

MC714

Sistemas Distribuídos

2º semestre, 2014

Exclusão mútua

Exclusão mútua

- SDs Concorrência e colaboração entre vários processos.
- Acesso simultâneo aos mesmos recursos.
- Necessário evitar que recurso seja corrompido ou torne-se inconsistente.
- Acesso mutuamente exclusivo pelos processos.

Exclusão mútua

- Duas categorias de algoritmos de exclusão mútua
 1. Soluções baseadas em ficha (*token*).
 - Passagem de mensagem especial entre processos – ficha
 - Há somente uma ficha disponível
 - Quem tem a ficha pode acessar o recurso
 - Ao terminar, ficha é passada diante para o próximo processo
 - Se processo não precisar acessar o recurso, passa ficha adiante.

Exclusão mútua

1. Soluções baseadas em ficha (*token*).

- Pode garantir com razoável facilidade que todo processo terá oportunidade de acessar o recurso → evita inanição (starvation).
- Fácil evitar deadlocks → contribui para otimização do processo.
- Desvantagem: ficha pode se perder → precisa-se de procedimento distribuído para criar nova ficha única.

2. Abordagem baseada em permissão.

- Processo que quer acessar o recurso solicita permissão aos outros processos.

Algoritmo centralizado

- Simular como exclusão mútua é feita em monoprocessador.
- Processo é eleito coordenador.
- Outro processo quer acessar recurso compartilhado:
 - Envia mensagem ao coordenador declarando qual recurso quer acessar e solicitando permissão.
 - Se recurso está livre, coordenador responde com permissão.
 - Se não está livre, pode negar permissão ou não responder até estar livre.
- Fig. 101.

Algoritmo centralizado

- Garante exclusão mútua
- Permissões concedidas na ordem
- Não há inanição
- Fácil de implementar. Três tipos de mensagem:
 - Requisição
 - Concessão
 - Liberação
- Coordenador é ponto de falha único
 - Falha implica queda do sistema
 - Se processo bloqueia depois de pedir recurso, pode ficar bloqueado
 - Coordenador pode ser gargalo

Algoritmo descentralizado

- Lin et al. (2004): algoritmo de votação que pode ser usado sobre uma DHT.
- Ampliação do coordenador central.
- Premissa: cada recurso é replicado n vezes.
- Toda réplica tem seu próprio coordenador para controlar acesso concorrente.
- Quando processo quer acessar um recurso, precisa de voto majoritário $m > n/2$ coordenadores.
- Coordenador informa ao requisitante se não der permissão.

Algoritmo descentralizado

- Torna solução centralizada original menos vulnerável a falhas.
- Premissa: quando um coordenador falha, se recupera rapidamente, mas esquece votos que tenha dado antes de falhar (coordenador *reinicia*).
 - Risco de conceder permissão dupla ao recurso que gerencia.

Algoritmo descentralizado

- p : probabilidade de um coordenador reiniciar durante um intervalo de tempo Δt .
- $P[k]$ probabilidade de que k entre m coordenadores se reiniciem durante o mesmo intervalo.

$$P[k] = \binom{m}{k} p^k (1 - p)^{m-k}$$

Algoritmo descentralizado

- Pelo menos $2m - n$ coordenadores precisam reiniciar para violar (não garantir) correção do mecanismo.

- Probabilidade de ocorrer: $\sum_{k=2m-n}^n P[k]$

- Ex: nós ficam 3h em DHT

- $\Delta t = 10s$

- $n=32, m=0,75n$

- Probabilidade de violação menor que 10^{-40}

Algoritmo descentralizado

- Usando DHT → recurso replicado n vezes
- Nome exclusivo *rname*
- i -ésima réplica: *rname-i*, usada para calcular chave
- Dado o nome do recurso, todo processo pode gerar as n chaves, podendo então consultar os nós responsáveis pelas réplicas.

Algoritmo distribuído

- Algoritmo distribuído determinístico para exclusão mútua
- Lamport (1978), aprimorado por Ricart e Agrawala (1981)
- Requer ordenação total de todos os eventos no sistema
 - Para qualquer par de eventos, como mensagens, não pode haver ambiguidade sobre o qual aconteceu primeiro.
 - Algoritmo de Lamport é um modo de conseguir essa ordenação.

Algoritmo distribuído

- Processo quer acessar recurso compartilhado:
 - Monta mensagem que contém nome do recurso, seu número de processo e hora (lógica) corrente.
 - Envia mensagem a todos os outros processos, incluindo ele mesmo.
 - Premissa: envio de mensagens é confiável.

Algoritmo distribuído

- Processo recebe uma mensagem com requisição: ação depende do seu próprio estado em relação ao recurso solicitado na mensagem.
 1. Receptor não está acessando recurso e não quer acessá-lo → responde OK.
 2. Receptor já tem acesso ao recurso → Não responde e põe requisição na fila.
 3. Receptor quer acessar mas ainda não o fez → compara marca de tempo da mensagem que chegou com a que enviou para todos. Mais baixa vence: OK se sua é mais alta; c.c. enfileira sem responder

Algoritmo distribuído

- Após enviar requisições de permissão, processo espera até que todos tenham dado permissão.
- Ao terminar, envia mensagem de OK para todos que estão em sua fila e remove todos da fila.
- Fig. 102.

Algoritmo distribuído

- Exclusão mútua garantida sem deadlock
- Número de mensagens: $2(n-1)$.
- Não existe ponto de falha único... mas existem n pontos de falha.
 - Se um processo falha, não responde;
 - Requisitante pode ficar bloqueado, assumindo erroneamente que processo que falhou está ocupando recurso.
 - Bloqueará todas as tentativas de todos os processos de entrar em região crítica.
 - Solução?

Algoritmo distribuído

- Solução?
- Receptor enviar respostas negativas.
- Requisição/resposta perdida: refaz até receber resposta ou declarar processo morto.
- Requisição negada: bloqueia até receber um OK.
- Precisa de primitiva de comunicação multicast
 - Ou cada nó mantém lista de associados com controle de entrada/saída de nós
- Gargalo continua existindo: todos lidam com todas as requisições.
- Pior que centralizado, mas mostra que é possível

Token ring

- Participantes formam anel lógico
- Ficha (token) passada ao longo do anel
 - Quem recebe usar recurso, se precisar, e repassa ficha.
- Ficha pode ser perder
- Precisa diferenciar falha de demora no uso do recurso

Algoritmos de eleição

Algoritmos de eleição

- Objetivo: escolher um líder – processo que seja coordenador dentre um conjunto de processos.
- Suposição: cada processo tem um número exclusivo (endereço de rede, por exemplo).
- Abordagem geral: procurar processo com número mais alto e designá-lo como líder.
- Algoritmos variam na maneira de localizar tal processo.

Algoritmos de eleição - suposições

- Todo processo sabe o número de todos os outros.
- Processos não sabem quais estão funcionando e quais estão inativos.
- Meta do algoritmo de eleição: garantir que, quando uma eleição começar, ela terminará com todos os processos concordando com o novo coordenador escolhido.

Algoritmos de eleição tradicionais

- **1. Algoritmo do valentão**
- Processo P qualquer nota que coordenador não está mais respondendo, inicia eleição da seguinte forma:
 - Envia mensagem ELEIÇÃO a todos os processos de número mais alto
 - Se nenhum responder, P vence a eleição e se torna coordenador
 - Se algum responder, este toma o poder e P conclui seu trabalho.

Algoritmos de eleição tradicionais

- Processos podem receber mensagem ELEIÇÃO a qualquer momento de nós com número mais baixo.
- Receptor envia OK de volta ao remetente, indicando que está vivo e que tomará o poder.
- Receptor convoca uma eleição (a não ser que já tenha convocado uma)
- Converge para situação onde todos desistem, exceto um, que será o novo coordenador.
 - Este anuncia a vitória enviando a todos os processos informando que ele é o novo coordenador.

Algoritmos de eleição tradicionais

- Processo inativo convoca eleição quando volta.
- Se for processo de número mais alto, ganha.
- Mais “poderoso” sempre ganha.
- Fig. 103.

Algoritmos de eleição tradicionais

- **2. Algoritmo de anel**
- Processos ordenados por ordem física ou lógica
 - Cada um sabe quem é seu sucessor
- Processo nota que coordenador não responde → envia mensagem ELEIÇÃO, contendo seu número de processo.
 - Se sucessor não estiver respondendo, envia ao próximo membro ao longo do anel. Repete até encontrar um processo funcional.
- Cada nó adiciona seu número de processo à mensagem, candidatando-se a coordenador.

Algoritmos de eleição tradicionais

- Quando mensagem retornar ao iniciador da eleição:
 - Extrai maior número de processo que encontrar na mensagem.
 - Circula mensagem COORDENADOR com tal número, além dos participantes do anel.
 - Mensagem COORDENADOR chegou ao iniciador, é removida.
- O que ocorre se dois processos iniciarem eleição?
- Fig. 104.

Eleição em ambiente sem fio

- Sem premissa de que troca de mensagem é confiável e topologia não muda.
 - Em especial redes ad hoc.
- Vasudevan et al. → pode eleger o melhor líder

Eleição em rede ad hoc sem fio

- Qualquer nó, chamado nó fonte, pode iniciar uma eleição
 - Envia mensagem ELEIÇÃO a seus vizinhos imediatos (nós que estão ao seu alcance).
- Nó recebe ELEIÇÃO pela primeira vez:
 - Designa remetente como seu pai
 - Envia mensagem ELEIÇÃO a todos seus vizinhos imediatos (exceto pai)
 - Se nó recebe ELEIÇÃO de vizinho que não é seu pai, se limita a reconhecer o recebimento.

Eleição em rede ad hoc sem fio

- Quando um nó **R** designou um nó **Q** como seu pai, repassa mensagem ELEIÇÃO aos vizinhos imediatos, exceto **Q**.
- Aguarda que reconhecimentos cheguem antes de reconhecer e ELEIÇÃO recebida de **Q**.
 - Vizinhos que já tem pai respondem imediatamente a **R**.
 - Se todos os vizinhos já tem pai, **R** é um nó folha e pode responder a **Q** rapidamente.
 - Resposta inclui informações (tempo de vida útil da bateria, capacidades dos recursos)

Eleição em rede ad hoc sem fio

- **Q** enviou mensagem ELEIÇÃO porque seu pai, **P**, havia enviado a ele.
- Quando **Q** reconhecer mensagem de **P**, **Q** passará o nó mais qualificado a **P**.
- O nó fonte saberá qual o melhor nó
 - Seleciona como líder
 - Transmite decisão em broadcast
- Fig. 105.

Eleição em sistema de grande escala

- Seleção de mais de um nó
 - Ex.: Superpares
- Requisitos:
 - Nós normais devem ter baixa latência de acesso a superpares
 - Superpares devem estar uniformemente distribuídos pela rede de sobreposição
 - Deve haver uma porção definida de superpares em relação ao número total de nós na rede de sobreposição
 - Cada superpar não deve precisar atender mais do que um número fixo de nós normais

Eleição em sistema de grande escala

- DHT: idéia básica é reservar uma fração do espaço de identificadores para superpares.
- Ex.: reservar os primeiros k bits da esquerda dos identificadores (de m bits).
- Se precisamos de N superpares, então os primeiros $\lceil \log_2(N) \rceil$ bits de qualquer chave podem ser usados para identificar esses nós.

Eleição em sistema de grande escala

- Ex.: no. $m=8$, $k=3$.
- Consultar chave **p**: nó responsável por **p** AND 11100000, que é tratado como superpar
- Cada nó **id** pode verificar se é um superpar: **id** AND 11100000 para ver se requisição é roteada para si.

Eleição em sistema de grande escala

- Outra abordagem: baseada em posicionamento geográfico dos nós.
 - N fichas distribuídas para N nós
 - Cada nó só pode ter 1 ficha
 - Ficha tem força de repulsão para outras
 - Nós devem saber da existência de outras fichas
 - Gossiping para disseminar força das fichas.
 - Se nó descobre que força agindo sobre ele é maior que um patamar, transfere ficha.
- Fig. 106.