Sistemas Distribuídos

7 Coordenação e Acordo

- **Coordenação e Acordo**

Profª Ana Cristina B. Kochem Vendramin
DAINF / UTFPR

Exclusão Mútua

- Evitar interferência entre um conjunto de processos e garantir a consistência no acesso aos recursos compartilhados.
- Quando um processo precisar atualizar, por exemplo, uma estrutura de dados compartilhada, ele primeiro precisa entrar em uma seção crítica (SC).
- Após entrar na SC e acessar o recurso desejado, o processo sairá da SC.

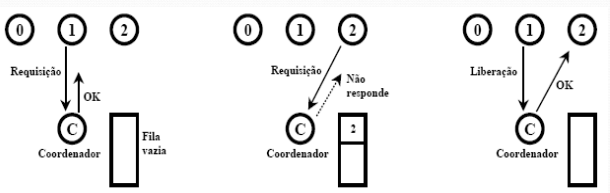
Exclusão Mútua

- Requisitos de exclusão mútua:
  - **Segurança**: no máximo um processo por vez pode ser executado na seção crítica;
  - **Ordenação**: utilizar a ordem antes do acontecido (*happened-before*) para ordenar a entrada na seção crítica;
  - **Subsistência**: os pedidos para entrar/sair de seção crítica têm sucesso.
    - Implica em independência de:
      - **impasse** (interdependência mútua entre dois ou mais processos fazendo com que eles travem indefinitamente);
      - **inanição**: adiamento indefinido da entrada de um processo que a solicitou.

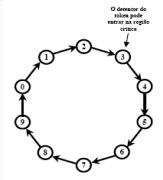
Algoritmo do Servidor Central

- Para entrar em uma seção crítica, um processo envia uma mensagem de pedido para o servidor e espera a resposta.
- A **resposta** representa um ***token*** significando permissão para entrar na **seção crítica**
  - Se nenhum processo tiver o *token*:
    - O servidor concede o *token*
  - Se o *token* estiver de posse de outro processo:
    - O servidor não responderá, mas colocará o pedido em uma fila
- Na saída da seção crítica, uma mensagem é enviada para o servidor, devolvendo o *token*.

Algoritmo do Servidor Central



Algoritmo em Anel



- Não existe papel de servidor ou coordenador
- Processos são organizados em um anel lógico
- Cada processo k precisa ter um canal de comunicação com o processo seguinte do anel
- A obtenção do *token* se dará na forma de uma mensagem passada de um processo para outro em uma única direção.
- Um processo que queira entrar na SC espera o token.
- Para sair da SC, processo envia o token ao vizinho.
- Se um processo não pede para entrar na seção crítica, ao receber o *token* ele o encaminhará ao seu vizinho.

Algoritmo com *multicast* e relógios lógicos

- Desenvolvido por Ricart e Agrawala para implementar exclusão mútua entre N processos pares usando *multicast*.
- Processos solicitam a entrada em uma SC enviando uma mensagem para o grupo *multicast* e só podem entrar nela quando todos os outros processos do grupo responderem.
- As mensagens que solicitam entrada na SC são da forma:
  - <Ti, Pi>
  - Onde:
    - Ti é a indicação de tempo do remetente;
    - Pi é o identificador do remetente.

Algoritmo com multicast e relógios lógicos

- Cada processo registra seu estado em uma variável
  - **RELEASED –** processo fora da SC;
  - **WANTED** – deseja entrar na SC;
  - **HELD** – encontra-se na SC.
- Se um processo pede a entrada na SC e os estados dos outros processos for RELEASED, então todos responderão imediatamente ao pedido e o solicitante obterá a entrada.
- Se algum processo estiver no estado HELD, então esse processo não responderá aos pedidos até que tenha terminado com a SC.

Algoritmo com multicast e relógios lógicos

- Se um ou mais processos solicitarem entrada ao mesmo tempo, o pedido do processo que apresentar a indicação de tempo mais baixa será o primeiro a coletar as respostas garantindo a próxima entrada.
- Se apresentarem indicações de tempo iguais, serão ordenados de acordo com os identificadores correspondentes dos processos.

Algoritmo com multicast e relógios lógicos

- **Inicialização**
  - estado:= RELEASED;
- **Para entrar na SC**
  - estado:= WANTED;
  - Envia o pedido para o grupo multicast ;
  - **T**:= indicação de tempo do pedido;
  - espera até (num resposta = (N-1));
  - estado:= HELD;
- **Pj recebe um pedido <Ti, pi>**
  - **if** (estado = HELD or (estado = WANTED and (Tj, pj) < (Ti, pi))) **then**
    - coloca na fila o pedido de pi sem responder;
  - **else** responde imediatamente para pi;
- **Para liberar o RC**
  - estado := RELEASED;
  - responde a todos os pedidos enfileirados.

Algoritmo de Votação Maekawa

- Um processo não precisa da autorização de todos os outros processos para entrar na SC
  - Precisa obter permissão de apenas um subconjunto votante.
  - Os subconjuntos usados por quaisquer dois processos precisam se sobrepor.
- Cada processo está em tantos conjuntos de votação quantos são os elementos em cada um desses conjuntos.

Algoritmo de Votação Maekawa

- Processos não precisam da autorização de todos para acessar o RC → somente de um subconjunto
- Como dividir os processos em subconjuntos ?
  - Para todo p_i (i = 1, 2, …, N)
    - Associa-se um conjunto votante (ou de eleitores) V_i tal que
    - $p_i \in V_i$
    - $V_i \cap V_j \neq \emptyset$ deve haver pelo menos um membro comum dados dois subconjuntos votantes
    - $|V_i| = K$ todos os subconjuntos votantes tem mesmo tamanho
    - Cada processo p_j está contido em M subconjuntos votantes

Algoritmo de Votação Maekawa

- Para **acessar a SC**, pi envia pedido *multicast* para todos os K membros de Vi (incluindo ele mesmo)
- **Não pode entrar na SC** até que tenha recebido as K respostas
- **Quando libera a SC**, envia msg de liberação para todos os membros de Vi
- Quando um processo pj em Vi **recebe o pedido** de pi
  - **Se estado=HELD** ou **já votou** desde a última msg de liberação
    - Não responde e enfileira o pedido de pi
  - **Senão**
    - Envia msg de resposta imediatamente (voto)
- Quando pj **recebe mensagem de liberação** da SC de pi
  - **Remove cabeça da fila** e **envia uma resposta** (voto)

Algoritmos de Eleição

- Escolher um único processo (coordenador) para desempenhar uma função em particular
- **Algoritmo baseado em anel**
  - Conjunto de processos organizados em anel lógico
  - Cada processo tem um canal de comunicação com o processo seguinte no anel (sentido horário)
  - Escolhe o coordenador que possui maior identificador
  - Qualquer processo pode iniciar um eleição colocando seu ID em uma mensagem de eleição e enviando a seu vizinho.

Algoritmos de Eleição

- **Algoritmo baseado em anel**
  - Quando um processo recebe uma msg de eleição, ele compara o ID presente na msg com o seu próprio.
  - Se o ID recebido é menor que o seu, o receptor substitui o ID pelo seu próprio ID e encaminha a mensagem.
  - Se o ID recebido for o do próprio receptor, então este se tornará o coordenador.
    - Anuncia sua eleição ao vizinho

Algoritmos de Eleição

- **Algoritmo valentão (*bully*)**
  - Permitem falhas de processos durante uma eleição
  - Presume troca de mensagens confiáveis
  - Presume sistema síncrono - tempos limites para detectar falhas de processos
  - Presume que cada processo sabe quais processos têm ID mais altos
  - Mensagens de Eleição;
  - Mensagens de Resposta à uma mensagem de eleição;
  - Mensagens de Coordenador: anuncia ID do processo eleito.

Algoritmos de Eleição

- **Algoritmo valentão (*bully*)**
  - O processo que sabe que possui o ID mais alto pode eleger a si mesmo como coordenador
    - Envia mensagem de coordenador a todos os processos com ID menor
  - Processo com ID mais baixo, envia mensagem de eleição para os processos com ID mais alto e espera resposta
    - Se nenhuma resposta chegar dentro de T segundos, o processo se considera coordenador
    - Se recebe resposta, configura sua variável eleita com o ID do coordenador

Referências Bibliográficas

- Coulouris, George; Dollimore, Jean; Kindberg, Tim. Distributed Systems Concepts and Design. Third Edition. Addison-Wesley 2001.
- Coulouris, George; Dollimore, Jean; Kindberg, Tim; tradução João Tortello. Sistemas Distribuídos: conceitos e projeto. 4. ed. Bookman 2007.
- Tanenbaum, A.S. Distributed Operating Systems. Prentice-Hall International, 1995.