Neighbor Discovery in Wireless Networks and the Coupon Collector's Problem

UMass Computer Science Technical Report UM-CS-2009-032

Sudarshan Vasudevan[*]
svasu@cs.umass.edu

Don Towsley[†]
towsley@cs.umass.edu

Dennis Goeckel[‡]
goeckel@ecs.umass.edu

Ramin Khalili[§]
ramin.khalili@epfl.ch

ABSTRACT

Neighbor discovery is one of the first steps in the initialization of a wireless ad hoc network. In this paper, we design and analyze practical algorithms for neighbor discovery in wireless networks. We first consider an ALOHA-like neighbor discovery algorithm in a synchronous system, proposed in an earlier work. When nodes do not have a collision detection mechanism, we show that this algorithm reduces to the classical *Coupon Collector's Problem*. Consequently, we show that each node discovers all its n neighbors in an expected time equal to $ne(\ln n + c)$, for some constant c. When nodes have a collision detection mechanism, we propose an algorithm based on receiver status feedback which yields a $\ln n$ improvement over the ALOHA-like algorithm.

Our algorithms do not require nodes to have any estimate of the number of neighbors. In particular, we show that not knowing n results in no more than a factor of two slowdown in the algorithm performance. In the absence of node synchronization, we develop asynchronous neighbor discovery algorithms that are only a factor of two slower than their synchronous counterparts. We show that our algorithms can achieve neighbor discovery despite allowing nodes to begin execution at different time instants. Furthermore, our algorithms allow each node to detect when to terminate the neighbor discovery phase.

[*]Department of Computer Science, University of Massachusetts, Amherst, MA 01003

[†]Department of Computer Science, University of Massachusetts, Amherst, MA 01003

[‡]Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003

[§]School of Computer and Communication Sciences, EPFL, CH-1015, Lausanne, Switzerland

1. INTRODUCTION

Wireless ad hoc networks and sensor networks have generated considerable attention recently due to many applications such as habitat monitoring [13], environmental observation [3], surveillance and tracking [23], and community networking [16]. These networks are typically deployed without any communication infrastructure and are required to "configure" themselves upon deployment, in order to establish an efficient communication infrastructure. For instance, immediately after deployment, a node has no knowledge of other nodes in its transmission range and needs to discover its neighboring nodes in order to efficiently communicate with other nodes in the network. Knowledge of one-hop neighbors is required by medium access control protocols [2], routing algorithms [9], and topology control algorithms [11]. Thus, neighbor discovery is one of the first steps in the configuration of a large wireless network.

Neighbor discovery algorithms can be classified into two categories, viz. *randomized* or *deterministic*. In a randomized neighbor discovery algorithm, each node transmits at randomly chosen times and yet discovers all its neighbors by a given time with high probability. In a deterministic neighbor discovery algorithm, each node transmits according to a pre-determined transmission schedule that allows it to discover all its neighbors by a given time with probability one. Guaranteed neighbor discovery typically comes at the cost of increased running time and often requires unrealistic assumptions such as synchronization between nodes and apriori knowledge of the number of neighbors [10]. We, therefore, choose to investigate randomized neighbor discovery algorithms in this paper.

The neighbor discovery problem is non-trivial due to several reasons:

1. A neighbor discovery algorithm, either randomized or deterministic, needs to cope with collisions. A randomized neighbor discovery algorithm needs to minimize the probability of collisions, while a deterministic algorithm requires determining a collision-free transmission schedule among nodes.

2. In many practical settings, nodes have no knowledge

of the number of neighbors, which makes coping with collisions even harder.

3. When nodes do not have access to a global clock, they need to operate asynchronously and still be able to discover their neighbors efficiently.

4. In asynchronous systems, nodes can potentially start the neighbor discovery process at different time instants and consequently, may miss each other's transmissions.

5. Furthermore, when the number of neighbors is unknown, nodes do not know apriori when/how to terminate the neighbor discovery process.

In this paper, we study the neighbor discovery problem when nodes have omni-directional antennas and propose algorithms that comprehensively handle each of these challenges. Unlike existing approaches that require apriori estimates of the number of neighbors or synchronization among nodes, we propose neighbor discovery algorithms that:

P1 do not require nodes to have any apriori knowledge of the number of neighbors,

P2 do not require synchronization among nodes,

P3 allow nodes to begin execution at different time instants, and

P4 enable each node to detect when to terminate the neighbor discovery process.

To the best of our knowledge, this is the first solution to the neighbor discovery problem that satisfies the properties **P1-P4**. We start out making a number of unrealistic assumptions violating each of **P1-P4**. As we will see, the analysis in such a simplistic setting provides us with valuable insight about the problem. Armed with this insight, we progressively relax each of the assumptions, each time taking a significant step towards a comprehensive and a practical solution to the neighbor discovery problem.

Existing approaches to the neighbor discovery problem when nodes have omni-directional antennas can be classified into 3 categories, viz. *randomized* [15, 4], *deterministic* [10], and *multi-user detection-based* [1, 12]. However, these solutions require either apriori knowledge of the number of neighbors [15, 4, 10], or node synchronization [15, 10]. The solutions proposed in [1, 12] are based on correlating the received signal with node signatures. However, each node is assumed to know apriori the correspondence between nodes and their signatures. In [5], the authors consider the problem of rendezvous between two duty cycling nodes in the absence of prior synchronization information. In comparison to [5], we are interested in the problem of efficient discovery of all one hop neighbors.

Neighbor discovery algorithms when nodes have directional antennas have been proposed in [22, 8, 20]. Again, the proposed solutions assume knowledge of the number of neighbors [22], or node synchronization [8]. In [20], the authors propose antenna scanning mechanisms for directional neighbor discovery. However, the focus in [20] is on a complete systems solution for building ad hoc networks with directional antennas and no significant analysis of the neighbor discovery algorithms is provided.

1.1 Main Results

The main results of this paper can be summarized as follows:

1. We consider the ALOHA-like neighbor discovery algorithm proposed in [15] and show that its analysis reduces to that of the *Coupon Collector's Problem*. Consequently, we show that each node discovers all its n neighbors in an expected amount of time equal to $ne(\ln n + c)$, for some constant c.

2. We then propose a neighbor discovery algorithm that assumes nodes can detect collisions i.e., nodes can distinguish between collisions and idle slots. We show that collision detection results in a $\ln n$ improvement over the ALOHA-like algorithm.

3. We next show that absence of an estimate of the number of neighbors results in a slowdown of no more than a factor of two, compared to when nodes know n.

4. We further show that lack of synchronization among nodes results in at most a factor of two slowdown in the algorithm performance from the case when nodes are synchronized.

5. Finally, we show that despite starting execution at different time instants, each node can discover all its neighbors. Furthermore, when nodes do not know n, we show that with high probability (w.h.p), each node can determine that it has discovered all its neighbors.

We thus have an $O(ne \ln n)$ ALOHA-like discovery algorithm when nodes do not have a collision detection mechanism and an $O(ne)$ Collision Detection-based algorithm for the case when nodes can detect collisions, both of which satisfy properties **P1-P4**.

1.2 Organization of the paper

The rest of the paper is structured as follows. In Section 2, we describe our model and its assumptions. Section 3 describes the ALOHA-like neighbor discovery algorithm and its analysis. We next present the collision detection-based algorithm for the case when nodes have collision detection in Section 4. In Section 5, we consider the case when nodes have no estimate of the number of neighbors. In Section 6, we describe asynchronous versions of the neighbor discovery algorithms. In Section 7, we handle initiation and termination of the neighbor discovery phase. Finally, we conclude in Section 8, and present future directions.

2. NETWORK MODEL AND ASSUMPTIONS

In this section, we introduce our network model which will be used throughout the rest of this paper. We make the following assumptions about the multi-hop wireless ad hoc network:

- **Unique Node IDs:** We assume that the nodes have unique identifiers. Unlike the stronger requirement that the identifiers be *globally* unique, we only require nodes to have *locally* unique identifiers i.e. no two neighbors of a given node have the same identifier. The identifier could be the MAC address of the node, the location of the node, or a random bit string of length chosen to be locally unique w.h.p.
- **Radio Model:** Each node is equipped with a radio transceiver that allows a node to either transmit or receive messages, but not simultaneously. Furthermore, we assume that enough frequency or time diversity exists in the transmission of a message that the short-scale variation due to multipath fading is averaged out. Thus, two nodes are neighbors if the received signal strength (after path-loss and shadowing) at one node given a transmission from the other exceeds a decoding threshold, and, under the assumption of a static network, this neighbor designation does not change during the neighbor discovery period.
- **Collision Model:** When two or more nodes transmit concurrently, a collision occurs at the recipient node. When a collision occurs, we assume that no partial recovery of packets is possible at the receiving node.
- **Collision Detection:** A collision detection mechanism allows a node to distinguish between the case where two or more nodes are transmitting and one where no node is transmitting. Indeed, practical solutions for collision detection have been proposed in [6, 7]. In this paper, we study two neighbor discovery algorithms, first when nodes do not have a collision detection mechanism and next when nodes do and study the impact of collision detection on algorithm performance.

We observe that the radio and collision model considered in the paper, although idealized, help in analyzing the performance of neighbor discovery algorithms. As we will see, the insights provided by our analysis in turn are used to design practical algorithms for the neighbor discovery problem.

3. ALOHA-LIKE NEIGHBOR DISCOVERY ALGORITHM

In this section, we consider the ALOHA-like neighbor discovery algorithm first proposed in [15], which assumes that nodes do not have a collision detection mechanism. We start out making several simplifying assumptions:

1. We consider a single clique of size n.
2. n is known to all nodes in the clique.
3. Time is divided into slots and nodes are synchronized on slot boundaries.

Each of these assumptions will be relaxed as we proceed. Importantly, these assumptions allow us to view the ALOHA-like neighbor discovery as a *Coupon Collector's Problem.* Consequently, the time to discover the n neighbors is the same as the minimum time to collect at least one of each of n coupon types.

3.1 Algorithm Description

The ALOHA-like neighbor discovery algorithm is a randomized algorithm that operates as follows. In each time slot, a node independently transmits with probability p_t, a parameter to be determined, and listens with probability $1 - p_t$. A discovery is made if exactly one node transmits in a slot; otherwise no discovery is made in that time slot.

The goal then is to choose p_t so as to maximize the expected fraction of neighbors discovered in a given time slot (as considered in [15]), or the probability of discovering a neighbor within a given time (as considered in [22]). Not surprisingly, the optimal choice of p_t can be shown to be $1/n$, where n is the clique size. Intuitively, this choice of p_t maximizes the per-slot throughput of a given node, as has been widely studied and well-understood in the context of the ALOHA protocol, leading to maximization of the discovery rate.

However, the crucial question of the time required to discover all the neighbors when nodes transmit with $p_t = 1/n$ were unanswered in [15, 22]. In the remainder of this section, we assume that nodes transmit with probability $p_t = 1/n$ and proceed to analyze the time to discover all n neighbors. Before doing so, we first describe the *Coupon Collector's Problem* and subsequently describe how it relates to the ALOHA-like discovery algorithm.

3.2 Classical Coupon Collector's Problem

The classical *Coupon Collector's Problem* [1] is defined as follows. There are n distinct objects that are repeatedly drawn (with replacement) from an urn with a probability of $1/n$ of picking an object at each trial. The question then is: what is the minimum number of trials needed to pick each of the n objects at least once? If the random variable W denotes the number of trials, it can be shown that $E[W] = n(\ln n + c)$, for some constant c. Furthermore, it can be shown that the random variable W is sharply concentrated around its mean. More formally, $P[W > n \ln n + cn] = 1 - e^{-e^{-c}}$, which tends to zero for large positive c and tends to one for large negative c.

3.3 Neighbor Discovery As Coupon Collection

In this section, we describe how the neighbor discovery problem maps into the Coupon Collector's Problem. Consider a clique of n nodes numbered $1, \ldots, n$. The probability that node i successfully transmits in a given time slot is given by:

$$p_s = p_t(1 - p_t)^{n-1} = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \approx \frac{1}{ne} \qquad (1)$$

Note that p_s is the same for each node i, $1 \leq i \leq n$.

The process of neighbor discovery can be then be treated as a coupon collection problem in the following manner: consider a fictitious node C which can be regarded as the coupon collector. In each slot, C draws one of the n coupons (i.e. discovers a given node) with probability p_s, and draws no coupon (corresponding to an idle slot or a collision) with probability $1 - np_s$. It is easy to see that when C collects n distinct coupons, in our case, it means that each node in the clique has discovered all of its $n - 1$ neighbors. Most of

[1] See [17] for an elegant derivation of the main results for the Coupon Collector's Problem.

the following analysis proceeds in a similar manner to that of the Coupon Collector's Problem.

Let the random variable W denote the number of slots needed for the fictitious node C to discover all n nodes. We can think of the neighbor discovery process as divided into epochs, each epoch consisting of one or more time slots. Let W_i denote the length of epoch i, $0 \leq i \leq n-1$, that starts when the i-th node is discovered and ends when the $i+1$-st node is discovered. Thus, in the i-th epoch there are $n-i$ nodes yet to be discovered, each of which has a probability p_s of being discovered in a given time slot. It is easy to see that the epoch length,W_i, is a Geometric random variable with parameter $(n-i)p_s$. Thus, noting that $W = W_0 + \ldots + W_{n-1}$, we get

$$E[W] = \sum_{i=0}^{n-1} E[W_i] = \frac{1}{(n-i)p_s} = \frac{1}{p_s}\sum_{i=1}^{n}\frac{1}{i} = \frac{1}{p_s}H_n \approx neH_n$$

where H_n denotes the n-th Harmonic number and is given by $\ln n + \theta(1)$. Therefore,

$$E[W] = ne(\ln n + \theta(1)) = ne\ln n + O(n) \qquad (2)$$

In Appendix A.1, we obtain an upper bound on the error introduced in p_s due to the approximation in (1) and show that the error goes to 0, for large n. In other words, $E[W] \rightarrow ne$, for large n.

3.4 Sharp Concentration Around The Mean

We next show that W is sharply concentrated around its mean. As described in [17], we use the Poisson approximation to the binomial distribution to provide an approximate argument. In Appendix B, we derive the sharp threshold result using a more rigorous analysis that employs Boole-Bonferroni Inequalities, similar to that described in [17].

Let $N_i(t)$ be a random variable that denotes the number of successful transmissions by node i in the first t time slots. It is easy to see that $N_i(t)$ is a binomial random variable with probability mass function:

$$P(N_i(t) = k) = \binom{t}{k} p_s^k (1-p_s)^{t-k}$$

Using the Poisson approximation (assuming large t and small p_s),

$$P(N_i(t) = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

where $\lambda = tp_s$. Let E_i^t denote the event that node i is not discovered in t slots. Therefore,

$$P(E_i^t) = P(N_i(t) = 0) = e^{-tp_s}$$

Substituting $p_s = \frac{1}{ne}$ as given by (1) into the expression for $P(E_i^t)$ yields

$$P(E_i^t) = e^{-\frac{t}{ne}}$$

Therefore,

$$P(\neg E_i^t) = 1 - e^{-\frac{t}{ne}}$$

We are interested in the probability that all n nodes are discovered by time t i.e. $P[\neg(\cup_{i=1}^n E_i^t)]$.

$$P[\neg(\cup_{i=1}^n E_i^t)] = P[\cap_{i=1}^n(\neg E_i^t)] \qquad (3)$$

We next show that E_i^ts can be treated as independent events.

Claim: For $1 \leq i \leq n$, and for any set of indices $\{j_1, \ldots j_k\}$ not containing i, $P[E_i^t | \cap_{l=1}^k E_{j_l}^t] \approx P[E_i^t]$

PROOF.

$$P[E_i^t | \cap_{l=1}^k E_{j_l}^t] = \frac{P[E_i^t \cap (\cap_{l=1}^k E_{j_l}^t)]}{P[\cap(\cap_{l=1}^k E_{j_l}^t]} = \frac{(1-(k+1)p_s)^t}{(1-kp_s)^t}$$

Using the approximation $1+x \approx e^x$ for small x, in the above equation yields:

$$P[E_i^t | \cap_{l=1}^k E_{j_l}^t] \approx \frac{e^{-t(k+1)p_s}}{e^{-tkp_s}} = e^{-tp_s} = e^{-\frac{t}{ne}} = P[E_i^t]$$

□

Substituting into (3), we get

$$P[\neg(\cup_{i=1}^n E_i^t)] = (1 - e^{-\frac{t}{ne}})^n \approx e^{-ne^{\frac{-t}{ne}}}$$

Therefore,

$$P[W > t] = 1 - P[\neg(\cup_{i=1}^n E_i^t)]$$

Let $t = ne(\ln n + c)$ for some constant $c \in \Re$. Thus,

$$P[W > t = ne(\ln n + c)] = 1 - e^{-ne^{-(\ln n + c)}} = 1 - e^{-e^{-c}}$$

Observe that $e^{-e^{-c}}$ is close to 1 for large positive c and is negligibly small for large negative c, thus implying a sharp concentration around the mean.

3.5 Validation of Clique Assumption

Our analysis of the ALOHA-like neighbor discovery algorithm was restricted to a single clique. In this section, we show that the analytical results compare remarkably well against results obtained from simulation of a multi-hop network, even for small clique sizes. The metric used for the comparison is the expected time required by a node to discover all its neighbors.

Our simulation setting consists of a uniform distribution of nodes over a 2D plane of area 3km × 3km. Each node has a fixed transmission range of 150m and is assumed to know the average number of neighbors apriori. For instance, with a total of 2000 nodes, the average number of neighbors per node is 16. Hence, the transmission probability of each node in our simulations is set to $p_t = 1/17$. After each run, we compute the average time required by a node to discover all its neighbors. For each node density, we run the simulation 20 times, each corresponding to a different node placement. Each data point shown in Figure 1 is an average over 20 runs. We plot the 95% confidence intervals around each data point (too small to be visible in the graph).

Note that the x-axis in the graph denotes the expected number of neighbors per node. To obtain the results from our analytical model for this deployment, we set the clique size, n, to be one more than the average number of neighbors per node i.e. substitute $n = 17$ in equation (2). In Figure 1, we show a comparison of the simulation and analytical results for different node densities. We immediately observe a close match between the analytical and simulation results. In fact,

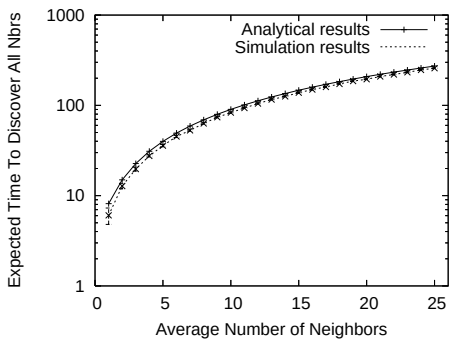

Figure 1: Validation of the Clique assumption

the analytical results are within 15% of the simulation results, even for small values of n, e.g. $4 \leq n \leq 8$. For larger values of n, the analytical results are always within 10% of the simulation results. When $n = 2$, we observe a 34% error between the analytical and simulation results. However, the large percentage error for small values of n is mainly due to the approximation in (1). Using an exact calculation, our analytical results are within 1% of the simulation results for $n = 2$ and within 3% for $n = 3$.

One reason for the close match between the analytical and simulation results is that nodes in the center of the plane have approximately the same number of neighbors, each of which transmits with the same probability p_t, which is well approximated by the clique assumption. The boundary nodes belong to smaller sized cliques compared to the central nodes, but they are far fewer in number and as a result, the average discovery time is dominated by the discovery times of the central nodes. In order to eliminate the boundary effects, we also simulate the algorithm by placing the nodes on a torus, producing more homogeneous deployments. As expected, absence of boundary effects results in a closer match between the simulation and analytical results for every value of n, as compared to the simulations on a 2D plane. For $n = 2$, we observe a 26% error, and less than 10% error for $n \geq 5$. Again, the large percentage error for smaller values of n is dominated by the approximation in (1).

4. COLLISION DETECTION-BASED NEIGHBOR DISCOVERY ALGORITHM

We next consider the scenario when nodes have a reliable collision detection mechanism. As will soon become clear, collision detection allows each node to know when it has been discovered by its neighbors. Thus, nodes can stop transmitting once they have been discovered, producing a significant improvement in performance over the ALOHA-like algorithm.

We start out with the same simplifying assumptions as described in Section 3, viz. we consider a single clique of size n, where n is known to all nodes and nodes are synchronized with each other.

4.1 Algorithm Description

Let the local variable i (maintained by each node) denote the number of neighbors discovered so far. Initially, $i = 0$. Each time slot is further sub-divided into two sub-slots, the significance of which will become clear as we describe the algorithm. The algorithm operates as follows.

At each time slot, each node does the following:

1. In the first sub-slot, transmit with probability $p_i = 1/(n - i)$ and receive with probability $1 - p_i$.
2. In the second sub-slot:
   (a) a node in receive mode in the first sub-slot, checks if the transmission was successful. If yes, it sets $i \leftarrow i + 1$. Else, it deterministically transmits in the second sub-slot.
   (b) if a node in the transmit mode in the first sub-slot, detects energy in the second sub-slot, it assumes its transmission was unsuccessful. Else, it will switch to the receive mode for the rest of the discovery process.

Thus, we see that collision-detection allows each node to keep track of the number of nodes yet to be discovered and adapt its transmission probability at each time slot. We also observe that, if a collision occurs in the first sub-slot, each node needs to transmit only one bit of information in the second sub-slot. As a result, the second sub-slot is much smaller in comparison to the first sub-slot and introduces only a small overhead.

There is, however, one problem with the algorithm described above. Consider the case when $i = 0$. Now, if all n nodes transmit in the first sub-slot, there is no receiver at all to provide feedback in the second sub-slot. Consequently, all n nodes erroneously assume that their transmissions were successful and therefore, switch to the receive mode for the rest of the discovery process. We next propose a solution to handle this problem.

4.2 The Case of All Nodes Transmitting

When $i = 0$, the probability, p_a, of all n nodes transmitting in the first sub-slot equals $1/n^n$. When n is large, this probability is negligibly small. For instance, when $n = 5$, $p_a = 1/5^5 = 0.0003$. But for smaller values of n, this probability is non-negligible. In order to provide feedback to the transmitters, we divide the second sub-slot of each time slot into r *mini-slots*, where r is fixed. Each transmitter in the first sub-slot then transmits in a randomly chosen k of the r mini-slots. Now, if a transmitter detects energy in any of the remaining $r - k$ mini-slots, it knows that there was at least one other transmitter in the first sub-slot and its transmission must, therefore, have been unsuccessful. A node in receive mode transmits in each of the r mini-slots on unsuccessful transmission and remains silent otherwise.

We now derive the probability P_e that each of the n nodes transmits in the first sub-slot and none of the nodes get feedback about their unsuccessful transmission. This happens when each of the n nodes transmit in the same k of the r mini-slots following the first sub-slot. Therefore,

$$P_e = \frac{1}{n^n} \frac{1}{\binom{r}{k}^n}$$

For example, when $r = 8$,$k = 4$, and $n = 2$, we see that $P_e = 0.00005$, which is very small. For larger values of n, P_e is even smaller. Thus, we see that even a small r is sufficient to keep P_e very small.

Note that the duration of each of the mini-slots can be as small as that of a single bit. Hence, the overhead introduced by the r mini-slots is very small. Furthermore, the r mini-slots are needed only until the discovery of the first node. As per our algorithm description, after the first node has been discovered, it will remain in receive mode for the rest of the discovery process and can reliably provide feedback to the transmitters. Based on the solution for handling simultaneous transmissions, we henceforth assume that P_e is negligibly small and ignore it in our analysis hereafter.

4.3 Expected Time To Discover All Neighbors

Again, we are interested in the time, W, required to discover all n nodes in the clique. As before, let us assume that discovery is divided into "epochs", each epoch consisting of one or more slots. Each "epoch" i, $0 \leq i \leq n-1$, starts when the i-th node is discovered and ends upon discovery of the $i+1$-st node. Let W_i denote the duration of epoch i. Then,

$$W = \sum_{i=0}^{n-1} W_i$$

Note that as per the description of the algorithm, there are $n-i$ undiscovered nodes in epoch i, each of which transmits with probability $p_i = 1/n - i$ at the beginning of each time slot throughout the duration of the epoch. The probability of a successful transmission in a time slot of epoch i is given by:

$$p_s(i) = p_i(1-p_i)^{n-i-1} = \frac{1}{n-i}\left(1 - \frac{1}{n-i}\right)^{n-i-1}$$

It is easy to see that each W_i is a geometrically distributed random variable with parameter $(n-i)p_s(i)$. Thus,

$$E[W] = \sum_{i=0}^{n-1} E[W_i] = \sum_{i=1}^{n} \frac{1}{\left(1-\frac{1}{i}\right)^{i-1}} \leq ne \qquad (4)$$

where the last inequality follows from Lemma 1 in Appendix A. Compared to the expression for $E[W]$ for the ALOHA-like discovery algorithm, we observe an improvement by a factor of at least $\ln n$. In Appendix A.2, we show that $ne - eH_n \leq E[W] \leq ne$, where H_n denotes the n-th Harmonic number. In other words, $E[W] = \Theta(ne)$.

4.4 Bounds on Deviation From Expectation

We next bound the probability that random variable W is significantly greater than its mean. Let W' be another random variable, which is a sum of n iid Geometric random variables with parameter $1/e$. It follows that W' is a negative binomial random variable with parameters n and p and has a probability mass function:

$$P(W' = t) = \binom{t-1}{n-1} p^n (1-p)^{t-n}, t = n, n+1, \ldots$$

Also, we know that $P(W' > t) = P(X < n)$, where $X \sim$ Binomial(t, p).

We now use Chernoff bounds for a binomial random variable to bound the probability that $P(X < n)$. In particular, for a binomial random variable X with mean $\mu = tp$, it can be shown [17, pp.70] that:

$$P(X < (1-\delta)\mu) < e^{-\mu\delta^2/2}, 0 < \delta \leq 1 \qquad (5)$$

Let $\delta = 1 - ne/t$. Note that $0 < \delta < 1, \forall t > ne$. Substituting into (5) yields:

$$P(W' > t) = P(X < n) < e^{\frac{-t}{2e}[1-\frac{ne}{t}]^2}, \forall t \geq ne$$

Therefore,

$$P(W' > t = 2ne) < e^{\frac{-n}{4}}$$

Noting that $W = \sum_{i=1}^{n} W_i$, where each W_i is geometrically distributed with parameter at least $1/e$, it is easy to see that

$$P(W > t) \leq P(W' > t) < e^{\frac{-n}{4}}$$

It is easy to see that the bound on the right hand side goes to 0 for large values of n. In other words, $W \leq 2ne$ w.h.p.

5. UNKNOWN NUMBER OF NEIGHBORS

Thus far, we assumed that each node has apriori knowledge of the number of neighbors n. Knowledge of number of neighbors allowed each node to set its transmission probability to the optimal value of $1/n$. Furthermore, knowledge of n makes the problem of termination of neighbor discovery trivial. In particular, we can configure each node to execute the algorithm for a duration of t slots, where t is chosen large enough that the probability of not discovering all n neighbors by time t (as determined by our analyses in earlier sections) is negligibly small.

We are now ready to relax the assumption that each node knows n. In fact, we consider the most stringent condition in which each node has no estimate of n. As we will see, rather surprisingly, the lack of an estimate of n, results in at most a factor of two slowdown. We postpone the problem of detecting termination when nodes do not know n to Section 7.

5.1 ALOHA-like Neighbor Discovery Algorithm

We first show how the ALOHA-like algorithm can be extended to operate when n is unknown to the nodes.

The execution of the algorithm proceeds in phases, each phase consisting of one or more time time slots. The algorithm operation is very simple. In phase i, each node transmits with probability $1/2^i$, where each phase i lasts a duration of $2^i e(\ln 2^i + c)$ slots, where c is a positive constant.

The overall idea is that nodes geometrically reduce their transmission probabilities until they enter the desired phase of execution. In our case, this occurs when nodes enter the $\lceil \log_2 n \rceil$-th phase. In this phase, each node transmits with probability close to $1/n$ for a duration of $ne(\ln n + c)$ slots. Based on our analysis in Section 3, we know that the probability that there is an undiscovered node after the $\lceil \log_2 n \rceil$-th phase is $1 - e^{-e^{-c}}$. In other words, all n nodes are discovered within $\lceil \log_2 n \rceil$ phases w.h.p.

The total time W spent (in terms of number of slots) before all nodes are discovered w.h.p is then the sum of the total

number of slots in the first $\lceil \log_2 n \rceil$ phases of the algorithm execution and is given by:

$$W = \sum_{m=1}^{\lceil \log_2 n \rceil} 2^m e(\ln 2^m + c)$$

$$W = e \ln 2 \sum_{m=1}^{\lceil \log_2 n \rceil} m2^m + ce \sum_{m=1}^{\lceil \log_2 n \rceil} 2^m \quad (6)$$

The first term in the above summation can be evaluated as follows. Let

$$f(x) = \sum_{m=0}^{k-1} x^m = \frac{x^k - 1}{x - 1}$$

Differentiating $f(x)$ with respect to x, we get:

$$f'(x) = \sum_{m=0}^{k-1} mx^{m-1} = \frac{(x-1)kx^{k-1} - (x^k - 1)}{(x-1)^2}$$

It is easy to see that the first summand in (6) is equal to $2f'(2)$.

$$2f'(2) = \sum_{m=1}^{k-1} m2^m = (k-2)2^k + 2$$

Substituting $k = \lceil \log_2 n \rceil + 1$ yields:

$$\sum_{m=1}^{\lceil \log_2 n \rceil} m2^m = (\log_2 n - 1)2n + 2$$

The second summand in (6) is simply a geometric series and simplifies to:

$$\sum_{m=1}^{\lceil \log_2 n \rceil} 2^m = 2n - 2$$

Simplification of the above equation yields:

$$W = 2ne(\ln n + c) - 2e(\ln 2^{n-1} + c) \leq 2ne(\ln n + c) \quad (7)$$

Thus, comparing (7) with the expression for $E[W]$ obtained in (2), we conclude that the lack of knowledge of the number of neighbors n results in no more than a factor of two slowdown.

5.2 Collision Detection-based Neighbor Discovery Algorithm

As described in Section 5.1, we next extend the collision detection-based algorithm to the case when nodes do not have an estimate of n.

Again, we divide the execution of the algorithm into phases. Each node maintains a local variable i, initialized to 0, that denotes the number of nodes discovered so far. Each phase m lasts a duration of $2^{m+1}e$ slots. The algorithm operates as follows:

In each of the $2^{m+1}e$ slots of phase m, a node does the following:

1. In the first sub-slot, it transmits with probability $p_i = 1/(2^m - i)$ and receives with probability $1 - p_i$.
2. In the second sub-slot:
   (a) a node in receive mode in the first sub-slot deterministically transmits in the second sub-slot, if a collision occurs in the first sub-slot; else it sets $i \leftarrow i + 1$ and goes to step 1.
   (b) if a node in transmit mode in the first sub-slot detects no energy in the second sub-slot, it switches to receive mode for the rest of the process. Else, it assumes its transmission was unsuccessful and goes back to step 1.

It is easy to see that when $m = \lceil \log_2 n \rceil$, the phase lasts a total of $2ne$ time slots. For each slot in the $\lceil \log_2 n \rceil$-th phase, the probability of a successful transmission in that slot is given by:

$$p_s(i) = \frac{1}{n-i}\left(1 - \frac{1}{n-i}\right)^{n-i-1} \geq \frac{1}{(n-i)e}$$

where the inequality follows from Lemma 1. Assuming there are k nodes yet to be discovered at the beginning of phase m, we can use the analysis in Section 4.4 to conclude that

$$P(W_k > 2ne) < e^{-n(1-\frac{k}{2n})^2} \leq e^{-\frac{n}{4}}, k = 0, 1, \ldots, n$$

where the random variable W_k denotes the time required to discover the remaining k nodes, when the probability of discovering a node in a time slot is at least $1/e$. In other words, all n nodes are discovered within $\lceil \log_2 n \rceil$ phases w.h.p.

Hence, the total time W required to discover all the neighbors w.h.p is given by:

$$W = \sum_{m=1}^{\lceil \log_2 n \rceil} 2^{m+1} e = 4(n-1)e \quad (8)$$

If we assume that the our desired probability of not discovering all n neighbors within a given time is upper bounded by $e^{-\frac{n}{4}}$, we know based on the analysis in Section 4.4 that when n is known, the collision detection-based algorithm takes $2ne$ time slots to achieve this desired probability bound. From (8), we conclude that lack of knowledge of n results in no more than a factor of two slowdown.

6. ASYNCHRONOUS NEIGHBOR DISCOVERY ALGORITHMS

We next relax the assumption that time is divided into slots and that nodes are synchronized on slot boundaries. In particular, we now consider an unslotted system in which nodes operate asynchronously. We propose asynchronous versions of the discovery algorithms described earlier and show that the asynchronous algorithms are only two times slower than their synchronous counterparts. Our results are, therefore, consistent with other results [21] which observe a factor of two reduction in throughput in going from slotted to unslotted ALOHA.

For the moment, we assume that all nodes start executing neighbor discovery at the same time instant, an assumption which is relaxed in Section 7.1.

6.1 ALOHA-like Neighbor Discovery

The asynchronous ALOHA-like algorithm is based on a similar algorithm described in [22]. The algorithm operates as follows. In between successive transmissions, each node remains in receive mode for an exponentially distributed time interval with mean $1/\lambda$. Each transmission lasts a duration of τ, which is assumed to be small relative to $1/\lambda$.

By trivially extending the analysis in [22] to the case of omni-directional antennas, we can derive the optimal λ that maximizes the rate of discovery of neighbors:

$$\lambda = \frac{1}{2\tau n}$$

where n denotes the clique size.

6.1.1 Analysis For Known n

We first analyze the asynchronous ALOHA-like neighbor discovery algorithm when each node is assumed to know n. We derive the expected time, $E[W]$, required for all n nodes to be discovered.

Due to inter-transmission times being exponentially distributed, the total traffic from all the nodes constitutes a Poisson process with rate $n\lambda$. Now, a transmission from a node at time instant t is successful only if there is no other transmission for the time interval $[t-\tau, t+\tau]$. The probability p_s of a successful transmission is therefore given by:

$$p_s = e^{-2n\tau\lambda} = 1/e$$

As in the synchronous case, we assume that neighbor discovery is divided into epochs, where epoch i, of duration W_i, starts with the discovery of i-th node and ends with the discovery of the $i+1$-st node. Thus, $W = \sum_{i=0}^{n-1} W_i$, as before.

Let us restrict our attention to epoch i, in which there are $n-i$ nodes yet to be discovered. The transmissions from these $n-i$ nodes constitute a Poisson process with rate $(n-i)\lambda$, each having a probability p_s of being successful. In other words, W_i is exponential with mean $1/((n-i)\lambda p_s)$. Therefore, $E[W_i] = 2\tau ne/(n-i)$, and

$$E[W] = \sum_{i=0}^{n-1} E[W_i] = 2\tau neH_n = 2\tau ne(\ln n + \theta(1))$$

In other words, the loss of synchronization results only in a factor of two slowdown, as expected.

6.1.2 Sharp Concentration Around the Mean

As described in Section 3.4, we next show that W is sharply concentrated around its mean. Let $N_i(t)$ denote the number of successful transmissions from node i by time t. Let $Q_i(t)$ denote the total number of transmissions from node i by time t. The conditional pmf $P(N_i(t) = k | Q_i(t) = t)$ is then given as:

$$P(N_i(t) = k | Q_i(t) = m) = \binom{m}{k} p_s^k (1-p_s)^{m-k}$$

Now, $Q_i(t)$ is a Poisson random variable with rate λ. Removing the conditioning, we get

$$P(N_i(t) = k) = \sum_{m=0}^{\infty} \binom{m}{k} p_s^k (1-p_s)^{m-k} e^{-\lambda t} \frac{(\lambda t)^m}{m!}$$

Let E_i^t denote the event $\{N_i(t) = 0\}$. Therefore,

$$P(E_i^t) = \sum_{m=0}^{\infty} (1-p_s)^m e^{-\lambda t} \frac{(\lambda t)^m}{m!} = e^{-\lambda t p_s} = e^{-\frac{t}{2\tau ne}}$$

Proceeding exactly as described in Section 3.4, we get,

$$P[W > t] = 1 - P[\neg(\cup_{i=1}^{n} E_i^t)]$$

Therefore,

$$P[W > t = 2\tau ne(\ln n + c)] = 1 - e^{-ne^{-(\ln n + c)}} = 1 - e^{-e^{-c}}$$

6.1.3 Analysis For Unknown n

We now extend the asynchronous ALOHA-like algorithm to the case where n is not known to the nodes. As before, the algorithm execution is divided into phases. We assume that nodes are synchronized on phase boundaries, an assumption which will be relaxed in Section 7.1.

The algorithm operates as follows. In phase i, each node remains in the receive mode for an exponential amount of time with mean $1/\lambda_i = 2^{i+1}\tau$ between successive transmissions. Each phase i lasts for a duration of $2^{i+1}\tau e(\ln 2^i + c)$ time units.

It is easy to see that the $\lceil \log_2 n \rceil$-th phase lasts a duration of $2\tau ne(\ln n + c)$ time units. In this phase, each node transmits with rate $\lambda = 1/2\tau n$. Based on our analysis in Section 6.1.1, we know that the probability that a node remains undiscovered at the end of this phase is $1 - e^{-e^{-c}}$, which is very small for large positive constant c.

Thus, the total time W required to discover all neighbors w.h.p equals the total time until and including the $\lceil \log_2 n \rceil$-th phase and is given by:

$$W = \sum_{i=1}^{\lceil \log_2 n \rceil} 2^{i+1}\tau e(\ln 2^i + c)$$

A calculation similar to that in Section 5.1 yields:

$$W = 4\tau ne(\ln n + c) - 4\tau e(\ln 2^{n-1} + c) \quad (9)$$

Again, we observe no more than a factor of two slowdown from the case when n is known. Furthermore, comparing W in (9) with that in (7), we conclude that the asynchronous algorithm is only twice as slow as its synchronous version.

6.2 Collision Detection-based Neighbor Discovery

We next present an asynchronous version of the collision detection-based neighbor discovery algorithm. Before describing the algorithm, we illustrate the feedback mechanism in an unslotted system. As before, each message is assumed to be of duration τ and is followed by a *feedback period* of duration Δ to obtain receiver status feedback. Let $\kappa = \tau + \Delta$. Consider the timeline of an asynchronous collision detection-based algorithm, as shown in Figure 2, which

consists of (i) *Unsuccessful Busy Periods*, during which two or more nodes transmit concurrently; (ii) *Feedback Periods* immediately following a message transmission; (iii) *Idle Periods* during which no transmissions occur; and (iv) *Successful Busy Periods* during which exactly one transmission occurs. We assume that at any time instant each node in the receive mode can detect if the wireless channel is *busy* or *idle*. As in the synchronous version, the key idea here is that

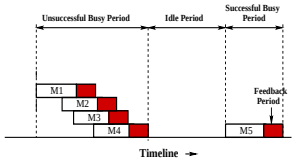

Figure 2: Asynchronous Collision Detection-based Neighbor Discovery

each node that has been successfully discovered stops transmitting, thus allowing other nodes to be discovered faster. Conversely, nodes yet to be discovered increase their transmission rate as other nodes get discovered. We start by describing our algorithm for the case when the clique size, n, is known to all nodes.

6.2.1 Known n

Let the local variable i, initialized to 0, denote the number of neighbors discovered thus far. The algorithm operates as follows.

1. Each node remains in the receive mode for an exponential duration with mean $1/\lambda_i = 2\kappa(n-i)$. While in the receive mode, a node does the following:
   (a) If a collision occurs, it deterministically transmits at the end of the busy period, after which it continues to remain in the receive mode.
   (b) If a message is successfully received and the node detects no energy until the end of the feedback period, it sets $i \leftarrow i+1$.
2. A node in the transmit mode does the following:
   (a) If its transmission begins in a busy period, it assumes that its transmission is unsuccessful.
   (b) If its transmission starts in an idle period and it detects no energy during the feedback period, its transmission was successful. The node switches to the receive mode for the remainder of the discovery process.

Based on the algorithm description, we make three important observations:

1. A message transmission at time t is successful only if no other transmission occurs within the time window $[t-\kappa, t+\kappa]$.
2. A node can begin a transmission during the transmission of another node, despite detecting a busy period. The algorithm performance can be improved by suppressing such transmissions. However, as we will see, despite transmitting in a busy period, we obtain an algorithm which is only two times slower than its synchronous version. Thus, transmission suppression can only improve this constant and not the asymptotic order.
3. Unlike its synchronous version, the asynchronous collision detection-based algorithm does not require minislots. In an unslotted system, the probability of two or more nodes transmitting simultaneously is 0. This fact and the ability of each node to distinguish between busy and idle periods allows each transmitting node to reliably detect the status of its transmission.

We are now ready to analyze the performance of the algorithm. Let W denote the time to discover all n nodes. As before, divide the discovery process into epochs, where the i-th epoch is of duration W_i. In epoch i, there are $n-i$ nodes yet to be discovered, each transmitting with rate $\lambda_i = 1/(2\kappa(n-i))$. Since the transmission events of an individual node constitute a Poisson process with rate λ_i, the transmission events from the $n-i$ yet to be discovered nodes also follow a Poisson process with rate $(n-i)\lambda_i$.

Now, the probability that a transmission by one of the $n-i$ yet to be discovered nodes is successful is given by:

$$p_s(i) = e^{-2(n-i)\kappa\lambda_i} = 1/e$$

Now, the successful transmission events from the $n-i$ yet to be discovered nodes follow a Poisson process with a rate given by $(n-i)\lambda_i p_s(i) = 1/2\kappa e$. In other words, the random variables W_i are iid and exponentially distributed with mean $2\kappa e$.

Noting that $W = \sum_{i=0}^{n-1} W_i$, it follows that the random variable W is the sum of n iid exponential random variables. Therefore, W is an n-stage Erlang random variable with mean:

$$E[W] = \sum_{i=1}^{n} E[W_i] = 2\kappa n e$$

We immediately conclude that the the asynchronous version of the collision detection-based neighbor discovery is only twice as slow as its synchronous version.

6.2.2 Bounds on Deviation From Expectation

We now bound the probability that W deviates significantly from its expectation. In particular, for any random variable W and any $t > 0$, we have the following Chernoff bound [18]:

$$P(W \geq a) \leq \inf_{t>0} e^{-ta} M_W(t) \tag{10}$$

where $M_W(t) = E[e^{tW}]$ denotes the moment generating function of the random variable W. For an n-stage Erlang random variable W with rate λ:

$$M_W(t) = \left(\frac{\lambda}{\lambda - t}\right)^n$$

Using elementary calculus, the value of t that minimizes the right hand side of (10) can be obtained as:

$$t^* = \lambda - \frac{n}{a}$$

Therefore,

$$P(W \geq a) \leq e^{-(a\lambda - n)} \left(\frac{a\lambda}{n}\right)^n$$

Setting $a = \alpha E[W] = \alpha n/\lambda$ yields:

$$P(W > \alpha E[W]) \leq \left(\frac{\alpha}{e^{\alpha-1}}\right)^n$$

When $\alpha = 2$, we get

$$P(W > 4\kappa ne) = (2/e)^n$$

which goes to 0 for large values of n.

6.2.3 Unknown n

We now consider the case when nodes have no knowledge of n. Again, we divide the algorithm execution into phases. Nodes are assumed to be synchronized on phase boundaries.

The m-th phase has a duration $2^{m+2}\kappa e$. Each node maintains a variable i that denotes the number of nodes discovered so far. In phase m, a node remains in the receive mode for an exponential duration with mean $1/\lambda_i = 2\kappa(2^m - i)$ between successive transmissions.

We see that the $\lceil \log_2 n \rceil$-th phase lasts a duration of $4\kappa ne$ time units. Let k denote the number of neighbors yet to be discovered at the beginning of the $\lceil \log_2 n \rceil$-th phase. From our analysis in Section 6.2.1, we know that the probability that it takes longer than $4\kappa ne$ to discover the remaining k nodes is:

$$P(W_k > 4\kappa ne) \leq \left(\frac{\alpha}{e^{\alpha-1}}\right)^k$$

where W_k denotes the time to discover the remaining k nodes. W_k is a k-stage Erlang random variable with mean $E[W] = 2\kappa ke$ and α is given by:

$$\alpha = \frac{4\kappa ne}{E[W_k]} = \frac{2n}{k} > 1$$

It is easy to see that $P(W_k > 4\kappa ne) \to 0$ for large n. In other words, all n nodes are discovered by the $\lceil \log_2 n \rceil$-th phase w.h.p. Hence the total time W before all nodes are discovered w.h.p is given by:

$$W = \sum_{m=1}^{\lceil \log_2 n \rceil} 2^{m+2}\kappa e = 8(n-1)\kappa e \qquad (11)$$

Thus, we observe no more than a factor two slowdown from the case when n is known. Comparing the expression for W in (11) with that in (8), we again conclude that the asynchronous algorithm is only two times slower than its synchronous version.

7. HANDLING START AND STOP TIMES

So far, we have treated neighbor discovery as if all nodes begin at the same time. We also assumed that the nodes are synchronized on the phase boundaries in the case of asynchronous algorithms. Furthermore, when nodes do not know n, we avoided the question of how nodes determine when to terminate neighbor discovery. We address each of these questions in this section.

7.1 Initiating Neighbor Discovery

It is realistic to assume that not all nodes in a wireless network are deployed at the same time instant. In fact, the deployment of the network may occur over a period of days. Since wireless nodes are battery powered, minimizing energy consumption is of utmost importance. Thus, it is appropriate to deploy the nodes in a sleep mode, in which nodes cannot send or receive messages. Suppose that the deployment of a wireless network takes place during the time interval $[t, t+\eta]$, where η is an upper bound on the deployment period and is known in advance. When nodes have access to a global clock, as in the case of a slotted, synchronous system, initiating neighbor discovery is trivial, as each node can begin execution at a globally agreed upon time instant $t' \geq t + \eta$.

In an asynchronous system, however, nodes do not have access to a global clock and furthermore, the clocks at different nodes may proceed at different rates resulting in clock offset between nodes. We assume that the maximum clock offset between any two nodes in the network is bounded by δ. In reality, clock offset between nodes can potentially grow unboundedly as clocks tick at different rates. However, the neighbor discovery occurs over much shorter time scales and therefore, it is reasonable to assume a fixed δ for the duration of the discovery phase. Each node wakes up from the sleep mode when its local clock reaches t' and starts executing the neighbor discovery algorithm. To account for clock offsets, we simply add δ time units to each phase i.e. phase i lasts for $2^{i+1}e(\ln 2^i + c) + \delta$ time units for the ALOHA-like algorithm and $2^{i+1}e + \delta$ time units for the Collision Detection-based algorithm. Thus, all nodes are simultaneously in phase i for at least $2^{i+1}e(\ln 2^i + c)$ time units in the case of ALOHA-like algorithm and $2^i e$ time units in the case of Collision Detection-based algorithm, thus guaranteeing all nodes are discovered w.h.p when $i = \lceil \log_2 n \rceil$, as desired.

To get a sense of how large δ is, we consider Mica2 motes equipped with a 32.768 kHz quartz crystal oscillator and with a real-time clock accuracy of ± 10 ppm [19]. This corresponds to an accuracy of ± 864 milliseconds per day or a maximum clock offset of 1.7 seconds per day between any two nodes. Thus, if the deployment spans a period of 3 days, δ is set to 5.1 seconds. With actively compensated oscillators [14] that provide an accuracy of $\pm$ 160 milliseconds/day, δ reduces to 1 second for the same deployment period.

7.2 Terminating Neighbor Discovery

We now discuss how nodes can terminate the neighbor discovery process, when n is unknown to nodes. We start with the ALOHA-like discovery algorithm.

7.2.1 ALOHA-like Neighbor Discovery Algorithm

We propose a probabilistic solution that allows nodes to decide at the end of each phase of algorithm execution, whether to proceed to the next phase, based on the number of distinct neighbors that successfully transmit in that phase. For ease of explanation, we consider the slotted, synchronous version of the discovery algorithm. The termination detection holds without change for the asynchronous version as well.

Termination in a Clique. We first consider the problem of termination in a clique. To simplify discussion, we initially assume nodes are synchronized on phase boundaries and relax this assumption later. Let X_j be the number of distinct nodes discovered by each node in phase j, including itself. Then the termination condition used by each node is:

TC Stop at the end of phase $j+1$ if $X_j > 2^{j-1} \wedge X_{j+1} \leq 2^j$.

Let m be the largest integer such that the clique size, $n = 2^m + k, 0 < k \leq 2^m$. We first argue that the algorithm terminates by the end of $m+2$-th phase w.h.p. The probability of successful transmission by a given node in a time slot of phase $m+1$ is given by:

$$p_s^{m+1} = \frac{1}{2^{m+1}} \left(1 - \frac{1}{2^{m+1}}\right)^{2^m+k-1}$$

Therefore,

$$p_s^{m+1} \geq \frac{1}{2^{m+1}} \left(1 - \frac{1}{2^{m+1}}\right)^{2^{m+1}-1} \geq \frac{1}{2^{m+1}e}$$

where the second inequality follows from Lemma 1 in Appendix A. Using an analysis exactly as that in Section 3, we immediately conclude that the event $\{X_{m+1} = n\}$ occurs with probability at least $e^{-e^{-c}}$. Similarly,

$$p_s^{m+2} = \frac{1}{2^{m+2}} \left(1 - \frac{1}{2^{m+2}}\right)^{2^m+k-1} \geq \frac{1}{2^{m+2}e}$$

Again, proceeding exactly as described in Section 3, we can conclude that the event $\{X_{m+2} = n\}$ occurs with probability at least $e^{-e^{-c}}$. Since the events in successive phases are independent of each other and $n > 2^m \wedge n \leq 2^{m+1}$, it follows that the event $\{X_{m+1} > 2^m \wedge X_{m+2} \leq 2^{m+1}\}$ occurs with probability at least $e^{-2e^{-c}}$, which goes to one for large positive c. In other words, the algorithm is terminated by the end of the $m+2$-th phase w.h.p.

However, it is possible that the algorithm may terminate too early i.e. in a phase $j \leq m$, before all nodes are discovered. We next derive the probability, P_e, of the algorithm terminating in a phase $j \leq m$. From Section 3.4, we know that the time between successful transmissions by each node i is exponentially distributed. Let Y_i^j be the time of first successful transmission by node i in phase j. Thus, Y_i^j is exponentially distributed with the following distribution:

$$P(Y_i^j \leq t) = 1 - e^{-tp_j}$$

where p_j is the probability of a successful transmission by node i in phase j and equals $\frac{1}{2^j}\left(1 - \frac{1}{2^j}\right)^{n-1}$. It is easy to see that the Y_i^js are iid random variables. Let the random variables $Y_{(1)}^j \leq \cdots \leq Y_{(n)}^j$ be obtained by permuting the random variables Y_i^js in increasing order. Thus, $Y_{(k)}^j$ denotes the k-th order statistic.

Let A_j denote the event $\{X_j > 2^{j-1} \wedge X_{j+1} \leq 2^j\}$. Since events in successive phases are independent of each other, we get:

$$P(A_j) = P[Y_{(2^{j-1})}^j \leq t_j] P[Y_{(2^j)}^{j+1} > t_{j+1}] \qquad (12)$$

where the distribution of the k-th order statistic is given by:

$$P[Y_{(k)}^j \leq t] = \sum_{a=k}^{n} \binom{n}{a} [P(Y_i^j \leq t)]^a [1 - P(Y_i^j \leq t)]^{n-a}$$

Therefore,

$$P_e = P(\bigcup_{j=1}^{m-1} A_j) \leq \sum_{j=1}^{m-1} P(A_j) \qquad (13)$$

where $P(A_j)$ is obtained from equation (12). We numerically evaluate the upper bound for P_e in (13) by varying n and setting the constant $c = 8$. For each n in the range $[2, 100]$, we find that $P_e \leq 10^{-5}$, which is clearly very small. We simulate cliques of sizes ranging from 2 to 100, repeating the simulation 100 times for each clique size. The simulation results confirm that each node always terminates in the $m+2$-th phase, as desired.

We finally relax the assumption that nodes are synchronized on phase boundaries. We propose that each node include in its messages the phase number it is currently in. Each node i which is currently in phase j discards any message it receives from a neighbor which is in a phase $k \neq j$. Thus, X_j is computed only from the messages transmitted during phase j, as desired.

Termination in a Multi-Hop Network. In a multi-hop network, neighboring nodes can potentially terminate at different time instants. Consider the case where a node A belongs to a smaller clique in comparison to another node B, but that they are connected to each other via an edge $A-B$. Thus, A has fewer interfering nodes compared to B and is likely to discover B before B discovers A. This might cause A to terminate (as per **TC**), before B discovers A. In order to avoid this situation, we propose the following change to the termination criterion in a multi-hop environment. We double the duration of each phase i.e. phase i lasts a duration of $2^{i+1}e(\ln 2^i + c)$. Each phase i is divided into two halves of duration $2^i e(\ln 2^i + c)$ each. In the first half, each node transmits with a probability $1/2^i$ in each time slot, as before. In the second half, however, each node transmits in the same slots as in the first half and announces to its neighbors, whether it will proceed to the next phase (as determined by **TC**). In our example, node A remains in the discovery process until it has been discovered by B and stops only after B decides to terminate the discovery process.

We simulate the ALOHA-like discovery algorithm in a multi-hop setting with the termination condition described in this section. As before, nodes are uniformly distributed in a 2D plane of size 3km × 3km. The number of nodes is varied from 200 to 4000. For each node density, the simulation is repeated 20 times, each corresponding to a different node placement. We observe that in every simulation each node terminates only after it has been discovered by all its neighbors, as desired.

7.2.2 Collision Detection-based Neighbor Discovery Algorithm

The issue of termination of neighbor discovery is trivial for the collision detection-based algorithm. When nodes are

synchronized on phase boundaries, we dedicate a single time slot at the end of each phase of algorithm execution. Each node which has not been discovered by other nodes (as determined by negative acknowledgments), deterministically broadcasts in this time slot, thereby signaling to the already discovered nodes to proceed to the next phase. Absence of a transmission in this time slot signals the end of the neighbor discovery process. In the absence of synchronization on phase boundaries, each node that has already been discovered, simply waits an additional phase duration and terminates the algorithm, if it detects no energy on the channel during the entire phase duration.

8. CONCLUSIONS

In this paper, we studied the problem of neighbor discovery in wireless networks. We proposed an $O(ne \ln n)$ ALOHA-like algorithm when nodes do not have collision detection and an $O(ne)$ algorithm when nodes can detect collisions. Unlike existing approaches, our neighbor discovery algorithms do not require nodes to have knowledge of the number of neighbors and also do not require synchronization among nodes. Furthermore, our algorithms allow nodes to begin execution at different times and also allow nodes to detect the termination of the neighbor discovery phase.

In the future, we would like to extend the analysis to a multi-hop network setting. Modeling physical layer phenomena such as fading and multipath is another interesting future direction.

Acknowledgments

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

9. REFERENCES

[1] D. Angelosante, E. Biglieri, and M. Lops. Neighbor discovery in wireless networks:a multiuser-detection approach. In *Information Theory and Applications Workshop*, pages 46–53, 2007.

[2] L. Bao and J. J. Garcia-Luna-Aceves. A new approach to channel access scheduling for ad hoc networks. In *ACM MOBICOM*, pages 210–221, 2001.

[3] A. Baptista, T. Leen, Y. Zhang, A. Chawla, D. Maier, W. Feng, W. Feng, J. Walpole, C. Silva, and J. Freire. Environmental observation and forecasting systems: Vision, challenges and successes of a prototype. *Encyclopedia of Physical Science and Technology, Academic Press*, 5(3):565–581, 2003.

[4] S. A. Borbash, A. Ephremides, and M. J. McGlynn. An asynchronous neighbor discovery algorithm for wireless sensor networks. *Ad Hoc Networks*, 5(7):998–1016, 2007.

[5] P. Dutta and D. E. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *ACM SenSys*, pages 71–84, 2008.

[6] P. Jacquet, P. Minet, P. Mühlethaler, and N. Rivierre. Priority and collision detection with active signaling - the channel access mechanism of hiperlan. *Wireless Personal Communications*, 4(1):11–25, 1997.

[7] P. Jacquet and P. Mühlethaler. Data transmission device and method for random access network having advanced collision resolution, December 1996. `http://www.freepatentsonline.com/EP0635184B1.html`.

[8] G. Jakllari, W. Luo, and S. V. Krishnamurthy. An integrated neighbor discovery and mac protocol for ad hoc networks using directional antennas. *IEEE Transactions on Wireless Communications*, 6(3):1114–1024, 2007.

[9] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[10] A. Keshavarzian and E. Uysal-Biyikoglu. Energy-efficient link assessment in wireless sensor networks. In *IEEE INFOCOM*, 2004.

[11] L. Li, J. Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer. A cone-based distributed topology-control algorithm for wireless multi-hop networks. *IEEE/ACM Transactions on Networking*, 13(1):147–159, 2005.

[12] J. Luo and D. Guo. Neighbor discovery in wireless ad hoc networks based on group testing. In *Annual Allerton Conference*, 2008.

[13] A. M. Mainwaring, D. E. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA*, pages 88–97, 2002.

[14] Maxim. 32.768khz temperature-compensated crystal oscillator. *DS32kHz Data Sheet*.

[15] M. J. McGlynn and S. A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *ACM MOBIHOC*, pages 137–145, 2001.

[16] Self-organizing neighborhood wireless mesh network. `http://research.microsoft.com/~bahl/MS\_Projects/projects.html`.

[17] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[18] R. Nelson. *Probability, Stochastic Processes and Queueing Theory*. Springer-Verlag, 1995.

[19] K. Pister and L. Doherty. Tsmp: Time synchronized mesh protocol. In *IASTED Distributed Sensor Networks*, pages 391–398, 2008.

[20] R. Ramanathan, J. Redi, C. Santivanez, D. Wiggins, and S. Polit. Ad hoc networking with directional antennas: a complete system solution. *IEEE Journal on Selected Areas in Communications*, 23:496–506, 2005.

[21] L. Roberts. Aloha packet system with and without slots and capture. *Computer Communications Review*, 1972.

[22] S. Vasudevan, J. F. Kurose, and D. F. Towsley. On neighbor discovery in wireless networks with directional antennas. In *IEEE INFOCOM*, pages 2502–2512, 2005.

[23] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 2002.

APPENDIX

A. APPROXIMATION ERROR

In this section, we calculate bounds on the approximation error for the results derived in Section 3.3 and Section 4.3. We first prove the following lemma.

LEMMA 1.

$$\left(1-\frac{1}{k}\right)^{k-1} \geq \frac{1}{e}, \forall k = 1, 2, \ldots$$

PROOF. It is easy to check that the lemma holds true when $k = 1$. For $k > 1$,

$$\left(1-\frac{1}{k}\right)^{k-1} = \frac{1}{\left(1+\frac{1}{k-1}\right)^{k-1}} \geq \frac{1}{e}$$

The last inequality follows from a well-known mathematical fact that $(1 + 1/x)^x \leq e, x \geq 1$. □

A.1 ALOHA-like Neighbor Discovery

Recall from (1), we have the following approximation:

$$p_s = p_t(1-p_t)^{n-1} = \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \approx \frac{1}{ne} \qquad (14)$$

From Lemma 1, we have:

$$p_s = \frac{1}{ne} + \delta_e(n)$$

where $\delta_e(n) > 0$ denotes the approximation error in (14). Using Taylor's Theorem, we can write

$$(n-1)\ln(1-\frac{1}{n}) = -1 + R(n)$$

where $R(n)$ represents the error term and is given by:

$$R(n) = \frac{1}{n} - \frac{n-1}{2n^2(1-c)^2}, 0 \leq c \leq 1/n$$

$R(n)$ is maximized when $c = 0$ and therefore,

$$R(n) \leq \frac{n+1}{2n^2} < \frac{1}{n}$$

Therefore,

$$0 < \delta_e(n) < \frac{1}{ne}(e^{\frac{1}{n}} - 1) \qquad (15)$$

It is easy to see that $\delta_e(n) \to 0$, as $n \to \infty$.

Recall from (2) that $E[W] = \frac{1}{p_s}H_n$. Hence, we obtain:

$$E[W] = \left(\frac{1}{\frac{1}{ne} + \delta_e(n)}\right) H_n$$

It is easy to see that $E[W] \to neH_n$, as $n \to \infty$.

A.2 Collision Detection-based Neighbor Discovery

Recall from (4) that

$$E[W] = \sum_{i=0}^{n-1} E[W_i] = \sum_{i=1}^{n} \frac{1}{\left(1-\frac{1}{i}\right)^{i-1}} \leq ne$$

We now obtain a lower bound for $E[W]$. From (15), we conclude that:

$$E[W] = \sum_{i=1}^{n} \frac{1}{\frac{1}{e} + i\delta_e(i)} = \sum_{i=1}^{n} \frac{e}{ie\delta_e(i) + 1} \geq \sum_{i=1}^{n} \frac{e}{e^{\frac{1}{i}}}$$

where $e^{-\frac{1}{i}} = 1 - \frac{1}{i} + \frac{1}{2!i^2} - \frac{1}{3!i^3} + \ldots$

It is easy to see that $\frac{1}{2!i^2} \geq \frac{1}{3!i^3} \geq \frac{1}{4!i^4} \geq \ldots$. Hence,

$$\sum_{i=1}^{n} e^{-\frac{1}{i}} \geq n - H_n + c \geq n - H_n$$

where $c \geq 0$ denotes the sum of remaining terms. Hence, we obtain

$$ne - eH_n \leq E[W] \leq ne$$

B. SHARP CONCENTRATION OF ALOHA-LIKE NEIGHBOR DISCOVERY

In this section, we provide a more rigorous proof of the sharp concentration result for the ALOHA-like neighbor discovery algorithm based on Boole-Bonferroni inequalities. The proof is very similar to the proofs available in [17] for the sharp concentration for the Coupon Collector's Problem.

LEMMA 2. *Let c be a real constant and $m = ne(\ln n + c)$ for positive integer n. Then, for any fixed positive integer k,*

$$\lim_{n\to\infty} \binom{n}{k}\left(1-\frac{k}{ne}\right)^m = \frac{e^{-ck}}{k!}$$

PROOF. We make use of the following inequality in our proof: For all $t, a \in \Re$, such that $a \geq 1$ and $|t| \leq a$,

$$e^t\left(1-\frac{t^2}{a}\right) \leq \left(1+\frac{t}{a}\right)^a \leq e^t$$

Let $t = -\frac{km}{ne}$ and $a = m$. Substituting in the above inequality, we get

$$e^{-\frac{km}{ne}}\left(1-\frac{k^2m}{n^2e^2}\right) \leq \left(1-\frac{k}{ne}\right)^m \leq e^{-\frac{km}{ne}}$$

$$e^{-\frac{mk}{ne}}\left(1-\frac{k(\ln n + c)}{ne}\right)^{\frac{mk}{ne}} \leq \left(1-\frac{k}{ne}\right) \leq e^{-\frac{km}{ne}}$$

Note $e^{-\frac{km}{ne}} = n^{-k}e^{-ck}$. Also,

$$\lim_{n\to\infty}\left(1-\frac{k(\ln n + c)}{ne}\right) = 1$$

For large n, we know that

$$\binom{n}{k} \sim \frac{n^k}{k!}$$

Putting all this together yields the desired result. □

THEOREM 1. *Let W denote the time required to discover all the n neighbors. Then, for any constant $c \in \Re$ and $m = ne(\ln n + c)$,*

$$\lim_{n\to\infty} P[W > m] = 1 - e^{-e^{-c}}$$

PROOF. The proof is exactly the same as described in [17] and has been reproduced here for completeness. The event $\{W > m\} = \bigcup_{i=1}^{n} E_i^m$, where E_i^m denotes the event that a node i is not discovered within m time slots. By the Principle of Inclusion-Exclusion, we have

$$P[\bigcup_{i=1}^{n} E_i^m] = \sum_{k=1}^{n} (-1)^{k+1} P_k^n$$

where

$$P_k^n = \sum_{1 \le i_1 \ldots \le i_k \le n} P[\bigcap_{j=1}^{k} E_{i_j}^m]$$

Let $S_k^n = P_1^n - P_2^n + \ldots + (-1)^{k+1} P_k^n$ denote the partial sum formed by the first k terms in this series. By the Boole-Bonferroni inequalities, for odd $k \ge 1$,

$$P[\bigcup_{i} E_i^m] \le \sum_{j=1}^{k} (-1)^{j+1} P_j^n$$

and for even $k \ge 2$,

$$P[\bigcup_{i} E_i^m] \ge \sum_{j=1}^{k} (-1)^{j+1} P_j^n$$

Putting the two Boole-Bonferroni inequalities together, we can write

$$S_{2k}^n \le P[\bigcup_{i} E_i^m] \le S_{2k+1}^n$$

Since all k-wise intersections of E_i^m are equally likely,

$$P_k^n = \binom{n}{k} P[\bigcap_{i=1}^{k} E_i^m]$$

Now, the probability of intersection of the k events $E_1^m, \ldots, E_k^m$ is the probability of not discovering any of the first k nodes in m time slots and is equal to $\left(1 - \frac{k}{ne}\right)^m$. Therefore,

$$P_k^n = \binom{n}{k} \left(1 - \frac{k}{ne}\right)^m$$

Therefore,

$$\lim_{n \to \infty} P_k^n = P_k = \frac{e^{-ck}}{k!}$$

Let $S_k = \sum_{j=1}^{k} (-1)^{j+1} P_j = \sum_{j=1}^{k} (-1)^{j+1} \frac{e^{-cj}}{j!}$. Note that the right hand side of the expression for S_k consists of the first k terms of the power series expansion of $f(c) = 1 - e^{-e^{-c}}$. We conclude that

$$\lim_{k \to \infty} S_k = f(c)$$

That is for all $\epsilon > 0$, there exists a $k^* > 0$, such that for any $k > k^*$,

$$|S_k - f(c)| < \epsilon$$

Since $\lim_{n \to \infty} P_k^n = P_k$, it follows that $\lim_{n \to \infty} S_k^n = S_k$. Equivalently, for all $\epsilon > 0$ and k, when n is sufficiently large, $|S_k^n - S_k| < \epsilon$. Thus, for all $\epsilon > 0$, any fixed $k > k^*$, and n sufficiently large, $|S_k^n - S_k| < \epsilon$ and $|S_k - f(c)| < \epsilon$. Therefore,

$$|S_k^n - f(c)| < 2\epsilon$$

and

$$|S_{2k}^n - S_{2k+1}^n| < 4\epsilon$$

$$|P[\bigcup_{i} E_i^m] - f(c)| < 4\epsilon$$

This implies the desired result that

$$\lim_{n \to \infty} P[\bigcup_{i} E_i^m] = f(c) = 1 - e^{-e^{-c}}$$

□