Two-Sided Expanding Ring Search

Simon Shamoun
Bar Ilan University
Email: srshamoun@yahoo.com

David Sarne
Bar Ilan University
Email: david.sarne@gmail.com

***Abstract*—We consider a new search paradigm, in which two nodes simultaneously conduct an expanding ring search for a path between each other. The new method generalizes expanding ring search, which is a well-studied flooding-based technique for discovering paths and resources in ad hoc networks. The idea behind the two-sided search is that all nodes that receive a query cache the shortest path to the source of the query; it is therefore only necessary for one node to find a cached route to the other. A two-sided search offers ways to reduce expected costs in ways that a search by one node alone cannot, and as such it weakly dominates a one-sided search. We show how to derive the sequence of search ranges with the minimum expected total search cost in polynomial time. We further consider optimal strategies under time constraints and study various properties and benefits of the two-sided search.**

***Index Terms*—ad hoc networks; constrained optimization; expanding ring search; query and search**

I. Introduction

Ad hoc networks are typically characterized by dynamic topology changes that make it difficult to properly maintain routing tables. The amount of updates needed to avoid stale routes and routing loops would incur a significant amount of overhead. An alternative to proactive route maintenance in these networks is routing on demand [1]. In on demand routing, routes are established and maintained only when needed, thus avoiding the problems and costs associated with proactive route maintenance. At the heart of on demand routing is search by forwarding queries from one node to another. The paths the queries follow to the destinations are the paths used by the routing protocols. This process does not require knowledge of the network topology and is used in all types of ad hoc networks to find paths to nodes, data, and services [2].

Most on demand protocols use broadcast flooding to search for paths. In broadcast flooding, the search node broadcasts its query, which is then rebroadcast by all nodes that receive the query from the source node or any other node that subsequently broadcasts the query. Broadcast flooding is guaranteed to find the shortest path to the target when one exists, but it is highly inefficient. All nodes that are connected to the searcher receive, process, and broadcast the query, even if they are very far from the path from the searcher to its target. This has a measurable cost to the network, such as power and bandwidth consumption. Although there may be more efficient ways to propagate a query throughout the network, the fundamental problem remains—the query extends beyond the necessary range.

A well-studied improvement upon flooding alone is expanding ring search (ERS) [3], [2], [4], [5]. In this method, the search node assigns a query a time-to-live (TTL) value and broadcasts it to all of its neighbors. The TTL value defines the maximum number of links the query traverses along any path from the searcher. Each node that receives a query decrements the TTL value and rebroadcasts the query if the TTL value is greater than one. If the target is not found, the searcher sends a new query with a larger TTL value, thus expanding the search extent. The underlying paradigm—controlled flooding—is common to many search techniques. When designed properly, the expected costs of such techniques are less than or equal to the cost of full flooding. The advantage of expanding ring search is the simplicity of implementation [6], but it has its own fundamental shortcoming. Each new query repeats the costs of the previous query, since it must traverse the same links before it can reach the further nodes. The cost of increasing the search extent sometimes renders ERS no more efficient than full flooding alone [5].

In this paper, we propose a two-sided expanding ring search, which addresses the shortcomings of ERS in significant ways. The idea behind the two-sided expanding ring search is that each node conducts an expanding ring search for the other, and all intermediate nodes that receive a query cache the route back to the source of the query. This way, it is only necessary for one node to find an intermediate node with a cached path back to the other node. In other words, the path found is composed of the subpath found by each node. The expected cost of finding each subpath can be significantly lower than the expected cost of the finding the full path, since the cost of increasing the search extent is lower for shorter distances than longer distances. This concept is depicted in Figure 1. In such a case, the expected cost of a two-sided search is lower than the expected cost of a one-sided search. Moreover, a two-sided search is more efficient than full flooding alone even in some cases that a one-sided search is not.

We also study two-sided search under time constraints. One might suppose that a two-sided search is faster than a one-sided search since both nodes can search at the same time. However, if speed alone was a concern, then the fastest way to search is to fully flood the network and not to use an expanding ring search. Therefore, a discussion of speed must be coupled with a discussion of efficiency [7]. Here we see even more versatility of the two-sided search. First, costs can be reduced when both nodes search at the same time, which is not so when there are no time constraints. Furthermore, costs

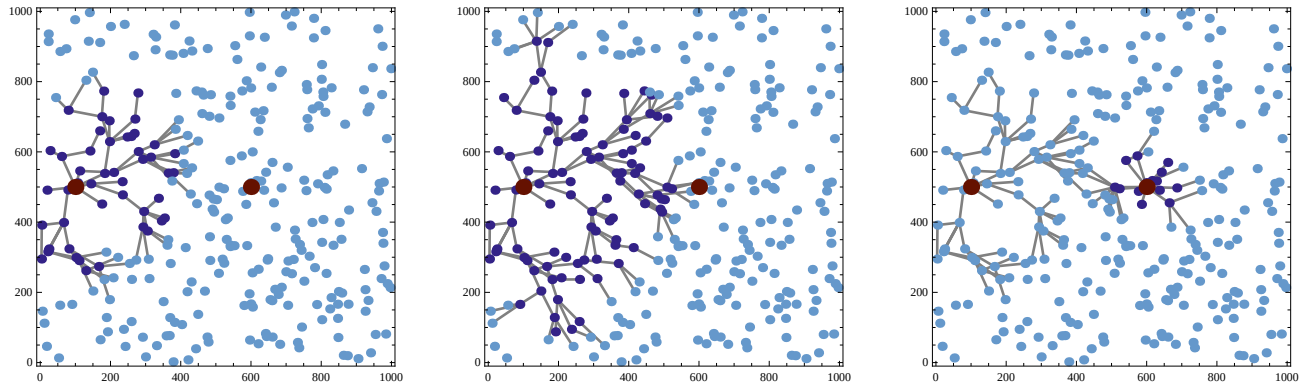

Fig. 1. Depiction of a one-sided and two-sided expanding ring search. The three figures depict a network of 300 nodes, with the two nodes searching for each other enlarged and highlighted in red, the transmitting nodes highlighted in dark blue, and the links traversed in gray. The transmission radii of all nodes are $100m$. In the leftmost figure, the leftmost node sends a query with a TTL value of 5. The middle figure shows the results when the TTL value is increased to 7. The rightmost figure shows the result when the rightmost node instead queries with TTL value 2, which is equivalent to searching for a path of length 7 from both nodes. The difference in the number of blue nodes between the middle and rightmost figures is the reduction in the number of transmitting nodes when using a two-sided search.

can be reduced even when one node begins a query before it completes a prior query, which is not so in the one-sided search.

The two-sided search is useful when both nodes have a mutual interest in contacting each other. This occurs, for example, when sensors and other users need to regularly upload data to a base station and report to one another or when they need to reestablish a lost connection. Since both parties are seeking a connection, they can search for one another at fixed times.

The main contributions of this paper are polynomial-time methods to derive search strategies that minimize the expected total search costs over both nodes and an analysis of certain properties of the two-sided search. Deriving the strategy for the unconstrained case is a matter of determining the order in which the nodes broadcast their queries and with which TTL values. The time-constrained case is more complex, since the search can be made more efficient when several queries are actively being flooded at a time. In this case we present a heuristic for low cost solutions that can also be derived in polynomial time.

The rest of the paper is organized as follows. In Section II, we review related work. In Section III, we define the notation used in the paper and the assumptions made. We show how to derive the optimal unconstrained strategy in Section IV and the time-constrained heuristic in Section V. We conclude the paper in Section VI.

II. Related Work

A significant amount of work has been dedicated to optimizing expanding ring search sequences [8], [9], [3], [10], [4], [5]. It is best optimized when the probability distribution of the minimum number hops between nodes and the cost of searching with each TTL value is known *a priori*. Most analysis assumes that all nodes are distributed uniformly at random throughout the field [8], [3], [10]. Chang and Liu [5] show how to derive the optimal TTL sequence for arbitrary cost functions and hop count distributions. When the probability distribution is not known in advance but the cost function is, the best worst-case approximation factor of any deterministic strategy and randomized strategy are 4 [4] and e [9], respectively, and these bounds are tight. Search latency was analyzed in [3], [10]. Chang and Liu [7] analyzed strategies with the best worst-case approximation factors of expected search costs under expected time constraints.

Popular methods designed to avoid the inherent redundancy of ERS are Blocking Expanding Ring Search (BERS) and its variations [11], [12], [13]. In these so-called "stopping broadcast methods", the searcher sends a route request only once, and a chase packet is sent when a route is found in order to terminate the search. Each node delays forwarding the request to allow enough time to receive the chase packet. Each method differs in the waiting time and how the chase packets are sent, with tradeoffs in latency and cost. These methods do not guarantee lower expected costs than ERS because of the cost of the cancellation flood. Other flooding-based methods limit queries to geographical areas [14], [15] and along time and distance gradients [16], but these methods do not guarantee improvements over ERS [6].

The two-sided search has some similarities to route caching [17], data replication [2], and directed diffusion [18]. In the first case, routes are cached for a limited amount of time, so nodes can discover cached routes to the destination and save on search costs and time. This is similar to the two-sided search since the search is for a cached route, but it is still effectively a one-sided search. In the second case, data is replicated in peer-to-peer and sensor networks so nodes only need to find a cached instance of that data. This is more like the two-sided search than the first case since the source nodes cache the data in a strategic manner, but ultimately the search is conducted by the sink alone. In the third case, sinks in a sensor network publish interests in certain types of data, gradients are established along links, and sources push data along links to the most likely interested parties. This too is similar to the

two-sided search in that both sides cooperatively establish a connection, but no search is actually conducted. Instead, the link gradients are used to guide the data pushed by the sources to the sink.

III. Model

Throughout the discussion, the two nodes conducting the search will be referred to as v_1 and v_2. Links and path length are frequently referred to as hops and hop count, respectively.

We make the standard assumptions found in the expanding search literature [8], [3], [5]. We assume that all links are bidirectional. Let x_M be the max-minimum path length between v_1 and v_2. The searchers therefore assign queries TTL values from the set $R = \{x_0 = 0, \ldots, x_M\}$, where $x_i = i$ $\forall 0 \leq i < M$. The resources required for querying with a TTL value x (referred to as the cost function) depends on the querying node's location. As such, the costs of searching from v_1 and v_2 are denoted $C_1(x)$ and $C_2(x)$, respectively. The minimum number of hops between the nodes is associated with a probability mass function (PMF) $f(x)$ and a corresponding cumulative distribution function (CDF) $F(x)$ (referred to as the hop count distribution). The term "location" loosely refers to the node-specific factors that contribute to these functions. In the one-sided search, only the cost of searching from the searching node needs to be considered, which is generally denoted $C(x)$. Note that the PMF and CDF of the minimum hop count between the locations of v_1 and v_2 is the same when searching in either direction, since we assume that all links are bidirectional.

IV. Unconstrained Search

In this section, we show how to derive the optimal two-sided search strategy when there are no time constraints. An unconstrained search is said to occur in rounds, with only one query initiated per round by one of the nodes. A new round begins only after it is certain that a path was not found in the previous round. Although it is possible for both nodes to search in the same round, this is no better than one node searching after the other. A search strategy is therefore a sequence defining which node initiates a query each round and the TTL value it assigns to the query. Our solution uses dynamic programming to calculate the optimal *cost-to-go*—the cost of continuing to search if a path was not found—given the last TTL value used by each node. The optimal search strategy is the sequence of TTL values that minimizes the cost-to-go when no search has yet been conducted.

The two-sided search is guaranteed to find any path found by the one-sided search, since a path of length x from v_1 to v_2, for example, can be divided into a subpath of length x' from v_1 to an intermediate node v' and a subpath of length $x - x'$ from v_2 to v'. Queries by v_1 and v_2 using TTL values x' and $x - x'$, respectively, are guaranteed to find each subpath to v', since the queries extend to all nodes within each respective number of hops from v_1 and v_2.

Before we continue, we show how two-sided ERS is more efficient than full flooding even when the one-sided search is not. Corollary 1 in [5] establishes that, for the the one-sided search, full flooding (only searching with x_M and incurring cost $C(x_M)$) is optimal if and only if $F(x) \leq C(x)/C(x_M)$ for all $1 \leq x \leq x_M$. This is not so for the two-sided search. Consider the case where $C_1(x) = C_2(x) = x^2/100$, $F(x) = x^3/1000$, and the value of x is in the range $[1, 10]$. The cost of full flooding is 1, and the expected cost of any other one-sided search sequence is strictly greater than 1. However, if both nodes search up to 5, thus guaranteeing that a path is found, the expected cost is $0.25 + 0.25 = 0.5$. Where the one-sided search cannot improve expected costs over full flooding, the two-sided search cuts costs in half.

Next we show how to derive the optimal one-sided search strategy, and then we show how to derive the optimal two-sided search strategy. We analyze the necessary length of each round in Section V.

A. One-Sided Search

In the one-sided search, one node (the searcher) searches for the other (the target). In this case, there is only one cost function $C(x)$, which is the cost to the searcher of searching with TTL value x. A search strategy $S = \{u_1, \ldots, u_L = x_M\}$ is a sequence of L TTL values that the searcher uses until it completes the search. The expected cost of using strategy S, $J(S)$, can be calculated as follows:

$$J(S) = C(u_1) + \sum_{i=2}^{L} C(u_i)(1 - F(u_{i-1})) \tag{1}$$

The cost $C(u_1)$ must be incurred since an initial query using TTL value u_1 is necessarily issued. The remaining costs, $C(u_i)$, $\forall 1 < 1 \leq L$, are incurred only if the target was not found in the previous round, which has a probability of $1 - F(u_{i-1})$. The expected cost can also be formulated as follows:

$$\begin{aligned} J(r, \{u_i, \ldots, u_L\}) &= C(u_i) \\ &\quad + (1 - F(u_i|r))J(u_i, \{u_{i+1}, \ldots, u_L\}), \forall 1 \leq i < L \\ J(u_L, \{\}) &= 0 \end{aligned} \tag{2}$$

In this formula, $J(r, \{u_i, \ldots, u_L\})$ is the cost of searching from round i onwards when r was the last TTL value used. This is the cost of querying with TTL value u_i, $C(u_i)$, plus the expected cost of continuing the search, $(1 - F(u_i|r))J(\{u_{i+1}, \ldots, u_L\})$, if the target is not found. $F(u_i|r))$ is the cumulative probability of finding the target with TTL value u_i conditioned on the probability that the target was not found with TTL value r. The cost of the entire strategy is $J(0, S)$. In general, the PMF and CDF of finding the target with TTL value x, given that it was not found with TTL value r, respectively denoted $f(x|r)$ and $F(x|r)$, can be calculated as follows:

$$f(x|r) = \begin{cases} \frac{f(x)}{1 - F(r)} & r < x \leq x_M \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$F(x|r) = \begin{cases} \frac{F(x) - F(r)}{1 - F(r)} & r < x \leq x_M \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Based on (2), we formulate the optimal cost-to-go in a way that the optimal strategy can be derived in $O(M^2)$ time using dynamic programming. The optimal cost-to-go for some TTL value x_i, $V(x_i)$, is the minimum cost of continuing the search, over all TTL values greater than x_i, if the search with x_i failed to find the target. The optimal search sequence is the sequence of TTL values that minimizes the cost-to-go for TTL value x_0. $V(x_i)$ can be formulated as follows:

$$V(x_i) = \min_{i+1 \le j \le M} \{C(x_j) + (1 - F(x_j|x_i))V(x_j)\} \quad (5)$$

$$V(x_M) = 0 \quad (6)$$

Equation (5) reflects the fact that failing to find the target using x_i, $0 \le i < M$, requires continuing the search with some TTL value x_j, $i < j \le M$, that is larger than x_i. The x_j that minimizes the cost-to-go for x_i is the one that is used to continue the search. Equation (6) reflects the fact that the search ends when searching up to x_M. $V(x_0)$ reflects the expected cost of the optimal strategy. By recording the x_j's that minimize the cost-to-go for each x_i, the optimal strategy can be extracted by following these links forward from x_0.

B. Two-Sided Search

We define a two-sided search strategy as a sequence S of L ordered pairs (u_i^1, u_i^2), in which u_i^1 and u_i^2 are the TTL values used by the nodes v_1 and v_2, respectively, in round i. Note that since there is no benefit to both nodes searching in the same round, either u_i^1 or u_i^2 will be null. We use $s(i)$ to identify the node that searches in round i, and m_i to indicate the maximum value used by $s(i)$ in any round prior to round i. That is, $m_i = \max_{1 \le j < i} u_j^{s(i)}$. Likewise, we use $s'(i)$ to indicate the node that is not searching in round i, and m'_i to indicate the maximum value used by $s'(i)$ in any round prior to round i ($\max_{1 \le j < i} u_j^{s'(i)}$). The expected cost of using a search strategy S, $J(S)$, can be calculated as follows:

$$J(S) = C_{s(1)}(u_1^{s(1)}) + \sum_{i=2}^{L} C_{s(i)}(u_i^{s(i)})(1 - F(m_i + m'_i)) \quad (7)$$

Here, the probability of finding a path in round i is different than in (1), because the path length is at least the maximum TTL value used so far by $s(i)$ plus the maximum TTL value used so far by $s'(i)$.

The optimal two-sided strategy can be derived in a similar way as the optimal one-sided strategy. Here we formulate the optimal cost-to-go given the last TTL values used by v_1 and v_2, denoted r_1 and r_2, respectively. The conditional PMF and CDF of v_1 finding a path with TTL value x given r_1 and r_2, denoted $f(x|r_1, r_2)$ and $F(x|r_1, r_2)$, respectively, are calculated slightly differently than in the one-sided case. The minimum path length in this case is necessarily in the interval $[r_1 + r_2 + 1, x_M]$; however, the first node searches in the range $[r_1 + 1, x_M - r_2]$, and the second node searches in the range $[r_2 + 1, x_M - r_1]$. Therefore, the conditional PMF and CDF are $f(x + r_2|r_1 + r_2)$ and $F(x + r_2|r_1 + r_2)$ (using (3) and (4)).

We denote the optimal cost-to-go $V(r_1, r_2)$. The base cases are $V(r_1, r_2) = 0$, $\forall r_1, r_2 | r_1 + r_2 \ge x_M$, because one node necessarily finds an intermediate node with a path to the other node in these cases. The cost-to-go in all other cases ($\forall r_1, r_2 | r_1 + r_2 < x_M$) is the minimum of the cost-to-go when either v_1 searches or v_2 searches, which consequently determines whether v_1 or v_2 searches in each of those cases. The formula below shows how this can be calculated. In the first case, when only v_1 searches, the cost-to-go is the minimum over all $r_1 < x_i \le x_M - r_2$ of the expected cost of v_1 searching with TTL value x_i. The cost-to-go when only v_2 searches is calculated in a similar manner. The optimal strategy is determined by $V(0, 0)$.

$$\begin{aligned} V(r_1, r_2) = \min\{ & \\ & \min_{r_1 < x_i \le x_M - r_2} \{C_1(x_i) + (1 - F(x_i|r_1, r_2))V(x_i, r_2)\} \\ & \min_{r_2 < x_j \le x_M - r_1} \{C_2(x_j) + (1 - F(x_j|r_2, r_1))V(r_1, x_j)\} \end{aligned} \quad (8)$$

The time complexity of this solution is $O(M^3)$, polynomial as in the case of the one-sided search and only one degree higher. The sequence returned is at worst a one-sided search sequence from one of the nodes, so it always returns a solution that is at most as costly as the optimal one-sided search.

C. Comparison of Unconstrained Search Techniques

We illustrate the expected costs of the one- and two-sided searches using synthetic data generated by randomly placing 300 nodes in a $1000m$ x $1000m$ field. All nodes within $100m$ of each other are linked. The (x, y) coordinates of v_1 are $(100, 500)$. The y coordinate of v_2 is 500 and the x coordinate is taken from the set $\{300, 400, \ldots, 900\}$, creating seven different settings. Figure 1 visualizes the case when the x coordinate of v_2 is 600. The enlarged dot on the left represents v_1, and the enlarged dot in the middle represents v_2. For each setting, the nodes were randomly placed 500 times. The normalized frequency distribution of the minimum number of hops between v_1 and v_2 over the 500 random placements is used for $f(x)$. The average cumulative number of nodes within successive hop counts from v_1 is used for $C_1(x)$, and likewise for v_2 and $C_2(x)$.

Figure 2 shows the normalized cost function of searching from v_1 and v_2 and the CDF of the minimum path length between both nodes when the x coordinate of v_2 is 600. In this case, the optimal one-sided strategy from v_1 is $(8, 11, 13)$. The optimal two-sided strategy is:

$round$	1	2	3	4	5	6	7
v_1	0	5	0	0	0	7	0
v_2	1	0	2	3	5	0	6

Notice here that no path will be found in the first round, but the combination of the search with TTL value 1 by v_2 and value 5 by v_1 is equivalent to a search for a path length 6, for which there is some probability of finding a path.

Figure 3 shows the expected costs of a one-sided search from v_1, a one-sided search from v_2, and a two-sided search

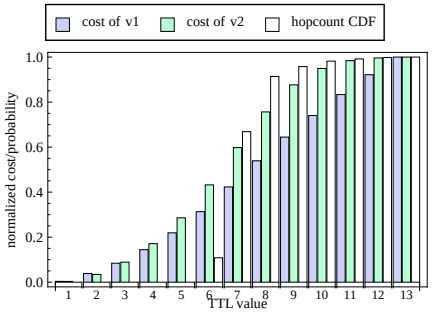

Fig. 2. Normalized cost of searching from v_1 and v_2 and the hop count CDF when v_2 is located at $(600, 500)$

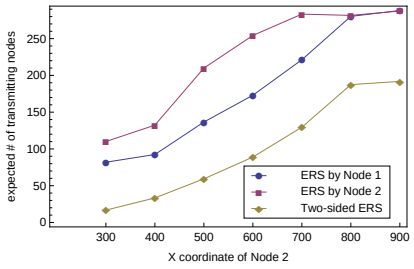

Fig. 3. Expected costs of using ERS from v_1, ERS from v_2, and a two-sided search with respect to location of v_2

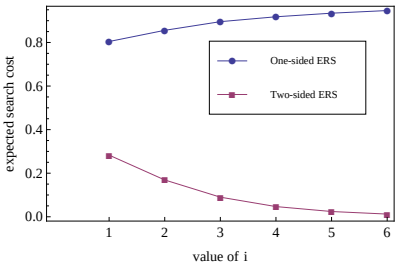

Fig. 4. Expected costs of using ERS from v_1 and a two-sided search with respect to $F(x) = (x/10)^i$ and $C(x) = (x/10)^{i+1}$

for the different locations of v_2. This figure shows that the two-sided search is more efficient than the one-sided search by either node. It requires about 70 fewer transmitting nodes on average. When v_2 is at location $(900, 500)$, the cost of the two-sided search is about 190, while the costs of the one-sided searches are close to the cost of full flooding. This figure also shows that it is less costly to search from v_1 than v_2. This is because there are fewer nodes to the left of v_1 than there are to the left of v_2 (see Figure 1).

Figure 4 shows the expected costs for a synthetic scenario where $F(x) = (x/10)^i$ and $C_1(x) = C_2(x) = (x/10)^{i+1}$, $1 \leq i \leq 6$ and $1 \leq x \leq 10$. $F(x)$ is shown on the x-axis, while $C_1(x)$ and $C_2(x)$ can be inferred from $F(x)$. Two important observations are made. First, as i increases, the expected cost of the one-sided search converges to 1 (the maximum cost), while the expected cost of the two-sided decreases. This strengthens the argument that the two-sided search is more efficient than full flooding in cases that the one-sided search is not. More importantly, this figure shows that the improvement of the two-sided search over the one-sided search is unbounded.

V. Time-Constrained Search

Here we analyze the problem of finding the search sequence with minimum expected cost under worst-case delay constraints. By worst-case delay constraints, we mean that the length of the search is not to exceed some time limit. Throughout the discussion, time is measured in time steps that correspond to the time it takes to forward a message one hop. We say that the entire search is limited to T time steps.

We again analyze the one-sided search before the two-sided search. In both cases, we first show how to calculate the maximum and expected search time required by a unconstrained search strategy. This answers the question in the previous section of how long each round should be, and it also assists in the analysis of the optimal time-constrained strategy. We prove some properties of the optimal time-constrained strategy, and then we show how to derive it.

A. One-Sided Search

In the unconstrained search, each round must be long enough for the searcher to be notified that a path was found, if so. This is the time it takes to forward a query the maximum number of hops allowed by the current TTL value plus the time it takes for a reply to be sent back to the searcher, which is two times the TTL value. The worst-case time for search strategy $S = \{u_1, \ldots, u_L\}$ is therefore

$$\sum_{i=1}^{L} 2u_i \tag{9}$$

The expected time includes the expected round trip time to the target when it is successfully found plus the expected time for failed queries. This is reflected by the first and last terms, respectively, in the following formula for the expected time.

$$\sum_{i=1}^{M} 2x_i f(x_i) + \sum_{i=1}^{L-1} 2u_i(1 - F(u_i)) \tag{10}$$

In the time-constrained search, one may think there might be some benefit to beginning a new query before the previous query is complete. In the following proposition, we establish that this is not true.

Proposition 1: In the one-sided expanding ring search, initiating a new query before the previous query is complete does not reduce expected costs.

Proof: We prove this by showing how, in the construction of a search strategy $u_1, \ldots, u_L$, overlapping queries does not improve costs. First note that $u_i < u_{i+1}$, because otherwise it would not be necessary to search with u_{i+1}. Next, note that the last TTL value to be used, u_L, is x_M. The most information available when the last query is issued is the maximum round trip made by the previous query. Therefore, there is no benefit to setting u_{L-1} to a larger TTL value than is needed to make a complete round trip before the last query begins. Now, given that $u_{L-2} < u_{L-1}$, the use of u_{L-2} provides no more information by the time u_L is used than will be available by u_{L-1}. Therefore, the same principle applies: u_{L-2} only needs to be set to the maximum TTL value needed to make a complete round trip before u_{L-1} is used. This same principle applies for all u_i, thus proving that overlaps do not reduce search costs. ■

Since the queries do not overlap in the optimal one-sided search strategy, it is only necessary find the optimal sequence

of TTL values that meets the time constraints. As such, without loss of generality, we assume that T is an even number. We can derive the optimal sequence using the following dynamic programming equations for the expected cost-to-go given the last TTL value used, r, and the time the current query is to be conducted, t.

$$V(r,t) = \min_{r' \in Q}\{C(r') + (1 - F(r'|r))V(r', t + 2r')\}$$
$$V(x_M, t) = 0 \tag{11}$$

Here $V(r,t)$ is the expected cost-to-go when the searcher already searched up to range r before time t, and $Q = \{r + 1, \ldots, \min\{(T-t)/2 - x_M, x_M\}, x_M\}$. The first case is the cost-to-go of terminating the search using the maximum range x_M. The second is the expected cost when there is enough time to search up to some range less than or equal to x_M. In this case, it is necessary to allow enough time for a final search with x_M, so the set of possible search ranges is limited to $r+1, \ldots, \min\{(T-t)/2 - x_M, x_M\}$ in addition to x_M.

B. Two-Sided Search

In the two-sided search, a search round must be long enough for the search node in the next round to be notified that a path was found. This is different from the time required by the one-sided search, since the search node in the current round may not be the same as the search node in the next round. Each round must be long enough for the current search node to forward a query to the hop extent defined by the current TTL value plus the maximum length of any cached route to the next round's search node, which is effectively the largest TTL used so far by the next round's search node. The last round must be long enough for both nodes to be notified about the established route, which is the TTL value of that round plus the maximum TTL value used in any round. This is more formally defined by the following formula.

$$\sum_{i=1}^{L-1}(u_i^{s(i)} + m_{i+1}) + u_L^{s(L)} + \max_{1 \le j \le L} u_j^{s(j)} \tag{12}$$

The expected time includes the expected round trip time to the target when it is successfully found plus the expected time for failed queries. This is reflected by the first and last terms, respectively, in the following formula for the expected time.

$$\sum_{i=1}^{L} \sum_{j=m_i+1}^{u_i^{s(i)}} (j + \max\{j, m_i'\}) f(j + m_i') + \sum_{i=1}^{L-1}(u_i^{s(i)} + m_{i+1})(1 - F(u_i^{s(i)} + m_i')) \tag{13}$$

We previously stated that allowing more than one node to search in a round does not improve expected costs when there are no time constraints. This is not the case when there are time constraints. Not only that, but a time-constrained search can be made efficient by overlapping queries, even those of a single node. That is, a node may start a query before enough time passed for it to receive results from prior queries. We prove this with the following example. Let k be a very large integer; the distance between v_1 and v_2 be either 2, 4 or 11, with probabilities $\frac{1}{2}$, $\frac{1}{2} - \frac{1}{k}$, and $\frac{1}{k}$; the time limit be 14; and the cost functions be

$$C_1(r) = \begin{cases} r & r \le 2 \\ k + r & 2 < r \le 6 \\ k^3 + r & r > 6 \end{cases} \tag{14}$$

and

$$C_2(r) = \begin{cases} r & r \le 2 \\ k + r & 2 < r \le 5 \\ k^3 + r & r > 5 \end{cases} \tag{15}$$

First, observe that the cost of covering the maximum distance of 11 is $2k + 11$ by using TTL values 6 and 5 for v_1 and v_2, respectively, and at least $k^3 + 11$ by using any larger TTL value for either v_1 and v_2, even if the TTL value is lower for the other node. This means that using TTL values 6 and 5 for v_1 and v_2 covers the maximum distance of 11 with the least cost when $k > 2$. Hence, we may assume, without loss of generality, that v_1 searches with TTL value 6 at time 2, and v_2 searches with TTL value 5 at time 4, thus terminating the search by the deadline. Consider the following two strategies:

t	r_1	r_2
0	1	1
2	6	–
4	–	5

Strategy 1

t	r_1	r_2
0	2	2
2	6	–
4	–	5

Strategy 2

If overlapping queries are not allowed, then the maximum search distance of v_1 before its final search is 1 (starting at time 0), while the maximum distance for v_2 is 2 (also starting at time 0). Together this is not enough to find a path of length 4, so it follows that Strategy 1 is the best strategy without overlapping queries. The expected cost is $\frac{1}{2} \cdot (1 + 1) + \frac{1}{2} \cdot (1 + 1 + (k + 6) + (k + 5)) = k + 7.5$. On the other hand, the expected cost of Strategy 2, which contains overlapping queries, is $\frac{1}{2} \cdot (2+2) + (\frac{1}{2} - \frac{1}{k}) \cdot (2 + 2 + (k+6)) + \frac{1}{k} \cdot (2+2+(k+6)+(k+5)) \le \frac{k}{2} + 9$ if $k \ge 5$. As k approaches infinity, this is half the expected cost of the non-overlap strategy, thus demonstrating that the optimal strategy may contain overlaps between queries.

A solution like (8) to derive the optimal sequence requires tracking the history of TTL values that overlap future queries, which is highly complex. Even if the query of one node can only overlap the query of the other, the solution requires a lengthy enumeration of all cases. Here we show how to derive a rounds-based strategy, in which one or both nodes make a query at the start of each round. We use $V_1(r_1, r_2, t)$ to indicate the cost-to-go when only v_1 searches at time t and the maximum search ranges used so far by v_1 and v_2 are r_1 and r_2, respectively. Likewise, we use $V_2(r_1, r_2, t)$ when only v_2 searches and $V_3(r_1, r_2, t)$ when both nodes search.

For each function, there are three cases to consider. If $t > T$, these functions are set to infinity because the strategy is

not valid. If $t \leq T$ and $r_1 + r_2 \geq x_M$, they are set to zero because the search will terminate properly. Finally, if $t \leq T$ and $r_1 + r_2 < x_M$, they are set to the lowest cost-to-go over all ways of continuing the search. We first show how this is calculated for $V_1(r_1, r_2, t)$. There are two ways to continue the search. The first way is for v_1 to query with the TTL value $r_1' = x_M - r_2$, thus terminating the search. The expected cost of doing so is:

$$C_1(r_1') + V_3(r_1', r_2, t + r_1' + \max\{r_1', r_2\}) \tag{16}$$

This is the cost $C_1(r_1')$ of querying with r_1' plus the cost of continuing the search, so to speak, after enough time has elapsed for both nodes to be notified that a path was found. This occurs at time $t' = t + r_1' + \max\{r_1', r_2\}$. Because $r_1' + r_2 = r_M$, $V_3(r_1', r_2, t + r_1' + \max\{r_1', r_2\})$ equals either infinity or zero, depending on the value of t'. The second way for v_1 to continue the search is to use a TTL value r_1' from the range (r_1, x_M), such that it is possible that a path will not be found and either v_1, v_2, or both nodes will need to continue the search. The expected cost of each of these cases is respectively:

$$C_1(r_1') + (1 - F(r_1'|r_1, r_2))V_1(r_1', r_2, t + 2r_1') \tag{17}$$

$$C_1(r_1') + (1 - F(r_1'|r_1, r_2))V_2(r_1', r_2, t + r_1' + r_2) \tag{18}$$

$$C_1(r_1') + (1 - F(r_1'|r_1, r_2))V_3(r_1', r_2, t + r_1' + \max\{r_1', r_2\}) \tag{19}$$

The main difference between these formulas is the time that the next round should begin, which was already explained with regards to (12).

$V_2(r_1, r_2, t)$ and $V_3(r_1, r_2, t)$ are calculated similarly. In the case of $V_3(r_1, r_2, t)$, however, either the search can be terminated with a pair of TTL values (r_1', r_2') such that $r_1 < r_1' < r_M - r_2$, $r_2 < r_2' < r_M - r_1$, and $r_1' + r_2' = x_M$, or continued by limiting the sum $r_1' + r_2'$ to some value less than x_M. The cost-to-go must reflect that both nodes are searching. For example, the equivalent formula to (18) is

$$C_1(r_1') + C_2(r_2') + (1 - F(r_1' + r_2'|r_1 + r_2))V_1(r_1', r_2', t + 2r_1') \tag{20}$$

C. Comparison

Figure 5 shows the expected and maximum time of an unconstrained one-sided search from v_1 and an unconstrained two-sided search for the settings described in Section III. The expected time increases with the distance between v_1 and v_2 since more time is required for the round trip of each query. The maximum time decreases at $x = 500$ because there are fewer rounds in the optimal strategy for this setting than for the case when $x = 400$, and similarly when $x = 800$. The trends in the one-sided and two-sided search are similar. Notice here that the expected time of the one-sided and two-sided searches are very similar, but the maximum time of the two-sided search is mostly less than the maximum time of the one-sided search.

Figure 6 shows on top the lower and upper bounds on the time-constraints of an ERS from v_1 (in blue) and a two-sided search (in pink). The lower bound is the minimum time required when the target is at the furthest range. For the one-sided search, this is two times the maximum TTL value, the time it takes to send a query to the maximum distance and back. For the two-sided search, this is two times the ceiling on half the maximum TTL value, the time to send a query from one node to half the maximum distance to the other node and back. The upper bound is the maximum time required for the unconstrained search. This figure shows that these bounds are mostly lower for the two-sided search. However, the most constrained one-sided search is still shorter than the unconstrained two-sided search.

Figure 6 shows on the bottom the lower and upper bounds on the costs of a time-constrained ERS from v_1 (in blue) and a two-sided search (in pink). The lower bound is the cost of the unconstrained search, and the upper bound is the cost of the most time-constrained search. There are two interesting trends here. The most time-constrained two-sided search is less costly than the least time-constrained one-sided search, meaning that there is no benefit to using the one-sided search. Additionally, the difference in cost between the most and least time-constrained two-sided search is usually very small, meaning that the two-sided search can be both fast and efficient.

Figure 7 shows the cost of a one-sided search from v_1 and a two-sided search with respect to the constraints on the time allowed when the location of v_2 is $(600, 500)$. For the one-sided search, although there is a significant drop in the cost when the time allowed is changed from 39 rounds to 42 rounds, there is no change in the cost when the time constraint is in the ranges $[26, 39]$ and $[42, 61]$. This means that there are no improvements in cost when there is some flexibility in the time constraints unless the threshold from 39 to 42 rounds is crossed. This trend occurs for all locations of v_2. For the two-sided search, there is an improvement in cost for just about any relaxation in the time constraints.

VI. Conclusion

In this paper, we introduced a new method to search for a path between two nodes in an ad hoc network. This method generalizes expanding ring search by searching from both nodes, hence weakly dominating it in terms of the overhead of unnecessary transmissions. This method is general and can also be applied to any of the extensions suggested for and inspired by expanding ring search itself. One interesting result obtained in this paper is that the two-sided search is more efficient than full flooding alone even in some cases that a one-sided search is not.

The minimum cost unconstrained two-sided search strategy can be derived in polynomial time, with an order of complexity not much greater than the time needed to derive the optimal one-sided search strategy. While overlapping queries do not improve the efficiency of the time-constrained one-sided search, they improve the efficiency of the two-sided search. However, deriving the minimum cost time-constrained two-sided search is highly complex. As such, we show how to derive a simpler albeit less efficient strategy, also in polynomial

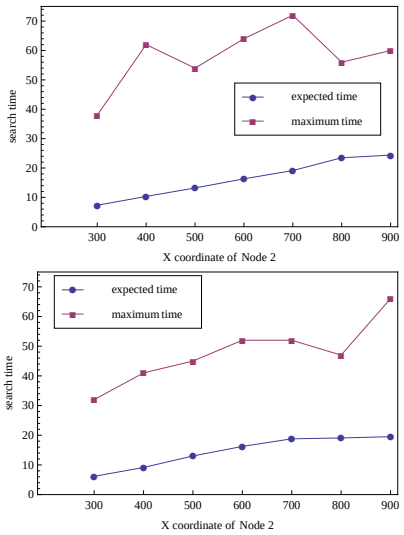

Fig. 5. Expected and maximum time of using a one-sided ERS (top) and two-sided ERS (bottom)

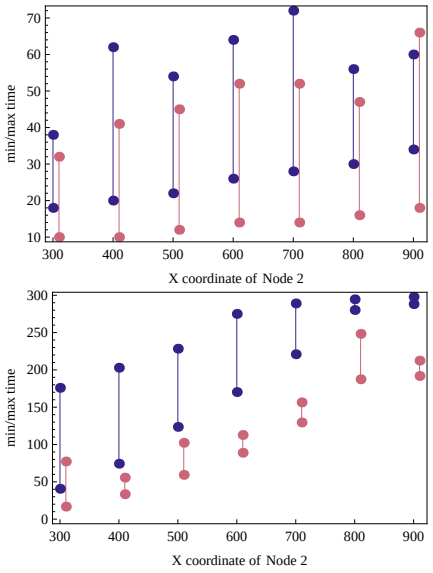

Fig. 6. Upper and lower bounds on the time constraints (top) and expected cost (bottom) of a time-constrained one-sided ERS from v_1 (blue) and two-sided search (pink)

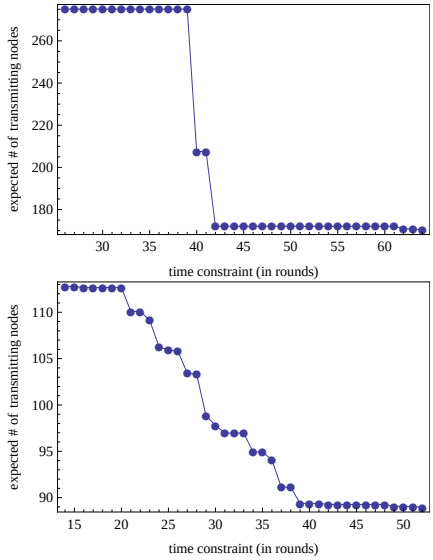

Fig. 7. Expected cost of a time-constrained one-sided ERS from v_1 (top) and two-sided ERS (bottom) when v_2 is located at $(600, 500)$

time, that still guarantees expected costs less than or equal to those of the time-constrained one-sided search. We show in select examples, based on realistic scenarios, that the most time-constrained two-sided search strategy is more efficient than the least time-constrained one-sided search.

All related ERS literature assumes that the cost and probability functions can be obtained by analysis or some other means. Without this assumption, it would not be possible to make any performance guarantees. This assumption is valid because no performance guarantees can be made about any protocol without some basic assumptions about network properties. Future work should address how these functions can be practically obtained for use by these techniques.

Acknowledgments

This work was supported in part by the Israeli Ministry of Industry and Trade under project RESCUE and ISF grant 1083/13. Thanks to Dror Rawitz for his assistance with the proof in Section V-B.

References

[1] E. M. Royer and C. k Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, pp. 46–55, 1999.

[2] B. Krishnamachari and J. Ahn, "Optimizing data replication for expanding ring-based queries in wireless sensor networks," in *WiOpt*. IEEE, 2006, pp. 361–370.

[3] Z. Cheng and W. B. Heinzelman, "Searching strategies for target discovery in wireless networks," *Ad Hoc Networks*, vol. 5, no. 4, pp. 413–428, 2007.

[4] Y. M. Baryshnikov, E. G. Coffman Jr., P. R. Jelenkovic, P. Momcilovic, and D. Rubenstein, "Flood search under the california split rule," *Oper. Res. Lett.*, vol. 32, no. 3, pp. 199–206, 2004.

[5] N. B. Chang and M. Liu, "Revisiting the TTL-based controlled flooding search: optimality and randomization," in *MOBICOM*, Z. J. Haas, S. R. Das, and R. Jain, Eds. ACM, 2004, pp. 85–99.

[6] Q. J. Chen, S. S. Kanhere, and M. Hassan, "Performance analysis of geography-limited broadcasting in multihop wireless networks," *Wireless Communications and Mobile Computing*, vol. 13, no. 15, pp. 1406–1421, 2013.

[7] N. B. Chang and M. Liu, "Controlled flooding search with delay constraints," in *INFOCOM*. IEEE, 2006.

[8] J. Deng and S. A. Zuyev, "On search sets of expanding ring search in wireless networks," *Ad Hoc Networks*, vol. 6, no. 7, pp. 1168–1181, 2008.

[9] N. B. Chang and M. Liu, "Controlled flooding search in a large network," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 436–449, 2007.

[10] J. Hassan and S. Jha, "On the optimization trade-offs of expanding ring search," in *IWDC*, ser. Lecture Notes in Computer Science, N. Das, A. Sen, S. K. Das, and B. P. Sinha, Eds., vol. 3326. Springer, 2004, pp. 489–494.

[11] R. Lima, C. Baquero, and H. Miranda, "Stopping ongoing broadcasts in large manets," ser. ARMOR '12. New York, NY, USA: ACM, 2012, pp. 4:1–4:5.

[12] M. Al-Rodhaan, L. M. Mackenzie, and M. Ould-Khaoua, "Efficient expanding ring search for manets," *IJCNIS*, vol. 2, no. 3, 2010.

[13] I. M. Pu and Y. Shen, "Analytical studies of energy-time efficiency of blocking expanding ring search," *Mathematics in Computer Science*, vol. 3, no. 4, pp. 443–456, 2010.

[14] K. Rachuri and C. S. R. Murthy, "Energy efficient and scalable search in dense wireless sensor networks," *IEEE Trans. Computers*, vol. 58, no. 6, pp. 812–826, 2009.

[15] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks," *Wireless Networks*, vol. 6, no. 4, pp. 307–321, 2000.

[16] H. Dubois-Ferrière, M. Grossglauser, and M. Vetterli, "Age matters: efficient route discovery in mobile ad hoc networks using encounter ages," in *MobiHoc*. ACM, 2003, pp. 257–266.

[17] Z. Cheng and W. B. Heinzelman, "Adaptive local searching and caching strategies for on-demand routing protocols in ad hoc networks," *IJHPCN*, vol. 4, no. 1/2, pp. 51–65, 2006.

[18] C. Intanagonwiwat, R. Govindan, D. Estrin, J. S. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, 2003.