Hierarchical Leader Election Algorithm With Remoteness Constraint

Mohamed Tbarka

November 10, 2019

Outline

ENSIAS

Outline

What is the problem of leader election ?

Leader election is an important primitive for distributed computing, useful as a sub-routine for any application that requires the selection of a unique processor among multiple candidate processors (video conferencing, multi-player games, ...).

What is distributed computing ?

Distributed computing is a model in which components of a software system are shared among multiple computers to improve efficiency and performance.

State of art

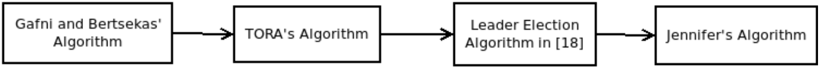


Figure:

Outline

ENSIAS

System Model

- The system is consisting of a set P of computing nodes and a set χ of directed communication channels from one node to another node.

System Model

- The system is consisting of a set P of computing nodes and a set χ of directed communication channels from one node to another node.
- The whole system as a set of (infinite) state machines that interact through shared events.

System Model

- The system is consisting of a set P of computing nodes and a set χ of directed communication channels from one node to another node.

System Model

- The system is consisting of a set P of computing nodes and a set χ of directed communication channels from one node to another node.
- The whole system as a set of (infinite) state machines that interact through shared events.

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.

ENSIAS

Outline

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.

Nodes, Neighbors and 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 components are referred to as the leader pair (LP). In more detail, the components are defined as follows:

Nodes, Neighbors and 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 components are referred to as the leader pair (LP). In more detail, the components are defined as follows:

- 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.

Nodes, Neighbors and 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 components are referred to as the leader pair (LP). In more detail, the components are defined as follows:

- 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.

- 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 an integer that is set to ensure that links are directed appropriately to neighbors with the same first three components.

Initial State

– $forming_u$ is empty,
N_u equals the set of neighbors of u in G_{chan}^{init}

Initial State

- $forming_u$ is empty,
  N_u equals the set of neighbors of u in G_{chan}^{init}
- $height_u[u] = (0, 0, 0, \delta_u, 0, l, u)$ where l is the id of a fixed node in $u's$ connected component in G_{chan}^{init} (the current leader), and δ_u equals the distance from u to l in G_{chan}^{init},

Initial State

- $forming_u$ is empty,
  N_u equals the set of neighbors of u in G_{chan}^{init}
- $height_u[u] = (0, 0, 0, \delta_u, 0, l, u)$ where l is the *id* of a fixed node in $u's$ connected component in G_{chan}^{init} (the current leader), and δ_u equals the distance from u to l in G_{chan}^{init},
- for each v in N_u, $height_u[v] = height_v[v]$ (i.e., u has accurate information about v's height), and

Initial State

- $forming_u$ is empty,
  N_u equals the set of neighbors of u in G_{chan}^{init}
- $height_u[u] = (0, 0, 0, \delta_u, 0, l, u)$ where l is the *id* of a fixed node in $u's$ connected component in G_{chan}^{init} (the current leader), and δ_u equals the distance from u to l in G_{chan}^{init},
- for each v in N_u, $height_u[v] = height_v[v]$ (i.e., u has accurate information about v's height), and
- $\mathcal{T}_u$ is initialized properly with respect to the definition of causal clocks.

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

ENSIAS

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 \ {v}
N := N ∪ {v}
myOldHeight := height[u]
if ((nlts^u, lid^u) = (nlts^v, lid^v)) // leader pairs are the same
    if (SINK)
        if (∃ (τ,oid,r) | (τ^w, oid^w, r^w) = (τ,oid,r) ∀ w ∈ N)
            if ((τ > 0) and (r = 0))
                REFLECTREFLEVEL
            else if ((τ > 0) and (r = 1) and (oid = u))
                ELECTSELF
            else // (τ = 0) or (τ > 0 and r = 1 and oid ≠ 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 ≠ height[u])
    send Update(height[u]) to all w ∈ (N ∪ forming)
end if
```

Subroutines

ElectSelf

1. $height[u] := (0,0,0,0,-\mathcal{T}_u,u,u)$

ReflectRefLevel

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

PropagateLargestRefLevel

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

StartNewRefLevel

1. $height[u] := (\mathcal{T}_u, u, 0, 0, nlts^u, lid^u, u)$

AdoptLPIfPriority (v)

```
1.    if ((nlts^v < nlts^u) or ((nlts^v = nlts^u) and (lid^v < lid^u)))
2.          height[u] := (τ^v, oid^v, r^v, δ^v + 1, nlts^v, lid^v, u)
3.    else send Update(height[u]) to v
4.    end if
```

Outline

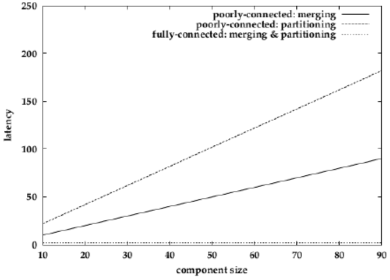
Outline

What's JBotSim ?

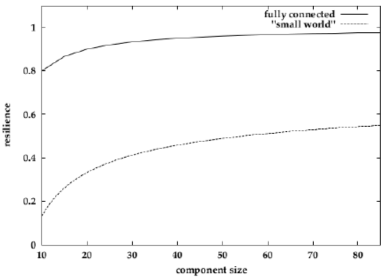
JBoTSim is a java library that offers basic primitives for proto-typing, running, and visualizing distributed algorithms in dynamic networks.

Simulation

How is our algorithm's performance ?



(a) Latency



(b) Resilience

Figure: Simulation Results

Outline

Is The Algorithm Perfect ?

An open question is how to extend our algorithm and its analysis to handle a wider range of clocks, such as approximately synchronized clocks and vector clocks.

Question ?