A Probabilistically Correct Leader Election Protocol for Large Groups

Indranil Gupta[1]
Robbert van Renesse
Kenneth P. Birman
{gupta, rvr, ken}@cs.cornell.edu

Technical Report (April 2000)
Department of Computer Science
Cornell University
Ithaca, NY 14853

[1]This work was funded by DARPA/RADC grant F30602-99-1-6532.

A Probabilistically Correct Leader Election Protocol for Large Groups

Indranil Gupta, Robbert van Renesse, Kenneth P. Birman

Abstract

This paper presents a scalable leader election protocol for large process groups with a weak membership requirement. The underlying network is assumed to be unreliable but characterized by probabilistic failure rates of processes and message deliveries. The protocol trades correctness for scale, that is, it provides very good probabilistic guarantees on correct termination in the sense of the classical specification of the election problem, and of generating a constant number of messages, both independent of group size. After formally specifying the probabilistic properties, we describe the protocol in detail. Our subsequent mathematical analysis provides probabilistic bounds on the complexity of the protocol. Finally, the results of simulation show that the performance of the protocol is satisfactory.

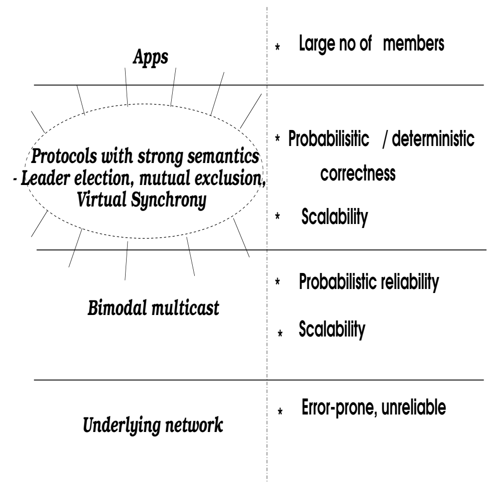

Figure 1: Target Setting

1 Introduction

Computer networks are plagued by crashing machines, message loss, network partitioning, etc., and these problems are aggravated with increasing size of the network. As such, solving protocol specifications which require strong guarantees to distributed applications is difficult, if not impossible, over large-scale networks. Probabilistic and randomized methodologies are increasingly being used to counter this unreliability by reducing strict correctness guarantees to probabilistic ones, and gaining scalability in return. A good example of such a protocol is the Bimodal Multicast protocol [3], an epidemic protocol that provides only a high probability of multicast delivery to group members. In exchange, the protocol gains scalability, delivering messages at a steady rate even for large group sizes.

Our current work is targeted toward realizing the same goals for the important class of protocols that classically have been formulated over reliable multicast message delivery. The specifications of these protocols, which include leader election, mutual exclusion, and virtual synchrony, require deterministic correctness guarantees. However, in results stemming from the famous Impossibility of Consensus proof by Fischer-Lynch-Paterson [12], most of these problems have been proved to be unsolvable in failure-prone asynchronous networks.

In contrast, we envision a world where applications would run over a new class of probabilistic protocols (Figure 1) and receive probabilistically guaranteed services from the layer below. By virtue of the proposed approach, these applications would scale arbitrarily, while guaranteeing correctness with a certain minimal probability even in the face of an unreliable network. For example, these protocols could be used to build a replicated file system with probabilistic guarantees on consistency.

As a step towards this goal, this paper presents a probabilistic leader election protocol. Leader election arises in settings ranging from locking and synchronization to load balancing [16] and maintaining membership in virtually synchronous executions [17]. The classical specification of the leader election problem for a process group states that at the termination of the protocol, exactly one non-faulty group member is elected as the leader, and every other non-faulty member in the group knows about this choice. In this paper, we show that, given probability guarantees on point-to-point (unicast) and multicast message delivery, process failure rates, and multicast group view content, our protocol gives a very high probability of correct termination. In return, it gains on the scalability: with very high probability, the protocol involves only a constant number of messages *regardless of group size*. We also show how to augment our protocol to adapt to changing failure probabilities of the network (w.r.t. processes and messages).

Sabel and Marzullo [24] proved that leader election over a failure-prone asynchronous network is impossible. This and a variety of other impossibility results all stem from the FLP result [12], which proves that there is no protocol by which an asynchronous system of processes can agree on a binary value, even with only one faulty process.

To provide a taxonomy of the complexity of the class of consensus protocols, Chandra and Toueg [6] proposed extending the network with *failure detectors*. For example, the leader election problem can be solved if and only if a perfect failure detector is available - one that suspects no alive processes, and eventually suspects every faulty one [24]. [10] discusses several weakened system models and what types of consensus are possible in these models, while [11] presents a weakened asynchronous model which assumes that message deliveries are always time-bounded. Since "real" systems lack such guarantees, these results have been valuable mostly in a theoretical rather than a practical sense.

Non-randomized leader election algorithms for a failure-prone asynchronous network model broadly fall into the following flavors. 1) Gallager-Humblet-Spira-type algorithms [1, 13, 21] that work by constructing several spanning trees in the network, with a prospective leader at the root of each of these, and recursively reduce the number of these spanning trees to one. The correctness guarantees of these algorithms are violated in the face of pathological process and message failures. 2) Models that create logical partitions in the network when communication becomes unreliable, each logical partition electing one leader [11]. This approach does not solve the scalability problem but circumvents it. 3) Models that involve strong assumptions, for example that all (process) failures occur before the election protocol starts [15, 28], or that all messages are delivered reliably [5].

Probabilistic solutions to leader election in general networks are usually classified as randomized solutions to the consensus problem [8], but these focus on improving either the correctness guarantee [23], or the bound on the number of tolerated failures [29]. The (expected or worst case) message complexities in these algorithms are typically at least linear in the group size, and fault tolerance is usually guaranteed by tolerating process failures up to some fraction of the group size. Further, most of these protocols involve several rounds of $O(N)$ simultaneous multicasts to the group (where N is the group size), and this can cause deterioration of the delivery performance of the underlying network. Our solution assumes a much weaker asynchronous network model than these algorithms, and promises at most a constant number of simultaneous multicasts with very high probability.

A lot of work has also been done on the election problem in specific network topologies such as rings [25], complete networks [26], the full information model that assumes reliable broadcasts [9], anonymous rings [31], wireless broadcast models [7] etc. More recently, *self-stabilizing* leader election has been a focus area as this involves protocols that work correctly even when the system starts in any arbitrary state [2]. An investigation of good criteria for choosing leaders in a group appears in [27].

Our take on the leader election problem is in a more practical setting than any of the above cited works. We are motivated by practical considerations of scaling in a real network where failures can be characterized by probabilities. The spirit of our approach is close to that of [3] and [30]. Our protocol's probabilistic guarantees are similar to those of leader election algorithms for the perfect information model [18, 32], while our guarantee on the number of messages resembles that of [14], which presents an election protocol for anonymous rings. To the best of our knowledge, ours is the first protocol that trades correctness of the leader election problem for better scalability.

The rest of the paper is organized as follows. Section 2 describes the assumed model and statement of the election problem we solve. Section 3 describes the protocol in detail. Section 4 analyses the protocol mathematically, while Section 5 presents simulations results. In Section 6, we present our conclusions.

2 The Model and Problem

2.1 Model

In our model, all processes have unique identifiers (*e.g.*, consisting of their host address and local process identifier). All processes that might be involved in the election are part of a *group*, which can have an arbitrarily large number of members. Each process has a possibly incomplete list of other members in the group, called the process' *view*. A process can communicate to another process in its view by `ucast` (unicast, point-to-point) messages, as well as to the entire group by `mcast` (multicast) messages.

Processes and message deliveries are both unreliable. Processes can undergo only fail-stop failures, that is, a process halts and executes no further steps. Messages (either `ucast` or `mcast`) may not be delivered at some or all of the recipients. This is modeled by assuming that processes can crash with some probability during a protocol round and a `ucast` (`mcast`) message may not reach its recipient(s) with some probability. Probabilistically reliable multicast can be provided using an epidemic protocol such as Bimodal Multicast [3]. The Bimodal multicast protocol guarantees a high probability of multicast message delivery to all group members in spite of failures by having each member periodically gossip undelivered multicasts messages to a random subset of group members in its view.

A few words on the weak group model are in order. As we define them, views do not need to be consistent across processes, hence a pessimistic yet scalable failure detection service such as the gossip heartbeat mechanism of [30] suffices. New processes can join the group by multicasting a message to it, and receiving a reply/state transfer from at least one member that included it in its view.

Our analysis later in this paper assumes a uniform distribution for process failure probabilities (p_{fail}), `ucast`/`mcast` message delivery failure probabilities (p_{ucastl}/p_{mcastl}), as well as the probability that a random member has another random member in its view, which we call the view probability (*view_prob*).

2.2 Problem Statement

An election is initiated by an `mcast` message. This might originate from, say, a client who wants to access a database managed by the group, or one or more member(s) detecting a failure of a service or even the previous leader. In our discussion, we will assume only one initiating message, but the extension of our protocol to several initiating messages follows from it.

In classical leader election, after termination there is exactly one non-faulty process that has been elected leader, and all non-faulty processes know this choice. In probabilistic leader election, with *known high probability*,

- (*Uniqueness*) there is exactly one non-faulty process that considers itself the leader;
- (*Agreement*) all non-faulty group members know this leader; and
- (*Scale*) a round of the protocol involves a total number of messages that can be bounded independent of the group size.

3 Probabilistic Leader Election

This section describes the proposed leader election protocol. The protocol consists of several rounds, while each round consist of three phases. In Section 3.1, we present the phases in a round. In Section 3.2, we describe the full protocol. Section 3.3 presents the pseudo-code and discusses several implementation issues.

3.1 Phases in a Round

Filter Phase

We assume that the initiating `mcast` I is uniquely identified by a bit string A_I. For example, A_I could be the *(source address, sequence number)* pair of message I. Each group member M_i that receives this message computes a *hash* of the concatenation of A_I and M_i's address, using a hash function H that deterministically maps bit strings to the interval $[0, 1]$. Next, M_i calculates the *filter value* $H(M_i A_I) \times N_i$ for the initiating message, where N_i is the size of

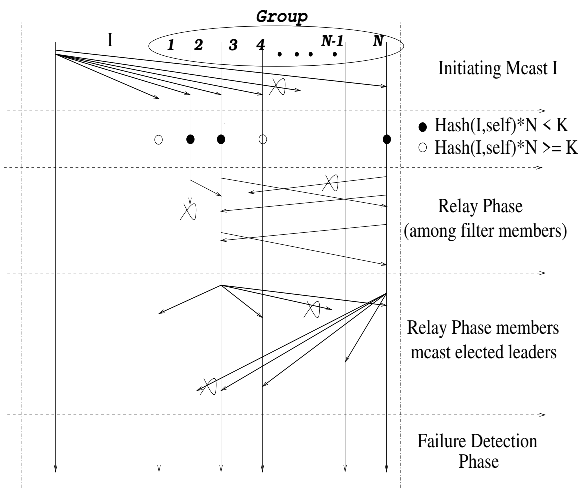

Figure 2: One (Successful) Protocol Round

(number of members in) M_i's current view. M_i participates in the next phase of this round, called the *Relay Phase*, if and only if this filter value is less than a constant K; otherwise it waits until the completion of the Relay phase. We require that H and K be the same for all members. We show in Section 4 that for a good (or fair) hash function H, large total number of group members N, the probability that the number of members throughout the Relay phase lies in an interval near K, quickly goes to unity at small values of K. This convergence is independent of N and is dependent only on the process failure, message delivery failure and view probabilities.

If the N_i's are the same for all members, each member M_i can calculate the set of members $\{M_j\}_i$ in its view that will satisfy the filter condition. It does so by checking if $H(M_j A_I) \times N_i < K$ for each member M_j in its view. In practice, the N_i's may differ, but this will not cause the calculated set $\{M_j\}_i$ to differ much from the actual one. (A more practical approach is to use an approximation of the total number of group members for N_i. This can be achieved by gossiping this number throughout the group. Thus, N_i's of different members will be close and the above filter value calculation will be approximately consistent.)

Figure 2 shows an illustration of one protocol round. The initiating multicast I is multicast to the entire group, but some group members may not receive it since `mcast` delivery is unreliable. The ones who do receive it evaluate the filter condition in the next step. The members labeled with solid circles (2, 3, N) find this condition to be true and hence participate in the Relay phase.

Relay Phase

As explained earlier, a member M_i that has passed the filter and is participating in the Relay phase can calculate the subset of members $\{M_j\}_i$ in its view that would have passed the filter condition if they received I. In the Relay phase, M_i first sends `ucast` messages to all such members in the set $\{M_j\}_i$ specifying M_i's preferred choice for a leader from among its view members. This choice is determined by the ordering generated by a *choice function* which evaluates the advantages of a particular member being elected leader. We require all members to use the same choice function in evaluating their preferred leaders, breaking ties by choosing the process with a lower identity.

Second, whenever M_i is contacted by another member M_k in a similar manner, it includes M_k in its view (and adds it to $\{M_j\}_i$), and compares M_k's choice with its own. If M_k's choice is "better" than its own according to the choice function, M_i *relays* this new choice to all the members in the set $\{M_j\}_i$ by `ucast` messages, and replaces its current best choice for leader. Otherwise, M_i replies back to M_k specifying its current best leader choice.

In the example of Figure 2, the 2^{nd}, 3^{rd} and N^{th} group members enter the Relay phase, but the 2^{nd} member subsequently fails. If either of the 3^{rd} and the N^{th} members has the other in its view, they will be able to exchange relay messages regarding the best leader.

Consider the undirected graph with nodes defined by the set of members participating in the Relay phase (relay members), and an edge between two members if and only if at least one of them has the other in its view throughout the phase. We call this the *relay graph*. Assuming timely message deliveries and no process failures, each connected component of this graph will elect exactly one leader, with a number of (`ucast`) messages dependent only on the size of the component. In Sections 4 and 5, we show that for a good hash function, the likelihood of the relay graph having exactly one component (and thus electing exactly one leader in the Relay Phase), approaches unity quickly at low values of K. Further, this convergence is independent of N and is dependent only on the process failure, message delivery failure and view probabilities. In Section 5, for an example choice function that is widely used in many distributed systems, we show that message delivery and process failures do not affect this convergence. Note that the number of `ucast` messages exchanged in a Relay phase with m members is $O(m^3)$, since each relay member's best choice might be communicated to every other relay member.

Finally, at the end of the Relay phase, when each component has decided on one leader, each member M_i participating in the Relay phase multicasts the identifier of the leader selected by M_i's component (M_i's current best choice) to the entire group—this is the set of *final multicasts* of this election round. The total number of multicast messages in the Relay phase is thus $O(m)$. Since it is likely that m lies in an interval near the protocol parameter K which is chosen regardless of N (analysis of Section 4), this implies only a constant number of `ucast` and `mcast` messages in the Relay Phase with high probability, regardless of the value of N.

In the example of Figure 2, once the 3^{rd} and N^{th} members have agreed on a leader, each of them multicasts this information to the group. Some of the group members may not receive both multicasts, but it is unlikely that every non-faulty member will receive neither.

Notice the possibility of using *any* arbitrary deterministic choice function for selecting good leaders (as long as it is the same for all group members). A good choice function can account for load, location, network topology, etc. [27].

Failure Detection Phase

Consider a situation in which there is more than one connected component in the relay graph. Each of these components may select and multicast different leaders in the Relay phase. Having each Relay phase member broadcast its component's selected leader to the entire group using a probabilistic multicast mechanism (such as Bimodal Multicast [3]) would give us a high probability that this inconsistency is detected by some group member (which need not have participated in the Relay phase). If a member detects an inconsistency such as two leaders elected in the same round, it immediately sends out a multicast to the entire group re-initiating the next election round. If no member detects any such inconsistency, the election protocol round would satisfy the Uniqueness and Agreement conditions of Section 2.2 if and only if there was exactly one component in the Relay phase, this component selected exactly one leader, every other non-faulty group member received at least one of the multicast messages specifying this selected leader, and this elected leader did not fail during the election round.

To reduce the probability of many group members sending out a (re-)initiating multicast message at the same time, we could have each member M_i calculate the hash (using H) of its own id concatenated with the message identifier of one of the resulting messages, and send out a re-initiating multicast only if this is lower than K/N_i. This would again give an expected constant number of re-initiating multicasts. Alternatively, we could use a randomized delay before sending the request: if a process receives a re-initiation request, it need not send one of its own.

3.2 General Protocol Strategy

Our complete election protocol strategy is to use the election round described in the previous section as a building block in constructing a protocol with several rounds. A complete protocol strategy specifies 1) the value of K to be used (by each member) in the first round of each election, 2) the value of K to be used in round number $l+1$ as a function of the value of K used in round l, and 3) a maximum number of rounds after which the protocol is aborted. Note that this strategy is deterministic and known to all members, and is not decided dynamically.

As we will see in Section 4, the initial value of K can be calculated from the required success probability, view probabilities, process and message delivery failure probabilities for the network in which the group members are based, and the total maximum number of group members. Unfortunately, in practice, failure probabilities may vary over time. Since a higher value of K leads to a higher probability of success in a round (Section 4), we conclude that round $l+1$ must use a higher value of K than round l. One possible strategy is to use twice the value of K in round l, for round $l+1$. Other strategies are possible, and make our general protocol strategy adaptive to varying network failure rates and process view probabilities. Notice also that a lower value of the maximum number of rounds in the protocol implies fewer messages but a higher value results in a better probability of correct termination.

3.3 Implementation

Member M_i::Election ($Sequence, RoundNum$):

1. On receiving "Init election" message I specifying
   ($Sequence, RoundNum$),
   select K from $RoundNum$ using strategy
   if $H(M_i A_I) \times N_i < K$ go to step 2
   else
   wait for timeout period $Time\_Out\_1$ (time for step 2 to complete) and jump to step 3

2. Find the set of members $\{M_j\}_i$ in my view such that
   $H(M_j A_I) \times N_i < K$
   find best preferred leader in my view and send this using
   `ucast` messages to members in $\{M_j\}_i$
   do until $Time\_Out\_2$
   receive similar preferred leader messages for this
   ($Sequence, RoundNum$) from other members M_k
   include M_k in $\{M_j\}_i$ and M_i's view
   compare current best leader choice with M_k's preference
   (using choice function)
   if M_k's preference better
   update current best leader choice and send `ucast`
   messages to all members in $\{M_j\}_i$ specifying this
   else
   inform M_k using a `ucast` of M_i's current best choice
   multicast current best leader choice to entire group
   wait $Time\_Out\_3$ to receive everyone's final leader choice.

3. if received none or more than one leader as final choice
   choose one of the final choice messages F
   if $H(M_i A_F) \times N_i < K$, multicast an initiating message I'
   specifying ($Sequence, RoundNum + 1$)
   else
   declare received choice as elected leader and include it in
   M_i's view.

Figure 3: Complete Election Protocol

Figure 3 contains the pseudo-code for the steps executed by a group member M_i during a complete election protocol. Each distinct election is specified by a unique sequence number $SequenceNum$, and $RoundNum$ refers to the current round number in this election. $M_i :: Election(SequenceNum, 1)$ is called by M_i at the start of every new leader election (on receipt of the initiating message for that election).

One practical matter we have not mentioned yet is the use of time-outs by members to detect (or, rather, estimate) completion of a particular part of the protocol round in an asynchronous network.[1] Three $Time\_Out\_*$ values appear in the pseudocode. $Time\_Out\_2$ is the expected time for termination of the Relay phase (before the final multicasts at the end). This is just the worst case propagation and processing delay needed for a message containing a relay member's initial preferred leader to reach all other relay members (if it is not lost by a process or link failure). Although the number of relay members is not known a priori, we show in Section 4 that with known high probability, the number of relay members who do not fail until the end of the Relay phase is at most (3K/2). Thus $Time\_Out\_2$ can be taken to be the product of $(3K/2)$ (the maximum length of any path in a relay graph with $3K/2$ members) and the maximum propagation and processing delay for a `ucast` packet in the underlying network. $Time\_Out\_3$ is just the worst case time for delivery of a `mcast` message. In the Bimodal multicast protocol [3], this would be the maximum time a message is buffered anywhere in the group. $Time\_Out\_1$ is the sum of the maximum time needed at member M_i to calculate the set $\{M_j\}_i$, and the values of $Time\_Out\_2$ and $Time\_Out\_3$.

The above protocol presentation imposes no restriction on the hash function used. However, our analysis of Section 4 assumes a fair hash function, that is, one which distributes its outputs uniformly in $[0, 1]$. For a particular hash function (*e.g.*, the one described in [19]), we would need to know its distribution function and plug it into a similar analysis.

4 Analysis - Properties of the Protocol

In this section, we analyze the probability of success of a round of our protocol. We also analyze the detection on incorrect termination and message and time complexity of a round.

Let N be the number of group members at the start of the election round - we will assume that this value is approximately known to all group members so that the filter value calculation is consistent across members. Let $view\_prob$ be the probability of any member M_i having any other member M_k in its view throughout the election round. Let p_{fail}, p_{ucastl}, p_{mcastl} be the failure probabilities of a

[1] Although an asynchronous network model does not admit real time, in practice timers are readily available, and we do not assume any synchronization nor much in the way of accuracy in measuring intervals.

process during an election round, a `ucast` message delivery, and a `mcast` message delivery, respectively. The protocol round analyzed uses the parameter K as in Figure 2. We denote the terms $((1-p_{mcastl})\cdot K)$ and $((1-p_{fail})\cdot(1-p_{mcastl})\cdot K)$ as K_1 and K_2 respectively.

For simplicity, we assume that the probabilities of deliveries of a `mcast` message at different receivers are independent, as well as that the $Time\_Out\_*$ values in the protocol of Figure 2 are large enough. Our analysis can be modified to omit the latter assumption by estimating the $Time\_Out\_*$ values from K and the worst-case propagation and processing delays for `ucast` and `mcast` messages (as described in Section 3.3), and redefining $p_{ucastl}(p_{mcastl})$ to be the failure probabilities of a `ucast` (`mcast`) message delivery within the corresponding worst-case delays, as well as calculating $view\_prob$, p_{fail} for the round length. We also assume that the hash function H used is fair, that is, it distributes its outputs uniformly in the interval $[0, 1]$.

Round Success Probabilities

Consider the following events in an election round with parameter K:

E1: between $K_1/2$ and $3K_1/2$ members are chosen to participate in the Relay phase, and between $K_2/2$ and $3K_2/2$ relay members do not fail before sending out the final multicast;
E2: the set of relay members who do not fail before their final multicasts form a connected component in the relay graph throughout the Relay phase;
E3: at the end of the Relay phase, each non-faulty relay member has selected the same leader;
E4: by the end of the election round, each group member either fails or receives at least one of the final multicast messages (specifying the selected leader) from each component in the relay graph at the end of the Relay phase;
E5: the elected leader does not fail.

Theorem 1: The event [E3, E4, E5] in an election round in the protocol of Figure 2 implies that it is successful, that is, the election satisfies the Uniqueness and Agreement properties of Section 2.2. □

Thus, the probability of success in an election round with parameter K, denoted as $\mathsf{P}^{\mathsf{Success}}(K, H, view\_prob, p_{mcastl}, p_{ucastl}, p_{fail}, N)$, is

$$\begin{aligned} &\geq \mathtt{Pr}[\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4},\mathsf{E5}] \\ &= \mathtt{Pr}[\mathsf{E1}]\cdot\mathtt{Pr}[\mathsf{E2}|\mathsf{E1}]\cdot\mathtt{Pr}[\mathsf{E3}|\mathsf{E1},\mathsf{E2}] \\ &\quad\cdot\mathtt{Pr}[\mathsf{E4}|\mathsf{E1},\mathsf{E2},\mathsf{E3}]\cdot\mathtt{Pr}[\mathsf{E5}|\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4}]. \end{aligned}$$

We will lower-bound each of these five terms. Pr[E1] can be bounded by assuming that N, the group size at the start of the round, is large enough ($\gg K$) so that, from the fairness of H, the number of members entering the Relay phase can be approximated by a normal distribution with mean $(1-p_{mcastl})\cdot K = K_1$ and variance $K_1(1-K_1/N)\simeq K_1$ [20]. The $(1-p_{mcastl})$ term comes from the expected fraction of members receiving the initial multicast message. Similarly, the number of relay members that do not fail until their final multicast is a normal distribution with mean $(1-p_{fail})\cdot(1-p_{mcastl})\cdot K = K_2$ and variance $K_2(1-K_2/N)\simeq K_2$. Thus [20],

$$\begin{aligned} \mathtt{Pr}[\mathsf{E1}] &\geq 1 - 2\sqrt{\frac{2}{\pi K_1}}e^{-K_1/8} - 2\sqrt{\frac{2}{\pi K_2}}e^{-K_2/8} \\ &= \text{(say) } \mathtt{Pr_{lb}}[\mathsf{E1}]. \end{aligned}$$

For a random graph with m members and edge probability $view\_prob$, [4] states that the probability of the graph having exactly one component is $\mathcal{P}_m \simeq e^{-m(1-view\_prob)^{m-1}}$, and this rises with m. Since E1 implies that there are at least $K_2/2$ relay members at the end of the Relay phase, we have

$$\mathtt{Pr}[\mathsf{E2}|\mathsf{E1}] \geq \mathcal{P}_{K_2/2} = \text{(say) } \mathtt{Pr_{lb}}[\mathsf{E2}|\mathsf{E1}].$$

Given E1 and E2, E3 is satisfied if the Relay phase member that chooses the "best" leader (according to the choice function) among all the initial preferred leaders in the Relay phase, does not fail before its final multicast, and this message is delivered to all other relay members that do not fail before their final multicasts (since this will ensure that all relay members select the same leader in their final multicasts of the round). The conditional probability of the latter is calculated as follows: given E1, at most $3K_2/2$ members send out final multicasts, and given [E1, E2], at least one spanning tree (with $(3K_2/2-1)$ edges defined by the view relationship) can be imposed on the relay graph at the end of the Relay phase. The probability that the message containing the "best" leader is delivered along all edges of that spanning tree is $(1-p_{ucastl})^{3K_2/2-1}$. Hence,

$$\begin{aligned}\texttt{Pr}[\mathsf{E3}|\mathsf{E1},\mathsf{E2}] &\geq (1-p_{fail})\cdot(1-p_{ucastl})^{3K_2/2-1}\\ &= \text{(say) } \texttt{Pr}_{\texttt{lb}}[\mathsf{E3}|\mathsf{E1},\mathsf{E2}].\end{aligned}$$

$\texttt{Pr}[\mathsf{E4}|\mathsf{E1},\mathsf{E2},\mathsf{E3}]$ is bounded below by $\texttt{Pr}[\mathsf{E4}|\mathsf{E1},\mathsf{E2},\mathsf{E3}$ and $K_2/2$ members send out final multicasts]

$$\begin{aligned}&= (p_{fail}+(1-p_{fail})(1-p_{mcastl}^{K_2/2}))^N\\ &= \text{(say) } \texttt{Pr}_{\texttt{lb}}[\mathsf{E4}|\mathsf{E1},\mathsf{E2},\mathsf{E3}].\end{aligned}$$

Finally, we have $\texttt{Pr}[\mathsf{E5}|\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4}] = 1-p_{fail}$.

Thus, the success probability in an election round $\mathsf{P}^{\mathsf{Success}}(K,H,view\_prob,p_{mcastl},p_{ucastl},p_{fail},N)$ is bounded below by

$$\begin{aligned}&\mathsf{P}_{\texttt{lb}}^{\mathsf{Success}}(K,H,view\_prob,p_{mcastl},p_{ucastl},p_{fail},N)\\ = &\quad \texttt{Pr}_{\texttt{lb}}[\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4},\mathsf{E5}]\\ = &\quad \texttt{Pr}[\mathsf{E1}]\cdot\texttt{Pr}_{\texttt{lb}}[\mathsf{E2}|\mathsf{E1}]\cdot\texttt{Pr}_{\texttt{lb}}[\mathsf{E3}|\mathsf{E1},\mathsf{E2}]\\ &\quad \cdot\texttt{Pr}_{\texttt{lb}}[\mathsf{E4}|\mathsf{E1},\mathsf{E2},\mathsf{E3}]\cdot\texttt{Pr}[\mathsf{E5}|\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4}]\end{aligned}$$

Figure 4 shows the typical variation of the first four product terms and $\texttt{Pr}_{\texttt{lb}}[\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4},\mathsf{E5}]$, with K. The plots are shown for values of K up to 65 with $(view\_prob,p_{mcastl},p_{ucastl},p_{fail},N) = (0.4,0.01,0.01,0.01,10000)$. The quick convergence of $\texttt{Pr}_{\texttt{lb}}[\mathsf{E1}]$ and $\texttt{Pr}_{\texttt{lb}}[\mathsf{E2}|\mathsf{E1}]$ to unity at small K (here $\simeq$ 40) is independent of the value of N. In fact, $\texttt{Pr}_{\texttt{lb}}[\mathsf{E4}|\mathsf{E1},\mathsf{E2},\mathsf{E3}]$ is the only one among the five factors of $\texttt{Pr}_{\texttt{lb}}[\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4},\mathsf{E5}]$ that seems to depend on N. However, its value remains close to unity for $N\cdot p_{mcastl}^{3K_2/2} \ll 1$, or $N \ll p_{mcastl}^{-3K_2/2}$, which, for $K=40$, turns out as 10^{107}, a number beyond the size of most practical process groups. Thus, for all practical values of initial group size N, the minimum probability that an election round of the protocol of Figure 2 satisfies the Uniqueness and Agreement conditions is dependent only on the failure and view probabilities in the group, but is independent of N.

However, this minimal success probability in an election round peaks at 0.6 for the above parameters. This is because our estimate for $\texttt{Pr}_{\texttt{lb}}[\mathsf{E3}|\mathsf{E1},\mathsf{E2}]$ is very pessimistic in assuming a weak global view knowledge, and thus including the possibility that all the initially preferred leaders in the Relay phase might be distinct. In a practical setting however, a fair number of the m (non-faulty) relay members would have the same initial leader choices (eg., if the choice function preferred a candidate leader with lower identity), so the probability $\texttt{Pr}_{\texttt{lb}}[\mathsf{E3}|\mathsf{E1},\mathsf{E2}]$ (and hence $\texttt{Pr}_{\texttt{lb}}[\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4},\mathsf{E5}]$) would be much higher than the curve shows. The simulation results in Section 5 confirm this for the choice function mentioned above.

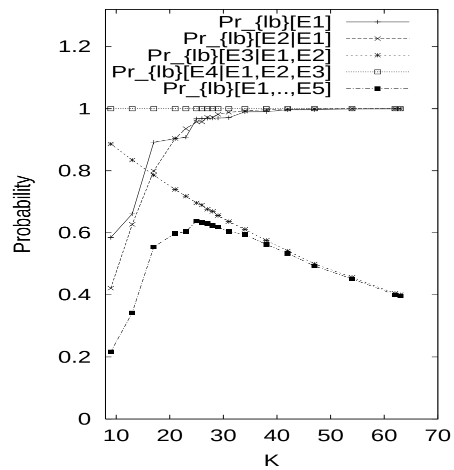

Figure 4: Pessimistic Analysis

Detection of Incorrect Termination in a Round

An election round may elect none or more than one leader with probability

$$1-\mathsf{P}^{\mathsf{Success}}(K,H,view\_prob,p_{mcastl},p_{ucastl},p_{fail},N) \leq 1-\texttt{Pr}_{\texttt{lb}}[\mathsf{E1},\mathsf{E2},\mathsf{E3},\mathsf{E4},\mathsf{E5}].$$

If no leader is elected in a round, no final multicast will be sent out and every non-faulty member will time out and detect incorrect termination. If more than one leader is elected in an election round with m non-faulty relay members (i.e., relay members that do not crash before their final multicasts), the probability that the incorrect termination is detected is greater than $\texttt{Pr}$[at least one member does not fail and receives all m multicasts] $= 1-(1-(1-p_{fail})(1-p_{mcastl})^m)^N$. Therefore,

Theorem 2: $\texttt{Pr}$[a re-initiating `mcast` is sent out to the group or all group members fail by the end of the round | election round with parameter K does not succeed], denoted as

$\mathtt{P}^{\mathtt{Detection}}(K, H, view\_prob, p_{mcastl}, p_{ucastl}, p_{fail}, N)$, is bounded below by

$$\mathtt{P}_{\mathtt{lb}}^{\mathtt{Detection}}(K, H, view\_prob, p_{mcastl}, p_{ucastl}, p_{fail}, N)$$
$$= \mathtt{Pr}_{\mathtt{lb}}[\mathtt{E1}] \cdot (1 - (1 - (1 - p_{fail})(1 - p_{mcastl})^{3K_2/2})^N).$$

With K fixed so that the term $\mathtt{Pr}_{\mathtt{lb}}[\mathtt{E1}]$ is arbitrarily close to unity, the probability of detection of incorrect election in a round of the presented protocol goes to unity as N tends to infinity. □

Round Message Complexity

The total number of ucast messages involved in an election round with m members entering the Relay phase is at most $m^3/2$ (since there are at most $m^2/2$ edges in a relay graph with m members, and m initial preferences for leaders). The total number of mcast messages is 1 (initiating message) + # messages in the final multicast phase (m) + # re-initiation messages on detection of inconsistent election. Thus,

Theorem 3: With (high) probability $\mathtt{Pr}_{\mathtt{lb}}[\mathtt{E1}]$ ($(\mathtt{Pr}_{\mathtt{lb}}[\mathtt{E1}])^2$), the number of ucast (mcast) messages in an election round is $O(K^3)$ ($O(K)$) (since K_1, K_2 are both $O(K)$). Thus, with (high) probability $(\mathtt{Pr}_{\mathtt{lb}}[\mathtt{E1}])^2$, the number of simultaneous multicasts in the network anytime during the round is $O(K)$. □

Theorem 4: Further, the *expected* number of ucast (mcast) messages in a round of the protocol is $O(K^3)$ ($O(K)$). (The proof follows from a simple binomial summation and is omitted.) This is $O(1)$ when K is fixed independent of N. The suggested election protocol round thus achieves the optimal message complexity for any global agreement protocol on a group of size N. □

Round Time Complexity

The time complexity of an election round with m members entering the relay phase is = the total message delivery and processing time of the ucast / mcast messages + computation time = $O(m) \times O(\mathcal{N}) + O(N) = O(\mathcal{N}m + N)$. The first $O(\mathcal{N})$ term comes from the delivery time of a ucast/mcast message by the underlying network with $\mathcal{N}$ nodes. The second $O(N)$ term is due to the dominating step where each member M_i has to calculate the set $\{M_j\}_i$ of relay members in its view. Thus,

Theorem 5: With (high) probability $(\mathtt{Pr}_{\mathtt{lb}}[\mathtt{E1}])^2$, the time complexity of an election round is $O(\mathcal{N}K + N)$. This is $O(\mathcal{N} + N)$ for K fixed independent of N, which is the optimal time complexity of for any global agreement protocol for a group of size N over a network with $\mathcal{N}$ nodes. □

5 Simulation Results

In this section, we analyze, through simulation, the performance of an election protocol strategy from the class described in Section 3.2. The correctness, scalability and fault tolerance of the proposed protocol are more evident here than from the limited analysis of Section 4. The strategy we analyze is specified by 1) an initial (first round) parameter $K_{init} = 7$; 2) for $l \leq 4$, the value of K in round l is twice the value used in round $l-1$; and at $l = 5$, $K = N$; and finally 3) the election protocol aborts after 5 rounds. The protocol is initiated by one mcast to the group, which initially has N members.

The unreliability of the underlying network and process group mechanism is characterized by the following parameters and assumptions:

1. p_{ucastl} : ucast message loss probability.
2. p_{mcastl} : mcast message loss probability.
3. p_{fail} : probability of a member failing in one protocol round. Process failures are uniformly and randomly distributed throughout the round.
4. $view\_prob$: probability that a member has another member in its view.
5. The choice function used in the simulation is the simple one that prefers candidates with lower identities.
6. The hash function is assumed to be a fair one.

The metrics used to measure the performance of the protocol are the following. *P(Success)* evaluates the final success probability of the protocol, and appears in two forms. "Strong" success probability refers to the (average) probability that a protocol run satisfies the Uniqueness and Agreement conditions. "Weak" success probability is in fact the (average) majority fraction of the non-faulty group members that agree on one leader at the end of the protocol. This is a useful metric for situations where electing more than one leader may be allowed, such as [22]. *# Rounds* refers to the average number of rounds after which the protocol terminates, either successfully, or without detecting an inconsistent election, or because the maximum number of rounds specified by the strategy has been reached. *# Messages* refers to the average number of `ucast` and `mcast` messages generated in the network during the protocol.

Figure 5 shows the results from the simulations. This figure is organized with each column of graphs indicating the variation of a particular performance metric as a function of each of the system parameters, and each row of graphs showing the effect of varying a system parameter on each of the performance metrics. Each point on these plots is the average of results obtained from 1000 runs of the protocol with the specified parameters. In Figures 5(a-c), $p_{ucastl} = p_{mcastl}$ is varied in the range $[0, 0.5]$ for fixed $N = 2000, p_{fail} = 0.001, view\_prob = 0.5$. The graphs for varying p_{fail} are very similar and not included here. In Figures 5(d-f), N is varied in the range $[1000, 5000]$ for fixed $p_{fail} = 0.001, view\_prob = 0.5, p_{ucastl} = p_{mcastl} = 0.001$. In Figures 5(g-i), $view\_prob$ is varied in the range $[0.2, 0.5]$ for $N = 5000, p_{fail} = 0.001, p_{ucastl} = p_{mcastl} = 0.001$.

Figures 5(a,d,g) show the very high success probability (strong) guaranteed by the above strategy even in the face of high message loss rates (up to $p_{ucastl} = p_{mcastl} = 0.4$, up to and beyond $N = 6000$ and $view\_prob = 0.2$). Notice that even the "weak" success ratio is close to 1 for these ranges, and as expected, is higher than the strong success probability. Figures 5(b,e,h) show the time scalability of the protocol for the same ranges of parameters that produced high success probabilities. Note Figure 5(e), which shows termination within 1 expected round for values of N up to 6000 (!) group members. Figures 5(c,f,i) show the message scalability for the same variation of parameters. Note again the lack of variation in the expected number of messages exchanged (Figure 5(f)) as N is varied up to 6000 members.

Figures 5(a-c) display the level of *fault tolerance* the protocol possesses with respect to message failures. Figures 5(d-f) show how much our protocol *scales* even as the number of group members is increased into the thousands. Finally, Figures 5(g-i) show that our protocol performs well even in the presence of only *partial membership* information at each member.

6 Conclusions

This paper described a novel leader election protocol that is scalable, but provides only a probabilistic guarantee on correct termination. Mathematical analysis and simulation results show that the protocol gives very good probabilities of correct termination, in the classical sense of the specification of leader election, even as the group size is increased into the tens of thousands. The protocol also (probabilistically) guarantees a low and almost constant message complexity independent of this group size. Finally, all these guarantees are offered in the face of process and link failure probabilities in the underlying network, and with only a weak membership view requirement.

The trade-off among the above guarantees is determined by one crucial protocol parameter–the value of K in an election round. From the simulation results, it is clear that choosing K to be a small number suffices to provide acceptable guarantees for the specified parameters. Increasing the value of K would enable the protocol to tolerate higher failure probabilities, but would increase its message complexity. Varying K thus yields a trade-off between increasing the fault tolerance and correctness probability guarantee on one hand and lowering the message complexity on the other.

References

[1] B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems", *Proc. 19th Symp. on Theory of Computing*, 1987, pp. 230-240.

[2] J. Beauquier, M. Gradinariu, C. Johnen, "Memory space requirements for self-stabilizing leader election protocols", *Proc. 18th Annual ACM Symp. Principles of Distributed Computing*, 1999, pp. 199-208.

[3] K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, Y. Minsky, "Bimodal multicast", *ACM Trans. on Computer Systems*, vol. 17, no. 2, May 1999, pp. 41-88.

[4] B. Bollobas, A. Thomason, "Random graphs of small order", *Annals of Discrete Mathematics, Random Graphs '83*, vol. 8, 1983, pp. 47-97.

[5] J. Brunekreef, J.-P. Katoen, R. Koymans, S. Mauw, “Design and analysis of dynamic leader election protocols in broadcast networks”, *Distributed Computing*, vol. 9, no. 4, Mar 1997, pp. 157-171

[6] T.D. Chandra, S. Toueg, “Unreliable failure detectors for asynchronous systems”, *Proc. 10th Annual ACM Symp. Principles of Distributed Computing*, 1991, pp. 325-340.

[7] I. Cidon, O. Mokryn, “Propagation and leader election in a multihop broadcast environment”, *Proc. 12th International Symp. Distributed Computing 1998 (DISC 98)*, pp. 104-118.

[8] B. Chor, C. Dwork, “Randomization in Byzantine agreement”, *Advances in Computing Research*, vol. 5, 1989, pp. 443-498.

[9] J. Cooper, N. Linial, “Fast perfect-information leader-election protocol with linear immunity”, *Proc. 25th Annual ACM Symp. on Theory of Computing*, 1993, pp. 662-671.

[10] D. Dolev, C. Dwork, L. Stockmeyer, “On the minimal synchronism needed for distributed consensus”, *JACM*, vol. 34, no. 1, Jan 1987, pp. 77-97.

[11] C. Fetzer, F. Cristian, “A highly available local leader election service”, *IEEE Trans. Software Engineering*, vol. 25, no. 5, Sep-Oct 1999, pp. 603-618.

[12] M.J. Fischer, N.A. Lynch, M.S. Paterson, “Impossibility of distributed consensus with one faulty process”, *Journ. ACM*, vol. 32, no. 2, Apr 1985, pp. 374-382.

[13] R. Gallager, P. Humblet, P. Spira, “A distributed algorithm for minimum weight spanning trees”, *ACM Trans. on Programming Languages nd Systems*, vol. 4, no. 1, Jan 1983, pp. 66-77.

[14] A. Itai, “On the computational power needed to elect a leader”, *Lecture Notes in Computer Science*, vol. 486, 1991, pp. 29-40.

[15] J.L. Kim, G.G. Belford, “A distributed election protocol for unreliable networks”, *Journal of Parallel and Distributed Computing*, vol. 35, no. 1, May 1996, pp 35-42.

[16] C.-T. King, T.B. Gendreau, L.M. Ni, “Reliable election in broadcast networks”, *Journal of Parallel and Distributed Computing*, vol. 7, 1989, pp. 521-540.

[17] C. Malloth, A. Schiper, “View synchronous communication in large scale networks”, *Proc. 2nd Open Workshop of the ESPRIT project BROADCAST (#6360)*, Jul 1995.

[18] R. Ostrovsky, S. Rajagopalan, U. Vazirani, “Simple and efficient leader election in the full information model”, *Proc. 26th Annual ACM Symp. Theory of Computing*, 1994, pp. 234-242.

[19] O. Ozkasap, R. van Renesse, K.P. Birman, Z. Xiao, “Efficient buffering in reliable multicast protocols”, *Proc. 1st Intnl. Workshop on Networked Group Communication*, Nov. 1999, Lecture Notes in Computer Science 1736.

[20] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill International Edition, 3^{rd} edition, 1991.

[21] D. Peleg, “Time optimal leader election in general networks”, *Journ. Parallel and Distributed Computing*, vol. 8, no. 1, Jan, 1990, pp. 96-99.

[22] R. De Prisco, B. Lampson, N. Lynch, “Revisiting the Paxos algorithm”, *Proc. Of WDAG 97, Lecture Notes in Computer Science*, vol. 1320, pp. 111-125

[23] M.O. Rabin, “Randomized Byzantine generals”, *Proc. 24th Annual Symp. Foundations of Computer Science*, Nov. 1983, pp. 403-409.

[24] L.S. Sabel, K. Marzullo, “Election vs. consensus in asynchronous systems”, Computer Science Technical Report ncstrl.cornell/TR95-1488, Cornell University, 1995.

[25] G. Singh, “Leader election in the presence of link failures”, *Proc. 13th Annual ACM Symp. Principles of Distributed Computing*, 1994, page. 375.

[26] G. Singh, “Leader election in complete networks”, *SIAM Journ. Computing*, vol. 26, no. 3, Jun 1997, pp. 772-785.

[27] S. Singh, J.F. Kurose, “Electing good leaders”, *Journ. of Parallel and Distributed Computing*, vol. 21, no. 2, May 1994, pp. 184-201.

[28] G. Taubenfeld, “Leader election in the presence of n-1 initial failures”, *Information Processing Letters*, vol. 33, no. 1, Oct 1989, pp. 25-28.

[29] S. Toueg, “Randomized Byzantine agreements”, *Proc. 3rd Annual ACM Symp. Principles of Distributed Computing*, 1984, pp. 163-178.

[30] R. van Renesse, Y. Minsky, M. Hayden, “A gossip-style failure detection service”, *Proc. of Middleware '98*, IFIP, Sept 1998, pp. 55-70.

[31] M. Yamashita, T. Kameda, ”Computing on anonymous networks. 1. Characterizing the solvable cases”, *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 1, Jan. 1996, pp. 69-89.

[32] D. Zuckerman, “Randomness-optimal sampling, extractors, and constructive leader election”, *Proc. 28th annual ACM Symp. Theory of Computing*, 1996, pp. 286-295.

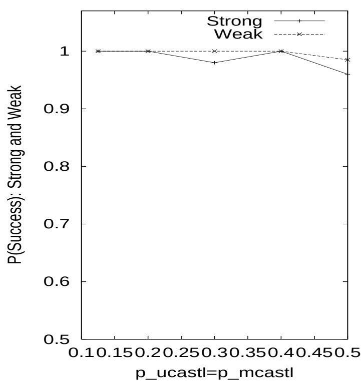

Fig (a). P(Success) vs message loss probability.

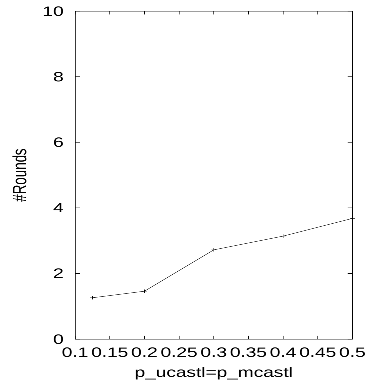

Fig (b). Avg. #Rounds vs message loss probability.

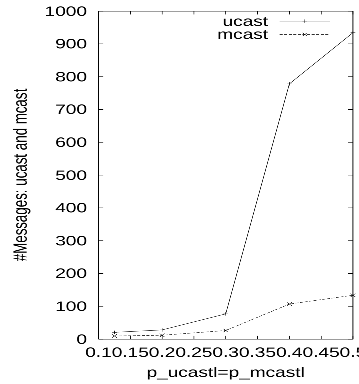

Fig (c). Avg. #Messages vs message loss probability.

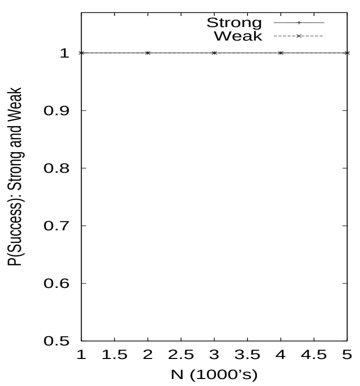

Fig (d). P(Success) vs # Group members.

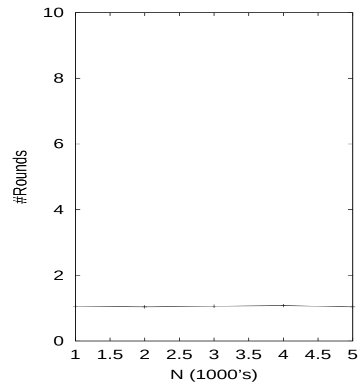

Fig (e). Avg. #Rounds vs # Group members.

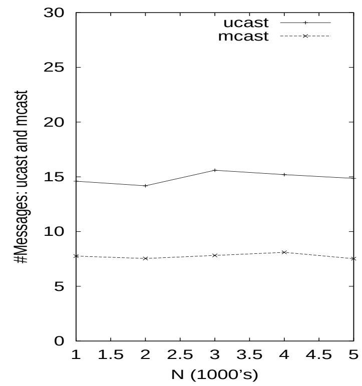

Fig (f). Avg. #Messages vs # Group members.

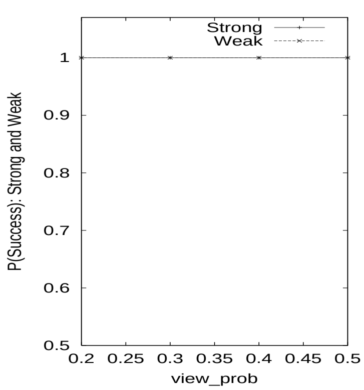

Fig (g). P(Success) vs view probability.

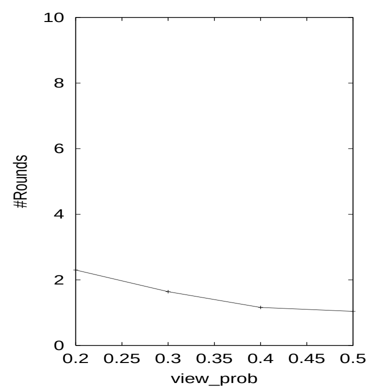

Fig (h). Avg. #Rounds vs view probability.

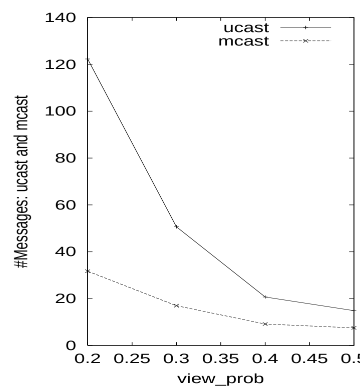

Fig (i). Avg. #Messages vs view probability.

Figure 5: Simulation Results