Searching strategy for multi-target discovery in wireless networks

Zhao Cheng, Wendi B. Heinzelman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, NY 14627
(585) 275-{8078, 4053}
{zhcheng, wheinzel}@ece.rochester.edu

***Abstract*—In this paper, we address a fundamental problem concerning the optimal searching strategy in terms of searching cost for the multi-target discovery problem in wireless networks. In order to find the nearest k targets from a total of m members with the least cost, how many searching attempts should we use, and how large should each searching area be? After providing the applications that motivate our research, we model the problem and derive a general formula for the expected cost as a function of the parameters of the number of searching attempts n and the searching area for each attempt, A_i. Based on this formula, we propose several algorithms to determine the optimal parameters to achieve the minimal cost, either pre-calculated or performed online. Using the optimal parameters derived from analysis, we experiment with these algorithms on general wireless network scenarios. The results show that our algorithms perform consistently close to optimal, and they exhibit much better performance than other heuristic schemes. The desired performance is achieved by adapting the searching radius to estimates of network parameters such as the total number of nodes and the total number of targets.**

I. Introduction

The target discovery process exists extensively in applications of nowadays fast growing wireless ad hoc networks and sensor networks. Usually, query packets are propagated inside the network to search for the targets. The target nodes will respond upon receiving the query packets. Unlike most unicast traffic, the query process usually includes a flooding process. Thus, it is crucial to determine the best way to search for a target to minimize the searching cost. More specifically, the question is: what are the number of searching attempts and the searching area for each attempt for the optimal searching strategy?

Of course, the simplest searching strategy is to search the entire interested area only once. Some other heuristic solutions are proposed and implemented as well. In DSR [1], the one-hop neighbors are first queried and the entire area is searched if the target is not among the one-hop neighbors. In AODV [2], an exponential expansion ring (EXP) scheme is applied, which is to start searching from one hop and increase the searching radius exponentially upon each failure.

Depending on the number of targets available, the target discovery problem can be divided into single-target discovery and multi-target discovery. The single-target discovery process is oriented for unique information such as the node ID in routing protocols [1], [2] or a unique service provided by a specific service provider [3], [4]. Previously, we showed that for single-target discovery, the expansion ring scheme cannot reduce the expected searching cost [7]. Instead, it increases both the cost and the latency dramatically. We also showed that the cost saving of even the optimal scheme is negligible, and the simplest searching scheme, which is to search the entire area only once, actually is the best scheme from both the cost and the latency's perspective. The reason that using multiple searches does not reduce cost is that the cost saving from a successful search in the local area cannot cover the extra cost when the local search fails since in wireless networks, the previously searched area has to be covered again for the new query to reach the farther area.

In this paper, we study a more general case, the multi-target discovery problem. Unlike the single-target discovery problem, using multiple searching attempts with increasing searching areas can reduce the cost for the multi-target discovery problem. The chance of finding a target in the local area increases as the total number of targets increases, and this increase is exponentially related to the total number of available targets. Thus, the cost saving from the local searching may eventually cover the extra cost from the possible failure of the local search.

Determining the optimal scheme for the multi-target discovery problem is crucial for applications in large wireless networks, especially those whose components are battery-supplied and power sensitive. For some applications, k servers must be found in order to function. For example, in NTP (Network Time Protocol) [5], the three closest servers are needed to synchronize a node's clock. In sensor networks, a node may need to find out the hop-distance to the nearest k anchors in order to perform location estimation [11]. Also in sensor networks, a mobile host may need to collect, say 20, temperature samples from the nearby sensors to have an accurate overview of the local temperature situation. There may be some other applications that perform multi-target discovery in order to distribute the load evenly among the network. For example, in a peer-to-peer file sharing network [9], a peer may locate a number of nearby peers and distribute the load among them. Another example is to discover an ensemble of special nodes nearby to distribute the computa-

tion among them. Also, multi-target discovery may be intentionally performed for robustness. A simple example is to locate more service providers than necessary. When the primary service provider cannot function well, there will be some backup to take the place to avoid interruption without initializing another search. For security sensitive applications such as NTP [5] and NIS (Network Information System) [6], multiple-target discovery is almost a necessity, both for security and robustness concerns.

Despite the extensive existence and importance of the multi-target discovery problem in wireless networks, the study of this field is almost non-existent. The schemes being used are merely from intuition without analytical support. To the best of our knowledge, this paper is the first formal study undertaken to generalize the problem and solve it both analytically and experimentally.

The rest of this paper is organized as follows. Section II provides an overview on the previous efforts in reducing discovery overhead and some other related work. Section III models the multi-target discovery problem in an infinite network and proposes several algorithms to determine the optimal number of searching attempts and the searching area of each attempt. In Section IV, we turn to realistic small-scale networks and illustrate how our previous analysis and algorithms can be applied into these scenarios. Extensive simulations are performed to compare our algorithms with existing schemes. Section V concludes the paper and discusses potential future work.

II. Related work

In [7], we analyzed the single-target discovery problem and showed that searching the entire area only once is actually the best scheme in terms of both cost and latency. In this work, we will study the multi-target discovery problem, which can be seen as an extension to the single-target discovery problem.

The multi-target discovery problem can be further divided into two branches. The first branch is to find at least 1 target from a total of m targets. The most common use of the this one-out-of-m discovery is in routing protocol implementations. Typical examples are DSR [1] and AODV [2]. Although the target is a specific node ID, there may be caches among the other nodes and the searching becomes a multi-target problem. Also, the searching schemes adopted by DSR and AODV, especially the expansion ring scheme, can be used for comparison with our approaches.

The other branch is a more general case, which is to find at least k targets from m members. The k-out-of-m multi-target discovery problem also has extensive applications, as we mentioned earlier. Examples that require a mandatory multi-target discovery are NTP [5], ITTC (Intrusion Tolerance via Threshold Cryptography) [8], sensor localization [11], and sensor information collecting [12]. Examples that require a multi-target discovery for robustness are NIS, NTP and any application requiring auxiliary backups. Examples that require a multi-target discovery for load distribution are peer-to-peer systems [9] and distributed computing systems [10]. Depending on various application requirements, different portions out of the total targets are to be found. For NTP, only three servers are required. For temperature monitoring sensor networks, quite a few sensors are required. For peer-to-peer systems or distributed computation systems, as many as possible peers are usually preferred.

In [13], the concepts of anycast and manycast are introduced to ad hoc networks. The so-called anycast is close to our 1-out-of-m problem in nature, and the manycast is comparable to our k-out-of-m problem. Although the authors propose several mechanisms to perform manycast delivery, their primary goal is to provide an investigation on the trade-offs of these mechanisms between performance, reliability and ease of implementation. In this paper, we only focus on the discovery phase, and we provide optimal solutions based on analysis.

III. Multi-target discovery in infinite networks: modeling and algorithms

A. Problem modeling, assumptions and terminology

Without loss of generality, we assume a large number of nodes are placed randomly and independently in a two-dimensional space $\mathbb{R}^2$. A source node wants to find at least one target within a unit area of interest. Suppose that m targets are distributed uniformly within this unit area. What is the optimal scheme to search this unit area to have the minimum cost? In other words, how many searching attempts n should be performed and what should be the searching area set $\mathcal{A}^{(n)} = \{A_1, A_2, \cdots, A_n\}$ for these n searching attempts?

Using this model, every searching strategy, including those mentioned earlier, can be exclusively expressed by $\mathcal{A}^{(n)}$. For example, for the simplest searching strategy, which is to search the entire interested area only once, it is $\mathcal{A}^{(1)} = \{1\}$. For DSR's searching strategy, which is to query the one-hop neighbors first and then search the entire area, it is $\mathcal{A}^{(2)} = \{\frac{1}{M^2}, 1\}$ if we denote M as the maximum hop limit allowed. For the exponential expansion ring scheme applied in AODV, the parameter set becomes $\mathcal{A}^{(\lceil \log_2(M) \rceil + 1)} = \{\frac{1}{M^2}, \frac{2^2}{M^2}, \frac{4^2}{M^2}, \cdots, \frac{(2^{\lceil \log_2(M) \rceil - 1})^2}{M^2}, 1\}$ if we assume that the searching area is on the order of the searching hop squared.

Here, we define the cost as the total area that has been searched. This general assumption does not contradict the traditional cost definition as the number of transmissions. In wireless networks, a node usually needs to forward packets for other nodes, and in order to search a certain area, the nodes within this area have to forward the queries. Thus, the number of query transmissions to search an area of A is proportional to A by a constant coefficient determined by the forwarding mechanism such as flooding and gossiping. Also, by defining the cost directly as the searching area, we minimize the number of variables and simplify our analysis without loss of generality. The conclusions drawn from this definition can be specified for different applications simply by mapping the area to realistic application parameters.

Also, we ignore the potential increase of the packet length and the cost it brings during packet propagation. For simplicity, we also ignore potential packet collisions, which can be effectively decreased by inserting a random delay time before for-

TABLE I

NOTATIONS USED THROUGHOUT THIS PAPER.

m	the total number of targets
k	the number of targets to be found
n	the number of attempts performed
C^n	cost of an n-ring scheme
D	cost difference between two schemes
A_i	searching area of the ith attempt
$\mathcal{A}^{(n)}$	optimal searching set for n-ring search

warding. Routing protocols are not needed for target discovery since broadcasted query packets will set up paths towards the source node that reply packets can follow.

During our analysis, we assume we are studying a snapshot of the network and nodes are static during the analysis. However, even if nodes are mobile, there are several reasons that our analysis is still valid. First, the flooding search time is short and nodes will not move too far away. Second, since nodes are moving randomly and independently, the number of nodes in a certain region is stable and will not have adverse effects on our analysis.

The model we are going to use in this section is based on the assumption that the source node is at the center of the searching area and the searching areas are concentric circles within the unit area as shown in Fig. 1. This assumption is valid for infinite networks, or practical large-scale networks. This simplified model expedites our current analysis and is easy to extend for realistic small-scale networks, as we will illustrate in Section IV.

We also assume that the targets are uniformly distributed throughout the searching area. For applications such as sensor networks, even if some sensors run out of energy, this general assumption should still be valid. This is because for a good sensor network architecture, sensors should deplete their energy in a balanced and uniform manner. However, there do exist some scenarios where the target distribution is non-uniform. We will discuss these non-uniform target distributions in Section V.

For quick reference, we use the term *n-ring* as a strategy that nodes attempt at most n times to discover the targets. Other notations are listed in table I.

B. Finding 1 out of m targets

Let us first look at the simplest case, finding only one target out of a total of m targets. Let us restate this 1-out-of-m problem briefly. Now, there are m targets distributed randomly and uniformly in the unit area. The source node located at the center wants to find at least one target from these m targets with the least cost by using the optimal n searching attempts.

1) A two-ring approach: Suppose a two-ring approach is applied, and for the first searching attempt, the searching area is A_1. For the second searching attempt, the searching area A_2 is, of course, the entire area and hence equals 1. As long as not all the m targets are located outside the A_1 area, the target will be found within the first attempt. Therefore, the probability P_1 to discover at least one target in the first attempt and the cost for the first searching attempt are

$$P_1 = 1 - (1 - A_1)^m, \qquad C_1 = A_1 \tag{1}$$

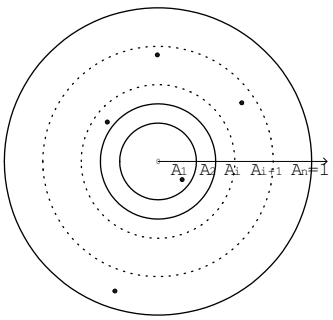

Fig. 1. The simplified model of the multiple target discovery. The searching areas are concentric circles. Both the target nodes (black dots) and non-target nodes (not shown) are uniformly distributed in the searching area.

However, if the first attempt fails, another search has to be performed, and the total searching cost for these two searches C_2 is

$$C_2 = A_1 + A_2 = A_1 + 1 \tag{2}$$

Note that if a second search needs to be performed, the total cost is not only just the second searching area, but includes the cost from the previous failed searching attempt.

If a second search is required, it means that all the m targets are located in the second ring outside the A_1 area, and the probability P_2 for this case to happen is

$$P_2 = (1 - A_1)^m \tag{3}$$

Thus, the expected cost C^2 for a two-ring scheme to complete the 1-out-of-m target discovery is

$$\begin{aligned} C^2 &= P_1 C_1 + P_2 C_2 = (1 - (1 - A_1)^m) A_1 + (1 - A_1)^m (A_1 + 1) \\ &= A_1 + (1 - A_1)^m \end{aligned} \tag{4}$$

It is easy to determine the minimum C^2 for $A_1 \in [0, 1]$ by solving $\frac{\partial C^2}{\partial A_1} = 0$, which results in

$$A_1 = 1 - m^{-\frac{1}{m-1}} \tag{5}$$

2) An n-ring approach: To aid the expression, let us define a virtual 0th attempt search for the area of $A_0 = 0$. If the ith search attempt succeeds, the total cost C_i is simply the cost summation of the first i attempts

$$C_i = \sum_{j=1}^{i} A_j \tag{6}$$

Similarly, in order to perform an ith search attempt and complete the task, there must be no targets in the area A_{i-1} and there must be at least one target in the area A_i. Thus, the probability P_i for the task to be completed in the ith attempt is

$$P_i = (1 - A_{i-1})^m - (1 - A_i)^m \tag{7}$$

Therefore, the expected cost C^n for a general n-ring searching approach is

$$\begin{aligned} C^n &= \sum_{i=1}^{n} P_i C_i = \sum_{i=1}^{n}((1-A_{i-1})^m - (1-A_i)^m))(\sum_{j=1}^{i} A_j) \\ &= \sum_{i=0}^{n-1} A_{i+1}(1-A_i)^m \end{aligned} \tag{8}$$

The final equality above can be easily proven through mathematical induction. Due to space constraints, we skip the intermediate steps.

C. Finding k out of m targets

Now, we can easily extend the study to a general case of finding at least k targets out of a total of m targets. Again, let us start from a two-ring approach.

1) A two-ring approach: Given the first searching area A_1, the probability p_i for exactly i nodes to be located within the A_1 area is actually of binomial distribution

$$p_i = C_m^i A_1^i (1-A_1)^{m-i} \tag{9}$$

In order to find at least k nodes within the first attempt, there must be greater than or equal to k nodes within the first area A_1. The probability P_1 for this case to happen is the summation of the probabilities p_i for $i \geq k$.

$$P_1 = \sum_{i=k}^{m} p_i = \sum_{i=k}^{m} C_m^i A_1^i (1-A_1)^{m-i} \tag{10}$$

Of course, the probability P_2 is the summation of the probabilities that there are less than k nodes within A_1.

$$P_2 = \sum_{i=0}^{k-1} C_m^i A_1^i (1-A_1)^{m-i} \tag{11}$$

To simplify the expression, let us first define

$$I(p;m,k) = \sum_{i=k}^{m} C_m^i p^i (1-p)^{m-i} \tag{12}$$

For a given (m,k) pair, we may further simplify $I(p;m,k)$ as $I(p)$ without causing confusion.

Eventually, we can write the cost for a two-ring searching scheme in a simpler form

$$\begin{aligned} C^2 &= P_1 C_1 + P_2 C_2 = I(A_1)A_1 + (1-I(A_1))(A_1+1) \\ &= 1 + A_1 - I(A_1) \end{aligned} \tag{13}$$

2) An n-ring approach: In order to find k targets in the ith searching attempt, there must be more than k nodes within the area A_i. Also, there must be fewer than k nodes within the area A_{i-1}, or else the search would end in the $(i-1)$th attempt. The probability P_i for the ith search to complete the searching task is

$$P_i = I(A_i) - I(A_{i-1}) \tag{14}$$

The cost of the ith search, the same as before, is

$$C_i = \sum_{j=1}^{i} A_j \tag{15}$$

Thus, we have the expected cost for a general n-ring search

$$\begin{aligned} C^n &= \sum_{i=0}^{n} P_i C_i = \sum_{i=0}^{n}((I(A_i) - I(A_{i-1})(\sum_{j=1}^{i} A_j)) \\ &= \sum_{i=0}^{n-1} A_{i+1}(1-I(A_i)) \end{aligned} \tag{16}$$

In the next section, we will propose several algorithms to determine the optimal searching area set $\mathcal{A}^{(n)}$ to minimize C^n based on equations 8 and 16.

D. Algorithms

We classify the algorithms into pre-planned algorithms and online algorithms, depending on when the parameters are determined. For pre-planned algorithms, $\mathcal{A}^{(n)}$ for various n are all calculated before the search starts. The source node will refer to these precalculated values during the searching process. For online algorithms, the source node only calculates the current searching area exactly before this search starts. Online algorithms need less computation than pre-planned algorithms, while they may perform less than optimal due to the lack of global knowledge.

1) Brute force (BF): Given n, there are $n-1$ searching area variables from A_1 to A_{n-1} (A_n is set to one). BF searches every possible $A_i \in [0,1]$ and calculates the cost based on equation 8 or 16. It picks the smallest cost as the optimal cost and the corresponding area set as the optimal solution. Despite its simplicity, this brute force technique requires excessive computation time and may be infeasible. We perform it offline just to provide a lower bound on achievable cost for the other algorithms. During realization, the interval of [0,1] for each A_i has to be discretized, and the results from this discretization may not be optimal. With a granularity of δ for each dimension A_i, the computational complexity is on the order of $(\frac{1}{\delta})^{n-1}$ for an n-ring scheme.

2) Ring-splitting (RS): Since BF cannot find the optimal solution within tolerable time, we may better focus on an alternative algorithm that is able to find good solutions using fewer computations. An intuitive solution is to insert a new searching ring between existing searching rings to reduce the cost as much as possible. We implement this idea in the Ring-splitting scheme. Suppose we already have an n-ring scheme. By inserting another searching attempt with searching area A_j between the ith attempt and the $(i+1)$th attempt, an $(n+1)$-ring scheme can be derived from the original n-ring scheme. From equation 16, the cost difference D between the old n-ring scheme and the new $(n+1)$-ring scheme is

$$\begin{aligned} D &= C^n - C^{n+1} \\ &= A_{i+1}(1-I(A_i)) - A_j(1-I(A_i)) - A_{i+1}(1-I(A_j)) \end{aligned} \tag{17}$$

RS starts from the one-ring searching scheme with $[A_0 = 0, A_1 = 1]$ and splits the ring that provides the largest cost reduction among all the possible ring splitting choices. This continues until there are no possible choices to split a ring to achieve any more cost savings. The procedure is as follows.

1) Start with the ring $[0, 1]$.
2) With an existing n-ring scheme, a given ring set of $\{[0, a_1], [a_1, a_2], \cdots, [a_{n-1}, 1]\}$ already exists. Check all these n rings and find out the candidates that can be split to further reduce the cost.
3) Terminate if there are no more candidates. Else, go to Step 4.
4) Pick the candidate that will reduce cost the most and split it. Go back to Step 2.

Whether a ring between $[A_i, A_{i+1}]$ should be split and become a candidate is a maximization problem of D and is determined as follows.

1) By solving $\frac{\partial D}{\partial A_j} = 0$, we have the potential splitting point A_j. Numerical methods for root-finding are required to find A_j.
2) First, check if A_j is within $[A_k, A_{k+1}]$. Second, check if $D(A_j)$ is larger than zero. Only when both requirements are satisfied, should A_j be a ring splitting candidate for $[A_k, A_{k+1}]$.

Since each splitting only brings two rings for calculations in the next step, if the optimal scheme is found at the n_0 ring, the total computation is only $2n_0 - 3$. The number of comparisons is $i-1$ for the i-ring scheme, so the total number of comparisons is only $\sum_{i=1}^{n_0}(i-1) = \frac{n_0(n_0-1)}{2}$.

Although RS does not guarantee the final solution set to be optimal, it reduces the computation time dramatically compared with the BF scheme. Also, it is scalable to n by providing a sub-optimal solution for all n-rings within one sequence of calculation, while BF has to calculate the solution for each n-ring scheme separately. This property makes RS more desirable for realistic implementations than scheme BF.

3) Online ring-splitting (ORS): BF and RS are pre-planned algorithms. The optimal number of searching attempts and the optimal searching area set are determined before the first search begins. ORS, instead, calculates the searching area only for the current search and only when necessary, either when the search is just beginning or the last search failed and a new search has to be performed.

In this algorithm, the source node always plans to finish the search within two attempts by splitting the remaining area. However, once it fails, it performs another splitting on the remaining searching area and performs another attempt. This process continues until the target is found, or there will be no more cost saving in splitting the remaining area.

Suppose the source node has already searched the area of S_0 and k_0 targets have been found. The new goal is to find $k - k_0$ targets from the remaining $m - k_0$ targets in the remaining $1 - S_0$ area. If the source node plans to finish the searching within two attempts by using A as the first searching area, the new cost would be

$$\begin{aligned} C_e &= I(\frac{A - S_0}{1 - S_0}; m - k_0, k - k_0))A \\ &\quad + (1 - I(\frac{A - S_0}{1 - S_0}; m - k_0, k - k_0))(A + 1) \\ &= 1 + A - I(\frac{A - S_0}{1 - S_0}; m - k_0, k - k_0) \end{aligned} \tag{18}$$

Again, some numerical methods for root-finding are required to solve $\frac{\partial C_e}{\partial A} = 0$. Also, the root $\tilde{A}$ has to pass the following two checks to provide the maximum cost saving: $\tilde{A} \in [S_0, 1]$ and $C_e < 1$. If the check fails, just use $A = 1$ to finish the last searching attempt. If not, use $\tilde{A}$ to perform the next search.

ORS is similar to RS but is performed online. Upon each failure, it only determines how to split the remaining area, although there may be some better splitting methods in the previously searched area. Thus, it performs even less than optimal compared to RS. However, it requires even less computation. There is only one computation for each additional searching attempt, and there is no wasted computation.

E. Numerical results

1) Algorithm analysis: In the previous section, we proposed several algorithms that vary in their computational requirements and their performance. Let us first determine the optimal searching area set $\mathcal{A}^{(n)}$ using these algorithms and compare the expected cost of these algorithms.

In Fig. 2, the expected costs for the solution of the 1-out-of-m problem calculated by each algorithm are shown. BF and RS have such close performance that their curves overlap with each other. ORS performs at most 5% worse than the other two algorithms. As mentioned earlier, this is because ORS is an online algorithm and lacks global knowledge. However, its performance is still very close to that of the pre-planned schemes. For the pre-planned schemes, although a different number of rings and different area parameters may be required to achieve their own optimal point (see column 3 in Table II), these algorithms perform nearly identically in terms of cost (see column 2 in Table II). The BF performance shown in Fig. 2 is on a limited brute force search on up to 4-ring schemes with a granularity of 0.001. It uses over 165 million computations to achieve the cost of 0.560852, while RS achieves a very close cost 0.567105 using only 9 computations. From this view, RS is much more practical for realistic implementations.

In Fig. 2, we also show the performance of the optimal 2-ring and 3-ring schemes. For 2-ring schemes, all the algorithms perform almost the same. For 3-ring schemes, ORS performs a little worse than the pre-planned algorithms, but it is still close. Notice that despite the fact that the real optimal solution may occur at a large value of n, the two-ring schemes have a major impact on the cost reduction compared with the 1-ring scheme whose cost is 1, but the three-ring schemes only further reduce the cost by around a trivial 2-5%. This informs us that it is very important to find a good searching area at the first attempt.

We also show the results for the k-out-of-m problem using (m,k) pairs of (6,2), (6,3), (6,4), (20,2), (20,10), (20, 18), (60,2),

TABLE II

A COMPARISON OF DIFFERENT ALGORITHMS FOR FINDING K=1 OUT OF $m = 3$ TARGETS.

Scheme	Cost	n	$\mathcal{A}^{(n)}$	Computations
BF	0.560852	4	{0.251000,0.594000,0.851000,1.0}	165,667,502
RS	0.567105	6	{0.111926,0.422650,0.746721,0.926407,0.988474,1.0}	9
ORS	0.581804	5	{0.422650,0.746721,0.926407,0.988474,1.0}	4

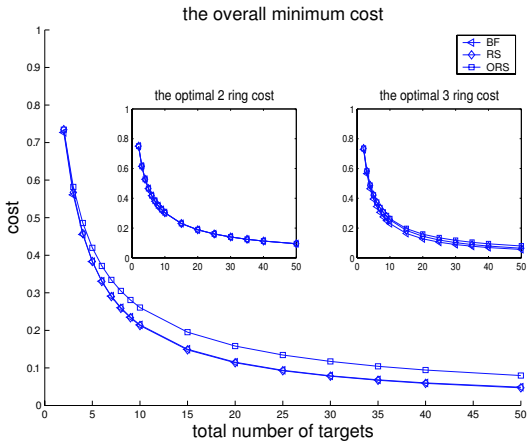

Fig. 2. The optimal expected cost of finding one target for each algorithm and the optimal 2-ring and 3-ring cost for each algorithm. The x-axis indicates the targets available in the searching area. The y-axis indicates the expected cost. Although the number of rings n to achieve the overall optimal cost is usually larger than 3, the optimal 3-ring scheme already performs very close to the real optimal.

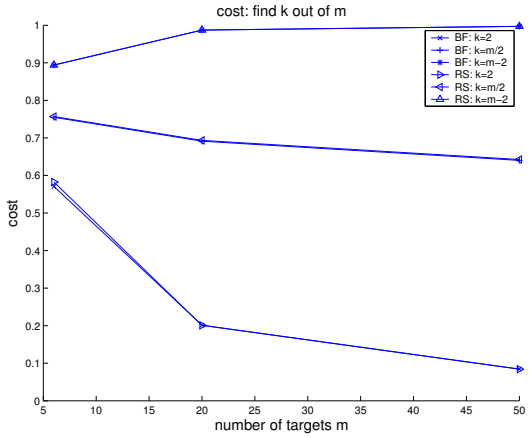

Fig. 3. The optimal expected cost for k-out-of-m discovery by different algorithms. The x-axis indicates the targets available in the searching area. The y-axis indicates the expected cost. 2, $\frac{m}{2}$,$m-2$ targets are to be searched for each algorithm.

(60,30), (60,58). By investigating the results of these discovery requests, we can have an idea of the trend of the searching cost and the searching radius for different total numbers of targets and for cases of searching few/half/most out of these targets.

Only the results from BF and RS are shown. For ORS, after finding k_0 targets, the goal of the next search becomes finding $k - k_0$ out of $m - k_0$. Therefore, the expected cost of ORS is dependent on each searching result; hence it is hard to determine analytically. The performance of ORS will be shown later through simulations.

As we can see from Fig. 3, the performance of these algorithms is still very close to each other and the curves overlap with each other. The larger the number of targets that need to be found, the less the cost can be reduced. Although the details are not shown here, the 2-ring and 3-ring schemes are still dominant in the cost reduction and more than 3-ring is unnecessary, which is the same conclusion as in the 1-out-of-m case.

In summary, we find that the two-ring RS scheme can provide close to optimal cost performance, and the three-ring RS scheme can further reduce the cost by at most 5%. More searching attempts can only reduce the cost by a negligible amount of less than 1% and are unnecessary. When only a few number of targets are to be found, or when $k << m$, the cost saving is significant. When most of the targets are to be found, the cost is close to the simple flooding searching scheme.

2) Algorithm verification: Since our algorithms are all stochastically based and our model is built over an infinite network, we experiment with these algorithms in a large-scale geography-based network to verify that the experimental results match our analytical expected cost. Also, we would like to examine how these algorithms affect the discovery latencies compared to the one-ring searching scheme, which is a missing part in our analysis. Hence, we place a large number of nodes, N_T, in a disk area randomly and independently. Each node has the same transmission range of R_t and the density is large enough for a well-connected network. The source node is located at the center of the unit area. The targets are chosen randomly from these N_T nodes, and the number of targets $m << N_T$. In this geography-based scenario, the source node controls its searching area by appending a searching radius limit on the query packet, and only nodes inside the distance limit will forward the query packet. Thus, it is assumed that nodes know about their own geographic location. In this scenario, latency is defined as the round trip distance from the source node to the target. For example, for the source node to hear the response from the border, the latency is $2 \times 1 = 2$.

We experiment on 3-ring BF, 3-ring RS and 3-ring ORS using the area set obtained from analysis for BF and RS and record their cost and latency. The cost is compared to the expected cost of the 3-ring RS scheme from analysis. In the top row of Fig. 4, we show the results of the 1-out-of-m discovery, and on

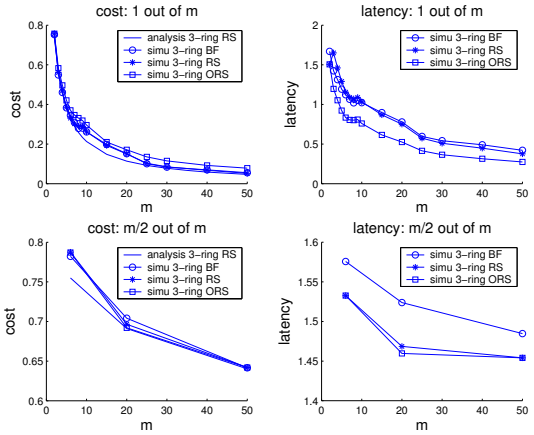

Fig. 4. The average cost and latency performance for each algorithm for geographical scenarios. The x-axis indicates the targets available in the searching area. The top row shows the results of 1-out-of-m discovery, and the bottom row shows the results of $\frac{m}{2}$-out-of-m discovery.

the bottom row, we show the results of the $\frac{m}{2}$-out-of-m discovery. In both cases, the cost of these algorithms is very close to the expected cost of 3-ring RS. This verifies our model and analysis. For latency, ORS performs a little better than the other algorithms. This is because it is more aggressive in searching a larger area and tends to take fewer attempts to complete the task. Thus, the corresponding latency tends to be smaller.

IV. Multi-target discovery in small-scale networks

Previously, we studied the target discovery problem based on a simplified model for infinite networks. In this model, we do not differentiate source nodes since source nodes are always located at the center of the interested area to be searched. While these assumptions simplify analysis and are approximately valid for large-scale networks, they do not hold true for small-scale networks. In small-scale networks, different nodes located at different positions in the network have different views of the network, especially for those nodes that are close to the network borders. Also, in small-scale networks, source nodes are more likely to search the entire network to complete the discovery task instead of searching only a part of the network. Furthermore, since hop limits are more widely applied to restrict flooding in small-scale networks than geographical limits, we must determine how to transform the analytical parameter searching area set $\mathcal{A}^{(n)}$ into hop limit values to make our results practical.

Let us remodel the problem for small-scale networks. Suppose the network is a circle of unit radius, and there are a total of N nodes and a total of m targets uniformly distributed in this network. A node wants to find k targets out of these m targets. What is the best strategy for this node if it has knowledge about its distance from the network border x_0? What is the best strategy if nodes do not know their location and have to apply a consistent system-wide searching strategy?

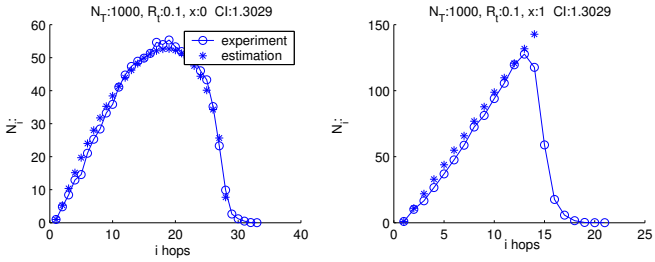

Fig. 5. The number of nodes N_{i,x_0} at i hops away from a node that is x_0 away from the border and its estimation $\tilde{N}_{i,x_0}$. The results are shown for 1000 nodes with transmission range 0.1. The left plot is for the border node $x_0 = 0$, and the right plot is for the center node $x_0 = 1$.

A. Self location aware

Although we modify our assumptions and network model to fit small-scale networks, it is interesting to find that equations 8 and 16 still hold true no matter where the source node is located. This is because we assume that both the nodes and targets are uniformly distributed in the entire area, which makes the deduction of the searching cost and the probability of finding targets follow exactly the same procedures. Therefore, the algorithms proposed based on these equations can still be applied to determine the searching areas. The only question incurred by the small-scale network model is how to utilize these calculated areas in practical applications by transforming them into hop limits to restrict flooding.

1) Mapping areas into hops: Previously, we provided detailed analysis on how to complete this mapping [7]. To avoid redundancy, we will just briefly discuss it here. The authors in [14] point out that each node has to be connected to more than $5.1774 \log N$ neighbors to have the network asymptotically connected with probability approaching one. In other words, in a well connected network, each node's transmission range R_t must be large enough to satisfy this requirement. By investigating the number of nodes at different hops from a node located x_0 from the border, we provided a method to estimate the number of nodes $\tilde{N}_{i,x_0}$ at i hops away from the source node based on the value of N and the transmission range R_t. To concentrate on our analysis, we assume that we have the estimated sequence $\tilde{N}_{i,x_0}$ for a node located at x_0. Fig. 5 shows N_{i,x_0} in a 1000-node network with $x_0 = 0$ and $x_0 = 1$, along with their estimates $\tilde{N}_{i,x_0}$.

Since the calculated area A can be seen as the percentage of nodes to be searched, we only need to find out which hop limit leads to a ratio of nodes covered close to the value of A. Therefore, each node first calculates the ratio of nodes within k hops T_{k,x_0} using $T_{k,x_0} = \frac{\sum_{j=0}^{k} \tilde{N}_{j,x_0}}{N}$. Given the calculated area A_i, the corresponding number of hop h_i can be found from [1,M] where T_{h_i,x_0} is closest to the value of A_i. M is the network diameter, which can be estimated using the methods provided in [7].

We show a mapping example in Fig. 6. In a network with a total of 1000 nodes and transmission radius of $R_t = 0.1$, a node

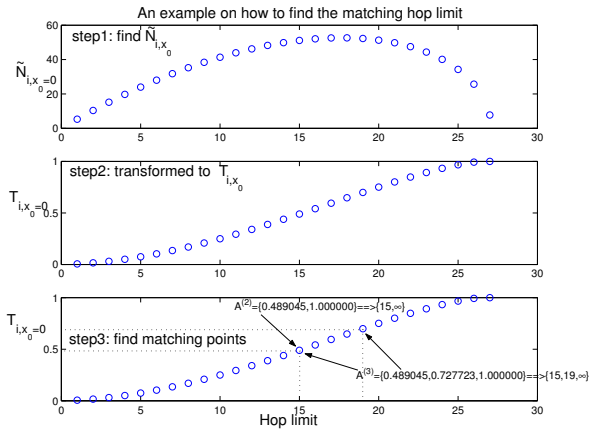

Fig. 6. An example on how to transform calculated areas into hop limits. First, N_{i,x_0} is estimated. Second, normalized T_{i,x_0} is determined. Finally, hop limits can be found by matching the areas with T_{i,x_0}.

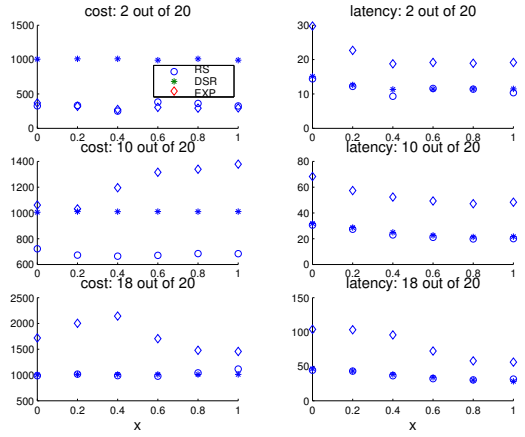

Fig. 7. Searching cost and latency comparison in small-scale networks for self-location knowledgable nodes. The x-axis indicates the source node location. RS performs consistently better than schemes DSR and EXP in terms of both cost and latency for all searching tasks.

located at the network border with $x_0 = 0$ is able to estimate the number of nodes at exactly i hops $\tilde{N}_{i,x_0}$ as shown in the first plot. Then, it calculates T_{i,x_0} as shown in the second plot. T_{i,x_0} indicates what portion of nodes are located within i hops from the source node. For a task of finding 3 targets out of 20, we have $\mathcal{A}^{(2)} = \{0.489045, 1.0\}$ for a two-ring RS scheme, and $\mathcal{A}^{(3)} = \{0.489045, 0.727723, 1.0\}$ for a 3-ring RS scheme. In the last plot, we find that $T_{15,x_0} \approx A_1 = 0.489045$ and therefore the first hop is 15 for the two-ring scheme. Similarly, $T_{19,x_0} \approx A_2 = 0.727723$, and 19 should be the second hop limit for the 3-ring searching scheme. Therefore, the hop limit set should be $\{15, \infty\}$ for the two-ring scheme and $\{15, 19, \infty\}$ for the three-ring scheme. By using ∞ as the last hop limit, we mean that any large enough integer can be used just to flood the entire network. From this example, we can see that with the estimation of N_{i,x_0}, we can easily map the calculated area set $\mathcal{A}^{(n)}$ into hop limits and apply these hop limits in small-scale networks.

2) Simulation results: Using the above mapping method, we experiment in a small network composed of $N_T = 1000$ nodes with transmission radius $R_t = 0.1$ in a unit radius circle area. We compare our two-ring RS scheme with existing schemes DSR and EXP in a scenario where nodes know their distance to the border x. DSR is actually a two-ring scheme with searching radius $\{1, M\}$. EXP uses a searching radius set $\{1, 2, 4. \cdots, M\}$. For our scheme, two-ring RS is applied due to its simplicity and its major impact in cost reduction. Fig. 7 shows that when m is large and k is small, EXP performs close to RS despite the latency being nearly twice that of RS. Although we cannot explain this phenomena analytically, one possible explanation is due to the discretization when mapping areas into hops. DSR shows a close performance to the one-ring searching, whose cost is 1000. Fig. 7 shows that for nodes at different locations, RS performs consistently low while the performance of EXP varies greatly. This is because RS chooses different hop limit values based on the source node's location to match the same optimal searching area to achieve the same optimal cost, while EXP chooses the same searching strategy for every node. Finally, scheme EXP is only suitable for searching a small number of targets in terms of cost. In addition, the excessive searching expansion procedure incurs unnecessary latency increase. For finding $\frac{m}{2}$-out-of-m targets, the EXP scheme may have a cost larger than 1000, even worse than the simple one-ring searching scheme.

B. Self location unaware

In a more general scenario, nodes do not know their positions in the network, and they have to apply the same searching strategy. The optimal searching strategy that minimizes the searching cost has to be reconsidered from the system's point of view.

We limit our scope to the two-ring RS searching scheme since the three-ring RS scheme does not bring significant cost improvement. Although this conclusion is drawn from the idealized infinite network model, we believe that it should still be valid for hop-based cases as well. Now, there is only one parameter we need to determine, the first searching hop k_{sys}.

1) Determining k_{sys}: It is easy to determine that for a uniformly distributed network, the probability that a random node is located x away from the border is

$$f_X(x) = 2(1-x) \qquad 0 \le x \le 1 \tag{19}$$

Given a sampling of x and an arbitrary k_{sys} value, we can find the corresponding searching area by calculating $A(k_{sys}, x) = T_{k_{sys},x}$ based on $\tilde{N}_{i,x}$ as in the last section. Then putting $A(k_{sys}, x)$ into equation 8 or 16, we can obtain the searching cost $C(k_{sys}, x)$ for this specific node with the searching hop limit at k_{sys}. Therefore, the system-wide expected cost that takes into account every node can be expressed as

$$C_{sys}(k_{sys}) = \int_0^1 f_X(x) C(k_{sys}, x) dx \approx \sum_{i=1}^{\frac{1}{\delta}} f_X(i\delta) C(k_{sys}, i\delta) \tag{20}$$

Here we propose our two-ring scheme based on Eq. 20. For each possible value of k less than the estimated network diameter M, we sample x from [0,1] using sampling interval δ and determine the corresponding $C(k,x)$. We then use equation 20 to calculate the system cost $C_{sys}(k)$, and determine the optimal first searching hop k_{opt} where the minimal C_{sys} is obtained.

2) Simulation results: We simulate this location unaware scenario following the above procedure. First, we need to clarify the sampling interval δ and its effects on the accuracy of the first hop limit and the computational complexity. From table III, we find that when δ decreases as the sequence $\{0.1, 0.05, 0.02, 0.01, 0.05\}$, the hop limit of the 2 out of 20 task is always 7, and the hop limit of the 10 out of 20 task is $\{14, 15, 14, 13, 13\}$, while the number of computations increases linearly with $\frac{1}{\delta}$. This informs us that although a small interval may bring about more accurate hop count calculation, the improvement is restricted since the hop limit must be chosen as an integer. We believe that the interval of 0.1 is good enough for use, and we apply $\delta = 0.1$ in the rest of our simulations. The ∞ for the 18-out-of-20 task indicates that there is no better scheme to find 18 targets more efficiently than just searching the entire area once by using a large enough hop limit.

TABLE III
THE IMPACT OF SAMPLING INTERVAL δ.

δ	Computations	First searching hop of $\{2, 10, 18\}$-out-of-20
0.1	10	$\{7, 14, \infty\}$
0.05	20	$\{7, 15, \infty\}$
0.02	50	$\{7, 14, \infty\}$
0.005	200	$\{7, 13, \infty\}$

In Fig. 8, we compare the system-wide cost and latency performance of different schemes. In RS, the first hop limit of $\{7, 14, \infty\}$ are used for finding $\{2, 10, 18\}$ out of 20 targets. Again, RS performs consistently well for all the tasks. When k is small as for the 2-out-of-20 task, EXP performs close to RS in terms of cost with a much longer latency. The estimated network diameter M, as seen from the left plot of Fig. 5, is 29, and the largest possible network diameter from statistical results is 33. Therefore, using any number larger than 33 as the first hop limit is actually a one-ring searching scheme.

3) Robustness validation: As mentioned earlier, our algorithm RS outperforms schemes DSR and EXP because it utilizes knowledge of the network parameters N and m to choose the optimal searching hop limits. The EXP scheme, on the other hand, also requires this topology information to a certain degree. First, EXP needs m to determine if the task of k-out-of-m is feasible by checking $k < m$. Then, it requires N to estimate the network diameter M so that it knows when it should stop the expansion searching. Failure to estimate M may lead to redundant attempts to flood the entire network, especially when the task cannot be completed. During the network design phase, the scale of the network is usually determined and the value of N may be roughly estimated. The information of the server numbers m can be achieved by letting each server announce its existence through broadcasting when a service is available. Due to the dynamic nature of the network, knowing the existence of a service does not explicitly help in finding a server. Therefore, the announcement can be so infrequent that the extra cost of this announcement can be seen as negligible when the number of service requests is large. Also, nodes newly joining in the network can ask their neighbors for the number of services available and have a rough idea about the services available.

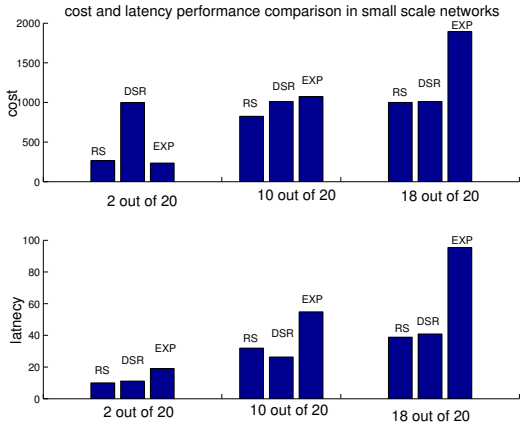

Fig. 8. Searching cost and latency comparison in small-scale networks for self-location unaware nodes. RS performs consistently close to optimal in terms of both cost and latency for all searching tasks.

Despite the fact that both RS and EXP require knowledge of N and m, RS needs it to be much more accurate. Erroneous N and m may lead to erroneous calculation of the first hop limit and thus affect the final searching cost. In this section, we will study the impact of erroneous parameters and test the robustness of our algorithms.

First, for a network of N nodes, let us define the error of N as $e_N = \frac{\tilde{N}-N}{N}$. $\tilde{N}$ is the estimated total number of nodes in the network by a specific source node. Similarly, we can define the error for the number of targets m as $e_m = \frac{\tilde{m}-m}{m}$, where $\tilde{m}$ is the estimated total number of targets in the network.

Although e_N and e_m are two different types of errors, when applying RS using these erroneous values, they eventually end up in an erroneous value of the first hop limit. For example, for the 2-out-of-20 task, the hop limits calculated based on erroneous e_N or e_m are shown in table IV.

An example of how these erroneous first hop limits affect the cost can be found in Fig. 9. Only when the error is very large, e.g., as large as $e_m = 100\%$, does the cost increase from the optimal 265 transmissions per search to 364 transmissions per search. For not so large errors, the cost will be 279 or 315 transmission, which is not so far away from the cost of the optimal 2-ring searching scheme, 265 transmissions.

For small-scale networks, similar conclusions can be drawn as in the last section. When the knowledge of N and m is accu-

TABLE IV

THE IMPACT OF ERRONEOUS N AND m ON COST.

e_N	-50%	-40%	-30%	-20%	-10%	10%	20%	30%	40%	50%
1st hop	9	9	8	8	8	8	7	7	7	7
e_m	-100%	-80%	-60%	-40%	-20%	20%	40%	60%	80%	100%
1st hop	10	9	9	8	8	8	7	7	7	7

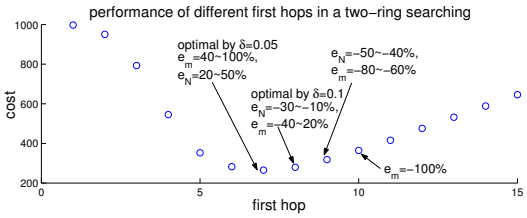

Fig. 9. The cost for all the possible first hops using a two-ring searching in small-scale networks. The x-axis indicates the first hop limit.

rate enough, RS can be applied to save cost while reducing the latency by about 50% compared to EXP. When N and m cannot be accurately estimated but the number of nodes to be found k satisfies $k << m$, EXP can be performed to reduce cost while doubling the latency. When k is close to m or when no information is known about the network topology, a simple flooding searching scheme is not bad since its latency is the smallest and it may perform even better than an arbitrary n-ring scheme. The DSR scheme shows a trivial cost improvement and a trivial latency reduction compared to the 1-ring scheme, and hence is of little practical value.

In summary, we illustrate how to apply our former analytical results to realistic hop-based small-scale networks. If a node has knowledge about its current location, it can calculate the optimal searching area $\mathcal{A}$ using RS and transform it into the desired hop limits. If nodes do not have the knowledge of their location in the network, a consistent two-ring searching strategy can be utilized for all nodes to reduce the searching cost from a system-wide perspective.

V. CONCLUSION AND DISCUSSIONS

In this paper, we studied the multi-target discovery problem in wireless networks. Through analysis, we provide several algorithms to determine the optimal searching strategies to reduce cost. We found that a 2-tier search usually performs close to optimal already and a 3-tier search can further reduce the searching by a trivial amount of around 3%. Our analysis is based on general assumptions, and the conclusions are universal to wireless networks. Simulations validate our analysis and show the practical value of our schemes in realistic scenarios.

One assumption of this paper is that nodes and targets are all uniformly distributed within the network. The limitation of this assumption is obvious in that our algorithms cannot be directly applied to non-uniform target distributions. One potential area for future work is to determine the optimal solution under the circumstances where nodes and targets are not uniformly distributed. In some scenarios, nearer areas may contain more targets of interest than farther areas. We expect that the first several searching rings should be even smaller than that in the uniform distributed network. In some other scenarios, farther areas may contain more targets of interest, e.g., caches are more likely to be around the targets and be far from the source nodes. In this case, a good strategy should be able to direct the query packets to the targets' surroundings as fast as possible. Also, the effects of more realistic wireless propagations and transmissions may be taken into account and investigated, although we expect them to have the same effects on RS and EXP.

REFERENCES

[1] D. B. Johnson and D.A.Maltz. *Mobile Computing*, Chapter Dynamic source routing in ad hoc wireless networks, pages 153-181. Kluwer Academic Publishers, Imielinski and Korth edition, 1996.

[2] C.Perkins and E.M.Royer. "Ad hoc on-demand distance vector routing" *Proceedings of IEEE WMCSA'99*, pp. 90–100, Feb. 1999.

[3] E. Woodrow and W. Heinzelman, "SPIN-IT: A Data Centric Routing Protocol for Image Retrieval in Wireless Networks," *Proc. International Conference on Image Processing (ICIP '02)*, Sep 2002.

[4] C. Intanagonwiwat and R. Govindan and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Mobile Computing and Networking*, pp. 56–67, 2000.

[5] Mills, D.L., "Network Time Protocol (Version 3)," RFC (Request For Comments) 1305, March 1992.

[6] Sun Mircorsystems. "System and Netowrk Administration," March 1990.

[7] Z. Cheng, W. Heinzelman,"Flooding strategy for target discovery in wireless networks," *Proc. of the 8th international workshop on Modeling analysis and simulation of wireless and mobile systems (MSWIM 2003)*, Sep 2003.

[8] T. Wu, M. Malkin, and D. Boneh. "Building intrusion tolerant application." *Proc. of the 8th USENIX Security Symposium*, 1999.

[9] Gnutella peer-to-peer file sharing system. http://www.gnutella.com

[10] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campell, and K. Nahrstedt. "Gaia: A middleware infrastructure to enable active spaces," *IEEE Pervasive Computing*,pp. 74–83, Oct-Dec 2002.

[11] N. Bulusu and J. Heidemann and D. Estrin. "Adaptive beacon placement," *In Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, pp. 489–498, Phoenix, Arizona, USA, April 2001.

[12] T. Oates, M. V. Nagendra Prasad and V. R. Lesser. "Cooperative Information Gathering: A Distributed Problem Solving Approach", Computer Science Technical Report 94-66-version 2 , University of Massachusetts, Amherst, MA.

[13] C. Carter, S. Yi, P. Ratanchandani, and R. Kravets. "Manycast: Exploring the Space Between Anycast and Multicast in Ad Hoc Networks," *Proc. on Mobile Computing and Networking archive Proceedings of the 9th annual international conference on Mobile computing and networking (MOBICOM)* pp. 273–285 San Diego, CA, USA, 2003.

[14] F. Xue and P. R. Kumar. "The number of neighbors needed for connectivity of wireless networks" Manuscript, 2002. Available from http://black1.csl.uiuc.edu/prkumar/postscript files.html

[15] G. Engeln-Mullges, F. Uhlig. "Numerical Algorithms with C,"Springer, 1996.

[16] L. L. Peterson, B. S. Davie. "Computer networks, a system approach," Morgan Kaufmann, 2001.