Sistemas Distribuídos
Aula 11

Roteiro

- Relógios
- Hora de referência
- Sincronizando relógios
- Algoritmo de Berkeley
- NTP

Relógio do Computador

- Como computador guarda o tempo?

- Igual a relógio digital: oscilador de cristal
  - cristal energizado oscila em frequência pré-determinada
  - cada oscilação incrementa um contador (*tic* de relógio)
- RTC: Real Time Clock
  - circuito integrado dedicado, alimentado também por bateria (computador desligado)
  - frequência de 32.768 kHz, que é 2^{15} tics por segundo
  - pode ser inicializado com um valor

Hora do Sistema

- RTC é chamado de *Hardware Clock* ou *BIOS Clock*
- Ao iniciar, SO utiliza o RTC para inicializar seu próprio relógio
  - *System Clock* ou *Kernel Clock*
  - também baseado em contagem de ticks (da CPU) e adaptações (por software)
- Hora do Sistema (*System Time*): número de tics desde uma certa data (*epoch*)
  - Unix: 1 de janeiro de 1970 às 00:00:00 UT
- Convertida para Hora do Calendário (*Calendar Time*)
  - leva em consideração fuso horário, horário de verão, etc
  - 1000000000 de seg → 9/9/2001 às 01:46:40 UT

Acertando a Hora

- Como ajustar a hora do seu computador?

- **Ideia 0**: olha a hora no celular, copia o valor para computador (ajusta Hora do Sistema, ajusta RTC)

Funciona se seu computador não se comunicar!

- Leituras sequenciais do relógio serão monotônicas

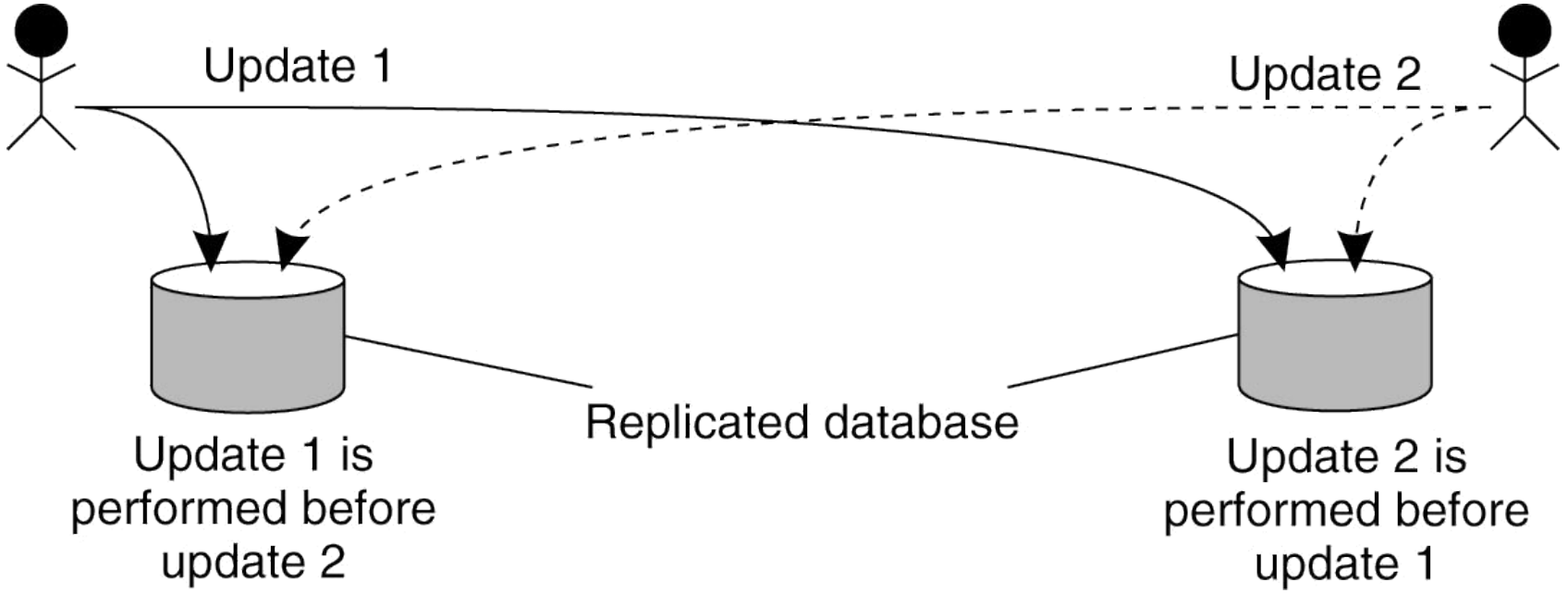
Exemplo com Dropbox

- Relógios ajustados com ideia 0
- Trabalho em grupo, usando Dropbox
  - supor que Dropbox utiliza hora de gravação do arquivo (*system time* local)
- Você edita, salva, Dropbox atualiza no seu amigo (seu relógio está na frente)
- Seu amigo edita, salva, mas Dropbox **não atualiza** no seu pois relógio dele está atrás (hora de gravação do arquivo é anterior)!

Evento ocorrendo depois é tido como anterior!

Exemplo com Banco de Dados Distribuído

- Dois usuários atualizam banco de dados distribuído (replicado)



- Réplica 1: U1 antes de U2
- Réplica 2: U2 antes de U1
- Como impor a mesma ordem nas réplicas?

Sincronizando Relógios

- Sincronização de relógios: dois relógios marcando exatamente a mesma hora

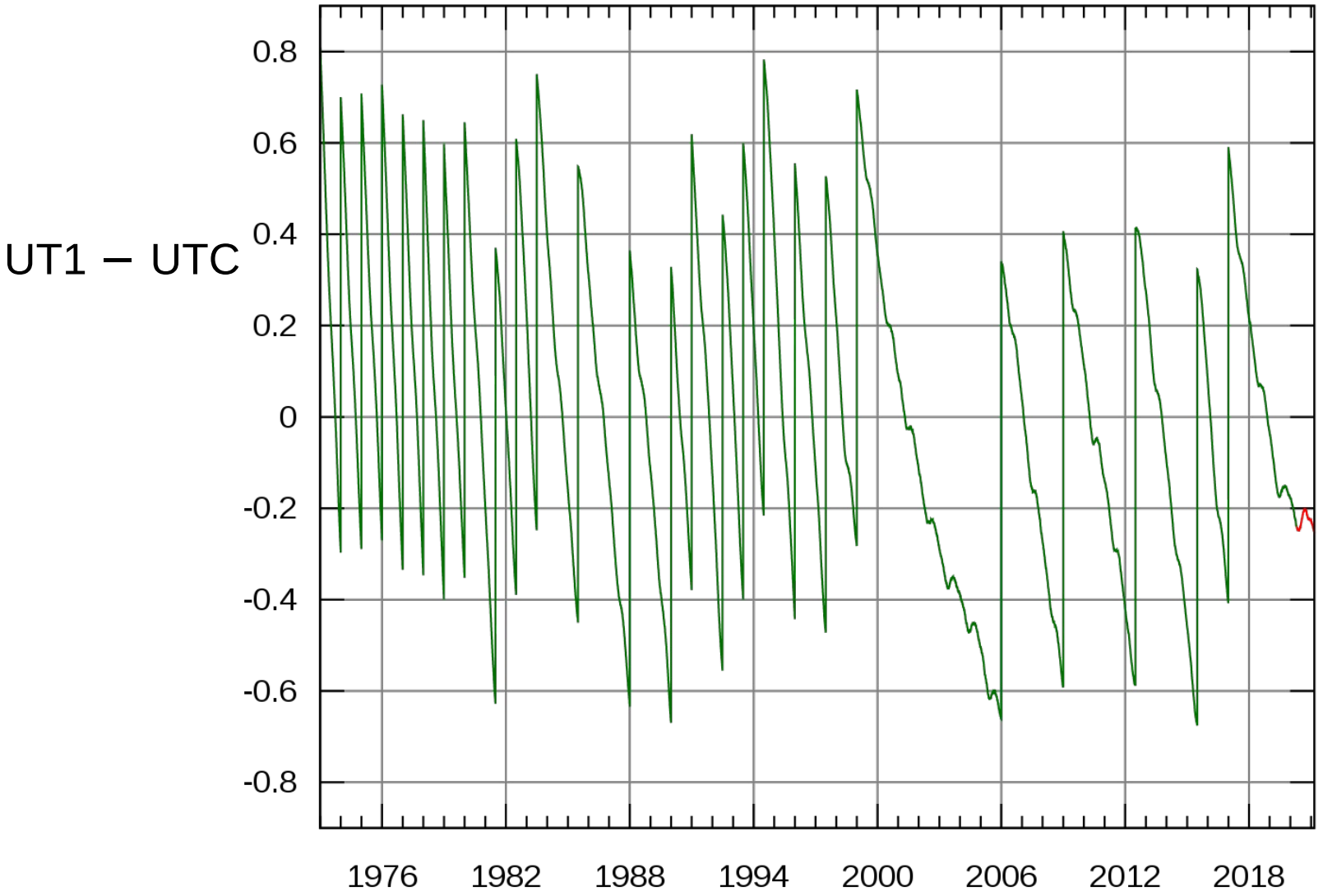
- Problema 1: Qual hora eles devem marcar?
- Problema 2: Como ajustar os relógios?

- Com relógios (perfeitamente) sincronizados, problemas anteriores não ocorrem
  - hora dos eventos pode ser utilizada

Hora Certa

- Diferentes referências da *hora certa*
- UT1: baseado em observações astronômicas
  - baixa precisão mas combina com o sol
- TAI: International Atomic Time
  - ciclos de radiação emitidos pelo Cesium
  - 9,192,631,770 ciclos p/seg (alta precisão)
  - diverge do UT1 pois rotação da terra está ficando mais devagar (menos dias por ano no futuro)
- UTC: TAI + ajustes para respeitar o UT1
  - mantém diferenças em menos de 1 segundo
  - adiciona um segundo (*leap second*) quando necessário

Divergência e *Leap Seconds*



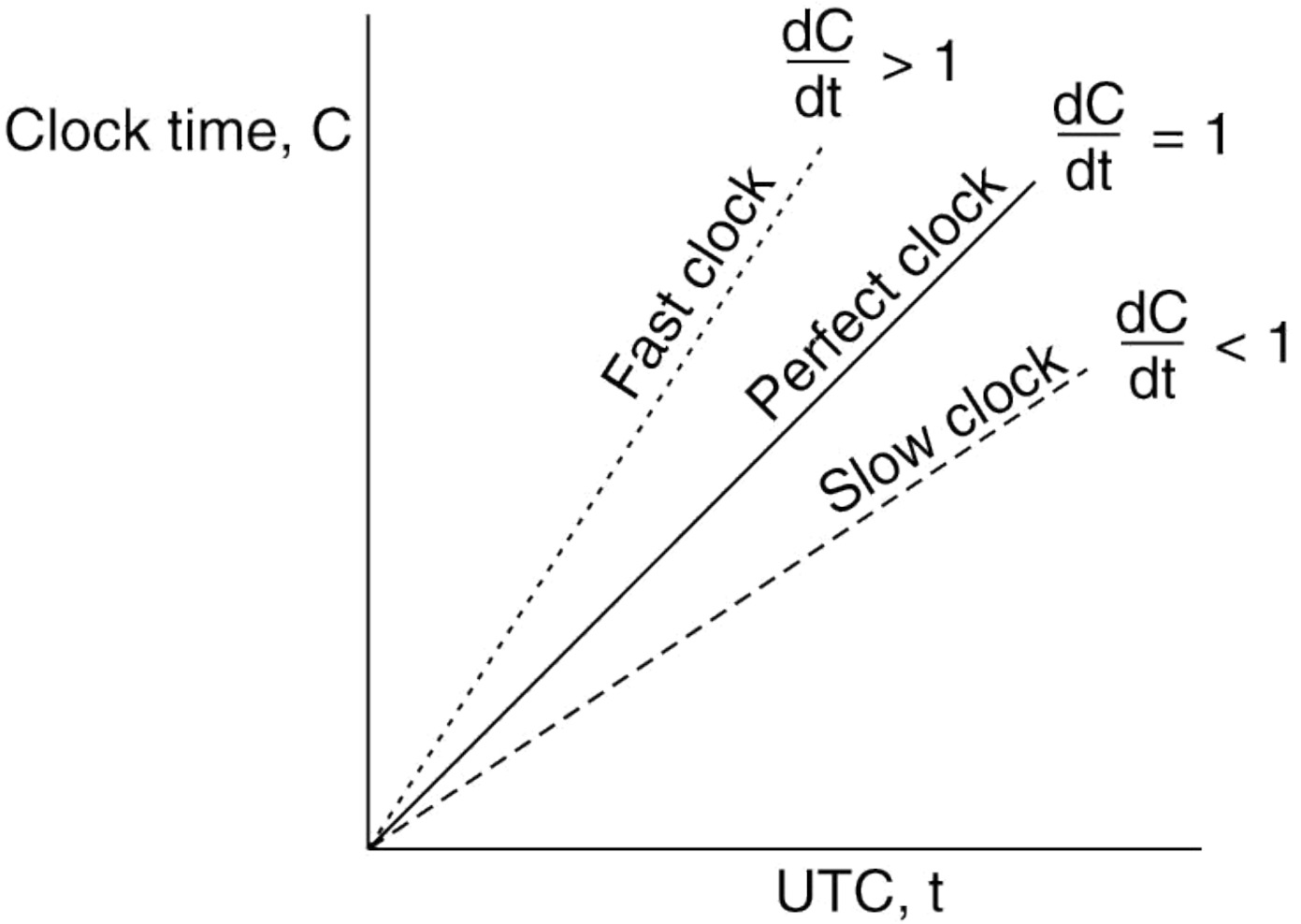
- *Leap second:* ajustes de 1s no UTC, último foi em 2016

Problemas com Relógios

- Nenhum oscilador é fundamentalmente idêntico a outro
- Diversos fatores afetam frequência: composição e fabricação, e fatores externos (tensão elétrica, temperatura, pressão, etc)
- Até os osciladores do TAI possuem variações

- **Problema fundamental: *clock drift***
  - frequência ligeramente diferentes, acumulam tempo (tics) em taxas diferentes
  - oscilador de cristal comum: erro aproximado de 1 segundo em 10 dias

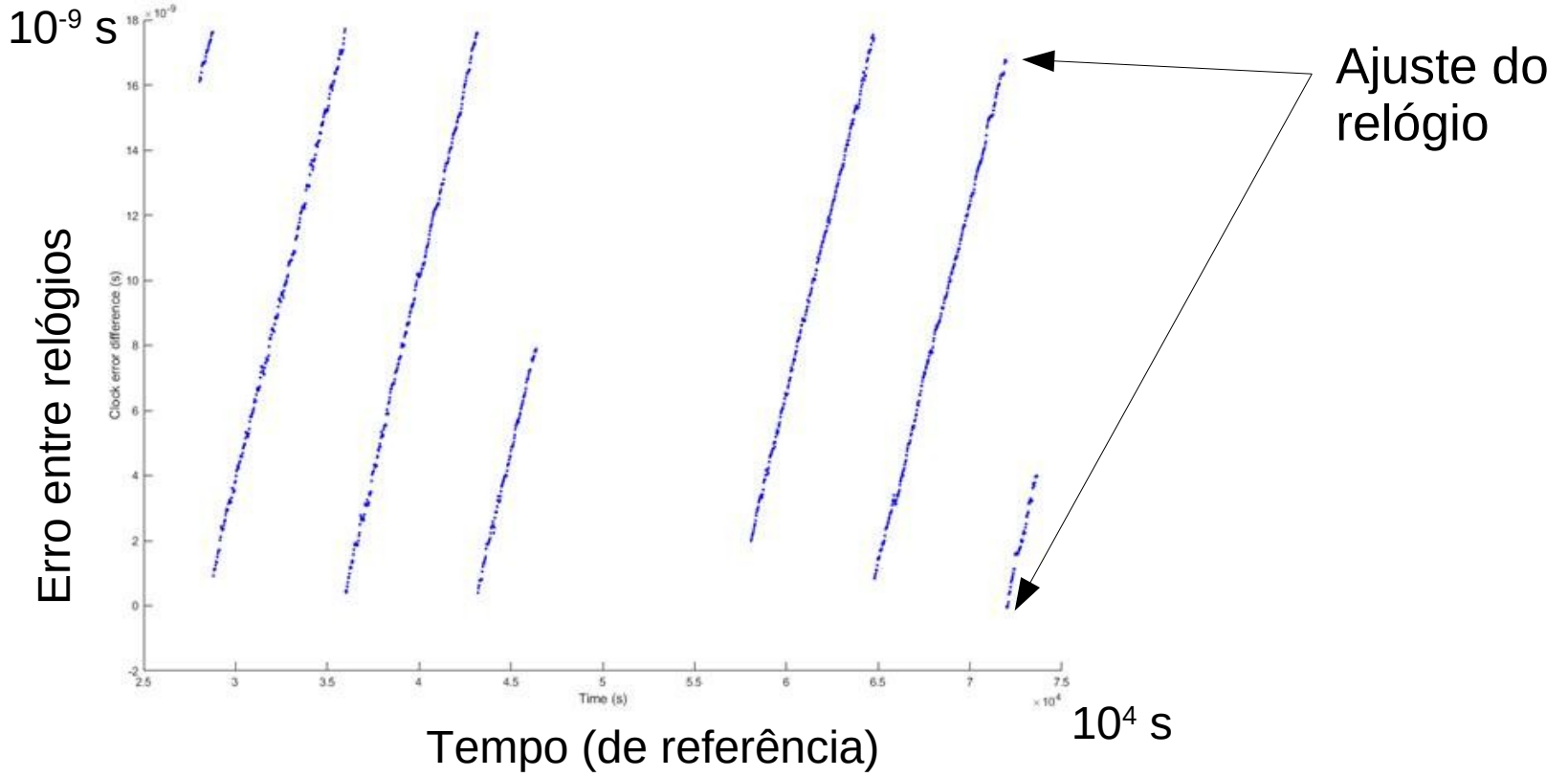
Exemplo de *Clock Drift*



- dC/dt = derivada do progresso do relógio com relação ao tempo (hora certa, UTC)

Exemplo Real

- Relógio atômico de um dos satélites usados no GPS (IIR-8 PRN 16)



Gustavo Mansur and Luiz Ferreira, “Error Behavior Of Atomic Clocks Aboard Gps Satellites” Bol. Ciênc. Geod. 25 (4) • 2019

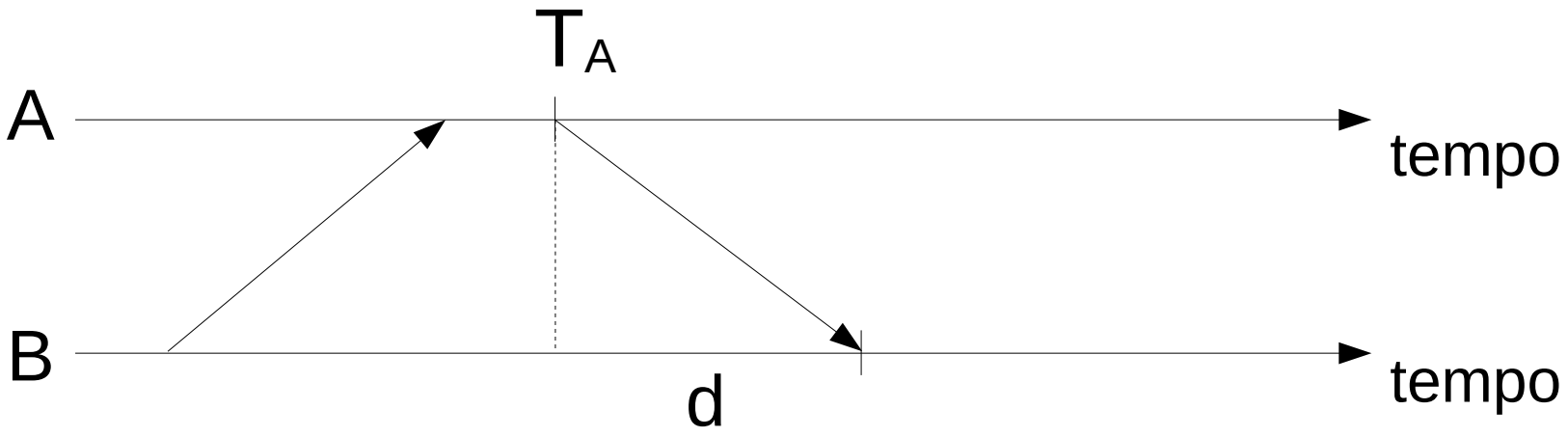
Ajustando Relógios

- *Clock drift:* ajustar constantemente os relógios dos computadores
  - verdade até para UTC!

- Como ajustar a hora do seu computador?

- **Ideia 1**: Perguntar a outro computador!
- Funciona? Quais serão os problemas?

Perguntando a Hora



- B envia pedido de hora para A
- A responde com sua hora, T_A
- Tempo entre verificar e mensagem chegar é d
- B acerta: $T_B = T_A + d$

- Problema: Como determinar o valor de d?

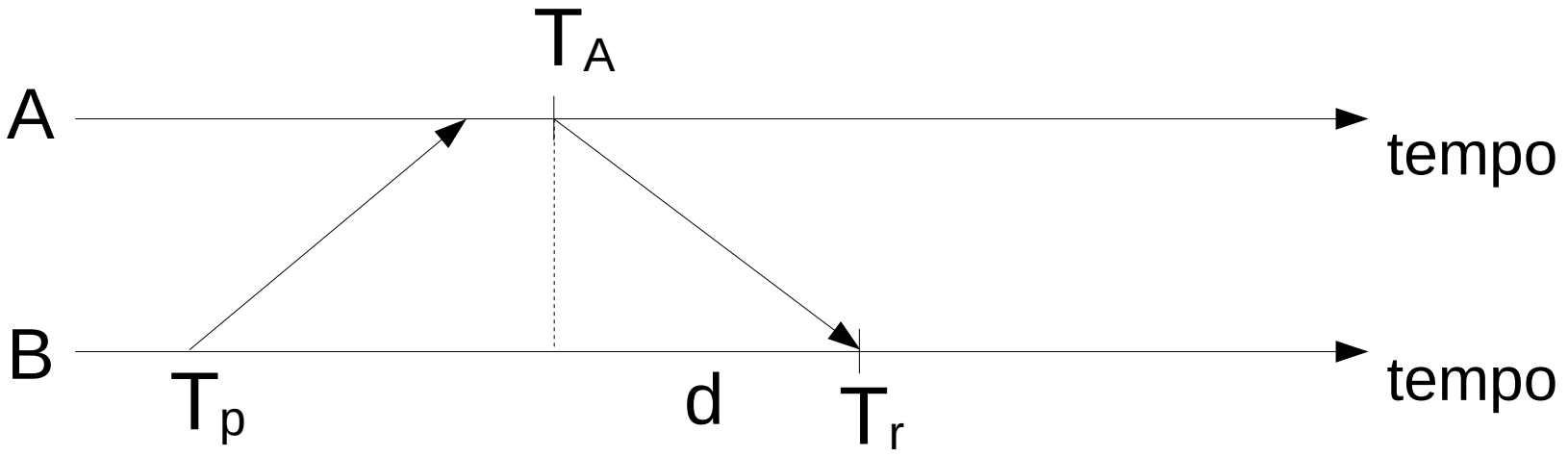
Lidando com Atraso

- Problema fundamental: Determinar d (na figura anterior)

- Tempo decorrido desde a leitura do RTC de A até a escrita no RTC de B!
- SO de A, interface de rede de A, transmissão pela rede (diversos enlaces), interface de rede de B, SO de B
- Atrasos são aleatórios, dependem de muitos fatores (ex. carga no SO, carga na rede)
- Tempo mais significativo: tempo de rede
  - principalmente com vários enlaces

Estimando d

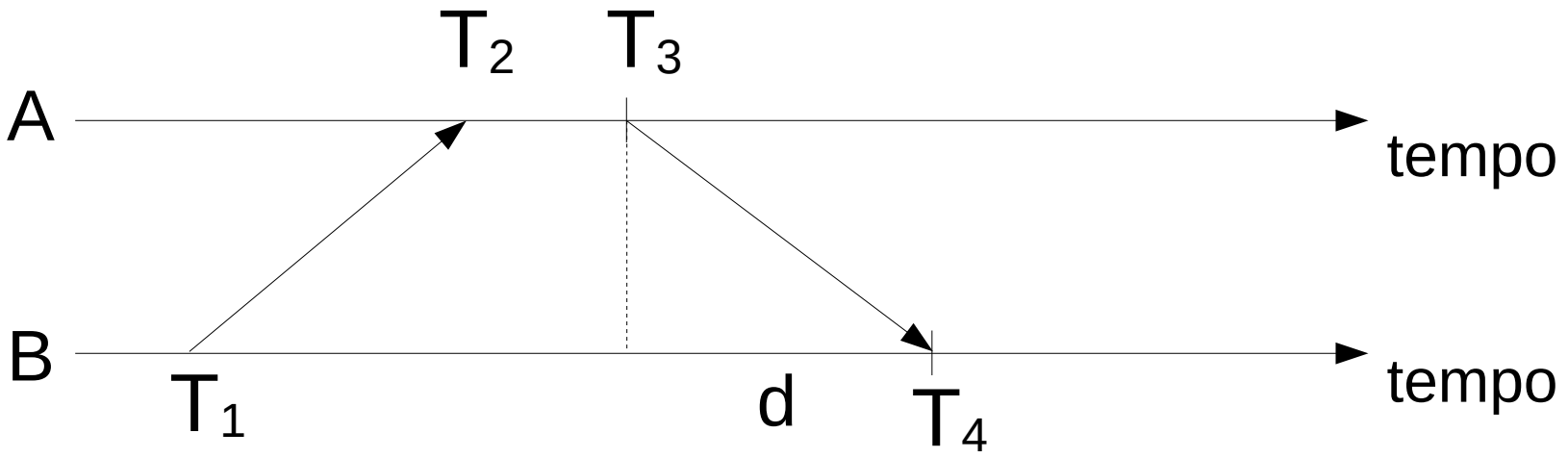
- **Ideia:** máquina B usa seu próprio relógio
  - T_p: horário que pedido é feito
  - T_r: horário que resposta chega
  - Estimar d = $(T_p - T_r) / 2$
  - B ajusta relógio para T_A + d



Problema: valor estimado é maior do que d (retardo entre chegada e saída em A)

Estimando d Melhor

- **Ideia:** marcar hora de chegada em A, descontar tempo entre chegada/saída

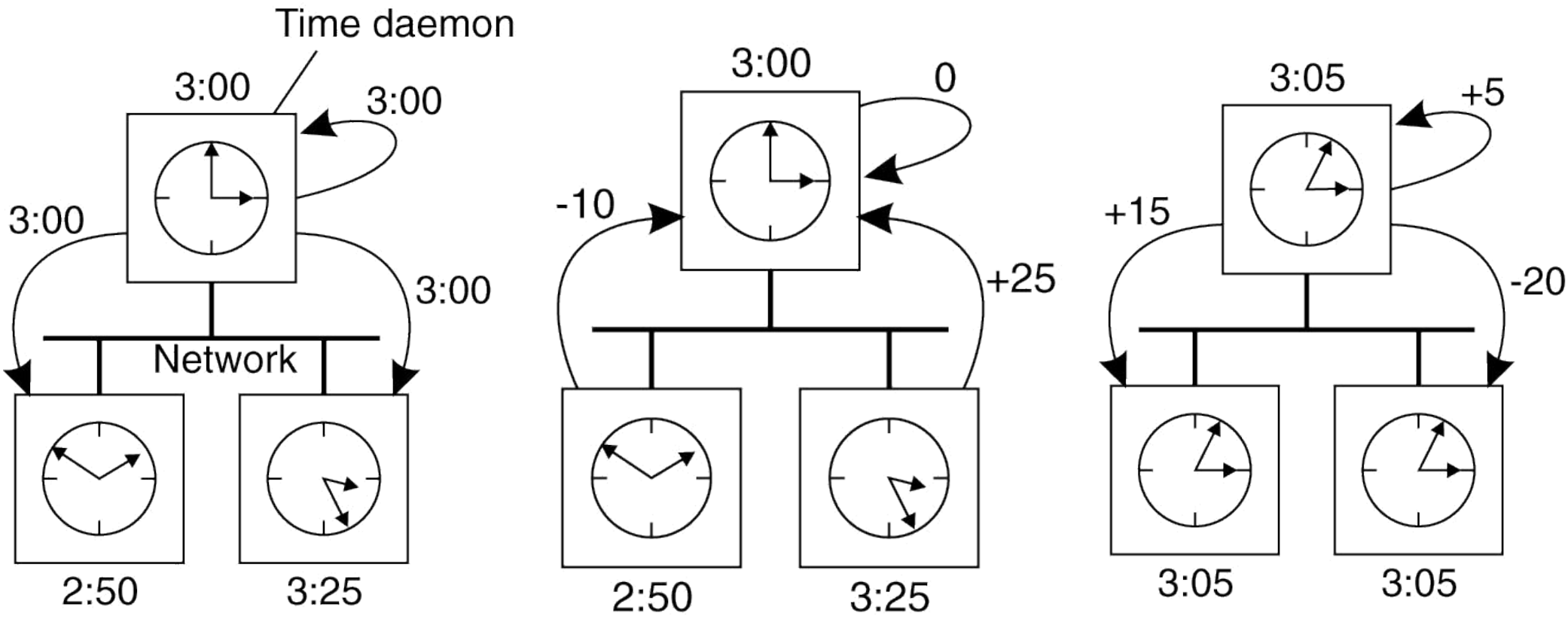


- Estimar d = $[(T_4 - T_1) - (T_3 - T_2)] / 2$
- B ajusta relógio para $T_3 + d$
- **Problema:** assume que retardo de ida igual ao de volta

Algoritmo de Berkeley

- Sincronização de relógio para máquinas na mesma rede local
  - originalmente, sem referência externa
- Servidor informa sua hora para máquinas clientes
- Máquinas clientes respondem com diferença entre valor recebido e seus relógios
- Servidor faz a média das diferenças recebidas (removendo possíveis outliers)
- Servidor transmite para cada cliente ajuste necessário
- Clientes fazem ajustes em seus relógios

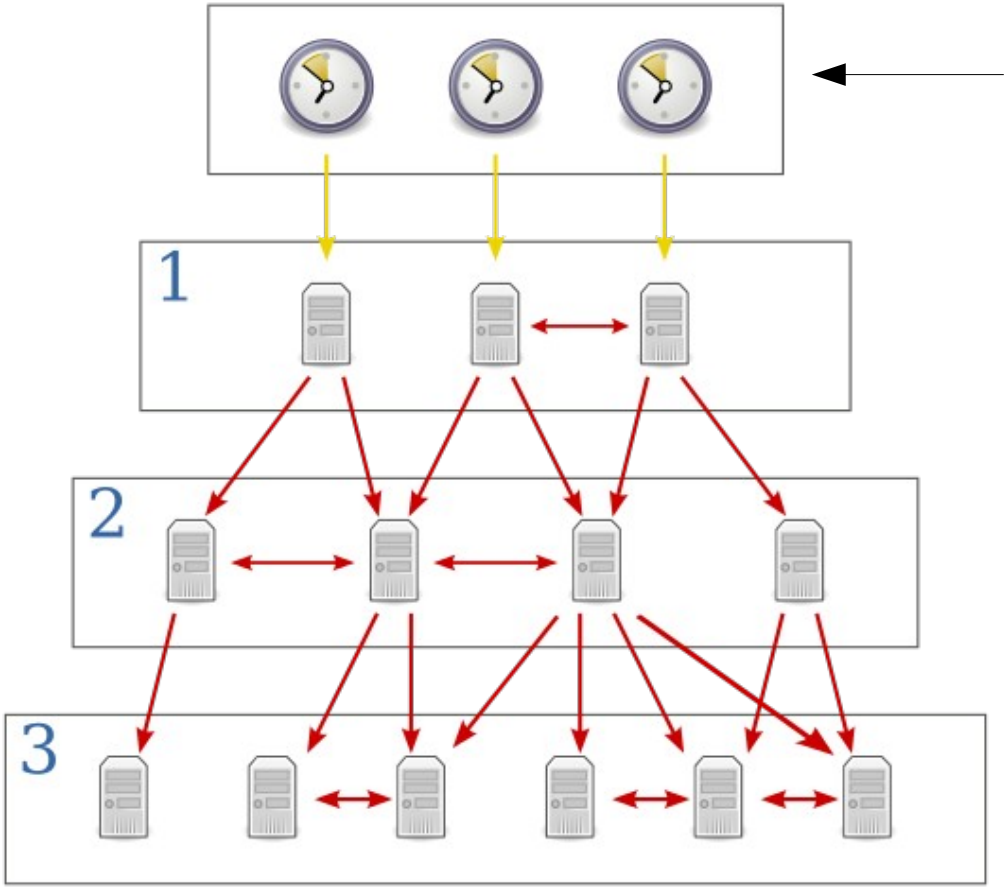
Exemplo



- 1) Servidor envia hora a todos clientes (*broadcast*)
- 2) Clientes respondem com diferenças
- 3) Servidor faz média e responde com ajustes
- Não possui referência externa (mas poderia)

NTP: Network Time Protocol

- Serviço de sincronização de relógios para a Internet, utilizando UTC como referência
- Hierarquia de servidores (*time server*), modelo cliente/servidor



Nível 0 são os servidores globais de UTC, precisão de relógio atômico

Menos precisão!

Computadores finais, precisão de alguns ms com UTC (mas varia)

NTP: Network Time Protocol

- Cliente periodicamente solicita hora a três ou mais servidores
- Cliente estima d como no algoritmo apresentado (slide 17)
- Determina hora certa fazendo média das horas estimadas, usando filtros (ex. elimina *outliers*)
- **Não seta a hora diretamente!**
- Atualiza a frequência do relógio do sistema
  - frequência de tics por segundo é variável e definida por software (*x* tics virtuais por tic do HW)
- Vantagens: hora do sistema nunca volta no tempo (monotônica); mudanças de hora mais suaves

NTP em Ação

- Sincronização continuada do relógio do sistema
- SO faz controle fino da frequência do relógio do sistema (usando NTP)
- Objetivo: Manter erro em torno de zero