Searching strategies for 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 target discovery problem in wireless networks. In order to find the nearest k targets from a total of m members using the minimum cost, should we search the network only once, or should we apply a so-called "expansion ring scheme?" Specifically, how many searching attempts should we use, and how large should each searching area be? To answer these questions, we provide a generic model and formulate 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. Using this model, we propose several algorithms to determine the optimal parameters, either pre-calculated or performed online. We experiment with these algorithms on general wireless network scenarios and show that our algorithms perform consistently close to optimal and better 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

Information dissemination and information retrieval are the ultimate goals of wireless networks. Before information propagates within the network, the target peers, from which information is retrieved or to which information is disseminated, first need to be discovered. This target discovery problem has extensive applications in wireless ad hoc networks and sensor networks, such as route discovery in several routing protocols [1], [2], sensor discovery in wireless sensor networks [3], and service discovery in wireless ad hoc networks [4]. 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 involves a costly flooding process. Therefore, choosing a proper searching strategy is crucial in reducing the searching overhead.

The simplest searching strategy is to search the entire interested area only once. Some other more complex 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 scheme is applied, which is to start searching from one hop and increase the searching radius exponentially upon each failure. However, a comprehensive study on these searching strategies is lacking. Specifically, under what network conditions is one scheme preferred over the other schemes? Is there any other scheme that outperforms the one-hop and exponential expansion ring schemes?

The answers to these questions vary for different searching requirements. When there is only one available target, the problem becomes a single target problem. When there are multiple available targets, the problem becomes either a one-out-of-multi target problem or a multi-out-of-multi target problem, depending on the required number of targets.

All these types of target discovery exist pervasively in ad hoc networks. 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]. The single-target discovery problem can easily turn into a one-out-of-multi target discovery problem, e.g., when intermediate nodes have route caches for the required node ID, or when there are several service providers offering the same services. Multi-out-of-multi target discovery is also necessary for many applications 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 [6]. 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 through the network. For example, in a peer-to-peer file sharing network [7], 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 computation among them. Distributing data to multiple sinks is another example for sensor networks. 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) [8], multiple-target

discovery is almost a necessity, both for security and robustness concerns.

Despite the extensive existence and importance of the 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. This paper fills this gap by generalizing the problem and solving it both analytically and experimentally. Practical conclusions and suggestions on searching strategies are revealed at the end of the paper.

The rest of this paper is organized as follows. Section II provides an overview on the previous efforts in reducing query overhead for information dissemination and retrieval and some other related work. Section III models the target discovery problem 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 networks and illustrate how to employ our algorithms to these scenarios. Extensive simulations are performed to compare our algorithms with existing heuristic schemes. Section V concludes the paper with practical searching strategies.

II. Related work

The target discovery problem can be divided into two branches. The first branch is to find at least one target from a total of m targets. The most common use of the this one-out-of-multi 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. However, notice that caches are likely to provide false information when they are stale, and thus treating this problem as a general general one-out-of-multi discovery problem ignores many implementation details. For a more comprehensive searching strategy concerning the possibility of invalid route caches, the reader is referred to [9].

The other branch is a more general case, which is to find multiple targets from m members. Examples that require a mandatory multi-target discovery are NTP [5], ITTC (Intrusion Tolerance via Threshold Cryptography) [10], sensor localization [6], and sensor information collecting [11]. 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 [7] and distributed computing systems [12]. 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 order to reduce the query overhead from flooding, the performance of two alternative query propagation techniques, gossiping and random walk, are examined and compared to that of traditional broadcasting. In gossiping [13], a node forwards a query with certain probability rather than automatically forwarding every new query it receives. Gossiping is able to achieve nearly full coverage with much less overhead compared to broadcasting. To query all the nodes, only a portion of nodes need to forward the query instead of having all the nodes repeat the query as in flooding. Our model can be directly applied to both the gossiping and flooding schemes since in both schemes, the cost to query all the nodes is proportional to the total number of nodes. In random walk, a node only forwards the query to one of its neighbors instead of broadcasting to all its neighbors. It has been shown that random walk does not reduce the searching cost for a moderate coverage ratio [14]. Therefore, random walk can only be used in some particular applications rather than general searching scenarios, and we do not consider random walk in our model.

In this paper, we model the target discovery problem to reveal the general trend and applicable strategies for most searching scenarios. Due to the limitation of our model, some particular searching scenarios are not covered. First, our model deals with either proactive data dissemination schemes from data sources such as SPIN [15] or reactive data query schemes such as general ad hoc routing protocols. It does not cover certain hybrid data query schemes that combine both proactive and reactive components, such as SHARP [16], SPAN [17], ZRP [18], TTDD [19] and rumor routing [20]. These solutions usually require extra hardware such as GPS for topology setup. Also, these schemes have to find the balance point between the proactive and reactive components. This process either requires certain global knowledge about the data/query ratio, or it requires some complex adaptation schemes. Second, our model only deals with one-shot queries. Although complex queries can be divided to multiple one-shot queries and solved individually using our model, it is more efficient to handle them together within one query process. For a complete overview of data querying and the solution for complex queries, the reader is referred to [21].

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

A. Problem modeling, assumptions and terminology

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. Our question is: what is the optimal scheme to search this unit area with 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, the searching strategies mentioned earlier can be exclusively expressed by $\mathcal{A}^{(n)}$. For example, the simplest searching strategy, which is to search the entire interested area only once, can be expressed as $\mathcal{A}^{(1)} = \{1\}$. The DSR searching strategy, which is to query the one-hop neighbors first and then search the entire area, can be expressed as $\mathcal{A}^{(2)} = \{\frac{1}{M^2}, 1\}$ 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 ad hoc wireless networks, a node 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 forwarding. We also ignore packet loss from unreliable wireless links since a node fails to receive a packet only when all its neighbors' query forwarding fails. This is a very low probability event in well-connected networks. For example, with a packet loss probability of 30%, if three neighbors forward the same query, the probability for a node to fail to receive it is only $(0.3)^3 = 0.027$.

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 simplified model expedites our current analysis and is easy to extend for realistic small-scale networks, as we will illustrate in Section IV. Another assumption, that targets are uniformly distributed within the area, may be invalid for certain scenarios as well. We will discuss other possible 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 of multi-target discovery, finding only one target out of a total of m targets. The single-target problem can be seen as $m = 1$, and we will discuss it as a special case of the 1-out-of-m problem. 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.

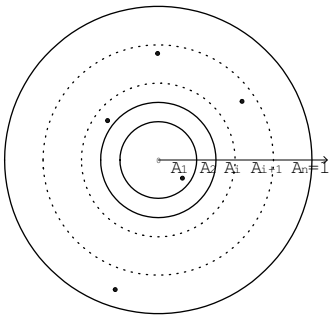

Fig. 1. The simplified model of target discovery. The searching areas are concentric circles. Both the target nodes (black dots) and non-target nodes (not shown) are uniformly distributed in the searching area.

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

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

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

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 com-

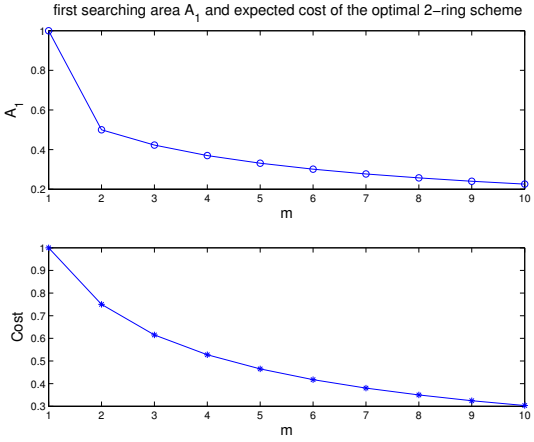

Fig. 2. 1-out-of-m, the optimal two-ring scheme. The optimal first searching area A_1 (top graph) and the corresponding cost C (bottom graph). The values vary according to the number m of existing targets. The more targets available, the smaller the first searching area and the less expected cost are.

plete the 1-out-of-m target discovery is

$$C^2 = P_1C_1 + P_2C_2 = (1-(1-A_1)^m)A_1 + (1-A_1)^m(A_1+1) = A_1 + (1-A_1)^m \tag{4}$$

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

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

In Fig. 2, we show the optimal A_1 calculated from equation 5 for different selections of m. Also, the minimum cost calculated from equation 4 for the corresponding m and A_1 is shown in the bottom figure.

From this figure, we can see that when the number of existing targets m increases, the first searching area should decrease and the expected cost decreases as well. This is obvious since when more targets are available, it is more likely to find a target by searching a smaller area, resulting in a smaller cost.

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

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

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

3) Single-target discovery: The single-target discovery, as a specific case with $m = 1$, is briefly discussed here. When $m = 1$, Equation 4 becomes

$$C^2 = A_1 + (1-A_1)^1 = 1 \tag{9}$$

The optimal cost equals 1 no matter the choice of A_1. For a general n-ring approach, it is easy to prove through mathematical induction that Eq. 8 is larger than 1 when $n > 2$. This means that if there is only one target, the cost of any two searching scheme is exactly the same as the cost of searching the entire area only once, and all the other searching schemes can only perform worse. Although specific ad hoc network cases may bring some cost saving as pointed out in [22], the cost saving is so negligible that the above conclusion drawn from the model still holds true.

C. Finding k out of m targets

Now, we can 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 a binomial distribution

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

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 $i \geq k$.

$$P_1 = \sum_{i=k}^{m} p_i = \sum_{i=k}^{m} C_m^i A_1^i (1-A_1)^{m-i} \tag{11}$$

The probability P_2 for the first attempt to fail is when there are less than k nodes within A_1.

$$P_2 = \sum_{i=0}^{k-1} C_m^i A_1^i (1-A_1)^{m-i} \tag{12}$$

To simplify the expression, we define

$$I(p;m,k) = \sum_{i=k}^{m} C_m^i p^i (1-p)^{m-i} \tag{13}$$

For a given (m, k) pair, we further simplify $I(p; m, k)$ as $I(p)$. The appendix shows some properties of the function $I(p; m, k)$.

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

$$\begin{aligned} C^2 &= P_1C_1 + P_2C_2 = I(A_1)A_1 + (1 - I(A_1))(A_1 + 1) \\ &= 1 + A_1 - I(A_1) \end{aligned} \tag{14}$$

2) An n-ring approach: In order to find k targets in the ith searching attempt, there must be more than k targets within the area A_i. Also, there must be fewer than k targets 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

$$P_i = I(A_i) - I(A_{i-1}) \tag{15}$$

The cost of the ith search, similar to the 1-out-of-m case, is

$$C_i = \sum_{j=1}^{i} A_j \tag{16}$$

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

$$\begin{aligned} C^n &= \sum_{i=0}^{n} P_iC_i = \sum_{i=0}^{n}((I(A_i) - I(A_{i-1})(\sum_{j=1}^{i} A_j)) \\ &= \sum_{i=0}^{n-1} A_{i+1}(1 - I(A_i)) \end{aligned} \tag{17}$$

Now, we have the searching cost in equations 8 and 17. In the next section, we will determine how to determine the optimal searching area set $\mathcal{A}^{(n)}$ to minimize the cost.

D. Algorithms

We will examine two types of algorithms to minimize the cost depending on when the parameters are determined: pre-planned algorithms and online algorithms. For pre-planned algorithms, the entire searching set $\mathcal{A}^{(n)}$ is calculated before the first searching attempt. The source node will refer to these precalculated values during the searching process. For online algorithms, the source node only calculates the next searching area right before the search. Online algorithms need less computation than pre-planned algorithms since they only calculate when necessary. However, 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 tries every possible combination of $A_i \in [0, 1]$ and calculates the cost based on equation 8 or 17. It picks the smallest cost as the optimal cost and the corresponding area set as the optimal solution. During implementation, the interval of [0,1] for each A_i is discretized. 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.

This scheme, although simple to implement, requires excessive computation time and becomes infeasible when n increases. Also, due to discreization, the results will only be quasi-optimal. We perform BF offline just to provide a benchmark on achievable minimal cost for the other algorithms.

2) Ring-splitting (RS): Since BF cannot find the optimal solution within tolerable time, especially when n increases and the granuality δ reduces, we attempt to find an alternative "good" algorithm with fewer computations. One solution is to insert a new searching ring between existing searching rings to reduce the cost as much as possible until there is no more cost reduction by inserting new rings. We implement this idea in the Ring-splitting scheme described as follows.

1) Start with the ring $[0, 1]$.
2) For the nth calculation, the area 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 ring candidates that can be split to further reduce the cost. (We will describe how to find out the candidates right after the procedure description.)
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.

Now, we discuss how we determine a ring candidate. Suppose we already have an n-ring scheme. An $(n+1)$-ring scheme can be derived from this n-ring scheme by inserting another searching attempt with searching area A_j between the ith attempt and the $(i+1)$th attempt. From equation 17, the cost difference D between the old n-ring scheme and the new $(n+1)$-ring scheme is

$$\begin{aligned} D &= C^n - C^{n+1} \\ &= A_{i+1}(1 - I(A_i)) - A_j(1 - I(A_i)) - A_{i+1}(1 - I(A_j)) \end{aligned} \tag{18}$$

Whether the 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 achieve the possible splitting point A_j. Numerical methods are required to find A_j.
2) In order to reduce cost by inserting A_j, the splitting point has to be within the ring and the cost difference should be larger than 0. Therefore, check if A_j is within $[A_k, A_{k+1}]$ first. Then, check if $D(A_j) > 0$. Only when both requirements are satisfied, should A_j be a ring splitting candidate for $[A_k, A_{k+1}]$.

Since each splitting only adds two rings for calculations in the next step, the total computation will be $2n_0 - 3$ if the algorithm stops at the n_0 ring. The number of comparisons is $i-1$ for the i-ring scheme, and the total number of comparisons is $\sum_{i=1}^{n_0}(i-1) = \frac{n_0(n_0-1)}{2}$, which is much fewer than that of the BF scheme $(\frac{1}{\delta})^{n_0-1}$.

Although RS does not guarantee the solution to be optimal, it reduces the computation time dramatically compared with the BF scheme. Also, while BF has to calculate for each n-ring solution separately, RS is scalable to n by providing the solution for all n-rings within one sequence of calculation. The only question remaining about RS is how its performance is compared to that of BF. We will come to this issue right

after we introduce the ORS algorithm in the next section.

3) Online ring-splitting (ORS): Both BF and RS are pre-planned algorithms. The optimal number of searching attempts and the entire searching area set are determined before the first search begins. ORS, instead, calculates the searching area only for the next search right before the search starts. In this algorithm, the source node always plans to finish the search within two attempts by splitting the remaining area. Upon its failure, ORS performs another splitting on the remaining unsearched area to find how far the next search should reach. This process continues until either the target is found, or there will be no more cost saving in splitting the remaining area.

ORS is very similar to RS. The only difference is that ORS can only split the remaining unsearched area, while RS can split any of the existing rings. This is because ORS is performed online, while RS is performed before the search starts and thus is able to do the global splitting.

Here is how ORS splits the remaining searching 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

$$\begin{aligned} C_e &= I(\frac{A-S_0}{1-S_0}; m-k_0, k-k_0))A \\ &\quad + (1-I(\frac{A-S_0}{1-S_0}; m-k_0, k-k_0))(A+1) \\ &= 1 + A - I(\frac{A-S_0}{1-S_0}; m-k_0, k-k_0) \end{aligned} \tag{19}$$

Again, some numerical methods 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. Otherwise, use $\tilde{A}$ to perform the next search.

As can be expected, ORS performs even less than optimal compared to RS since it can only split the remaining ring. However, it requires even less computation. There is only one computation for each additional searching attempt, and the computation stops as long as the searching goal is met.

E. Numerical results

1) Algorithm evaluation: This section evaluates the performance of algorithms BF, RS and ORS. We will reveal how many searching attempts are generally enough and how these algorithms perform compared to each other.

In Fig. 3, the expected costs for the solution of the 1-out-of-m problem calculated by each algorithm are shown. The X-axis indicates the total number of available targets. Let us first examine the the performance of the algorithms. 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,

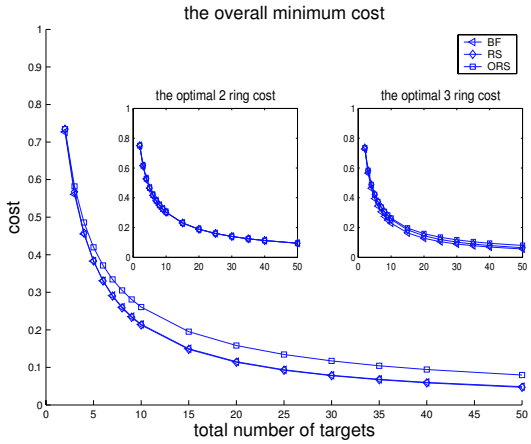

Fig. 3. The optimal expected cost of finding one target for each algorithm and the optimal 2-ring and 3-ring cost for each algorithm. The x-axis indicates the targets available in the searching area. The y-axis indicates the expected cost. Although the number of rings n to achieve the overall optimal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

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. 3 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.

Fig. 3 also reveals how many searching attempts are enough to achieve near-optimal performance. 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. Although the real optimal solution may occur at a larger 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, and 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, and more than 3 attempts are unnecessary.

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), (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 is changed to 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.

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

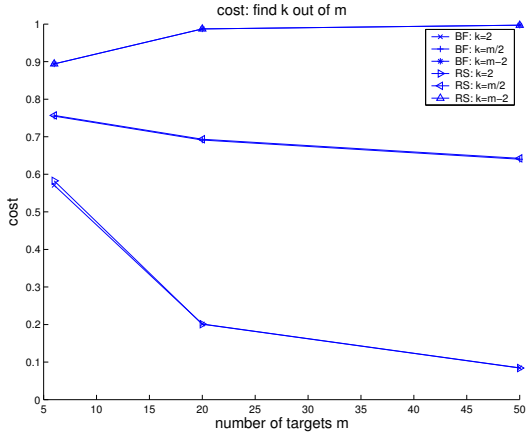

Fig. 4. The optimal expected cost for k-out-of-m discovery by different algorithms. The x-axis indicates the targets available in the searching area. The y-axis indicates the expected cost. 2, $\frac{m}{2}$,$m-2$ targets are to be searched for each algorithm.

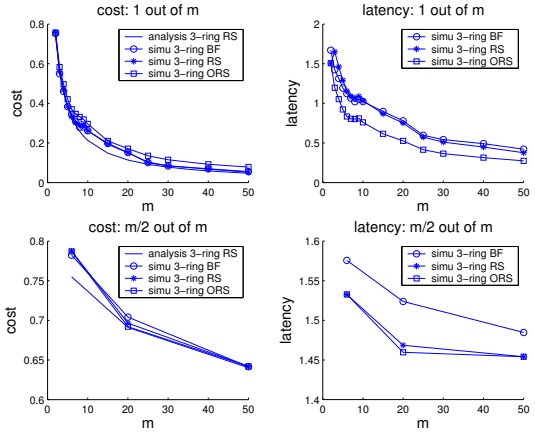

Fig. 5. The average cost and latency performance for each algorithm for geographical scenarios. The x-axis indicates the targets available in the searching area. The top row shows the results of 1-out-of-m discovery, and the bottom row shows the results of $\frac{m}{2}$-out-of-m discovery.

As we can see from Fig. 4, 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, 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) Model validation: Since our model is stochastically based, we experiment with our algorithms in a large-scale network to verify that their cost performance matches the 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 not included 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 to form 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 $m << N_T$. 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. 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 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. 5, we show the results of the 1-out-of-m discovery, and on 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 is smaller.

IV. PRACTICAL TARGET DISCOVERY IN AD HOC NETWORKS

In the previous section, we studied the target discovery problem based on a simplified model. When applying the proposed algorithms to realistic ad hoc networks, we need to resolve several other issues. First, since hop limit is generally used to restrict query flooding, we need to map the searching area set $\mathcal{A}^{(n)}$ to hop values. Second, nodes are located at different positions in the network instead of the center as assumed in our

model. We want to discover how this location variation affects our model and the corresponding area-to-hop mapping. Third, since it is more likely that nodes do not know their locations, what should be the global searching hop values to save the searching cost from the network perspective? Finally, we want to determine the robustness of our algorithms under erroneously estimated network parameters.

A. Area-to-hop mapping

A more detailed area-to-hop mapping method can be found in [22]. To avoid repetition, we will just briefly discuss it in this paper. Note that the area A in our model indicates the portion of the total number of nodes that should be queried. The problem of mapping the area to a hop value is actually to find the hop value that can be used to cover that portion of nodes. Statistics on the number of nodes at different hop distances are required to help the mapping procedure.

In [22], an empirical estimation method is proposed to determine the number of nodes at each hop distance for connected networks. Given the total number of nodes N and the transmission range R_t, the number of nodes $\tilde{N}_{i,x_0}$ at i hops away from a source node can be estimated. Since nodes at different locations have different views at the network, x_0, which indicates the distance between the node and the border, incorporates this difference.

We show an area-to-hop mapping example for a node located at the border of the network with $x_0 = 0$ in Fig. 6. The network contains 1000 nodes and the transmission radius is $R_t = 0.1$. The number of nodes at each hop distance $\tilde{N}_{i,0}$ is shown in the uppermost plot, and the total number of nodes $T_{i,0}$ within hop distance i is shown in the middle plot. After we divide $T_{i,0}$ by the total number of nodes in the network, 1000 in this case, $T_{i,0}$ indicates how much portion of nodes are within hop i of this specific node. For example, about 50% of nodes are within its first 15 hops and around 75% of nodes are within its 20 hops, as shown by the dot lines in the middle plot.

Now we can consult T_i for the area-to-hop mapping. Suppose the source node needs to find 3 out of 20 targets. Using the two-ring RS scheme, we find the area set to be $\mathcal{A}^{(2)} = \{0.489045, 1.0\}$. For the first searching area 0.489045, we check T_i at each hop value i and find that $T_{15,x_0} = 0.48923$ at hop 15 is the closest to the area value 0.489045. Thus, we choose 15 as our first search hop distance. The second hop can be chosen as any integer large enough to search the entire network. By combining the node estimation method $\tilde{N}_{i,x_0}$ from [22] and the target discovery algorithm in this paper, we can find the optimal searching hop values for each individual source node.

B. Global searching parameters

Note that given a specific target searching requirement, the area set calculated using our algorithm is always the same. The variation of node location x_0 only varies the hop choices during area-to-hop mapping by using different $\tilde{N}_{i,x_0}$ values.

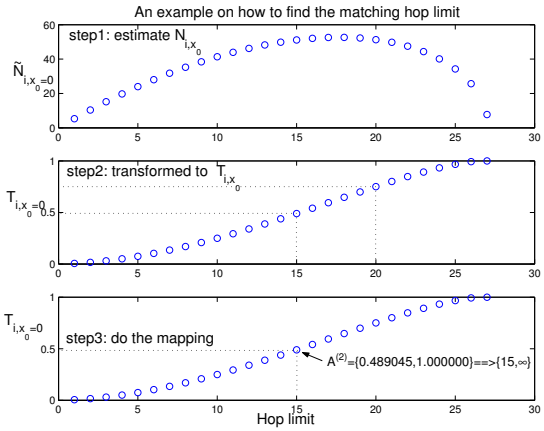

Fig. 6. An example on how to transform calculated areas into hop limits. First, N_{i,x_0} is estimated. Second, normalized T_{i,x_0} is determined. Finally, hop limits can be found by matching the areas with T_{i,x_0}.

In other words, nodes should choose different searching hop values based on their own locations.

However, in general, nodes do not know their locations in the network and thus cannot perform $\tilde{N}_{i,x_0}$ estimation. Instead of choosing their own searching hop values, all the nodes have to apply the same searching values. The global values should minimize the expected searching cost for the entire network rather than for each individual node.

We limit our scope to the two-ring RS searching scheme since the three-ring RS scheme does not bring significant cost improvement. This conclusion is drawn from the model, and we believe that it is also valid for hop-based networks. For now, there is only one parameter we need to determine: the first searching hop h_{sys}.

First, we can prove that for a uniformly distributed network, the Probability Distribution Function (pdf) of a random node location x away from the border is

$$f_X(x) = 2(1-x) \qquad 0 \le x \le 1 \tag{20}$$

Given a node located at x, we can reverse the earlier area-to-hop mapping to determine the searching area value $A(h_{sys}, x)$ for the searching hop h_{sys}.

$$A(h_{sys}, x) = T_{h_{sys},x} = \sum_{i=0}^{h_{sys}} \tilde{N}_{i,x} \tag{21}$$

Putting $A(h_{sys}, x)$ into equation 8 or 17, we can obtain the searching cost $C(h_{sys}, x)$ for the node at x using the searching hop limit h_{sys}. Considering nodes can be at any location with the pdf $f_X(x)$, the system-wide expected searching cost can be expressed as

$$C_{sys}(h_{sys}) = \int_0^1 f_X(x) C(h_{sys}, x) dx \approx \sum_{i=1}^{\frac{1}{\delta}} f_X(i\delta) C(h_{sys}, i\delta) \tag{22}$$

Here is how we determine the systematic searching hop using Eq. 22. For each possible hop value of h less than the

estimated network diameter M, we sample x from [0,1] using sampling interval δ and determine the corresponding $C(h,x)$. We then use equation 22 to calculate the system cost $C_{sys}(h)$, and determine the optimal first searching hop h_{opt} where the minimal C_{sys} is obtained. This computation only needs to be done once, and the optimal searching hop h_{opt} will be applied by all the nodes in the network.

We tested the above global first searching hop procedure in a network with 1000 nodes. We first investigate the effect of the sampling interval δ 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 $\{14, 15, 14, 13, 13\}$. Although a small interval may lead to more accurate hop count calculation, the improvement is restricted since the hop limit must be chosen as an integer. Since computation increases linearly with $\frac{1}{\delta}$, we believe that the interval of 0.1 is good enough for use, and we apply $\delta = 0.1$ for 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. 7, we compare the system-wide cost and latency performance of our scheme with that of the DSR and the EXPansion ring 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 searching tasks. When the number of targets to be found is small as for the 2-out-of-20 task, EXP performs close to RS in terms of cost with a much higher latency. The estimated network diameter is 29 in the tested scenario. Therefore, using ∞ as the first hop means to choose any number larger than 29 and search the entire network just once.

C. Robustness validation

Our algorithm RS outperforms 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 network 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 search. 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 ad hoc networks, knowing the existence of a service does not mean that nodes will know where the server is and how to reach it.

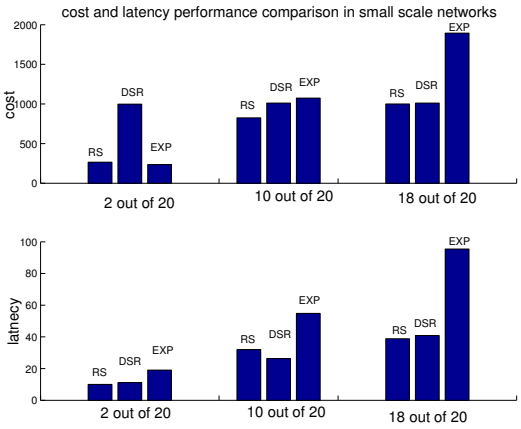

Fig. 7. Searching cost and latency comparison in small-scale networks for self-location unaware nodes. RS performs consistently close to optimal in terms of both cost and latency for all searching tasks.

Although both RS and EXP require knowledge of N and m, RS needs it to be 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. 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 both 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. 8. 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. Even so, the cost saving is still substantial. 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.

D. Choosing the right strategy

Depending on the amount of information about the network parameters and the searching task, different searching strategies should be chosen. When N and m can be accurately estimated, RS can be applied to save cost while reducing the latency by about 50% compared to EXP. When N and m cannot

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

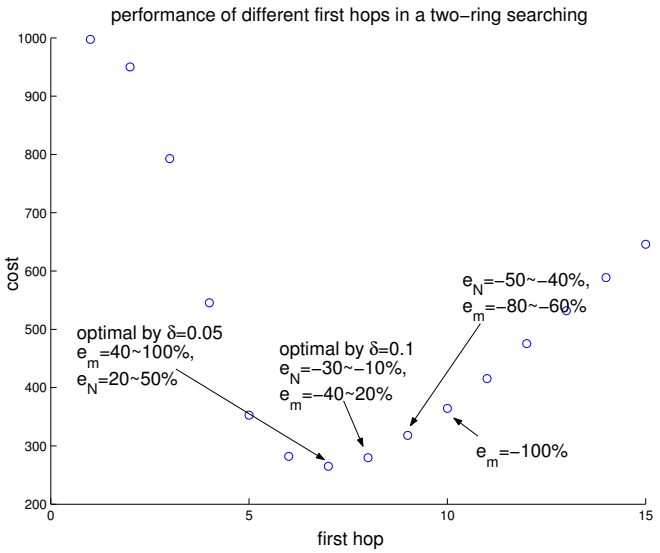

Fig. 8. The cost for all the possible first hops using a two-ring searching in small-scale networks. The x-axis indicates the first hop limit. Erroneous network parameter estimations result in erroneous hop limit choice. Substantial cost saving can still be achieved using erroneous parameters.

be accurately estimated but the number of required targets 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 best 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 degradation compared to the 1-ring scheme, and hence is of little practical value.

V. CONCLUSION

In this paper, we studied the target discovery problem in wireless networks. We model the problem and propose RS to discover the optimal searching parameters. We illustrate how to apply the model to realistic network scenarios. General searching strategies are concluded after we investigated the performance of our scheme and compared it with that of other schemes. The amount of information about the network parameters and the desired task determines the best searching scheme.

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] Chalermek Intanagonwiwat and Ramesh Govindan and Deborah Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Mobile Computing and Networking*, pp. 56–67, 2000.

[4] 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.

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

[6] 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.

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

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

[9] Z. Cheng and W. Heinzelman, "Adaptive Local Searching and Caching Strategies for on-demand routing protocols in ad hoc networks," to appear in: *Mobile and Wireless Networking of International Journal of High Performance Computing and Networking (IJHPCN)*.

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

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

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

[13] L, Li, J. Halpem, Z. J. Hass, "Gossip-based ad hoc routing," *IEEE INFOCOM* June, 2002.

[14] C. Avin and G. Ercal, "Bounds on the mixing time and partial cover of ad hoc and sensor networks," *Second European Workshop on wireless sensor networks (EWSN)*, January 2005.

[15] W. R. Heinzelman, J. Kulik, H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *Proceedings of the fifth annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 99)*, Seattle, Washington, August 1999.

[16] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, "SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks," *ACM Symposium on Mobile Ad Hoc Networking andi Computing (MobiHOC)*, 2003.

[17] B. Chen, K. Jamieson, H Balakrishnan and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *In Proc. of 7th Annual International conference on Mobile Computing and Networking*, pages 85–96. ACM, July 2001.

[18] Z. J. Haas, M. R. Pearlman and P. Samar,"The Zone Routing Protocol (ZRP) for Ad Hoc Networks," *IETF MANET Internet Draft*, July 2002.

[19] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: Two-tier Data Dissemination in Large-scale Wireless Sensor Networks," *ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on ACM MOBICOM*, 2003.

[20] D. Braginsky and D. Estrin, "Rumor Routing Algorithm For Sensor Networks,"*First Workshop on Sensor Networks and Applications (WSNA)*, September 2002.

[21] N. Sadagopan, B. Krishnamachari and A. Helmy, "Active Query Forwarding in Sensor Networks (ACQUIRE)," *Ad Hoc Networks Journal - Elsevier Science* 1st Quarter 2004.

[22] 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.

APPENDIX

A. Properties of $I(x; m, k)$

$I(x; m, k)$ is defined as

$$I(x; m, k) = \sum_{i=k}^{m} C_m^i x^i (1-x)^{m-i} \tag{23}$$

Its derivative over x is

$$\begin{aligned} I'(x; m, k) &= m(I(x; m-1, k-1) - I(x; m-1, k)) \\ &= m C_{m-1}^{k-1} x^{k-1} (1-x)^{m-k} \end{aligned} \tag{24}$$

And I'' is

$$I''(x; m, k) = m(I'(x; m-1, k-1) - I'(x; m-1, k)) \tag{25}$$

We show the I_x and I'_x for a given pair of values of m and k in Figs. 9 and 10.

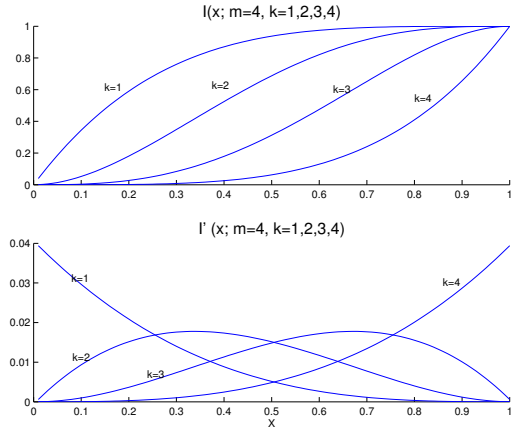

Fig. 9. The $I(x)$ function with $m = 4$ and k chosen from {1,2,3,4}. The x-axis is x, and the y-axis is $I(x)$ for the upper plot and $I'(x)$ for the lower plot.

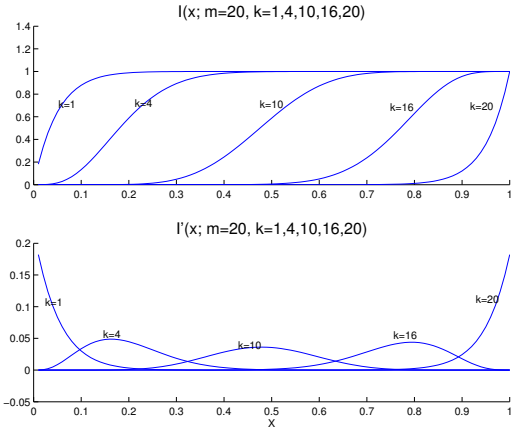

Fig. 10. The I_x function with $m = 20$ and k chosen from {1,4,10,16,20}. The x-axis is x, and the y-axis is I_x for the upper plot and I'_x for the lower one.

All the expressions of $I(x; m, k)$, I' and I'' are required for the numerical solutions in Section III.