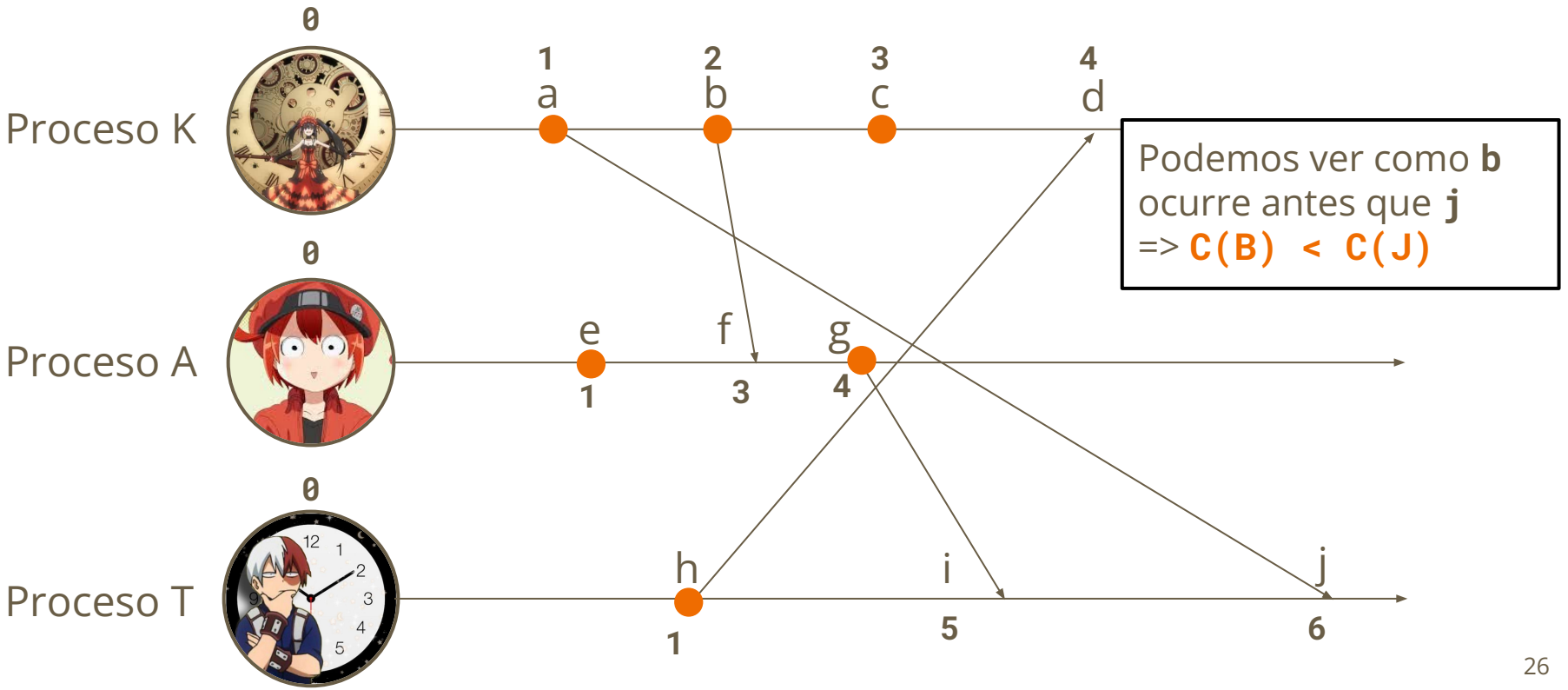
# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport



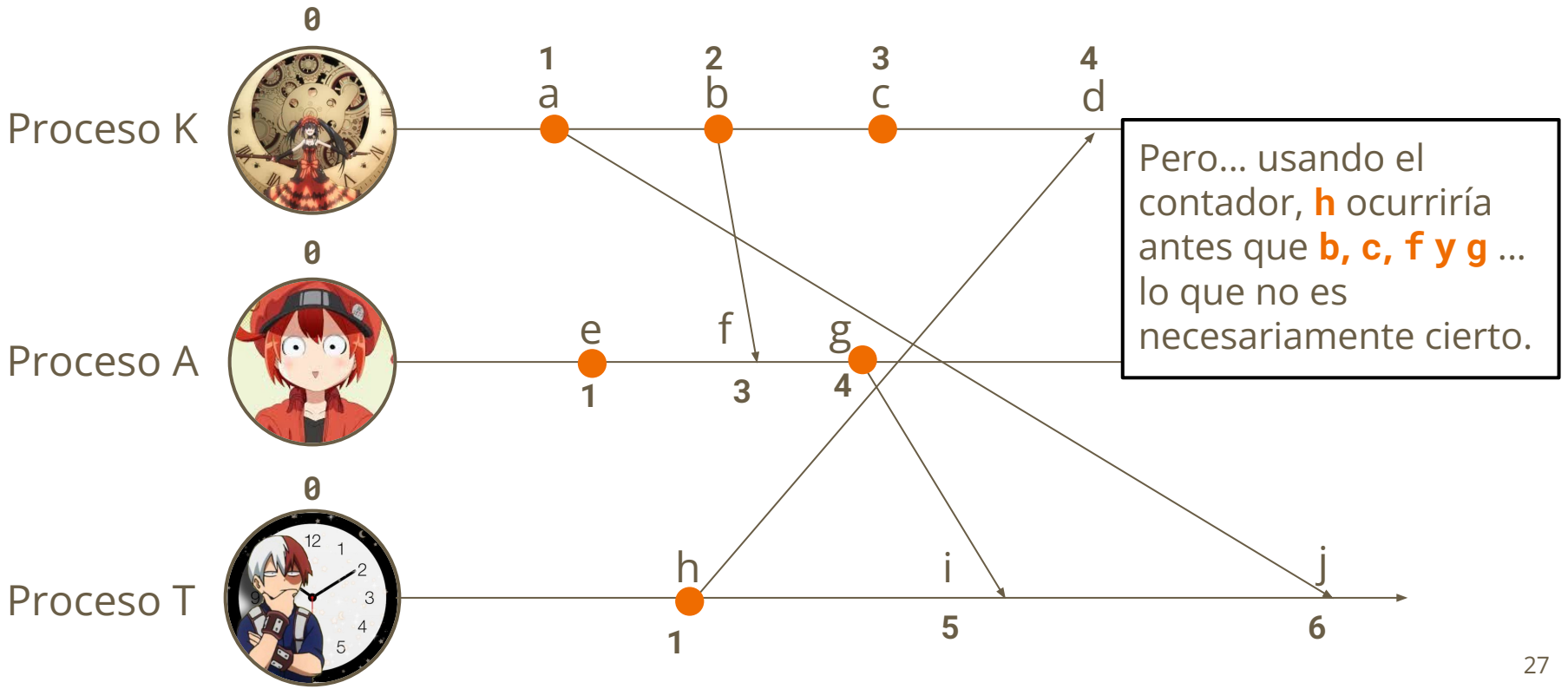
# Sincronización de relojes - Algoritmos (Reloj Lógico)

**Relojes Lógicos de Lamport**



# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport



# Sincronización de relojes - Algoritmos (Reloj Lógico)

## Relojes Lógicos de Lamport

- Con la implementación anterior, si a → b entonces el contador al momento de ocurrir **a** será menor al momento de ocurrir **b**.
- Esta relación **es solo en una dirección**, si el contador al momento de ocurrir **a** es menor al momento de ocurrir **b**, no podemos concluir que a → b.
  - **No podemos usar el contador para confiar plenamente en la causalidad.**
- Nos permite generar un **orden parcial** de los elementos en función de su contador, pero no permite garantizar que ese orden refleje fielmente la realidad.

# Sincronización de relojes - Algoritmos (Reloj Lógico)

**Relojes Lógicos de Lamport**

- Con la implementación anterior, si a → b entonces el contador al momento de ocurrir **a** será menor al momento de ocurrir **b**.
- Esta relación **es solo en una dirección**, si el contador al momento de ocurrir **a** es menor al momento de ocurrir **b**, no podemos concluir que a → b.
  - **No podemos usar el contador para confiar plenamente en la causalidad.**
- Nos permite generar un **orden parcial** de los elementos en función de su contador, pero no permite garantizar que ese orden refleje fielmente la realidad.

**La próxima clase vamos a ver un mecanismo de Reloj Lógico que si permita abordar este problema de unidireccionalidad.**

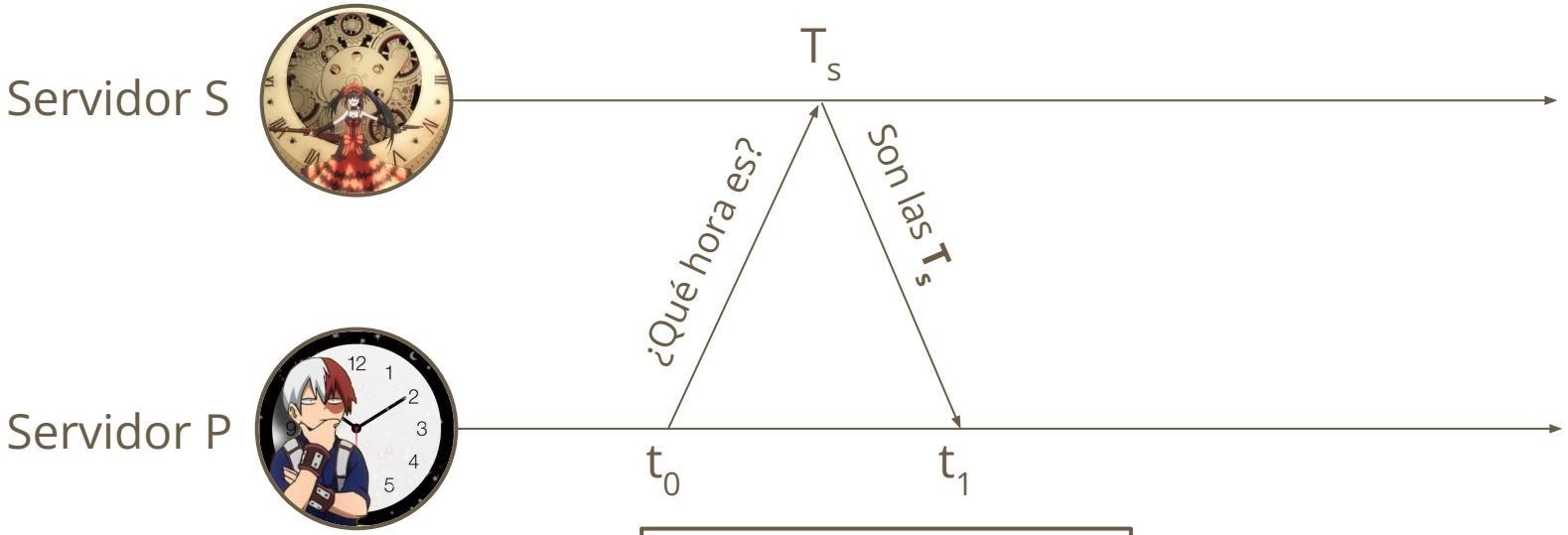
# Sincronización de relojes - Algoritmos (Reloj Físico)

**Método de Cristian**

- Un servidor ajusta su reloj a lo que diga el otro servidor.
- Se asume que el otro servidor tiene un mejor reloj, por ejemplo, sincronizado a UTC (Tiempo universal coordinado) o con reloj atómico.
- Entiende que existe un *delay* de comunicación, que asume que es el mismo de ida que de regreso.
- Asume que una vez recibe el mensaje, el otro nodo responde de inmediato.
- Con la respuesta del servidor y el *delay*, **estima la hora que tendría el servidor y la utiliza para sobreescribir su propia hora.**

# Sincronización de relojes - Algoritmos (Reloj Físico)

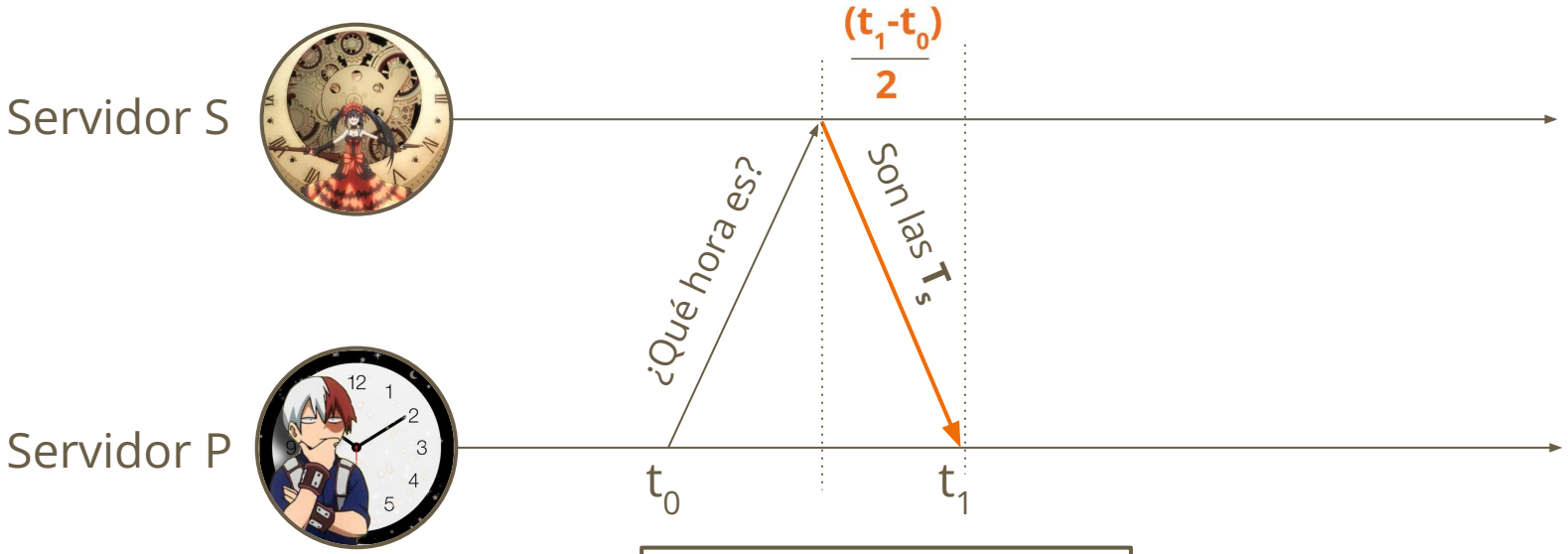
**Método de Cristian**




$$T_{servidor\ p} = T_s + \frac{(t_1 - t_0)}{2}$$

# Sincronización de relojes - Algoritmos (Reloj Físico)

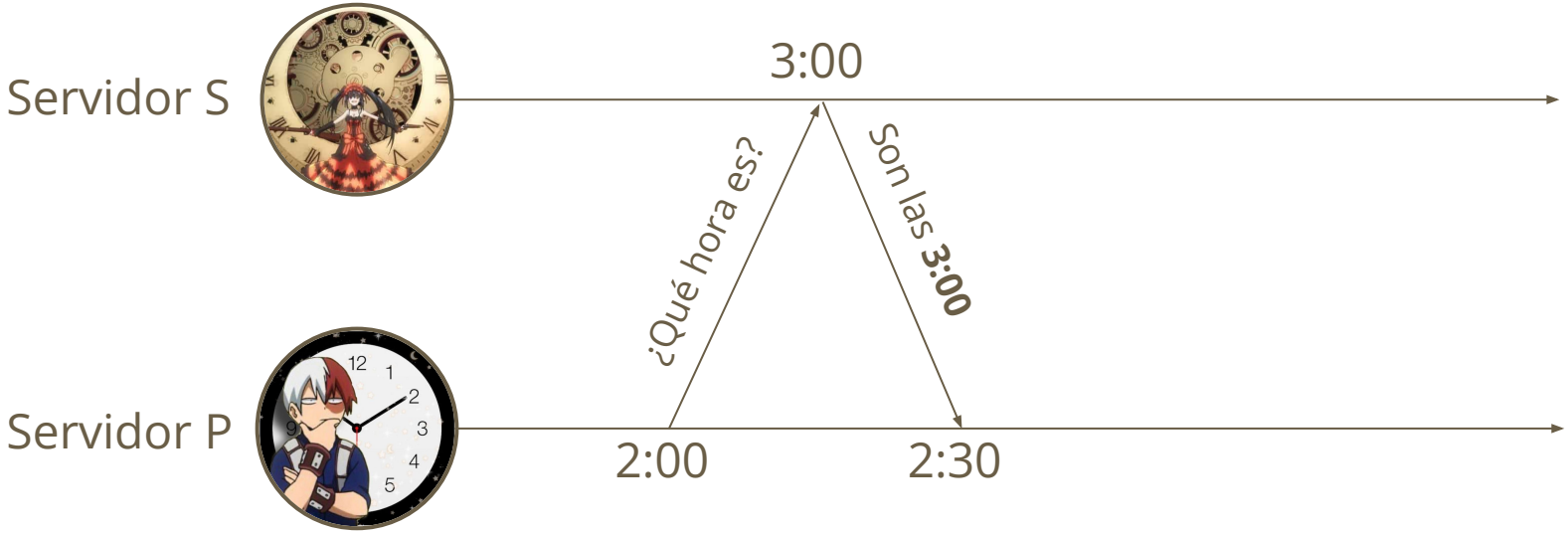
**Método de Cristian**




$$T_{servidor\ p} = T_s + \frac{(t_1-t_0)}{2}$$

# Sincronización de relojes - Algoritmos (Reloj Físico)

**Método de Cristian**




$T_{servidor\ p} = 3:00 + \frac{(2:30-2:00)}{2} = 3:15$

# Sincronización de relojes - Algoritmos (Reloj Físico)

**Algoritmo de Berkeley**

- Se creó para entornos en los cuales no se tienen receptores de tiempo UTC.
- Se selecciona, entre los nodos conectados, a un **nodo líder** mientras que los demás serán considerados los **nodos seguidores**.
  - Dato *freak*: hay literatura donde se ocupa "nodo maestro" y "nodo esclavos".
- El nodo líder se encarga de definir una hora para que él y los nodos seguidores la utilicen.

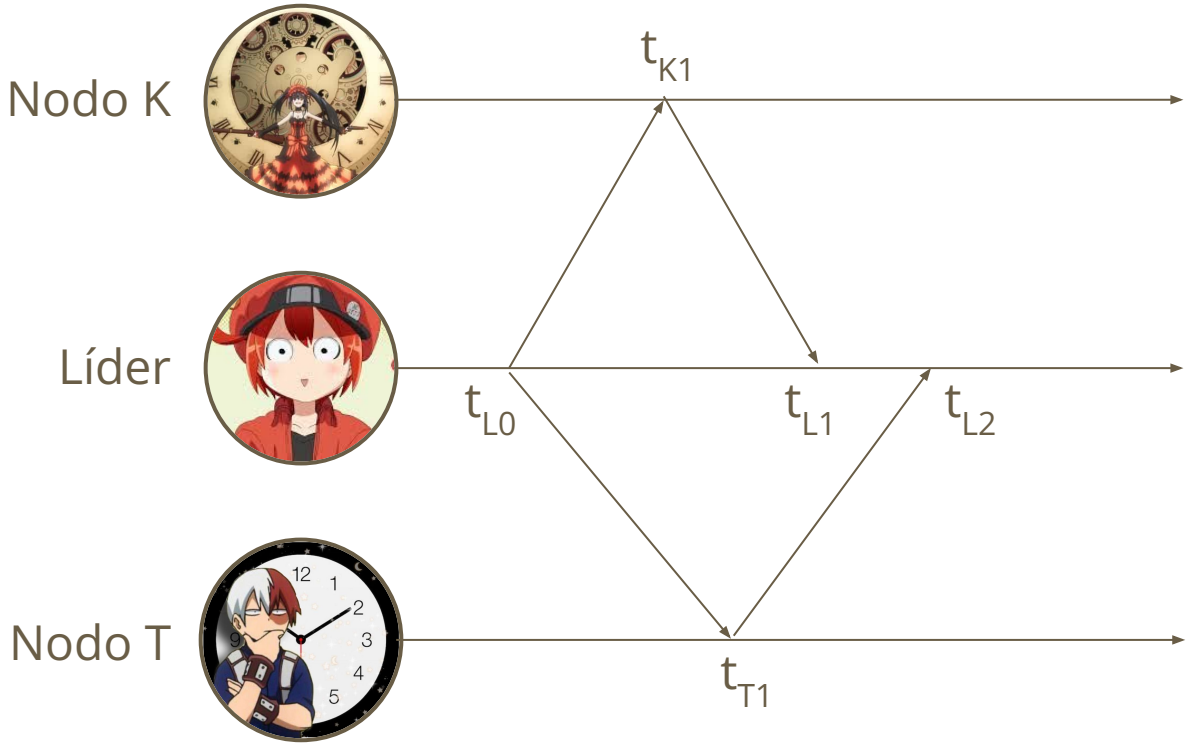
# Sincronización de relojes - Algoritmos (Reloj Físico)

**Algoritmo de Berkeley - Funcionamiento**

1. El líder aplica el método de Cristian para estimar la hora actual de cada nodo seguidor considerando el *delay* de comunicación.
2. Calcula el promedio entre las horas de todos los nodos, incluido el líder
3. Líder ajusta su reloj al promedio calculado.
4. Se manda la diferencia entre la hora estimada y la hora promedio a cada nodo.
5. Los nodos aplican ajuste a sus relojes según la diferencia indicada.

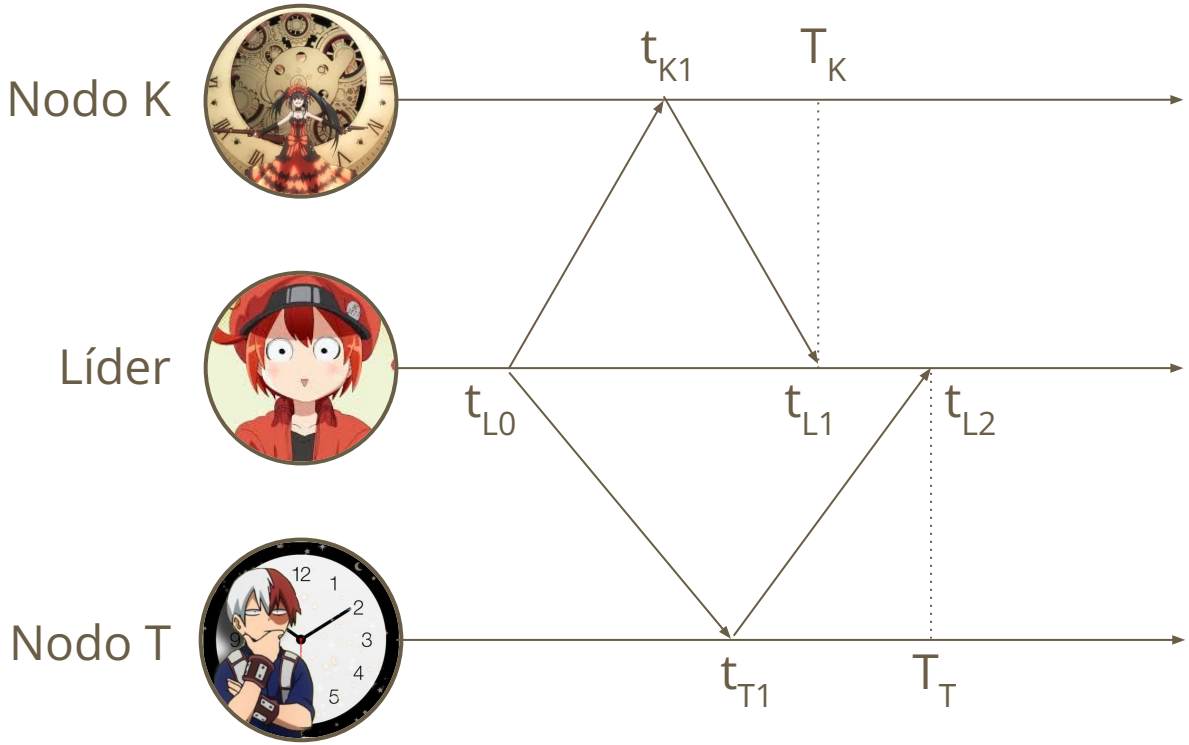
# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo



# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo

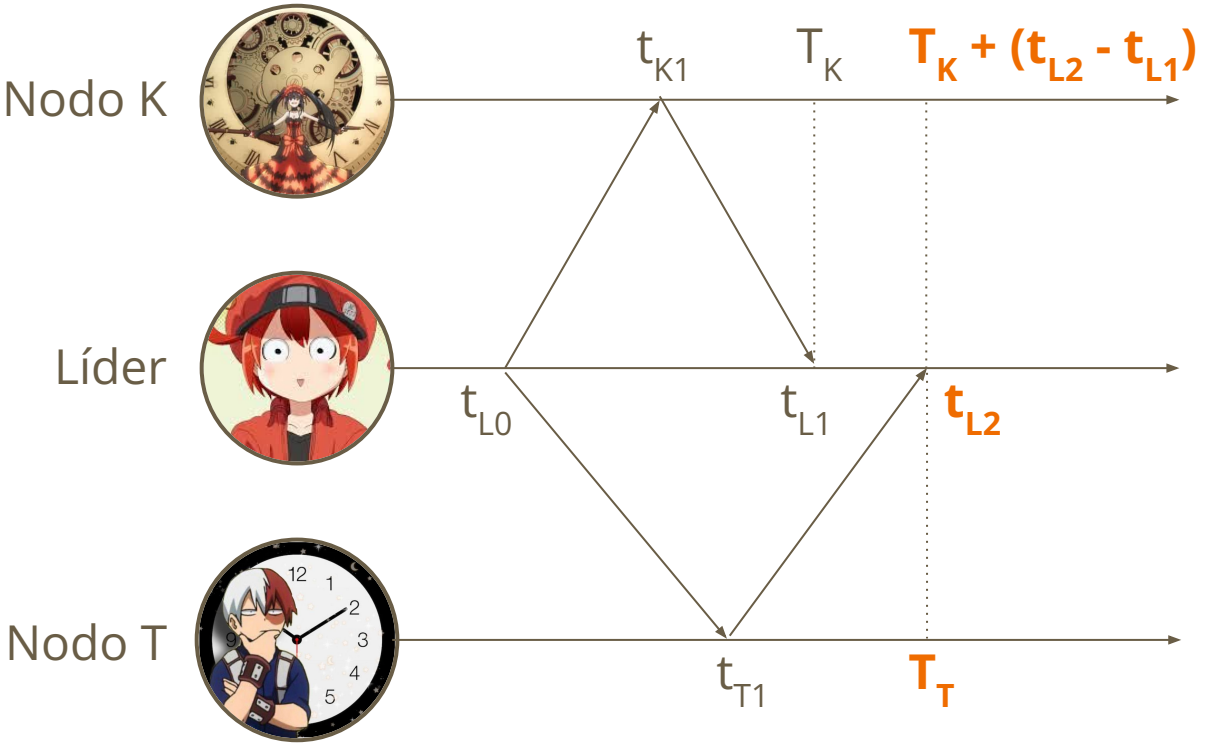



$$T_K = t_{K1} + (t_{L1} - t_{L0})/2$$

$$T_T = t_{T1} + (t_{L2} - t_{L0})/2$$

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo



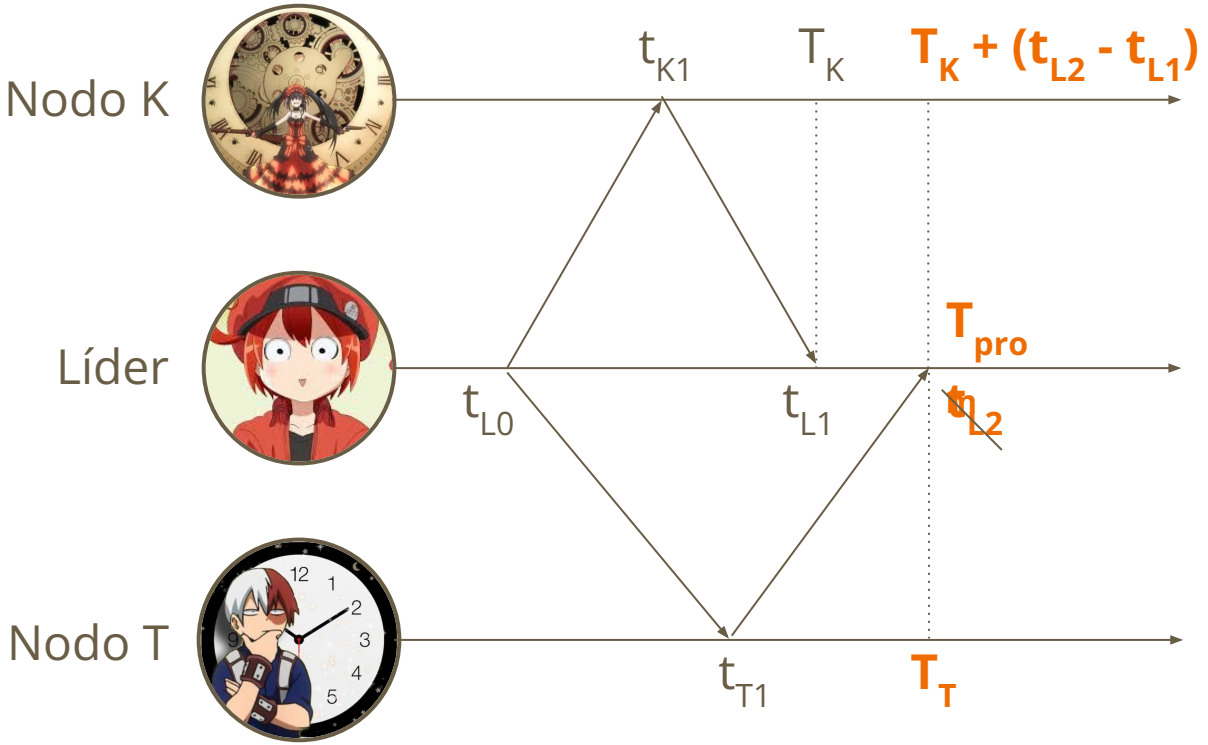

$T_K = t_{K1} + (t_{L1} - t_{L0})/2$

$T_T = t_{T1} + (t_{L2} - t_{L0})/2$

Se obtiene el promedio de las 3 horas ($T_{prom}$)

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo



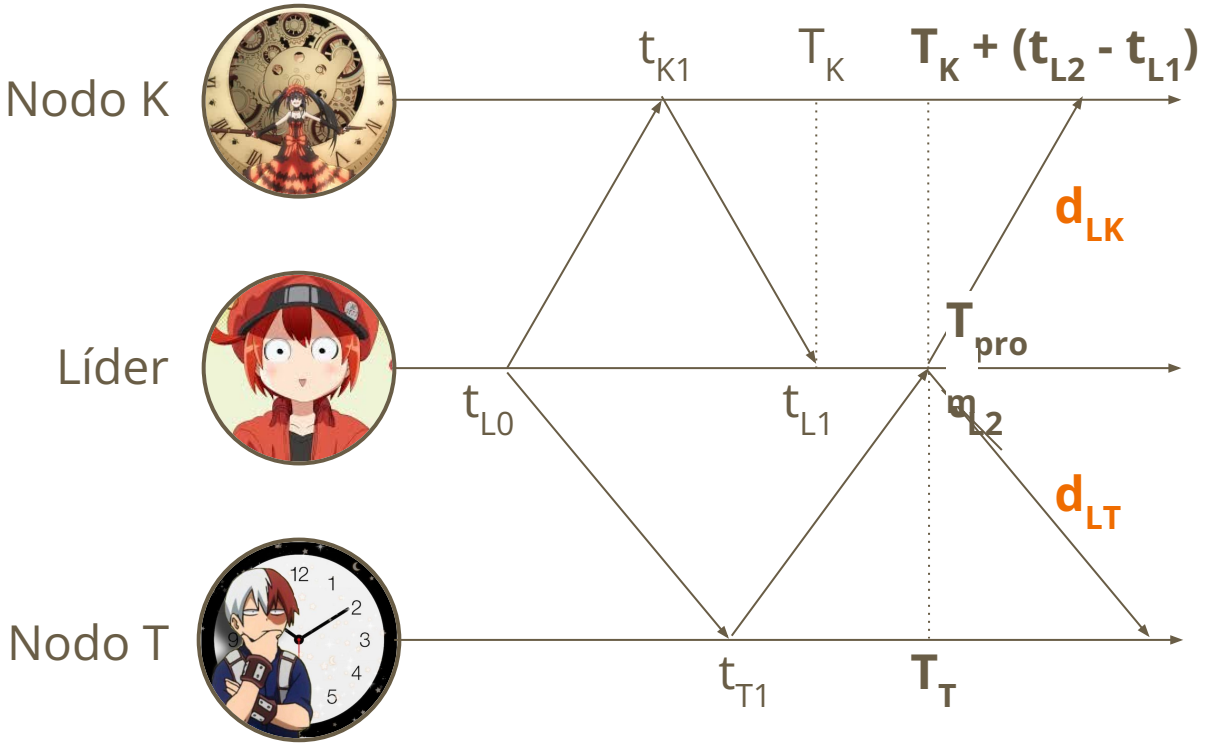

$T_K = t_{K1} + (t_{L1} - t_{L0})/2$

$T_T = t_{T1} + (t_{L2} - t_{L0})/2$

Se obtiene el promedio de las 3 horas ( $T_{prom}$ )

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo



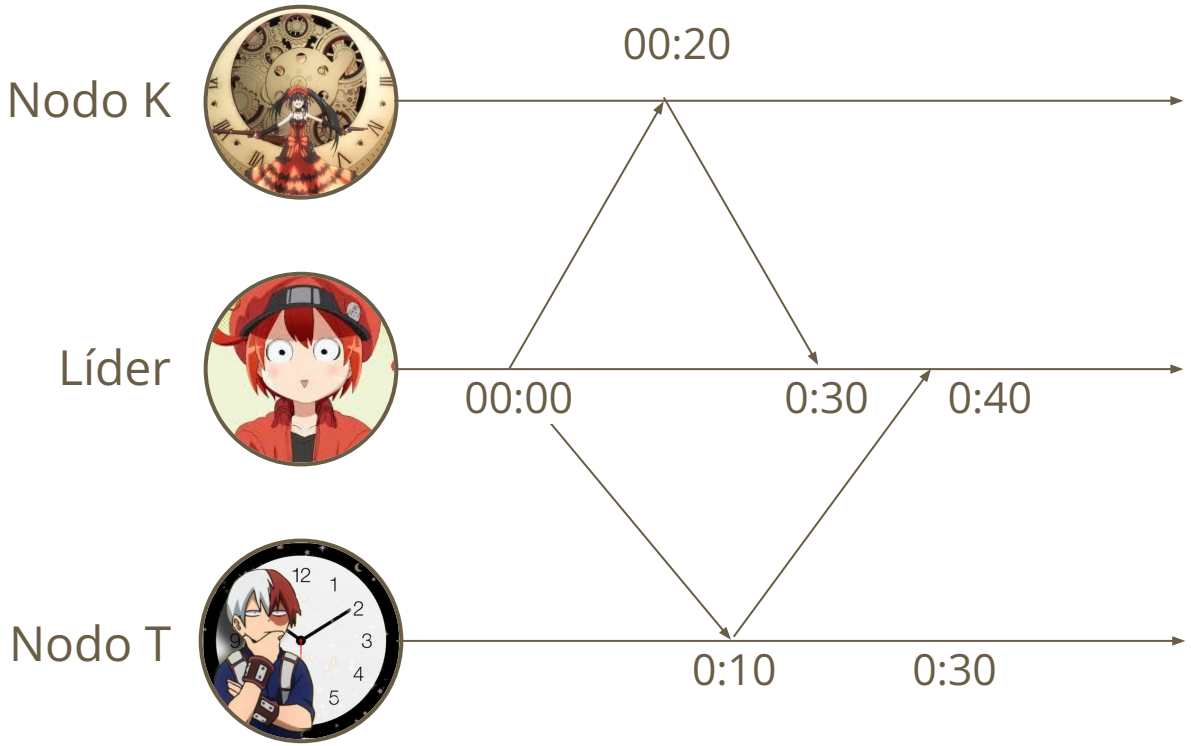

$T_K = t_{K1} + (t_{L1} - t_{L0})/2$

$T_T = t_{T1} + (t_{L2} - t_{L0})/2$

Se manda la diferencia entre el tiempo estimado en $t_{L2}$ y el $T_{prom}$ para cada nodo.

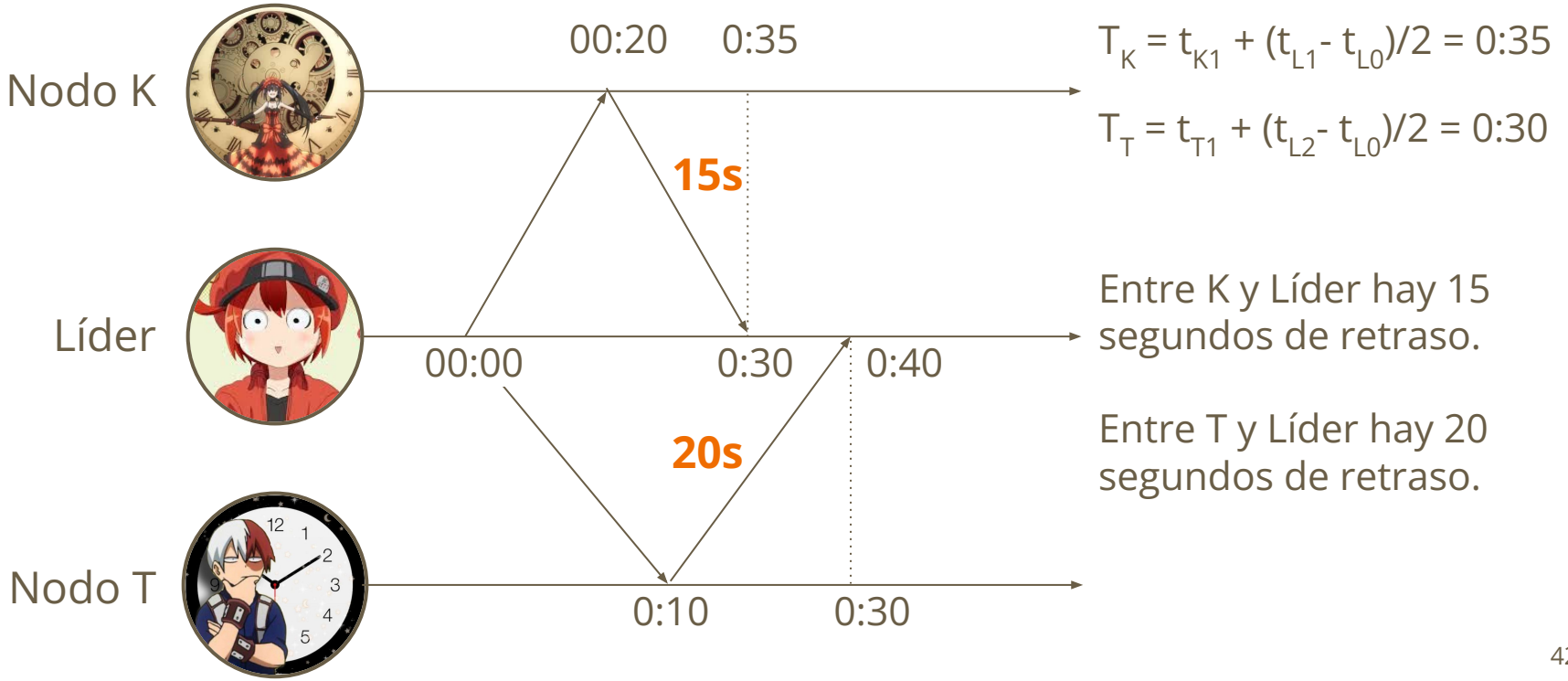
# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo



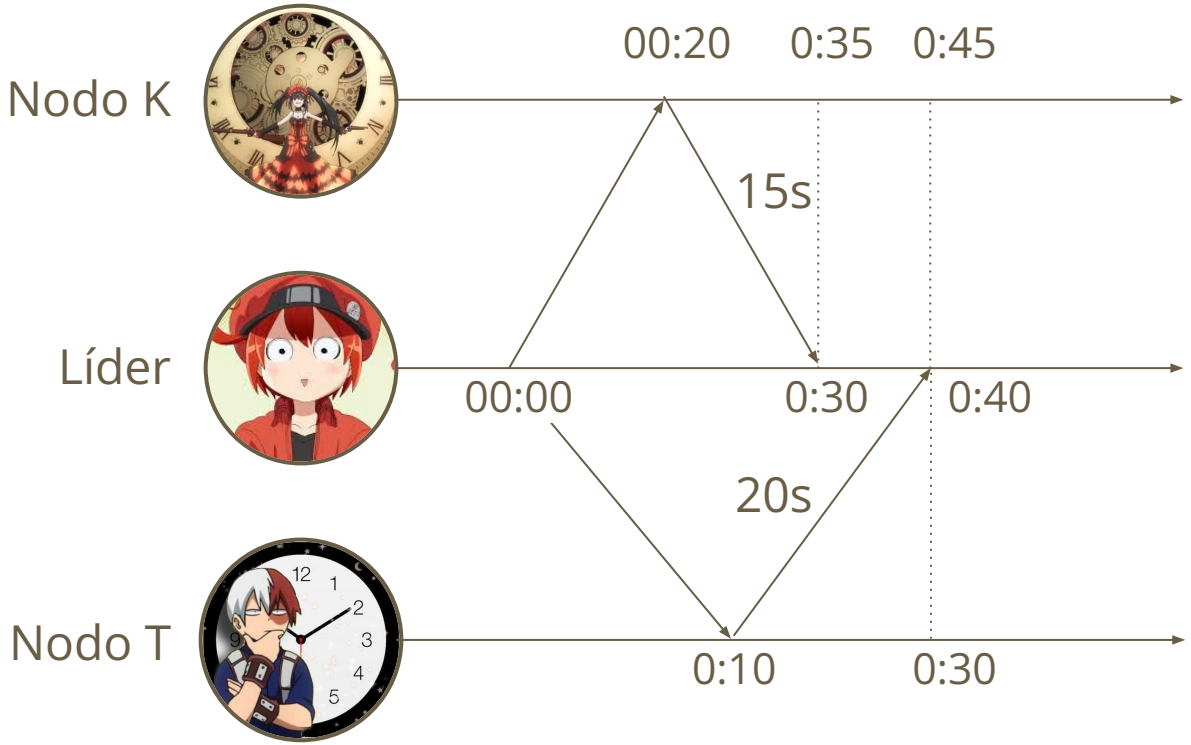
# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo



# Sincronización de relojes - Algoritmos (Reloj Físico)

**Algoritmo de Berkeley - Ejemplo**

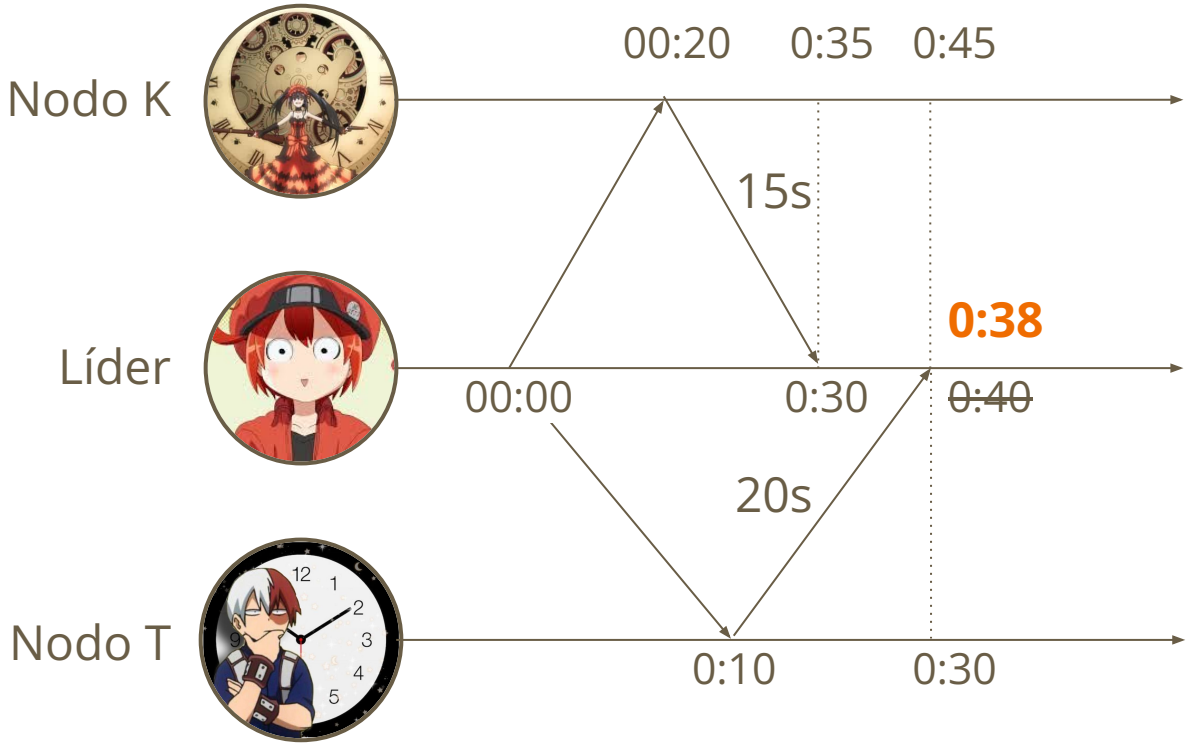



En el tiempo del último mensaje (0:40), se calcula promedio entre tiempos estimados

(0:45 + 0:40 + 0:30)/3 = 0.38

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo

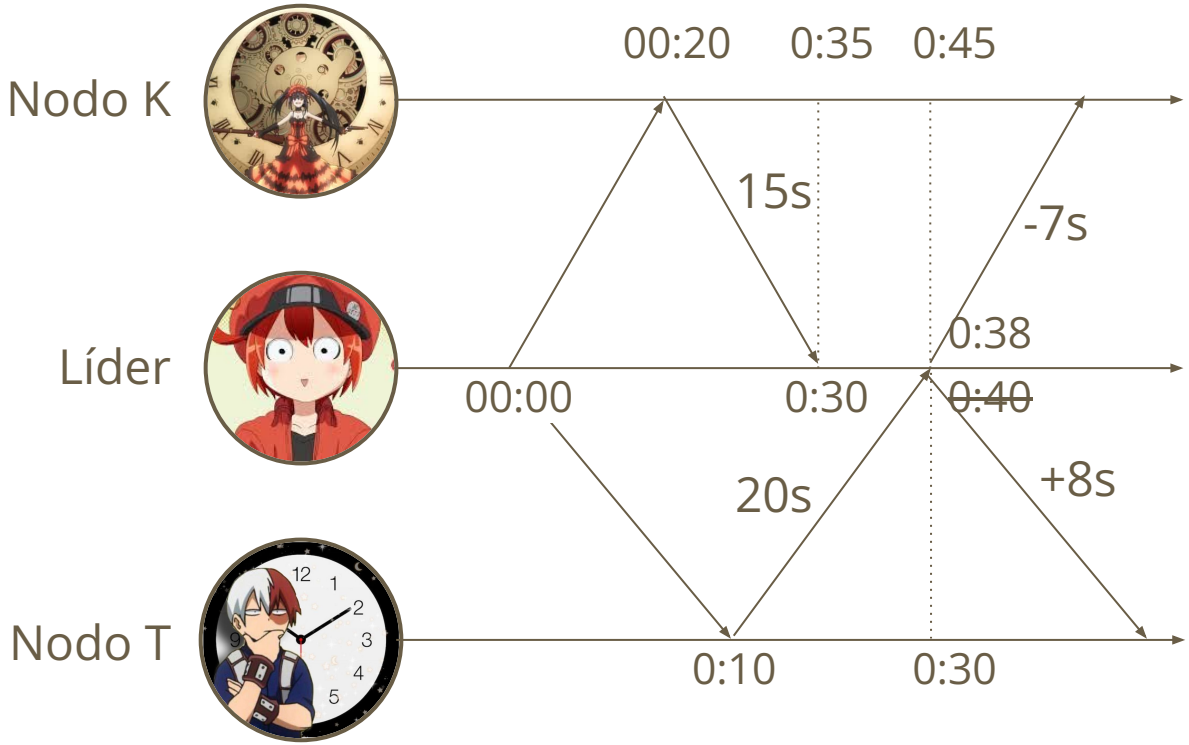



En el tiempo del último mensaje (0:40), se calcula promedio entre tiempos estimados

(0:45 + 0:40 + 0:30)/3 = 0.38

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo



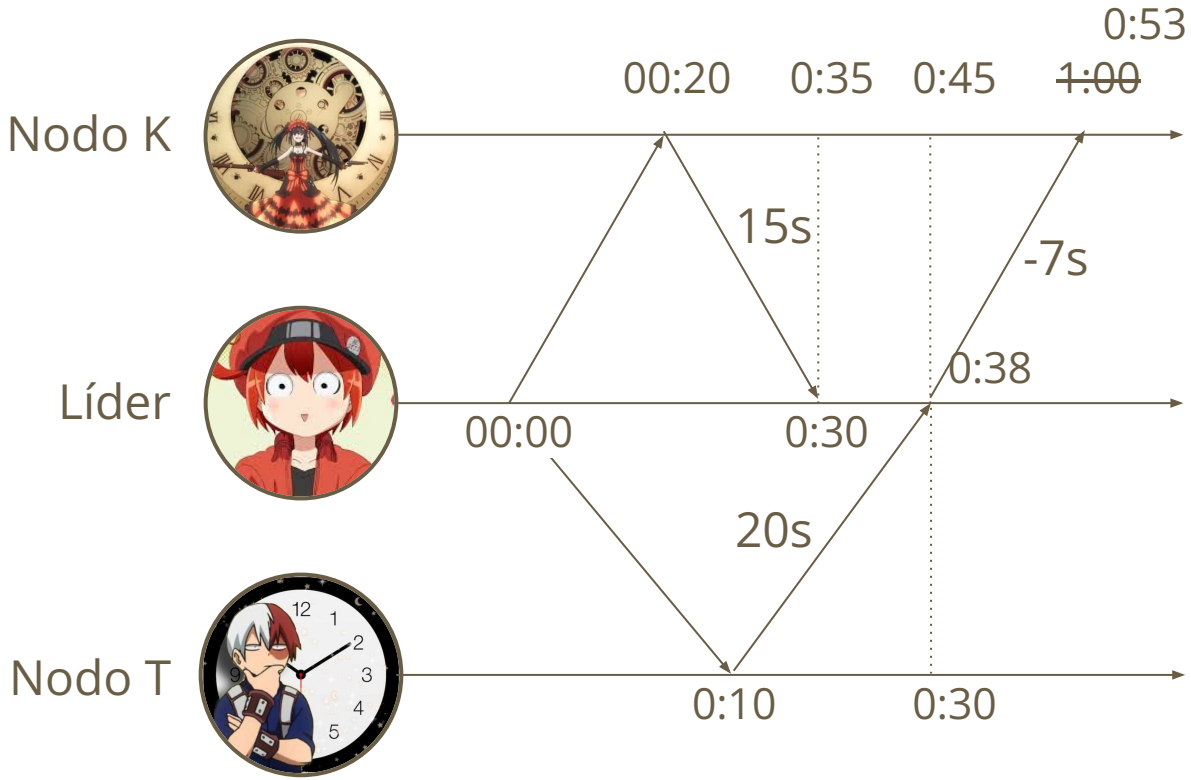

Se envía diferencia de tiempo para que todos calcen con el tiempo promedio

K: 0:45 - 0:38 = 7s
[7 segundos adelantado]

T: 0:30 - 0:38 = -8s
[8 segundos atrasado]

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo

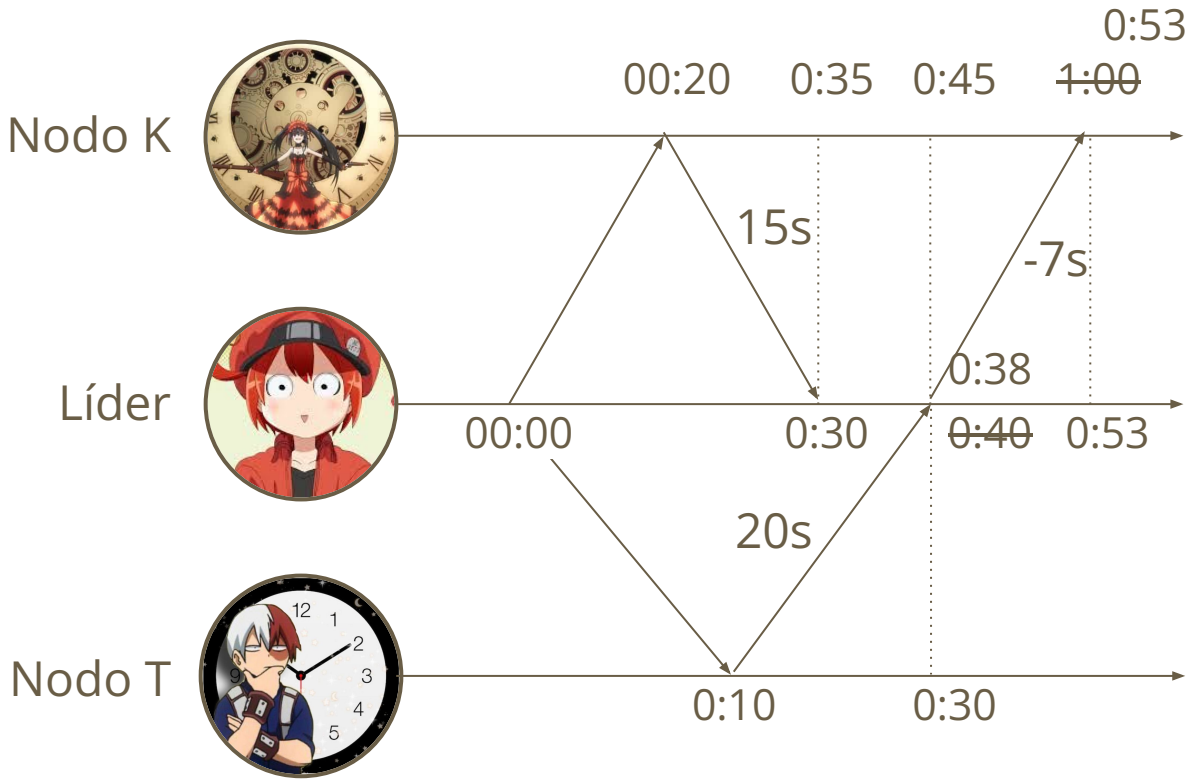



**Comprobemos nodo K**

Cuando su reloj marque 1:00, le llegará el mensaje de "-7" dejándolo en 0:53

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo




**Comprobemos nodo K**

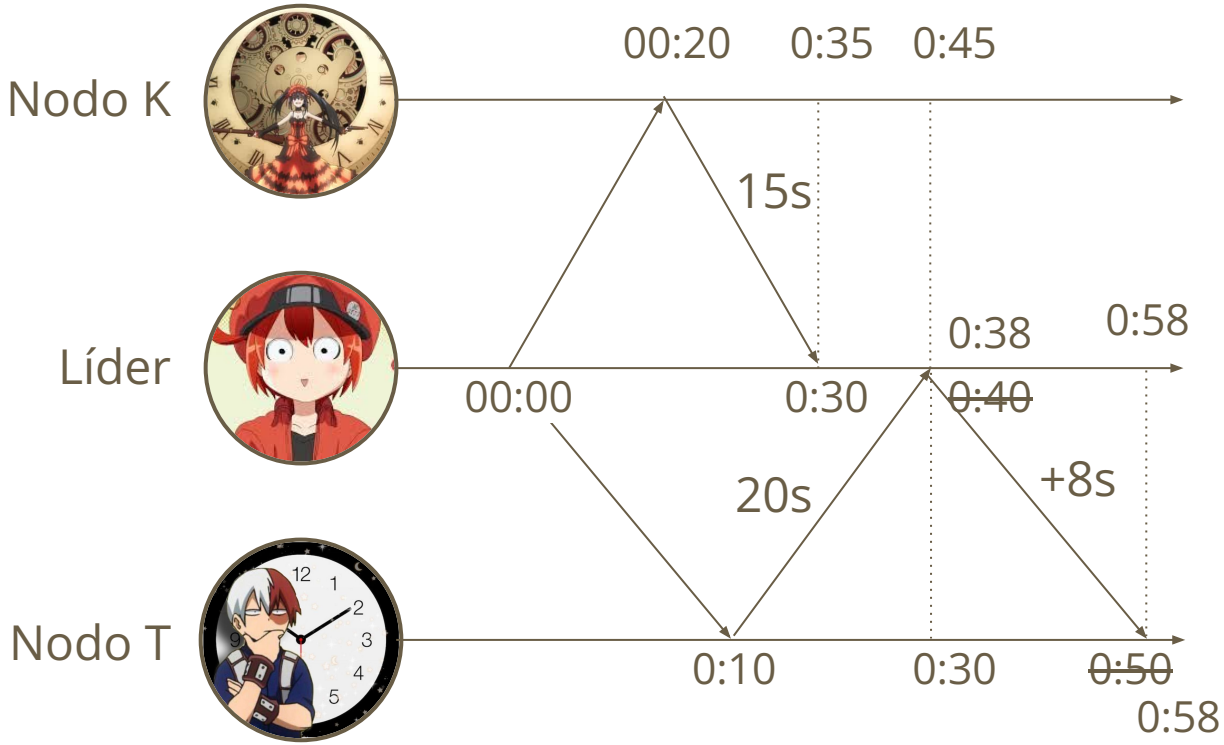
Cuando su reloj marque 1:00, le llegará el mensaje de "-7" dejándolo en 0:53

A partir de 0:38, luego de 15 segundos será 0:53

**¡Coincide con Nodo K!**

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Algoritmo de Berkeley - Ejemplo




**Comprobemos nodo T**

Cuando su reloj marque 0:50, le llegará el mensaje de "+8" dejándolo en 0:58

A partir de 0:38, luego de 20 segundos será 0:58

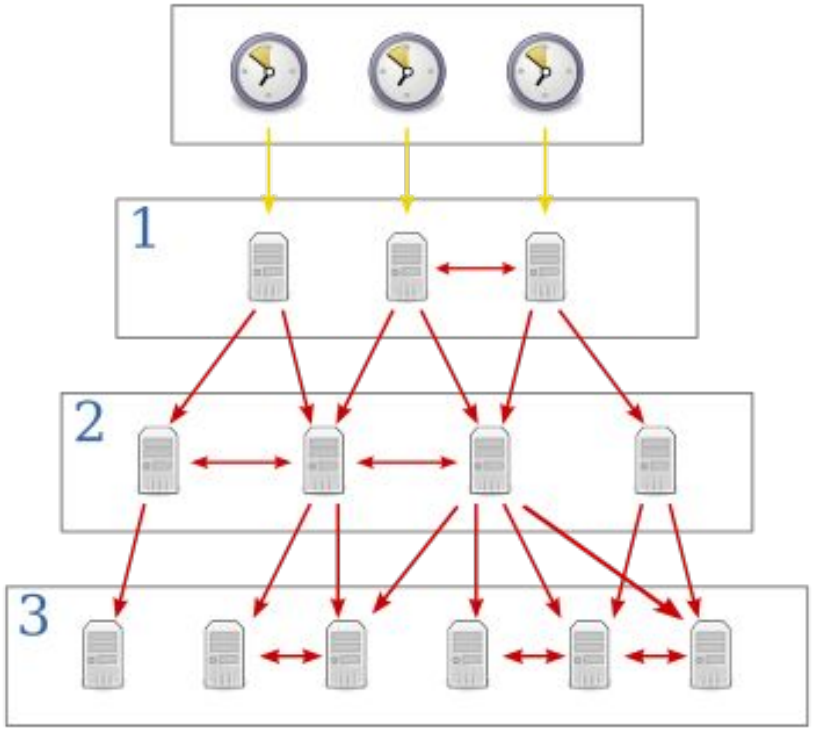
**¡Coincide con Nodo T!**

# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP)**

Funciona a partir de estratos.

- Estrato 0: relojes más eficientes.
- Estrato N se sincroniza entre ellos y con el estrato anterior
- A medida que N es mayor, la sincronización será menos precisa.



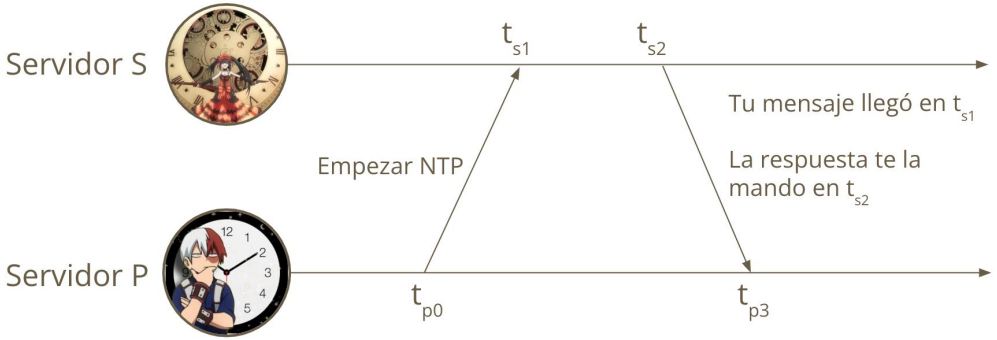

**Fuente:** Network Time Protocol

# Sincronización de relojes - Algoritmos (Reloj Físico)

## Protocolo de Tiempo de Red (NTP)

Mediante el envío de 2 mensajes, se calculan 2 métricas:

- **Retraso**: tiempo total de ida y vuelta para los dos mensajes.
- **Desfase**: estimación del desplazamiento entre el reloj de ambos servidores

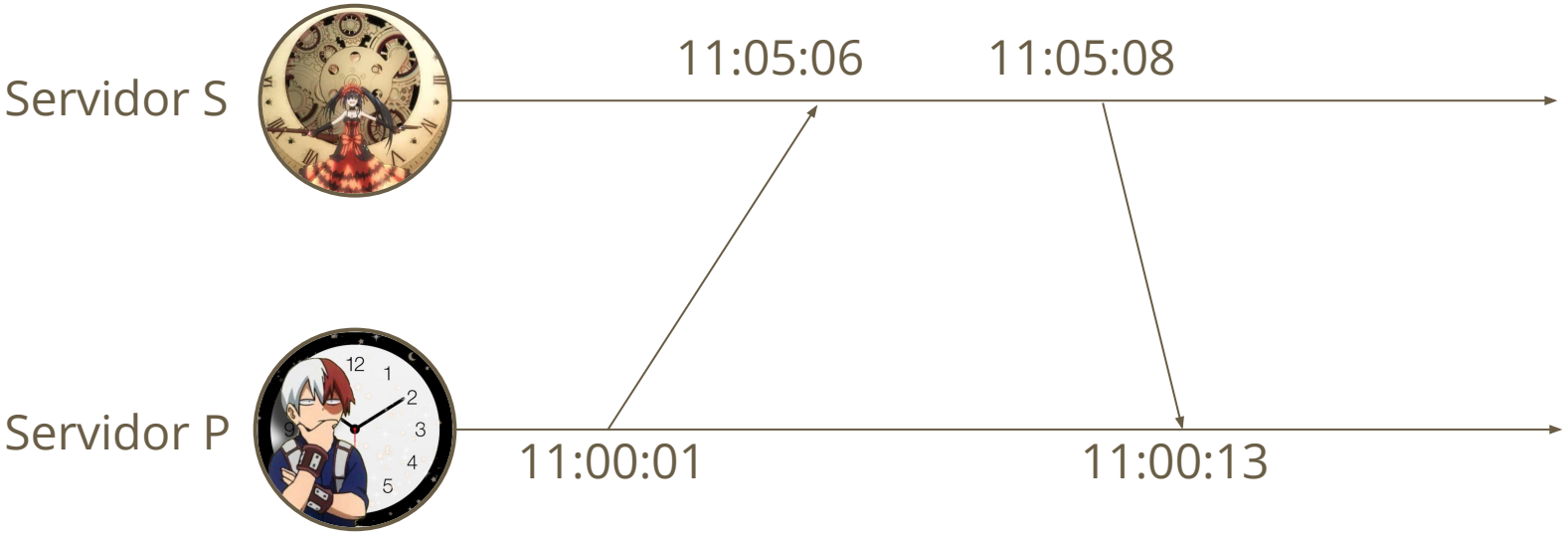



$$\text{Retraso} = (T_{s1} - T_{p0}) + (T_{p3} - T_{s2}) = (T_{p3} - T_{p0}) - (T_{s2} - T_{s1})$$

$$\text{Desfase} = \frac{(T_{s1} - T_{p0}) + (T_{s2} - t_{p3})}{2}$$

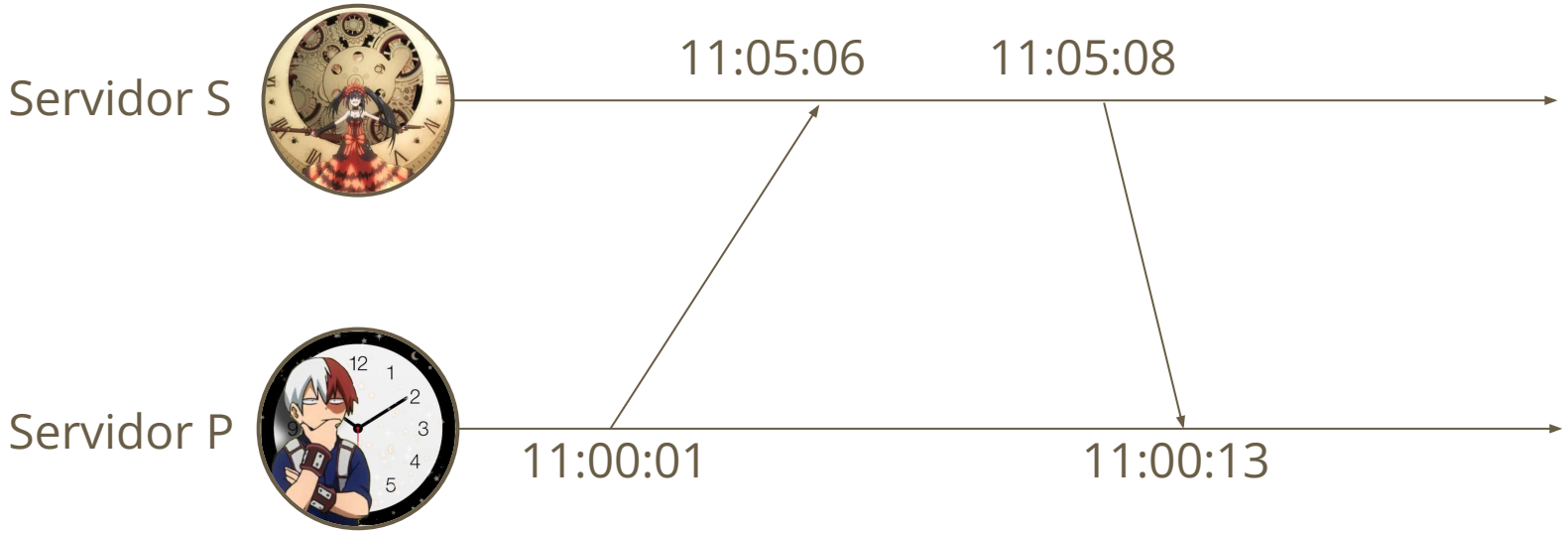
# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP) [Ejemplo 1]**



# Sincronización de relojes - Algoritmos (Reloj Físico)

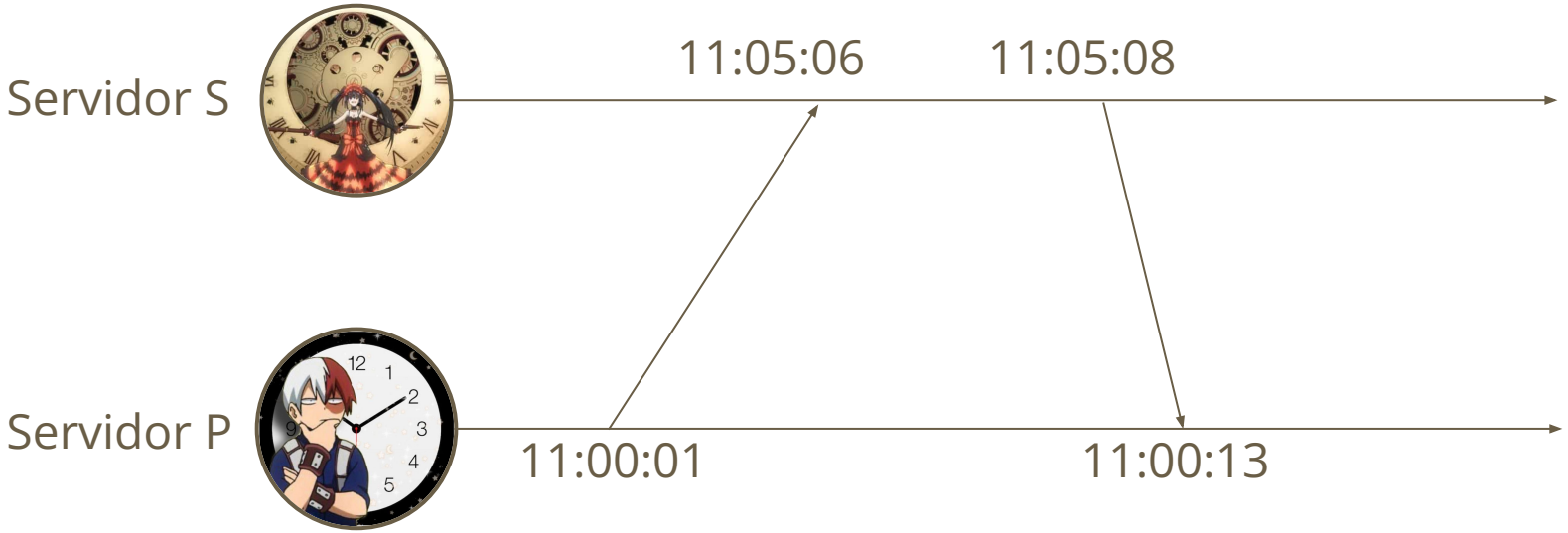
**Protocolo de Tiempo de Red (NTP) [Ejemplo 1]**




Retraso = $(T_{s1} - T_{p0}) + (T_{p3} - T_{s2})$ = (05:06 - 00:01) + (00:13 - 05:08) = 5:05 - 4:55 = 10 segundos

# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP) [Ejemplo 1]**

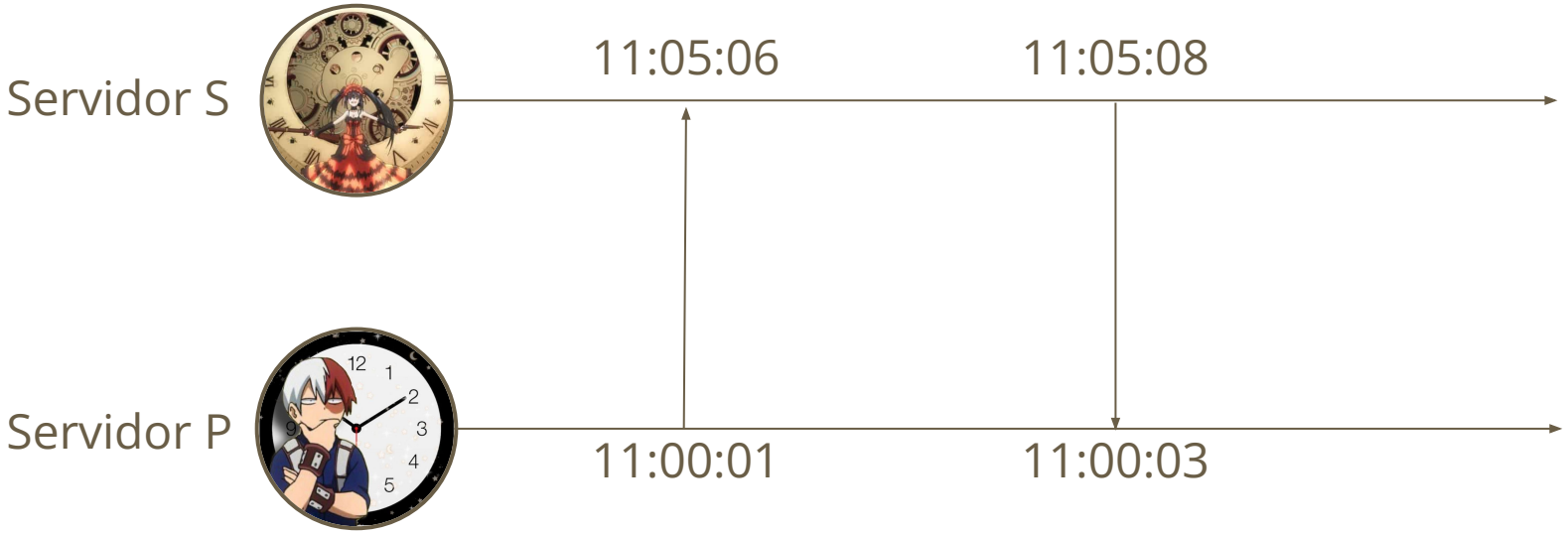



Retraso = $(T_{s1} - T_{p0}) + (T_{p3} - T_{s2}) = (05:06 - 00:01) + (00:13 - 05:08) = 5:05 - 4:55 =$ 10 segundos

Desfase = $\frac{(T_{s1} - T_{p0}) + (T_{s2} - t_{p3})}{2} = \frac{5:05 + (05:08 - 00:07)}{2} = \frac{5:05 + 4:55}{2} = \frac{10:00}{2}$ = 5 minutos.

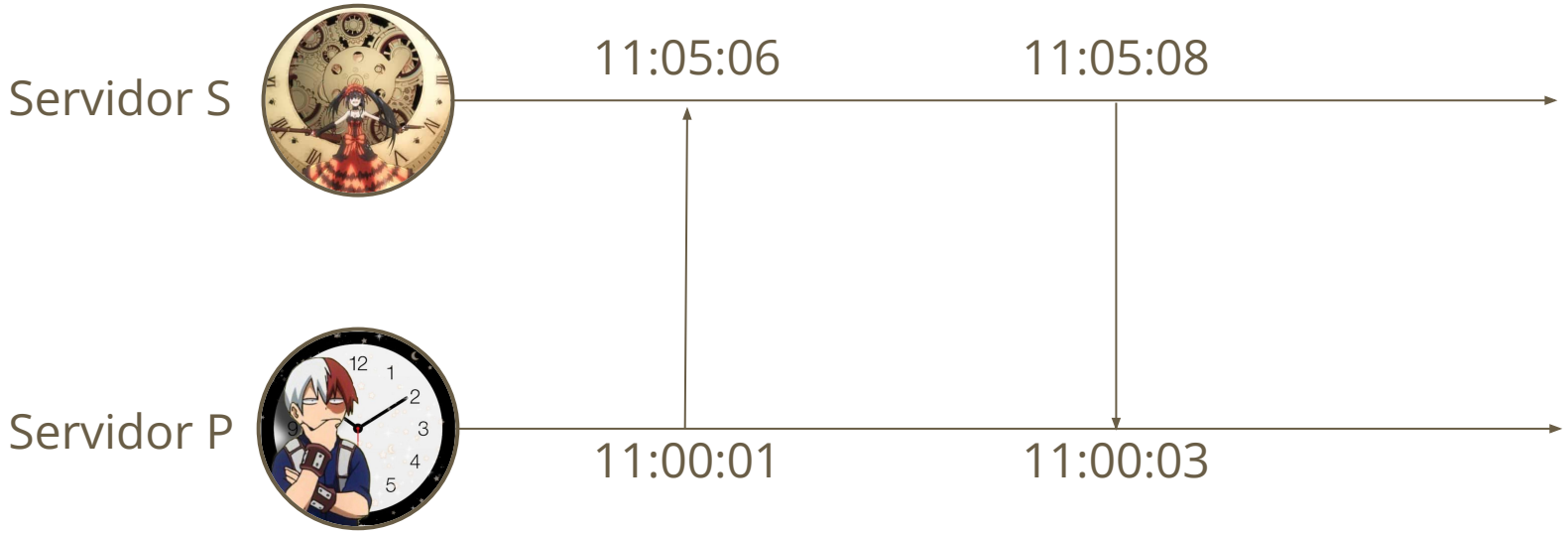
# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP) [Ejemplo 2]**



# Sincronización de relojes - Algoritmos (Reloj Físico)

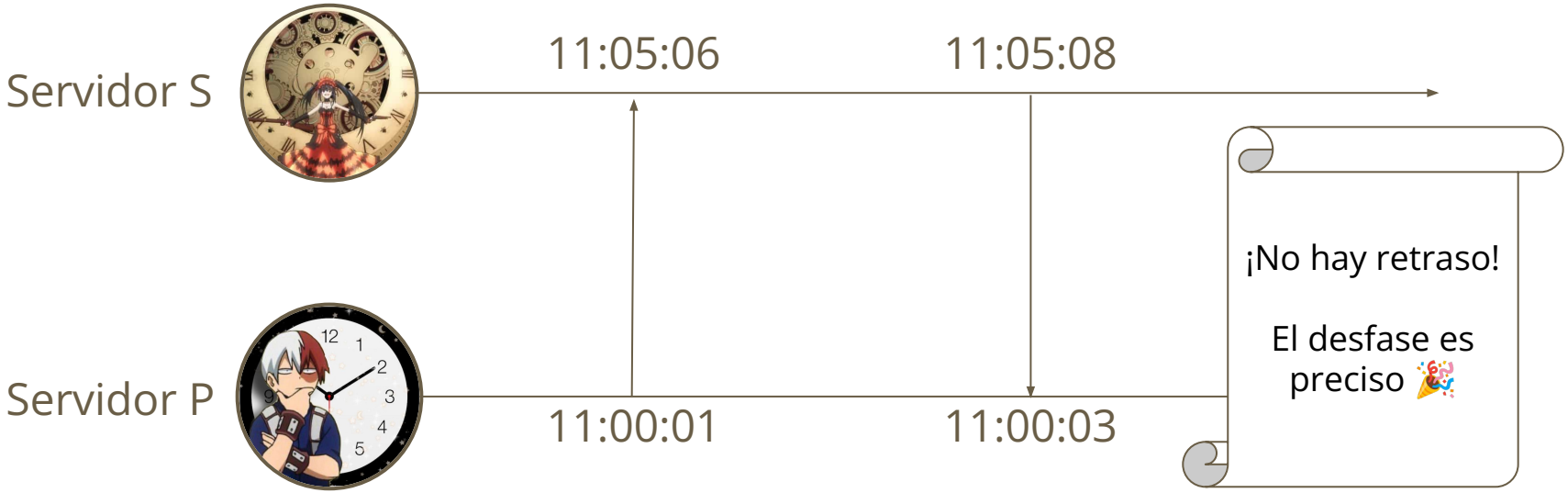
**Protocolo de Tiempo de Red (NTP) [Ejemplo 2]**




Retraso = $(T_{s1} - T_{p0}) + (T_{p3} - T_{s2})$ = (05:06 - 00:01) + (00:03 - 05:08) = 5:05 - 5:05 = 0 segundos

# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP) [Ejemplo 2]**

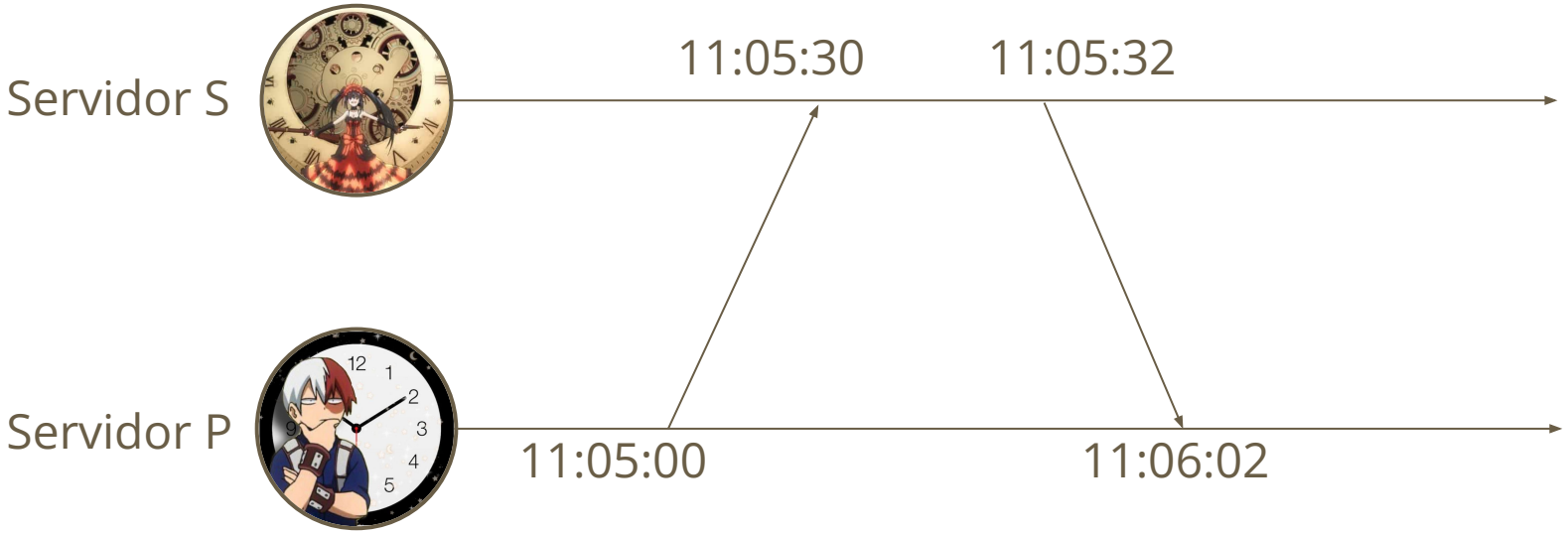



Retraso = $(T_{s1} - T_{p0}) + (T_{p3} - T_{s2})$ = (05:06 - 00:01) + (00:03 - 05:08) = 5:05 - 5:05 = 0 segundos

Desfase = $\frac{(T_{s1} - T_{p0}) + (T_{s2} - t_{p3})}{2} = \frac{5:05 + (05:08 - 00:03)}{2} = \frac{5:05 + 5:05}{2}$ = 5 minutos y 5 segundos

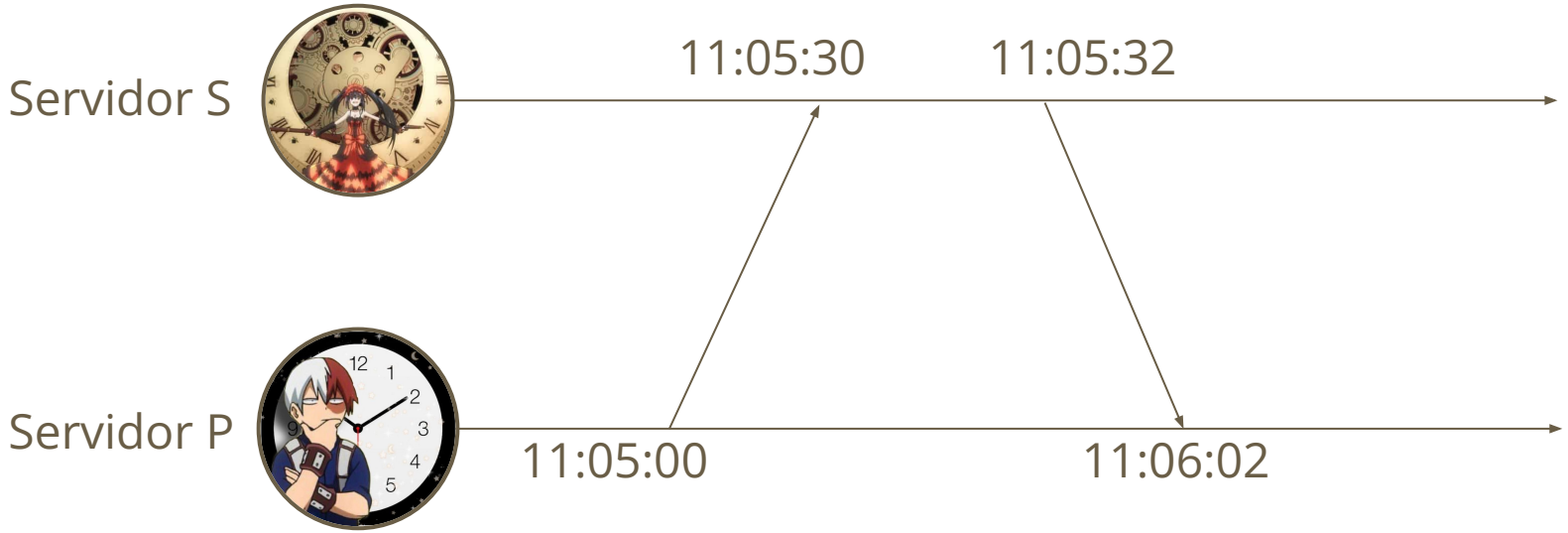
# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP) [Ejemplo 3]**



# Sincronización de relojes - Algoritmos (Reloj Físico)

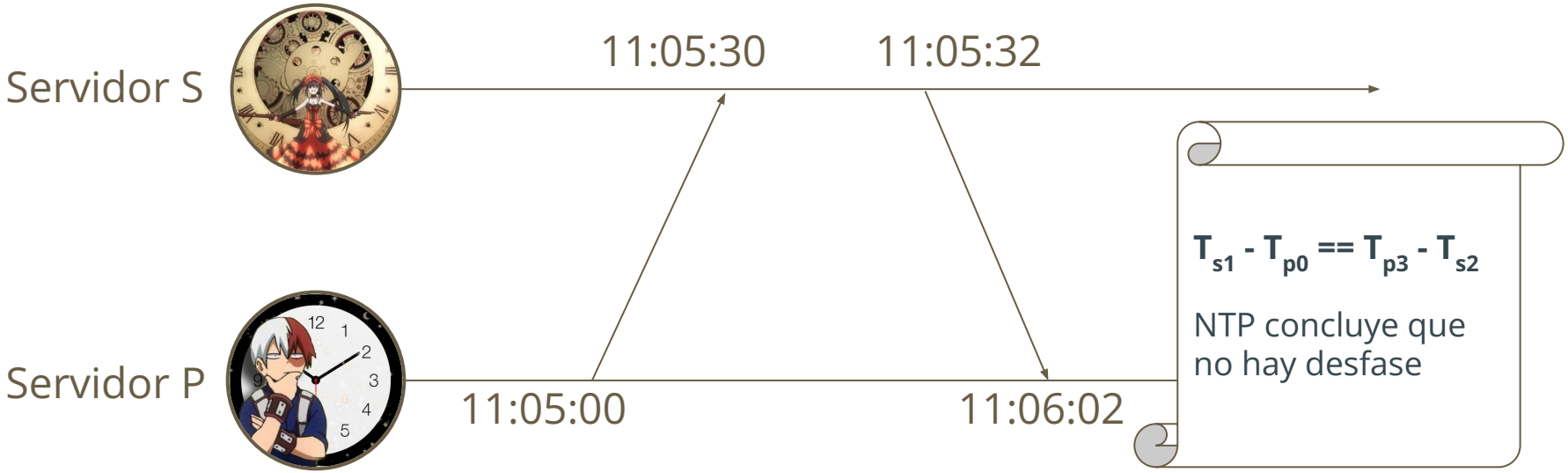
**Protocolo de Tiempo de Red (NTP) [Ejemplo 3]**




Retraso = $(T_{s1} - T_{p0}) + (T_{p3} - T_{s2})$ = (05:30 - 05:00) + (06:02 - 05:32) = 0:30 + 0:30 = 1 minuto.

# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP) [Ejemplo 3]**



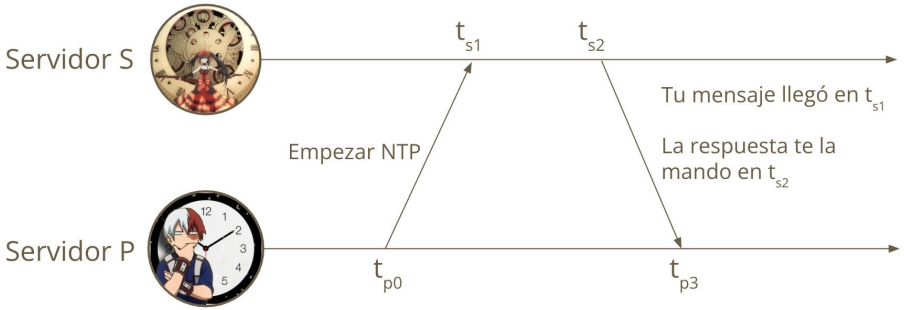

Retraso = $(T_{s1} - T_{p0}) + (T_{p3} - T_{s2})$ = (05:30 - 05:00) + (06:02 - 05:32) = 0:30 + 0:30 = 1 minuto.

Desfase = $\frac{(T_{s1} - T_{p0}) + (T_{s2} - t_{p3})}{2} = \frac{0:30 + (05:32 - 06:02)}{2} = \frac{0:30 - 0:30}{2}$ = 0 segundos.

# Sincronización de relojes - Algoritmos (Reloj Físico)

**Protocolo de Tiempo de Red (NTP)**

- NTP realiza este proceso varias veces y selecciona aquel con menor retraso para aplicar su desfase al servidor.
- Este proceso se puede hacer con diferentes servidores para obtener más valores.




$$\text{Retraso} = (T_{s1} - T_{p0}) + (T_{p3} - T_{s2}) = (T_{p3} - T_{p0}) - (T_{s2} - T_{s1})$$

$$\text{Desfase} = \frac{(T_{s1} - T_{p0}) + (T_{s2} - t_{p3})}{2}$$

# Poniendo a prueba lo que hemos aprendido 👀

¿Con cuál de los siguientes mecanismos **es más seguro** que los nodos queden sincronizados lo más cercano al Tiempo Universal Coordinado (UTC)?

a. Relojes lógicos de Lamport.
b. Algoritmo de Berkeley.
c. Protocolo de Tiempo de Red.
d. Algoritmo de Raft.
e. Método de Cristian.

# Próximos eventos

## Próxima clase

- Sincronización de relojes: Vectores lógicos
- Estados globales. ¿cómo coordinarnos para hacer un "*checkpoint*" en un sistema distribuido.

## Evaluación

- Tarea 1 se publicará la otra semana y evalúa hasta aquí.
  - *Spoiler 1*: levantar más de un servidor TCP con *socket* que no se quede bloqueado.
  - *Spoiler 2*: los servidores TCP tendrán que interactuar con un RCP.
  - *Spoiler 3*: los servidores interactúan entre ellos usando reloj lógico.

# IIC2523
# Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 05)

# Créditos (animes utilizados)

## Date a Live

## Boku no Hero Academia


## Hataraku Saibou