Aula 15- Exclusão Mútua Distribuída

Monday, May 2, 2016 13:43

Considere um recurso (resource) acessado pelos processos de um sistema distribuído.

No contexto da exclusão mútua, o recurso é chamado de "região crítica" (RC).
Na verdade: RC corresponde à parte do processo que executa quando este tem acesso ao re ecurso.

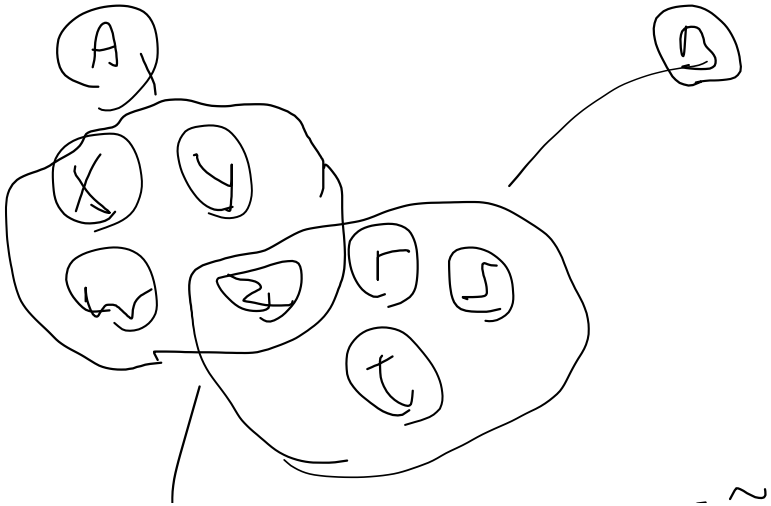
Definição da exclusão mútua

Em um determinado instante de tempo, no máximo 1 processo acessa a região crítica.

Existem 3 categorias de algoritmos de exclusão mútua distribuída.

1. Baseados em permissão. (<u>só este na disciplina</u>)
2. Baseados em token (ou bastão). Uma mensagem especial que dá acesso à RC.
3. Baseados em quorum. Um quorum é um subconjunto dos processos do sistema distrib uído. i.e. Se o sistema tem N processos, o quorum tem $\sqrt{N}$ ou $\frac{N}{2}+1$ ou $logN$ processos.

O quorum funciona de uma maneira tal que se dois ou mais processos fazem duas ou mais r equisições concorrentemente, pelo menos um processo do sistema recebe e ordena as requisições.



Quóruns são feitos para que sempre haja uma interseção.

Exclusão Mútua: modelo de sistema

Mutex = Mutual Exclusion

O sistema consiste de N processos (1, 2, ..., N) e é assíncrono.

Um processo que deseja acessar a região crítica envia uma mensagem REQUEST e, depois de uma troca de mensagens que resulta em sucesso, acessa a região crítica.

Uma premissa importante:
Após fazer uma requisição, um processo não pode fazer outra requisição antes de acessar e liberar a região crítica.

- Um processo pode estar em um de 3 estados:
  - Requisitando a R.C.
  - Usando ou "na" R.C.
  - Idle
- Em geral, os processos mantém uma fila (estrutura de dados FIFO) de requisições.

Outras premissas:
- Sistema representável por grafo completo (fully connected).
- Canais de comunicação confiáveis (não perdem mensagens), mas não necessariamente com entrega ordenada.
- Processos não falham.
- Processos não estão sincronizados, mas mantém um relógico lógico global. Vamos considerar que quanto menor o timestamp, maior a prioridade.

Propriedades MUTEX

Safety (segurança) -> nada de ruim acontece
- No máximo um único processo pode acessar a R.C. em um instante de tempo.

Liveness (progressão) -> algo de bom acontece
- Não ocorrem nem deadlocks nem starvation (manição)
  - Deadlock: dois ou mais processos aguardam indefinidamente por mensagens que nunca chegam

- Starvation: um processo aguard indefinidamente para acessar a RC, enquanto o: s demais continuamente acessam

Além da safety e da liveness, uma terceira propriedade é

Fairness (justiça) - todos os processos têm a mesma chance de acessa a RC (nenhum proces: so é priorizado).

Métricas de desempenho MUTEX

Notação:

- N: número de processos
- T: atraso médio de uma mensagem
- E: tempo médio de execução da R.C.

1. Número de mensagens necessárias para 1 processo acessar a RC
2. Atraso de sincronização: depois que um processo deixa a RC, quanto tempo demora p ara o processo seguinte executar a RC
3. Tempo de resposta: neste contexto, é o intervalo entre a REQUISIÇÃO e a entrada na I RC
4. Throughput (vazão): quantas requisições são atendidas por unidade de tempo

Em geral, o desempenho do MUTEX é avaliado em carga ALTA e BAIXA.

Carga baixa - pequeno número de requisições, em geral 1 por instante de tempo.

Carga alta - sempre há pelo menos 1 requisição pendente; em outras palavras: após a execu ção da RC por um processo, outro processo vai executar a RC.

Desempenho melhor e pior casos:

Melhor caso neste contexto. É o próprio round trip time: RTT = 2T+E
(2* atraso médio da mensagem mais tempo médio de execução) - acontece em carga baixa)

Pior caso: depende do algoritmo. Em geral: carga alta.

Algoritmo de Lamport para exclusão mútua distribuída

- Os processos mantêm um relógio lógico. Seja t o contador (timestamp) do processo i.
- Cada processo mantém uma fila de requisições **ordenadas pelo relógio lógico**.
- Canais de comunicação FIFO

- Canais de comunicação FIFO

Canal de comunicação FIFO: ordena as mensagens de 1 fonte (1 origem)

O algoritmo tem três partes: Requisição da RC, execução da RC, liberação da RC.

Requisição da RC:

- Quando o processo i deseja executar a RC, envia REQUEST(ti, i) para todos os processos e enfilera a requisição na RQi.
- Quando o processo j recebe REQUEST(ti, i) do processo i, enfilera em RQj e retorna REPLY para i com tj.

Execução da RC:

- O processo i entra na RC quando
  - L1: i recebeu mensagens com timestamp maior que ti de todos os processos
  - L2: a requisição de i está na frente de RQi.

Liberação da RC:

- Quando sai da RC, o processo i remove a sua própria requisição da RQi e envia uma mensagem RELEASE para todos os outros processos
- Quando o processo j recebe o RELEASE do processo i, remove o REQUEST(ti, i) do RQj.