Sistemas Distribuídos
Exclusão Mútua

Disciplina: Sistemas Distribuídos
Prof.: Edmar Roberto Santana de Rezende

Faculdade de Engenharia de Computação
Centro de Ciências Exatas, Ambientais e de Tecnologias
Pontifícia Universidade Católica de Campinas

Exclusão Mútua

Introdução

- Processos concorrentes:
  - necessidade de ler ou atualizar dados compartilhados
  - → dois processos não podem acessar os dados compartilhados ao mesmo tempo
- Regiões críticas:
  - processo entra em uma região crítica para garantir a exclusão mútua na execução de determinado trecho de código
  - → nenhum outro processo vai usar os dados compartilhados ao mesmo tempo
- Um único processador:
  - regiões críticas protegidas com o uso de:
    - semáforos
    - monitores
    - construções similares
- Como implementar exclusão mútua em sistemas distribuídos?

Exclusão Mútua

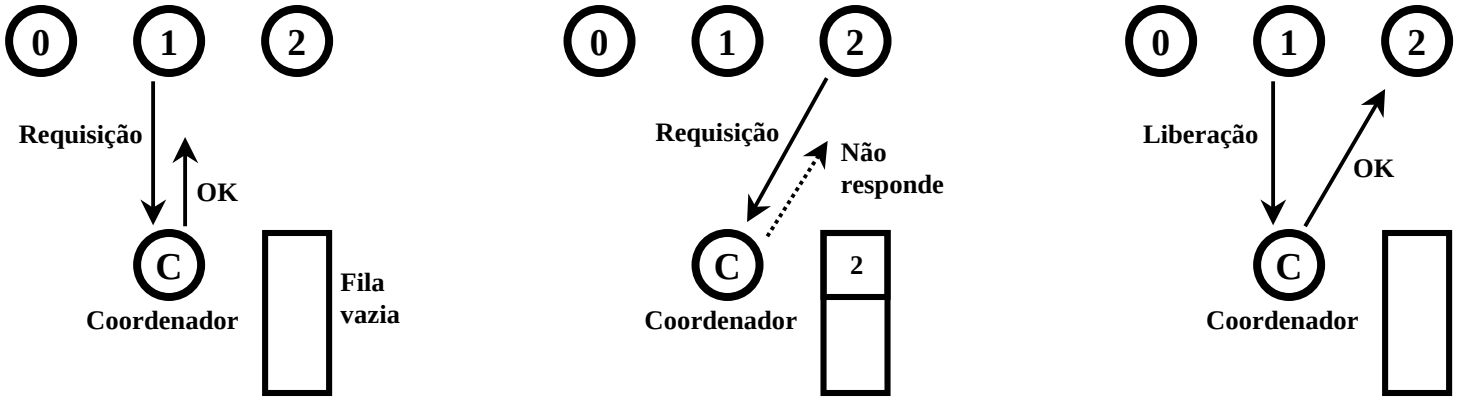
Algoritmo Centralizado

- Idéia:
  - simulação da metodologia de um sistema com um único processador
- Algoritmo:
  1. eleger um processo coordenador:
     - pode ser o processo com maior endereço de rede, maior ID, etc
     - coordenador pode ser estático ou eleito utilizando algum algoritmo distribuído
  2. quando um processo deseja acessar uma região crítica:
     - envia uma mensagem ao coordenador solicitando permissão
     - se nenhum outro processo estiver executando a região crítica:
       - » coordenador envia permissão ao processo que a solicitou
     - se algum processo estiver executando a região crítica:
       - » coordenador não concede a permissão (não responde ou envia mensagem negando a permissão)
       - » requisição é colocada em uma fila
  3. quando o processo deixa a região crítica:
     - envia uma mensagem ao coordenador abrindo mão de seu acesso
     - coordenador envia permissão ao primeiro processo da fila

Exclusão Mútua

Algoritmo Centralizado

- Algoritmo:



- Características do algoritmo:
  - é justo (requisições atendidas na ordem de suas chegadas)
  - nenhum processo espera indefinidamente (não há *starvation*)
  - fácil de ser implementado:
    - necessita de apenas 3 mensagens (requisição, permissão e liberação) para se garantir o acesso a determinada região crítica

Exclusão Mútua

Algoritmo Centralizado

- Problemas:
  - coordenador é um ponto crítico (ponto único de falha)
    - falha do coordenador pode derrubar todo o sistema
  - se coordenador não responde em caso de permissão negada:
    - processos bloqueados não têm como distinguir a situação “coordenador fora do ar” da situação “permissão de acesso negado”
  - sistema muito grande:
    - um único coordenador pode degradar a performance do sistema (possível gargalo)
- Existência de um ponto único de falha:
  - indesejável em sistemas distribuídos

Exclusão Mútua

Algoritmo Distribuído

- Lamport (1978):
  - algoritmo para sincronização de clocks lógicos
  - primeiro algoritmo distribuído para exclusão mútua
- Ricart e Agrawala (1981):
  - tornaram o algoritmo de Lamport mais eficiente
- Exigência:
  - ordenação global de todos os eventos do sistema
  - → para qualquer par de eventos deve haver um consenso sobre qual deles aconteceu antes
- Solução:
  - algoritmo de Lamport para sincronização de clocks lógicos

Exclusão Mútua

Algoritmo Distribuído

- Algoritmo:

1. quando um processo deseja entrar em uma região crítica:
   - envia mensagem para todos os processos (inclusive ele próprio) contendo:
     - o nome da região crítica, seu próprio número e o tempo corrente
2. quando um processo recebe uma mensagem de requisição:
   a) se o receptor não estiver executando a região crítica e não deseja executar:
     - envia de volta ao transmissor um mensagem de OK

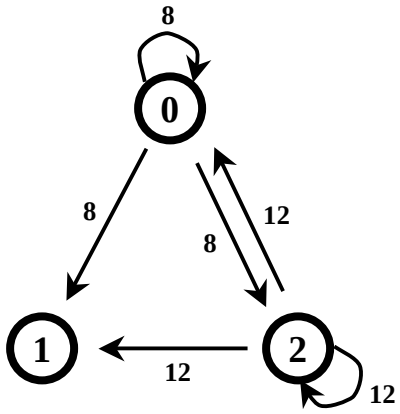
   b) se o receptor estiver executando a região crítica:
     - não deve responder
     - guarda a requisição em uma fila

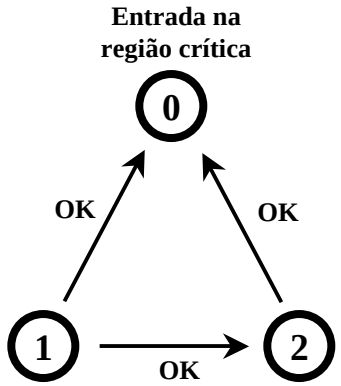
   c) se o receptor também deseja executar a região crítica, mas ainda não entrou:
     - compara o tempo da mensagem recebida com o tempo da mensagem de requisição que ele enviou
       - » se o tempo da mensagem recebida for menor envia um OK ao transmissor
       - » se o tempo de sua própria mensagem for menor coloca a requisição recebida em uma fila e não responde
3. após enviar uma requisição para executar uma região crítica:
   - aguarda até que todos os demais processos lhe dêem permissão
4. ao terminar a execução da região crítica:
   - envia mensagem de OK a todos os processo de sua fila

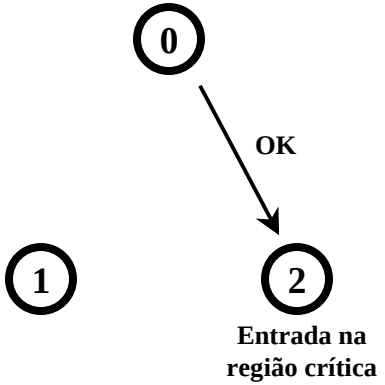
Exclusão Mútua

Algoritmo Distribuído

- Algoritmo:







- Características do algoritmo:
  - nenhum processo espera indefinidamente (não há *starvation*)
  - não há impasses (*deadlocks*)
  - número de mensagens por entrada na região crítica:
    - **2 (n-1)** mensagens
    - **n:** número de processos do sistema

Exclusão Mútua

Algoritmo Distribuído

- Problemas:
  - o ponto único de falha foi substituído por **n** pontos de falha
    - a falha de um processo bloqueia todas as tentativas subsequentes de acesso à região crítica
  - tráfego gerado na rede é muito maior
  - sistema muito grande:
    - todos os processos se tornam possíveis gargalos
- Resultado:
  - funciona melhor para pequenos grupos de processos
  - pode ser melhorado:
    - envio de negação de acesso à região crítica
    - processo pode entrar na região crítica quando obtiver a permissão da maioria dos processos

Exclusão Mútua

Algoritmo Token Ring

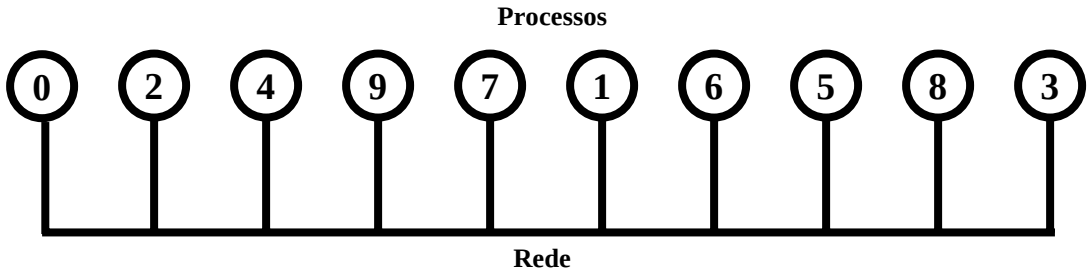
- Método completamente diferente:
  - construção de um anel lógico (ordenação dos processos)
    - atribui-se a cada processo uma posição no anel:
      - » ordem numérica de endereços de rede
      - » qualquer outro meio conveniente
    - não importa a ordenação:
      - » o importante é que cada processo conheça o próximo na sequência

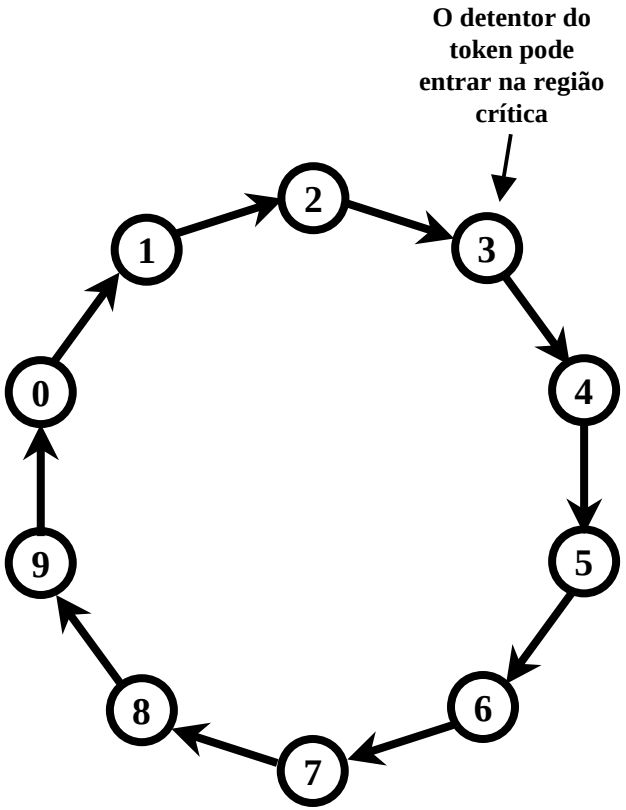
- Algoritmo:
  1. quando o anel é inicializado o processo **0** recebe um bastão (*token*)
  2. se um processo estiver de posse do *token* e deseja entrar em uma região crítica:
     - executa a região crítica (não é permitido que ele entre em uma segunda região crítica)
     - ao terminar a execução da região crítica envia o token ao seu vizinho
  3. se um processo estiver de posse do *token* e não deseja entrar em uma região crítica:
     - envia o token ao seu vizinho

Exclusão Mútua

Algoritmo Token Ring

- Algoritmo:





Exclusão Mútua

Algoritmo Token Ring

- Características do algoritmo:
  - não há ocorrência de *starvation*
  - quando um processo deseja entrar em uma região crítica:
    - o que pode acontecer de pior é ter que esperar que todos os processos antes dele executem alguma região crítica
- Problemas:
  - se o token se perder
    - há a necessidade de regenerá-lo
  - difícil detectar a perda do token:
    - o fato do token não aparecer por muito tempo não significa que ele esteja perdido
  - se algum processo falhar
    - seu vizinho deve identificar a falha e removê-lo do anel
      - » é necessário que todos conheçam a configuração do anel

Exclusão Mútua

Comparação

- Algoritmo Centralizado:
  - mais simples e mais eficiente
  - nº de mensagens por entrada na região crítca: **apenas 3** (requisição, autorização e liberação)
- Algoritmo Distribuído:
  - nº de mensagens por entrada na região crítca : **2(n-1)**
    - **n-1** requisições
    - **n-1** autorizações
- Algoritmo Token Ring:
  - nº de mensagens por entrada na região crítca: **variável** e **imprevisível**
    - se todos os processos desejam entrar na região crítica: **1 mensagem**
    - token pode circular por horas sem nenhum processo interessado: **imprevisível**

Exclusão Mútua

Comparação

- Algoritmo Centralizado:
  - retardo para entrar na região crítica:
    - apenas o tempo de **2 mensagens**

- Algoritmo Distribuído:
  - retardo para entrar na região crítica:
    - o tempo equivalente a **2(n-1) mensagens**

- Algoritmo Token Ring:
  - retardo para entrar na região crítica:
    - o tempo varia:
      - token acaba de chegar: **0 mensagens**
      - token acaba de ser passado: **n-1 mensagens**