



Sincronização de Tempo e Coordenação

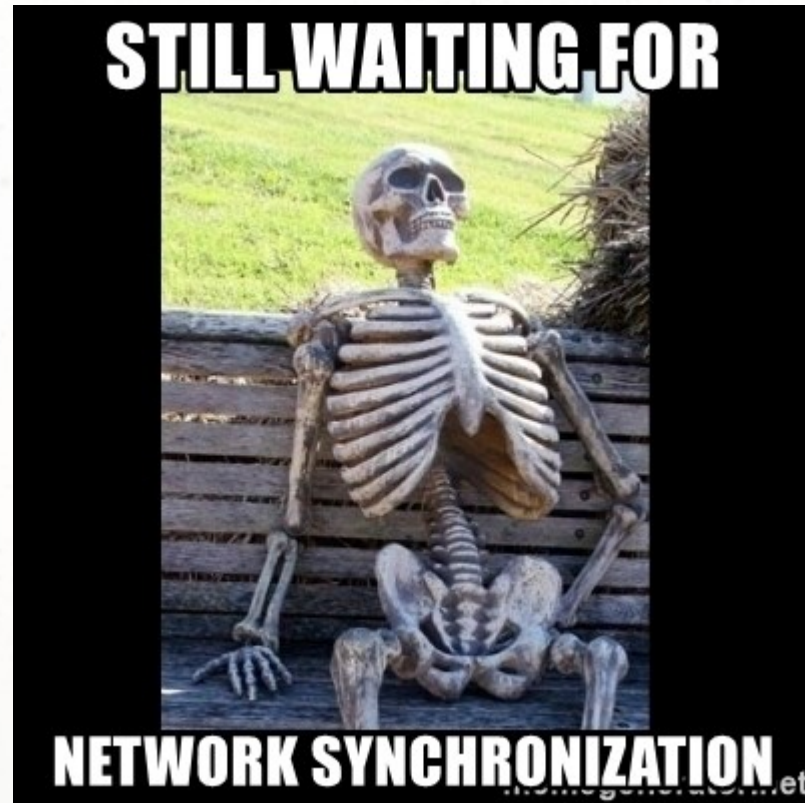
Sistemas Distribuídos e Mobile

Prof. Gustavo Torres Custodio
gustavo.custodioi@anhembi.br

Agenda

- Sincronização
 - Conceito
 - Algoritmos
- Coordenação e Acordo
 - Conceito
 - Algoritmos

Sincronização em Sistemas Distribuídos



ã Sincronização

- Como as regiões críticas são implementadas em um SD?
- Como os recursos são alocados aos processos?



SYNCHRONIZED
SWIMMING

It's weirder than you originally thought.

ã Sincronização

- Em SO, os problemas de exclusão mútua e região crítica são solucionados através de semáforos ou monitores.
 - Tais métodos utilizam memória compartilhada para implementar a solução, portanto, impossível de ser feito em um SD.
- Pergunta: como promover a sincronização em um ambiente distribuído?

• Sincronização através do CLOCK

- Os Sistemas Distribuídos utilizam algoritmos distribuídos na implementação de serviços e aplicações.
- Geralmente não é desejável ter todas as informações sobre o sistema em um único lugar.

Propriedades dos Sistemas Distribuídos

- As informações relevantes são espalhadas pelas múltiplas máquinas;
- Os processos tomam as decisões baseadas somente em informações locais;
- Um ponto de falha que paralise todo o sistema deve ser evitado;
- Não existe relógio comum ou um tempo global.

• Sincronização em Sistemas Distribuídos

- Sincronização de “Clock”
 - É mais complicado que em Sistemas Centralizados porque usam algoritmos distribuídos;
 - Geralmente não é possível, ou desejável, ter todas as informações sobre o sistema em um único lugar (desta forma a decisão poderia ser feita da mesma forma que nos Sistemas Centralizados).

- **Clock Físico**
 - os clocks não podem divergir de mais que uma certa quantidade.
- **Clock Lógico**
 - consistência interna é o que importa.



- Em alguns sistemas (tempo-real por exemplo) um clock real é importante. Nestes sistemas necessitamos de clocks físicos externos.
 - *International Atomic Time* (TAI) - baseado nas transições do Cesium 133;
 - *Coordinated Universal Time* (UTC) - em fase com o movimento do sol.

- Lamport (1978) - *“Time, Clocks, and the Ordering of Events in a Distributed System”*
 - Mostra que a sincronização dos relógios é possível e apresenta o algoritmo para se conseguir isto.
- Outro Artigo do Lamport
 - Mostra que a sincronização dos *clocks* não precisa ser absoluta:
 - Se dois processos não interagem, não é necessário que seus *clocks* sejam sincronizados (não é observável);
 - Usualmente o que importa não é que todos os processos concordem com o exato tempo em que os eventos aconteceram, mas que concordem na ordem em que os eventos ocorreram.

Algoritmo de Lamport

- Relação “acontece-antes”
 $a \dashrightarrow b$ (a acontece antes de b)
- Significa que todos os processos concordam que primeiro o evento a ocorreu e depois disto, o evento b ocorreu.

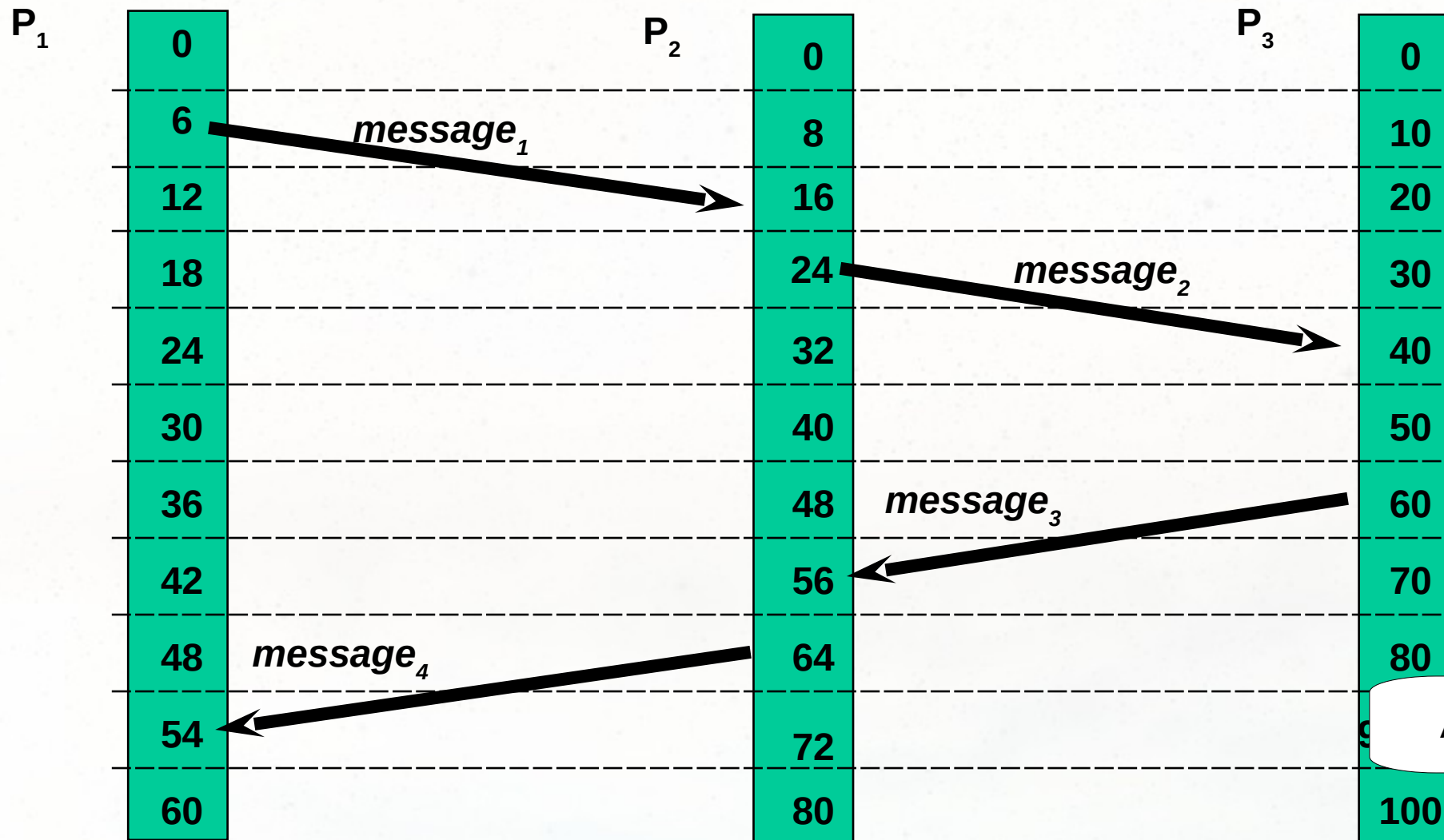
Algoritmo de Lamport

- Esta relação pode ser observada em duas situações:
 - 1- Se a e b são eventos no mesmo processo, e a ocorre antes de b , então $a \rightarrow b$ é verdadeiro
 - 2- Se a é o evento de uma mensagem sendo enviada por um processo, e b é o evento da mensagem sendo recebida por outro processo, então $a \rightarrow b$ é também verdadeiro. Uma mensagem não pode ser recebida antes de ser enviada, ou no mesmo tempo em que foi enviada.

Algoritmo de Lamport

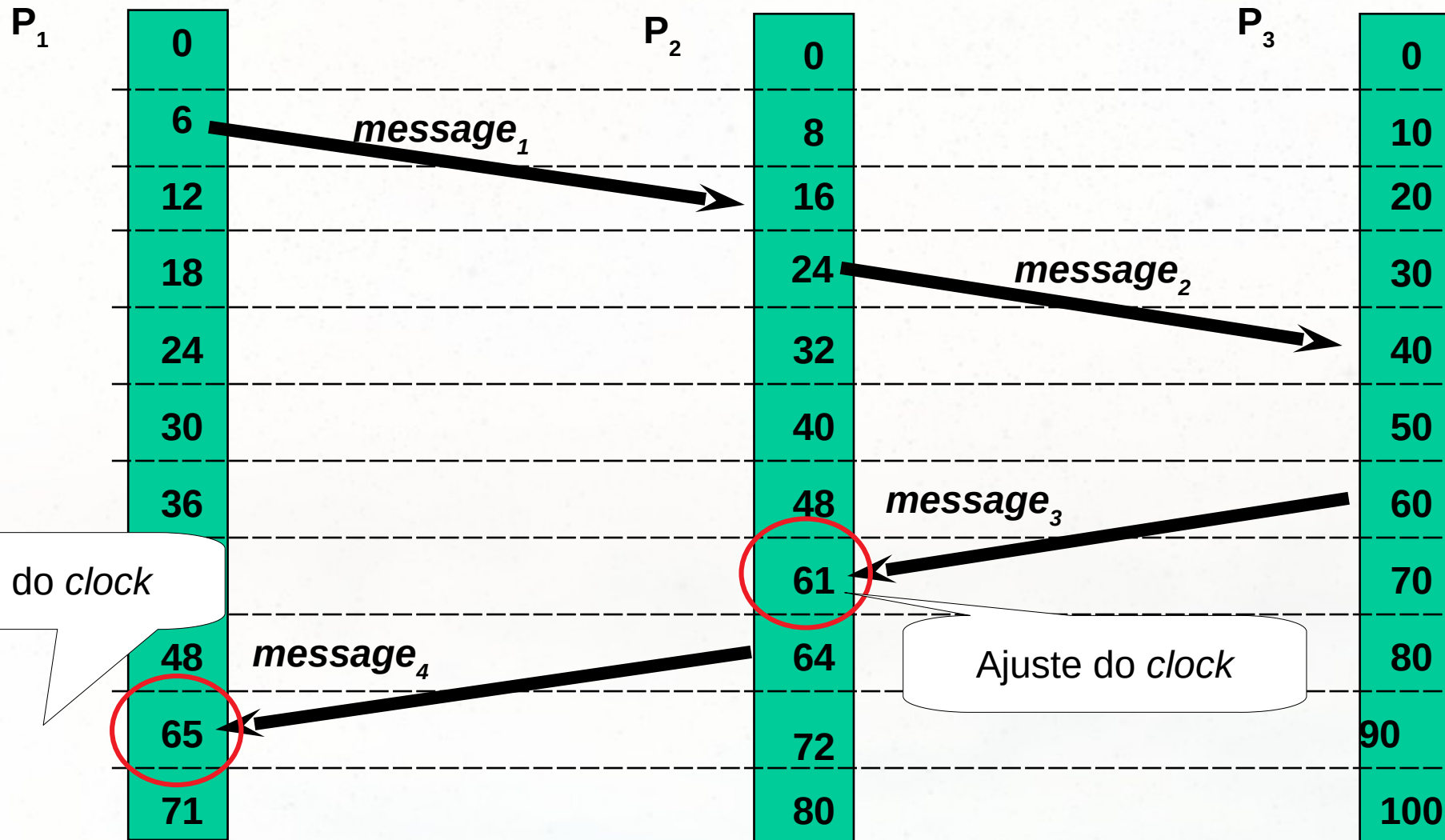
- “acontece-antes” é uma relação transitiva:
 - Se $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$
- Se dois eventos x e y , acontecem em diferentes processos que não trocam mensagens, então
 - $x \rightarrow y$ não é verdadeiro, nem $y \rightarrow x$ é verdadeiro.
 - Estes eventos são ditos concorrentes, o que significa que nada pode ser dito sobre quando eles aconteceram.

Visão de Três Processos Quaisquer



Algo Estranho?

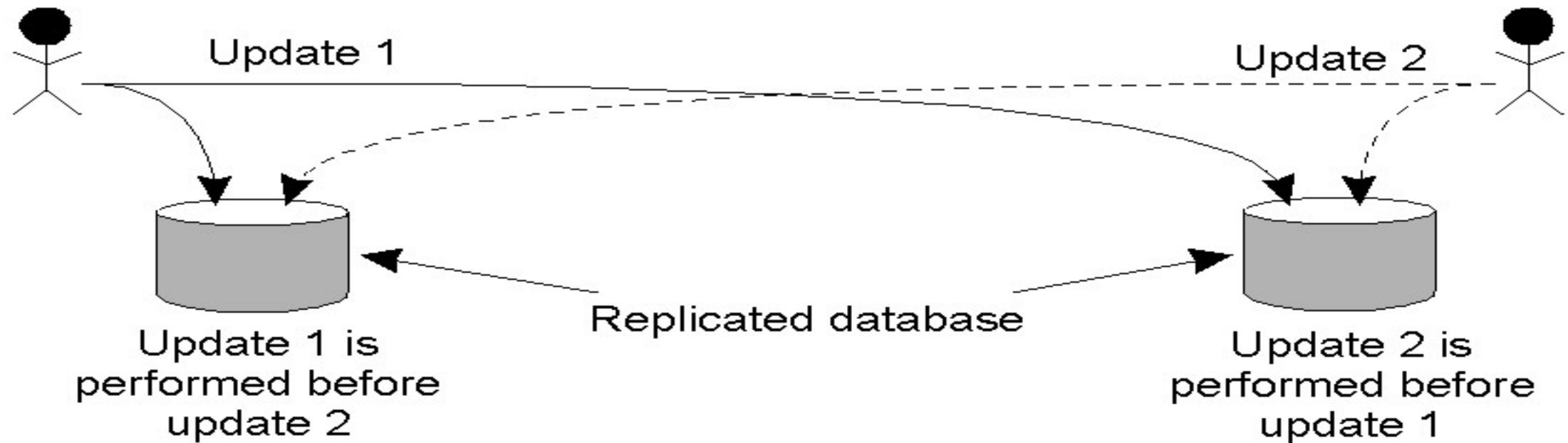
Algoritmo de Lamport



Algoritmo de Lamport

- Desde que a mensagem saiu no tempo 60, ele precisa chegar no mínimo no tempo 61.
- Cada mensagem carrega o tempo de envio, de acordo com o *clock* do processo que enviou a mensagem.
- Quando a mensagem chega e o relógio do receptor mostra um valor anterior ao da mensagem, o receptor avança seu relógio para o tempo de envio da mensagem mais um.
- Para que o algoritmo atenda os requisitos para um tempo global, necessita ainda que entre dois eventos, o relógio avance no mínimo de um.

Lamport Timestamps



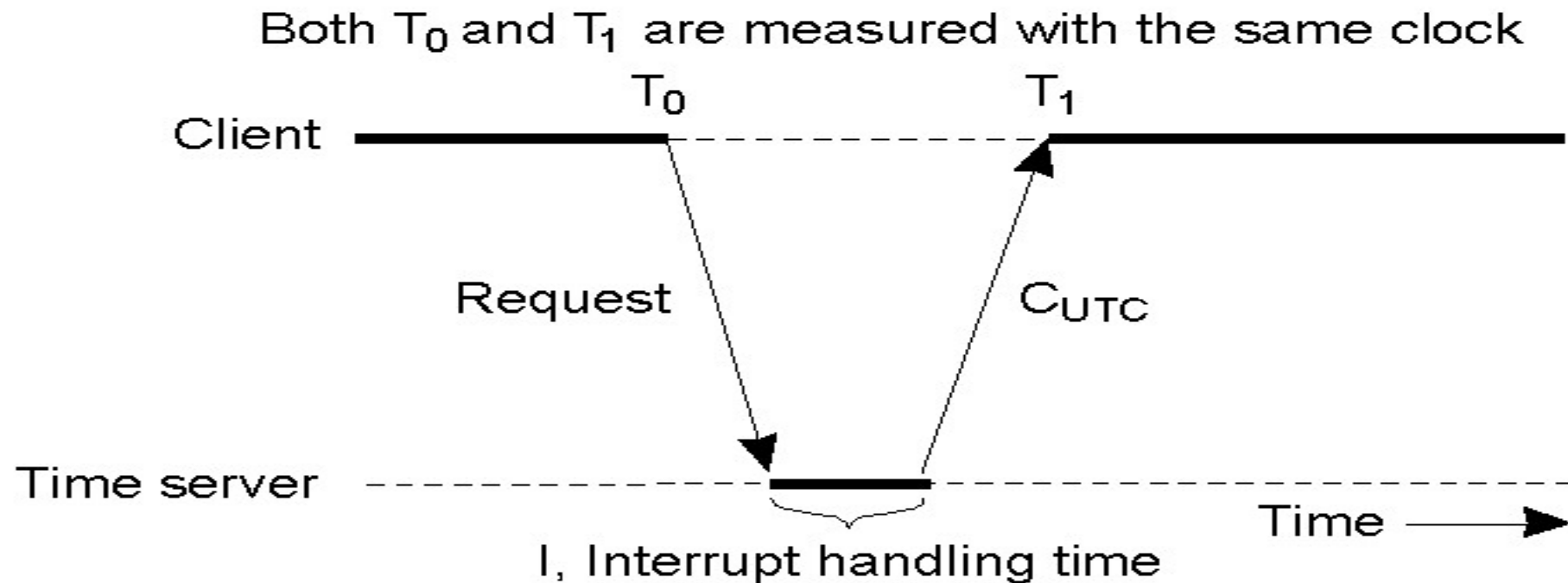
- Três processos, cada um deles com o seu próprio *clock*. Os *clocks* executam com diferentes tempos.
- Algoritmo de Lamport corrige os *clocks*.

Algoritmos de Sincronização de Relógios

- Algoritmo de Cristian (1989)
 - Para sistemas com uma máquina com receptor de UTC.
 - A máquina que tem o receptor UTC é chamada Servidor de Tempo.
 - Periodicamente, cada máquina envia uma mensagem para o *Servidor de Tempo* perguntando pelo tempo corrente (atual).
 - Esta máquina responde o mais rápido possível com uma mensagem contendo o tempo corrente C_{utc} .

Algoritmo do Christian

- Retirando o tempo atual de um servidor

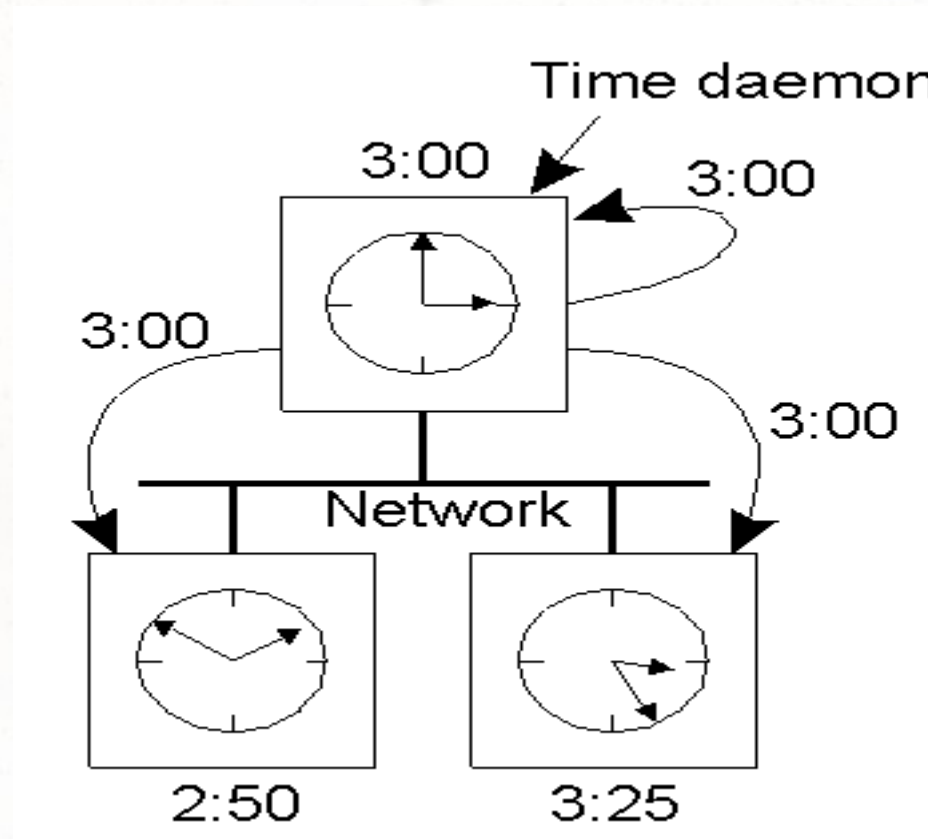


Algoritmo de Berkeley

- Nenhuma máquina tem um receptor de Tempo Universal Coordenado (UTC).
- Como funciona:
 - Servidor de Tempo é ativo, e requer, periodicamente de cada máquina, o tempo do seu relógio.
 - Servidor de Tempo calcula a média dos tempos e diz para cada máquina como ajustar seu relógio.

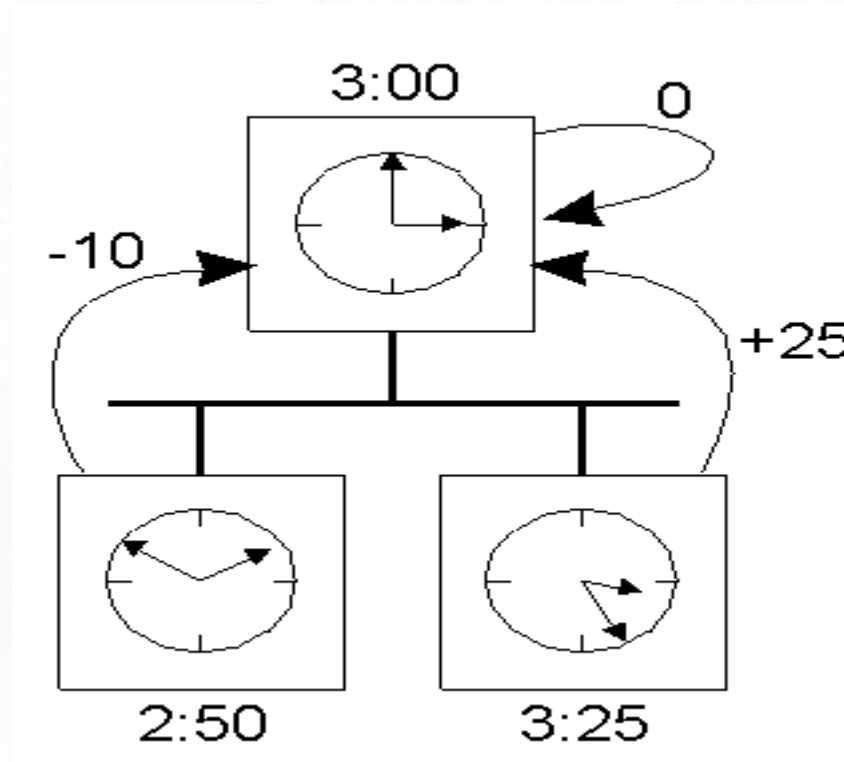
Algoritmo de Berkeley

O processo do tempo pergunta a todas as outras máquinas o seu clock;



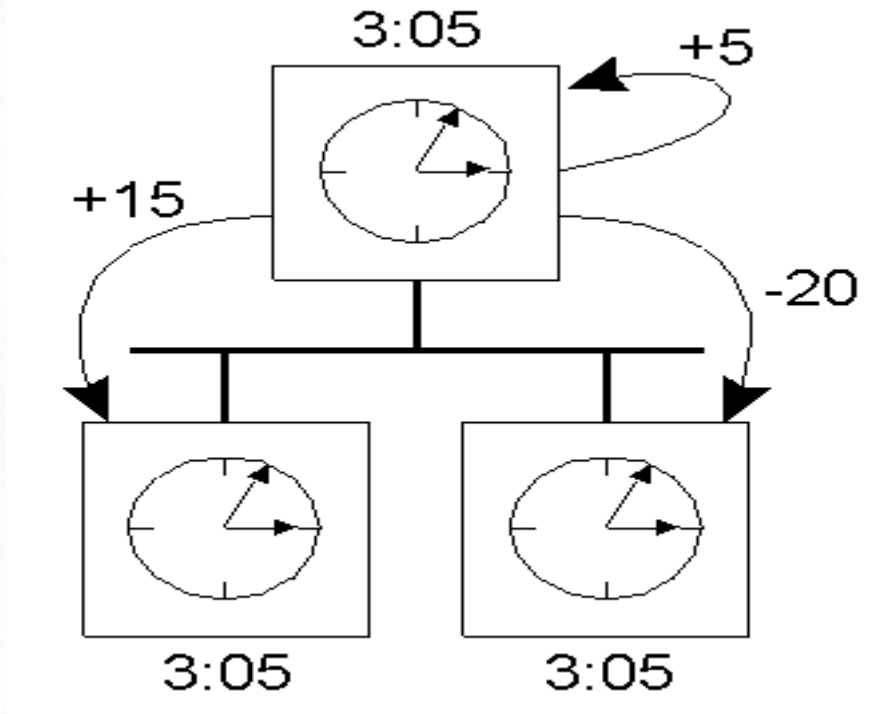
Algoritmo de Berkeley

As máquinas respondem;



Algoritmo de Berkeley

O processo do tempo envia para as máquinas como elas devem se ajustar.



Coordenação e Acordo



Coordenação e Acordo

- Muitos algoritmos distribuídos requerem um processo como coordenador.
 - Exemplo: Algoritmos de Exclusão Mútua
- Em geral não importa qual seja o processo coordenador, mas um deles inequivocamente tem que exercer esta função.

- **Objetivos:**

- Conjunto de processos que coordene as ações ou concorde com um ou mais valores.

- **Exemplo:**

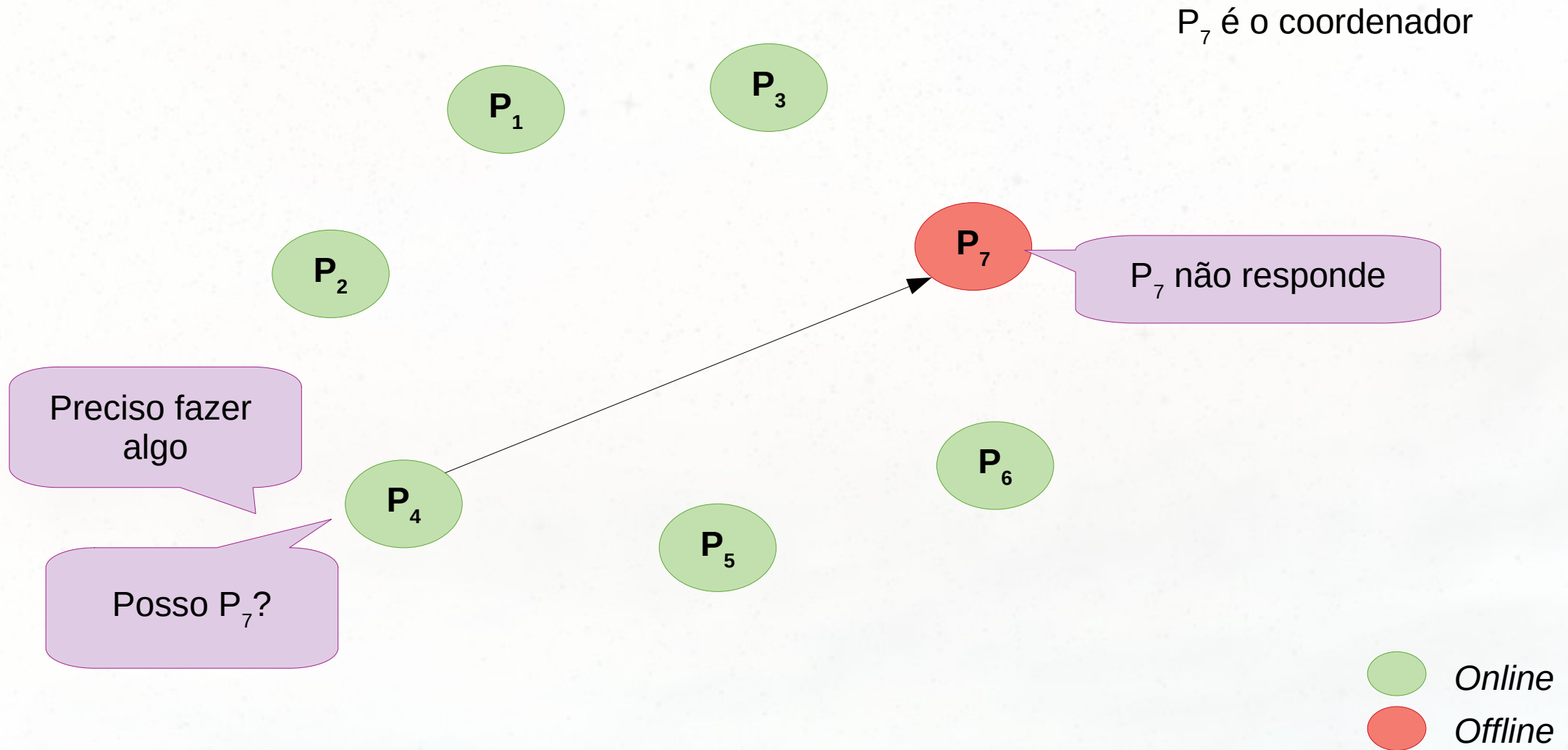
- Nave espacial
 - Todos os computadores devem concordar com as condições que são apresentadas pela nave.
 - Abortar missão – Alguma falha ocorreu.
 - Tudo OK – Prosseguir com a missão.

Algoritmos para Coordenação

- Algoritmo Bully
 - “Força” a ser o coordenador
- Algoritmo de Anel

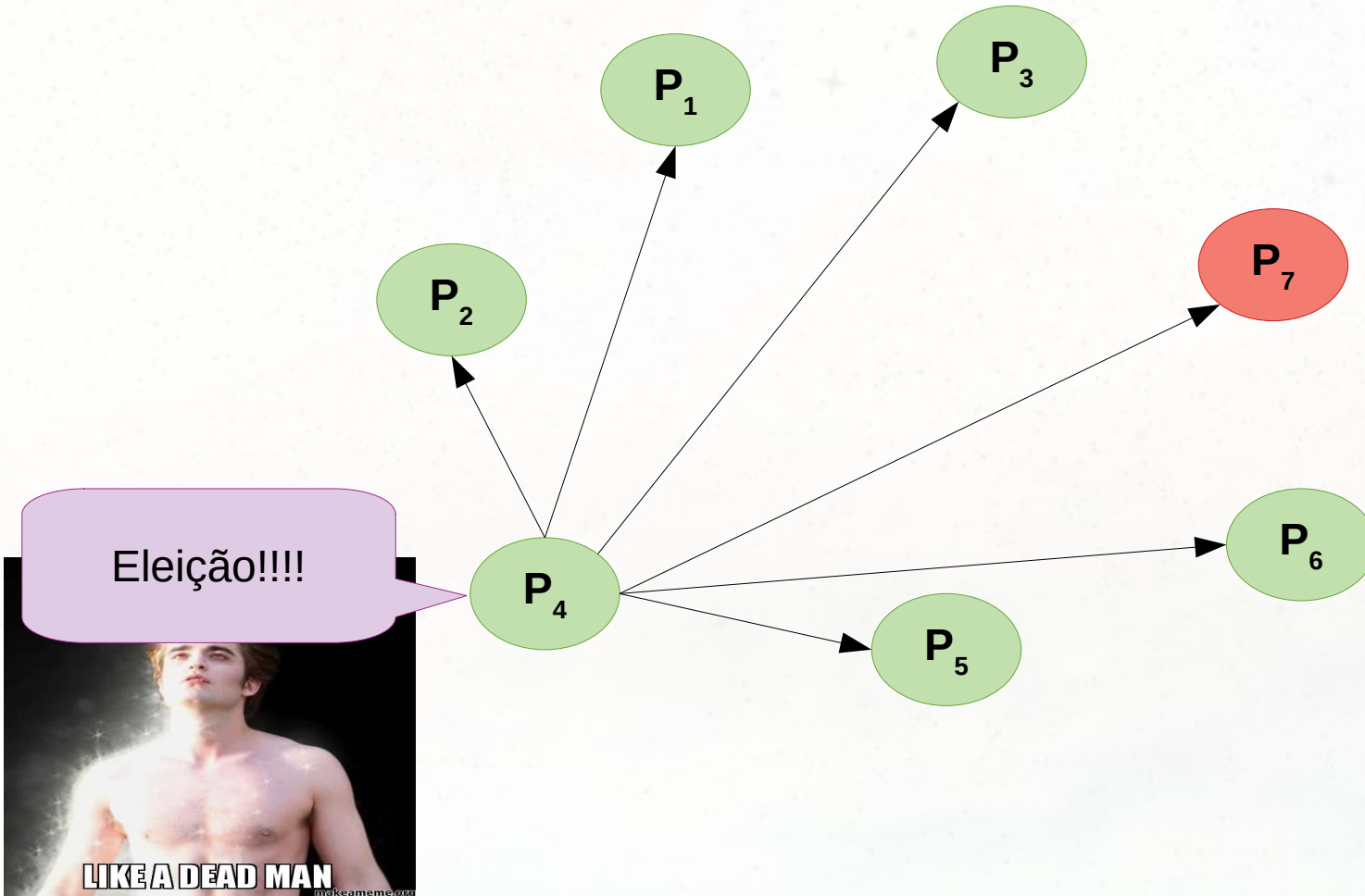


Algoritmo "Bully"



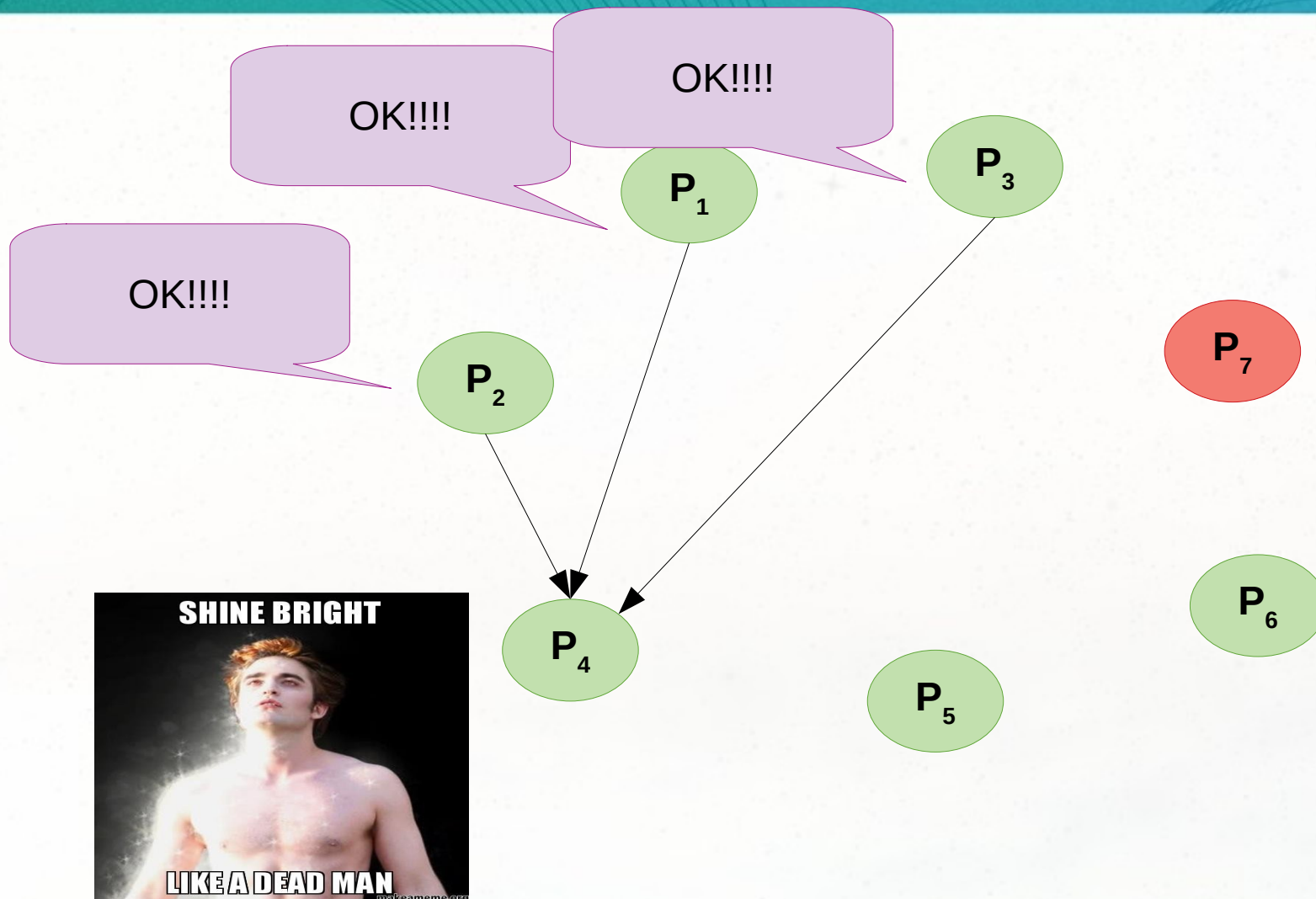
Algoritmo “Bully”

P_7 é o coordenador



● Online
● Offline

Algoritmo "Bully"

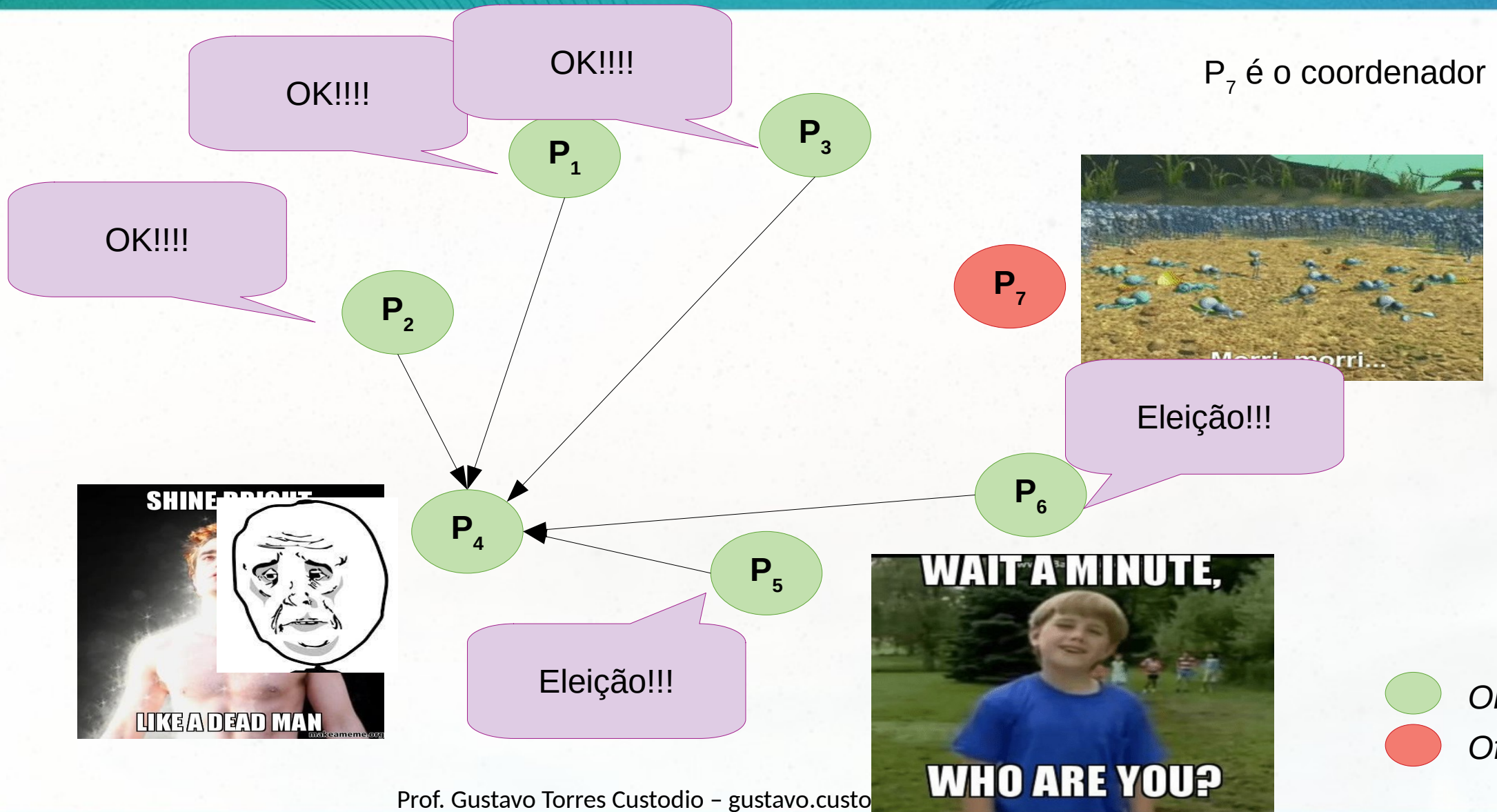


P_7 é o coordenador

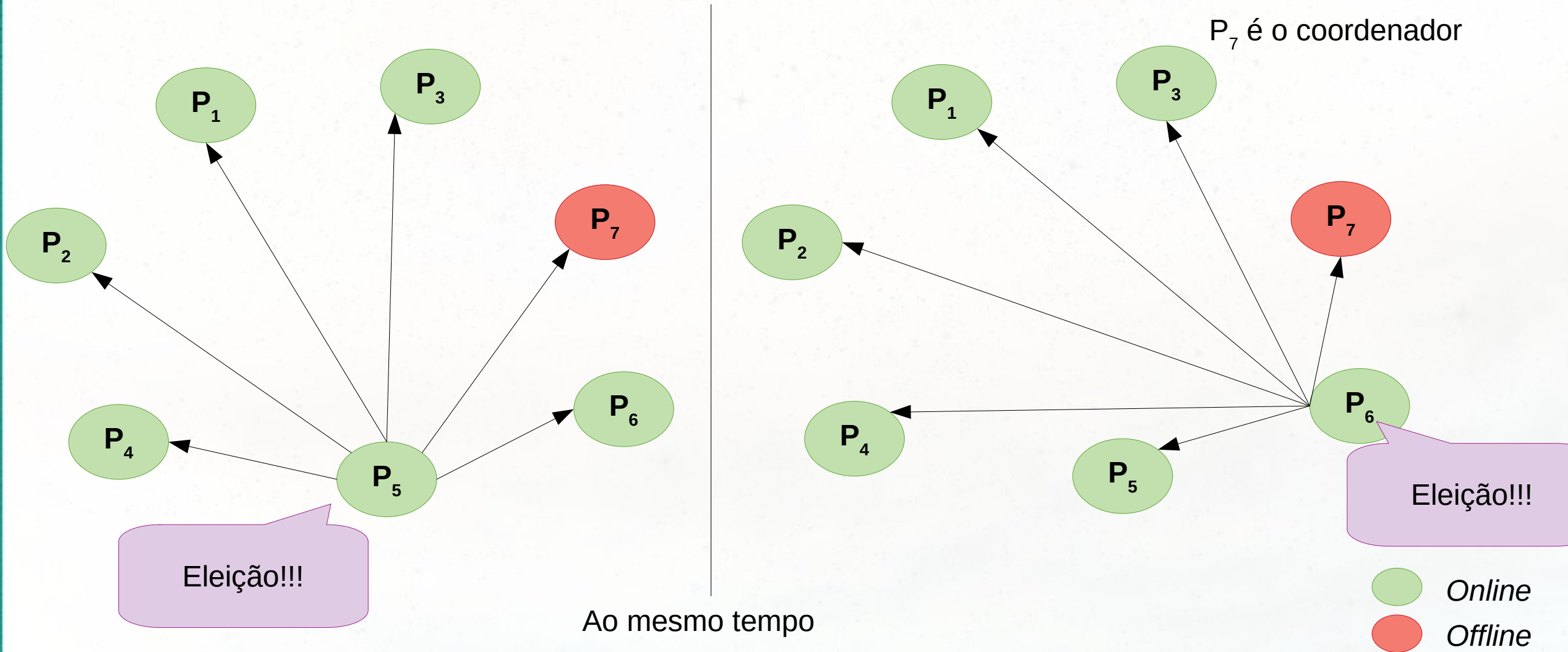


Online
Offline

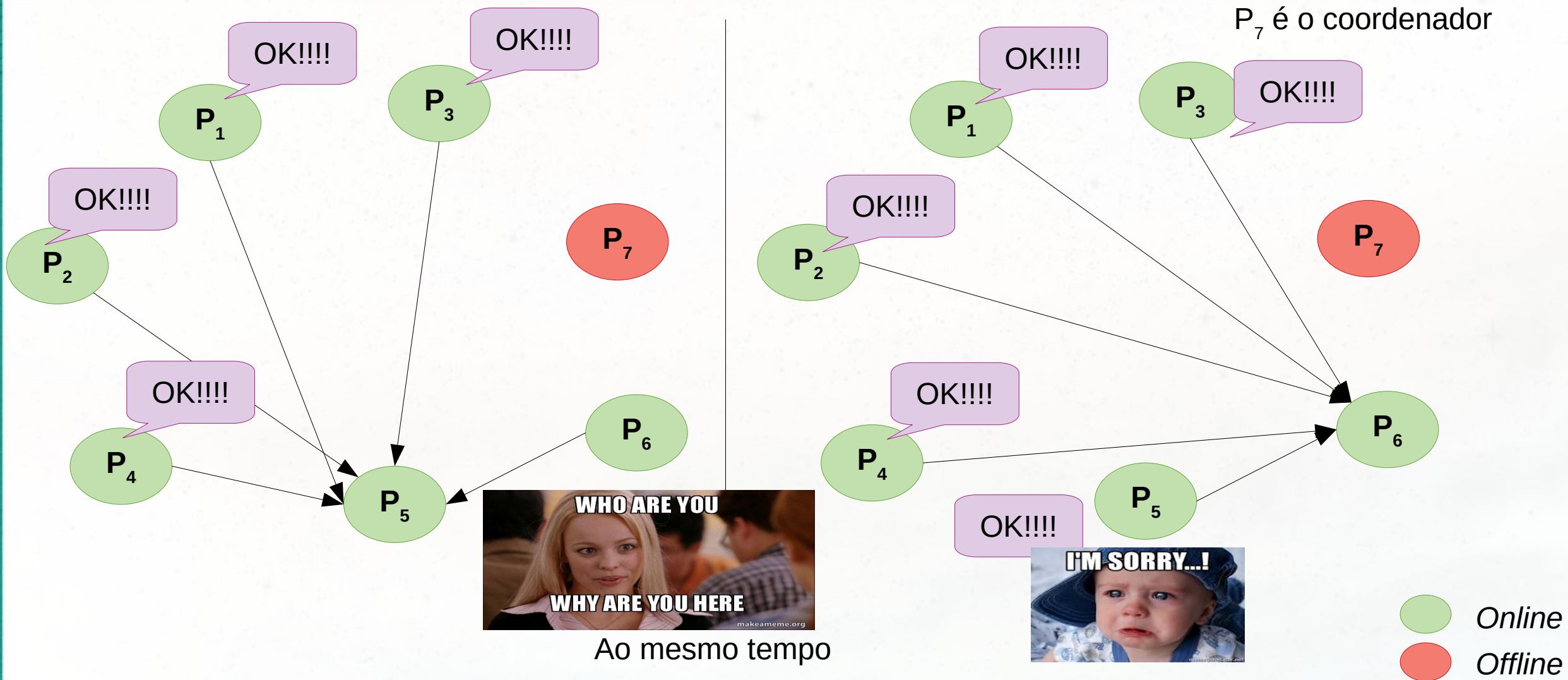
Algoritmo "Bully"



Algoritmo "Bully"

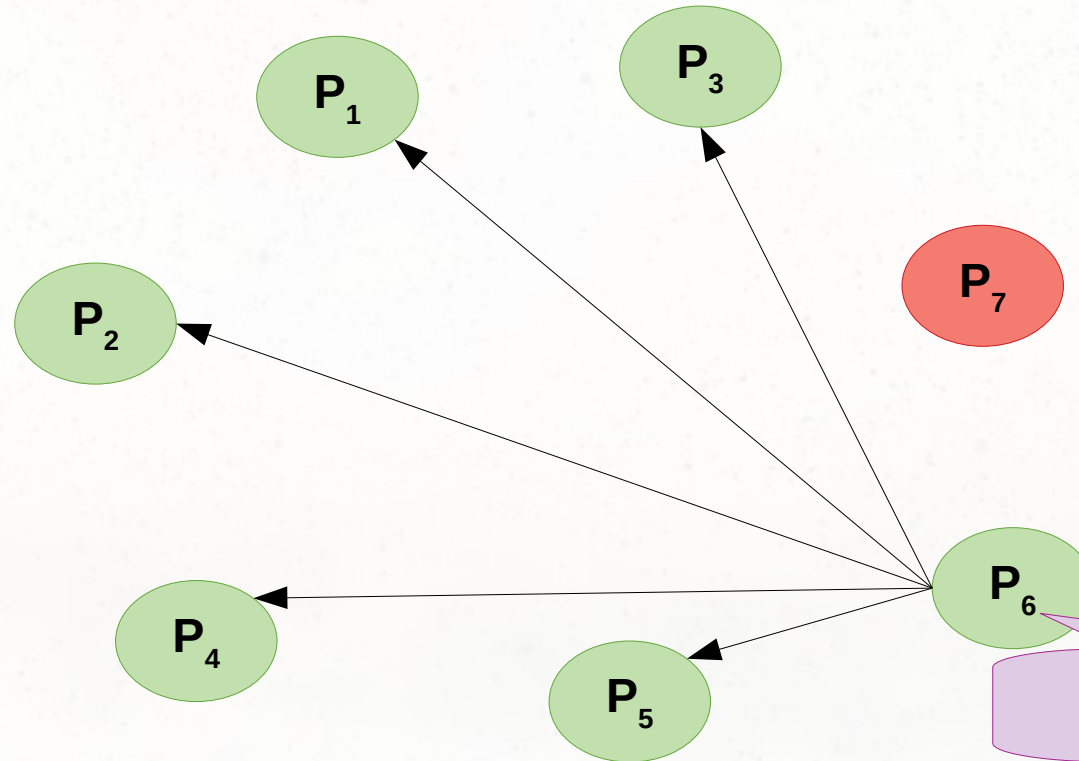


Algoritmo "Bully"



Algoritmo “Bully”

P_6 é o coordenador



Eu sou o Coordenador!!!

● Online
● Offline

Algoritmo “Bully”

- Quando um processo nota que o coordenador não está respondendo a uma requisição, ele inicia uma eleição:
 - 1- P_1 envia uma mensagem de ELEIÇÃO para todos os processos com números maiores que o seu;
 - 2- Se nenhum responde, P_1 ganha a eleição e se torna coordenador;
 - 3- Se um processo com número maior responde, ele assume a coordenação.

Algoritmo “Bully”

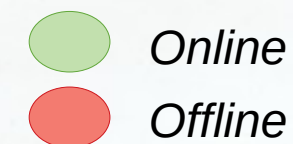
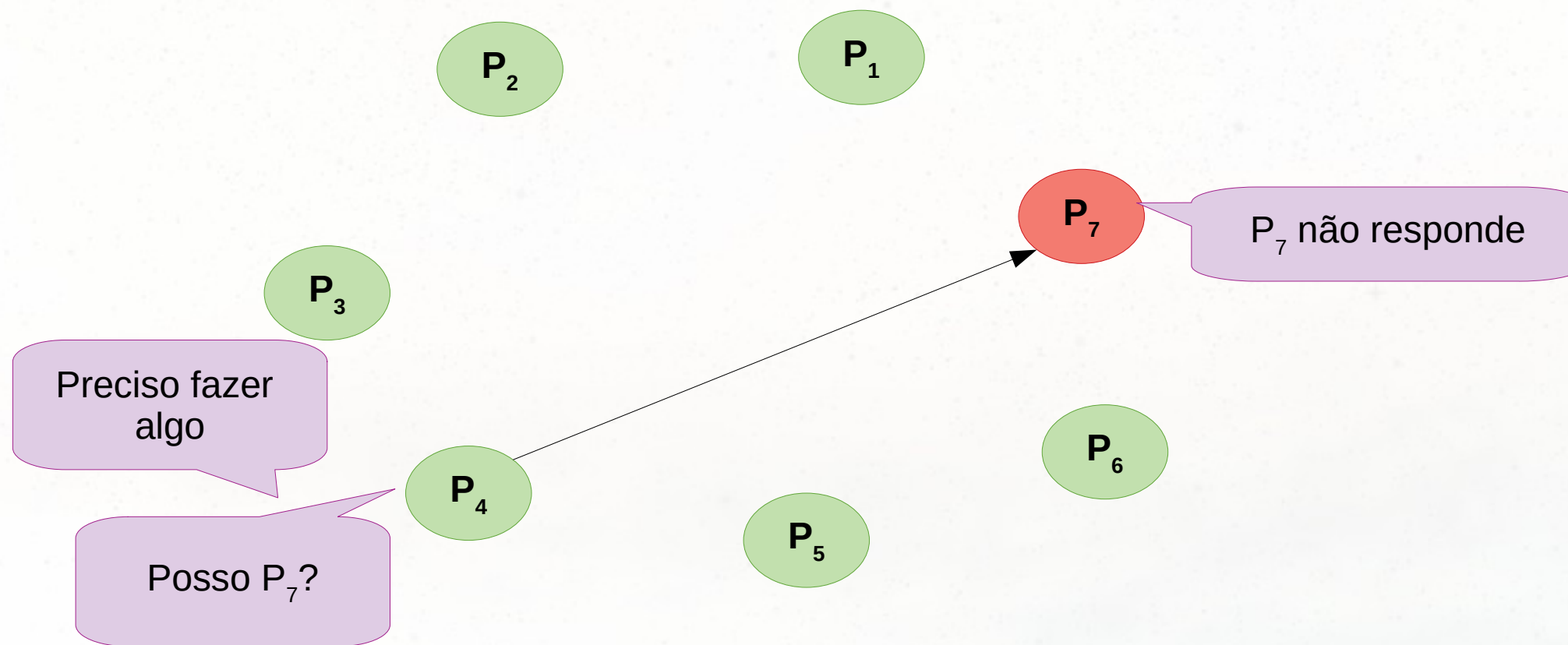
- Quando um processo recebe uma mensagem de ELEIÇÃO de um processo de menor número, ele envia uma mensagem OK de volta indicando que ele vai tomar o comando.
- Depois disso ele inicia uma eleição.
 - Eventualmente todos os processos abandonam a disputa, com exceção de um que é o novo coordenador.
 - O coordenador envia uma mensagem a todos os processos avisando que é o novo coordenador.
- Se um processo que estava “fora do ar” volta, ele inicia uma eleição.

Algoritmo de Anel



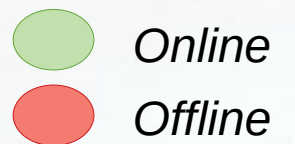
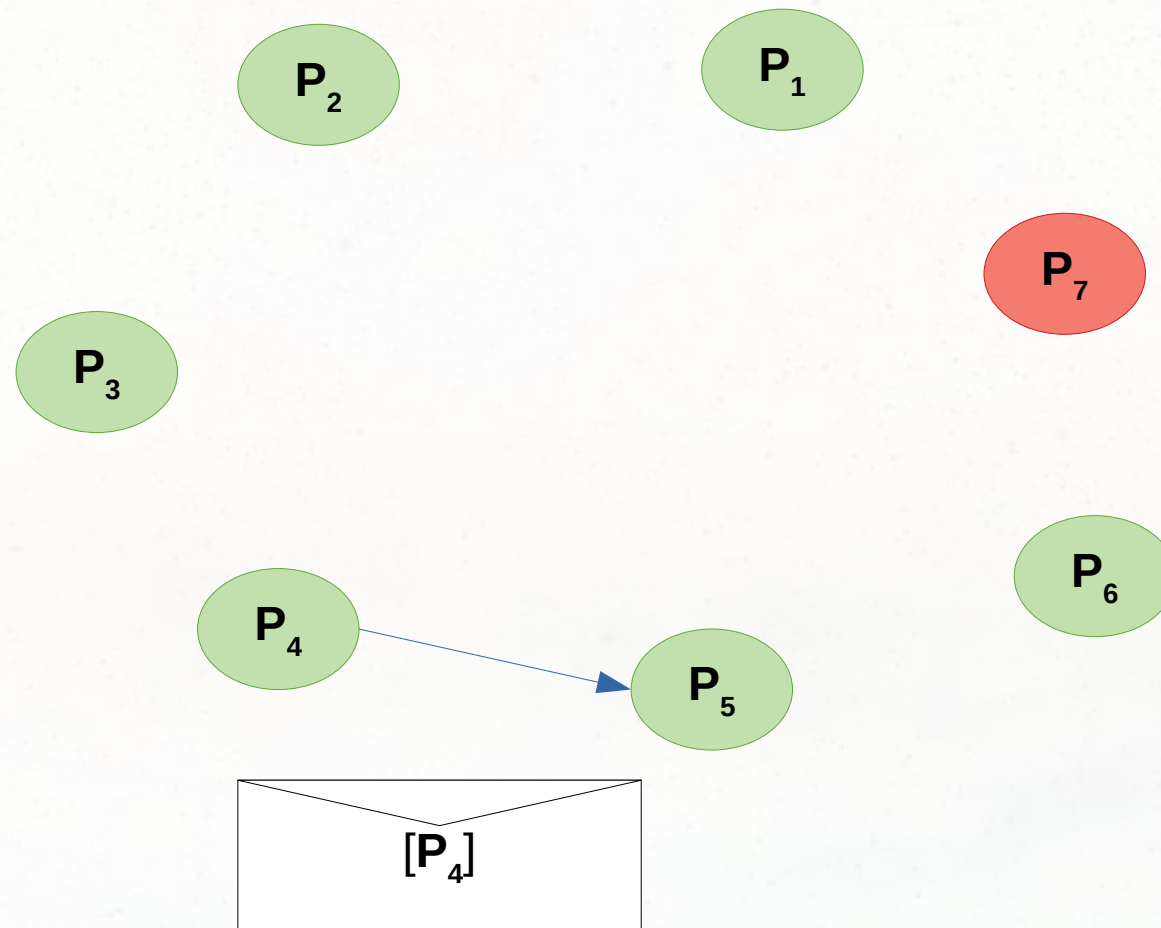
Algoritmo de Anel

P_7 é o coordenador



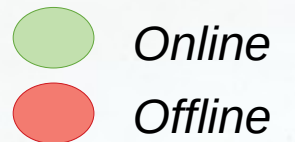
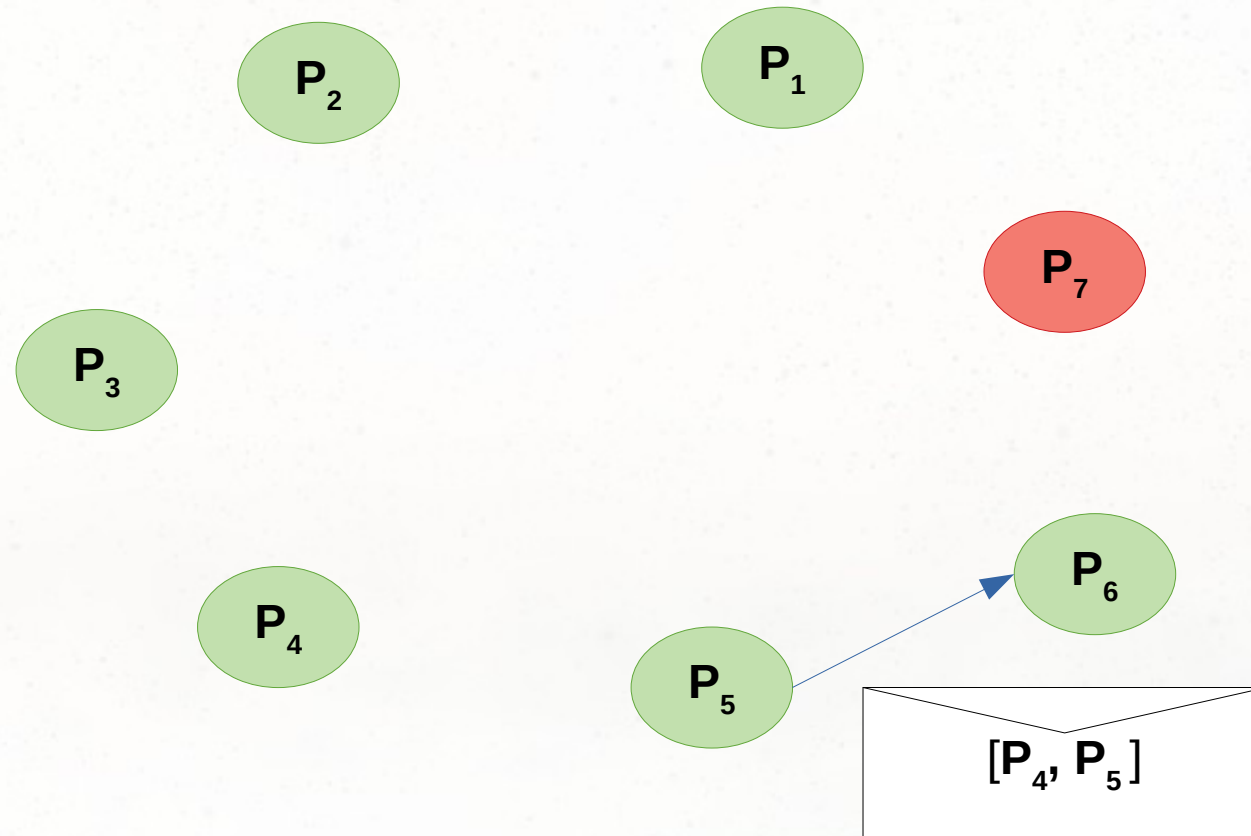
Algoritmo de Anel

P_7 é o coordenador



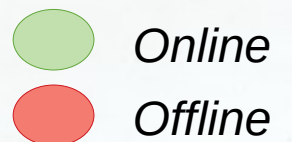
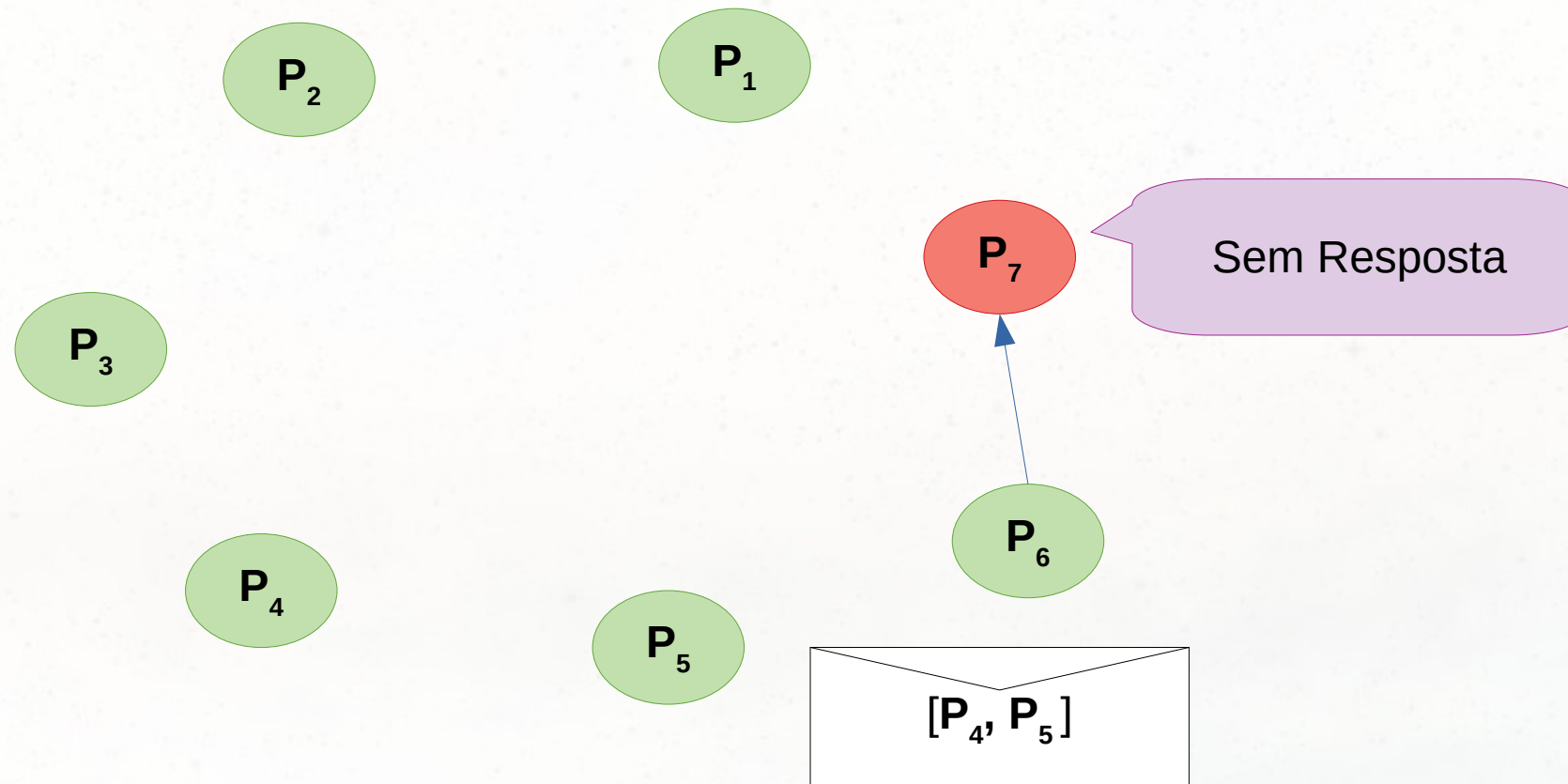
Algoritmo de Anel

P_7 é o coordenador



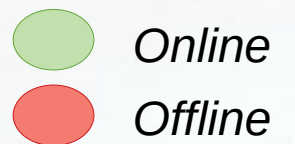
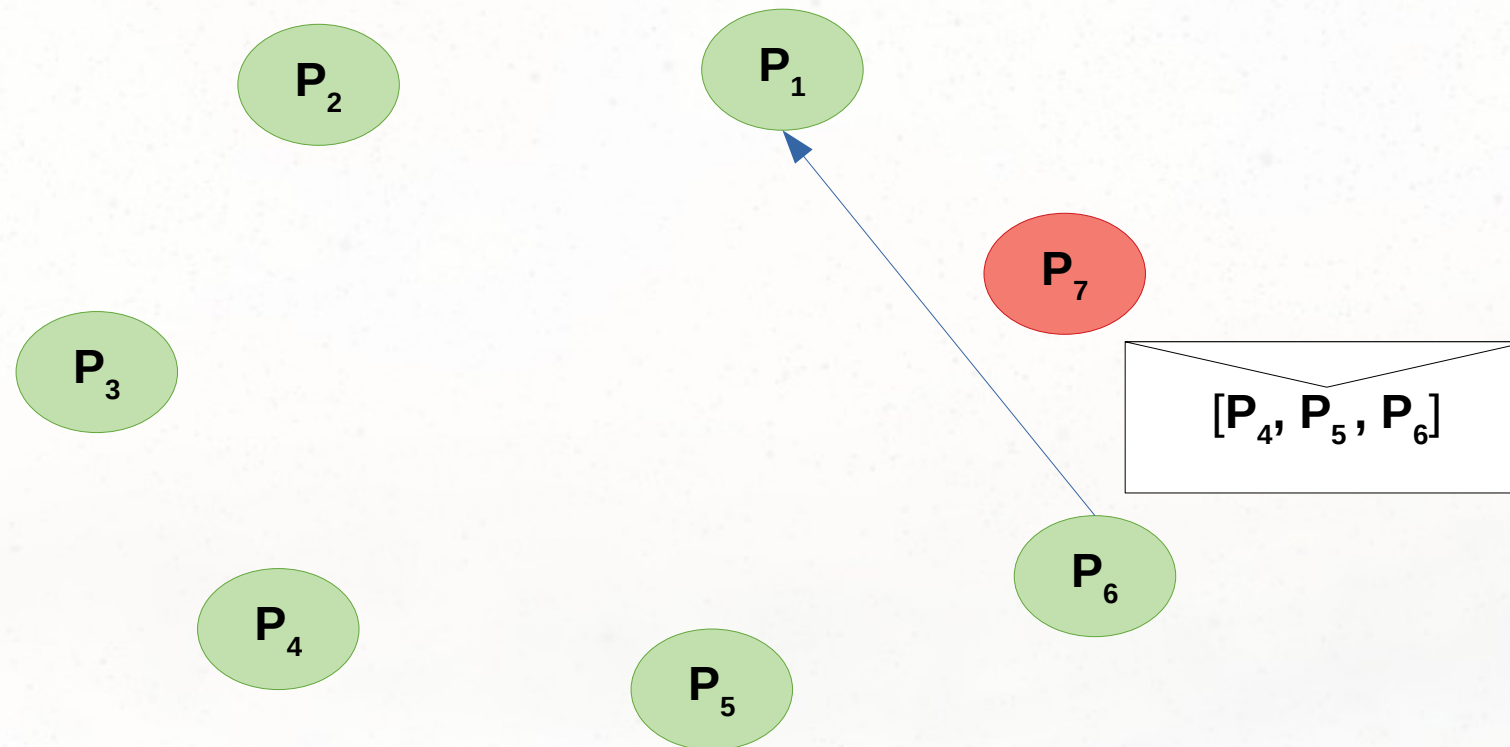
Algoritmo de Anel

P_7 é o coordenador

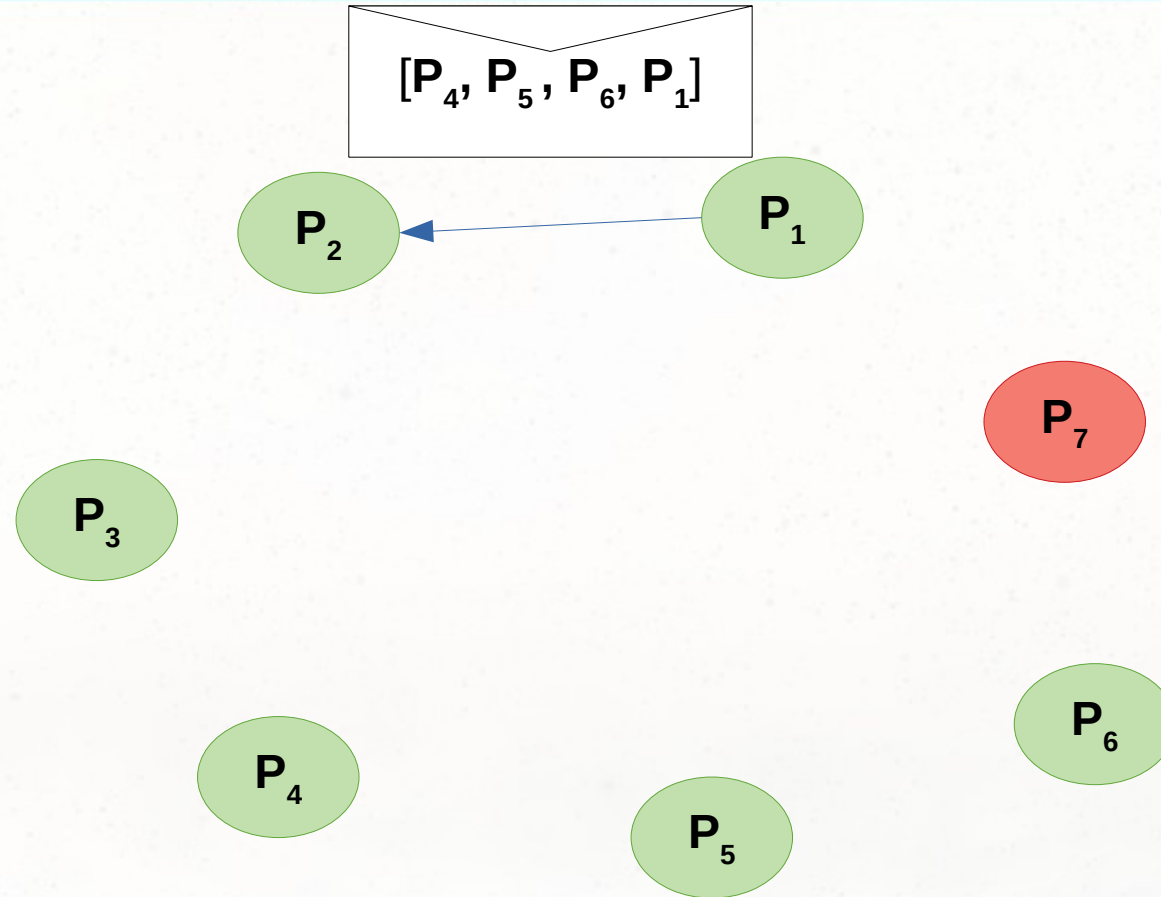


Algoritmo de Anel



P_7 é o coordenador



Algoritmo de Anel

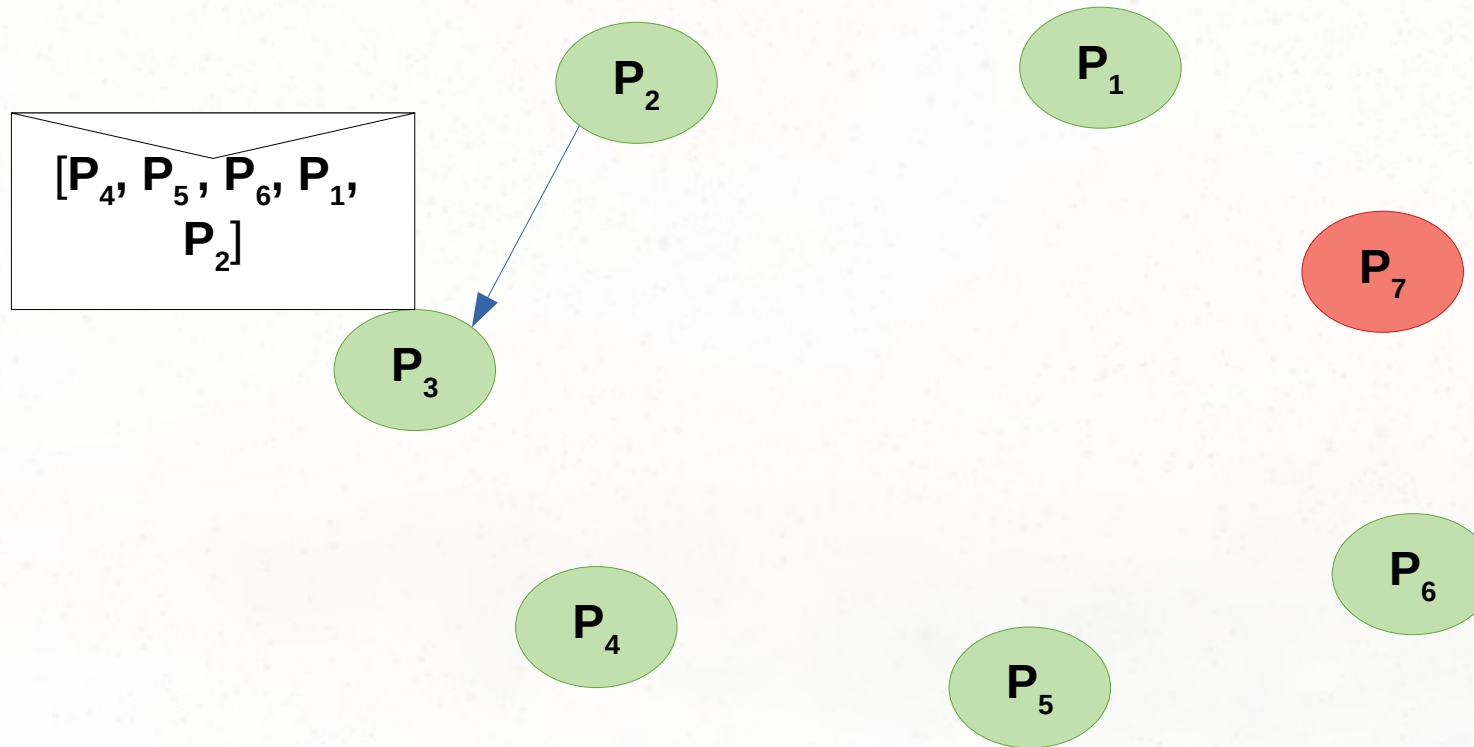




P_7 é o coordenador

 *Online*
 *Offline*

Algoritmo de Anel

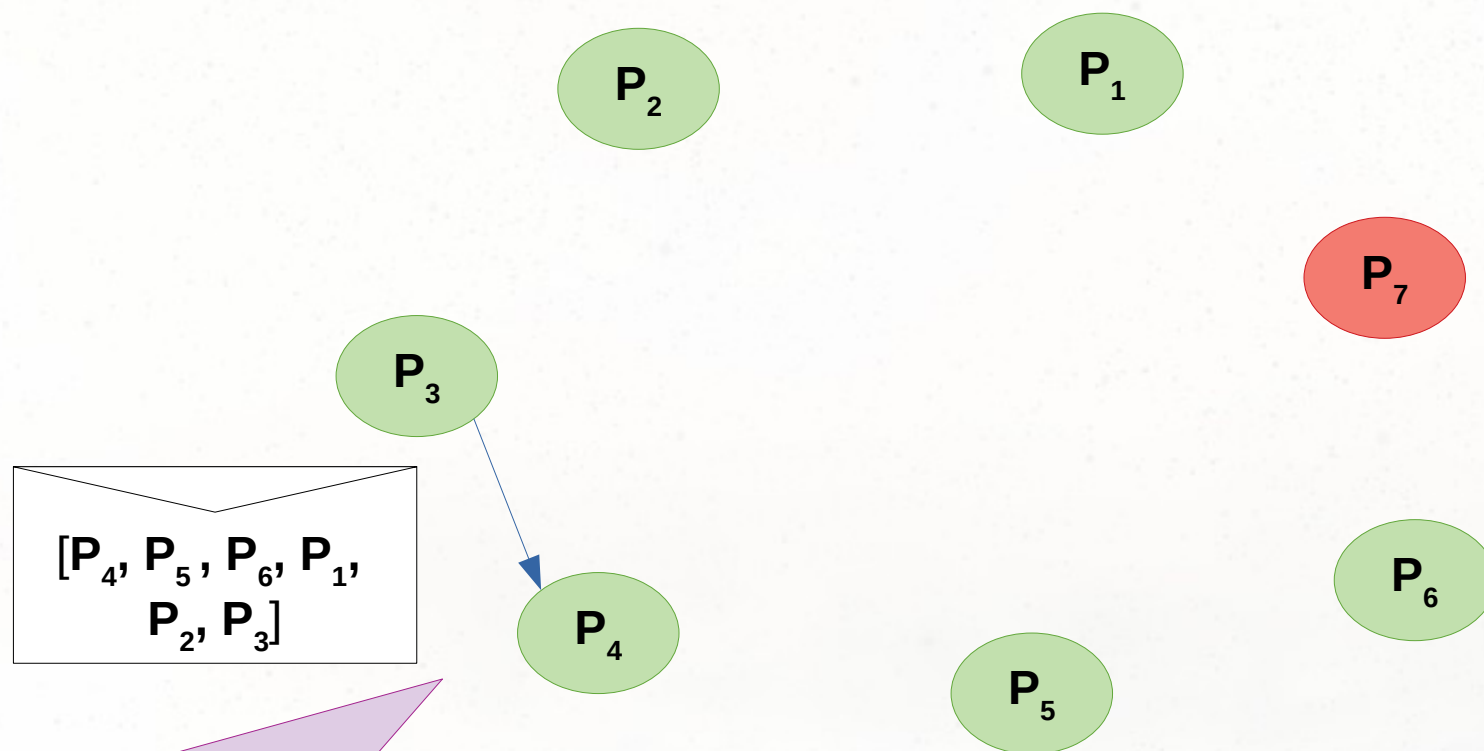
P_7 é o coordenador




 Online
 Offline

Algoritmo de Anel

P_6 é o coordenador

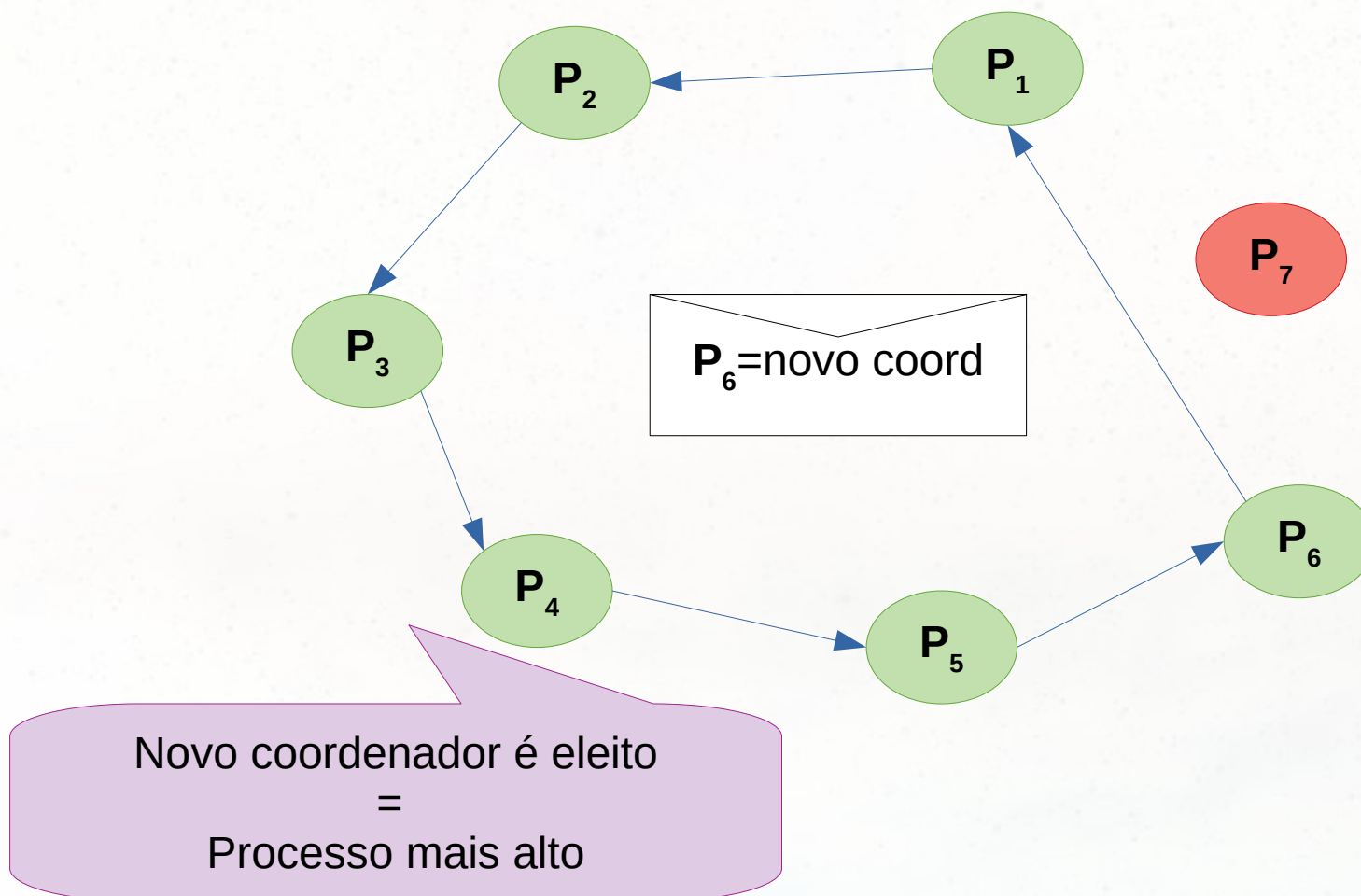


Novo coordenador é eleito
=
Processo mais alto

 Online
 Offline

Algoritmo de Anel

P_6 é o coordenador

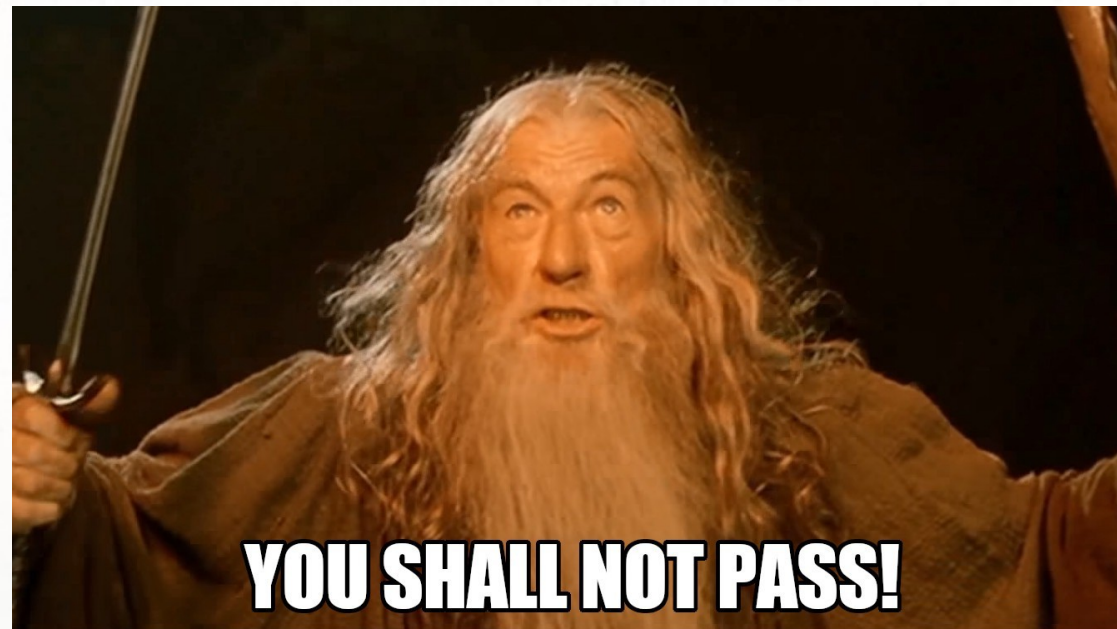


● Online
● Offline

Algoritmo de Anel

- Baseado no uso do anel.
- Quando um processo nota que o coordenador não está funcionando, ele envia uma mensagem de ELEIÇÃO para o seu sucessor contendo o seu número de processo.
- A mensagem segue e a cada passo é incluído o número do processo na lista da mensagem.
- Quando a mensagem dá a volta no anel o novo coordenador é determinado (o processo com mais alto número na lista) e uma mensagem circula novamente informando quem é o novo coordenador.
- Quando a mensagem dá a volta no anel ela é retirada.

Exclusão Mútua



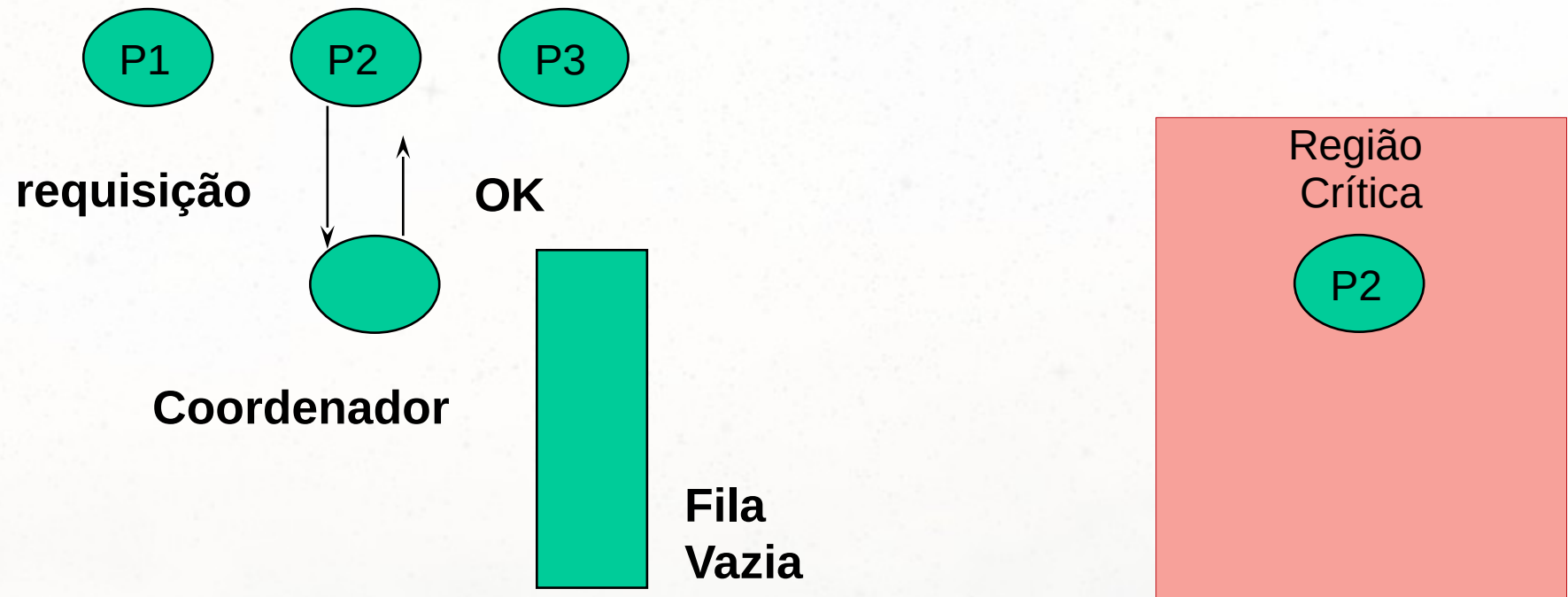
Exclusão Mútua

- A maneira mais direta de conseguir exclusão mútua em um sistema distribuído é imitar o que é feito no sistema centralizado:
 - Um processo é eleito como Coordenador. Quando um processo quer entrar na região crítica, ele envia uma mensagem requisitando ao Coordenador.
 - Se nenhum outro processo está na região crítica, o coordenador envia uma resposta dando permissão.
 - Quando a mensagem resposta chega, o processo entra na região crítica.

Exclusão Mútua

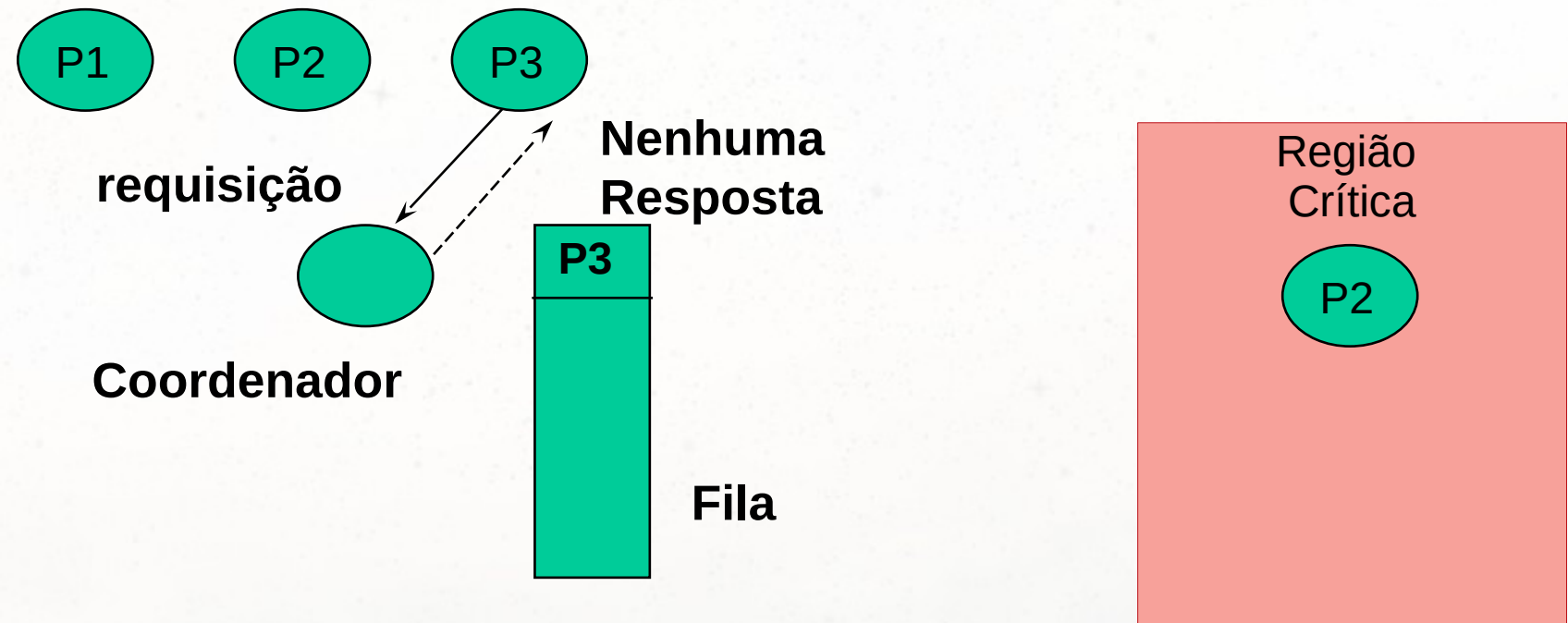
- Supondo que outro processo peça permissão para entrar na região crítica, e o Coordenador sabendo que outro processo está na região,
 - Ele não envia resposta alguma (bloqueando ou permitindo) este processo até que possa entrar na região.
- Quando o processo deixa a região, ele envia para o Coordenador uma mensagem liberando a região.

Exclusão Mútua - Algoritmo Centralizado



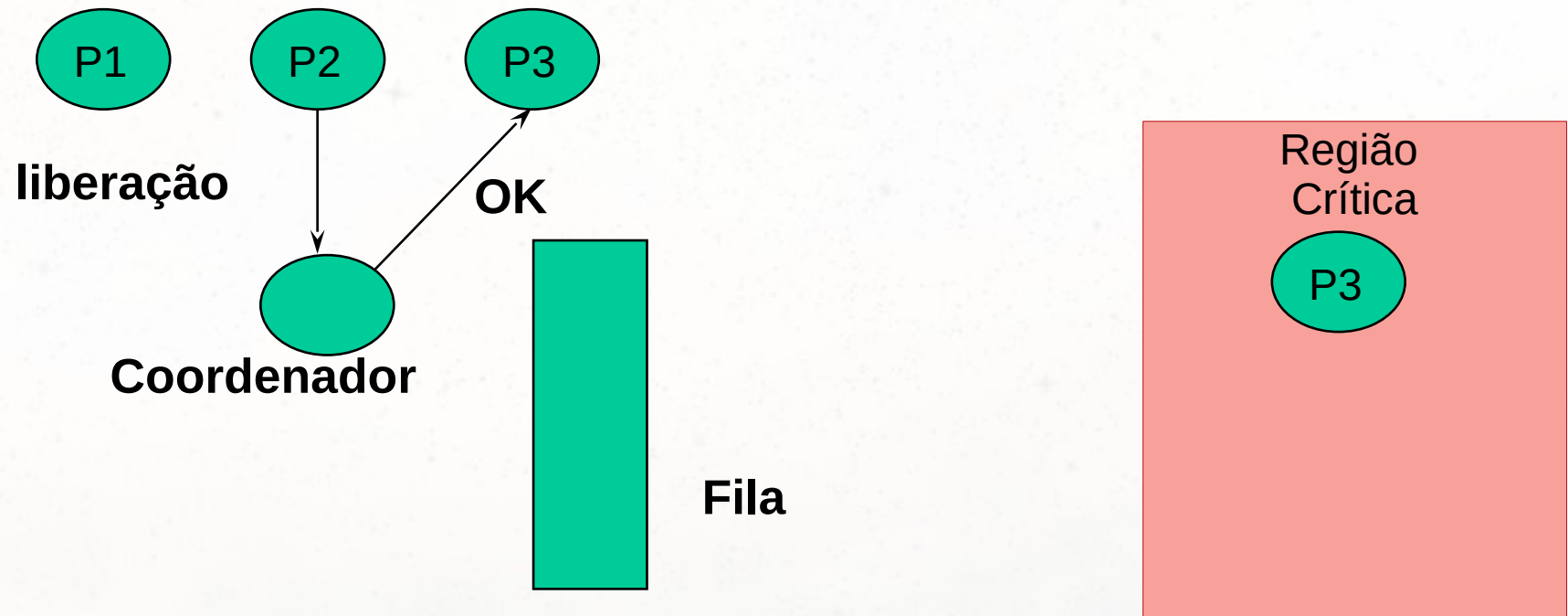
- Processo 2 pede permissão para o coordenador para entrar em uma região do sistema. O coordenador permite a entrada.

Exclusão Mútua - Algoritmo Centralizado



- Processo 3 pede permissão para entrar na mesma região do sistema. O coordenador não envia resposta nenhuma.

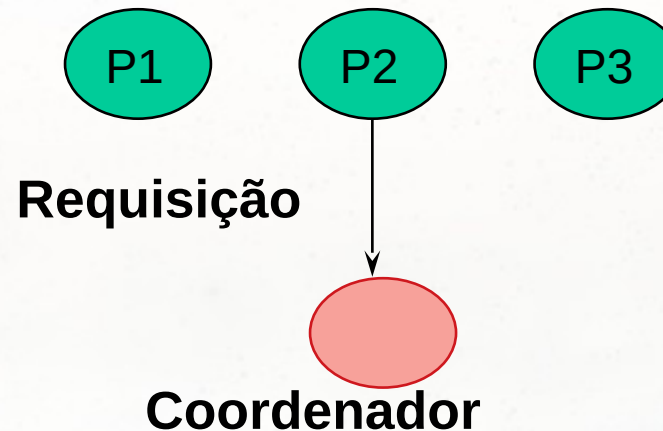
Exclusão Mútua - Algoritmo Centralizado



- Quando o processo 2 termina de utilizar a região, ele envia uma mensagem para o coordenador, que envia uma mensagem para o processo 3

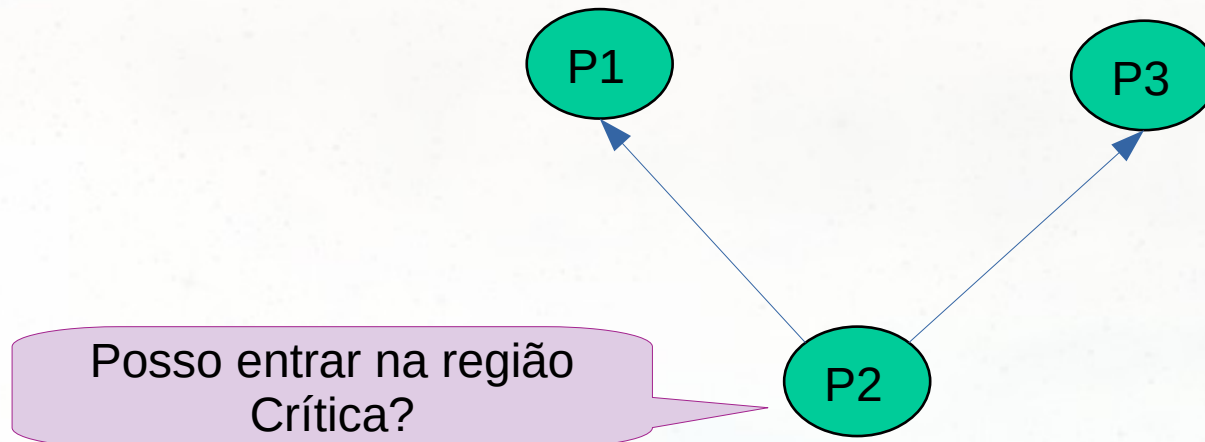
Exclusão Mútua - Algoritmo Centralizado

- Problema?
 - Falha no Coordenador inviabilizar o mecanismo.
 - Como eu resolvo?



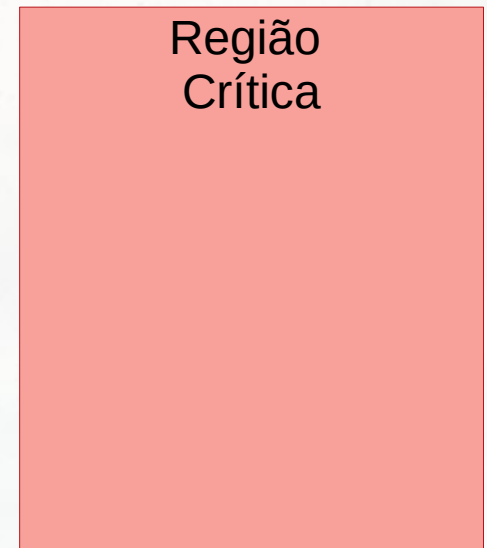
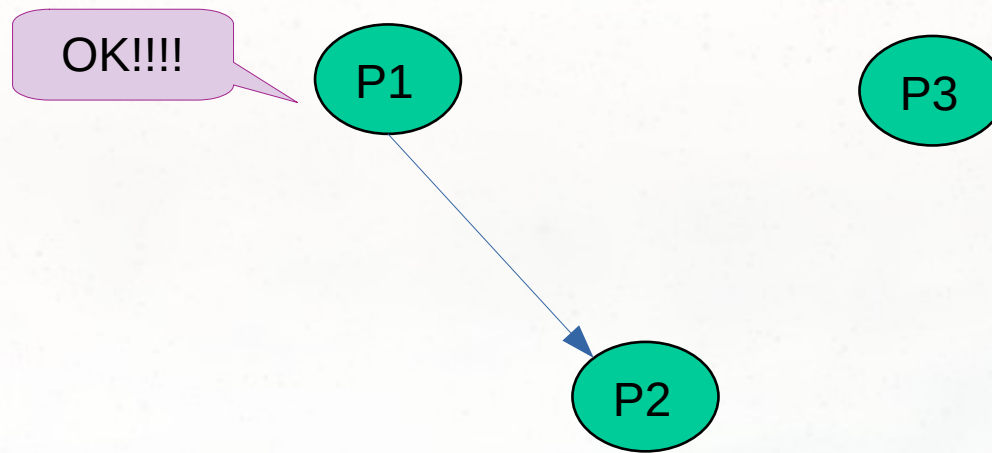
Exclusão Mútua - Algoritmo Distribuído

- Quando um processo quer entrar na região crítica:
 - ele constrói uma mensagem contendo o nome da região, número do processo e o tempo corrente.
- Ele envia a mensagem para todos os outros processos.



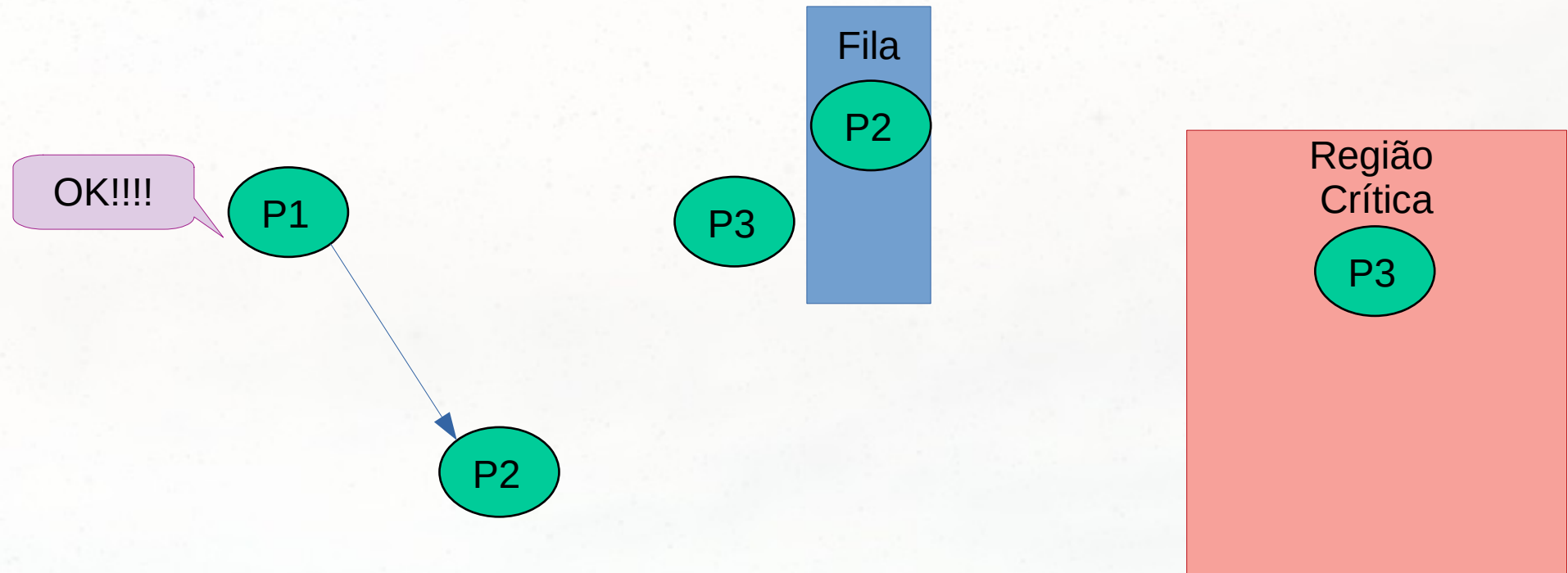
Exclusão Mútua - Algoritmo Distribuído

- Quando um processo recebe uma mensagem de requisição de outro processo:
 - 1- Se o receptor não está na região crítica e não quer entrar, ele envia de volta uma mensagem OK;



Exclusão Mútua - Algoritmo Distribuído

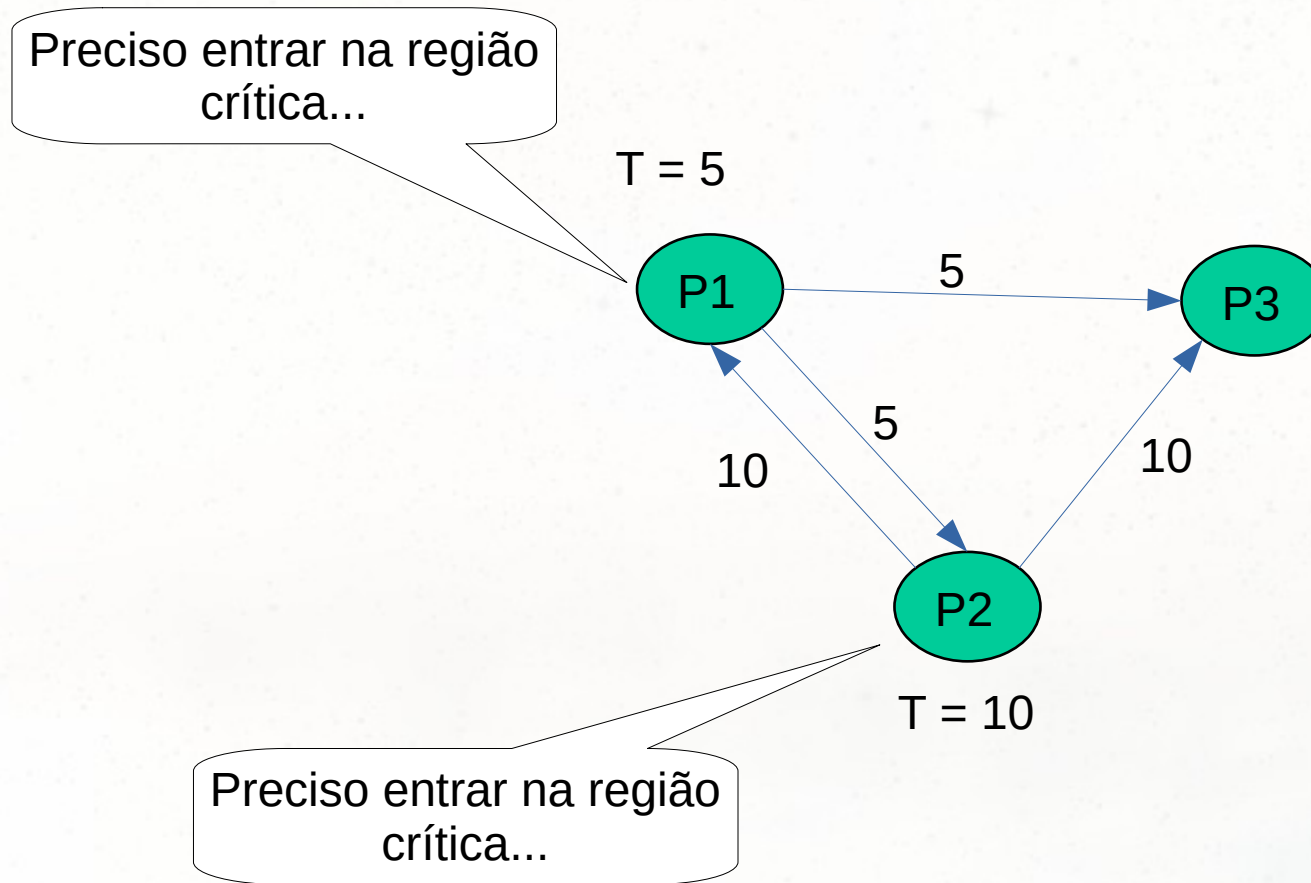
- Quando um processo recebe uma mensagem de requisição de outro processo:
2- Se o receptor já está na região, ele não responde e coloca a requisição na fila;



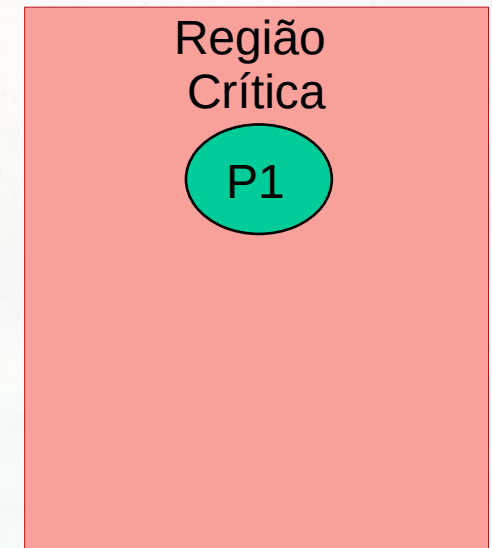
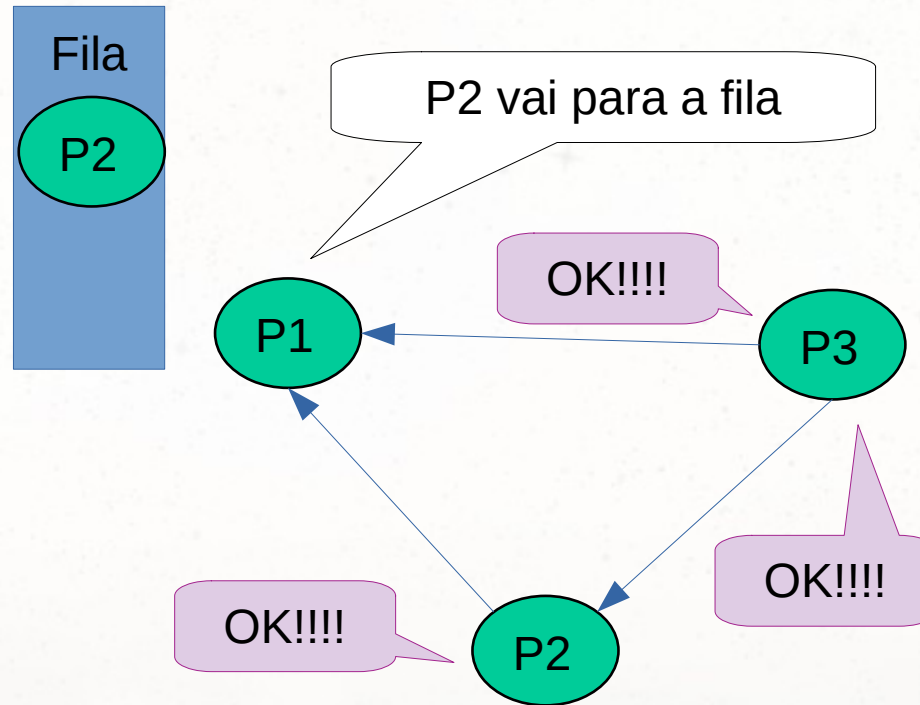
Exclusão Mútua - Algoritmo Distribuído

- Quando um processo recebe uma mensagem de requisição de outro processo:
 - 3- Se o receptor quer entrar na região crítica mas ainda não o fez, ele compara o tempo da mensagem que chegou com o tempo da mensagem que ele enviou para os outros processos.
 - O menor tempo vence.
 - Se a mensagem que chegou é menor ele envia de volta uma mensagem com OK.
 - Se a sua mensagem tem o menor tempo ele coloca na fila a requisição que chegou e não envia nada.

Exclusão Mútua - Algoritmo Distribuído



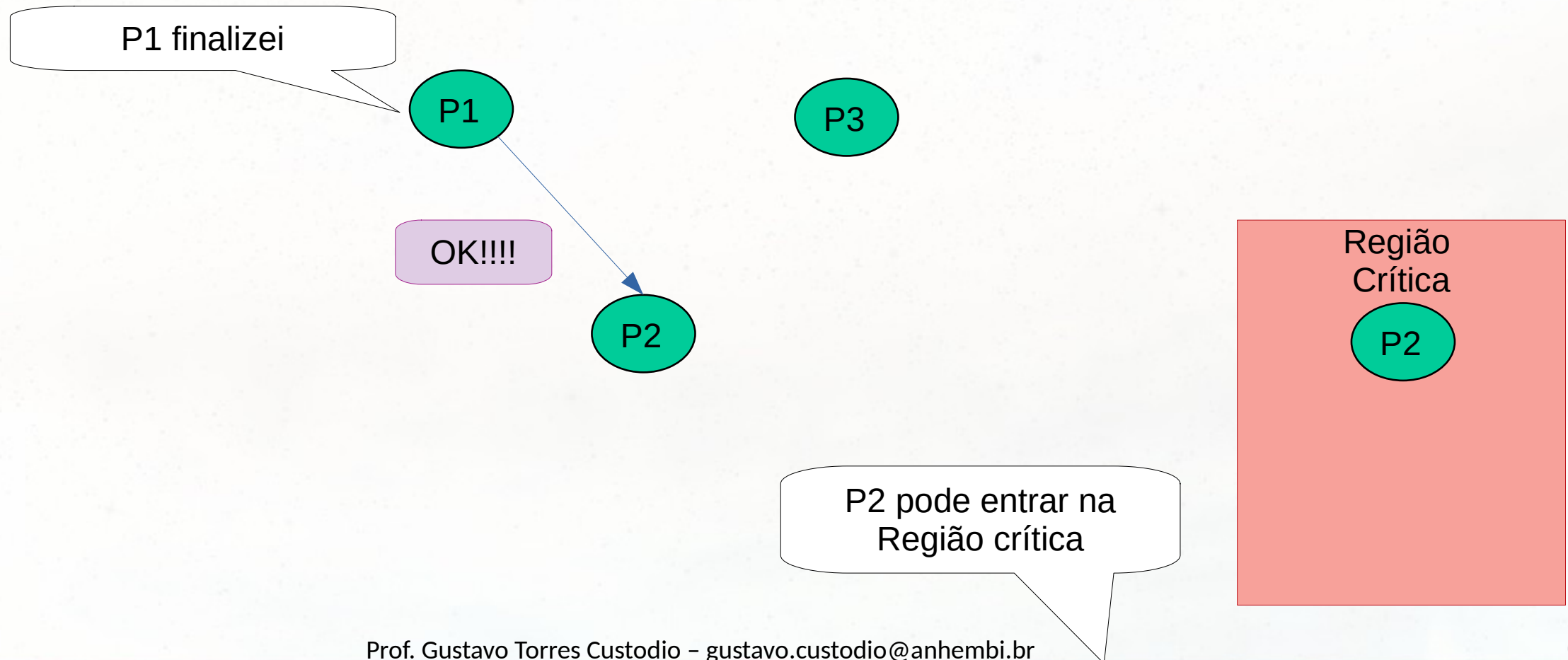
Exclusão Mútua - Algoritmo Distribuído



Cada processo verifica o tempo do processo e descobre o menor

Menor => P1

Exclusão Mútua - Algoritmo Distribuído

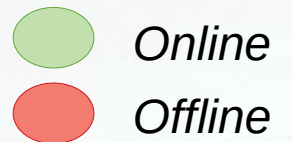
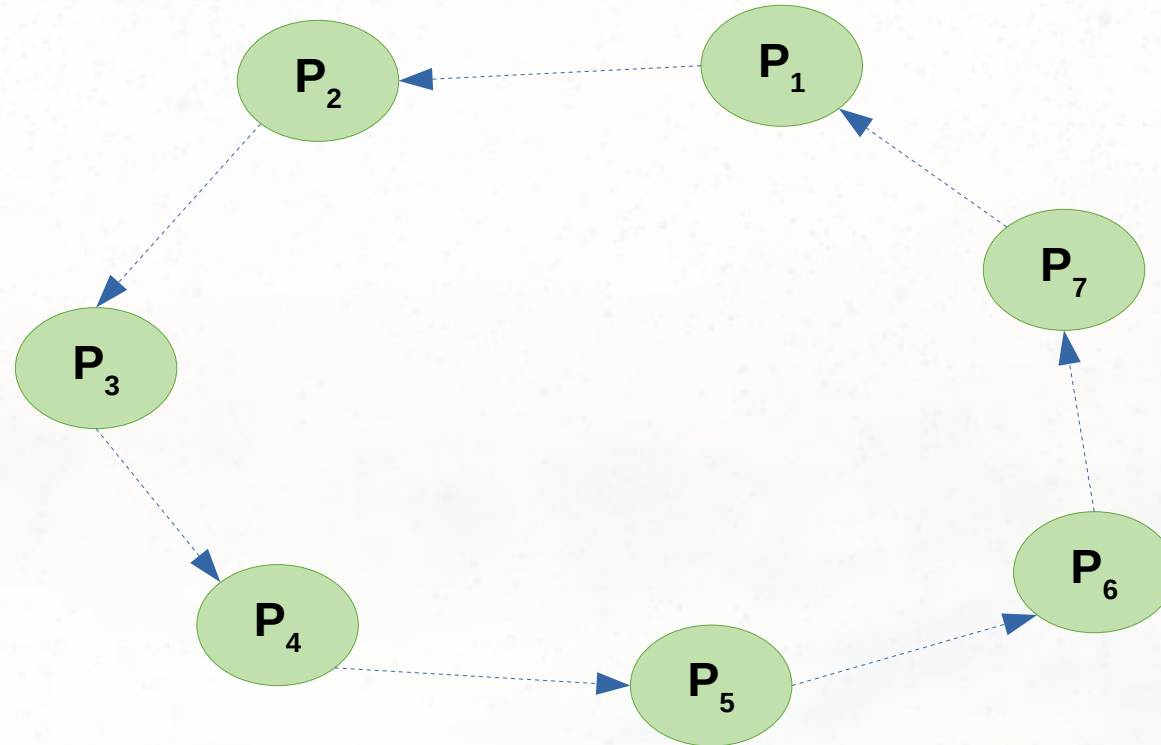


Exclusão Mútua – Algoritmo Distribuído

- Observações:
 - Após pedir permissão um processo espera até que todos tenham dado a sua permissão.
 - Quando todas as permissões chegam o processo pode entrar na região crítica.
 - Quando ele sai da região crítica, ele envia uma mensagem OK para todos os processos na sua fila.

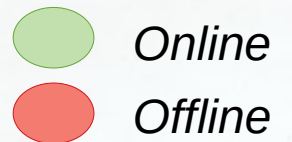
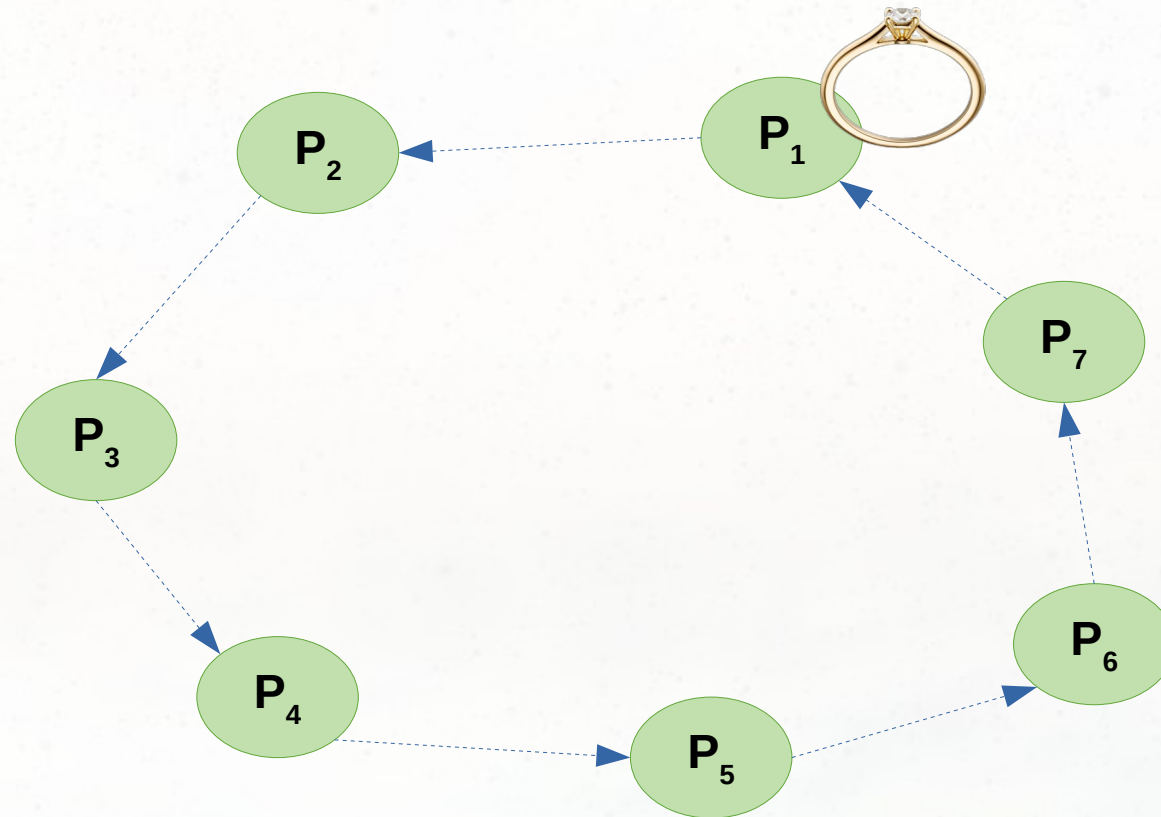
Algoritmo Token Ring

- É construído um anel lógico por software no qual a cada processo é atribuído uma posição no anel.



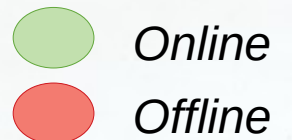
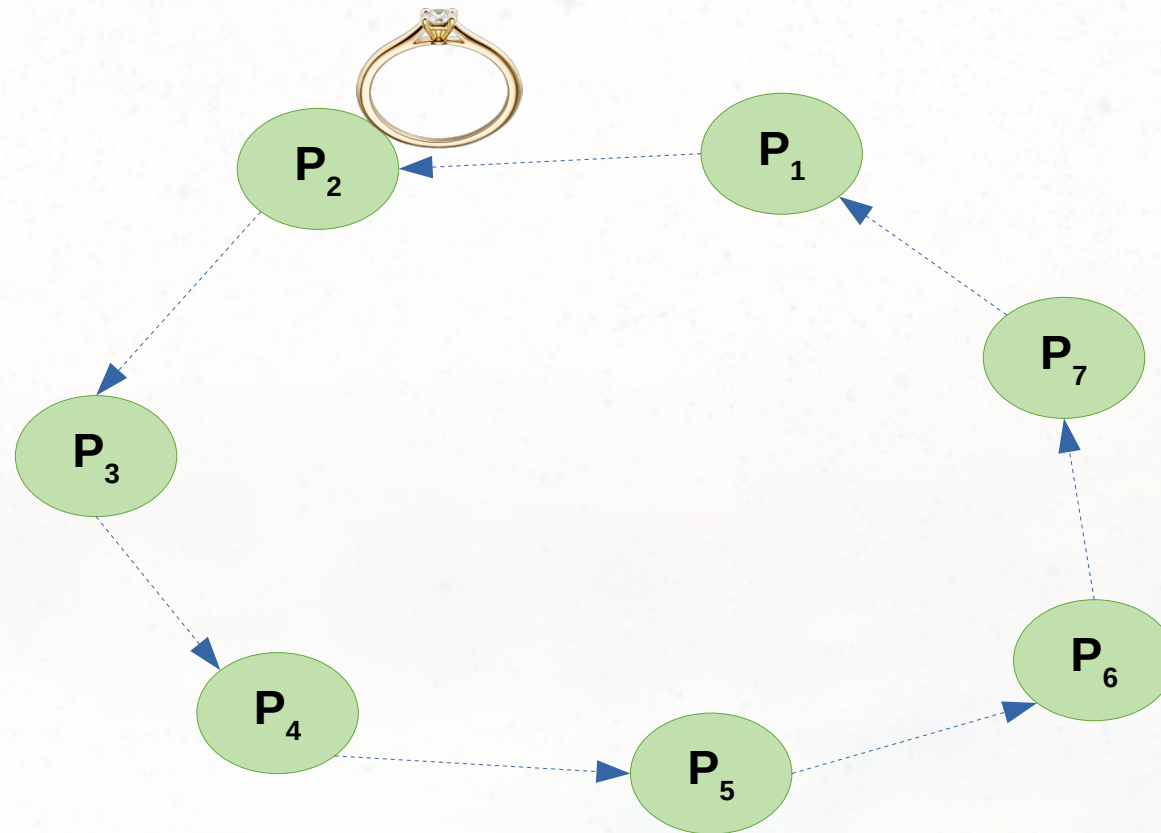
Algoritmo Token Ring

- Quando o anel é inicializado, o processo 1 ganha o “token”.



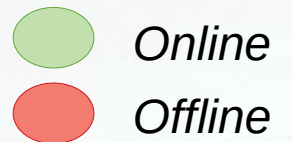
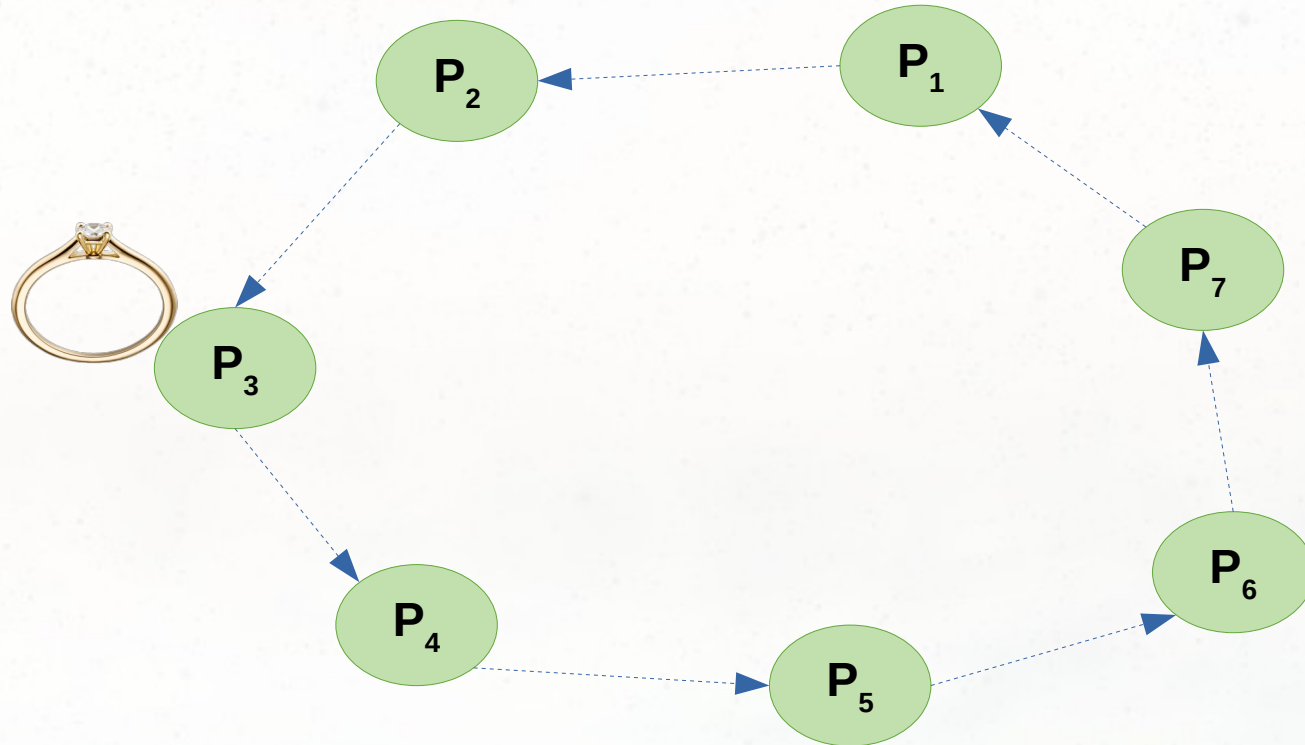
Algoritmo Token Ring

- O “token” circula no anel (passa do processo k para o $k+1$).



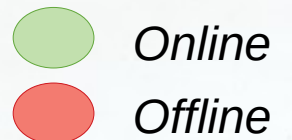
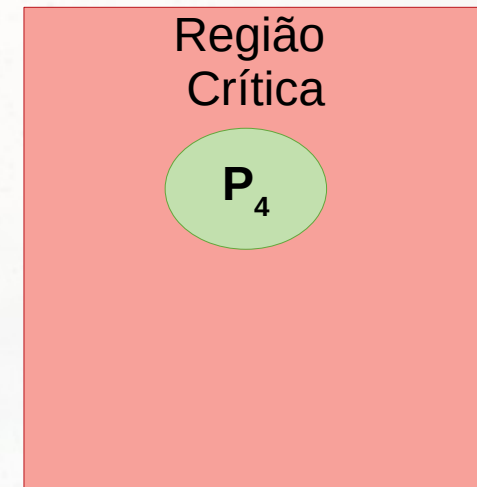
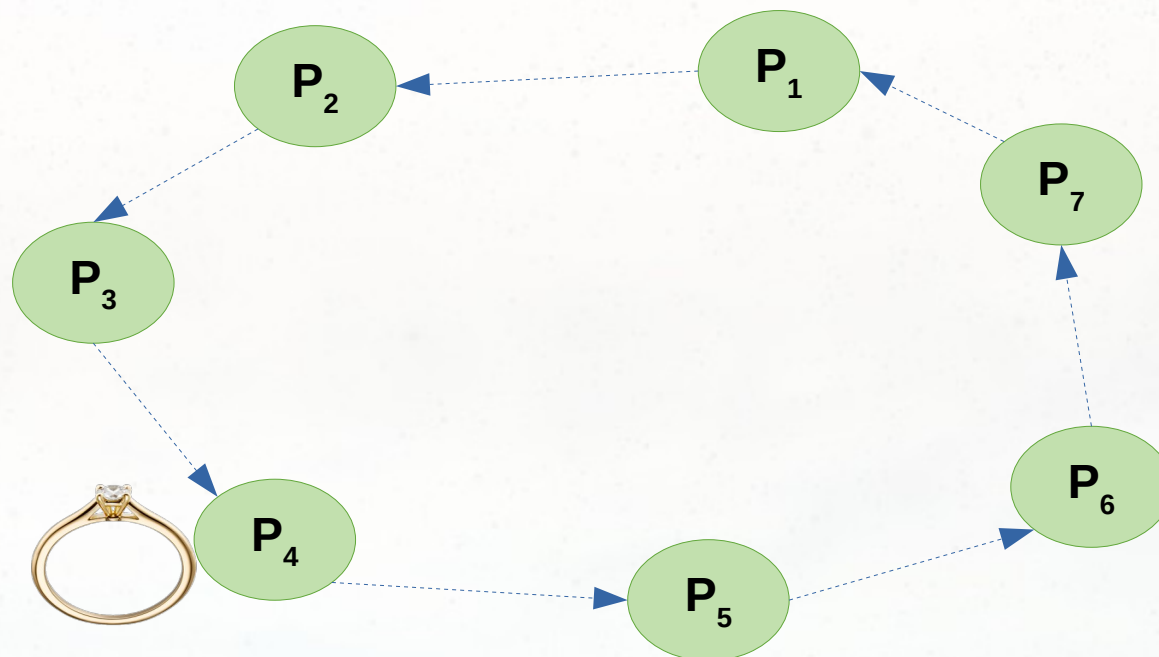
Algoritmo Token Ring

- O “token” circula no anel (passa do processo k para o $k+1$).

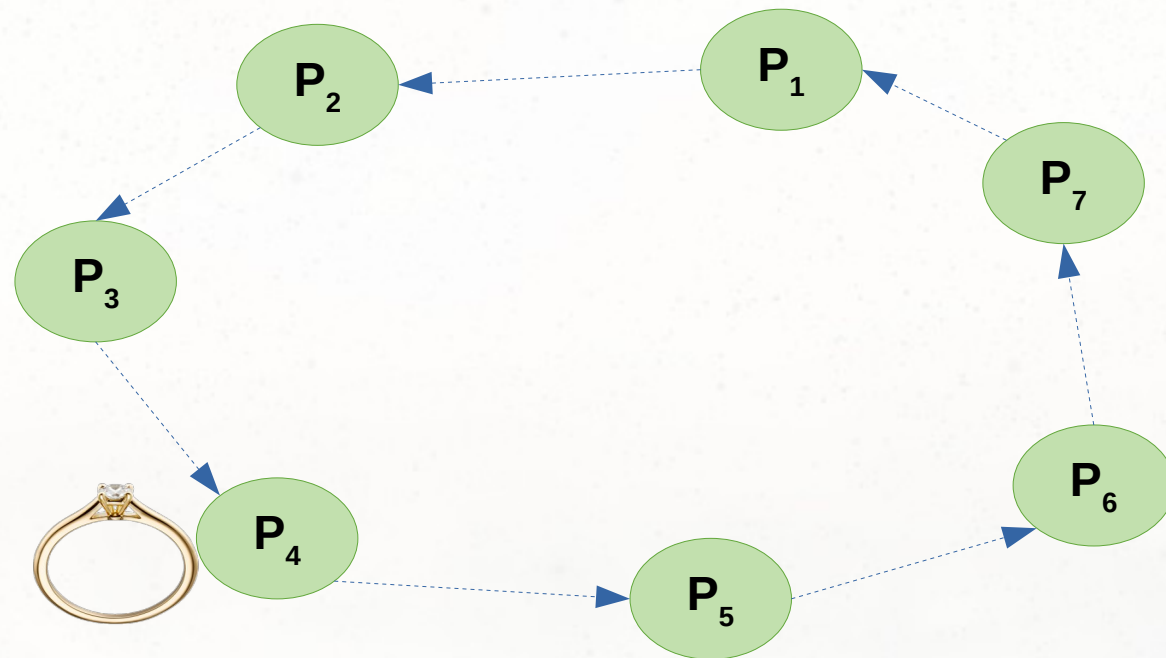


Algoritmo Token Ring

- Quando o processo ganha o “token” ele verifica se ele quer entrar na região crítica.
 - Caso positivo, ele entra na região, realiza o seu trabalho e ao deixar a região passa o “token” para o elemento seguinte do anel.



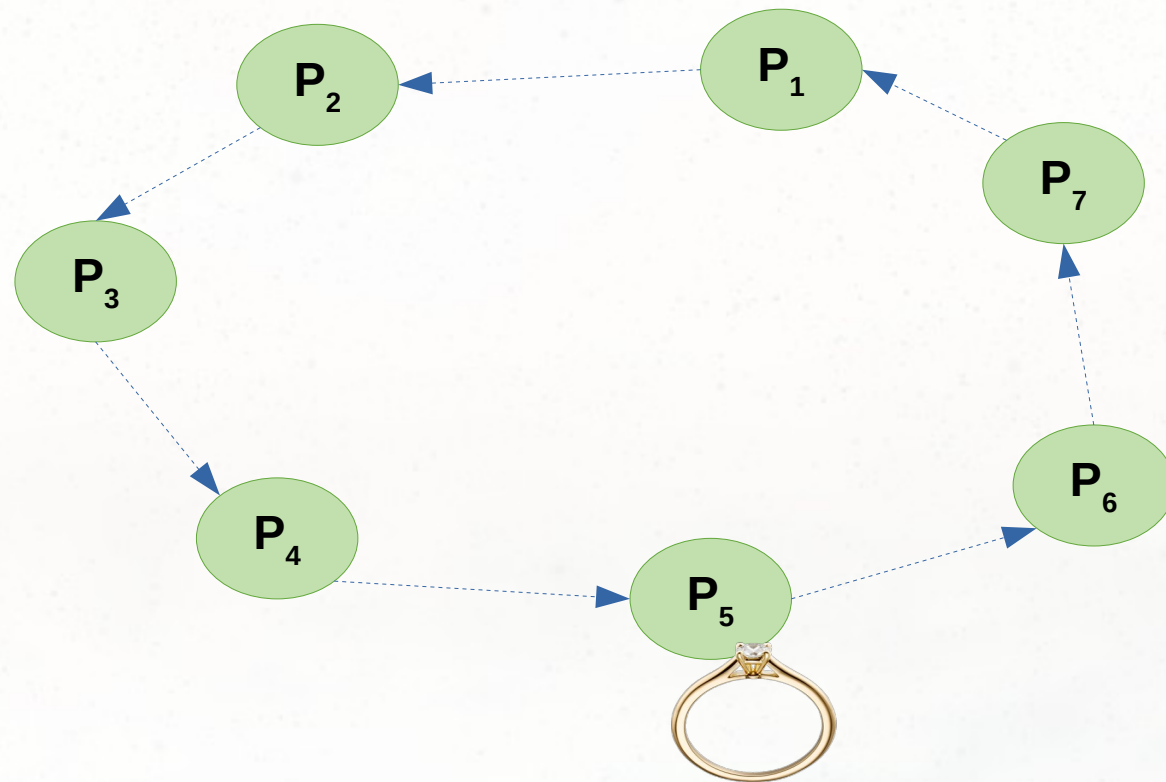
Algoritmo Token Ring



Região
Crítica

● Online
● Offline

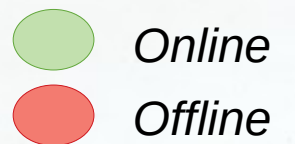
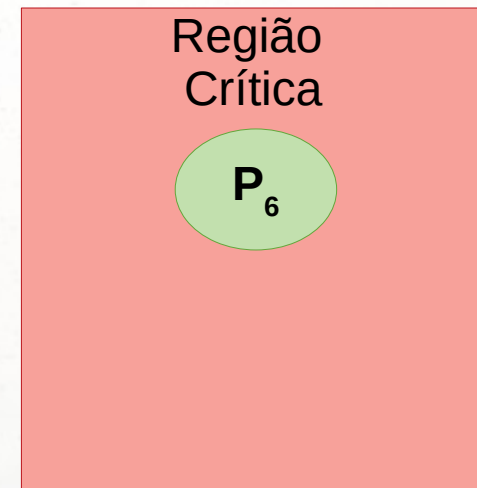
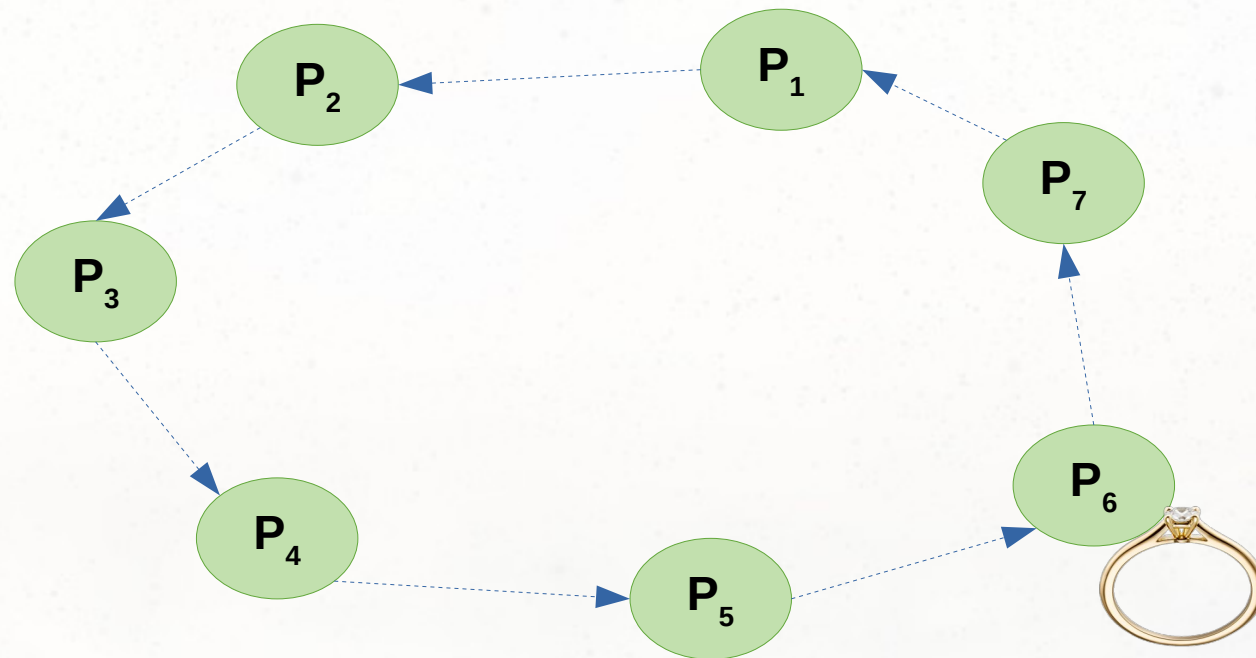
Algoritmo Token Ring



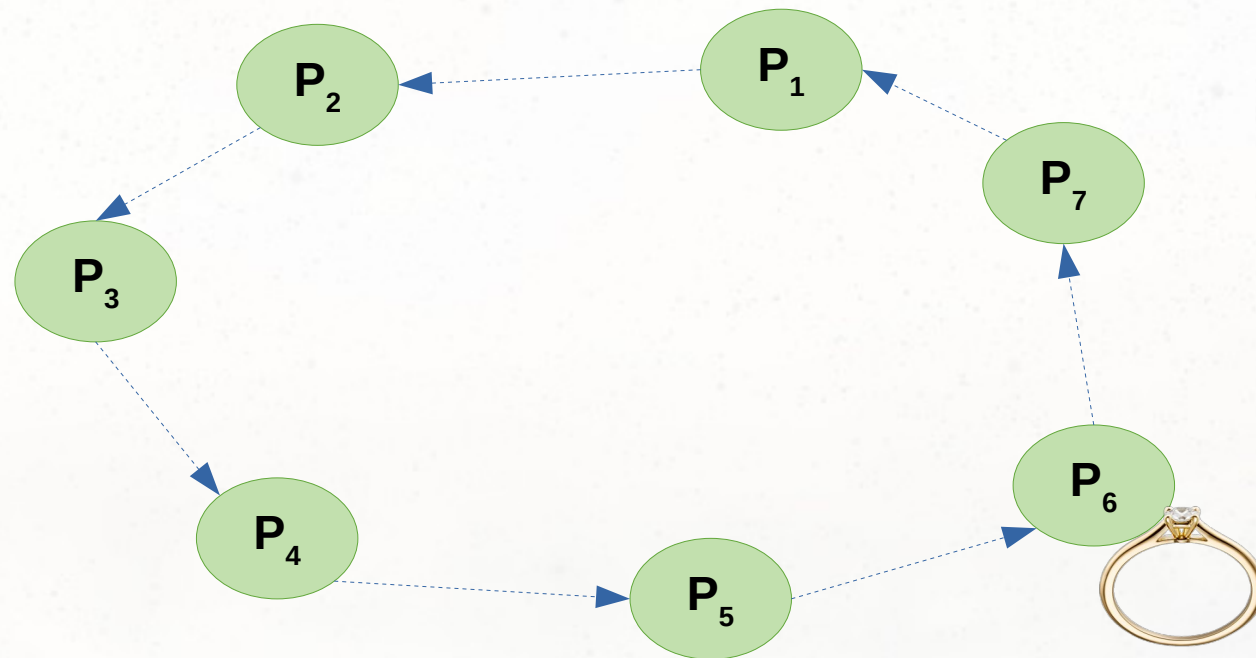
Região
Crítica

Online
Offline

Algoritmo Token Ring



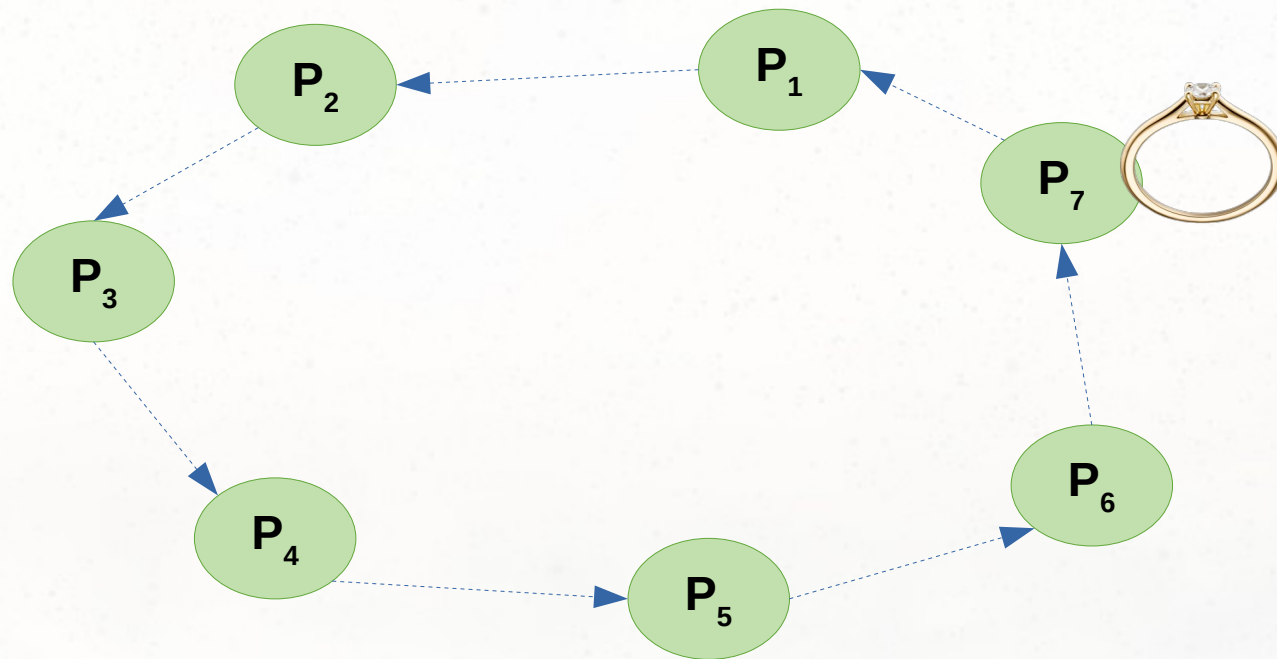
Algoritmo Token Ring





Região
Crítica

● Online
● Offline

Algoritmo Token Ring



Região
Crítica

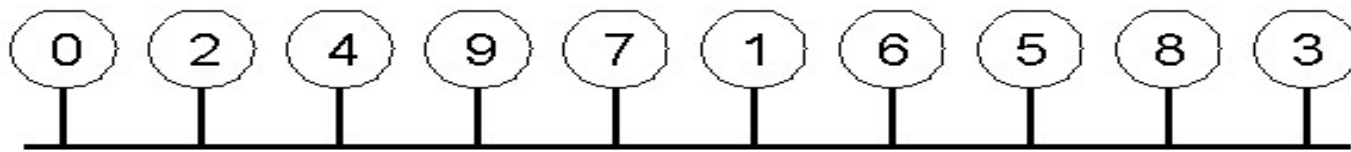
 Online
 Offline

Algoritmo Token Ring

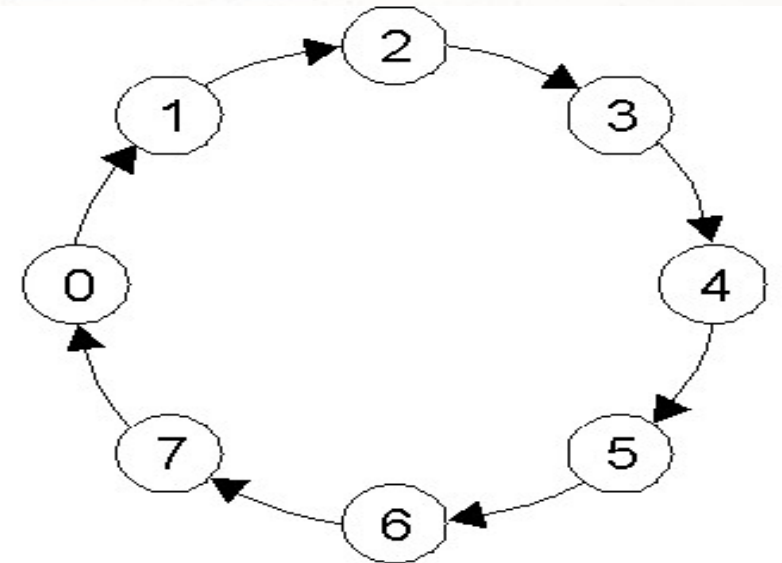
- Se o processo não quer entrar na região crítica ele simplesmente passa o “token”. Como consequência quando nenhum processo quer entrar na região crítica o “token” fica circulando pelo anel.
- Problemas:
 - Se o “token” é perdido ele precisa ser regenerado. A detecção de um “token” perdido é difícil.
 - Se um processo falha também ocorrem problemas. A solução é fazer o processo que recebe o “token” confirmar o recebimento. O processo que falhou pode ser retirado do anel, e o “token” enviado para o processo seguinte. Essa solução requer que todos os processos conheçam a configuração do anel.

Algoritmo Token Ring

- (a) Um grupo não ordenado de processos em uma rede.
- (b) Um anel lógico é construído via software.



(a)



(b)

Comparação entre os Algoritmos

Algoritmos	Mensagens por entrada/saída	Passos até a entrada	Problemas
Centralizado	3	2	Coordenador para
Distribuído	$2 (n - 1)$	$2 (n - 1)$	Qualquer processo para
Token ring	1 até ∞	0 até $n - 1$	Perdeu o token por parada de processo

Dúvidas



Exercícios



Exercício

- Utilize o Tipo Abstrato de Dados → Processo
 - Implemente os algoritmos de coordenação:
 - Algoritmo “Bully”
 - Algoritmo de Anel
 - Implemente os algoritmos de exclusão mútua:
 - Algoritmo Centralizado
 - Algoritmo Distribuído
 - Algoritmo Token Ring



Obrigado!
Bom Dia!
Boa Tarde!
Boa Noite!

ã



ecossistema
ănimă