Sincronização

- Tempo e Relógios

- **Sincronização de Relógios**
  - **Algoritmo de Cristian**
  - **Algoritmo de Berkeley**
  - **Network Time Protocol**

Algoritmos de Sincronização

Caso mais simples:

- Sincronização interna entre dois processos num sistema distribuído síncrono.

- São conhecidos os limites máximo (Max) e mínimo (Min) para o envio de mensagens, assim como para o desvio do relógio e para o tempo de execução dos processos.

Assumindo que o processo 1 envia uma mensagem ao processo 2 com o tempo que marca o seu relógio, t

Algoritmos de Sincronização

A incerteza no envio da mensagem será u = (Max – Min)

Se o processo 2 acerta o seu relógio para t + Min, o máximo desvio (skew) será u porque a mensagem pode ter demorado Max.

Se o processo 2 acerta o seu relógio para t + Max, o máximo desvio será também u porque a mensagem pode ter demorado Min.

Mas, se o processo 2 acertar o seu relógio para, t + (Max + Min) /2

O desvio (skew) entre os dois relógios será no máximo,

(Max - Min) / 2

Algoritmos de Sincronização

- Como acertar um relógio

- obter UTC e corrigir o software do relógio

- Problemas

➔ O que acontece se um relógio está adiantado e é acertado?

- O tempo nunca anda para trás.

- O valor lido do relógio físico deverá ser escalado pelo software de forma a ir atrasando lentamente, sempre como uma função crescente.

Algoritmos de Sincronização

Sistemas Assíncronos

- Algoritmo de **Cristian (**sincronização externa)

Funcionamento: Um cliente solicita o tempo a um servidor confiável. O servidor responde com o seu tempo atual e o cliente ajusta o seu relógio com base no tempo de ida e volta (RTT) estimado.

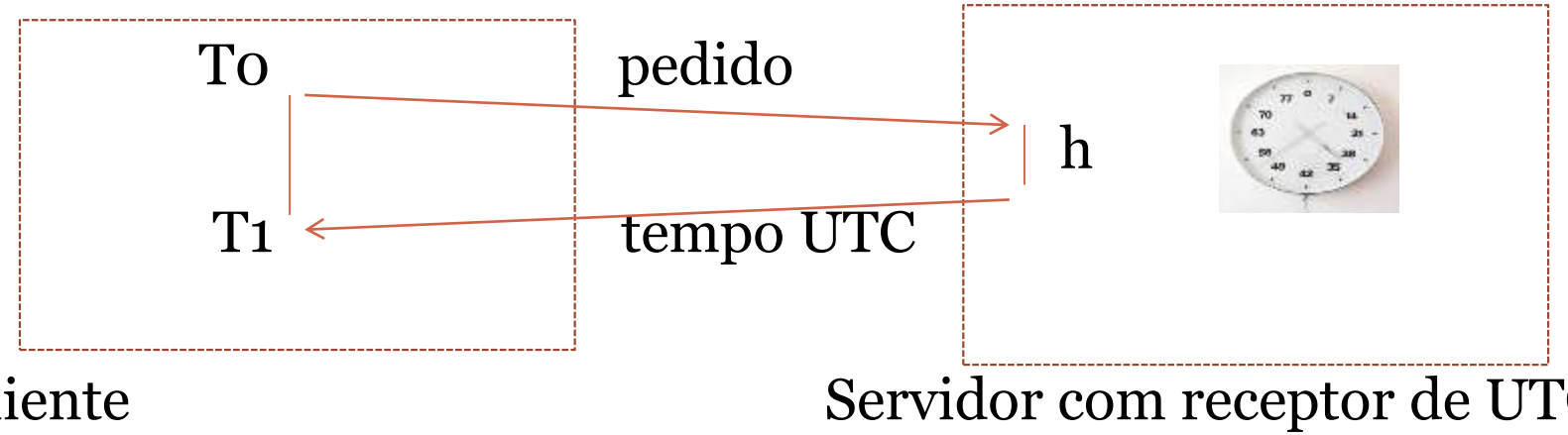
Assunção: O tempo de ida e volta da mensagem é simétrico.

Problema: Se o atraso de rede for assimétrico ou variável, a precisão cai.

Algoritmos de Sincronização

- Algoritmo de Cristian (1989)

- obter UTC e corrigir o software do relógio



Estimativa para o tempo de propagação da mensagem:

p = (T1 – To – h) /2 (= metade do "round-trip time" = 1/2 RTT)

Algoritmos de Sincronização

- Algoritmo de Cristian

- Acertar o relógio do cliente para UTC + p

- Fazer várias medições para obter o valor de T1-T0
  - Descartar valores acima de um determinado limite
  - Ou assumir os valores mínimos

Algoritmos de Sincronização

- Algoritmo de Cristian (cont)

- Algoritmo probabilístico:

- a sincronização é conseguida se o RTT é pequeno quando comparado com a exatidão desejada

- a exatidão é tanto maior quanto o tempo de transmissão está perto do mínimo

Algoritmos de Sincronização

- Algoritmo de Cristian (cont)

- Problemas
  - Ponto único de falha e congestionamento (*bottleneck*)

Possível solução: - utilizar um conjunto de servidores com receptores de UTC

- o cliente faz o pedido em *multicast* para o conjunto de servidores e usa a primeira resposta que recebe

Algoritmos de Sincronização

- Algoritmo de Cristian (cont)

- Problemas

  - Um servidor em falha ou malicioso pode provocar estragos.

  Possível solução:

  - autenticação

  - protocolo de acordo entre vários servidores que permita mascarar falhas.

Algoritmos de Sincronização

- Algoritmo de Cristian - exemplo

• Cliente regista tempo de envio: T0=10.000 ms

• Cliente envia requisição ao servidor.

• Servidor recebe mensagm e responde com o seu tempo atual: Ts=12.000 ms

• Cliente recebe a resposta e regista tempo de receção: T1′=10.060 ms

Com que valor vai atualizar o seu relógio?

Algoritmos de Sincronização

- Algoritmo de Cristian - exemplo

Tempo de ida e volta: T1′ - T0 = 10.060 – 10.000 = 60ms

Estimativa para o tempo de envio da mensagem: 60 /2 = 30 ms

Tempo estimado para o servidor no momento em que o cliente recebe a resposta: Ts = 12.000 + 30

o cliente ajusta o seu relógio para: 12.030

Algoritmos de Sincronização

- Algoritmo de Cristian

O que acontece se o servidor estiver atrasado em relação ao cliente?

- O cliente sincroniza incorretamente para um tempo no passado, o que pode causar falhas em aplicações que dependem de ordenação temporal correta (logs, eventos, transações, ...)

- Erro não corrigido pelo algoritmo de Cristian

Algoritmos de Sincronização

- Algoritmo de Cristian

Soluções possíveis:

- Ignorar sincronizações incorretas;

- Não atrasar o relógio, mas abrandar a sua velocidade, isto é ajustar a taxa de avanço do relógio até que alcance o tempo correto.

Algoritmos de Sincronização

- Algoritmo de Berkeley (sincronização interna)

Funcionamento: Um nó coordenador consulta os relógios de todos os nós participantes e calcula uma média (ignorando desvios muito grandes). Depois, envia o valor a ajustar para cada nó.

Assunção: Não há relógio de referência global; o coordenador calcula um tempo "médio".

Vantagem: Útil quando nenhum relógio externo é confiável.

Limitação: Pode ser afetado por nós com desvios extremos (outliers), embora isso possa ser mitigado.

Algoritmos de Sincronização

- Algoritmo de Berkeley (sincronização interna)

  - É escolhido um computador para ser o co-ordenador (*master)*

  - O *master* periodicamente contacta os outros computadores, (*slaves*)

  - O *master* faz uma estimativa do tempo local de cada slave, baseado no rtt.

Algoritmos de Sincronização

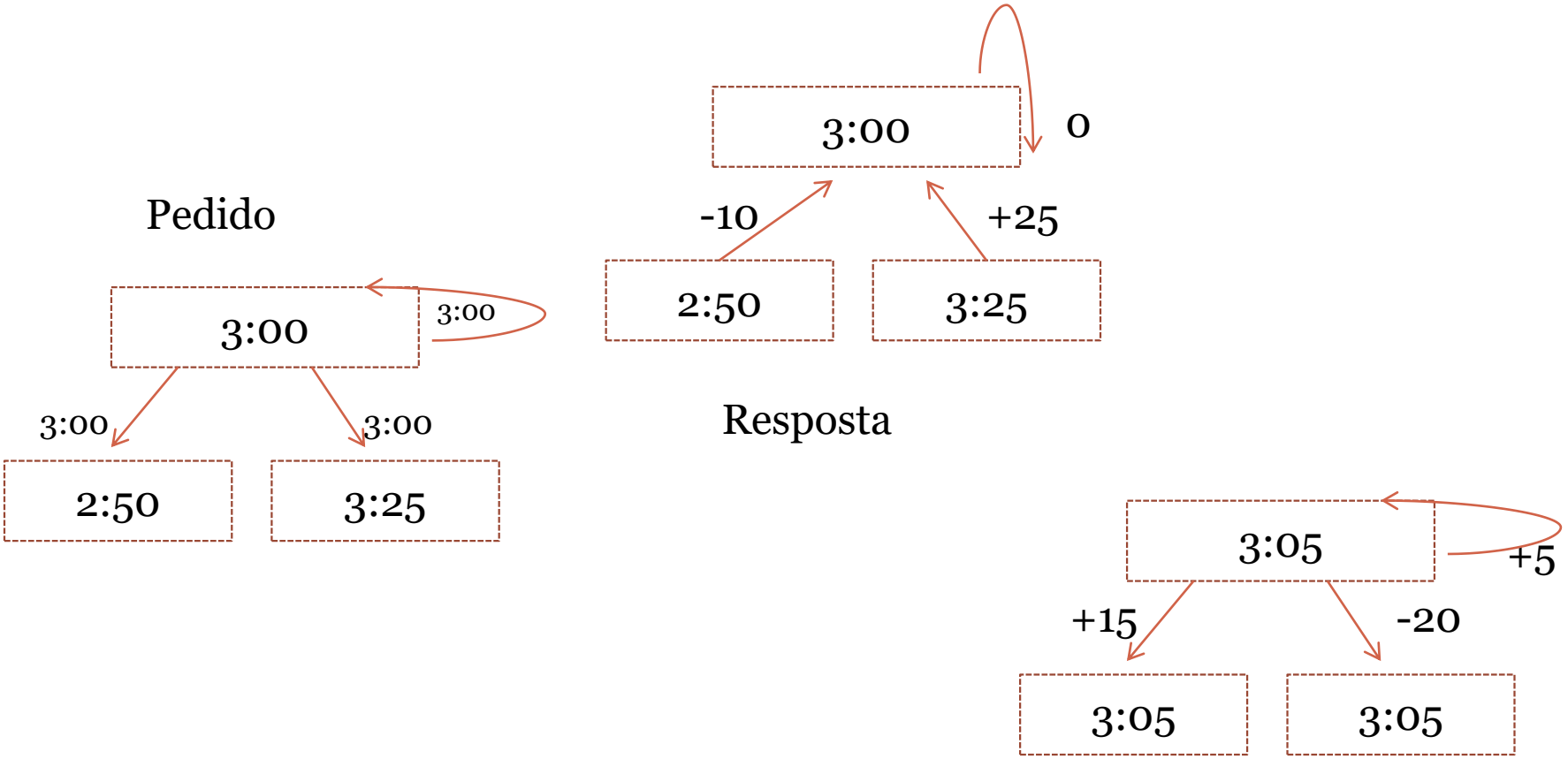
- Algoritmo de Berkeley (sincronização interna)

- O *master* calcula o tempo médio de todos os computadores, ignorando valores de transmissão demasiado elevados e máquinas com tempos muito diferentes dos outros.

- Finalmente o *master* envia a cada computador o valor de que o seu relógio deve ser ajustado (esse valor pode ser positivo ou negativo)

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo simplificado)



180 + 170 + 205 = 555 / 3 = 185

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

- Cenário: 4 máquinas

  Coordenador,

  Máquina A

  Máquina B

  Máquina C

- O coordenador envia requisições e mede o RTT (round-trip time) de cada resposta

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

- 🕒 **Tempos locais e RTTs medidos**

Máquina	Tempo local (enviado)	RTT estimado (ms)
Coordenador	10.000	--
Máquina A	10.500	40
Máquina B	9.700	20
Máquina C	10.200	60

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

- 🕒 **Tempo corrigido recebido pelo coordenador**

Máquina	Tempo local	RTT (ms)	RTT / 2	Tempo corrigido
Coordenador	10.000	--	--	10.000
Máquina A	10.500	40	20	10.480
Máquina B	9.700	20	10	10.690
Máquina C	10.200	60	30	10.170

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

- 🕒 **Cálculo do desvio em relação ao coordenador**

Máquina	Tempo corrigido	Desvio
Coordenador	10.000	--
Máquina A	10480	+480
Máquina B	9.690	-310
Máquina C	10.170	+170

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

- **Média dos desvios:**

Média = (0 + 480 – 310 + 170) / 4 = 340 /4 = 85 ms

Tempo de referência: 10.000 + 85 = **10.085**

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

🕒 **Ajustes enviados pelo coordenador**:

Máquina	Desvio	Ajuste = 85 - desvio
Coordenador	--	+85
Máquina A	+480	-395
Máquina B	-310	+395
Máquina C	+170	-85

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

Tempos após o ajuste:

Máquina	Tempo original	Ajuste	Novo tempo
Coordenador	10.000	+85	10.085
Máquina A	10.500	-395	10.105 !!
Máquina B	9.700	+395	10.095
Máquina C	10.200	-85	10.115 !!

Algoritmos de Sincronização

- Algoritmo de Berkeley (exemplo, considerando os tempos de transmissão)

- Todos os tempos estão **aproximadamente sincronizados em torno de 10.100 ms**, com pequenas variações.

- A sincronização considera **latência de rede** e não depende de um relógio de referência confiável.

- Em sistemas reais, essa abordagem pode ser refinada com múltiplas rodadas, eliminação de outliers e suavização (slewing) ao aplicar os ajustes.

Algoritmos de Sincronização

- Algoritmo de Berkeley

  - **Precisão:** depende do round trip time

  - **Tolerância a falhas:** Calcula a média dos tempos para um subconjunto de computadores que diferem até um certo valor máximo.

    Ignora mensagens cujo tempo de transmissão é demasiado elevado.

  - **Que fazer se o *master* falha?**

    Eleger um novo coordenador.

Algoritmos de Sincronização

- Network Time Protocol (NTP)

Funcionamento: Protocolo hierárquico baseado numa estrutura de níveis. Usa múltiplos servidores para estimar o tempo com algoritmos estatísticos robustos.

Precisão: Milissegundos em redes locais; dezenas de milissegundos na Internet.

- Compensação de latência de rede.
- Deteta e corrige desvios de relógios (drift).

Algoritmos de Sincronização

- Network Time Protocol (NTP)
  - Múltiplos servidores de tempo espalhados pela Internet
  - Servidores primários (ligados directamente aos receptores de UTC
  - Servidores secundários sincronizam com os primários
  - Servidores terciários sincronizam com secundários, etc
  - Permite sincronizar um elevado número de máquinas

Algoritmos de Sincronização

- Network Time Protocol (NTP)
  - Permite lidar com avarias de servidores

    Se um servidor secundário não consegue aceder a um primário, tenta aceder a outro. Existem servidores redundantes e caminhos redundantes entre servidores.

  - Usa autenticação para verificar se a informação vem de fonte fiável

Algoritmos de Sincronização

- Modos de sincronização do NTP

O Modo de sincronização determina como os dispositivos comunicam para sincronizar o tempo.

Modos principais:

- "Multicast"
- "Procedure Call"
- "Symmetric"

Algoritmos de Sincronização

- Modos de sincronização do NTP

  - **Modo "multicast"**

    - Usado em LANs de alta velocidade

    - Um ou mais servidores faz periodicamente *multicast* do seu tempo para os outros servidores.

    - Os receptores acertam os seus relógios assumindo um pequeno atraso de transmissão.

Algoritmos de Sincronização

- Modos de sincronização do NTP

  - **Modo "procedure call"**

    - Similar ao algoritmo de Cristian

    - os clientes solicitam o tempo de um ou vários servidores, e estes enviam o valor do seu relógio.

    - adequado quando o *multicast* não está disponível

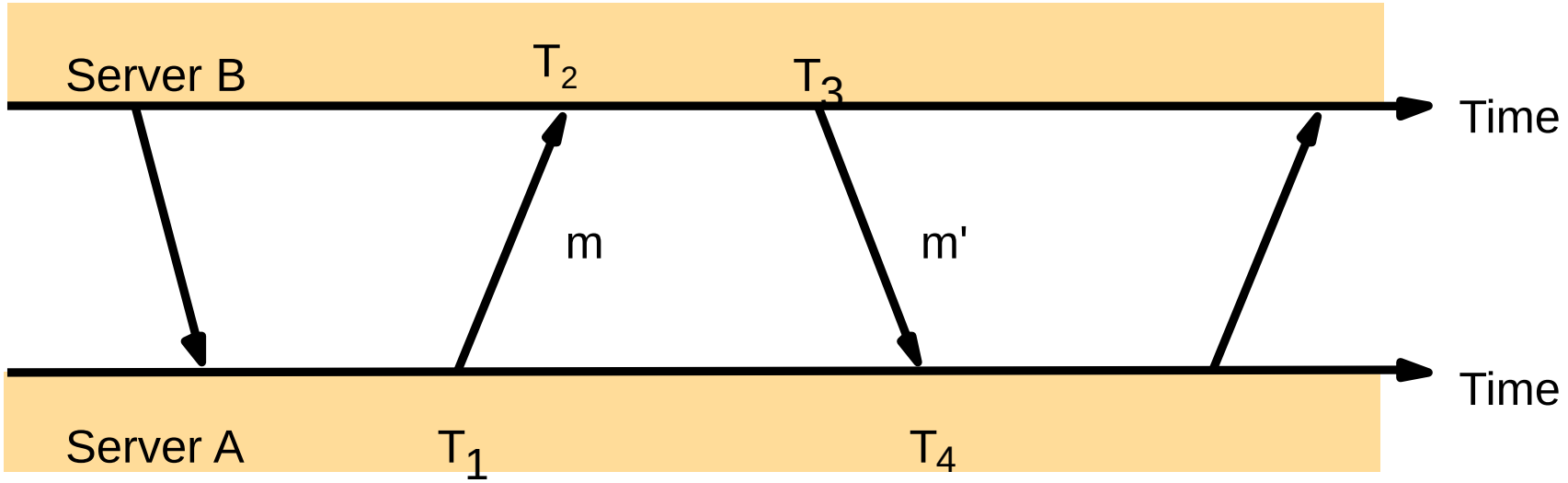
Algoritmos de Sincronização

- Modos de sincronização do NTP

  - **Modo “symmetric”**

    (maior exactidão, usado em servidores primários ou próximos)

    - Pares de processos solicitam o tempo um ao outro



Algoritmos de Sincronização

- ”NTP symmetric mode”

Para cada par de processos calcula-se um *offset*, **o**, que corresponde à diferença entre os dois relógios,

e um *delay*, **d**, que é o tempo total de transmissão das duas mensagens

Se o offset do relógio de A em relação ao de B for **o** (Tb = Ta +o) e os tempos de transmissão de mensagens de **m** e **m’** forem **t** e **t’**

Então:

$T_2 = T_1 + t + o$ e $T_4 = T_3 + t' - o$

$d_i = t + t' = T_2 - T_1 + T_4 - T_3$ **(round trip delay)**

Algoritmos de Sincronização

- ”NTP symmetric mode”

$T_2 = T_1 + t + o$ e $T_4 = T_3 + t' - o$

$d_i = t + t' = T_2 - T_1 + T_4 - T_3$ **(round trip time)**

- Supondo que T2 - T1 ≈ T4 - T3, isto é, que t ≈ t’

Podemos estimar o offset de A relativo a B como:

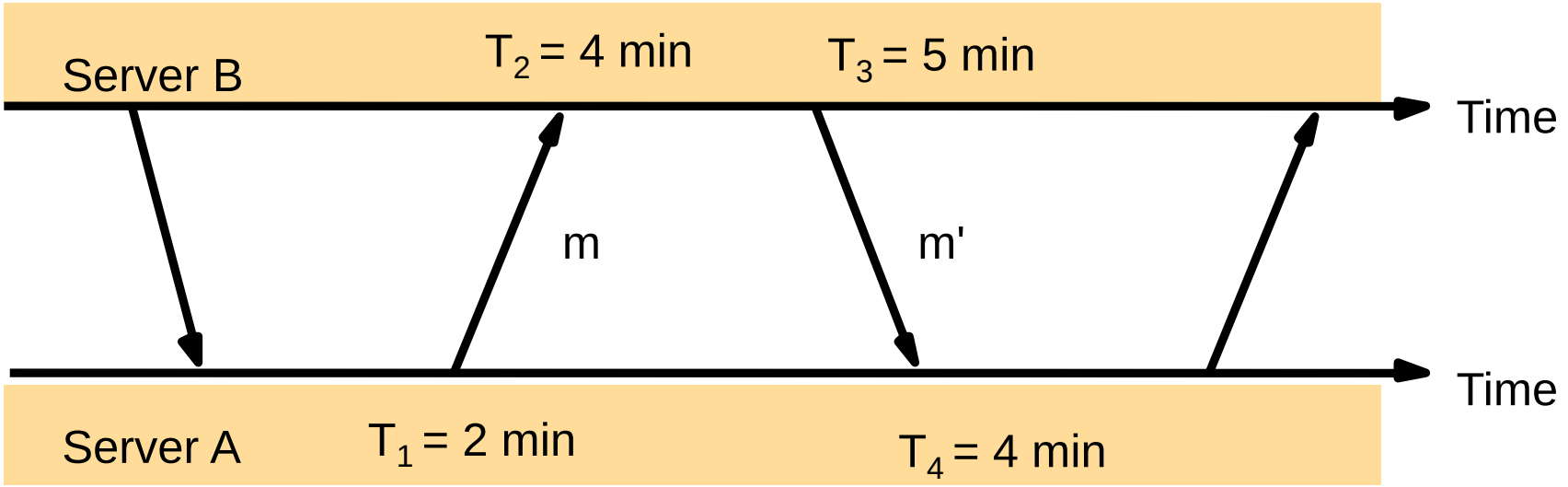
$O = T3 + ((T_2 - T_1) + (T_4 - T_3)) /2 - T4$

$= ((T_2 - T_1) + (T_3 - T_4)) /2$

Se o relógio de A é mais rápido, O < 0;

Algoritmos de Sincronização

Exemplo 1 :



O = ((T_2 - T_1) + (T_3 - T_4)) /2

O = ((4 – 2) + (5 – 4)) /2 = (2 + 1) /2 = 1.5 ➔ Tb = Ta + 1.5

RTT = T2 – T1 + T4 – T1 = 4 – 2 + 4 – 5 = 1 , T = T' = RTT /2 = 0.5

Algoritmos de Sincronização

Exemplo 2 :

O cliente NTP e o servidor trocam os timestamps durante a sincronização:

T1: tempo em que o cliente envia o pedido (marcado pelo cliente)
T2: tempo em que o servidor recebe o pedido (marcado pelo servidor)
T3: tempo em que o servidor envia a resposta (marcado pelo servidor)
T4: tempo em que o cliente recebe a resposta (marcado pelo cliente)

T1 = 100.00 T2 = 150.25
T3 = 150.30 T4 = 100.35

Qual o offset qual o RTT?

Fórmulas: **Offset:** $= ((T_2 - T_1) + (T_3 - T_4)) / 2$

RTT = T2 – T1 + T4 – T3

Algoritmos de Sincronização

- **Offset = +50.10** segundos

→ O relógio do cliente está atrasado 50.10 segundos em relação ao servidor.

- **Delay = 0.30** segundos

→ Atraso estimado da rede ida/volta.