Network Neighborhood Size Estimation Using Distributed Random Sampling

Rui Fan, Allen Miu, Eugene Shih

Massachusetts Institute of Technology
{rfan, aklmiu, eugene}@lcs.mit.edu

December 19, 2000

1 Introduction

Wireless sensor networks used for surveillance are typically deployed in environments where positions and locations of network entities cannot be controlled. Rather, nodes tend to be randomly scattered over some terrain where objects of interest may traverse. One main purpose of these sensor nodes is to provide information about interesting events that occur in the environment. For example, in the military, tiny sensor nodes can be used to monitor enemy soldier and tank maneuvers in an area of conflict. When a moving tank is detected by some array of sensors, some information (e.g. velocity) about the tank should be reported by the sensor to the basestation of an ally.

In such a network, nodes have limited energy resources due to their size and the low energy density of small batteries. Therefore, computation and communication at each node should not be used without carefully considering the energy costs. In addition, the communication protocol should be designed in such a way so that nodes do not wastefully communicate. Note that in such a system, both transmitting and receiving can be costly. Therefore, collisions should be minimized and constant transmission prohibited.

Clearly, a media-access control (MAC) protocol will be needed to allow the nodes to share the wireless media. A good MAC protocol will provide high throughput while maintaining an equal share of the available bandwidth among the nodes. However, designing a good MAC protocol without any information about the topology of a network can be difficult. One beneficial piece of information that can help the MAC designer is the neighborhood size. Thus, we introduce the network neighborhood size estimation (N-EST) problem. Given a set of N nodes randomly scattered over some terrain, we wish to determine the number of neighbors n for each node. A node u is considered a neighbor of v if u is within a communication range r of v. All nodes have the same r and it is not necessary for any node to know the number of neighbors for any other node.

In this paper, we will present a parallel fully polynomial randomized approximation scheme with $O(\log^2 N)$ running time to estimate the neighborhood size for every node in the network.

2 Problem Statement

As described, we wish to consider the problem of estimating the number of nodes reachable by each node in a wireless ad-hoc network. In such a network, we have several constraints:

- Every node has knowledge of the total number of nodes N at the time of deployment. However, no node has knowledge about their locations.
- Each node possesses a unique identifier i, where $1 \leq i \leq N$, but does not have knowledge about any other node in the network.
- Nodes will be deployed over some region, but the deployment of the nodes cannot be controlled. In other words, the physical distribution of nodes over the region is unknown.
- Nodes must communicate during time slots of fixed length. A node may broadcast and receive broadcasts simultaneously. In addition, we will assume that nodes have synchronized clocks.
- Each node can transmit a distance of r; thus, each node can communicate with nodes within a circle of radius r. However, suppose two nodes a, b are both within r of a third node c. If a and b transmit within the same time slot, the messages from a and b will *collide* and node c will receive a corrupted message.

As stated, the solution to the N-EST problem will give an estimate for the number of neighbors for each node in the network.

3 Related Work

Much research has been conducted in the area of random sampling, approximation algorithms, and parallel approximation algorithms. The use of random sampling for size estimation was used in [1]. In that paper, a size-estimation framework was introduced to estimate the size of the transitive closure. The author also outlines a technique that can be used to estimate the number of vertices with a distance d of a given vertex v. Unlike the N-EST problem, however, the algorithm is given $G = (V, E)$; the algorithm is given a complete description of the vertices and edges in the graph. In the N-EST problem, the set of edges is unknown.

Randomization techniques have also been used to estimate the multicast session size in a wide-area network [2]. Multicast is a method that is used to distribute information to a subset of users in a wide-area network. These users are collectively known as a multicast group. According to [2], estimating the size of a multicast session is essential for scalability. That is, in order to support extremely large multicast groups, multicast protocols require group size estimates to achieve scalability. The scheme proposed by the authors uses probabilitistic polling of users in the wide-area network to find the group size.

4 Solution Criteria

To evaluate the performance of the algorithms presented, we will consider the following criteria:

- *Accuracy.* Clearly, the optimal solution for each node is when every node exactly estimates its neighborhood size. However, any algorithm that can provide a probabilistic guarantee that all nodes have an estimate within ε of their actual neighborhood size is also useful.
- *Time to determine an approximation to n.* Another important criteria is the running time required to produce an estimate that is within ε of n.

The notion of a polynomial approximation scheme captures how well an algorithm satisfies these two criteria. A *polynomial approximation scheme* (PAS) for a counting problem Π is a deterministic algorithm $\mathcal{A}$ that takes an input instance I and a real number $\varepsilon > 0$, and in time polynomial in $n = \|I\|$ produces an output $A(I)$ such that

$$(1-\varepsilon)\#(I) \leq A(I) \leq (1+\varepsilon)\#(I)$$

where $\#(I)$ is the number of distinct solutions for an instance I of Π [5].

A *polynomial randomized approximation scheme* (PRAS) is a PAS that uses a randomized algorithm $\mathcal{A}$ instead of a deterministic one. Furthermore, a PRAS produces an output $A(I)$ such that

$$\Pr[(1-\varepsilon)\#(I) \leq A(I) \leq (1+\varepsilon)\#(I)] \geq \frac{3}{4}.$$

A *fully polynomial randomized approximation scheme* (FPRAS) is a polynomial randomized approximation scheme whose running time is polynomial in N and $1/\varepsilon$. Finally, an (ε, δ)-FPRAS for a counting problem Π is an FPRAS that takes an input instance I and computes a ε-approximation to $\#(I)$ with probability at least $1-\delta$ in time polynomial in N, $1/\varepsilon$, and $\log(1/\delta)$.

In an algorithm that solves the neighborhood size estimation problem, we would like to find an approximation for every node. A fully polynomial approximation algorithm (deterministic or randomized) to determine the neighborhood size for a particular node κ would therefore take as input $\kappa \in S$, ε, δ, and N, the number of nodes. Then, $A(\kappa, \varepsilon, \delta, N)$ should produce an estimate of the number of distinct neighbors for κ. An algorithm to determine the neighborhood size for every node would take as input the set of nodes, δ, and ε. It should output an N element vector[1], where the i'th entry represents an estimate for n_i where n_i is the number of neighbors for node i, $1 \leq i \leq N$. Since the nodes are autonomous, one can imagine that all nodes will estimate n_i in parallel. Therefore, a *parallel randomized approximation algorithm* is presented to find all n_i's simultaneously. In general, a good parallel algorithm should have a running time that is polylogarithmic in the input size, in this case, N.

To analyze the feasibility of the algorithm presented and to evaluate its performance, we have implemented a custom simulator.

Before presenting the parallel FPRAS, a deterministic algorithm will be introduced. Then, we will describe our algorithm and give a performance and running time analysis. A discussion of the simulator and simulation results will follow. Finally, we will conclude and offer some ideas for future research.

5 A Deterministic Algorithm

One straightforward way to allow every node i to obtain an exact estimate for n_i is to do the following. Each node broadcasts at the i'th time slot. Since each node has a unique identity, it is clear that no collisions will occur. Every node maintains a counter c_i that increments each time a message is received. After $\Theta(N)$ slots, we have $n_i = c_i$. In other words, c_i will be the exact number of nodes around node i.

6 A Randomized Sampling Estimation Algorithm

We present a parallel FPRAS which in $O(\log^2 N)$ time outputs a (ε, δ) approximation for the number of neighbors of every node. Intuitively, each node broadcasts a signal to all its neighbors

[1] In reality, each node obtains an estimate for n_i and no vector is output.

with some probability in every time slot. We assume that broadcasts are synchronized, and that nodes can broadcast and receive broadcasts simultaneously. A node receives an uncorrupted signal if only one of its neighbors broadcasts in a time slot. If more than one neighbor broadcasts, the node registers a collision. By counting the number of uncorrupted signals received over $O(\log N)$ time slots, constituting one epoch, a node can estimate its number of neighbors. Depending on its number of neighbors, some broadcast probabilities will ensure the node receives many clear signals in an epoch, and will achieve a good estimate for its number of neighbors. The algorithm runs for $O(\log N)$ epochs, in each of which a different broadcast probability is used. This ensures that every node will receive many clear signals in some epoch.

For $\delta, \varepsilon > 0$, define $\mu_{\varepsilon,\delta} \triangleq \frac{3}{\varepsilon^2} \log \frac{1}{\delta}$. Define $\beta \triangleq \lceil 395 \mu_{\varepsilon,\delta} \rceil$. Define $\rho(n,p) \triangleq np(1-p)^n$. For $0 < p < 1$, define $\gamma(z,p)$ to be a solution for n to $\rho(n,p) = z$. Note that for $z \neq (1-p)^{1/p}$, $\rho(n,p) = z$ has 2 solutions. We will describe which solution to choose later. Given a set S of numbers $s_1, ..., s_n$, define maxarg(S) to be the index of the maximum element in S. We summarize some of the definitions in the table below. The algorithm is presented following the table.

Variable	**Definition**
$\mu_{\varepsilon,\delta}$	$\frac{3}{\varepsilon^2} \log \frac{1}{\delta}$
β	$\lceil 395 \mu_{\delta,\varepsilon} \rceil$
$\rho(n,p)$	$np(1-p)^n$
$\gamma(z,p)$	A solution for n to $\rho(n,p) = z$.
maxarg(S)	The index of the maximum element of a set S.
N	The total number of nodes.
p_j	The broadcast probability used in epoch j.
λ_i^j	The number of clear signals received by node i in epoch j.
Λ_i	The set $\{\lambda_i^j\}$, for $1 \leq j \leq \lceil \log N \rceil + 2$.

Algorithm 1
Input: A set of nodes with unknown distribution. $\delta, \varepsilon > 0$.
Output: Every node obtains a (ε, δ) *approximation for its number of neighbors.*
1. Every node broadcasts.
2. If a node doesn't receive any broadcasts, it records 0 as its number of neighbors and stops.
3. Set $j \leftarrow 1$, $p_j \leftarrow \frac{1}{2}$.
4. Repeat the following $\lceil \log N \rceil + 2$ times:
   4.1 Every node i sets $c_i \leftarrow 0$.
   4.2 Repeat the following β times:
      4.2.1 Every node broadcasts with probability p_j.
      4.2.2 Every node i which receives an uncorrupted broadcast increments c_i.
   4.3 For every node i, set $\lambda_i^j \leftarrow c_i$, $j \leftarrow j + 1$.
   4.4 Set $p_j \leftarrow p_{j-1}/2$.
5. For every node i which didn't stop in step 1, let $\iota_i = \text{maxarg}(\Lambda_i)$.
6. Every node i which didn't stop in step 1 records $\gamma(\frac{\lambda_i^{\iota_i+1}}{\beta}, p_{\iota_i+1})$ as its number of neighbors.

7 Analysis

We first present a lemma which relates an estimate for ρ to an estimate for γ.
Lemma Let $0 \leq \nu \leq 0.1$, $0 \leq p \leq \frac{1}{2}$, and $\frac{1}{4p} \leq n \leq \frac{1}{2p}$. If $(1 - 0.263\nu)\rho(n,p) \leq z \leq (1 +$

$0.263\nu)\rho(n,p)$, then $(1-\nu)n \le \gamma(z) \le (1+\nu)n$.

Proof We consider the amount of relative change in ρ given a ν change in n, compared to ν. This quantity equals

$$\frac{1}{\nu}\frac{\rho((1+\nu)n,p)-\rho(n,p)}{\rho(n,p)} = \frac{(1+\nu)(1-p)^{\nu n}-1}{\nu} \triangleq \psi(n,p,\nu)$$

We seek to lower bound $|\psi(n,p,\nu)|$, for $0 \le p \le \frac{1}{2}$, $\frac{1}{4p} \le n \le \frac{1}{2p}$, and $-0.1 \le \nu \le 0.1$. We will see shortly that $\psi(n,p,\nu) > 0$ in this region, so it suffices to lower bound $\psi(n,p,\nu)$. We have $\frac{\partial\psi}{\partial n} = (1+\nu)\log(1-p)(1-p)^{\nu n} \le 0$, since $\log(1-p) \le 0$. Thus ψ is minimized for $n = \frac{1}{2p}$.

$$\psi(n,p,\nu) \ge \psi(\frac{1}{2p},p,\nu)$$

Also, $\frac{\partial\psi}{\partial p} = -(1+\nu)(1-p)^{\nu n-1}n \le 0$. Thus ψ is minimized $p = \frac{1}{2}$, $n = \frac{1}{2p} = 1$.

$$\psi(n,p,\nu) \ge \psi(1,\frac{1}{2},\nu) = \frac{(1+\nu)(\frac{1}{2})^{\nu}-1}{\nu} \triangleq \psi_2(\nu)$$

We can verify that for $-0.1 \le \nu \le 0.1$, ψ_2 is minimized for $\nu = 0.1$. Thus,

$$\psi(n,p,\nu) \ge \psi(1,\frac{1}{2},0.1) \ge 0.263$$

Thus, for every factor of ν change in n, ρ changes at least a factor of 0.263ν. Hence, if $(1-0.263\nu)\rho(n,p) \le z \le (1+0.263\nu)\rho(n,p)$, then $(1-\nu)n \le \gamma(z) \le (1+\nu)n$. □

Claim Algorithm 1 produces a (δ,ε) approximation for the number of neighbors of all nodes in $O(\log^2 N)$ time.

Proof Fix a node i, and let n be its number of neighbors. If $n = 0$, i will not receive any broadcasts in step 1, and will correctly record 0 as its number of neighbors. Assume then that $n > 0$, and let $q = \frac{1}{2n}$. Let epoch 0 refer to step 1. Then $p_0 = 1$, and $p_{\lceil \log N\rceil+2} \le \frac{1}{4N}$. Thus, there exists a j such that $p_j \le q \le p_{j-1}$, where $1 \le j \le \lceil N\rceil + 2$. Define $k = p_j n$. i receives a clear signal in each iteration of step 4.2 if only one of its neighbors broadcasts. This event occurs with probability $\rho(n,p_j) = np_j(1-p_j)^n = n\frac{k}{n}(1-\frac{k}{n})^n$. We have $(1-\frac{k}{n})^n \ge e^{-k}(1-\frac{k^2}{n})$. Since $\frac{1}{2}q \le p_j \le q$, then $\frac{1}{4} \le k \le \frac{1}{2}$. Thus $(1-\frac{k}{n})^n \ge e^{-k}(1-\frac{(1/2)^2}{1})$, and $k(1-\frac{k}{n})^n \ge \frac{3}{4}ke^{-k}$. Thus, the probability i receives a clear signal in each iteration is at least $\frac{3}{16}e^{-1/4}$. Define $\epsilon \triangleq 0.263\varepsilon$. The expected time for i to receive $\mu_{\epsilon,\delta} = \frac{3}{\epsilon^2}\log\frac{1}{\delta}$ clear signals is $\frac{16}{3}e^{1/4}\mu_{\epsilon,\delta}$. By the Markov inequality, the probability i does not receive $\mu_{\epsilon,\delta}$ clear signals in $\frac{32}{3}e^{1/4}\mu_{\epsilon,\delta}$ time is $\le \frac{1}{2}$, and the probability i does not receive $\mu_{\epsilon,\delta}$ clear signals in $\frac{64}{3}e^{1/4}\mu_{\epsilon,\delta}\log N$ time is $\le \frac{1}{N^2}$. Since there are N nodes, the probability any of them do not receive $\mu_{\epsilon,\delta}$ clear signals is at most $\frac{1}{N}$. If i receives $\mu_{\delta,\epsilon}$ clear signals, then it has a (ϵ,δ) estimate for $\rho(n,p_j)$. Then, since $\frac{\lambda_i^{\iota_i+1}}{\beta} \ge \mu_{\epsilon,\delta}$ with high probability, $\frac{\lambda_i^{\iota_i+1}}{\beta}$ is an (ϵ,δ) approximation for $\rho(n,p_j)$ with high probability. Since $\epsilon = 0.263\varepsilon$, then lemma 1 implies that i's estimate $\gamma(\frac{\lambda_i^{\iota_i+1}}{\beta},p_{\iota_i+1})$ is an ε approximation for n. Each iteration of step 4 takes at most $\frac{64}{3}e^{1/4}0.263^{-2}\mu_{\varepsilon,\delta}\log N \le 395\mu_{\varepsilon,\delta}\log N$ time. Thus, the total running time is at most $395\mu_{\varepsilon,\delta}\log^2 N = O(\log^2 N)$. Algorithm 1 is fully polynomial in N, $\log(1/\delta)$, and $1/\varepsilon$, and is in RNC. □

8 Additional Analyses

It might at first seem counterintuitive to use $\gamma(\frac{\lambda_i^{\iota_i+1}}{\beta}, p_{\iota_i+1})$ instead of $\gamma(\frac{\lambda_i^{\iota_i}}{\beta}, p_{\iota_i})$ to estimate n, where n is node i's number of neighbors. Indeed, since $\lambda_i^{\iota_i} \geq \lambda_i^{\iota_i+1}$, $\frac{\lambda_i^{\iota_i}}{\beta}$ is a better estimate for $np_{\iota_i}(1-p_{\iota_i})^n$ than $\frac{\lambda_i^{\iota_i+1}}{\beta}$ is for $np_{\iota_i+1}(1-p_{\iota_i+1})^n$. However, we are interested in estimating $\gamma(\rho, p)$. As we will see, $\gamma(\rho_{\iota_i}, p_{\iota_i})$ is a poor estimate for n, even if ρ_{ι_i} is well estimated. We see that ι_i corresponds in expectation to the epoch with the optimal broadcast probability for n. Thus in expectation, $\iota_i = j$, where $p_j \leq \frac{1}{n} \leq p_{j-1}$. Here we have $\frac{1}{2n} \leq p_j \leq \frac{1}{n}$. Restricting $0 \leq \nu \leq 0.1$, $0 \leq p \leq \frac{1}{2}$ as in the lemma and following a similar analysis, we have that $|\psi(n, p_j, \nu)|$ is minimized for $(n, p_j, \nu) = (\frac{1}{p_j}, p_j, 0)$. Hence,

$$\begin{aligned} |\psi(n,p_j,\nu)| &\geq \lim_{\nu\to 0} \frac{(1+\nu)(1-p_j)^{\nu n}-1}{\nu} \\ &= \lim_{\nu\to 0} \frac{(1-p_j)^{\nu/p_j}\left(p_j+(1+\nu)\log(1-p_j)\right)}{p_j} \\ &= \frac{p_j+\log(1-p_j)}{p_j} \end{aligned}$$

The second equality follows from L'Hôpital's rule. Thus, we have

$$\lim_{n\to\infty, p_j\to 0} |\psi(n,p_j,\nu)| = \lim_{p_j\to 0} \frac{p_j+\log(1-p_j)}{p_j} = \lim_{p\to 0} 1 - \frac{1}{1-p} = 0$$

The second equality again follows from L'Hôpital's rule. This shows that in the limit, any ε approximation for n leads to a perfect approximation for $np_j(1-p_j)^n$, or conversely, any ϵ approximation for $np_j(1-p_j)^n$ leads to an infinitely poor approximation $\gamma(np_j(1-p_j)^n, p_j)$ for n. For large n, we have $np_j(1-p_j)^n = ke^{-k}$, where $k = np_j$. Then, the previous result is intuitively clear by observing that the graph of ke^{-k} (shown in Figure 1) is flat for $k = 1$. This means that for $k = 1$, any change in ke^{-k} leads to an unbounded relative change in k. We can also verify that for small ϵ, an ϵ approximation for ke^{-k} leads to an approximation for k which is exponentially poor in ϵ. Hence, achieving an ε approximation for k when k is close to 1 takes exponential time in ε. However, by basing our estimate around epoch $\iota_i + 1$, we ensure with high probability that we have $\frac{1}{4} \leq k \leq \frac{1}{2}$. Since the slope of ke^{-k} is bounded from below by a positive constant in this region, we ensure that an ϵ estimate for $np_j(1-p_j)^n$ leads to a $c\epsilon$ estimate for n, for some bounded $c > 0$. Hence, estimating k in this region takes the same asymptotic time as estimating ke^{-k}. Note that we do not choose $\iota_i + s$ for $s > 1$, because this dramatically lowers the probability of a successful broadcast.

Another consideration is computing $\gamma(z, p)$. As mentioned previously, for $z \neq (1-p)^{1/p}$, $\rho(n, p) = z$ has 2 solutions. We always let $\gamma(z, p)$ be the smaller of the 2 solutions. To justify this, note that we only apply γ to values from epoch $\iota_i + 1$ (for any i). With high probability, we have $\frac{1}{4n} \leq p_{\iota_i+1} \leq \frac{1}{2n}$, so that $2n \leq \frac{1}{p_{\iota_i+1}} \leq 4n$. In general, one of the solutions of $\gamma(z, p_{\iota_i+1})$ is less than $\frac{1}{p_{\iota_i+1}}$, and the other solution is greater than $\frac{1}{p_{\iota_i+1}}$. Hence, the larger of the 2 solutions is greater than $2n$, and only the smaller of the 2 solutions can be a good estimate for n.

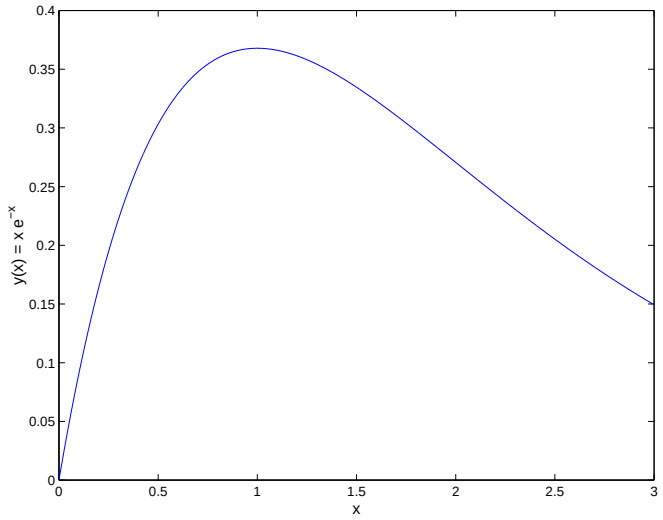

Figure 1: $f(x) = xe^{-x}$

9 The Simulator

9.1 Description of the Simulator

We wrote a Java-based simulator to help us understand and test the performance of our approximation algorithm. The simulator allows the user to specify various parameters. The key parameters are shown in Table 1. A detailed discussion of some of these parameters will be given later.

Table 1: Table of simulation parameters.

Parameter	**Meaning**
Distribution	Nodes can be physically distributed using the normal or uniform distribution
Maximum steps	The maximum number of simulation steps
μ	μ is specified instead of δ and ε
N	Number of nodes to simulate
p	Starting broadcast probability
radius	Fixed broadcast radius for each node
x	Width of simulation region
y	Height of simulation region

The simulator also implements a graphical user interface that displays the broadcast activities in each step and draws a bar graph reporting the current neighbor estimate generated by each node running Algorithm 1. Figure 2 shows a screen shot of the user interface. The top portion animates the broadcast activities as the simulation runs. Each dot represents a node and each circle represents a broadcast event. The radius is proportional to the transmission range. Each node receives a different color, depending on whether it lies in one circle (successful reception), overlapping circles (collision), or no circle (no reception). At the bottom is a bargraph. The height

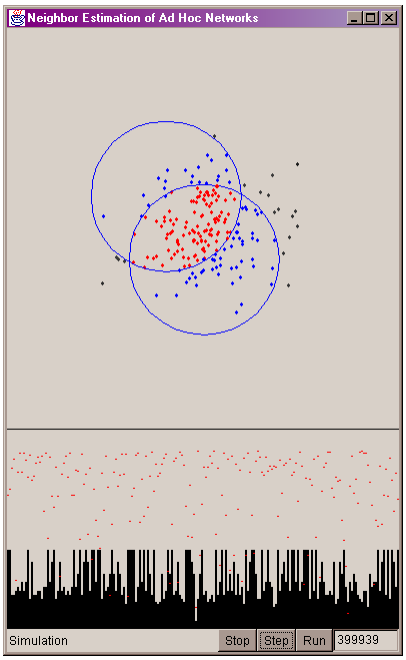

Figure 2: GUI for our simple Java networking simulator. Note that the nodes are physically distributed using the normal distribution.

of each bar is the current estimate for a given node. Each column also contains a mark whose height represents the actual number of neighbors for the node. Hence, the gap between the height of a bar and the mark shows the absolute error of the current estimate for a node in the current iteration.

Finally, we note that the quality of our results may be affected by the quality of the random function generator and floating-point operations implemented on our development platform. Hence, we used Java 1.3's StrictMath library, which implements the well-known "Freely Distribution Math Library (`fdlibm`) [7], to reduce the possibility of producing erroneous results caused by non-standard generator and operator implementations. This also helps us to obtain uniform results on different platforms running the Java simulation.

9.2 Implementation Issues

9.2.1 Computing $\gamma(z, p)$

Our estimation algorithm depends on computing the value of $\gamma(z, p)$. However, this computation is difficult in practice because it is not possible to get a closed-form solution for $\gamma(z, p)$. In addition, $\gamma(z, p)$ has two solutions for $z \neq (1-p)^{1/p}$, and the method we use for computing $\gamma(z, p)$ must produce the solution we want.

We considered two basic approximation schemes to solve this problem. First, we can use Newton's method to approximate $\gamma(z, p)$. However, this approach increases the running time of our simulation, as Newton's method requires many iterations of expensive floating-point operations. Moreover, Newton's method may return an arbitrary solution when multiple solutions exist, and so may return an incorrect solution for our purposes.

Instead, we implemented a table lookup method over a set of precomputed values of $\rho(n, p) =$

$np(1-p)^n$ to approximate $\gamma(z,p)$. We apply a binary search on the value of $\rho(n,p)$ among the precomputed values to obtain the correct n. Binary search allows us to retain high precision while saving memory, by keeping only N entries in the table. Our running time remains the same, as binary search is accomplished in $O(\log N)$ time.

Note that the table is also parameterized by p. Because we have a different p in each epoch, and there are $O(\log N)$ epochs, we need to construct $O(\log N)$ tables. Hence, the total space used is $O(N \log N)$.

We have considered an alternative table lookup scheme to eliminate the $O(\log N)$ factor in the memory consumption. If we let $p = k/n$, then

$$\rho(z,p) = np(1-p)^n = k(1-k/n)^n \approx ke^{-k}$$

Rather than keeping $O(\log N)$ tables with different values of p, we keep one table, map z, p to a solution for k, and then find n by computing $n = k/p$. However, this method works only when n is large, as $(1-k/n)^n \not\approx e^{-k}$ for small n. Hence, this scheme does not work well unless the node density is sufficiently high.

9.2.2 Stopping Estimates After Receiving $\mu(\varepsilon,\delta)$ Packets

The algorithm implemented in our simulator actually deviates from Algorithm 1. In particular, nodes stop updating the counters c_i and j after they have received $\mu_{\varepsilon,\delta}$ uncorrupted packets[2]. When this is done, the accuracy of the neighbor size estimate does not improve by increasing β arbitrarily. Using this implementation, the accuracy is controlled by choosing the desired (ε,δ) bounds on the estimates, and then using the Chernoff bound to derive the required $\mu_{\varepsilon,\delta}$ to acheive the bounds. Then, we use our analysis to find the appropriate β which allows every node to receive $\mu_{\varepsilon,\delta}$ packets with high probability. Finally, we verify our analysis by comparing the observed (ε',δ') values against the chosen (ε,δ).

10 Simulation Results

10.1 Methodology

As described in Algorithm 1, the user gives as input the number of nodes N, ε, and δ. Using these input parameters, we have shown how to derive a parallel (ε, δ)-FPRAS.

In the simulator, instead of specifying δ and ε, the user specifies $\mu_{\varepsilon,\delta}$. In our simulations, we wanted to get a ($\delta = 0.05$, $\varepsilon \in [0.05, 1]$) approximation for the N-EST problem. Since $\mu_{\varepsilon,\delta} \triangleq \frac{3}{\varepsilon^2}\log\frac{1}{\delta}$, we set $\mu_{\varepsilon,\delta} = 2500$, which gives an $\varepsilon \approx 0.07$ for $\delta = 0.05$. In addition, we tested $\mu_{\varepsilon,\delta} = 500$. With $\delta = 0.05$, this corresponds to a $\varepsilon \approx 0.16$.

Once $\mu_{\varepsilon,\delta}$ is input, we can determine the maximum number of steps required to execute the algorithm. Let n_i be the number of neighbors for node i, then i will have a good approximation to n_i after receiving $\mu_{\varepsilon,\delta}$ good signals. Hence, we need to determine the total sampling time required for all nodes to receive $\mu_{\varepsilon,\delta}$ uncorrupted packets with high probability. As discussed previously, the total sampling time required is at most $\lceil 395\mu_{\varepsilon,\delta}\log^2 N\rceil$. Thus, given $\mu_{\varepsilon,\delta}$, we can determine the maximum number of steps required to produce our guarantee.

[2]One can view this as an optimization for saving energy. The nodes shut down their receivers after they received μ packets. Even though the nodes stop recording receptions, they must continue to broadcast, because neighboring nodes may not have received μ packets. The energy saving from shutting down receivers can be quite significant, because receiving data takes more power than transmitting data [6].

In addition to N and $\mu_{\varepsilon,\delta}$, the user needs to specify a region size, the maximum communication range r, and a distribution to the simulator (see Table 1). A distribution needs to be specified because we must scatter the nodes in some fashion. Scattering the nodes using a uniform probability distribution would be the most desirable. However, since deployment is dependent on the terrain and method of deployment, this is often not achievable in practice. Hence, the simulation allows the user to choose to scatter nodes using a Gaussian or uniform distribution.

In simulating our algorithm, we performed a set of simulation runs using a uniform distribution and a single run using a Gaussian distribution. In all cases, the extent of our region was 200×200. Our simulation was a straightforward $O(N^2)$ sequential implementation of Algorithm 1. For this reason, it was infeasible to simulate values of $N > 1000$, and so the asymptotic behavior of our algorithm was not directly observed.

For the uniform distribution, we first fixed $r \in \{25, 50\}$. Then we varied $N \in \{50, 100, 200, 500\}$ and $\mu_{\varepsilon,\delta} \in \{500, 2500\}$. Therefore, a total of 16 simulations were performed using a uniform distribution of the nodes. For the Gaussian case, we fixed $r = 50, N = 500, \mu = 1000$. For $\mu = 1000$ and $\delta = 0.05$, we expect at least ε factor deviation of at least 11% for 95% of the nodes.

10.2 Results

10.2.1 Uniformly distributed nodes

First, we show the results obtained from simulating nodes distributed uniformly. We only give a subset of all the results we gathered, however, the results obtained are sufficient for comparison with the theoretical analyses. Table 2 shows the mean, standard deviation, median, minimum, and maximum percentage errors from the actual number of neighbors for the nodes for various N and $\mu_{\varepsilon,\delta}$. Notice that the mean of the error is well within our desired error of 0.07.

Table 2: Statistics concerning the accuracy of our algorithm. The nodes were distributed using a uniform distribution and $r = 50$. The error is reported as a fraction of the actual.

N	$\mu_{\varepsilon,\delta}$	mean	std	median	min	max
50	500	0.0301	0.0674	0	0	0.3333
	2500	0.0019	0.0130	0	0	0.0909
100	500	0.0331	0.0459	0	0	0.2222
	2500	0.0157	0.0268	0	0	0.1000
200	500	0.0358	0.0316	0.0323	0	0.1379
	2500	0.0135	0.0199	0	0	0.0909
500	500	0.0488	0.0370	0.0435	0	0.2083
	2500	0.0189	0.0154	0.0169	0	0.0690

To get an idea of the error obtained by all the nodes, we created several "cumulative" plots. Figure 3 shows a cumulative plot of the percentage of total nodes at a specific error deviation for $r = 50, \mu_{\varepsilon,\delta} = 2500$, and various N. The meaning of these graphs and how to interpret them is best illustrated through an example. For instance, for $N = 100$, approximately 95% of the nodes have at most a 7% deviation from their actual number of neighbors. This matches our theoretical predictions. We expected 95% of the nodes to be within a factor of $\varepsilon \approx 0.07$ nodes. Note that for other N, our experimental results matched our theoretical predictions as well.

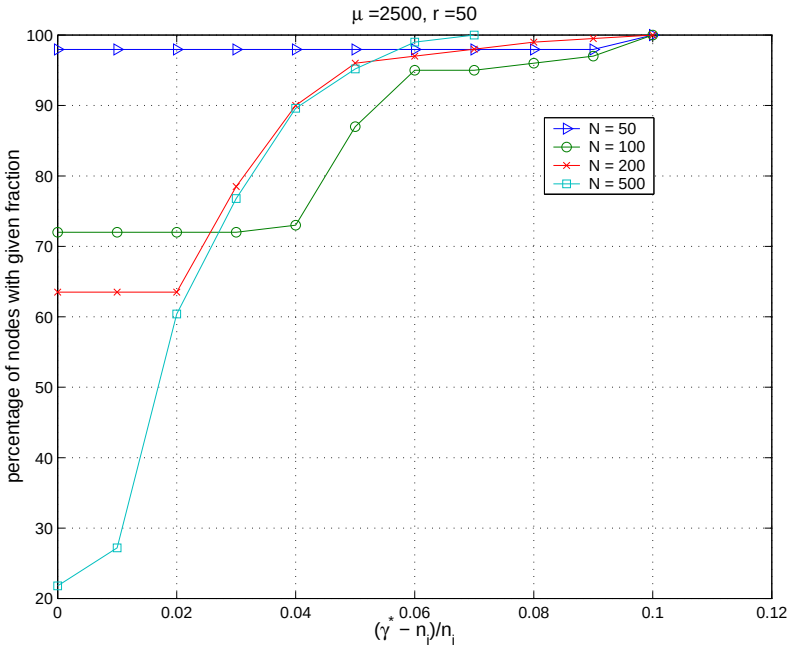

Figure 3: Cumulative plot of the error percentage for nodes uniformly distributed over a 200×200 region with $r = 50, \mu = 2500$, and N variable. Note that $\gamma^* = \gamma(\frac{\lambda_i^{\iota_i+2}}{\beta}, p_{\iota_i+2})$. For example, for $N = 100$, approximately 95% of the nodes have at most a 7% deviation from their actual number of neighbors.

An interesting feature in Figure 3 is the graded nature of the results. This is because some nodes have few neighbors and hence, any slight error in the absolute estimate in the number of neighbors causes a large jump in the relative estimate. Figure 4 shows a cumulative plot of the percentage of total nodes at a specific error deviation for $r = 50, \mu_{\varepsilon,\delta} = 500$ and various N. Again, the experimental results match or exceed our theoretical predictions.

In addition to the accuracy of our algorithm, we wished to assess the running time needed to obtained the desired accuracy. Table 3 shows the mean, standard deviation, median, minimum, and maximum number of time steps for each node to obtain $\mu_{\varepsilon,\delta}$ uncorrupted packets and thus, a good approximation of the neighborhood size of the node. In theory, we expect that all nodes will receive $\mu_{\varepsilon,\delta}$ in at most $\lceil 395\mu_{\delta,\epsilon} \log^2 N \rceil$ time. This theoretical upper bound on the expected time is shown in the last column of Table 3.

Clearly, the mean finishing times are much less than the expected times. Thus, the upper bound of the running time that we have computed theoretically is loose. In fact, all of the nodes successfully receive $\mu_{\varepsilon,\delta}$ good signals two orders of magnitude before the allotted time. This shows that in practice the upper bound is very weak. In addition, we have again produced plots to track the various finishing times of the nodes. Figure 5 shows a cumulative plot of the percentage of total nodes at a specific error deviation for $r = 50, \mu_{\varepsilon,\delta} = 2500$ and various N. While Figure 6 shows a cumulative plot of the percentage of total nodes at a specific error deviation for $r = 50, \mu_{\varepsilon,\delta} = 500$ and various N.

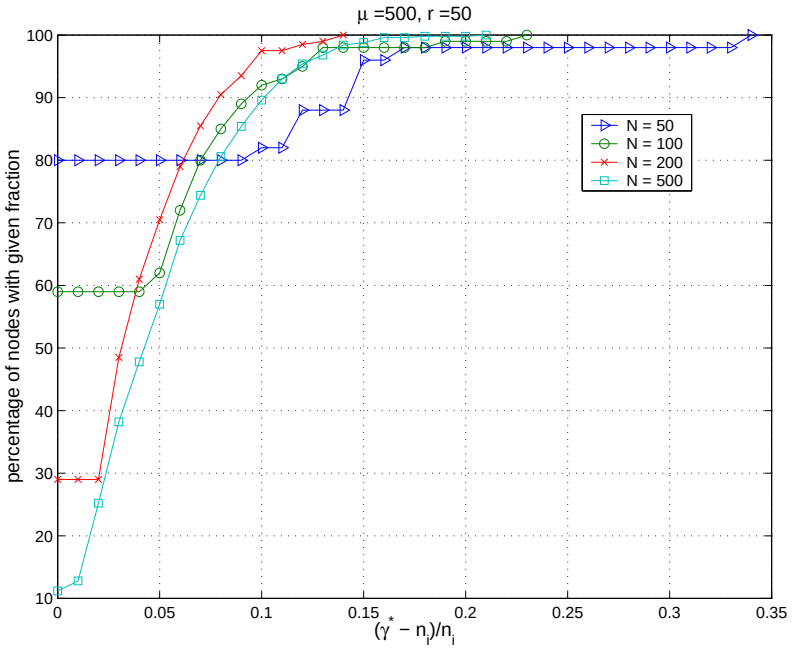

Figure 4: Cumulative plot of the error percentage for nodes uniformly distributed over a 200×200 region with $r = 50, \mu = 500$, and N variable. Note that $\gamma^* = \gamma(\frac{\lambda_i^{\iota_i+2}}{\beta}, p_{\iota_i+2})$.

Table 3: Statistics concerning the time (in steps) at which a node received $\mu_{\varepsilon,\delta}$ good packets. The nodes were distributed using a uniform distribution and $r = 50$.

N	$\mu_{\varepsilon,\delta}$	mean	std	median	min	max	expected
50	500	2401.98	385.48	2374.00	1706	3270	6.2×10^6
	2500	12422.65	2001.57	12227.00	8903	17068	3.1×10^7
100	500	2584.08	367.37	2547.50	1865	3479	8.7×10^6
	2500	12808.72	2117.58	12230.50	9973	17608	4.4×10^7
200	500	2597.53	486.35	2444.00	1933	3847	1.2×10^7
	2500	12827.43	1896.12	12408.50	9744	17994	5.8×10^7
500	500	2827.44	570.56	2973.00	1736	3752	1.6×10^7
	2500	14017.98	2239.93	14692.00	9650	17785	7.9×10^7

10.2.2 Gaussian distributed nodes

In the Gaussian simulations, we set $\mu = 1000$, which corresponds to a $\varepsilon \approx 0.11$ for $\delta = 0.05$. The results obtained through simulation are shown in Table 4, Figure 7, and Figure 8. As before, the errors and time taken are well within our theoretical predictions.

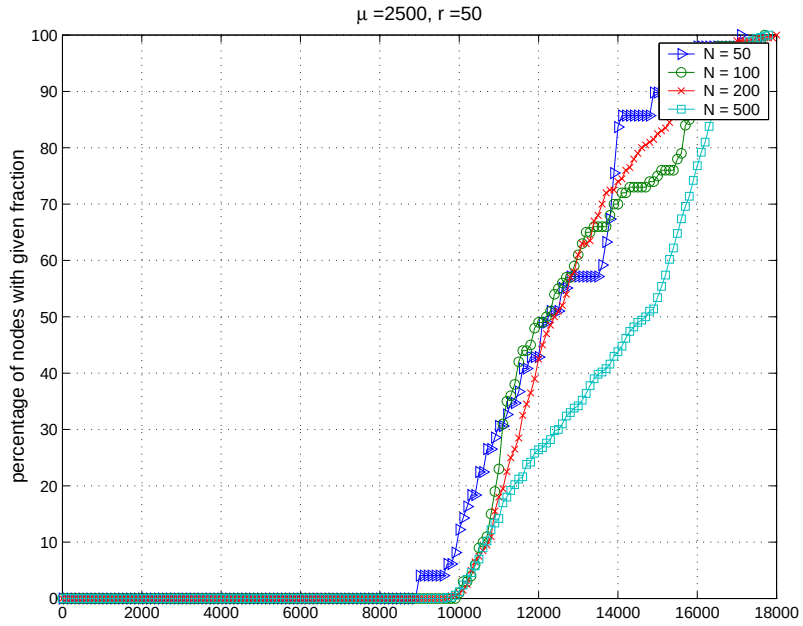

Figure 5: Cumulative plot of the time (in steps) at which a node received $\mu_{\varepsilon,\delta}$ good packets for nodes uniformly distributed over a 200×200 region with $r = 50, \mu = 2500$, and N variable.

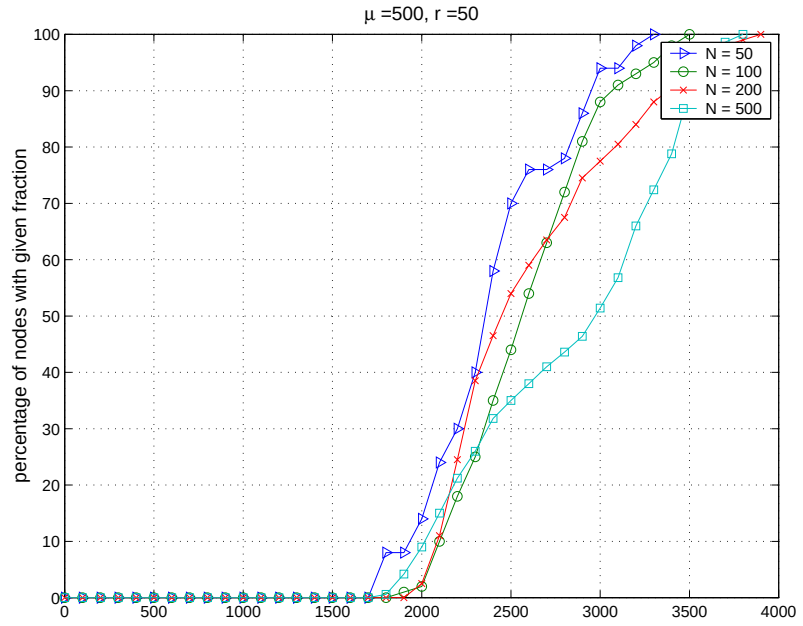

Figure 6: Cumulative plot of the time (in steps) at which a node received $\mu_{\varepsilon,\delta}$ good packets for nodes uniformly distributed over a 200×200 region with $r = 50, \mu = 500$, and N variable.

11 Future Work

Our analysis currently assumes that all nodes have synchronized clocks and make the decision to broadcast at the same time. However, this assumption may not be practical, and the algorithm

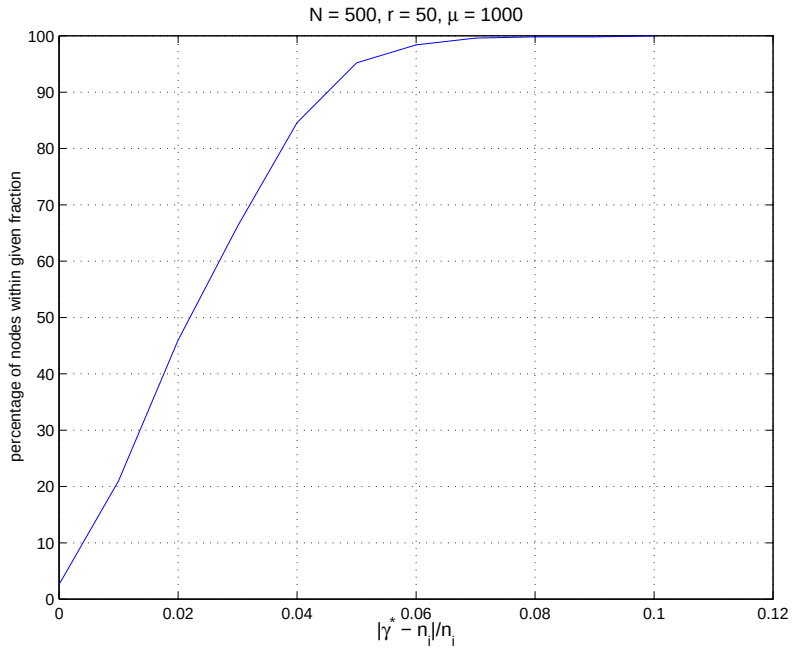

Figure 7: Cumulative plot of the error percentage for nodes distributed using a Gaussian distribution over 200×200 region ($N = 500, r = 50, \mu = 1000$). Note that $\gamma^* = \gamma(\frac{\lambda_i^{\iota_i+2}}{\beta}, p_{\iota_i+2})$.

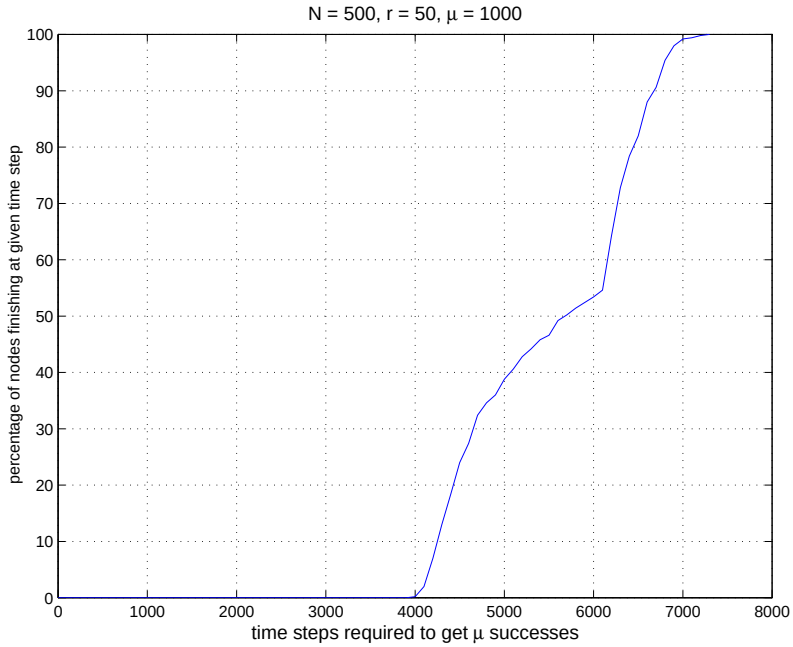

Figure 8: Cumulative plot of the time (in steps) at which a node received $\mu_{varepsilon,\delta}$ good packets. The nodes were deployed using a Gaussian distribution over a 200×200 region (N = 500, r = 50, $\mu = 1000$).

Table 4: Some statistics concerning the performance of our algorithm. The nodes were distributed using a normal distribution and $N = 500, r = 50, \mu = 1000$.

	mean	std	median	min	max
Error (%)	2.39	1.51	2.20	0	10
Time (steps)	5519.95	949.57	5681.50	3996	7254

should be analyzed for the case when the nodes are not synchronized. Under this relaxed assumption, we believe that our algorithm should not perform much worse. Without synchronization, a single packet can overlap and collide anywhere within a time slot. Hence, the only value that is affected under this relaxed assumption is $\rho(n, p)$, the probability of successfully receiving a packet. Note that if a broadcast packet consumes one time slot, it can overlap in at most two time slots. Hence, we can derive a new expression for $\rho(n, p)$, and derive similar analyses using the framework presented in this paper.

In addition, as described in the results section, our simulations illustrate that the bound on the running time of Algorithm 1 is rather weak. Perhaps through the use of additional probabilistic tools or by changing the analsis approach, we can reduce the constant factor in our bound of the running time.

Finally, our algorithm is designed to estimate the number of neighboring nodes in a static network. It would be interesting to see if this technique can be extended and/or modified to handle a dynamic network. In such a network, nodes change positions or become active and inactive over time. Since the number of active neighbors varies over time, it is unclear if random sampling can be used solve the problem.

12 Conclusion

In this paper, we have presented N-EST, a novel problem of estimating the number of neighboring nodes in an ad-hoc wireless network. We have presented a parallel FPRAS that runs in $O(\log^2 N)$ time, which is much asymptotically faster than the simple deterministic algorithm that runs in $\Theta(N)$ time. The algorithm makes no initial assumption about the physical node distribution, the area over which the nodes are distributed, and the broadcast radius of the nodes. We also presented a variety of simulation results and confirmed the theoretical performance of our algorithm.

References

[1] Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Comput System Sci*, 55(3):441–453, 1997.

[2] T. Friedman and D. Towsley. Multicast session size membership estimation. In *IEEE INFOCOM 1999*, March 1999.

[3] David R. Karger. Private Communication, November 2000.

[4] Richard M. Karp, Michael Luby, and Neal Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10:429–448, 1989.

[5] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[6] National Semiconductor Corporation. *LMX3162 Evaluation Notes and Datasheet*, April 1999.

[7] Kwok C. Ng. C Math Library for machines that support IEEE 754 floating point. http://www.netlib.org, January 1995.

[8] Sheldon M. Ross. *Introduction to Probability and Statistics for Engineer and Scientists.* John Wiley and Sons, 1 edition, 1987.