- Exclusão Mútua Distribuída

- Algoritmos para eleição de um coordenador ou líder

Exclusão Mútua Distribuída

Suponhamos N processos, pi, i=1,2,…,N que não partilham variáveis, mas que partilham algum recurso partilhado ao qual devem aceder numa secção critica (s.c.).

Assumimos que o sistema é assíncrono, que os processos não falham e que a comunicação é fiável.

O protocolo ao nível da aplicação é o seguinte:

enter() // entrar na secção crítica, bloquear se necessário

resourceAccesses() // aceder aos recursos partilhados na s.c.

exit() // sair da secção crítica, outro processo pode entrar

Exclusão Mútua Distribuída

Os requisitos para a exclusão mútua são os seguintes:

EM1: (segurança)

Apenas um processo de cada vez pode executar a s.c.

EM2: (nem impasse nem asfixia = *liveness*)

Pedidos para entrar ou sair da secção critica deverão ter sucesso num período razoável de tempo.

Por vezes é importante que:

EM3: (ordenação)

Se um processo pede o acesso à s.c. antes de outro, o acesso deve ser-lhe garantido primeiro.

Exclusão Mútua Distribuída

Para avaliar a performance de cada algoritmo, vamos analisar:

- **Bandwidth** (largura de banda consumida, proporcional ao número de mensagens enviadas em cada operação de entrada e saída da s.c.)

- **Client delay** (tempo necessário para o processo entrar na s.c.)

- **Synchronization delay** (efeito do algoritmo sobre o throughput do sistema. Tempo entre um processo sair da s.c. e outro processo entrar)

Exclusão Mútua Distribuída

Algoritmo Centralizado

A forma mais simples de garantir a exclusão mútua é ter um processo que garante a permissão para entrar na secção crítica

- Para entrar na s.c. um processo envia ao servidor um pedido de entrada

(A resposta do servidor constitui um “token” que significa permissão para entrar na secção crítica)

- Se nenhum processo detém o token, o servidor responde imediatamente, garantindo o acesso

Exclusão Mútua Distribuída

Algoritmo Centralizado

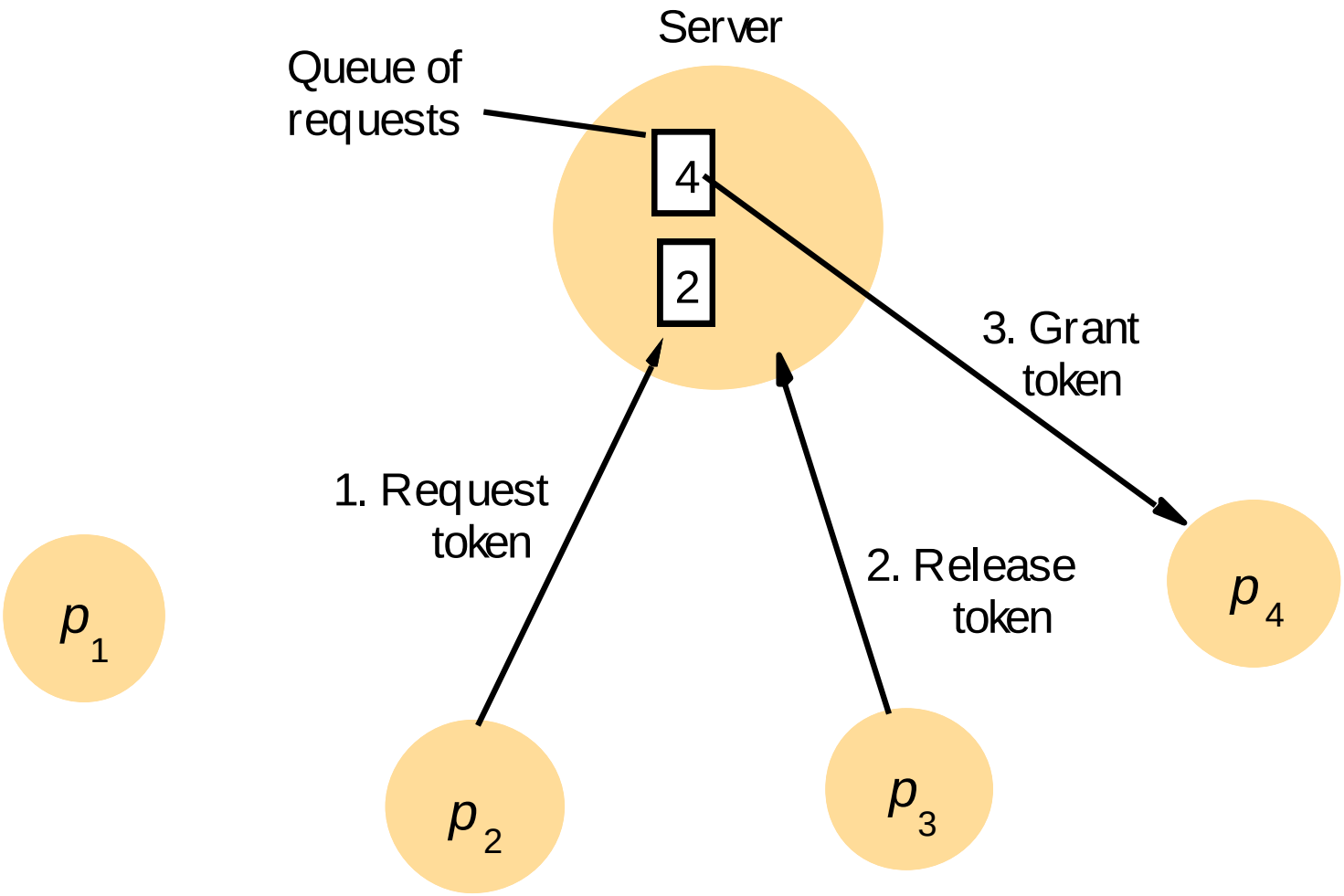
- Se o token está na posse de algum outro processo o servidor não responde, colocando o processo em espera

- Quando um processo sai da s.c. envia uma mensagem ao servidor, devolvendo o token.

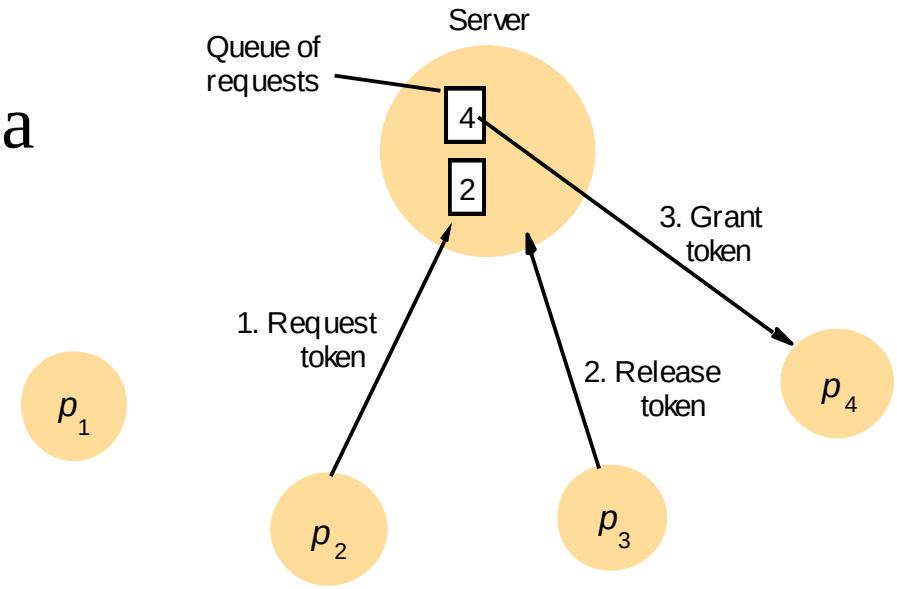
- Se a fila de processos em espera não está vazia, o servidor escolhe a entrada mais antiga na fila e envia o token a esse processo

Exclusão Mútua Distribuída

Algoritmo Centralizado



Exclusão Mútua Distribuída



- O pedido de P2 foi colocado em espera, onde já estava P4
- Quando p3 sai da s.c. devolve o token ao servidor.
- O servidor dá permissão a P4 enviando-lhe o token
- Quando p4 sair da s.c. p2 poderá entrar.

Exclusão Mútua Distribuída

Algoritmo Centralizado

Bandwidth

Entrar na secção crítica requer 2 mensagens; Sair da s.c. requer 1 mensagem

Client delay

atraso do processo para entrar na s.c. será o tempo necessário para enviar pedido e receber resposta (round trip time)

Synchronization delay

O atraso na sincronização será também um round trip time: processo que sai da s.c. envia mensagem de release e o servidor envia o token a um novo processo

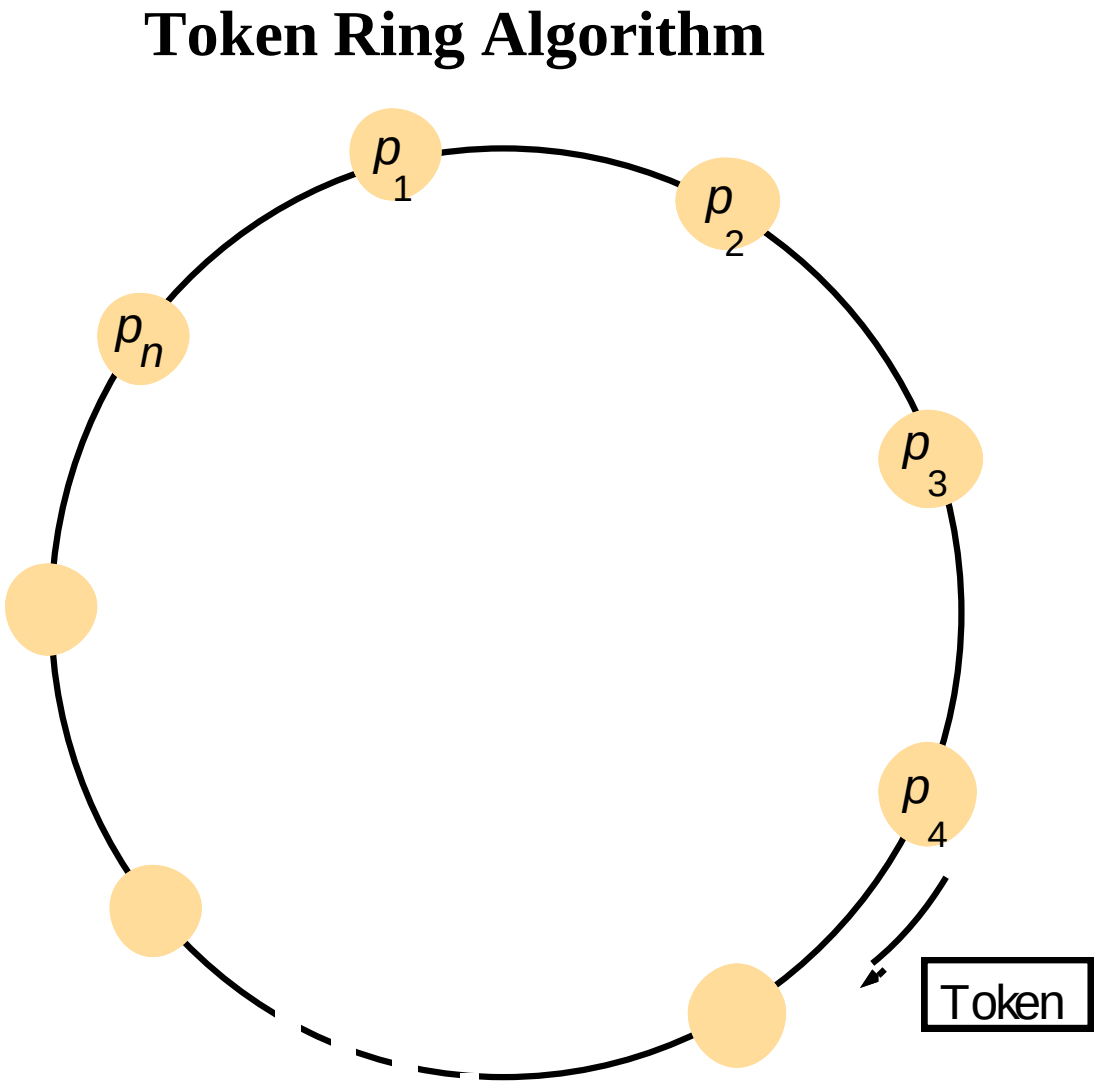
- O servidor pode ser um bottleneck para o desempenho do sistema.

Exclusão Mútua Distribuída

Token Ring Algorithm

- Os processos são organizados em anel, segundo o seu IP ou outra numeração.
- Cada processo Pi tem de ter um canal de comunicação com o processo que se segue no anel P(i+1) mod N.
- O token que dá acesso à s.c. é passado de processo em processo numa única direcção.
- Se um processo não pretende aceder à s.c. simplesmente passa o token ao seguinte.
- Um processo que queira aceder à s.c. espera pelo token, fica com ele enquanto acede à s.c. e no final passa-o ao processo seguinte.

Exclusão Mútua Distribuída



Exclusão Mútua Distribuída

Token Ring Algorithm

Nota: este algoritmo não verifica a regra EM3. (Porquê?)

- **Bandwidth**

O algoritmo consome continuamente largura de banda (excepto quando está a aceder à s.c.) 1 .. ∞

- **Client delay**

O delay de um processo que quer entrar na secção crítica pode ir de **1 a N** mensagens.

- **Sinchronization delay**

O tempo de sincronização entre a saída de um processo da s.c. e a entrada de um novo pode ir de **1 a N** mensagens.

Exclusão Mútua Distribuída

Um Algoritmo Distribuído

Ricart and Agrawala

Os processos, p1,p2, … pN têm identificadores distintos. Todos os processos podem comunicar com todos e cada processo possui um relógio Lógico (Lamport) actualizado com as regras que vimos anteriormente.

Mensagens de pedido para entrar na s.c. são da forma <T,pi>> onde T é o timestamp do emissor e pi o seu identificador.

Exclusão Mútua Distribuída

Um Algoritmo Distribuído

Ricart and Agrawala …

Cada processo regista o seu estado como,

RELEASED // fora da secção crítica

WANTED // quer aceder à secção crítica

HELD // está na secção crítica

Exclusão Mútua Distribuída

Um Algoritmo Distribuído

Ricart and Agrawala

Se um processo pretende entrar na secção crítica envia o pedido a todos os processos (multicast).

Se o estado de todos os processos for RELEASED todos respondem imediatamente e o processo acede à s.c.

Se algum processo está no estado HELD, então esse processo não responde ao pedido até terminar a s.c.

Exclusão Mútua Distribuída

Um Algoritmo Distribuído

Ricart and Agrawala …

Se dois ou mais processos requerem o acesso à s.c. ao mesmo tempo, então o processo que tem o menor timestamp irá ser o primeiro a obter as N-1 respostas que lhe garantem o acesso à s.c.

Se dois pedidos tiverem o mesmo timestamp, então os pedidos são ordenados pelo identificador do processo.

Exclusão Mútua Distribuída

Um Algoritmo Distribuído

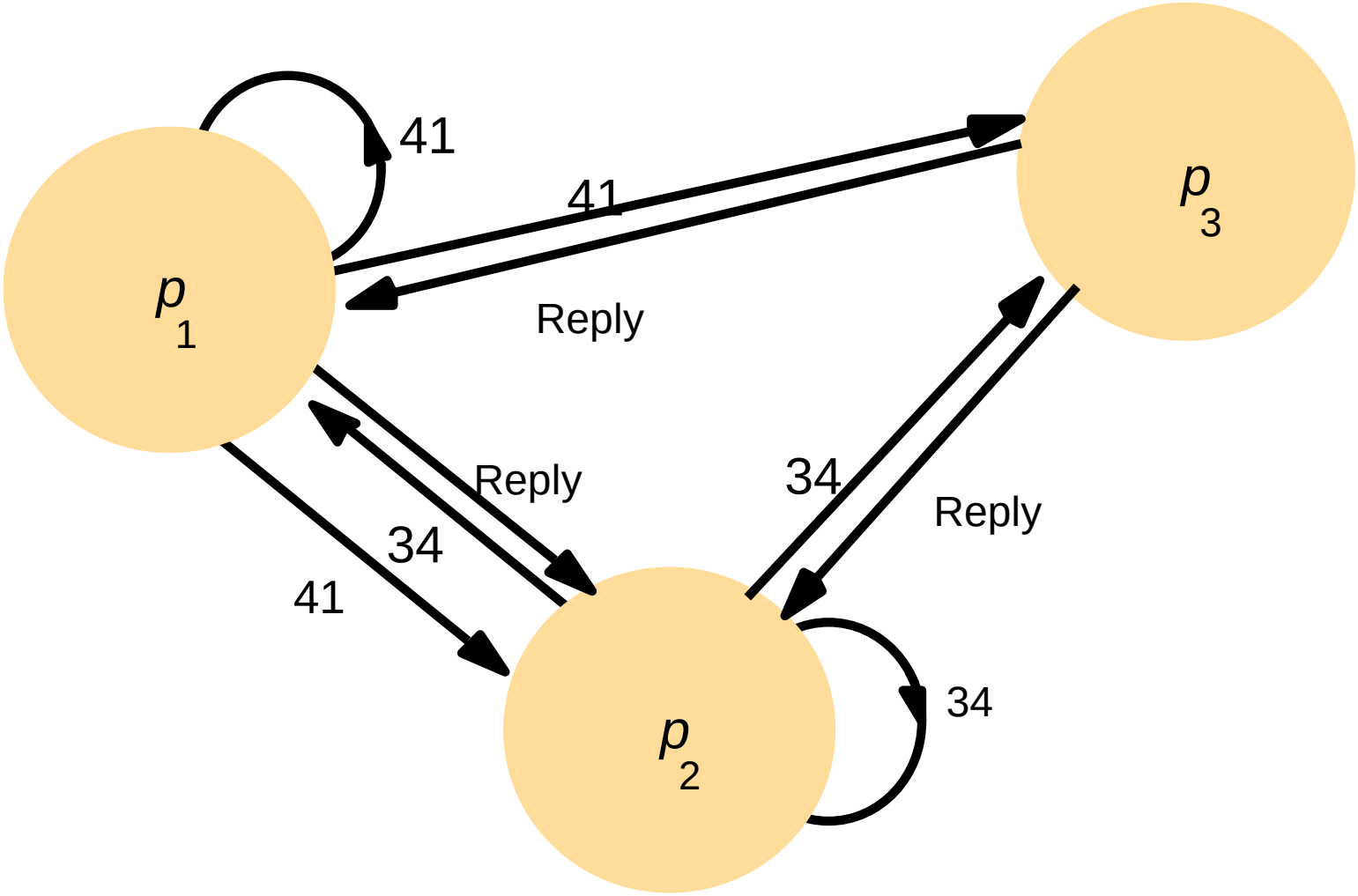
Ricart and Agrawala

On initialization
    state := RELEASED;
To enter the section
    state := WANTED;
    Multicast *request* to all processes;
    T := request's timestamp;
    Wait until (number of replies received = (*N* – 1));
    state := HELD;

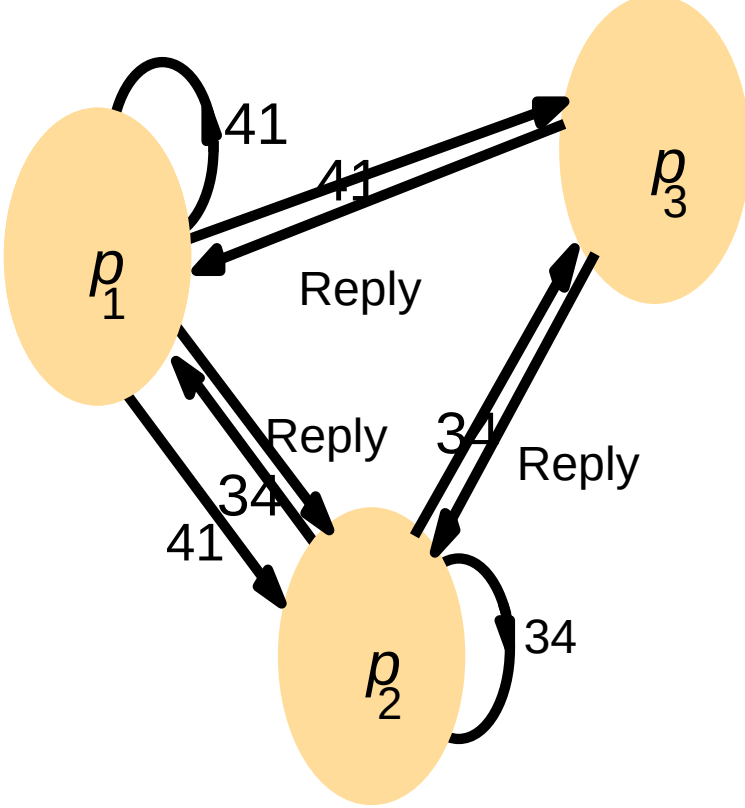
request processing deferred here

On receipt of a request $<T_i, p_i>$ *at* p_j $(i \neq j)$
    if (*state* = HELD or (*state* = WANTED *and* $(T, p_j) < (T_i, p_i)$))
    then
        queue *request* from p_i without replying;
    else
        reply immediately to p_i;
    end if
To exit the critical section
    state := RELEASED;
    reply to any queued requests;

Exclusão Mútua Distribuída



Exclusão Mútua Distribuída



Suponhamos,

P3 não pretende entrar na s.c.

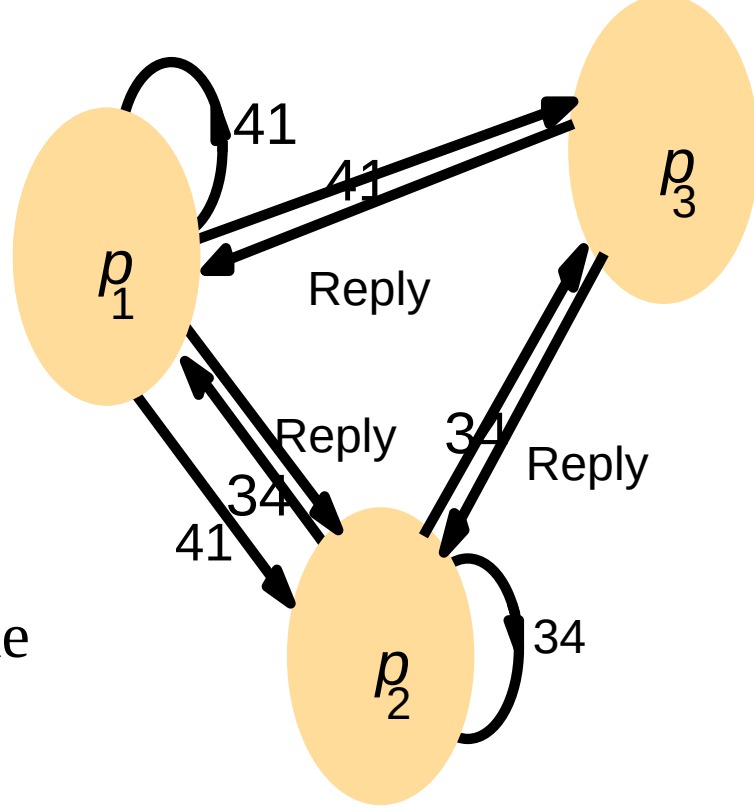
P1 e P2 fazem o pedido em simultâneo

O timestamp de p1 é 41 e o de p2 é 34

P3 recebe o pedido e responde imediatamente

P2 recebe o pedido de P1 e como o seu próprio timestamp é <, não responde, guardando o pedido

Exclusão Mútua Distribuída



P1 detecta que o timestamp do pedido de P2 é < que o seu e portanto responde imediatamente.

P2 ao receber a segunda resposta, entra na s.c.

Quando P2 sair da s.c. irá responder ao pedido de P1, que por sua vez poderá aceder à s.c.

Exclusão Mútua Distribuída

Um Algoritmo Distribuído

- **Bandwidth**

Obter o acesso à s.c. necessita **2(N-1)** mensagens. N-1 para o pedido e N-1 respostas. (Se existir suporte em hardware para o Multicast, o pedido necessita de 1 mensagem + N-1 respostas = total de N mensagens.

Client delay

O atraso do cliente para obter o acesso é um round trip time. **(2)**

Synchronization delay

O tempo para um processo sair da s.c. e outro entrar é o tempo de envio de apenas uma mensagem. **(1)**

Exclusão Mútua Distribuída

Em caso de falha?

O que acontece se um processo avaria *(craches)?*

O que acontece se uma mensagem se perde?

Nenhum dos algoritmos, tolera a perda de mensagens.

- O algoritmo centralizado, tolera avarias em processos clientes que não detenham nem tenham pedido o token
- O algoritmo em anel não tolera a avaria de nenhum processo
- O algoritmo de Ricart e Agrawala não tolera a avaria de nenhum processo.

Eleição de um líder (coordenador)

Um algoritmo para escolher um processo único para desempenhar uma determinada função, designa-se por (election algorithm)

É essencial que:

Todos os processos concordem na escolha

Geralmente é escolhido o processo com o identificador mais elevado.

Os algoritmos tentam localizar o processo com maior identificador e designá-lo como coordenador.

Eleição de um líder (coordenador)

Assumimos que todos os processos sabem o identificador dos outros processos, mas não sabem quais os processos que estão activos e quais não estão.

- **Bully Algorithm (Garcia-Molina)**

- **Ring Algorithm**

Eleição de um líder (coordenador)

Bully Algorithm (Garcia-Molina)

Assume que:

- **o sistema é síncrono.**
- **a entrega de mensagens é fiável**

Funciona mesmo no caso em que um processo falhe durante e eleição

Eleição de um líder (coordenador)

Bully Algorithm (Garcia-Molina)

- Quando um processo não tem resposta de um coordenador, inicia uma eleição

- Quando P inicia uma eleição,
  - P envia a mensagem “ELECTION” para todos os processos com identificador (ID) superior ao seu.
  - Se nenhum processo responde, P vence a eleição e torna-se o coordenador. Envia a mensagem “Coordinator” para todos os outros
  - Se algum processo com maior ID responde, o papel de P na eleição termina.

Eleição de um líder (coordenador)

Bully Algorithm (Garcia-Molina)

- Quando um processo recebe a mensagem (ELECTION) de processos com ID mais baixo,

  - envia a mensagem “OK” para o remetente

- Quando um processo recebe a mensagem “COORDINATOR”

regista o identificador do processo emissor como o novo coordenador.

Eleição de um líder (coordenador)

Bully Algorithm (Garcia-Molina)

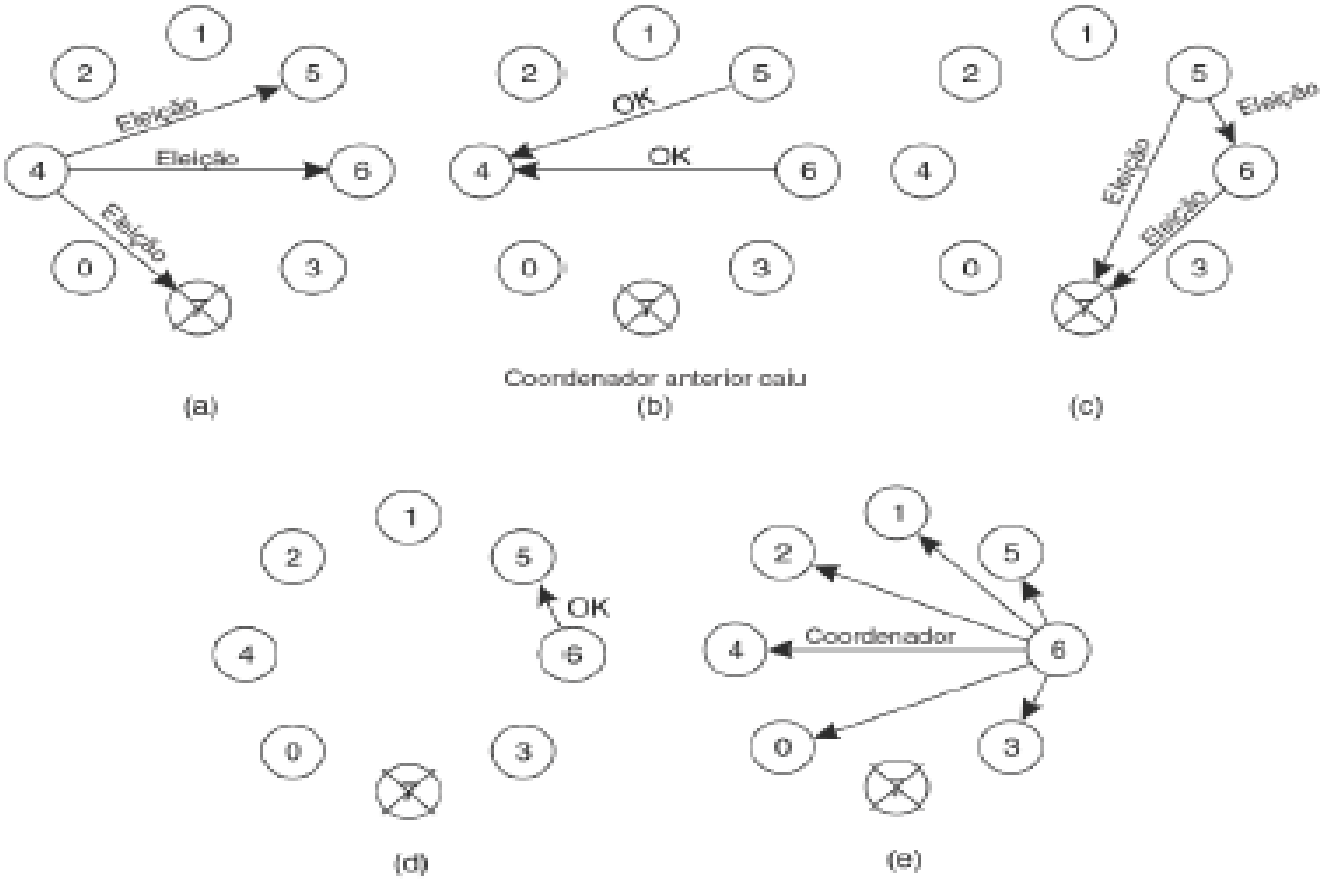
- Se um processo é reinicializado para ocupar o lugar de um processo que avariou, vai iniciar uma eleição.

- Se esse processo tiver o maior dos identificadores, assume-se como o novo coordenador. Envia a mensagem "COORDINATOR" aos outros.

(algoritmo do valentão)

Eleição de um líder (coordenador)

Bully Algorithm (Garcia-Molina)



Eleição de um líder (coordenador)

Ring Algorithm

- Processos organizados num anel lógico.
- As mensagens são enviadas no sentido dos ponteiros do relógio

- ➔ Se um processo verifica que o coordenador não funciona:
- (como?!)

Eleição de um líder (coordenador)

Ring Algorithm

- Envia mensagem de eleição (“ELECTION”) para o seu sucessor

- A mensagem contém o ID do emissor

- Se o sucessor não estiver disponível, envia para o seguinte e assim sucessivamente.

- Em cada passo, o processo que recebe a mensagem, anexa o seu ID e envia para o processo seguinte.

Eleição de um líder (coordenador)

Ring Algorithm

- Quando a mensagem regressa ao processo que iniciou a eleição, o que é verificado porque contém o ID do processo, a mensagem é alterada para "COORDINATOR"

- Vai circular novamente, agora para informar todos os processos de:

-"quem é" o coordenador (processo com o maior ID)

- "quem são" os novos membros do anel.

- Quando a mensagem volta novamente ao processo que iniciou a eleição, é removida.

Eleição de um líder (coordenador)

Ring Algorithm