PARSe: Power-efficient Architecture for Resource-discovery and Small-transfers in Large-Scale Ad Hoc Networks
Ahmed Helmy
Department of Electrical Engineering – Systems
University of Southern California
helmy@usc.edu

Abstract-
Resource discovery requests and small transfers are likely to constitute a significant portion of the flows in emerging classes of ad hoc networks. In these applications the query or transferred object is carried within the request message itself. Route discovery for such requests incurs much more communication overhead (and hence power) than the actual data transfer. Especially for large-scale networks, it is quite costly to establish shortest path routes for such types of requests. Existing approaches for routing in ad hoc networks attempt to find and maintain high quality routes. Such approaches may be suitable for prolonged transfers, but not for small ones. In this paper, we present a new architecture that is geared towards resource discovery and small transfers in large-scale ad hoc networks. In our approach we aim at reducing the total energy consumption of successful delivery as opposed to finding high quality routes. We do not assume knowledge of node locations.

In our architecture, a mobile node knows information about nodes in its zone, up to R hops away. We introduce the notion of contacts that extend beyond the zone and act as short cuts to reduce the degrees of separation between the request source and the target. We introduce several protocols to implement different policies for the search. We also propose a simple mobility-based scheme for object replication. We evaluate the performance of our protocols, mainly in terms of energy consumption and request success rate. We carry out extensive simulations to investigate the design dimensions and to compare our architecture to other related protocols, including flooding and border-casting. Our results show that significant savings may be achieved using our contact-based PARSe technique. For large networks and high request rates, PARSe consumes as little as 5.5% of flooding energy and 13.5% of border-cast energy. The study also shows reasonable settings of parameters that work well for a wide range of network sizes, up to 32,000 nodes. Our protocols may be easily implemented using simple extensions to zone-routing protocols.

1. Introduction
Classes of emerging ad hoc wireless networks are attracting lots of research attention. These networks are expected to have a significant impact and have the potential for many applications. Research challenges in such networks are manifold. First, the scale of such networks is expected to grow into the tens of thousands of nodes (or more). Second, unlike wired networks, nodes in ad hoc networks, in many cases, are expected to be power constrained, such as small battery-powered devices (e.g., PDAs), adding a major constraint on protocol design. Third, node mobility creates a highly dynamic environment to which protocols must adapt gracefully. Fourth, the infrastructure-less nature of ad hoc networks requires self-configuring protocols, and renders resource discovery one of the essential elements for proper operation. Hence, for the scalability of ad hoc networks it is imperative to design scalable, power-efficient, mobility-adaptive, self-configuring resource discovery protocols. Potential examples of resource discovery in future ad hoc networks include, but are not limited to, DNS-like queries, discovering servers (web, multicast, files or other), and object discovery in distributed databases or distributed computing. In addition to resource discovery requests, many applications may use small transfers and very short flows, such as short transactions and messaging, object access, paging, to name only a few.

Several ad hoc routing protocols have been proposed recently (e.g., DSR [5], AODV [4], ZRP [13][14]). Usually, one of the main design goals for ad hoc routing is to discover and maintain routes of high quality, to achieve efficient prolonged data transfers over those routes. However, in cases of resource discovery and small transfers, the factor dominating network overhead is not the data transfer, rather the discovery overhead. Such transfers are very short and do not extend beyond the discovery phase. Hence, the main design goal for such transfers is not to establish high quality routes, but to achieve successful delivery of requests (or resolution of queries) with very low consumption of network resources, in terms of energy and bandwidth. Communication is a major consumer of energy (a scarce resource). Thus, it becomes crucial for power-efficient protocols to reduce communication overhead, both in terms of transmission power and reception power. Unfortunately, little work has been done on power-efficient resource discovery in large-scale ad hoc networks.

One simple technique to use for resource discovery is flooding. For frequent requests in large-scale networks, however, flooding may incur significant communication overhead. Expanding ring search (ERS) techniques are also commonly used for discovery. For large-scale wireless networks, however, the network diameter tends to be quite high (due to clustering of nodes), and ERS may be quite inefficient in such networks. For scalability, several hierarchical approaches have been proposed [21][22]. Many such architectures are cluster-based, in which each cluster of nodes elects a cluster-head to relay traffic to other clusters. A cluster-head may become a single point of failure or a point of traffic concentration. The landmark approach [6][15][16] 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, for election, promotion and demotion. Hence, in highly dynamic environments, such as ad hoc networks, complex hierarchical approaches are susceptible to major re-configuration overhead with node mobility and failure. Other architectures use location information for routing. In our study we assume that location information is not available, and that the target node (that holds the resource being discovered) may not be known a priori. Also we do not assume that the boundary or extent of the network is known.

In this paper we introduce a new architecture for power-efficient resource discovery and small-transfers in large-scale ad hoc networks, called PARSe. 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. In our architecture, every node independently collects information from neighboring nodes up to \( R \) hops away. This is called a node’s zone. A similar concept was introduced for zone routing protocol (ZRP) [12][13][14]. In ZRP, border-casting is used to discover routes, in which borders of a zone relay the request to their borders, and their border, so on. In our scheme, however, we use fundamentally different concepts of small worlds [1][18] and contacts. Use of contacts is key for efficient discovery in our scheme. Wireless networks are spatial graphs (in which links are limited by distance) and tend to be highly clustered, leading to very high path length.

For a node, contacts are a few nodes outside of the zone 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 zone-overlap and to elect contacts that increase the coverage of the search. The search proceeds according to several possible policies; single-shot, level-by-level or step-search. The contact distance, depth of search, extent of zone, and number of contacts are design parameters investigated in detail in this study. Also, we analyze effects of object replication on the performance of our protocols and suggest a simple scheme for using mobility to distribute replicas in a communication-efficient manner.

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 network sizes, without the need for parameter optimization for each network size. 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 average delay (or number of attempts). The step search policy exhibits the least energy consumption, while the single-shot exhibits the least delay (with only one attempt), both with near perfect success rates for large-scale networks. We compare our protocols to flooding and border-casting over networks ranging in size from 200 to 32000 nodes. Our results show that significant overall energy savings may be achieved using our contact-based PARSe technique. For large networks and high request rates, PARSe consumes as little as 5.5% of flooding energy and 13.5% of border-cast energy. Our protocols may be easily implemented using simple extensions to zone-routing protocols.

The rest of the paper is outlined as follows. Section 2 introduces the architectural overview for PARSe. Section 3 presents the contact-selection protocol and search policies. Section 4 provides request processing and forwarding rules and presents mechanisms for performance improvement. Section 5 provides evaluation and comparison results. Section 6 discusses related work and Section 7 concludes.

2. PARSe Architectural Overview

In our PARSe architecture, each node in the ad hoc network keeps track of a number of nodes in its vicinity within \( R \) hops away. This defines the zone of a node. Each node chooses its zone independently, and hence no major re-configuration is needed when a node moves or fails. There is no notion of zone head, and no elections that require consensus among nodes. The zone is maintained using a very localized proactive discovery protocol (similar to link state or distance vector protocols). We assume the existence of a neighbor discovery protocol by which each node identifies nodes 1 hop away. Then the intra-zone exchange occurs with the neighbor information. Typically the number of nodes in the zone is a very small fraction of network nodes (100 nodes is the upper limit for number of zone nodes in our study). As part of the zone information each node keeps routes to all nodes in its zone, including the borders \( (R \) hops away), and pointers to resources in its zone. When a node, \( Q \), issues a query for a resource or a small transfer request, it first checks to see if the resource (or destination) is in its zone. If not, then it issues a request to a (small) number of its borders \( (R \) hops away). Each border, \( B \), receiving the request would in turn select another node, \( C \), to which to forward the request. To increase search efficiency, \( C \) should have low zone overlap with \( Q \). We call \( C \) a contact node. Contact nodes act as short cuts that bridge between (almost) disjoint zones. This helps to reduce the degrees of separation between the querier nodes and the target nodes. Degrees of separation in this context refer to the number of intermediate nodes to get to the target from the querier node. The main architecture is shown in Figure 1, where the querier node \( Q \) (potentially any node in the network) chooses three of its borders, \( B1, B2, B3 \) to which to send a request message. The borders in turn choose three contacts, \( C1, C2, C3 \) to which to forward the request. The number of contacts (or borders) chosen, \( NoC \), is a design parameter. Contacts may be chosen at a distance of \( r \) hops.

\(^1\) We use the term request and query interchangeably throughout the document. When used, any of these terms indicate a request (for object), small-transfer, or query.
away from the border, where \( r \) is also a design parameter. If \( r=R \) then the contact is a border of a border of \( Q \).

Several questions arise regarding setting the design parameter space, such as how many contacts (NoC) should be used? What is the distance (\( r \)) at which the contacts should be selected? and what is the zone radius (\( R \))? We shall investigate these design dimensions later in the evaluation section. First, we address the question of how to select useful contacts.

![Figure 1](image-url)  
Figure 1. Each node in the network has a zone of radius \( R \). 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, \( B_i \), chooses one of its borders, \( C_i \), to be the direction for forwarding the request \( r \) hops further until it reaches the contact. In this example \( r=3 \) and \( C_i \)'s are the contacts.

3. 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.

3.1. Contact Selection Protocol

As was alluded to before, a contact node acts 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 useful contact to have a zone that does not overlap significantly with that of the querier node or the other contacts. This is a distributed algorithm in which contacts do not know about each other, and in which the contacts do not know their shortest distance from the querier (remember that contacts are outside of the querier’s zone). This problem lends itself to finding the minimum dominating set (MDS) of nodes, which has been proven to be NP-Complete [27]. Instead of attempting to find an optimal or near optimal solution, we develop simple heuristics that attempts to reduce zone overlaps, while incurring very low communication overhead and scaling to a large number of nodes, potentially tens of thousands.

The first kind of overlap occurs between the contact’s zone and the querier’s zone. To attempt to reduce this overlap the main idea is to push the request as far out from the querier’s zone as possible. 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 zone rendering the contact ineffective and the query success rate becomes low. Another simple approach is to avoid sending the request through the node from which it was received. However, wireless networks have a high clustering (or cliquishness) coefficient [1][18]). This means that, on average, the neighbors\(^2\) of a neighbor of \( B \) are also neighbors of \( B \) with high probability. 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 zone. 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 \( C_1 \) through node \( x \). Node \( x \) is a neighbor of node \( L \) and is within \( Q \)'s zone, and hence would lead to a contact less than \( R+r \) hops away. In many cases the contact chosen this way may not be useful, with a zone heavily overlapping with \( Q \)'s zone.

The main problem in forwarding the request outside of \( Q \)'s zone to a useful contact is the loss of direction for the forwarded message at the border of the zone (since \( Q \) knows only about nodes \( R \) hops away). To achieve a sense of direction, we introduce a simple heuristic that uses information about the neighbors of \( B \)'s previous hop, \( L \). We shall explain this heuristic next and illustrate how well it performs in the evaluation section.

A querier node, \( Q \), sends a request to NoC of its borders. Let us focus on 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 zone. As was mentioned before, due to high clustering\(^3\) many of \( L \)'s neighbors (all of which are in \( Q \)'s zone) 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, \( C_2 \), that has significantly less zone overlap with \( Q \) than \( C_1 \) does. If \( B \) cannot find a contact without passing through \( L \)'s neighbors, then it chooses a contact (randomly) that does not pass through \( L \). Note that this scheme does not guarantee non-overlap but it reduces it drastically as we shall show later on in our evaluation section. We call this scheme the zone overlap reduction (ZOR) scheme. Note that for the above examples we have used \( r=R \) for illustration. From a design perspective there is no reason why \( r \) has to equal \( R \). In cases where \( r \) is not equal to \( R \), ZOR 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 ZOR 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 ZOR again to find its own direction border, and so on. ZOR is performed without incurring any extra communication overhead and in general is performed $\lceil r/R \rceil$ times at each chosen border.

![Figure 2](image.png)

Figure 2. (a) The border node, $B$, forwards the request towards its border $C_1$ via node $x$. $C_1$'s zone has significant overlap with $Q$'s zone. By using only 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, all of which are in $Q$'s zone. Hence, $B$ is more likely to choose a useful contact, $C_2$. The overlap between $C_2$'s zone and $Q$'s zone is a lot less than overlap between $C_1$'s zone and $Q$'s zone. ($L$'s neighbors are those inside the solid line circle of radius $tr$, including nodes $x, y, z$).

The second type of zone overlap occurs amongst contacts. To attempt to reduce this overlap we introduce a scheme that does not guarantee non-overlap between contacts’ zones, but performs quite efficiently during requests, as we shall show. We call this scheme the route overlap reduction (ROR) scheme.

### 3.2. Levels of Contacts – putting the first pieces together

The above contact selection schemes (ZOR 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, if applicable depending on $\lceil r/R \rceil$) perform ZOR 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 maximum number/depth of levels called maxDepth or $D$. 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 ZOR and ROR. That is, a level-$i$ contact, selects borders with maximally disjoint routes from its set of borders that do not pass through its previous hop ($L$’s) neighbors.

### 3.3. Search Policies – putting all the pieces together

Given a request and a maximum depth, $D$, 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 (or lbl, for short), 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., $d_i = 2d_{i-1}$.

#### 3.3.1. 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 reached (or visited) is $(NoC)^i$, and the theoretical number of hops traversed is $(R+r)(NoC)$. Hence, the number
of transmissions is given by \((R + r) \sum_{i=1}^{D} (NoC)^i\). We emphasize that this is only a theoretical (very loose) upper bound. The search employs loop and re-visit prevention mechanisms (described later), the effect of which are not considered in this simple theoretical analysis. After considering these mechanisms via detailed simulations, the overhead estimates are reduced drastically, as will be shown in the evaluation section.

### 3.3.2. 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 after increasing \(d\). This is analogous, in a sense, to expanding ring search but at the contact level. Hence, the number of contacts visited with each attempt is given by \(\sum_{i=1}^{d} (NoC)^i\), and the overall number of hops traversed (or requests transmitted) is \((R + r) \sum_{d=1}^{D} \sum_{i=1}^{d} (NoC)^i\). Again, these are only theoretical estimates and the detailed simulation results will be given in the evaluation section.

### 3.3.3. 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 \(d_{\text{max}}\) is reached, where \(d_{\text{max}}\) is the first \(d_i\) that satisfies the inequality \(2d_{\text{max}} > D\) for \(D > 2\). (For \(D \leq 2\), \(d_{\text{max}} = D\). For example, if \(D = 20\) then \(d_{\text{max}} = 16\).

An example of lbl (or step) with \(D = 2\), \(R = r = 3\), and \(NoC = 3\) is given in Figure 3 (b). Both schemes are identical for \(D = 2\). Note that level-1 contacts visited on the first attempt are not necessarily similar to 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 effect. We shall investigate this effect further in the evaluation section. Another performance implication due to the different policies is in the request delivery time. 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.

---

For lbl and step, between the different attempts the querier waits for a time \(t\), proportional to the hops traversed for that attempt; i.e., \(t \propto d_i (R + r)\). Single-shot does not use \(t\), since it uses a single-attempt per request.
4. Request Processing & Performance Enhancement

The rules for processing the requests are the same for all of the above policies. This section presents details of request processing and forwarding. In addition, we present mechanisms for improving the performance of the search. The first, loop prevention, is an essential part of our architecture. The second, mobility-based replication, is optional, and provides improvement in cases of mobility.

4.1. Request Forwarding and Processing

4.1.1. The request message

The request message contains the target ID, which could be the node ID or the resource key (for resource discovery). For small transfers, the request may also carry the data. The destination ID in the request 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 \(SN\).

4.1.2. Loop prevention and re-visit avoidance

As the message is forwarded, each node traversed records the \(SN, Q\) and the previous hop node, \(P\), 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 \(SN\) and \(Q\), 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, at any level, exists in the same zone as the querier (e.g., due to a request branch going in a circle), then the contact drops the request since this part of the network will have been already covered.

4.1.3. Search, processing and forwarding

A contact (or a border node) receiving the request, first performs a target search via a local lookup in the zone information\(^5\). If the target is found, the request is delivered and a response is forwarded on the reverse path, 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 to determine which functions to perform, and whether it is a contact, two fields are included in the request message. The first field is the level-count, and the second field is the hop-count for each level. 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:

\[ \text{If hop-count reaches '0', 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 ZOR and ROR as in section 3), and sends the request to those borders.}\]

\[ \text{If the hop-count is not '0', and current node is the destination of the request message, the receiving node acts as a border node. It selects a direction border (using ZOR as in section 3), and sends the request to it.}\]

\[ \text{Otherwise, the request is simply forwarded to the next hop to the destination.}\]

Note that the request message is \emph{unicast} hop by hop, it is \emph{not} broadcast hop by hop. This has a direct implication on the energy consumed at each hop. In broadcast, all the sender’s neighbors consume energy to receive the message, whereas in unicast only the intended recipient consumes full reception energy, after a handshake for channel reservation, other neighbors may go into idle/sleep mode. We shall re-visit this point in detail in the evaluation section.

4.2. Mobility-based replication

In many cases the requests are issued for small objects that may be replicated\(^6\). Examples include small-file sharing, sensed data, distributed mobile databases, computing and data storage, and peer-to-peer networks. In order to reduce the search overhead, increase success rate, and increase robustness against node failures and network partitions, we propose a simple replication scheme that takes advantage of node mobility. In this scheme the producer of the data (the node holding the object) sends a copy of the object to its direct neighbors (1 hop away). The node may use the intra-zone information to identify those nodes likely to move out of the zone. As the nodes move away, the replicas become distributed throughout the network. The more the mobility, the more the distribution of the replicas. We expect this to reduce overhead and delay of object access. We study performance of this scheme in the evaluation section.

5. Evaluation and Comparison

In this section we study the various dimensions of the design space for our architecture, PARSe. In addition, we compare our protocols to other related approaches including flooding, expanding ring search\(^7\) (and its variants) and border-casting (as proposed in [13][14]).

---

\(^5\) Optionally, other nodes en route may also perform the lookup. In our experience, this accounts for less than 2% performance gain.

\(^6\) There is a large body of work that addresses object replication, including recent work in [25]. Many such techniques (e.g., uniform, proportional replication) may incur high replication overhead. Our mechanism is complimentary to that body of work in the sense that we provide an additional (or alternative) mechanism that may be used for communication-efficient replication using mobility.

\(^7\) We have investigated several variants of expanding ring search: with increments of 1 and 10, and with exponential increments starting from 1 and starting from 10. We found that the best of those strategies (the exponential increment-1) performs slightly worse than flooding while the others perform
particularly, for PARSe, we attempt to 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 zone radius (R)? (5) What is the best search policy, single-shot, lbl or step? (6) How does replication affect the protocol performance? and most importantly, (7) Is there a specific combination of settings that scales well? (That is, is there a specific combination that performs well for a wide variety of network sizes; small, medium and large?).

the main performance metrics include communication overhead (more precisely the energy consumption resulting from communication) and the request success rate. 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 failure beyond an acceptable level will be recovered using flooding. Hence, the scheme used here is contact-based search, if failed then fallback to flooding. Since this penalty 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.

Next we describe our simulation setup and scenarios, then we present and analyze the results.

5.1. 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,R,NoC,D and replication. We also evaluate the overall communication overhead and energy for our architecture. Overhead consists of two components: (a) zone establishment and maintenance (or intra-zone overhead), and (b) request (or query) overhead.

the intra-zone overhead is a function of the number of nodes within the zone (among other factors). This number increases as a function of $R^2$ and node density. In addition, each mobile node needs to store the zone information. Hence, we should not increase $R$ arbitrarily. In fact, we put a limit of 100 nodes per zone for our study, and choose $R=3$. Transmission radio range ($tr$) is taken as 110m. We study a wide range of network sizes, as shown in Table 1. We also vary the area of the network to maintain network connectivity (with $\approx 2.5\%$ unreachability due to partitions), and to keep the zone nodes under 100 (for zone radius of $R=3$). N nodes are randomly placed in a square of $\ell m \times \ell m$, where $\ell$ is the side of the square in meters.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Area (m²)</th>
<th>Node Degree</th>
<th>Border Nodes</th>
<th>Zone Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>1000x1000</td>
<td>7.6</td>
<td>15.1</td>
<td>35</td>
</tr>
<tr>
<td>500</td>
<td>1400x1400</td>
<td>8.9</td>
<td>20.5</td>
<td>44.8</td>
</tr>
<tr>
<td>1000</td>
<td>2000x2000</td>
<td>9.1</td>
<td>21.7</td>
<td>46.6</td>
</tr>
<tr>
<td>2000</td>
<td>2800x2800</td>
<td>9.7</td>
<td>24.7</td>
<td>52.9</td>
</tr>
<tr>
<td>4000</td>
<td>3700x3700</td>
<td>11</td>
<td>30.3</td>
<td>62.2</td>
</tr>
<tr>
<td>8000</td>
<td>4800x4800</td>
<td>13</td>
<td>38.8</td>
<td>77.8</td>
</tr>
<tr>
<td>16000</td>
<td>6500x6500</td>
<td>14.3</td>
<td>44.6</td>
<td>88.2</td>
</tr>
<tr>
<td>32000</td>
<td>9200x9200</td>
<td>14.3</td>
<td>45</td>
<td>88.9</td>
</tr>
</tbody>
</table>

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

For mobility we use the random way point model, where nodes choose velocities and destinations independently. A destination is chosen randomly (within the network area) and a velocity is chosen randomly from $[0, V_{max}]$. Once the destination is reached, another random destination is chosen, so on. We use $V_{max}=0$ to 60m/s, and various request/query rates varying from 0.01 query/km up to 1000 query/km. We implement the protocols under study in a network simulator using C++. For the purposes of our simulation we assume there is no contention at the MAC layer and do not model collisions. However, instead of measuring only number of transmissions, we implement a communication energy model at each hop.

Hop-by-Hop Communication Energy Model

The energy consumed when a request is sent (or forwarded) at each hop is due to packet transmission at the sender and packet reception at the recipient(s). Depending on the mode of the message, whether unicast, multicast or broadcast, the number of actual recipients vary. By recipients we do not mean only the intended recipients but also other nodes (within the transmission range) that are in receive state. In general, a wireless node may be in one of three power states: (i) transmit state, (ii) receive state, or (iii) idle/sleep state. The power expended in each of these states may vary drastically. Also, the overall power consumed is a function of the duration of stay in any of these states (mainly a function of the packet size). We refer to the amount of energy consumed during the transmission of a request packet as $E_{tx}$. Similarly, $E_{rx}$ refers to the energy of request reception. If a message is broadcast, it is received by all other nodes within radio range; i.e., all neighbors. The average number of neighbors per node is the average node degree (g). For a unicast message, usually there is a small handshake phase to inform the neighbors of the impending transmission. In IEEE 802.11 (the model we adopt), CSMA/CA is used with handshake and medium reservation. The handshake involves broadcast of a small message, request-to-send (RTS), to which the intended recipient responds with a broadcast of a small clear-to-send (CTS) message. This RTS/CTS exchange causes the neighbors to transit into the idle/sleep state until the end of request transmission. We refer to the power

---

8 Request failures may be tolerated up to an acceptable level due to possible network partition (due to mobility or imperfect node distribution, such as isolated islands of nodes). We found that level to be $\sim 2.5\%$ for our study as will be explained later.

9 For a constant node density (per m²) the number of nodes in the zone is directly related to the zone area ($\pi R^2$).

10 The radio range was based on reasonable averages from data for the Lucent Orinocco, Cisco Aironet and 3Com wireless cards.

11 Modeling MAC layer details and collisions using packet-level simulator (os[28]) was found not to scale to large networks. However, in general, collisions have greater impact on protocols using broadcast messages due to the higher likelihood of collisions as opposed to using unicast messages with RTS/CTS mechanisms that reduce the probability of collisions drastically.
consumption due to handshake as \( Eh \). Based on this understanding we use the following energy model:

- Energy consumed by a unicast message (\( Eu \)): 
  \[
  Eu = Etx + Erx + Eh = Etx(1 + f + h),
  \]
  where \( f = Erx/Etx \) and \( h = Eh/Etx \).

- Energy consumed by a broadcast message (\( Eb \)): 
  \[
  Eb = Etx + g\cdot Erx = Etx(1 + f + g),
  \]
  where \( g \) is the average node degree.

For this study we use \( f = 0.64 \), and \( h = 0.11^2 \). Hence, the simulator differentiates between (hop-by-hop) unicast and broadcast messages and applies the energy model accordingly. To have the results be independent of the packet size used, we measure in the energy consumed as a function of \( Etx \); i.e., we have the energy measure in \( Etx \) units.

Following we discuss the results of our simulations. The first part of the results discusses the effect of various parameters, \( r \), \( NoC \), \( D \), and \( replication \), on the performance of different search policies using contacts (with flooding penalty for failures). For this set of simulations we use the 1000 node topology in Table 1. Then we present results using all the network topologies in Table 1 to compare per query overhead and overall overhead (including intra-zone maintenance) for the different search policies of PARSe, in addition to flooding and border-casting. For this second set of simulations we use specific parameter settings for PARSe policies based on the first set of simulations. To combine intra-zone overhead (a function of mobility) and query overhead (a function of the query rate) we use normalized per m/s metrics, and will be explained later.

Each data point represents an average of 10 simulation runs with different random seeds. Low variability between runs was observed. Qurer-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.

We first present the overhead per request (hereafter referred to as overhead per query), then we provide the intra-zone overhead, and finally we present the overall overhead.

### 5.2. 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 including mobility-based replication. Our aim is to understand the behavior of the different PARSe policies with the various design parameters, and study trends to aid us in identifying desirable parameter settings.

#### 5.2.1. 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 \( r \) and clearly indicates favorable settings for the different search policies. In general, as \( r \) grows, the contacts’ location extends farther away from the querier’s zone. For single-shot policy, as \( r \) increases we see a consistent drop in the request success rate due to reduced coverage of areas near (just outside the zone of) the request sender (be it \( Q \) or any of its contacts). This effect was qualitatively illustrated in Figure 3. This drop in success rate translates into fallback to flooding, which consistently produces more energy consumption. Hence lower values of \( r \) (0 ≤ \( r ≤ 2 \)) are preferred for single-shot. 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 still maintain good coverage (and hence high request success rate) with increasing \( r \) up to a certain distance, after which increasing \( r \) generally leads to drop in success rate. At very low values of \( r \) (e.g., \( r ≤ 2 \)), although \( lbl \) and \( step \) achieve high success rate, they also incur added overhead due to zone overlap between \( Q \) and level-1 contacts (and in general between level-\( i \) contacts and their respective level-\( i+1 \) contacts). This overlap reduces with increase in \( r \), with the best values around 3-8 hops (3 being best for \( step \), 5 and 8 being best for \( lbl \)).
policies, a very low number of contacts (for single-shot), power consumption by increasing the success rate. It is revealed that increasing NoC is drastically reduced. After certain visited is high, but due to the loop and re-visit prevention mechanisms this number is drastically reduced. 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 energy consumption almost saturates (with very slight, almost negligible, increase with increase of D). Note that D=33 for step translates into a maximum of 6 attempts. It is important to mention that an increase in D does not necessarily translate into increase in number of attempts for a successful request, and hence increase in delivery (or response) delay. In fact, the average number of attempts in the simulations (for successful as well as unsuccessful requests 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.

5.2.4. Effect of random and mobility-based replication

The degree of replication represents the number of copies of a target object within the network, with degree of 1 meaning no replication. We present two sets of simulations to show the effect of replication on the protocol performance. The first set uses replication degrees from 1 to 10 and assumes that these replicas exist randomly across the network. The second set uses the mobility based replication scheme discussed in section 4.2 in which the object is broadcast to only the first hop neighbors. Figure 7 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, and the different policies converging toward similar overheads with high degrees of replication. For lbl and step this mainly happens due to the decrease in the average required number of attempts before success (i.e., reaching the first/nearest 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 when more branches terminate earlier because they are able to locate a replica.

Results for mobility based replication are shown in Figure 8, where mobility is varied between 0m/s (no mobility) and 60m/s. In this simulation, 200 different objects were assigned to (or generated by) randomly chosen nodes, which we call sources. The objects were replicated at the sources’ neighbors at time 0. After 15 seconds of simulation (using random way point mobility) random objects were requested by 500 random nodes. The simulation was repeated 10 times with random seeds. Replication overhead varies with the size of the object itself and was not considered in the analysis. (If the number of requests per object is reasonably high then replication overhead will not affect the trend. We assume that these objects are in general relatively small). We notice that as mobility increases the overhead decreases in a way similar to the decrease with increase in degree of replication. Hence, mobility in this case effectively distributes the initially clustered replicas. With the increase of mobility beyond a certain point (45m/s in the graph) the performance saturates, as the maximum effect of replication is reached. A similar effect may be observed with extension of simulation time.

5.2.2. 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=2 (for single-shot), r=8 (for lbl) and r=3 (for step). For all policies, a very low number of contacts (NoC<3) achieves low success rate and incurs high energy consumption due to fallback flooding. 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. For lbl and step the best setting is at NoC=3, while for single-shot favorable settings include NoC=3-5 with the best setting being NoC=4 mainly due to the inability of 3 contacts to establish complete coverage near contacts’ zones.

5.2.3. 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 power consumption by increasing the success rate. It is not the case, however, as one might expect, that increasing D exponentially increases the number of contacts visited. It is true that the number of contacts that may be potentially visited is high, but due to the loop and re-visit prevention mechanisms this number is drastically reduced. After certain
In this section we evaluate the scalability characteristics of the PARSe protocols. In particular, we want to investigate how the energy consumption grows with the increase in number of nodes in the network. There are two main overhead components for PARSe: (a) query overhead, and (b) intra-zone establishment. 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. Intra-zone overhead, on the other hand, is a function of the degree of mobility (m/s), the number of nodes in a zone, the number of zones (or 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 present a metric called $QMR$ (query-mobility-ratio, simply $q$) defined per node as query/s/(m/s) or simply query/km. This is very similar to $CMR$ (call-mobility-ratio) in [13] but is defined for queries or request rate as opposed to route discovery or call rate. For simplicity, let us assume that the size of the intra-zone exchanges is, on average, comparable to the size of a request\textsuperscript{14}, and let us call the intra-zone overhead $Z(R)$, defined in terms of $Etx$ energy units and is a function of the zone radius, $R^2$. $Z(R)$ has units of ‘energy ($Etx$) per sec per node per m/s’. Also, let us call the energy consumption per query $E_{protocol}$, so the energy consumption per query for single-shot, lbl and step are $E_{single}$, $E_{lbl}$ and $E_{step}$, respectively. Similarly for flooding and border-casting, we have $E_{flood}$ and $E_{border}$. The units for $E_{protocol}$ are given in ($Etx$) units per query. Hence, we get the overall query overhead for the lbl policy (for example), $E_{Qlbl} = q.E_{lbl}$, where $q$ is $QMR$ as defined above. The units of $E_{Qlbl}$ are in `$Etx$ units per sec per node per m/s’, compatible with $Z(R)$ above. The total overhead for lbl (for example) becomes $E_{Tlbl} = Z(R) + E_{Qlbl}$.

Our goal in this section is to obtain trends and comparisons for $E_{protocol}$ and $Z(R)$, for PARSe protocols as well as related schemes, over various sizes of networks ranging from 200 nodes to 32,000 nodes. (SeeTable 1).

**Related schemes**

We compare our protocols to flooding and border-casting\textsuperscript{16}. For flooding, in a network of $N$ nodes, the request is transmitted, in general, by $N^{-1}$ nodes. These transmissions are sent over ‘2.L-g’ links, where $L$ is the number of links in the network (a link being defined by a pair of nodes that lie within transmission range), and $g$ is the node degree. By definition $g = 2.L/N$, so the number of receptions is given by

\textsuperscript{14} Note that this assumption may not be accurate, as the control messages (for intra-zone) are usually smaller than regular packets. This, however, acts in favor of flooding which does not incur zone overhead. So, in a way the obtained overhead may be considered as an upper bound. Also, our goal is to study trends and not obtain precise numbers. Obtaining precise numbers is a task at which simulations (vs. experiments) are not suitable in general, especially for wireless networks.

\textsuperscript{15} Here, $R$ is the number of hops over which the intra-zone exchange occurs (from a node’s perspective). For PARSe, this exchange occurs over the zone radius, $R$. For border-casting, however, the intra-zone exchange occurs over $2R-1$. We shall get back to this point during comparisons.

\textsuperscript{16} In border-casting [13][14] the querier, $Q$, sends the request to its borders, and the borders send it to their borders, so on. As the request is forwarded, borders record previous border information (route information does not accumulate in the request message but is stored in the nodes to reduce overhead). 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. (For the neighbors to record such information the request message must be broadcast (or multicast) at each forwarding node. If unicast messages are used, then the number of resulting transmissions increases significantly.) 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). This incurs more intra-zone overhead but reduces request overhead.
g.\( (N-1) \). Knowing the approximate number of expected transmissions and receptions we can estimate the energy consumed by flooding. Our flooding simulations show very close results to this simple analytical model of flooding, but incur slightly less overhead, mainly due to network partitions.

We first analyze results of the query overhead, then intra-zone overhead, followed by analysis of total overhead.

5.3.1. Scalability of Query Overhead

Simulation results are analyzed for the different PARSe protocols (or search policies). Parameter setting was based on earlier analysis. For single-shot we present results for two settings, the first includes \( r=2, \text{NoC}=4 \), and the second includes \( r=3, \text{NoC}=3 \). The maximum depth, \( D \), was increased to 65 to achieve better success rate for single-shot. For \text{step} and \text{lbl} we used \( D=33 \) and \( \text{NoC}=3 \). For \text{step} we used \( r=3 \), and for \text{lbl} \( r=8 \). Results are presented in Figure 9. For all network sizes it is clear that the \text{step} policy achieves the best performance (with success rate of 97.5\% or better for all network sizes). Achieving similar success rate but with more overhead is the \text{lbl} approach. Single-shot exhibits an interesting behavior, for settings of \( (\text{NoC}=4, r=2) \) single-shot achieves between 90-96\% success rate for sizes below 4000, then the success rates go above 97.5\% from 4000 nodes and on, consistently rising with increase in \( N \). For single-shot with \( (\text{NoC}=3, r=3) \) lower success rates (82-89\%) are reached for sizes below 4000 (as expected and was explained in our earlier analysis), the success rate increases to (94-97\%) for 4000-8000 nodes. After 8000 nodes this setting achieves 97.5\% and above success rate. It is interesting that single-shot with this \((3,3)\) setting achieves less success rate than the previous \((4,2)\) setting, but yet there is a cross-over point at 4000 nodes after which single-shot\((3,3)\) performs much better than single-shot\((4,2)\). For higher \text{NoC} and lower \( r \), single-shot achieves better success rate at medium networks (as was explained earlier). However, for very large-scale networks (above 8000) single-shot\((3,3)\) performs much better than single-shot\((4,2)\), and approaches performance of \text{step} for 32,000 nodes. There are two reasons for this. First, for lower \text{NoC} single-shot incurs less overhead. Second, 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.

Figure 10 shows the trend for average number of attempts with increase in nodes, which reflects the response delay (to an extent). The single-shot average is always around 1 (as expected), and the largest increase occurs for \text{lbl} (reaching 13.7 attempts for 32,000 nodes). \text{Step} has good scaling characteristics, with 5.2 average attempts for 32,000 nodes.

5.3.2. Comparison with Related Schemes (Query Overhead)

Based on this analysis, our feeling is that \text{lbl} provides no advantage over single-shot or \text{step}. \text{Step} provides the best performance in terms of energy consumption, and possesses desirable scaling characteristics in terms of overhead and 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 \text{step}, however, is its persistent good performance over a wide spectrum of network sizes, with the setting \((\text{NoC}=3, r=3, D=33 \ [\text{max attempts}=6])\).

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 \text{step}, but for getting the best response delay we may use single-shot). It is conceivable that different policies may be used for different kinds of requests, simply by putting 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.

Figure 9. Scalability of query overhead for PARSe policies

Figure 10. Average attempts per query for PARSe policies
improvement in performance using contacts, especially in large-scale networks. This is due to two main reasons: (i) decrease in number of transmitted packets per query, and (ii) avoiding broadcast and using unicast for all messages. Figure 12 shows the query energy-consumption ratio, QER, of step to the other protocols. That is, $QER_{protocol} = \frac{E_{step}}{E_{protocol}}$ where protocol is lbl, single(4,2), single(3,3), flood and border. $QER_{flood}$ ranges between 5.5% (for large networks) and 8.5% (for small-medium networks), while $E_{CR_{border}}$ ranges between 13.5% (for large networks) and 22% (for small-medium networks). As for comparing PARSe policies, $QER_{lbl}$ ranges from 56% (for large networks) to 83.5% (for small-medium networks), while $QER_{single}$ ranges from 34.5% (for small networks) to 96.5% (for large networks). The figure shows the two settings of single-shot that we have alluded to (and analyzed) before.

Figure 12. Query overhead of PARSe, flooding and border-casting

### 5.3.3. Intra-zone Overhead

The intra-zone overhead includes the energy consumed during intra-zone message exchange. Although not necessarily required for PARSe, we use a link state protocol similar to that used in [13]. Alternatively, we can use more efficient zone maintenance protocols (e.g., [17] or other). For link state, the zone exchange is in the form of broadcast messages within the zone. This exchange increases linearly with mobility (as more zone changes occur). As was described before, we normalize this overhead with respect to mobility using $Z(R)$. The intra-zone overhead is also a function of the number of nodes in the zone. This number is a function of $R$, and increases with the zone area (i.e., with $R^2$).

Figure 13 shows $Z(R)$ for PARSe as $Z(3)$ and for border-casting as $Z(5)$ (remember that border-casting uses link exchange of 2R-1 to employ efficient early termination).

Figure 13. Normalized Intra-zone overhead for the basic zone $R=3$, $Z(3)$ and the extended zone of 2R-1, $Z(5)$

#### 5.3.4. Comparisons of Total Overhead

The total energy consumed is the combined effect of intra-zone 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: $E_{TER_{step}} = Z(R) + q.E_{step}$. For flooding, the intra-zone overhead is not incurred, so $E_{TER_{flood}} = E_{flood}$. For border-casting the intra-zone overhead is incurred for an extended zone of 2R-1, hence $E_{TER_{border}} = Z(2R-1) + E_{border}$. We evaluate the total energy-consumption ratio, $TER$, of step to the other protocols. We get:

$$TER_{flood} = \frac{E_{TER_{flood}}}{E_{TER_{step}}} = \frac{Z(R) + q.E_{step}}{q.E_{flood}},$$

and

$$TER_{border} = \frac{E_{TER_{border}}}{E_{TER_{step}}} = \frac{Z(2R-1) + q.E_{border}}{Z(R) + q.E_{step}}.$$

Figure 14 and Figure 15 show $TER_{flood}$ and $TER_{border}$ respectively, as function of the QMR (query-mobility ratio) $q$ (query/km). We note that a logarithmic scale was used for $q$ to resolve the rapid drop in the total energy-consumption ratio. Also note the difference in the y-axis scale for $TER$. We first analyze the behavior of $TER_{flood}$ with the change in $q$. To re-iterate, $q$ is the number of queries/sec per mobility (km/s) per node (or simply query/km per node). We investigate a wide range of query rates. Results are shown in
mainly because border-casting is also a zone-based approach and incurs more intra-zone overhead by using the extended zone of radius (2R-1). Effect of the extended zone is clearest for small QMR where the intra-zone overhead has the dominant effect, whereas for high QMR the effect is mainly due to the query overhead. For a small network (200 nodes) and for low q, we get \( TERR_{border} \approx 48\% \), while for high q, \( TERR_{border} \) is just below 25%. For medium to large-scale networks (500-32000 nodes) and for low q, \( TERR_{border} \) ranges from 37% to 44%, and for high q, \( TERR_{border} \) ranges from 13% to 20%. Hence, the best gains for PARSe can be observed for higher values of QMR, where \( TERR_{border} \) approaches 13.5% for large networks.

6. Related Work

We address the problem of resource discovery in infrastructure-less wireless mobile ad hoc networks. Hence, architectures that require infrastructure (e.g., DNS) 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.

Ad hoc routing protocols, in general, may be classified as reactive (on-demand), proactive (table-driven), hybrid, or hierarchical. Proactive schemes such as DSDV [3], WRP [20] and GSR [19] cause updates to be periodically flooded throughout the network. These schemes may be suitable for small scale networks (of 10s of nodes) and may be used as intra-zone protocols in our scheme. Reactive schemes such as AODV [4] and DSR [5] attempt to reduce the overhead due to periodic updates by maintaining state only for the active sources. In these schemes a search may be initiated for each new request. However, the search procedure generally involves flooding (or expanding ring search). These schemes may be suitable for small-medium size networks (of 100s of nodes). The above (proactive and reactive) protocols were generally designed to attempt to find shortest path routes. In many cases, establishing such routes incurs much more overhead than is needed for resource discovery or small-transfers. Especially for large scale networks (of 10s of thousands of nodes), the above schemes incur excessive overhead unsuitable for our target applications.

The zone routing protocol (ZRP) [9][11][12][13][14] uses a hybrid approach, where link state is used intra-zone and on-demand routing (border-casting) is used inter-zone. A feature of ZRP is that a zone is node-specific, and no complex coordination is used. The zone concept used in PARSe is similar to that used in ZRP. However, we avoid border-casting by using contacts out-of-zone. The main concepts upon which contacts were designed (short cuts to create a small world and on-the-fly contact selection) are fundamentally different than ZRP. Also, the target application is different. ZRP, as a routing protocol, attempts to find high quality routes, for prolonged transfers. In our

17 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 intra-zone exchange. Maintaining zone information without being active is not desirable.
study, we target resource discovery and small transfers, and we make a clear design trade-off between route optimality and the reduction of overall communication overhead. In section 5 we presented detailed comparisons with border-casting and showed that the contact-based approach incurs significantly lower overhead for our purposes. We believe that our work may be complementary to zone routing. With simple extensions to zone routing, it is quite conceivable that the two maybe integrated in a unified architecture where border-casting is used for route discovery while PARSe is used for resource discovery and small transfers.

Hierarchical schemes, on the other hand, e.g., CGSR [21] and [22], involve election of cluster-heads. The cluster-head is responsible for routing traffic in and out of the cluster. Also, a cluster head may be a single point of failure and a potential bottleneck. Other hierarchical schemes use landmarks [6][15][16]. Landmark routing avoids traffic concentration by using the direction of landmarks for routing. Landmarks do not necessarily forward the packets for their respective zones. Advertisement, promotion and demotion schemes are used for node coordination to construct the hierarchy. Cluster-based and landmark-based hierarchies rely on complex coordination and thus are susceptible to major reconfiguration due to mobility, leading to serious performance degradation. We do not employ any coordinated election schemes. In our architecture each node maintains its zone independently, so no major re-configuration is incurred with mobility.

Related work on smart flooding is given in [17][23][24]. The main idea is to exploit node density to reduce redundant transmissions, sometimes reducing coverage. Such work is complementary to our work. For high density networks, this work maybe integrated with our work to provide more efficient intra-zone exchange.

In GLS [7] an architecture is presented for location discovery that is based on a grid map of the network. This map must be known a priori by all nodes. Nodes use geographic routing and the network map to recruit location servers to maintain their location. Nodes use a consistent mapping algorithm to update and search for node locations. This is a useful architecture given the network map is known and geographic data is available (through GPS or other). These assumptions may not hold in our case. Also the ID of the target node must be known in GLS. By contrast, for resource discovery, the target resource may reside at a node with an ID unknown to the querier.

The algorithms proposed in [2][8] use global information about node locations to establish short cuts or friends, and use geographic routing to reach the destination. It is unclear how such architectures are feasible with mobility. Also, the destination ID (and location) must be known in advance, which may not be the case in resource discovery.

7. Conclusion and Future Work

We have presented a novel architecture for resource-discovery and small-transfers in large-scale ad hoc 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 zone, 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 zone. 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).

This work is the first work, of which we are aware, for resource discovery in large-scale ad hoc networks that does not use tightly coordinated hierarchical schemes. This renders our scheme more robust and resilient to mobility. We do not assume availability of geographic information.

The main contributions of this paper include:

- Introducing the contact-based PARSe architecture for power-efficient search in large-scale ad hoc networks
- Designing a simple, effective, on-the-fly contact selection protocol for zone-overlap reduction
- Supporting various search policies and presenting mechanisms for loop-prevention and mobility-based replication 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 border-casting 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 large networks and high request rates, PARSe consumes as little as 5.5% of flooding energy and 13.5% of border-cast energy. We provide different search policies that may be suitable for different situations. 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 use of geographic information, when available, to further enhance the performance of PARSe. Knowing geographic location of the contacts may allow for better zone-overlap reduction schemes. We shall also explore utility of our architecture in location-aware networks. PARSe can still be used for resource discovery and small transfers in large-scale ad hoc networks. Our protocols may be easily implemented using simple extensions to zone-routing protocols. We also plan to further study mobility-based replication using various mobility models [26].
References


