Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Computer Networks 43 (2003) 437–458
COMPUTER NETWORKS
www.elsevier.com/locate/comnet

Mobility-assisted resolution of queries in large-scale mobile sensor networks (MARQ)

Ahmed Helmy *

Department of Electrical Engineering—Systems, University of Southern California, 3740 McClintock Ave, EEB 232, Los Angeles, CA 90089-2562, USA

Abstract

One of the most crucial aspects of the design of sensor networks is provisioning of efficient query resolution and resource discovery. In many cases sensor networks are expected to be large-scale, and in some cases these sensors maybe installed on moving objects, rendering the query resolution problem even more challenging. Flooding techniques, including global flooding or expanding ring search techniques, may be very inefficient in large-scale networks, especially in wireless (spatial) networks where the diameter of the network tends to be quite high. More so is the case when queries are *one-shot* and frequent.

In this study, a novel architecture is presented for query resolution in large-scale mobile sensor networks. A salient feature of our architecture is that it takes advantage of mobility to increase the efficiency of query resolution. The architecture borrows from the concept of small worlds and introduces the concept of *contacts* that act as short cuts to reduce the degrees of separation between the sources of the query and the targeted objects. Contacts are initially chosen from nearby neighbors, as they move away they discover new neighbors and hence become more effective in query resolution. Unlike conventional approaches for routing protocols, our primary design goal is *not* to optimize routes or response delays, but to reduce communication overhead. This is particularly important in energy-constraint environments, as are many sensor networks, particularly for one-shot queries, where the communication is short lived. We design our protocols to be scalable, self-configuring, and highly adaptive to mobility. In fact, it utilizes mobility.

We evaluate our protocols through extensive simulations and present a detailed analysis of its performance. We further compare our approach to other query resolution protocols. Our results clearly indicate the drastic improvement obtained by using contacts, especially in high mobility scenarios. For non-replicated objects, we obtain 60–70% improvement over zone routing approaches, 80–90% improvement in communication overhead over flooding, and even greater improvements over expanding ring search approaches. Our protocols respond extremely well to replication, as the number of transmitted packets per query drops significantly.

Keywords: Sensor networks; Mobile wireless networks; Query resolution; Energy-efficient protocols; Network simulation

1. Introduction

'Sensor networks' is a new and emerging field with many potential applications. The design of sensor networks, unlike traditional computer

* Tel.: +1-213-821-1329; fax: +1-213-740-4418.
E-mail address: helmy@usc.edu (A. Helmy).

1389-1286/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1389-1286(03)00353-0

networks, is geared towards specific applications, or classes of applications that have common characteristics. Those characteristics may vary widely from one application-domain to another, in terms of communication and query semantics, energy-constraints, capabilities and mobility patterns, among others. As such it is quite hard (and sometimes even undesirable) to design general protocols for all operating conditions. Design of sensor networks can, and in fact should, take advantage of domain-specific information. Many such domain-specific sensor networks are designed to collect and disseminate data upon request, and may be viewed as databases. Hence, one of the most crucial aspects of the design of sensor networks is provisioning of efficient query resolution and resource discovery. Examples of query resolution include searching for sensor readings of a given value (or range) or sensors with certain capabilities. In many cases, sensor networks are expected to be large-scale, consisting potentially of thousands of nodes, and in some cases these sensors maybe installed on moving objects, rendering the query resolution problem even more challenging. Examples of such cases include networks of distributed robots, and person, animal or object monitoring/tracking, among others.

Semantics of data collection and query may be quite different for different applications and situations. For example, queries maybe continuous or one-shot. For continuous queries the interest in the data collected may extend over periods of time beyond the query time. Also queries may be frequent or infrequent, simple or complex (asking for multiple pieces of information that may exist on multiple sensors). The data may also be unique or replicated.

Our design goal is to provide efficient query resolution for one-shot, frequent, simple queries. In addition, our protocols should be able to respond well to data replication when applicable.

We do not assume dependence on availability or precision of any location or geographic information. That is, we assume that node, data, and interest locations are unavailable, partially available or imprecise (such that geographic routing cannot be used), or that the boundaries of the network are unknown or dynamic (due to node mobility) such that consistent location hashes cannot be used to store-retrieve information. In other words, there is no fixed frame of reference to provide a rendezvous mechanism between producers and consumers of data. This renders our architecture applicable even in cases where location (e.g., GPS) information is not available (e.g., indoors or for very small sensors). Taking advantage of location information if and when it becomes available is not addressed in our study, and can be potentially pursued in future work. So, for this work we assume that location information is not available.

In such situations, simple approaches of flooding may be used. Flooding techniques, including global flooding or expanding ring search techniques, may be very inefficient in large-scale networks, especially in wireless (spatial) networks where the diameter of the network tends to be quite high. More so is the case when queries are *one-shot*, and for potentially replicated objects.

In sensor networks, data tend to be of low-bandwidth and volume, and hence communication due to inefficient control protocols (e.g., query resolution) are likely to dominate the overall cost of communication. Also, in many cases of sensor networks, energy consumption of communication far exceeds that of processing. [1] Since wireless sensors, in general, have very limited power, it becomes essential to provide efficient control protocols for query resolution. This is the problem we address in our study.

We present a novel architecture for query resolution in large-scale mobile sensor networks, for the aforementioned environments. We refer to our architecture as mobility-assisted resolution of queries (MARQ). In this architecture each node uses a proactive protocol to maintain information about other nodes in its *zone*, up to R hops away. Each node also maintains information about a

[1] Note that energy to provide mobility may exceed the communication overhead. In the situations we target, however, we assume that mobility is either provided by the sensor-carrying person or object (e.g., animal), or that provisioning of mobility (e.g., in robots) is done by a separate source that can not be shared with the communication subsystem in the wireless sensor, or that mobility is already part of the robots task or mission.

small number of nodes farther away, called *contacts*. A challenging problem is to choose and maintain useful contacts with reasonable overhead. For that we introduce a mobility-assisted contact selection protocol, coupled with a light-weight maintenance mechanism. During a query, instead of using flooding to search for an answer, only the contacts are queried for information they maintain about their zones. The contacts may in turn query their contacts, and so on, until the answer is found. A salient feature of our architecture is that it takes advantage of mobility to select far away contacts to increase the efficiency of query resolution. The architecture borrows from the concept of small worlds [1–4,16] and introduces the concept of contacts that act as short cuts to reduce the degrees of separation between the sources of the query and the targeted objects. Degrees of separation in this context represent the number of nodes (or contacts) to query before reaching the target. Contacts are initially chosen from nearby neighbors, as they move away they discover new neighbors and hence become more effective in query resolution. Unlike traditional approaches for routing protocols, our architectural design goal is not route optimization, but reducing communication overhead. This is especially useful in energy-constraint environments, as are many sensor networks, particularly for one-shot queries, where the discovered route is not used for extended communication and the connection is short-lived.

We make a conscious design choice of trading off route and delay optimality for reduction in overall communication overhead. We borrow from existing work on zone routing, but extend beyond that work to provide much more efficient query resolution by using contacts and by changing the design trade-offs as mentioned above.

Major contributions of this work lie in the introduction of the concept of contacts as short cuts in the wireless network (that attempt to construct a small world), and the proposal of novel mobility-assisted protocols for contact selection and maintenance (MACS). That is in addition to the drastic improvement in query resolution overhead.

We evaluate our protocols through extensive simulations and present a detailed analysis of its performance. We further compare our approach to other query resolution protocols. Our results clearly indicate the drastic improvement obtained by using contacts, especially in high mobility scenarios. For non-replicated objects, we obtain 60–70% improvement over edge-of-zone flooding (ZRP-like [12]), 70–90% improvement in communication overhead over probabilistic and simple flooding, and even more improvement over expanding ring search approaches. These improvements are enhanced significantly further with replication.

The rest of the paper is outlined as follows. Section 2 provides an overview of the contact-based architecture. Section 3 introduces the mobility-assisted contact selection protocol. Section 4 gives elaborate description of the contact-based query mechanisms. Evaluation and comparison simulation experiments and analysis of results is detailed in Section 5. Related work is discussed in Section 6. Section 7 concludes and provides future work directions.

2. Contact-based architecture overview

First we start by stating our assumptions and the context in which our architecture is applicable; then we provide an overview of our contact-based architecture.

2.1. Assumptions

First, we state the assumptions upon which our architecture is built. (1) The source of the query may not know the ID of the target node that holds the resource. (2) Nodes only have local knowledge of their neighbors (e.g., using 1 hop Hello or data link connectivity). (3) Nodes do not know their own location or any other geographical location of any other node (i.e., our architecture does not require GPS or any other GPS-less distance estimation techniques). [2] (4) Infrastructure-less network:

[2] Availability of such location information may simplify our architecture. Studying such simplifications is part of on-going and future work.

we assume there are *no* well-known servers or landmarks.

These assumptions differentiate our work from other works that require any of the above elements.

2.2. Architectural overview

We propose what we call a loosely coupled simple hierarchy. It is loosely coupled because each node has its own view of the network (its zone), and it is simple because there are no complex coordination mechanisms to elect cluster-heads or leaders. In our architecture, each node knows a number of neighboring nodes in a neighborhood or zone. The zone may be defined, for example, by a number of hops R. Information about nodes within the zone are obtained by an intra-zone link state protocol. Outside of the zone, a node may maintain (a small number of) contacts. A contact is chosen at r hops distance from the selecting node. The contact may be maintained up to $r_{\max}$ hops away. The idea behind the contacts stems from the small world concept [16], for increased coverage and reachability of other nodes. A contact outside a node's zone will also have its own zone, thus providing an extended view of the network. This is shown in Fig. 1. Our architecture is not to be confused with other cluster-head or landmark based hierarchies. In our scheme there is virtually no coordination between nodes in a zone. There are no *special* nodes, whose failure or movement may incur significant reconfiguration overhead. By contrast, we shall show that our scheme incurs non-significant overhead with mobility.

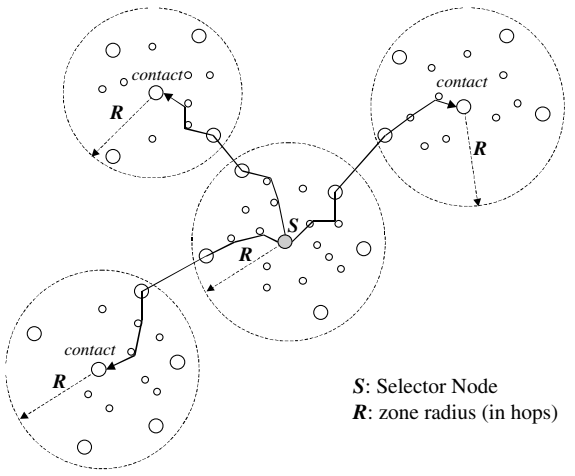

Fig. 1. A node in the contact-based architecture chooses a small number of contacts outside its zone to increase its network view during query resolution or resource discovery.

The main components of our architecture include: (a) zone establishment, (b) contact selection and maintenance and (c) query processing and forwarding.

Zone establishment is performed by each node independently by sending link state messages to R hops away. We call R the zone radius. For zone establishment we use mechanisms similar to those in ZRP [9–12]. Alternatively, other energy-efficient link state [15] mechanisms may be used.

In this paper, we focus on the remaining components of the architecture; namely contact selection and maintenance and query processing and forwarding. We think of contacts as short cuts to the outside world (i.e., out-of-zone), that provide useful information when needed. To reduce the discovery delay these contacts are established *in anticipation* of queries. With network dynamics and mobility it may be quite expensive to establish and maintain routes to all far away contacts. Instead, we propose to establish candidate contacts from within the zone. As these candidates move out of the zone they become contacts and can be used in the query process, thus taking advantage of mobility. One unique feature of our architecture is that its performance improves with increased mobility, as we shall show in the evaluation section.

Not all nodes in the network need to establish contacts. In fact, if all nodes establish contacts this may constitute a large overhead for large-scale networks. Only a small subset of nodes, called selectors, independently choose to establish contacts. Selectors are not fixed, but are dynamic and may be chosen (in a distributed manner, without extra overhead) in a way that achieves load balancing. A selector keeps a list of (a subset of) its zone borders, and chooses its contacts from those border nodes that move out of the zone. This choice takes advantage of zone information to attempt to reduce overlap between contact zones. Once the contact is out-of-zone a simple contact discovery mechanism

is invoked to keep track of it. Routes maintained to contacts are loose (perhaps sub-optimal) routes. Since each node knows about neighboring nodes up to R hops away through the zone maintenance protocol, the contact route (that has initial length of R hops) may be extended up to R^2 hops without any extra overhead as the nodes en-route move away. This is explained further in Section 3.2. Once a contact (or one of its en-route nodes) moves too far away, then the contact is dropped and another is chosen. We investigate the choice of the number of selectors, contacts and the value of R as part of our study.

Once these contacts are chosen they may be used in the query process for resource discovery. A querying node sends messages to its contacts, and their contacts and their contacts, so on, up to maximum contact level or until the object is found. We introduce mechanisms to prevent loops and revisits of already-searched zones.

We evaluate our protocol through extensive simulations and compare it to various other approaches for resource discovery, including flooding, expanding ring search (and its variants), and the border-casting (or edge-flooding) approach. [3] We evaluate the query success ratio and the total overhead (due to query and the architecture), and use them as basis of our comparison. We note that the zone and contact establishment and maintenance vary with mobility and simulation time and are amortized over the number of queries performed (which is in turn a function of the query rate). We consider all these factors in our evaluation and show for which range of query rates and mobility our approach is best suited.

3. Mobility-assisted contact selection

One of the main challenges that we need to address in our architecture is the effect of mobility and its dynamics on contacts. We pose this challenge in the form of the following question. How will contacts be selected with reasonable overhead under mobility conditions to significantly reduce query overhead? To attempt to answer this question we propose a mobility-assisted contact selection (MACS) scheme.

The problem of contact selection is challenging for two main reasons. First, mobility seems an adversary, providing sometimes random node movement and contributing to link and path failures. Second, the selecting node is likely to know little or no information about the mobility characteristics and capabilities of nodes in far away regions of the network, and hence may not be able to make intelligent decisions as to which node may be useful to resolve a query. We argue that in order to achieve significant reduction of query overhead contacts need not be randomly placed in the network (a concept that follows from the small world concept [4]). This argument is substantiated by our results later in the paper. We instead propose a scheme that selects contacts from the selector's zone, and tracks those contacts as they drift out-of-zone. This idea is illustrated in Fig. 2.

Our proposed scheme *takes advantage of mobility*. In our approach, a selecting node, S, makes initial selection of a list of candidate contacts (CCs) from its own zone. These are nodes that lie within R hops away. S knows routes to these nodes via intra-zone routing. This way, S may also collect information about CCs' mobility or abilities. This information may be piggybacked over the intra-zone link state exchange. The future contacts are to be chosen from this list of CCs. Once these candidate contacts move out-of-zone, overlap between their zones and S's zone will be reduced and

[3] Border-casting was introduced for the zone routing protocol ZRP [9]. Although designed for routing, we believe that the general approach is also attractive for resource discovery. We study a variant of ZRP/border-casting that we refer to as 'edge-flooding' (Efld) in this document.

[4] In small world analysis on relational graphs it is suggested that adding a few 'random' links achieves significant reduction in degrees of separation. Degrees of separation in our case is the number of intermediate contacts to query before reaching the target. In [16], we show that for wireless networks (that belong to spatial graphs with boundaries) placement of contacts may be restricted in distance while still achieving significant reduction in degrees of separation. Treatment of this problem is out of scope of this document.

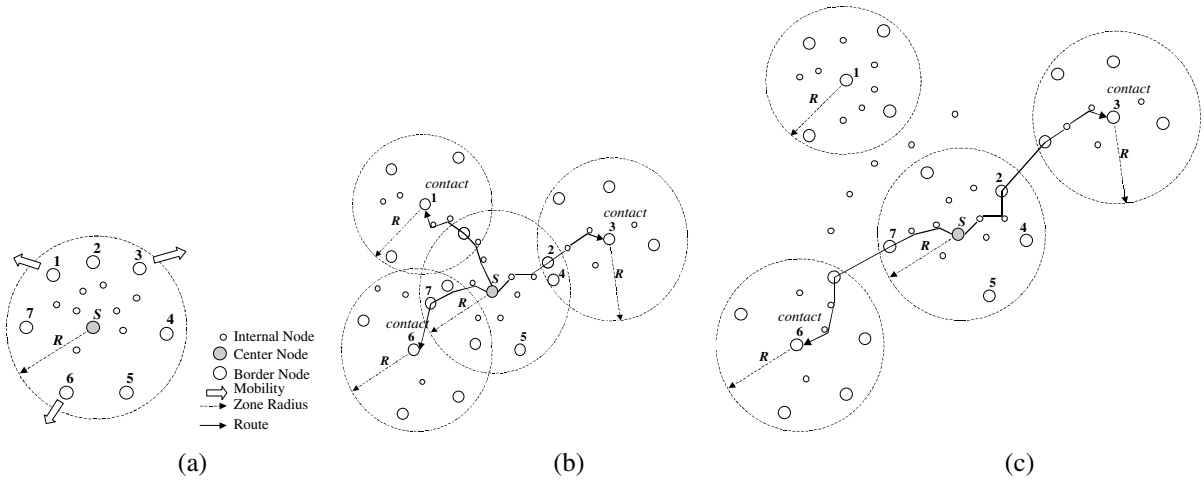

Fig. 2. Example of zoning, contacts and effect of mobility: (a) Zone for source node S is shown (with radius R). Border nodes are numbered (1–7). Nodes 1, 3 and 6 are moving/drifting out-of-zone. (b) Radii for the drifting nodes are shown. S stays in contact with the drifting nodes, which enables it to obtain better network coverage with low overhead. (c) After moving away, contact nodes drift up to a point where their zones no longer intersect with S's zone. In this example, S maintains contact with those nodes not more than $(2R + 1)$ hops away, i.e., nodes 3 and 6, and loses contact with node 1 as it drifts farther than the contact zone.

the added network view (or coverage) will be significant, as shown in Fig. 2. Following we describe in more detail the contact selection, maintenance and their integration.

3.1. Contact selection

A selector node, S, starts selecting a list of candidate contacts (CCs) from the zone (i.e., within R hops). Based on its mobility pattern, a node on the CCs list may get promoted to contact or get dropped (or evicted) from the candidacy list based on *promotion/eviction* rules. By mobility pattern we mean the sequence of distances (in hops) between these candidates and the selecting node, over time. When and if a candidate crosses the promotion boundary (PB) it is considered a contact and may be queried during searches. Once a contact crosses the upper or lower eviction boundaries (UEB, LEB) it is evicted from the *contact list* and is not queried in further searches (see Fig. 3). That is, if a node moves too far away (beyond the eviction boundary) it is harder to maintain, whereas if it comes closer to S its new zone may overlap with that of S, and is evicted in both cases. We investigate two contact selection protocols called *border-based* and *neighbor prediction-based* contact selection protocols.

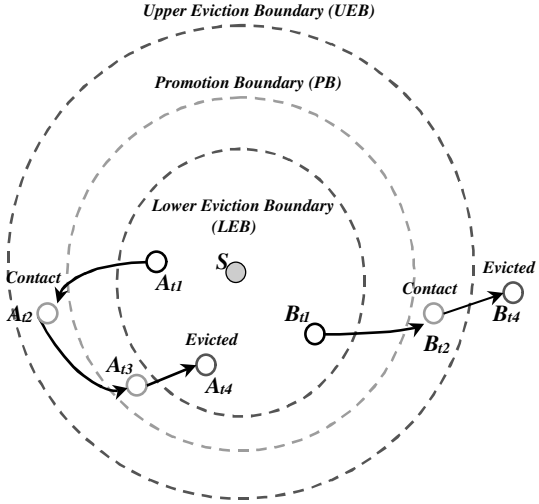

Fig. 3. Promotion and eviction boundaries for S: nodes A and B originally in S's zone, get promoted at time t2 when they cross the promotion boundary (PB), then they get evicted from the contact list at time t4 when they cross the lower or upper eviction boundary.

Border based contact selection protocol: Selection of candidate contacts is done from nodes at R hops, i.e., at the border of the zone. Nodes R hops away are tracked, i.e., S keeps track of its movement/distance over time. If border nodes move closer to S, they are evicted. If they cross the promotion boundary, those candidates become contacts. When the contacts cross an eviction boundary they are evicted.

Neighbor prediction-based contact selection protocol: This protocol takes advantage of the fact that S readily knows the routes/hop distance to nodes within its zone (i.e., within R hops away). It also takes advantage of the likelihood that (even in random way point movement) mobility often remains constant for short periods. In this protocol, node S selects neighbors (those nodes that are 1 hop away) to *track* their movement. When a neighbor node becomes 2 hops away, then 3 hops away, this sequence may indicate that this neighbor is heading out-of-zone and has a high probability to continue moving away. Other neighbors, that do not show this consistency in mobility, may take longer (if ever) to get to the desirable region and thus waste more resources before becoming useful, and so are evicted from the CCs list. We expect this simple prediction scheme to help identify good candidates that have higher probability of becoming contacts in the case where there is regular mobility pattern. But in case of random movement, as our initial results show [16] this prediction scheme may not prove advantageous over the border selection scheme, but will lead to delays in contact selection. For the rest of this paper we use a border based contact selection scheme.

3.2. Contact maintenance

As the CCs move, S keeps track of their distance (in hops) through intra-zone routing, and when they move out-of-zone a lightweight local route discovery mechanism is used (see Fig. 4). The precise route need not be kept at all times, but only a loose route is maintained to reduce maintenance overhead. So long as every node en-route to the contact has the next node en-route in its zone (i.e., at most R hops away), then S can route to the contact.

3.3. Integrating selection and maintenance

In order to reduce overlaps and gaps between S's zone and the zones of its contacts we use two heuristics. First, we note that the initial route to the contact is of R hops (e.g., S–A–B–Z as in Fig. 4(a)), which may be allowed to grow up to R^2 hops without extra overhead (as shown in Fig. 4(c)). On average, a contact will be at $R + (R^2 - R)/2$ hops away, with overlap or gap (between S's zone and the contact's zone) of $|(R^2 - 3R)/2|$. So in order to minimize the overlap or gap we set $R = 3$ (this also incurs very reasonable link state overhead for the zone [12]). Second, in order to reduce overlap

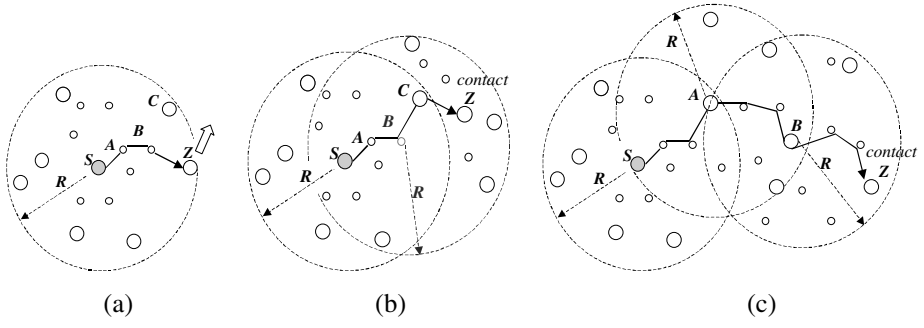

Fig. 4. The local route discovery protocol (a) Z is at the border of S's zone with a route 'S–A–B–Z'. Z is moving out-of-zone. (b) Z is no longer within S's zone, but it highly likely to exist in B's zone. The route 'S–A–B–C–Z' is identified and Z is selected as a contact. (c) Only loose routing may be kept by S without the need to update the exact route, so long as each node en-route has the next node en-route in its zone, then S can route to Z. This dampens route oscillations and reduces overhead.

between the contacts' zones, S chooses contacts to which it has disjoint routes.

Based on this scheme we use a simplified version of the border selection scheme that incurs very low overhead and has proven to perform very well. We use a promotion boundary and lower eviction boundary of R, and an upper eviction boundary (of at most R^2) depending on the connectivity between nodes en-route to the contact. In this version, a selector node keeps track of the border nodes; nodes that are exactly R hops away. This information is readily available through the intra-zone exchange. Every time a link state update changes route information about a border node, the new information is checked against the old information. If the update leads to a previous border (say node Z) disappearing from S's routing tables, then this indicates that Z has moved out-of-zone and it becomes a potential contact. The local route discovery protocol (described previously) is performed to establish a new route to the contact, Z. The selector node, S, sends a *self-monitoring* message to what used to be the (last hop) node leading to Z, say node B, if reachable. The self-monitoring message creates state in the nodes en-route to Z (e.g., nodes A and B in Fig. 4). It is highly likely that Z will be in B's zone, and the self-monitoring state will be established in about R nodes. As the nodes move, the path leading to Z may be kept up to R^2 hops, without the need to exchange any additional messages. If Z is not in B's zone then B sends a *remove monitor* message back to the selector. The self-monitoring state established in the nodes contains the next and previous hops on the path to Z, e.g., the self-monitoring state in B includes A as the previous hop and Z as the next hop. If the self-monitoring state exists in a node, it is checked with every route change. If the previous node on the path is not in-zone, then the node removes the self-monitoring state. Otherwise, if the next node on the path is out-of-zone, a remove monitor message is sent to the previous node. This message is propagated back to the selector node and eradicates the self-monitoring state in the previous nodes on the path. It causes the selector, S to evict the contact, Z. Also, if Z moves back into S's zone, it is evicted. Once the desired number of contacts has been selected the border check procedure need not be performed until and unless some of the existing contacts get evicted.

4. Contact-based query

In order to perform contact-based queries, selector nodes must be chosen; then they select and maintain their contacts (as described above); a policy is decided upon to perform the query (either level-by-level or single-shot); then the query is forwarded and processed. We detail these mechanisms in this section.

4.1. Choosing selectors

The simplest scheme for choosing selectors is that every node becomes a selector with probability 1. This means, however, that the selection and maintenance overhead is multiplied by the number of nodes in the network and the number of contacts per selector. Instead, we propose that only a small fraction (x%) of nodes in the network become selectors to decrease the selection overhead. Our study shows that after a certain point (about 10% of the nodes or 2% of the links), creation of new contacts in the network adds very little to improve performance (we shall discuss this further in the evaluation section). For robustness reasons we may also want multiple selectors (e.g., 3 on average) to be accessible to every node. The selection may also be a function of available energy, or capability (for example, in some sensor networks a few nodes may be designated to collect information), or other criterion. For simplicity, we do not assume prior knowledge of such heterogeneity and use a simple probabilistic method where every node independently chooses itself as a selector with probability (5–10%). A selector node sets a selector bit in the intra-zone information such that other nodes in the zone may use its contacts. Nodes that do not have selectors in their zones, promote themselves to be selectors after a random time, select their own contacts and set the selector bit in their intra-zone information. A node running out of energy (or overloaded) may stop advertising itself as a selector thus triggering other nodes to select contacts, and achieving load balancing.

4.2. Query policy

When a node issues a query, it first checks if the target resource exists in its zone (readily available through the intra-zone link state exchange). If not, then it issues a query message to a selector node (either itself is a selector or another selector in its zone) [5] that in turn forwards the query message to its contacts. The process by which the search sequence progresses depends on the query policy. We present two policies for query sequence: (i) the *level-by-level* contact query, and (ii) the single-shot contact query.

In the level-by-level contact query approach the querying node, Q, sends the query message to its contacts (or those of a selector in its zone), called the 1st level contacts. If the target is not found (i.e., no positive reply is returned to Q within a period t), then Q issues a new query that extends to the contacts of the previously visited contacts, called the 2nd level contacts. Unless the target is found, the process repeats up to the nth level contacts. (For our study we set $n = 5$. Our study shows that for $n > 5$ we get diminishing returns; i.e., query success rate almost saturates but overhead rises.) If the target is still not found, a fall-back mechanism is used, such as flooding or border-casting edge-flooding (i.e., ZRP-like [12]); both alternatives are investigated in the evaluation section. At any point in the search, if the target is found, a response is sent back to Q, and the search terminates at that level. (ii) In the single-shot contact query the same query is propagated from the 1st level contacts to the 2nd, and so on until the nth level contacts.

These two different policies have different merits. The level-by-level approach ends search when a target is found, and so may take advantage of object replication by reducing query overhead drastically (as we will show). The single-shot approach incurs less delay, and has very comparable per query overhead to the level-by-level approach with the proper setting, in case of no replication. We shall investigate both approaches.

[5] For this study we assume that if a node is not a selector and does not have other selectors in its zone then it resorts to flooding (or edge-flooding as described later). This represents an upper bound on our overhead. In enhanced versions of the protocol a node may select contacts on-the-fly using TTL of $2R$, but this may entail using contacts with unknown capabilities, energy, etc. This is part of on-going work.

4.3. Query forwarding and processing

The same rules of query processing are applied in both query policies. Each new query issued by Q is given a new sequence number, SN. As the query is passed on from contact to contact, nodes en-route (and their neighbors for single channel networks) process it hop-by-hop. Hop-by-hop processing includes a lookup in the zone table to check if the target is in-zone. We call this type of processing 'sweeping queries'. In addition, each node processing the query records its SN and Q. This record is needed for the time expected to complete the query, and so is timed out after a few seconds to be robust to SN wrap-arounds and failure of Q. If the target is found by the destined contact, a node en-route, or any of its neighbors, a response is returned to Q. Each contact, upon receiving a query message (for which it is the destination), checks its records for SN and Q; if found then the query message is simply dropped. We call this scheme loop prevention. The same cannot be done for en-route nodes, because the query message may be destined to far away contacts, the zones of which have not been covered before. For en-route nodes, upon reception of a query message, the records are checked for SN and Q, if found, then another check is performed on the destination of the query message. If the destination lies within less than 3 hops away, then the query message is dropped, otherwise it is propagated. This reduces search overhead leading to contacts, the zones of which heavily overlap with zones that have been investigated before. We call this scheme re-visit reduction. These schemes proved effective in reducing the query overhead while having very little effect on query success.

5. Evaluation and comparison

In this section we describe a set of simulation experiments used to evaluate our proposed

contact-based query schemes. We analyze the overhead and success rate of our architecture over various dimensions; mobility, network size, query rate, degree of replication and various numbers of contacts. In addition, we simulate several other query mechanisms and compare their performance to the contacts approach, including flooding, expanding ring search and several of its variants, and the border-casting approach (which we refer to as edge-flooding).

For the overhead analysis, we investigate the different overhead components of our schemes in detail. Particularly, we study packets transmitted for contact selection and maintenance, the zone establishment and maintenance and the query resolution. We also introduce new metrics, based on the call-to-mobility ratio (CMR), to capture the effects of the overall traffic as a function of query rate per mobility unit. This facilitates our comparisons.

5.1. Simulation setup

The networks simulated vary in size from 200 to 1000 nodes over a 1 km $\times$ 1 km area and varying radio range (55 m for the 1000 node topology, 70 m for 500 node, and 105 m for the 200 node topology) to keep the node density almost constant. By node density we mean the average node degree or number of nodes per zone. Unless stated otherwise, the nodes are randomly distributed over the area. Also, unless stated otherwise, for mobility we use the random way point model, in which a node chooses a random destination and picks a random velocity from $[0, V_{\max}]$. Once the destination is reached another destination and velocity are picked randomly, so on. We use a pause time of 0, i.e., continuous mobility. $V_{\max}$ was varied from 2 to 40 m/s. A single channel model was used in which a broadcast message may be heard by all neighbors within range. We do not simulate MAC layer protocols. [6] We use a discrete event simulator that borrows code from NS-2 [33] (mainly mobility and route computation) in addition to our own protocol modules for implementing the various query mechanisms. We present overhead in terms of packet transmission. [7] Most contact-based simulations were run using 5–10% selectors (i.e., 50 selectors for the 1000 node topology, 37 selectors for the 500 node topology and 20 selectors for the 200 node topology), usually using 4 maximum contacts each unless stated otherwise. Average values shown were averaged over 9 different runs of varying random seeds.

5.2. Contact selection and maintenance overhead

The contact selection and maintenance overhead reflects the packets sent during the contact selection, promotion and eviction procedures described above. This is expected to be a function of node mobility and the number of total contacts selected (i.e., number of selectors and maximum number of contacts per selector). Therefore, we use metrics that attempt to capture per contact, per node and per mobility measures. Also we investigate how the contact selection process is affected by mobility.

As shown in Fig. 5(a), the contact selection protocol becomes more efficient (meaning it selects more contacts) with the increase of velocity. In Fig. 5(b) the overhead of contact selection and maintenance is shown for two measures: (i) the overhead per contact second and (ii) the overhead per node per second. The former measure (i) indicates number of messages transmitted to keep a contact for 1 s; calculated as the ratio of the total packets transmitted to the sum of contacts through the simulation (counted every second). Both measures show linear relationship between overhead and velocity. The difference between the measures (almost a factor of 5) reflects the ratio of nodes to contacts. One additional overhead measure of interest is the packets per node per second per (m/s), where (m/s) is the average node velocity ($V_{\max}/2$ in

[6] Collisions at the MAC layer will be exacerbated by the excessive query traffic (especially broadcast traffic). Not modeling collisions actually works in favor of flooding-based protocols, that would have shown even worse performance had collisions been modeled.

[7] We have conducted studies that consider both transmission and reception and obtained similar trends and conclusions.

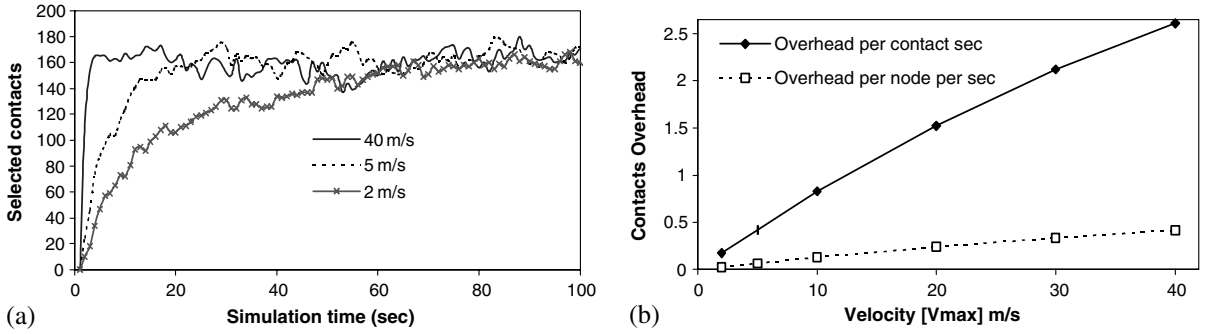

Fig. 5. Contact selection and maintenance analysis: (a) the number of selected contacts over time for different mobility degrees, the higher the mobility the more efficient the contact selection. (b) Overhead of selection and maintenance of contacts with various degrees of mobility, the overhead increases linearly with mobility. Solid line represents the overhead per contact second (the total packets transmitted during the simulation for contact selection and maintenance over the cumulative count of contacts through the simulation), dashed line represents the overhead (packets transmitted) per node per second.

our case). This measure captures overhead regardless of the mobility degree. We refer to such measure as the normalized overhead of selection and maintenance (or SM for short). Our simulation results consistently show that SM for our scheme is 0.02 packets/node/s/(m/s), for all velocities and over various topologies. This will be used later on in our comparisons. Simulation results are shown for 1000 node topology with 55 m range, 50 selector nodes and 4 contacts per selector over 100 s of simulation time. Similar SM overhead results were obtained for the other topologies.

5.2.1. *Effect of different mobility models*

Because MARQ utilizes mobility, we expect that it may exhibit different performance under various mobility models. Therefore, we introduce and study the effect of two deployment and mobility scenarios in addition to random way point. We refer to the two models as *expansion* and *contraction*. Expansion scenarios may be encountered when sensors are dropped over an area (e.g., using aircraft), which leads in general to an initial normal distribution of node locations. The sensors move uniformly randomly to cover the whole region of interest and then they stop. This scenario is shown in Fig. 6(I). Contraction scenarios may be encountered when sensors move towards a common area of interest after being initially randomly distributed. The sensors pick a destination location within the area of interest based on a normal distribution and stop when destination is reached. This scenario is shown in Fig. 6(II).

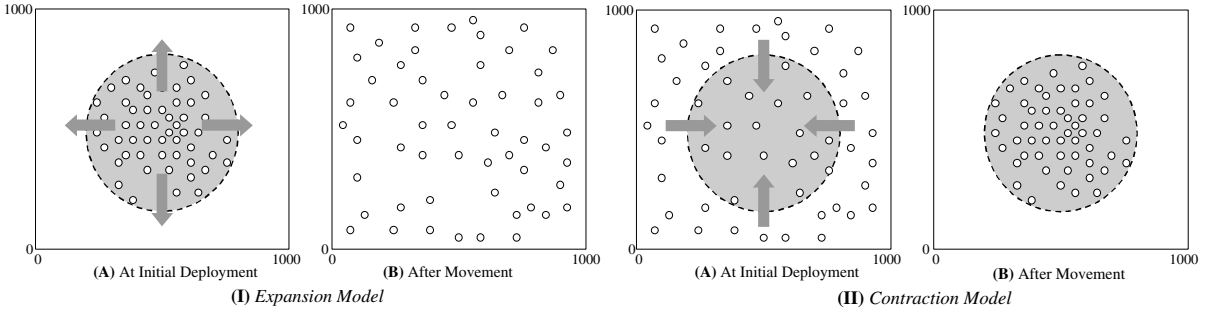

Fig. 6. The expansion and contraction deployment and mobility models. Arrows indicate direction of mobility. In (I)(A) and (II)(B) node locations have a normal distribution (avg. 500 and stdev. 250 for the x, y coordinates).

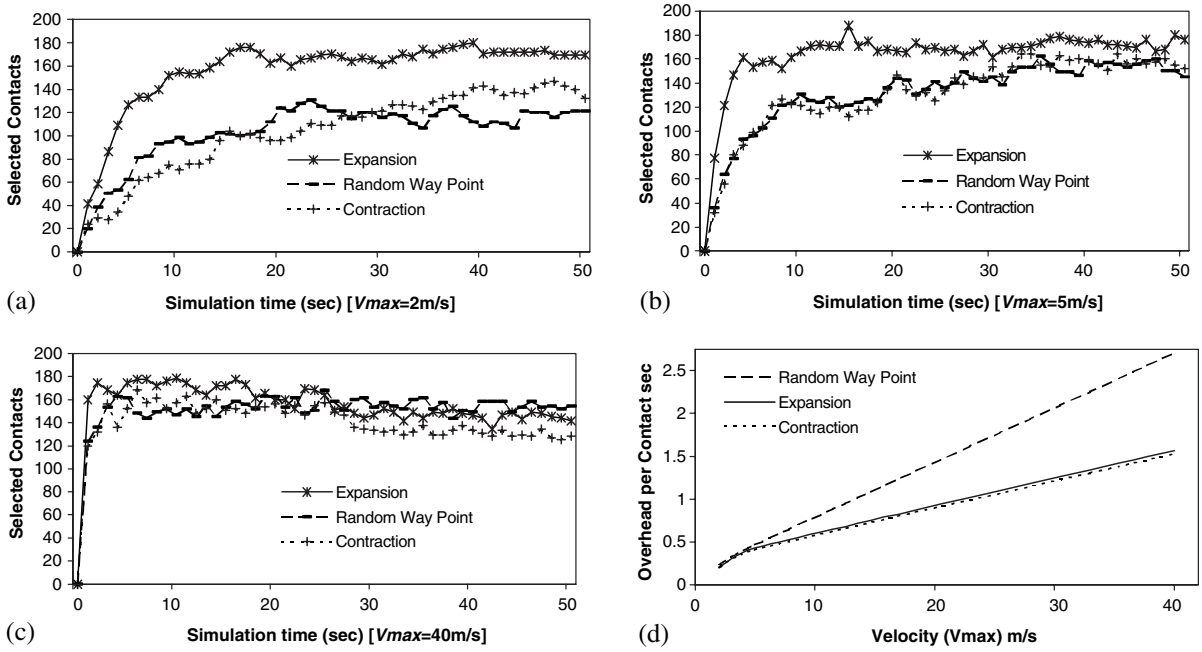

Fig. 7. Performance of the mobility-assisted contact selection (MACS) mechanism with the various mobility models; *expansion*, *contraction* and *random way point*. (a), (b) and (c) show the number of selected contacts with $V_{\max} = 2$, 5 and 40 m/s, respectively. (d) shows the contact selection overhead for the different mobility models in this study.

We evaluate the performance of MARQ for expansion and contraction models, in terms of number of selected contacts and contact selection overhead, and compare it to that under the random way point mobility model. The results are shown in Fig. 7. We see that the number of contacts selected is higher for the expansion model, due to the fact that many nodes are moving away from the initial deployment region, which creates greater opportunity for contact selection. For the contraction model the number of contacts is slightly worse than random way point, with all models achieving very good performance during higher mobility. In terms of contact selection overhead, all models experience very similar overhead during low speeds, with random way point exhibiting more overhead during higher speeds, due to its continuous movement. Hence in general we expect the performance of MARQ for expansion and contraction models to be close to (if not better than) random way point, with the general trends being quite similar. For the rest of this document, for brevity, we only show results using the random way point mobility model.

5.3. Zone overhead

The intra-zone information dissemination protocol uses link state to update routes to nodes within R hops away. The overhead incurred by this protocol is a (linear) function of mobility as shown in Fig. 8(a). Hence, we use a per mobility metric to capture overhead independent of mobility degree, the packets per node per second per (m/s). Fig. 8(b) shows this metric for different radii of link state exchange. [8] Let us call these values LS(h), where h indicates the hops extent of link state exchange. We are particularly interested in the case where $R = 3$. For our scheme LS(3) is needed to establish a zone of radius $R = 3$, and the overhead

[8] A similar study was conducted for ZRP [12] and similar results were obtained.

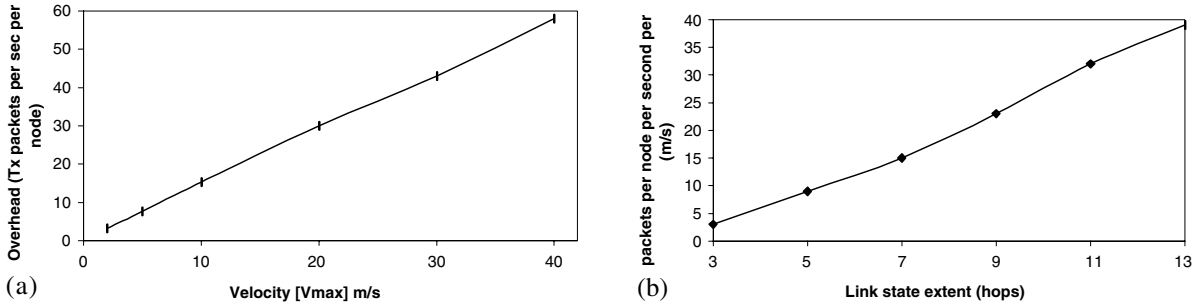

Fig. 8. Overhead of the zone establishment for link state: (a) exchange over $R = 3$. The overhead increases linearly with mobility. (b) Overhead per (m/s) of mobility for various settings of R.

incurred is ~3 packets/node/s/(m/s). For edge-flooding (Efld) or ZRP $R = 3$ translates into $2R - 1 = 5$ link state exchange to perform the early termination (ET) algorithm. Hence, for $R = 3$, Efld incurs $\mathrm{LS}(5) = 9$ packets/node/s/(m/s) overhead. More on this in the comparison section for total overhead.

5.4. Query overhead and success rate

The overhead is measured in transmitted packets per query. Success is the number of successful queries (or reachable nodes) in the network. Because the query success of contact-based query may be lower than the other schemes, we propose a fall-back (or penalty) scheme. For every unsuccessful contact-based query, we trigger a flooding or Efld message. This achieves the same reachability for all schemes. We refer to the contact-based scheme as 'c', and the fall-back schemes as 'c + flood' and 'c + Efld', respectively. We first show results for the level-by-level contact query, then discuss the single-shot contact query. In this subsection we study effects of mobility, number of contacts, number of nodes; i.e., network size (or topology), and effect of replication. For single-shot contact query we study the effect of varying the maximum contact level.

5.4.1. Performance with mobility

In this experiment the velocity ($V_{\max}$) was varied from 0 to 40 m/s (note that 0 velocity means that only nodes within a zone are reachable and out-of-zone nodes can only be reached through fall-back). [9] Results are shown in Fig. 9 for the 1000 node topology; with 25 s simulation, for the first 15 s no queries were triggered to stabilize the network, then queries were triggered with a rate of 0.15 query per node per second generating 1500 queries. It is very clear that the query overhead decreases drastically with *increase* of mobility. This is explained by investigating the contact-based reachability; i.e., success rate. With low mobility, less contacts are selected, and hence there is a higher percentage of failures, and hence a higher percentage of fall-backs, and vice versa for high mobility. Note that the reachability curve shown is for the contact-based scheme only. After using fall-back the success rate is same as the other protocols. The overhead per query shown is for both contact-based + fall-back mechanism (i.e., that achieving maximum success rate). Fig. 9(c) shows the overhead ratio for contact-based + fall-back over the fall-back; mainly illustrating the advantages in query overhead obtained by using contact-based queries (70% improvement over flooding and 50% improvement over Efld, in high mobility).

[9] Further enhancement of the contact-based query for low (or no) mobility may be achieved by integrating a static-based contact selection as in CARD [27,28], where the list of zone borders is used to direct the contact selection message out-of-zone and achieve reduced zone overlap with the contacts. This, however, incurs additional backtracking overhead. Integration of CARD and MARQ is part of our future work.

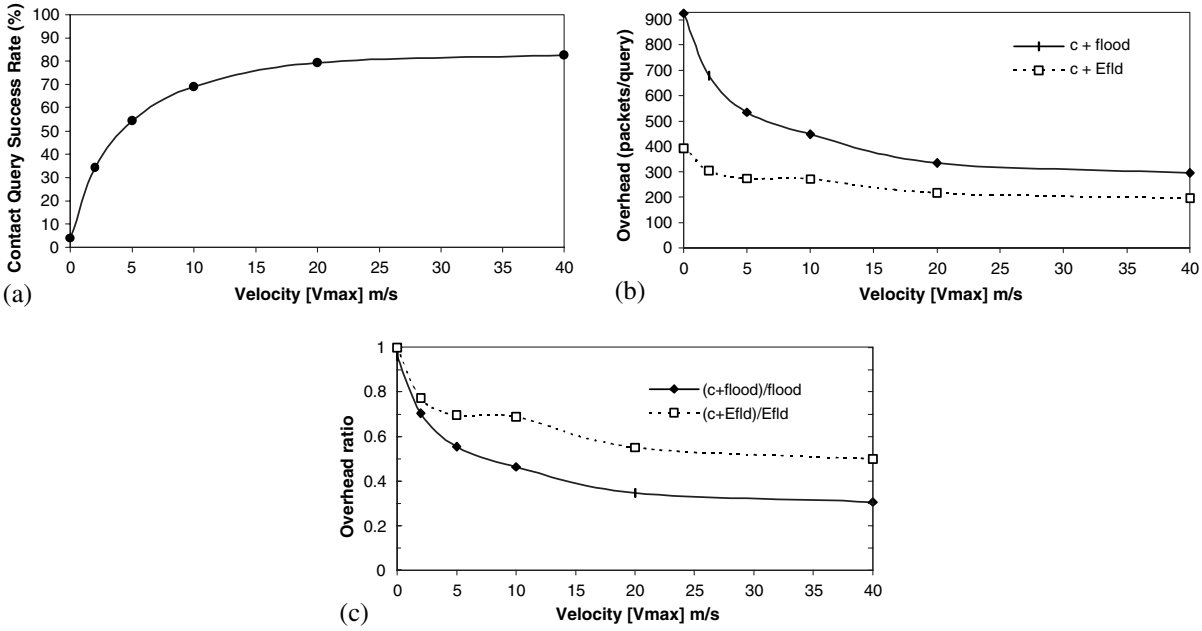

Fig. 9. Effect of mobility on query overhead for the contact-based query with fall-back, $c + \text{flood}$ uses flooding as a fall-back mechanism while $c + \text{Efld}$ uses the edge-of-zone flooding as fall-back: (a) reachability; i.e., query success of the contact-based scheme (without fall-back) increases, (b) overall query overhead decreases, (c) reduction in query overhead obtained by using contact-based queries. The best ratio achieved at high speeds (70% over flooding and 50% over Efld). Ratios shown for contact-based queries with fall-back (i.e., achieving the same success rate as the other schemes).

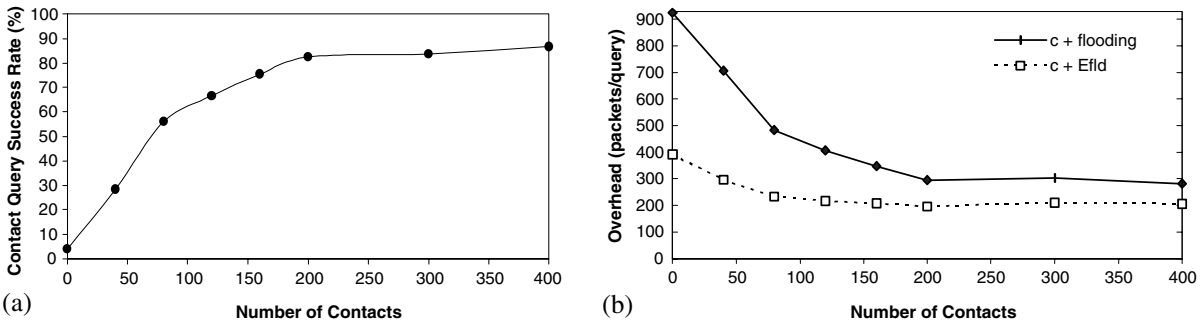

Fig. 10. Effect of increasing the number of contacts in the network. After a certain value (around 200 contacts) there is saturation in the success rate and the per query overhead. (Simulation results shown for 1000 node topology, at 40 m/s.) (a) Success rate for the contact-based queries (without fall-back) and (b) overhead per query for contacts with fall-back.

5.4.2. Varying number of contacts

In order to understand the performance improvement due to addition of contacts we conduct experiments with varying the number of overall contacts in the network. As shown in Fig. 10, the success rate increases (and subsequently the query overhead decreases) with the increase in number of contacts up to a certain point (around 200) then it saturates. Increase in number of contacts beyond this point does not lead to an increase in perfor-

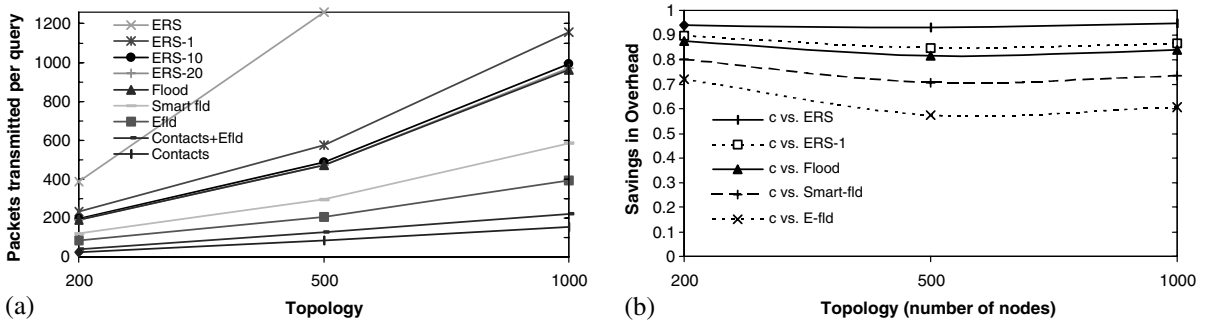

Fig. 11. Comparing overhead per query for the various query schemes: (a) the query overhead of various query techniques, (b) improvement margin of using contacts over other approaches. '*c* vs. ERS' means (1-overhead of '*c*'/overhead of 'ERS'), similarly for the others.

mance, but leads to an increase in contact selection and maintenance (SM) overhead (discussed earlier).

5.4.3. Varying topologies

We have conducted extensive simulations to compare our contact-based approach to other approaches for query such as flooding, expanding ring search and their variants, and Efld (a variant of ZRP). The comparison is done for various network topologies. This first set of comparison results is shown in Fig. 11. Every point on the graph represents an average of 9 runs of 500 random queries each. Results are shown for three of the topologies used with 200 (with 105 m range), 500 (with 70 m range) and 1000 nodes (with 55 m range), respectively. This kept the average node degree almost constant. The random way point mobility model was used with a maximum speed ($V_{\max}$) of 40 m/s. For the contact-based mechanism 5–10% of the nodes were randomly chosen to select four contacts each. For our mechanism (called simply contacts) the zone radius, R, the LEB and PB were set to 3 and the UEB to 9. All the flooding techniques used implemented loop prevention by multi-transmission suppression (each node upon receiving a query, records its sequence number and does not forward another query with the same sequence number). For smart flooding (*smart fld*) we used probabilistic flooding [30]. 'Smart fld' results are shown for settings that achieved success rate of ~85%. This was found to be equivalent to probabilistic flooding with $p = 0.65$. Efld incorporated loop prevention, and two levels of query detection and early termination. [10] The same zone radius (3) is also used for Efld. Expanding ring search techniques varied the step by which the TTL is chosen; for ERS it is simply incremented by 1 for every re-try. For ERS-1 is it incremented exponentially starting with 1 (i.e., TTL increases with every try as 1, 2, 4, 8, etc.), for ERS-10 the TTL is also incremented exponentially but starting from 10, and similarly for ERS-20, starting from 20. The results show a very clear advantage of using the contacts mechanisms over any other approach we have studied. Contacts results in query overhead savings ranging from 80–90% over flooding, 70–80% over smart flooding and 60–70% over Efld, as shown in Fig. 11(b). The savings are even greater for all other expanding ring search techniques studied. Expanding ring search (ERS) performs particularly poorly due to the

[10] In [12] the querier, Q, sends the request to its borders, and the borders send it to their borders, so on. Query control mechanisms are used to reduce redundant querying. Query detection mechanisms QD-1 (and QD-2) indicate that intermediate nodes along the forwarding path (and their neighbors) record the request information. Upon receiving a request sent to a border that has been previously visited, the intermediate node terminates such request. The intermediate node has knowledge of the previously visited borders in its zone by maintaining intra-zone information of up to $2R-1$ hops (instead of the basic zone of R hops). Hence, the redundant request can be terminated early. This scheme is called early termination (ET). We further reduce ZRPs overhead by having a query forwarded by a border node, suppress queries from its neighboring border nodes.

large-network diameter (around 35). Note however, that the average query success rate for contacts (82.6%) was lower than most other schemes (96%) (unreachability was due to the highly dynamic mobile network). Although the parameters of contacts (e.g., number of contacts or selectors) may be tuned to improve reachability, it is hard to obtain an optimal setting in a highly dynamic environment. Instead we propose to use Efld as a fall-back mechanism when a query fails using contacts. We call such an extended mechanism contacts + Efld. This extended mechanism has savings over Efld ranging from 38–54% in query overhead.

5.4.4. Effect of replication

In many cases, object replication may be performed in a sensor network, and hence reachability; i.e., success rate, of the contacts scheme alone may be sufficient for successful queries without having to fall-back. We have conducted several replication experiments to estimate the per query overhead incurred by level-by-level contact-based query and its success rate as a function of the replication ratio. Replication ratio represents the number of times the object is inserted into the network; ratio of 1 means only 1 copy (no replication), which was the case in our discussions above, ratio of 2 means the object is inserted in 2 randomly selected nodes in the network, so on. Results in Fig. 12 are shown for 1000 node topology, at 40 m/s, with 50 selectors and 4 contacts per selector. We notice a drastic improvement in performance. Adding only 1 copy of the objects results in over 60% reduction in query overhead, and increase of success rate from 81.5% to 92.2%. Note that except for ERS and its variants, overhead of the other schemes (flooding and Efld) per query does not change drastically with replication. For ERS and its variants the overhead is still very high (even with replication ratio of 4 the best ERS variant, ERS-1, is still well above 500 packets per query). For replication ratio of 4, the contact-based query incurs only 2.2% per query overhead as does flooding, and 5.5% as does Efld, for about 3.5% less success rate (92.2% vs. 96%).

5.4.5. Effect of contact level

For the above experiments we set the maximum level (nth) of contacts to 5. We did not attempt to optimize this value thus far because it had little effect on level-by-level contact query because the search terminates when the object is found. For single-shot contact query, however, this value may be important because the search does not terminate when an object is found, but it continues until the nth level (or before if looping is detected on all search branches). Hence, we conducted a set of experiments to investigate the effect of varying the maximum contact level on query overhead and reachability. We also investigated the change in number of contacts per selector. Simulation results are shown in Fig. 13 for 1000 node topology, with 50 selectors at 40 m/s. Fig. 13(a) shows the results for single-shot contact ($c1$ for short) with flooding as fall-back, while (b) shows results for $c1$ with Efld as fall-back (notice the difference in y-axis

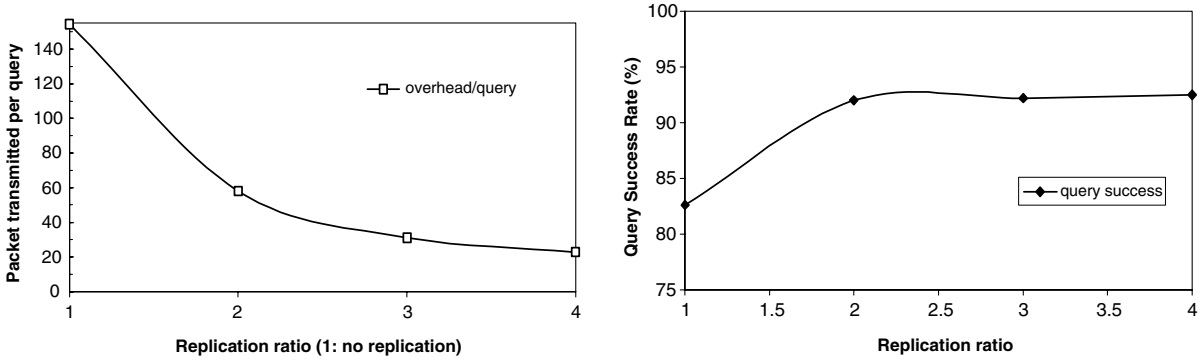

Fig. 12. Effect of replication on the query overhead and the success rate of level-by-level contact-based query.

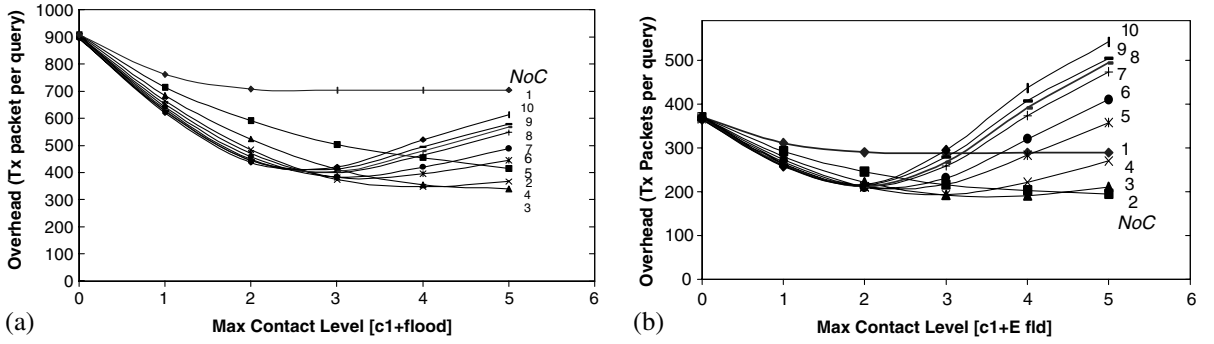

Fig. 13. Effect of contact level and number of contacts per selector: (a) performance of single-shot contacts $(c1)$ + flooding, (b) performance of single-shot conatcts $(c1)$ + Efld.

scale). It is quite clear that there are optimum operating points for combinations of number of contacts (NoC for short) and contact level. This is equivalent to solving the following optimization problem: minimize $c1 + (1 - \text{success}) \cdot \text{fall-back}$, where $c1$ is the average overhead for the single-shot contact query, *success* is the rate of success for the single-shot contact query, and *fall-back* is the average overhead for the fall-back mechanism. It is surprising that the results for optimum single-shot contact are quite close to those for level-by-level contact, with the caveat that for the single-shot scheme the parameters (contact level and NoC) need to be optimized. If we over-estimate the parameters this may cause the single-shot scheme to perform poorly. It seems, however, that for 3 or 4 contacts, the performance is relatively insensitive to contact levels above 3. Similar curves (i.e., against contact level and NoC) for level-by-level contacts show that performance does not degrade if we over-estimate, but it almost saturates due to termination of search for found objects.

5.5. Total overhead

The total overhead of the contacts protocol includes components other than query overhead, such as (i) zone maintenance, and (ii) contact selection and maintenance. The link state overhead repairs link failures due to mobility and hence is a function of mobility (as we have shown). A general way to represent such overhead is per node per m/s of mobility. For link state exchange over $R = 3$ (what we called LS(3)) hops we get 3 packets per second per node per m/s of mobility. The contact selection and maintenance is also a function of mobility. Hence, the effectiveness of our scheme depends on the call-to-mobility ratio (CMR), or the query rate per m/s (called q). The cost to select and maintain contacts and zones must be amortized by the query rate or CMR. For very small CMR the benefits of the contacts architecture may not show. We plot the relationship between the CMR query rate q, and the overhead normalized by the overhead of flooding. q represents the average per node query per km; q is small for very high mobility.

The total overhead (in packets/s/node/(m/s)) is function of q and is obtained as follows. Let T_{Efld}, T_{Efld}, T_{c} and $T_{\text{c+Efld}}$ be the total overhead for flooding, Efld, contacts, and contacts with fall-back to Efld, respectively. Also, let fld, Efld, c and ce be the overhead per query (in packets) for those protocols, in order. Note that with fall-back $ce = c + (1 - \text{success})\text{Efld}'$, where Efld$'$ is the Efld overhead with $R = 3$ link state exchange. Let SM be the contact selection and maintenance overhead (as explained earlier), and LS be the zone link state overhead. We get:

$$T_{\text{fld}} = q \cdot \text{fld},$$

$$T_{\text{Efld}} = \text{LS}(2R - 1) + q \cdot \text{Efld},$$

$$T_{\text{c}} = \text{LS}(R) + q \cdot c + \text{SM},$$

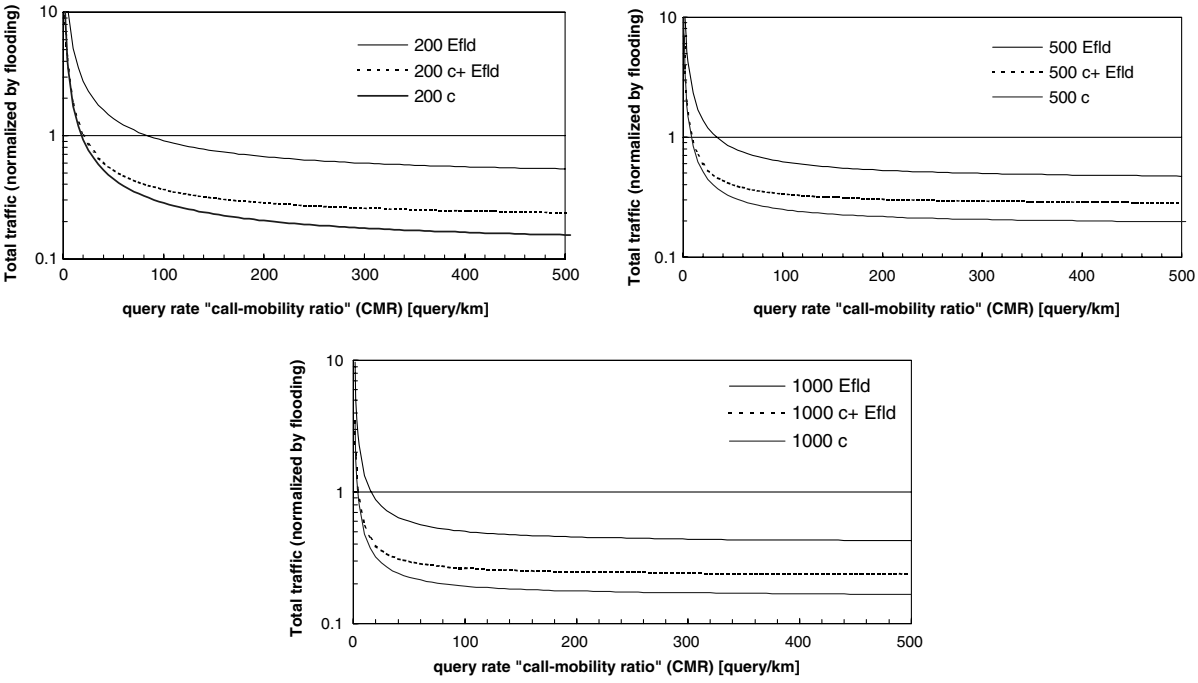

Fig. 14. Benefits of contacts compared to flooding and Efld as function of the call-mobility-ratio q for different topologies [200, 500 and 1000 nodes] (c: contacts, c + Efld: is the extended contacts).

$$T_{\text{c+Efld}} = \text{LS}(R) + q \cdot ce + \text{SM}$$
$$= \text{LS}(R) + q \cdot (c + (1 - success) \cdot \text{Efld}') + \text{SM}.$$

For $R = 3$, we get $\text{LS}(3) = 3$ and $\text{LS}(5) = 9$, and for the settings on our study for the contact-based scheme, we have $\text{SM} = 0.02$. fld, Efld and c (or ce) are obtained in our simulations for the different topologies. We plot the ratios

$$\left(\frac{T_{\text{c}}}{T_{\text{fld}}}\right), \quad \left(\frac{T_{\text{c+Efld}}}{T_{\text{fld}}}\right) \quad \text{and} \quad \left(\frac{T_{\text{Efld}}}{T_{\text{fld}}}\right)$$

in Fig. 14.

As shown in Fig. 14 benefits of contacts over flooding start to show at q between 5 and 10 query/km. For higher values of q (around 400 queries/km) the ratio of the total contacts overhead to the flooding overhead approaches 14–20% for the contacts approach and 22–28% for the extended contacts + Efld approach.

6. Related work

We address the problem of query resolution and resource discovery in *infrastructure-less* wireless mobile sensor networks. Hence, architectures that require infrastructure (e.g., DNS) or that assume existence of underlying routing are not suitable for our problem. Centralized approaches are neither robust nor scalable.

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 or efficient flooding has been proposed in [24,29–31]. 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 MARQ is that smart flooding reduces the redundant messages in querying every node in the network, whereas MARQ 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. In our simulations we have compared MARQ to probabilistic flooding.

Other related work lies in the areas of sensor networks and ad hoc wireless networks.

Approaches in ad hoc networks that address scalability employ hierarchical schemes based on clusters or landmarks (e.g., LANMAR [5,6]). 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. Hence, in general we avoid the use of complex coordination schemes for hierarchy formation, and we avoid using cluster-heads.

In GLS [7] an architecture is presented that is based on a *grid* map of the network (that all nodes know of). Nodes recruit location servers to maintain their location. Nodes update their location using an ID-based algorithm. Nodes looking for location of a specific ID use the same algorithm to reach a location server with updated information. This is a useful architecture when a node knows the network grid map, knows its own location (through GPS or other means), and knows the ID of the node it wishes to contact. These assumptions may not hold in our case, especially that a source node has to know the specific ID of the target node and uses that ID to locate the traget's location servers. 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.

The algorithm proposed in [4,8] uses global information about node locations to establish short cuts or friends, and uses geographic routing to reach the destination. It is unclear how such architecture is feasible with mobility. Also, such work does not specify the number of short cuts to create. In addition, the destination ID (and location) must be known in advance, which may not be the case in resource discovery.

In ZRP [9–12] the concept of hybrid routing is used, where link state is used intra-zone and on-demand routing is used inter-zone. Border-casting (flooding between borders) is used to discover inter-zone routes. A good feature in ZRP is that a zone is node-specific. Hence, there is no complex coordination susceptible to mobility as in cluster-head approaches. 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 mobility-assisted contact selection) are fundamentally different than ZRPs border-casting. Also, as a routing protocol, ZRP attempts to search for shortest paths and minimize the response time for route discovery. In our design we make a clear trade-off between route optimality (that is *not* the primary goal in query resolution) and communication overhead. We have compared the performance of ZRP and the contact-based approach through simulations. The contacts based approach incurs significantly lower overhead and has much more desirable scalability characteristics than ZRP for query resolution. Detailed comparisons against ZRP-like approach (that we refer to in this paper as edge-of-zone flooding, or Efld) were presented in Section 5.

For object tracking, in SCOUT [14] an architecture was presented that is based on hierarchy formation. Using concepts borrowed from landmark hierarchy [13], 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 mobility. This is mentioned clearly in their work. The root nodes of the hierarchy use global flooding to send advertisements. These advertisements are sent periodically. If the root nodes 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. The work presents two schemes that may be supported by the hierarchy, called SCOUT-AGG and SCOUT-MAP. In SCOUT-AGG object information is aggregated up the hierarchy. Queries travel up the hierarchy tree until the object is found. This query scheme may degenerate to flooding if the object summaries in many devices in the hierarchy indicate that they may have the object. SCOUT-MAP uses indirection through a device locator. A hashing scheme is used to route to the device locator (which has information about the tracking device). The hash depends on the number and identity of children of each device in the hierarchy that will be involved in this routing. Hence, a change in the number or identity of children for any of the en-route devices will cause re-hashing. Children for any devices at any level often change with mobility. A re-hash of the objects and their device locators is needed with mobility, in addition to re-configuration of the hierarchy. The performance of the SCOUT architecture degrades drastically with node mobility and network dynamics.

Directed diffusion [17,18] provides a data dissemination paradigm for sensor networks. This scheme targets continuous queries in sensor networks. Without availability of geographic 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 may be amortized over the amount of information exchanged over possibly extended periods of time. For high mobility or for short-lived/one-shot queries, however, directed diffusion may incur excessive overhead especially in large-scale networks, where flooding is quite costly. We believe that in such situations, our contact-based architecture can be integrated with directed diffusion to discover resources in a scalable manner instead of using flooding mechanisms.

In [23] 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 the node nearest to that location. Geographic routing is used to route the data to that location. Nodes interested in the data use consistent hashing on that object's identifier and get the location at which the data is stored. 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 and routing are 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 situations where geographic information is not available. These are situations we address in this paper.

In [20–22] 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 MARQ 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 [32] 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. That work does not use mobility-assisted techniques nor does it attempt to reduce zone overlaps. 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 MARQ in future work.

Rumor routing is proposed in [26] as an alternative to reduce flooding overhead for interests in directed diffusion. It was designed for continuous (long-term) queries.

In [19] 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 [24], Gossiping, and LEACH [25]. These protocols are designed for data dissemination (not query resolution for potentially replicated data that we address in our scheme).

Also, these protocols do not utilize mobility to increase performance. This seems to be a unique feature of our MARQ scheme that has proven to be quite valuable in high mobility scenarios.

7. Conclusions and future work

In this paper we have presented a novel architecture for efficient query resolution in sensor networks, for one-shot, simple queries of potentially replicated data. Our architecture, MARQ, uses the concept of zones, formed using link state exchange up to R hops, and introduces the concept of contacts that extend beyond zones by utilizing mobility. Contacts represent short cuts in the wireless network that attempt to reduce the degrees of separation between the queriers and the targets. Contacts are initially selected from within the zone, and become more valuable with mobility as they move out of zone. This allows the selecting node a chance to intelligently choose its contacts from nodes it already knows.

Mechanisms for efficient, mobility-assisted, contact selection and maintenance have been presented and evaluated. Our evaluation shows that performance of these mechanisms *improves* with mobility; a feature that is unique to our MARQ architecture.

Detailed evaluation of the performance (in terms of overhead and success rate) was carried out, for various mobility degrees, network sizes, and degrees of replication. The results were compared to several other well-known query resolution schemes (including flooding, expanding ring search and its variants, and edge-of-zone flooding). For non-replicated objects, results indicate improvement of 70–90% over flooding, 60–70% over edge-of-zone flooding, and greater for expanding ring search mechanisms. For replicated objects, the savings are even more drastic with our level-by-level contact-based query mechanism; for 3 randomly placed object replicas savings of over 90% are obtained.

Our mechanisms do not assume availability of geographic information and are totally distributed and self-organizing. Robustness to failures is obtained by using a loosely coupled simple hierarchy scheme based on independent node-specific zones.

Plans of our future work include further improvement of our mechanisms by introducing heuristics for contact selection for static networks. One possible direction that is currently being investigated is to actively send selection messages out-of-zone (e.g., up to $2R$ hops) using borders to which the selecting node has disjoint routes.

Other related research issues to investigate include mobility-assisted object replication to improve per query overhead. The main idea is to replicate the data by sending it to nearby nodes that are moving away. This idea also utilizes mobility and reduces the overhead of replication in the sensor network.

Acknowledgements

The author thanks the anonymous reviewers for their insightful comments on mobility models and smart flooding. The author also acknowledges the very useful feedback provided during discussions with the wireless networking laboratory research group at USC, specifically Karim Seada on contact selection and Shao-Cheng Wang on overhead analysis. Ahmed Helmy was supported by the NSF CAREER Award 0134650, and a research grant from Intel.

References

[1] S. Milgram, The small world problem, Psychology Today 1 (1967) 61.
[2] D.J. Watts, Small Worlds the Dynamics of Networks Between Order and Randomness, Princeton University Press, Princeton, NJ, 1999.
[3] D. Watts, S. Strogatz, Collective dynamics of small-world networks, Nature 393 (1998) 440.
[4] J. Kleinberg, Navigating in a small world, Nature 406 (August 2000).
[5] B. Das, V. Bharaghavan, Routing in ad-hoc networks using minimum connected dominating sets, in: Proceedings of IEEE ICC'97.
[6] P. Guangyu, M. Gerla, X. Hong, LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility, MobiHOC'00, 2000, pp. 11–18.
[7] J. Li, J. Jannotti, D. Couto, D. Karger, R. Morris, A scalable location service for geographic ad hoc routing, ACM Mobicom, 2000.

[8] L. Blazevic, S. Giordano, J.-Y. Le Boudec, Anchored path discovery in terminode routing, in: Proceedings of the Second IFIP-TC6 Networking Conference (Networking 2002), Pisa, May 2002.
[9] M. Pearlman, Z. Haas, Determining the optimal configuration for the zone routing protocol, IEEE Journal on Selected Areas in Communications 17 (8) (1999) 1395–1414.
[10] Z. Haas, M. Pearlman, The zone routing protocol (ZRP) for ad hoc networks, IETF Internet Draft for the Manet Group, June'99.
[11] Z. Haas, M. Pearlman, The performance of query control schemes for the zone routing protocol, ACM SIGCOMM'98.
[12] Z. Haas, M. Pearlman, ZRP: a hybrid framework for routing in ad hoc networks, in: C. Perkins (Ed.), Ad Hoc Networks, Addison-Wesley, Reading, MA, 2001, pp. 221–254.
[13] P.F. Tsuchiya, The landmark hierarchy: a new hierarchy for routing in very large networks, Computer Communications Review 18 (4) (1988) 35–42.
[14] S. Kumar, C. Alaettinoglu, D. Estrin, SCOUT: scalable object tracking through unattended techniques, ICNP, 2000.
[15] J.J. Aceves, M. Spohn, Bandwidth-efficient link-state routing in wireless networks, in: C. Perkins (Ed.), Ad Hoc Networks, Addison-Wesley, Reading, MA, 2001, pp. 323–350.
[16] A. Helmy, Small large-scale wireless networks: mobility-assisted resource discovery, Technology Research News (TRN) (August 2002).
[17] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom 2000), Boston, Massachusetts, August 2000.
[18] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, Impact of network density on data aggregation in wireless sensor networks, in: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, July 2002.
[19] M. Chu, H. Haussecker, F. Zhao, Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks, International Journal of High Performance Computing Applications 16 (3) (2002) 90–110.
[20] 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, University of Southern California, September 2002.
[21] P. Bonnet, J.E. Gehrke, P. Seshadri, Querying the physical world, IEEE Personal Communications 7 (5) (2000) 10–15.
[22] P. Bonnet, J. Gehrke, P. Seshadri, Towards sensor database systems, in: Mobile Data Management'2001, Hong Kong, 2001.
[23] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker, GHT—a geo-graphic hash-table for data-centric storage, in: First ACM International Workshop on Wireless Sensor Networks and their Applications, 2002.
[24] W.R. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, in: Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, WA, August 1999, pp. 174–185.
[25] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: 33rd International Conference on System Sciences (HICSS), January 2000.
[26] D. Braginsky, D. Estrin, Rumor routing algorithm for sensor networks, in: First Workshop on Sensor Networks and Applications (WSNA), September 2002.
[27] N. Nahata, P. Pamu, S. Garg, A. Helmy, Efficient resource discovery for large scale ad hoc networks using contacts, ACM SIGCOMM Conference, Pittsburgh, August 2002 (poster), Extended Abstract in ACM SIGCOMM Computer Communications Review (CCR), July 2002.
[28] A. Helmy, S. Garg, P. Pamu, N. Nahata, Contact based architecture for resource discovery (CARD) in large scale MANets, in: Third IEEE/ACM International Workshop on Wireless, Mobile and Ad Hoc Networks (WMAN), IEEE/ACM IPDPS, April 2003.
[29] W. Lou, J. Wu, On reducing broadcast redundancy in ad hoc wireless networks, IEEE Transactions on Mobile Computing 1 (2) (2002) 111–122.
[30] S. Ni, Y. Tseng, Y. Chen, J. Sheu, The broadcast storm problem in a mobile ad hoc network, in: Proceedings of ACM MOBICOM, August 1999, pp. 151–162.
[31] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, L. Viennot, Optimized link state routing protocol, Proceedings of IEEE INMIC'01.
[32] N. Sadagopan, B. Krishnamachari, A. Helmy, The ACQUIRE mechanism for efficient querying in sensor networks, in: First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA), in Conjunction with IEEE ICC, May 2003.
[33] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu, Advances in network simulation, IEEE Computer 33 (5) (2000) 59–67.

Ahmed Helmy received his Ph.D. in Computer Science ('99), M.S. in Electrical Engineering ('95) from the University of Southern California, M.S. in Engineering Math ('94) and B.S. in Electronics and Communications Engineering ('92) from Cairo University, Egypt. He is currently an Assistant Professor of Electrical Engineering at the University of Southern California. In 2002, He received the National Science Foundation (NSF) CAREER Award, and in 2000 he received the USC Zumberge Award. In 2000, he founded—and is currently directing—the wireless networking laboratory at USC. His current research interests lie in the areas of protocol design and analysis for mobile ad hoc and sensor networks, mobility modeling, design and testing of multicast protocols, IP micro-mobility, and network simulation. His email address is: helmy@usc.edu, and his webpage is at http://ceng.usc.edu/~helmy.