ELECTING LEADERS BASED UPON PERFORMANCE: THE DELAY MODEL

Suressh Singh* and James Kurose†

Abstract

In a distributed system, an algorithm used to select a distinguished node or leader in the system is known as a leader election algorithm. In this paper, we examine leader election algorithms that attempt to locate the leader at a "good" node (from a performance standpoint) in the system. In the "preference-based" approaches examined here, each node in the system uses locally available information to "vote" for the various candidates (potential leaders) on the basis of the performance level it would realize under each of them. The preference-based leader election algorithms we propose and examine are simple and are shown to perform almost as well as a traditional optimization-based approach towards leader election.

1 Introduction

In a number of distributed computer system applications, a distinguished node or leader is used to coordinate some activity in the system. For example, the leader might coordinate reorganization of the system following a failure (e.g., when a leader fails or a distributed system is partitioned), help manage k-resiliency in a reliable distributed system [3], or manage a system resource that requires mutually exclusive access [9, 1]. The algorithms that choose such a leader are called leader election algorithms.

Existing leader election algorithms may be characterized as extrema-finding algorithms in which all nodes are assumed to have a unique ID number, and the leader which is elected is simply that node which has the largest ID number, see for example, [7, 5, 2, 6]. While these efforts are of considerable theoretical interest, they suffer from practical considerations. When a leader is elected on the basis of an ID number, it is chosen without regard to the level of performance that each node would receive from the leader. For example, a node which already has a high workload demand, or which is connected to other nodes via unreliable links, or which is in a congested area of the underlying communication network would make a poor choice for a leader, since, as leader, it would provide a poor level of performance (in the above examples, high delay or unreliable communication) to the nodes in the system.

The research described in this paper is predicated on the belief that when a leader is to be elected, that leader should ideally represent a "good" choice for the system, from a performance standpoint, as opposed to a choice determined by some arbitrary criterion such as node number.

In our preference-based approach, each node in the system uses locally available information to "vote" (as discussed later) for the various candidates for leader on the basis of the performance level it would realize under each possible leader (e.g., the expected communication delay or path reliability from the voting node to the candidate node). The votes of the individual nodes are then combined (as discussed later) to determine the elected leader. There are two advantages to such a preference-based approach. First, preference-based algorithms by definition incorporate performance considerations into the election process. Second, there is already a large

*with SUNY New Paltz
†with Univ. of Massachusetts, Amherst
body of existing literature on leader election (voting) in the area of Social Choice theory [10] upon which we may draw.

In this paper we examine a number of socially-inspired, preference-based leader election algorithms which are designed to overcome the previously-mentioned deficiencies in existing leader election algorithms. A major result of this paper is that five of our proposed leader-election algorithms perform significantly better than the other algorithms studied.

The remainder of this paper is organized as follows. Section 2 presents a system model (the Delay Model) in which we study the problem of preference-based leader election. In section 3, we introduce our preference-based approach to leader election and contrast it with an optimization-based approach. Section 4 presents different election schemes to be studied in this paper. Two different methodologies which will be used to study the relative merits of the various election schemes are presented in section 5. Section 6 discusses our experimental and theoretical results. In section 7, we briefly discuss algorithms to implement the election schemes presented in this paper. Finally, section 8 summarizes the paper and provides a brief discussion of some open problems.

2 The Delay Model

Despite the differences in their function, most distributed computer systems can be represented as connected networks of computers. In such systems, the leader is necessarily a node that is connected to the rest of the system via the underlying communications network. All interactions between the leader and the system take place via messages exchanged over this network.

One natural performance measure in such a system is the communication delay between nodes. Based on this performance measure, a node that is more "central" would be a better choice for leader than a node that is on the periphery of the network. In Figure 1, node a is intuitively a better choice than node b to be the leader, because it is more centrally located w.r.t. the other nodes in the system. This notion of central location is the basis for the delay model.

2.1 System Model

The distributed system is represented as a graph $G(V, E)$, where $V$ is the set of nodes and $E$ is the set of edges. Furthermore, it is assumed that every edge is bidirectional and is weighted. Thus if $(i, j)$ is an edge, then so is $(j, i)$ and, the delay on $(i, j)$ is independent from the delay on the edge $(j, i)$. This assumption is quite realistic because, the delay along a link is a function of both the transmission delay and the queuing delay at the transmitting node. While the transmission delay might be the same along a physical link, the queues might be of different lengths at the two nodes i and j. Figure 2 shows an example of such a system. We can now state the property that distinguishes one node in the system as the best leader.

2.2 Optimal Leader

In a system that consists of $n$ nodes, there are $n$ possible leaders. However, not all of these nodes make good leaders. Let us define the system-wide performance of a node $j$ to be the sum of the shortest paths $d(i, j)$ from every node $i$ in the system to node $j$. Then, the optimal leader is defined as the node with the smallest value for the system-wide performance. Formally,
Definition 1: (Performance of a node) The system-wide performance of a node \( j \) is given by 
\[ d(j) = \sum_{i=1}^{n} d(i, j), \]
where \( d(i, j) \) is the shortest path length from node \( i \) to node \( j \).

Definition 2: (Optimal Leader) The optimal leader is a node \( k \) such that, 
\[ d(k) < d(j), \quad 1 \leq j \leq n, j \neq k. \]

Let us examine the five node system shown in Figure 2. We can represent the shortest paths \( d(i, j) \) by means of a matrix \( W \) as shown below. This matrix is called a weight matrix and it is easy to check that the column sum of the \( j \)th column yields the value for \( d(j) \). In this example, node \( d \) possesses the smallest column sum and is therefore the optimal leader.

\[
W = \begin{bmatrix}
  d(i, j) & a & b & c & d & e \\
  a & 0 & 2 & 4 & 5 & 8 \\
  b & 4 & 0 & 2 & 3 & 6 \\
  c & 3 & 2 & 0 & 1 & 4 \\
  d & 2 & 4 & 5 & 0 & 3 \\
  e & 3 & 5 & 6 & 1 & 0
\end{bmatrix}
\]

\[ d(j) \] 12 13 17 10 21

We finally assume that, every node can only estimate its own distance to the other nodes in the system. In other words, node \( i \) can only estimate the values for \( d(i, j) \), \( \forall j \). This set of values corresponds to the \( i \)th row of the weight matrix \( W \).

3 Preference-Based Approach

Given a system model as described above, we are now interested in designing an algorithm that would elect the optimal leader. An obvious algorithm is for every node to send its row of the weight matrix \( W \) to all the other nodes in the system. At the end of this series of exchanges, every node can independently compute the column sums of the weight matrix and determine the identity of the optimal leader.

Unfortunately, this approach is susceptible to failures in which a node misreports some of its weights of the weight matrix\(^1\) causing an arbitrary leader to be elected. Since failures of this kind cannot be detected by any other node, more sophisticated algorithms are required to handle them. In the preference-based approach discussed here, nodes only exchange votes, representing their relative preference for the different candidates, that are combined to determine the leader. Furthermore, the vote of every node is counted exactly once, thus reducing the impact of misreported votes upon the final outcome.

3.1 Preference Orders and Votes

Let us illustrate the idea of preference-based leader elections using the example shown in Figure 2. However, before doing so, we need to define a preference order.

Definition 3: (Preference Order) Let \( i, j, k \) be three nodes. We say that node \( i \) prefers \( j \) over \( k \), written \( i : j > k \), if \( d(i, j) < d(i, k) \). If we assume that all the delays are distinct, then for every node \( i \), corresponding to the \( i \)th row of the weight matrix, we can associate a total order of the nodes in the system. This order is called a preference order.

Given the weight matrix \( W \) (from above), we derive the following set of preference orders\(^2\).

\[
\begin{align*}
  a : & \ b > c > d > e \\
  b : & \ c > d > a > e \\
  c : & \ d > b > a > e \\
  d : & \ a > e > b > c \\
  e : & \ d > a > b > c
\end{align*}
\]

Let us now assume that every node has one vote which it could assign to any of the other nodes. In order to determine the winner (elected leader), the simplest scheme would be to tally all the votes and declare the node with the largest number of votes to be the leader. In the example above, node \( a \) votes for \( b \) (its best preference) and so on. Thus node \( d \) will be declared the leader since it gets two votes while all the rest get one vote apiece.

There are several noteworthy features about the election scheme described above. First, and most important, each node has precisely one vote which it may assign to any of the other nodes. In order to determine the winner (elected leader), the simplest scheme would be to tally all the votes and declare the node with the largest number of votes to be the leader. In the example above, node \( a \) votes for \( b \) (its best preference) and so on. Thus node \( d \) will be declared the leader since it gets two votes while all the rest get one vote apiece.

There are several noteworthy features about the election scheme described above. First, and most important, each node has precisely one vote which it may assign to any of the other \( n - 1 \) nodes. Therefore, even if a node has poor estimates for the delays, since it only has one vote,
it cannot affect the outcome of the elections in a significant way. Second, the individual preferences of the nodes are considered in determining the leader, unlike traditional leader election schemes that elect leaders based on ID numbers. The preference-based approach we propose can be summarized simply as,

1. Nodes estimate their delays to other nodes in the system.
2. Given these estimates, a node determines its vote.
3. The votes are exchanged among the nodes using some algorithm (described in section 7).
4. The votes are then combined using a given election scheme (several election schemes are described in section 4) to determine a leader. This leader is called the elected leader.

It should be noted that preference-based election schemes do not always elect the optimal leader. However, as we will show in sections 5 and 6, the leader elected is optimal with a very high probability and on the average, has a system-wide performance very close to that of the optimal leader.

In the next section we present several election schemes that we will analyze in this paper.

4 Election Schemes

We have studied the following election schemes, the first four of which are borrowed from social choice theory.

**Plurality:** This is probably the most familiar scheme. Each node assigns one vote to its most preferred choice. The votes are summed up to determine the leader.

**Approval:** In this scheme, a node assigns one vote to every candidate it considers above average (i.e., to each candidate which will provide a level of performance that is better than the average performance level provided to the node, averaged over all candidate leaders). All the votes are then tallied to determine the elected leader.

**Borda:** Let $a_1, \ldots, a_{n-1}$ be $n-1$ nodes excluding node $i$. For node $i$, let $d(i, a_1) < d(i, a_2) \ldots < d(i, a_{n-1})$; then node $i$ assigns $n-1$ votes to $a_1$, $n-2$ votes to $a_2$ and so on. All the votes are tallied to determine the elected leader.

**Copeland:** In this scheme, a plurality election is run for each pair of nodes. The *copeland score* of a candidate is defined as the number of candidates it beats in the pairwise elections. The candidate with the highest copeland score is declared the winner.

**PluApp:** This scheme is a combination of plurality and approval election schemes. Nodes assign two votes to the most preferred candidate and one vote to all others it considers above average. A tally of votes determines the eventual winner.

**Fractional:** In this scheme, node $i$ assigns a real-valued vote equal to $d(i, j)/(\sum_{j=1}^{n} d(i, j))$ to candidate $j$. The votes are tallied to determine the winner. Note that the sum of the real-valued votes is 1, and the votes assigned reflect the relative performance level of the different candidates.

**Proportional:** A node $i$ assigns to a node $j$, $[vd(i, j)/(\sum_{j=1}^{n} d(i, j))]$ votes, where $v$ is a constant. For large $v$, this scheme approximates the fractional scheme discussed above.

**Logscale:** This scheme was motivated by the observation that a preference might have an unusually high weight for a particular node that, as a result, is in fact the optimal leader. Thus, we need to emphasize the difference in weights. Each node $i$ assigns a weight of $2d(i, j)$ to candidate $j$. Given these new weights and a set of votes $v$, nodes assign votes by the same technique as is used by proportional voting.

**Random:** As the name suggests, one of $p$ candidates is chosen to be the leader at random. It serves as a benchmark algorithm against which to compare all of our other election schemes. Note that as previously mentioned, traditional leader election

---

3 In the case of a tie, the elected leader is chosen randomly from the set of winners.

4 Where $\lfloor a \rfloor$ is the integer closest to $a$. 
algorithms are random in the sense that the preferences of
the nodes are not considered while electing a leader.

5 Measures of Performance

In the previous section, we presented several leader
election schemes. As we have mentioned earlier however,
these election schemes do not always elect the optimal leader. It is of interest
to determine the quality of the elected leader w.r.t. the optimal leader. To this end, we
define two measures of performance of an election scheme.

1. The probability that an election scheme
   elects the optimal leader, and

2. The expected value of the relative difference
   in the performance level provided to the sys-
   tem by the optimal leader and the elected
   leader. We refer to this difference as the
erelative error, $r_e$. If node $j$ is elected, node $o$ is optimal and node $k$ is the worst possi-
ble choice for leader, then for a given weight
matrix we define,

$$r_e = \frac{\sum_{i=1}^{n} d(i,o) - \sum_{i=1 \neq j}^{n} d(i,j)}{\sum_{i=1}^{n} d(i,o) - \sum_{i=1 \neq k}^{n} d(i,k)}$$

In the next section we will evaluate the per-
formance of different election schemes based on
the two metrics defined above.

6 Results

It is clear that the particular system topology
will determine the performance of different election
schemes. Since we do not want to restrict
ourselves to a particular system topology (be-
cause the results that we would obtain would
then be very specific and not applicable gener-
ally), we study the performance of the election
schemes over a set of randomly generated sys-
tem topologies. In the literature, these objects
are called weighted random graphs.

The process by which such graphs are generated
is as follows. Let the number of nodes in the
graph be $n$. Then a random graph is constructed
by the following process. First $n$ labeled nodes
are thrown onto a plane. Then for each of the
$n(n-1)/2$ possible edges, a coin is tossed. The probability of the coin coming up “heads” is $p$.

If the coin comes up heads, we include that edge
in the graph, otherwise it is left out. At the end
of this process, we have a random graph. Given
such a random graph, we replace the edges by
bidirectional links and attach weights to these
links. The weights are chosen i.i.d. from some
probability distribution function. The graph ob-
tained by this process is denoted by $G(n,p)$ (in
general, $p$ is a function of $n$).

6.1 Simulation Results

For our simulations, we generated random
weighted graphs using the process discussed
above. The number of nodes ranged from 3 to
15. Link weights were chosen from a uniform dis-
tribution over $[0,1]$ in one set of experiments and
from an exponential distribution with parameter $\lambda = 1$, for the second set of experiments. The
purpose of studying these two cases was to deter-
nine the sensitivity of different election schemes
to the choice of distribution. An election scheme
that performs well and that is indifferent to a
choice of distribution would be well suited to sys-
tems with unknown pdf's for the link delays.

For each of the two cases described above, we
ran two sets of experiments. One with the proba-
bility $p = 0.25$ and another with $p = 0.75$. When
the probability $p$ is small, we expect the graphs
to be sparse and when $p$ is large, the graphs tend
to be dense. We were interested in determining
the effect of graph density on the performance of
different election schemes.

The two measures used to determine the per-
formance of different election schemes were, the
probability that the optimal leader is elected and
the expected relative error. For each experi-
ment, 20,000 graphs were generated and 5%
confidence intervals were computed for each of
the two measures. The confidence interval half-
widths kept to less than 5% of the point values.

For $p = 0.25$, Figure 3 shows the plot of the
probability of electing the optimal leader versus
$n$ for the different voting schemes when the link
weights are uniformly distributed. Note that,
Fractional, Borda, Proportional and PluApp are
most likely to elect the optimal leader. It also
appears that most of the election schemes possess
an asymptotic limit, although we have not been
able to characterize this limit.

In Figure 4 we plot the expected relative er-
or between the optimal and the elected lead-
ers when \( p = 0.25 \). Almost all the leader election algorithms with the exception of random, logscale and plurality perform equally well, and have achieved a relative error to within 5% of optimum, even for small values of \( n \). This is a significant result because it implies that even though the optimal leader may not be elected always, the leader that is elected has a performance level very close to optimal! It is noteworthy that when the distribution chosen was exponential, almost identical results were obtained. Furthermore, the results remained unaffected for \( p = 0.5 \) and \( p = 0.75 \).\cite{12}

To summarize our simulation results then, we note that the election schemes, in general, do not always (but frequently do) elect the optimal leader. Also, for most of the schemes studied, the performance of the elected leader is very close to that of the optimal leader. The size of the graph has little effect on the relative orderings of the election schemes in terms of performance. However, the absolute performance of the election schemes does change. Furthermore, the probability of an edge (i.e., \( p \)) does not affect the behaviour of the different election schemes. These two observations would suggest that the election schemes can be used in a wide range of distributed systems.

### 6.2 Theoretical Results

In this section, we present various theoretical results about the asymptotic behaviour of various election schemes. We begin by defining some terms. We are interested in determining the behaviour of an election scheme over the space \( G(n,p) \), as \( n \to \infty \). We say that a typical element (graph) has a property \( Q \) when the probability that a random graph on \( n \) vertices has \( Q \) tends to 1 as \( n \to \infty \).

**Definition 4:** Almost every (a.e.) graph has a property \( Q \) if,

\[
\lim_{n \to \infty} \Pr(G(n,p) \text{ has } Q) = 1
\]

A property \( Q \) is said to hold almost surely (a.s.) if a.e. graph has property \( Q \). We prove below that most election schemes elect the optimal leader almost surely and that the expected relative error is 0, almost surely. Before we can proceed, we need to state some results from the theory of random graphs that will be useful.

**Theorem 1:** (see \cite{4}) Let \( c \in \mathbb{R} \) be fixed and let \( p(n) = (\log n + c + o(1))/n \). Then,

\[
\lim_{n \to \infty} \Pr(G(n,p) \text{ is connected}) = e^{\exp(-e^{-c})}
\]

This theorem implies that for a very large range of values of \( p(n) \), almost every random graph is connected. The following theorem also proved in \cite{4}, shows that for a large range of values of \( p(n) \), almost every graph has a diameter of 2.

**Theorem 2:** Suppose \( p^2n - 2\log n \to \infty \) and \( n^2(1 - p) \to \infty \), then almost every graph has a diameter of 2.

We use these two results to prove asymptotic properties of some of the election schemes we have studied. Assume that the link weights are all 1. Then we have,

**Lemma 1:** (see \cite{12}) If \( p(n) \) is as in the theorem above, then the optimal leader is the node with the highest degree a.s.

**Lemma 2:** (see \cite{12}) In a diameter 2 graph, the leader elected by plurality, approval, plusapp, borda and copeland election schemes has the highest degree.

As a result of the two lemmas above, we obtain,

**Theorem 3:** If \( p(n) \) satisfies the conditions of the theorem above, the leader elected by plurality, approval, plusapp, borda and copeland is optimal a.s.

The probability of electing the optimal leader thus approaches 1 as \( n \) approaches infinity. However, this in itself does not constitute a proof that the relative error approaches zero. The next theorem characterizes the difference in the expected performance of the optimal leader and the elected leader. Let \( \mu^{opt} \) and \( \mu^{elec} \) denote the expected column sums of the optimal and the elected leaders respectively.

**Theorem 4:** (see \cite{12}) In random graphs with \( p(n) \) as in theorem 2, \( \lim_{n \to \infty} \mu^{opt} - \mu^{elec} = 0 \).

The proof techniques that have been used unfortunately cannot be generalized to weighted.
random graphs. While we have several preliminary results that deal with weighted random graphs, we have as yet been unable to generalize theorems 3 and 4 above.

In the next section, we present algorithms that may be used to implement the election schemes we have discussed.

7 Algorithmic Implementation

We assume that the system is represented as a connected graph on \( n \) nodes. We assume that nodes only suffer from fail-stop failures\(^6\), and that messages are not lost or garbled. Messages are delivered in the order in which they were sent.

Space restrictions prevent us from presenting our algorithms here. However, we would like to briefly describe the two sets of algorithms we have developed. In the first set of algorithms, we assume the existence of a spanning tree over which nodes exchange votes. We have shown that the total number of messages exchanged is \( 3(n - 1) \) for all election schemes except copeland. For copeland, \( O(n^2) \) messages are required. The second set of algorithms are based upon existing algorithms developed in the literature on gossiping and broadcasting (see [8]). The message complexity of these algorithms is also \( O(n) \), however no assumption of a spanning tree is made. A detailed description of these algorithms may be found in [12] and in an upcoming paper.

8 Conclusions

The major contribution of this paper is the novel approach to the problem of leader election in distributed systems. The preference-based algorithms that we have presented are both general and from a practical standpoint, directly applicable to real systems.

The problem of preference-based leader election has been studied in some depth in [12]. In addition to the delay model, two other system models, the reliability model and the weight matrix model, have been studied. We have addressed the issue of the robustness of election schemes. That is, we have developed models to study the effect of a few misreported votes (i.e., some nodes vote randomly and their votes are unrelated to the weight matrix entries) on the performance of election schemes. These results will be published in a future paper.

In our work, we have addressed several issues of preference-based leader election, however, many interesting open problems still remain. Our current work focuses on generalizing the theoretical results to the case of weighted random graphs.

References

Figure 3: Probability of electing the optimal leader, $p = 0.25$.

Advanced Textbooks in Economics, Amsterdam (1983).


Figure 4: Expected relative error, $p = 0.25$.

Legend:

- a – Approval
- b – Borda
- c – Copeland
- f – Fractional
- g – Logscale
- n – Proportional
- p – Plurality
- r – Random
- y – PluApp