Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Computação
Bacharelado em Ciência da Computação

Sistemas Distribuídos

Tempo e Estados Globais

Prof. Rodrigo Campiolo

05/10/20

Tópicos

- Introdução
- Definições
- UTC
- Sincronização
- NTP
- Relógios Lógicos
- Relógios Vetoriais
- Atividade

Introdução

- Tempo
  - Quando um evento ocorreu em um computador? Importante para auditoria.
  - Algoritmos dependem da sincronização do relógio:
    - consistência de dados distribuídos.
    - autenticidade de uma requisição ao servidor.
    - eliminar atualizações duplicadas.
- Estado Global
  - Observar os estados dos processos.

Definições

- **Evento:** a ocorrência de uma única ação realizada pelo processo (ação de comunicação ou uma ação de transformação de estado).
- **Relógio (Clock):** um dispositivo eletrônico que conta oscilações ocorrendo em um cristal segundo uma frequência definida. Geralmente dividem a contagem e armazenam em um registrador.

Definições

- **Clock skew:** a diferença na leitura de dois relógios.
- **Clock drift:** divergência na contagem do tempo.
- **Clock drift rate:** divergência na contagem do tempo de um relógio em relação a um relógio ideal (para relógios quartz equivale a 1s a cada 11,6 dias).

UTC

- Universal Time Coordinated
  - Padrão internacional para registro de tempo.
  - Baseado em relógio atômico.
  - Às vezes é necessário somar ou subtrair segundos (*leap second*) para manter equivalência com o tempo astronômico.

  - Para saber mais:
    - http://oal.ul.pt/hora-legal/tempo-universal-coordenado/

Sincronização

- Relógios de computadores podem ser sincronizados com fontes externas de alta precisão.
- Sincronização de relógios físicos:
  - **Sincronização externa**: sincronizar os relógios dos processos com uma fonte autoritativa externa.
  - **Sincronização interna**: sincronizar o relógio de um processo com outro relógio com um limite de precisão.

Sincronização

- Sincronização em um SD síncrono
  - Limites conhecidos: *drift rate*, *delay*, tempo nos processos.
  - Processo envia o tempo t e o processo receptor atualiza o relógio com t + $T_{transmissão}$.
  - Pode-se considerar:
    - $T_{transmissão}$ = min
    - $T_{transmissão}$ = max
    - $T_{transmissão}$ = (min + max) / 2

Sincronização

- Em geral, os SD são assíncronos, logo o tempo de transmissão não tem um limite superior.

Sincronização

- Algoritmo de Cristian
  - Uso de um servidor de tempo conectado a uma fonte UTC.
  - Cliente solicita “tempo” ao servidor.
  - Servidor devolve “tempo t”.
  - Cliente atualizaria: $t + T_{round\ trip} / 2$

- Para saber mais: https://www.youtube.com/watch?v=yvuy0rPkv8Q

Sincronização

- Algoritmo de Berkeley
  - Um computador é escolhido como *mestre*.
  - O *mestre* consulta os computadores a serem sincronizados (*escravos*).
  - Os *escravos* enviam os valores de tempo.
  - O mestre faz uma estimativa do tempo considerando o RTT (*Round Trip Time*) e o seu próprio relógio.
  - O mestre calcula a média dos tempos, ignorando valores que estão muito divergentes da maioria.
  - O mestre envia o tempo de ajuste (positivo ou negativo para cada escravo).

Network Time Protocol (NTP)

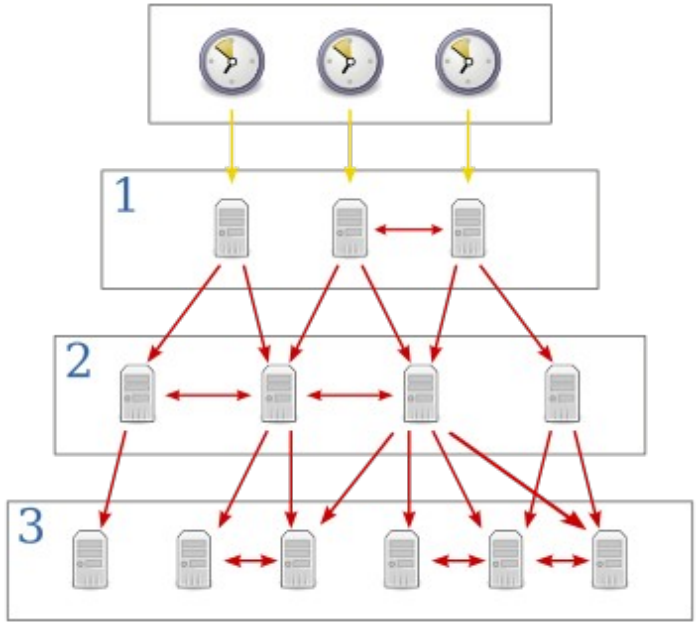
- NTP provê um serviço de tempo e um protocolo para distribuir informações de tempo na Internet.

Relógios Atômicos

Servidores Primários (Stratum 1)

Servidores Secundários (Stratum 2)

Servidores Terciários (Stratum 3)



Fonte: https://en.wikipedia.org/wiki/Network_Time_Protocol

Network Time Protocol (NTP)

- Os servidores NTP são sincronizados entre si de três formas:
  - Multicast
  - RPC
  - Simétrico
- Uso do protocolo UDP e porta 123
- Pode-se obter precisão de sincronização:
  - Internet: dezenas de milissegundos
  - Rede local: 1 milissegundo.

Tempo e Relógio Lógico

- Como não podemos sincronizar perfeitamente relógios em um SD, não podemos usar relógios para descobrir a ordem de eventos arbitrários.
- Lamport - https://en.wikipedia.org/wiki/Leslie_Lamport
- Esquema simples baseado na causalidade:
  - Dois eventos em um mesmo processo, então ocorreram na mesma ordem que o processo observou.
  - Quando uma mensagem é enviada entre dois processos, o evento de envio ocorreu antes do evento de recepção.

Tempo e Relógio Lógico

- Lamport generalizou esse conceito como:
  - Relação **happened-before**. Também é conhecida por ordenação causal ou ordenação causal potencial.

- Relação **happened-before**:

HB1: If $\exists$ process $p_i : e \rightarrow_i e'$, then $e \rightarrow e'$.

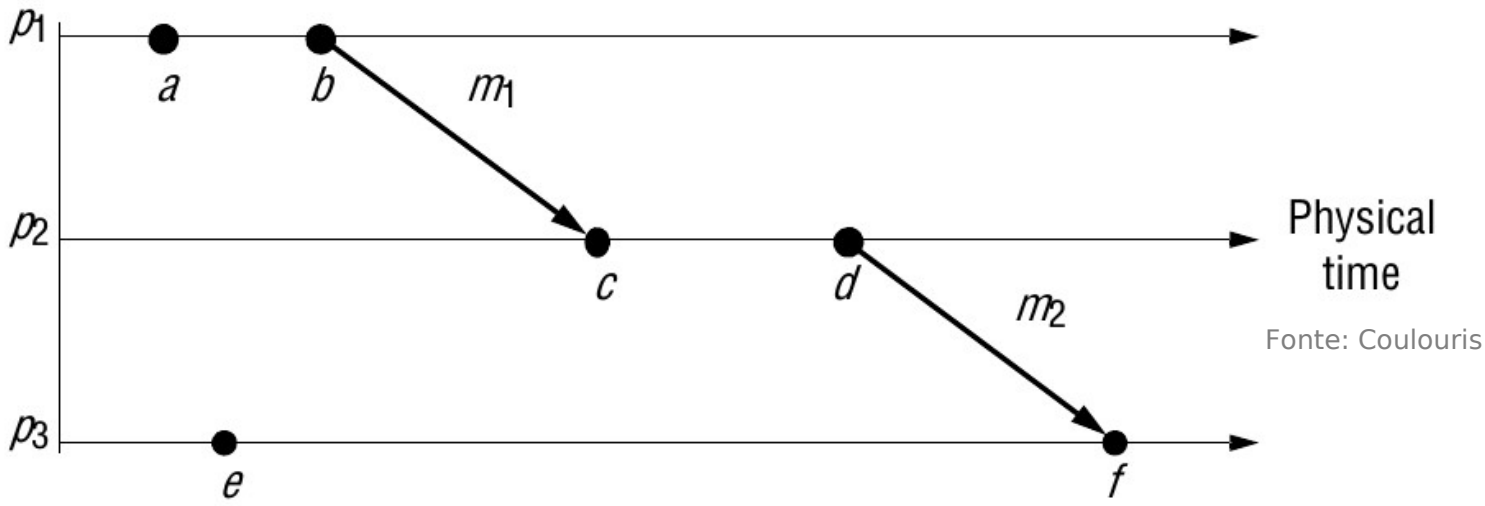
HB2: For any message m, $send(m) \rightarrow receive(m)$
– where $send(m)$ is the event of sending the message, and $receive(m)$ is the event of receiving it.

HB3: If e, e' and e'' are events such that $e \rightarrow e'$ and $e' \rightarrow e''$, then $e \rightarrow e''$.

Fonte: Coulouris

Tempo e Relógio Lógico

- Exemplo



Fonte: Coulouris

a → b, b → c, c → d, d → f
Os eventos **a** e **e** não são ordenados pela relação happened-before. Dizemos que são concorrentes: a ‖ e

Relógios Lógicos

- Lamport inventou um mecanismo denominado de **relógio lógico** que a ordenação *happened-before* por ser representada numericamente.
- Cada processo p_i mantém seu próprio relógio lógico L_i .
- $L_i(e)$ indicação de tempo do evento **e** em $\mathbf{p_i}$.
- L(e) indicação de tempo no evento que ocorreu.

Relógios Lógicos

- Funcionamento:

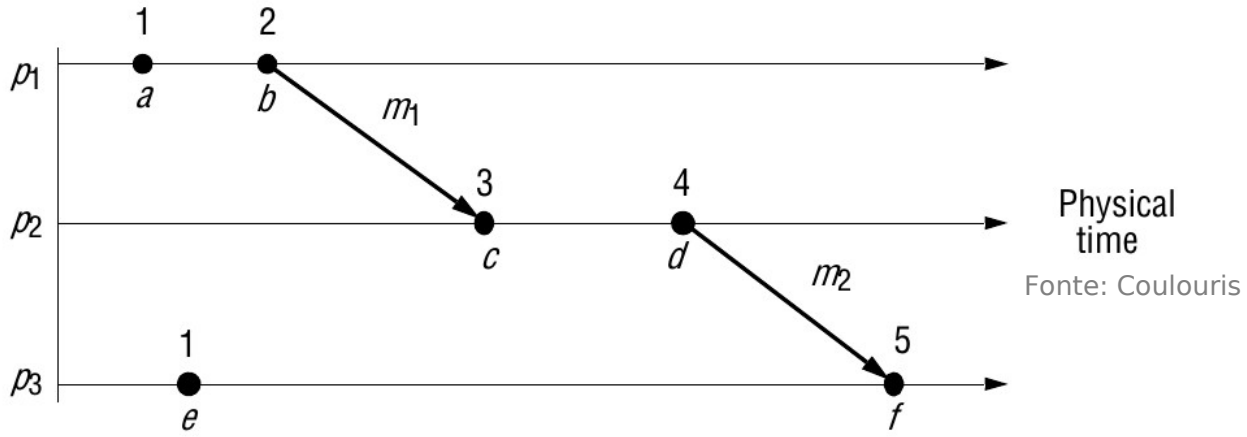
LC1: L_i is incremented before each event is issued at process p_i: $L_i := L_i + 1$.

LC2: (a) When a process p_i sends a message m, it piggybacks on m the value $t = L_i$.

(b) On receiving (m, t), a process p_j computes $L_j := max(L_j, t)$ and then applies LC1 before timestamping the event *receive*(m).

Fonte: Coulouris

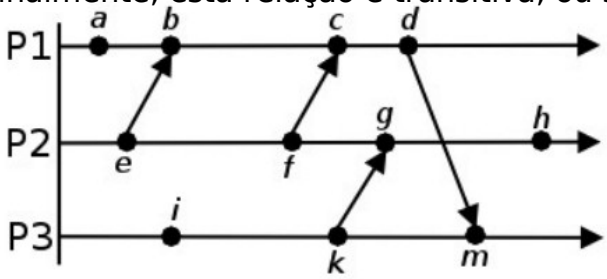
- Exemplo:



Fonte: Coulouris

Questão (PosComp 2015)

Em um Sistema Distribuído, a ordenação causal assegura que todos os processos reconheçam que um evento deve acontecer somente após a ocorrência de todos os eventos dos quais ele é dependente. A ordenação causal pode ser implementada pela relação acontece antes, representada como a → b. Esta relação determina que se a e b são eventos de um mesmo processo e a aconteceu antes de b, então a → b. Esta relação também estabelece que, se o evento a for o envio de uma mensagem e o evento b for o recebimento desta mesma mensagem, então a → b. Finalmente, esta relação é transitiva, ou seja, se a → b e b → c, então a → c.



Considere a existência de três processos: P1, P2 e P3, cada um residindo em um nó de processamento distinto. Estes processos estão representados no diagrama espaço-tempo abaixo. A direção vertical representa o espaço (ou seja, processos diferentes) e a direção horizontal representa o tempo. Uma seta em diagonal indica uma mensagem enviada de um processo para outro. As letras minúsculas representam os eventos.

De acordo com o diagrama apresentado, uma ordenação causal destes eventos, consistente com a relação acontece antes, seria:

(A) a b c d e f g h i k m

(B) a e i b f k m c g d h

(C) e a b i c d f k g m h

(D) e i a b k f c g d m h

(E) i a b e f k m c g h d

Relógios Vetoriais

- Cada processo mantém seu próprio relógio vetorial V_i.
- Com N processos o tamanho de V_i será N.
- Funcionamento:

VC1: Initially, $V_i[j] = 0$, for $i, j = 1, 2 \ldots, N$.

VC2: Just before p_i timestamps an event, it sets $V_i[i] := V_i[i] + 1$.

VC3: p_i includes the value $t = V_i$ in every message it sends.

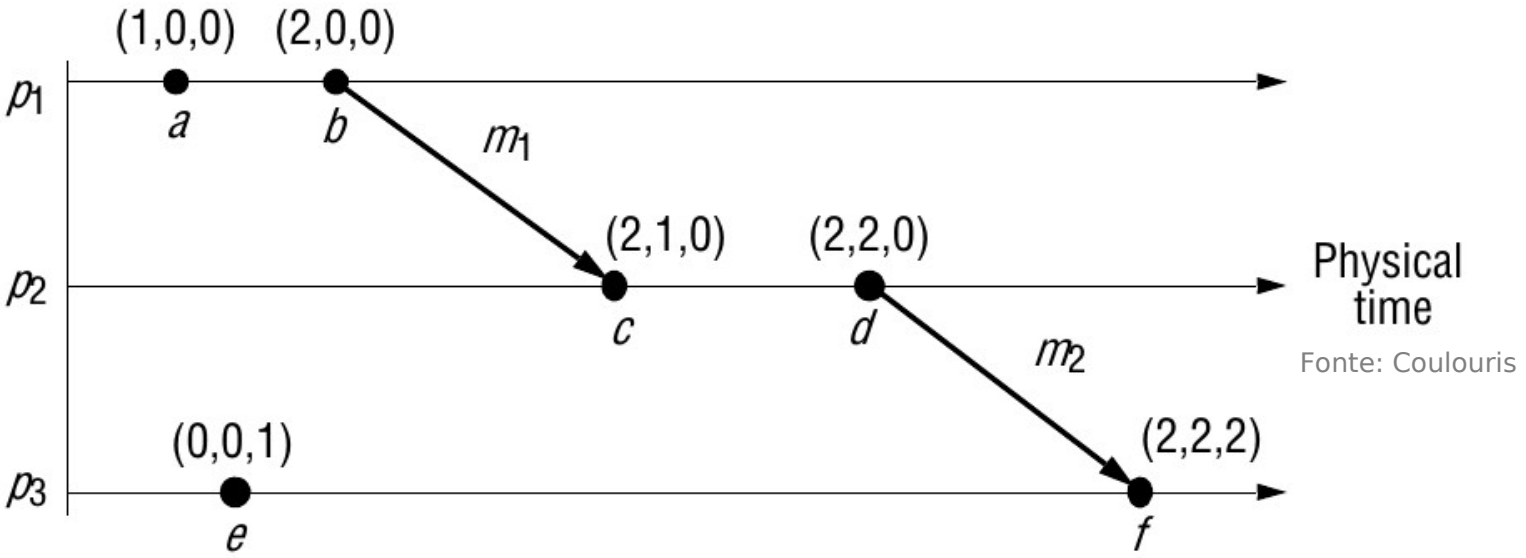
VC4: When p_i receives a timestamp t in a message, it sets $V_i[j] := max(V_i[j], t[j])$, for $j = 1, 2 \ldots, N$. Taking the component-wise maximum of two vector timestamps in this way is known as a *merge* operation.

Fonte: Coulouris

A ordenação é obtida comparando os vetores nas posições dos eventos.

Relógios Vetoriais

- Exemplo:

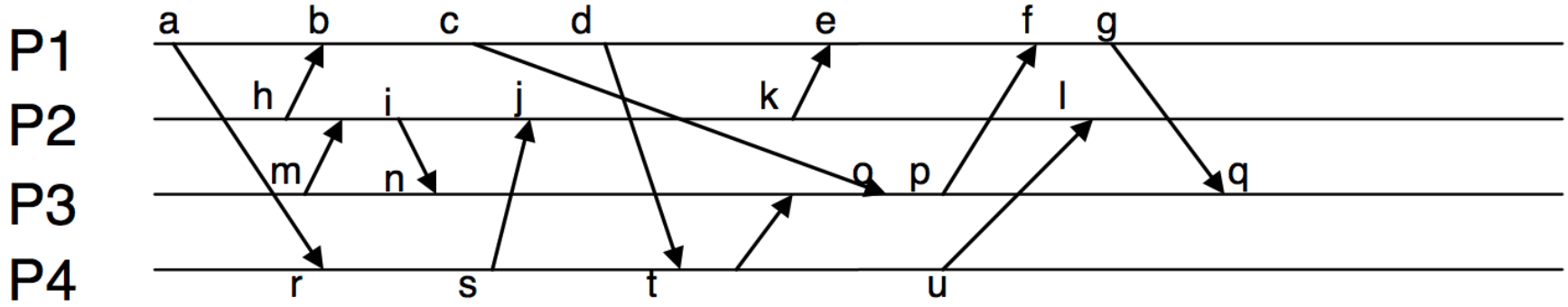


Fonte: Coulouris

```
V(a) < V(f)
Os eventos e e c são concorrentes, pois V(c) <= V(e)
(Falso) e V(e) <= V(c) (Falso)
```

Atividade

- Coloque os rótulos segundo os algoritmos de relógio lógico de Lamport e de relógio vetorial.



Referências

COULOURIS, George F; DOLLIMORE, Jean; KINDBERG, Tim; BLAIR, Gordon. **Sistemas distribuídos: conceitos e projeto**. 5. ed. Porto Alegre: Bookman, 2013.