Eleição de Líder

Algoritmos Distribuídos

Professora: Lúcia Drummond

Eleição de Líder

- Um **Líder** é um membro em N que todos reconhecem como diferente para executar alguma tarefa especial.

- Assumimos que um nó conhece:
  - Sua própria identificação, e;
  - Que todas as identificações são diferentes e podem ser ordenadas.

Eleição de Líder

- **Versão síncrona** em um grafo completo:
  - No pulso s = 0, todo candidato envia sua identificação para todos os outros nós.
  - No pulso s = 1, todo nó recebeu a identificação dos outros candidatos e pode decidir sobre o líder.

Complexidade de mensagens: $O(n^2)$
Complexidade de tempo global: $O(1)$

Eleição de Líder

- A fim de reduzir a complexidade de mensagens para O(n *log* n) mensagens:
  - Um candidato não envia sua identificação para todos os vizinhos no mesmo pulso.
  - Primeiro se comunica com 1 dos vizinhos, depois com 2 de seus vizinhos, 4 e assim por diante.
  - Assim, o k-ésimo conjunto de vizinhos que um nó envia tem 2^{k-1} vizinhos.
  - Logo, são necessários *log* n conjuntos para abranger todos os vizinhos.

Eleição de Líder

- Quando um candidato envia uma mensagem para um vizinho:
  - Ele está tentando capturar esse vizinho, tornando-se seu dono, de forma que um candidato que tenha capturado todos os nós no final seja eleito o líder.

Eleição de Líder

- Um candidato tem sucesso na captura de um nó, se:

  Sua identificação for
  - Maior do que a dos outros candidatos que estão tentando capturar o mesmo nó no mesmo pulso;
  - Maior do que a identificação do atual “dono” do nó.

Eleição de Líder

- Um nó só prossegue na tentativa de capturar o próximo nó:
  - Se ele obtiver sucesso na atual captura.

    Caso contrário, ele desiste.

Eleição de Líder

- **Versão Assíncrona:**
  - As identificações não são mais usadas como base de comparação, mas apenas para desempate.
  - As comparações são baseadas em nível (*level*) de cada candidato, que representa o número de nós que um candidato conseguiu capturar.

Eleição de Líder

- Para evitar que dois candidatos consigam capturar o mesmo nó, um candidato só considera um nó como capturado quando o "dono" do nó desiste.

- Para isso é enviada uma mensagem de *check* e *eliminate* para o atual "dono" para eliminá-lo.

Eleição de Líder

- Um nó ao receber uma mensagem de captura (*capture*), responde com *ack* ou *nack*, dependendo se a captura teve sucesso ou não.

Algoritmo A_Elect_Leader_C

- **Variáveis**

  $candidate_i$ = **false**;
  $tried_i^j$ = **false** para todo n_j;
  $owner\_id_i$ = **nil**;
  $level_i = 0$;
  $owns_i = 0$;
  $p\_owner\_id_i$ = **nil**;
  $p\_owned\_id_i$ = **nil**;

∈

Algoritmo A_Elect_Leader_C

- **Algoritmo**

Input:

msg_i = **nil**;

Ação if $n_i \in N_0$**:**

$candidate_i$:= **true**;

$owner\_id_i := id_i$;

Seja n_j um nó vizinho de n_i:

$tried_i^j$:= **true**;

Envie $capture(level_i, id_i)$ para n_j;

Algoritmo A_Elect_Leader_C

- **Algoritmo**

Input:

$msg_i = capture(level_j, id_j)$ tal que $origem(msg_i) = n_j$;

Ação:

if $p\_owner\_id_i = id_j$ **then**

begin

$owner\_id_i := id_j$;

Envie *ack* para n_j;

end

Algoritmo A_Elect_Leader_C

- **Algoritmo**

else
  if ($level_i$,$p\_owner\_id_i$) < ($level_j$,id_j) **then**
   begin
     $level_i := level_j$;
     if $candidate_i$ **then**
      begin
        $candidate_i :=$ **false;**
        $p\_owner\_id_i := id_j$;
        Envie *ack* para n_j;
      end

*

Algoritmo A_Elect_Leader_C

- **Algoritmo**

```
        else
          begin
                p_owner_idᵢ := idⱼ;
                Seja nₖ tal que owner_idᵢ := idₖ;
                Envie check(k) para nⱼ;
          end
      end
    else
      Envie nack para nⱼ;
```

*

Algoritmo A_Elect_Leader_C

- **Algoritmo**

Input:

$msg_i = nack$;

Ação:

if $candidate_i$ **then**

$candidate_i :=$ **false**;

Input:

$msg_i = check(j)$;

Ação:

if $candidate_i$ **then**

Envie $eliminate(level_i, id_i)$ para n_j;

Algoritmo A_Elect_Leader_C

- **Algoritmo**

Input:

$msg_i = eliminate(level_j, id_j)$ tal que

$origem(msg_i) = n_j$;

Ação:

If not $candidate_i$ **then**

Envie *eliminated* para n_j;

else

if $(level_i, id_i) < (level_j, id_j)$ **then**

begin

$candidate_i$:= **false**;

Envie *eliminated* para n_j;

end

else Envie *nack* para n_j;

Algoritmo A_Elect_Leader_C

- **Algoritmo**

Input:

$msg_i = eliminated$

Ação:

If $candidate_i$ **then**

begin

Seja n_j tal que $p\_owned\_id_i = id_j$:

Envie $capture(level_i, id_i)$ para n_j;

end

Algoritmo A_Elect_Leader_C

- **Algoritmo**

Input:

$msg_i = ack$

Ação:

$owns_i := owns_i + 1;$

$level_i := \lfloor \log(\mathrm{owns_i} + 1) \rfloor$;

Seja S tal que $tried_i^{\,j}$ = **false** para todo $n_j \in$ S;

Algoritmo A_Elect_Leader_C

- **Algoritmo**

If $S \neq \{\}$ **then**

begin

Seja n_j um nó em S*;*

$tried_i^{\,j}$:= **true**;

$p\_owned\_id_i = id_j$:

Envie *capture(*$level_i$*,*id_i*)* para n_j;

end

Teoremas

Teorema 5.1:

Para $1 \le k \le \lceil \log n \rceil - 1$, o número máximo de candidatos restantes no pulso *s=2k* no algoritmo S_Elect_Leader_C é $\left\lfloor n / 2^{k-1} \right\rfloor$.

Prova:

No pulso *s=2k,* por (5.3) um candidato deve ter capturado 2^{k-1} nós para ainda ser um candidato, (isto é, deve ter recebido 2^{k-1} *acks*).

A afirmação segue então do fato que, por (5.2), quaisquer dos nós de *n* só podem ser capturados por no máximo um candidato em qualquer pulso.

Teoremas

Corolário 5.2:

O algoritmo S_Elect_Leader_C envia no máximo $2n\lceil \log n \rceil - n$ mensagens de captura e no máximo $n\lceil \log n \rceil$ *acks*.

Prova:

O número inicial de candidatos é no máximo *n*, no pulso *s*=0 no máximo *n* mensagens de captura são enviadas.

Para $1 \leq k \leq \lceil \log n \rceil - 1$, no pulso *s=2k* um candidato envia no máximo 2^k mensagens de captura.

Teoremas

- **Prova (cont):**

  Pelo teorema 5.1, o número de candidatos neste pulso é menor que $\left\lfloor n / 2^{k-1} \right\rfloor$, então o número total de capturas é no máximo:

  $$n + \sum_{k=1}^{\lceil \log n \rceil - 1} \left\lfloor n / 2^{k-1} \right\rfloor \cdot 2^k \leq n + 2n(\lceil \log n \rceil - 1)$$

  $$= 2n(\lceil \log n \rceil - n)$$

  Por (5.2), um nó envia um *ack* no máximo por um pulso ímpar, de forma que o número total de acks não é maior que $n \lceil \log n \rceil$.

Teoremas

Teorema 5.3:

Para $1 \leq k \leq \lfloor \log n \rfloor$, o número máximo de candidatos no nível *k* em qualquer estado global de uma execução do Algoritmo A_Elect_Leader_C é $\left\lfloor n/(2^k - 1) \right\rfloor$.

Prova:

Pela definição de nível, um candidato n_i no nível *k* deve ter capturado pelo menos 2^k-1 nós

$$k = \lfloor \log(owns_i + 1) \rfloor \leq \log(owns_i + 1).$$

O teorema segue então do fato que nenhum dois candidatos pode ser os donos de um mesmo nó em qualquer estado global.

Teoremas

Corolário 5.4:

O Algoritmo A_Elect_Leader_C envolve no máximo $2n\lfloor \log n \rfloor + n$ tentativas de capturas por candidato.

Prova:

Antes de alcançar o nível 1, um candidato tenta capturar exatamente um nó.

Para $1 \leq k \leq \lfloor \log n \rfloor$, enquanto no nível *k* um candidato tenta capturar 2^k nós.

Teoremas

Prova (cont):

Através do Teorema 5.3, o número total de capturas do algoritmo é:

$$n + \sum_{k=1}^{\lceil \log n \rceil} \left\lfloor n/2^k - 1 \right\rfloor \cdot 2^k \le n + \sum_{k=1}^{\lfloor \log n \rfloor} \left(n/2^{k-1} \cdot 2^k \right)$$

$$= 2n\lfloor \log n \rfloor + n$$

Cada captura de nós por um candidato envolve no máximo seis mensagens (captura, confere, elimina, eliminou, captura e *ack*.)