Leader Election in Asynchronous Distributed Systems

Scott D. Stoller[*]

January 17, 1999

Abstract

In a classic paper, Garcia-Molina specifies the leader election problem for synchronous and asynchronous distributed systems with crash and link failures and gives an elegant algorithm for each type of system. This paper points out a flaw in Garcia-Molina's specification of leader election in asynchronous systems and proposes a new specification.

Index terms: leader election, group membership, asynchronous distributed systems, crash failures

In a classic paper, Garcia-Molina specifies the leader election problem for synchronous and asynchronous distributed systems with crash and link failures and gives an elegant algorithm for each type of system; the algorithm for asynchronous systems is called the Invitation Algorithm [3]. The group communication system in Amoeba [5, 6] uses the Invitation Algorithm to reconfigure a group after a node crashes. In a recent textbook [1], Chow and Johnson write: "We will examine the classic election algorithms of Garcia-Molina. Several variations of election have been proposed, but the Garcia-Molina algorithm best defines and handles the possible failures."

Garcia-Molina's specification of leader election in asynchronous systems is based on the idea of *groups*: a group is a set of nodes that agree on a leader. To prohibit trivial algorithms, Garcia-Molina's specification requires, roughly, that if a set R of nodes can all communicate with each other during an election, then at the end of the election, the nodes in R are in a single group. However, Garcia-Molina's specification is unintentionally strong: contrary to his Theorem A4, the Invitation Algorithm, through no fault of its own, does not satisfy it. Furthermore, Garcia-Molina's specification is undesirably strong for some systems: it sometimes forces nodes that cannot directly communicate to be in the same group. These problems are not mentioned in [1]. This paper proposes a new specification, which is satisfied by the Invitation Algorithm and never forces nodes that cannot directly communicate to be in the same group.

The Invitation Algorithm works roughly as follows; for details, see [3] or [1]. Each node has a unique priority. Each node i maintains a variable $status_i$ containing its status, a variable grp_i identifying the group it is in, and a variable ldr_i identifying the leader of that group. A node's status is Normal except while the node is in the process of joining a new group. Periodically, each node that is not the leader of a group calls a Timeout procedure that checks whether the leader of its group is still alive, by sending a message to the leader and waiting for a reply. If the node

[*]Email: stoller@cs.indiana.edu Web: http://www.cs.indiana.edu/~stoller/ Address: Computer Science Department, Indiana University, Bloomington, IN 47405, USA

does not receive a reply within the timeout period, the node invokes a Recovery procedure. The Recovery procedure puts a node i into a singleton group with node i as the leader. Periodically, each leader i calls a Check procedure, which sends messages to every other node asking whether that node is a leader. If one or more other nodes replies that it is a leader, node i pauses for a time inversely proportional to its priority (this helps prevent multiple nodes from initiating elections concurrently) and then calls a Merge procedure. The Merge procedure sends messages to all of the other leaders, inviting them to join a new group with the inviting node as leader. When a leader i receives an invitation, it forwards the invitation to the other members of its group. A node i that receives an invitation (directly or indirectly), sends an accept message to the proposed leader of the group. If i receives a reply to its accept message within some timeout period, then i joins the new group; otherwise, i calls the recovery procedure described above.

Garcia-Molina gives two correctness requirements for leader election in asynchronous systems. The first requirement, called Assertion 3, says that if two nodes are in the same group, then they have the same leader; we express this requirement as follows:

> *Assertion 3.* At all times, for all operational nodes i and j, if $status_i =$ Normal and $status_j =$ Normal and $grp_i = grp_j$, then $ldr_i = ldr_j$.

Note that ALE1 is satisfied by algorithms that always leave each node in a singleton group. To express that a good algorithm leaves the system in a state with a reasonably small number of groups, Garcia-Molina proposes a second requirement, called Assertion 4. Two nodes are *connected* in a given time interval if all messages sent between them during that time interval are delivered within δ time units, where δ is a known constant.

> *Assertion 4.* Suppose there is a set R of nodes which are operational and pairwise connected for the duration of an election. Suppose also that there is no superset of R with this property. Then the election leaves the system in a state in which: (a) there is a node i in R with $status_i =$ Normal and $ldr_i = i$, and (b) for every other node j in R, $status_j =$ Normal and $ldr_j = i$ and $grp_j = grp_i$.

At first glance, Assertion 4 seems not to specify whether nodes that are not connected must end up in the same group. Surprisingly, this is not always true. To see why, consider the system shown in Figure 1. The number in each node indicates its priority; small numbers correspond to high priorities. The edge between nodes 1 and 2 indicates that those nodes are connected. Similarly, nodes 2 and 3 are connected, but nodes 1 and 3 are disconnected. Note that Garcia-Molina explicitly considers non-transitive connectivity [3, page 52]. The dotted and dashed lines indicate two sets $R_1 = \{1, 2\}$ and $R_2 = \{2, 3\}$ that meet the preconditions of Assertion 4. Thus, Assertion 4 requires that nodes 1 and 2 end up in the same group, and that nodes 2 and 3 end up in the same group, and hence that all three nodes end up in the same group, even though nodes 1 and 3 are not connected.

The following scenario shows that the Invitation Algorithm does not satisfy Assertion 4, despite Theorem A4 in [3]. The system contains 3 nodes. Initially, all nodes are operational, and all pairs

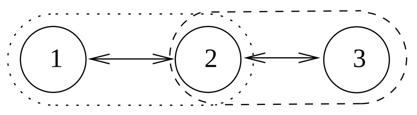

Figure 1: Scenario in which Assertion 4 requires disconnected nodes to be in the same group.

of nodes can communicate; also, all nodes are in the same group, and node 1 is the leader. The following events occur:

1. Node 1 crashes.
2. Nodes 2 and 3 each call Timeout and then Recovery; each forms a singleton group.
3. Node 1 recovers, but communication between nodes 1 and 3 has been lost. In other words, Assumptions 8 and 9 hold for the nodes in the set $\{1,2\}$ and for the nodes in the set $\{2,3\}$, but not for the nodes in the set $\{1,2,3\}$.
4. Node 1 calls Recovery and then Check. Since nodes 1 and 2 can communicate, node 1 calls Merge.

The outcome is that nodes 1 and 2 are in one group, and node 3 is in a singleton group (node 3 sends an accept message to node 1 but does not receive a reply, so it calls Recovery). If no more failures occur, these groups will not change. The set $\{2,3\}$ satisfies the hypotheses about R in Assertion 4, so we have the requirements: (a) some node i in $\{2,3\}$ is a coordinator, and (b) every other node j in $\{2,3\}$ has node i as its coordinator. Requirement (b) is not satisfied, since node 2 has node 1 as its coordinator.

We propose a weaker requirement that never forces disconnected nodes to be in the same group. Two nodes are *disconnected* in a time interval if all messages sent between them during that time interval are lost. A system is *stable* in a time interval if, during that interval, no crashes or recoveries occur and every pair of nodes is either connected or disconnected.[1] When a system is stable, its *connectivity graph* is the undirected graph whose vertices correspond to the nodes of the computer system and with an edge between vertices i and j iff nodes i and j are connected. A *clique cover* of a graph is a partition of that graph's nodes into cliques (*i.e.*, fully connected components). The predicate up_i is true in a state iff node i is operational. For a relation E, let E^* denote the reflexive and transitive closure of E. The requirement is:

> *Assertion 4'.* For a given system, there exists 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, letting $\langle V, E \rangle$ denote the system's connectivity graph, the system reaches a state such that: (a)

[1] This definition of stability is based on [2]. The definition of stability in [2] is stronger because it requires transitive connectivity.

($\forall i : status_i =$ Normal $\wedge\ up_{ldr_i} \wedge\ grp_{ldr_i} = grp_i \wedge \langle i, ldr_i\rangle \in E^*$), and (b) the number of groups is at most the size of a minimum-sized clique cover of $\langle V, E\rangle$. Furthermore, the system remains in that state as long as the system remains stable.

Assertion 4' *allows* the number of groups to equal the size of a minimum clique cover; this ensures that disconnected nodes are never forced to be in the same group. Assertion 4' does not *force* the number of groups to equal the size of a minimum-sized clique cover; this is important, because we do not want leader election to be NP-hard (recall that computing a minimum-sized clique cover is NP-complete [4]). When the connectivity graph is transitive, Assertion 4' is equivalent to Assertion 4.

We sketch a proof that the Invitation Algorithm satisfies part (b) of Assertion 4'; the proof that it satisfies part (a) is straightforward. An *independent set* in a graph is a set S of vertices such that no two vertices in S are connected by an edge. Under the hypotheses of Assertion 4', when the Invitation Algorithm quiesces, the number of groups is at most the size of a maximum-sized independent set in $\langle V, E\rangle$, because the leaders must form an independent set, because if any two leaders were connected, one of them would call Merge. Let S be an independent set in $\langle V, E\rangle$. Note that, in every clique cover for $\langle V, E\rangle$, each element of S must be in a different clique. Thus, the size of a minimum-sized clique cover for $\langle V, E\rangle$ is greater than or equal to the size of a maximum-sized independent set in $\langle V, E\rangle$.

References

[1] Randy Chow and Theodore Johnson. *Distributed Operating Systems and Algorithms.* Addison Wesley, 1997.

[2] Flaviu Cristian and Frank Schmuck. Agreeing on processor group membership in asynchronous distributed systems. Technical Report CSE95-428, University of California, San Diego, 1995.

[3] Hector Garcia-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers*, C-31(1):47–59, January 1982.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[5] M. F. Kaashoek and A. S. Tanenbaum. Group communication in the Amoeba distributed operating system. In *Proc. IEEE 11th International Conference on Distributed Computing Systems (ICDCS)*, pages 222–230. IEEE Computer Society Press, 1991.

[6] M. Frans Kaashoek and Andrew S. Tanenbaum. Efficient reliable group communication for distributed systems. Rapport IR-295 IR-295, Faculteit Wiskunde en Informatica, Vrije Universiteit, 1992. Revised version available from ftp://ftp.cs.vu.nl/pub/papers/amoeba/group94.ps.Z.