Sincronização em Sistemas Distribuídos

Prof. Raul Ceretta Nunes

Curso de Ciência da Computação

ELC1018 - Sistemas Distribuídos

informática
Universidade Federal de Santa Maria

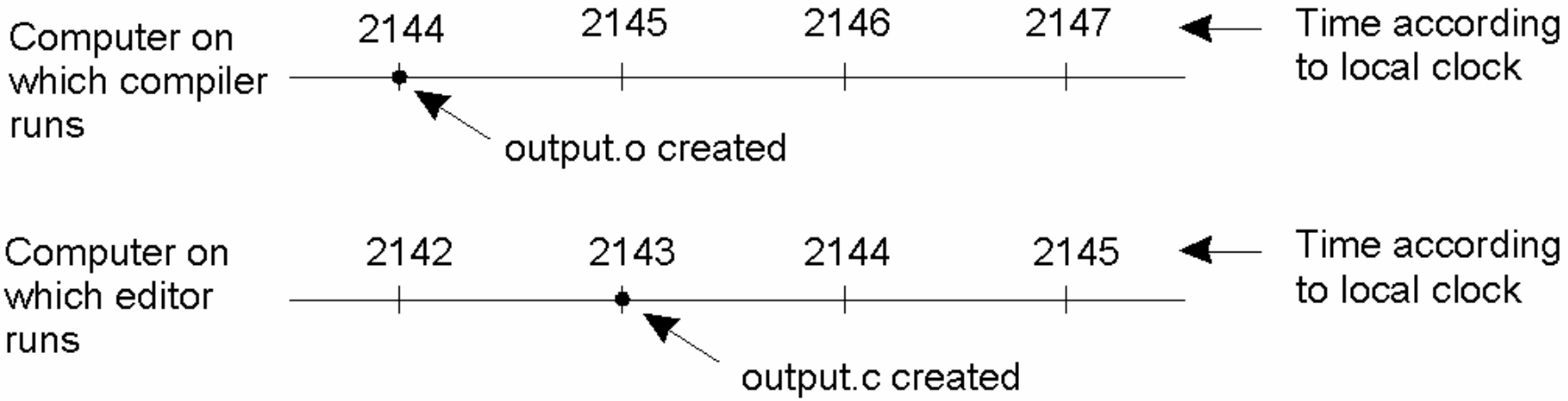
Sincronização em SD

- baseada no tempo real (absoluto)
- baseada na ordem relativa dos eventos
- baseada no estado global distribuído
- baseada no uso de um coordenador
  - mecanismos de eleição
- baseada em exclusão mútua
  - proteção contra acessos múltiplos
- baseada em transações distribuídas
  - mecanismos de concorrência avançados

informática
Universidade Federal de Santa Maria

Sincronização de relógios

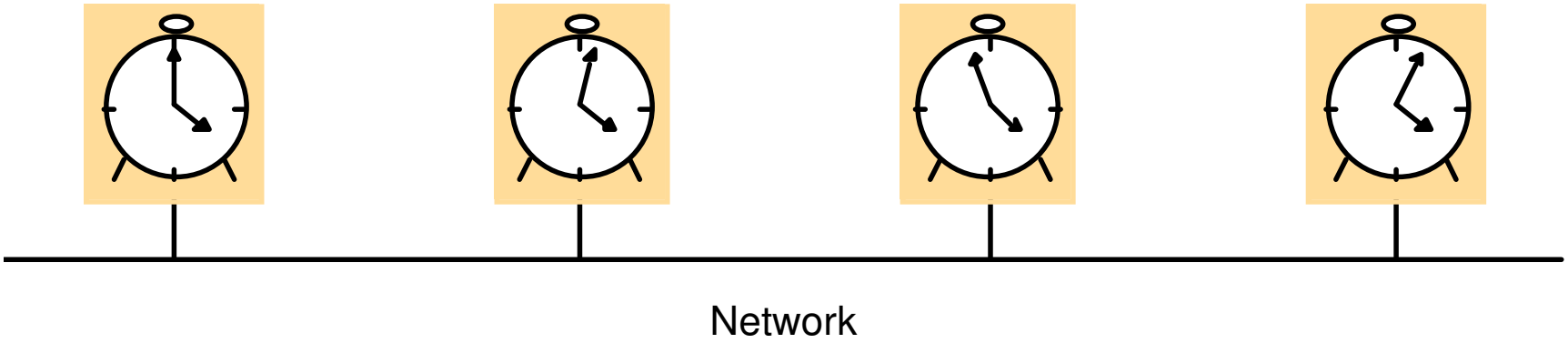
- Relógios em SD são independentes
- Relógios físicos são imprecisos (oscilação de cristais)
- Quando cada máquina tem seu próprio relógio, um evento A que ocorreu após um evento B pode ser marcado com um horário anterior ao de B



como `output.o` tem data posterior, o `make` não irá recompilar o `output.c`

informática
Universidade Federal de Santa Maria

Distorção (skew) entre relógios



informática
Universidade Federal de Santa Maria

Sincronização de relógios

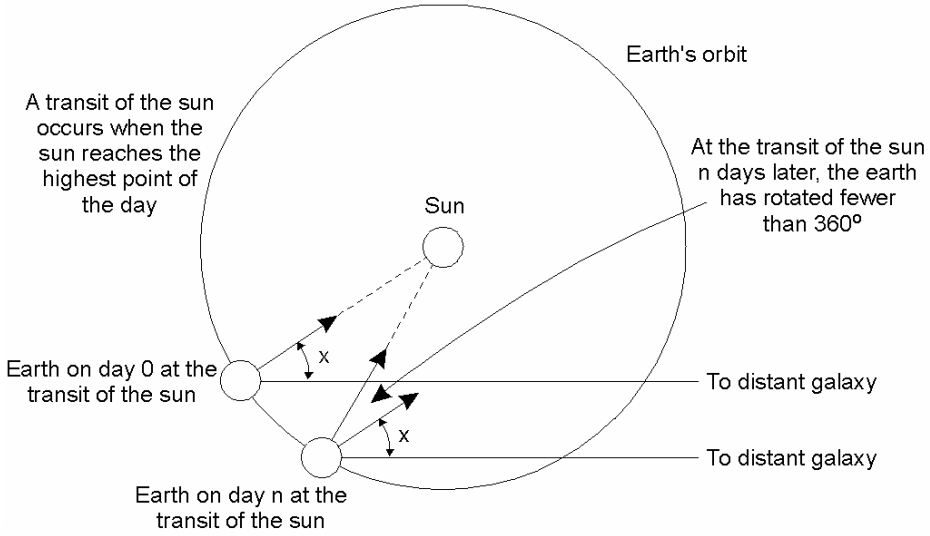
- Ponto chave: *timestamp* para cada evento

- Perguntas:
  - Como sincronizar os relógios com o mundo real?
  - Como sincronizar os relógios um com o outro?
  - Como determinar a seqüência correta dos eventos num sistema distribuído?

informática
Universidade Federal de Santa Maria

Hora terrestre

- Medida em relação ao sol



TAI - Tempo Atômico Internacional

média de tempo de 50 labs com cesium133

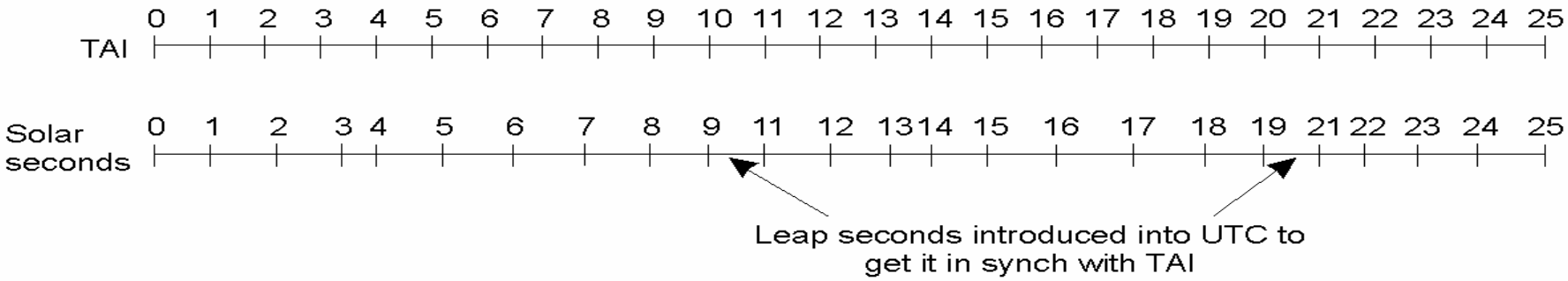
9192631770 oscilações = 1 segundo solar

86400s TAI = 1 dia solar ± 3ms

UTC - Universal Coordinated Time

a cada 800ms ajusta-se saltando um seg.

distribuído via WWV (rádio) ou GEOS (sat)



informática
Universidade Federal de Santa Maria

Hora no computador

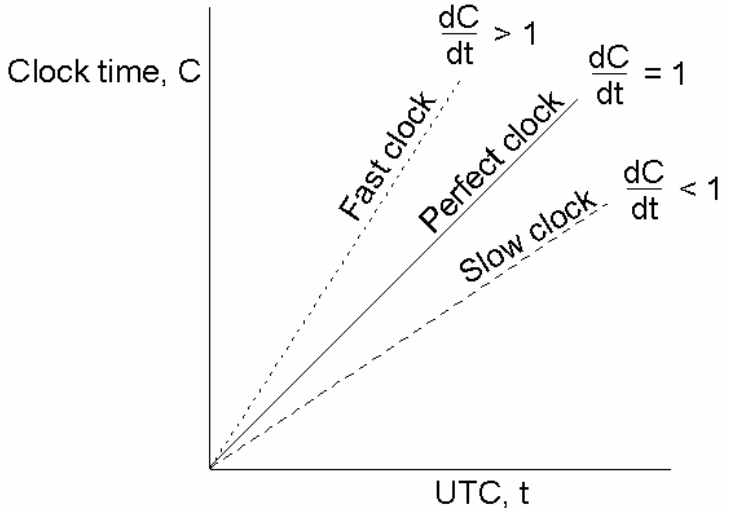
- Num computador:
  - Arquitetura: um cristal, um contador e um reg. de controle
  - Oscilação base a partir de um cristal ordinário sob tensão
  - Cada oscilação (pico de onda) decrementa um contador
  - Quando contador = 0, gera interrupção e contador ajustado para valor de um registrador de controle
  - Cada interrupção corresponde a um *tick*
  - Valor do registrador de controle define freqüência do *tick*
- Num Sistema Distribuído:
  - cristais não são idênticos
  - vários relógios derivam um do outro (*skew* - distorção)
  - cada relógio deriva em relação ao tempo ideal (*drift* rate)

informática
Universidade Federal de Santa Maria

Sincronização de relógios

- UTC como referência: *t*
- Relógio da máquina *p*: *Cp(t)*
- No mundo perfeito: *Cp(t) = t*, ou seja, *dC/dt = 1*
- Realidade: cada relógio tem um *drift* máximo ρ a cada Δt, logo $1 - \rho \leq dC/dt \leq 1 + \rho$
- para impedir uma derivação

maior do que δ, necessita-se sincronizar os relógios a cada $\delta/(2.\rho)$



informática
Universidade Federal de Santa Maria

Exercício

- Considere dois relógios C1 e C2
- C1 adianta e C2 atrasa
- t segundos depois C1 e C2 divergem em 2pt
- Se o projetista de um sistema distribuído espera garantir que C1 e C2 não se distanciem mais do que δ segundos, os relógios devem ser sincronizados no mínimo a cada quantos segundos?

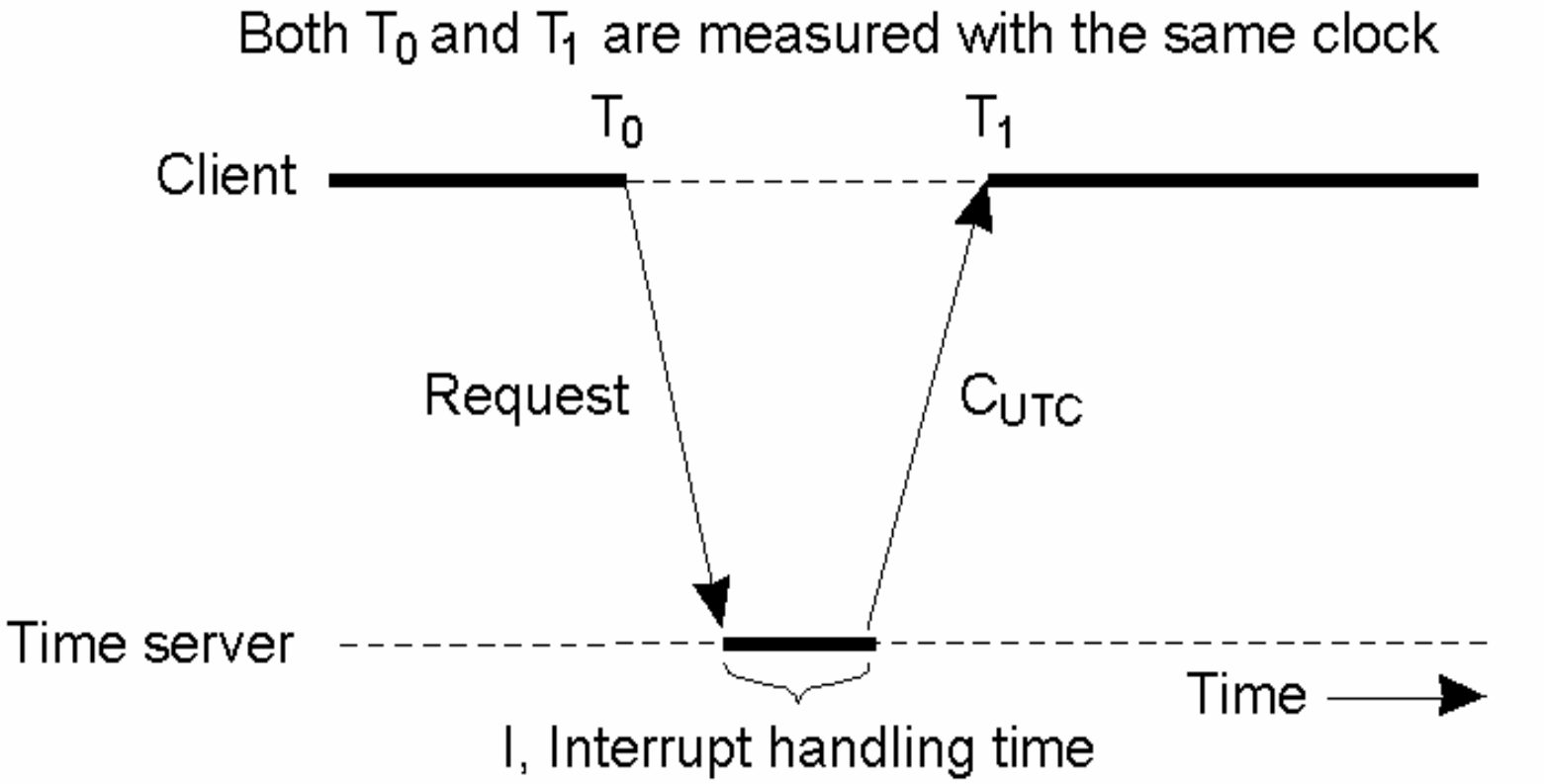
informática
Universidade Federal de Santa Maria

Algoritmo de Cristian

- Usa método de sincronização externa
- Periodicamente, cada máquina cliente contata servidor de tempo e requisita a hora.
- O servidor (WWV), informa a hora “oficial”.
- O cliente assume hora informada pelo servidor.

informática
Universidade Federal de Santa Maria

Algoritmo de Cristian



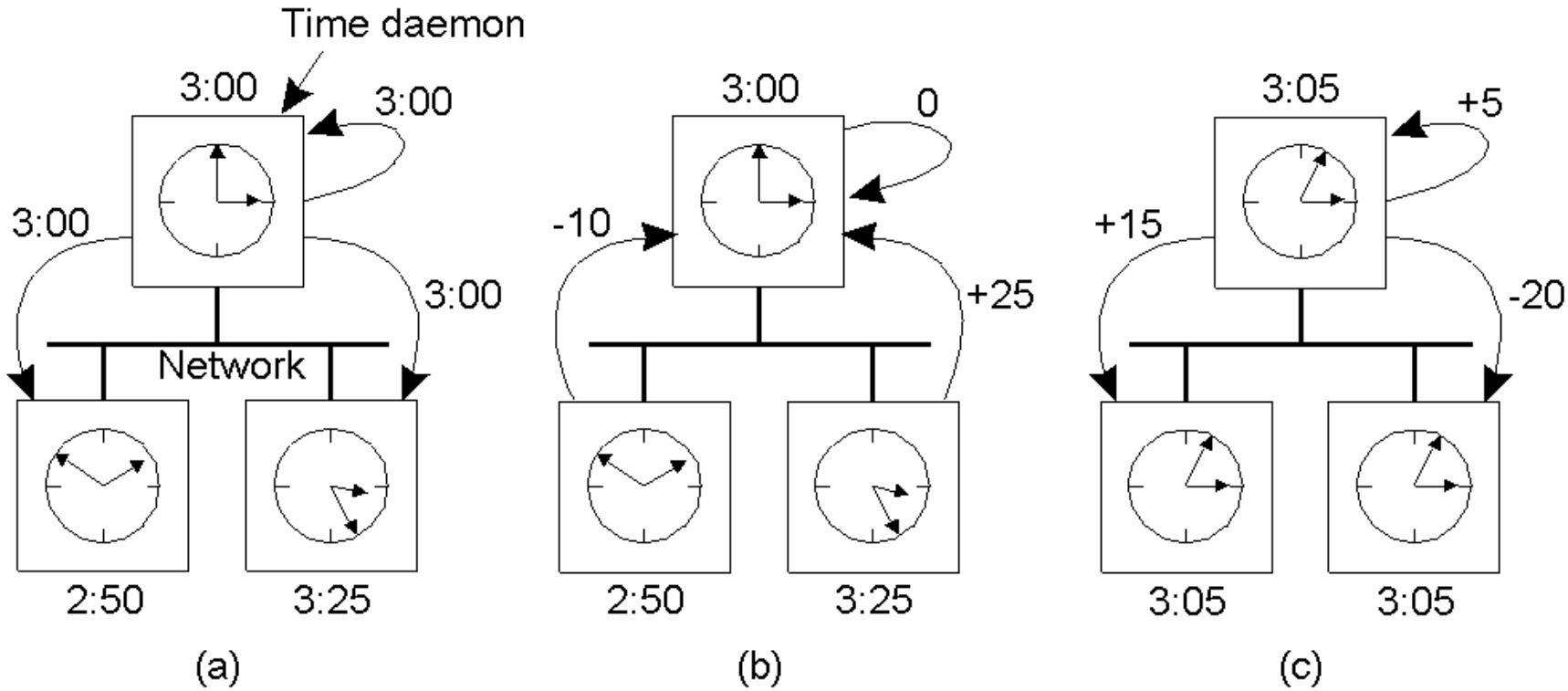
informática
Universidade Federal de Santa Maria

Algoritmo de Cristian

- Problemas:
  - Hora do cliente não pode “decrescer”.
    - Solução – fazer o relógio andar mais lento por um dado tempo, a fim de sincronizar com servidor
  - O atraso de transmissão pode ser relevante
    - Soluções:
      - Contabilizar o rtt e descontar tempo de viagem
      - Tentar descobrir tempo de computação do servidor
      - Tentar classificar atrasos curtos e longos

informática
Universidade Federal de Santa Maria

Algoritmo de Berkeley



(a) Daemon requisita valores dos relógios de todas as outras máquinas

(b) As máquinas respondem requisição

(c) Daemon comunica a todos como ajustar os seus relógios

informática
Universidade Federal de Santa Maria

Algoritmo pela média

- Divide tempo em intervalos de sincronização
- No início do intervalo broadcast a hora local
- Após broadcast, aguarda msgs no intervalo R
- Quando terminar R ou chegar todos broadcasts, computa a hora média

- V1: discarta os *m* superiores e *n* inferiores
- V2: tenta estimar atraso de comunicação
- Network Time Protocol (NTP) é muito usado

informática
Universidade Federal de Santa Maria

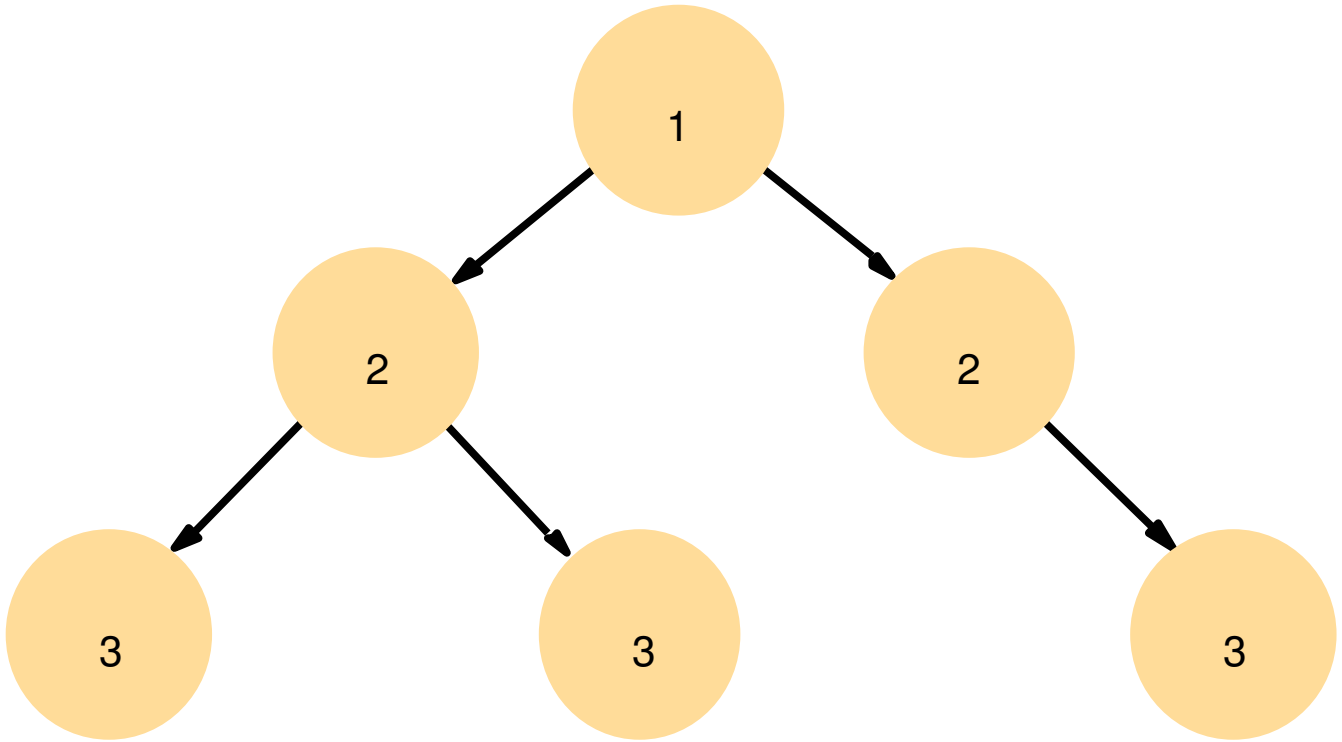
Network Time Protocol

- Soluções Cristian e Berkley são para intranets
- NTP é para Internet
  - Servidores primários conectados a fontes UTC
  - Servidores secundários sincronizam com primários ...
- Considera atrasos de comunicação

informática
Universidade Federal de Santa Maria

Network Time Protocol

- Sub rede de sincronização



Arrows denote synchronization control, numbers denote strata.

informática
Universidade Federal de Santa Maria

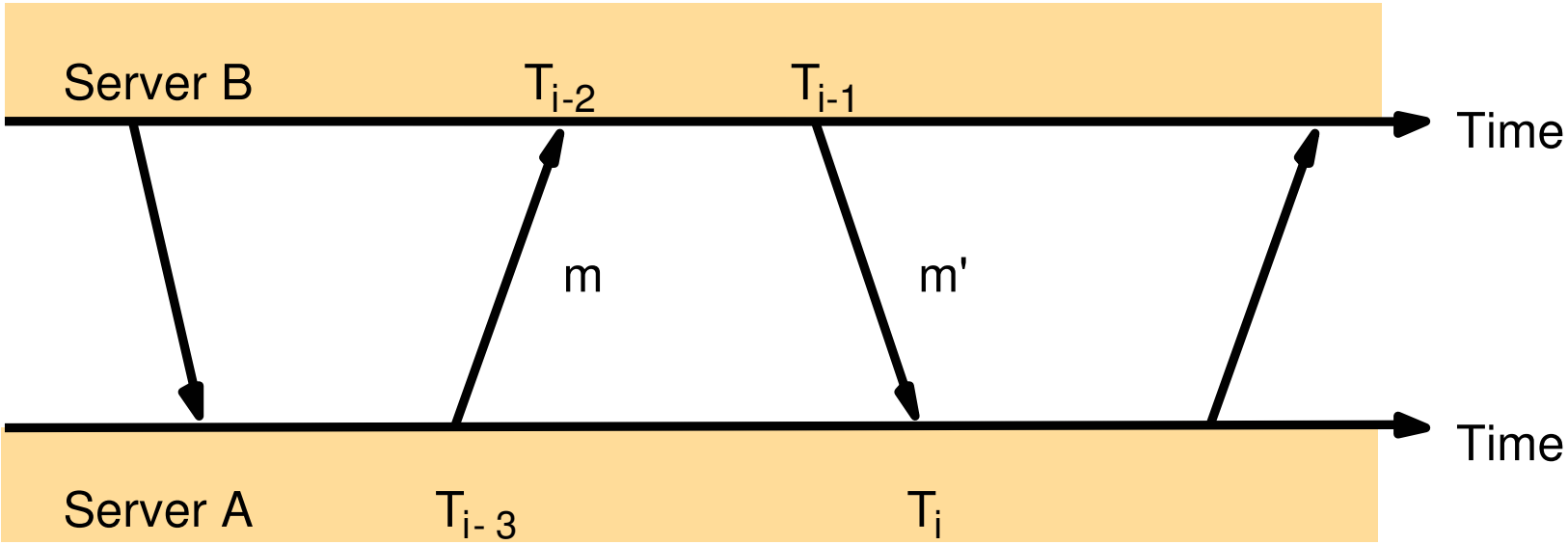
Network Time Protocol

- Oferece:
  - Sincronização de clientes com o UTC
  - Servidores e caminhos redundantes = serviço confiável
  - Escalabilidade (muitos clientes e servidores)
  - Segurança (usa mecanismo de autenticação)

informática
Universidade Federal de Santa Maria

Network Time Protocol

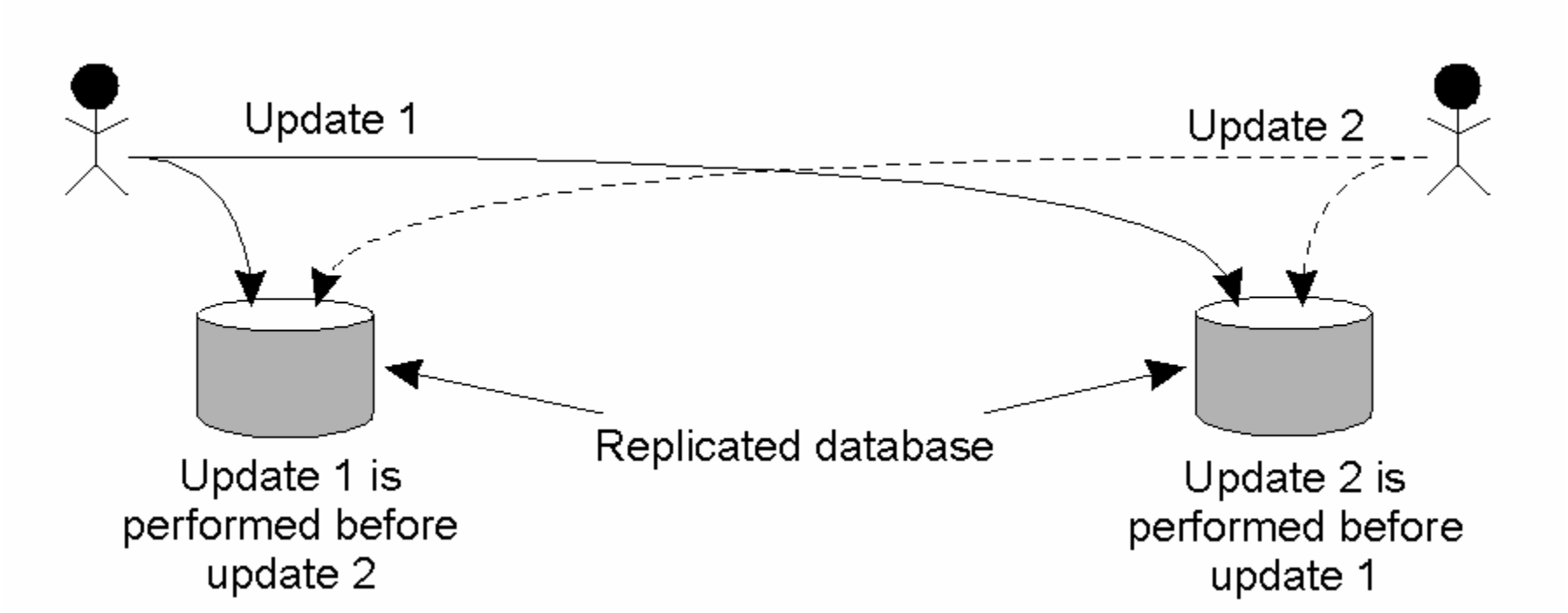
- Funcionamento
  - Mensagens entre um par de peers NTP



Compensação e atraso => filtro de dispersão

informática
Universidade Federal de Santa Maria

Ordenação de eventos



- Atualização da base de dados replicada deixando-a num estado inconsistênte.
- Multicast totalmente ordenado

informática
Universidade Federal de Santa Maria

Ordenação de eventos

- dificuldade de determinar relações temporais
  - RAZÃO: inexistência de relógio global
- problema
  - determinar ordenação temporal de eventos que ocorrem em nodos diferentes, medidos por relógios diferentes
  - relação: a "**aconteceu antes de**" b : $a \rightarrow b$

informática
Universidade Federal de Santa Maria

Ordenação de eventos

- ordem parcial
  - se **a** e **b** são eventos do mesmo processo e **a** é executado antes de **b** então **a** $\rightarrow$ **b**
  - se **a** é um **send** e **b** é um **receive** da mesma mensagem então **a** $\rightarrow$ **b**
  - **a** $\rightarrow$ **b** e **b** $\rightarrow$ **c** então **a** $\rightarrow$ **c**
  - se **a** e **b** são eventos concorrentes:
    - nem **a** $\rightarrow$ **b**, nem **b** $\rightarrow$ **a**

informática
Universidade Federal de Santa Maria

Clocks lógicos

- Lamport (78)
  - meio de assinalar um número a um evento
  - nenhuma relação com o tempo físico
  - sistema de relógio lógico
    - **C** - sistema de relógio, **Ci** - relógio local ao proc Pi
    - um sistema de relógio lógico é correto se é consistente com a relação $\rightarrow$

      para quaisquer eventos a e b,

      se **a** $\rightarrow$ **b** então **C(a) < C(b)**

informática
Universidade Federal de Santa Maria

Relógios lógicos

a maior parte dos problemas de ordenação podem ser resolvidos com relógios lógicos, sem necessidade de relógios físicos sincronizados

- relógio lógico
  - carimba um evento de forma que a **relação de ordem parcial** é mantida
  - pode ser facilmente implementado
    - usando contadores
    - numerando mensagens com numeração crescente
  - exemplo de implementação: *timestamp* T

informática
Universidade Federal de Santa Maria

Timestamps

- *Pi* inclui seu *timestamp* nas mensagens *m* que envia
  - *Tm:* *timestamp* da mensagem *m*

Ci - relógio local ao processo Pi

- 2 condições:
  - cada *Pi* incrementa *Ci* entre 2 eventos sucessivos
  - recebendo mensagem *m* com *Tm* ,
    Pj torna *Cj* maior ou igual ao seu valor atual **e maior que** *Tm*

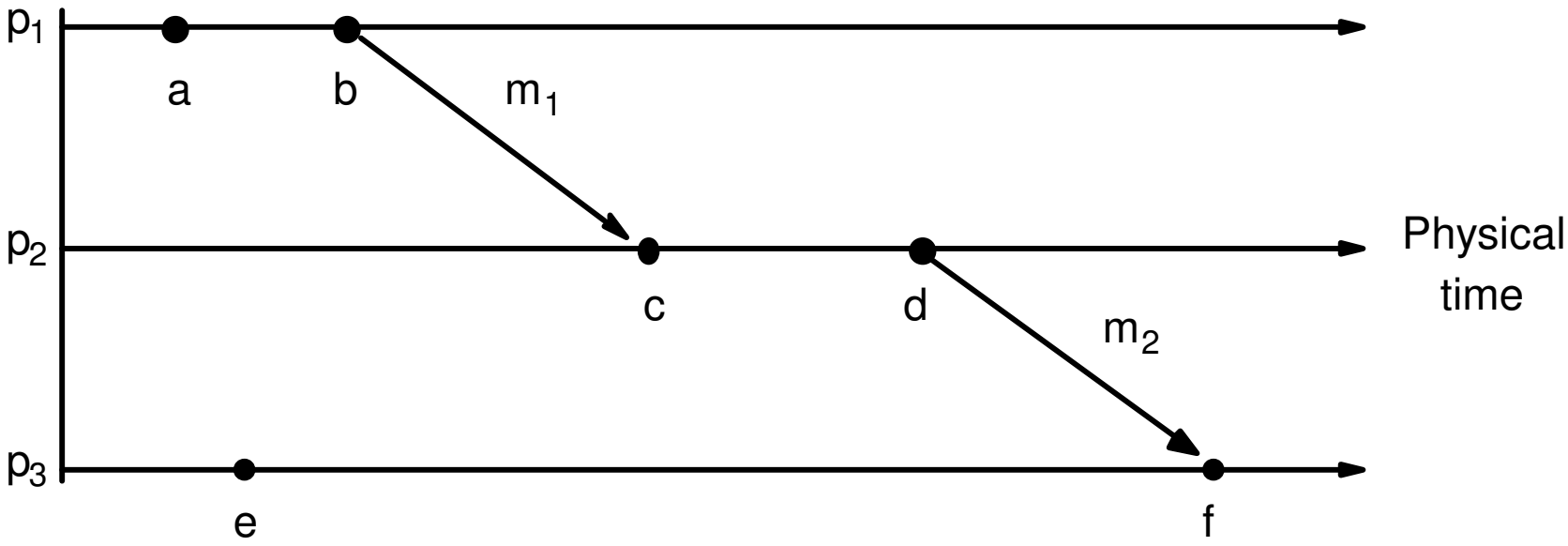
informática
Universidade Federal de Santa Maria

Timestamps

- as 2 condições asseguram realização da ordem parcial
- implementação:
  - um contador por nodo é suficiente
  - impossível assinalar *timestamps* consistentes com a relação de ordem parcial puramente usando relógios físicos
  - relógios físicos precisariam ser sincronizados para assegurar a relação de ordem parcial

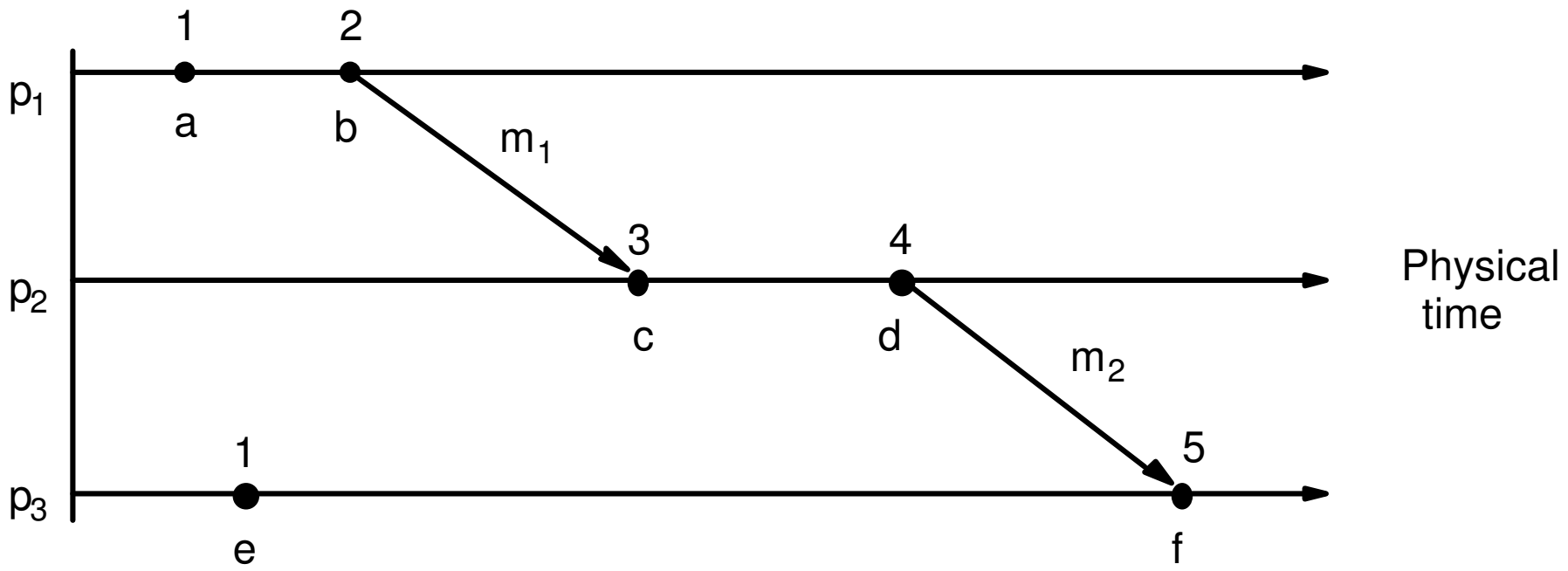
informática
Universidade Federal de Santa Maria

Eventos entre três processos



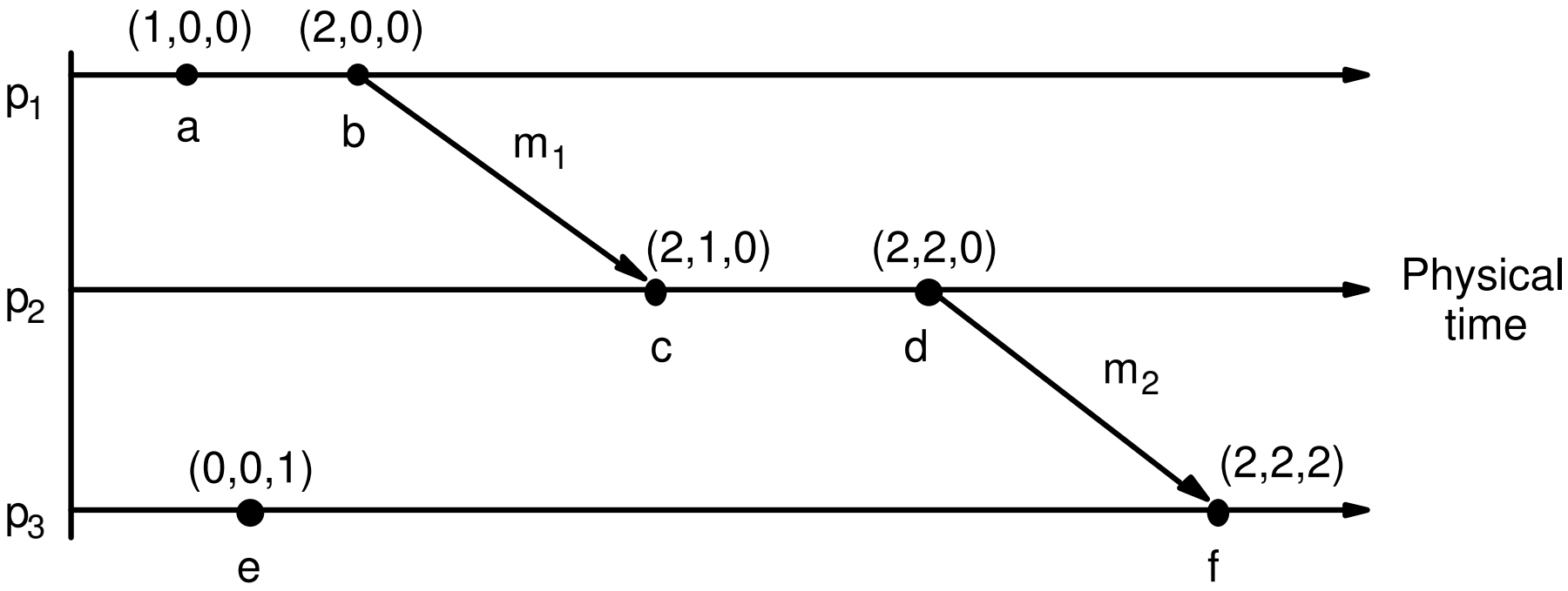
informática
Universidade Federal de Santa Maria

Relógio lógico de Lamport



informática
Universidade Federal de Santa Maria

Vetor de relógios



informática
Universidade Federal de Santa Maria

Ordenação total com relógio lógico

- relógio lógico pode ser usado para ordenação total (Lamport 78)
  - mensagens de diferentes processos podem possuir o mesmo *timestamp*
  - se duas mensagens possuem o mesmo *timestamp, então* devem ter sido originadas em diferentes processos
  - para estabelecer uma ordem, basta ordenar os processos de alguma forma

a mesma forma de ordenação deve ser seguida por todos os processos

informática
Universidade Federal de Santa Maria

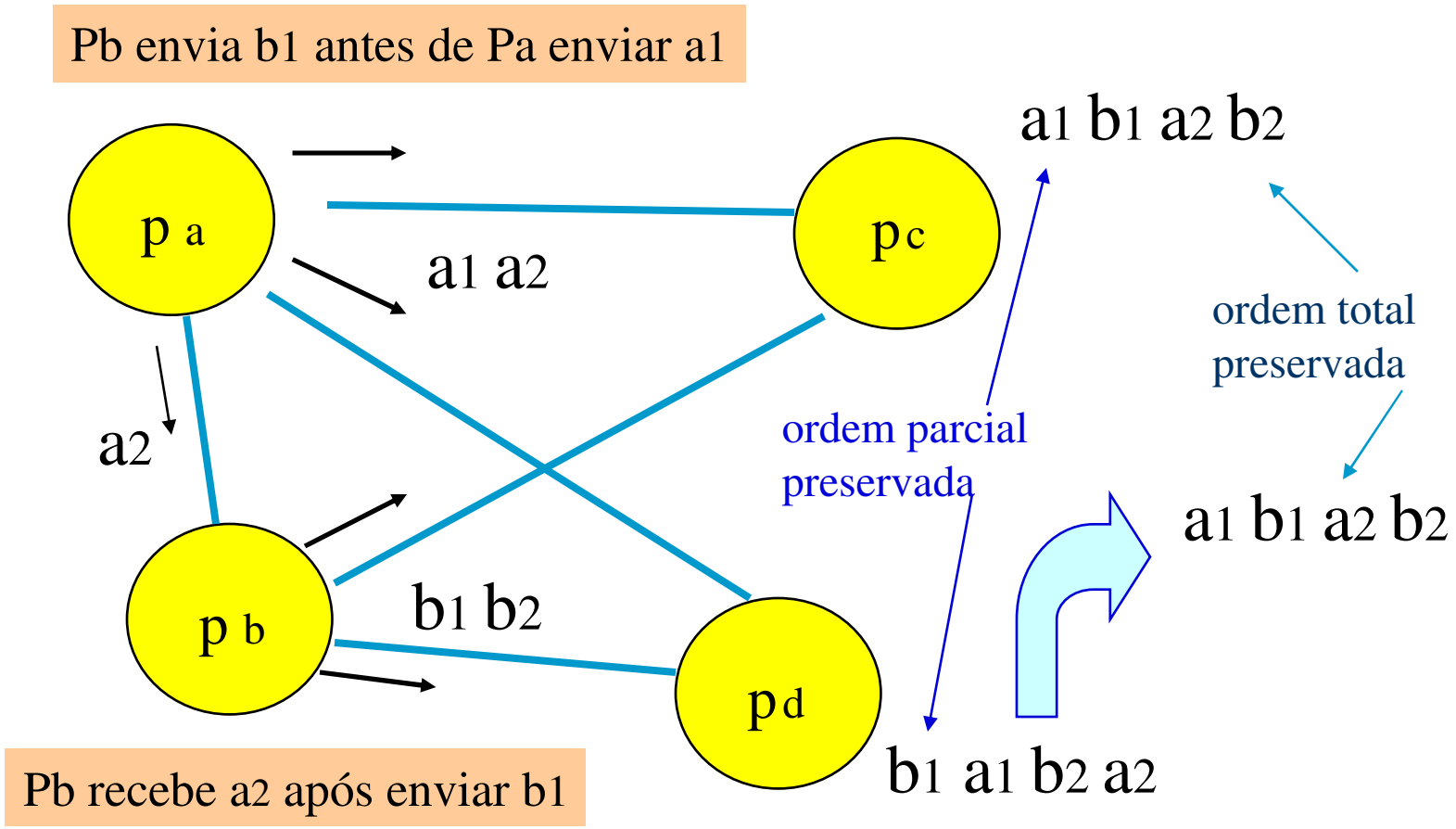
Ordenação total

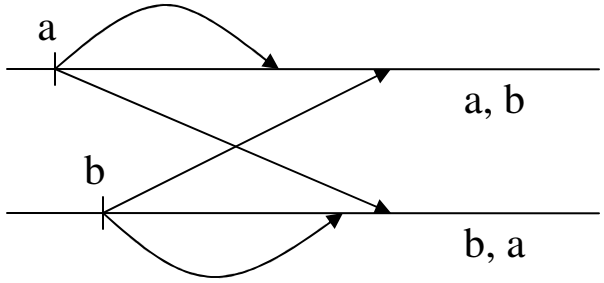
- uma implementação possível
  - eventos com mesmo *timestamp* são ordenados pela ordem lexicográfica dos nomes dos processos
- preserva a relação de ordem parcial $\rightarrow$
  - todos os nodos classificam os eventos na mesma ordem,
  - não assegura preservação de ordem temporal dos eventos concorrentes

se a ordem temporal for importante, relógios lógicos não podem ser empregados

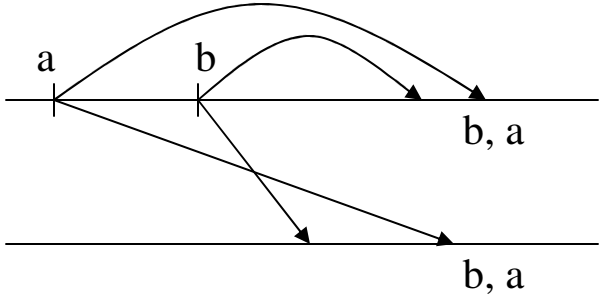
informática
Universidade Federal de Santa Maria

Exemplo

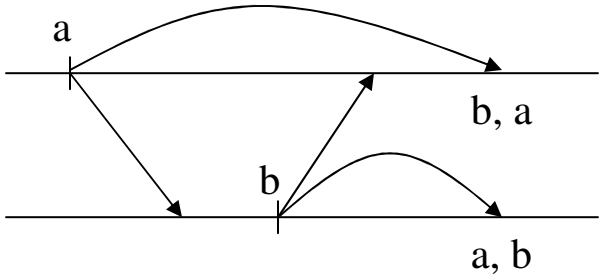




(a) ordem causal



(b) ordem total



(c) ordem FIFO

Nota: Ordem causal implica em ordem FIFO

Relações entre requisitos de ordem