Sincronização de Eventos, Exclusão Mútua e Acordos

Sistemas Distribuídos e Programação Concorrente

Prof. MSc. Rodrigo Malara

Motivação

- Comunicação é importante – Troca de informações
- Tão importante quanto:
  - Sincronização
  - Regiões críticas
- Soluções convencionais depende de memória compartilhada
  - Semáforos, mutex e monitores
- Precisamos de alguma forma de sincronizar processos sem usar memória compartilhada e isso envolve temporização

Tópicos

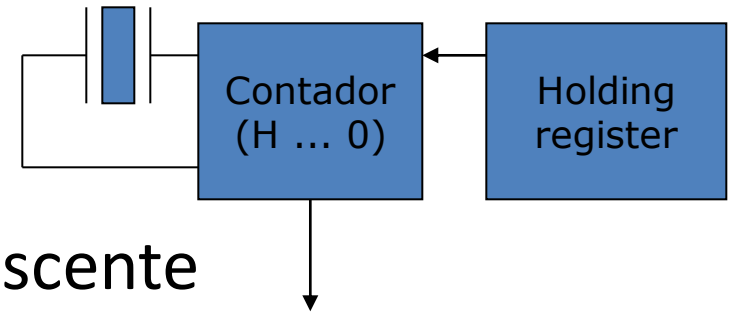
- Medidas de tempo
  - Físicas
  - Lógicas
- Exclusão mútua
- Eleição

Sincronização de relógio

- Em sistema centralizado, tempo não é ambíguo
  - Algo é responsável por manter o tempo
- Em sistemas distribuídos, tempo pode ser ambíguo
  - Pois existem vários sistemas computacionais
    - Cada um com seu relógio individual
  - Ex: Sistemas de arquivos distribuídos
    - Qual arquivo é o mais novo?

Sincronização de relógio

- Computadores têm temporizadores, e não relógios
  - SO ajusta “H”
  - Clock em contagem decrescente
  - Zerou? Gera interrupção



- Sistema operacional usa interrupções para acertar relógio em software
- Cada computador computador possui
  - Um cristal, uma freqüência, um relógio

Sincronização de relógio

- Cristais de quartzo não são perfeitos
  - Têm pequenas variações na freqüências
- Vários computadores
  - Relógios ficarão gradualmente fora de sincronismo
    - Drift ou deslize

Relógios físicos

- Alguns sistemas precisam de tempo físico sincronizado
- Há a necessidade de um relógio externo para sincronização
  - Ponto de referência
  - Somente 1!?
    - Vários, para garantir eficiência e redundância
- Vários relógios? Como sincronizá-los?

Conhecimentos Gerais

Como o tempo físico é medido?

Dia solar médio

- Tempo entre dois trânsitos do sol
  - Sol chegar no ápice aparente celeste
  - 24 horas, 3600 segundos por hora
  - Segundo solar = 1/86400 dia solar
- Período de rotação não é constante
  - Arrasto aerodinâmico da atmosfera
  - 300 milhões anos atrás → 1 ano = 400 dias
  - Dias ficando mais longos
- Segundo solar médio!
  - Precisão insuficiente para muitas aplicações

Tempo atômico internacional

- Relógio atômico em 1948
  - 1 segundo = 9 192 631 700 transições Ce_{133}
- 50 laboratórios no mundo
  - USP São Carlos tem um!
- De tempos em tempos, cálculo do TAI
  - Média do número de ticks desde 01/01/1958 dividido por 9 192 631 700
  - Número de segundos desde de 01/01/1958

Tempo atômico universal

- Problemas
  - Relógio atômico é caro para ter num SD...
  - Estável, mas o dia solar não!
    - Em 1995, 3mseg menos que o dia solar médio
- Introdução de segundos bissextos (leap) no dia solar
  - Caso a diferença entre Tempo Atômico Internacional e solar seja maior que 800ms
  - Tempo Universal Coordenado (UTC)

Como obter o UTC?

- Se encontra no fuso horário 00:00
  - Greenwich - Inglaterra
- Propagados pelas estações de medição do tempo
  - Estação de ondas curtas
    - Beep a cada segundo UTC
  - GPS
    - Possui uma precisão ±1ms
  - Network Time Protocol
    - https://ntp.br

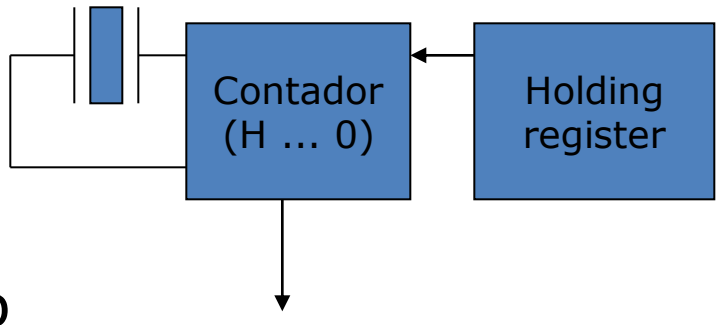
Continuando com algoritmos de sincronização de relógios

Algoritmos de sincronização

- Objetivo
  - Manter outros relógios sincronizados com o de uma máquina com relógio correto
  - Manter as máquinas com os relógios o mais próximo possível
- Idéia básica
  - $C_p(t) = t$
    - Hora do computador p é igual à hora ideal t (UTC)
  - $dC/dt = 1$
    - Variação do relógio do computador deve ser zero

Algoritmos de sincronização

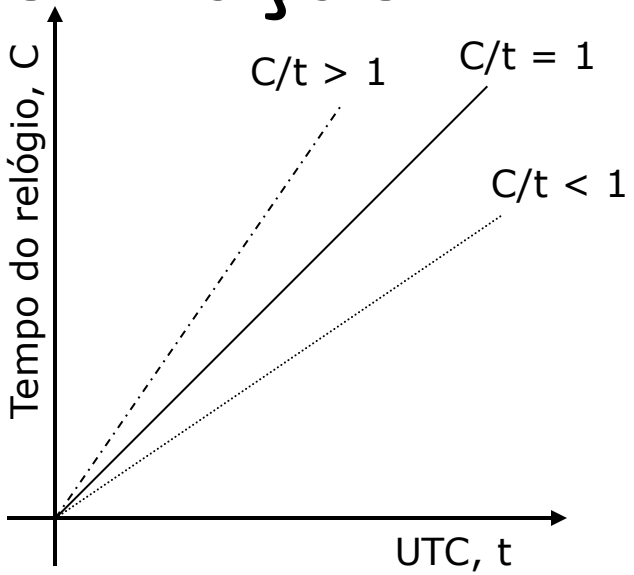
- Temporizadores não são perfeitos
  - Precisão típica de 10^{-5}
    - Atrasa/Adianta 1s a cada 27,7 horas
    - ρ = 0,00001 (taxa máx. flutuação)
    - Fornecida pelo fabricante
  - Para estar dentro da especificação



$$(1-\rho) \leq \frac{dC}{dt} \leq (1+\rho)$$

  - Exemplo: H = 60 (interrupções por seg.)
    - Ou 216000 interrupções/h
  - (1- 10^{-5})*216000 < int/h < (1+10^{-5})* 216000
  - 215998 < interrupções/hora < 216002

Algoritmos de sincronização



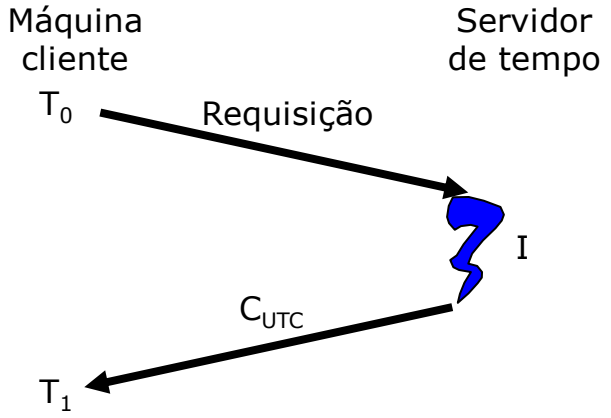
- Dois relógios
  - Flutuando em direções opostas
- Para mantê-los com diferença máxima δ
  - Sincronizá-los a cada ($\delta/2\rho$)
  - Considerando exemplo anterior. Se quisermos um δ máximo de 1 seg temos: $1/2 * 10^{-5}$ = 13,85 horas
- Todos os algoritmos fazem isso, só muda como!

Algoritmo de Cristian

- Necessita de um servidor de tempo
- No máximo a cada ($\delta/2\rho$), clientes pedem o tempo ao servidor
  - Resposta: C_{UTC}
- Acertar o relógio com o novo tempo

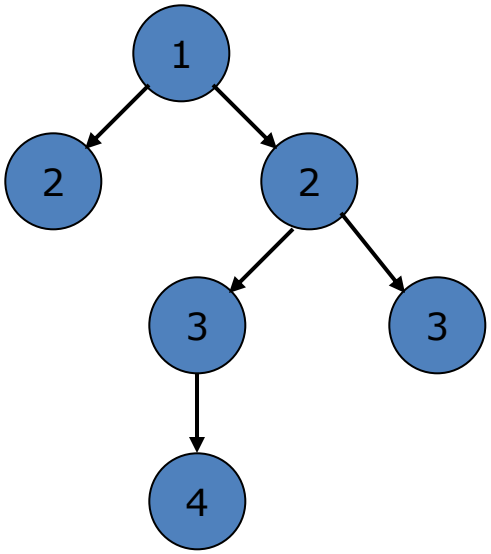
Cristian: problema

- Mensagens não são instantâneas
  - Tempo de propagação da msg
    - $T_1 - T_0 = \Delta$
    - $C_C(t) = C_{UTC} + \Delta/2$
  - Tempo de tratamento da mensagem (geralmente desconhecido
    - $T_1 - T_0 - I = \Delta$
    - $C_C(t) = C_{UTC} + \Delta/2$
  - Efetuar várias medidas e usar a média



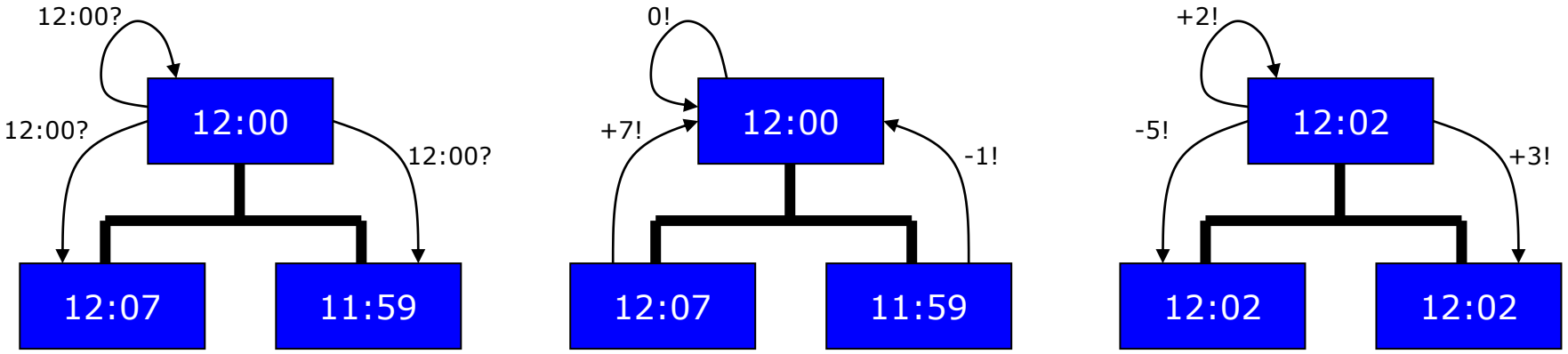
Network Time Protocol

- Forma de sincronização física
  - Usa protocolo UDP/IP
- Hierarquia de servidores
  - Níveis são *strata*
  - Sincronizar com nível superior
    - Menos suscetíveis a erros de propagação
- Três formas de sincronização
  - Multicast: só para redes com baixa latência
    - Servidores divulgam timestamp aos clientes
  - Cliente solicita: usa Cristian com médias



Algoritmo de Berkeley

- Cristian assume servidor passivo
- Em Berkeley, servidor é ativo – não depende de entidade externa para saber a hora exata
  - Pergunta às máquinas o tempo
  - Calcula média
  - Notifica novo tempo



Questões

1. Defina
   1. Deslize de relógio
   2. Skew do Relógio
2. Supondo que a precisão do clock de um computador (ρ) é 10^{-4}. Calcule qual é a freqüência de sincronização mínima entre o computador e o servidor de tempo para um atraso de max. de 1 segundo
3. Já que a precisão é muito importante na sincronização de relógios, como o algoritmo de Cristian lida com o tempo variável de requisição e resposta do clock do servidor de tempo ?
4. Explique com as suas palavras o que é o NTP
5. Qual a diferença entre o algoritmo de Berkley e o de Cristian?

Tempo lógico

- Processos vêm evento ordenados pelos seus relógios locais
- É impossível garantir sincronismo absoluto
- Lamport mostrou que:
  - Não é necessário usar tempo físico em algumas situações
  - Ordenação não precisa de precisão, basta saber quem veio antes de quem...

Tempo lógico

- Ordenação lógica parte de dois pontos intuitivos e óbvios
  - Dois eventos no mesmo processo ocorrem na ordem em que são observados
    - Pode-se confiar na ordem de 2 eventos se ela ocorre no mesmo computador
  - Quando processos trocam mensagens, o envio tem que ocorrer antes do recebimento

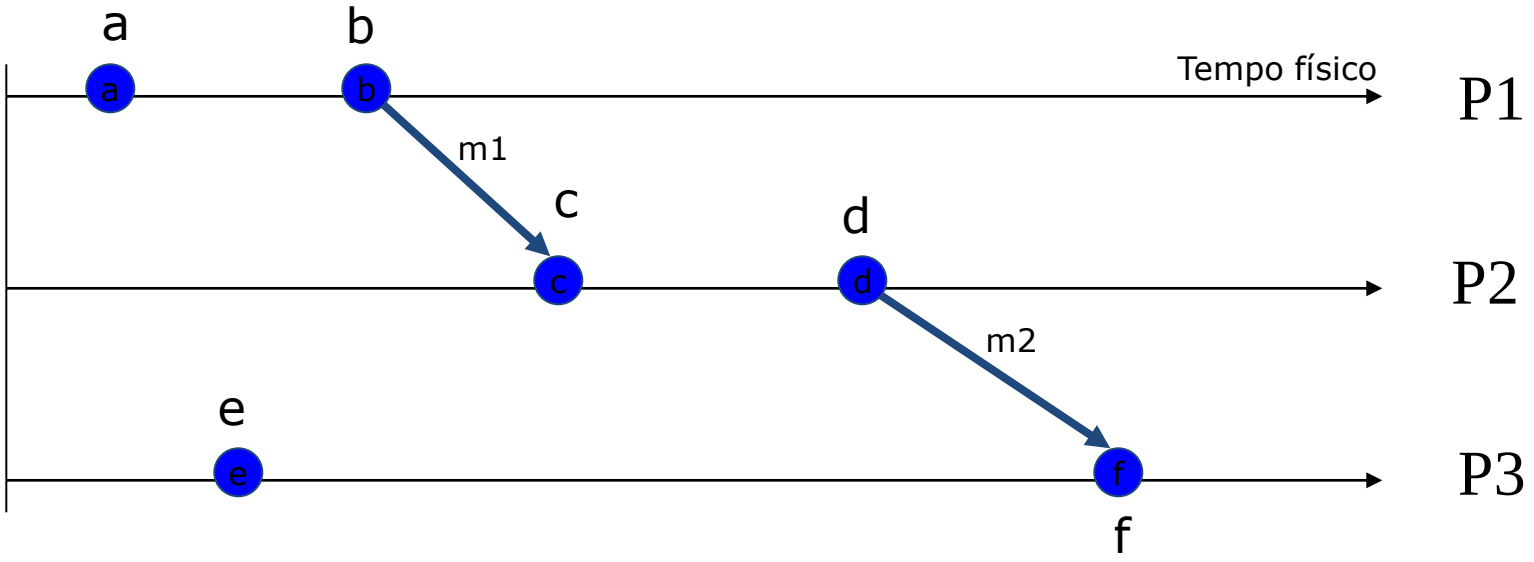
Relação de happens-before

- Happens before (HB) ou “Acontece Antes”
  - (Potencial) ordenação causal
  - Notação: x → y
    - Leia-se X acontece antes que Y
- Se dois eventos, *x* e *y*, ocorrem no mesmo processo *p*, e *x* acontece antes de *y*
  - $x \xrightarrow{P} y$
- Mais formalmente...

Relação de happens-before

- Considere que X, Y e Z são eventos
- HB1: Se $\exists$ processo *p*: $x \xrightarrow{P} y$ então
  - x → y
- HB2: Para qualquer mensagem *m*
  - send(m) → recv(m)
- HB3: se *x*, *y* e *z* são eventos tal que x → y e y → z, então
  - x → z

Relação de happens-before



a → b; b → c; c → d; d → f ⇒ a → f

Não há cadeia de mensagens entre *a* e *e*, logo a ↛ e ou (a || e)

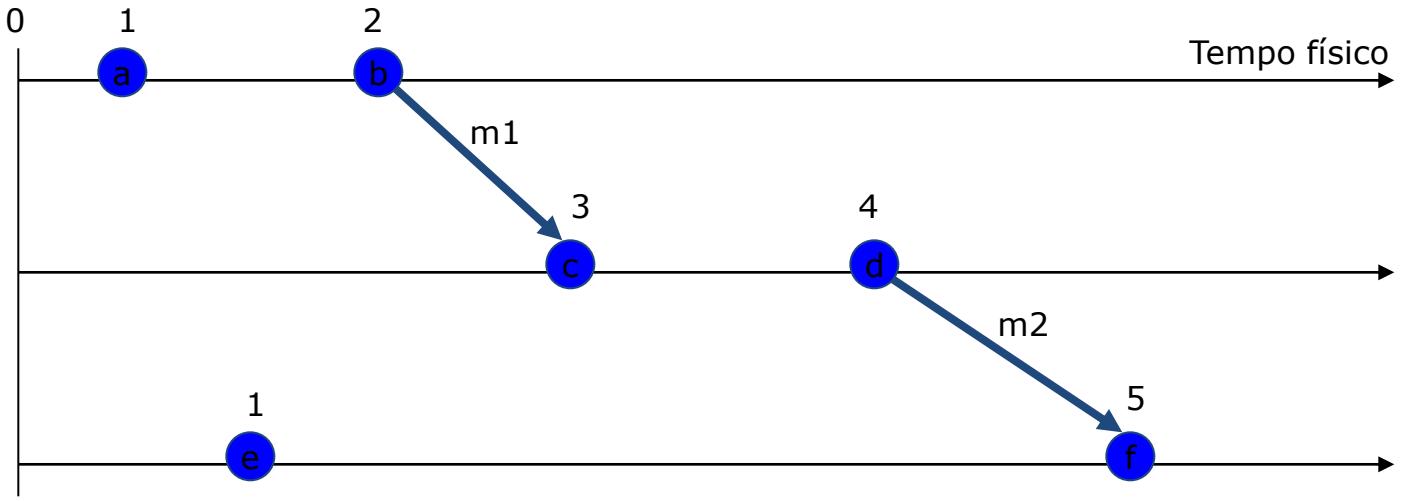
Relógio lógico (Lamport)

- Captura numérica da relação Happens Before
  - Contador em software incrementado de forma monótona (uma única direção)
  - Valores do contador não tem relação com relógio físico
- Cada processo com seu contador C_P
  - Tempo de A no processo P = $C_P(A)$
  - Tempo de B em qualquer processo = C (B)

Relógio lógico

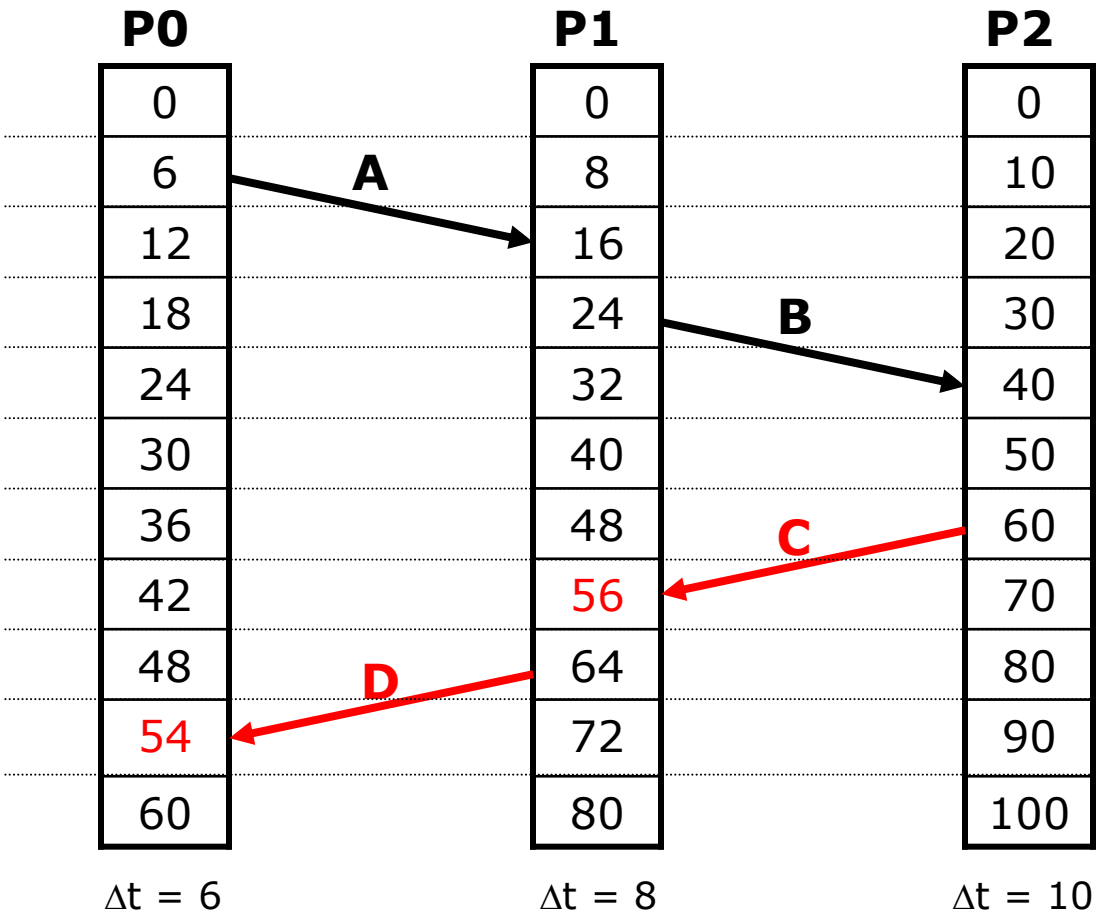
- Regras de incremento/envio de relógio
  - Antes de cada computação local em P
    - C_P++
  - Quando P envia uma mensagem M, ele manda seu relógio lógico t=C_P
    - (M,t)
  - Para receber (M,t), um processo Q calcula C_Q=max(C_Q,t) e recebe (M,t) no tempo C_Q++

Relação de happens-before

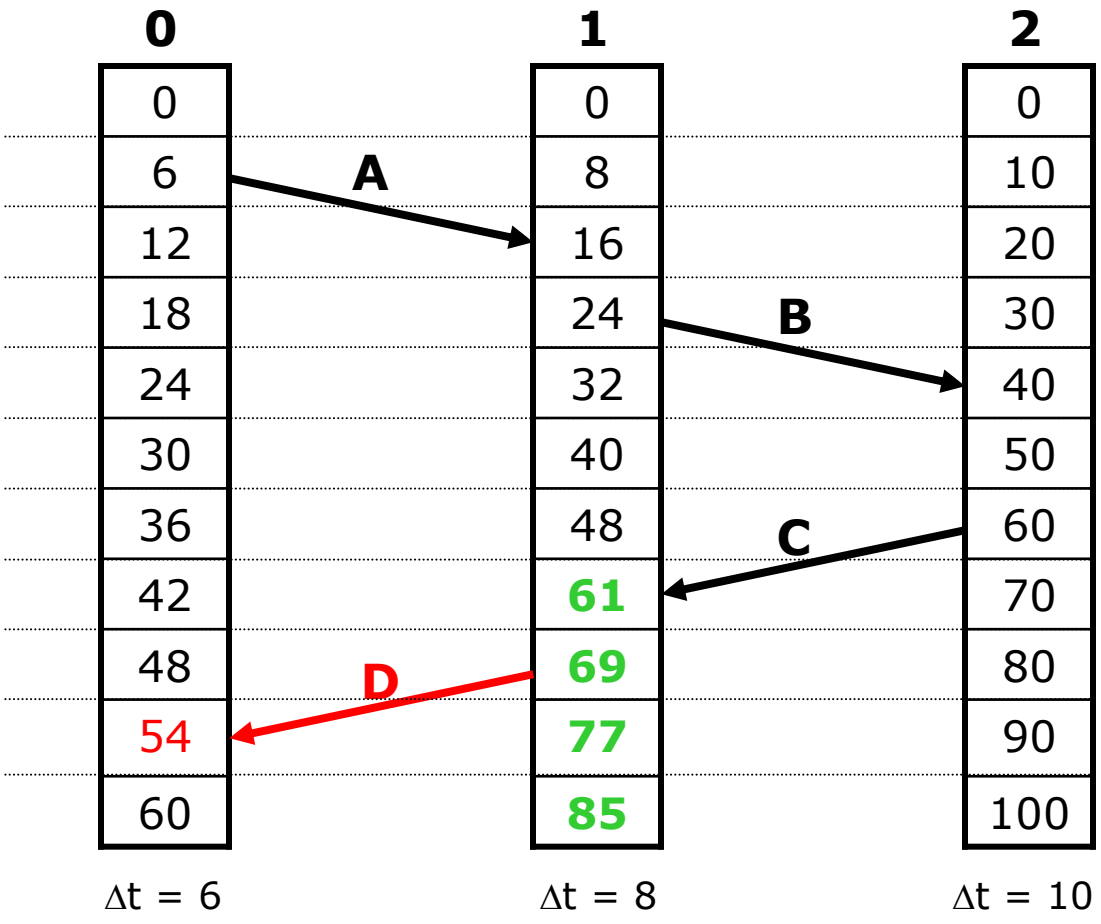


a → b ⇒ C(a) < C(b)
c → f ⇒ C(c) < C(f)

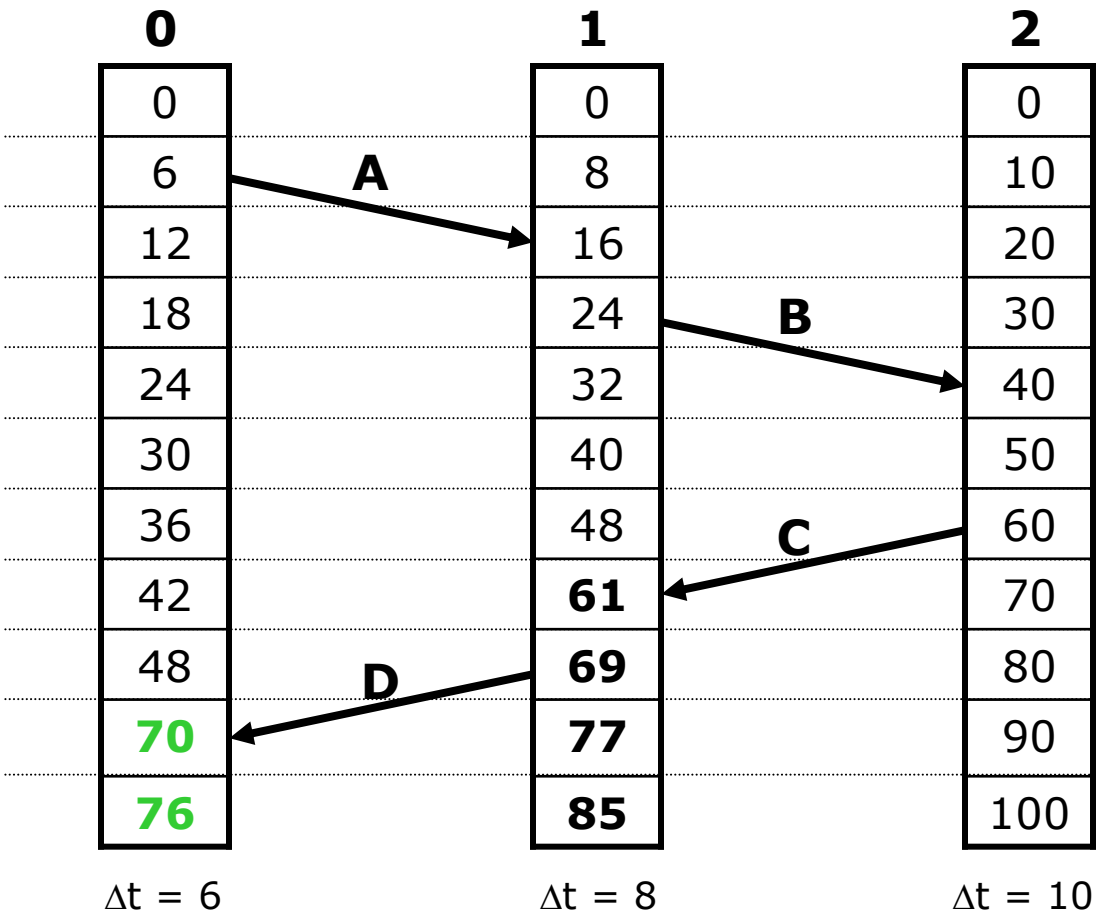
Algoritmo de Lamport



Algoritmo de Lamport



Algoritmo de Lamport



Ordenação Total de Eventos usando Relógios Lógicos

- O Algoritmo de Lamport não é infalível
  - Pode acontecer de 2 processos terem o mesmo valor de relógio lógico ao mesmo tempo
- A ordenação total usa o número do processo
  - Pode ser útil em algumas situações
- Evento *a*, processo P_A no tempo T_A
  - (T_A, P_A)
- Evento *b*, processo P_B no tempo T_B
  - (T_B, P_B)
- $(T_A, P_A) \rightarrow (T_B, P_B)$ se, e somente se
  - $T_A < T_B$
  - $T_A = T_B$ e $P_A < P_B$