Computação Distribuída

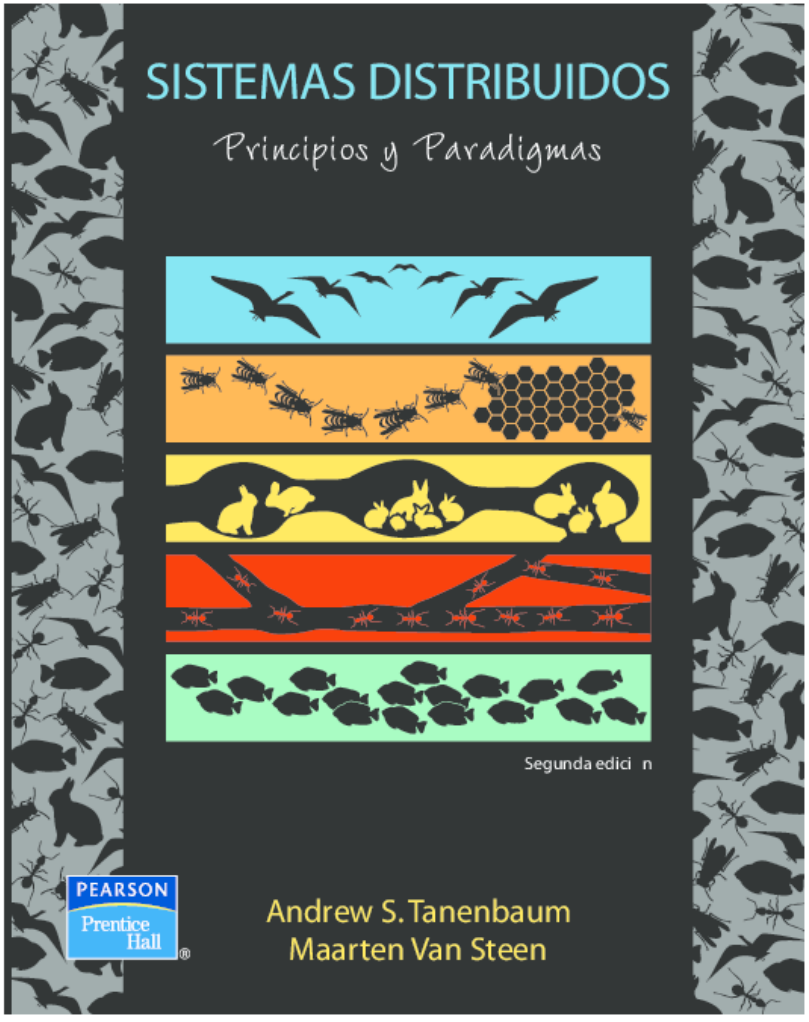
Exclusão Mútua e Eleição

CHAPTER 6

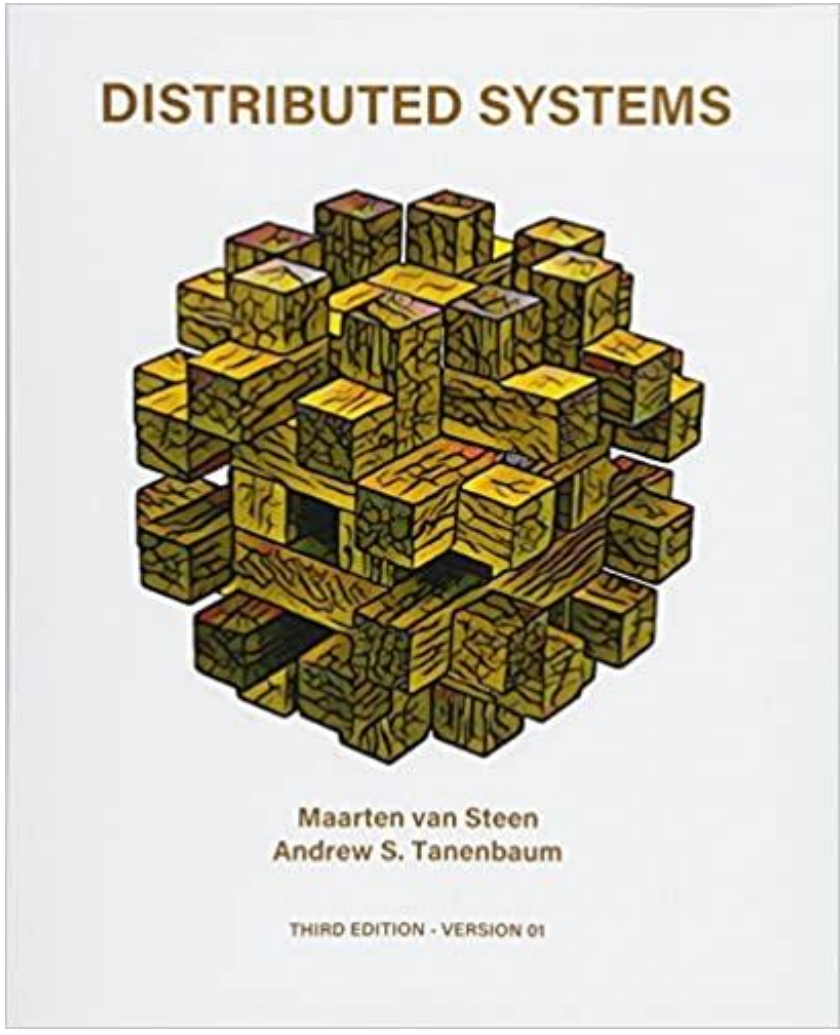
COORDINATION

Vladimir Rocha (Vladi)

CMCC - Universidade Federal do ABC


SISTEMAS DISTRIBUIDOS
Principios y Paradigmas
Segunda edici n
PEARSON
Prentice
Hall
Andrew S. Tanenbaum
Maarten Van Steen

2007


DISTRIBUTED SYSTEMS
Maarten van Steen
Andrew S. Tanenbaum
THIRD EDITION - VERSION 01

2017

Disclaimer

- Estes slides foram baseados nos do professor **Emilio Francesquini** para o curso de Sistemas Distribuídos na UFABC.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.
- Estes slides foram adaptados daqueles originalmente preparados (e gentilmente cedidos) pelo professor **Daniel Cordeiro, da EACH-USP** que por sua vez foram baseados naqueles disponibilizados online pelos autores do livro "Distributed Systems", 3ª Edição em: https://www.distributed-systems.net.

Agenda

- Exclusão Mútua (acesso exclusivo)
- Eleição (escolha de um líder)

Agenda

- Exclusão Mútua (acesso exclusivo)
  - . centralizado
  - . distribuído
  - . token
  - . descentralizado

Exclusão mútua

Problema

Alguns processos em um sistema distribuído querem acesso exclusivo a algum recurso.

Soluções:

Baseado em permissão: um processo que quiser entrar na seção crítica (ou acessar um recurso) precisa da permissão de outros processos

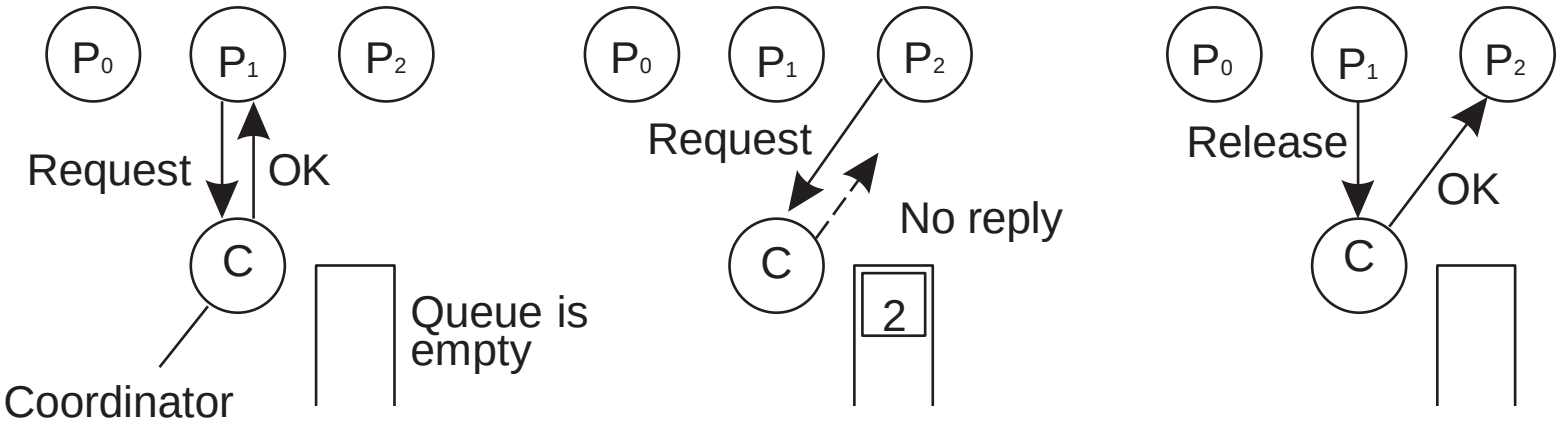
Baseado em tokens: um *token* é passado entre processos. Aquele que tiver o *token* pode entrar na seção crítica ou passá-lo para frente quando não estiver interessado.

Agenda

- Exclusão Mútua
  - centralizado
  - distribuído
  - token
  - descentralizado

Exclusão mútua: permissão, centralizado

Use um coordenador



(a) Processo P_1 pede permissão ao coordenador para acessar o recurso compartilhado. Permissão concedida.

(b) Processo P_2 então pede permissão para acessar o mesmo recurso. O coordenador não responde.

(c) Quando P_1 libera o recurso, avisa o coordenador, que então responde para P_2.

Agenda

- Exclusão Mútua
  - centralizado
  - distribuído
  - token
  - descentralizado

Exclusão mútua: permissão, distribuído

Ricart & Agrawala [1981]

Princípio

O processo que precisa do recurso envia uma requisição de permissão a todos os outros processos (inclusive para ele mesmo).

A resposta à permissão (denominada de ACK) é enviada quando:

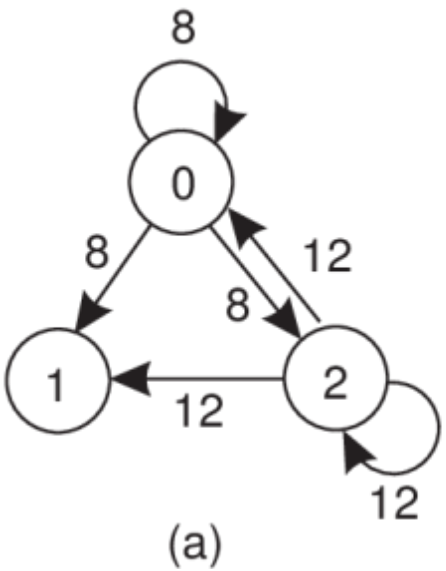
- o processo receptor não tem interesse no recurso compartilhado; ou
- o processo receptor está esperando por um recurso, mas tem menos prioridade (a prioridade é determinada via comparação de timestamps*)

Em todos os outros casos, o envio da resposta é adiado.

* *timestamp* é o horário do relógio ou um valor (e.g., contador).

Exclusão mútua: permissão, distribuído

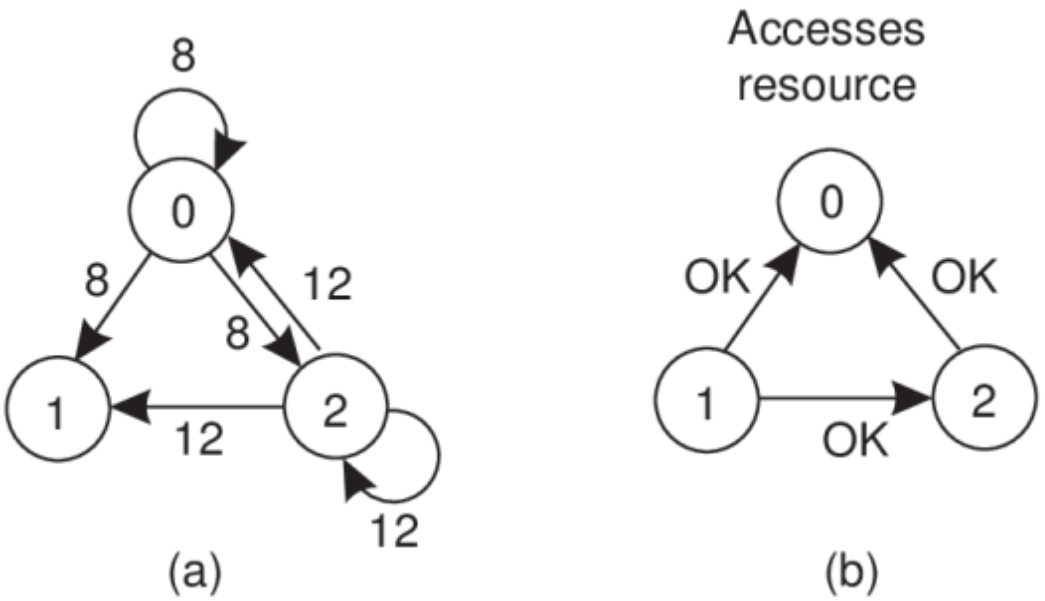
Exemplo com três processos:



(a) dois processos (P_0 e P_2) querem acessar um recurso compartilhado ao mesmo tempo, mas só um pode

Exclusão mútua: permissão, distribuído

Exemplo com três processos:

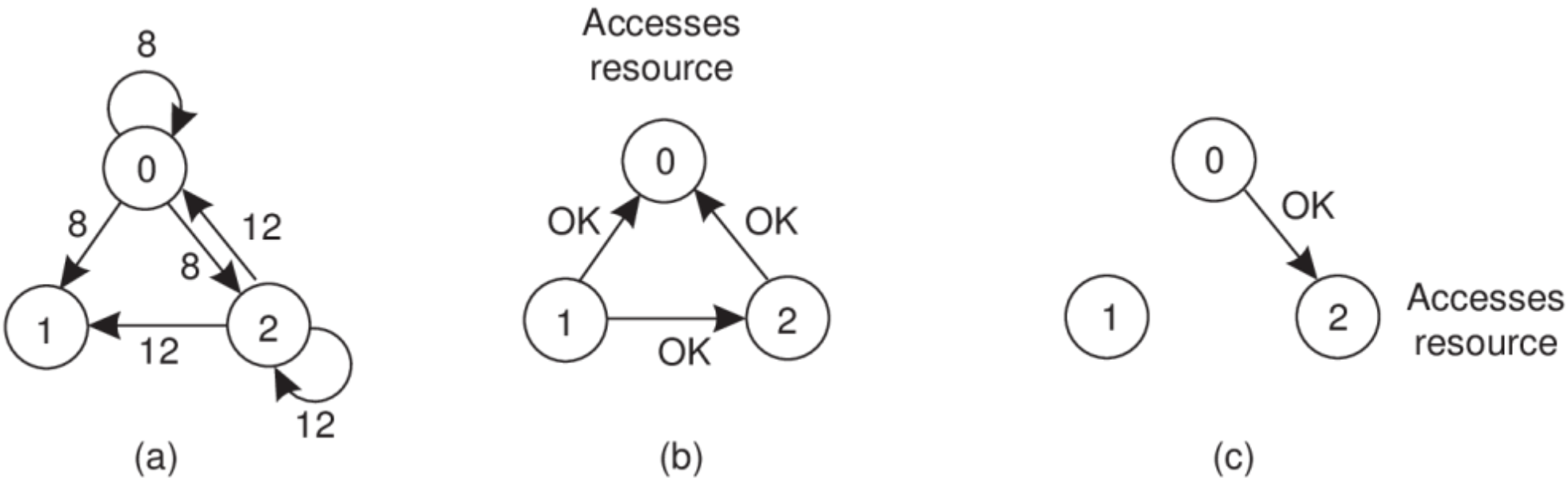


(a) dois processos (P_0 e P_2) querem acessar um recurso compartilhado ao mesmo tempo, mas só um pode

(b) P_0 tem o menor *timestamp*; ele ganha

Exclusão mútua: permissão, distribuído

Exemplo com três processos:



(a) dois processos (P_0 e P_2) querem acessar um recurso compartilhado ao mesmo tempo, mas só um pode

(b) P_0 tem o menor *timestamp*; ele ganha

(c) quando P_0 terminar, envia os OK adiados; assim P_2 agora pode continuar

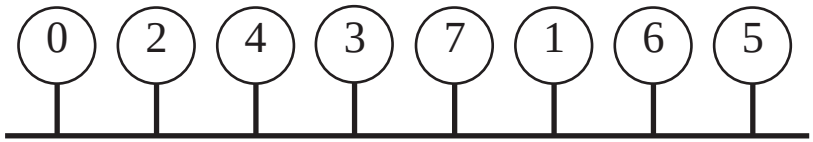
Agenda

- Exclusão Mútua
  - centralizado
  - distribuído
  - token
  - descentralizado

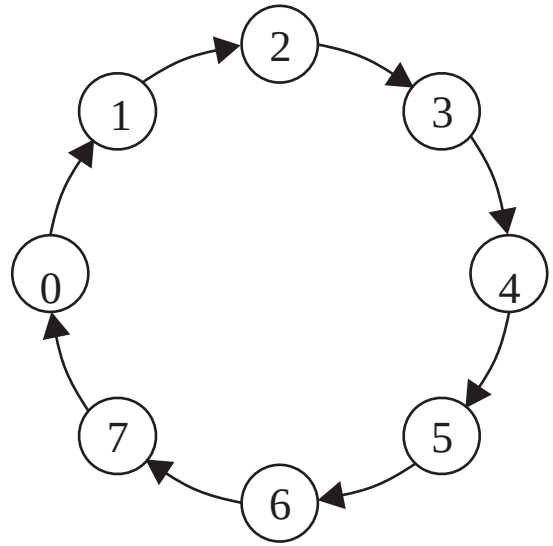
Exclusão mútua: token ring

Ideia

Organizar os processos em anel lógico e passar um *token* entre eles. Aquele que estiver com o *token* pode entrar na seção crítica (se ele quiser).



(a)



(b)

Agenda

- Exclusão Mútua
  - centralizado
  - distribuído
  - token
  - descentralizado

Exclusão mútua: descentralizada

Princípio

Assuma que todo recurso é replicado N vezes (para não ter SPoF, i.e., maior disponibilidade).
Cada replica está associada a seu próprio coordenador [Lin 2004]

$\Rightarrow$ acesso requer a maioria dos votos de $m > N/2$ coordenadores.

Hipótese

Quando um coordenador morrer, ele se recuperará rapidamente, mas terá esquecido tudo sobre as permissões que ele deu.

Risco? Após a recuperação, dar incorretamente a permissão de aceso para um processo

Exclusão mútua: descentralizada

Quão robusto é esse sistema?

Seja $p = \Delta t/T$ a probabilidade de que um coordenador morra e se recupere (reset) em um período Δt e que tenha uma esperança de vida T. Por exemplo, $\Delta t \sim 2s$, $T \sim 2.7h$.

A probabilidade P[k] de que k dos m coordenadores sejam resetados, i.e., morra e volte, durante o mesmo intervalo é:

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

Lembre que m corresponde a uma maioria. No caso, só serve como parâmetro que toma valores entre (N/2, N]

Exclusão mútua: descentralizada

Quão robusto é esse sistema?

Seja $p = \Delta t/T$ a probabilidade do reset do coordenador.

Seja P[k] a probabilidade de que k dos m coordenadores reset:

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

Assuma que f dos m coordenadores resetam.

Corretude é violada quando $m - f$ coordenadores (corretos, sem reset) são minoria

- I.e., existe uma maioria dando a exclusão mútua para outro.

Isso acontece quando: $m - f \leq N/2$ ou $f \geq m - N/2$

Assim, devemos analisar o comportamento de f quando há violação

A probabilidade de violação é $\sum_{k=m-N/2}^{N} \mathbb{P}[k]$

Exclusão mútua: descentralizada

$p = \Delta t/T$, a probabilidade do reset do coordenador.
N, total de coordenadores.
m, maioria

Probabilidade de violação em função dos parâmetros

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

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

Exclusão mútua: comparação

Ver no Tanenbaum 3ed.

Algorítimo	# msgs por entrada/saída	Atraso para entrar (em qde msgs)	Problemas
Centralizado	3	2	Morte do coordenador
Descentralizado	$2mk + m$, $k = 1,2,\ldots$	$2mk$	*Starvation*, ineficiente.
Distribuído	$2\,(n-1)$	$2\,(n-1)$	Morte de qualquer
Token ring	1 à ∞	0 à $n-1$	Perder token, proc. morre

Agenda

- Exclusão Mútua
- Eleição
  - Anel
  - Bully

Eleição: princípio

Um algoritmo precisa que algum dos processos assuma o papel de coordenador. A pergunta é: como selecionar esse processo especial dinamicamente?

Nota

Em muitos sistemas o coordenador é escolhido manualmente (ex: servidores de dados). Isso leva a soluções centralizadas com um ponto único de falha – SPoF.

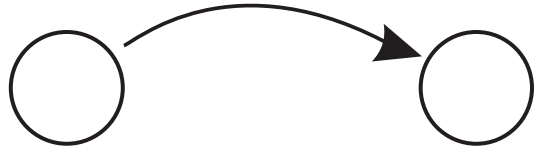
Perguntas

1. Se um coordenador é escolhido dinamicamente, até que ponto podemos dizer que o sistema será centralizado e não distribuído?
2. Um sistema inteiramente distribuído (ou seja, um sem um coordenador) é sempre mais robusto que uma solução centralizada/coordenada?

Eleição: Hipóteses básicas

- Todos os processos possuem um id único
- Todos os processos conhecem os ids de todos os outros processos no sistema (mas eles não têm como saber se os nós estão funcionando ou não)
- A **eleição significa identificar o processo de maior id** que está funcionando em um dado momento

Eleição: anel

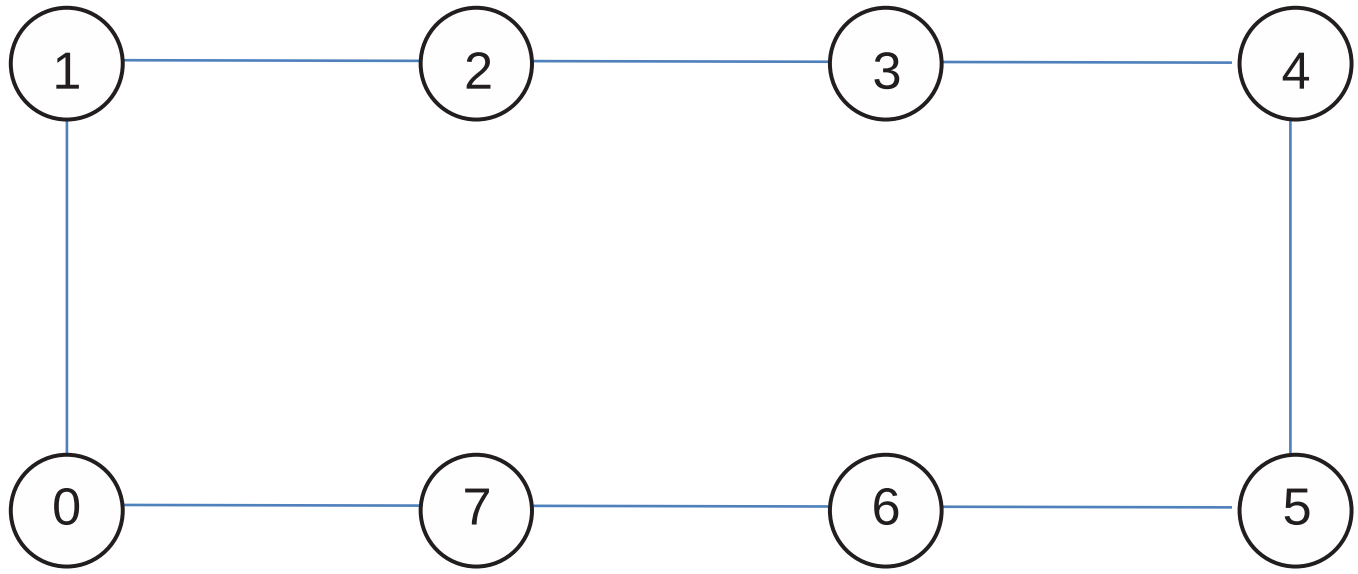


Princípio

As prioridades –identificadores– dos processos são obtidas organizando-os em um anel (lógico). O processo com prioridade mais alta deve ser eleito como coordenador.

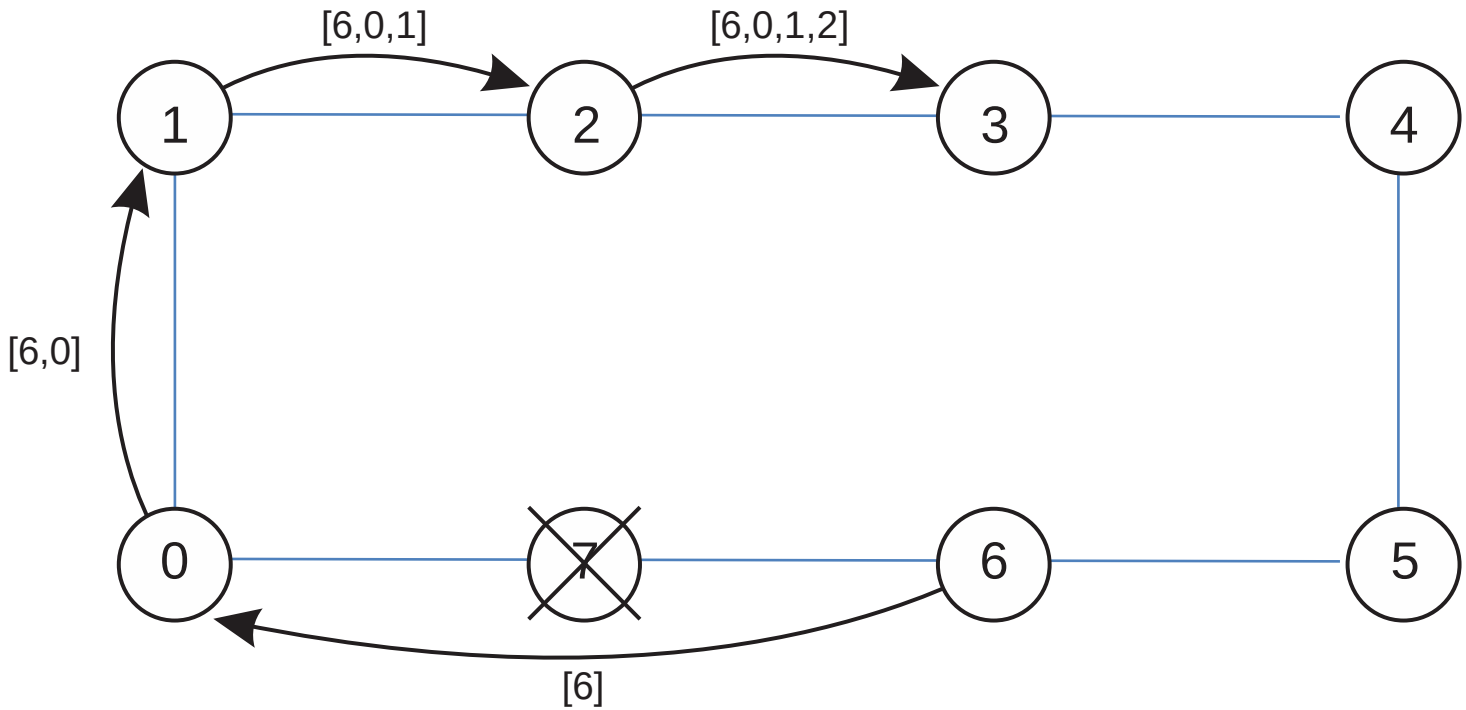
- Qualquer processo pode iniciar a eleição ao enviar uma mensagem de eleição ao seu sucessor. Se um sucessor estiver indisponível, a mensagem é enviada ao próximo successor.
- Se uma mensagem for repassada, o remetente se adiciona na lista. Quando a mensagem voltar ao nó que iniciou, todos tiveram a chance de anunciar a sua presença.
- O nó que iniciou circula uma mensagem pelo anel com a lista de nós “vivos”. O processo com maior prioridade é eleito coordenador.

Eleição: anel



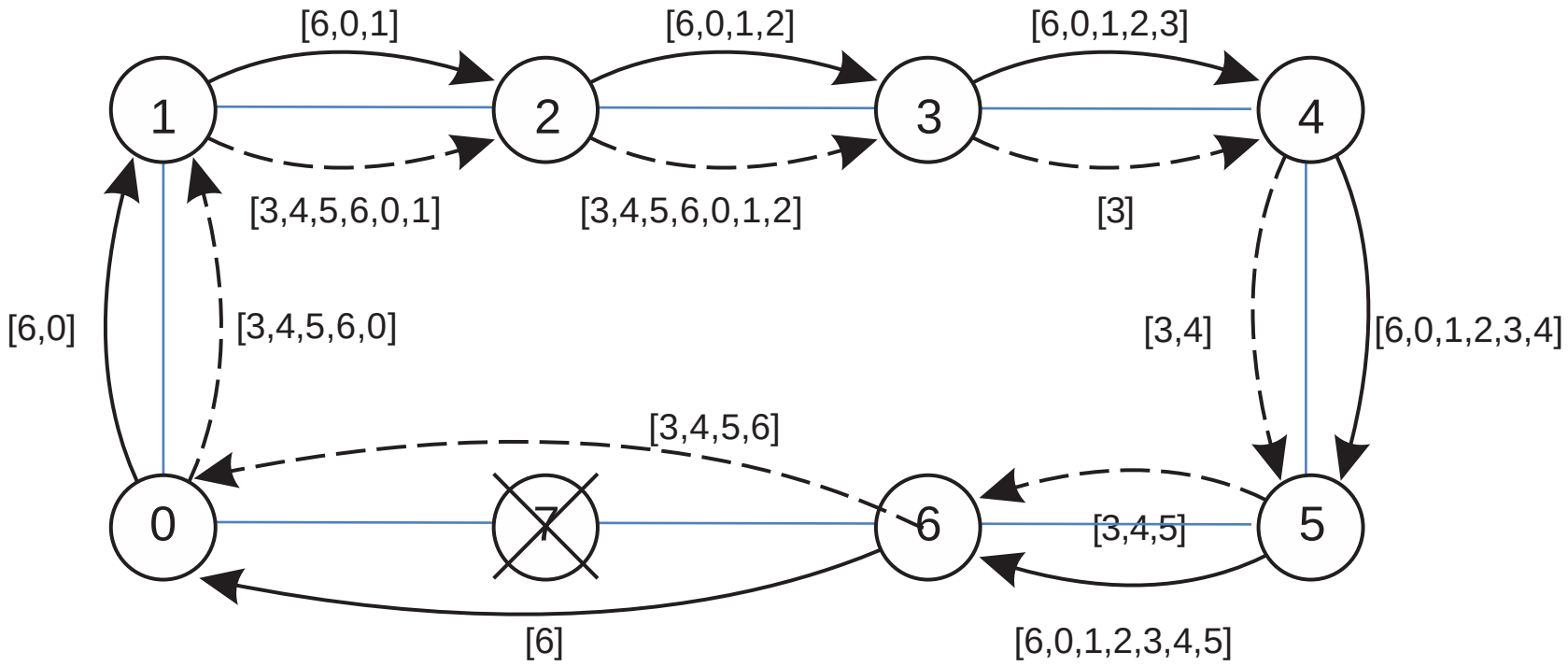
Link lógico para o seguinte processo

Eleição: anel



- As linhas contíguas mostram as mensagens da eleição iniciada por P_6

Eleição: anel



- As linhas contíguas mostram as mensagens da eleição iniciada por P_6
- As linhas pontilhadas se referem a eleição iniciada por P_3

Pergunta: Supondo que só P_3 iniciasse a eleição. Como P_4 saberia que P_6 é o coordenador?

Eleição: “Bully”

Princípio [Garcia-Molina 1982]

- os N processos se conhecem
- ganha o maior id

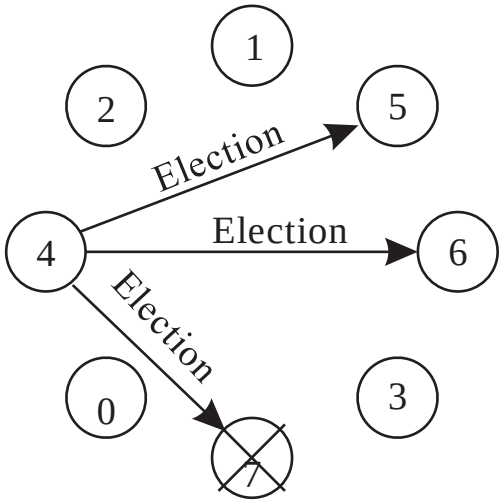
Considere N processos $\{P_0,..., P_{N-1}\}$ e seja $id(P_k) = k$. Quando qualquer processo P_k perceber que o coordenador não está mais respondendo às requisições, ele começa uma nova eleição:

1. P_k envia uma mensagem ELECTION para todos os processos com identificadores maiores que o seu: P_{k+1}, P_{k+2},..., P_{N-1}.
2. Se ninguém responder, P_k ganha a eleição e se torna o coordenador
3. Se um dos nós com maior id responder, esse assume a eleição e o trabalho de P_k termina[2].
4. O coordenador escolhido envia COORDINATOR a todos

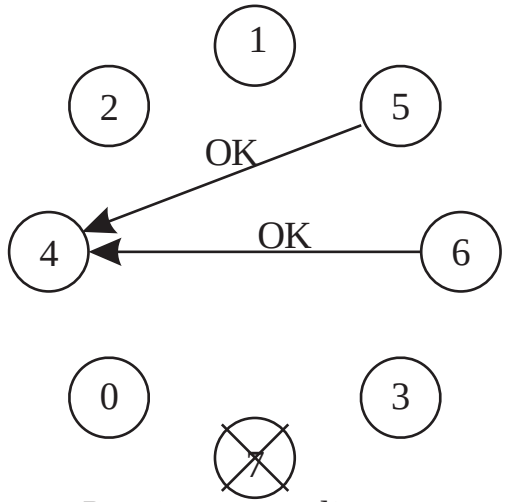
---

2 O maior sempre ganha, por isso o nome de *Bully* “valentão”.

Eleição: “Bully”

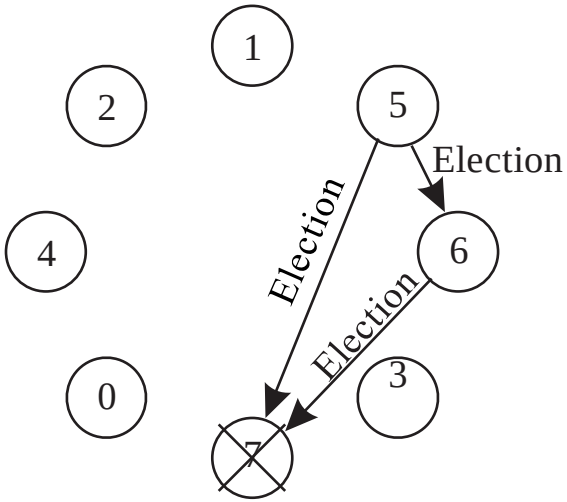


(a)

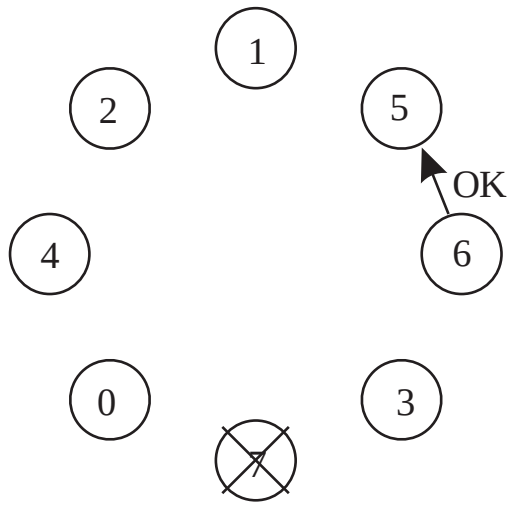


Previous coordinator has crashed

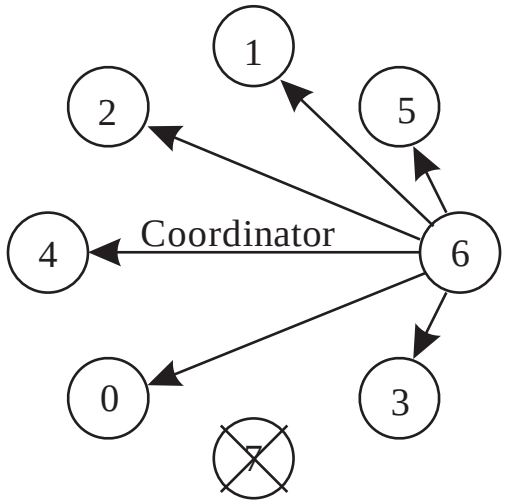
(b)



(c)

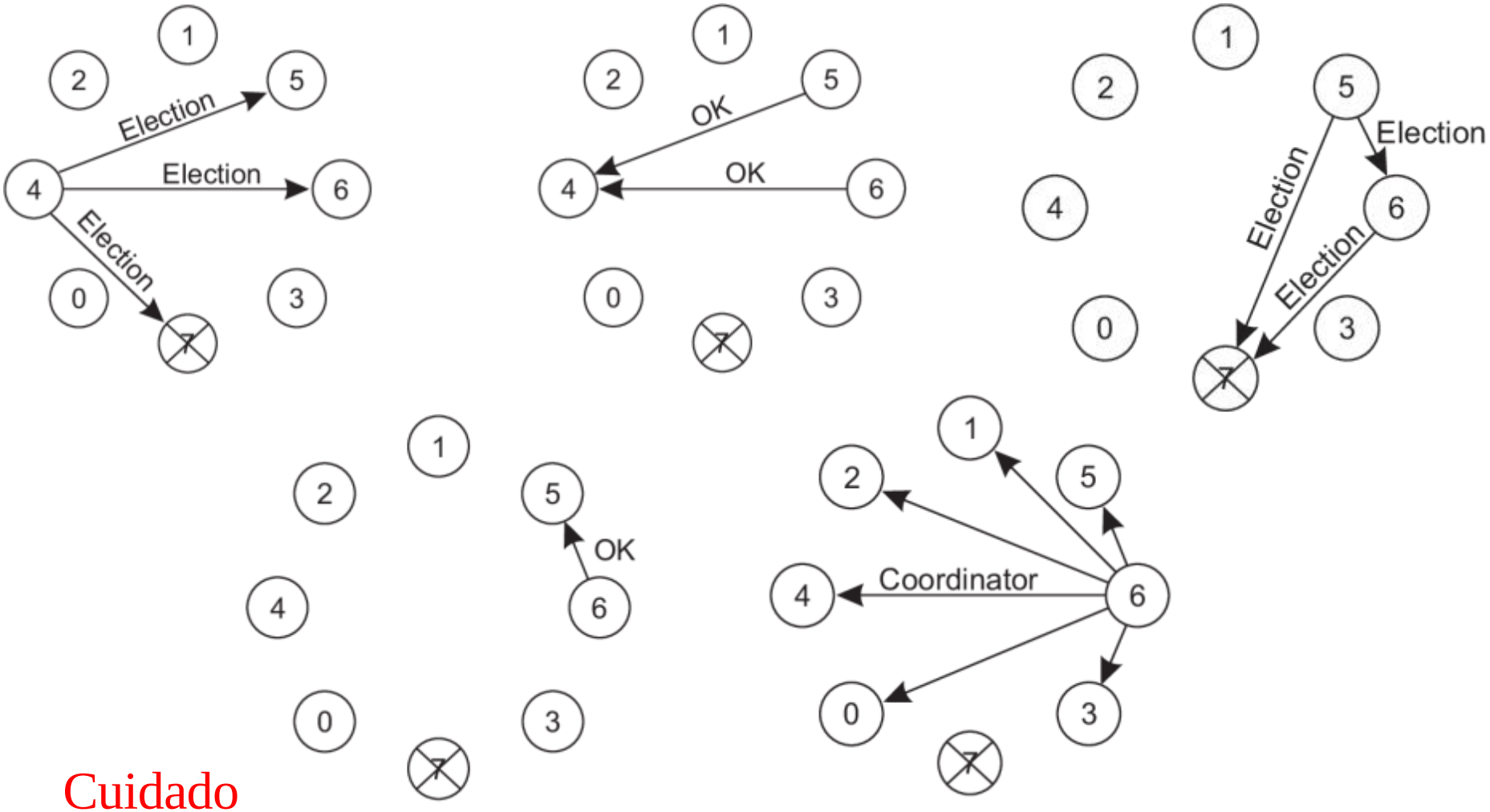


(d)



(e)

Eleição: “Bully”



Cuidado

Estamos assumido algo importante sobre a comunicação. O quê?

Assumimos que a comunicação é confiável

Perguntas:

- Posso usar eleição para exclusão mútua? Como?
- Posso usar exclusão mútua para eleição? Como?
- Se os algoritmos apresentados não são usados na prática, o que fazer?

Apache ZooKeeper™

Capítulo 8

Conceitos adquiridos

- Exclusão mútua:
    - centralizado, distribuído ricart & agrawala, token, decentralizado.
- Eleição: anel, bully.