Election in Trees and Rings

T-79.4001 Seminar on Theoretical Computer Science

Ilari Nieminen

21.02.2007

Outline

Outline

Notation

- n is the number of nodes, m is the number of edges
- Standard set of restrictions
  $\mathbf{R} = \{$*Bidirectional Links*, *Connectivity*, *Total Reliability*$\}$
- $\mathrm{N}(x)$ is the set of neighbours of x
- $\mathbf{M}[P]$ is the number of messages needed in protocol P
- $\mathbf{T}[P]$ is the time required in protocol P
- $\mathbf{B}[P]$ is the number of bits needed in protocol P

Election

Leader Election (**Elect**)

- In the initial configuration all entities are in the same state ("available")
- In the goal configuration all but one are in the same state ("follower")
- Can be thought as enforcing restriction *Unique Initiator*

Impossibility Result

- Problem **Elect** is *deterministically unsolvable* under **R**
- Means that there is no protocol that will terminate correctly in finite time
- Easy to prove with two entities when communication delays are unitary

Election's Standard Set of Restrictions

Restriction *Initial Distinct Values* (ID) is chosen to break the symmetry between entities. Set **IR** $=$ **R** $\cup\{\mathrm{ID}\}$ is called the *standard set for election*. id(x) is used to denote the distinct value of entity x.

Elect Minimum

1. Find the smallest value id(x)
2. Elect the entity with that value as a leader

This strategy also solves **Min.**

Elect Minimum Initiator

1. Find the smallest value id(x) among initiators
2. Elect the entity with that value as a leader

Does not solve **Min.**

Elect Root

1. Construct a rooted spanning tree
2. Elect the root of the tree as the leader

Outline

Elect Minimum in Trees

Tree:Elect_Min

- Using saturation, find the smallest value
- $\mathbf{M}\,[Tree : Elect\_Min] = 3n + k_* - 4 \leq 4n - 4$

Elect Root

- *Full Saturation* selects two saturated nodes
- *Tree:Elect_Root* compares the identities of the saturated nodes
- $\mathbf{M}[\mathit{Tree} : \mathit{Elect\_Root}] = 3n + k_* - 2 \leq 4n - 2$

Tree:Elect_Root

```
SATURATED
  Receiving(Election, id)
  begin
      if id(x) < id then
          become LEADER
      else
          become FOLLOWER
      end
      send (Termination) to N(x)-{parent}
  end

PROCESSING
  Receiving(Termination)
  begin
      become FOLLOWER
      send (Termination) to N(x)-{parent}
  end
```

```
Procedure Resolve
begin
    send (Election,id(x)) to parent
    become SATURATED
end
```

Bit Complexity

- Tree:Elect_Root sends two more messages than Tree:Elect_Min
- Number of bits needed is lower for Tree:Elect_Root
- $\mathbf{B}\,[\mathit{Tree} : \mathit{Elect\_Min}] = n(c + \log \mathrm{id}) + c(2n + k_* - 2)$
- $\mathbf{B}\,[\mathit{Tree} : \mathit{Elect\_Root}] = 2(c + \log \mathrm{id}) + c(3n + k_* - 2)$

where $c = O(1)$ denotes the number of bits needed to distinguish between messages.

Outline

Rings

- A ring consists of a single cycle of length n
- Each entity has exactly two neighbours, whose ports are called “right” and “left”
- It is important to note that this labeling might be inconsistent between entities
- Notation: **other** is used to denote N(x)-sender
- Any protocol that elects a leader in a ring can be made to find the minimum value with n additional messages

All the Way

- On becoming awake entity sends a message to one of its neighbours containing its id
- On receiving a message it forwards the message and keeps note of the smallest id seen
- Because the *Message Ordering* restriction is not used, an entity won't know that the election is finished when it receives its value back
- To calculate the size of the ring, a counter is added to the message
- Does not actually need the *Bidirectional Links* restriction

All the Way Protocol

States: S = {ASLEEP, AWAKE, FOLLOWER, LEADER}
S_{INIT} = {ASLEEP}
S_{TERM} = {FOLLOWER, LEADER}

```
ASLEEP
  Spontaneously
  begin
      INITIALIZE
      become AWAKE
  end

  Receiving("Election", value*, counter*)
  begin
      INITIALIZE
      send ("Election", value*, counter*+1) to other
      count := count+1
      min := Min{min, value*}
      become AWAKE
  end
```

```
AWAKE
  Receiving("Election", value*, counter*)
  begin
      if value* ≠ id(x) then
          send ("Election", value*, counter*+1) to other
          min := MIN{min, value*}
          count := count+1
          if known then
              CHECK
          end
      else
          ringsize := counter*
          known := true
          CHECK
      end
  end
```

All the Way Procedures

```
Procedure INITIALIZE
begin
    count := 0
    size := 1
    known := false
    send ("Election", id(x), size) to right; min := id(x)
end

Procedure CHECK  begin
    if count = ringsize then
        if min = id(x) then
            become LEADER
        else
            become FOLLOWER
        end
    end
end
```

All the Way and All the Way Minimum Initiator

- The cost of *All the Way* is easily seen
- $\mathbf{M}\,[AlltheWay] = n^2$
- $\mathbf{T}\,[AlltheWay] \leq 2n - 1$
- By modifying the protocol to find the smallest value among the initiators number of messages can be reduced
- $\mathbf{M}\,[AlltheWay : Minit] = nk_* + n$
- $\mathbf{T}\,[AlltheWay : Minit] \leq 3n - 1$
- The additional n is required to inform the ring of termination.

As Far As It Can

- The drawback of *All the Way* is that every message travels the whole ring
- *All the Way* is modified so that an entity will only forward Election messages if the id in the message is smaller than than the smallest seen so far
- The message with the smallest id will travel the entire ring, so if an entity receives its own id, it knows it is the leader
- The leader notifies the ring to ensure termination

As Far As It Can Message Complexity

- Worst case happens if the ring is "ordered" and all the messages are sent in the "increasing" direction
- $\mathbf{M}\,[AsFar] = n + \sum_{i=1}^{n} i = \frac{n(n+3)}{2}$
- Average case is harder. Let $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$
- $\mathbf{M}\,[AsFar] = nH_n \approx .69n\log_2 n + O(n)$ on average in oriented (or unidirectional) rings
  $\mathbf{M}\,[AsFar] = \frac{\sqrt{2}}{2}nH_n \approx .49n\log_2 n + O(n)$ on average in unoriented rings (assuming half of the "rights" correspond to the clockwise direction)
- $\mathbf{T}\,[AsFar] = \mathbf{T}\,[AlltheWay] + n - 1$

Controlled Distance

- The downside with *As Far As It Can* is that $O(n^2)$ performance is still possible
- *Controlled Distance* has guaranteed $O(n \log n)$ message performance
- Idea is to limit the distance a message can travel and send multiple messages if necessary

Controlled Distance Basics

1. Entity *x* sends a message with its own id, and the message will travel until it is terminated (by a smaller id) or until it reaches a distance *dis*.
2. If the message is not terminated, it will be sent back to its originator. After receiving the returned message, it knows there are no smaller ids on that side of the ring within distance *dis*.
3. To confirm that there are no smaller ids on either side, the entity will send the message in both directions. If they both come back, next time the message will be allowed to travel further.

Controlled Distance Basics (cont.)

4. If at any time an entity receives a message with a smaller id, it will stop trying to win the election
5. If an entity receives its own message back from the other side, it knows it is the leader and notifies the ring

Controlled Distance Correctness

- The correctness can intuitively be understood through the following observations
- Messages containing the smallest id will always travel the maximum allocated distance.
- Every candidate that meets the messages will give up
- Allocated distance is increased monotonically, so at some point, a message with the minimum id will travel through the whole ring

Protocol Control

States: $S = \{ASLEEP, CANDIDATE, DEFEATED, FOLLOWER, LEADER\}$
$S_{INIT} = \{ASLEEP\}$
$S_{TERM} = \{FOLLOWER, LEADER\}$

```
ASLEEP
  Spontaneously
  begin
        INITIALIZE
        become CANDIDATE;
  end

  Receiving("Forth", id*, stage*, limit*)
  begin
        if id* < id(x) then
              PROCESS-MESSAGE
              become DEFEATED
        else
              INITIALIZE
              become CANDIDATE
        end
  end

DEFEATED
  Receiving(*)
  begin
        send (*) to other
        if * = "Notify" then
              become FOLLOWER
        end
  end

CANDIDATE
  Receiving("Forth", id*, stage*, limit*)
  begin
        if id* < id(x) then
              PROCESS-MESSAGE
              become DEFEATED
        else
              if id* = id(x) then
                    NOTIFY
              end
        end
  end

  Receiving("Back", id*)
  begin
        if id* = id(x) then
              CHECK
        end
  end

  Receiving("Notify")  begin
        send ("Notify") to other
        become FOLLOWER
  end
```

Procedures of protocol Control

Procedure INITIALIZE
begin
    stage := 1
    limit := dis(stage)
    count := 0
    send ("Forth", id(x), stage, limit) **to** N(x)
end

Procedure PROCESS-MESSAGE
begin
    limit* := limit* -1
    if *limit* = 0* **then**
        send ("Back", id*, stage*) **to sender**
    else
        send ("Forth", id*, stage*, limit*) **to other**
    end
end

Procedure CHECK
begin
    count := count+1
    if *count = 2* **then**
        count := 0
        stage := stage+1
        limit := dis(stage)
        send ("Forth", id(x), stage, limit) **to** N(x)
    end
end

Procedure NOTIFY
begin
    send ("Notify") **to** right
    become LEADER
end

Message Complexity of Control

- Performance depends on choice of dis(i)
- Let $\text{dis}^{-1}(n)$ denote smallest integer k such that $\text{dis}(k) \geq n$.
- $\mathbf{M}\left[Control\right] \leq n \sum_{i=1}^{dis^{-1}(n)} \left(3 \frac{dis(i)}{dis(i-1)} + 1\right) + n$
- If distance is doubled at each stage
  $\mathbf{M}\left[Control\right] \leq 7n \log n + O(n)$
- $\mathbf{T}\left[Control\right] \leq 2n + \sum_{i=1}^{dis^{-1}(n)} 2dis(i)$