CAPTURE: location-free Contact-Assisted Power-efficienT qUery REsolution for sensor networks

Ahmed Helmy*
helmy@usc.edu

Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-2562

Queries and small transfers are likely to constitute a significant portion of the flows in emerging classes of sensor networks. Route discovery for such queries incurs much more communication overhead than the actual data transfer. Especially for large-scale sensor networks, it is quite costly to establish shortest path routes for such types of requests. Flooding-based approaches for routing are designed to search for high quality routes. Such approaches may be suitable for prolonged transfers, but not for small ones. In this paper, we present an architecture that is geared towards one-shot frequent queries in sensor networks. In our approach we aim at reducing the total energy cost query resolution as opposed to searching for high quality routes.

Our architecture uses a hybrid approach, where each node collects information about nodes in its proximity, up to R hops away, using a link state protocol. Beyond the proximity, we introduce the novel notion of contacts that act as short cuts to reduce the degrees of separation between the request source and the target. A new efficient, on-demand, contact selection protocol is integrated into the search process. We do not assume knowledge of node locations. Several protocols to implement different policies for the search are introduced. Extensive simulations are used to systematically evaluate the performance of our protocols. Our results show substantial communication overhead reduction for our contact-based technique as compared to related schemes. The study also shows reasonable settings of parameters that work well for a wide range of networks.

I. Introduction

Motivation: Classes of emerging wireless sensor networks are expected to have a significant impact and have the potential for many applications. Such networks are infrastructure-less, power-constraint, scalable networks, in which high quality route discovery and maintenance may be quite costly. In many applications of sensor networks the network may be treated as a distributed database that is queried for information. In future sensor networks, it is highly likely that queries and small transfers will constitute a significant portion of the supported traffic. Examples of small transfers include resource discovery, monitoring queries, and data centric storage, among others. For such small transfers, it is quite inefficient to establish optimal (shortest path) routes, where the cost of such routes exceeds (by far) the cost of the actual data transfer. Hence, it is crucial for the efficiency of sensor networks to provide routing protocols geared towards queries and small transfers. In this work, we propose a novel architecture for small transfers in sensor networks. We design our protocols to be self-configuring, power-efficient and scalable.

Background: Sensor networks consist of wireless, power-constrained devices that may be used to instrument a physical setting (e.g., habitat monitoring, object tracking). As such, sensor networks may be viewed in several classes of applications as a distributed *database*. One of the main functions of such networks is to *resolve queries* and carry out transactions and small transfers. Queries may be classified based on their semantics into different categories. A query may be simple (e.g., inquiring about one variable, such as temperature), or complex (e.g., an expression of several variables). A query may be one-shot or persistent; the latter leading to flow of information for an extended amount of time after the query

* This work was funded by grants from NSF CAREER Award 0134650, Intel and Pratt&Whitney ICT.

transfer. Also, the query may be for unique or replicated data. In this paper we target simple, one-shot queries, for potentially replicated data. One distinguishing characteristic of such queries is that the communication cost for route discovery (to resolve the query) may exceed the cost of data transfer. Since communication is a major consumer of energy, the cost of route discovery for one-shot queries and small transfers should be minimized to conserve energy.

Routing protocols for wireless networks [4] [3] [11] [12] have been traditionally designed to discover and maintain routes of high quality, to achieve efficient prolonged data transfers over those routes. This is a suitable approach for long transfers, where the cost of the initial route discovery is amortized over the savings during efficient data transfer. However, in cases of query resolution and small transfers - those that do not extend beyond the discovery phase – *high-cost-high-quality* routes may *not* be justified. In our approach the main design goal is *not* high quality routes, but to achieve successful delivery with very low overhead.

Flooding is a commonly used technique for resource discovery. For frequent requests in large-scale networks, however, flooding may incur significant communication overhead. Expanding ring search techniques are also commonly used for discovery, but are quite inefficient for large-scale sensor networks, where the network diameter tends to be quite high (due to clustering of nodes). For scalability, several hierarchical approaches have been proposed [18][19]. Many such architectures are cluster-based, in which nodes elect cluster-heads (or dominating set) to relay the traffic. A cluster-head may become a single point of failure or a point of traffic concentration. The landmark approach [5][13][14] avoids using the landmarks for communication, but uses them as directions for routing. However, the highest level landmark needs to periodically flood information throughout the network. One major concern in these hierarchical approaches is their reliance on complex coordination mechanisms that are susceptible to major re-configuration with node failures, node sleep/wake-up schedules and node mobility.

On-demand routing approaches in ad hoc networks (such as DSR[4] and AODV[3]) use caching schemes to alleviate the cost of flooding. These schemes can be quite efficient for small scale, static networks. The efficacy of caching, however, degrades severely with dynamics of the network, especially for large-scale networks, where the cache validity drops significantly.

Location-based or geographic routing is becoming a very attractive alternative when location information is available. Geographic routing (e.g., GPSR[34]) is stateless and only needs local neighbor location information to forward the packet towards the destination. An inherent assumption in geographic routing, however, is that the destination address is known. In the applications we target in this study (i.e., resource discovery and queries) **the destination location is *not* known a priori**. Hence, even if location information is available (which may not be the case for many sensor networks) an efficient resource discovery protocol is still needed. Some approaches use consistent distributed hashing for location discovery or data-centric storage (e.g., GLS[6], GHT[29], RRs[35]). Those approaches work well when the network boundary is known, there are no gaps or unoccupied areas, and a map of the network is configured into each node. In cases where the network boundary is not known a priori (e.g., due to rapid sensor deployment from an aircraft or vehicle) or is dynamic (due to node mobility), or when node failure creates gaps, these approaches may fail. The *CAPTURE* architecture avoids all these limitations by virtue of being *location-free*.

In this paper we introduce a new architecture for power-efficient resource discovery and small-transfers in large-scale sensor networks, called *CAPTURE*. *Instead of using shortest path or optimal routes, our design goal is to conserve network energy, while achieving high request success ratio.* We avoid the use of flooding or complex coordination mechanisms in our approach. *We design what we call on-demand on-the-fly loosely coupled hierarchy, in which instances of the hierarchy are efficiently constructed during the query process, without having to be maintained or re-configured.* In our architecture, every node independently collects information from neighboring nodes up to *R* hops away. This is called a node's *proximity*. Targets beyond the proximity of the node are discovered with the aid of *contacts*. *Use of **contacts** is key for efficient discovery in our scheme.* The idea behind the contacts borrows from *small worlds* [1] [16] [17]. Unlike relational or random graphs, wireless networks are spatial graphs (in which links are a function of distance, among other factors) that tend to be highly clustered, leading to very high path length. For a node, contacts are a few nodes outside of the proximity that act as *short cuts* to transform the wireless network into a small world and hence reduce the average degrees of separation between

the querier and the target. When a request[1] is made, the contact-selection protocol is invoked. Contact selection employs a simple, yet effective, mechanism to reduce proximity-overlap and to elect contacts that increase the search coverage. The search proceeds according to a search policy until the target is located or the query is resolved.

Salient features of our architecture include its ability to select useful contacts on-the-fly without having to maintain contact information a priori. Also, our protocols exhibit very good performance over a wide range of networks, without the need for parameter optimization for each network. Furthermore, our protocols respond well to replication with a drastic decrease in query overhead.

We use extensive simulations to evaluate the performance of our protocols in terms of energy consumption, success rate and latency. We compare our protocols to flooding, expanding ring search and ZRP. Our results show significant savings for our technique. For medium to high request rates, *CAPTURE* incurs 20-30% of flooding overhead, and achieves even greater savings for expanding ring search. Our protocols may be implemented using simple extensions to zone-routing protocols.

The rest of the paper is outlined as follows. Section II introduces an architectural overview of *CAPTURE*. Section III presents the contact-selection protocol and search policies. Section IV provides request processing and forwarding rules. Section V provides evaluation and comparison results. Section VI discusses related work and Section VII concludes.

II. CAPTURE Architectural Overview

In the *CAPTURE* architecture, each node in the network keeps track of a number of nodes in its vicinity within *R* hops away. This defines the *proximity* of a node, where *R* is called the proximity children. The lead subset for my technology. The proximity is maintained using a proactive localized link state broadcast. Each node chooses its proximity independently, and hence no major re-configuration is needed when a node moves or fails. There is no notion of cluster head, and no elections that require consensus among nodes. We assume the existence of a neighbor discovery protocol by which each node identifies nodes 1 hop away (through periodic beacons). The link state protocol provides neighbor information to other nodes in the proximity. Typically the number of nodes in the proximity is small (in our study we limit the number of proximity nodes to 100, by limiting *R*). As part of the proximity information each node keeps routes to nodes and pointers to resources in its proximity. Nodes *R* hops away are called *borders*. Overhead and dynamics of the link state protocol are evaluated in Section 5.3.3.

When a querier node *Q* (potentially any node in the network) issues a query or request, it first checks to see if the resource (or destination) is in its proximity. If not, then it seeks the assistance of a number of contacts (*NoC*) outside the proximity, as follows. First, a request is issued to *NoC* (say 3) of *Q*'s borders (*R* hops away). Each border, *B*, receiving the request would in turn select another node, *C*, at *r* hops away to which to forward the request. We call *C* a *contact* node and *r* the contact distance. To increase search efficiency, *C* should have low proximity overlap with *Q*. Proper setting of the parameter *r* helps to reduce such overlap. Contact nodes act as short cuts that bridge between disjoint proximities. This helps to reduce the degrees of separation between *Q* and the target nodes. Degrees of separation in this context refer to the number of intermediate nodes to get from the querier node to the target.

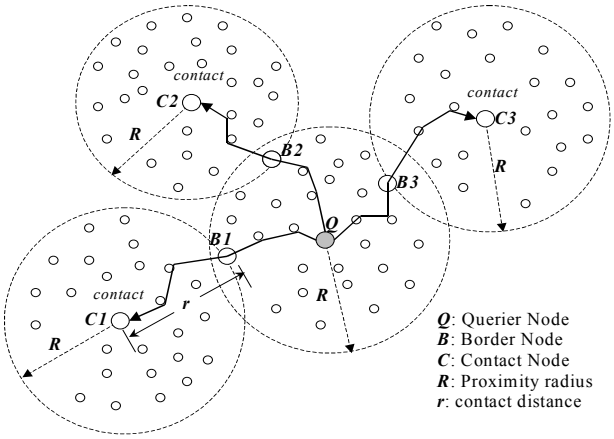


Figure 1. Each node in the network has a proximity of radius R hops. A querier node, Q, sends a request through a number of its borders equal to the number of contacts (NoC), in this case NoC=3. Each border node, Bi, chooses one of its borders, Ci, to be the direction for forwarding the request r hops further until it reaches the contact. The contacts are up to (R+r) hops away from Q. In this example r=R=3.

The main architecture is shown in Figure 1, where the querier node *Q* chooses three of its borders, *B1, B2, B3* to which to send a request

[1] We use the terms *request* and *query* inter-changeably.

message. Each of the borders in turn chooses one contact at r hops away to which to forward the request. *C1, C2,* and *C3* represent the contacts. The number of borders (and subsequently contacts) chosen, *NoC*, and the contact distance (r hops) are design parameters. If $r=R$ then the contact is a border of a border of Q.

Questions regarding setting the design parameters, such as number of contacts (*NoC*), contact distance (r), and proximity radius (R), shall be investigated in the evaluation section. First, we describe our contact selection scheme.

III. Contact Selection and Search Policies

This section introduces the contact-selection protocol and the notion of levels of contacts. Then presents various policies by which these levels may be traversed during the search, in a single attempt or multiple attempts.

III.A. Contact Selection Protocol

The main purpose of a contact node is to act as a short cut to increase the view of the network by searching for the target in uncovered parts of the network. Hence, it is important for a contact to have a proximity that does not overlap significantly with that of the querier node, Q, or the other contacts of Q. This is a distributed algorithm in which contacts do not know about each other, and do not know their shortest distance to the querier (remember that contacts are outside of the querier's proximity). Instead of attempting to find an optimal solution, the basic idea of our approach is to develop a simple, yet efficient algorithm, that attempts to reduce proximity overlaps, thus increasing coverage and reducing search overhead. The algorithm should incur low communication overhead, achieve high success rate and should scale to large networks.

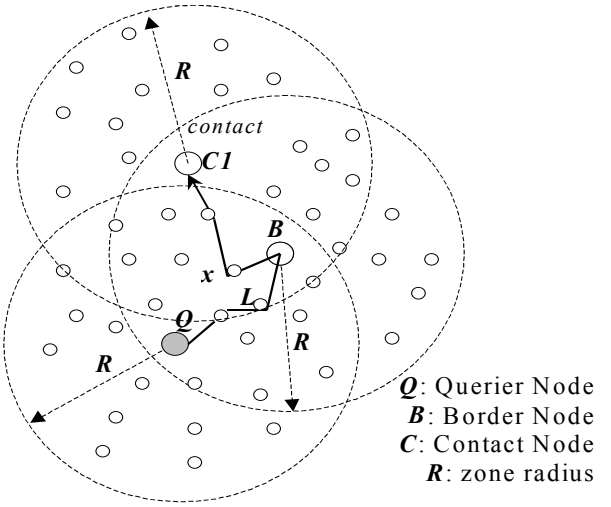
The first kind of overlap occurs between the contact's proximity and the querier's proximity. To reduce this overlap the request is directed out of the querier's proximity. One simple approach to try to achieve this is for the border node to randomly choose one of its borders to which to forward the request. This, however, often leads to significant overlap with the querier's proximity rendering the contact ineffective and the query success rate becomes low. Another simple approach is for the border node to avoid sending the request through the node from which it was received. However, wireless networks have a high clustering coefficient [1][16][2]. This means that, on average, there is relatively high probability that the neighbors of a neighbor of B are also neighbors of B. Therefore, it is not sufficient to avoid only the previous hop since there may still be a good chance that the border may forward the request through nodes that belong to Q's proximity. This is illustrated in Figure 2 (a), where the border node B receives the request from node L (the previous hop), and forwards it to contact $C1$ through node x. Node x is a neighbor of node L and is within Q*'s* proximity, and hence would lead to a contact less than $R+r$ hops away. In many cases the contact chosen this way may have a proximity heavily overlapping with Q's proximity.

The problem in forwarding the request outside of Q's proximity to a useful contact is the loss of direction for the forwarded message at the border of the proximity (since Q knows only about nodes R hops away). Remember that we do *not* assume or rely on knowledge of location information, since such information may not be available in some scenarios. This renders our scheme applicable to a wider class of sensor networks. To achieve a sense of direction without location information, we introduce a mechanism that uses information about the neighbors of B's previous hop, L, as explained next.

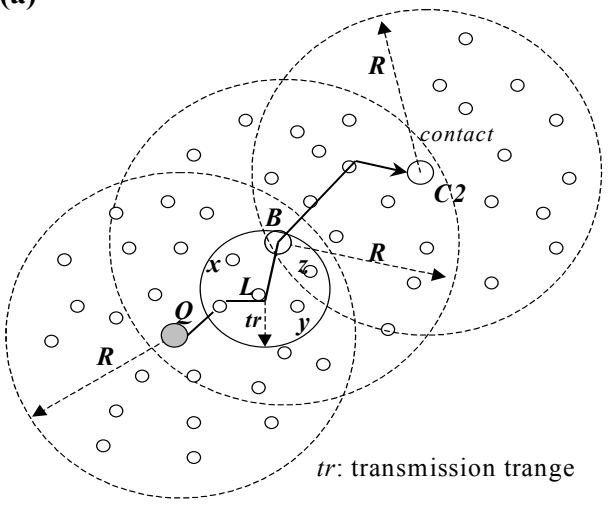
A querier node, Q, sends a request to *NoC* of its borders. Consider one of those borders, B. Let node L be the last hop before B on that path. Note that B is R hops away from Q, and L is R-1 hops away from Q. All L's neighbors (including B) are 1 hop away from L, and hence are at most R hops away from Q. That is, all L's neighbors are within Q's proximity. As was mentioned before, due to high clustering many of L's neighbors (all of which are in Q's proximity) may also be B's neighbors. Hence, B should attempt to avoid forwarding the request through any of L's neighbors. As illustrated in Figure 2 (b), B avoids L's neighbors (x,y,z) and is able to forward the request to a contact, $C2$, that has significantly less proximity overlap with Q than $C1$ does. If B cannot find a contact without passing through L's neighbors, then it randomly chooses a contact that does not pass through L. This scheme reduces overlaps drastically, as we shall show later on in the evaluation section. We call this scheme the *proximity overlap reduction* (*POR*) scheme. Note that for the above examples we have used $r=R$ for illustration. In cases where r is not equal to

[2]The clustering coefficient (cc) measures the probability that neighbors of a node are also neighbors of each other. In [16][17] it was shown that for wireless networks cc=0.58 (high clustering) for settings similar to our study.

R, *POR* is used to select a border for *B* that provides *direction* for choosing the contact, we call this the *direction border*. If $r<R$ then *POR* is performed by *B* and then the contact is selected between *B* and its direction border. If $r>R$ then the direction border needs to perform *POR* again to find its own direction border, and so on. *POR* is performed without incurring any extra communication overhead and in general is performed $\lceil r/R \rceil$ times at each chosen border.



(a)



(b)

Figure 2. (a) The border node, *B*, forwards the request towards its border *C*1 via node *x*. *C*1's proximity has significant overlap with *Q*'s proximity. By only using random forwarding or avoiding only node *L* (the previous hop) *B* can easily lose sense of direction and choose a poor contact. (b) By using neighbor information of *L*, *B* avoids forwarding the request to *L* or any of its neighbors (*x*,*y*,*z*), all of which are in *Q*'s proximity. Hence, *B* is more likely to choose a useful contact, *C*2. The overlap between *C*2's proximity and *Q*'s proximity is a lot less than overlap between *C*1's and *Q*'s proximities.

The second type of overlap occurs between proximities of contacts. To reduce this overlap the querier node, *Q*, attempts to select borders to which it has disjoint routes. This is done using the proximity information (with no extra overhead). If *NoC* borders are chosen by the end of this procedure then *Q* sends the request to the chosen borders. Otherwise, borders are chosen with minimum route overlap (i.e., with different 2nd hop nodes, then 3rd hop nodes, etc.). Otherwise, new borders are chosen randomly until *NoC* borders are chosen. This scheme does not guarantee non-overlap between contacts' proximities, but performs quite efficiently during requests, as we shall show. We call this scheme the *route overlap reduction* (*ROR*) scheme.

It is quite conceivable that a power-related metric may be integrated into the contact selection process. For example, in addition to resource information (e.g., sensor type and capability) exchange in the proximity, the power and drainage levels may also be piggybacked on the proximity-limited link state algorithm. When selecting contacts, those nodes with the highest power metric among the nodes that reduce the overlap will be chosen.

III.B. Levels of Contacts – putting the first pieces together

The above contact selection schemes (*POR* and *ROR*) provide a mechanism to select *NoC* contacts that have distances up to $R+r$ hops away from *Q*. We call these contacts *level*-1 contacts. To select the level-1 contacts *Q* performs *ROR* to reach *NoC* borders, then those borders (and their respective direction borders, and so on, $\lceil r/R \rceil$ times) perform *POR* to get the direction for the contacts.

To select farther contacts, this process is further repeated as needed at the level-1 contacts, level-2 contacts and so on, up to a number of levels called *maxDepth, D*. We shall study the effect of *D* in the evaluation section. The only difference between *Q* selecting the level-1 contacts, and level-*i* contacts selecting level-i+1 contacts is that level-*i* contacts need to perform *POR* and *ROR*. That is, a level-*i* contact, selects borders with disjoint routes from its set of borders that do not pass through its previous hop (*L*'s) neighbors.

III.C. Search Policies – putting all the pieces together

Given a request and a number of levels, *D*, the target search process may proceed using different policies. We investigate three different policies for

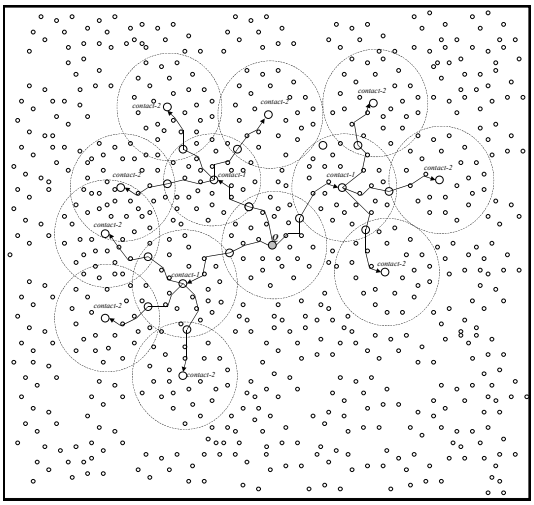
target search. The first is called *single-shot*, in which the querier sends out a request, in a single attempt, to traverse the contact levels in succession, up to D levels. The second policy is called *level-by-level* (*lbl*), in which the request is sent out in several attempts. The first attempt is performed with level depth of 1. Until and unless the target is found, each subsequent attempt, i, is performed with level depth $d_i=1+d_{i-1}$. Attempts continue up to $d_i=D$. The third policy is called *step* search (or simply *step*), and is very similar to *lbl* except that increasing the depth occurs in steps instead of increments of 1. For our study we choose an exponential step increase; i.e., $d_i=2d_{i-1}$.

Single-shot Policy

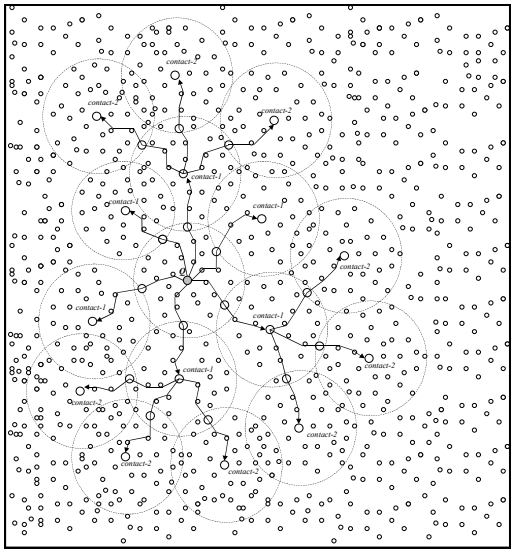
In this policy the request is sent out from the querier node once, in a single attempt. The request is forwarded directly from level-1 contacts to level-2 contacts, up to level-D contacts. In a sense, this policy is analogous to flooding between contacts. An example of single-shot with D=2, R=r=3, and NoC=3 is given in Figure 3 (a). To further clarify this policy we give a simple, first order, theoretical estimate of its overhead. These estimates are given only for illustration purposes. At each level-i, the theoretical number of contacts visited is $(NoC)^i$, and the theoretical number of hops traversed is $(R+r).(NoC)^i$. Hence, the number of transmissions is given by $[(R+r).\sum_{i=1}^{D}(NoC)^i]$. We note that this is only a theoretical upper bound. The search employs loop and re-visit prevention mechanisms, the effect of which are not considered in this simple theoretical analysis. After considering these mechanisms via detailed simulations, the overhead is reduced drastically, as will be shown Section 5.

Level-by-level (lbl) Policy

In *lbl* the querier node, Q, may need to send the request several times, in multiple attempts, until the target is reached or D is reached. Starting with 1 level, the number of levels visited in each attempt d is incremented by 1. If the querier does not get a positive response, it initiates another attempt[3] after increasing d. Hence, the number of contacts visited in each attempt is given by $\sum_{i=1}^{d}(NoC)^i$, and the upper limit on number of transmissions is $[(R+r).\sum_{d=1}^{D}\sum_{i=1}^{d}(NoC)^i]$. Again, these are only illustrative theoretical estimates. Detailed simulation results are given in the evaluation section.



(a)



(b)

Figure 3. Examples of search policies with D=2, R=r=NoC=3: (a) The single-shot policy forwards the request in one attempt reaching level-1 and level-2 contacts (called contact-1 and contact-2), (b) The level-by-level lbl policy forwards the request in multiple attempts with increasing the visited levels. In the first attempt only '3' level-1 contacts are visited. In the second attempt 3 different level-1 contacts are visited and the request is forwarded to '9' level-2 contacts. It is clear that different policies reach different parts of the network. Single-shot may not be able to achieve good coverage near Q with low NoC.

[3] For lbl and step, the querier waits for time t between attempts; t α d.(R+r). Single-shot does not use t, since it uses a single-attempt per request.

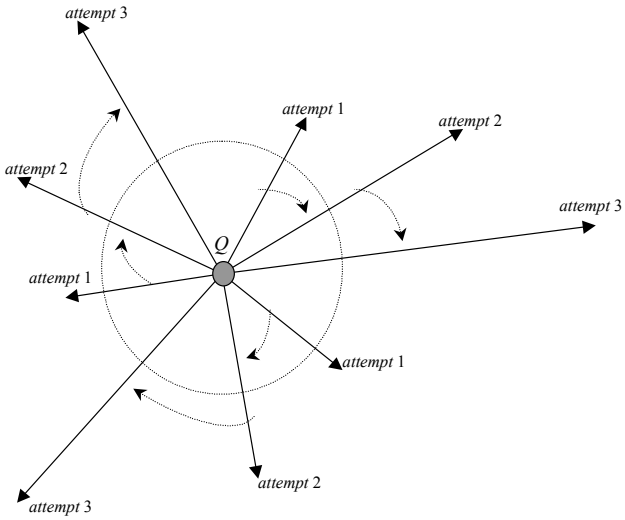


Figure 3(c). The rotation-like effect between attempts in step and lbl increases network coverage. In lbl, attempti reaches the leveli contacts.

Exponential *Step* Search Policy

Step search is similar to *lbl*, except that the number of levels visited in attempt i, d_i, is incremented exponentially; i.e., $d_i=2d_{i-1}$ (e.g., 1,2,4,8..) until the target is found or d_{max} is reached, where d_{max} is the first d_i that satisfies the inequality $2d_{max}>D$ for $D>2$. (For $D\leq 2$, $d_{max}=D$). For example, if $D=20$ then $d_{max}=16$. For the *step* policy the upper limit on number of transmissions is given by $[(R+r).\sum_{d=1,2,4,8...}^{d_{max}}\sum_{i=1}^{d}(NoC)^i]\cdot$

An example of *lbl* (or *step*) with $D=2$, $R=r=3$, and $NoC=3$ is given in Figure 3 (b). Schemes *lbl* and *step* are identical for $D=2$. It is important to note that level-1 contacts visited on the first attempt are not necessarily the same as level-1 contacts visited on the second attempt. This is due to the randomization of the first border selection. From Figure 3 this effect is clear, and it results in *different policies reaching different parts of the network*. It seems, however, that *single-shot* may not reach parts of the network near the querier, but those parts are likely to be reached by *lbl* and *step* due to the randomization (rotation-like) effect, as illustrated in Figure 3 (c). We shall investigate this effect further in the evaluation section. Another performance implication due to the different policies is in the request latency. Intuitively, *single-shot* incurs less delivery time than the other policies because it completes its search in a single attempt. *Step* search is expected to complete its search in less number of attempts than *lbl*. We shall investigate this further in the evaluation section.

IV. Request Forwarding and Processing

The rules for processing the requests are the same for all of the above policies. This section presents details of request processing, forwarding, and loop prevention.

IV.A. The Request Message

The request message contains the target ID, which could be the node ID or the resource key. The destination-ID in the request message contains the ID of the border node (or the direction border). The request message also contains the maximum number of levels to visit (d) for that attempt, the querier ID (Q) and a sequence number (SN). For every new attempt the querier issues a new SN.

IV.A.1. Loop Prevention and Re-visit Avoidance

As the message is forwarded, each node traversed records the SN, Q and P, where P is the previous hop node, from which the request was received. P may be used later to send a response to the querier, Q, through the reverse path. If a node receives a request with the same (SN,Q), it drops the request. This provides for loop prevention and avoidance of re-visits to the covered parts of the network. This mechanism is important to keep the overhead from exponentially growing at each level. The recorded (SN,Q,P) is kept as soft state, associated with a short timer, adding robustness against querier failure and SN wrap around. Also, if a contact reached at any level finds the querier in its own proximity, indicating a loop, then the contact drops the request.

IV.A.2. Search, Processing and Forwarding

A contact (or border) receiving the request, first performs a target search in its local proximity information. If the target is found, the request is delivered and a response is forwarded on the reverse path (if needed), with each node forwarding the response to its recorded previous hop, P. Otherwise, further processing is performed as follows.

In order for a recipient of a request message to determine which functions to perform, and whether it is a contact, two fields are included in the request message; *level-count* and the *hop-count*. Initially, the level-count is set to d and the hop-count set to ($R+r$). The hop-count is decremented with every hop and is checked:

- If hop-count reaches '0', then the receiving node acts as a contact. A contact decrements the level-

count and resets the hop-count field to (R+r). If level-count reaches '0' the contact drops the request. If level-count is not '0', the contact selects *NoC* borders (using *POR* and *ROR* as in section 3), and sends the request to those borders.

- If the hop-count is not '0', and the current node ID is same as the destination ID of the request message, the receiving node acts as a border node. It selects a direction border (using *POR* as in section 3), and sends the request towards it.
- Otherwise, the request is simply forwarded to the next hop to the destination.

Note that only nodes along the forwarding path need to process the request. No processing is necessary by the neighbors of those nodes. This is an important feature that allows nodes in the sensor network to sleep and wake-up to save energy without affecting the behavior of the protocol. This feature, along with *on-the-fly* contact selection, distinguish this work from other related work that does not consider sensor nodes sleep/wake-up cycles (e.g., in ZRP the query is broadcast and is processed by nodes neighboring those forwarding the message, and in dominating set or backbone based approaches, sleep/wake-up cycles trigger major re-configuration of the network).

Evaluation and Comparison

In this section we study the various design parameters of *CAPTURE*. In addition, we compare *CAPTURE* to other related approaches including flooding, expanding ring search [4] and ZRP ([11][12]).

Particularly, for *CAPTURE*, we attempt to systematically answer the following questions: (1) How many contacts (*NoC*) to choose? (2) What is the best contact distance (r)? (3) What should be the maximum depth (D) for the search? (4) How should we set the proximity radius (R)? (5) What is the best search policy, *single-shot*, *lbl* or *step*? (6) How does replication affect the protocol performance? and (7) Is there a specific combination of settings that performs well for a wide variety of networks?

The main performance metrics include communication overhead and the request success rate. Overhead is measured in number of transmitted and forwarded messages during query resolution and proximity maintenance. Note the trade-off between success rate and overhead; the more the success rate the more the overhead and vice versa. In order to balance these conflicting goals we introduce a penalty for request failures. Any request failure for the contact-query mechanism will be recovered using flooding. Hence, *one scheme used in our simulations is contact-based search, if failed then fallback to flooding*. Since the penalty of flooding is quite expensive it will be natural for our best performing parameters to avoid resorting to flooding by achieving a very high request success rate. We define the term 'contact search' to refer to only that part of the protocol that uses contact-based search without fallback to flooding. For clarity we present results for the *contact search* only – in average number of transmitted messages per query, and success rate (or packet delivery ratio) – and present results for contact search with fallback to flooding.

IV.A.3. Simulation setup

We use extensive simulations to investigate the design space parameters and evaluate the performance of our proposed protocols under various settings of *r, NoC, D* and replication. We also evaluate the overall communication overhead for our architecture. This overhead consists of two components: (a) proximity establishment and maintenance, and (b) request (or query) overhead.

In our architecture, each node keeps track of other nodes in its proximity. To keep storage requirements and proximity overhead at a reasonable limit, we limit the number of nodes per proximity to 100 nodes. This limit is achieved for all simulated networks by setting *R=3*.

Nodes	Area (mxm)	Node Degree	Border Nodes	Proximity Nodes
200	1000x1000	7.6	15.1	35
500	1400x1400	8.9	20.5	44.8
1000	2000x2000	9.1	21.7	46.8
2000	2800x2800	9.7	24.7	52.9
4000	3700x3700	11	30.3	62.2
8000	4800x4800	13	38.8	77.8
16000	6500x6500	14.3	44.6	88.2
32000	9200x9200	14.3	45	88.9

Table 1. Networks used in the simulation. Nodes are initially randomly distributed. Number of border and proximity nodes are given for R=3.

[4] We investigated several variants of expanding ring search with various constant and exponential TTL increments. All these variants were found to perform worse than flooding due to the large network diameter. For brevity we omit results for the expanding ring search.

The transmission range (*tr*) is set to 110m. We study a wide range of networks, as shown in Table 1. We vary the area of the network to maintain network good connectivity, and to keep the proximity nodes under 100 (for proximity radius of *R*=3). *N* nodes are randomly placed in a square of 'ℓm x ℓm'.

We developed a discrete event simulator and implemented the protocols under study. The first part of the results systematically discusses the effect of *r, NoC, D,* and *replication*, on the performance of the different search policies. For this set of simulations we use the 1000 node topology in Table 1. Then we present scalability analysis for *CAPTURE*, flooding and ZRP. For this set of simulations we use specific parameter settings for *CAPTURE* policies based on the first set of simulations and use various topologies.

Each data point represents an average of 10 simulation runs with different random seeds. Low variability between runs was observed. Querier-target pairs were chosen randomly. 1000 such queries were performed in each run; i.e., a total of 10,000 queries (or requests) for each data point.

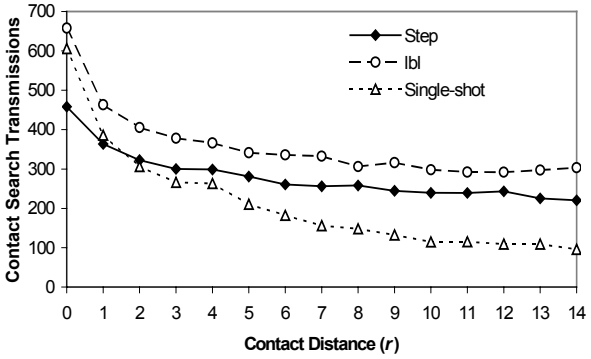
IV.B. Overhead per Query

The overhead per query is affected by the various design parameters. Here we investigate the effect of the contact distance (*r*), the number of contacts (*NoC*), the maximum depth (*D*), and the degree of replication. Our aim is to understand the behavior of the different *CAPTURE* policies with the various design parameters, and study trends to aid us in identifying desirable parameter settings. For each parameter we show the overhead and success rate for the contact search only (without fallback to flooding), then the overhead for the contact search with fallback to flooding.

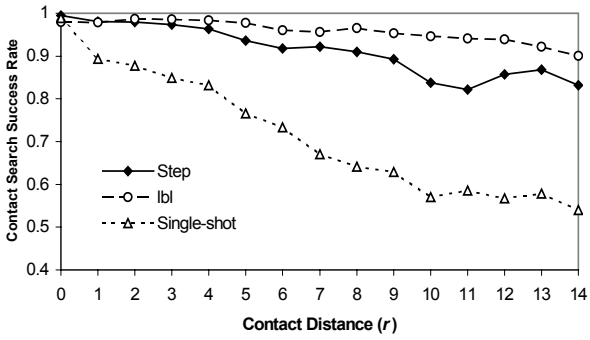
IV.B.1. Effect of contact distance (r)

We have conducted several experiments with various *NoC* and *D*. We only show partial results that represent the trend, using *NoC*=3 and *D*=33 in a 1000 node network. Figure 4 shows the effect of varying *r*. Figure 4 (c) indicates favorable settings for the different search policies. In general, as *r* grows, the contacts' location extends farther away from the querier's proximity.

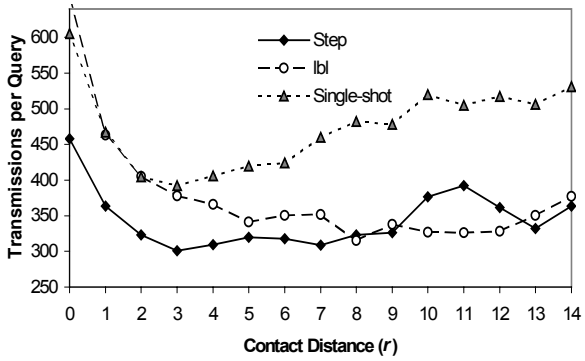
For single-shot policy, with $r \leq 3$ the transmission overhead decreases as *r* increases, then it rises noticeably with further increase in *r*. This is due to a sharp drop in the contact-based request success rate above *r*=3. Remember that drop in success rate translates into fallback to flooding, which consistently produces more transmissions. The drop in success rate is due to reduced coverage of areas near *Q*'s proximity or the contacts' proximities. This effect was qualitatively illustrated earlier in Figure 3. Hence, higher values of *r* (*r*>3) are not preferred for single-shot.



(a) Overhead of the contact search per query (without fallback to flooding)



(b) Success rate for the contact search (without fallback to flooding)

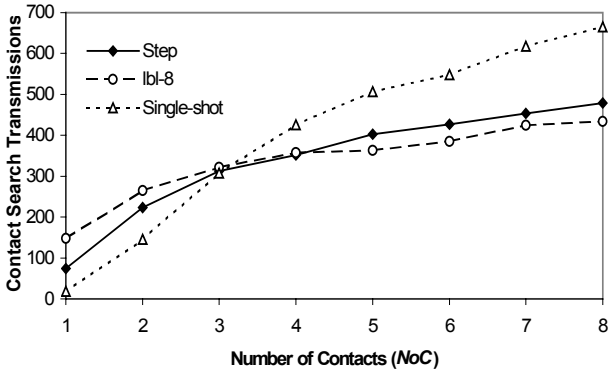


(c) Overhead per query for contact search with fallback to flooding

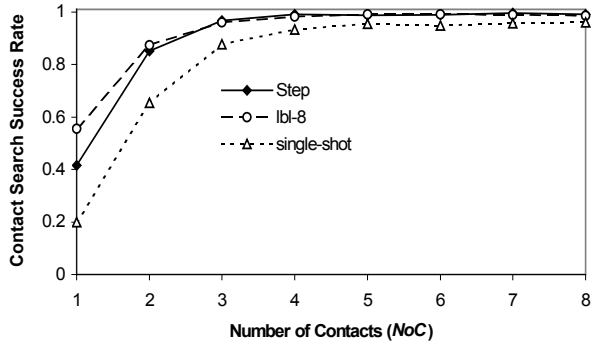
Figure 4. Effect of Contact Distance (*r*)

On the other hand, for *lbl* and *step* policies, the trend is different. Due to multiple attempts and randomization of contact selection between attempts, *lbl* and *step* can maintain good coverage

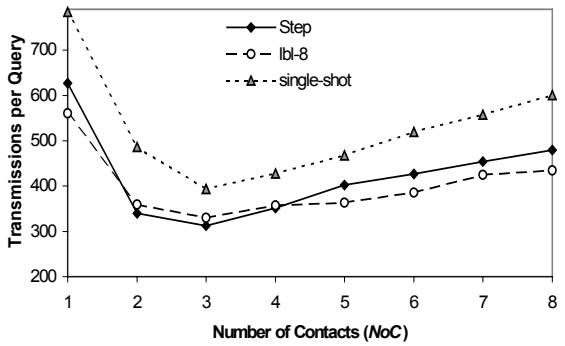
with increase in *r*. Hence, high request success rate is achieved with less transmissions due to fallback to flooding. Further increase in *r* generally leads to more transmissions due to drop in success rate. At very low values of *r* (e.g., $r \leq 2$), although *lbl* and *step* achieve high success rate, they also incur added overhead due to proximity overlap between *Q* and level-1 contacts (and in general between level-*i* contacts and level-*i*+1 contacts). This overlap reduces with increase in *r*, with the best values around 3-8 hops (3 being best for *single-shot* and *step* and 8 being best for *lbl*).



(a) Overhead of the contact search per query (without fallback to flooding)



(b) Success rate for the contact search (without fallback to flooding)



(c) Overhead per query for contact search with fallback to flooding

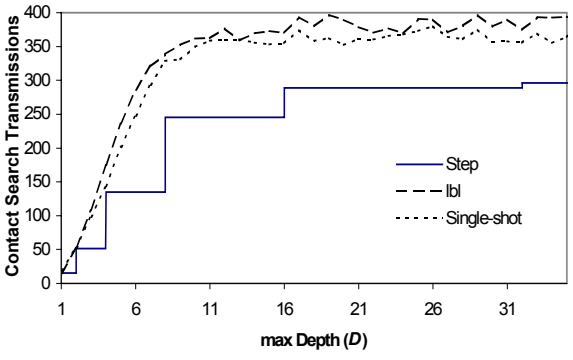
Figure 5. Effect of Number of Contacts (*NoC*)

IV.B.2. Effect of Number of Contacts (NoC)

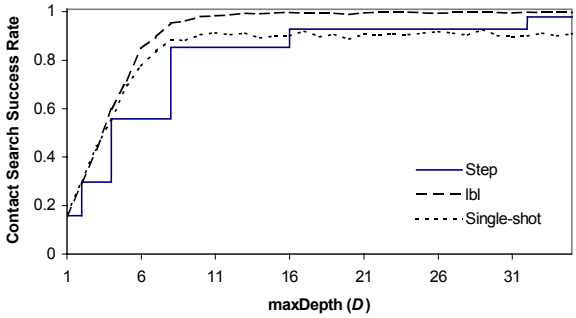
To understand the effects of *NoC* on the different policies we evaluate different favorable settings of *r* based on our previous analysis. Results in Figure 5 are shown for $r=3$ (for single-shot), $r=8$ (for *lbl*) and $r=3$ (for *step*). For all policies, a very low number of contacts ($NoC<3$) incurs high number of transmissions due to fallback to flooding because of low success rate. Increasing *NoC* increases success rate until almost all requests succeed then we see an increase in overhead due to additional (unnecessary) search branches with increase in *NoC*. From Figure 5 (c) we see that for all search policies the best setting is at $NoC=3$.

IV.B.3. Effect of Maximum Depth (D)

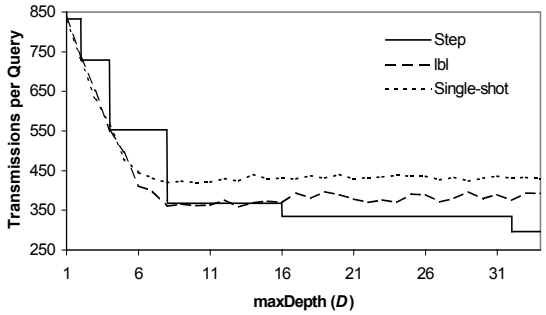
Using favorable settings for *r* and *NoC* we investigate the effect of increasing the maximum contact depth, *D*. Results in Figure 6 show that increasing *D* generally decreases the transmissions by increasing the success rate and subsequently reducing fallback to flooding. It is *not* the case that increasing *D* exponentially increases the number of contacts visited. Although the number of potential contacts grows, loop prevention drastically reduces the number of visited contacts. After certain values of *D* (10 for *lbl*, 13 for single-shot and 33 for *step*) most requests (97.5% or more) become successful and overhead almost saturates. Note that $D=33$ for *step* translates into a maximum of 6 attempts. Increase in *D* does not necessarily translate into increase in number of attempts. The average number of attempts (for $D>10$) is 3.1 attempts for *step*, 4.0 for *lbl*, and of course 1 for single-shot. For larger networks we expect this number to rise and we suspect that *D* required for high success rates may rise as well. We shall return to this point later in this section.



(a) Overhead of the contact search per query (without fallback to flooding)



(b) Success rate for the contact search (without fallback to flooding)



(c) Overhead per query for contact search with fallback to flooding

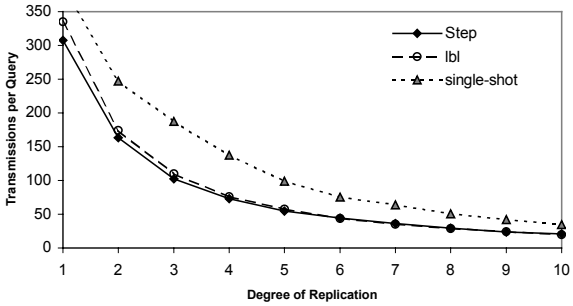
Figure 6. Effect of maximum depth (*D*)

IV.C. Effect of replication

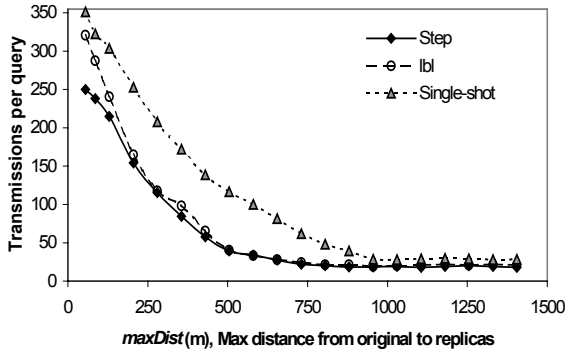
The degree of replication represents the number of copies of a target object within the network (degree of 1 means no replication). We present simulations to show the effect of replication on the protocol performance. We use various replication models. The first assumes the replicas are randomly distributed across the network, for which we use replication degrees from 1 to 10. The second replication model assumes replication degree of 10, but restricts the maximum distance between a replica and the original copy. We do not consider the cost of replication. So, for the shown results these replications must have been created naturally (due to the phenomena monitored or sensed) or placed when the network was deployed. We also assume a model of *anycast* in which the source is looking for *any* one of the copies and not some or all of them.

Figure 7 (a) shows the effect of random replication on the different search policies. We see significant decrease in overhead in all policies with the increase in replication degree. For *lbl* and *step* this mainly happens due to the decrease in the average required number of attempts before success (i.e., reaching any replica). This number drops drastically from 3.1 (without replication), to 2.4 (with 1 replica) to 1.1 (with 8 replicas). For single-shot the drop occurs due to branch termination upon success.

Figure 7 (b) shows the effect of restricted replication for degree of replication of 10, where any replica is allowed to exist only at a random distance [0,*maxDist*] from the original copy. We investigate various settings of *maxDist*. From the figure, it is clear that the overhead decreases with the increase in *maxDist*, which means wider distribution of the replicas. However, the overhead reduction saturates at some point reaching that of random distribution of replicas around *maxDist*~650m for *step* and *lbl* and around ~950m for *single-shot*.



(a) Effect of random replication



(b) Effect of restricted replication

Figure 7. Effect of replication

Related schemes, e.g., flooding and ZRP, do not show as drastic improvement with replication. Among related schemes, expanding ring search responds best to replication. However, related schemes incur overhead significantly higher than *CAPTURE* protocols. For the rest of this paper we do *not* assume replication.

Note on the effect of traffic patterns

Note that the performance of flooding and ZRP is not greatly affected by the location of the target

(whether near or far from the querier), since there is no notion of search termination. By contrast, *CAPTURE*'s *step* and *lbl* do terminate the search when the target is found and no further attempts (with larger search depth) are triggered. Simulation results for various traffic patterns are omitted for brevity.

IV.D. Scalability Analysis of Total Overhead

In this section we evaluate the scalability of the *CAPTURE* protocols. In particular, we want to investigate how the overhead grows with the increase in number of nodes in the network. There are two main overhead components for *CAPTURE*: (a) query overhead, and (b) proximity maintenance. In the previous section we have studied the overhead per query. The overall query overhead is a function of the overall number of queries, which in turn is a function of the query rate (query/sec) per node, the number of nodes, and the simulation time. Proximity overhead, on the other hand, is a function of the degree of mobility (m/s), the number of nodes in a proximity, the number of nodes in the network and the simulation duration. In order to be able to combine these two overhead components in a meaningful way we represent the query rate as a function of mobility. We also normalize all the measures per node per second per m/s of mobility. We use a metric called *QMR* (query-mobility-ratio, q) defined per node as query/s/(m/s) or simply query/m. Let us call the proximity overhead *Z(R)*, defined in terms of packets transmitted and is a function of the proximity radius, R. *Z(R)* has units of 'transmissions per sec per node per (m/s)'. Also, let us call the transmissiosn per query for *single-shot*, *lbl* and *step* are T_{single}, T_{lbl} and T_{step}, respectively. The overall query overhead for *lbl* (for example), $T_{Qlbl} = q.T_{lbl}$. The units of T_{Qlbl} are in 'transmissions per sec per node per m/s', compatible with *Z(R)*. The total overhead for *lbl* (for example) becomes $T_{Tlbl}=Z(R)+T_{Qlbl}$.

Our goal in this section is to obtain trends and comparisons of total overhead for *CAPTURE* protocols as well as related schemes, for a wide range of query rates and over various networks (200 to 32,000 nodes) (See Table 1).

IV.E. Related schemes

We compare our protocols to flooding, ZRP and a variant of ZRP that we call ZRP*. Let T_{flood} be the transmissions per query for flooding. In a network of N nodes, the request is transmitted by N-1 nodes; that is, $T_{flood} \approx N-1$. In ZRP [11][12] the querier sends the request to its zone borders, and the borders send it to their borders, so on. Query control is used to reduce redundant querying. Request messages are broadcast (or multicast) hop by hop and nodes along the forwarding path (and their neighbors) record the request information. Requests that are sent to previously visited borders are terminated. For a zone of radius R, each node keeps track of nodes up to $2R$-1 hops. We modify ZRP such that a request from a border node suppresses the requests from its neighboring border nodes. In our simulations, this reduced ZRP's overhead without decrease in query success rate. We call this modified version ZRP*.

Next, we analyze scalability of the query overhead, then proximity overhead, followed by analysis of total overhead.

IV.F. Scalability of Query Overhead

Simulation results are analyzed for the different *CAPTURE* protocols. Parameter setting was based on earlier analysis. For single-shot we present results for r=3, NoC=3. The maximum depth, D, was increased to 65 to achieve better success rate for single-shot.

For *step* and *lbl* we used D=33 and NoC=3. For *step we used* r=3, and for *lbl* r=8. Results are presented in Figure 8. Remember that the scheme used in the simulations always achieves (at least) 97.5% success rate by falling back to flooding if the contact-based search fails. For all network sizes it is clear that the *step* policy achieves the best performance (with success rate of 97.5% or better for all network sizes). *lbl* achieves similar success rate but with more overhead.

Single-shot with (NoC=3, r=3) exhibits an interesting behavior. For small-medium networks, *single-shot* performs worse than *lbl*, but for large networks (above 4000 nodes) its performance approaches that of *step* and becomes superior to that of *lbl*. The reason for this can be explained by examining the query success rate. For sizes below 4000, lower success rates (82-89%) are reached, the success rate increases to (94-97%) for 4000-8000 nodes. After 8000 nodes this setting achieves 97.5% and above success rate.

More specifically, with the increase in number of nodes there are more branches to search, giving more chance to cover, at higher contact-levels, what was not covered at lower contact-levels (near Q), thus increasing the success rate for contact-based search and decreasing the overall overhead.

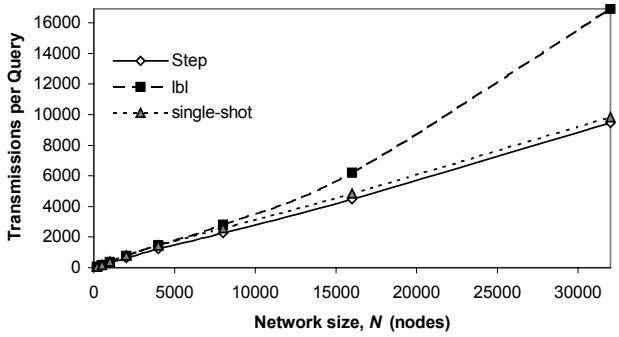


Figure 8. Scalability of query overhead for *CAPTURE* policies

IV.F.1. Latency Analysis

Figure 9 shows the trend for average number of attempts with increase in nodes. The single-shot average is always around 1, and the largest increase occurs for *lbl* (reaching 13.7 attempts for 32,000 nodes). *Step* scales well, with 5.2 average attempts for 32,000 nodes. Based on this analysis, we feel that *lbl* provides no advantage over *single-shot* or *step*. *Step* provides the best performance in terms of number of transmissions, and possesses desirable scaling characteristics in terms of delay. Single-shot exhibits the best delay among these policies and may be set to achieve good performance at higher scale. One nice feature of *step*, however, is its persistent good performance over a wide spectrum of network sizes, with the setting (*NoC*=3,*r*=3,*D*=33 [max attempts=6]).

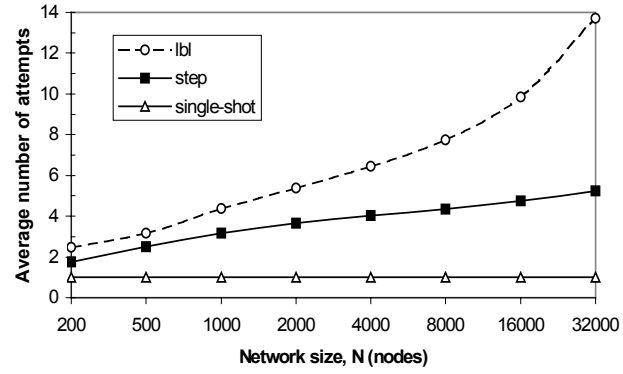


Figure 9. Average attempts per query for *CAPTURE* policies

It is important to note that the purpose here is not to decide on a winning policy in all situations. Rather, by developing an understanding between the different characteristics of the different policies, each policy may have an advantage depending on the requirement (e.g., for getting consistently the lowest energy we may use step, but for getting the best response delay we may use single-shot). It is conceivable that different policies be used for different kinds of requests. This is achieved by simply setting the right parameters in the request message. For example, to implement single-shot, the querier sets the maximum level of contacts to visit (d) to the maximum depth (D) and performs a single attempt.

IV.F.2. Comparison with Related Schemes (Query Overhead)

We compare our *CAPTURE* protocols to flooding and *ZRP** for the various networks. In Figure 10 we show the results for query overhead for *step*, *lbl*, *single-shot, flooding*, *ZRP* and *ZRP**. It is quite clear that there is a drastic improvement in performance using contacts, quantified next.

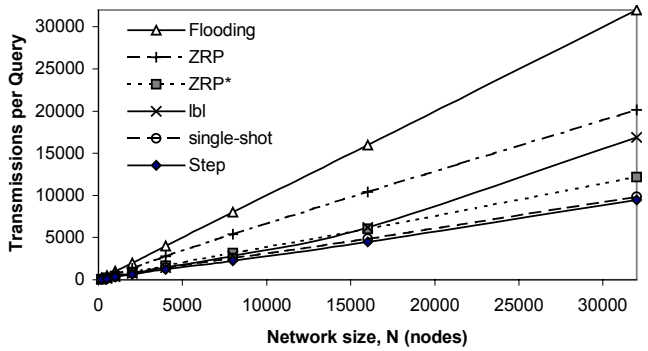


Figure 10. Ratio of *step* query overhead vs. other approaches

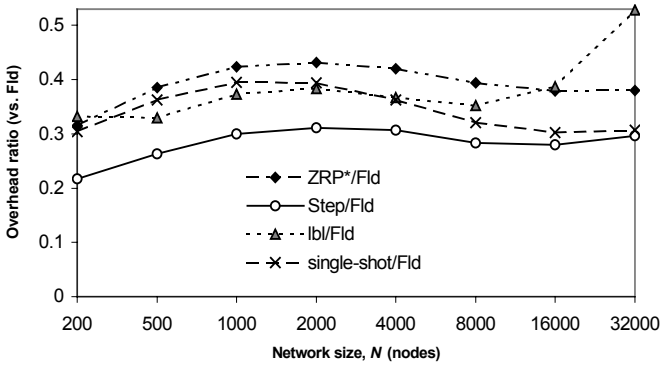


Figure 11. Query overhead of *CAPTURE*, flooding and ZRP

Figure 11 shows the query overhead ratio for the various protocols with respect to flooding, with various network sizes. It is clear that *step* incurs the least overhead for all network sizes with per-query overhead ratio between 0.21 (for small networks) and 0.3 (for large networks). *Single-shot* performs worse than *step*, but approaches 0.3 for large networks. ZRP* has overhead ratio of 0.31 (for

small networks) and 0.44 (for larger networks).

IV.F.3. Proximity Overhead

The proximity overhead includes the energy consumed by the link state message exchange. Alternatively, we can use more efficient proximity maintenance protocols (e.g., [15] or other). For link state, the proximity exchange is in the form of broadcast messages by each node, up to R-1 hops away. This exchange increases linearly with mobility (with more link changes). So, this overhead is normalized with respect to mobility using $Z(R)$. The proximity overhead is also a function of the number of nodes in the proximity. This number is a function of R, and increases with the proximity area (i.e., with R^2). Figure 12 shows $Z(R)$ for *CAPTURE* and for ZRP. (ZRP uses link exchange of $2R$-1 to employ efficient early termination).

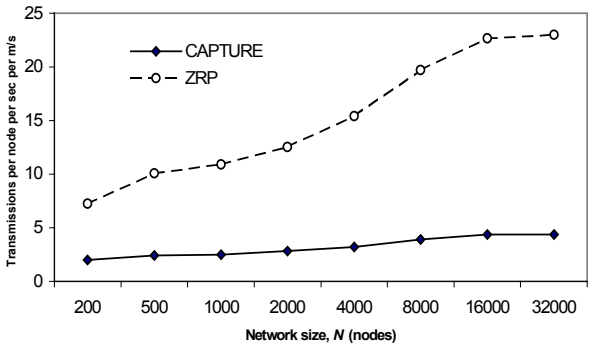


Figure 12. Normalized proximity overhead for the basic proximity R-1=2, and the extended zone of 2R-1=5 used by ZRP

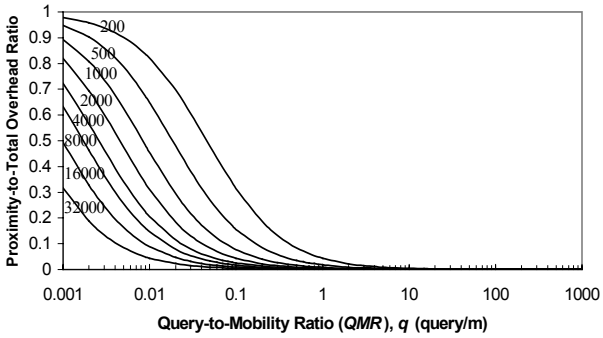


Figure 13. Ratio of proximity maintenance overhead to the total overhead of *CAPTURE*

IV.F.4. Comparisons of Total Overhead

The total overhead is the combined effect of proximity maintenance and query overhead. As was mentioned, metrics used to measure these two components need to be normalized in order to be combined in a meaningful way. This normalization is per second per node per mobility unit (m/s). The equation for total overhead formulated above for the *step* policy is as follows:

$$T_{Tstep}=Z(R)+T_{Qstep}= Z(R)+q.T_{step}$$

To understand the effect of proximity maintenance on the total overhead we plot the ratio $\frac{Z(R)}{Z(R)+q.T_{step}}$ against q in Figure 13. We can observe that the contribution of the proximity maintenance overhead is most significant for low values of q and rapidly reduces with increase in q.

For flooding, no proximity overhead is incurred, so $T_{Tflood}=T_{Qflood}= q.T_{flood}$. We evaluate the total overhead ratio, OHR_{flood}, of *step* to the other protocols. We get:

$$OHR_{flood} = \frac{T_{Tstep}}{T_{Tflood}} = \frac{Z(R)+q.T_{step}}{q.T_{flood}},$$

Figure 14 shows OHR_{flood}, as function of the *QMR* (query-to-mobility ratio) q (query/m) per node. We note that a logarithmic scale was used for q to resolve the rapid drop in the total overhead ratio.

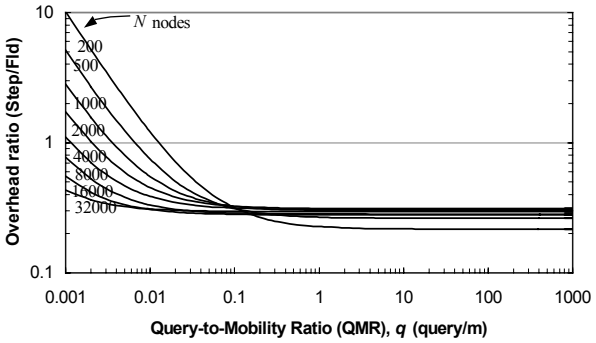


Figure 14. Total overhead ratio vs. flood (*OHR_{flood}*)

As shown in the figure, for very low values of q=0.001 query/m and small to medium network sizes (200-4000 nodes) flooding performs better. This is due to the very low number of queries triggered as compared to the proximity maintenance overhead[5]. Note that, in general, zone-

[5] We suspect that a scenario of very low q, indicating relatively inactive nodes, is unlikely in large-scale ad hoc networks. A more likely scenario is that when the nodes are inactive for extended periods of time, they may go to sleep or 'off' mode and not participate in proximity exchange. Maintaining zone information without being active is not desirable.

based protocols perform well when the proximity overhead is amortized over a reasonable number of queries in order to achieve overall gain. Hence, the gains of the contact-based approach will appear with increase in the query rate, q. Also, note that for non-mobile (static) or low mobility sensor networks, q will be high.

Let the *cross-over-point (CoP)* be the point after which *CAPTURE* shows benefits over flooding. For a small network size (200 nodes) *CoP* is q=0.01query/m. For medium size networks (500-4000 nodes) *CoP* is q=0.001-0.006query/m. For large networks (8000-32000 nodes) *CoP* is below q=0.001 query/m. For q>0.1the overhead ratio ranges between 0.2 and 0.31 for all network sizes.

V. Related Works

Perhaps the simplest form of resource discovery is global *flooding*. This scheme does not scale well as we have shown. Hence, it is our design goal to avoid global flooding. Expanding ring search uses repeated flooding with incremental TTL. This approach and its derivatives also do not scale well as we have shown.

Related work on smart flooding has been proposed in [15][20][21]. These techniques attempt to reduce the redundancy inherent in flooding, and may be integrated in our work to provide more efficient zone establishment instead of regular link state protocol. One major difference between smart flooding and *CAPTURE* is that smart flooding reduces the redundant messages in querying every node in the network, whereas *CAPTURE* attempts to create a small world and only queries a small number of nodes (the contacts) on the order of the degrees of separation from source to target. In relatively sparse networks (some of which we include in our study) smart flooding will not be very effective since there is no significant redundancy in flooding anyway.

Approaches in ad hoc networks that address scalability employ hierarchical schemes based on clusters or landmarks[5][13][14]. Other mechanisms use minimum dominating or covering sets [23][22]. These architectures, however, require complex coordination between nodes, and are susceptible to major re-configuration (e.g., adoption, re-election schemes) due to mobility or failure of the cluster-head or landmark. Furthermore, usually the cluster-head becomes a bottleneck. More importantly, in sensor networks nodes may have a *sleep/wake-up* schedule to conserve energy. This may require significant re-configuration of the network in case of the above tightly coupled hierarchies. We avoid the use of complex coordination schemes for hierarchy formation, and we avoid using cluster-heads.

In GLS [6] an architecture is presented that is based on a *grid* map of the network. Nodes recruit location servers to maintain their location. Nodes perform location updates and lookups using an ID-based algorithm. The algorithm proposed in [2] and [7] uses global information about node locations to establish short cuts or friends, and uses geographic routing to reach the destination. These are useful architectures when a node knows the network map, its location, and the ID of the target node. These assumptions may not hold in our case. By contrast, in our architecture, a source node may be looking for a target *resource* residing at a node with an ID *unknown* to the source node. Conceptually, however, GLS may still be used (after modification) to hash any resource name using similar algorithm to [6]. In cases where the network boundary is not known or dynamically changing (due to node mobility or failure), or where there are gaps in the grid map, GLS-like approaches will not work.

Location-based or geographic routing is becoming a very attractive alternative when location information is available. Geographic routing (e.g., GPSR[34]) is stateless and only needs local neighbor location information to forward the packet towards the destination. An inherent assumption in geographic routing, however, is that the destination address is known. In the applications we target in this study (i.e., resource discovery and queries) the destination location is *not* known a priori. Hence, even if location information is available (which may not be the case for many sensor networks) an efficient resource discovery protocol is still needed. *CAPTURE* may be used in conjunction with geographic routing to provide efficient location and resource discovery. Once the location of the resource/node is known geographic routing may be used to deliver the packets.

In ZRP [8][11][9][9] the concept of hybrid routing is used, where link state is used intra-zone and on-demand border-casting (flooding between borders) is used inter-zone. A good feature in ZRP is that a zone is node-specific. Hence, there is no complex coordination. We use the concept of zone in our architecture. However, we avoid border-casting by using contacts out-of-zone. The main concepts upon which contacts were designed (small world graphs and contacts) are fundamentally different than ZRP's bordercasting. We have compared the performance of a variant of ZRP and

our approach through simulations in Section 5. The contact-based approach incurs significantly lower overhead.

For object tracking, in SCOUT [14] an architecture was presented that is based on hierarchy formation. Using concepts borrowed from landmark hierarchy [11], where wireless devices self-configure in a multi-level hierarchy of parent nodes and children nodes. Each level is associated with a radius to which the device advertises itself. To configure the hierarchy complex mechanisms for promotion, demotion, and adoption are used. These mechanisms are susceptible to major re-configurations with dynamics. This is stated clearly in the work. The root nodes of the hierarchy use global flooding to send advertisements. If the root nodes sleep, fail or move, new root nodes may be elected, and all nodes in the network may need to re-map all tracked objects. This does not scale well under dynamic conditions.

Directed diffusion [24] provides a data dissemination paradigm for sensor networks. This scheme targets continuous queries in sensor networks. Without location information about the sensors or the sensed information, directed diffusion uses flooding to advertise the interests from sinks to sources throughout the network. Data delivery occurs over diffusion paths re-inforced by the sources. Interests are periodically refreshed by the sinks. For continuous queries the cost of flooding is amortized over the amount of information exchanged over possibly extended periods of time. For one-shot queries or with mobility, directed diffusion with flooding may incur excessive overhead especially in large-scale networks. In such situations, our contact-based architecture may be integrated with directed diffusion to discover resources in a scalable manner instead flooding.

In [29] a data-centric storage architecture was proposed for sensor networks. The architecture uses distributed hash tables that map objects into locations in the network. The object/data is stored in (or retrieved from) the node nearest to that location. Gegoraphic routing is used to route the data/request to that location. Data is replicated in nodes near to that location in case of movement of the node nearest to that location. This scheme may be well-suited for scenarios in which geographic information is available, and in which the network boundary is fixed and known a priori such that consistent hashing leads to a location within the boundaries of the network. This scheme was not designed for location-free networks, or when the boundaries of the network change with time.

In [26][27][28] approaches are proposed that treat the sensor network as a database. Concepts of data-centric and in-network processing are emphasized, and query resolution is presented as one of the essential mechanisms for sensor networks. The *CAPTURE* architecture presented in this paper fits in that model, and provides a very efficient alternative for query resolution of one-shot, simple queries for potentially replicated data.

The *ACQUIRE* algorithm [33] was proposed for complex query resolution in sensor networks, where the query message is active, querying up to d hops away in each step. This is similar to the zone concept used in this paper. The amortization factor, c, has some parallels to the query rate, q. The *ACQUIRE* paper presented an analytical framework to evaluate query resolution mechanisms. We plan to leverage such framework to model *CAPTURE* in future work.

We first presented the general idea of contacts at a very high level, with no details or evaluation in [41]. Our previous work on contact-based architectures and contact-selection protocols includes *CARD* [42][43], and *MARQ* [44]. Both *CARD* and *MARQ* use a pro-active approach that selects and maintains contacts. *CARD* uses zone-edge information to select useful contacts, while *MARQ* exploits mobility by choosing contacts moving away from the zone. The *CAPTURE* architecture, on the other hand, uses a re-active approach, by choosing contacts dynamically, on-the-fly, when the request is issued. The reactive nature of this protocol reduces the maintenance overhead and is more resilient to network dynamics.

Rumor routing is proposed in [32] as an alternative to reduce flooding overhead for interests in directed diffusion. It was designed for continuous (long-term) queries. In [25] diffusion mechanisms are presented in which sensors are selectively queried for correlated data based on gain vs. cost.

Other data dissemination protocols for sensor networks include SPIN [30], Gossiping, and LEACH [31]. These protocols are designed for data dissemination (not query resolution for potentially replicated data that we address in our scheme).

VI. Conclusions and Future Work

We have presented a novel architecture for resource-discovery and small-transfers in large-scale sensor networks. For such applications, the

overhead incurred for obtaining high quality routes is not justified as compared to the transfer of the actual data. Hence, the main design goal in such target applications is to reduce communication overhead and power consumption, rather than route optimization. In our architecture each node knows information about nodes within its proximity, up to *R* hops away. To service a request, we provide a simple, yet very effective, mechanism by which the querying node selects a number of contacts outside its proximity. These contacts act as short cuts to transform the network into a small world and reduce the degrees of separation between the querying node and the requested target (or resource).

Our architecture does not use tightly coordinated hierarchical schemes. This renders our scheme more robust and resilient to mobility. We do not assume or rely on availability of geographic information.

The main contributions of this paper include:

- Introducing the contact-based *CAPTURE* architecture for power-efficient search in large-scale sensor networks
- Designing a simple, on-demand, location-free contact selection protocol for effective overlap reduction
- Supporting various search policies and presenting mechanisms for loop-prevention to improve performance
- Evaluating, in detail, the different dimensions of the design space and scalability of our protocols
- Comparing performance of our protocols against flooding and ZRP using extensive simulations over a wide array of networks and request rates

Our results show that significant savings may be achieved using our contact-based techniques. For medium to and high request rates, *CAPTURE* incurs overhead as little as 20% of flooding overhead and achieves even greater savings over variants of expanding ring search. We provide different search policies that may be suitable for different situations. A search policy may be simply chosen by setting different parameters within the request message at the time of query. Among the policies investigated: *step* achieves the best energy efficiency and responds best to replication, while single-shot achieves minimum delay. The study also shows reasonable settings of parameters that work well for a wide range of network sizes (from 200-32000 nodes).

For future work we plan to investigate the utility of our architecture in location-aware networks, where *CAPTURE* can still be used for location discovery. Once the target location is known geographic routing is used for delivery.

References

[1] D. Watts, S. Strogatz, “Collective dynamics of ‘small-world’ networks”, *Nature* 393, 440 (1998).

[2] J. Kleinberg, “Navigating in a small world”, *Nature,* 406, Aug. 2000.

[3] C. E. Perkins, E. M. Royer, Ad-hoc On-Demand Distance Vector Routing, IEEE Wksp. Mobile Comp. Sys. And Apps., Feb. 1999.

[4] D. B. Johnson, D. A. Maltz, Dynamic Source Routing in Ad-Hoc Wireless Networks, Mobile Computing, 1996, pp.153-181.

[5] P. Guangyu, M. Gerla, X. Hong, “LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility”,MobiHoc’00

[6] J. Li, J. Jannotti, D. Couto, D. Karger, R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing", *ACM Mobicom* 2000.

[7] L. Blazevic, S. Giordano, J.-Y. Le Boudec *“*Anchored Path Discovery in Terminode Routing”. *Proceedings of the Second IFIP-TC6 Networking Conference (Networking 2002), Pisa, May 2002.*

[8] M. Pearlman, Z. Haas, “Determining the optimal configuration for the zone routing protocol”, *IEEE JSAC*, p. 1395-1414, Aug 1999.

[9] Z. Haas, M. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", IETF Internet draft for the Manet group, June '99.

[10] Z. Haas, M. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol", *ACM SIGCOMM* '98.

[11] Z. Haas, M. Pearlman, “ZRP: A Hybrid Framework for Routing in Ad Hoc Networks”, Book Chapter in Ad Hoc Networks, Editor C. Perkins, Addison Wesley, pp. 221-254, 2001.

[12] Z.J. Haas, M.R. Pearlman, "The Perforamnce of Query Control Schemes for the Zone Routing Protocol," *ACM/IEEE Transactions on Networking*, vol. 9, no. 4, pp. 427-438, August 2001.

[13] P. F. Tsuchiya, "The Landmark Hierarchy: A

new hierarchy for routing in very large networks", CCR, Vol. 18, no. 4, pp. 35-42, Aug. 1988.

[14] S. Kumar, C. Alaettinoglu, D. Estrin, "SCOUT: Scalable object tracking through unattended techniques", *IEEE ICNP* 2000.

[15] J.J. Aceves, M. Spohn, "Bandwidth-Efficient Link-State Routing in Wireless Networks", Book Chapter in Ad Hoc Networks, Editor C. Perkins, Addison Wesley, pp. 323-350, 2001.

[16] A. Helmy, "Small Large-Scale Wireless Networks: Mobility-Assisted Resource Discovery", *LANL Technical Report cs.NI/0207069, featured in the Technology Research News (TRN) Journal (trnmag.com)*, August 2002.

[17] A. Helmy, "Small Worlds in Wireless Networks", *IEEE Communications Letters*, pp. 490-492, Vol. 7, No. 10, October 2003.

[18] C.-C. Chiang, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel", *Proc. IEEE SICON'97*, Apr.1997.

[19] J. Liu, Q. Zhang, W. Zhu, J. Zhang, B. Li, "A Novel Framework for QoS-Aware Resource Discovery in Mobile Ad Hoc Networks",*ICC '02*

[20] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum et L. Viennot, Optimized Link State Routing Protocol, *IEEE INMIC* 2001.

[21] S. Ni, Y. Tseng, Y. Chen and J. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network", *ACM Mobicom*, '99.

[22] W. Lou, "A cluster-based backbone infrastructure for broadcasting in mobile ad hoc networks", IEEE/ACM IPDPS WMAN, April 03.

[23] H. Gupta, S. Das, Q. Gu, "Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution", *ACM,MobiHoc* 03.

[24] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *ACM MobiCom*, August 2000.

[25] M. Chu, H. Haussecker, F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks." *Int'l J. High Performance Computing Applications*, 2002.

[26] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, S. Shenker, The Sensor Network as a Database, Technical Report 02-771, Computer Science Department, UCLA, September 2002.

[27] P. Bonnet, J. E. Gehrke, and P. Seshadri, "Querying the Physical World, " *IEEE Personal Communications*, Vol. 7, No. October 2000.

[28] P. Bonnet, J. Gehrke, P. Seshadri, "Towards Sensor Database Systems," *Mobile Data Management*, 2001.

[29] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker, "GHT – A Geo-graphic Hash-Table for Data-Centric Storage," *First ACM WSNA Workshop*, 2002.

[30] W.R. Heinzelman, J. Kulik, and H. Balakrishnan "Adaptive protocols for information dissemination in wireless sensor networks," *ACM MobiCom*, pp. 174-185, Aug. 1999.

[31] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan "Energy-efficient communication protocol for wireless microsensor networks," 33rd International Conference on System Sciences (*HICSS*), 2000.

[32] David Braginsky and Deborah Estrin, "Rumor Routing Algorithm For Sensor Networks," First *WSNA* Workshop, September 2002.

[33] N. Sadagopan, B. Krishnamachari, A. Helmy, "The *ACQUIRE* Mechanism for Efficient Querying in Sensor Networks", *First IEEE ICC SNPA Workshop* May 2003.

[34] B. Karp, H. Tung, "Greedy Perimeter Stateless Routing for Wireless Networks", *ACM MobiCom* 2000.

[35] K. Seada, A. Helmy, "Rendezvous Regions: A Scalable Architecture for Service Provisioning in Large-Scale Mobile Ad Hoc Networks", *ACM SIGCOMM*, Refereed poster, 2003.

[36] D. B. Johnson, D. Maltz, J. Broch, "DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks", Book Chapter in Ad Hoc Networks, Editor C. Perkins, Addison Wesley, pp. 139-172, 2001.

[37] Y. Hu, D. Johnson, "Caching Strategies in On-Demand routing protocols for Ad Hoc wireless networks", *ACM MobiCom*, 2000.

[38] Yih-Chun Hu and David B. Johnson. Ensuring Cache Freshness in On-Demand Ad Hoc Network Routing Protocols. *Proc. POMC*

Workshop on Principles of Mobile Computing, pp. 25-30, October 2002.

[39] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, *ACM MobiCom,* 1998.

[40] David A. Maltz. On-Demand Routing in Multi-hop Wireless Ad Hoc Networks. Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 2001.

[41] A. Helmy, "Architectural Framework for Large-Scale Multicast in Mobile Ad Hoc Networks", *IEEE International Conference on Communications (ICC 2002)*, Vol. 4, pp. 2036-2042, April 2002.

[42] A. Helmy, S. Garg, P. Pamu, N. Nahata, "Contact Based Architecture for Resource Discovery (*CARD*) in Large Scale MANets", *IEEE/ACM IPDPS Int'l Workshop on Wireless, Mobile and Ad Hoc Networks (WMAN)*, pp. 219-227, Apr 2003.

[43] A. Helmy, S. Garg, P. Pamu, N. Nahata, "*CARD*: A Contact-based Architecture for Resource Discovery in Ad Hoc Networks", *ACM Baltzer Mobile Networks and Applications (MONET) Journal, Kluwer publications, Special issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks*. To appear (1st Quarter 2004).

[44] A. Helmy, "Mobility-Assisted Resolution of Queries in Large-Scale Mobile Sensor Networks (*MARQ*)", *Computer Networks Journal - Elsevier Science (Special Issue on Wireless Sensor Networks)*, Vol. 43, Issue 4, pp. 437-458, November 2003.

Appendix

A. On-demand Queries with Caching

A technique that is most commonly used in ad hoc routing is on-demand with caching; e.g., DSR-like [4] or AODV-like [3]. In this appendix we build a caching model for those approaches and evaluate its efficacy for large-scale wireless networks for small transfers. Note that most (if not all) previous studies on on-demand ad hoc routing protocols used 40-100 node networks, with long lived connections [4] [36] [37] [38] [39] [40]. We are not aware of any previous study for cache performance with the scale of the network or for small transfers.

The caching model follows the dynamic source routing (DSR) design [4] [36]. A source looking for a traget (or a destination) triggers a route request (RREQ) on demand. First, the source looks up its own local cache for a path to the destination. If local cache is not found then the source sends a query to its first hop neighbors and they perform cache lookup. If a cached path is not found, or if the found cache does not result in positive response from the target (e.g., due to invalidity of the cache), then the source floods the route request throughout the network.

We make minor adjustments to DSR to make it more suitable to small transfers. For example, the traget responds only to the first query (as opposed to responding to several route requests as in DSR for long transfers such that the source gets multiple routes). This reduces the reply traffic overhead. Also, no intermediate caches are used. That is, when a local or neighbor path cache is not found for the destination, then the request is flooded and is answered by the target. A node more than 1 hop from the source does not respond to the source (since its cache may be invalid). This reduces the number of potential floods to reach the target.

The reply (from the target) traverses the reverse path to the source, and nodes along the path (and their neighbors) cache the path information (i.e., aggressive caching).

When a cached path is used and is found to be invalid or out-of-date, it is attached to the flooded route request to invalidate all copies of that path in the network.

Performance of DSR depends heavily on cache performance. We define the following metrics for evaluating the cache performance:

(1) *Cache hit ratio (p)*: is the ratio of route requests that are found in the local or neighbor cache.

(2) *Valid cache ratio (q)*: is the ratio of the cache hits that are valid (i.e., not out-of-date).

(3) *Cache efficacy (p.q)*: is the ratio of route requests that are answered correctly from a valid cache.

Simply put, if we ignore the cost of local and neighbor cache lookups and the cost of replies, then the overhead of DSR per query is given by:

$$T_{DSR}=(1-p.q).T_{flood}$$

where T_{flood} is the number of transmissions triggered by flooding per query, and p, q as stated above.

We observe the performance of the cache for simulation settings similar to those used above in the paper. Each data point represents an average of 10 simulation runs with different random seeds. Querier-target pairs were chosen randomly. 1000 such queries were performed in each run with 10 queries per sec; i.e., a total of 10,000 queries (or requests) for each data point. Each node moves using a "random waypoint" model [39]. The node chooses a random destination and moves toward it with a constant speed chosen uniformly between zero and a maximum speed (*Vmax*). When the node reaches the destination, it chooses a new destination and begins moving toward it immediately. These simulations do not involve a pause time. A cache warm-up period was allowed before measurements were taken in each run.

The network area and radio range were setup for all topologies to have almost a constant average node degree (i.e., number of neighbors per node) equivalent to 40 nodes with 250m radio range in 1000mx1000m network area, or 200 nodes with 110m radio range in 1000mx1000m network area.

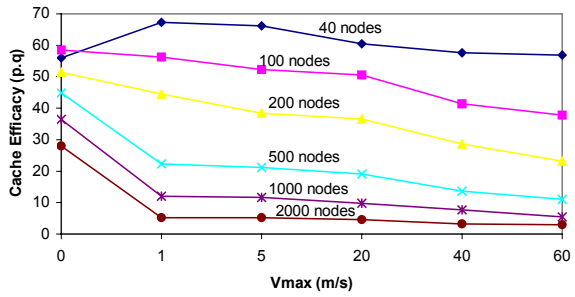


Figure A.1. The cache efficacy with various velocities and various network sizes. The cache performance degrades drastically with scale of the network and with (even very low) mobility.

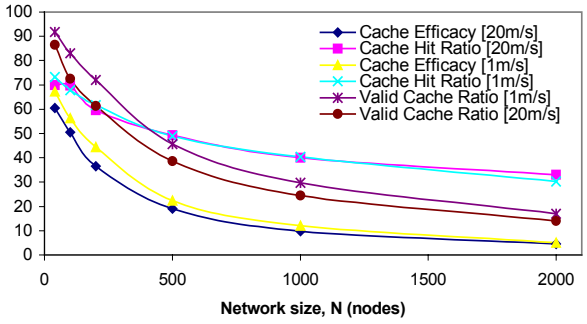


Figure A.2. Cache hit ratio (*p*), valid cache ratio (*q*) and cache efficacy (*p.q*) with the network size for 1m/s and 20m/s.

The results are given in figures A.1 and A.2. Figure A.1 shows the cache efficacy with the max velocity (Vmax) for various network sizes. For very small scale networks (40-100 nodes) the efficacy is relatively high (~50-70%) especially for the static or low mobility cases. This result is consistent with previous studies on on-demand ad hoc routing. As the number of nodes increases, however, the cache efficacy drops dramatically, even for very low mobility (1m/s), to ~10% for 1000 nodes and ~5% for 2000 nodes!

Figure A.2. gives a closer look at the cache metrics. It seems that the cache performance depends on mobility, but much more so on network size. The cache hit ratio (*p*) drops from ~73% (for 40 nodes) to ~30% (for 2000 nodes). The more drastic drop occurs in the valid cache ratio (*q*), from ~92% (for 40 nodes) to ~14% (for 2000 nodes), which brings the overall cache efficacy (*p.q*) down.

For moderate to large-scale networks (above 1000 nodes) the performance of on-demand routing with caching approaches flooding, where the on-demand routing protocol resorts to flooding more than 90% of the time due to cache miss or invalid cache hit.

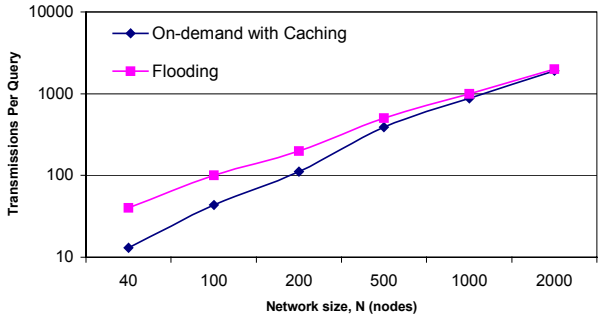


Figure A.3. Per-query overhead of on-demand routing and flooding for the same simulation setup, with 1m/s mobility.

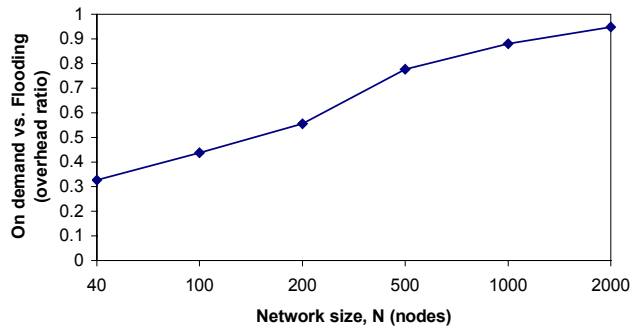


Figure A.4. Ratio of On-demand routing overhead to flooding overhead. As the network size increases caching performance degrades and on-demand routing overhead approaches flooding overhead.

Figures A.3 and A.4 show the simulation results (in terms of number of transmissions per-query) for the DSR-like on-demand routing as compared to flooding.

Based on this analysis we believe that on-demand routing with caching approaches are not suitable for resource discovery, query resolution, and small transfers in large-scale wireless networks.