

Modelos Fundamentais de um SD

- Modelo de Falhas
- **Modelo de Interação ou Sincronismo**
- Modelo de Segurança

Modelo de Avarias:

Caracteriza o sistema em termos das falhas/avarias, i.e., dos desvios em relação ao comportamento especificado, que os seus componentes podem apresentar.

Modelo de Sincronismo:

Caracteriza o sistema em termos do comportamento temporal dos seus componentes:

- processos;
- relógios locais;
- canais de comunicação;

Modelo de Sincronismo:

- Síncrono
- Assíncrono

Um sistema diz-se **síncrono** sse:

- 1 – É conhecido um limite inferior e um limite superior para o tempo que cada processo leva a executar;
- 2 – É conhecido um limite inferior e um limite superior para o atraso na comunicação entre os seus componentes;
- 3 - Cada processo tem um relógio local e se é conhecido um limite superior para o desvio na sua taxa de incremento.

Modelo de Sincronismo:

Um sistema diz-se **assíncrono** se:

- Nada se assume sobre o comportamento temporal do sistema.

Dilema:

É relativamente fácil resolver problemas com sistemas síncronos, mas é extremamente difícil construir um sistema síncrono.

Sincronização

- Tempo e Relógios
- Sincronização de Relógios
 - Algoritmo de Cristian
 - Algoritmo de Berkeley
 - Network Time Protocol

O papel do tempo

- Precisa de ser medido com elevada precisão
- Precisa de ser medido de forma consistente pelas diversas componentes de um sistema
- Crucial na ordenação de eventos
observadores distintos podem testemunhar eventos por ordens diferentes

O papel do tempo

- Tempo Real?

- Função monótona contínua e crescente
- Unidade: segundo

“Actualmente um segundo é a duração de 9.192.631.770 períodos da radiação correspondente à transição entre os dois níveis hiperfinos do estado fundamental do átomo de césio 133”

Graficamente pode ser representado por uma sequência de pontos sobre uma linha recta → ***Timeline***

O papel do tempo

O uso do tempo em sistemas distribuídos é feito em dois aspectos:

- Registrar e observar a localização de eventos na timeline

Queremos saber qual a sequência em que ocorreu um conjunto de eventos (possivelmente distribuídos por várias máquinas)

- Forçar o futuro posicionamento de eventos na timeline

Sincronização do progresso concorrente do sistema

O papel do tempo

Para conhecermos qual a sequência de um conjunto de acontecimentos podemos marcar o instante de ocorrência atribuindo um,

- ***Timestamp***: sequência de caracteres que marcam a data e/ou tempo no qual um certo evento ocorreu.
(ex. data de criação/alteração de um ficheiro)
- um timestamp está associado a um ponto na timeline.

Se queremos comparar a duração de vários acontecimentos podemos usar,

- **Intervalos de tempo**: cadeia de tempo composta por vários intervalos adicionados

Durações Distribuídas

- **Timers / relógios locais:** implementam a abstracção da *timeline*.

Num sistema distribuído cada evento pode ocorrer em diferentes locais cada um com a sua *timeline*.

- Como conciliar diferentes *timelines*?
- Como medir durações distribuídas?

Tempo Global vs Tempo Absoluto

- **Tempo Global (*global time*):** implementa a abstracção de um tempo universal, através de um relógio que fornece o mesmo tempo a todos os participantes no sistema.
- **Tempo Absoluto (*absolute time*):** padrões universalmente ajustados, disponíveis como fontes de tempo externo para o qual qualquer relógio interno se pode sincronizar.

Relógios locais

- **Relógio físico local (physical clock - pc):** o modo mais comum para fornecer uma fonte de tempo num processo.

- Equipamento físico, que conta as oscilações que ocorrem num cristal de quartzo a uma dada frequência.
- Cada oscilação do cristal decrementa o contador de uma unidade.
- Quando o contador chega a zero, é gerado um *interrupt* e o contador é recarregado com o valor inicial.
- Cada *interrupt* é designado como um “clock tick”

Relógios locais

- Um relógio num processo correcto, k , implementa uma função discreta, monótona crescente, pc_k , que mapeia o tempo real t em tempo de relógio $pc_k(t)$.

- Problemas dos relógios físicos:

Granularidade: relógios físicos são granulares, isto é, avançam uma unidade em cada *tick* t_{tk} .

$$pc_k^{tk+1} - pc_k^{tk} = g$$

Problemas dos relógios físicos

- A frequência das oscilações varia com a temperatura
- Diferentes taxas de desvio em diferentes computadores
- **Taxa de desvio do relógio físico:** existe uma constante positiva r_p , a *taxa de desvio (rate of drift)*, que depende não só da qualidade do relógio mas também das condições ambientais.

$$0 \leq 1 - r_p \leq (pc_k(t_{tk+1}) - pc_k(t_{tk}))/g \leq 1 + r_p$$

para $0 \leq t_{tk} \leq t_{tk+1}$

(Para uma taxa de desvio de 10^{-5} , após 60 minutos o erro acumulado pode ser superior a 30 milissegundos)

Clock skew / Clock drift

Skew – *é a diferença do valor do tempo lido de dois relógios diferentes.*

Drift – *é a diferença no valor lido de um relógio e o valor do tempo fornecido por um relógio de referência perfeito por unidade de tempo do relógio de referência.*

Ex. *drift de 10^{-5} segundos / segundo, significa que em cada segundo o relógio tem um desvio de 0,00001 segundos.*

Para que serve um relógio local?

- Fornecer *timestamps* para eventos locais.
- Medir durações locais
O erro causado pelo desvio é normalmente insignificante para pequenas durações.
- Pode ser usado como um “timer” para estabelecer *timeouts*
- Medir durações distribuídas *round-trip*

Relógios Globais - Características

- Fornecer o mesmo tempo para todos os intervenientes do sistema
- *Timestamping* de eventos distribuídos
- Medição de durações distribuídas

Relógios Globais - funcionamento

- É criado um relógio virtual vc_p para cada processo p a partir do relógio físico
- É feita a sincronização de todos os relógios locais com o mesmo valor inicial $vc_p(t_{init})$
- Periodicamente os relógios virtuais são re-sincronizados

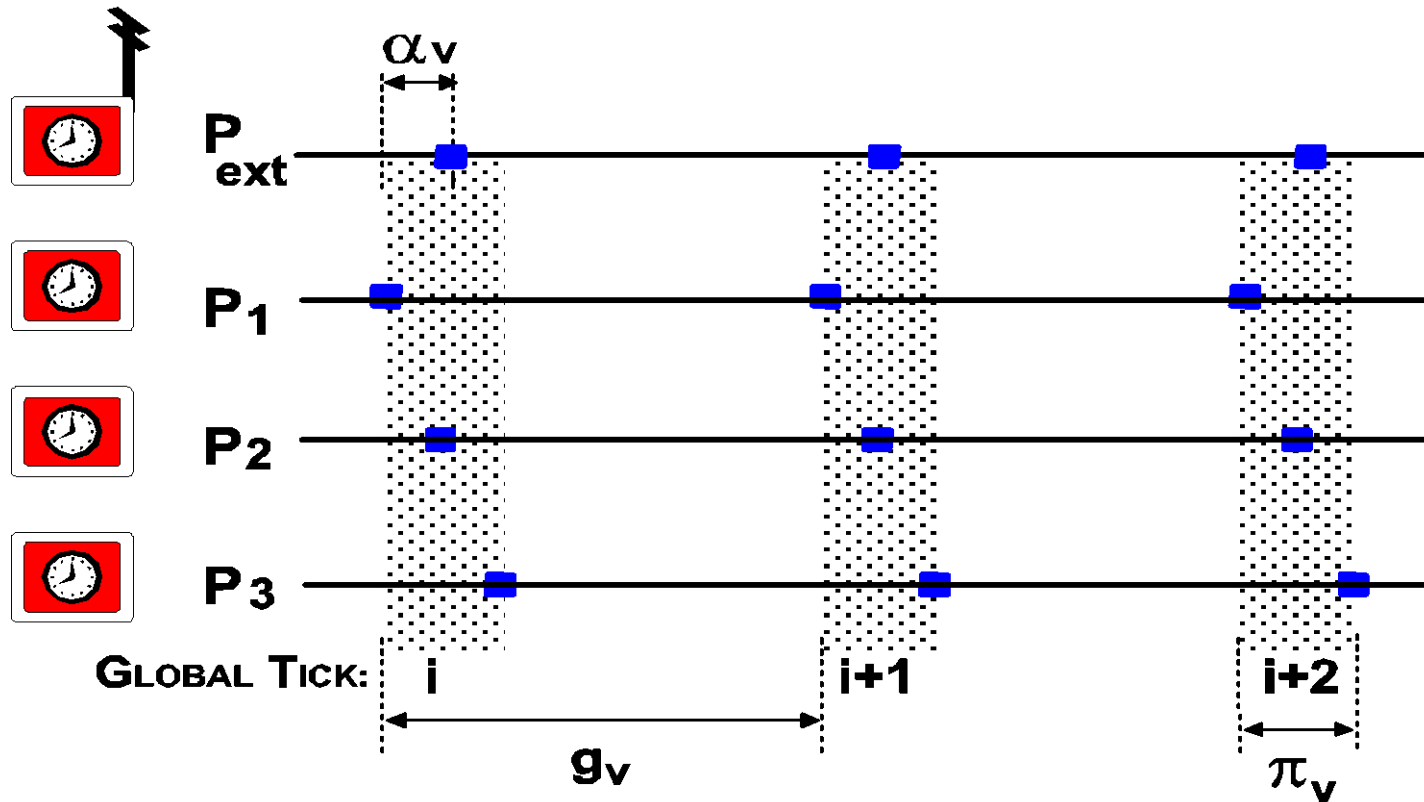


Algoritmos de Sincronização

Propriedades dos Relógios Globais

- Granularidade $g_v = vc_p(t_{k+1}) - vc_p(t_k)$
- Precisão (π_v): quão próximos os relógios se mantêm sincronizados entre si em qualquer instante do tempo.
- Exactidão (accuracy - α_v): quão próximos os relógios estão sincronizados em relação a uma referência de tempo real absoluto (sincronização externa)

Propriedades dos Relógios Globais (cont.)



Sincronização interna vs Sincronização externa

- Sincronização interna:

- relógios têm que obter precisão relativamente a um tempo interno ao sistema

- Sincronização externa:

- relógios tem que estar sincronizados com uma fonte externa de tempo universal

Referências de Tempo universal - Normas

- **Tempo Atómico Internacional (TAI - Temps Atomique International)**

função contínua monótona crescente a uma taxa constante
(tempo médio dos relógios atómicos de césio existentes)

- **Universal Time, Coordinated (UTC)**

referência de tempo política (correção do TAI de forma a
ajustar o tempo com o dia solar)

- Forma mais simples de obter o UTC: por GPS (Global Position System) – é assegurada uma exactidão, em terra, $\leq 100\text{ns}$ para os relógios dos receptores de GPS

Medição de durações *round-trip*

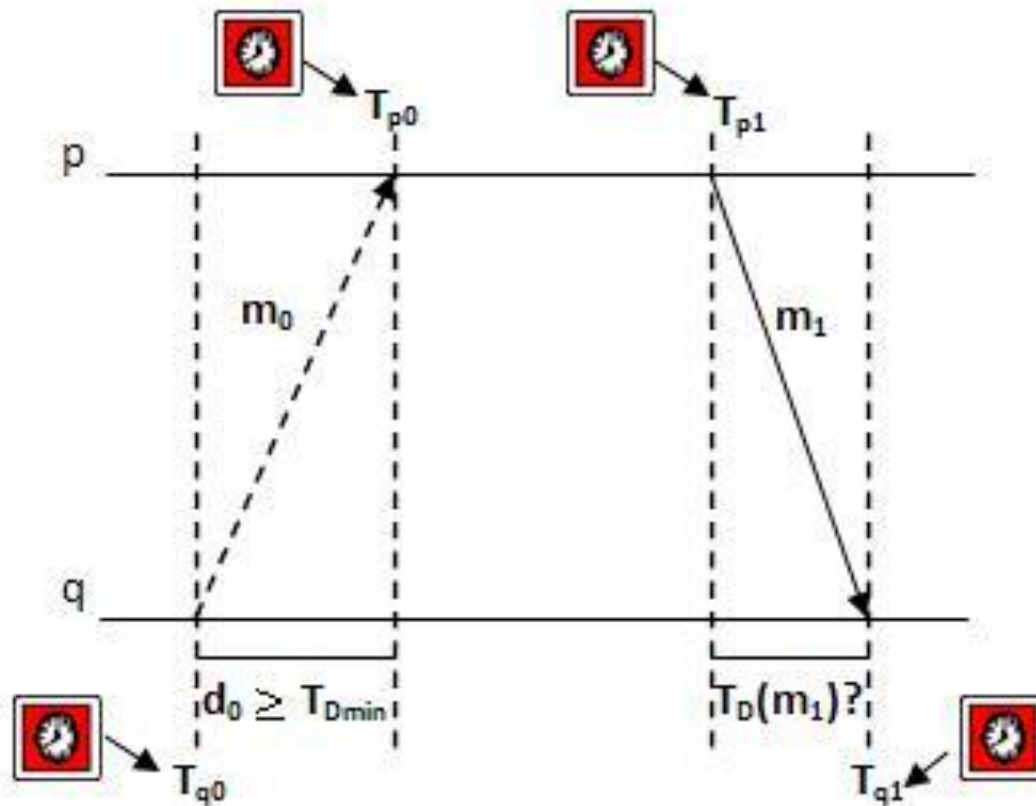
- Certas durações distribuídas podem ser medidas sem a existência explícita de relógios globais
- O atraso de entrega de uma mensagem pode ser calculado com um erro conhecido e limitado, se existir uma mensagem prévia recente no sentido inverso

Medição de durações *round-trip* (cont.)

- Pré-requisitos para o uso deste método:

- Assegurar troca de mensagens frequente entre os *sites* relevantes
- Assegurar que o *timestamping* das transmissões de mensagens e entregas, também sejam trocados entre os *sites* relevantes

Medição do atraso de entrega de mensagens



$$t_D(m_1) \leq (T_{q1} - T_{q0}) - (T_{p1} - T_{p0}) - T_{Dmin}$$

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

Se o processo 2 acerta 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 acertar o seu relógio para, $t + (\text{Max} + \text{Min}) / 2$

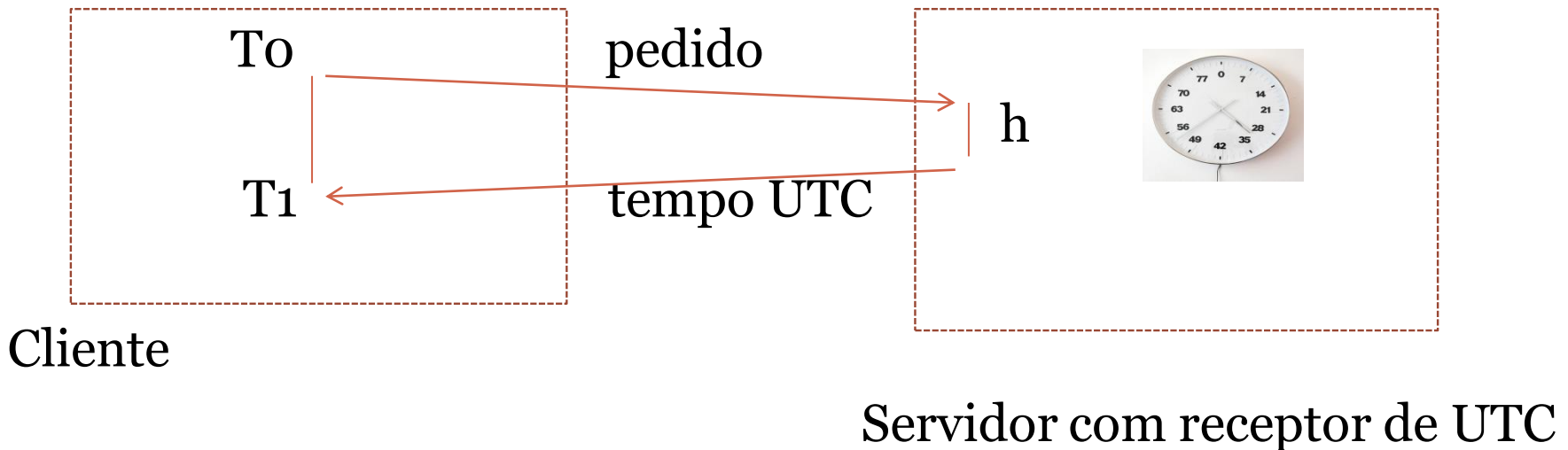
O desvio (skew) entre os dois relógios será no máximo,
 $(\text{Max} - \text{Min}) / 2$

Algoritmos de Sincronização

- Como acertar um relógios
 - 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 T1-To

- Descartar valores acima de um determinado limite
- Ou assumir os valores mínimos

Algoritmo probabilístico:

- a sincronização é conseguida se o RTT é pequeno quando comparado com a exactidão desejada
- a exactidão é tanto maior quanto o tempo de transmissão está perto do mínimo

Algoritmos de Sincronização

- Algoritmo de Cristian

- 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

- Um servidor em falha ou malicioso pode provocar estragos.

Possível solução: - autenticação

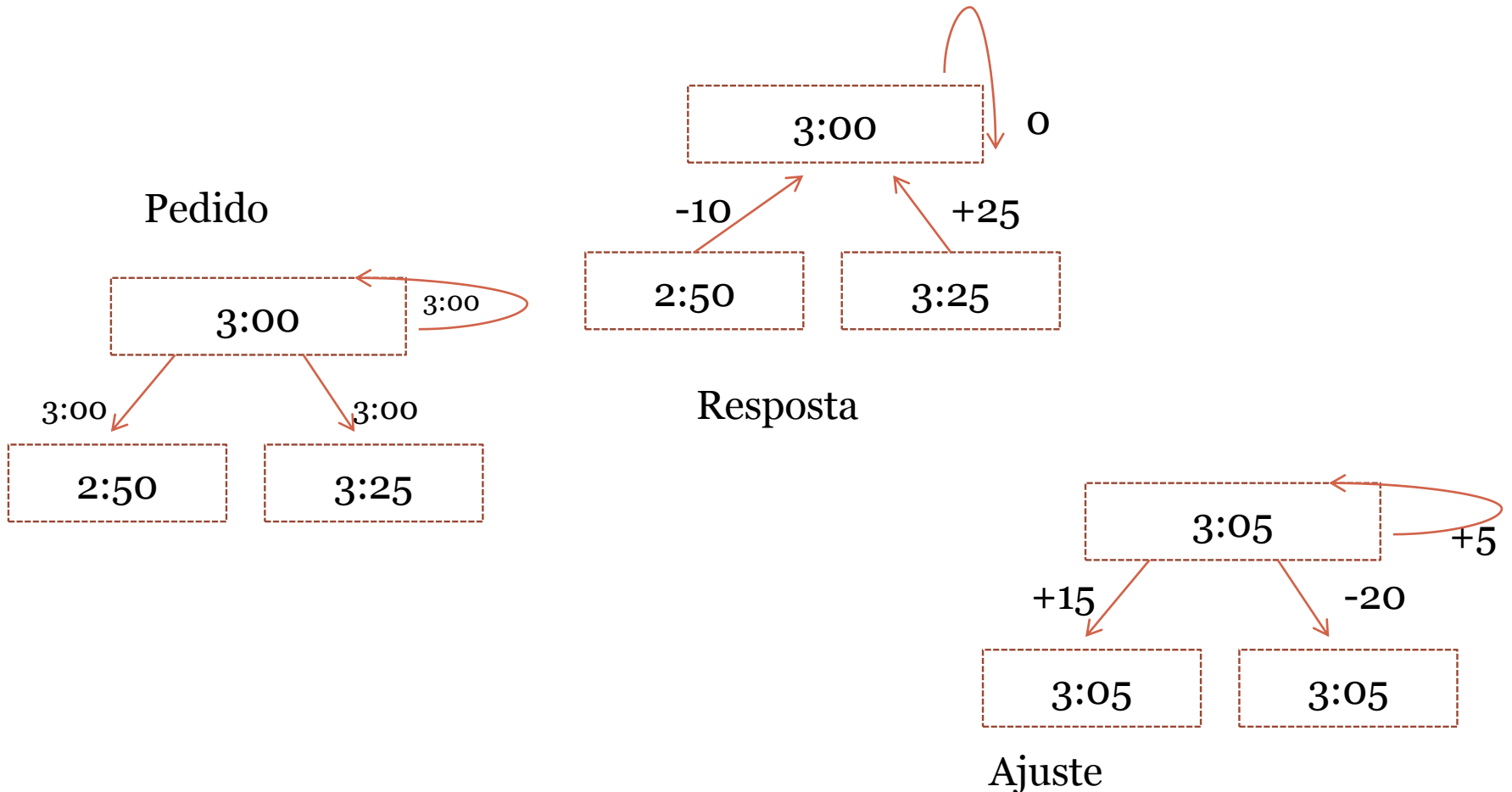
- protocolo de acordo entre vários servidores que permita mascarar falhas.

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.
 - 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.

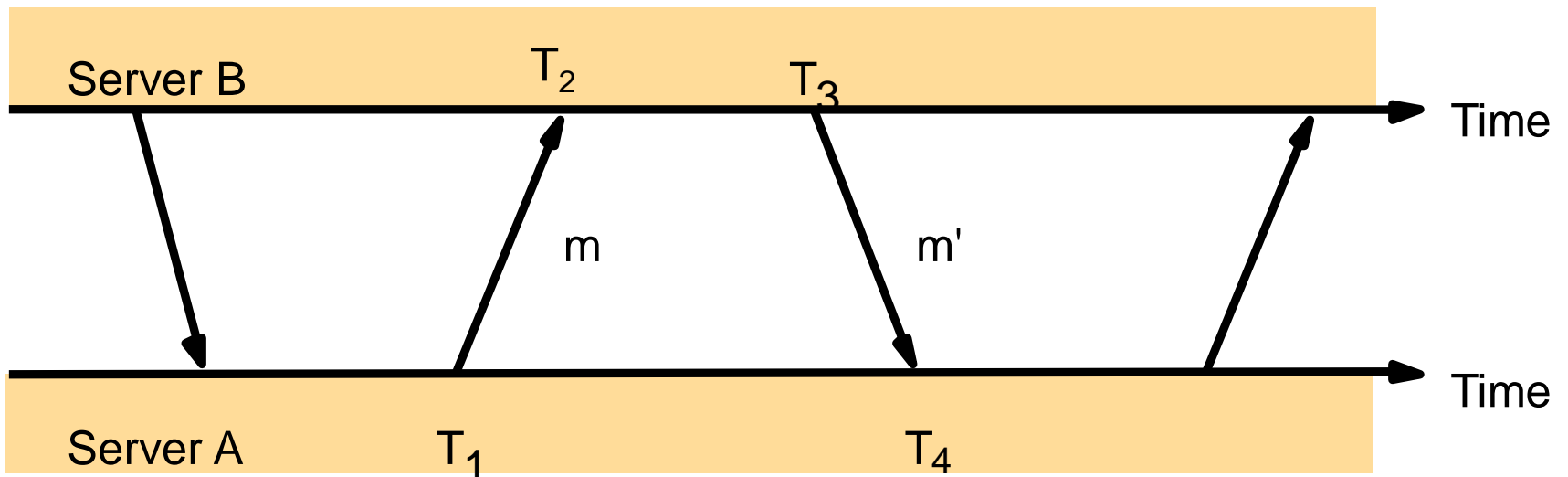
Algoritmos de Sincronização

- 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 se pretende maior nível de exactidão
ou
 - 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*, \mathbf{o} , que corresponde à diferença entre os dois relógios, e um *delay*, \mathbf{d} , que é o tempo total de transmissão das duas mensagens

Se o offset do relógio de A em relação ao de B for \mathbf{o} os tempos de transmissão de mensagens de \mathbf{m} e \mathbf{m}' forem \mathbf{t} e \mathbf{t}'

Então:

$$\mathbf{T}_2 = \mathbf{T}_1 + \mathbf{t} + \mathbf{o} \quad \text{e} \quad \mathbf{T}_4 = \mathbf{T}_3 + \mathbf{t}' - \mathbf{o}$$

$$\mathbf{d}_i = \mathbf{t} + \mathbf{t}' = \mathbf{T}_2 - \mathbf{T}_1 + \mathbf{T}_4 - \mathbf{T}_3 \quad (\text{round trip delay})$$

Algoritmos de Sincronização

- "NTP symmetric mode"

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

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

➤ Supondo que $T_2 - T_1 \approx T_4 - T_3$

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, $O < 0$;