Distributed Council Election

Danny Raz Yuval Shavitt Lixia Zhang

Abstract— **This paper studies the problem of electing a small number of representatives (council) out of a (possible large) group of anonymous candidates. The problem arises in scenarios such as multicast where, to avoid feedback implosion, a small subset of the receivers is chosen to provide feedback on network conditions.**

We present several algorithms for this problem and analyze the expected number of messages and rounds required for their convergence. In particular, we present an algorithm that almost always converges in one round using a small number of messages (for typical council size) when the number of hosts is known. In the case where the number of hosts is unknown (and too large to be polled), our algorithms converge in a small number of rounds that improves previous results by Bolot et al [BTW94].

I. INTRODUCTION

In many distributed applications there is a need to distributively elect a small group of hosts out of a potentially large population. The elected group needs to be maintained under dynamic network conditions that include new members joining and leaving and network temporary partition.

A typical application for this problem appears in multicast protocols. Due to the feedback implosion problem, there is a need to elect a small group of representatives out of the (possibly) large set of receivers (multicast group members). An election mechanism was proposed by Bolot et al. [BTW94] to elect a small number of receivers and collect from them feedback regarding loss rate and congestion in the multicast group. Other applications that require distributed group-election are electing hot backup servers, distributed databases, and distributed network management.

A particular example where such algorithms are implemented is the IDMaps project [JJJ$^+$00]. As part of this project, measurement stations (Tracers) are installed in various locations in the Internet. These tracers measure the distance among themselves and to other areas in the Internet. The measurement results are then sent to (potentially many) distance information servers by multicast. To reduce measurement overhead, distance information servers provide feedback for the usefulness of each measurement to the Tracer. To avoid the feedback implosion problem at Tracers there is a need to select one or a few representatives out of the server population.

In both the multicast congestion control and the IDMaps example, communication from some data transmitting entity to a group of hosts is done using multicast while the reverse direction is done using unicast transmissions. In both examples the population size may vary over several orders of magnitude, the population may change over time, and the transmitting entity has no initial knowledge of the population size. In the above examples, choosing a single representative is usually undesirable both for redundancy and better statistical representation (in the multicast example). Thus, our algorithms are capable of electing a group of any size in a predefined range, $\mathcal{R} = [L..U]$. Note that if the transmitting entity knows the ids of all the receivers it can (deterministically) choose the representatives, but we cannot assume this due to the feedback implosion problem.

We model this environment using a synchronous distributed election process where the hosts do not communicate among themselves but through a central entity that sends a global feedback message to the entire population. Two measures of interest in this environment are the expected number of rounds for election, T, and the expected number of unicast messages needed, N.

We present in this work several distributed randomized algorithms and analyze their performance. The algorithms explore the tradeoff between the two measures above and the state maintained in the hosts. We first present a basic algorithm that requires each host to maintain a single bit state. This algorithm converges in a small constant number of rounds regardless of n, the initial population size, and in sub-linear (in n) number of messages (for typical $\mathcal{R}$s). If the hosts can maintain the state from the previous round, we present an improved algorithm that reduces T by 10-15% (for typical $\mathcal{R}$s) and dramatically reduces N to a constant factor of T, thus omitting N dependency on n. We then present an algorithm that further reduces the expected number of rounds, T, to be close to 1 (e.g., when $\mathcal{R} = [1..8]$ we achieve $T = 1.001$ regardless of n). However, this last algorithm requires almost double the number of messages of the previous algorithm (for the above $\mathcal{R} = [1..8]$ this is still only $N \approx 8$ regardless of n).

The above algorithms assume that n is either known or alternatively can be polled in a single round and n messages. We show the robustness of the suggested algorithms to an inaccurate estimation of n. This leads to an efficient algorithm for the case where n is unknown and maybe too big to be polled. The algorithm searches for n in a way similar to the one suggested by Bolot et al. [BTW94] but achieves superior performance.

II. Related Work

Our problem is similar to the collision resolution problem in multiple access (MA) networks. In this type of network hosts try to transmit packets via a shared medium. If multiple transmissions overlap, usually the packet collides and the targeted receiver cannot decode the signal. The problem of collision resolution in MA networks had been extensively studied during the 1970s and the 1980s, and several sub-models that differ in the feedback the hosts receive from the channel were analyzed. Here we emphasize only the most relevant ones, and refer the interested reader to a book by Rom and Sidi [RS90].

Most of the work in MA networks concentrated on two feedback models. The binary model assumes hosts knows whether a collision occurred or not. The ternary feedback model assumes hosts are aware of three channel conditions: silence, no packets transmission; success, one packet successfully transmitted; and collision, where more than one packet was transmitted and none was successfully received by the receiver. The host receives no feedback about the number of the packets that collided.

Some works dealt with a model with richer feedback from the channel. Tsybakov [Tsy80] examined a model where the number of colliding hosts is known. Georgiadis and Papantoni-Kazakos [GPK82] studied a channel where the number of colliding hosts is known up to some bound. Pippenger [Pip81] proved that the capacity of a collision channel with full multiplicity feedback is 1, Ruszinkó and Vanroose [RV97] gave an algorithm that achieves this bound.

A few works examined a model of a channel with multiple reception capability, where up to $k-1$ simultaneous transmissions may be decoded. Tsybakov et al. [TML83] assumed that the channel has a ternary feedback. The hosts know if the previous slot was idle; if it has some successful transmissions, up to $k-1$; or if a collision of k or more packets occurred. Likhanov et al. [LPS+93] examined a similar model but assumed that in the case of successful transmissions the hosts are aware of the exact number of successful transmissions.

It is important to note that even in the case where the model in the MA literature is the same as the one used here, the objective is different and thus the optimization problem is different. This makes a solution that works well for one of the problems perform badly in the other. For example, the model where full multiplicity feedback is given matches our model for host feedback. Thus, the results by [Pip81], [RV97] might seem to suggest that as n grows there exists an algorithm that elects a group (at least in the case where $\mathcal{R} = [1..1]$) with an expected number of rounds that approaches one. However, an examination of the algorithm structure in [Pip81], [RV97] reveals an initialization phase which takes $n/\log n$ rounds whose cost is amortized over the transmission of n packets. Such a solution will not fit our goal since we are only interested in the first successful transmission (in MA jargon).

Other (somewhat less) related line of work is the distributed leader election. Many variants to this problem were studied [Lyn97]. The closest model to our problem is the case of anonymous cliques, i.e., network with full connectivity but no unique identifiers for the processors. For this a Monte-Carlo solution was given by Afek and Matias [AM94], and we are not aware of a Las-Vegas solution which better fits our model. Note, that in our model there is a central entity that does not exist in the distributed leader election literature.

Feedback suppression in multicast was studied extensively in the recent years. The main two solution approaches suggested are timer based and structural based (cf. [NB98]). One exception is the work by Bolot et al. [BTW94] that suggested a mechanism for feedback control in a multicast video distribution. In their solution each host is assigned a random 16 bit vector. The sender polls the receivers by sending a bit mask to which only receivers with a matching vector reply. By polling with masks of decreasing size the sender can ensure a small number of feedback messages, with high probability, but the polling may take up to 15 rounds when the number of receivers is small. Our solution for the unknown n case is similar in nature, but as demonstrated in section VI-A can be tuned to achieve faster convergence and any specific target range of feedback messages.

III. Model

We have n hosts and wish to elect a subgroup in the size range $\mathcal{R} = [L, U]$. For the election, the hosts communicate only with a central trusted entity, $\mathcal{C}$; no direct communication among the hosts is allowed. Communication is done is synchronous rounds. In each such a round, $\mathcal{C}$ broadcasts a feedback message to all hosts and each host may send a reply message. We distinguish between solutions based on the amount of information in the feedback message, $\mathcal{F}$.

The following parameters are of interest. First we would like to minimize the expected number of rounds needed for the election process. Another important parameter we would like to minimize is the election overhead, which is the number of messages sent by the hosts through the en-

```
basic algorithm
Init: s = 0

N_c ← received feedback
if N_c < L
    send reply
    s ← 1
else if s = 1
    if N_c ≤ U
        send reply
    else
        if (rnd() < c/N_c)
            send reply
        else
            s ← 0
```

Fig. 1. The basic algorithm for the host

```
basic algorithm
N_c ← 0
while (N_c ∉ [L..U])
    Send Feedback(N_c)
    N_c ← number of replies
```

Fig. 2. The basic algorithm for $\mathcal{C}$

tire election process, excluding the first n messages.

IV. Algorithms with no history

We first study a simple algorithm that requires each host to maintain only one bit of state, s (see Figure 1). $s = 1$ indicates that the host participates in the current election process. At each round (see Figure 2), $\mathcal{C}$ sends the number of reply messages it received in the previous round, N_c, i.e., $\mathcal{F}$ size is $\log n$ bits. A host decides to reply (and suggest itself as a representative) with probability p defined below.

$$p = \begin{cases} 1 & \text{if } N_c < L \\ 1 & \text{if } N_c \leq U \text{ and } s = 1 \\ c/N_c & \text{if } N_c > U \text{ and } s = 1 \end{cases} \tag{1}$$

where c is a pre-configured constant. The state bit s is set when the feedback from $\mathcal{C}$ is less than L, and reset when the host drops its candidacy (with probability $1 - c/N_c$). The election process starts when $\mathcal{C}$ sends the value 0 as feedback, and it ends when the number of active hosts is in the desired range. Figures 1 and 2 give a pseudo-code of the algorithm for the host and $\mathcal{C}$, ($rnd()$ returns a random number uniformly distributed in [0..1]).

The value of the constant c is pre-determined. Next we will evaluate the algorithm performance and in particular show how to calculate the optimal value of c. For this end, let us first define $Pr\{i|m\}$ to be the probability that out the m hosts that were active (have $s = 1$) in the current round exactly i hosts decide to remain active (not reset s)

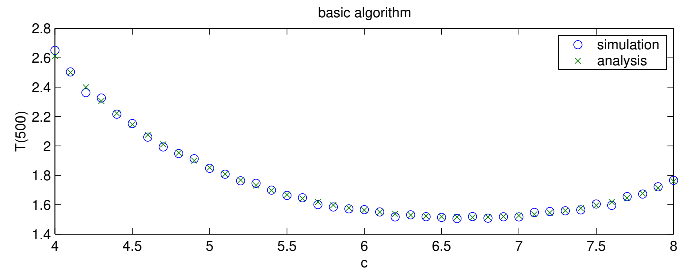

Fig. 3. A comparison of the simulation results to the analytical results for $T(500)$, $L = 4, U = 8$, and $4 \leq c \leq 8$

in the next round.

$$Pr\{i|m\} = \binom{m}{i} \left(\frac{c}{m}\right)^i \left(1 - \frac{c}{m}\right)^{m-i} \tag{2}$$

The expected number of messages, $N(n)$, is given by the following relation. In the formulation we omit the cost of the first initialization round for reasons that will become clear in section VI-A.

$$N_n(m) = \begin{cases} \sum_{i=0}^{m} Pr\{i|m\}(i + N_n(i)) & U < m \leq n \\ 0 & L \leq m \leq U \\ m + N_n(n) & m < L \end{cases} \tag{3}$$

In the first line, we sum over all the possibilities that i hosts stayed active after this round. We pay i messages for this round and $N_n(i)$ for the rest of the election. The relation for $L \leq m \leq U$ reflects the fact that the election ends when the active host number reaches its target range. The last relation is due to the fact that when we undershoot we restart the algorithm.

Equation 3 defines a linear system of $n + 1$ equations with the $n + 1$ variables $N_n(m)$ $m = 0, 1, \ldots n$. The equation can be solved with c as a free parameter to minimize the message overhead. Note that in the calculation for a certain n we obtain $N(n) = N_n(n)$ but the rest of the values $N_n(i), i < n$ cannot be used to obtain N values for $i < n$.

A similar system can be defined if one wishes to minimize the number of rounds it takes the system to elect a number of representatives in the desired target range. In this case, we pay a unit price per round.

$$T_n(m) = \begin{cases} 1 + \sum_{i=0}^{m} Pr\{i|m\}T_n(i) & U < m \leq n \\ 0 & L \leq m \leq U \\ 1 + T_n(n) & m < L \end{cases} \tag{4}$$

Although the analysis can be easily used to produce numerical results, we found out that running high confidence simulations is much faster for large n values. Thus, most of the graphs presented in the paper are driven by high

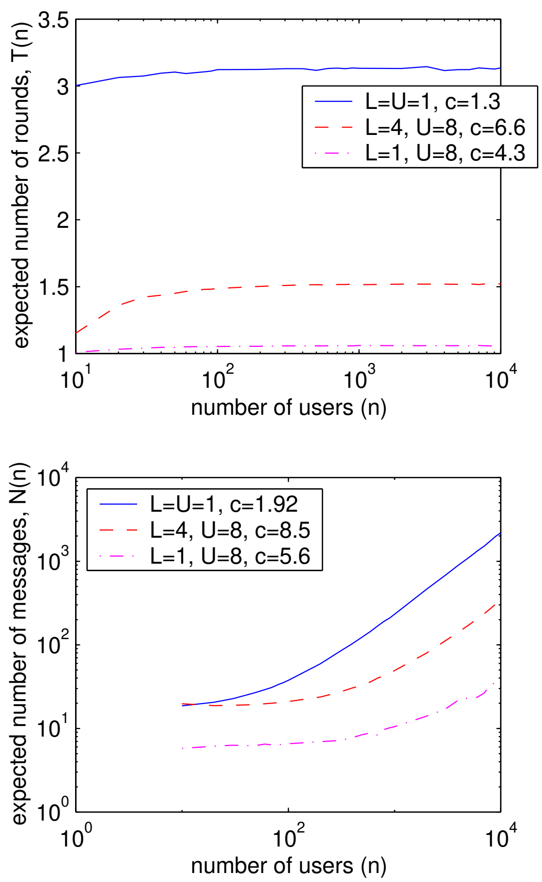

Fig. 4. The basic algorithm: $T(n)$ and $N(n)$ as a function of n

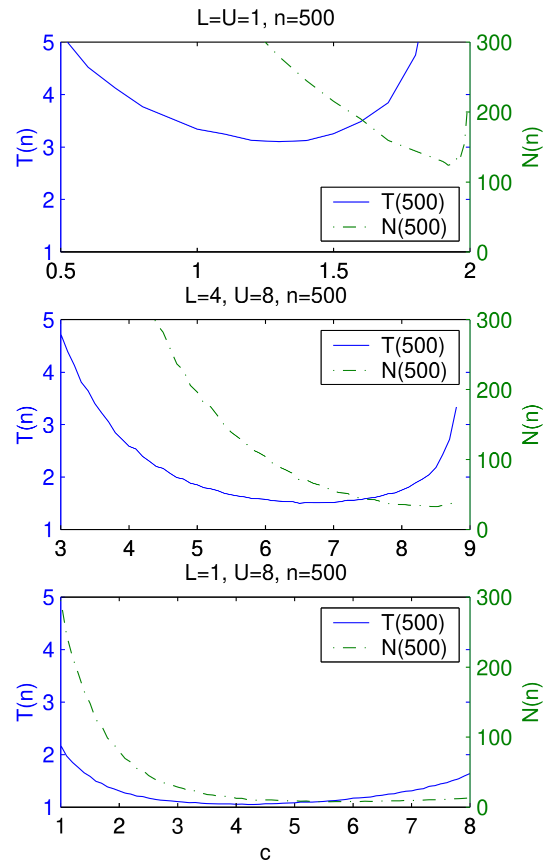

Fig. 5. The basic algorithm: $T(n)$ and $N(n)$ as a function of c for $n = 500$

confidence simulations where each point is the average of 10,000 runs and in all cases the 1% confidence interval is less than 1%. Figure 3 compares results achieved from simulations and analysis for $T(500)$ where $L = 4, U = 8$ and $3 \leq c \leq 8$. A simulation point is derived from 10,000 different runs. All the simulation points fall very close to the analysis results. The simulation can be used both to predict with high accuracy the value of the function, and be used for the selection of the optimal c.

Figure 4 depicts the values of $T(n)$ and $N(n)$ for several target ranges. For each graph we picked c to be a value closer to the optimum. The most obvious fact from these graphs is the fast converges of the algorithm, and the little dependency of $T(n)$ on n. Astonishingly, even for very large values of n the expected number of rounds (after the initialization round) is less than 2. The number of messages needed for the algorithm is also well below $n/10$ for a large enough n, and grows at a sub-linear rate.

Figures 5 show T and N dependency on c for $n = 500$. The dependency is fairly stable for a wide range of n values. Both functions are fairly flat around the optimum but become quite steep farther away.

Next we present a simple observation that can help in selecting c which is close to the optimal. The shape of $Pr\{i|m\}$ has a mode at $i = c$ and it rapidly drops on both sides. Thus, by selecting $L < c < U$ one can expect that most likely an 'undershoot' of the range will be at $i = L - 1$, and an 'overshoot' at $i = U + 1$. For T the penalty of undershoot is two rounds while the penalty of overshoot is one round ($T < 2$ suggests that in most cases no more than one additional round is needed), thus selecting c at the 2/3 point between L and U, i.e., $c = L + 2(U-L)/3$ should most likely lay in the flat region around the optimum. Similarly for N the penalty for overshoot is $U + 1$ and for undershoot is $L + n$, and thus the optimal c can be approximated by $U - (U - L)(U + 1)/(L + n)$.

A. *Improved algorithm (Skip-Reset)*

Although the basic algorithm performs well, there is still room for improvement. For this, we use the observation that when the feedback is less than L there is a wasted round in which all the hosts suggest themselves as candidates. The only purpose of this round is for the hosts to learn the value of n, but this value can be advertised by $\mathcal{C}$. This saves a round and more significantly the n messages

```
Algorithm Skip-Reset
Init: s = 0

(N_c, r) ← received feedback
if r = 1
   s ← 1
if N_c < L
   send reply
   s ← 1
else if s = 1
   if N_c ≤ U
      send reply
   else
      if rnd() < c/N_c
         send reply
      else
         s ← 0
```

Fig. 6. Algorithm Skip-Reset for the host

```
Algorithm Skip-Reset
N_c ← 0
N ← 0
while (N_c ∉ [L..U])
   if N_c < L
      Send Feedback(N, 1)
   else
      Send Feedback(N_c, 0)
   N_c ← number of replies
   N ← max{N, N_c}
```

Fig. 7. Algorithm Skip-Reset for $\mathcal{C}$

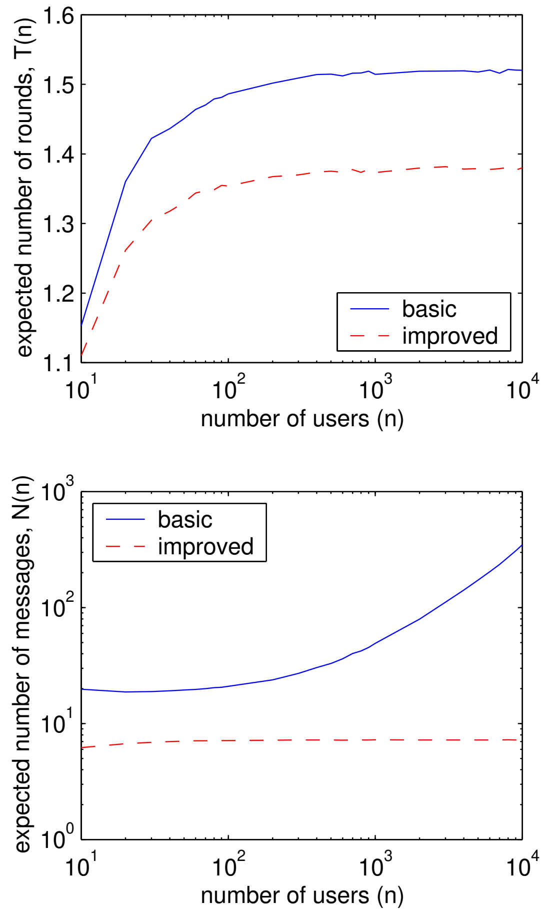

Fig. 8. The improved algorithm vs. the basic algorithm: $T(n)$ and $N(n)$ for $L = 4, U = 8$, and optimal c values

that are transmitted in this round[1].

A pseudo-code of the improved algorithm, called Skip-Reset, is given in figures 6 and 7. As can be seen, the addition to the host algorithm is only the ability to receive a reset bit in the feedback message. The algorithm of $\mathcal{C}$ needs to store in memory the population size (in variable N), and use it when a reset is required.

The change in the formulation for this enhancement is simple. When $m < L$ we can avoid payment for the skipped round (which is m for N and 1 for T). Thus, Equation 3 and Equation 4 should be written as

$$N_n(m) = \begin{cases} \sum_{i=0}^{m} Pr\{i|m\}(i + N_n(i)) & U < m \leq n \\ 0 & L \leq m \leq U \\ N_n(n) & m < L \end{cases} \tag{5}$$

$$T_n(m) = \begin{cases} 1 + \sum_{i=0}^{m} Pr\{i|m\}T_n(i) & U < m \leq n \\ 0 & L \leq m \leq U \\ T_n(n) & m < L \end{cases} \tag{6}$$

Figure 8 shows the improvement in the algorithm performance for $L = 4$ and $U = 8$. While the improvement in $T(n)$ is modest (a fifth of a round or about 13%) the improvement in $N(n)$ is dramatic. The almost linear dependency of N in n is eliminated and we get $N(n) \approx 9.2$ for $n > 100$. For other parameter selection similar improvement is achieved.

B. *The power of choice*

We've seen above that in most cases the election is done in one round. To increase the chances of succeeding in the first round one can use the following technique. At each round, the host draws two coins with probability p, and sends an election message containing both drawings if at least in one of the draws it remains active. This requires two more bits to be sent with the election message. When $\mathcal{C}$ receives the election messages from the hosts it can choose to use either of the two rounds of drawing, whichever optimizes its operation[2]. In particular, this al-

[1] Our observation is similar in spirit to the one made by Massey for the binary tree collision resolution protocols [RS90, Section 5.2.1]

[2] This technique is similar in spirit to the one used by Azar et al. for the balanced allocation of balls into bins [ABKU99].

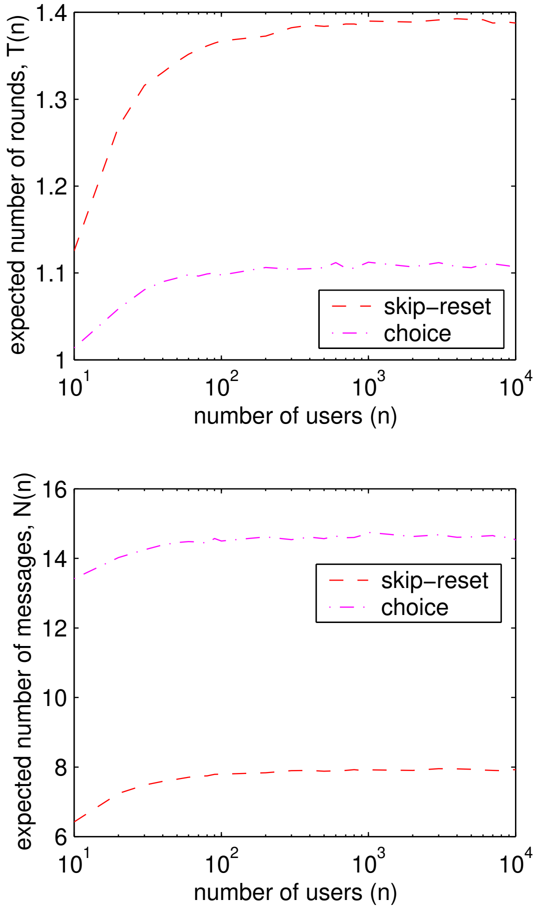

Fig. 9. Algorithm Skip-Reset vs. the choice algorithm: $T(n)$ and $N(n)$ for $L = 4, U = 8$, and $c = 5.6$

gorithm will take one round if either of the two drawings is within the target range. For example, if the probability of success in one round is $1/3$ then by using the free choice of two rounds the probability to succeed in one round increases to $1 - (1 - 1/3)^2 = 5/9$.

In the feedback $\mathcal{C}$ sends, it must include a bit indicating which of the two election rounds have been used. This requires the hosts to maintain additional two bits as part of their state. Figure 9 shows the gain from the algorithm when combined with the previous improvement of Skip-Reset. As one can see the number of rounds for the appropriate choice of c is almost always 1. However, there is a penalty in an increase in the number of messages (almost a factor of two). Note however that if we use this improvement with the basic algorithm it will improve both T and N.

V. Algorithms with History

In this section we present an algorithm that requires the hosts to remember their state in the previous round. The motivation for this type of algorithm is that once less than

Algorithm Skip-Reset with history

```
Init: s = 0, s_h = 1

(N_c, r) ← received feedback
if r = 1 AND s_h = 1
   s ← 1
if N_c < L
   send reply
   s ← 1
else if s = 1
   if N_c ≤ U
      send reply
   else
      if rnd() < c/N_c
         send reply
      else
         s ← 0
else if s_h = 1
   s_h ← 0
```

Fig. 10. Algorithm Skip-Reset with history for the host

Algorithm Skip-Reset with history

```
N_c ← 0
N ← 0
while (N_c ∉ [L..U])
   if N_c < L
      Send Feedback(N, 1)
   else
      Send Feedback(N_c, 0)
      N ← N_c
   N_c ← number of replies
```

Fig. 11. Algorithm Skip-Reset with history for $\mathcal{C}$

L hosts are left active it is better to return to the pervious round which may have less active participants than the initial number of active hosts. This should potentially reduce the number of messages of Algorithm Skip-Reset without adding to the number of rounds.

Figures 10 and 11 show a pseudo-code for Algorithm Skip-Reset with one round history. The host maintains its previous state in the variable s_h and returns to an active mode if it receives a reset notification *and* was active in the previous round. $\mathcal{C}$ maintains the smallest number (that is still larger than L) of active hosts it learned, and sends it with a reset upon receiving less than L replies.

Equations 3 and 4 seem to change only slightly to

$$N(m) = \begin{cases} \sum_{i=L}^{m} Pr\{i|m\}(i + N(i)) & \\ \quad + \sum_{i=0}^{L-1} Pr\{i|m\}(i + N(m)) & U < m \leq n \\ 0 & L \leq m \leq U \end{cases} \tag{7}$$

$$T(m) = \begin{cases} 1 + \sum_{i=U+1}^{m} Pr\{i|m\}T(i) & \\ \quad + \sum_{i=0}^{L-1} Pr\{i|m\}T(m) & U < m \leq n \\ 0 & L \leq m \leq U \end{cases} \tag{8}$$

However, the change is more substantial. By introducing history of the last state, the algorithm never rolls back, and the equation becomes oblivious to the initial host population. The main advantage of this is the ability to solve the equations iteratively, and obtain in $O(n^2)$ all the $N(i)$ values for $L \le i \le n$. The same holds for T.

The iterative form of equations 7 and 8 (for $i > U$) is

$$N(m) = \frac{\sum_{i=U+1}^{m-1} Pr\{i|m\}N(i) + \sum_{i=0}^{m} Pr\{i|m\}i}{1 - Pr\{m|m\} - \sum_{i=0}^{L-1} Pr\{i|m\}} = \frac{c + \sum_{i=U+1}^{m-1} Pr\{i|m\}N(i)}{1 - Pr\{m|m\} - \sum_{i=0}^{L-1} Pr\{i|m\}} \quad (9)$$

$$T(m) = \frac{1 + \sum_{i=U+1}^{m-1} Pr\{i|m\}T(i)}{1 - Pr\{m|m\} - \sum_{i=0}^{L-1} Pr\{i|m\}} \quad (10)$$

Note that at *every* round during the election the expected number of messages $\mathcal{C}$ receives is c regardless of the number of active hosts. This means that $N(m) = c \cdot T(m)$ for all $L \le m$ (these functions are not defined for $m < L$). Of course this does not mean that for each election the number of messages sent is c times the number of election rounds. This fact can be easily proved by induction. For $m \le U$ $N(m) = T(m) = 0$. Assuming the hypothesis holds for some $m - 1$ we have

$$\frac{N(m)}{T(m)} = \frac{c + \sum_{i=U+1}^{m-1} Pr\{i|m\}N(i)}{1 + \sum_{i=U+1}^{m-1} Pr\{i|m\}T(i)} = \frac{c + \sum_{i=U+1}^{m-1} Pr\{i|m\}cT(i)}{1 + \sum_{i=U+1}^{m-1} Pr\{i|m\}T(i)} = c$$

which proves the hypothesis.

It is clear, that $N(m) = cT(m)$ holds also for Algorithm Skip-Reset since there also at every round the number of messages is expected to be c. However, finding an algebraic proof for Algorithm Skip-Reset is still open (though the logical proof is suffice). For the basic algorithm $N(m) > cT(m)$ and the constant factor relation does not hold. The reason is that in some rounds, namely after the feedback message shows less than L active hosts, the number of active hosts is n with probability 1 (and not expected to be c).

Figure 12 depicts the improvement in Algorithm Skip-Reset when history is used. The gain in $T(n)$ is 5-6% while the gain in $N(n)$ is around a quarter of a percent. For reference, we also plot algorithm choice that improves T by roughly 40% but increases N by about 5%.

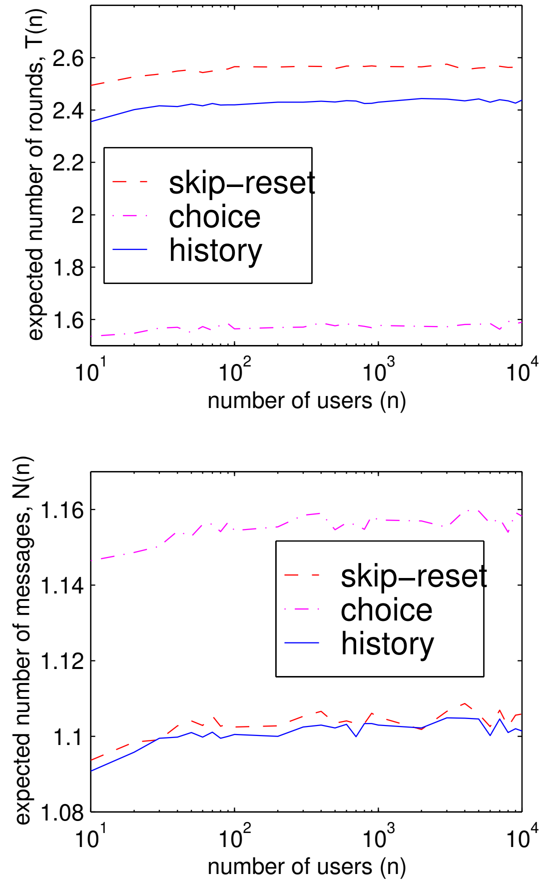

Fig. 12. A performance comparison of Algorithm Skip-Reset the choice algorithm and Algorithm Skip-Reset with history for 'leader election' scenario $L = 1, U = 8$, and optimal c values

VI. Robustness

In the previous sections we assumed that either n is known, or that $\mathcal{C}$ can poll all nodes for their number. In some applications, like multicast feedback, such polling may cause feedback implosion, in other applications the number of hosts may dynamically change as a result of hosts joining/leaving, or network partitioning. For such cases it is important to check the robustness of the algorithms presented in the previous section to an inaccurate estimation of n. Figure 13 presents the performance of Algorithm Skip-Reset when used with an estimation of the total number of hosts (n), instead of the real value. One can see that a factor of 10 mistakes in the value of n, does not increase $T(n)$ by more than one round, but it may generate 10 times more messages if the estimation is an undershot. Another fact that is clear from Figure 13 is that it always pays to overestimate. This is due to the fact that underestimation may cause to generate many messages since the nodes are using a much too big $p = c/n$, while overestimation will cause additional round(s) in which no message is transmitted since $p = c/n$ is too small.

A. Estimating n

The robustness of the algorithms with respect to n, indicates that there will be a very small degradation in per-

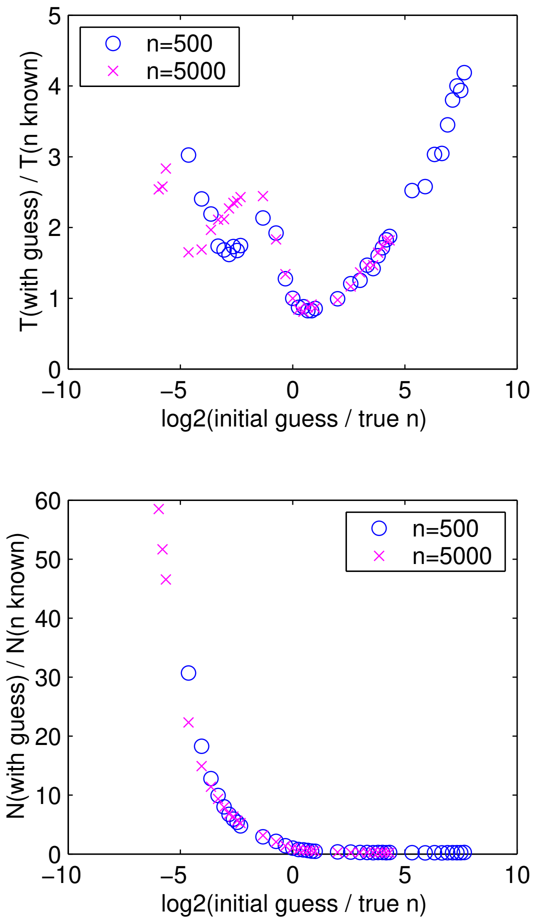

Fig. 13. Relative performance of Algorithm Skip-Reset under different estimations of n where $L = 1, U = 8$, and $c = 5.6$

formance in a scenario where a small number of nodes are either joining or leaving the system. However, in applications like multicast, the number of hosts may vary significantly, and it is important to keep the estimation always bigger than the actual n value to avoid feedback implosion.

We suggest a family of algorithms that starts with a very high estimation (say 100000 to compare with 2^{16} that is used in [BTW94]), and reduces it every time the number of responses is smaller than L. The rate at which the estimation is reduced is given by a function $\hat{n}(i+1) = f(\hat{n}(i))$. In this work we used a family of functions defined in the following way

$$f(\hat{n}) = \begin{cases} \hat{n}^{\alpha}/\beta & \hat{n} > (\beta U)^{1/\alpha} \\ 0 & \hat{n} \leq (\beta U)^{1/\alpha} \end{cases} \tag{11}$$

The second condition ensures that the estimation is not in the range $[1..U]$. When $\alpha = 1$ the estimated value of n decreases by a constant factor of $\beta > 1$. However, when $\alpha < 1$ the estimated value of n decreases much faster. The latter, has a shorter convergence time but may result in more messages.

Algorithm Skip-Reset with estimating n

$N_c \leftarrow 0$
$N \leftarrow MAXGUESS$
Send Feedback($MAXGUESS$, 1)
while ($N_c \notin [L..U]$)
  $N_c \leftarrow$ number of replies
  if $N_c < L$
    $N \leftarrow f(N)$
    Send Feedback(N, 1)
  else
    Send Feedback(N_c, 0)
  $N \leftarrow \max\{N, N_c\}$

Fig. 14. Algorithm Skip-Reset with estimating n for $\mathcal{C}$

Let $f(\hat{n})$ be the function used to decrease the estimate when the feedback is below L. Denote by $Pr_{\hat{n}}\{i|n\}$ the probability that exactly i out of n hosts will be elected in a round given an estimate of $\hat{n}$ hosts.

$$Pr_{\hat{n}}\{i|n\} = \binom{n}{i} \left(\frac{c}{\hat{n}}\right)^i \left(1 - \frac{c}{\hat{n}}\right)^{n-i} \tag{12}$$

We analyze the performance of the estimation algorithm assuming Algorithm Skip-Reset is used, and write expressions for the expected number of messages and rounds to elect a council given an initial estimation $\hat{n}_0$, and n hosts. Note that here (unlike in the previous analysis) we do count all rounds including the initialization round.

Following the previous formulation, let $\hat{N}_n(\hat{n}_0)$ and $\hat{T}_n(\hat{n}_0)$ be the expected number of messages sent and round (correspondingly) given n hosts and $\hat{n}_0$ initial guess. We can thus write the recursive relations (for $\hat{n} > U$):

$$\begin{aligned} \hat{N}_n(\hat{n}) &= \sum_{i=L}^{n} Pr_{\hat{n}}\{i|n\}(i + N_n(i)) \\ &\quad + \sum_{i=0}^{L-1} Pr_{\hat{n}}\{i|n\}(i + \hat{N}_n(f(\hat{n}))) \\ &= c\frac{n}{\hat{n}} + \sum_{i=L}^{n} Pr_{\hat{n}}\{i|n\} N_n(i) \\ &\quad + \sum_{i=0}^{L-1} Pr_{\hat{n}}\{i|n\} \hat{N}_n(f(\hat{n})) \end{aligned} \tag{13}$$

Where $N_n(i)$ are the ones calculated from the linear system in (5). If For $0 < \hat{n} \leq U$, $\hat{N}_n(\hat{n})$ is not defined since $f(\hat{n})$ does not return values in this range (see Equation 11). For $\hat{n} = 0$, $\hat{N}_n(0) = n + N(n)$ since each host will suggest itself as a candidate in the next round.

In the same way we can write equations for $\hat{T}_n$ (for $\hat{n} > U$):

$$\hat{T}_n(\hat{n}) = \sum_{i=L}^{n} Pr_{\hat{n}}\{i|n\}(1 + T_n(i))$$

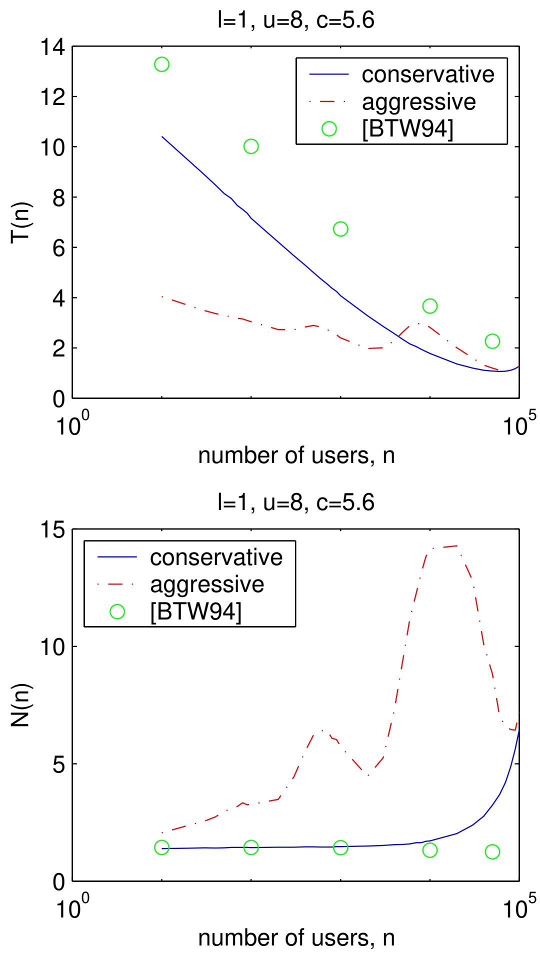

Fig. 15. Performance of algorithm Skip-Reset where n is unknown $L=1, U=8$, and $c=5.6$

$$\begin{aligned} & + \sum_{i=0}^{L-1} Pr_{\hat{n}}\{i|n\}(1+\hat{T}_n(f(\hat{n}))) \\ = \; & 1 + \sum_{i=L}^{n} Pr_{\hat{n}}\{i|n\}T_n(i) \\ & + \sum_{i=0}^{L-1} Pr_{\hat{n}}\{i|n\}\hat{T}_n(f(\hat{n})) \qquad (14) \end{aligned}$$

Figure 15 shows the performance of the algorithm in a typical scenario, where the target range is $\mathcal{R} = [1..8]$ i.e., we are willing to elect up to 8 members (compare to the 10 responders bound in [BTW94]). We selected two sets of parameters: a *conservative* set where $\alpha = 1$ and $\beta = 2$, and an *aggressive* set where $\alpha = 0.7$ and $\beta = 1$. We compare these two with the algorithm in [BTW94].[3] Even when the actual population is very small (10), our conservative search takes only about 10 rounds to elect representatives. Compare this with the over 13 rounds required by the [BTW94] algorithm. Both algorithms reduce their approximation by a factor of 2. Ours performs better since the probability of a host to become a candidate is $p = c/n$ and when the estimate reaches 60 we get $p \approx 1/10$. Thus, with high probability one representative will be elected when $\hat{n} \leq 60$. A much faster convergence is achieved using the aggressive parameter set. Every time there is no response the estimation is reduced to $\hat{n}^{0.7}$, and therefor four rounds are sufficient for all the n. The reduction in the number of rounds comes with a penalty of an increase in the expected number of messages. However, the maximum of 15 messages the aggressive algorithm achieves is well within the tolerable. The number of local maxima in the aggressive algorithm graphs corresponds to the maximal number of steps.

[3]Bolot et al. did not calculate the expected number of messages, instead they showed that the probability that this number is greater than 10 is small. We calculated the expected number of messages in their algorithm by $\sum_{j=1}^{i}\sum_{m=1}^{n} Pr(r_j = m)\cdot m$.

algorithm name	section number	host memory	$\mathcal{C}$ memory	feedback size (bits)
basic	IV	1 bit	–	$\log n$
Skip-Reset	IV-A	1 bit	$\log n$ bits	$1+\log n$
choice	IV-B	3 bits	$\log n$ bits	$2+\log n$
history	V	2 bits	$\log n$ bits	$1+\log n$

TABLE I

SUMMARY OF THE ALGORITHMS PRESENTED

VII. CONCLUDING REMARKS

Table I summarizes the algorithms presented in the paper. Note that all these algorithms can work continuously if we change the `while` condition in $\mathcal{C}$'s algorithm to `true`. This way, if the representative group size decreases over time below L a new election process automatically restarts. In this context, an interesting research direction is to design an efficient election algorithm that increases the group size when it is close to (but still above) L, thus avoiding the reactivation of the election.

REFERENCES

[ABKU99] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocation. *SIAM J. Computing*, 29:180 – 200, 1999.

[AM94] Yehuda Afek and Yossi Matias. Elections in anonymous networks. *Information and Computation Journal*, 113(2):312–330, September 1994.

[BTW94] J-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. In *ACM SIGCOMM 1994*, pages 58–67, London, UK, September 1994.

[GPK82] L. Georgiadis and P. Papantoni-Kazakos. A collision resolution protocol for random access channels with energy detectors. *IEEE Trans. on Communications*, 30(11):2413–2420, November 1982.

[JJJ+00] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, , and Lixia Zhang. On placement of internet instrumentation. In *IEEE INFOCOM 2000*, pages 295 – 304, Tel-Aviv, Israel, March 2000.

[LPS+93] N. Likhanov, E. Plotnik, Y. Shavitt, M. Sidi, and B. Tsybakov. Random access algorithms with multiple reception capability and N-ary feedback channel. *Problemy Peredachi Informatsii*, 29(1):82 – 91, 1993.

[Lyn97] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997.

[NB98] Jörg Nonnenmacher and Ernst W. Biersack. Optimal multicast feedback. In *IEEE INFOCOM '98*, San Francisco, CA, USA, March 1998.

[Pip81] N. Pippenger. Bounds on the performance of protocols for a multiple access broadcast channel. *IEEE Transactions on Information Theory*, 27(2):145–151, March 1981.

[RS90] Raphael Rom and Moshe Sidi. *Multiple Access Protocols: Performance and Analysis*. Springer-Verlag, 1990.

[RV97] Miklós Roszinkó and Peter Vanroose. How an Erdös-Rényi-type search approach gives an explicit code construction of rate 1 for random access with multiplicity feedback. *IEEE Transactions on Information Theory*, 43(1):368–373, January 1997.

[TML83] B.S. Tsybakov, V. A. Mikhailov, and N. Likhanov. Bounds for packet transmission rate in a random-multiple-access system. *Problemy Peredachi Informatsii*, 19(1):61–81, January–March 1983.

[Tsy80] B.S. Tsybakov. Resolution of a conflict of known multiplicity. *Problemy Peredachi Informatsii*, 16(2):134–144, April–June 1980.