Sistemas Distribuídos

Eleição de Líder

no modelo parcialmente síncrono - GST

Prof. Elias P. Duarte Jr.
Universidade Federal do Paraná (UFPR)
Departamento de Informática
www.inf.ufpr.br/elias/sisdis

Sumário

- O Modelo:
  - falhas por parada com recuperação (*crash-recovery*)
  - parcialmente síncrono com GST
- Eleição de Líder: Definição & Propriedades
- Detector de falhas ◊P
- Um Algoritmo para Eleição de Líder no modelo de parada com recuperação

Eleição de Líder

- “Líder” em sistemas distribuídos:
  - Pode funcionar como “coordenador” de diversas tarefas distribuídas
  - Ou pode ser simplesmente uma “referência” para os demais processos processos do sistema
- Vamos ver que um algoritmo de eleição de líder é, de certa forma, equivalente a um detector de falhas
- Ao invés de detectar quais processos falharam → escolher **1** processo correto como líder
- Objetivo: todos os processos têm o mesmo líder
- Na verdade é o Detector de Falhas Ω

Modelo de Falhas: Parada com Recuperação

- *Crash-Recovery Model*
- Veja no modelo de falha por parada (crash) simples: processos podem recuperar e voltar!
- Mas perdem o estado interno completamente
- Quando recuperam, voltam sem qualquer informação de estado

Modelo de Falhas: Parada com Recuperação

- No modelo crash-recovery: os processos *não* perdem informações de estado
- Como fazer?

Modelo de Falhas: Parada com Recuperação

- No modelo crash-recovery: os processos não perdem informações de estado
- Como fazer?
- O único jeito é ter memória não volátil para guardar as informações necessárias
- Quais informações são necessárias? Depende do algoritmo distribuído sendo executado ;-)

Modelo Temporal: Parcialmente Síncrono com GST

- GST: *Global Stabilization Time*
- O sistema é assíncrono até um instante de tempo, justamente denominado GST

Modelo Temporal: Parcialmente Síncrono com GST

- GST: *Global Stabilization Time*
- O sistema é assíncrono até um instante de tempo, justamente denominado GST
- A partir do GST: o sistema passa a se comportar como síncrono
- Dizemos que o algoritmo de eleição de líder que vamos estudar é “após um tempo” (*eventual,* em inglês)

Propriedades da Eleição de Líder

- **Precisão** após um tempo: existe um instante de tempo a partir do qual todo processo correto tem como líder um processo correto
- **Acordo** após um tempo: existe um instante de tempo a partir do qual todos os processos corretos têm como líder o mesmo processo correto
  - *no two correct processes trust as leader different correct processes*

Modelo GST: Há Período de Instabilidade

- Durante um tempo é possível que vários processos corretos tenham vários líderes...
- ... e que fiquem mudando de líder! na medida em que suspeitam de falhas e deixam de suspeitar
  - lembre-se: GST inicialmente assíncrono!
- Entretanto é garantido que a partir de um instante vai estabilizar
  - se não ocorrer em uma situação real: não corresponde ao modelo :-P

Algoritmo Eventual Leader Election

- O líder é determinado usando um Detector de Falhas:

  → o líder é o processo correto de menor identificador
- No modelo de falhas é crash-recovery: processos falham e recuperam
- O algoritmo mantém uma variável *encarnação*, que informa quantas vezes o processo falhou e recuperou
- O processo inicia com *encarnação* ← 1; cada vez que recupera faz *encarnação*++

Monitoramento com Heartbeats

- Periodicamente cada processo correto envia um *heartbeat* para todos os outros processos
- O *heartbeat* carrega a encarnação do processo: *send(heartbeat, encarnação)*

Seleção do Líder

- O algoritmo mantém um conjunto de *Candidatos:* processos dois quais recebeu hearbeat no último intervalo de heartbeats
- Critério de seleção de líder: candidato correto com:
  - que falhou/recuperou menos vezes, ou seja tem o menor valor de *encarnação*
  - com menor identificador
- Em outras palavras: daqueles processos que falharam e recuperaram menos vezes, o que tem menor id

Detector de Falhas ◊P

- Este algoritmo usa um detector de falhas *eventually perfect* (perfeito após um tempo): ◊P
- ◊P se diz “diamante P”
- A implementação clássica deste detector é assim:
  - Toda vez que muda o líder (portanto o líder atual falhou)
  - Incrementa o intervalo de envio de *heartbeats*
  - O motivo é evitar falsas suspeitas

  “the delay will keep increasing until it becomes large enough for the leader to stabilize when the system becomes synchronous”

O Algoritmo Eleição de Líder (1)

Algoritmo Eleição de Líder
// Modelo: Parcialmente Síncrono GST, Falhas Crash-Recovery

Init: Líder ← 0; // o processo 0 (zero) é o líder inicial
encarnação ← 1; **store**(encarnação);
para todo processo i ∈ S faça: send(heartbeat, encarnação);
Candidatos ← Φ;
startTimer(IntervaloHeartbeats);

Upon Recovery: Líder ← 0; **retrieve**(encarnação);
encarnação++; **store**(encarnação);
para todo processo i ∈ S faça: send(heartbeat, encarnação);
Candidatos ← Φ;
startTimer(IntervaloHeartbeats);

O Algoritmo Eleição de Líder (2)

Algoritmo Eleição de Líder: continuação
// Modelo: Parcialmente Síncrono GST, Falhas Crash-Recovery

Upon IntervaloHeartbeats: NovoLíder ← Seleciona(Candidatos);
se NovoLíder ≠ Líder
então IntervaloHeartbeats++; // após suspeita de falha, incr. intervalo heartbeats
Líder ← NovoLíder;
para todo processo i ∈ S faça: send(heartbeat, encarnação);
Candidatos ← Φ;

Upon receive(heartbeat,enc) from j:
se ∃ (j,e) ∈ Candidatos | e < enc // tinha o j em candidatos com enc menor
então Candidatos ← Candidatos - (j,e);
Candidatos ← Candidatos U (pj,enc); // insere o j com enc atual

Por que a eleição funciona?

- Atende a propriedade de precisão após um tempo
- Vamos supor que não, não atende a precisão: existe um processo correto que elege como líder um processo falho z :-0
- Podemos imaginar 2 situações:
  - z é um processo falho que nunca mais recupera
  - z é um processo que fica falhando, isto é: falha intermitente e continuamente

Precisão: Líder Falho?

- No caso do processo z falhar e não recuperar mais, não corre o risco de ser eleito
- Pois não vai mandar heartbeats mais!
- Depois de um tempo vai ser suspeitado e excluído no conjunto de *Candidatos*
- Outro processo vai ser eleito

Precisão: Líder Falha/Recupera Continuamente

- Cada processo que falha e recupera incrementa seu número de *encarnação*
- Esta informação é gravada em memória não-volátil
- E se mentir? Impossível! O modelo aqui não é bizantino!!
- Neste caso sua encarnação vai sendo incrementada e se houver um processo correto este vai ser eleito!

Observação Avançada

- Mesmo com canal de comunicação perfeito...
- Pode ser que os heartbeats deste processo que falha/recuperem continuamente nunca sejam recebidos!

Observação Avançada

- Mesmo com canal de comunicação perfeito...
- Pode ser que os heartbeats deste processo que falha/recuperem continuamente nunca sejam recebidos!
- Pois o canal de comunicação perfeito é construído sobre um enlace fair-loss e portanto se a origem falha não há garantia de entrega :-P

A Eleição Funciona: Acordo

- Será que a eleição garante o acordo?
- Todos os processos corretos elegem *o mesmo processo correto como líder?*
- Considere que chegamos ao período síncrono do modelo GST
- Vamos pensar nos processos corretos e que não falham mais: seja C o conjunto destes processos

A Eleição Funciona: Acordo

- Considere o conjunto de *Candidatos* mantido por um processo correto
- O conjunto C está contido em *Candidatos*: pois estão corretos transmitindo heartbeats sobre canais de comunicação perfeitos
- Além desses:
  - se houver um processo que falha para sempre: depois de um tempo será removido
  - se houver um processo que fica continuamente falhando e recuperando, sua *encarnação* vai crescendo
- Como a função que seleciona líder é determinística: todos elegem o mesmo líder – processo de menor id entre os que têm menor *encarnação*

Referências:

Este algoritmo é descrito no nosso livro texto

Subseção 2.5.5

Conclusão (1)

- Nesta aula vimos um algoritmo para um modelo diferente de todos os anterires da disciplina
- Modelo parcialmente síncrono com GST (leva em conta falsas suspeitas durante o período assíncrono)
- Modelo *crash-recovery*: processos mantém informações em memória secundária
- O algoritmo usa um detector de falhas ◊P: fica incrementando *timeout* a cada suspeita

Conclusão (2)

- No algoritmo da eleição do líder os processos corretos enviam *heartbeats* para todos os demais a cada intervalo com sua *encarnação*
- Cada processo correto mantém um conjunto de *Candidatos*, dos quais recebeu *heartbeat* na última rodada
- É eleito líder o processo de menor identificador dentre os *Candidatos* com menor *encarnação*
- A eleição garante a precisão e o acordo após um tempo

Obrigado!
Página da Disciplina
Sistemas Distribuídos:
www.inf.ufpr.br/elias/sisdis