SINCRONIZAÇÃO NOS SISTEMAS DISTRIBUÍDOS

PROF. DR. SÉRGIO CARLOS PORTARI JÚNIOR

PARTE II - ELEIÇÃO E COORDENAÇÃO

Muitos algoritmos requerem um processo para atuar como coordenador.

- Exemplo: Coordenador no algoritmo centralizado de exclusão mútua.
- Em geral, não importa qual processo irá atuar como coordenador, mas é preciso elegê-lo.

Como eleger um coordenador?

PARTE II – ELEIÇÃO E COORDENAÇÃO

Premissas:

- Cada processo possui um número único (por exemplo, seu endereço de rede) separado.

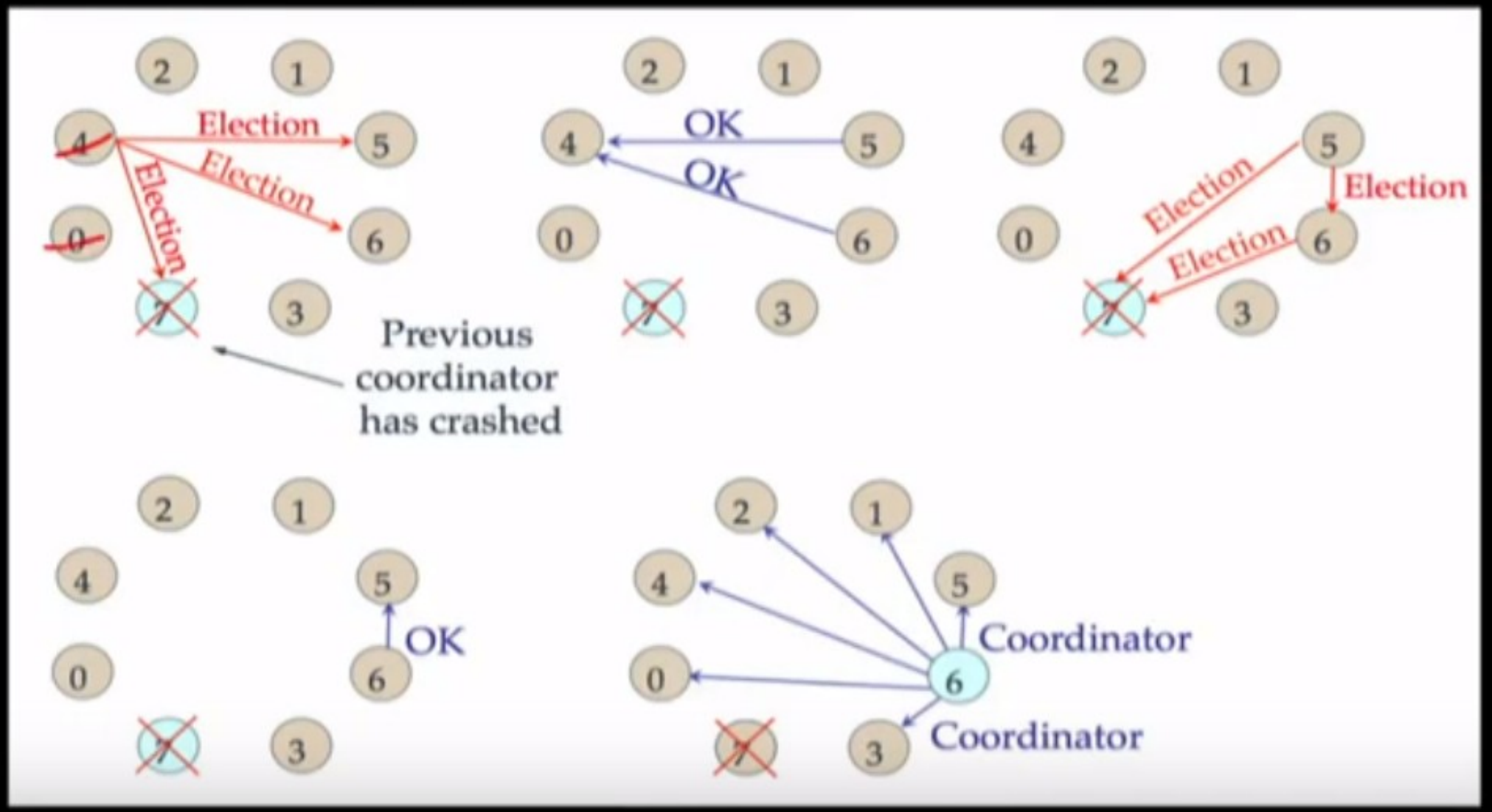
- Todo processo conhece o número do processo de qualquer outro processo

- Os processos não sabem quais processos estão “vivos” e quais estão “mortos”.

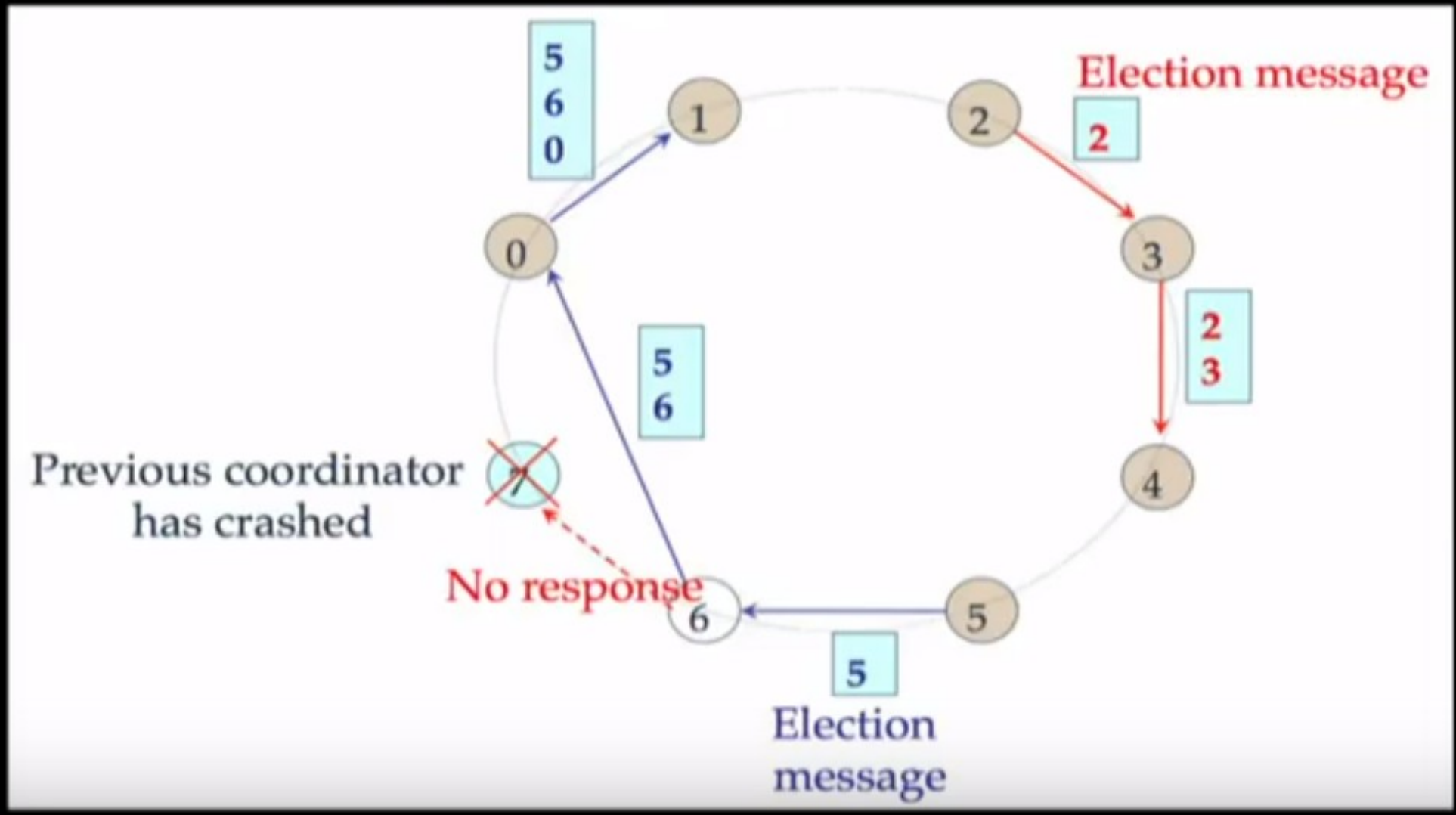
PROBLEMA DA SINCRONIZAÇÃO EM SD

- Abordagem:
  - Localize o processo com o número de processo maior e eleja-o como coordenador.
  - Os algoritmos de eleição diferem na forma com fazem esta localização.

ALGORITMO DO VALENTÃO (BULLYNG)



ALGORITMO DO ANEL



PARTE III – EXCLUSÃO MÚTUA EM SD

- A melhor maneira de se conseguir exclusão mútua em um SD é imitar o que é feito em um sistema local.
- 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 dá o OK.

ALGORITMO CENTRALIZADO

- Processo 1 pede permissão ao coordenador para entrar em uma região crítica. A permissão é concedida
- Processo 2 então pede permissão para entrar na mesma região crítica. O coordenador não responde.
- Quando o processo 1 sai da região crítica, ele avisa o coordenador, quando então responde a 2.

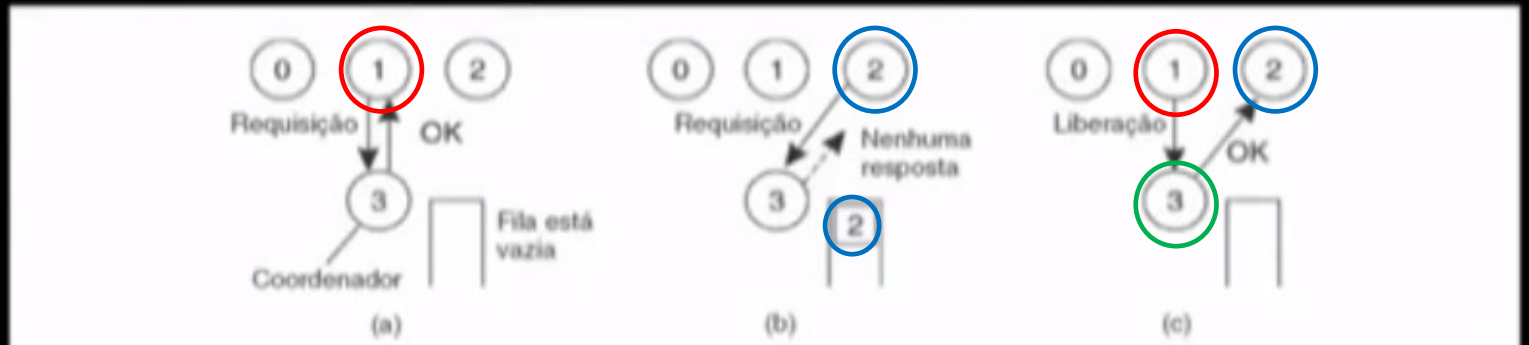


Figura 6.14 (a) O processo 1 solicita ao coordenador permissão para acessar um recurso compartilhado. A permissão é concedida. (b) Depois, o processo 2 solicita permissão para acessar o mesmo recurso. O coordenador não responde. (c) Quando o processo 1 libera o recurso, informa ao coordenador, que então responde a 2.

ALGORITMO CENTRALIZADO - VANTAGENS

- Obviamente garante exclusão mútua.
- É justo (os pedidos são concedidos na ordem em que são recebidos).
- Sem inanição (um processo não espera para sempre).
- Fácil de implementar (apenas 3 mensagens: solicitação, concessão e liberação).

ALGORITMO CENTRALIZADO - DESVANTAGENS

- Coordenador: um único ponto de falha.
- Traz um gargalo no desempenho
- Se os processos normalmente bloqueiam após fazer um pedido, eles não podem distinguir um coordenador morto da "permissão negada".

ALGORITMO DISTRIBUÍDO

a) Dois processos querem entrar na mesma região crítica ao mesmo tempo.

b) Processo 0 tem o timestamp mais baixo, então ganha.

c) Quando o processo 0 finaliza, ele envia um OK, então o processo 2 agora pode entrar na região crítica.

Problema: falha de qq

Processo "crashes".

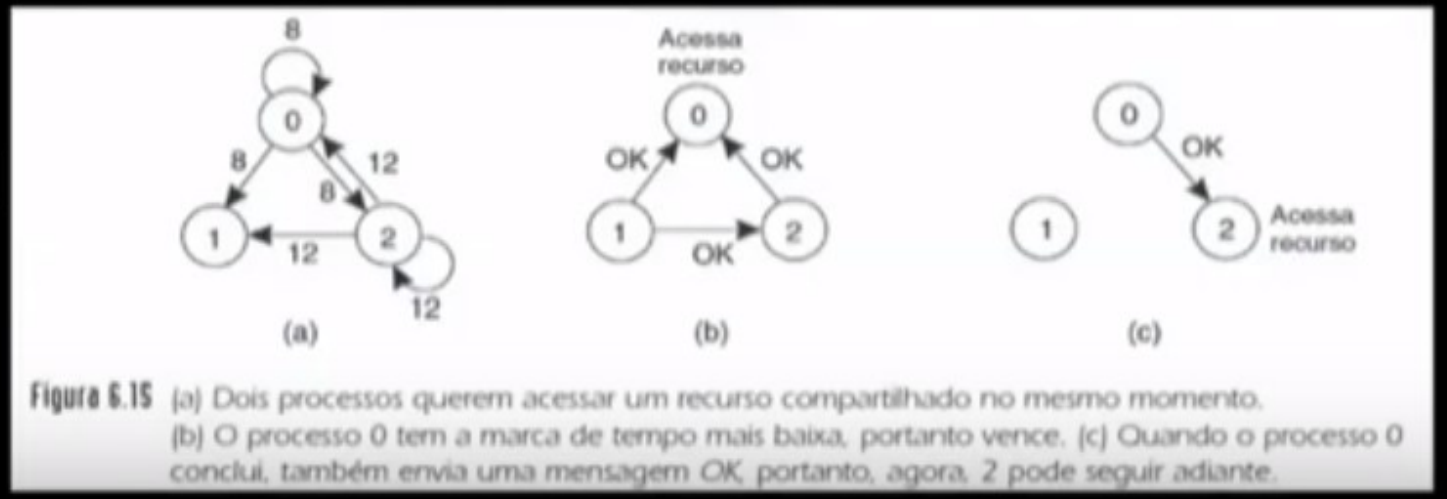
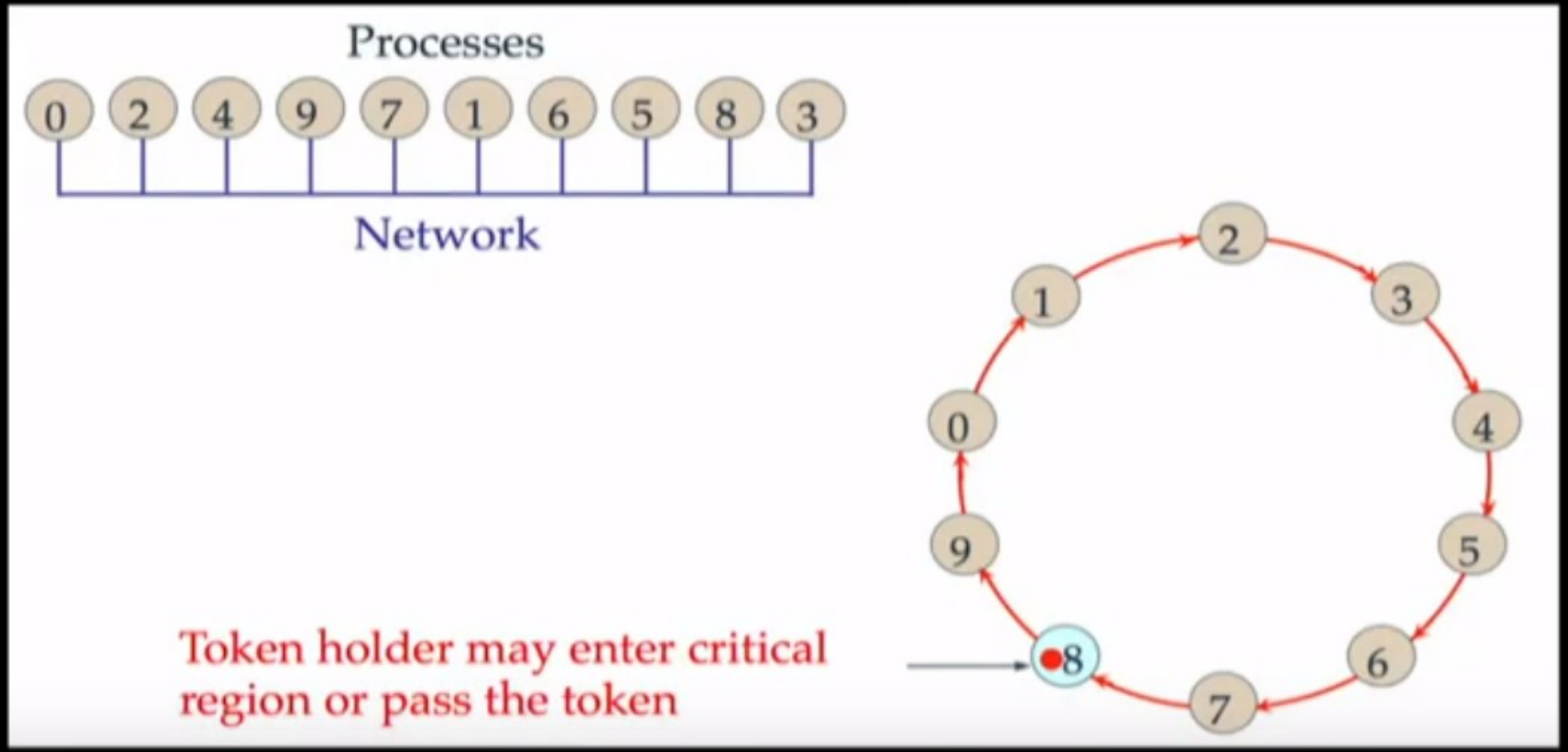


Figura 6.15 (a) Dois processos querem acessar um recurso compartilhado no mesmo momento. (b) O processo 0 tem a marca de tempo mais baixa, portanto vence. (c) Quando o processo 0 conclui, também envia uma mensagem *OK*, portanto, agora, 2 pode seguir adiante.

ALGORITMO DISTRIBUÍDO



ALGORITMO DISTRIBUÍDO

Vantagens:

- Garante a exclusão mútua (apenas 1 processo possui o token em qualquer instante).
- Sem inanição (o token circula entre os processos em uma ordem bem definida).

Desvantagens:

- Regeneração do token se ele for perdido.
- Detectar que o token está perdido é difícil – rede pode estar sobrecarregada.