Sistemas Distribuídos
Aula 11

Aula passada

- Modelo computação distribuída
- RPC
- Marshalling e stubs
- Semântica operacional
- RMI

Aula de hoje

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

Relógio do Computador

- Como computador mantém o tempo?

- Igual a relógio digital: oscilador de cristal
  - cristal oscila com 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
- *System time*: define zero em algum ponto de referência (depende do SO)

Acertando a Hora

- Como ajustar a hora do seu computador?

- **Ideia 0**: olha a hora no seu celular, copia o valor para computador (ajusta o RTC)
- Funciona? Problemas?

Funciona se seu computador estiver isolado

- uma leitura do relógio feita depois da outra sempre retorna um valor maior

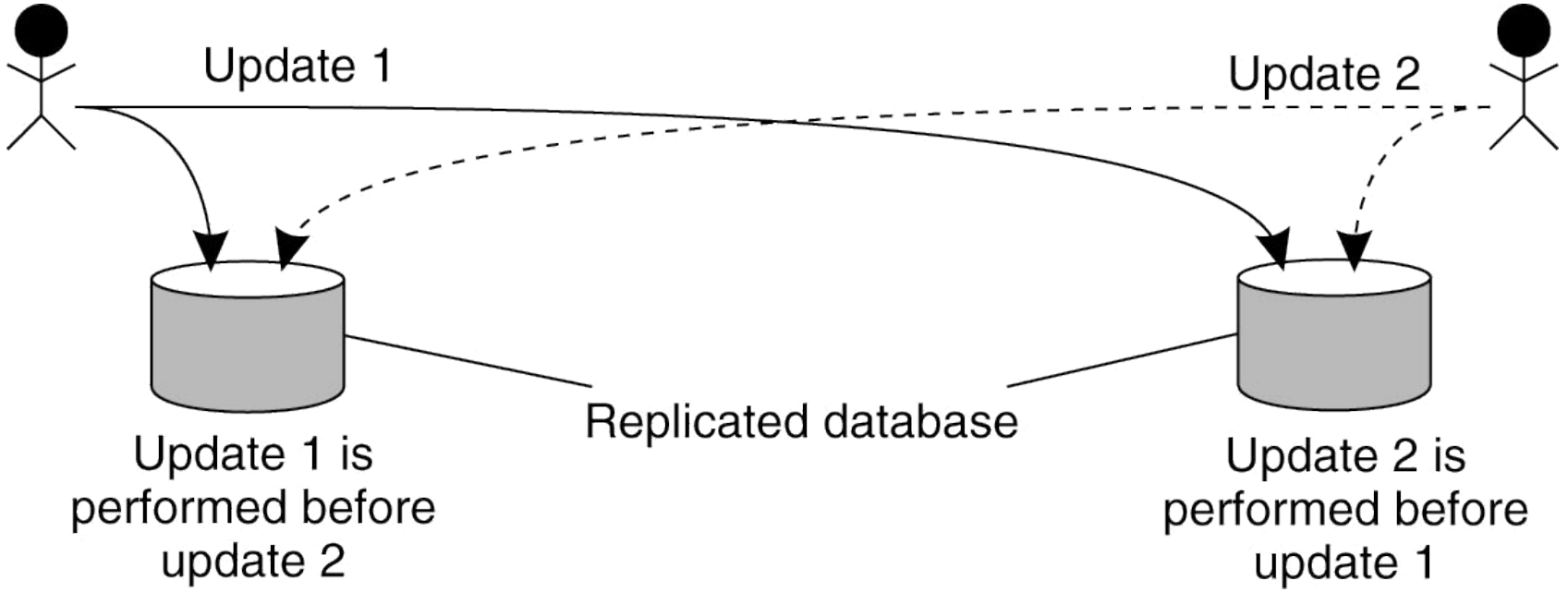
Exemplo com Dropbox

- Relógios ajustados com ideia 0
- Trabalho em grupo, usando Dropbox
  - supor que registro de tempo usa hora de gravação do arquivo (*system time* local)
- Você edita, salva, Dropbox atualiza (seu relógio está na frente)
- Seu amigo edita, salva, mas Dropbox **não atualiza** pois relógio dele está atrás!

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
- Qual aconteceu primeiro?

Fundamental!

Sincronizando Relógios

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

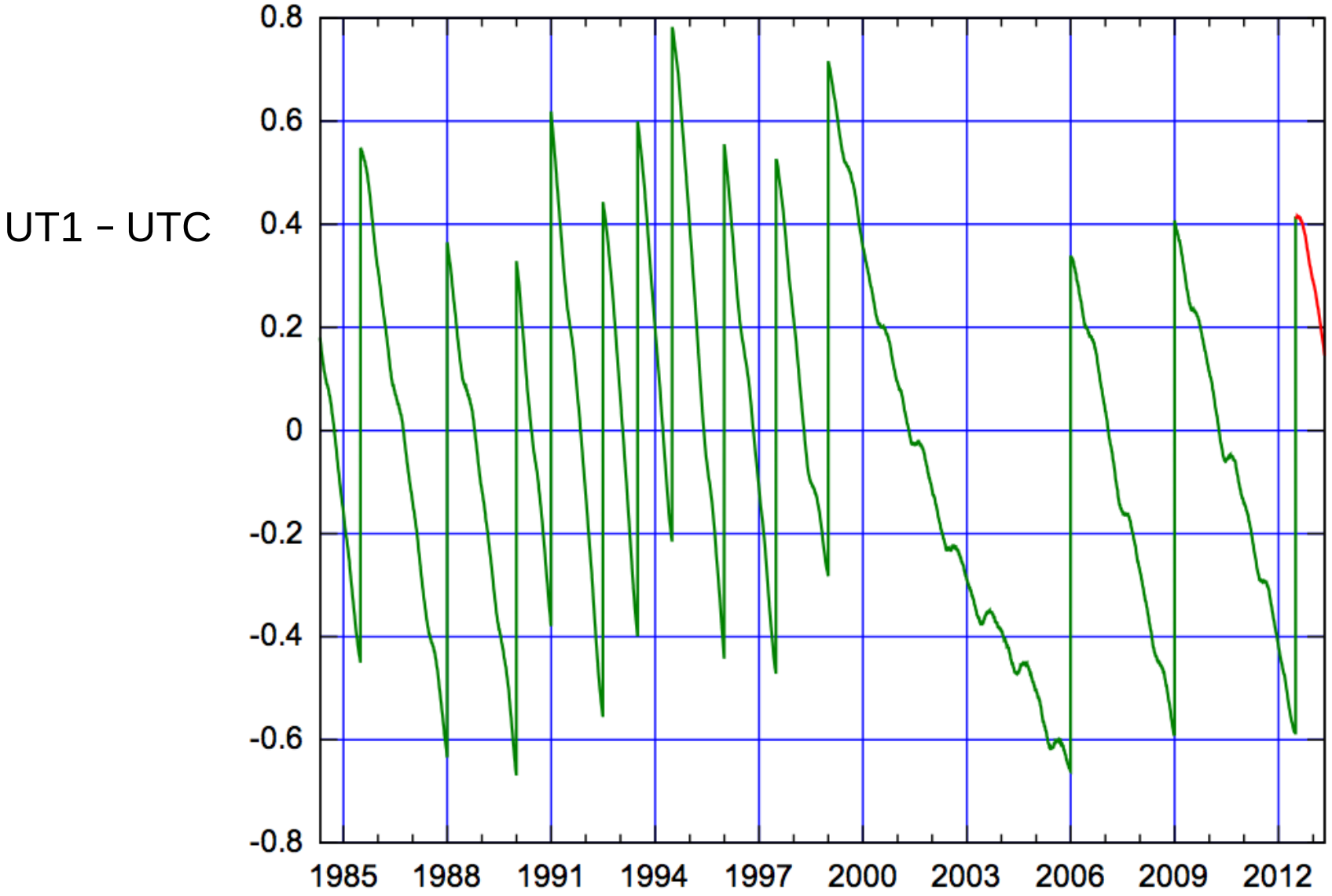
- Problema 1: Qual hora eles devem marcar?
- Problema 2: Como ajustar os relógios?

- Com relógios (perfeitamente) sincronizados, problemas anteriores não ocorrem!

Hora 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

Comparando Padrões

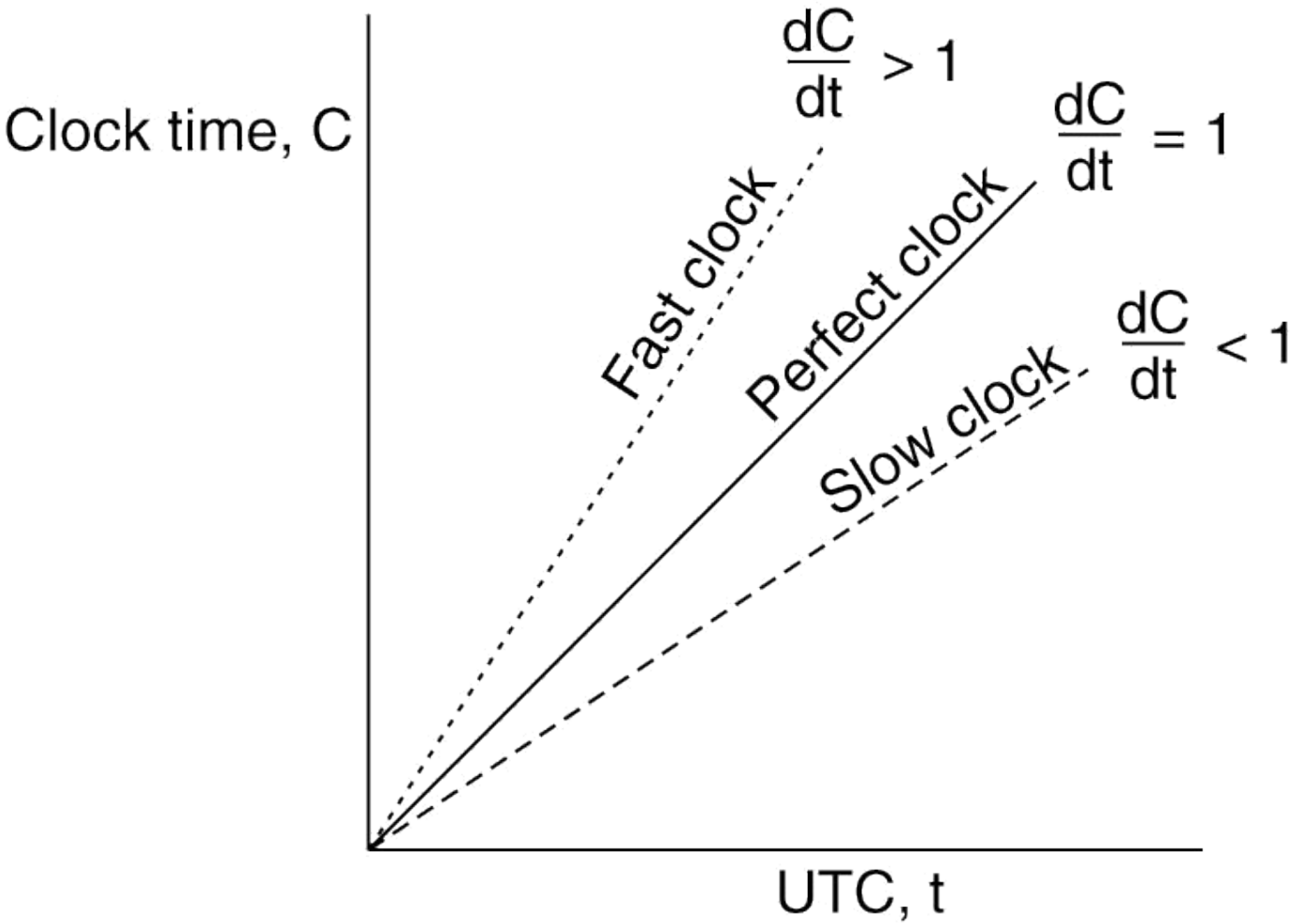


- *Leap second* são os ajustes!

Problemas com Relógios

- Nenhum oscilador é fundamentalmente idêntico a outro
- Diversos fatores afetam frequência: composição (fabricação) e fatores externos (tensão elétrica, temperatura, pressão, etc)
- Até os osciladores do TAI são diferentes!
  - vários são usados, média é calculada
- **Probema fundamental: *clock drift***
  - frequência ligeramente diferentes, acumulam tempo (tics) em taxas diferentes
  - oscilador de cristal: 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)

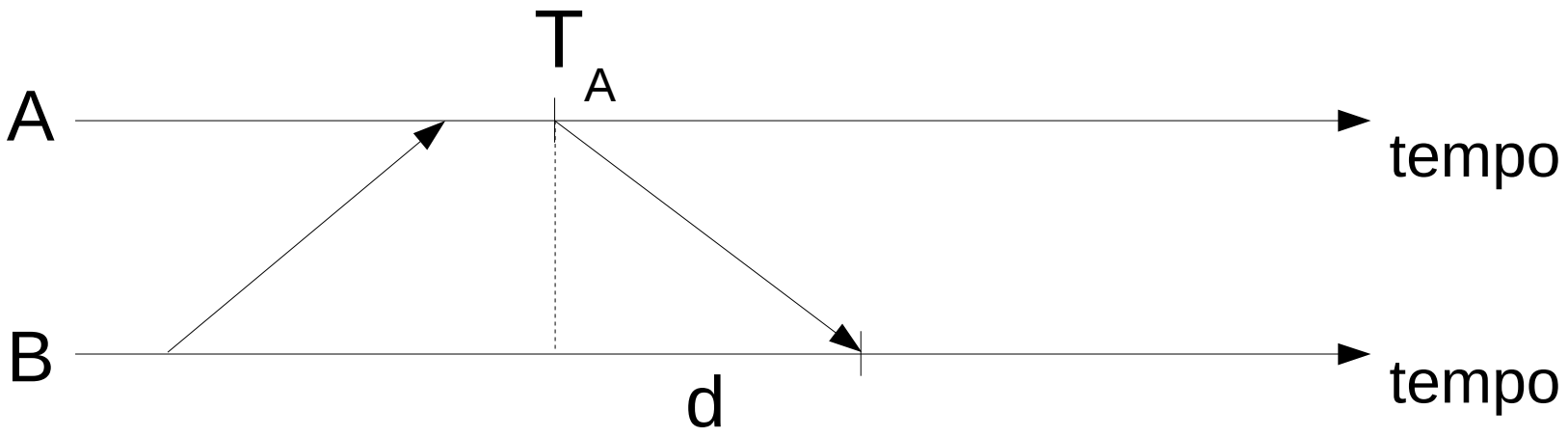
Ajustando Relógios

- *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? 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: $T_B = T_A + d$
- Funciona? Problemas?

Lidando com Atraso

- Problema fundamental: Estimando d (na figura anterior)
- O que determina d?

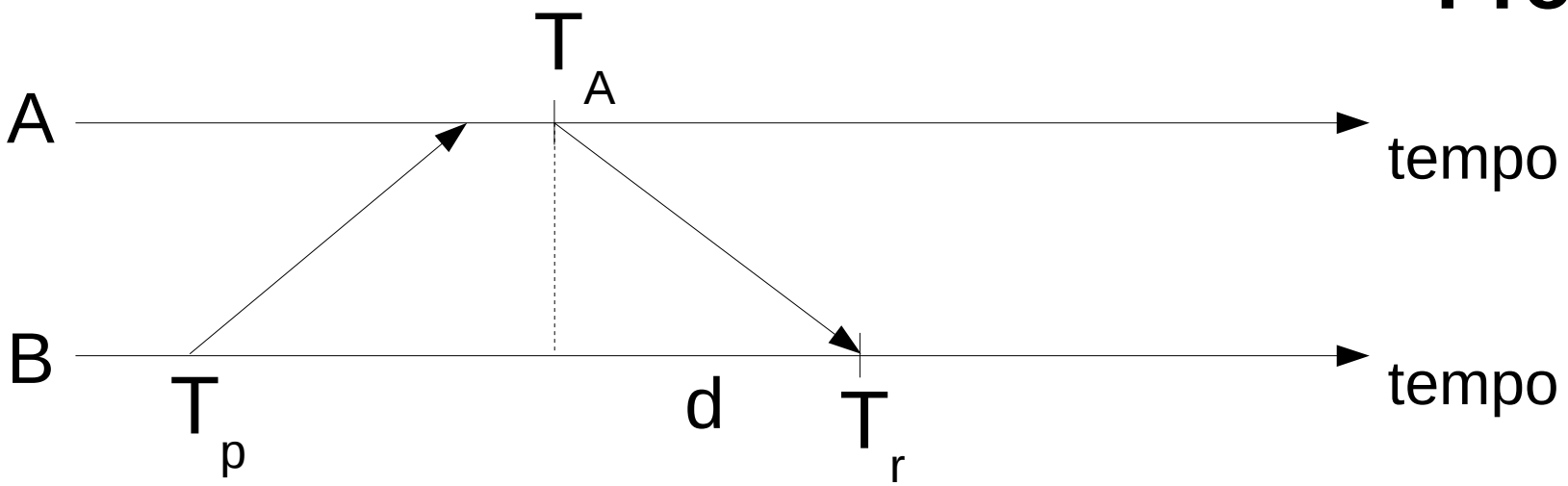
- Tempo decorrido desde a leitura do RTC de A até a escrita no RTC de B!
- SO de A, interface de rede de A, transmissão pela rede (diversos enlaces), interface de rede de B, SO de B
- Atrasos são aleatórios, dependem de muitos fatores (ex. carga no SO, carga na rede)
- Tempo mais significativo: tempo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

Estimando d

- Como estimar d?

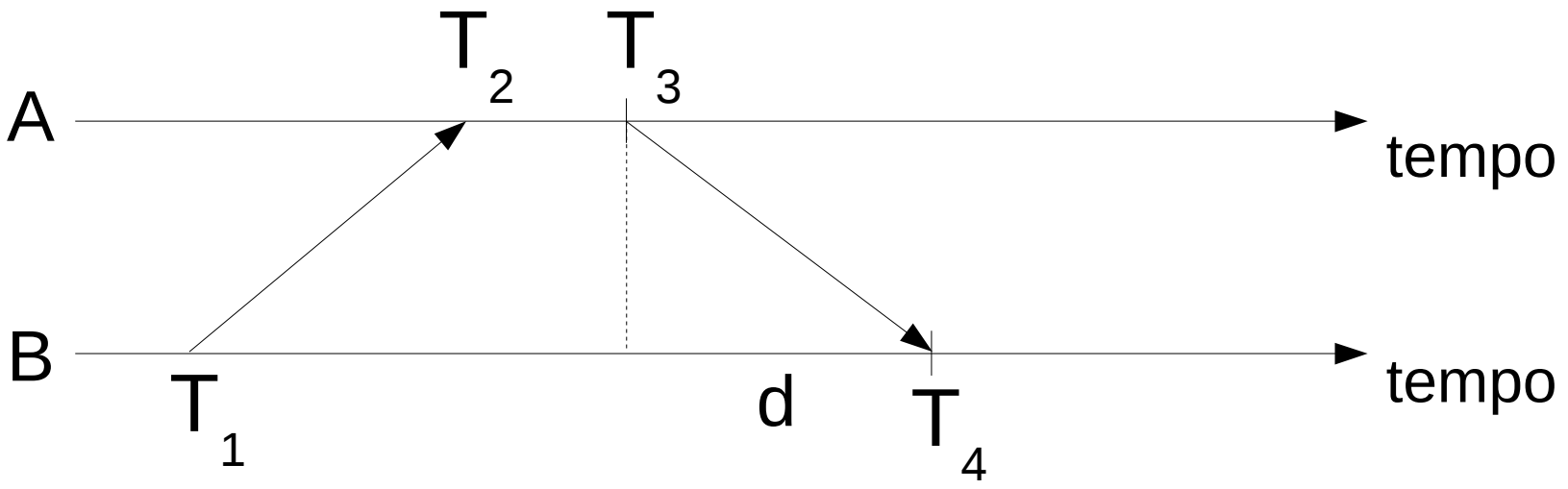
- **Ideia:** máquina B usa seu próprio relógio
  - T_p: horário que pedido é feito
  - T_r: horário que resposta chega
  - $d = (T_p - T_r) / 2$

Problemas ?



Estimando d Melhor

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

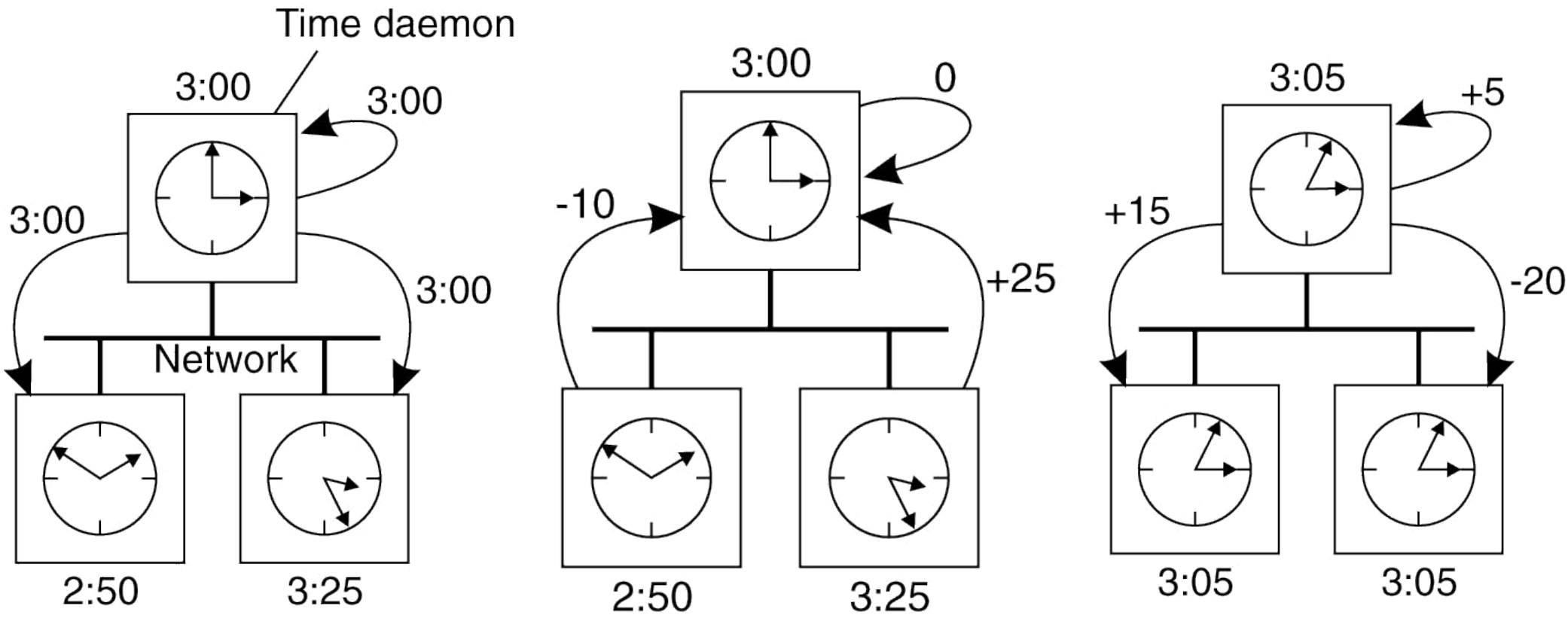


- Quanto vale d?
- $d = [(T_4 - T_1) - (T_3 - T_2)] / 2$
- B ajusta relógio para $T_3 + d$
- Ainda assume que retardo de ida igual ao de volta

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 com seus relógios
- Servidor faz a média dos valores recebidos (remove outliers)
- Servidor transmite a cada cliente ajuste necessário
- Clientes fazem ajustes em seus relógios

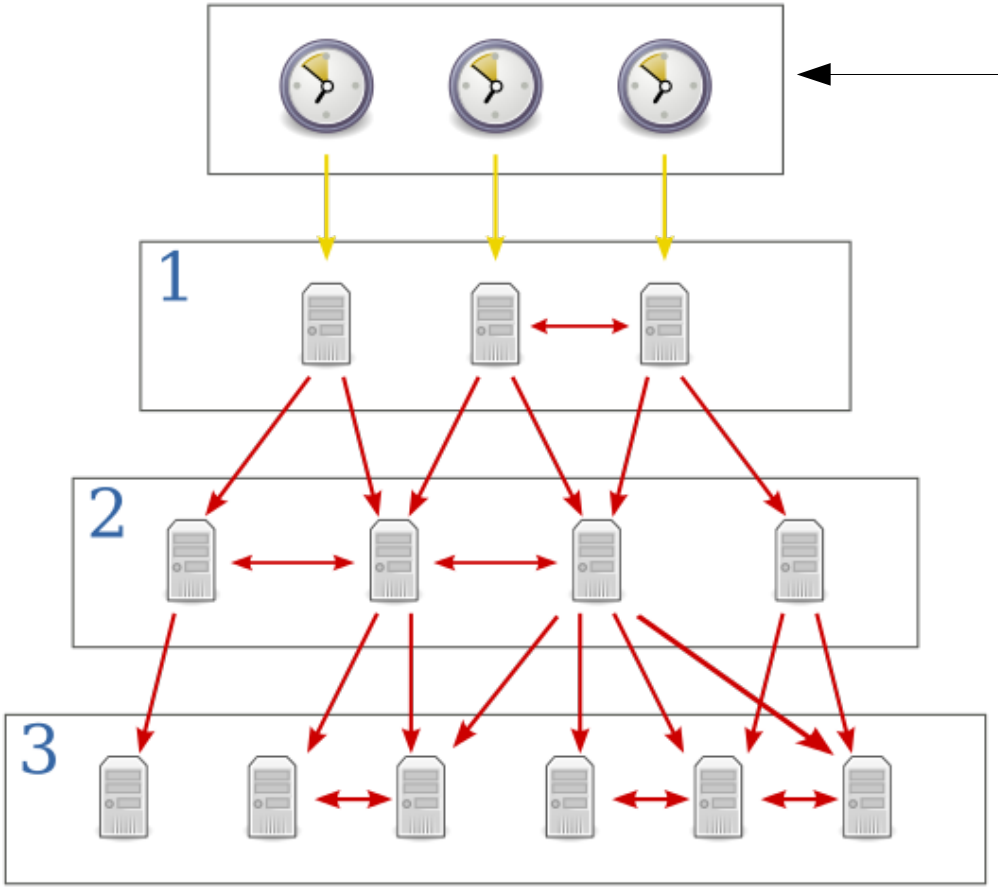
Algoritmo de Berkeley



- 1) Servidor pergunta 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, conectado a servidores 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 15)
- Determina hora certa fazendo média das horas estimadas, usando filtros (ex. elimina outliers)
- **Não seta a hora diretamente!**
- Atualiza a taxa de progressão do seu relógio
  - RTC é usado para construir relógio virtual, com taxa controlada (*x* tics virtuais por tic do RTC)
- Vantagens: hora nunca volta no tempo; mudanças de hora mais suaves

NTP em Ação

- Sincronização “constante” do relógio
- Controle fino da taxa de progressão do relógio