ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 14: Coordenação (parte 3)

Prof. Renan Alves

Escola de Artes, Ciências e Humanidades — EACH — USP

19/04/2024

Exclusão Mútua

Problema

Vários processos em um sistema distribuído desejam acesso exclusivo a algum recurso.

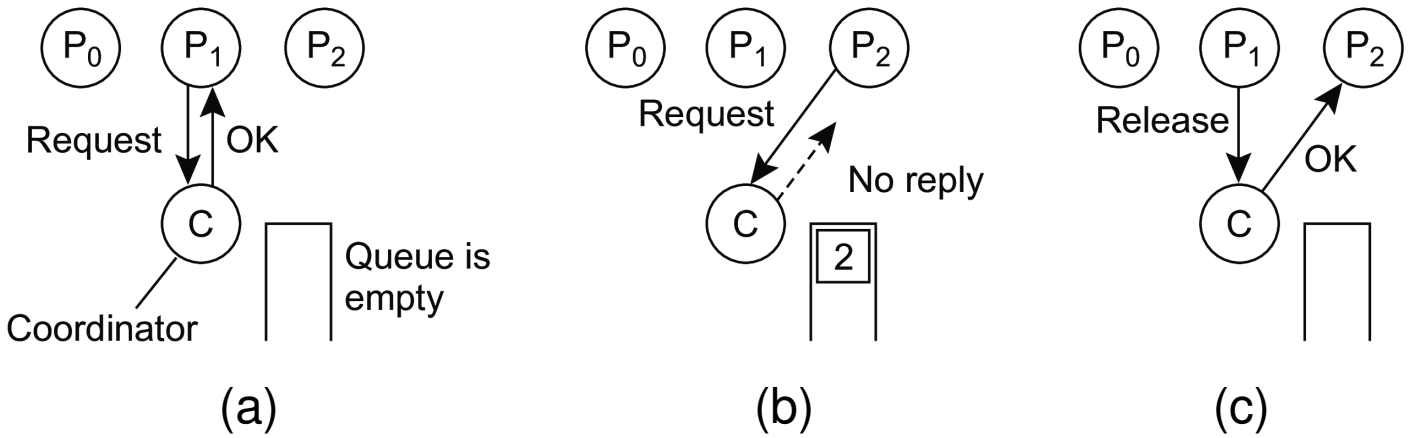
Soluções Básicas

Baseada em Permissão: Um processo que deseja entrar na sua região crítica, ou acessar um recurso, precisa ter permissão dos outros processos.

Baseada em Token: Um token é passado entre os processos. Aquele que possui o token pode prosseguir em sua região crítica, ou passá-lo adiante quando não estiver interessado.

Baseada em Permissão, centralizada

Simplesmente usar um coordenador



(a) O processo P_1 solicita ao coordenador permissão para acessar um recurso compartilhado. A permissão é concedida.

(b) Em seguida, o processo P_2 solicita permissão para acessar o mesmo recurso. O coordenador não responde.

(c) Quando P_1 liberar o recurso, ele informa ao coordenador, que então responde a P_2.

Baseada em Permissão, centralizada

Vantagens

- Exclusão mútua garantida
- Fairness
- Simplicidade de implementação
- Apenas 3 mensagens (REQUEST, OK, RELEASE)

Desvantagens

- Coordenador: único ponto de falha, possível gargalo
- Falha de coordenador indistinguível de espera

Exclusão mútua: Ricart & Agrawala

Semelhante ao Lamport, porém confirmações (ACK) não são enviadas

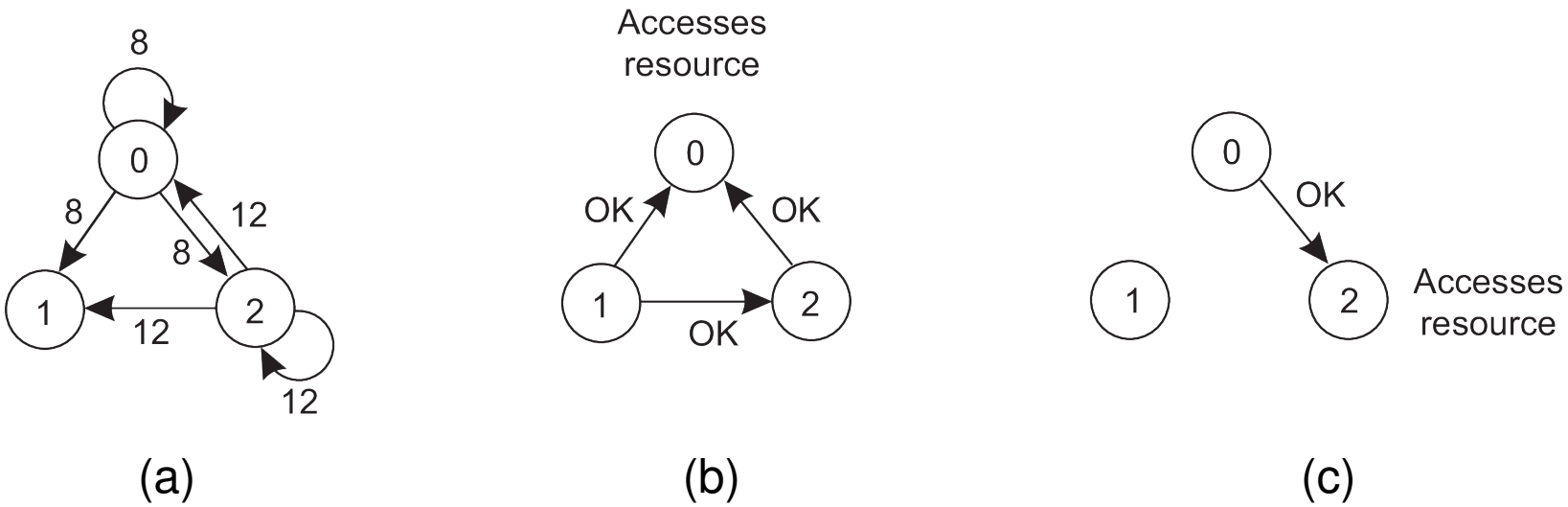
Uma mensagem de OK é enviada em resposta somente quando:

- O processo receptor não tem interesse no recurso compartilhado; ou
- O processo receptor está esperando pelo recurso, mas tem prioridade mais baixa (conhecida através da comparação de timestamps).

Em todos os outros casos (processo receptor já está usando o recurso ou tem uma requisição em aberto com timestamp menor), a resposta é adiada: processo receptor deve gerenciar localmente quando enviar o OK.

Exclusão mútua: Ricart & Agrawala

Exemplo com três processos



(a) Dois processos desejam acessar um recurso compartilhado no mesmo momento.

(b) P_0 possui o timestamp menor, então ele vence.

(c) Quando o processo P_0 termina, ele envia um *OK* para que P_2 possa prosseguir.

Exclusão mútua: Ricart & Agrawala

Análise

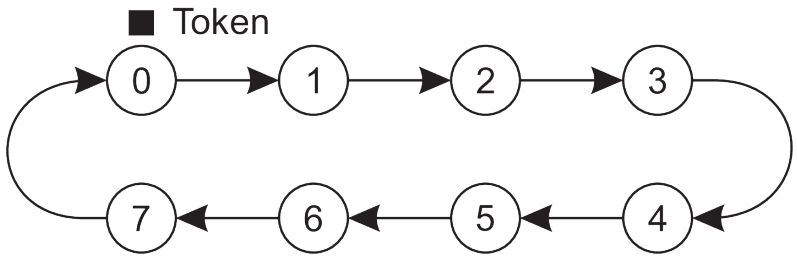
- Uma requisição para entrar na região crítica requer 2(N-1) mensagens
- Hipótese de conhecimento de todos os participantes
- Hipótese de entrega confiável de mensagens
- Todos os nós tornam-se pontos de falha
  - Mitigação:
  - Adicionar mensagem NOT-OK
  - Conseguir permissão da maioria (em vez da permissão de todos)

Exclusão mútua: Algoritmo de token ring

Essência

Organizar processos em um anel lógico e passar um token entre eles. Aquele que possuir o token pode entrar na região crítica (se desejar).

Uma rede overlay construída como um anel lógico com um token circulante



Exclusão mútua: Algoritmo de token ring

Análise

- É garantido que todos os processos terão a sua vez, porém pode haver tempo de espera
- Poucas mensagens são enviadas, porém, são enviadas mesmo que nenhum processo deseje utilizar o recurso
- Todos os nós são pontos de falha: é preciso adicionar mecanismos de detecção de falha

Exclusão mútua descentralizada

Princípio

Suponha que cada recurso seja replicado N vezes, com cada réplica tendo seu próprio coordenador $\Rightarrow$ o acesso requer um voto majoritário de $m > N/2$ coordenadores. Um coordenador sempre responde imediatamente a uma solicitação.

Hipótese

Quando um coordenador falha, ele se recupera rapidamente, mas esquece as permissões que concedeu.

Exclusão mútua descentralizada

Quão robusto este sistema é?

- Seja $p = \Delta t / T$ a probabilidade de um coordenador reiniciar durante um intervalo de tempo Δt, considerando que tem uma vida útil de T.

Exclusão mútua descentralizada

Quão robusto este sistema é?

- Seja $p = \Delta t / T$ a probabilidade de um coordenador reiniciar durante um intervalo de tempo Δt, considerando que tem uma vida útil de T.
- A probabilidade $\mathbb{P}[k]$ de k de m coordenadores reiniciarem durante o mesmo intervalo é:

$$\mathbb{P}[k] = \binom{m}{k} p^k (1-p)^{m-k}$$

Exclusão mútua descentralizada

Quão robusto este sistema é?

- Seja $p = \Delta t / T$ a probabilidade de um coordenador reiniciar durante um intervalo de tempo Δt, considerando que tem uma vida útil de T.
- A probabilidade $\mathbb{P}[k]$ de k de m coordenadores reiniciarem durante o mesmo intervalo é:

$$\mathbb{P}[k] = \binom{m}{k} p^k (1-p)^{m-k}$$

- f coordenadores reiniciar $\Rightarrow$ a corretude é violada quando há apenas uma minoria de coordenadores não defeituosos: quando $N - (m - f) \geq m$, ou, $f \geq 2m - N$.

Exclusão mútua descentralizada

Quão robusto este sistema é?

- Seja $p = \Delta t / T$ a probabilidade de um coordenador reiniciar durante um intervalo de tempo Δt, considerando que tem uma vida útil de T.
- A probabilidade $\mathbb{P}[k]$ de k de m coordenadores reiniciarem durante o mesmo intervalo é:

$$\mathbb{P}[k] = \binom{m}{k} p^k (1-p)^{m-k}$$

- f coordenadores reiniciar $\Rightarrow$ a corretude é violada quando há apenas uma minoria de coordenadores não defeituosos: quando $N - (m - f) \geq m$, ou, $f \geq 2m - N$.
- A probabilidade de uma violação é $\sum_{k=2m-N}^{m} \mathbb{P}[k]$.

Exclusão mútua descentralizada

Probabilidades de violação para vários valores de parâmetros

N	m	p	Violação
8	5	3 s/hora	$< 10^{-5}$
8	6	3 s/hora	$< 10^{-11}$
16	9	3 s/hora	$< 10^{-4}$
16	12	3 s/hora	$< 10^{-21}$
32	17	3 s/hora	$< 10^{-4}$
32	24	3 s/hora	$< 10^{-43}$

N	m	p	Violação
8	5	30 s/hora	$< 10^{-3}$
8	6	30 s/hora	$< 10^{-7}$
16	9	30 s/hora	$< 10^{-2}$
16	12	30 s/hora	$< 10^{-13}$
32	17	30 s/hora	$< 10^{-2}$
32	24	30 s/hora	$< 10^{-27}$

Exclusão mútua descentralizada

Análise

- É possível ter um sistema robusto a falhas
- No caso de não ser possível obter o recurso, backoff e tentar novamente
  - Se muitos nós estiverem tentando e atrasos forem grandes, pode levar a starvation

Exclusão mútua: comparação

Algoritmo	**Mensagens por entrada/saída**	**Atraso antes da entrada (em tempos de mensagem)**
Centralizado	3	2
Ricart & Agrawala	$2(N-1)$	$2(N-1)$
Anel de token	$1,\ldots,\infty$	$0,\ldots,N-1$
Descentralizado	$2kN+(k-1)N/2+N, k=1,2,\ldots$	$2kN+(k-1)N/2$

Exclusão mútua: comparação

Algoritmo	**Mensagens por entrada/saída**	**Atraso antes da entrada (em tempos de mensagem)**
Centralizado	3	2
Ricart & Agrawala	$2(N-1)$	$2(N-1)$
Anel de token	$1,\ldots,\infty$	$0,\ldots,N-1$
Descentralizado	$2kN+(k-1)N/2+N, k=1,2,\ldots$	$2kN+(k-1)N/2$

Tolerância a falhas

- Ricart & Agrawala e token ring: falha de qualquer nó traz problemas
- Descentralizado: falhas transientes são toleradas com dada probabilidade
- Centralizado: coordenador ponto único de falha, porém fornece ponto de foco para melhorar a robustez

Exemplo: ZooKeeper

Noções básicas

- Configuração centralizada do servidor
- Toda comunicação cliente-servidor é não-bloqueante: um cliente recebe imediatamente uma resposta
- O ZooKeeper mantém um espaço de nomes baseado em árvore, semelhante a um sistema de arquivos
- Os clientes podem criar, excluir ou atualizar nós, além de verificar a existência.

Condição de corrida no ZooKeeper

Nota

O ZooKeeper permite que um cliente seja notificado quando um nó ou um ramo na árvore muda. Isso pode facilmente levar a condições de corrida.

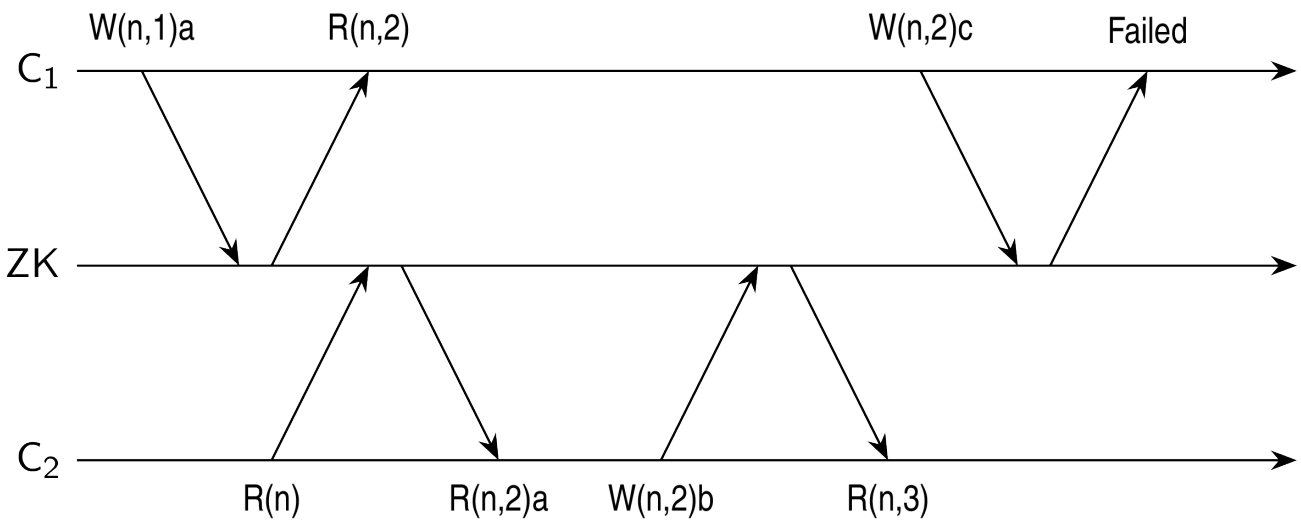
Considere um mecanismo de trava (lock) simples

1. Um cliente C_1 cria um nó $/lock$.
2. Um cliente C_2 deseja adquirir a trava, mas é notificado de que o nó associado já existe.
3. Antes que C_2 se inscreva para uma notificação, C_1 libera a trava, ou seja, exclui $/lock$.
4. O cliente C_2 se inscreve para alterações em $/lock$ e bloqueia localmente.

Solução

Usar números de versão

Versionamento no ZooKeeper



Notação

- $W(n,k)a$: solicitação para escrever a no nó n, assumindo que a versão atual é k.
- $R(n,k)$: versão atual do nó n é k.
- $R(n)$: cliente deseja saber o valor atual do nó n.
- $R(n,k)a$: valor a do nó n é retornado com sua versão atual k.

Protocolo de lock do ZooKeeper

Ficou simples:

1. bloquear: Um cliente C_1 cria um nó $/lock$.
2. bloquear: Um cliente C_2 deseja adquirir a trava, mas é notificado de que o nó associado já existe $\Rightarrow$ C_2 se inscreve para notificação sobre alterações de $/lock$.
3. desbloquear: O cliente C_1 exclui o nó $/lock$ $\Rightarrow$ todos os inscritos para alterações são notificados.

Algoritmos de eleição

Princípio

Um algoritmo requer que algum processo atue como coordenador. A questão é como selecionar este processo especial dinamicamente (i.e. em tempo de execução).

Nota

Em muitos sistemas, o coordenador é escolhido manualmente (por exemplo, servidores de arquivos). Isso leva a soluções centralizadas $\Rightarrow$ ponto único de falha.

Algoritmos de eleição

Princípio

Um algoritmo requer que algum processo atue como coordenador. A questão é como selecionar este processo especial dinamicamente (i.e. em tempo de execução).

Nota

Em muitos sistemas, o coordenador é escolhido manualmente (por exemplo, servidores de arquivos). Isso leva a soluções centralizadas $\Rightarrow$ ponto único de falha.

Questões

1. Se um coordenador é escolhido dinamicamente, até que ponto podemos falar sobre uma solução centralizada ou distribuída?
2. Uma solução totalmente distribuída, ou seja, sem coordenador, é sempre mais robusta do que qualquer solução centralizada/coordenada?

Hipóteses básicas

- Todos os processos têm IDs únicos
- Todos os processos conhecem os IDs de todos os processos no sistema (mas não sabem se estão ativos ou inativos)
- Eleição significa identificar o processo com o maior ID que está ativo

Eleição por intimidação

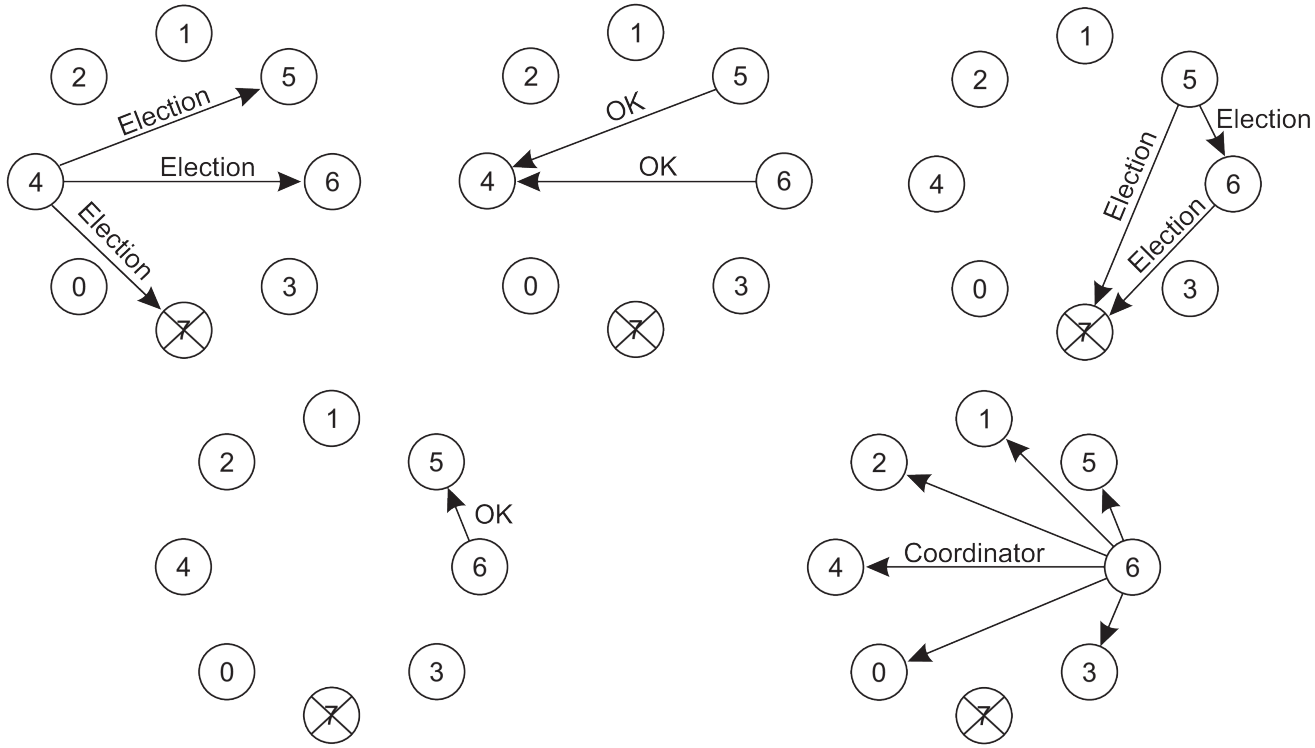
Princípio

Considere N processos $\{P_0, \ldots, P_{N-1}\}$ com $id(P_k) = k$. Quando um processo P_k percebe que o coordenador não está mais respondendo às solicitações, ele inicia uma eleição:

1. P_k envia uma mensagem *ELECTION* para todos os processos com identificadores mais altos: $P_{k+1}, P_{k+2}, \ldots, P_{N-1}$.
2. Se ninguém responder, P_k vence a eleição e se torna o coordenador.
3. Se um dos superiores responder, ele assume o controle e P_k não faz mais nada.

Eleição por intimidação

O algoritmo de eleição do valentão



(a) Processo 4 inicia eleição
(b) Processos 5 e 6 respondem, dizendo para 4 parar
(c) É a vez de 5 e 6 iniciarem uma eleição.
(d) Processo 6 diz para 5 parar.
(e) Processo 6 vence e comunica para todos o resultado.

Eleição em um anel

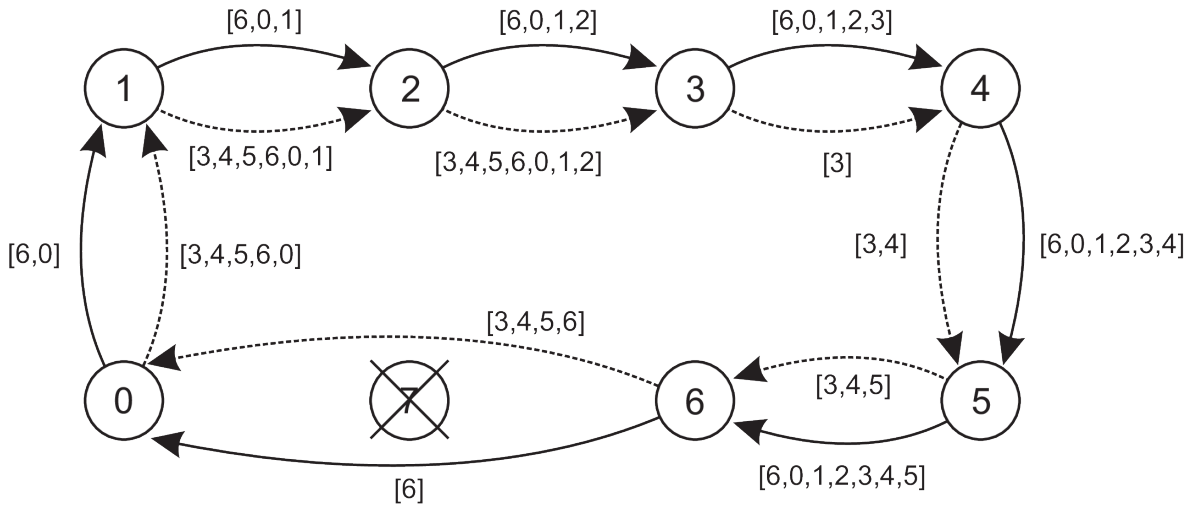
Princípio

A prioridade do processo é obtida organizando os processos em um anel (lógico). O processo com a maior prioridade deve ser eleito como coordenador.

- Qualquer processo pode iniciar uma eleição enviando uma mensagem de eleição ao seu sucessor. Se um sucessor estiver inativo, a mensagem é repassada para o próximo sucessor.
- Se uma mensagem for repassada, o remetente se adiciona à lista. Quando voltar para o iniciador, todos tiveram a chance de manifestar sua presença.
- O iniciador envia uma mensagem de coordenador ao redor do anel contendo uma lista de todos os processos ativos. Aquele com a maior prioridade é eleito como coordenador.

Eleição em um anel

Algoritmo de eleição usando um anel



- A linha sólida mostra as mensagens de eleição iniciadas por P_6
- A tracejada, as mensagens por P_3