Distributed Systems - Mutual Exclusion

O que é Exclusão Mútua?

Mecanismo que garante que apenas um processo por vez pode acessar um recurso compartilhado (ex: banco de dados, arquivo, seção crítica).

Em sistemas distribuídos, os processos estão em máquinas diferentes e a comunicação é feita via rede, **não há memória compartilhada nem relógio global,** então a coordenação precisa ser feita **via troca de mensagem**.

Objetivo principal:
Evitar condições de corrida (race conditions), inconsistência de dados e conflitos de acesso.

Algoritmos

O livro apresenta alguns algoritmos clássicos. Os mais importantes:

Algoritmo centralizado

Um processo coordenador controla o acesso ao recurso.
Processo envia **request**, recebe **grant**, e após uso, envia **release**.
Simples, porém tem ponto único de falha e pode causar gargalos.

Algoritmo descentralizado com votação

Cada processo solicita permissão a um conjunto de processos (quorum).
Precisa obter permissão da maioria para entrar na seção crítica.
Mais robusto, mas mais mensagens são trocadas.

Algoritmo distribuído de Ricart-Agrawala

Usa carimbos de tempo (Lamport's logical clocks) e mensagens REQUEST, REPLY, RELEASE.
Cada processo mantém uma fila de pedidos ordenada pelo timestamp.
Todos os processos devem acknowledger antes de entrar na seção crítica.
Requer (2n – 1) mensagens por entrada na seção crítica.

Token Ring

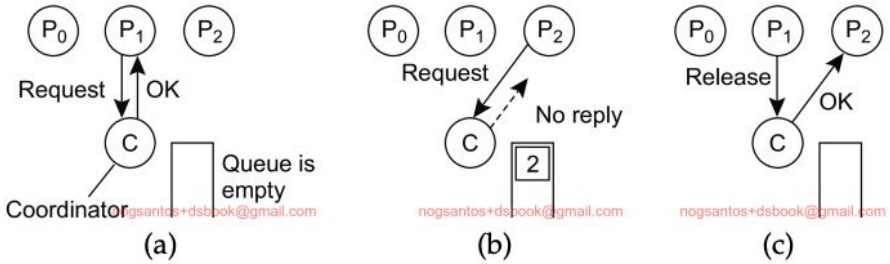
Um token é passado em anel lógico entre os processos.
Quem tem o token pode entrar na seção crítica.
Muito eficiente (1 mensagem por entrada), mas recuperar token perdido é difícil.

Algoritmo centralizado

Um coordenador central que atua como um "porteiro" do recurso crítico. Todo processo que quiser acessar o recurso deve pedir permissão ao coordenador. Ele mantém o controle e concede ou nega o acesso com base na disponibilidade.

Passos

1. **Requisição:**
   - Um processo P1 envia uma mensagem `REQUEST` para o coordenador.
2. **Decisão:**
   - Se **nenhum outro processo** está na seção crítica, o coordenador responde com `GRANT`.
   - Caso contrário, ele **enfileira o pedido**.
3. **Uso da seção crítica:**
   - O processo P1 entra na seção crítica após receber o `GRANT`.
4. **Liberação:**
   - Quando termina, o processo envia `RELEASE` ao coordenador.
   - O coordenador então concede `GRANT` ao **próximo processo na fila** (se houver).



Algoritmo centralizado

Vantagens

- Simples de entender e implementar
- Baixo número de mensagens por entrada na seção crítica (eficiente em redes pequenas)
- Coordenador pode implementar políticas de prioridade, fairness, timeout, etc.

Desvantagens

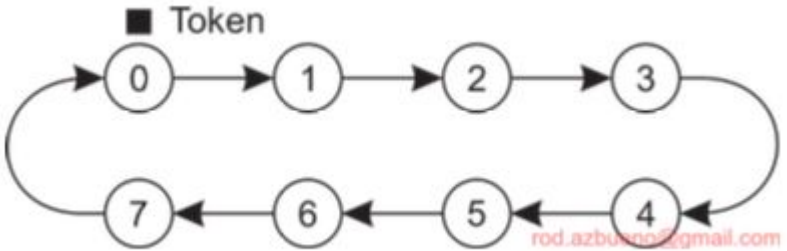
- Ponto único de falha (SPOF): se o coordenador falhar, o sistema trava.
- Gargalo de desempenho: o coordenador pode virar um bottleneck se muitos processos fizerem requisições simultaneamente.
- Não é escalável para sistemas com muitos nós ou tráfego intenso.

Aplicações típicas

- Útil para **ambientes controlados ou pequenos clusters**
- Situações com **baixa concorrência pelo recurso**
- Laboratórios, simulações ou ambientes onde a tolerância a falhas não é crítica

Token-ring algorithm

Algoritmo de exclusão mútua distribuída.
Cria um anel lógico entre os processos.
Cada processo sabe quem é o próximo no anel.
Um único token circula entre os processos.



Como o Token Ring Funciona

Inicialização: Um processo (ex: Processo 0) recebe o token.
O token circula de processo em processo, na ordem do anel.
Quando um processo recebe o token:
- Se quiser acessar a seção crítica → Entra → Executa → Sai → Passa o token.
- Se não quiser → Passa o token imediatamente.
Ninguém pode entrar duas vezes seguidas com o mesmo token.

Garantias Oferecidas

- Exclusão mútua garantida: Só o processo com o token pode acessar o recurso.
- Sem Starvation: Todos os processos recebem o token em sua vez.
  a. Starvation = Fome / Inanição:
     Quando um processo fica esperando indefinidamente e nunca consegue acessar o recurso.
  b. O Token Ring evita isso, porque o token circula por todos.
- Ordem definida: O token segue sempre a mesma sequência.

Problema de Falhas no Token Ring

- Se um processo **cai (crash)** ou **fica offline**, o token pode ficar "preso" tentando passar para ele.
- Se o token for perdido (ex: processo que tinha o token falha), o sistema precisa de uma forma de recuperação.

Recuperação de Falhas - Abordagem Simples

- O processo tenta passar o token para o próximo.
- Se falhar várias vezes seguidas (ex: 10 tentativas), o processo considera o vizinho como "morto".
- O token é então passado para o próximo processo da lista.
- A lista de participantes fica localmente atualizada em cada nó.

Casos de Uso Reais

- Jogos online baseados em turnos.
- Sistemas de controle de acesso a recursos compartilhados.

Show me the code:

https://github.com/rodolfobueno/distributed-system-mutual-exclusion