ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 12: Coordenação (parte 1)

Prof. Renan Alves

Escola de Artes, Ciências e Humanidades — EACH — USP

12/04/2024

Recapitulando...

Vimos:

- Processos
- Comunicação

Como fazer para que os processos:

- Sejam sincronizados (i.e. consigam estabelecer ordem dos eventos)
- Esperem algum evento
- Interajam para acessar recursos compartilhados

Sincronização de Relógios

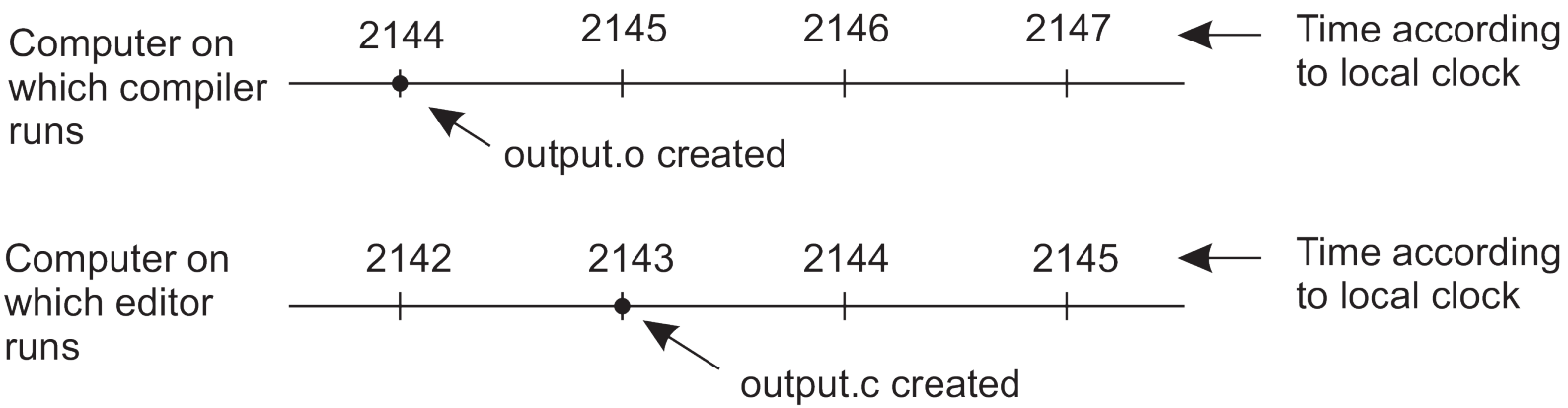
Em um sistema centralizado:

- Há uma única referência de tempo, monotonicamente crescente
- Qualquer processo pode fazer uma syscall para consultar o horário local

Em um sistema distribuído:

- Não há referência única de tempo
- Aplicações podem ser diretamente impactadas por desacordos temporais
- Como sincronizar?

Exemplo


Computer on which compiler runs
2144
2145
2146
2147
Time according to local clock
output.o created
Computer on which editor runs
2142
2143
2144
2145
Time according to local clock
output.c created

Relógios físicos

Problema

Às vezes, precisamos apenas do tempo exato, não apenas de uma ordenação.

Solução: Universal Coordinated Time (UTC)

- Baseado no número de transições por segundo do átomo de césio 133 (bastante preciso).
- Atualmente, o tempo real é considerado como a média de cerca de 50 relógios de césio ao redor do mundo.
- Introduz um segundo extra de tempos em tempos para compensar que os dias estão ficando mais longos.

Nota

- O UTC é transmitido em massa (broadcast) por meio de rádio de ondas curtas e satélite.
- Os satélites podem fornecer uma acurácia de cerca de ± 0.5 ms.
- Combinando recepções de vários satélites, podem ser construídos servidores de tempo terrestres, com uma precisão de 50 nanossegundos.

Sincronização de Relógios

Relógios nos sistemas computacionais

- Na prática não é viável todos com computadores possuírem um receptor UTC
- Sistemas computacionais em geral possuem um relógio baseado em oscilações de cristal (relógio de hardware)
  - Pode haver variações entre relógios
- Relógio de hardware é utilizado para gerar um interrupção que incrementar o valor de um contador (relógio de software)

Sincronização de Relógios

Precisão

O objetivo é manter a divergência entre dois relógios em duas máquinas quaisquer dentro de um limite especificado, conhecido como precisão π:

$$\forall t, \forall p, q : |C_p(t) - C_q(t)| \leq \pi$$

onde $C_p(t)$ é o valor de relógio calculado na máquina p no tempo UTC t.

Acurácia

No caso da acurácia, visamos manter o relógio dentro de uma faixa de valores limitado por α:

$$\forall t, \forall p : |C_p(t) - t| \leq \alpha$$

Sincronização

- Sincronização interna: mantém a precisão dos relógios
- Sincronização externa: mantém a acurácia dos relógios

Variação de relógio (drift)

Especificações de relógio

- Taxa de variação de relógio máxima ρ.
- $F(t)$ denota a frequência do oscilador do relógio de hardware no tempo t.
- F é a frequência ideal (constante) do relógio
- Para cumprir as especificações:

$$\forall t : (1-\rho) \leq \frac{F(t)}{F} \leq (1+\rho)$$

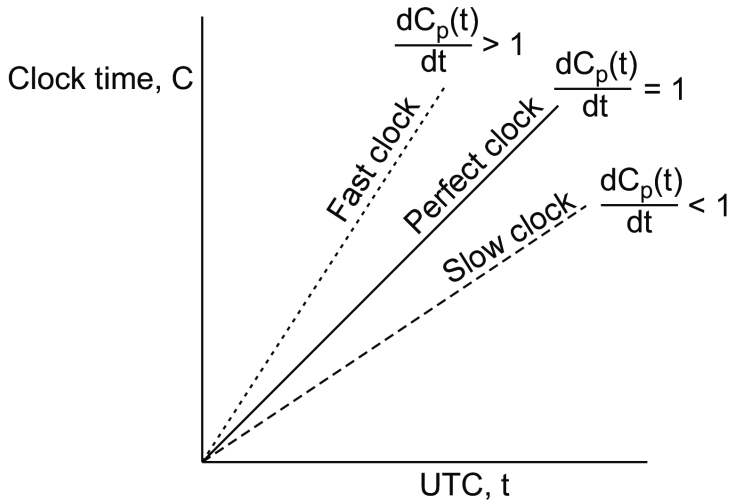
Observação

Ao usar interrupções de hardware, acoplamos um relógio de software ao relógio de hardware, e, assim, também sua taxa de variação:

$$C_p(t) = \frac{1}{F}\int_0^t F(t)dt \Rightarrow \frac{dC_p(t)}{dt} = \frac{F(t)}{F}$$

$$\Rightarrow \forall t : 1-\rho \leq \frac{dC_p(t)}{dt} \leq 1+\rho$$

Relógios rápidos, perfeitos, lentos

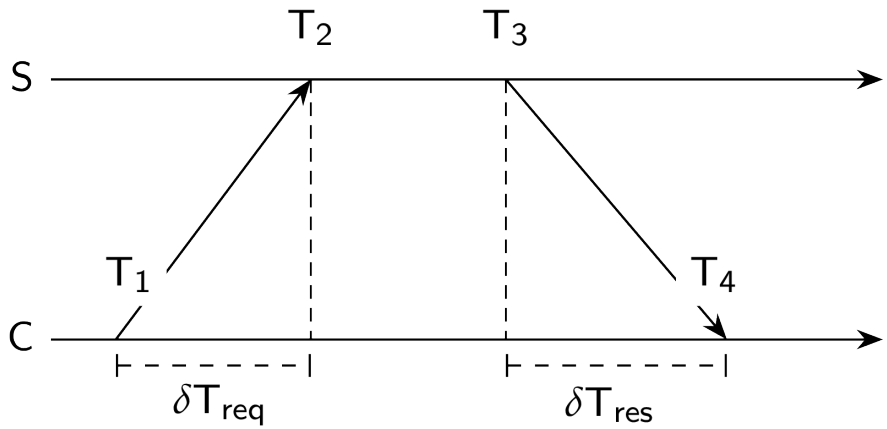


Quando sincronizar

- Considere dois relógios se desviando do UTC na direção oposta com drift ρ
- Após um tempo Δt depois de terem sido sincronizados: separados por até $2\rho * \Delta t$
- Se desejar precisão π: os relógios devem ser ressincronizados pelo menos a cada $\pi/(2\rho)$ segundos
- Os vários algoritmos diferem precisamente na forma como essa ressincronização é feita.

Detecção e ajuste de tempos incorretos

Obtendo o tempo atual de um servidor de tempo



Computando a diferença relativa θ e o atraso δ

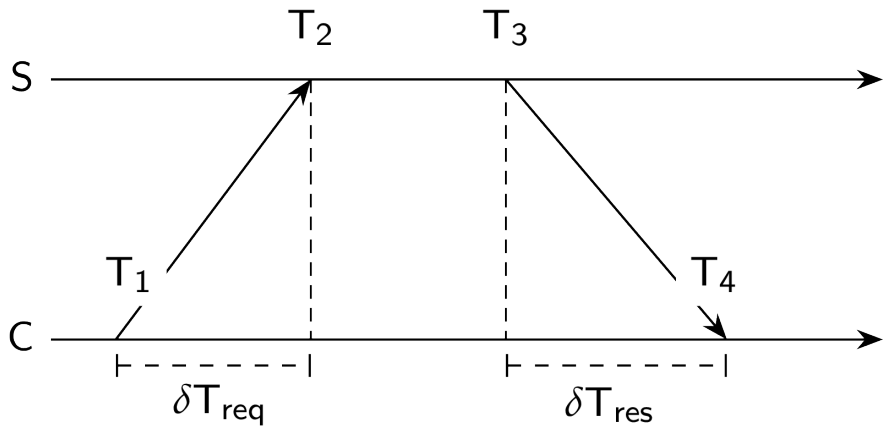
Suposição: $\delta T_{req} = T_2 - T_1 \approx T_4 - T_3 = \delta T_{res}$

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

$$\delta = \big((T_4 - T_1) - (T_3 - T_2)\big)/2$$

Detecção e ajuste de tempos incorretos

Obtendo o tempo atual de um servidor de tempo



Computando a diferença relativa θ e o atraso δ

Suposição: $\delta T_{req} = T_2 - T_1 \approx T_4 - T_3 = \delta T_{res}$

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

$$\delta = \big((T_4 - T_1) - (T_3 - T_2)\big)/2$$

Network Time Protocol (precisão: dezenas de ms)

Coletar pares (θ, δ). Escolher θ para o qual o atraso associado δ for mínimo.

Sincronização em redes sem fio

Nas redes cabeadas

- Há disponibilidade de servidores de tempo
- É mais fácil formar pares arbitrários para sincronização, em geral

Nas redes sem fio (especialmente ad hoc)

- Hipóteses anteriores podem ser falsas
- Roteamento através de múltiplos saltos
- Necessidade de economizar energia

Reference broadcast synchronization (RBS)

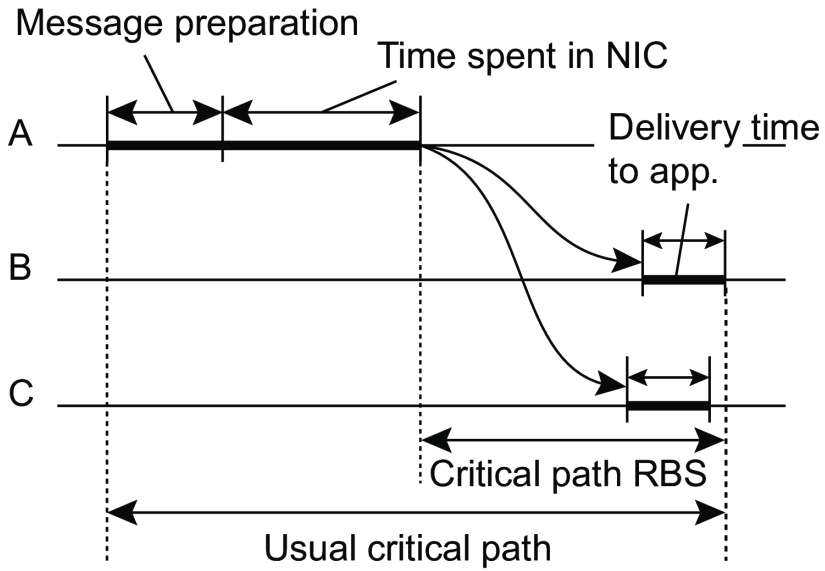
- Em vez de sincronizar com UTC, visa apenas sincronizar os relógios internamente
- Apenas os receptores sincronizam

Reference broadcast synchronization (RBS)

Essência

- Um nó transmite uma mensagem de referência $m \Rightarrow$ cada nó receptor p registra o tempo $T_{p,m}$ que recebeu m.
- Nota: $T_{p,m}$ é lido a partir do relógio local de p.

RBS minimiza o caminho crítico



Reference broadcast synchronization (RBS)

Sincronizando dois nós P e Q

- Cada um deles recebeu k mensagens RBS

Primeira ideia: calcular média das diferenças

- $Offset[p,q](t) = \frac{\sum_{k=1}^{M}(T_{p,k} - T_{q,k})}{M}$
- Problema: a média não capturará o drift

Solução: usar regressão linear

- $Offset[p,q](t) = \alpha t + \beta$

A relação do tipo ”ocorreu-antes” (happened-before)

Questão

O que geralmente importa não é que todos os processos concordem exatamente com que hora é, mas que eles concordem com a ordem em que os eventos ocorrem. Requer uma noção de ordenação.

A relação do tipo ”ocorreu-antes” (happened-before)

Questão

O que geralmente importa não é que todos os processos concordem exatamente com que hora é, mas que eles concordem com a ordem em que os eventos ocorrem. Requer uma noção de ordenação.

A relação do tipo ocorreu-antes

- Se a e b são dois eventos de um mesmo processo, e a vem antes de b, então $a \rightarrow b$.
- Se a é o envio de uma mensagem e b é o recebimento dessa mensagem, então $a \rightarrow b$.
- Se $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$.

Nota

Isso introduz uma ordem parcial de eventos em um sistema com processos operando concorrentemente.

Relógios lógicos

Problema

Como podemos manter uma visão global do comportamento do sistema que seja consistente com a relação de ocorreu-antes?

Relógios lógicos

Problema

Como podemos manter uma visão global do comportamento do sistema que seja consistente com a relação de ocorreu-antes?

Associar um timestamp $C(e)$ a cada evento e, satisfazendo as seguintes propriedades:

- P1 Se a e b são dois eventos no mesmo processo, e $a \rightarrow b$, então exigimos que $C(a) < C(b)$.
- P2 Se a corresponde ao envio de uma mensagem m, e b ao recebimento dessa mensagem, então também $C(a) < C(b)$.

Relógios lógicos

Problema

Como podemos manter uma visão global do comportamento do sistema que seja consistente com a relação de ocorreu-antes?

Associar um timestamp $C(e)$ a cada evento e, satisfazendo as seguintes propriedades:

- P1 Se a e b são dois eventos no mesmo processo, e $a \rightarrow b$, então exigimos que $C(a) < C(b)$.
- P2 Se a corresponde ao envio de uma mensagem m, e b ao recebimento dessa mensagem, então também $C(a) < C(b)$.

Problema

Como associar um timestamp a um evento quando não há relógio global $\Rightarrow$ manter um conjunto consistente de relógios lógicos, um por processo.