

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 $u = (\text{Max} - \text{Min})$

Se o processo 2 ajusta o seu relógio para $t + \text{Min}$, o máximo desvio (skew) será u porque a mensagem pode ter demorado Max .

Se o processo 2 ajusta o seu relógio para $t + \text{Max}$, o máximo desvio será também u porque a mensagem pode ter demorado Min .

Mas, se o processo 2 ajustar o seu relógio para, $t + (\text{Max} + \text{Min}) / 2$

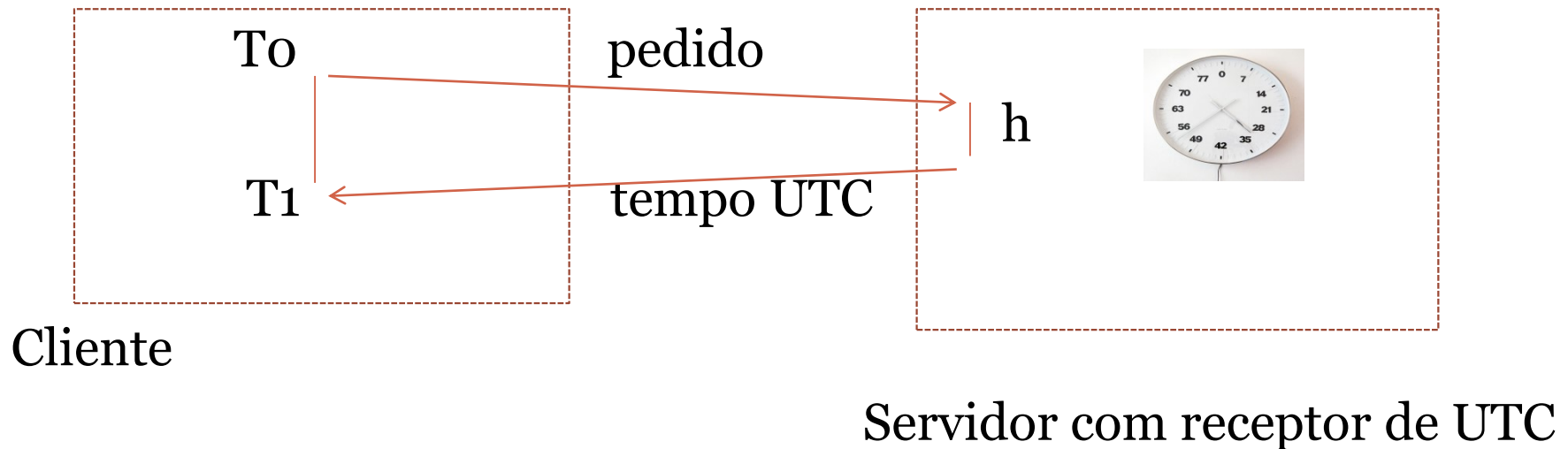
O desvio (skew) entre os dois relógios será no máximo,
 $(\text{Max} - \text{Min}) / 2$

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
 - obter UTC e corrigir o software do relógio



Estimativa para o tempo de propagação da mensagem:

$$p = (T_1 - T_0 - h) / 2 \quad (= \text{metade do "round-trip time"} = 1/2 \text{ RTT})$$

Algoritmos de Sincronização

- Algoritmo de Cristian
 - Acertar o relógio do cliente para $UTC + p$
- Fazer várias medições para obter o valor de $T_1 - T_0$
 - 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

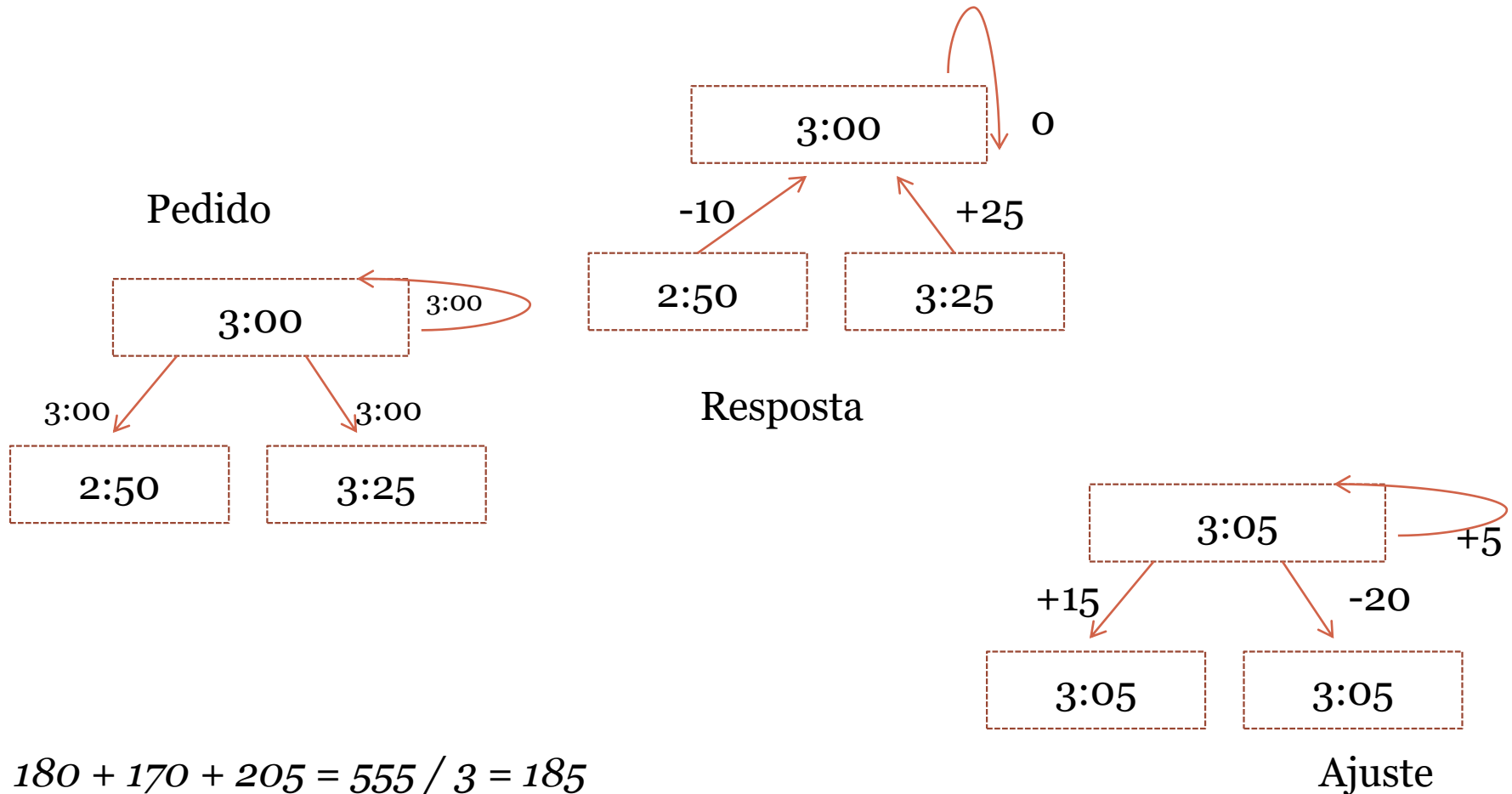
- Algoritmo de Berkeley (sincronização interna)
 - É escolhido um computador para ser o co-ordenador (*master*)
 - O *master* periodicamente contacta os outros computadores, (*slaves*)
 - O *master* faz uma estimativa do tempo local de cada slave, baseado no rtt.

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 (sincronização interna)



Algoritmos de Sincronização

- Algoritmo de Berkeley (sincronização interna)
 - **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)
 - 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

- **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

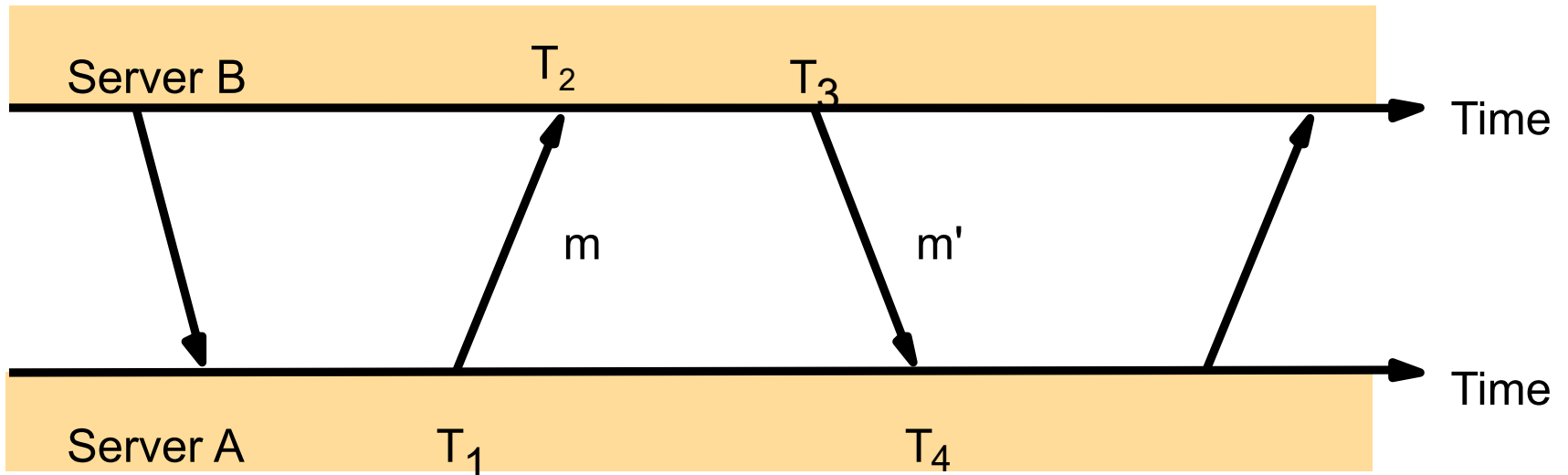
Algoritmos de Sincronização

- 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** ($T_b = T_a + o$) e os tempos de transmissão de mensagens de **m** e **m'** forem **t** e **t'**

Então:

$$T_2 = T_1 + t + o \quad \text{e} \quad T_4 = T_3 + t' - o$$

$$d_i = t + t' = T_2 - T_1 + T_4 - T_3 \quad (\text{round trip delay})$$

Algoritmos de Sincronização

- "NTP symmetric mode"

$$T_2 = T_1 + t + o \quad \text{e} \quad T_4 = T_3 + t' - o$$

$$d_i = t + t' = T_2 - T_1 + T_4 - T_3 \quad (\text{round trip delay})$$

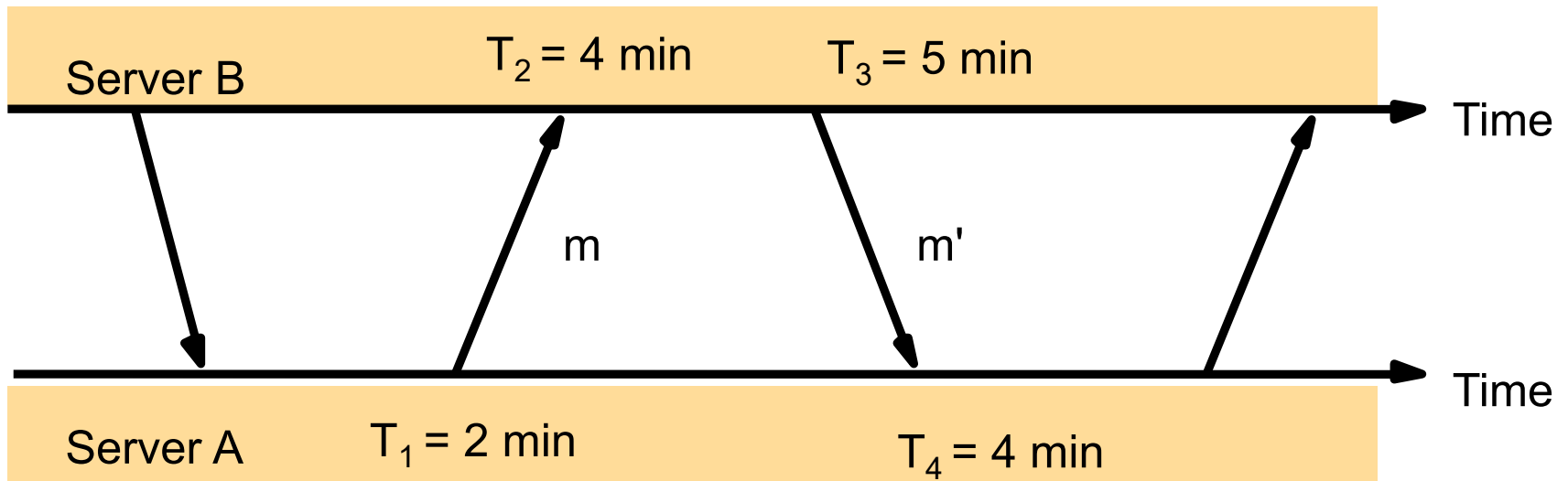
➤ Supondo que $T_2 - T_1 \approx T_4 - T_3$, isto é, que $t \approx t'$
Podemos estimar o offset de A relativo a B como:

$$\begin{aligned} O &= T_3 + ((T_2 - T_1) + (T_4 - T_3)) / 2 - T_4 \\ &= ((T_2 - T_1) + (T_3 - T_4)) / 2 \end{aligned}$$

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

Algoritmos de Sincronização

Exemplo:



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

$$O = ((4 - 2) + (5 - 4)) / 2 = (2 + 1) / 2 = 1.5 \quad \rightarrow T_b = T_a + 1.5$$