Elections

Pedro F. Souto (pfs@fe.up.pt)

October 21, 2014

Leader Election

Why Many distributed algorithms rely on a process that plays a special role – **coordinator/leader**. Such algorithms usually are:

- Simpler
- More efficient

What Upon completion of the algorithm all non-faulty nodes agree on who the coordinator is.

- Only one node is elected the coordinator
- All nodes know the identity of the coordinator

Garcia-Molina's Algorithms: Introduction

- The algorithms were proposed in the scope of **system reorganization** upon failure/recovery of system components.
- GM observes that we can ensure fault-tolerance by means of two approaches:

  By masking failures i.e. by using algorithms that continue to work even if some system components fail:
  - This is the only approach if we need continuous operation
  - Likely to be the more appropriate if failures are common also

  By reorganizing the system i.e. by taking some time out to reorganize the system
  - Likely to be allow simpler algorithms
- We abstract the leader election problem from this context
  - This leads to simpler algorithms

Some notes on the paper

This paper is really worth the reading

- It is very well written
- It is an early paper on distributed algorithms and GM explains the issues at length
- It touches on several recurrent issues in distributed systems/algorithms:
  - Fault-tolerance
  - Synchronous vs asynchronous systems
  - Failure detection (and its impossibility in asynchronous systems)
  - Groups of processes
  - RPCs
- GM is very careful/rigorous:
  - Assumptions
  - Specifications
  - Algorithms
- And, in spite of all that, the specification for asynchronous systems is buggy

System Model/Assumptions

1 All nodes cooperate and use the same algorithm

3 The communication subsystem does not spontaneously generate messages

6 There are no transmission errors (but messages may be lost)

7 Messages are delivered in the order in which they are sent

4 All nodes have some **stable(/safe)** storage

5 When a node fails, it immediately halts all processing.
- Crashed nodes may recover
- Data on stable storage is not lost, i.e. is as before the crash

8 The communication system does not fail and has an upper bound on the time to deliver a message, T

9 A node always responds to incoming messages with no delay

Observation Assumptions 8 and 9 mean the system is synchronous.
- The author claims that they are reasonable both for a LAN or a high-connectivity network
- They will be dropped below

Specification: State

- Virtually all distributed algorithms may be described by state machines:
  - Describing the operation of each node (process)
  - Changing their state in response to reception of messages or to the passage of time

$S(i).s$ state of the node i: one of DOWN, ELECTION and NORMAL [a]

- When a node crashes its state changes automatically to DOWN

$S(i).c$ the coordinator according to node i

[a] G-M considers an additional state, but here we are presenting election algorithms independently of their application

Specification of Leader Election

Assertion 1 At any time instant, for any two nodes, if they are both in NORMAL state, then they agree on the coordinator:

$$\forall_{i,j} : S(i).s = S(j).s = \text{NORMAL} \Rightarrow S(i).c == S(j).c$$

Assertion 2 If no failures occur during the election, the protocol will eventually transform a system in any state to a state where:

a) there is a node i such that $S(i).s = \text{NORMAL}$ and $S(i).c = i$
b) all other non-faulty nodes $j \neq i$ have $S(j).s = \text{NORMAL}$ and $S(j).c = i$

Leader Election vs. Mutual Exclusion

1. In an election fairness is not important
   - All we need is that one node becomes the leader
2. An election protocol must deal properly with the failure of the leader
   - Usually, mutual exclusion protocols assume that a process in a critical section does not fail
3. All nodes need to learn who the coordinator is

The Bully Election Algorithm (1/2)

Idea: A node wishing to become a leader challenges other nodes

- Weaker nodes back-off
- Winner brags about becoming the leader

Convention: The smaller a node's identifier the **stronger** it is [a]

---

[a] G-M uses the other convention, but this one has the advantage that we need not know what is the range of the identifiers

The Bully Election Algorithm (2/2)

Phase 1 The node that initiates the election challenges stronger nodes by sending them an ELECTION message

- A stronger node responds to the challenge and initiates a new election (by challenging the nodes stronger than it)
- An initiator whose challenge is answered backs off

Phase 2 Node i begins phase 2, if it does not receive any response to its challenges within T. It comprises two steps:

1. Sends a HALT message to weaker nodes
   - Upon receiving HALT , a node sets its state to ELECTION
2. T time units later, node i sends a COORDINATOR message to weaker nodes, and sets $S(i).c$ to i and $S(i).s$ to NORMAL
   - Upon receiving that message, node k sets $S(k).c$ to i and $S(k).s$ to NORMAL

Comment The HALT message (1st step) is required to ensure that **Assertion 1** is **not** violated

The Bully Election Algorithm and Failures

What about node failures?

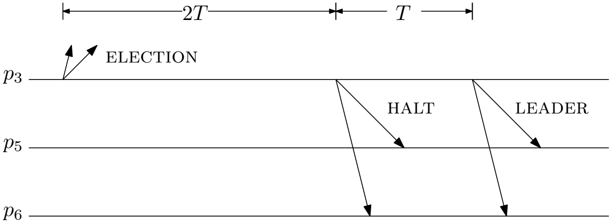
- Failure of the initiator triggers a new election
- Failures of nodes other than the initiator do not matter
  - In some applications, it may also trigger a new election

What about recovery of a node, after a failure?

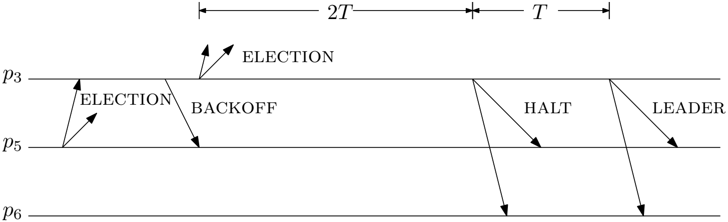
- The node may initiate a new execution of the election algorithm

Bully Algorithm: Example Execution

- Strongest node starts upon failure detection



- Strongest node starts upon challenge



Food for thought Is it possible to remove the HALT message?

Bully Algorithm: Example Execution

What if a node recovers when another node has already initiated an election?

Observation Need to consider two cases:

- Recovering node is stronger
- Recovering node is weaker

Leader Election without Assumptions 8 and 9

If we drop assumptions:

8. The communication system does not fail and has an upper bound on the time to deliver a message, T
9. A node always responds to incoming messages with no delay

Then assertions:

Assertion 1 At any time instant, for any two nodes, if they are both in NORMAL state, then they agree on the coordinator

Assertion 2 If no failures occur during the election, the protocol will eventually transform a system in any state to a state where there is a coordinator

cannot be satisfied **always**:

1. Assume node i is the coordinator, has not crashed but it does not respond to other nodes because it is too slow
2. From the point of view of other nodes, it has crashed, so to satisfy Assertion 2, they must elect a new coordinator
3. But, if they elect a new coordinator, Assertion 1 will be violated

Specification without Assumptions 8 and 9 (1/2)

Groups

Definition Is a set of nodes with a **group id**, i.e. an identifier

- All messages are tagged with the group id
- Not all messages with foreign group ids can be ignored

Node state Includes also the following pieces of information:

$S(i).g$ the current group id;

Specification without Assumptions 8 and 9 (2/2)

Assertion 3 At any time instant, for any two nodes i and j in NORMAL state and in the same group, then they must agree on the coordinator: $S(i).s = \text{NORMAL} \wedge S(j).s = \text{NORMAL} \wedge S(i).g = S(j).g \Rightarrow S(i).c = S(j).c$

- This alone is weak, as it can be satisfied by a singleton group

Assertion 4 Suppose that:

1. there is a set of operating nodes R which all have **two way communication with all other nodes** in R. That is Assumptions 8 and 9 hold for nodes in R
2. there is no superset of R satisfying the previous property
3. no node failures occur during the election

then the election algorithm will eventually transform the nodes in set R from any state to a state where there is i in R such that for every node j

$$S(j).s = \text{NORMAL} \wedge S(j).c = i$$

Note Assertions 1 and 2 are special cases of Assertions 3 and 4

The Invitation Algorithm (1/2)

Idea Rather than imposing itself as a leader, a node wishing to become a coordinator invites others to join in a group where it is the coordinator

- Initially, each node creates a singleton group, of which it is the coordinator
- Periodically, coordinators try to merge their group with other groups in order to form larger groups

Description

Failure detection a node that is not a leader periodically checks if its leader is still alive

- If not, it creates a singleton group of which it is the leader

Group merging a node that is a leader periodically probes all other nodes for leadership

- If one or more nodes reply, node *i* initiates the merging protocol after a delay inversely proportional to its priority
  - The variable delay helps preventing different nodes to initiate the merging concurrently

The Invitation Algorithm (2/2)

1. Node i sends an INVITATION message:
   - to all leaders that have responded
   - to the members of its current group
2. When a leader j receives an INVITATION, it forwards it to the other group members
3. All nodes that receive an INVITATION, directly or indirectly, respond with an ACCEPT message to the candidate (to leader)
4. The candidate adds the sender of each ACCEPT message as group member
5. After time T, **enough(?)** to receive ACCEPT messages from all group members, the new leader sends a READY message to all of them
6. Upon receiving the READY message to a previously sent ACCEPT, node k joins the new group
   - If a node does not receive a READY message after a timeout, it initiates a new election

The Invitation Algorithm: Example

- Consider a group of 6 nodes with ids from 1 to 6, with node 1 as leader
- Let node 1 fail
- Each of the other members forms a singleton group
- Assume that nodes 2 and 3, send invitations to the other nodes and that the conditions on the system are such that:
  - Node 4 accepts the invitation of node 2, leading to one group coordinated by node 2 and members 2 and 4;
  - Nodes 5 and 6 accept the invitation of node 3, leading to one group of coordinated by node 3 and members 5 and 6;
- Some time later, one of the nodes invites the other coordinator to join it in a group
- For a proof of correctness check the paper

Is the Invitation Algorithm Correct?

It appears correct

But Scott Stoller has shown that it does not satisfy Assertion 4

Suppose that:

1. there is a set of operating nodes R which all have **two way communication with all other nodes** in R. That is Assumptions 8 and 9 hold for nodes in R
2. there is no superset of R satisfying the previous property
3. no node failures occur during the election

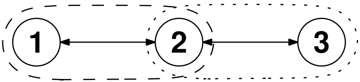
then the election algorithm will eventually transform the nodes in set R from any state to a state where there is i in R such that for every node j

$$S(j).s = \text{NORMAL} \wedge S(j).c = i$$

...when the connectivity is not transitive

Problematic Scenario

1. Node 1 crashes
2. Nodes 2 and 3, each forms a singleton group



3 Node 1 recovers, but communication between nodes 1 and 3 has been lost. Communication between all other pairs of nodes works normally

4 Node 1 forms a singleton group, and invites node 2

5 Nodes 1 and 2 become a group, whereas node 3 becomes a singleton group.

Observation 1 If no more failures occur, these groups will not change

Observation 2 The set $\{2, 3\}$ satisfies the hypotheses on set R in Assertion 4

Contradiction

- Assertion 4 requires that node 2 be coordinator of group $\{2, 3\}$
- Node 2 is a member of group $\{1, 2\}$

Solution (1/2)

Fix the specification The specification is too strong

- It requires processes that are not connected to belong to the same group
- If one of them is the leader, that is not going to happen

Weaken the requirements But that requires a more complex definition

Two nodes are disconnected in a time interval if all messages sent between them during that interval are lost

Stable system in a time interval if, during that interval, no crashes or recoveries occur and every pair of nodes is either connected or disconnected

Connectivity graph, when a system is stable is the undirected graph whose vertices correspond to the nodes and with an edge between vertices i and j iff nodes i and j are connected

Clique cover of a graph is a partition of that graph's nodes into cliques, i.e. fully connected subgraphs

E^* reflexive and transitive closure of relation E

Solution (2/2)

Let $\langle V, E \rangle$ be the system connectivity graph

Assertion 4' For a given system, there is a constant c such that if the system is **stable** for a time interval of duration at least c, then by the end of that interval, the system reaches a state such that

a) $S(i).s = \text{NORMAL} \wedge S(i).g = S(S(i).c).g \wedge (\langle i, S(i).c \rangle \in E^*)$

b) the number of groups is at most the size of a **minimum-sized** clique cover of $\langle V, E \rangle$

Note A clique cover is a partition of a graph's vertices in cliques. E.g. the sets

$$\{\{1,2\},\{3\}\},\{\{1\},\{2,3\}\},\{\{1\},\{2\},\{3\}\}$$

are clique covers of the problematic graph above.

Theorem The Invitation Algorithm satisfies **Assertion 4'**

Proof Check Scott Stoller's paper

Further Reading

- Subsection 6.5, Tanenbaum and van Steen, *Distributed Systems*, 2nd Ed.
- Hector Garcia-Molina, *Elections in a Distributed Computing System*, IEEE Transactions on Computers, Vol. C-31, No. 1, January 1982, pp. 48–59
- Scott Stoller, *Leader Election in Asynchronous Distributed Systems*, IEEE Transactions on Computers, Vol. C-59, No. 3, March 2000, pp. 283–284