Sistemas Distribuídos

Exclusão Mútua

Edeyson Andrade Gomes

www.edeyson.com.br

Roteiro da Aula

- Introdução
  - Coordenação e Acordo
  - Suposição de Falhas
- Exclusão Mútua
- Algoritmos
  - Centralizado
  - Distribuído
  - Anel
- Comparações

Coordenação e Acordo

Coordenação e Acordo

- Sistema Síncrono
  - Suposição de temporização
    - Limites para:
      - Atrasos máximos na transmissão de mensagens
      - Execução de etapas de um processo
      - Taxas de derivação de Relógio (Drift)
  - Suposições permitem usar limites para detectar falhas
  - Base para a coordenação (acordo)

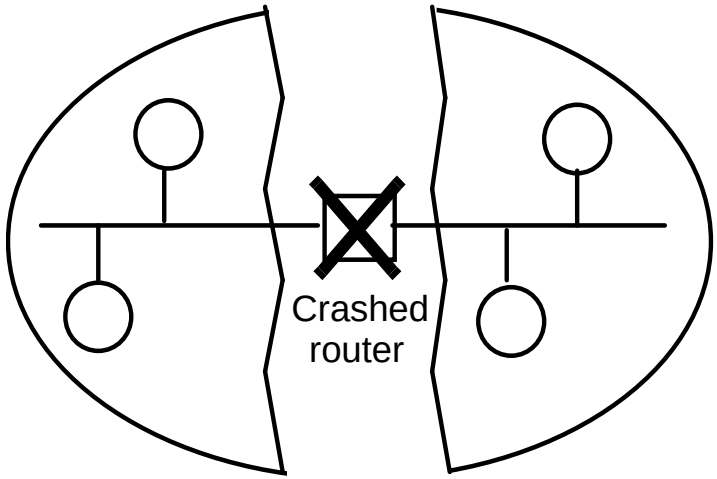
Suposição de Falhas

- Premissas:
  - Tarefas e Canais
    - Par de processos conectados por canais
      - Canal confiável
        - Mascaramento de falha (retransmissão)
    - Falha em um processo não impede os demais de se comunicarem
      - Independência de processos

- Se o canal é confiável:
  - Sempre ocorre o envio da mensagem à caixa postal do destinatário
    - Redundância (replicação) de hardware

Suposição de Falhas

- Se o canal é confiável:
  - A comunicação entre alguns processos pode acontecer
    - Pode haver particionamento de rede



Suposição de Falhas

- Internet
  - Topologias complexas
  - Ponto a ponto
  - Roteamentos independentes
    - Conectividade pode ser assimétrica, intransitiva
  - Logo, nem todos os processos podem se comunicar ao mesmo tempo

- Premissa:
  - Processos só falham por defeitos

Falhas de Processos e Canais

- Falha:
  - Por Omissão
    - Deixar de executar a ação que deveria
    - Processo colapsa e não executa demais passos.
      - Parada por falha
      - Usar timeout em sistema síncrono
    - Perda de mensagem
      - Descarte de mensagem
  - Arbitrárias
    - Qualquer tipo de erro pode ocorrer
      - Valores errados em variáveis/mensagens
  - Temporização
    - Limites de tempo não respeitados

Falhas de Processos e Canais

- Processo "correto"
  - Não apresenta falha em ponto algum
  - Processo sofre falha em ponto X
    - É não defeituoso até X
    - Não correto após X

- Como decidir se um processo falhou?
  - Detector de falhas
    - Serviço consultado pro processos
      - Vários Locais e um Global
    - Não é preciso
      - Pode ser não confiável

Falhas de Processos e Canais

- Detector de Falhas não Confiável
- Processo é:
  - Suspeito
    - Evidência de falha
      - Período máximo nominal de silêncio
        - Particionamento de rede?
        - Lentidão?
  - Não Suspeito
    - Evidência como mensagem recebida
      - E se falhou após mensagem?

Falhas de Processos e Canais

- Detector de Falhas Confiável
  - Preciso na detecção de falha
  - Processo é:
    - Não Suspeito
    - Falho
      - Em colapso (estado terminal)
  - Pode dar diferentes respostas a diferentes processos

Falhas de Processos e Canais

- Detector de Falhas Confiável
  - P_i envia uma mensagem P_i está vivo para cada P_k do grupo a cada intervalo T
  - Tempo máximo de transmissão é D (estimado)
  - Se o detector de falhas local em P_m não recebe mensagem de P_i no intervalo T + D
    - Relata a P_m de que é suspeito P_j
  - Caso contrário, relata OK

- Problema
  - Valores para T e D
    - Largura de banda x falho pode ser detectado como não falho.

Falhas de Processos e Canais

- Detector de Falhas Confiável
  - Valores para T e D
    - Usar valores que reflitam atrasos observados na rede
- Sistema síncrono
  - Detector pode se tornar confiável
    - D é limite de transmissão absoluto
- Detector não confiável:
  - Impreciso – suspeita de processo que não falhou
  - Incompleto – não suspeitar de processo que falhou

Exclusão Mútua

Exclusão Mútua

- Regiões (Seções)Críticas
  - Múltiplos Processos
    - Concorrência a Recursos
    - Consistência
- Semáforos
  - Monoprocessamento
- Solução em Sistemas Distribuídos?

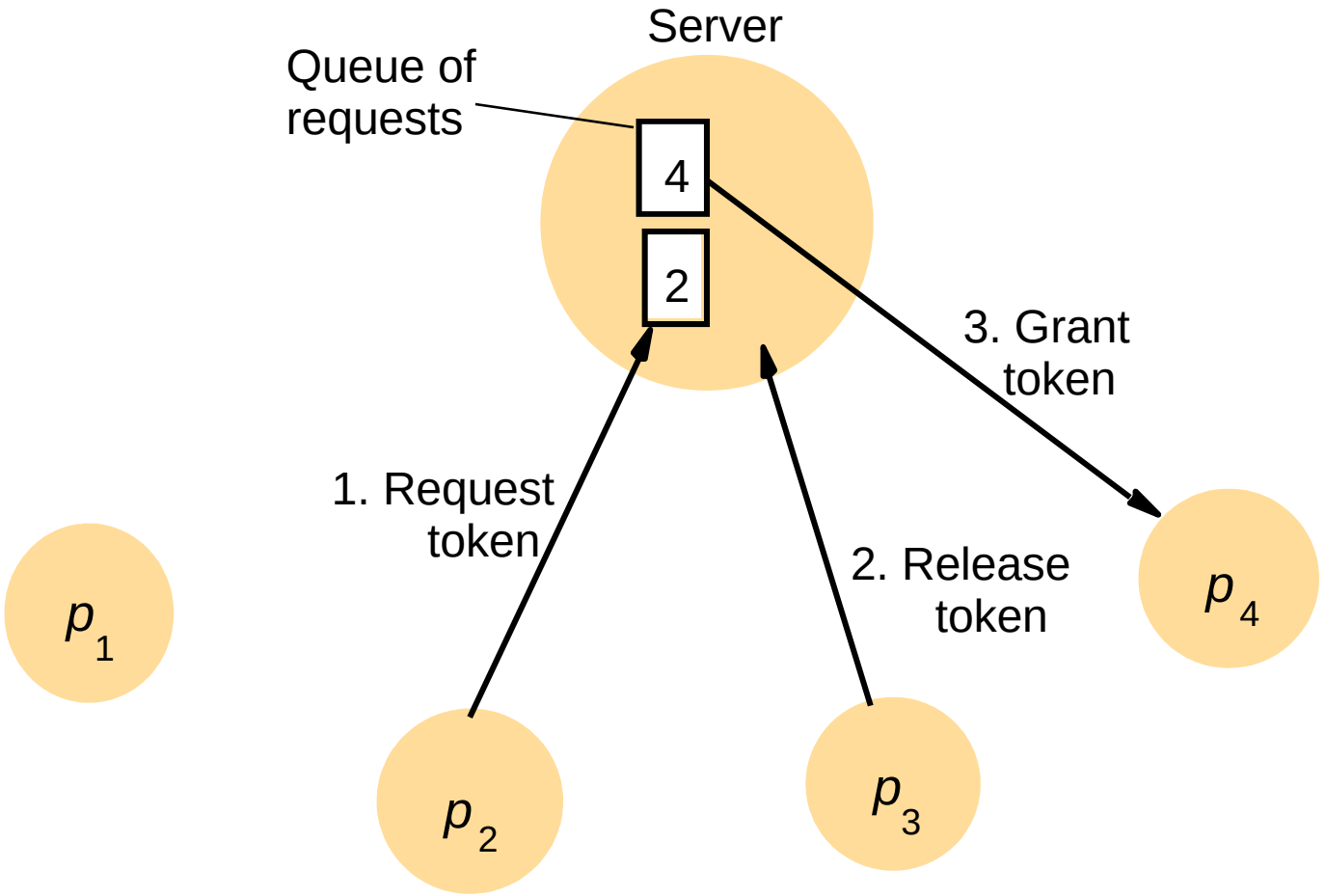
Exclusão Mútua

- Para resolver o problema de sincronização em SD, necessitamos de semáforos distribuídos. Existem três algoritmos para sua solução:
  - Centralizado
  - Distribuído
  - Anel

Algoritmo Centralizado

Algoritmo Centralizado

- Coordenador garante a Exclusão Mútua



Algoritmo Centralizado

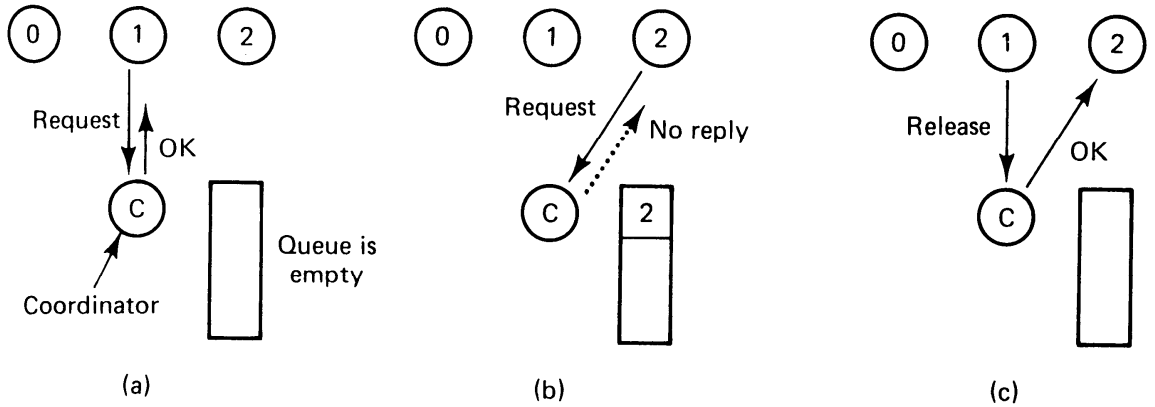
- Coordenador garante a Exclusão Mútua
- Duas Operações:
  - Request
  - Release
    - Permite Posse e Espera
- Três tipos de mensagem:
  - request, reply e release

Algoritmo Centralizado

a) Processo 1 solicita ao coordenador permissão para entrar na RC. Permissão dada.

b) Processo 2 solicita ao coordenador permissão para entrar na mesma RC. Permissão negada.

c) Quando o processo 1 sai da RC, ele avisa ao coordenador que libera o Processo 2.



Algoritmo Centralizado

```
Coordenador
  loop
    recv msg
      case msg of
        REQUEST: se RC está livre
                 então responder GRANTED
                 senão, enfileirar REQ [e
                      responder DENIED]
        RELEASE: se a fila não está vazia
                     remover 1o. da fila e
                     responder "GRANT"
                 senão, marca RC como livre
     end case
  end loop
```

Algoritmo Centralizado

- **Client**

```
send(REQUEST)
recv(msg) // Bloqueante: Reply indica RC Livre
    enter RC
send(RELEASE)
```

Algoritmo Centralizado

- **Client**

```
send(REQUEST)
recv(msg) // Bloqueante
if msg = GRANTED then enter RC
else recv(msg) // Bloqueante
     enter RC
send(RELEASE)
```

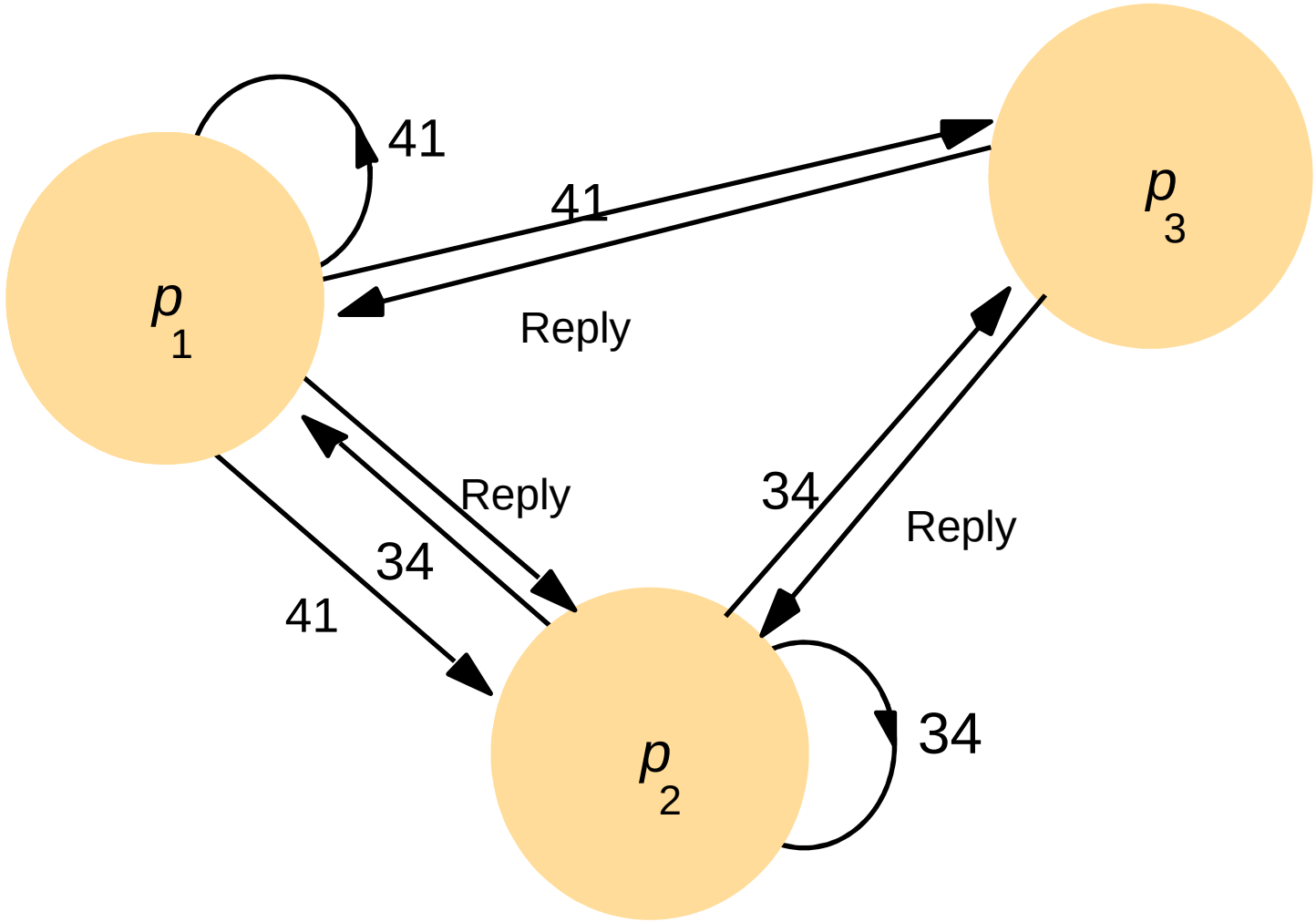
Algoritmo Centralizado

- **Client**

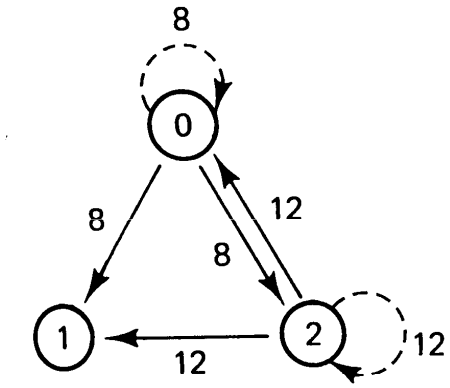
```
send(REQUEST)
WSARecv(msg) // Não Bloqueante
Test() // Testa resposta
if msg = GRANTED then enter RC
else WSARecv(msg)
     WAIT() // Bloqueante
      enter RC
send(RELEASE)
```

Algoritmo Distribuído

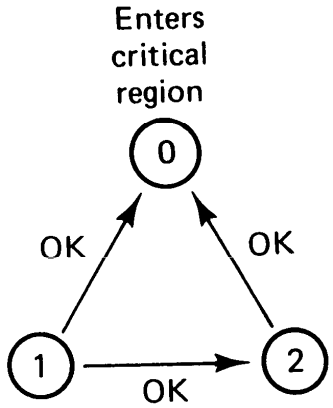
Algoritmo Distribuído



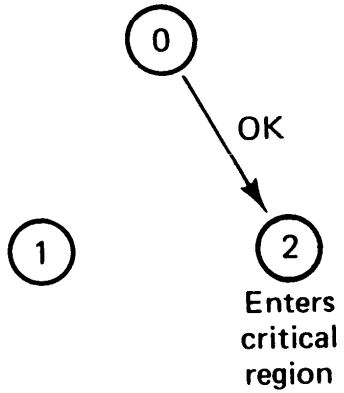
Algoritmo Distribuído



(a)



(b)



(c)

Algoritmo Distribuído

- Sem Coordenador
  - Decisão tomada pelo grupo
  - N processos pares
  - Multicast

- Dois processos desejam entrar na mesma RC ao mesmo tempo
  - Processo 0 tem o menor timestamp (TS), então ganha
  - Quando o processo 0 sair da RC ele envia um OK, então o 2 pode entrar na RC.

Algoritmo Distribuído

- Processo:
  - Quando um processo deseja entrar na Região Crítica X (RCx) gera um TS e envia request(RCx,TS) para todos os processos no sistema (grupo).
  - A RC está livre se o processo receber reply deTODOS os outros
  - Processo recebe Request e:
    - (A) se ele está na RC, não responde e coloca o requisitante na fila de espera
    - (B) se não deseja a RC, responde imediatamente
    - (C) se deseja entrar na RC, mantém uma fila de requisições e envia um reply para o request de menor TS.

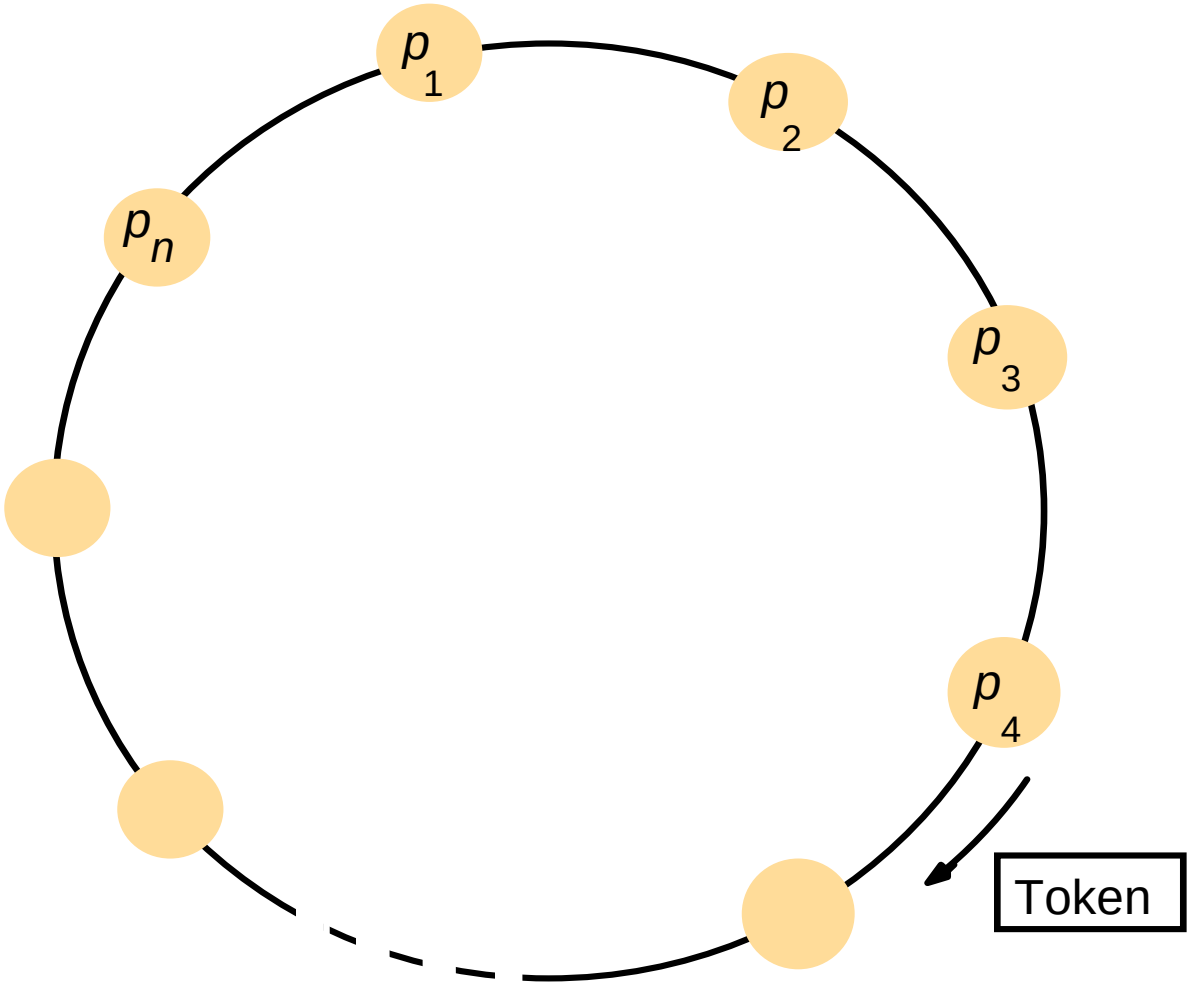
Algoritmo Distribuído

- Propriedades:
  - Garante EM
  - Sem starvation, assumindo ordenamento total de msgs
  - 2(N-1) msgs: (N-1) request e (N-1) reply
  - n pontos de falha
  - ack explícito e timeout

  - Overhead
    - Decisão por grupo
    - Voto de n-1 elementos do grupo
    - Performance

Algoritmo do Anel

Algoritmo do Anel



Algoritmo do Anel

- Um token circula num anel lógico
  - Um processo só entra na RC se possuir o token
- Perda do Token?
- Detecção
  - Tempo de Espera?

Comparação

Algorithm	Messages per entry/exit	Delay before entry (in message times)	Problems
Centralized	3	2	Coordinator crash
Distributed	2 (n – 1)	2 (n – 1)	Crash of any process
Token ring	1 to ∞	0 to n – 1	Lost token, process crash

Comparação

- Algoritmo com maior *Overhead* na troca de mensagens.
  - Caso exista grande concorrência entre processos pela RC, é o Distribuído.
  - Caso contrário, é o do Anel.

Algoritmos de Eleição

Algoritmos de Eleição

- Escolha do Coordenador
  - PID único
    - Não é do SO, é do Grupo.
  - Cada processo conhece todos os outros
    - Grupos?
  - Estado dos processos
    - Ativo?
      - AYA
    - Como determinar?

Algoritmos de Eleição

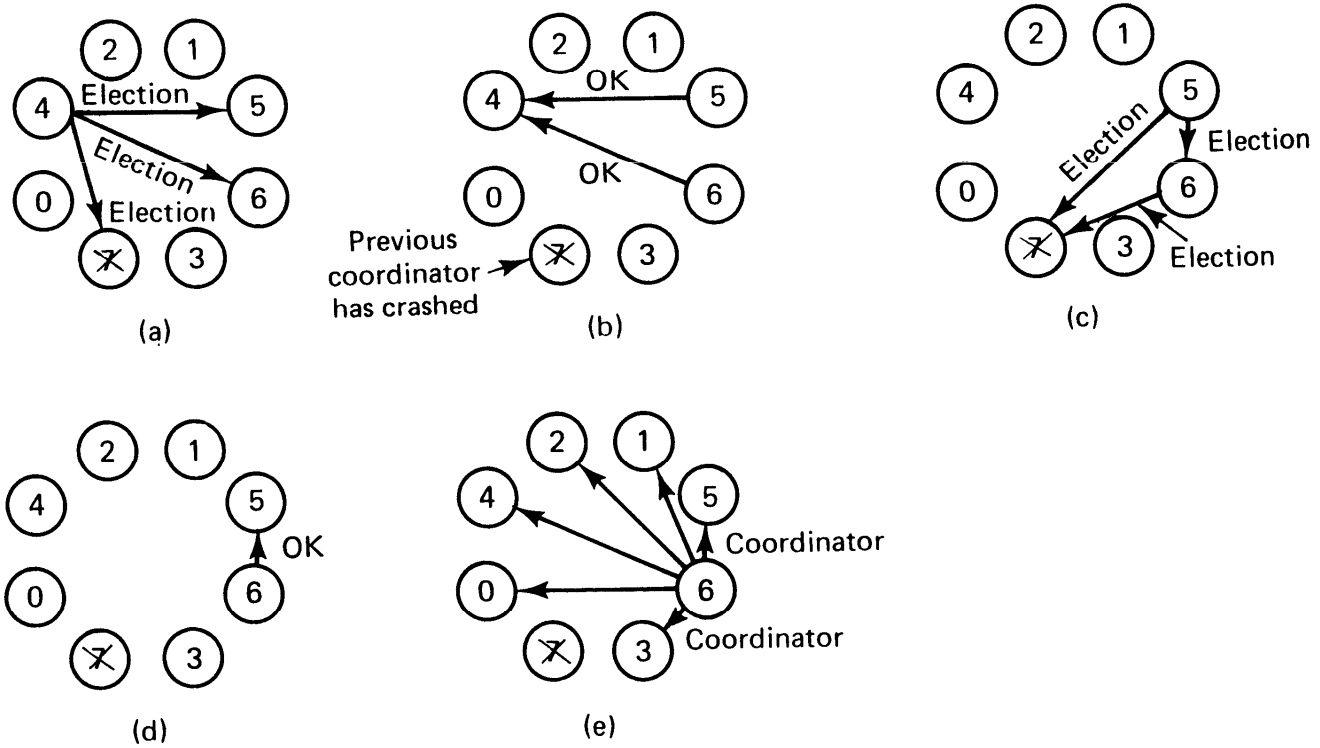
- Algoritmo do Ditador
  - Garcia-Molina (1982)
  - Maior PID indica o Coordenador
    - Válido para o grupo
  - Processo pode falhar durante eleição
  - Sistema síncrono
    - Detector de falhas confiável
      - Atraso max de msg = Tm
      - Atraso max de processo = Tp
      - T = 2 Tm + Tp
        - Atraso máximo

Algoritmos de Eleição

- Algoritmo do Ditador
  - Processo da Eleição:
    - p envia uma msg de ELEIÇÃO para todos os processos com PID > PIDp
    - p aguarda pelo tempo T
    - Se ninguém responde, p ganha a eleição
      - p envia mensagem de COORDENADOR a todos com PID < PIDp
    - Se alguém responde (PIDq > PIDp), p para (perde a eleição)
      - p aguarda T' para receber a mensagem COORDENADOR de q
      - Se T' expira e a mensagem não chega, p inicia nova eleição

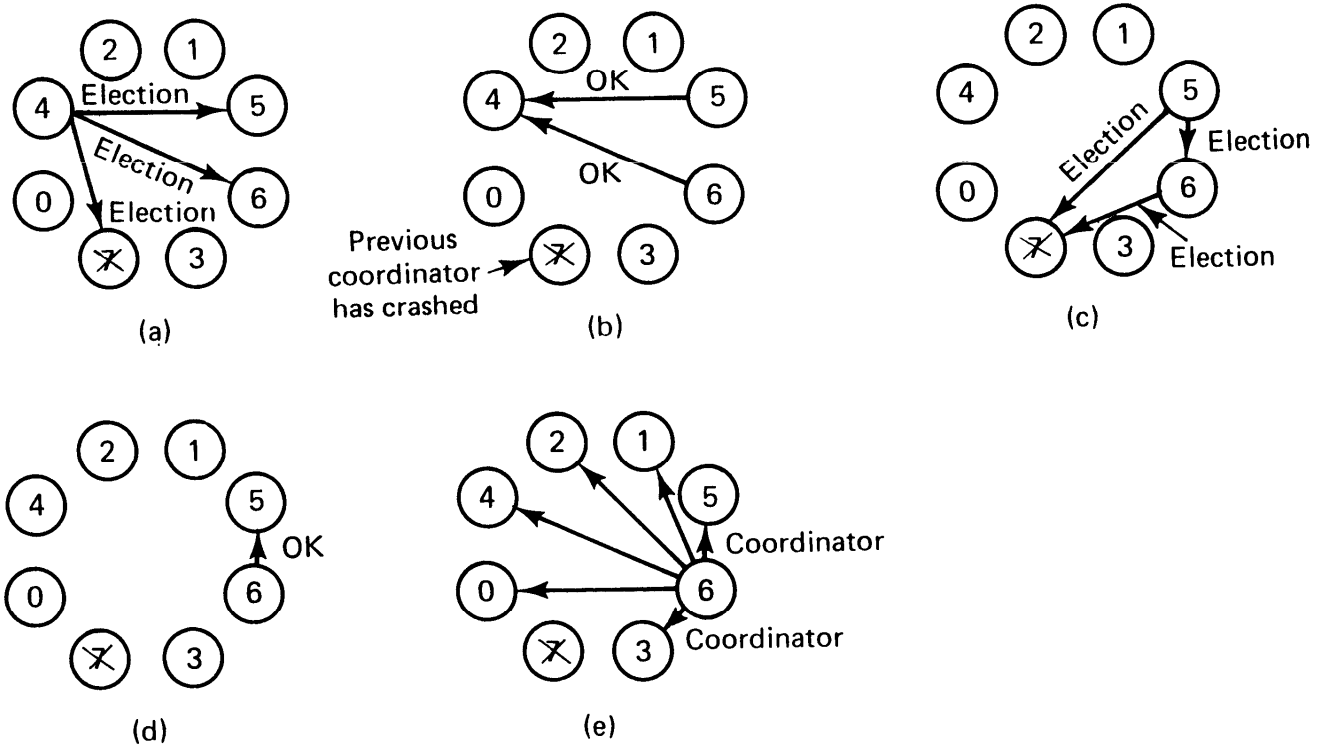
Algoritmos do Ditador

- (a) Processo 4 inicia uma eleição.



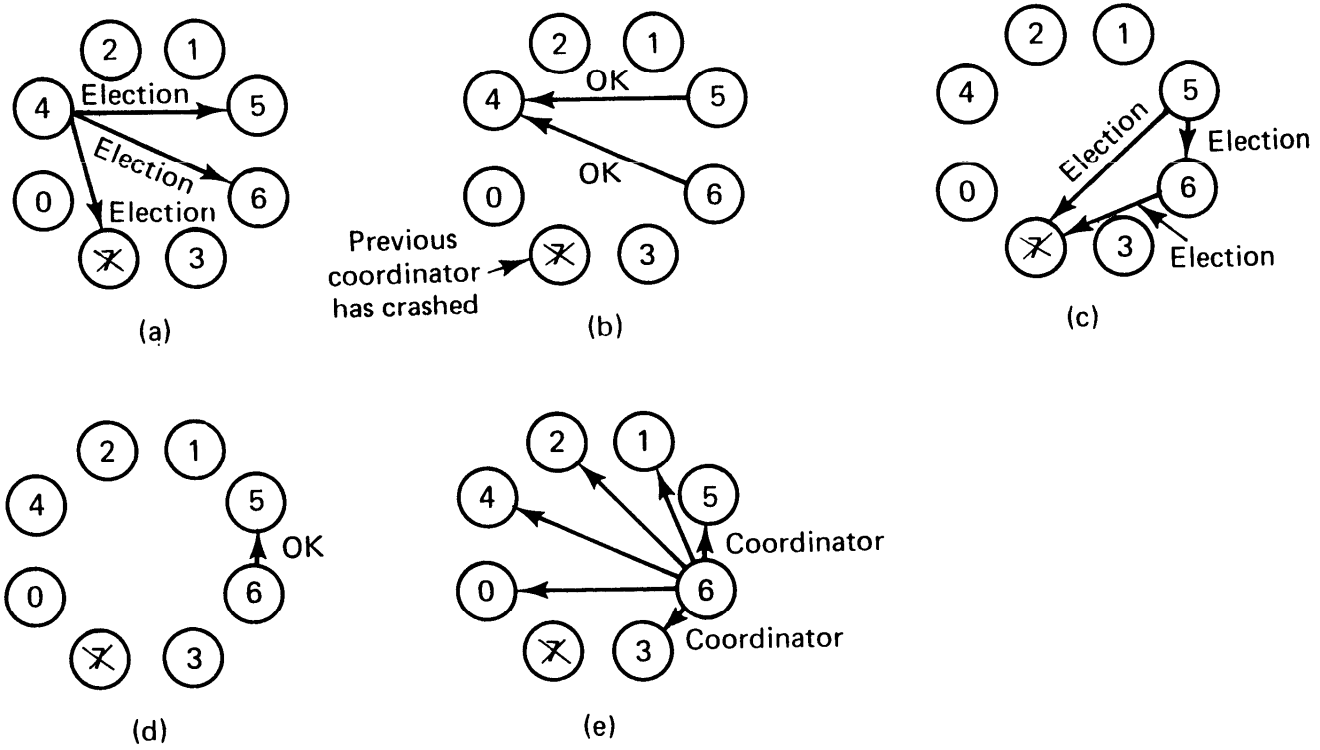
Algoritmos do Ditador

- (b) Processos 5 e 6 respondem, informando a 4 para parar



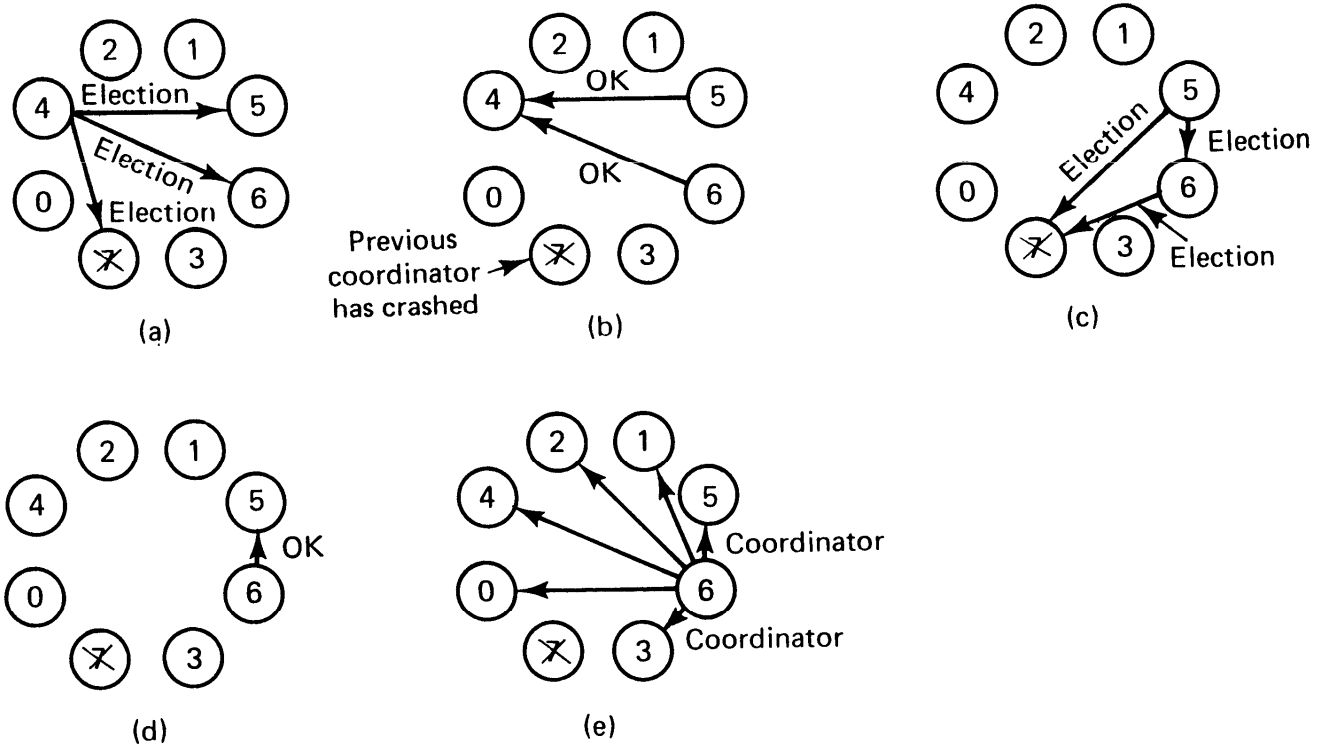
Algoritmos do Ditador

- (c) 5 e 6 iniciam a eleição.



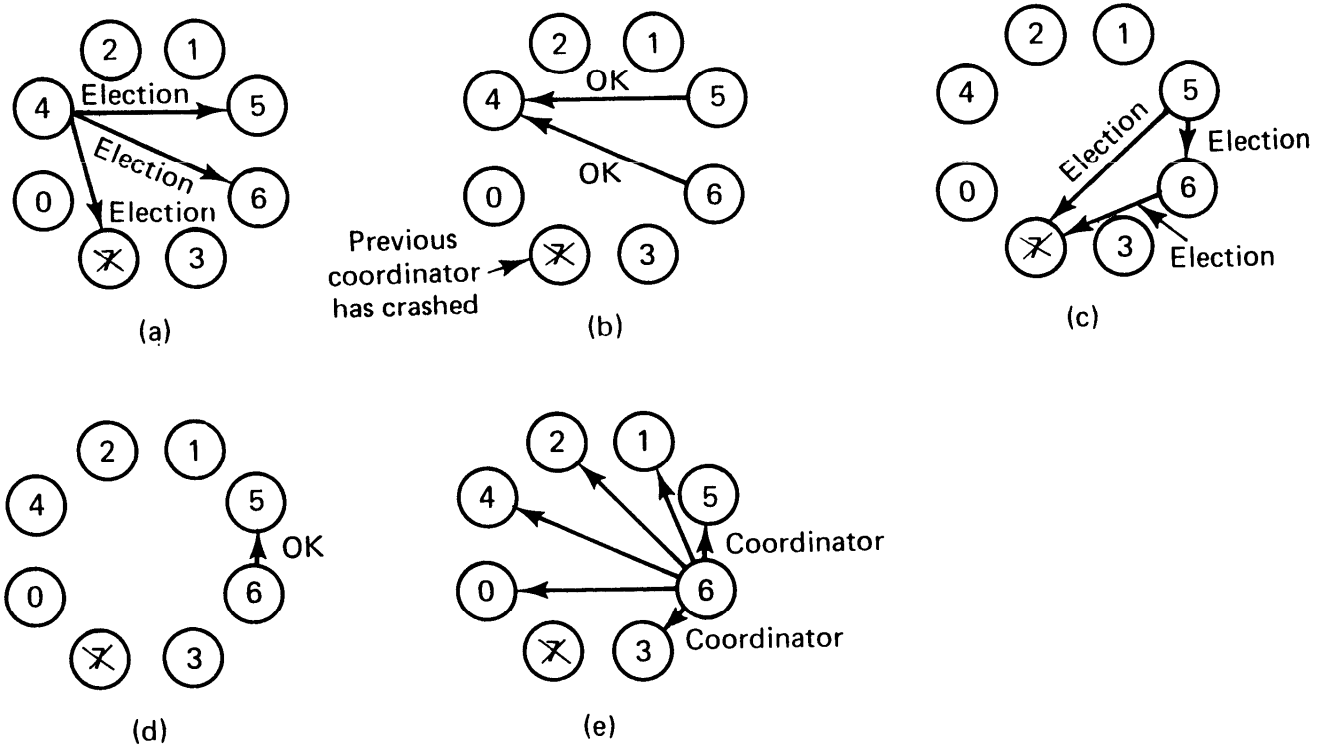
Algoritmos do Ditador

- (d) Processo 6 informa a 5 para parar.

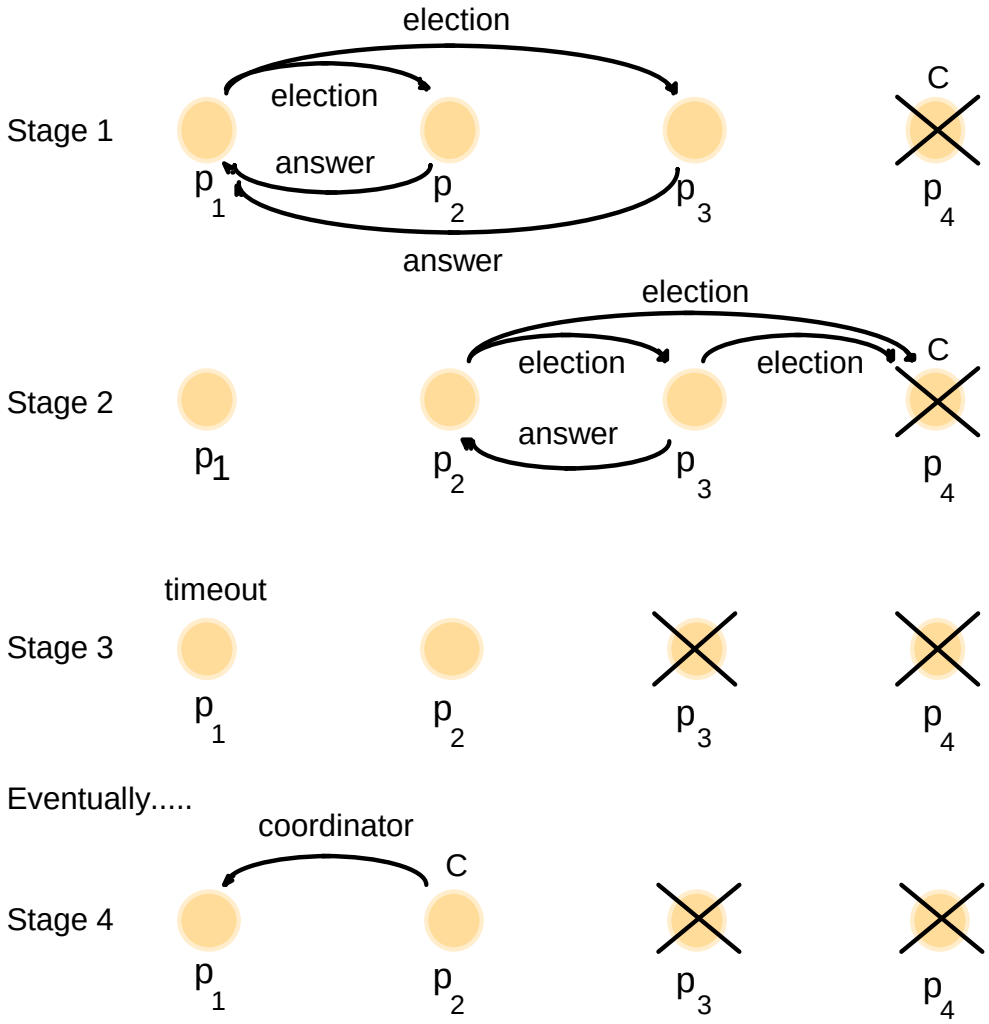


Algoritmos do Ditador

- (e) Processo 6 ganha e informa a todos.



Algoritmos do Ditador



Algoritmos do Ditador

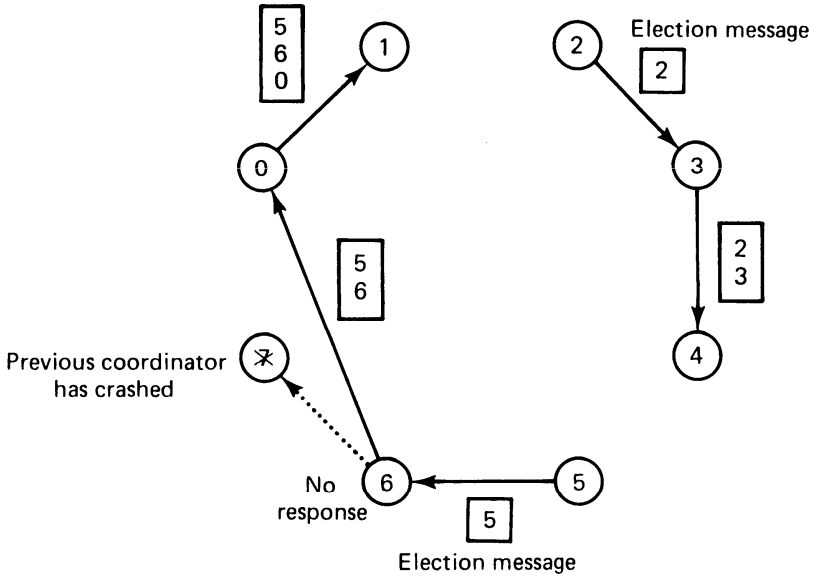
- Processo x entra no grupo
  - Novo
  - Após queda
    - Qual o coordenador?
      - Nova Eleição.

Algoritmos de Eleição

- Algoritmo do Anel
  - Não usa token
  - Ordem lógica ou física dos processos
    - Cada processo conhece os sucessores
  - Início
    - Detecção de queda do Coordenador.

Algoritmos de Eleição

- (a) Processos 2 e 5 descobrem, simultaneamente, que o coordenador falhou.
- (b) 2 e 5 iniciam uma msg de ELEIÇÃO.
- (c) msg circula.



DeadLock

Deadlock em
Sistemas Distribuídos

Deadlock em SD

- Deadlock
  - Imprevisível
  - Difícil reprodução

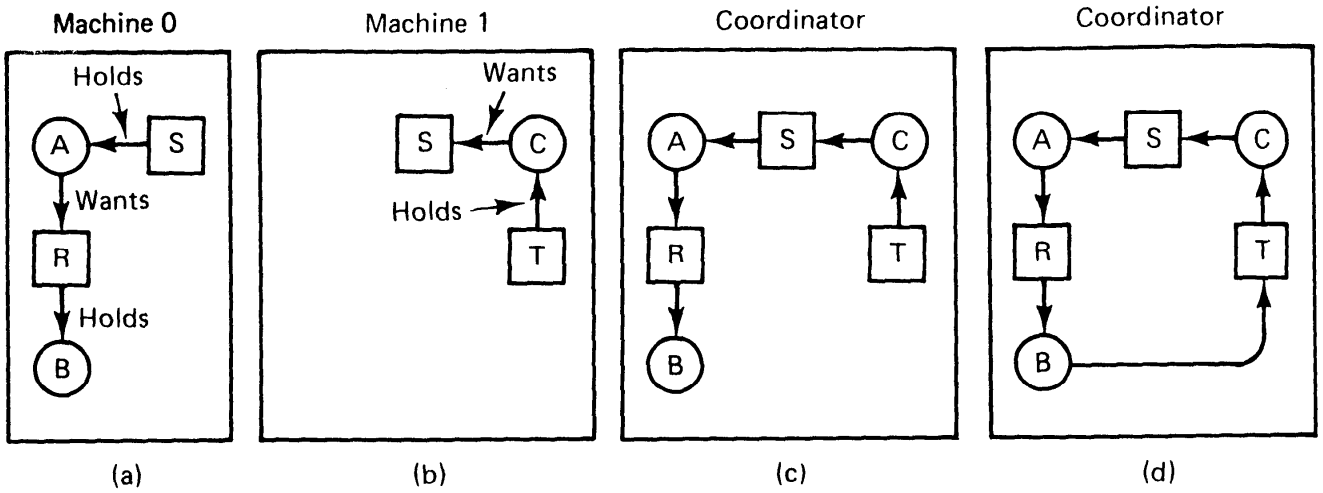
- Estratégias
  1. Detecção e Recuperação
  2. Prevenção
  3. Impedimento

Deadlock em SD

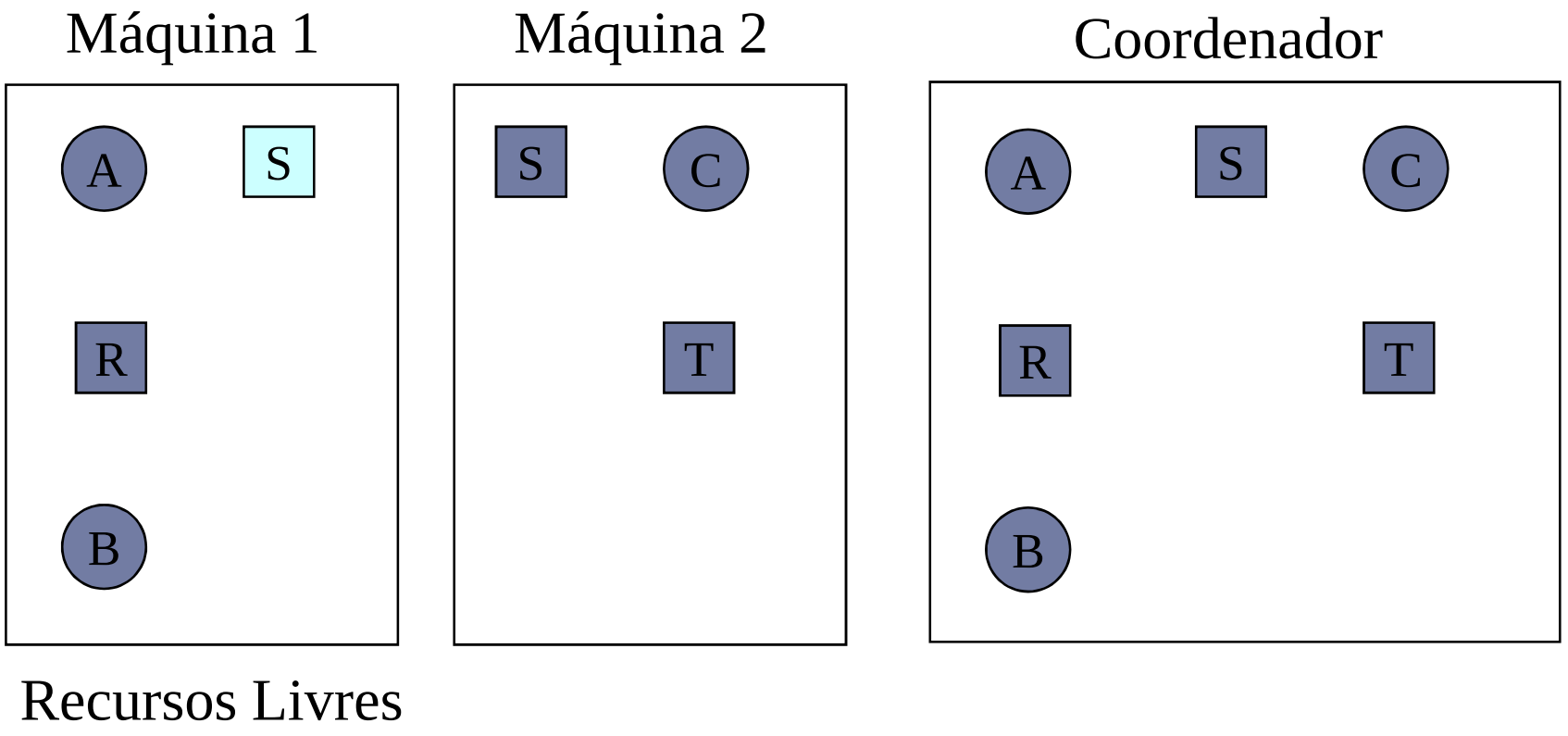
- Dificuldades:
  - Informação de alocação de recursos distribuída
  - Visão precisa (tempo real?) da alocação

Deadlock em SD

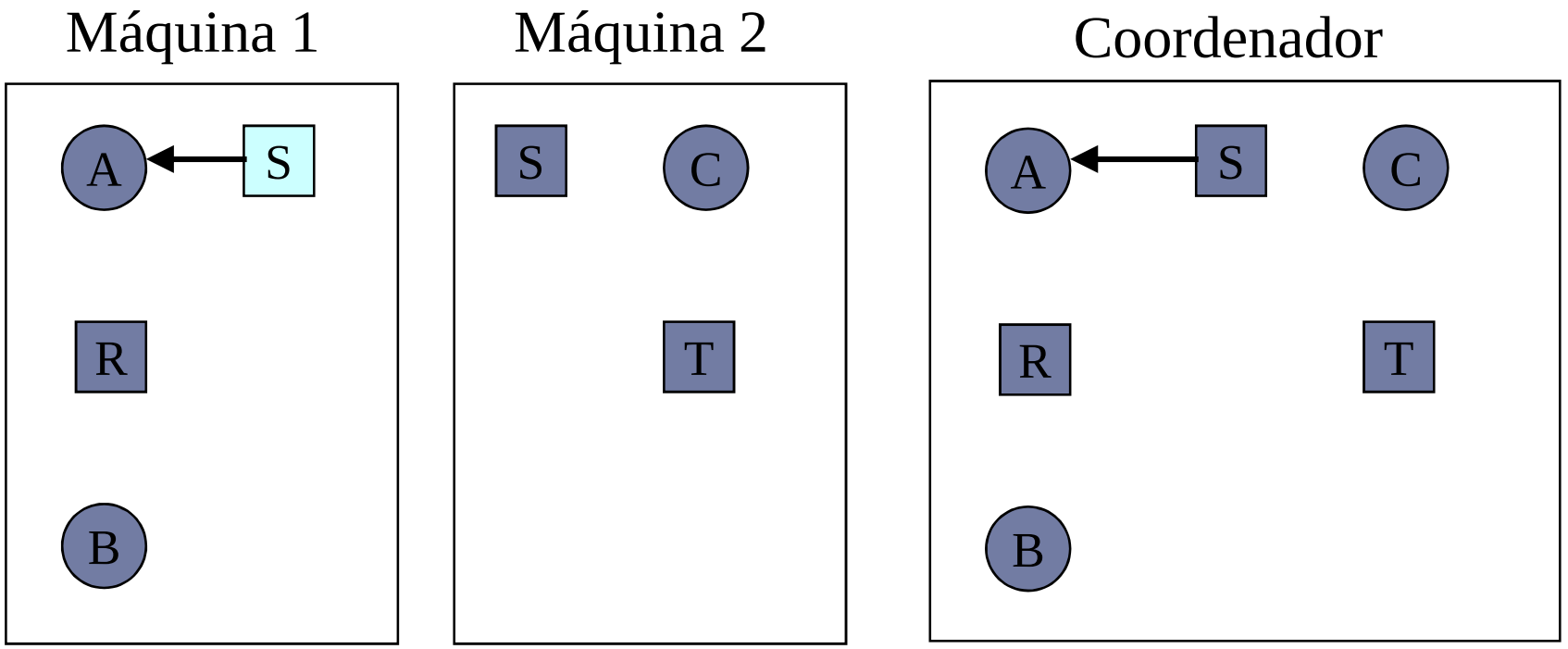
- Solução centralizada
  - Falso Deadlock



Deadlock em SD



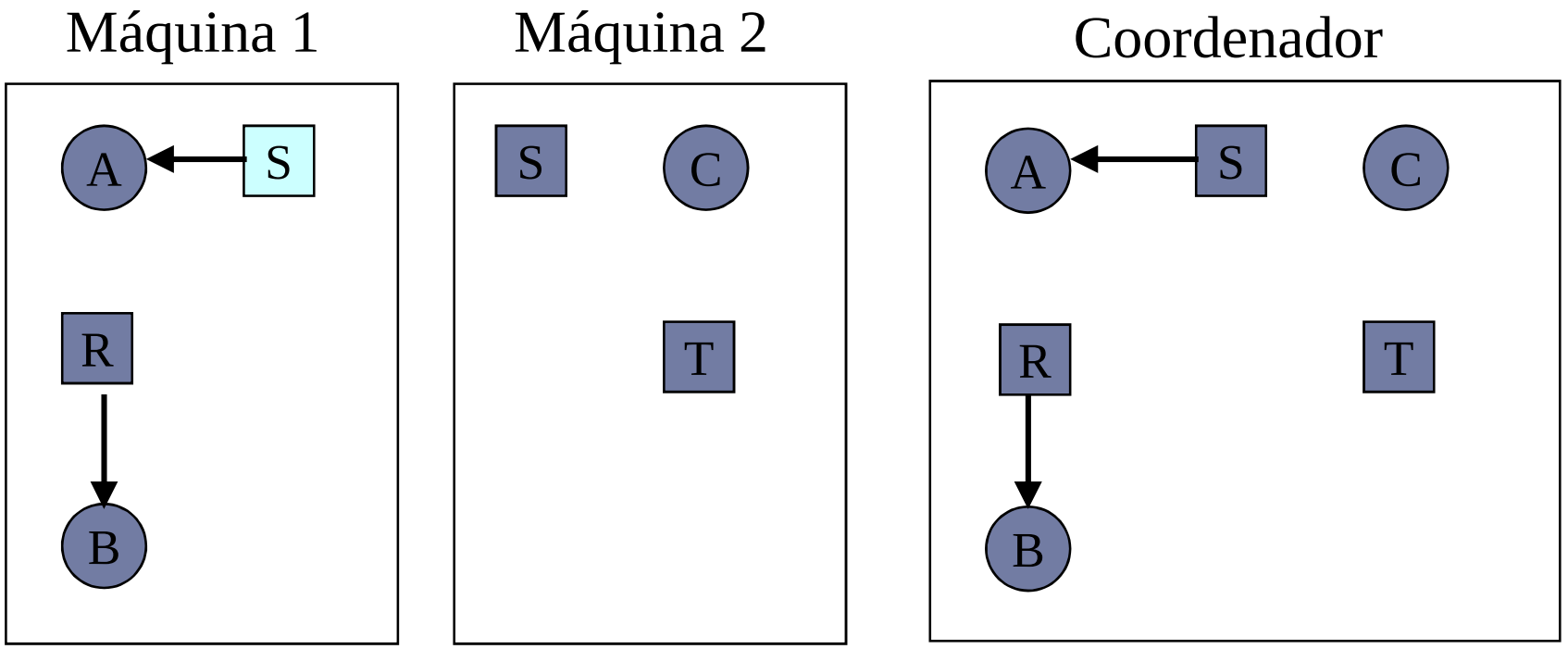
Deadlock em SD



A solicita S

Máquina 2 envia Alocação ao Coordenador

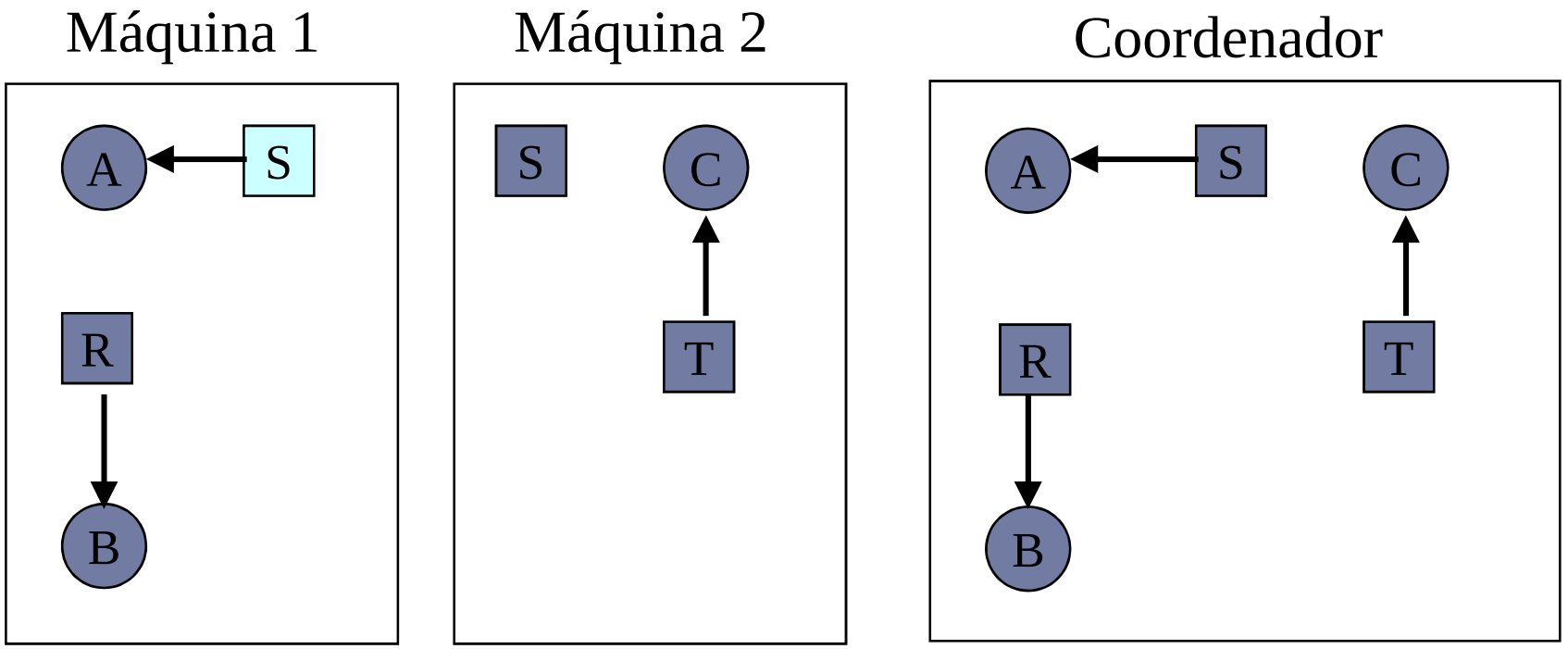
Deadlock em SD



B solicita R

Máquina 1 envia Alocação ao Coordenador

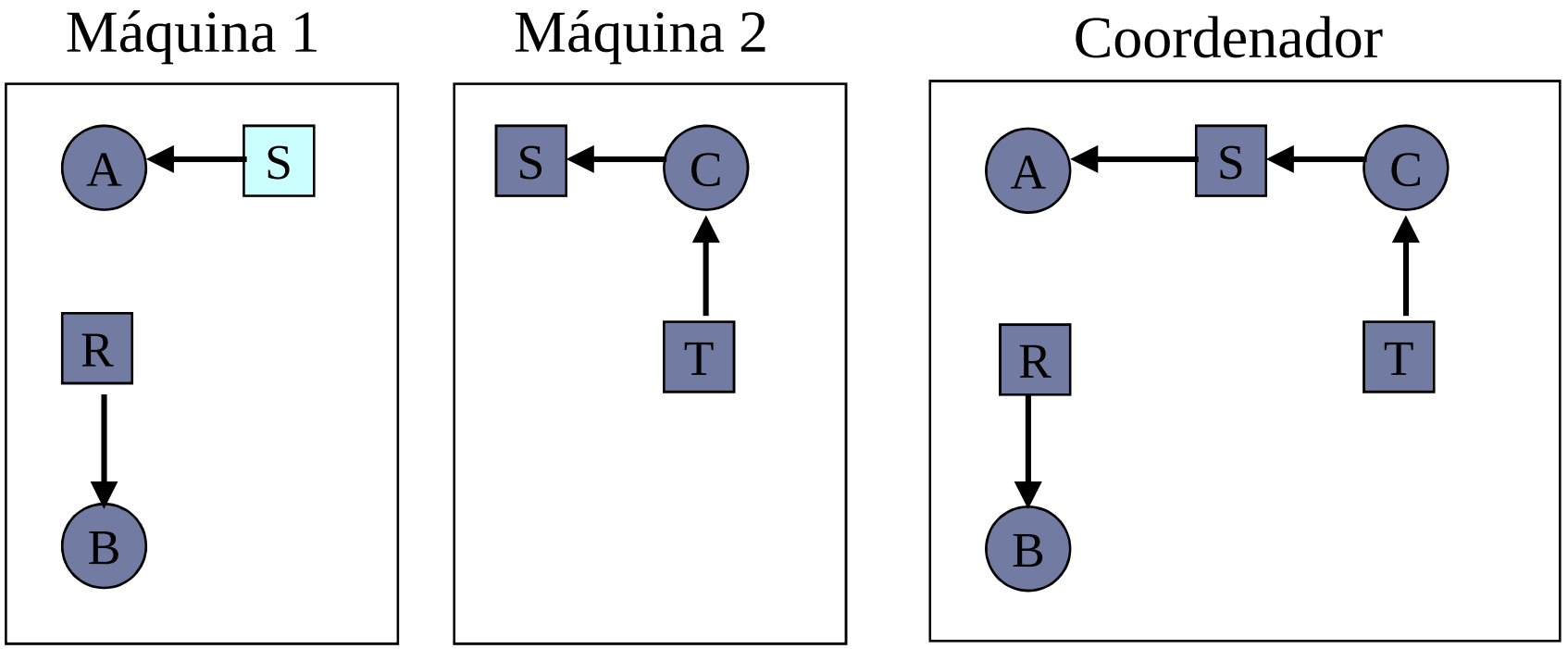
Deadlock em SD



C solicita T

Máquina 2 envia Alocação ao Coordenador

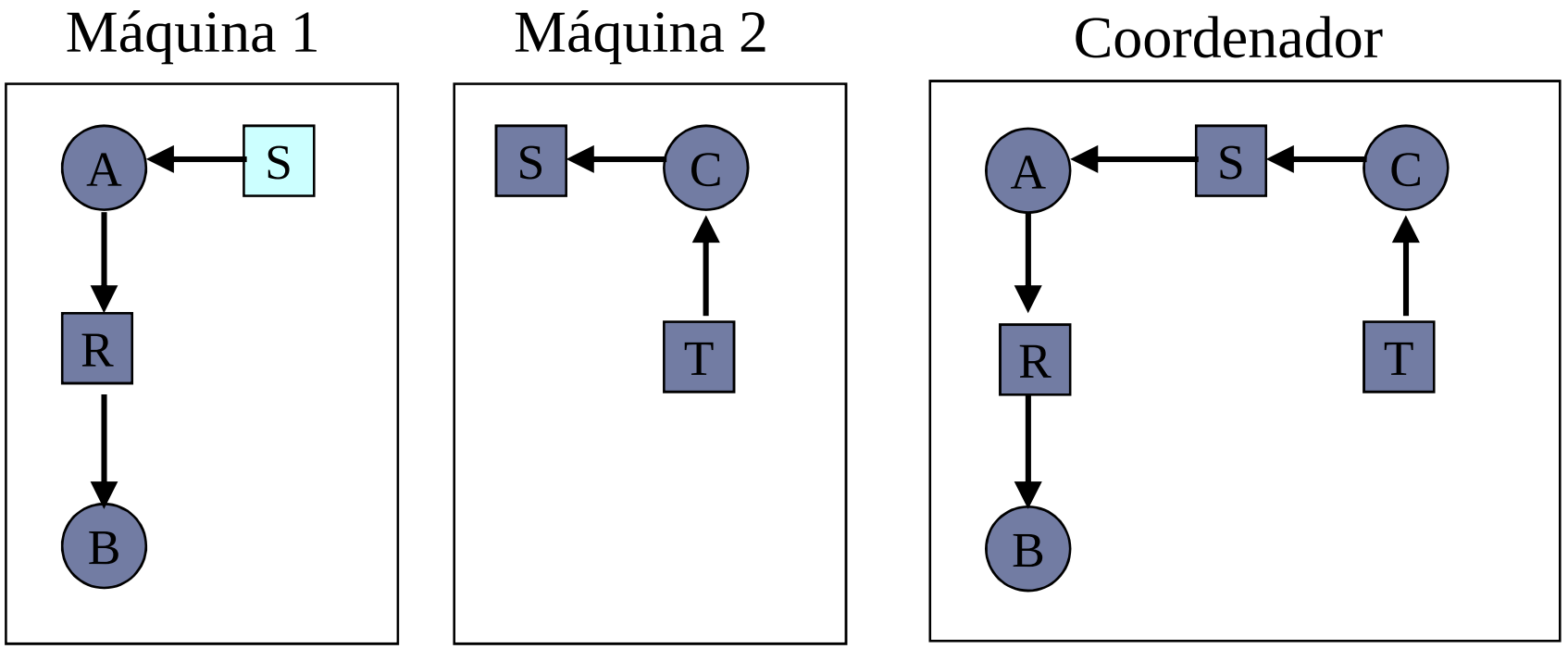
Deadlock em SD



C solicita S

Máquina 2 envia Requisição ao Coordenador

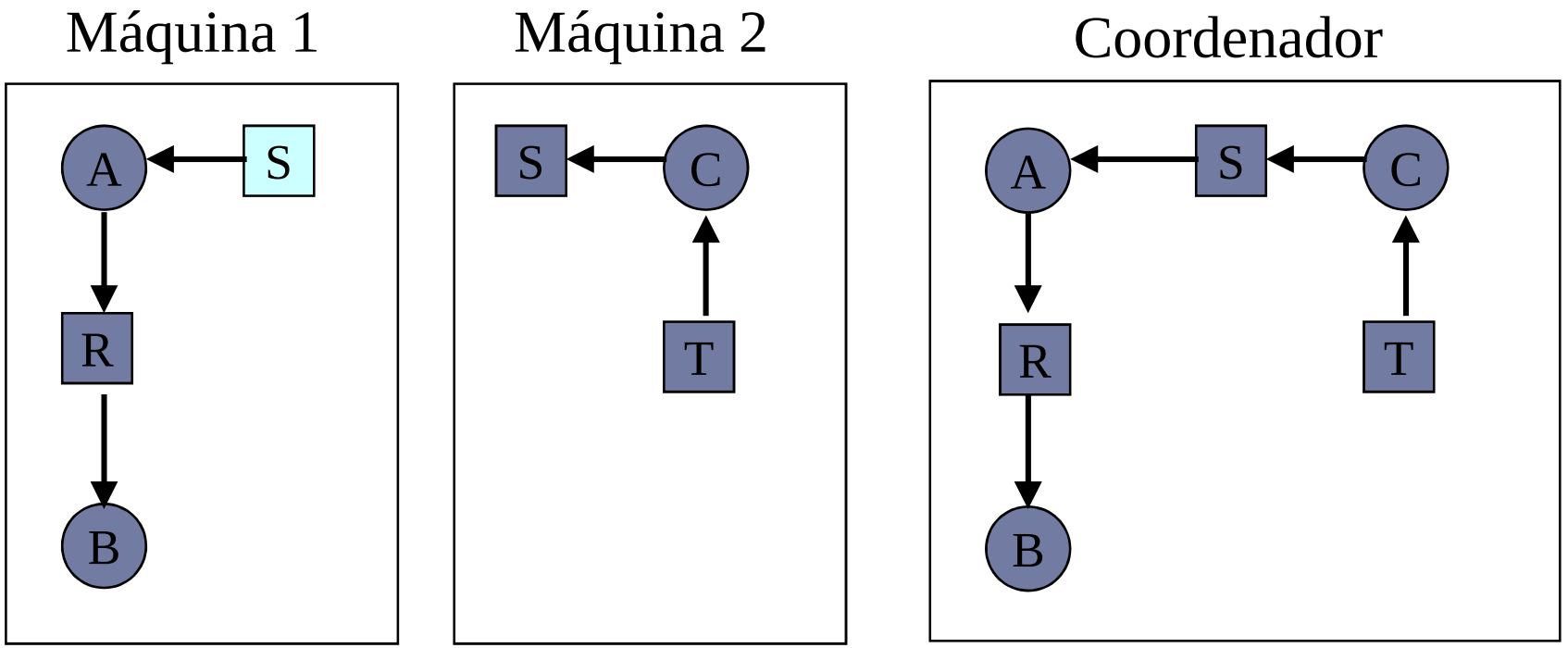
Deadlock em SD



A solicita R

Máquina 1 envia Requisição ao Coordenador

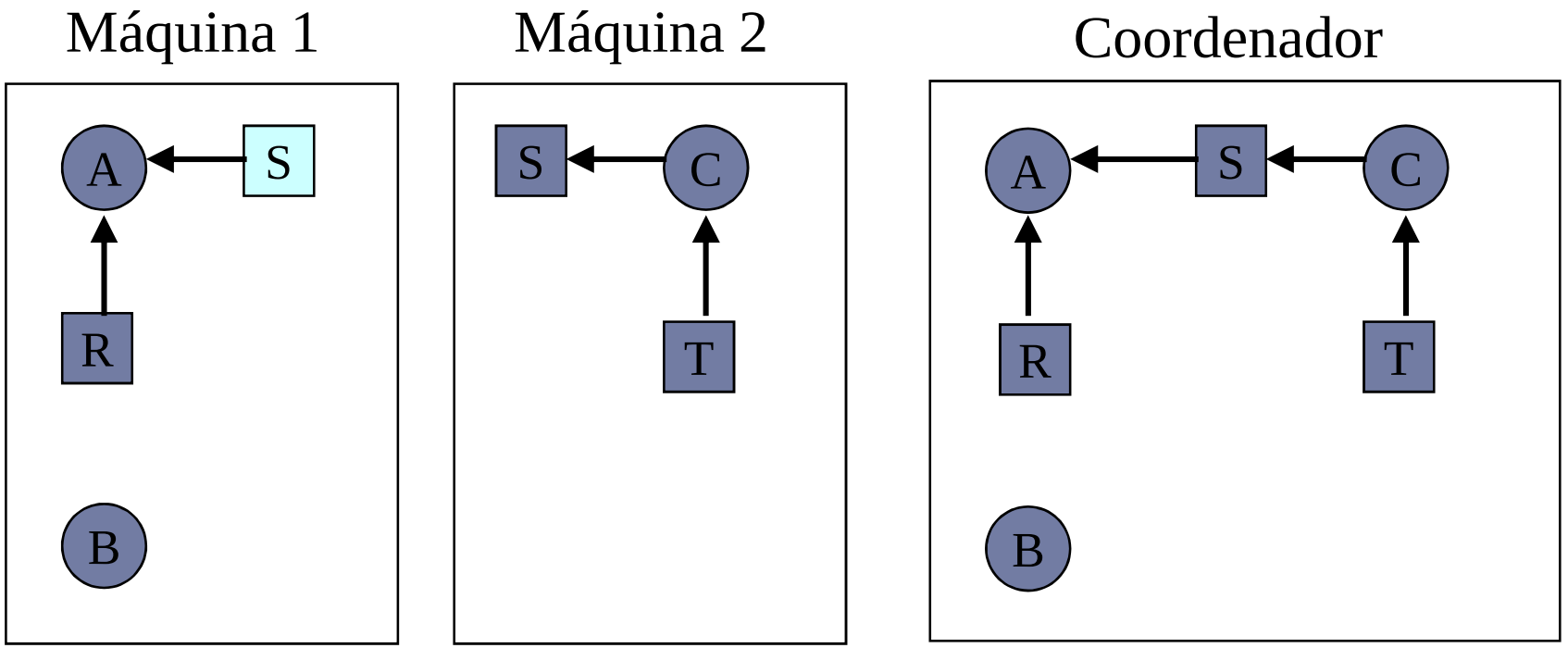
Falso Deadlock



T0 - B Libera R
T1- B Solicita T

Máquina 1 envia Liberação ao Coordenador
Máquina 2 envia Solicitação ao Coordenador

Falso Deadlock



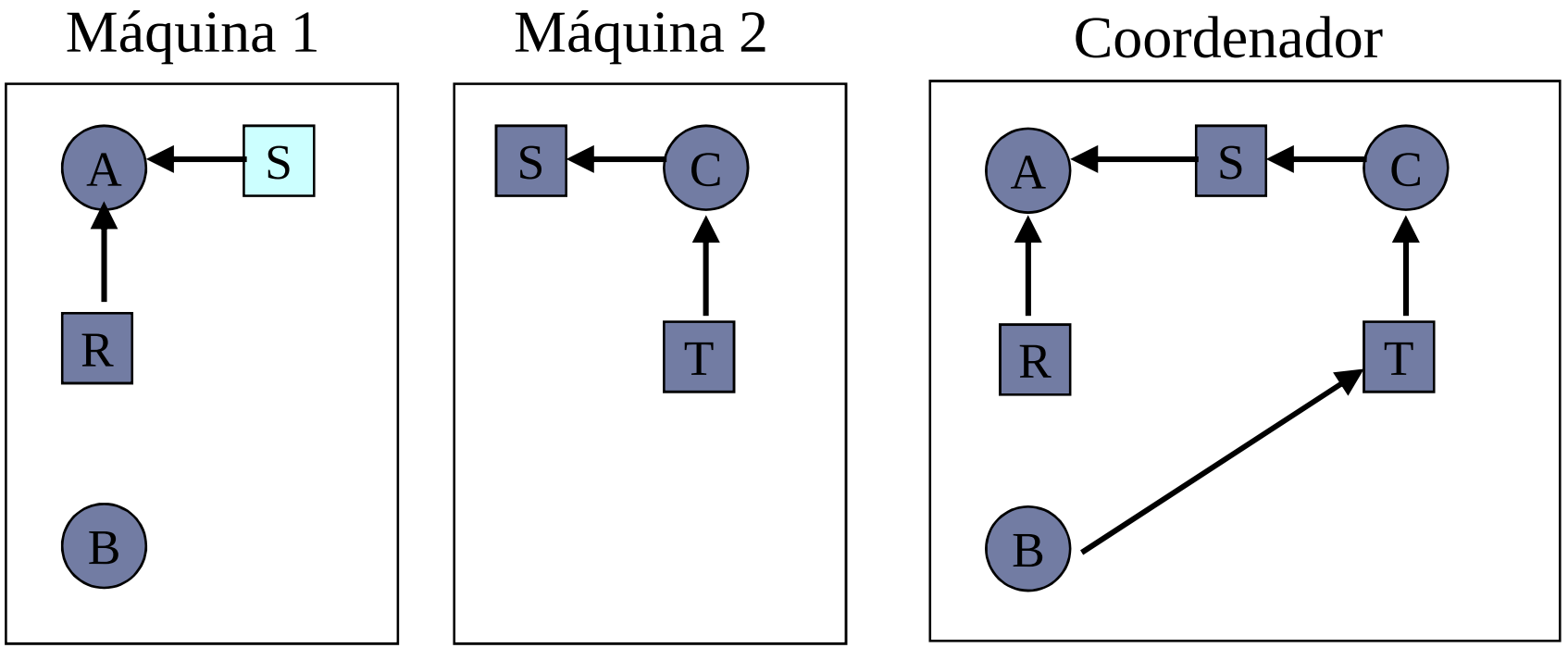
T0 - B Libera R

T1- B Solicita T

Máquina 1 envia Liberação ao Coordenador

Máquina 2 envia Solicitação ao Coordenador

Falso Deadlock



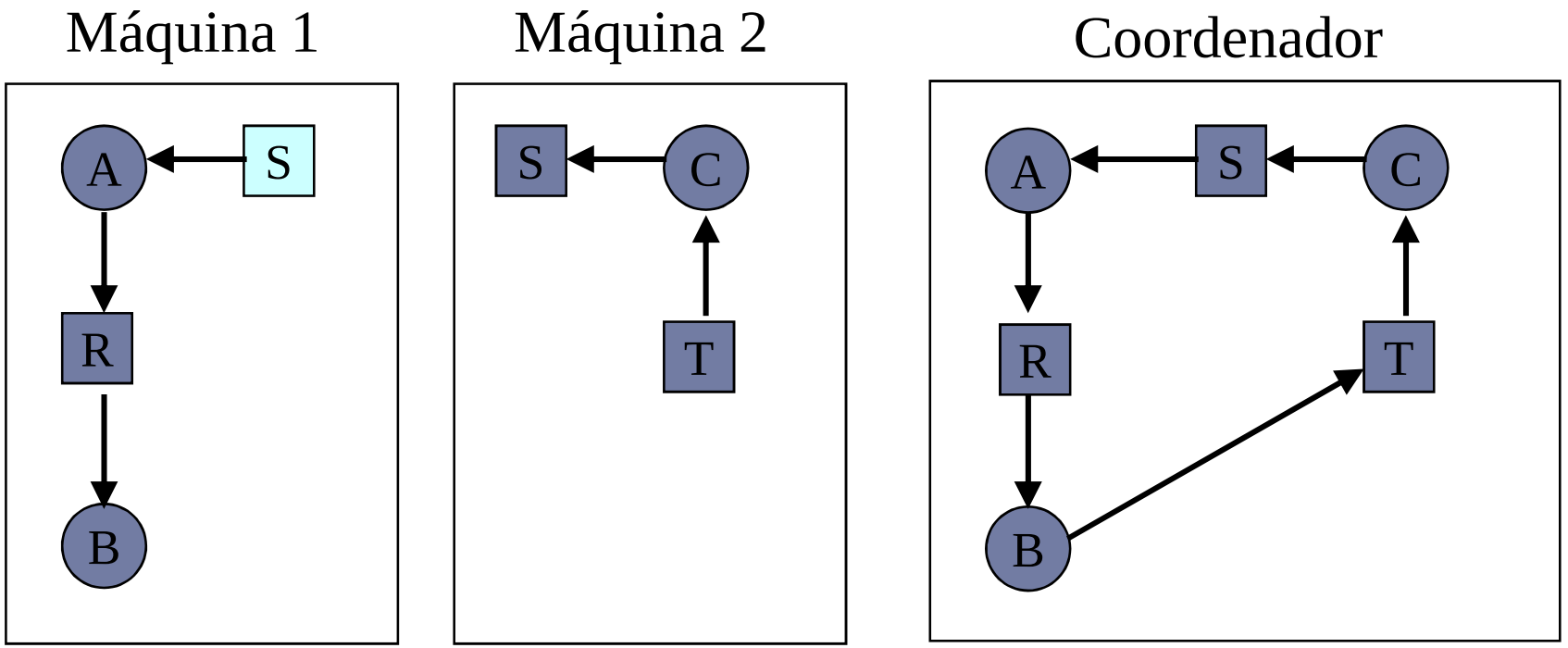
T0 - B Libera R

T1- B Solicita T

Máquina 1 envia Liberação ao Coordenador

Máquina 2 envia Solicitação ao Coordenador

Falso Deadlock

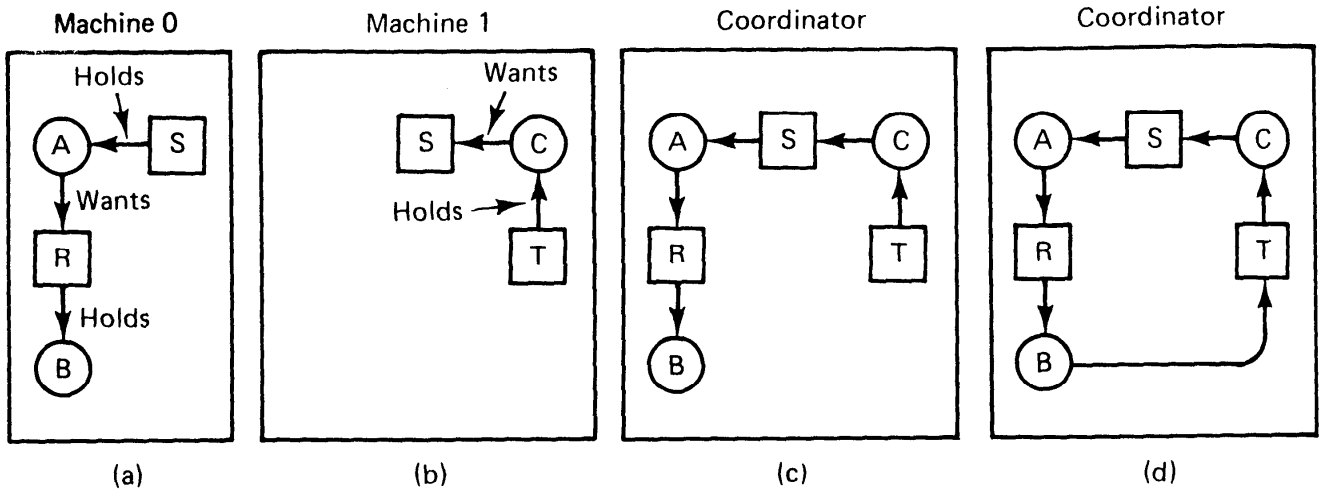


T0 - B Libera R
T1- B Solicita T

Máquina 1 envia Liberação ao Coordenador
Máquina 2 envia Solicitação ao Coordenador

Deadlock em SD

- Solução centralizada
  - Falso Deadlock



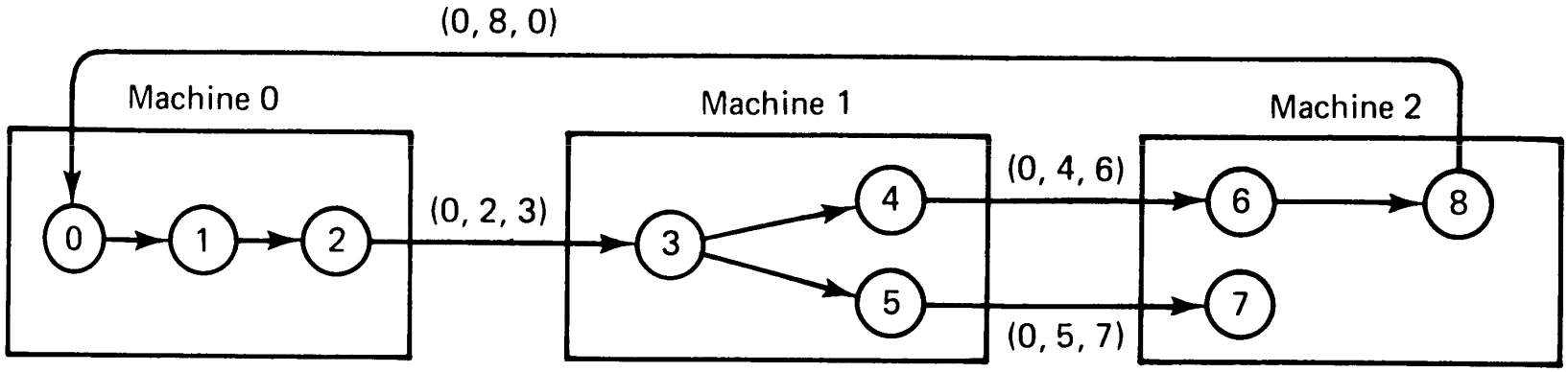
Deadlock em SD

- Solução centralizada
  - Lamport
    - Tempo Global
    - Análise de mensagem anterior
      - Overhead

Deadlock em SD

- Solução Distribuída
  - Chandy-Misra-Haas
    - Requisição de múltiplos recursos por vez
- Dificuldade
  - Arcos entre máquinas
    - Como encontrar ciclos?

Deadlock em SD



Deadlock em SD

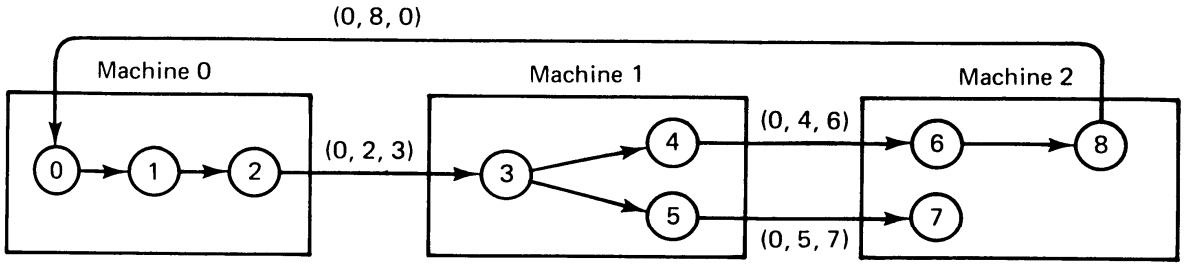


Figura apenas com processos. Cada arco passa por um recurso
Logo, P3 aguarda 2 recursos, um preso por P4 e outro por P5

Mensagem: (Processo bloqueado, Processo emissor da msg, Processo Destino)

msg inicial: (0,0,1)

Receptor checa se está aguardando (bloqueado) algum recurso.
Se sim, altera e envia mensagem.

Ciclo = Retorno à origem ➔ Deadlock

Deadlock em SD

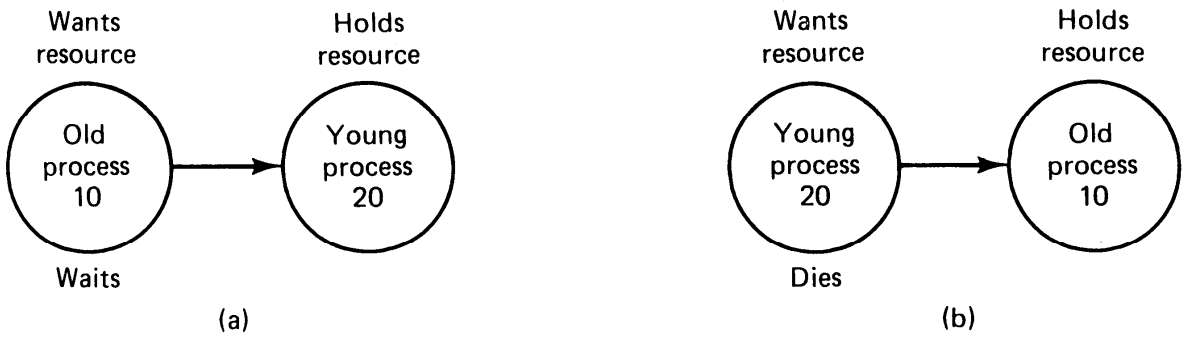
- Solução Distribuída
  - Processo inicial
    - Suicídio
    - Início simultâneo?
  - Alternativa
    - Adição dos PID no fim da msg
      - Anel
        - Matar processo de maior PID

Deadlock em SD

- Solução Distribuída
  - Como enviar a mensagem estando bloqueado?

Deadlock em SD

- Prevenção
  - Ordenamento hierárquico de processos
  - Wait-Die (WD): Não-preemptivo
    - Quando *P* requer um recurso alocado por *Q* , *P* aguarda apenas se for mais velho que *Q*. Em outro caso, *P* morre.



Deadlock em SD

- Prevenção
  - Wound-Wait (WW): Preemptivo
    - Quando *P* requer um recurso alocado por *Q* , *P* aguarda apenas se for mais novo que *Q*. Em outro caso, *Q* morre, liberando os recursos.