Introdução ao Processamento Paralelo e Distribuído

Videoaula

Notas dos slides

APRESENTAÇÃO

O presente conjunto de slides pertence à coleção produzida para a disciplina *Introdução ao Processamento Paralelo e Distribuído* ofertada aos cursos de bacharelado em Ciência da Computação e em Engenharia da Computação pelo Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas.

Os slides disponibilizados complementam as videoaulas produzidas e tratam de pontos específicos da disciplina. Embora tenham sido produzidos para ser assistidos de forma independente, a sequência informada reflete o encadeamento dos assuntos no desenvolvimento do conteúdo programático previsto para a disciplina.

computação
Universidade Federal de Pelotas

universidade federal de pelotas
CDTec
centro de desenvolvimento tecnológico

Sistemas Distribuídos

Coordenação

Notas da videoaula

DESCRIÇÃO

Nesta videoaula são apresentados questões relacionadas à coordenação em Sistemas Distribuídos

OBJETIVOS

Nesta videoaula são apresentados aspectos ligados à coordenação em Sistemas Distribuídos. Os assuntos tratados são refletido nos modelos de interação entre componentes.

“Coming together is a beginning. Keeping together is progress. Working together is success.

Henry Ford

Avant Propos

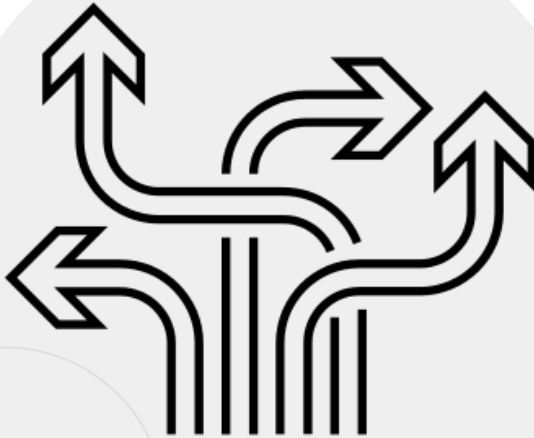
A **sincronização** foi tratada, na programação, como estratégia fundamental no controle de execução de processos (ou threads), permitindo o avanço da execução de um ou outro em função de um *estado*. Quando se fala em **sincronização de dados**, o objetivo é garantir que dois, ou mais, conjuntos de dados tenham as mesmas *informações*.

A **coordenação**, em um SD, objetiva gerenciar as interações entre os componentes e as dependências entre atividades de forma a manter as informações consistentes.

Avant Propos

Coordenação, estratégias para obter:

- Relógios globais
- Exclusão Mútua
- Algoritmos de Eleição



Relógio Global

O Problema

Os equipamentos possuem diferentes precisões nos seus relógios físicos e os atrasos, não controláveis nem previsíveis, nas comunicações não permite a existência de um relógio universal.

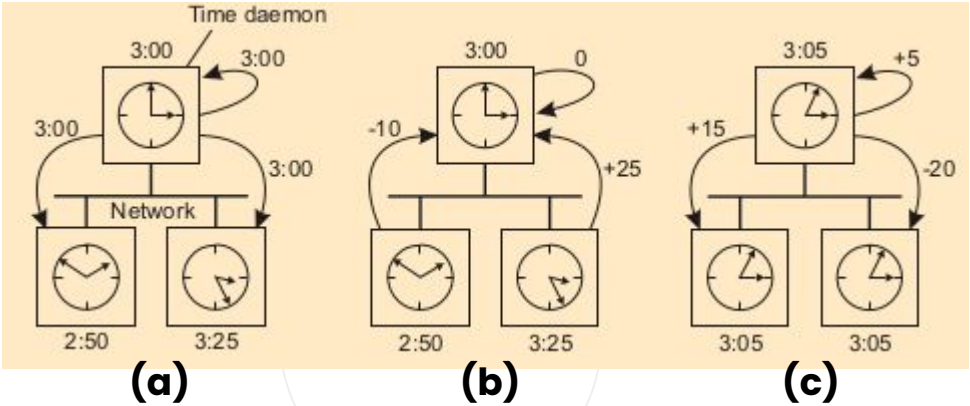
Questão de fundo:

- O que é mais importante conhecer? O tempo real em que eventos ocorrem ou a ordem de causalidade e efeito entre eventos?

Sincronizando relógios físicos

Algoritmo de Berkeley

- **(a)** O servidor é ativo, de tempos em tempos solicita a hora das outras máquinas, informando a sua própria hora
- **(b)** A resposta diz a diferença, para mais ou para menos, na máquina local
- **(c)** O servidor faz uma média dos tempos e envia uma mensagem a todos para correção.



Muito importante: Não é possível voltar o relógio de nenhuma máquina, de alguma forma deve ser ajustada as velocidades dos participantes.

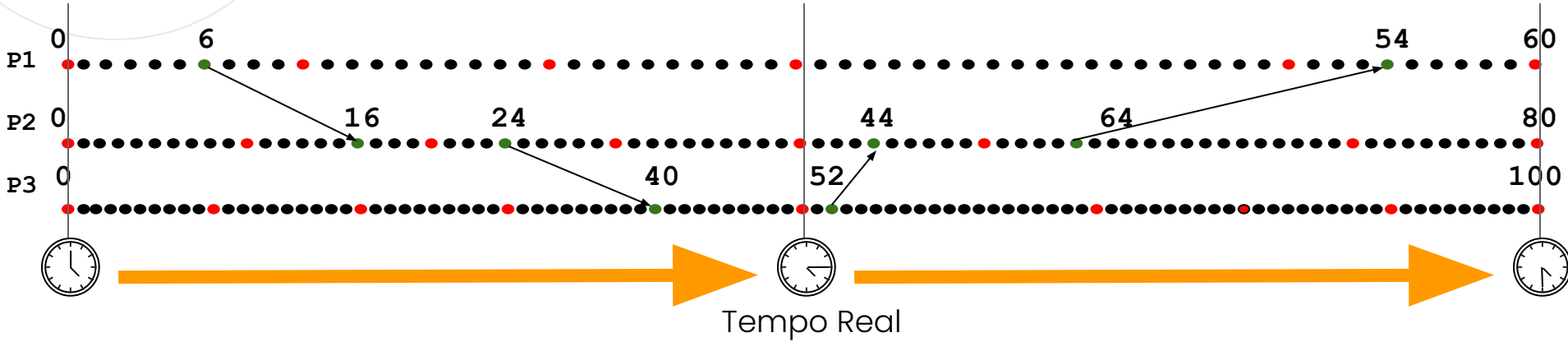
Sincronizando relógios lógicos

Questão: medir o tempo de forma que cada evento e possa receber um tempo $C(e)$ de forma que todos os processos envolvidos concordem com este tempo.

Dado dos eventos e_i e e_j, tal que $e_i \rightarrow e_j$, então $C(e_i) < C(e_j)$

Sincronizando relógios lógicos

Considere a seguinte situação simulando uma situação real.



- Evento contabilizado
- Evento de interação entre processos
- Evento cuja contagem é múltipla de 10

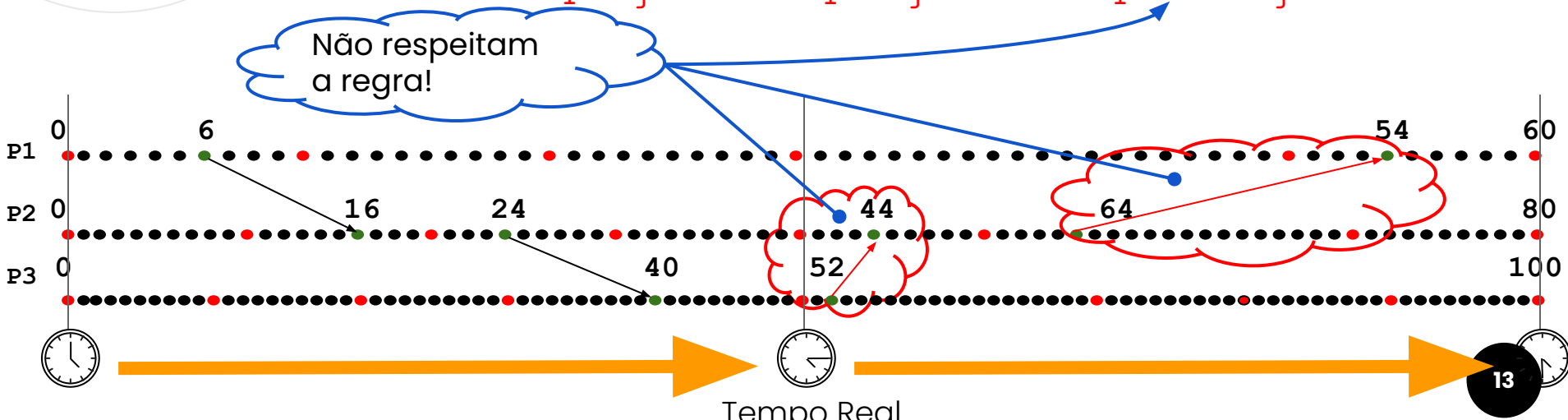
6, 16, 24... Contador do evento no referido processo

Observe: o tempo, em cada processo, é contado pelo número de eventos realizados (P3 é o mais rápido!)

Sincronizando relógios lógicos

Questão: medir o tempo de forma que cada evento e possa receber um tempo **C(e)** de forma que todos os processos envolvidos concordem com este tempo.

Dado dos eventos e_i e e_j, tal que $e_i \rightarrow e_j$, então $C(e_i) < C(e_j)$



Sincronizando relógios lógicos

Solução: Associar uma informação de tempo (timestamp) a cada evento, sendo este timestamp incremental e ajustado para que o timestamp de um evento resultante da interação entre 2 processos seja equivalente ao incremento do maior timestamp dos eventos que geraram a interação.

Assim, dado dos eventos e_i e e_j, tal que $e_i \rightarrow e_j$, então $C(e_i) < C(e_j)$ mesmo que executados em processos distintos

Algoritmo de Lamport

Seja:

- L_i o relógio lógico do processo p_i
- $L_i(e)$ o timestamp do evento e no processo p_i.

Regras:

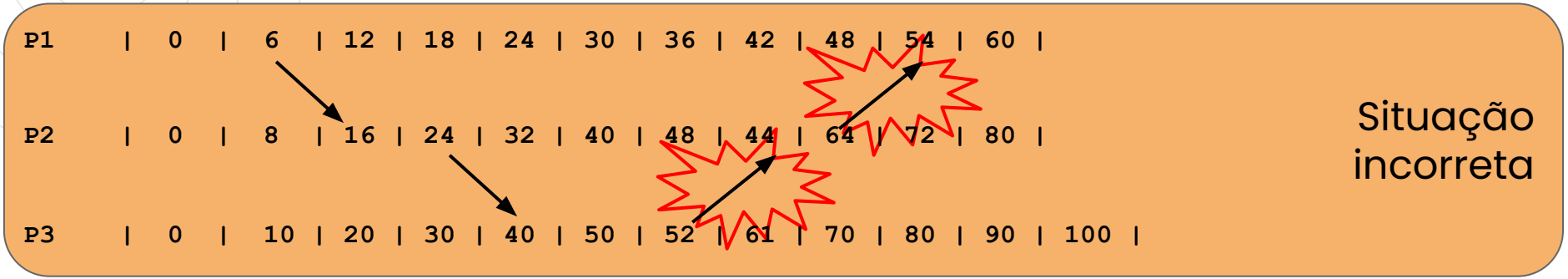
[R1]: L_i é incrementado antes da atribuição de um timestamp a qualquer evento em p_i.

$$L_i := L_i + 1$$

[R2]: Quando o processo p_i envia uma mensagem m, a ela é anexado o valor $t = L_i$ (já incrementado).

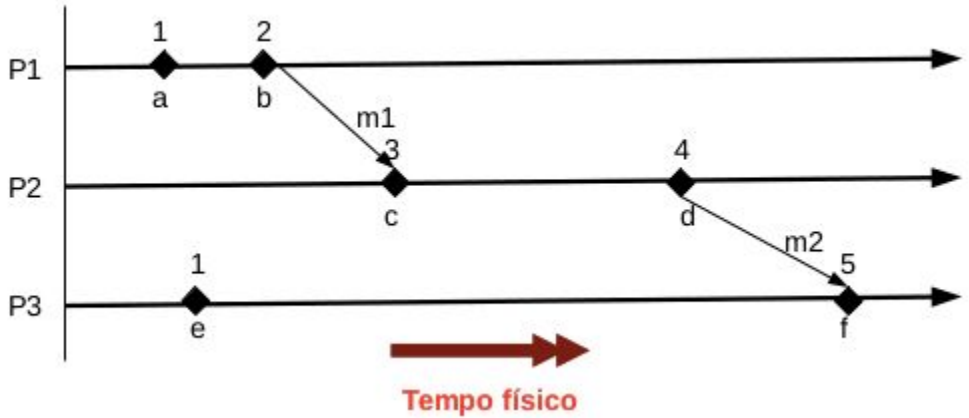
[R3]: Ao receber uma mensagem (m,t), um processo p_j calcula $L_j := \max(L_j, t)$, aplica a R1 e finalmente atribui o timestamp L_j ao evento de recepção da mensagem.

Algoritmo de Lamport



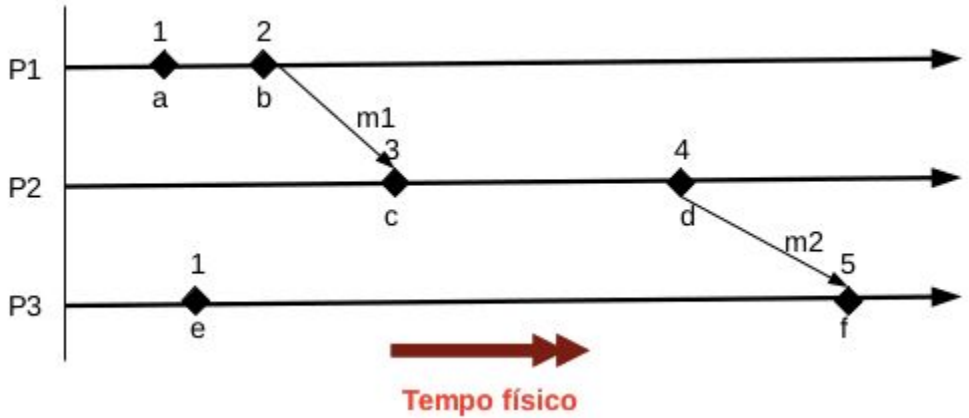
Relógios Lógicos

- $L_1(a) = L_2(e)$, embora os eventos a e e sejam distintos.
- É possível estabelecer uma ordem total, associando ao tempo a informação do processo que executou o evento.
- Sejam e e e' eventos ocorrendo nos processos p_i e p_j, com timestamps T_i e T_j, então, seus tempos lógicos globais são definidos por (T_i, i) e (T_j, j).



Relógios Lógicos

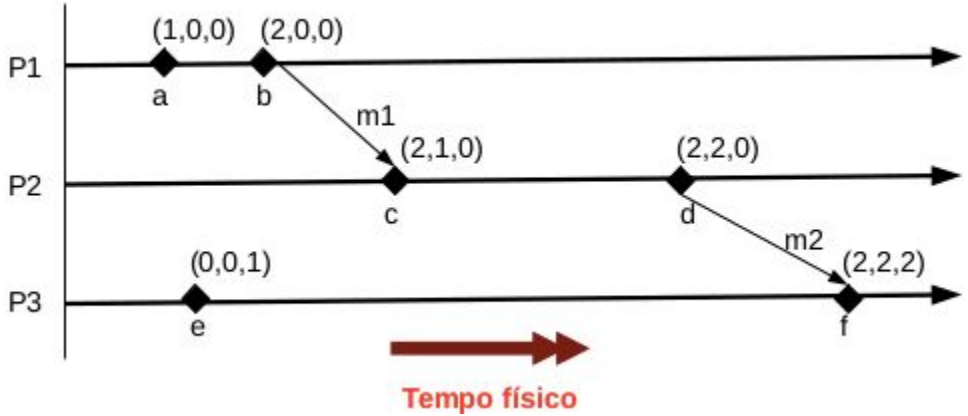
- Importante:
  `(T(a),i) < (T(b),i)` ⇒ `a → b`
  mas,
  `(T(a),i) < (T(e),j)` ⇏ `a → e`



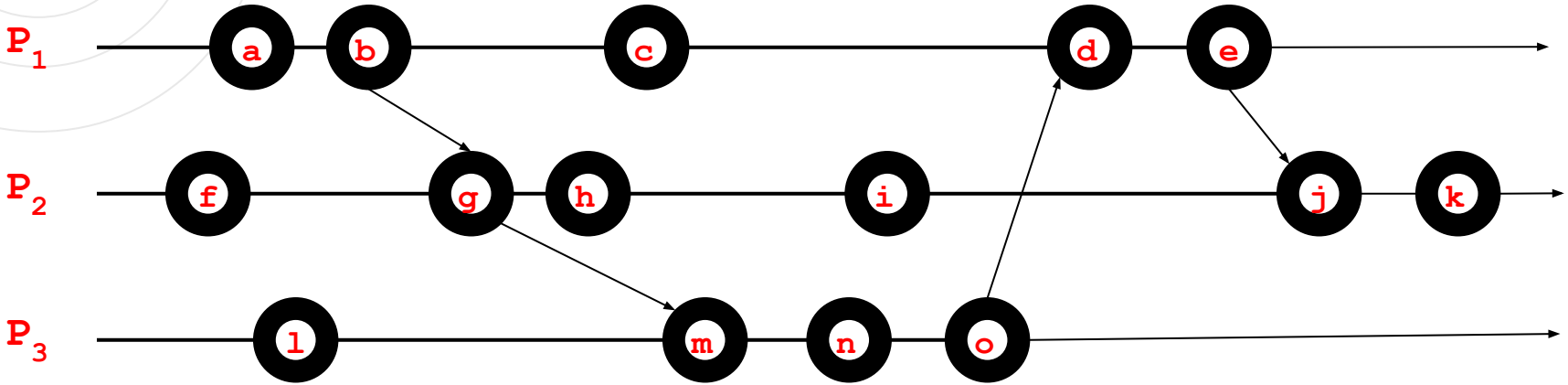
Relógios Vetoriais

- Cada processo `i` de um SD com ***n*** processos mantém um **vetor** V_i de ***n*** posições, cada posição correspondendo a um processo.
- O registro em cada posição de V_i informa ao último timestamp conhecido pelo processo `i` do processo correspondente àquela posição.
- Operação similar ao algoritmo de Lamport.
- Regras
  - **[R1]:** inicialmente $V_i[j] = 0$, para $j = 1, 2, \ldots n$.
  - **[R2]:** cada processo `i` incrementa seu timestamp no seu relógio vetorial, na posição $V_i[i]$.
  - **[R3]:** ao enviar uma mensagem, o processo `i` anexa seu relógio vetorial, ou seja $t = V_i$.
  - **[R4]:** ao receber uma mensagem, o processo `j` atualiza seu relógio vetorial considerando que $V_j[k] = \max(V_j[k], t[k])$, para $j = 1, 2, \ldots n$.

Relógios Vetoriais

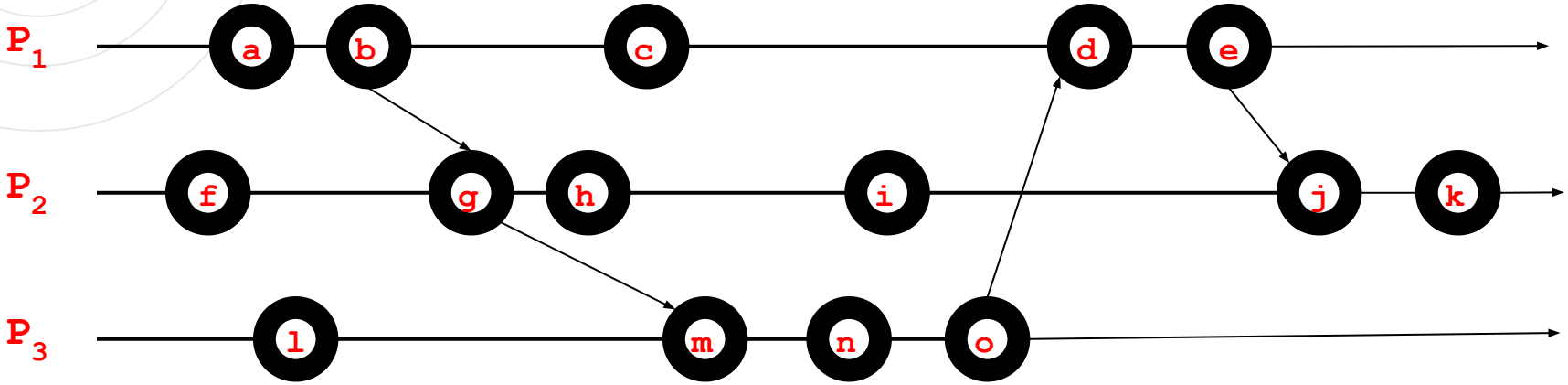


Situações



a → b	j → k	c // h
a → g	o → k	c // i
a → n	f → k	c // n
		c // f

Situações



a → b j → k c //
a → g o → k h
a → n f → k c //
i

Exclusão Mútua

Exclusão Mútua

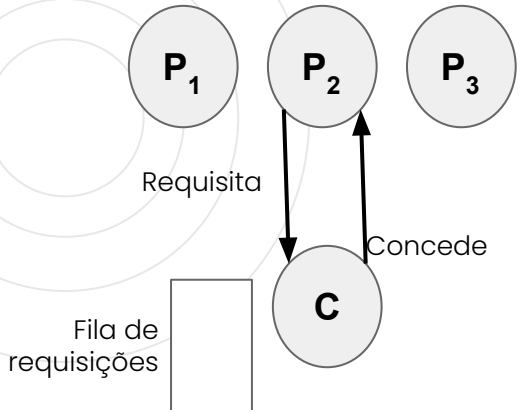
Problema:

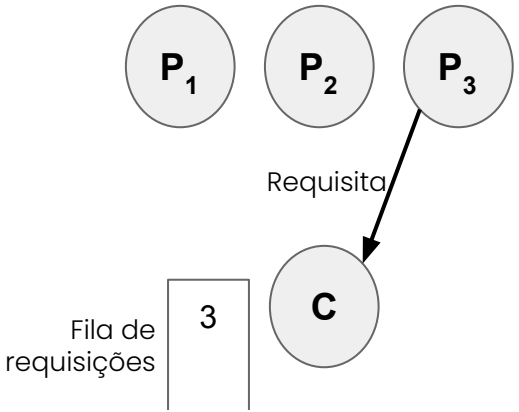
Os componentes de um SD necessitam acessar algum recurso compartilhado a uma seção crítica em regime de exclusão mútua. Este recurso pode ser representado por um dispositivo de hardware, como um dispositivo de E/S, ou mesmo um banco de dados.

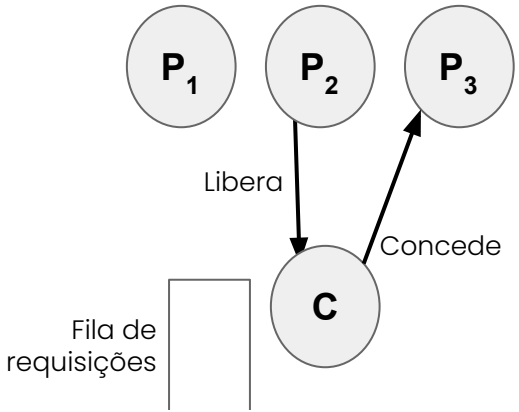
***As implementações podem ser baseadas em*:**

- **Permissão**: Um componente do SD possui autoridade para permitir que outros componentes tenham acesso à seção crítica.
- **Token**: Um token é passado entre os componente. O componente de posse do token tem direito a entrar na seção crítica

Solução: Centralizada e baseada em permissão



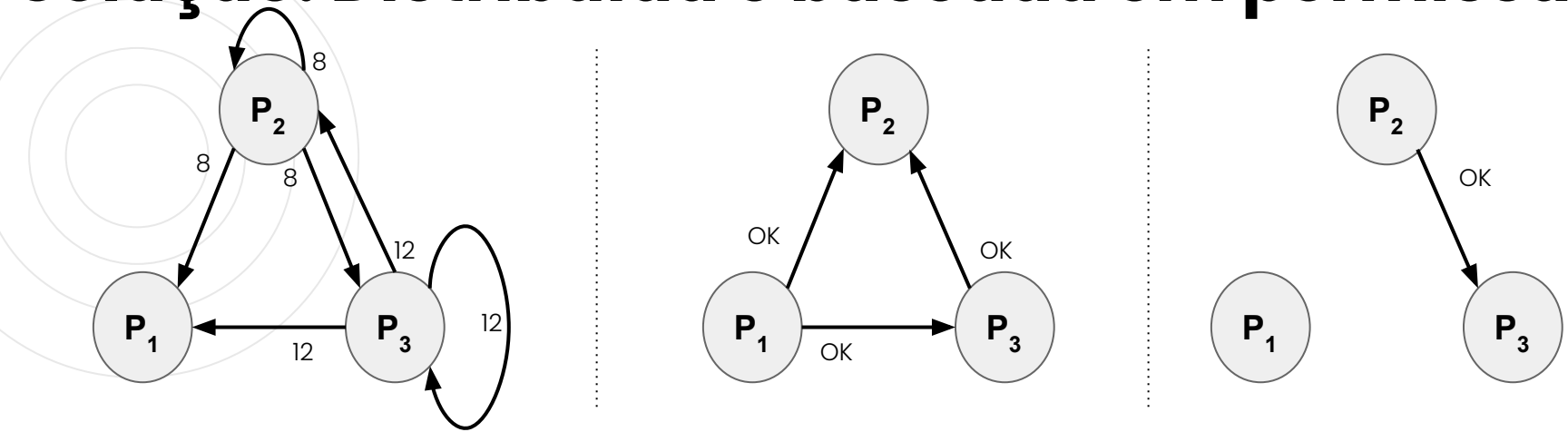




Um componente atua como coordenador, recebendo requisições e concedendo acesso à seção crítica.

De implementação simples, possui o inconveniente do processo que realizou uma requisição não saber se seu pedido foi recebido. Pode ocorrer sobrecarga no coordenador, se ocorreu falha no coordenador, a implementação básica não oferece solução.

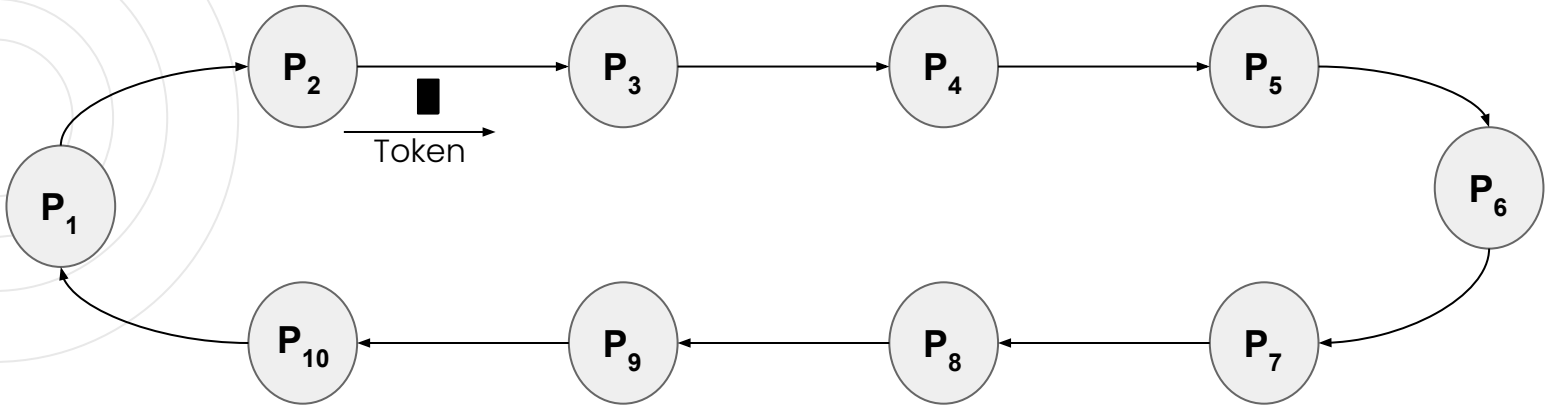
Solução: Distribuída e baseada em permissão



A decisão de quem vai receber a permissão é tomada de forma coletiva.

Todos os processos interessados (no exemplo, P1 e P3) mandam seus respectivos timestamps. Inclusive para si mesmo. Então, aquele localmente, todos os processos envolvidos decidem que quem vai ganhar o recurso é aquele com o timestamp mais baixo. A ordem de solicitação é mantida por quem ganhou o recurso para depois poder repassar o recurso.

Solução: Distribuída e baseada em token



O princípio é o de organizar os componentes em uma organização lógica, tipicamente um anel, e circular entre eles um token.

De implementação simples, possui também a vantagem de oferecer uma previsibilidade sobre o tempo que falta para poder obter o token. É possível elaborar medidas de tolerância a falhas. Os inconvenientes estão ligados à escalabilidade da solução e a alta dependência da rede.

Eleição

Eleição de Líder

Muitos algoritmos necessitam que um participante atue como coordenador. Em alguns casos, este líder é determinado manualmente e não há como alterá-lo, por exemplo, um servidor de arquivos. No entanto, quando for possível, é indicado poder determinar, dinamicamente, um líder.

Eleição de Líder

- Redes de Difusão de Rádio (Radio Broadcast Network)
  - Evitar sobreposição de sinais, determinando quem pode enviar
- Notificação Explícita de Congestionamento
  - Subdividir redes, identificando o líder dos subconjuntos criados para balancear a rede
- O sistema de arquivos da Amazon
  - Leitura interessante: https://aws.amazon.com/pt/builders-library/leader-election-in-distributed-systems
- Determinação de réplica principal
  - Tolerância a falhas

Algoritmo do Valentão (*bully algorithm*)

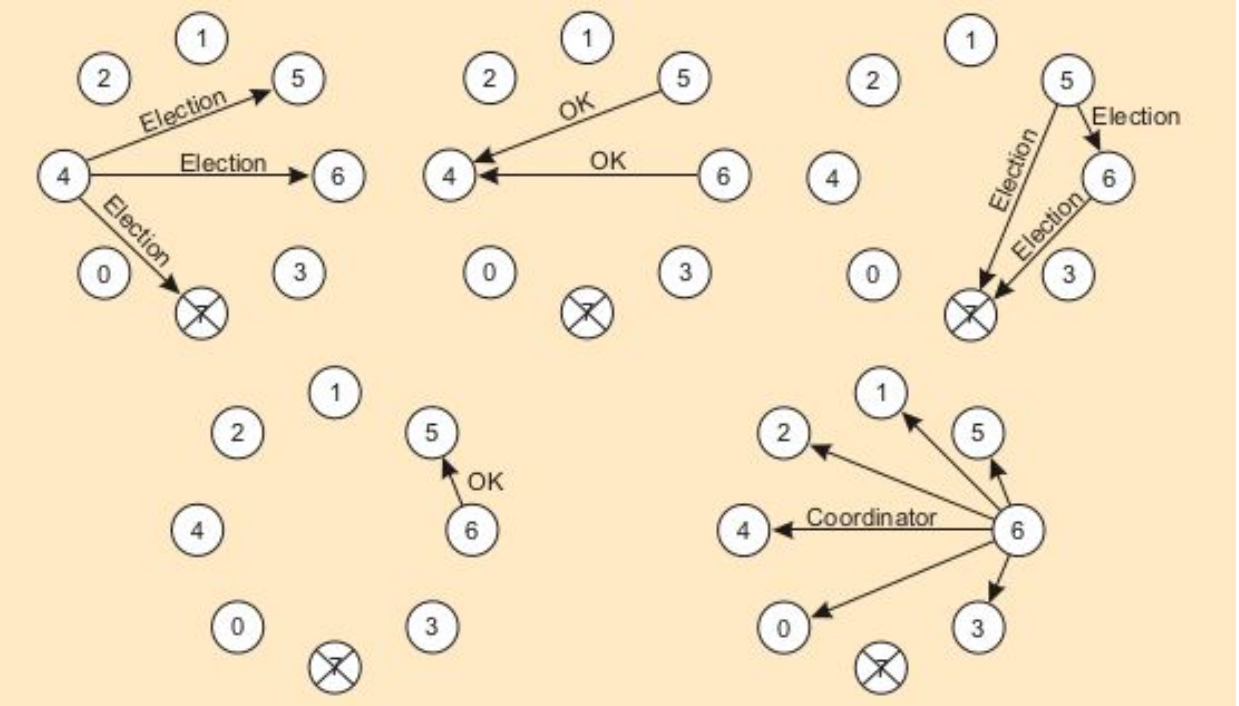
N processos envolvidos $\{P_0, P_1, \ldots P_{N-1}\}$, onde $id(P_k) = k$.

A eleição é iniciada quando o processo P_k detecta que o coordenador não está respondendo. O processo P_k, então inicia a eleição e o procedimento é o seguinte:

1. P_k Envia uma mensagem **ELEIÇÃO** a todos os processos cujo identificador é superior a k, ou seja P_{k+1}, P_{k+2}, ... P_{N-1}
2. Se não houver resposta, P_k venceu a eleição é é o novo coordenador.
3. Se houve resposta, P_k perdeu a eleição.
4. Os processos com identificador superior a k dão prosseguimento ao processo até o novo líder ser eleito.

Algoritmo do Valentão (*bully algorithm*)

1. P_k Envia uma mensagem **ELEIÇÃO** a todos os processos cujo identificador é superior a **k**, ou seja P_{k+1}, P_{k+2}, ... P_{N-1}
2. Se não houver resposta, P_k venceu a eleição é é o novo coordenador.
3. Se houve resposta, P_k perdeu a eleição.
4. Os processos com identificador superior a **k** dão prosseguimento ao processo até o novo líder ser eleito.



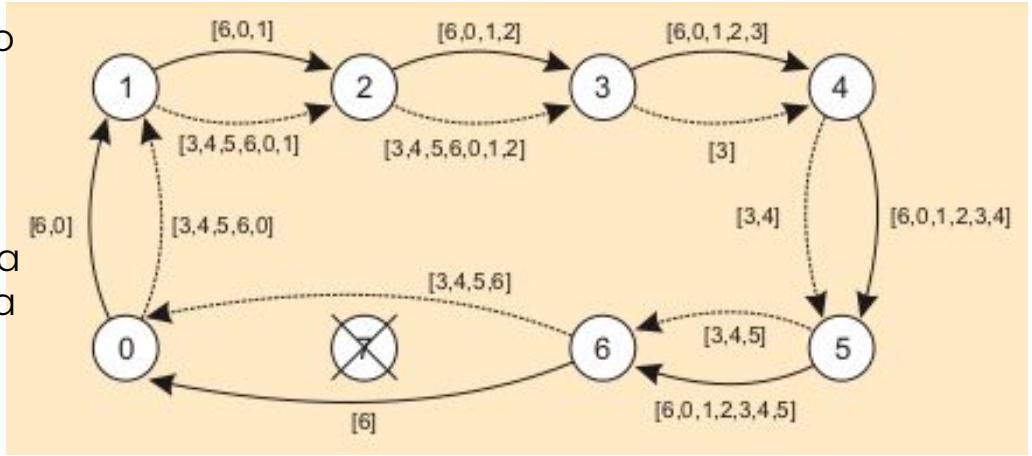
Eleição em Anel

Os processos encontram-se organizados em um anel (lógico), possuindo identificadores únicos. O identificador de um processo corresponde também à sua prioridade para assumir a liderança. O processo com maior identificador é eleito o líder, independente de quem iniciar a eleição..

1. Qualquer processo pode iniciar o processo de eleição, enviando ao próximo vizinho ativo uma mensagem de eleição contendo seu próprio identificador
2. Ao retornar ao processo inicial, todos processos do anel tomaram conhecimento de que uma eleição estava ocorrendo e a mensagem recebida anota todos os processos ativos.
3. O processo inicial então envia uma nova mensagem, informando que a eleição terminou e o identificador do processo eleito (o de id mais alto)

Eleição em Anel

1. Qualquer processo pode iniciar o processo de eleição, enviando ao próximo vizinho ativo uma mensagem de eleição contendo seu próprio identificador
2. Ao retornar ao processo inicial, todos processos do anel tomaram conhecimento de que uma eleição estava ocorrendo e a mensagem recebida anota todos os processos ativos.
3. O processo inicial então envia uma nova mensagem, informando que a eleição terminou e o identificador do processo eleito (o de id mais alto)



Duas eleições disparadas, uma pelo P_3, outra pelo P_6.

Obrigado!