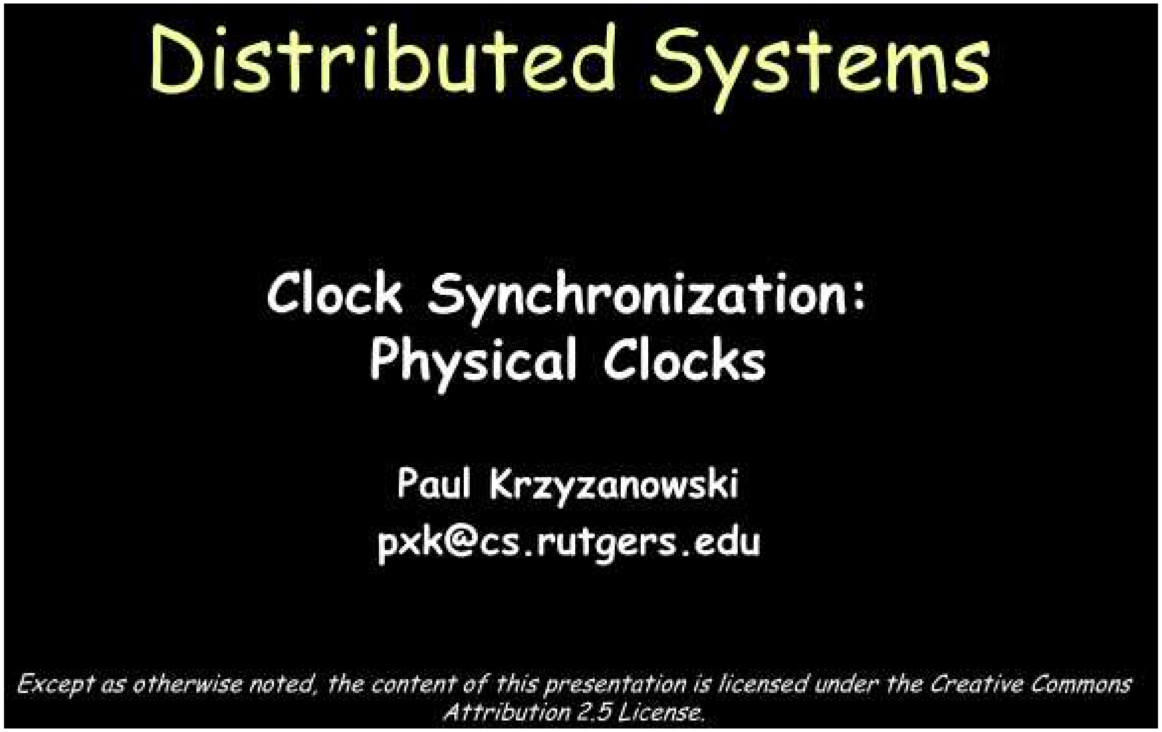
Programación Distribuida y Tiempo Real

Sincronización – Relojes Físicos

Sincronización de Relojes Físicos

- La mayoría de las explicaciones será de
  - CS 417: Distributed Systems, Paul Krzyzanowski https://www.cs.rutgers.edu/~pxk/417/index.html



Reloj Físico en un Sistema Operativo

- De una manera u otra, se contabilizan ciclos o transiciones de un material que vibra/varía con frecuencia *constante*

Reloj Físico en un Sistema Operativo

- De una manera u otra, se contabilizan ciclos o transiciones de un material que vibra/varía con frecuencia *constante*

- 1929: Quartz crystal clock
  - Resonator shaped like tuning fork
  - Laser-trimmed to vibrate at 32,768 Hz
  - Standard resonators accurate to 6 parts per million at 31° C
  - Watch will gain/lose < $\frac{1}{2}$ sec/day
  - Stability > accuracy: stable to 2 sec/month
  - Good resonator can have accuracy of 1 second in 10 years
    - Frequency changes with age, temperature, and acceleration

Reloj Físico en un Sistema Operativo

- En el “extremo” de precisión y exactitud

Atomic clocks

- Second is defined as 9,192,631,770 periods of radiation corresponding to the transition between two hyperfine levels of cesium-133
- Accuracy:
  better than 1 second in six million years
- NIST standard since 1960

Reloj Físico en un Sistema Operativo

OS generally programs a timer circuit to generate an interrupt periodically

- e.g., 60, 100, 250, 1000 interrupts per second
  (Linux 2.6+ adjustable up to 1000 Hz)
- Programmable Interval Timer (PIT) - Intel 8253, 8254
- Interrupt service procedure adds 1 to a counter in memory

Reloj Físico en un Sistema Operativo

OS generally programs a timer circuit to generate an interrupt periodically

- e.g., 60, 100, 250, 1000 interrupts per second (Linux 2.6+ adjustable up to 1000 Hz)
- Programmable Interval Timer (PIT) - Intel 8253, 8254
- Interrupt service procedure adds 1 to a counter in memory

Resonador + Contador

Reloj Físico en un Sistema Operativo

OS generally programs a timer circuit to generate an interrupt periodically

- e.g., 60, 100, 250, 1000 interrupts per second
  (Linux 2.6+ adjustable up to 1000 Hz)
- Programmable Interval Timer (PIT) - Intel 8253, 8254
- Interrupt service procedure adds 1 to a counter in memory

Este contador "representa" el tiempo transcurrido

Un incremento será 1/60s, 0.01s, 0.004s, 1ms

Reloj Físico en un Sistema Operativo

OS generally programs a timer circuit to generate an interrupt periodically

- e.g., 60, 100, 250, 1000 interrupts per second (Linux 2.6+ adjustable up to 1000 Hz)
- Programmable Interval Timer (PIT) - Intel 8253, 8254
- Interrupt service procedure adds 1 to a counter in memory

Este contador "representa" el tiempo transcurrido

Un incremento será 1/60s, 0.01s, 0.004s, 1ms

Reloj Físico en un Sistema Operativo

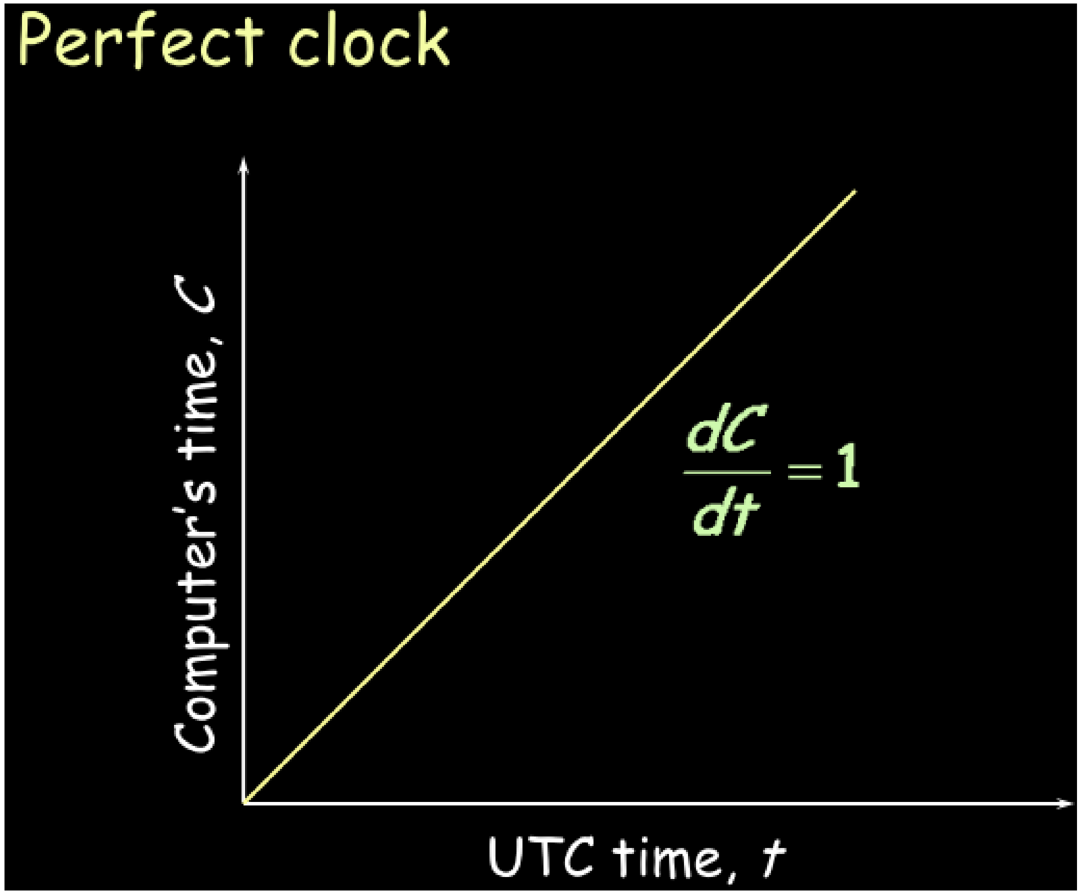
Problem

Getting two systems to agree on time
- Two clocks hardly ever agree
- Quartz oscillators oscillate at slightly different frequencies

Clocks tick at different rates
- Create ever-widening gap in perceived time
- Clock Drift

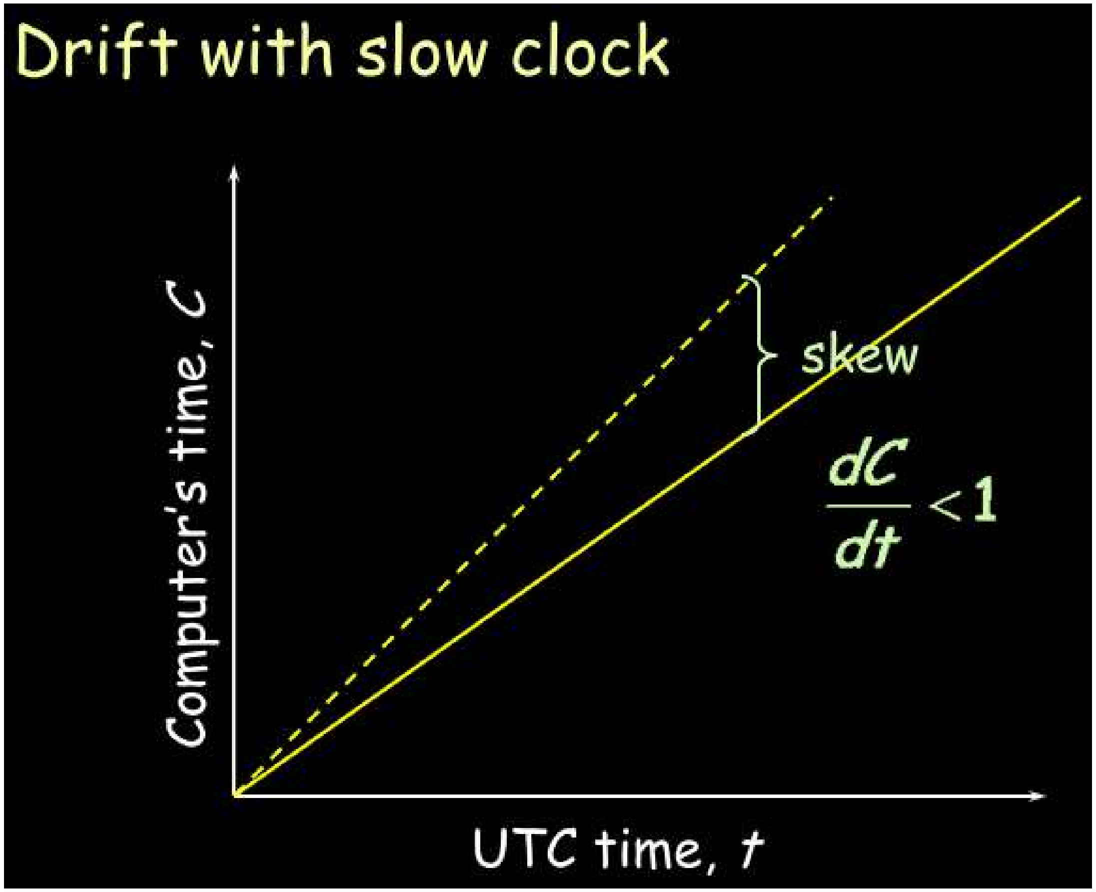
El primer problema: oscilador vs. reloj atómico
del sistema de cómputo vs. real

Reloj Físico en un Sistema Operativo



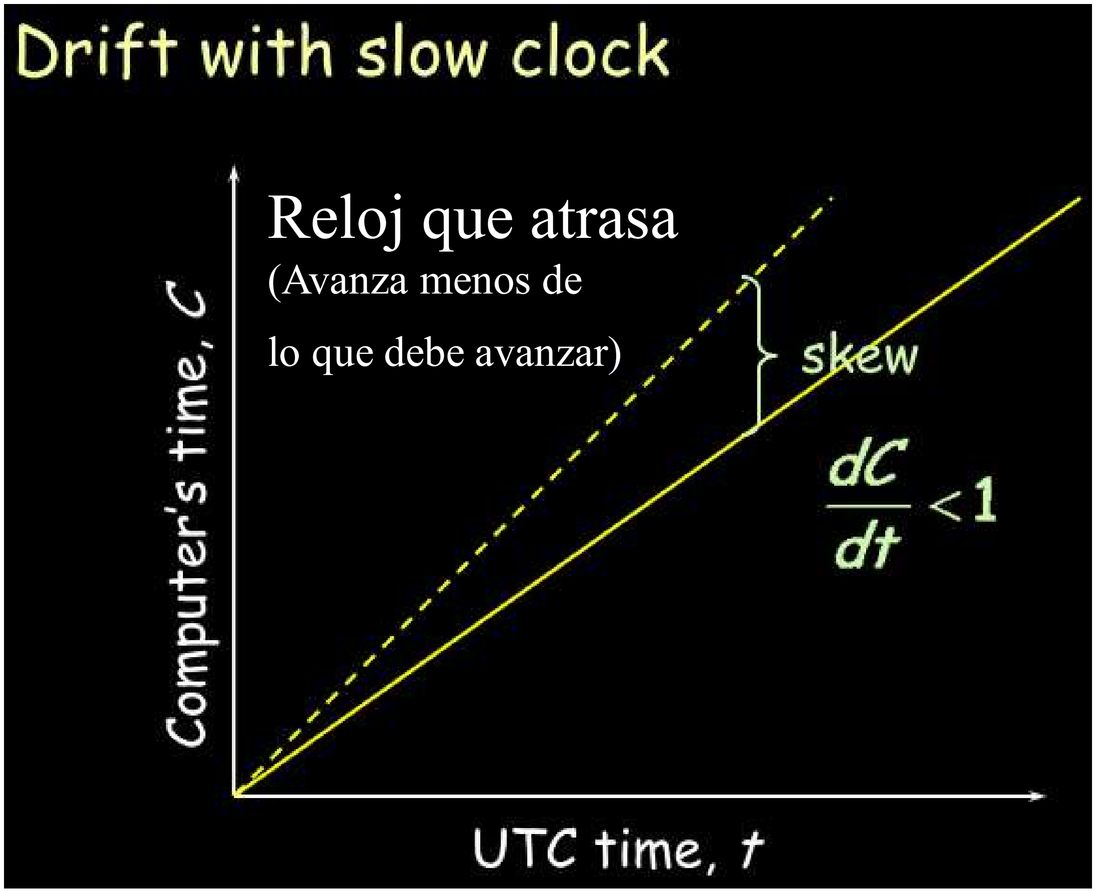
El primer problema: oscilador vs. reloj atómico

Reloj Físico en un Sistema Operativo



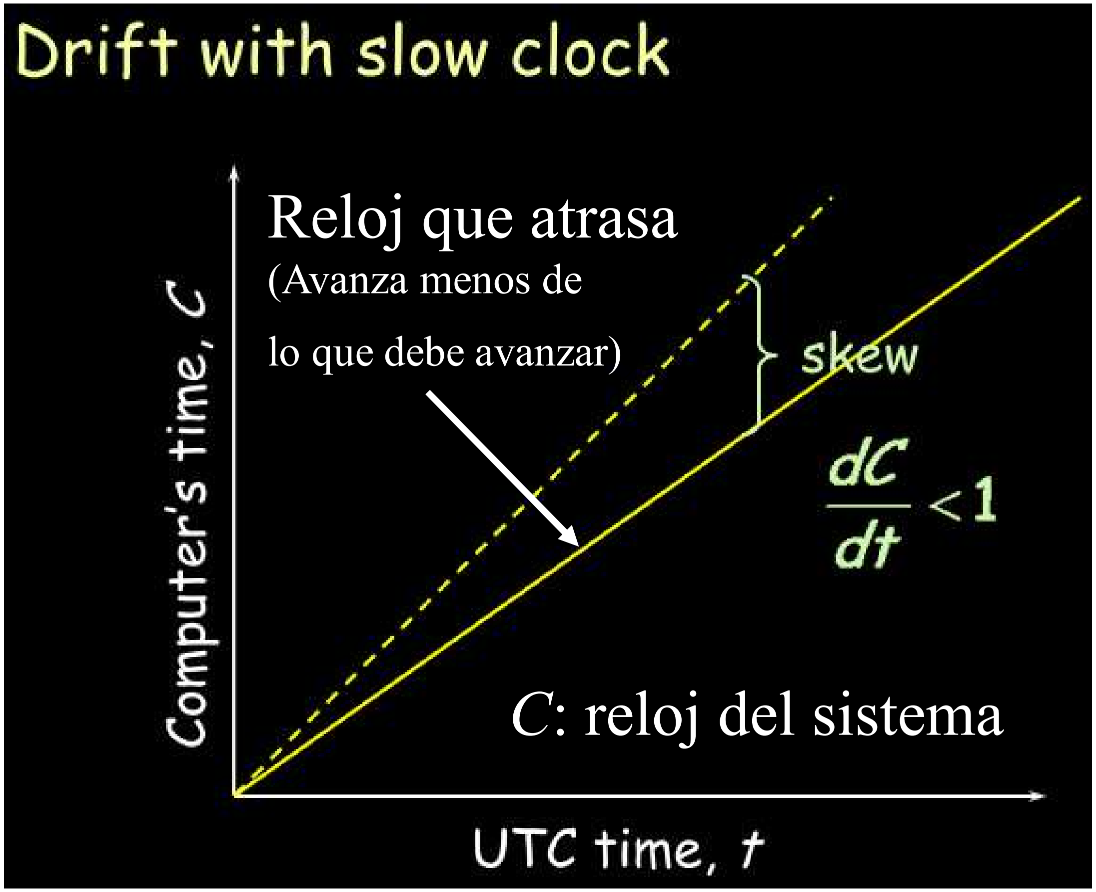
El primer problema: oscilador vs. reloj atómico

Reloj Físico en un Sistema Operativo



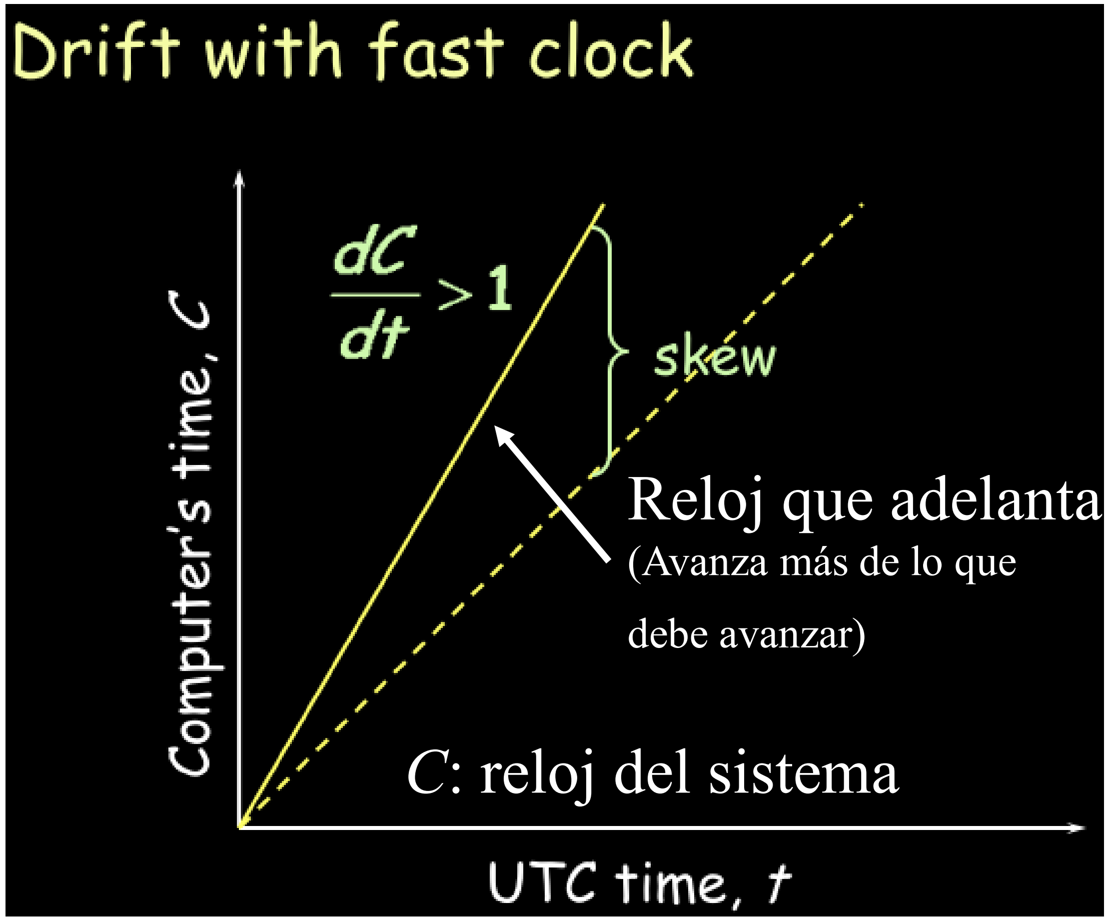
El primer problema: oscilador vs. reloj atómico

Reloj Físico en un Sistema Operativo



El primer problema: oscilador vs. reloj atómico

Reloj Físico en un Sistema Operativo



El primer problema: oscilador vs. reloj atómico

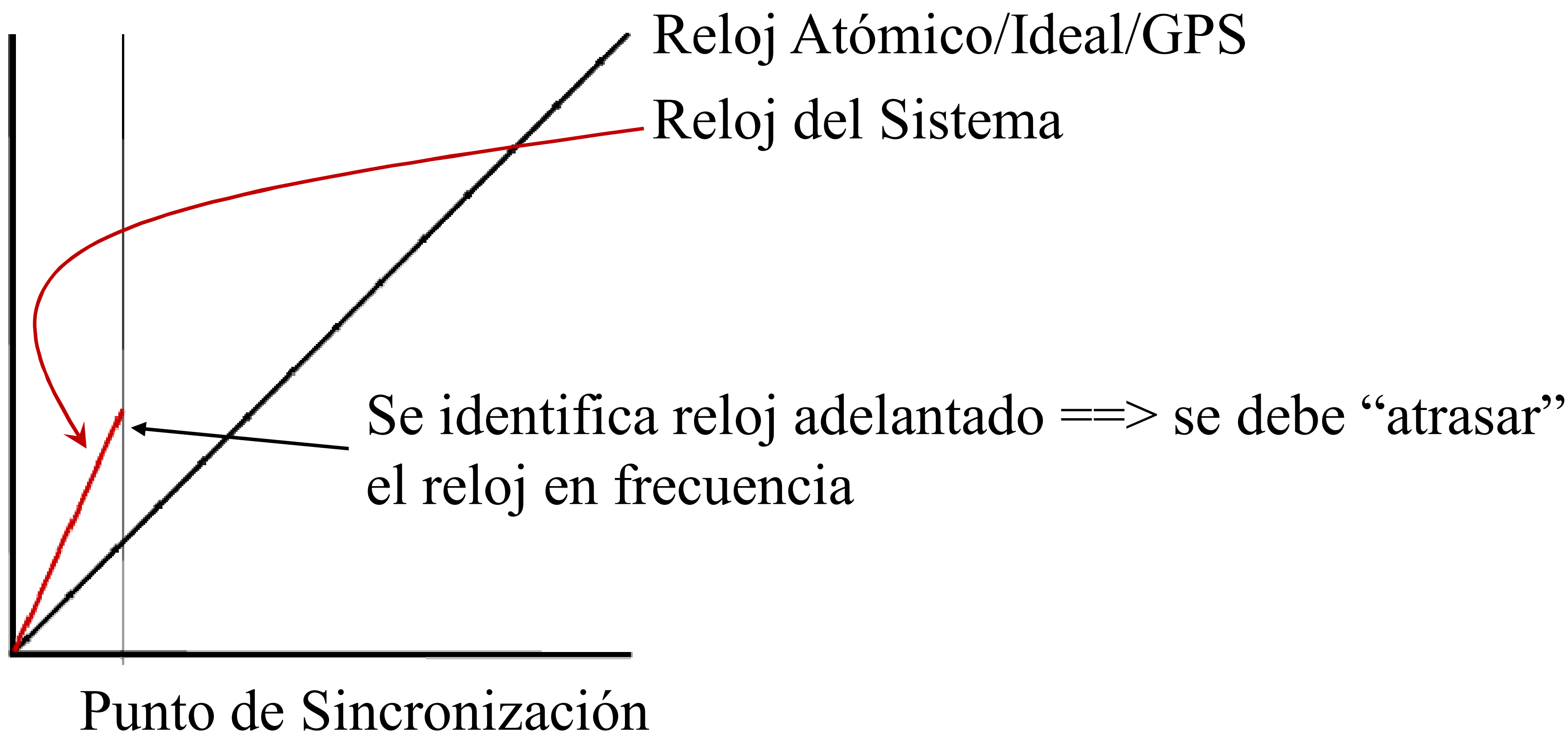
Reloj Físico en un Sistema Operativo

- Oscilador vs. reloj atómico:
  - Adelanta o Atrasa: problema de oscilador-frecuencia
  - Corregir diferencia
    - El problema es la frecuencia
    - Atrasar el reloj ==> "desordenar" eventos en el t

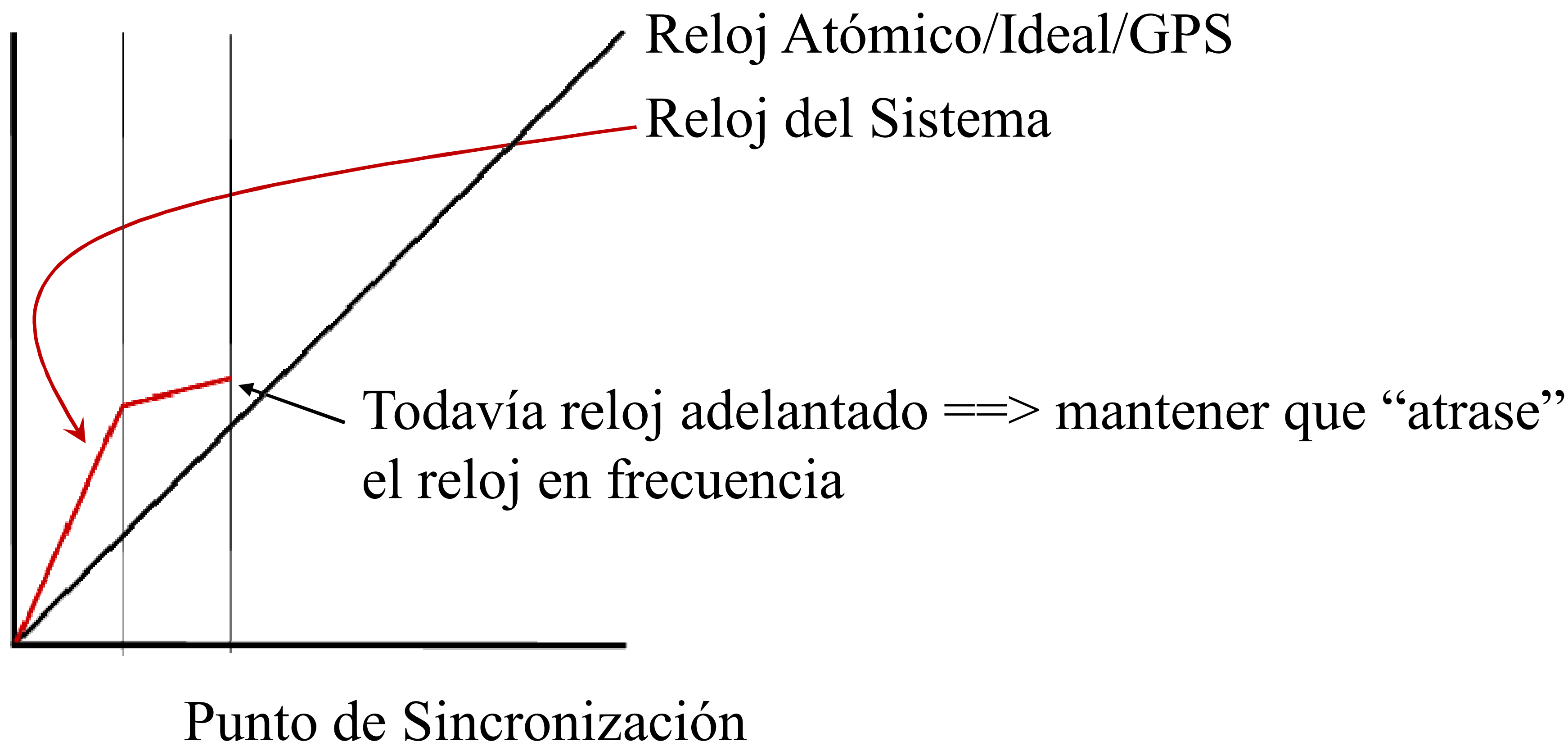
Reloj Físico en un Sistema Operativo

- Oscilador vs. reloj atómico:
  - Adelanta o Atrasa: problema de oscilador-frecuencia
  - Corregir diferencia
    - El problema es la frecuencia
    - Atrasar el reloj ==> “desordenar” eventos en el t
  - Corregir frecuencia
    - “Atrasar” un reloj adelantado. Ej: sumar 1 cada 105 oscilaciones en un reloj de 100 Mhz nominales (“esperar” más oscilaciones para sumar 1)
    - “Adelantar” un reloj atrasado. Ej: sumar 1 cada 95 oscilaciones en un reloj de 100 Mhz nominales (menos oscilaciones para sumar 1)

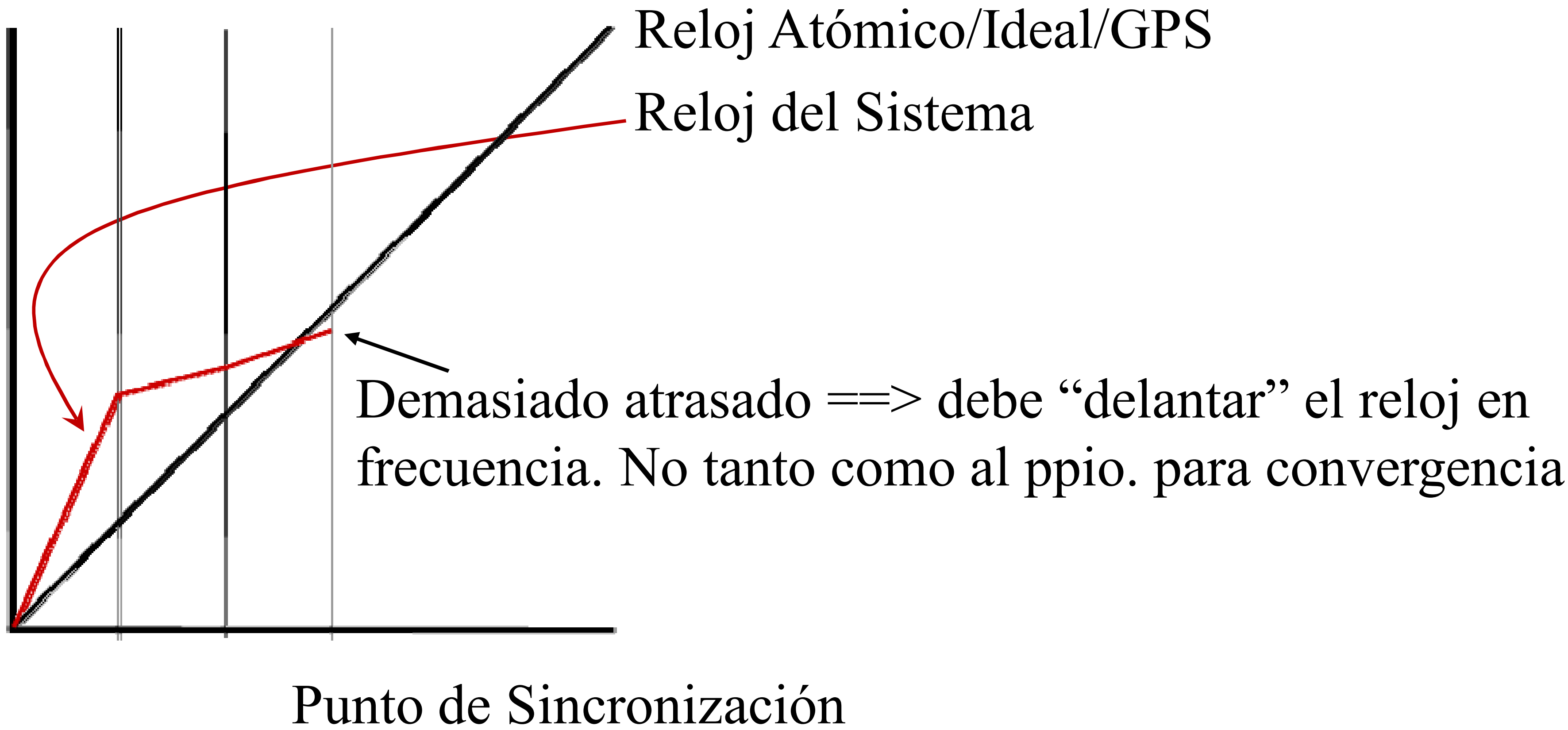
Reloj Físico en un Sistema Operativo



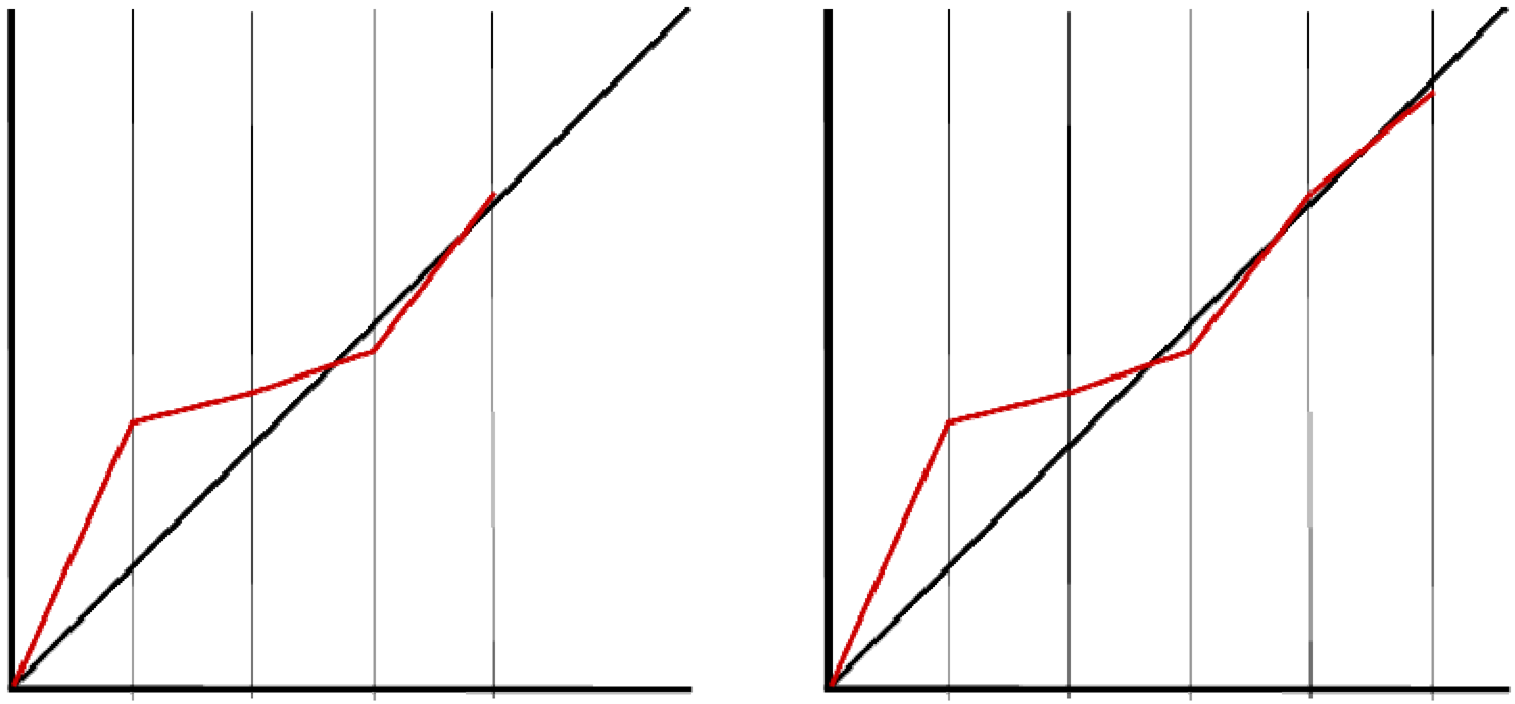
Reloj Físico en un Sistema Operativo



Reloj Físico en un Sistema Operativo

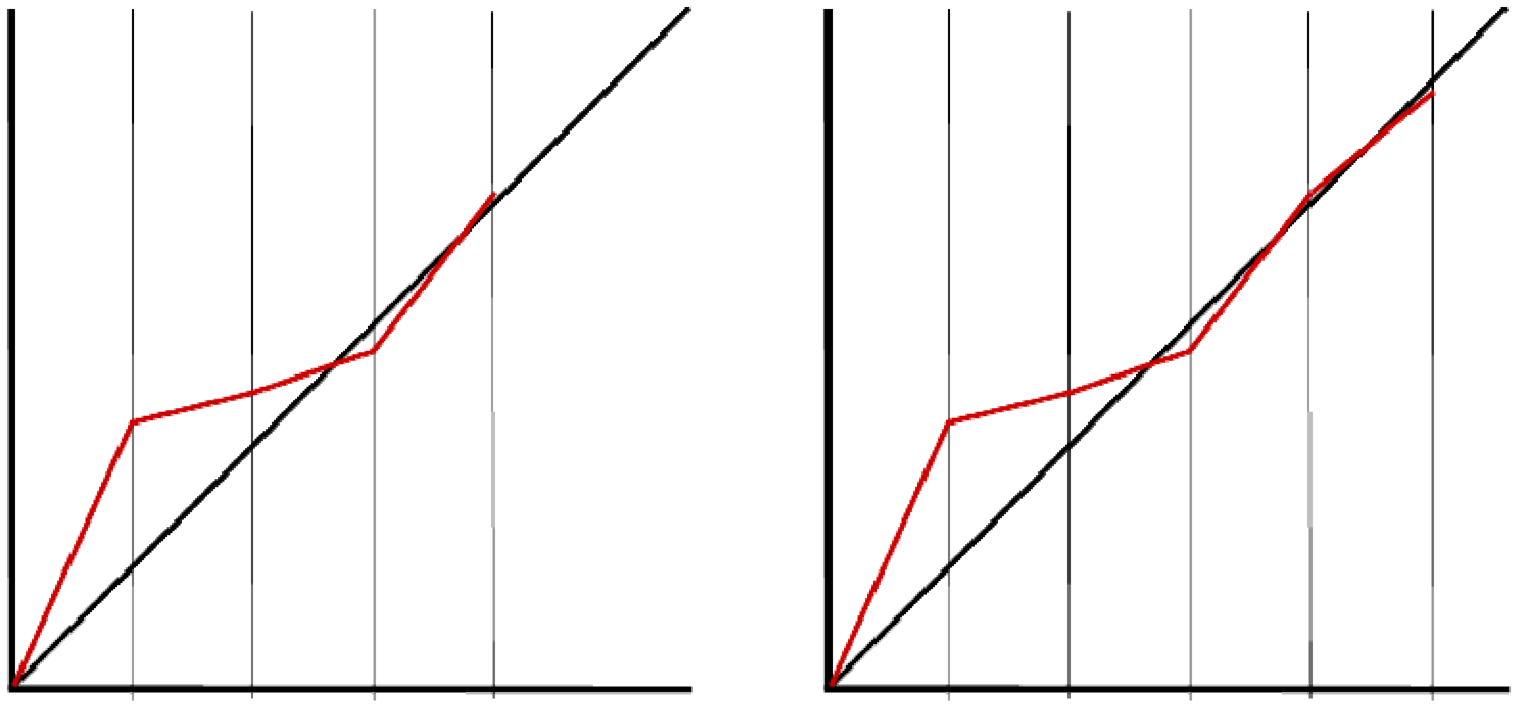


Reloj Físico en un Sistema Operativo



A medida que pasa más tiempo, en general se tiene menor error (las frec. físicas no suelen variar mucho)

Reloj Físico en un Sistema Operativo



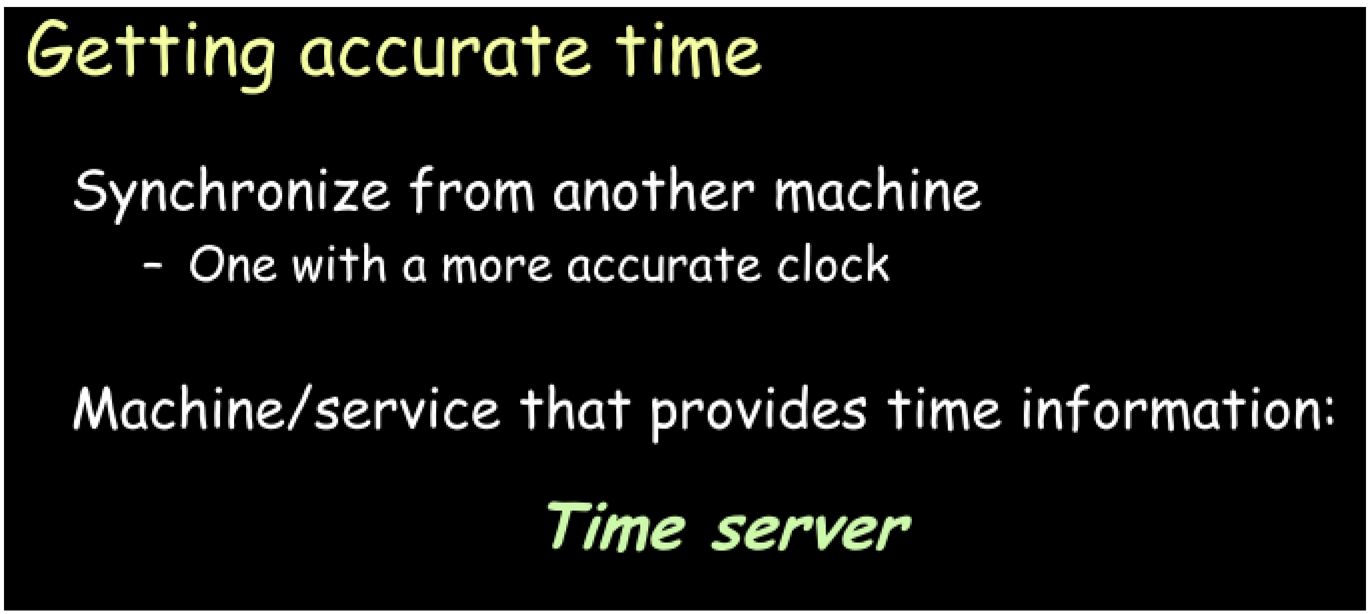
Se verifica/corrije la sincronización en intervalos constantes, pero podría definirse más frecuente al principio

Sincronización de Relojes

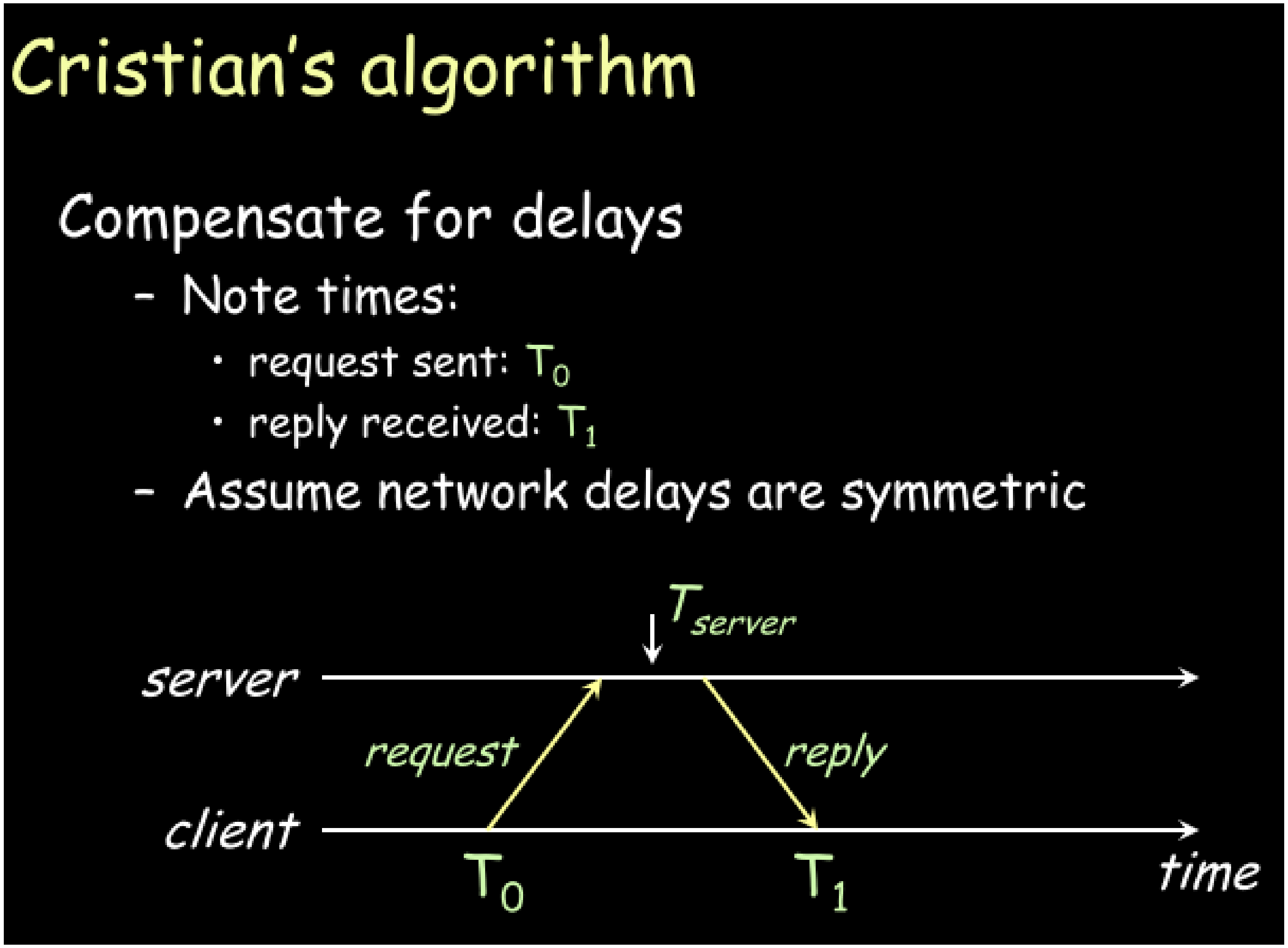
- Un reloj de referencia en otra computadora
    - Atómico
    - GPS
    - El reloj local de la otra computadora

Sincronización de Relojes

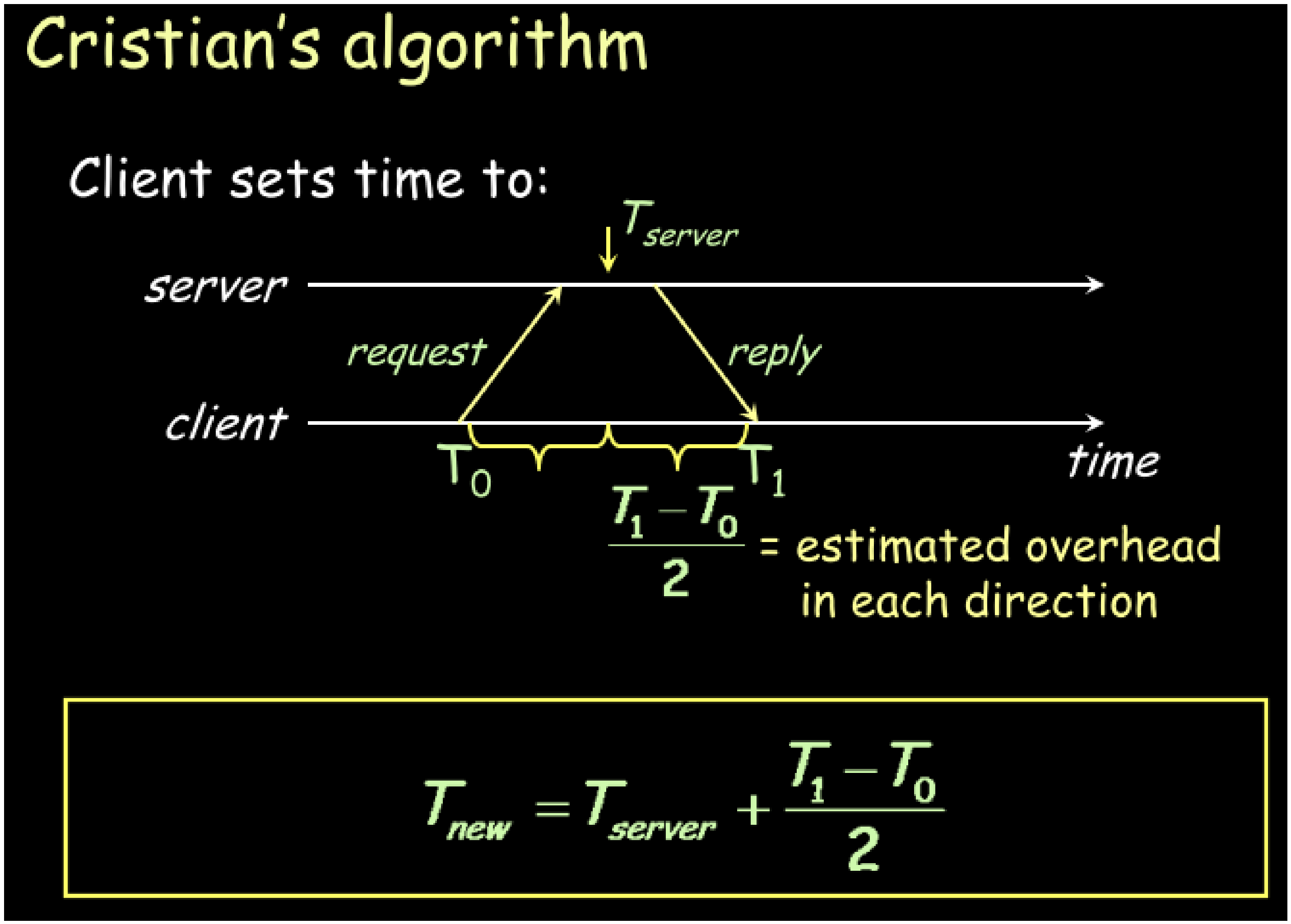
- Un reloj de referencia en otra computadora
  - Atómico
  - GPS
  - El reloj local de la otra computadora



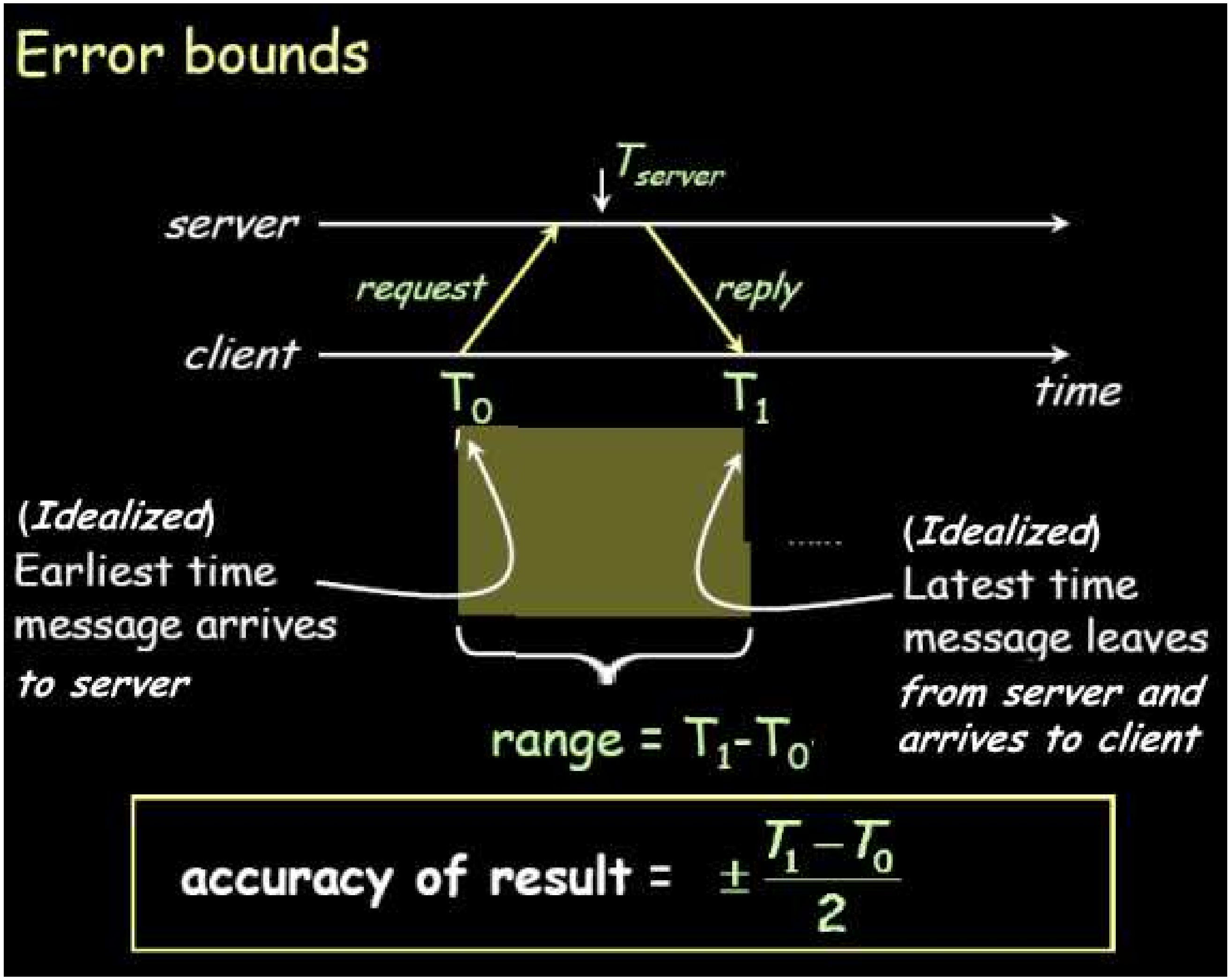
Sincronización de Relojes



Sincronización de Relojes



Sincronización de Relojes



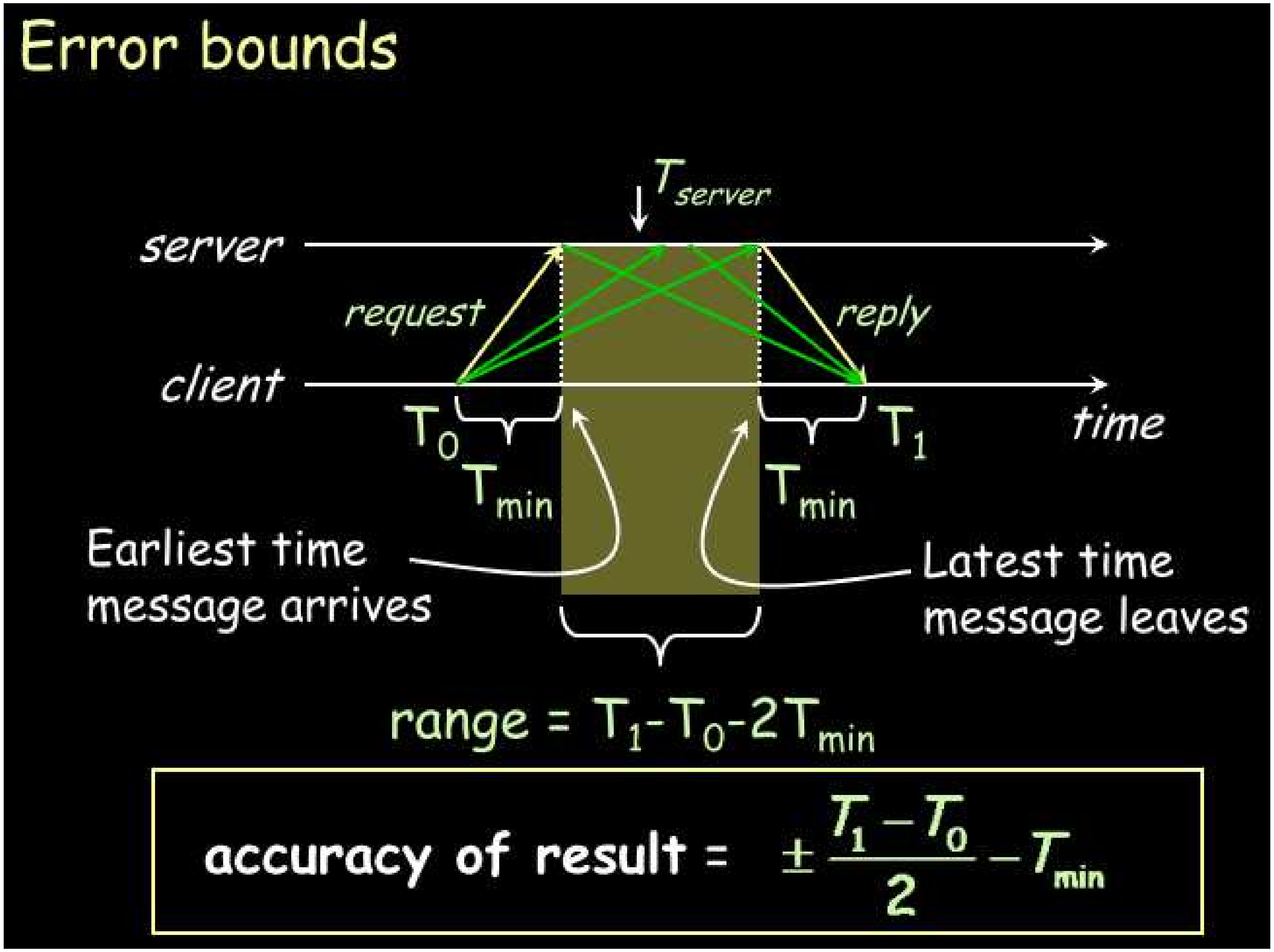
Sincronización de Relojes

Error bounds

If minimum message transit time (T_{min}) is known:

Place bounds on accuracy of result

Sincronización de Relojes



Sincronización de Relojes

- Cristian
  - Servidor de tiempo
  - Requerimiento/Respuesta
  - Localmente: servidor + reloj local
  - Error acotado por los tiempos de comunicaciones
    - Hay error aún en condiciones ideales
      - $T_1 - T_0$ (en el reloj local, “estimando” servidor)
  - Unico punto de falla: servidor
    - Se podría “cambiar” servidor, pero no es sencillo

Sincronización de Relojes

Berkeley Algorithm

- Gusella & Zatti, 1989

- Assumes no machine has an accurate time source
- Obtains average from participating computers
- Synchronizes all clocks to average

Sincronización de Relojes

Berkeley Algorithm

- Machines run **time dæmon**
  - Process that implements protocol
- One machine is elected (or designated) as the server (**master**)
  - Others are **slaves**

Sincronización de Relojes

Berkeley Algorithm

- Master polls each machine periodically
  - Ask each machine for time
    - Can use Cristian's algorithm to compensate for network latency
- When results are in, compute average
  - Including master's time
- *Hope: average cancels out individual clock's tendencies to run fast or slow*
- Send offset by which each clock needs adjustment to each slave
  - Avoids problems with network delays if we send a time stamp

Sincronización de Relojes

Berkeley Algorithm

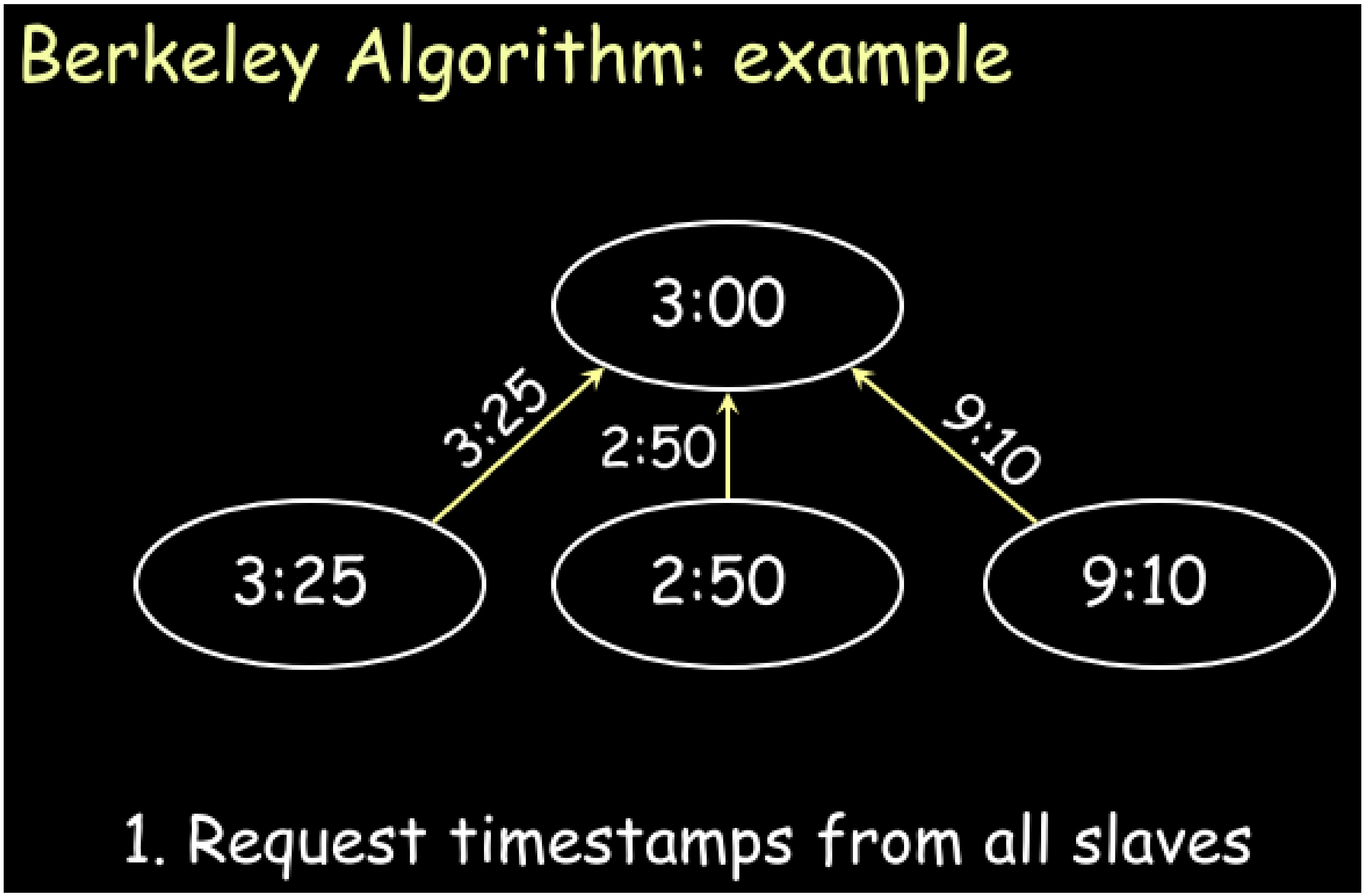
Algorithm has provisions for ignoring readings from clocks whose skew is too great

- Compute a **fault-tolerant average**

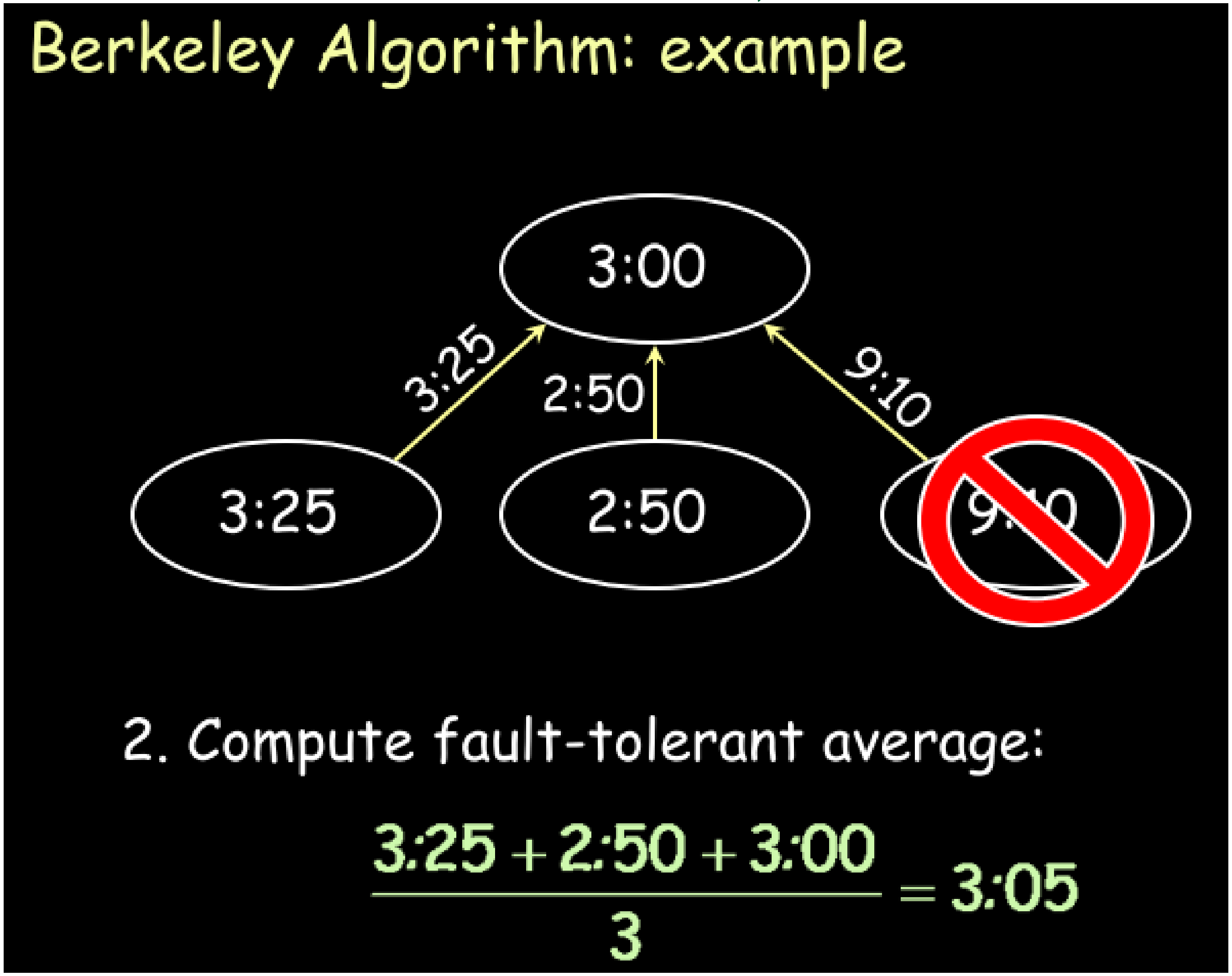
If master fails

- Any slave can take over

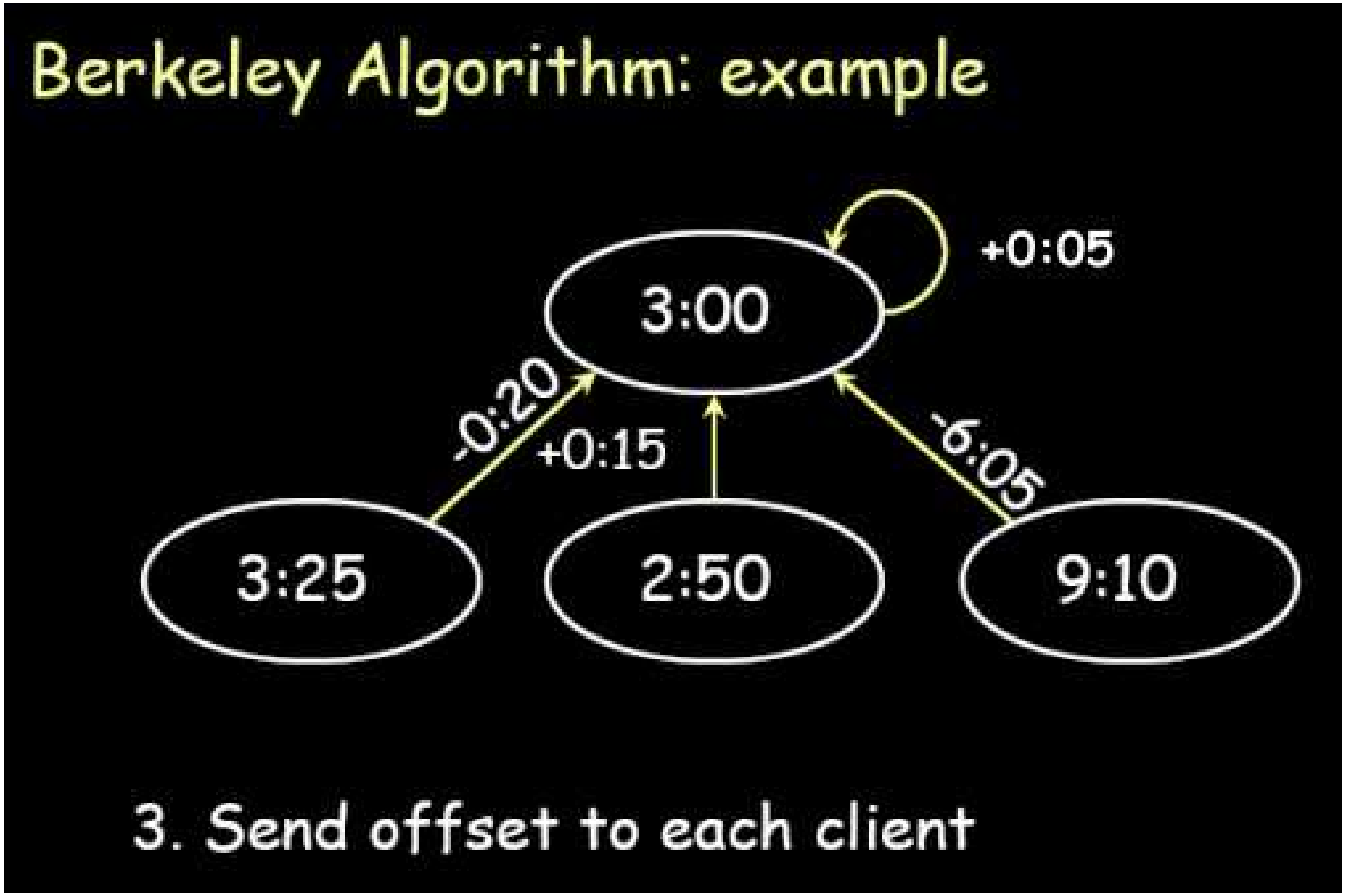
Sincronización de Relojes



Sincronización de Relojes



Sincronización de Relojes



Sincronización de Relojes

- Berkeley
  - Más práctico que teórico (a diferencia de Cristian)
  - Todos los relojes (excepto *descartados*)
  - Comunicaciones colectivas
    - Requerimiento de relojes locales (*broadcast*)
    - Envío de las correcciones (*scatter*)
  - Cálculo centralizado
  - Implementado
    - Cualquiera puede ser master
    - Reemplazo de master (sin punto único de falla)

Sincronización de Relojes

Network Time Protocol, NTP

1991, 1992

Internet Standard, version 3: RFC 1305

RFC 5905 - Network Time Protocol Version 4: Protocol and Algorithms Specification
Updated by RFC 7822, 8573

Sincronización de Relojes

NTP Goals

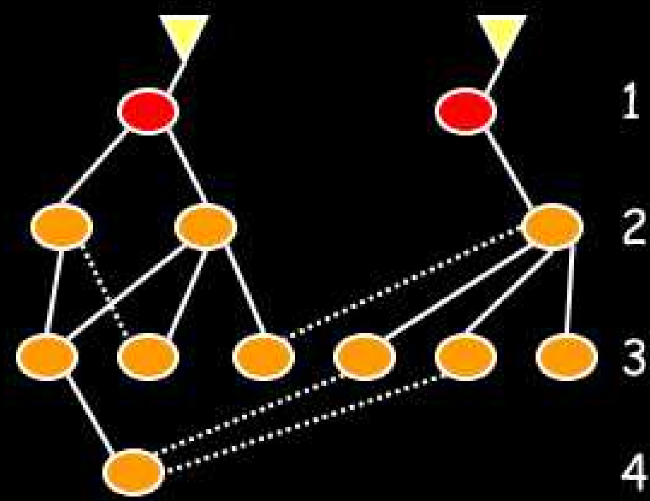
- Enable clients across Internet to be accurately synchronized to UTC despite message delays
  - Use statistical techniques to filter data and gauge quality of results
- Provide reliable service
  - Survive lengthy losses of connectivity
  - Redundant paths
  - Redundant servers
- Enable clients to synchronize frequently
  - offset effects of clock drift
- Provide protection against interference
  - Authenticate source of data

Sincronización de Relojes

NTP servers

Arranged in strata

- 1st stratum: machines connected directly to accurate time source
- 2nd stratum: machines synchronized from 1st stratum machines
- ...



Sincronización de Relojes

NTP Synchronization Modes

Multicast mode

- for high speed LANS
- Lower accuracy but efficient

Procedure call mode

- Similar to Cristian's algorithm

Symmetric mode

- Intended for master servers
- Pair of servers exchange messages and retain data to improve synchronization over time

All messages delivered unreliably with UDP

Sincronización de Relojes

NTP messages

- Procedure call and symmetric mode
  - Messages exchanged in pairs
- NTP calculates:
  - **Offset** for each pair of messages
    - Estimate of offset between two clocks
  - **Delay**
    - Transmit time between two messages
  - **Filter Dispersion**
    - Estimate of error – quality of results
    - Based on accuracy of server's clock ***and*** consistency of network transit time
- Use this data to find preferred server:
  - *lower stratum & lowest total dispersion*

Sincronización de Relojes

NTP message structure

- Leap second indicator
  - Last minute has 59, 60, 61 seconds
- Version number
- Mode (symmetric, unicast, broadcast)
- Stratum (1=primary reference, 2-15)
- Poll interval
  - Maximum interval between 2 successive messages, nearest power of 2
- Precision of local clock
  - Nearest power of 2

Sincronización de Relojes

NTP message structure

- Root delay
  - Total roundtrip delay to primary source
  - (16 bits seconds, 16 bits decimal)
- Root dispersion
  - Nominal error relative to primary source
- Reference clock ID
  - Atomic, NIST dial-up, radio, LORAN-C navigation system, GOES, GPS, ...
- Reference timestamp
  - Time at which clock was last set (64 bit)
- Authenticator (key ID, digest)
  - Signature (ignored in SNTP)

Sincronización de Relojes

NTP message structure

- T_1: originate timestamp
  - Time request departed client (client's time)
- T_2: receive timestamp
  - Time request arrived at server (server's time)
- T_3: transmit timestamp
  - Time request left server (server's time)

Sincronización de Relojes

NTP's validation tests

- Timestamp provided ≠ last timestamp received
  - duplicate message?
- Originating timestamp in message consistent with sent data
  - Messages arriving in order?
- Timestamp within range?
- Originating and received timestamps ≠ 0?
- Authentication disabled? Else authenticate
- Peer clock is synchronized?
- Don't sync with clock of higher stratum #
- Reasonable data for delay & dispersion

Sincronización de Relojes

SNTP

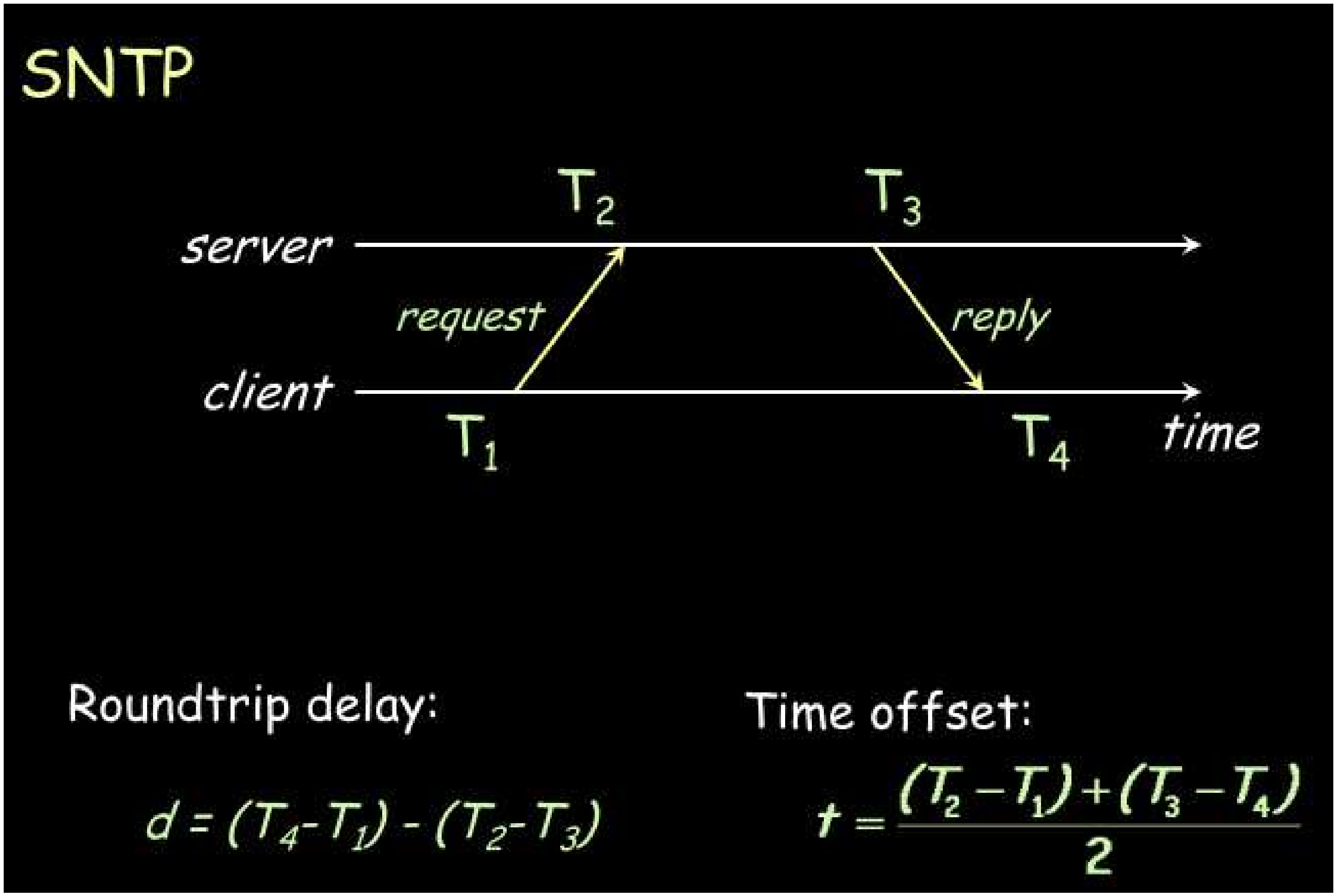
Simple Network Time Protocol

- Based on Unicast mode of NTP
- Subset of NTP, not new protocol
- Operates in multicast or procedure call mode
- Recommended for environments where server is root node and client is leaf of synchronization subnet
- Root delay, root dispersion, reference timestamp ignored

RFC 2030, October 1996

Incluido en RFC 5905 - Network Time Protocol Version 4: Protocol and Algorithms Specification

Sincronización de Relojes



Dudas/Consultas

- Plataforma Ideas