JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 21, 184–201 (1994)

Electing "Good" Leaders

SURESH SINGH*,† AND JAMES F. KUROSE‡

*Department of Computer Science, University of South Carolina, Columbia, South Carolina 29208; and ‡Department of Computer Science, University of Massachusetts, Amherst, Massachusetts 01003

In a distributed system, an algorithm used to select a distinguished node or *leader* to coordinate some activity is known as a *leader election algorithm*. While traditional approaches to leader election, based upon node ID numbers, are of considerable theoretical interest, we believe these approaches are not very practical since the leader elected will not necessarily deliver good performance. In this paper, we propose and examine socially inspired leader election schemes that attempt to locate the leader at a "good" node (from a performance standpoint) in the system. Each node uses locally available information to *vote* for the various candidates. These votes are combined using an election scheme to determine the leader. The election schemes we propose and examine are shown to perform almost as well as a traditional optimization-based approach toward leader election and are shown to be robust to node failures.

1. INTRODUCTION

In many distributed computer systems there exists a need to identify one or more nodes that perform some centralized function in the system. Such distinguished nodes are called *leaders*. For example a leader might coordinate system reorganization after failures [5] (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 [2, 8]. An application in which more than one leader is required is the case of *replicated files*. Here, k nodes out of n need to be chosen as locations at which copies of a replicated file are stored. All of these k nodes may be thought of as leaders. The algorithms that close such a leader are called *leader election* algorithms.

1.1. Problems with Traditional Approaches

Almost all existing leader election algorithms may be characterized as *extrema-finding algorithms* in which 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 [1, 4, 5]. While these efforts are of considerable theoretical interest, they suffer from several practical considerations. First, when a leader is elected on the basis of an ID number, it is chosen without regard to the *preferences* of the individual nodes or, equivalently, the performance level the system would realize if a particular node were elected 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 communications network would make a poor choice for a leader. When a leader is to be elected, that leader should ideally represent a *good* choice for the system, e.g., from a performance or reliability standpoint, as opposed to a choice determined by some arbitrary criterion such as node number. A second problem with existing leader election algorithms is that of *robustness* and *reliability:* a node can mistakenly have itself elected leader simply by reporting a extremely large ID number.

1.2. Preference-Based Approach to Leader Election

This paper argues that "preference-based" leader election algorithms, in which a leader is elected on the basis of the *preferences* of the nodes can overcome the deficiencies in existing leader election algorithms. In this approach each node in the system votes for the various candidates on the basis of the performance level it would realize under each possible leader (e.g., the expected communication delay or the 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 several advantages to such a preference-based approach. First, preference-based algorithms by definition incorporate *performance* considerations into the election process. Secondly, depending on the manner in which individual preferences are combined to form a system-wide decision, the preferences of an errant node

† E-mail: singh@cs.scarolina.edu.

0743-7315/94 $6.00

are but one set of preferences which are considered in electing a leader; we would thus expect preference-based leader election algorithms to exhibit a certain degree of *robustness* with respect to such failures. Finally, there is already a large body of existing literature on leader election (voting) in the area of Social Choice Theory [10] upon which we may draw. We believe that many of the specific voting mechanisms and models previously developed in this area provide useful blueprints for engineering similar decentralized leader election algorithms in distributed computer systems.

1.3. Problem Statement

The idea of preference-based elections is new in a computer science context and we therefore formalize it in the next section. The rest of the paper is then devoted to proving the two claims made above, that the elected leader is good from a performance standpoint and that the preference-based election process is robust.

In order to subtantiate the first claim, several election schemes are studied in a specific system model, the *reliability model*. We show that the elected leader is often optimal and on the average displays a system-wide performance that is close to optimal. The claim of robustness is studied in another system model called the *abstract weight matrix model*. We show that many of the election schemes elect a good leader even in the presence of several failures.

The rest of the paper is organized as follows. Section 2 provides a formal framework for the preference based approach. Section 3 defines two metrics that are used to determine the *quality* of the elected leader. Section 4 defines the two system models studied here, the reliability model and the weight matrix model. Section 5 focuses on leader election in the reliability model and shows that the elected leader is good from a performance standpoint. Section 6 discusses the issue of robustness in the context of the abstract weight matrix system model. Implementation issues are discussed in Section 7, and some open problems are discussed in Section 8.

2. FORMAL FRAMEWORK

There are two parts of our framework. The first part deals with a definition of the *optimal leader(s)* in the system. This definition is based upon the function that will be served by the leader(s) and may therefore be dependent upon several system-wide parameters, such as the system topology, the rate at which requests arrive at the leader, traffic patterns, and the reliability of the links.

The second part deals with the problem of actually finding the optimal leader(s) in a *decentralized* way. Our proposed approach assumes that nodes are *selfish*. Thus, from the point of view of some node i, a good leader will be a node that maximizes some *local function* at node i. In order to reach a consensus on a leader, all the nodes in the system vote. These votes are combined using some *election scheme* to find the leader(s). Note that, during the election process, the only information exchanged between nodes is their vote. Let us now formalize these ideas.

Let us assume that the system contains n nodes (labeled $S = \{1, 2, \ldots, n\}$) all of which are *potential* leaders (or candidates). Out of these, $1 \leq l \leq n$ nodes need to be elected to be the leader(s). We assume that there exists a function f: $S \rightarrow \Re$ (where $\Re$ represents the the real numbers) such that $f(j)$ represents the *system-wide performance* that would be realized if node j were to be elected the leader ($f(j)$ will also be called the *performance* of node j, as a shorthand). We will then say that node j is a *better choice* for leader than node k if $f(j) > f(k)$. A set $L^{opt} = \{a_1, a_2, \ldots, a_l\} \subseteq S$ will be called the *optimal* set of leader(s) if it satisfies the *optimality property* stated below.

DEFINITION 1 (Optimality Property). The set of nodes L^{opt} is called the set of optimal leaders iff, $f(a_j) \geq f(i)$ $\forall a_j \in L^{opt}$, $\forall i \in \overline{L^{opt}}$. The set L^{opt} thus contains the l best leaders.

Next, we assume that there exist functions g_i, $i \in S$, g_i: $S \rightarrow \Re$, where $g_i(j)$ represents the level of performance node i would receive if node j were elected the leader. The function g_i can only be *estimated* by node i and it is used by node i to determine its vote.

We say that node i *prefers* node j to be the leader over node k if $g_i(j) > g_i(k)$. Let the vote of node i, as determined by the function g_i, be represented by v_i and let the collection of n votes be represented by $V = \{v_1, v_2, \ldots, v_n\}$. If e is some election scheme, then we define the *elected leader(s)* as,

DEFINITION 2 (Elected Leaders). The set of nodes $L^e = \{b_1, b_2, \ldots, b_l\}$ that are *elected* by the election scheme e, i.e., $e(V, S) = L^e \subseteq S$, are called the elected leaders.

Let us now illustrate these concepts with an example.

EXAMPLE. Let a system consist of four nodes, $S = \{a, b, c, d\}$ as shown in Fig. 1, where the numbers on the links indicate edge delays. Let us further assume that $l = 2$, that is, *two* leaders are to be elected. The function $f(j)$ is defined to be the *negative* average point-to-point delay (we use a negative value only for convenience since Definition 1 requires the optimal leader(s) to maximize $f(j)$) from node j to every other node in the system. It is easy to see that $f(a) = -1.5$, $b(b) = -2.5$, $f(c) = -1.5$, and $f(d) = -2.5$. Therefore, by Definition 1, $L^{opt} = \{a, c\}$.

The nodes in the system can, however, only *estimate* these point to point delays. Let us assume that the esti-

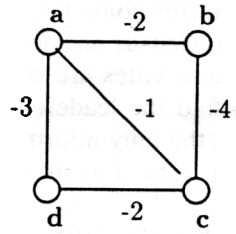

FIG. 1. Example of a four-node network.

mated values of these delays are as shown in the table below. $g_i(j)$, the number in the ith row of the jth column, is the estimated delay from node i to node j:

$$g = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0.0 & -3.0 & -2.2 & -4.1 \\ b & -1.0 & 0.0 & -5.0 & -3.0 \\ c & -2.0 & -3.0 & 0.0 & -3.0 \\ d & -5.0 & -1.5 & -1.0 & 0.0 \end{array}$$

Finally, for the election scheme e, assume that every node has one vote which it may assign to any one of the other candidates (we assume that nodes may not vote for themselves in order to avoid situations in which all nodes receive one vote each) and the winner(s) are the ones that obtain the maximum number of votes. Then it is easy to see that node a votes for c giving $v_a = c$, and similarly, $v_b = a$, $v_c = a$, and $v_d = c$ and therefore, $L^e = \{a, c\}$. In this example we see that the nodes that are *elected* are the same as the nodes that are *optimal*, however, this may not always be the case.

It should be pointed out that there are, in general, many different types of elections and correspondingly many different forms of voting. We will present several election schemes in the last part of this section and discuss the types of voting used.

A natural question that arises is are the two sets, L^{opt} and L^e identical? If they are not always identical, how often will they be identical? Furthermore, are there properties of the system that will guarantee that these sets be equal? Unfortunately there is no simple answer to these questions because there are several factors that influence the final outcome of an election. Specifically, the relationship between the functions g_i and f, the form of voting used and the election scheme used to determine the leader(s), all play an important role. We will discuss some of these questions in later sections.

2.1. Election Schemes

In [12, 14], we have begun to study the behavior of several election schemes that are presented below. When we first began looking at the problem of preference-based leader election, the election schemes we used were the ones that already existed in the social choice literature. However, with a better understanding of the problem, we developed our own election schemes that are better suited to a distributed computer system environment.

Election schemes from social choice theory (see [9, 17]):

Plurality: This is probably the most familiar election scheme. Each node gives one vote to its most preferred candidate. The votes are then summed and the node with the highest tally is declared elected. This is the election scheme used in the example discussed above.

Approval: A node assigns one vote to each candidate it considers above average. The leader is the node that gets the maximum number of votes. In the example above, node a estimates an average $-(3.0 + 2.2 + 4.1)/3 = -3.1$, and therefore assigns one vote each to c and b.

Veto: Each node vetoes one other node that it considers to be the worst possible choice. The candidate not vetoed is declared the leader. Node a vetoes node d in the example above. It is possible for all candidates to be vetoed, in which case a different election scheme is used to elect a leader.

Borda: For some node i, let $g_i(a_1) > g_i(a_2) > \cdots > g_i(a_n)$ for some labeling of the nodes, $a_1, a_2, \ldots, a_n$. Then node i assigns node (or candidate) a_1, $n - 1$ votes, node a_2, $n - 2$ votes, and so on. The leader is a node with the maximum number of votes.

Copeland: We form all possible pairs of nodes and run a plurality election for each pair. The node that beats the most other nodes in such elections is declared the elected leader.

We present our own election schemes below. In each, the elected leader is the node with the highest vote tally.

PluApp: This scheme is a combination of plurality and approval voting. Each node assigns two votes to its most preferred candidate and one vote to all other candidates it considers *above average*. The leader is the node with the most votes. During extensive simulation runs of both the Plurality and Approval election schemes, we noticed that very often the optimal leader was elected *either* by plurality or by approval (but not by both). By combining these two election schemes we hoped to increase the probability that the optimal leader is elected. As the next section will show, this hope was justified.

Fractional: We assume that each node has one *vote* that can be shared among all the candidate nodes. Therefore, a candidate node may receive 0.01 votes. Node i assigns node j a vote $g_i(j)/\sum_{j=1}^{n} g_i(j)$. The intuition behind this election scheme is that the fractional vote more accurately reflects the amount by which node i prefers node j in comparison to the others.

Logscale: This scheme was motivated by the observation that occasionally $g_i(j)$ might be unusually high for some j and this fact is reflected in the system-wide performance of node j (i.e., $f(j)$). Thus, rather than a simple

fractional vote as in the case of fractional voting, there is a need emphasize the *difference* in the values of the $g_i(j)$'s. Therefore, node i assigns to node j a vote equal to $[v2^{g_i(j)}/\sum_{j=1}^{n} 2^{g_i(j)}]$ ($[a]$ = the largest integer closest to a). Where v is a fixed number of votes at a node.

Proportional: This scheme is similar to fractional with the difference that instead of fractional votes only integer votes may be assigned. Thus, node i assigns node j a vote of $[vg_i(j)/\sum_{j=1}^{n} g_i(j)]$, where v is a fixed number of votes at a node.

Finally, we devised a simple election scheme to serve as a *benchmark* and also to represent the *traditional models* of leader election, where a leader is elected based upon an ID number with no regard to where a leader is elected based upon an ID number with no regard to its performance. We call this the *random* election scheme.

Random: A node is elected at *random* from among the set of n nodes.

Having presented several election schemes above, we can now make the distinction between a *correct* vote and an *incorrect* vote.

DEFINITION 3 (Correct Vote). The vote computed by a node, *for a given election scheme*, is correct if this vote is identical to the vote that would be computed by this node (based upon its function g_i) were it to have complete knowledge of the system configuration.

Consider the previous example. The delays computed by node a (first row of the matrix) are estimates of the actual delays to nodes b, c, and d, respectively. Even though these delay estimates are not accurate, for Plurality elections, node a correctly votes for node c. Therefore node a's vote is said to be *correct*. If the delay estimates had been, say, -1.0, -2.0, -3.0 instead, then node a would cast its vote for node b. This is an *incorrect* vote.

In Section 5, we assume that the votes of all nodes are correct. In other words, each node has *estimates* that are good enough to generate correct votes. It is noteworthy that in order to have correct votes, different election schemes require varying quality of estimates. Thus, the estimates used to generate a correct vote for Plurality elections may not be good enough to generate correct votes for Fractional elections. In Section 6 we study the situation where some nodes have incorrect votes.

2.2. Summary

It is important to remark that there are potentially an infinite number of election schemes possible. It should therefore be theoretically possible to construct an election scheme, given a particular system model, such that the probability of electing the optimal leader is maximized. However, this appears to be a difficult problem and the four election schemes we have developed thus far were based more on intuition, developed as a result of extensive simulations, than any formal theory. This is an area of our current and future research.

In this section, we have defined our framework for leader election and discussed the preference-based approach. In Sections 5 and 6, we will discuss the application of our approach to leader election in two specific system models. Section 3 discusses the methodology used to study this problem.

3. APPROACH AND METHODOLOGY

Given a specific system we would like to answer questions relating to the *quality* of the elected leader. We would like to answer questions such as, how often is the optimal leader elected? Is one election scheme better than another for a specific system model? What is the nature of information on which nodes base their votes? Is it possible to chose the functions g_i (for some f) such that an election scheme always choses the optimal leader?

In our work thus far we have concentrated upon providing *probabilistic* answers to some of the questions posed above. Specifically, we use the following two metrics to determine the quality of the leader that is elected by various election schemes.

1. What is the *probability* that the optimal leader is elected? i.e.,

$$\Pr[\text{optimal leader} = \text{elected leader}] = ?.$$

2. What is the *expected difference in performance* between the elected and optimal leaders (where the performance of a node is given by the function f)? Formally, we define the *expected relative error* as

$$r_e = \frac{f(\text{optimal leader}) - f(\text{elected leader})}{f(\text{optimal leader}) - f(\text{worst leader})}$$

An alternative metric for the expected difference in performance is

$$r_{abs} = \frac{f(\text{optimal leader}) - f(\text{elected leader})}{f(\text{optimal leader})}$$

We chose to use r_e in this work, however, because it is more "pessimistic" in the sense that $r_e > r_{abs}$.

The next section presents two specific system models in which we have studied the behaviour of various election schemes as measures by the two metrics defined above.

—The first model we study in the *reliability model* and we use it to *validate* the preference-based approach to leader election. We show that the elected leader is fre-

quently optimal and the relative error between the performance of the elected leader and the optimal leader is often very small.

—The second model is called the *abstract weight matrix model* and is used to address the issues relating to *robustness*. If some subset of nodes misreport their votes, clearly the leader that is elected may not represent the best choice. We show, however, that several election schemes are quite robust to these forms of failure.

4. TWO SYSTEM MODELS

In this section we present two of the three system models in which we have studied the problem of leader election. Results from the third model, called the *delay model*, have appeared elsewhere [14] and will not be discussed in this paper. The two models studied here are called the *reliability model* and the *abstract weight* matrix model. The reliability model is intended to be less abstract than the weight matrix model, and to be more representative of the concerns that arise in distributed systems. The weight matrix model is an abstract framework that we employ to discuss issues relating to *robustness*.

4.1. The Reliability Model

Consider a simple distributed system shown in Figure 2; a, b, and c are three nodes of the system connected by communication links. The numbers on each link denote the probability that the link is functioning; this may alternatively be viewed as the probability that a message sent over this link is successfully received at the receiving node. The *path reliability* from node a to node c is 0.5 along the link (a, c) and 0.81 along the path $a - b - c$. Therefore, a message sent from a to c via b has a higher chance of arriving at its destination than if it were sent out along the link (a, c). Intuitively, in Fig. 2, node b would be a better leader than either of nodes a or c because node b has paths of reliability 0.9 to both a and c, while node a (or c) has paths of reliability 0.9 and 0.81 to each of the other two nodes. This idea of maximum path reliability is used to define the performance metrics, g_i and f, in our model.

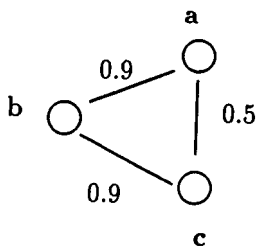

FIG. 2. Path reliabilities.

We assume the distributed system is represented by a weighted connected graph with *bidirectional edges*. The weight on a link is a number between 0 and 1 and represents the probability that the link is operational. We assume that the weight on a directed link (i, j) is different from the weight on the directed link (j, i). This assumption is valid because a message may be lost either from the queue or from the physical link itself during transmission. While the probability of the message being lost during transmission may be the same, the probability of the loss of a message from the queue at node i is independent of the same probability at node j.

Let us now define the system-wide performance of a node. Let T be a *directed spanning tree* rooted in some node j (all the nodes have directed paths *to* node j. There are no outgoing edges from node j in this tree). At any particular instant a link (i, k) will be operational with a probability of p_{ik}. Let x denote the number of nodes that are connected to node j in three T (note that because of link failures some nodes may not be connected to j in T). Then define

$$U_T(j) = \sum_{x=1}^{n-1} xq(x),$$

where

$q(x) = Pr(j$ is connected to exactly x nodes in $T)$.

$U_T(j)$ is the expected number of nodes that j is connected to in T. We then define $f(j)$, the system-wide performance of node j as,

DEFINITION 4. The system-wide performance of a node j is

$$f(j) = \max_T U_T(j)$$

The node with the highest value for $f(j)$ is defined to be the *optimal leader*. The number $f(j)$ is also called the *expected connectivity* of node j [12]. The *optimal leader* is a node that has the maximum expected connectivity. The intuition behind this definition is the following. Given a leader in a graph, every node sends information to the leader along the maximum reliability path. When a link failure occurs, even if the graph is not disconnected, some maximum reliability paths may have to be recomputed. By defining the optimal leader to be the node with the highest expected connectivity, the number of such recomputations in the case of link failure is minimized.

Every individual node in the system can realistically estimate its *path reliability* to the other nodes in the system (perhaps by keeping track of the number of retransmissions required). This defines the functions g_i.

DEFINITION 5. The performance of node j as estimated by node i, $g_i(j)$, is the *maximum* path reliability from node i to node j.

The relationship between the $g_i(j)$ and $f(j)$ is expressed by the theorem below. The proof of this theorem appears elsewhere [13].

THEOREM 1. *If $f(j)$ is the expected connectivity of node j, then*

$$f(j) = \sum_{i=1}^{n} g_i(j)$$

if $g_i(j)$'s are the correct maximum path reliabilities from node i to node j.

Consider the six-node system shown in Fig. 3. Construct a matrix W such that the entries w_{ij} represent the reliability of the *maximum reliability path* from node i to node j. For example, the path $e - d - c - b$ has reliability 0.2 (which is the product of the link reliabilities, ed, dc, cb) and represents the maximum reliability path from node e to node b:

$W =$

w_{ij}	a	b	c	d	e	f
a	1.0	0.2	0.35	0.7	0.25	0.5
b	0.3	1.0	0.9	0.54	0.108	0.15
c	0.24	0.5	1.0	0.6	0.12	0.12
d	0.4	0.25	0.5	1.0	0.2	0.2
e	0.48	0.2	0.4	0.8	1.0	0.6
f	0.8	0.16	0.28	0.56	0.5	1.0
$f(j)$	3.22	2.31	3.43	4.2	2.178	2.57

In the example above, node d is the optimal leader with an expected connectivity of 4.2.

In summary, in the reliability model, each node computes its maximum reliability paths to each of the other nodes in the system and uses these estimates to vote to chose a leader (this would correspond to each node, i, estimating the ith row of the matrix W). In a real system, these entries are *estimated* by the nodes. As we stated earlier, we assume that these estimates are good enough for a node to vote on; however, these estimates are not good enough to add together to determine the optimal leader.

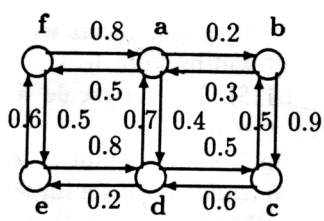

FIG. 3. Example of a six-node system.

4.2. The Random Weight Matrix Model

The random weight matrix model provides us with an abstract probabilistic framework in which to examine the properties of various election schemes. Let us assume that the system contains n nodes and p candidates ($p \leq n$). Let w_{ij} denote the performance level that would be seen by node i if candidate j were elected the leader. The collection of all these *weights* may be represented by a matrix $W(n \times p)$. We assume that the entries w_{ij} are i.i.d. random variables from some common probability distribution (pdf). It is noteworthy that the entries w_{ij} corresponded to maximum reliability paths in the reliability model.

A natural *measure of performance* of a given candidate j can be derived from the four axioms given below. Let $f(j)$ denote the system-wide performance of candidate j and assume that $f(j) > f(k)$ iff j has a higher level of system-wide performance.

Axioms.

A1. $f(a) = U(w_{1a}, \ldots, w_{na})$, where U: $\Re^n \to \Re$.

A2. *Anonymity:* If $x_i = w_{ia}$ then, $U(\ldots, x_i, \ldots, x_j, \ldots) = U(\ldots, x_j, \ldots, x_i, \ldots)$.

A3. *Monotone Increasing*: $U(x_1, \ldots, x_n) < U(x_1 + \Delta x, \ldots, x_n)$, $\Delta x > 0$.

A4. *Linear Importance*: $U(x_1, \ldots, x_i + \Delta x, \ldots, x_n) = U(x_1, \ldots, x_j + c_{ij}\Delta x, \ldots, x_n)$, $c_{ij} \geq 0$.

Axiom A1 says that the system-wide performance of any candidate is a function of its performance as seen by each individual node. The condition of anonymity means that the node ID is not important in the computation of f for different candidates. The axiom for monotonicity is that the system-wide performance of a candidate a increases if its performance w_{ia} as seen by any node i increases. Finally, according to the axiom of linear importance, the system-wide performance of a candidate when x_j increases by a small amount Δx is the same as when x_i increases by some proportional amount $c_{ij}\Delta x$. This axiom captures the linear relationship between the nodes in their effect upon the system-wide performance of a candidate.

From A2 and A4 we derive

$$U(\ldots, x_i + \Delta x, \ldots, x_j, \ldots) = U(\ldots, x_i, \ldots, x_j + c_{ij}\Delta x, \ldots)$$

and

$$\begin{aligned} U(\ldots, x_i + \Delta x, \ldots, x_j, \ldots) &= U(\ldots, x_j, \ldots, x_i + \Delta x, \\ &= U(\ldots, x_j + c_{ij}\Delta x, \ldots, x_i, \ldots) \\ &= U(\ldots, x_i, \ldots, x_j + c_{ji}\Delta x, \ldots). \end{aligned}$$

Therefore, $c_{ij} = c_{ji}$ for all pairs i, j.

The isoperformance curve in $x_i - x_j$ space (plot x_i vs x_j, leaving all other x_k unchanged, such that $f(a) = c$, a constant), has a slope of -1 at every point (x_i, x_j). Since this is true for every pair x_i and x_j, we get that

$$f(a) = \lambda \sum_{i=1}^{n} w_{ia},$$

where l is an arbitrary constant. Axiom A3 dictates that $\lambda > 0$.

Thus, we see that summing the weights is a "natural" measure of system-wide performance for a candidate since it satisfies the four reasonable axioms stated above. By the derivation above observe that it is the *only* function that satisfies the four axioms. A social welfare function that has the same form has also been derived by Ng [11].

We are now in a position to formally state the random weight matrix model. In the abstract framework that we have developed, we assume that each of the utility values w_{ia}, are chosen i.i.d. from some probability distribution. The system-wide performance level of a candidate a is then given by the sum, $\sum_{i=1}^{n} w_{ia}$. And an optimal candidate is one that maximizes system-wide performance. Formally,

DEFINITION 6 (Performance of a Candidate). If j is a candidate and if w_{ij} (equivalently $g_i(j)$) denotes its performance level for node i, then the system-wide performance of j is given by $f(j) = \sum_{i=1}^{n} w_{ij}$.

DEFINITION 7 (Optimal Leader). Candidate k is defined to be the *optimal leader* iff, $f(k) \geq f(j), j \neq k, 1 \leq j \leq p$.

We consider three different probability distributions for the w_{ij}'s: the exponential distribution, the uniform distribution and the normal distribution. In each case we study the performance of the *elected* leader versus the optimal leader.

5. RELIABILITY MODEL: RESULTS

In this section we discuss the behaviour of the different election schemes in the reliability system model. Observe that results obtained for specific system topologies may not be applicable generally. Thus, if the system was a *star* network then the node at the center of the star would always be the optimal leader and it would also be the node to be elected by most election schemes, irrespective of the link reliabilities.

Therefore, in order to make general statements about the election schemes it becomes necessary to consider a wide range of system topologies. For this reason we studied *randomly generated* system topologies and provide answers to some of the questions posed in Section 3 over this randomly generated set. In the next subsection we outline our method of constructing random system topologies that were used in our simulations.

5.1. Weighted Random Graphs

Let the number of nodes in the graph be denoted by n. Then a *random* graph is constructed by the following process. For each of the $n(n-1)/2$ possible edges between n nodes, 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. If the graph is not connected, we throw it away and generate another.

For our system model, we have assumed that the graphs are both weighted and bidirectional. Therefore, we need to add a second step to the process discussed above. Each edge in the random graph is first replaced by bidirectional links. Each directed link is then assigned a delay chosen from some common pdf. The graph thus obtained is taken to represent a distributed system.

The set of graphs generated by the process described above is represented as $G(n, p)$ and it is over this space that we study the performance of different election schemes as defined by the two measures presented earlier. Note that, in general, if $p \ll 1$ then the graphs will tend to be sparse while if p is large, the graphs will tend to be dense.

5.2. Simulation Results

For our simulations, we generated random weighted graphs with link weights drawn from a *uniform* distribution on [0, 1]. The number of nodes n varied from 3 to 20. Three sets of experiments were run for each value of n. In each the probability p (the probability that an edge exists) took on a different value, $p = 0.25$, 0.5, and 0.75. Different values of p were used to determine the effects of the graph density on the behavior of the election schemes.

For each experiment we generated 20,000 connected weighted random graphs. We computed the probability that the optimal leader was elected by each of the election schemes and the expected relative error in the performance between the optimal leader and the elected leader. In each case the number of leaders to be elected is one and if multiple leaders are elected, the tie is broken randomly. 95% confidence intervals were generated and the confidence half-widths kept to less than 5% of the point values, see [15]. For Logscale and Proportional election schemes, $v = n$.

For $p = 0.25$, Fig. 4 shows a plot of the probability of electing the optimal leader versus n for the different election schemes. Note that Fractional, Borda, and Copeland are most likely to elect the optimal leader (with a proba-

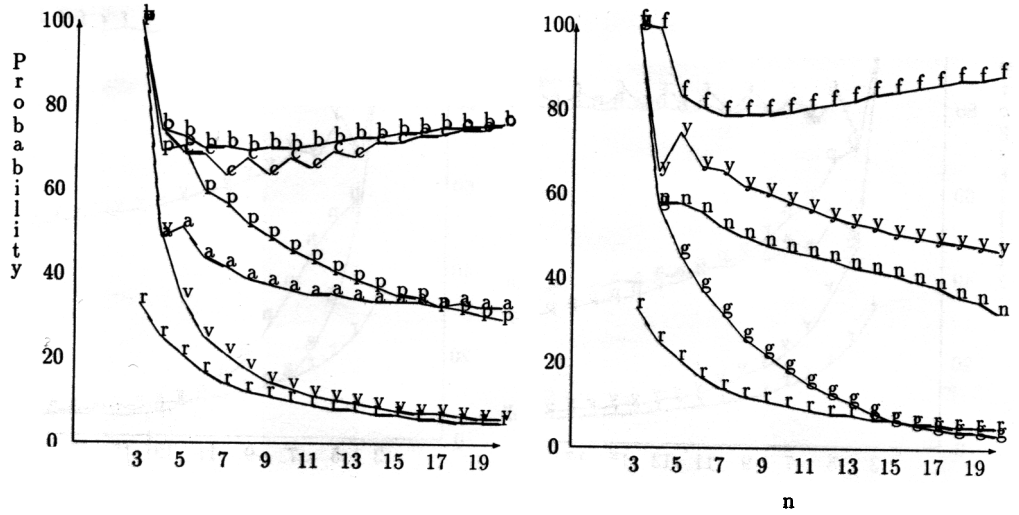

FIG. 4. Probability of electing the optimal leader, $p = 0.25$. a, Approval; b, Borda; c, Copeland; f, Fractional; g, Logscale; n, Proportional; p, Plurality; r, Random; v, Veto; y, Pluapp.

bility of around 0.8) while Veto, Random, and Logscale are the least likely to do so. Plurality and Approval appear to have the same asymptote and have a probability of electing the optimal leader that lies in between. PluApp elects the optimal leader with a higher probability than either Plurality or Approval.

Figures 5 and 6 show the same plots for when the edge probability is 0.5 and 0.75, respectively. First, we note that the *relative* performance of the election schemes is unchanged. However, it appears that in denser graphs, Fractional elects the optimal leader with a greater probability (around 0.95). A similar trend is observed in the behavior of Borda and Copeland. Plurality, Approval, and PluApp are relatively unaffected.

Figures 7, 8, and 9 plot the expected relative error versus n for the different election schemes for the three values of p. Note that Fractional, Borda, and Copeland have errors that are less than 3%, while PluApp has an error that is less than 5%. For $p = 0.25$, Proportional has a low error but this error quickly increases with the graph density. Veto, Random, and Logscale have errors approaching 40%.

To summarize our simulation results, we note that Fractional, Borda, and Copeland provide the best performance. The three probability appears to have little effect on most election schemes. However, Proportional is the most affected, while PluApp is least affected.

5.3. Summary

The results presented in this section demonstrate the power and feasibility of the preference-based approach to

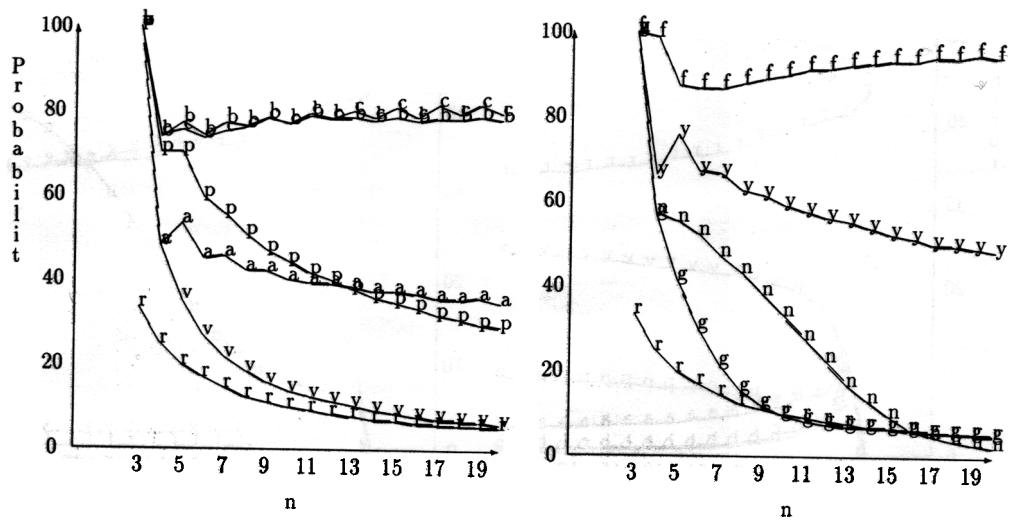

FIG. 5. Probability of electing the optimal leader, $p = 0.50$. Letters as in Fig. 4.

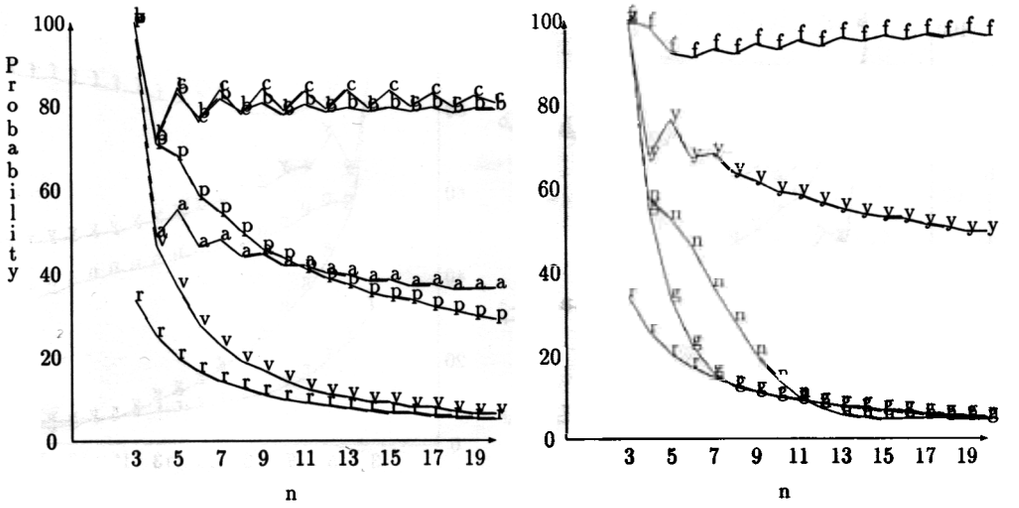

FIG. 6. Probability of electing the optimal leader, $p = 0.75$. Letters as in Fig. 4.

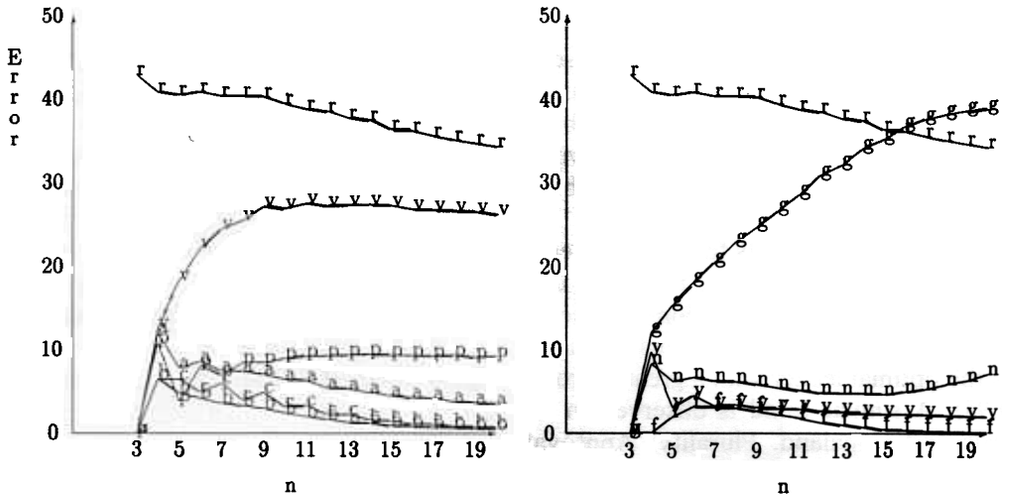

FIG. 7. Expected relative error, $p = 0.25$. Letters as in Fig. 4.

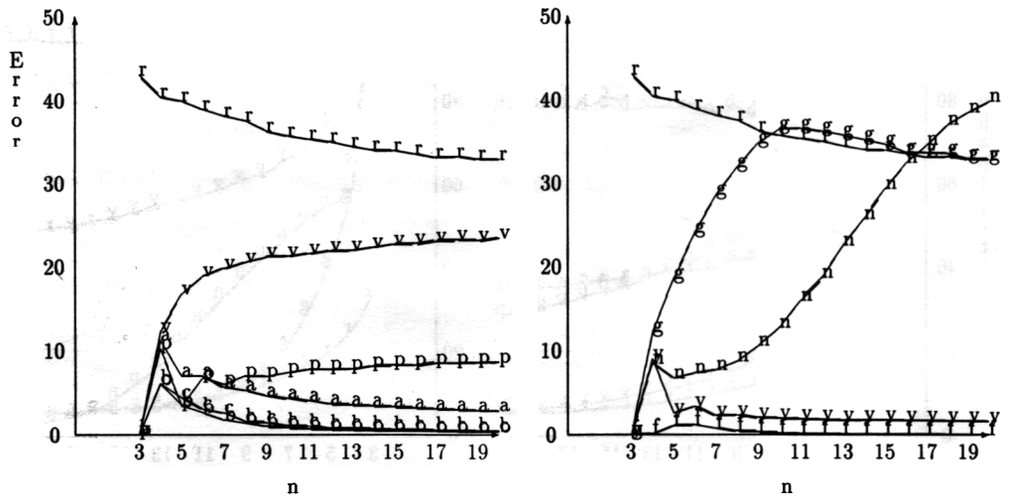

FIG. 8. Expected relative error, $p = 0.50$. Letters as in Fig. 4.

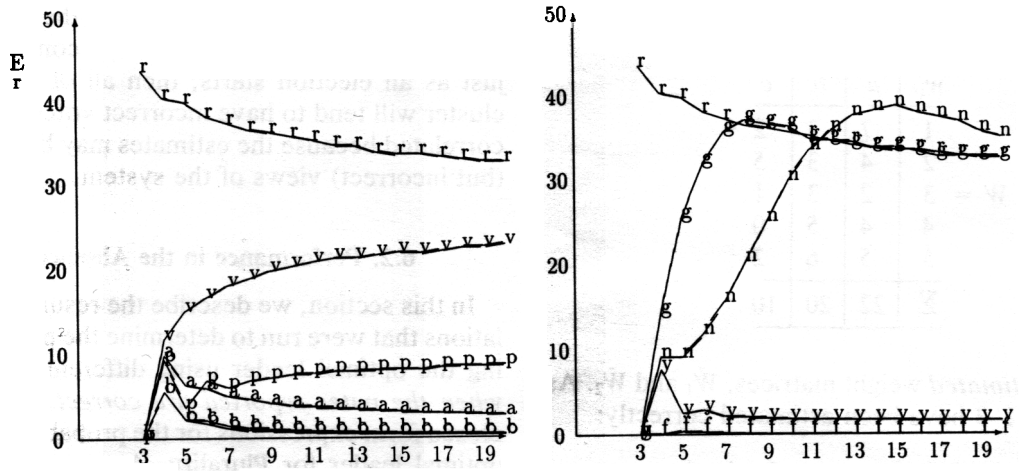

FIG. 9. Expected relative error, $p = 0.75$. Letters as in Fig. 4.

leader election. As we have seen several election schemes elect the optimal leader with a high probability and in cases when the optimal leader is not elected, the elected leader has a performance level close to optimal.

6. ROBUSTNESS AND THE WEIGHT MATRIX MODEL

In the previous section we studied the behavior of various election schemes for the *reliability* system model. As we saw, several election schemes elect the optimal leader with a high probability and in the cases where the optimal leader is not elected, the elected leader has a performance level very close to optimal. An assumption that was implicit in those studies was that every node reported its *correct* vote (see Section 2.1, Definition 3). In this section, we are interested in studying the behavior of election schemes when some nodes *misreport* their votes. Nodes that misreport their votes are said to have suffered a *failure*.

It is important at this point to distinguish our usage of the term *failure* from the manner in which it is most frequently used in the computer science literature. Failure usually means a failure of the *communication channel*, leading to lost or garbled messages, and to *crash* failures of nodes. In our usage, however, a *failed* node functions correctly; it only misreports its vote.

In computer systems, this scenario of misreported votes is not unrealistic. Nodes base their votes on their *estimates* of the performance of the various candidates. The quality of these estimates determines the correctness of the vote. Consider a situation in which an election is called before some particular node has been able to produce "good" estimates (possibly because it has only just recovered from a crash); in this case, the node's vote might well be incorrect.

As we will see, one way to minimize the damage done by failed nodes is to give the vote of each node equal importance. For example, we would not allow one node to have two votes while all the rest have one, because if the node with two votes were to fail, it would count as *two* failures. We believe that this property of assigning each node a single vote makes many of our election schemes robust.

The remainder of this section is organized as follows. We first define two models of failure in Section 6.1. In Section 6.2 we discuss the behavior of the election schemes in the *abstract weight matrix model* in the *absence of failure*. Sections 6.3 and 6.4 then discuss the effect failure has on the performance of the various election schemes in the weight matrix model.

6.1. Models of Failure

Before defining the two models of failure, we would like to reiterate the difference between a *correct* and an *estimated* weight matrix. Let w_{ij} of the correct weight matrix W represent the actual performance node i would realize if node j were elected the leader. However, as previously discussed, a node can only estimate these quantities. Thus node i would estimate row i of the matrix W. Let us denote this estimated matrix by $\tilde{W}$.

When nodes vote, the votes are based upon the estimated weight matrix $\tilde{W}$. It is not the purpose of this paper to address the problem of developing good estimation techniques; rather, we are interested here in determining the behavior of election schemes in the presence of differing W and $\tilde{W}$ matrices.

EXAMPLE. Assume that five nodes vote to elect a leader out of three possible candidates. The *correct*

weight matrix is shown below:

$W =$

w_{ij}	a	b	c
1	7	3	2
2	4	3	5
3	2	3	1.
4	4	5	0
5	5	6	2
Σ	22	20	10

Consider two *estimated* weight matrices, $\tilde{W}_1$ and $\tilde{W}_2$. Assume that rows 1–3 have been estimated correctly:

$\tilde{W}_1 =$

w_{ij}	a	b	c
.		.	
.		.	
4	6	7	2,
5	3	4	0

$\tilde{W}_2 =$

w_{ij}	a	b	c
.		.	
.		.	
4	6	7	20
5	3	4	30

Consider now the effect of the incorrect votes on the quality of the leader elected. Let Borda be the election scheme being used to determine the leader. Given the correct weight matrix W, candidate b is elected (a receives 11 votes, b receives 12, and c receives 7) the leader. The exact same votes are also reported with the estimated weight matrix $\tilde{W}_1$ (and therefore $\tilde{W}_1$ yields correct votes by Definition 3). However, if the estimates in $\tilde{W}_2$ are used, candidate c is elected since nodes 4 and 5 report incorrect votes.

6.1.1. Two Types of Failure

It is clear from the example above that misreported votes may significantly distort the final outcome of the election process. There are a large number of possible failure models. We however chose to study the two described below because the effectively represent a large set of possible failures in *real systems*.

DEFINITION 8 (Random Failure). The entries of one (or more) rows of the estimated weight matrix are unrelated to the corresponding entries of the correct weight matrix.

Thus the votes of the failed nodes will be *random*. The second model of failure works in much the same way as random failure discussed above; however it is more malacious in practice. We assume that the nodes which fail distort their vote *together*.

DEFINITION 9 (Correlated Failure). If $\mathcal{M}$ is the set of failed nodes, then every node $i \in \mathcal{M}$ assigns an extremely large weight w_{ij} to the same candidate j in the estimated weight matrix. The other candidates are assigned random weights.

An example of such a failure is the following. If one cluster of a network fails and then comes back up again just as an election starts, then all of the nodes in that cluster will tend to have incorrect votes that may well be correlated because the estimates may be based on similar (but incorrect) views of the system.

6.2. Performance in the Absence of Failure

In this section, we describe the results of several simulations that were run to determine the probability of electing the optimal leader using different election schemes *when the votes reported are correct*. We have derived closed form expressions for the probability of electing the optimal leader for Plurality, Approval, and Veto using the Partition Theorem from number theory, see [6, 12].

We present two sets of simulations. In one set, the number of voting nodes n is equal to the number of candidates p. In the second case, the set of candidates is a subset of the set of nodes, i.e., $p < n$.

Case 1: $n = p$. The first case we study is one in which the number of voting nodes is equal to the number of candidates running for election. The value of n is allowed to range from 2 to 15. The entries of the weight matrix are i.i.d. random variables chosen from three different probability distributions. The Exponential distribution with mean $\lambda = 1$, the Normal distribution $N(0,1)$, and the Uniform distribution between [0,1]. Our goal in using three different probability distributions was to determine the sensitivity of the election schemes to the underlying distribution of the weight matrices.

Under the assumptions stated above, the simulations were conducted as follows. For each value of n and for each choice of probability distribution, 20,000 weight matrices were generated. In each matrix, the optimal leader was determined as that having the maximum column sum (see Section 4.3, Definition 7). For each election scheme, the leader elected was found and compared with the optimal leader. In cases where more than one leader was elected, the ties were broken *randomly*.

We calculated 95% confidence intervals for the values of the probability, Pr[elected leader = optimal] and relative error for each election scheme, at each value of n for the different distributions. The resulting confidence halfwidths were less than 5% of point values.

In this paper we present results for the case when the weights are chosen from a uniform distribution only. The behaviour of most of the election schemes is unaffected by a choice of distribution. Figure 10 plots the probability of electing the optimal leader as a function of n for various election schemes. The first observation to be made is that Random performs very poorly. It elects the optimal leader with a probability $1/p$, as expected. Veto also performs very poorly because a large number of nodes are tied for leader and these ties are broken randomly. Both

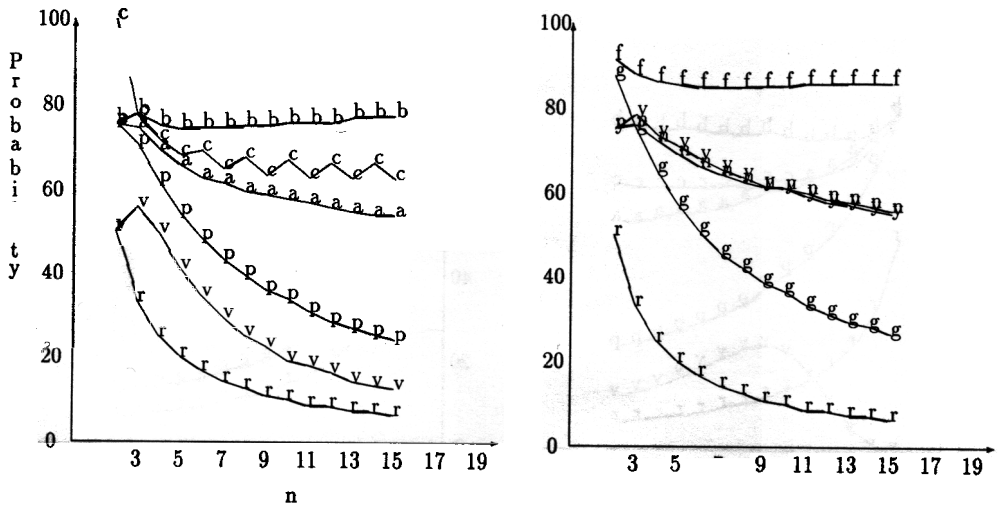

FIG. 10. Probability of electing and optimal leader (uniform). Letters as in Fig. 4.

Fractional and Borda perform very well. An intuitive explanation for this is that *on the average*, the highest weight in any row of the weight matrix lies in the interval $[(n-1)/n, 1]$, the second highest weight lies in the interval $[(n-2)/(n-1), (n-1)/n]$, and so on. The assignment of votes in Borda closely duplicates this pattern (i.e., the highest weight is given a vote of n, the second highest $n-1$, and so on) and therefore the optimal leader is elected with a high probability. Both Fractional and Proportional (with $v = n$) perform well because the voting information exchanged reflects not only the preferences of the nodes but the relative strength of those preferences. The "saw-tooth" behavior of Copeland is a result of there being an *odd* or *even* number of nodes in the system.

If the pdf was the exponential distribution, Logscale elects the optimal leader with a high probability, approaching 60%. The performance of Borda and Copeland drops to around 40% though. In the case of a normal distribution, Proportional elects the optimal leader with a probability approaching $1/n$. Copeland and Borda elect the optimal leader with a probability between 65–75%.

In Figure 11 we plot the expected relative error as a function of n for the case when the weight matrix entries are chosen from a uniform distribution. Observe that the relative error between the elected and optimal leaders for Borda, Copeland, Approval, Fractional, PluApp, and Proportional is less than 10%. In the case that the weights are chosen from an exponential distribution. Plurality and Logscale also have errors that are less than 10%. In

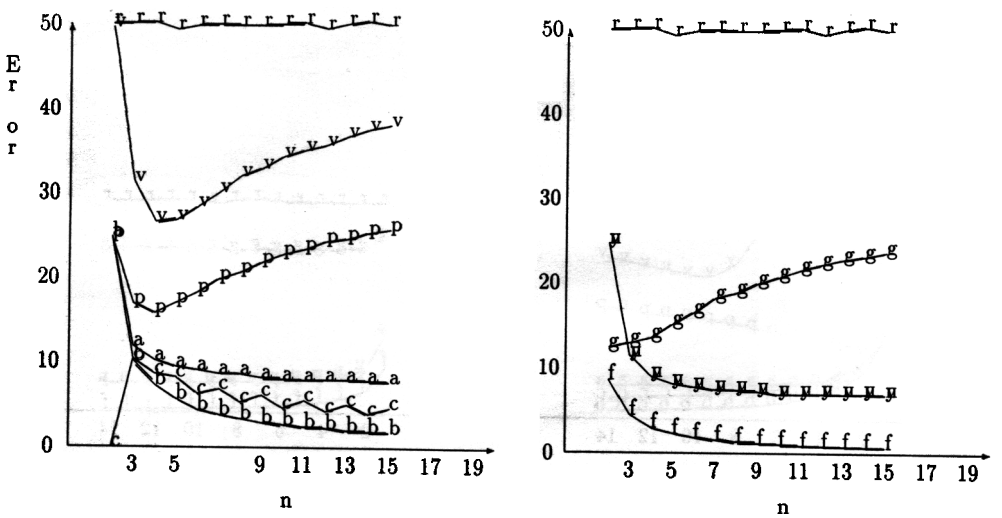

FIG. 11. Expected relative error (uniform). Letters as in Fig. 4.

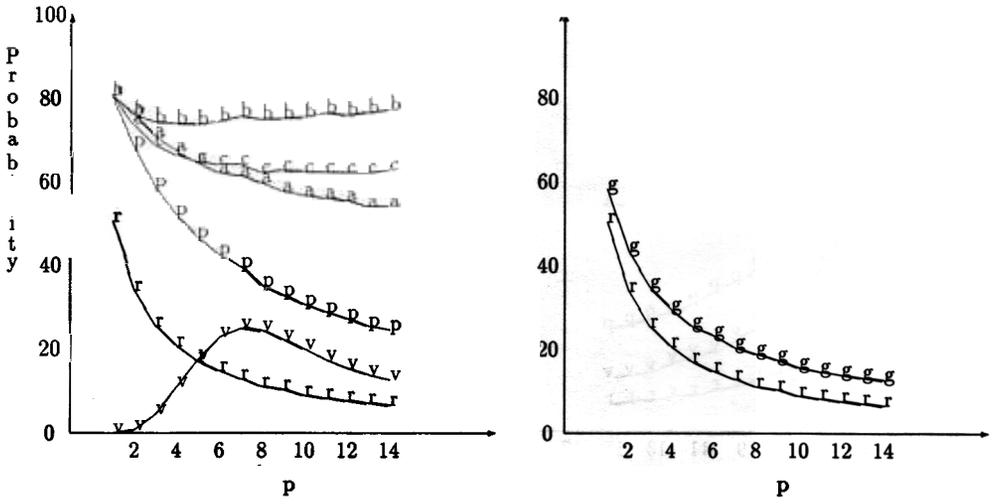

FIG. 12. Probability of electing the optimal leader (uniform). Letters as in Fig. 4.

the case of a normal distribution, Fractional and Plurality have errors in excess of 20%. Thus we see that with respect to the relative error metric, most of the election schemes elect a *good* leader.

Case 2: $p < n$. In this set of experiments we kept n fixed at 15 and varied p, the number of candidates, between 2 and 14. For each value of p, we generated 20,000 weight matrices and computed 95% confidence intervals. In each case the interval halfwidths were within 5% of the point values.

Figure 12 plots the probability of electing the optimal leader as a function of p for the case when the weight matrix entries are chosen from a uniform distribution. Random elects the optimal leader with a probability of $1/p$, while for $p \leq 7$ Veto elects the optimal leader with a probability *smaller* than $1/p$; this happens because in many instances all the candidates are vetoed! Both Borda and Fractional elect the optimal leader with a high probability for reasons already stated. As the number of candidates increases, the probability with which Plurality elects the optimal leader decreases rapidly because the number of tied winners increases rapidly. The same is true for Logscale.

In Fig. 13 we plot the expected relative error as a function of p. For small values of p since Veto may not elect any leader, its performance is very poor. Note that Borda, Fractional, Copeland, Approval, PluApp, and Proportional have a relative error that is less than 10%. In fact, Fractional has an error smaller than 5% in most instances.

In summary we note the following properties. If the pdf is *symmetric* (as in the case of the uniform or the normal

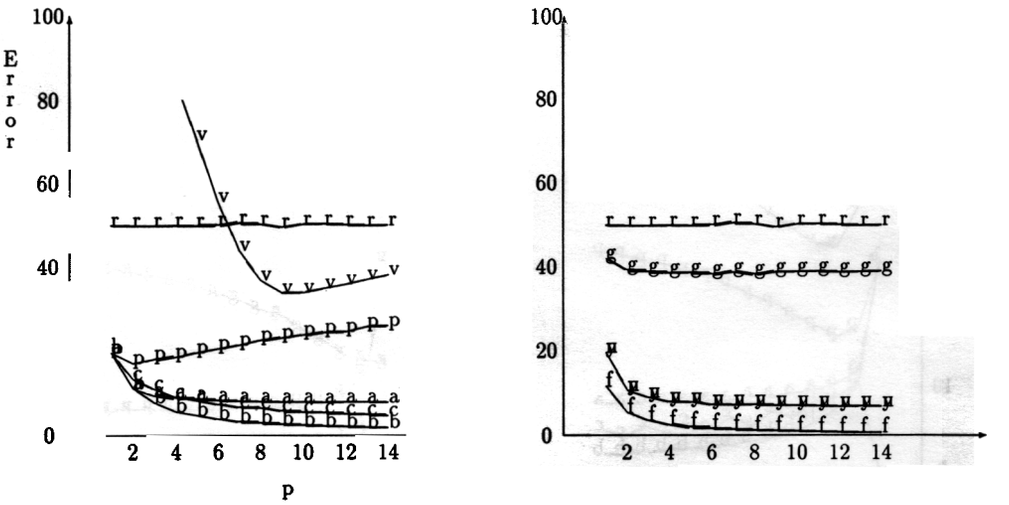

FIG. 13. Expected relative error (uniform). Letters as in Fig. 4.

distribution), PluApp, Borda, and Copeland consistently elect the optimal leader with a high probability. Fractional and Proportional show a good performance if the distribution is uniform. If the pdf is exponential, PluApp, Fractional, and Proportional elect the optimal leader with a high probability. In terms of sensitivity to choice of distribution, PluApp is by far the least sensitive. Finally, in general we observe that for a given number of voting nodes, the number of candidates will not drastically affect the probability with which the optimal leader is elected. A surprising result!

Comparing these results with those obtained for the reliability model we observe that Borda and Copeland consistently elect the optimal leader in both models with a high probability. This is also true for the delay model [14]. We conjecture that these two election schemes will do well in other system models as well.

6.3. Effects of Random Failure

Let us now turn our attention to the situation when some set of nodes report incorrect votes. Let m denote the number of such failed nodes. It is clear that some election schemes will be able to "tolerate" more failures than others for a comparable "degradation" in performance. Unfortunately, there is no single definition for the degree of robustness since we use two measures, the probability of electing the optimal leader and the relative error, to evaluate the performance of election schemes. However, corresponding to each of these two measures, we define two different measures of robustness.

DEFINITION 10 (Robustness w.r.t. the Probability of Electing the Optimal Leader). A number $\Re^{pr}_{r,\delta}$, that denotes the maximum number of failures m such that,

$$\frac{|\Pr[\text{optimal} = \text{elected} \mid m \text{ failures}] - \Pr[\text{optimal} = \text{elected} \mid \text{no failures}]|}{\Pr[\text{optimal} = \text{elected} \mid \text{no failures}]} \leq \delta.$$

Similarly, let ε denote the maximum tolerable increase in the relative error. Then,

DEFINITION 11 (Robustness in Error). A number $\Re^{err}_{r,\varepsilon}$, that denotes the maximum number of failures m such that,

$$|E[\text{relative error} \mid m \text{ failures}] - E[\text{relative error} \mid \text{no failures}]| \leq \varepsilon.$$

In both these definitions, the subscript r in the measure of robustness denotes the type of failure. In this case it is *random failure*; robustness in the presence of *correlated failure* will use a subscript l.

The two definitions above provide a way of representing the robustness of different election schemes in terms of the maximum number of permissible failures. A higher value for $\Re^{pr}_{r,\delta}$ or $\Re^{err}_{r,\varepsilon}$ would mean a more robust election scheme.

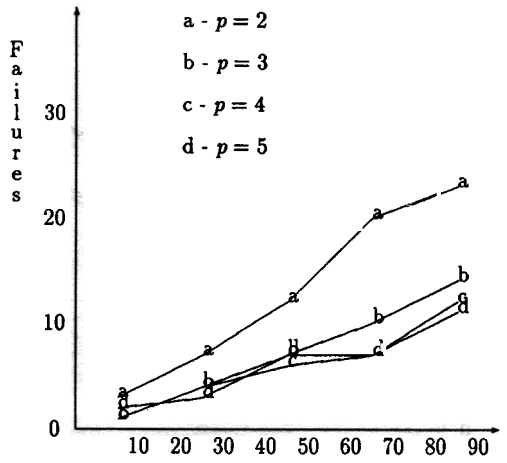

FIG. 14. $\Re^{pr}_{r,0.1}$ vs n for different p.

Case 1: $p \ll n$. In the first set of experiments we assume that the number of candidates p is much smaller than the number of voting nodes n. For the purposes of our experiment, the number of candidates ranged between 2 and 5 and n took values 10, 30, 50, 70, and 90. The larger values of n were chosen to determine asymptotes, if any. We computed the number of random failures required for $\delta = \varepsilon = 0.05$, 0.1, and 0.2.

In the simulations reported here, the weight matrix entries were chosen from an exponential distribution with mean $\lambda = 1$; 95% confidence intervals were generated and the confidence interval halfwidths kept to less than 5% of the point values.

In this set of experiments we found a surprising result. For a given δ (or ε), *all* election schemes can tolerate the same number of failures! This was true for both the measures $\Re^{pr}_{r,\delta}$ and $\Re^{err}_{r,\varepsilon}$. Figure 14 shows a plot of the number of failures $\Re^{pr}_{r,0.1}$ vs n for different values of p. Figure 15 plots $\Re^{err}_{r,0.1}$ vs n.

A possible explanation for this result that all election schemes exhibit the same degree of robustness to random failure is suggested by the following observation. The votes of the m failed nodes when taken together will effectively counterbalance each other out yielding a system in which $n - m$ nodes vote to elect one out of p candidates.

We conclude that all election schemes are *equally* robust under random failures when $p \ll n$ and infact there *appears* to be a linear relationship between the number of failures that can be tolerated and δ (or ε).

Case 2: $n = p$. In this set of experiments the number of nodes varied between 2 and 14. As in the above experi-

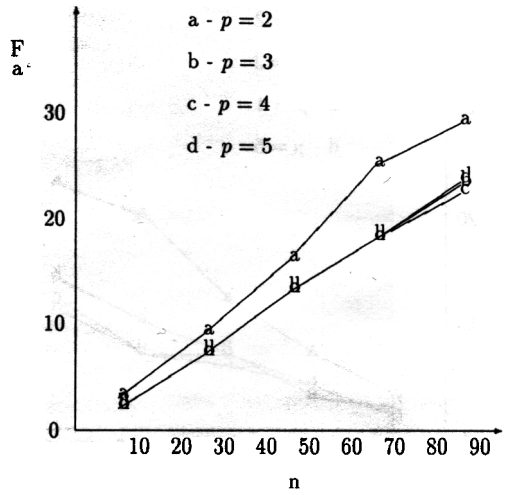

FIG. 15. $\mathfrak{R}^{err}_{r,0.1}$ vs n for different p.

ment, the values of $\delta = \varepsilon$ were chosen to be 0.05, 0.1, and 0.2.

The first experiment measured $\mathfrak{R}^{Pr}_{r,\delta}$. We found that for $\delta = 0.05$ and 0.1 no failures could be tolerated by any of the election schemes while for $\delta = 0.2$ one failure could be tolerated.

The second experiment measured $\mathfrak{R}^{err}_{r,\varepsilon}$. Figure 16 shows a plot of the number of failures that can be tolerated vs n for three values of ε. Note that all the election schemes were able to tolerate the same number of failures.

6.4. Effects of Correlated Failure

Unlike the random failure model studied above, the correlated failure model assumes that all the nodes that fail assign a large weight to the *same* candidate. We would thus expect a much lower degree of robustness as compared to the random failure model.

Case 1: $p \ll n$. As above, the weights of the weight matrix were chosen from an exponential distribution with mean $\lambda = 1$. To simulate correlated failure, a specific node was chosen to be the one to receive a large weight (set arbitrarily at 100). We chose the values $\delta = \varepsilon = 0.05$, 0.1, and 0.2. The number of candidates ranged between $2 \le p \le 5$ and n took values 10, 30, 50, 70, and 90. We computed 95% confidence intervals and determined that the interval half-widths were within 5% of the point values.

In Fig. 17 we plot $\mathfrak{R}^{pr}_{l,0.1}$ vs the number of nodes n for different values of p. The legend used there is: f, fractional; l, logscale; p, proportional; and o, others. We noted that the election schemes Plurality, Approval, PluApp, Borda, and Copeland exhibit the same degree of robustness and are therefore lumped together under the title *others*. Observe that Fractional, Proportional, and Logscale are less robust w.r.t. this metric. These election schemes require a greater accuracy in the estimation of the weight matrix because votes are determined as fractions of the weight matrix entries. The other election schemes require votes that are total orders of the candidates and these schemes can therefore tolerate more correlated failures.

Figure 18 shows a plot of $\mathfrak{R}^{err}_{l,0.1}$ as a function of n. We observe that the election schemes exhibit a similar pattern of behavior.

Case 2: $n = p$. We conducted experiments similar to the ones described above with $2 \le n = p \le 14$. The first result is that for $\delta = 0.05$ and 0.1, *none* of the election schemes can tolerate even one failure. With $\delta = 0.2$, however, all the election schemes except Fractional, Logscale, and Proportional can tolerate upto two correlated failures.

6.5. Summary

In this section we studied the effect of misreported votes on the performance of election schemes. We defined two failure models and showed via simulations that the random failure model is less severe than correlated failures.

In real systems we would expect a failure model that is a mix of both the models we have examined. There will be subsets of nodes that will form coalitions and there will be other nodes that will suffer random failure. Therefore, the robustness exhibited by election schemes in such systems will "lie in between" the robustness predicted by the two models studied here.

Finally, we note that Fractional, Proportional, and

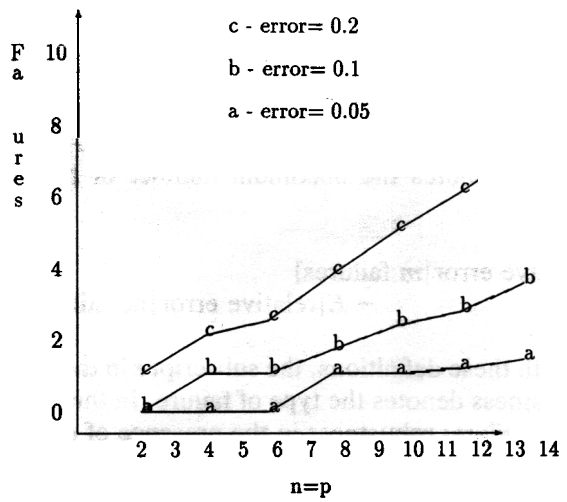

FIG. 16. $\mathfrak{R}^{err}_{r,\varepsilon}$ vs $n = p$ for different ε.

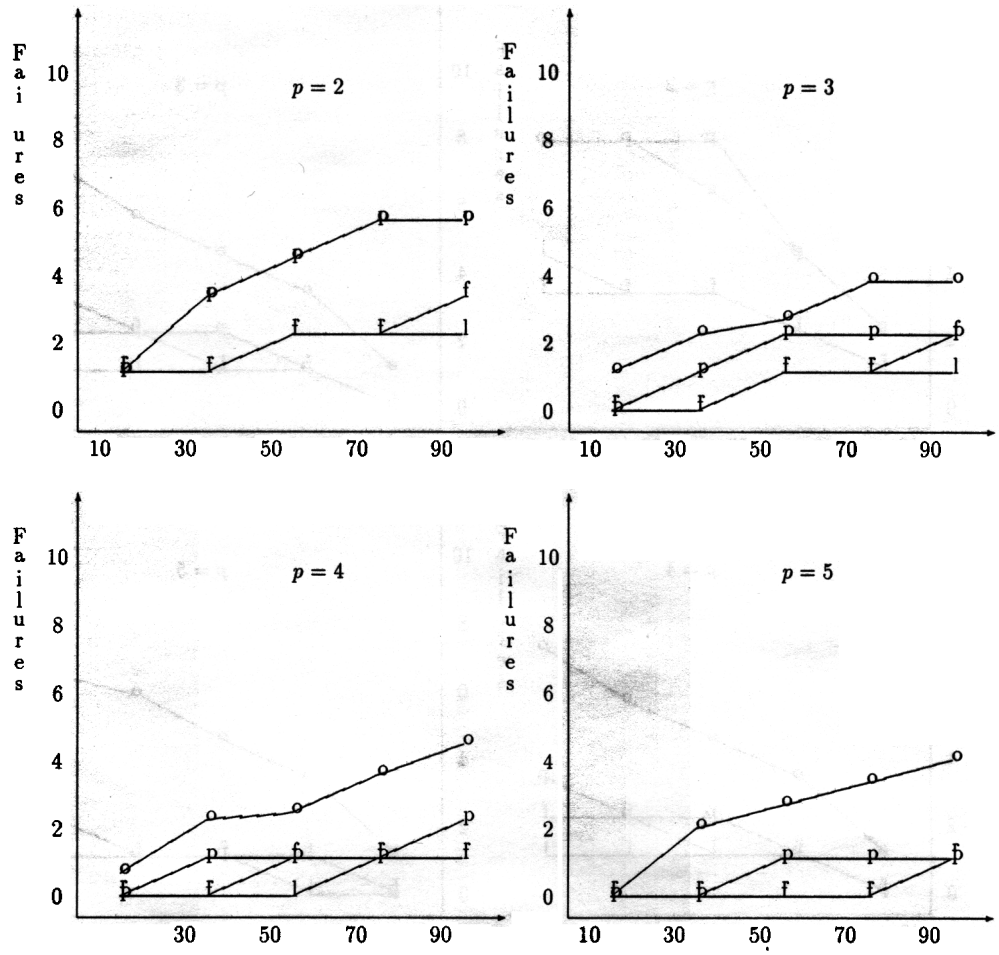

FIG. 17. Plot of $\Re^{pr}_{l,0.1}$ vs n.

Logscale elect the optimal leader with a high probability but are less tolerant to correlated failure than Borda, Plurality, Copeland, Pluapp, and Approval. All the election schemes are equally tolerant to random failure, however.

IMPLEMENTATION ISSUES

The process of leader election, as described in this paper, proceeds in two steps. In the first step, all the nodes determine their vote based upon the local function, g_i, being maximized. The implementation of this first step is system dependent and is not addressed in this paper. The second step involves the collection and collation of votes to determine the leader(s). There are several existing algorithms that can be easily adapted to implement this step.

Gossiping algorithms address the problem of *all-to-all* communication between n nodes. Each node is assumed to possess a "secret" that has to be communicated to all the others in the shortest time. Clearly these algorithms can be easily adapted to the problem of collection of the votes, with every vote representing a secret. The communication cost is shown to be $2n + c$ (c a constant) in [7, 16]. Once all the nodes have received the votes from all the others, they execute the appropriate election scheme to determine the leader(s). With the exception of Copeland every other election scheme requires time $O(np)$ to execute (where n is the number of nodes and p the number of candidates). Copeland requires time $O(np^2)$ since pairwise elections are run between all pairs of candidates.

8. CONCLUSIONS AND FUTURE WORK

We began this paper with the postulate that electing leaders based upon the individual *preferences* of the nodes would yield a leader that gives a high system-wide performance. As earlier sections have shown, this claim has been well justified. We also defined two new models of *failure* that occur when some nodes report incorrect votes. These kinds of failure cannot be detected because

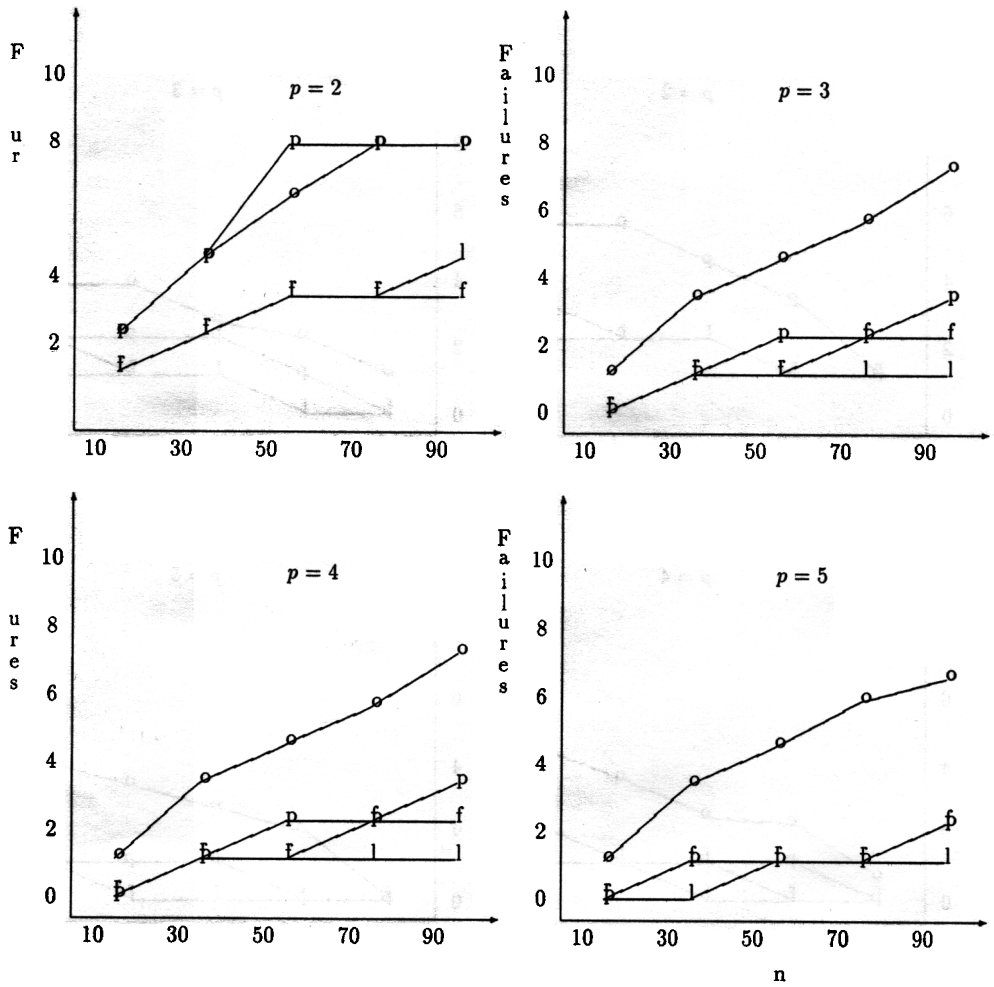

FIG. 18. Plot of $\mathfrak{R}^{err}_{l,0.1}$ vs n.

the incorrect votes are derived from poor estimates of system parameters. The nodes themselves have not failed in the traditional sense. We show that several election schemes can tolerate several such failures without a significant degradation in system performance.

The model for leader election studied here is relatively new and is a rich source of open problems. One interesting generalization is the situation where k leaders need to be elected out of n nodes. An application of this may be in the case of *replicated files*. Another interesting problem is "when is an election held and who calls for new elections?". The simple case is when the old leader dies. However, new elections should be held if the performance of the leader, as seen by the rest of the system, deteriorates. Another interesting question is to find some kind of characterization of graphs that have a high probability of electing optimal leaders (with some appropriate election scheme) to aid designers of networks interested in using leader election for some purpose in their systems. These and several other problems are currently being studied.

REFERENCES

1. Afek, Y. Distributed algorithms for election in unidirectional and complete networks. Ph.D. dissertation, Dept. of Computer Science, UCLA, Los Angles, CA, 1985.
2. Alsberg, P. A., and Day, J. A principle for resilient sharing of distributed resources. In *Proc. 2nd Intl. Conference on Software Engineering*, 1976.
3. Birman, K. Implementing fault tolerant distributed objects. *IEEE Trans. Software Engrg.* **SE-11,** 6 (1985), 502–508.
4. Fredrickson, G., and Lynch, N. Electing a leader in a synchronous ring. *J. Assoc. Comput. Mach.* **34,** 1 (1987), 98–115.
5. Garcia-Molina, H., Elections in a distributed computer system. *IEEE Trans. Comput.* **C-31,** 1 (1982), 48–59.
6. Grosswald, E., *Topics from the Theory of Numbers*. Birkhäuser, Basel, 1984.
7. Hedetniemi, S. M., *et al.* A survey of gossiping and broadcasting in communication networks. *Networks* **18** (1988), 319–349.
8. Menasce, D., Muntz, R., and Popek, J. A locking protocol for resource coordination in distributed databases. *ACM TODS* **5,** 2 (1980), 103–138.
9. Moulin, H., *The Strategy of Social Choice*, Advanced Textbooks in Economics, North-Holland, Amsterdam, 1983.

10. Mueller, D., Public choice: A survey. *J. Econom. Lit.* **14** (1976), 395–433.
11. Ng, Y., Bentham or Bergson? Finite sensibility, utility functions and social welfare functions. *Rev. Econom. Stud.* **42(4),** 132 (1975), 545–569.
12. Singh, S., Preference-based leader election in distributed systems. Ph.D. dissertation, Dept. of Computer Science, Univ. of Massachusetts, Amherst, MA, 1990.
13. Singh, S., Expected connectivity and leader election in distributed systems. *Inform. Process. Lett.* **42** (1992), 283–285.
14. Singh, S., and Kurose, J. Electing leaders based upon performance: The delay model. *11th Intl. Conf. on Distributed Computing Systems,* 1991, pp. 464–471.
15. Trivedi, K., *Probability and Statistics with Reliability, Queueing and Computer Science Applications,* Prentice–Hall, Englewood Cliffs, NJ, 1982.
16. West, D. B., Gossiping without duplicate transmissions. *SIAM J. Algebra Discrete Methods* **3** (1982), 418–419.
17. Young, H. P., Fair allocation. In *Proceedings of Symposia in Applied Mathematics,* Vol. 33. AMS Short Course Lecture Notes, Anaheim, CA, 1985.

SURESH SINGH received his Ph.D. in 1990 in computer science from the University of Massachusetts at Amherst. He received his M.S. from the same institution in 1986 and his B.Tech. degree in computer science from the Indian Institute of Technology, Kanpur, in 1984. He is presently an assistant professor in computer science at the University of South Carolina, Columbia, South Carolina. Professor Singh's research interests are in the areas of distributed computing, performance evaluation, and computer communication networks. Professor Singh belongs to the IEEE.

JAMES KUROSE received his Ph.D. in 1984 in computer science from Columbia University and joined the Department of Computer Science at the University of Massachusetts at Amherst in that year. He is currently an associate professor in that department. His research interests are in the area of high-speed networks (including protocol design, performance analysis and implementation), scheduling issues in real-time systems, and modeling and performance evaluation. Professor Kurose has also been quite active in professional societies. He is currently Editor-in-Chief of the IEEE/ACM Networking journal and he has been Editor-in-Chief of the IEEE Transactions on Communications. He has also been a Guest Editor for the IEEE Journal on Selected Areas in Communications special issues on computer-aided modeling analysis and design of communication systems. He was the Technical Program Co-chair of the 1992 IEEE Infocom Conference (the Joint IEEE Communication and Computer Societies Conference on Computer Communications). He has also served on Program and/or Organizing Committees for a number of conferences and workshops.

Received March 13, 1992; revised November 6, 1992; accepted December 9, 1992