

# Two-Tier Leadership Election

Ian Adams

Graduate Student

Department of Computer Science  
University of California, Santa Cruz

iadams@soe.ucsc.edu

## Abstract

Current leadership election algorithms do not explicitly plan for the failure of the leader, and thus must do system wide elections each time a leader fails. To this end we propose an algorithm that removes the need for system wide elections after a leadership failure. Our technique has an initial system wide election to elect a leader, and after the leader is confirmed chooses *subordinates* with which to form a clique. After the failure of the leader a subordinate assumes the leader's position, and a new subordinate is selected to take the vacated place. We demonstrate the feasibility of this new scheme with a simple message passing simulation and found that our scheme cuts in half the number of messages required to select and announce a new leader after a leadership failure.

## 1. Introduction

Leadership election in distributed systems is essentially a variation of termination detection in diffusing computations (3) wherein a leader, usually referred to as a process in the literature, is selected such that all devices/processes agree that it is the leader. This has a wide variety of potential applications. One such application is the regeneration of a lost token in a network (11), a token here essentially being a piece of data that provides unique privileges to the holder. If the token is lost an election can be used generate and guarantee there is only as single replacement token. Another example comes from ad hoc routing within dynamic wireless networks wherein wireless devices are elected as 'landmarks' for the coordination of networks (7). Yet another application (and the motivation for this project) is electing a device to control the physical coordination of activities in a distributed system (note we are considering a distributed system to be composed of autonomous independent units, much like (4)). It is in this latter category of applications that we question existing leadership election schemes.

Most existing leadership election algorithms treat all nodes in the system equally when selecting a leader. In essence, the attitude in these schemes is that it doesn't matter what device is selected, so long as each node in the system agrees on a leader, though there are schemes now to choose

the 'best' node rather than simply the one with the highest ID. There is a large body of work, some of which we will discuss in the next section, on algorithms to accomplish these goals. Most are variations on the 'Bully Algorithm' (5) wherein the device with the highest value, often an identifier or some other arbitrary static value, 'wins' the election.

Work has been done on reducing the overhead in messages and run time (2; 1), but these algorithms still run system wide each time a leader needs to be selected. This is acceptable if leadership election is infrequent and the system is small, but if the system is very large and has to deal with more frequent leadership failures, it is easy to see the possibility of decreases in both network bandwidth and system wide output due to the overhead of frequent elections. This slowdown can further be compounded if a leader in the system must gather data from constituent nodes before operation can resume in an effective fashion, such as would be the case in the proposed 'Logan' system for managing large numbers of archival storage nodes. (15).

The initial election needs to encompass the entire system to ensure only one leader is elected. However, after the leader is established, it is possible to take an idea from the fault tolerant systems area and nominate subordinates to take over in the event of a failure. If all nodes are to be treated equally, there is no reason why a nominated node should be any different from an elected one. Provided the subordinates promptly notice the failure of the leader, it is a relatively simple matter to select a new leader and subordinate with relatively few messages. We also describe, though do not test, an algorithm to deal with situations where nodes are not all equally suitable for leadership in the following section.

There are still situations where full election would be necessary, such as system startup, correlated failures, network partition and so forth, but with subordinates the election is reduced to a much smaller portion of the system. This will reduce the number of messages needed to be passed to elect a leader, though the ID of the new leader will still need to be propagated to the system. Additionally we can now leverage the subordinates to store duplicate information from the leader (e.g. system stats, job lists, etc.) so that they may

smoothly take over operation rather than having to halt to gather the necessary information from the system nodes.

The rest of the paper is structured as follows: Section 2 describes the 2 algorithms used, section 3 is experimental design, section 4 is results, section 5 is related work, and section 6 is conclusions and future work.

## 2. Election Algorithm

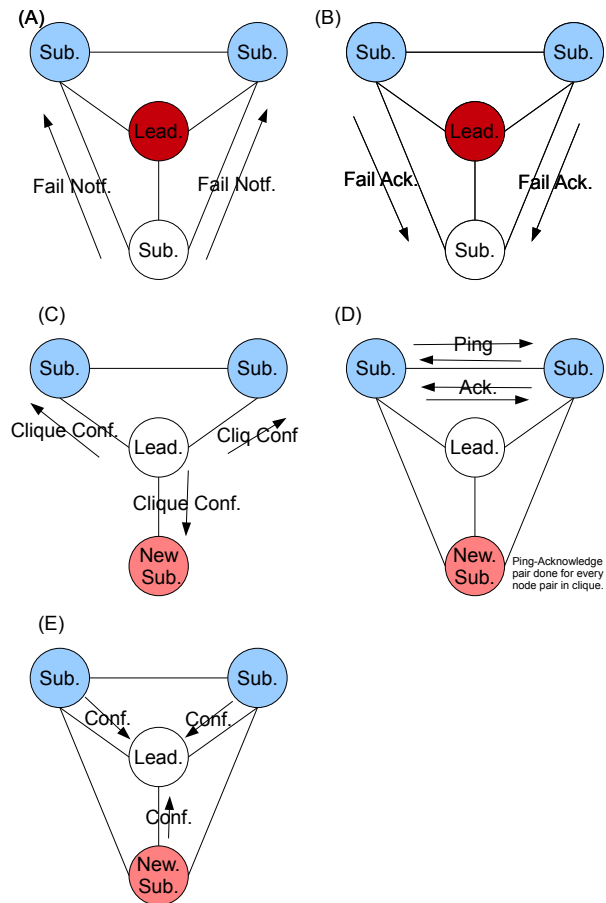
The core of our algorithm, and our basis for comparison, is a simple version of the Asynchronous Extrema Finding Algorithm (AEFA) found in (16). We chose it as it is simple to implement, robust (it can work in the presence of failures), can function in an asynchronous environment, and is designed to select the 'best' node in a system that may have nodes of non-uniform suitability for leadership. The last property will be particularly useful in future extensions of the 2-Tier algorithm.

### 2.1 AEFA

The election begins when a node recognizes that there is no leader, or the leader has recently failed. This recognition may be accomplished in a variety of ways, such as through an explicit failure broadcast from the leader, term expiration, timeouts from a heartbeat, etc.

Once a node notices leadership failure or absence it changes its state to ELECTION and broadcasts to all of its neighbors an ELECTION message and marks these neighbors as children. Upon receipt of an election message these child nodes change their state to ELECTION as well and set their parent to the node they received the ELECTION message from. They then also broadcast ELECTION messages to all of their neighbors besides their parent and correspondingly set these as children. If a node receives an ELECTION message and is already in the ELECTION state, it immediately sends back an ACKNOWLEDGE message with its own ID or the highest ID it has received an ACKNOWLEDGE message from. Once an node has received ACKNOWLEDGE messages from all of its children, it sends an ACKNOWLEDGE message to its parent with the highest ID (possibly its own) that it received. This process continues until the source node receives ACKNOWLEDGE messages from all of its children, and thus has the highest ID in the system. This entire process can be viewed as growing a spanning tree from the source node with ELECTION messages, and then shrinking it back down with ACKNOWLEDGE messages back to the source (16). See figure 2 for a small example.

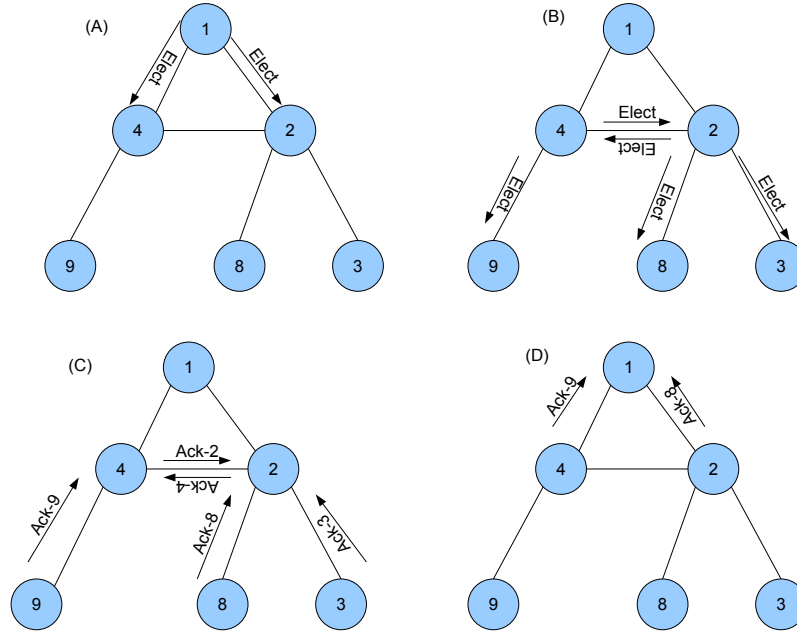
After the source node receives the highest ID, it contacts the highest ID node, notifies it that it is the leader, and it then broadcasts its notification of leadership to all nodes in a fashion essentially identical to the leadership algorithm itself, so we omit its description here. After the leadership announcement, all extant nodes in the system will agree on the leader.



**Figure 1.** An example of replacing a failed leader in a leadership clique. In A the bottom subordinate node notices the leader has failed and announces this to the surviving clique members. In B They confirm this with responses agreeing that the leader has failed. In C the bottom subordinate node that announced the failure promotes itself to leader, selects a new subordinate, and sends messages asking to verify the clique connections. In D a series of pings is used to verify connectivity. In E, after all connectivity is verified, the leader receives confirmation from its subordinates, and will then announce its leadership to the system.

### 2.2 2-Tiered Election

Our election method in an uninitialized system starts off identically with the same AEFA algorithm described above to select a single leader and propagate its identity through out the system. After the leader is confirmed, it chooses 3 arbitrary *subordinate* nodes and forms graph clique with them, verifying the clique with a series of simple pings between the members. We are currently assuming that all nodes are equally valid as choices, but with the AEFA scheme and the ability to leverage the fact that we now have a confirmed leader, we could easily select the 'best' leader and gather the necessary information to select the 'best' nodes for subordi-



**Figure 2.** A small example of the AEFA algorithm in action. Note that essentially it is growing out a spanning tree, and nodes return acknowledgments with the best value seen so far to their parent only when they have received acknowledgments from all children or have already received an election message.

nates.

After the initial election, if a leader fails or exits the system, the first subordinate node to notice the failure announces this to the other subordinates. The other subordinates confirm the leadership failure and return acknowledgment messages to the node that made the announcement. This node then considers itself the leader, and selects a new subordinate to replace it. After the selection of a new subordinate, the clique is verified to be complete with a series of pings, and the new leader announces to the system its assumption of leadership. See figure 1 for an example.

Because this scheme does not require a system wide election each time a leader fails, it will significantly reduce the number of messages passed in the course of selecting a new leader due to the small size of the subordinate-leader clique. Additionally the overhead of creating the clique is quite low and easy to understand due to its small size and simple nature. This clique may additionally be leverage existing work in fault tolerance techniques for distributed systems.

As a motivating example, in the Logan system (15) the leader uses statistics gathered from nodes under its control to dynamically create and maintain heterogeneous raid groups. In the event of a leader failure, it would be expensive in time and messages to regather all the necessary stats to resume operations, but with a subordinate with replicated state it could smoothly transition to a new leader.

### 2.2.1 2-Tier Algorithm-Selecting the Best Leader

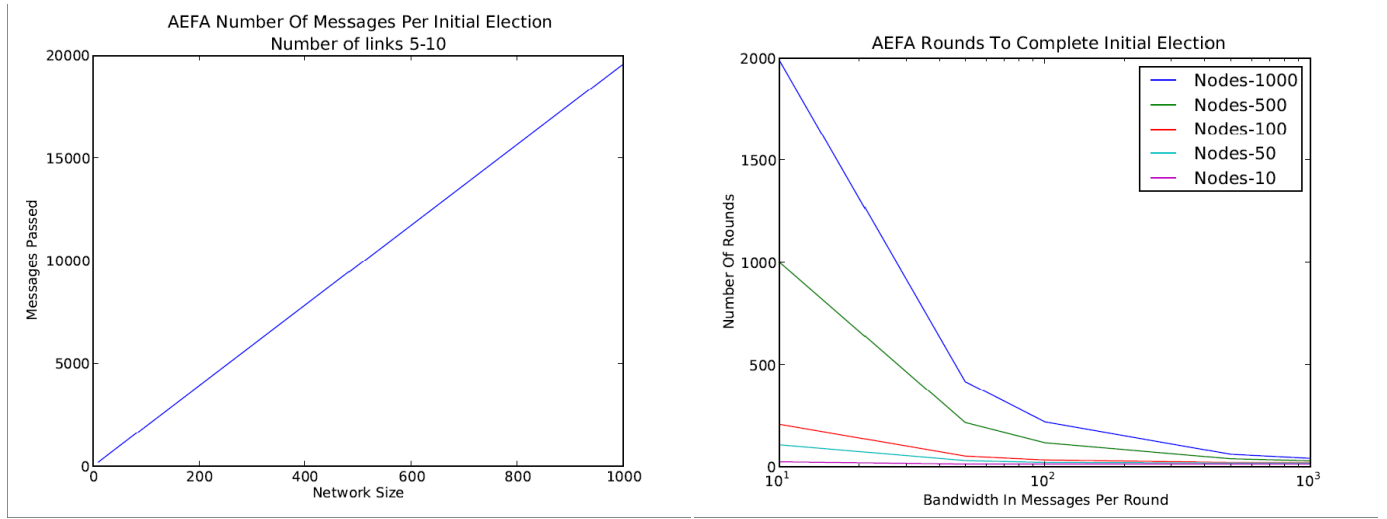
The aforementioned algorithms both work off of selecting the node with the highest ID in the initial case, though it is a simple matter to switch it to using some normalized evaluation value to select the 'best' node for leader as they do in the original AEFA algorithm.

For our 2-Tier algorithm there are a few ways to accomplish this goal. One would be to store statistics gathered from the system and replicated across the leadership clique, so that when the leader fails, we already have a centralized repository from which to select the best replacement. If we require the subordinates to also be the best possible nodes it becomes more complicated, where we must replace subordinate nodes after each failure or as better nodes enter the system.

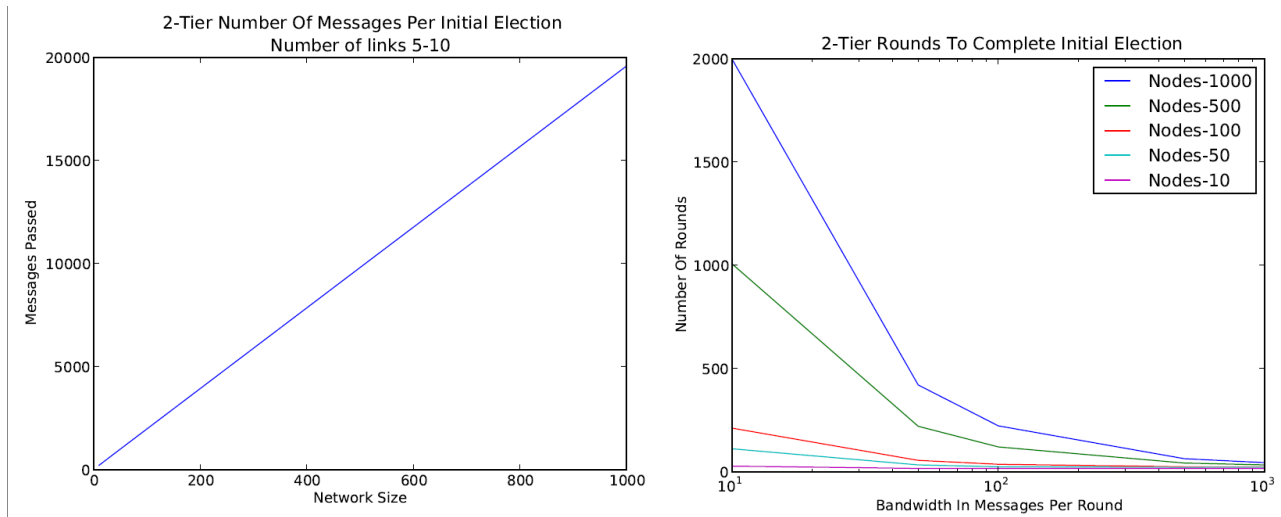
If we relax the requirement that the subordinates and leader always be the best possible we can simplify it a bit to where failover occurs in the same fashion as the normal 2-Tier election, and we select the best node we can find to replace the promoted subordinate. This could be done in a variety of ways such as through gathered and replicated statistics, or even a random sampling of nodes in the system.

## 3. Experimental Design

Our experiments were run with a simple message passing network model that runs in rounds. A round consists of 2 distinct phases. At the beginning of each phase, messages are



**Figure 3.** Left figure is Average number of messages for election with the AEFA scheme . Right figure is the number of rounds to complete an election over varying bandwidth for AEFA. Each node always has 5-10 links regardless of system size.



**Figure 4.** Left figure is Average number of messages for election in the 2-Tier scheme. Right figure is showing the number of rounds to complete an election over varying bandwidth for the 2 Tier scheme. Each node always has 5-10 links regardless of system size. Note that the startup cost with the 2-Tier scheme is essentially identical given the low number of extra messages required to create the leadership clique.

taken from the output queues of sending nodes and delivered to the input queues of the destination nodes. The number of messages that may be delivered in a round is determined by a global bandwidth setting. If the number of messages delivered in a round reaches this bandwidth, no more messages may be delivered until the next round, though no messages are actually lost. Afterwards each node processes its input queue and does the appropriate state change (if any) and

places any necessary messages onto its output queue.

Each node in the system has a single input and output queue of infinite length for the receiving and sending of messages. A node may process its entire input queue each round, and there is no limit on the number of messages it may place on its output queue.

For each experiment a network is created consisting of a single connected component with each node connected to

approximately 5 to 10 other nodes. These numbers are somewhat arbitrary and were chosen as due for ease of implementation and to maintain consistency across tests. In the future we would like to have more stringent controls on the topology of the graph network, but because of the short term nature of this project we stuck with our simplistic implementation.

### 3.1 Assumptions

Due to the limited time for this implementation our assumptions are fairly numerous and strong.

- No nodes or links may fail during or between elections.
- Each node may process an infinite number of messages during each round.
- All communications are point-to-point. Broadcasts simply consist of sending the message individually to each node on the neighbor list
- There are no simultaneous elections. This can be compensated for, and has been in the full version of the AEFA algorithm we are using, but in the interest of time we forgo dealing with this and acknowledge that it may affect our 2-Tier scheme in unpredictable ways.
- There are no partitions in our network during or between elections.

### 3.2 Experiments

Our first set of experiments track the number of messages passed in the 'start-up' elections for a new system, as well as the number of rounds they take to complete for given bandwidths.

The second set of experiments tracks the same statistics as above, but now tracking results that come from running our election schemes after the failure of a leader. For the pure AEFA election we fail a leader, and pick an arbitrary node to be the first to notice the leader loss and to be the source of a new election. For the 2-Tier scheme we pick an arbitrary subordinate node to be the first to notice the leader loss and announce its intention to be the new leader.

For each set of results, we ran the elections and failover elections with system sizes of 10, 50, 100, 500, 1000 nodes and varied the bandwidth to 10, 50, 100, 500, 1000 messages per round maximum for each network size. Each trial was repeated 20 times with a randomly generated system of the appropriate size newly generated each time.

## 4. Results

As we can see from figures 3 and 4 the number of messages and rounds scales almost perfectly linearly in both election algorithms for the initial election in a new system. This is due to the fact that the number of messages passed in an election in both schemes is dependent upon the number of links. Remember that we have a strictly point to point message passing model, and each node broadcasts to all of

its neighbors upon receipt of a new election or leadership announcement message. Each broadcast may have up to 10 (or slightly more) messages, and correspondingly there will be the same number of responses as acknowledgments. For example, If a network with  $N$  were fully connected with our scheme we would see approximately  $N^2$  messages passed. We do not intend to trivialize the importance of the network topology and number of links, but we did not have enough time in light of the short time span of the project to examine the effects of topology changes.

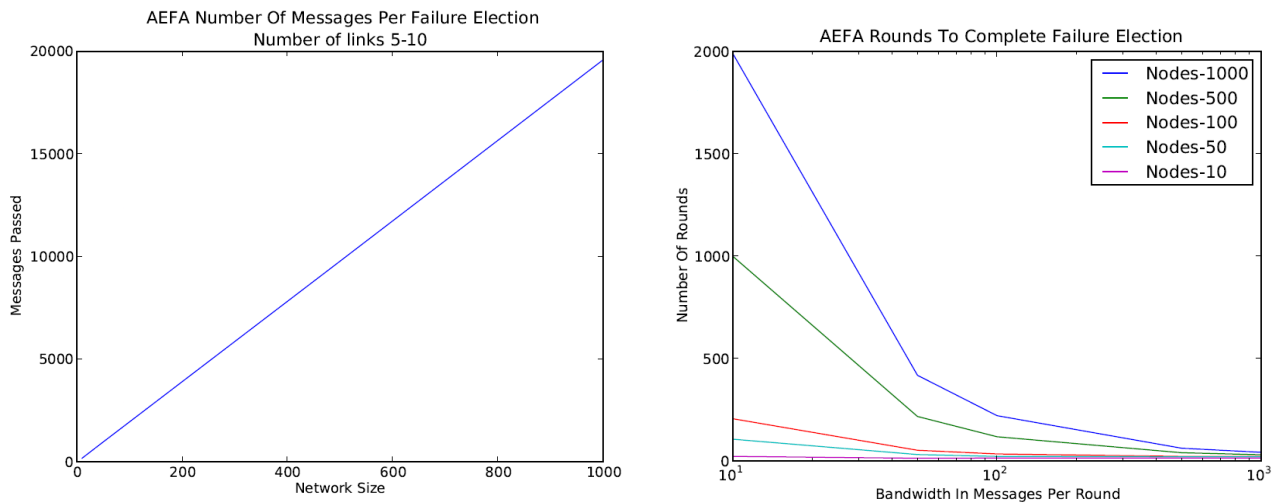
Regardless, with similar topology between tests we can see that the startup cost for both elections is essentially the same. This is because our 2-tier algorithm only adds a small number of message at the end of an AEFA election for the creation of the leadership clique.

It is when we fail a leader and select a new one that the advantages of the 2-Tier scheme become clear. As we can see from figures 5 and 6 the number of messages and rounds required to elect and announce a new leader are approximately half of that when running AEFA. The reason for this is because under the 2-Tier election, we can leverage the leadership clique and simply select a new leader from within that small subset of the system, and then announce the new leader. This makes the primary overhead burden announcing and confirming the new leader. When just using AEFA we must hold a system wide election, and then announce, both being fairly large diffusing computations.

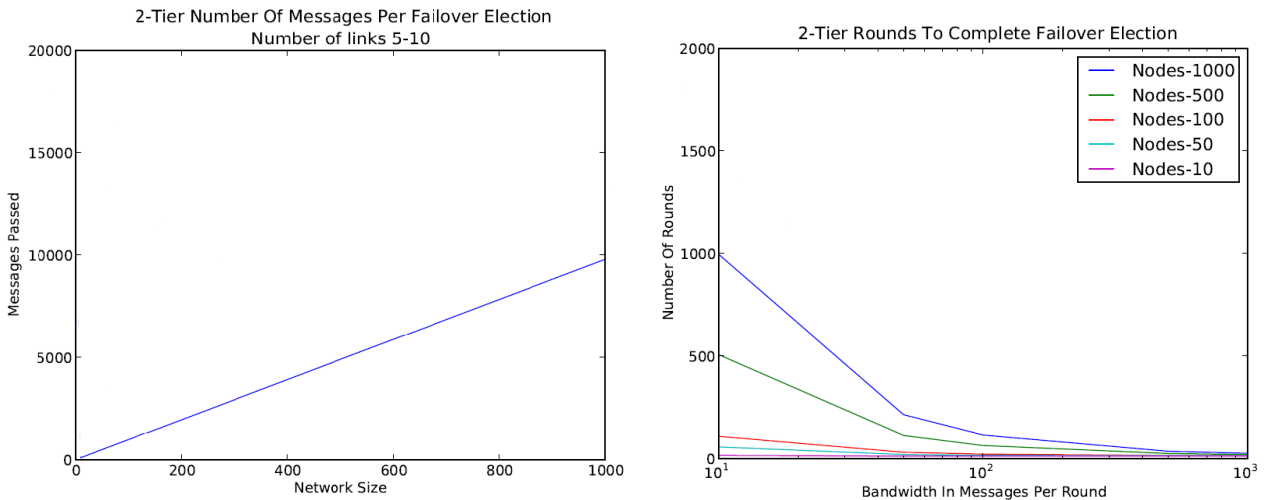
A fairly interesting and somewhat counterintuitive result we found was when examining the average number of messages processed by a given node per message passing round. We noticed that across all tests (see figure 7) as the system size increases, the average number number of messages processed actually decreases. This is, like mentioned above, due to the fact that we are not varying the number of links in our system on a per node basis. Because of this, in larger systems individual nodes will wait longer amounts of time to receive acknowledgments from their children when announcing a new leader or selecting one through AEFA. Though this result is mostly an artifact of our experimental design and implementation it demands further examination (there is very likely work already done on this topic, were simply out of time) of the effects of topology. A careful arrangement of links and nodes could reduce the necessary workload without sacrificing connectivity and robustness, since often times a node that has already received an election message has already received election messages from others and thus these are somewhat redundant.

## 5. Related Work

None of the election algorithms we found take a two stage approach as we propose, and instead rely on system wide elections after each leadership failure, which we believe to be needlessly wasteful. Additionally utilizing ideas from fault tolerant systems (state replication) can allow for quick



**Figure 5.** Left figure is the number of messages required to elect and announce new leader only using AEFA. Right figure is the number of rounds it takes for an election (selection, and announcement) to complete over varying bandwidths and system sizes. It is effectively identical to the initial election due to the fact that it must run a full system wide election again.

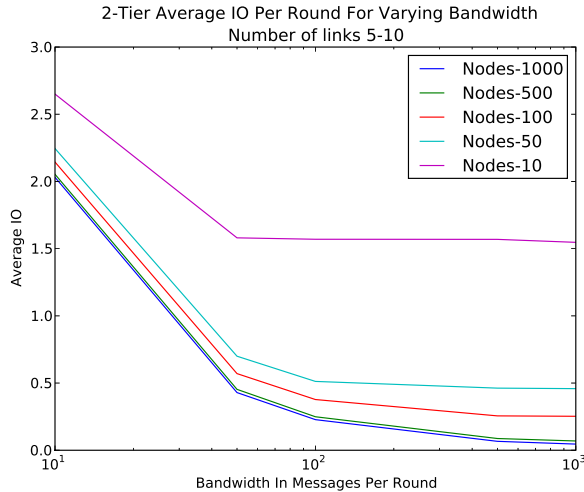


**Figure 6.** Left figure is the number of messages to elect and announce a new leader after a failure has occurred. The right is the number of rounds it takes for the election and announcement to complete under varying bandwidths. Compared to AEFA it takes approximately half the messages and time due to the fact that the 'election', such as it is, only takes a fixed number of rounds regardless of system size because it is only taking place within the leadership clique, and the rest of the time and messages is from announcing and confirming the new leader.

resumption of activity much as it does in traditional systems. Utilizing group communications techniques could also further reduce the overhead as that field of research is creating efficient methods of communicating with large sets of independent nodes/processes.

### 5.1 Election Algorithms

There is a significant body of work already extant on leadership election and algorithms. Most of these are variations on one of two algorithms: the invitation algorithm, and the bully algorithm (5). In the bully algorithm, each node (or process in the original paper) has a numeric ID. A node con-



**Figure 7.** This shows the average IO of nodes during initial elections under the 2-Tier scheme. The AEFA and associated failure results are omitted as they are essentially identical.

tacts all nodes it knows of with higher ID, which may be none at all. If none respond after a set amount of time then it declares itself leader. The invitation algorithm is slightly different. Each node forms its own 'group' initially. It then periodically contacts the other nodes it knows of, if it has a higher ID it invites the other node(s) to join its group. This continues until there is only a single group and leader.

Cidon and Mokryn's (2) work manages to reduce the number of necessary messages by a fairly significant margin, but depends on the system being in a broadcast environment such as a wireless network to enable multiple nodes to receive a message sent via a single broadcast. This property cannot be depended on in general so is of limited utility in limiting overhead for an arbitrary system.

Work on elections within wireless sensor and ad hoc networks has also been significant (12; 16; 7). These algorithms must deal with the additional complexity of potentially mobile nodes, link failures, node failures, and network partitions and merges all at a much higher frequency than seen in a traditional distributed environment. In these situations efficiency is key as frequent leadership elections are the norm. However, our proposed scheme would likely not be appropriate as the high rate of network partitions would obviate the usefulness of having subordinates as quick leader replacements due to the likelihood of them being cut off.

There has been less work done on choosing the 'Best' leader for a network rather than arbitrarily picking based on ID. (16)'s work addresses this issue for ad hoc networks (the AEFA algorithm we use is from this work). They proposed algorithms that took into account the 'goodness' of a node. One resembles the bully algorithm, but instead of a simple identifier, a node advertises its value based upon some common metric, and the node with the highest value wins. A

different approach they propose allows a node to 'vote' for the node it recognizes as best, and the node with the highest number of votes wins. This is a more complicated approach but is interesting nonetheless as it is a marked departure from the normal bully approach.

## 5.2 Fault Tolerant Systems

Replication of state across multiple machines is nothing new see (14; 9) among others. The basic idea is to replicate relevant state across multiple machines to attain fault tolerance in the event of the faults or failure of one or potentially more machines.

In the Quicksilver (6) distributed system, transactions are managed by a coordinator and each coordinator has a number of subordinates which it replicates to. This allows for byzantine fault tolerance and smooth transitioning in the event of a coordinator failure. To our knowledge they do not acknowledge the cost of choosing the coordinator and subordinates and it does not appear to be as general or as flexible as ours.

In the Harp file system (10), the primary server is replicated across multiple backup servers who step in the event of the primary failure. This is roughly analogous to having a our leader-subordinate hierarchy and actions, though it does not seem to be a generalized scheme for leadership in a system, but simply a fault tolerance technique.

Though we are not interested in conventional fault tolerance per se, we can leverage their idea of data replication so that our leadership scheme can quickly and efficiently recover from a failed leader via a subordinate promptly stepping in and announcing itself as the leader much like in (10). Furthermore, if the leader requires specialized information about system state, we could replicate this at the subordinate nodes, further smoothing out the leader failover process.

## 5.3 Group Communications

Group communication protocols like (8; 13) provide methods for communication with subsets of an entire distributed system. Use and or integration of group communications techniques could further reduce the cost of running an election and announcing a leader.

## 6. Conclusions and Future Work

We have demonstrated the feasibility of a simple modification to the AEFA election method that allows for a 50 percent savings in message overhead in the event of a leader failure by forming a clique around the initially elected leader and holding failover elections within the clique only.

There is much work to be done on the clique elections on several fronts. We need to modify it so that it can be guaranteed to function in the face of node and link failures, network partitions, simultaneous elections, and cheating resistance. Additionally further testing to demonstrate its use in systems that require specialized leadership information is needed,

and it is here that we believe the 2-Tier scheme will have a significant advantage as the leadership clique may be leveraged to implement fault tolerance techniques using replicated state.

**Acknowledgments:** I would like to personally thank Mark Storer for his suggestion of this as a project, and give credit to him for his initial work on the 2-Tier election. Additionally I would like to thank fellow classmates and members of the SSRC for their feedback and support.

## References

- [1] Awerbuch, B. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, 1987.
- [2] Cidon, I., Mokryn, O. Propagation and leader election in a multihop broadcast environment. *Lecture Notes In Computer Science*, volume 1499, 1998.
- [3] Dijkstra E., Scholten, C. Termination detection for diffusing computations. *Information Processing Letters* 11, 1 (Aug. 1980), 14.
- [4] Enslow, Phillip H. What is a 'Distributed' Data Processing System ?. *Computer*, Volume 11, Issue 1. Pages 13-21. January 1978.
- [5] Garcia-Molina, H. Elections in a distributed computing systems. *IEEE Transactions on Computers* 31, 1 (Jan. 1982), 4859.
- [6] Haskin, R., Malachi, Y., and Chan, G. 1988. Recovery management in QuickSilver. *ACM Trans. Comput. Syst.* 6, 1 (Feb. 1988), 82-108.
- [7] Hong X., Gerla, M. Dynamic Group Discovery and Routing in Ad Hoc Networks. *Annual Mediterranean Ad Hoc Networking Workshop*, 2002.
- [8] Kaashoek, Frans. Group Communications In Amoeba And Its Applications *Proceedings of 11th Int'l Conf. on Distr. Comp. Systems* , 222-230
- [9] Lamport, L. Paxos Made Simple. *ACM SIGACT News*, November 2001.
- [10] Liskov, B., Ghemawat, S., Gruber, R., Johnson, P., and Shriram, L. 1991. Replication in the harp file system. In *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles* (Pacific Grove, California, United States, October 13 - 16, 1991). SOSP '91. ACM, New York, NY, 226-238
- [11] Lynch, N. Distributed Algorithms. Pages 25-77. 1996.
- [12] Navneet M., Welch J., Vaidya, N. Leader Election for Mobile Ad Hoc Networks. *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*. 2000.
- [13] Renesse, R., Birman K., Maffei, S. Horus: A Flexible Group Communication System. *Communications of the ACM*, April 1996, Vol 39, No.4
- [14] Schneider, F. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Computing Surveys*, Vol. 22, No. 4. December 1990.
- [15] Mark W. Storer, Kevin Greenan, Ian Adams, Ethan L. Miller, Darrell D. E. Long, Kaladhar Voruganti, "Logan: Automatic Management for Evolvable, Large-Scale, Archival Storage," *Proceedings of the 2008 Petascale Data Storage Workshop (PDSW 08)*, November 2008.
- [16] Vasudevan, S., DeCleene, B., Immerman N., Kurose, J., Towsley, D. Leader Election algorithms for Wireless Ad Hoc Networks. *Proceedings of the DARPA Information Survivability Conference and Exposition*. 2003.