Hierarchical Leader Election Algorithm With Remoteness Constraint

Mohamed Tbarka

October 21, 2019

ENSIAS

Outline

1. Introduction
   - What's Distributed Systems ?
   - Election Algorithms
2. Preliminaries
   - System Model
   - Modeling Asynchronous Dynamic Links
   - Configurations and Executions
   - Problem Definition
3. Leader Election Algorithm
   - Informal Description
   - Nodes, Neighbors and Heights
   - Initial State
   - Goal Of The Algorithm

Outline

1 Introduction

- What's Distributed Systems ?
- Election Algorithms

2 Preliminaries

- System Model
- Modeling Asynchronous Dynamic Links
- Configurations and Executions
- Problem Definition

3 Leader Election Algorithm

- Informal Description
- Nodes, Neighbors and Heights
- Initial State
- Goal Of The Algorithm

What's Distributed Systems ?

A distributed system is a network that consists of autonomous computers that are connected using a distribution middleware. They help in sharing different resources and capabilities to provide users with a single and integrated coherent network.

The Bully Algorithm

As a first example, consider the bully algorithm devised by Garcia-Molina (1982). When any process notices that the coordinator is no longer responding to requests, it initiates an election. A process, P, holds an election as follows:

The Bully Algorithm

As a first example, consider the bully algorithm devised by Garcia-Molina (1982). When any process notices that the coordinator is no longer responding to requests, it initiates an election. A process, P, holds an election as follows:

- *P* sends an *ELECTION* message to all processes with higher numbers.

The Bully Algorithm

As a first example, consider the bully algorithm devised by Garcia-Molina (1982). When any process notices that the coordinator is no longer responding to requests, it initiates an election. A process, P, holds an election as follows:

- *P* sends an *ELECTION* message to all processes with higher numbers.
- If no one responds, P wins the election and becomes coordinator.

The Bully Algorithm

As a first example, consider the bully algorithm devised by Garcia-Molina (1982). When any process notices that the coordinator is no longer responding to requests, it initiates an election. A process, P, holds an election as follows:

- *P* sends an *ELECTION* message to all processes with higher numbers.
- If no one responds, P wins the election and becomes coordinator.
- If one of the higher-ups answers, it takes over. P's job is done.

Election Algorithms

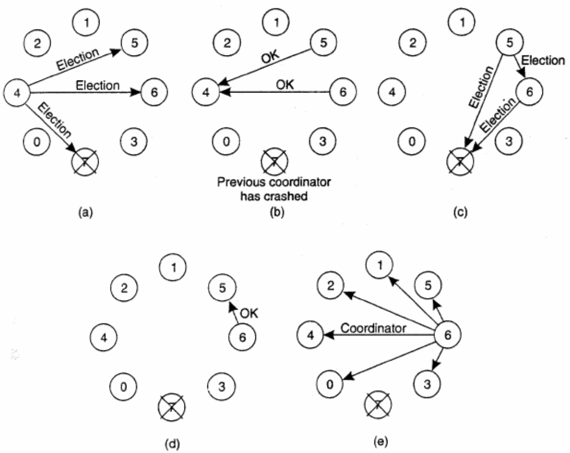


Figure: Bully Algorithm

Outline

1. Introduction
   - What's Distributed Systems ?
   - Election Algorithms
2. Preliminaries
   - System Model
   - Modeling Asynchronous Dynamic Links
   - Configurations and Executions
   - Problem Definition
3. Leader Election Algorithm
   - Informal Description
   - Nodes, Neighbors and Heights
   - Initial State
   - Goal Of The Algorithm

System Model

- We assume a system consisting of a set P of computing nodes and a set χ of directed communication channels from one node to another node. χ consists of one channel.

System Model

- We assume a system consisting of a set P of computing nodes and a set χ of directed communication channels from one node to another node. χ consists of one channel.
- We model the whole system as a set of (infinite) state machines that interact through shared events (a specialization of the IOA model [17]).

Asynchronous Dynamic Links' Model

The state of Channel(u, v), which models the communication channel from node u to node v, consists of:

- a $status_{uv}$ variable;

Asynchronous Dynamic Links' Model

The state of Channel(u, v), which models the communication channel from node u to node v, consists of:

- a $status_{uv}$ variable;
- and a queue $mqueue_{uv}$ of messages.

Configurations & Executions

- The notion of configuration is used to capture an instantaneous snapshot of the state of the entire system.

Configurations & Executions

- The notion of configuration is used to capture an instantaneous snapshot of the state of the entire system.
- A configuration is a vector of node states, one for each node in P, and a vector of channel states, one for each channel in χ.

Problem Definition

Each node u in the system has :

- a local variable lid_u to hold the id of the supreme leader;

Problem Definition

Each node u in the system has :

- a local variable lid_u to hold the id of the supreme leader;
- another local variable $slid_u$ to hold the identifier of the sub-leader whose remoteness towards u obeys the constraint.

Outline

1. Introduction
   - What's Distributed Systems ?
   - Election Algorithms
2. Preliminaries
   - System Model
   - Modeling Asynchronous Dynamic Links
   - Configurations and Executions
   - Problem Definition
3. Leader Election Algorithm
   - Informal Description
   - Nodes, Neighbors and Heights
   - Initial State
   - Goal Of The Algorithm

Heights

After a leader is gone, the algorithm consists on three waves:

- First wave : initiated by one of the lost leader's neighbors looking for it;

Heights

After a leader is gone, the algorithm consists on three waves:

- First wave : initiated by one of the lost leader's neighbors looking for it;
- Second wave : initiated by the node located at the edge of the network if the search has hit a dead-end;

Heights

After a leader is gone, the algorithm consists on three waves:

- First wave : initiated by one of the lost leader's neighbors looking for it;
- Second wave : initiated by the node located at the edge of the network if the search has hit a dead-end;
- Third wave : initiated by the same node which initiated the first wave updating the other nodes' heights.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- *oid*, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- oid, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- r, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- *oid*, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- *r*, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.
- δ

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- oid, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- oid, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- r, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- *oid*, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- *r*, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.
- δ

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- *oid*, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- *oid*, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- r, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- *oid*, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- *r*, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.
- δ

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- oid, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

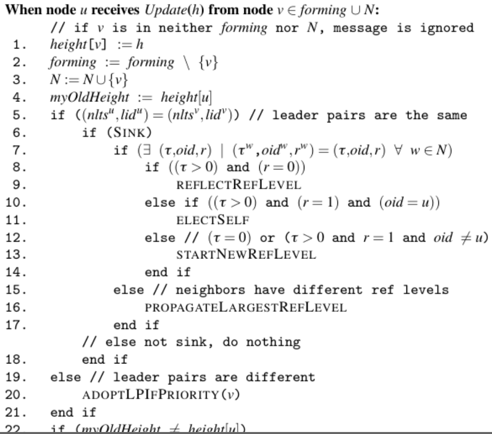
- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- *oid*, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- *r*, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.

Heights

The height for each node is a 7-tuple of integers $((\tau, oid, r), \delta, (nlts, lid), id)$, where the first three components are referred to as the reference level (RL) and the fifth and sixth.

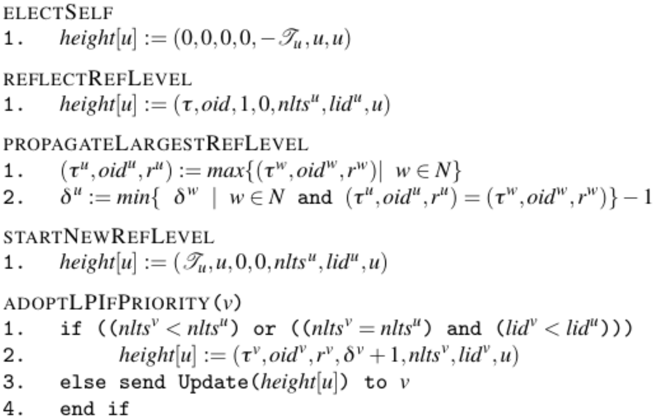
- τ, a non-negative timestamp which is either 0 or the value of the causal clock time when the current search for an alternate path to the leader was initiated.
- oid, is a non-negative value that is either 0 or the id of the node that started the current search (we assume node ids are positive integers).
- r, a bit that is set to 0 when the current search is initiated and set to 1 when the current search hits a dead end.
- δ

The Code triggered by Update Message



```
When node $u$ receives $Update(h)$ from node $v \in forming \cup N$:
    // if $v$ is in neither $forming$ nor $N$, message is ignored
    $height[v] := h$
    $forming := forming \setminus \{v\}$
    $N := N \cup \{v\}$
    $myOldHeight := height[u]$
    if $((nlts^u, lid^u) = (nlts^v, lid^v))$ // leader pairs are the same
        if (SINK)
            if $(\exists\ (\tau, oid, r)\ |\ (\tau^w, oid^w, r^w) = (\tau, oid, r)\ \forall\ w \in N)$
                if $((\tau > 0)$ and $(r = 0))$
                    REFLECTREFLEVEL
                else if $((\tau > 0)$ and $(r = 1)$ and $(oid = u))$
                    ELECTSELF
                else // $(\tau = 0)$ or $(\tau > 0$ and $r = 1$ and $oid \neq u)$
                    STARTNEWREFLEVEL
                end if
            else // neighbors have different ref levels
                PROPAGATELARGESTREFLEVEL
            end if
        // else not sink, do nothing
        end if
    else // leader pairs are different
        ADOPTLPIFPRIORITY$(v)$
    end if
    if $(myOldHeight \neq height[u])$
```

Subroutines



```
ELECTSELF
    $height[u] := (0,0,0,0,-\mathscr{T}_u,u,u)$

REFLECTREFLEVEL
    $height[u] := (\tau,oid,1,0,nlts^u,lid^u,u)$

PROPAGATELARGESTREFLEVEL
    $(\tau^u,oid^u,r^u) := max\{(\tau^w,oid^w,r^w) | \ w \in N\}$
    $\delta^u := min\{ \ \delta^w \ | \ w \in N$ and $(\tau^u,oid^u,r^u) = (\tau^w,oid^w,r^w)\} - 1$

STARTNEWREFLEVEL
    $height[u] := (\mathscr{T}_u,u,0,0,nlts^u,lid^u,u)$

ADOPTLPIFPRIORITY$(v)$
    if $((nlts^v < nlts^u)$ or $((nlts^v = nlts^u)$ and $(lid^v < lid^u)))$
        $height[u] := (\tau^v,oid^v,r^v,\delta^v+1,nlts^v,lid^v,u)$
    else send Update($height[u]$) to $v$
    end if
```

Outline

1 Introduction
- What's Distributed Systems ?
- Election Algorithms

2 Preliminaries
- System Model
- Modeling Asynchronous Dynamic Links
- Configurations and Executions
- Problem Definition

3 Leader Election Algorithm
- Informal Description
- Nodes, Neighbors and Heights
- Initial State
- Goal Of The Algorithm

Outline

1 Introduction
- What's Distributed Systems ?
- Election Algorithms

2 Preliminaries
- System Model
- Modeling Asynchronous Dynamic Links
- Configurations and Executions
- Problem Definition

3 Leader Election Algorithm
- Informal Description
- Nodes, Neighbors and Heights
- Initial State
- Goal Of The Algorithm

What's JBotSim ?

Outline

1 Introduction
- What's Distributed Systems ?
- Election Algorithms

2 Preliminaries
- System Model
- Modeling Asynchronous Dynamic Links
- Configurations and Executions
- Problem Definition

3 Leader Election Algorithm
- Informal Description
- Nodes, Neighbors and Heights
- Initial State
- Goal Of The Algorithm

Questions and Answers

Want to know more?

- Browse `http://web.mit.edu/smoot/history.htm`.

Questions and Answers

Want to know more?

- Browse http://web.mit.edu/smoot/history.htm.
- Smoot's Legacy http://alum.mit.edu/news/AlumniNews/Archive/smoots_legacy.

Questions and Answers

Want to know more?

- Browse http://web.mit.edu/smoot/history.htm.
- Smoot's Legacy http://alum.mit.edu/news/AlumniNews/Archive/smoots_legacy.
- Smoot Salute! http://web.mit.edu/spotlight/smoot-salute.