Contact-Based Architecture for Resource Discovery (*CARD*) in Large Scale MANets

Ahmed Helmy[*†], Saurabh Garg[†], Priyatham Pamu[§], Nitin Nahata[†]
[†]Electrical Engineering Department [§]Computer Science Department
University of Southern California
{helmy, sgarg, pamu, nnahata}@usc.edu

Abstract

In this paper we propose a novel architecture, CARD, for resource discovery in large scale MANets that may scale up to thousands of nodes. Our mechanism is suitable for resource discovery as well as routing very small data transfers or transactions in which the cost of data transfer is much smaller than the cost of route discovery. Our architecture avoids expensive mechanisms such as global flooding or complex hierarchy formation and does not require any location information. In CARD resources within the vicinity of a node, up to a limited number of hops, are discovered using a proactive scheme. For resources beyond the vicinity, each node maintains a few distant nodes called contacts. Contacts help in creating a small world in the network and provide an efficient way to query for distant resources. As the number of contacts of increases, the network view (reachability) of the node also increases, increasing the discovery success rate. Paths to contacts are validated periodically to adapt to mobility. We present mechanisms for contact selection and maintenance that attempt to increase reachability with reduced overhead. Our simulation results show that CARD can be configured to provide desirable performance for various network sizes. Comparisons with other schemes show overhead savings reaching 87% (vs. flooding) and 79% (vs. bordercasting) for high query rates in large-scale networks.

1. Introduction

Ad hoc networks are wireless networks composed of mobile devices with limited power and transmission range. These networks are rapidly deployable as they neither require a wired infrastructure nor centralized control. Because of the lack of fixed infrastructure, each node also acts as a relay to provide communication throughout the network. Applications of ad hoc networks include coordination between various units (e.g., in a battlefield), search and rescue missions, rapidly deployable networks, and vehicular networks, among others. Although research on MANets has attracted a lot of attention lately, little attention has been given to resource discovery in large-scale MANets that may scale up to thousands of nodes. Resource discovery examples include discovering capabilities (such as GPS capable nodes, printers, multicast directory servers, location-based servers, etc.), DNS-like queries (in the context of ad hoc networks), or even sensed information such as temperature, pollution, or congestion in sensor and vehicular networks. In addition, a very important mode of communication that has been largely ignored in ad hoc networks literature is that of short flows and small transactions, where the communication cost of discovering shortest routes is usually the dominant factor (not the data transfer as in long flows). For such short flows reducing overhead (not route optimization) is the main design goal.

In ad hoc networks, lack of infrastructure renders resource discovery a challenging problem. In addition, mobility induces frequent route changes. Traditional protocols proposed for resource discovery either involve global flooding or are based on complex hierarchy formation. While flooding is inefficient and therefore does not scale well, hierarchy formation involves complex coordination between nodes and therefore may suffer significant performance degradation due to frequent, mobility induced, changes in network connectivity.

To overcome these limitations we propose a new architecture for efficient resource discovery in large-scale ad hoc networks, called *CARD*. Our study targets resource discovery and routing for short flows, where route optimization is not a goal. *CARD* is *not* a general routing protocol, as we make a design decision to trade-off shortest paths for drastic reduction in discovery overhead. Our architecture is based on the concept of *small worlds* [10] [11]. In our architecture we adopt a hybrid approach in which a node uses periodic updates to reach its *vicinity* within a limited number of hops, *R,* and reactive querying beyond the vicinity via *contacts*. *Contacts* act as *short cuts* that attempt to transform the network into a small world by reducing the degrees of separation. They help in providing a view of the network beyond the vicinity during resource discovery. Each node maintains state for a few contacts beyond its vicinity. Contacts are polled periodically to validate their presence and routes. For discovering resources efficiently, queries are sent to the contacts that leverage the knowledge of their vicinity. As the number of contacts increases, the network view (reachability) increases. However, at the same time the overhead involved in contact maintenance also increases. Our results show this trade-off.

Our architecture has been designed to meet requirements for an efficient resource discovery scheme, even in large-scale networks with thousands of wireless devices. Scalability is one of our main design goals. Nodes in ad hoc networks are

[*] A. Helmy was supported by NSF CAREER Award 0134650.

usually portable devices with limited battery power. Therefore to save power the resource discovery mechanism should be efficient in terms of communication overhead. Simulation based comparisons with flooding and bordercasting [8][9] show our architecture to be more efficient. Simulation results also show that our protocol is scalable and can be configured to provide good performance for various network sizes. Overhead savings are function of the query rate, reaching 87% (vs. flooding) and 79% (vs. bordercasting) in communication savings for high query rates during high mobility in large-scale networks; a drastic improvement in performance.

The rest of this document is organized as follows. Section 2 discusses related work. Section 3 describes our design goals and provides an overview of our architecture, *CARD*, and introduces the contact selection and maintenance algorithms. Section 4 presents analysis of *CARD*, and compares it to flooding and bordercasting. We conclude in Section 5.

2. Related Work

Related research lies in the areas of routing and resource discovery in ad hoc networks. Due to lack of infrastructure in ad hoc networks, resource (and route) discovery is a challenging problem. Most of the routing protocols proposed so far can be broadly classified as: proactive (table-driven), reactive (on-demand), hybrid, or hierarchical.

Proactive schemes such as DSDV [1], WRP [3] and GSR [2] flood periodic updates throughout the network. This is resource consuming, especially for large-scale networks. Reactive schemes such as AODV [5] and DSR [4] attempt to reduce the overhead due to periodic updates by maintaining state only for the active resources. In these schemes a search is initiated for new discovery requests. However, the search procedure generally involves flooding (or expanding ring search), which is inefficient and does not scale well.

Routing protocols in general attempt to discover optimal (shortest path) routes. In many cases, establishing such routes incurs much more overhead than is needed to transfer the data (e.g., in short flows and small transactions). In our study, instead of obtaining shortest paths, we focus on reducing the overhead of resource (or route) discovery for short flows.

Hybrid schemes such as ZRP [9] try to combine the benefits of both the proactive and reactive schemes. ZRP limits the overhead of periodic updates to a limited number of hops (zone). Resources beyond the zone are discovered in a reactive manner by sending queries through nodes at the edges of the zones (bordercasting). The zone concept is similar to the vicinity concept in our study. However, instead of bordercasting we use contacts. The design principles upon which our architecture was designed (small world graphs) are fundamentally different than those used for ZRP. In our study, we show that the contact-based approach is much more efficient than bordercasting for our purposes.

Hierarchical schemes, such as CGSR [6] and [15], involve election of a cluster-head, which has greater responsibilities than other nodes. The cluster-head is responsible for routing traffic in and out of the cluster. Cluster-based hierarchies rely on complex coordination and thus are susceptible to major re-configuration due to mobility, leading to serious performance degradation. Also, a cluster head may be a single point of failure and a potential bottleneck. In our architecture each node has its own view of the network, and hence there is very little coordination between various nodes. This enables our architecture to adapt gracefully to network dynamics.

GLS [7] requires all nodes to know of a network grid mapping and assumes knowledge of node locations (via GPS or other). *CARD* does not rely on location information.

Related work on smart or efficient flooding has been proposed in [16][17]. These techniques do not use concepts of short cuts or vicinities (as we define in our architecture). Such work is complementary to our work and can be easily integrated within our work to provide even more efficiency in the vicinity establishment and maintenance phase.

In [12] a framework was proposed for multicast in large-scale ad hoc networks that introduced the concept of contacts. Our work here fits nicely in that framework. Also, in [13] the relationship between small worlds and wireless networks was shown. In this paper, we build upon that relationship.

3. *CARD* Architectural Overview

3.1. Design Requirements

The design requirements of our *CARD* resource discovery architecture for large-scale Ad hoc networks include:
(a) Scalability: Applications of large-scale ad hoc networks involve military and sensor network environments that may include thousands of nodes. Therefore the resource discovery mechanism should be scalable in terms of control overhead with increase in network size.
(b) Efficiency: Ad hoc networks include portable devices with limited battery power. Therefore, resource discovery mechanisms should be power-efficient.
(c) Robustness: The mechanism should be robust to handle frequent link failures due to mobility.
(d) Decentralized operation: For the network to be rapidly deployable, it should not require any centralized control.
(e) Independence of location information: Rapid deployability and self-configurability require network mechanisms independent of any external information.

3.2. Definitions

• *Vicinity (of a node):* All nodes within a particular number of hops (*R*) from the node. *R* is the radius of the vicinity.
• *Edge nodes (of a vicinity):* All nodes at a distance of *R* hops from the node.
• *Maximum contact distance (r):* the maximum distance (in hops) from the source within which a contact is selected.
• *Overlap:* Overlap between nodes represents number of common nodes between their vicinities.
• *Number of Contacts (NoC): NoC* specifies the value of the maximum number of contacts to be searched for each source node. The actual number of contacts chosen is usually less

than this value. This is due to the fact that for a particular value of R and r, there is only a limited region available for choosing contacts. Once this region has been covered by vicinities of the chosen contacts, choosing more contacts in the same region is not possible, as their vicinities would overlap with the vicinities of the already chosen contacts. This is according to our policy to minimize overlap.

- *Depth of search (D): D* specifies the levels of contacts (i.e., contacts of contacts) queried by a source.
- *Reachability:* The number of nodes that can be reached by a source node is termed as the reachability of the source node. This includes the nodes within the vicinity that can be reached directly and the nodes that lie in the contacts vicinity and are therefore reachable through the contact(s), etc. In a sense, this is also a measure of the discovery success rate.
- *Overhead:* Overhead is defined in terms of the control messages generated by a mechanism. For example, contact selection, maintenance or query. Procedures for these mechanisms are described in the next sections.

3.3. Mechanism Description

Our architecture employs a hybrid of proactive and reactive approaches for resource discovery. All nodes within R hops from a node form the node's vicinity. Each node proactively (e.g., using a link state protocol) maintains state for resources within its vicinity. Each node also maintains state for (a few) nodes that lie outside the vicinity. These nodes serve as contacts for accessing resources beyond the vicinity. Contacts are selected and maintained using the mechanisms described below.

3.3.1. Contact Selection Procedure.

(1) A node, ***s***, sends a Contact Selection (*CS*) message through each of its edge node, one at a time.

(2) An edge node receiving a *CS* forwards it to a randomly chosen neighbor (*X*).

(3) A node receiving a *CS* decides whether or not to be a contact for ***s***. This decision is made using either a probabilistic method (*PM*) or edge method (*EM*). These methods are described later in this section.

(4) After using either procedure *PM* or *EM* for deciding whether to be a contact, if the node receiving a *CS* does not choose to be the contact, it forwards the *CS* to one of its randomly chosen neighbor (excluding the one from which the *CS* was received).

(5) The *CS* traverses in a depth-first manner until a contact is chosen or it reaches a node at a distance r hops from ***s***. If a contact is still not chosen (due to overlap), *CS* backtracks to the previous node, which forwards it to another randomly chosen neighbor.

(6) When a contact is selected, the path to the contact is returned and stored at ***s***.

3.3.2. Contact Selection Methods. We introduce and compare two different methods for contact selection: ***(a)*** the probabilistic method (*PM*), and ***(b)*** the edge method (*EM*).

(a) *Probabilistic Method (PM):* Contacts help to increase a node's view (reachability) of the network beyond its own vicinity. To achieve maximum increase in reachability, the vicinities of any node, ***s***, and its contacts should be disjoint/separated, i.e., there should be no overlap between the vicinity of the ***s*** and the vicinity of any of its contacts. The vicinities of different contacts of the same node should also be non-overlapping, to ensure maximum increase in reachability. To achieve this, the *CS* contains the following information: (*i*) ID of node ***s***, (*ii*) a list of already chosen contacts of ***s*** (*Contact_List*; typically small of ~5 IDs*)*, and (*iii)* the hop count, *d*.

This information is used as follows. When a node *X* receives a *CS*, it first checks if ***s*** lies within its vicinity. This check is easily performed since each node has complete knowledge of its vicinity. So a node knows the IDs of all the other nodes in its vicinity. *X* also checks if its vicinity contains any of the node IDs contained in the *Contact_List.*

If neither ***s*** nor any of its already selected contacts lie in the vicinity of *X*, *X* probabilistically chooses itself as the contact. This probability (*P*) of choosing to be a contact is defined as follows:

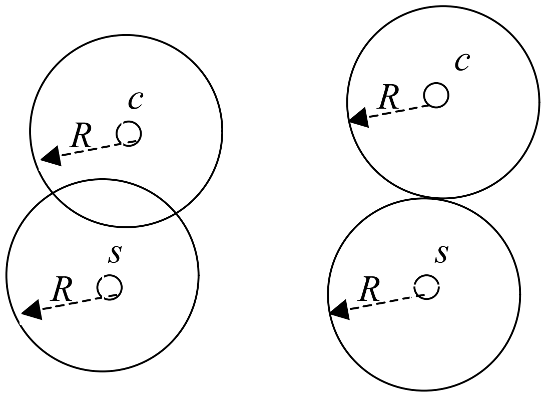
$$P = (d - R)/(r - R) \qquad \text{-- (1)}$$

Here, d is the distance of X from ***s***. d is included in the *CS* as hop count. From the above equation, when $d = R$, $P = 0$. When $d = r$, $P = 1$. This aims to select the contacts between R and r hops from ***s***. This has been formulated to provide maximum increase in reachablility with the addition of each new contact. However there can still be a case where equation (*1*) does not provide the maximum benefit of adding a contact. This case is shown in Fig.1 where ***c*** is the contact for node ***s***.

In this figure although the distance between ***s*** and its contact, ***c***, is greater than R hops, there is still an overlap between the two vicinities. Such a situation will arise whenever a node within R hops from the edge node becomes the contact. To prevent this case, equation (*1*) is modified to:

$$P = (d - 2R)/(r - 2R) \qquad \text{--(2)}$$

In this equation $P=0$ when $d=2R$ and $P=1$ when $d=r$, i.e., contacts are chosen between $2R$ and r hops from the source.



(a) overlap (b) non-overlap

Fig. 1 Overlap in (a) due to the use of *P*

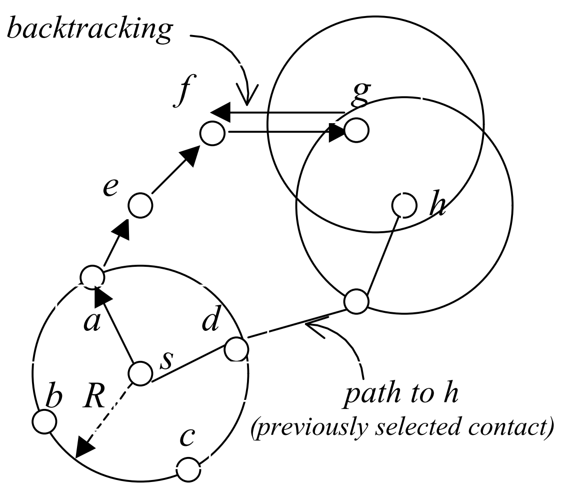


Fig. 2 Selecting contacts in the *P* method

Fig. 2 explains the contact selection procedure with an example. In the above figure *R*=3 and *r*=6. Nodes *a, b, c, d* are the edge nodes for node ***s***. Node ***s*** sends a Contact Selection (*CS*) message through *a*. Node *a* randomly chooses one of its neighbors *e* and forwards *CS* to that node. Node *e* calculates the probability *P* according to equation *(1)*. If the probability of being the contact failed at *e*, it forwards *CS* to one of its neighbors, *f* (chosen randomly). Node *f* again forwards *CS* to *g*. As *g* is at *r* hops from ***s***, the probability *P* at *g* is 1. However, *g* still cannot become a contact for ***s*** as there already exists another contact *h* (which was selected through a previous selection via another edge node *d*) in the vicinity of *g*. So *g* returns *CS* to *f* (*backtracking*). Node *f* then forwards *CS* to another neighbor.

(b) *Edge Method (EM):* Even with equation (*2*) the probabilistic method can result in a situation where there is some overlap between the contact and ***s***'s vicinities. This is possible due to the fact that the nodes do not have any sense of direction. Therefore, it is possible that a contact may be selected at a location where the *CS* has traveled more than *2R* hops in one direction, but the contact may in fact be closer than *2R* hops from the source.

More seriously, the probabilistic method for contact selection can be expensive in terms of the amount of traffic generated by the *CS*. This is due to the extra traffic generated due to backtracking, and lost opportunities when the probability fails, even when there is no overlap. To reduce the possibility of such a situation, probability equations (*1*) and (*2*) are eliminated. The probability equations were formulated to have a higher possibility of choosing the contact that lies either between *R* and *r* hops (equation *1*) or between *2R* and *r* hops (equation *2*). To maintain this non-overlapping property without the probability equations, the contact selection procedure is modified as follows.

The list of all edge nodes (*Edge_List*) of ***s*** is added to the *CS*. Also, the query and source IDs are included to prevent looping. On receiving a *CS*, apart from checking for overlap with ***s***'s vicinity and the vicinities of all the already selected contacts (*Contact_List*), the receiving node also checks for overlap with the vicinities of any of the nodes on the *Edge_List* as well[1]. Since any node that lies at a distance of less than *R* hops from the edge will have an overlapping vicinity with the ***s***'s vicinity, checking for non-overlap with the edges ensures that a contact is chosen between *2R* and *r* hops only. This eliminates the possibility of an overlap due to the lack of direction. Fig. 3 and Fig. 4 show a comparison of the probabilistic and edge methods. As can be seen from Fig. 3 the reachability saturates in both *PM* and *EM.* However the saturation occurs much earlier in the case of probabilistic method. Also as compared to *EM*, the reachability achieved is less for *PM*, for the same values of *NoC*. Fig. 4 shows the backtracking overhead for *PM* and *EM*. Due to the reasons explained earlier, overhead is significantly reduced for *EM*.

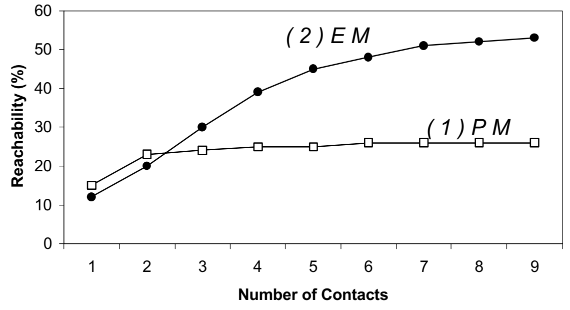


Fig. 3. Reachability for (1) *PM* and (2) *EM*

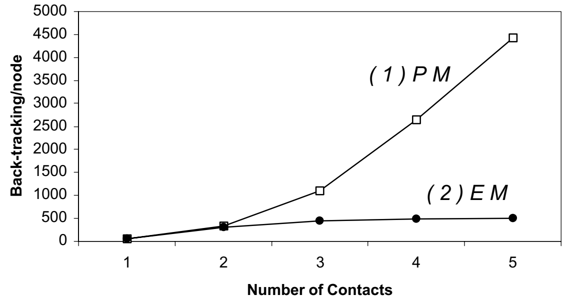


Fig. 4. Overhead for (1) *PM* and (2) *EM*
(Shown: 500 nodes, 710mx710m, tx range=50m, *R=3, r=20, D=1*)

3.3.3. Contact Maintenance Procedure. Node mobility may cause the path to a contact to change. Therefore a node needs to keep track of its contacts and their paths. This is done using periodic polling of the contacts as follows.

(1) Each node periodically sends *validation* messages to each of its contacts. These validation messages contain the path from a node, ***s***, to the contact.

(2) Each node on the path that receives the validation message checks if the next hop in the path is a directly connected neighbor. If so, it forwards the validation message to the next hop node. If the next hop is missing, the node tries to salvage the path using *local recovery*, discussed later in this subsection.

[1] This may be achieved in a communication-efficient manner by using bloom filters[18] to represent membership in the edge-list and/or the vicinity.

(3) If a path cannot be salvaged using local recovery, the contact is considered to be lost.
(4) If the path to a contact is validated but the number of hops to the contact does not lie between *2R* and *r*, the contact is considered to be lost.
(5) After validating all the contacts, if the number of contacts left is less than the specified *NoC*, new contacts are selected.

The *local recovery* mechanism is performed as follows. Assuming reasonable values of node velocities and validation frequency, there is a high probability that if a node has moved out of a contact path, it is still within the vicinity of the previous hop in the path. Even in the case when a node is completely lost (because it has moved out of the vicinity of the previous hop), some other node further down the path might have moved into the vicinity of the previous node. Local recovery takes advantage of these cases to recover from changes in the path when possible, without having to initiate new searches from ***s***. Thus local recovery provides an efficient mechanism for validating contacts and recovering from changes in the contact paths. If the next hop on the path is missing, the node that received the validation message looks for the next hop in its vicinity routing table. If the next hop is in the vicinity, the path is updated and the validation message is forwarded to the next hop. If the lookup for the next hop fails, lookup is done for the subsequent nodes along the path.

3.3.4. Query Mechanism. When a source node, ***s***, (potentially any node), needs to reach a destination or target resource, *T*, it first checks its vicinity routing table to see if *T* exists in its own vicinity. If *T* is not found in the vicinity, ***s*** sends a Destination Search Query (*DSQ*) to its contacts. The *DSQ* contains the following information: (1) depth of search (*D*), and (2) target resource ID (*T*). Upon receiving a *DSQ,* each contact checks the value of *D*. If *D* is equal to 1, the contact performs a lookup for *T* in its own vicinity. If *T* exists, then the path to *T* is returned to ***s***, and the query is considered successful. Otherwise, if *D>1*, the contact receiving the *DSQ* decrements *D* by 1 and forwards the *DSQ* to each of its contacts, one at a time. In this way the *DSQ* travels through multiple levels of contacts until *D* reduces to 1.

The source ***s***, first sends a *DSQ* with *D=1* to its contacts, one at a time. So only the first level contacts are queried with this *DSQ*. After querying all its contacts if the source does not receive a path to the target within a specified time, it creates a new *DSQ* with *D=2* and sends it again to its contacts, one at a time. Each contact observes that *D=2* and recognizes that this query is not meant for itself. So it reduces the value of *D* in the *DSQ* by 1 and forwards it to its contacts one at a time. These contacts serve as second level contacts for the source. Upon receiving the *DSQ*, a second level contact observes that *D=1* and it does a lookup for the target *T* in its own vicinity and returns the path to *T*, if found. In this way the value of *D* is used to query multiple levels of contacts in a manner similar to the expanding ring search. However, querying in *CARD* is much more efficient than the expanding ring search as the queries are not flooded at with different TTLs but are directed to indiviual nodes (the contacts). Contacts leverage knowledge of their vicinity topology (gained through the proactive scheme operating within the vicinity) to provide an efficient querying mechanism.

4. Evaluation and Analysis

In this section we present detailed simulation based evaluation and analysis of our architecture. NS-2 [14] along with our *CARD* extensions and other utilities were used to generate various scenarios of ad hoc networks. Mobility model for these simulations was random way-point model. Our simulations so far did not consider MAC-layer issues. In random way point model a node is assigned a random velocity from [0,Vmax] and assigned a destination location randomly. One the node reaches its destination it is assigned a random velocity and random destination again, so on.

First we try to understand the effect of various parameters such as vicinity radius (*R*), maximum contact distance (*r*), the number of contacts (*NoC*), depth of search (*D*) and network size (*N*) on reachability and overhead. Reachability here is defined as the percentage of nodes that are reachable from a source node. For overhead we consider the number of control messages. We consider overhead due to contact selection and contact maintenance. Having developed an understanding of the various parameters in our architecture, we then compare it with other schemes such as flooding and bordercasting in terms of query overhead and query success rate.

No.	Nodes	Area	Tx Range	No. of Links	Node Degree	Network Diameter	Av. Hops
1	250	500*500	50	837	6.75	23	9.378
2	250	710*710	50	632	5.223	25	9.614
3	250	1000*1000	50	284	2.57	13	3.76
4	500	710*710	30	702	4.32	20	5.8744
5	500	710*710	50	1854	7.416	29	11.641
6	500	710*710	70	3564	14.184	17	7.06
7	1000	710*710	50	8019	16.038	24	8.75
8	1000	1000*1000	50	4062	8.156	37	14.33

Table1 Description of various scenarios used for simulating *CARD*

Table 1 shows the scenarios used in our simulations. These scenarios vary in number of nodes, network size, and propagation range. The variation is considered to capture the effect of these factors on *CARD*. As shown in Fig. 3 and Fig. 4, the edge method outperforms the probabilistic method. (We obtained similar results for other scenarios.) Therefore, we present only the results for the edge method.

4.1. Reachability Analysis

Reachability Analysis was conducted to understand how contacts help in increasing the view of the network. Here we present results for a topology of 500 nodes spread over area of 710m by 710m. The details can be seen from Table 1, scenario number 5. Similar results were observed for other scenarios.

4.1.1. Varying Vicinity Size (*R*). Fig. 5 shows the effect of increasing the vicinity size (*R*) on reachability. As *R* increases, the reachability distribution shifts to the right; i.e., more nodes achieve higher percentage of reachability. This increase in reachability with the increase in *R* is due to increase in the number of nodes within the vicinity. As the value *2R* approaches the maximum contact distance (*r*), the region available for contact selection (between *2R* and *r*) is reduced. This results in less number of contacts being chosen. In Fig 5, when *R=7*, contacts can only be selected between *2R=14* and *r=16* hops from the source. This small region for contact selection significantly reduces the number of contact and hence the reachability distribution shifts to the left. At this point most reachability is due to the vicinity of the source.

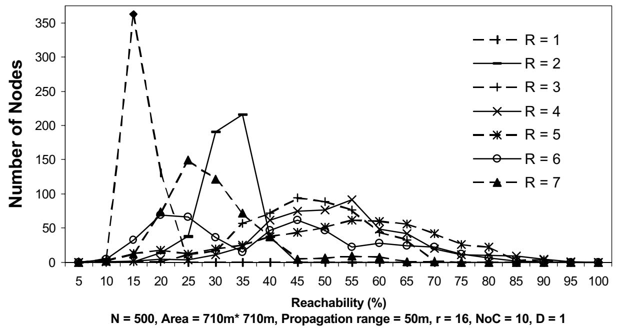


Fig. 5 Effect of Vicinity Radius (*R*) on Reachability

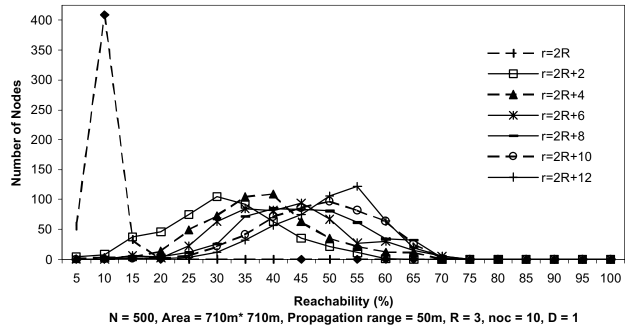


Fig. 6 Effect of Maximum Contact Distance (*r*) on reachability

4.1.2. Varying Maximum Contact Distance (*r*). Fig. 6 shows the effect of increasing *r* on reachability. Since contacts are selected between *2R* and *r* hops from the source, higher values of *r* provide a wider region for contact selection. The mechanisms for contact selection described earlier prevent selection of contacts that have overlapping vicinities. This implies that as *r* increases a larger number of contacts can be selected before their vicinities start to overlap. Therefore reachability increases with increase in *r*. Larger values of *r* also mean that the average contact path length would increase (as more contacts are chosen at larger distances from the source). However, once the vicinities of the contacts and the source become non-overlapping, for *r > (2R +8)*, we see no significant increase in reachability with further increase in *r*.

4.1.3. Varying Number Of Contacts (*NoC*). *NoC* specifies the maximum number of contacts to be selected for each node. The actual number of contacts chosen may be less than this value. This is because of the limited region available for choosing contacts for given *R* and *r*. Once this region has been covered by vicinities of chosen contacts, choosing more contacts in the same region is not possible as their vicinities would overlap with the vicinities of the already chosen contacts. Therefore contact selection mechanism prevents selection of more contacts. This can be seen in Fig. 7, in which the reachability initially increases sharply as more and more contacts are chosen. However, the increase in reachability saturates beyond *NoC=6* as the actual number of contacts chosen saturates due to the effect of overlapping vicinities.

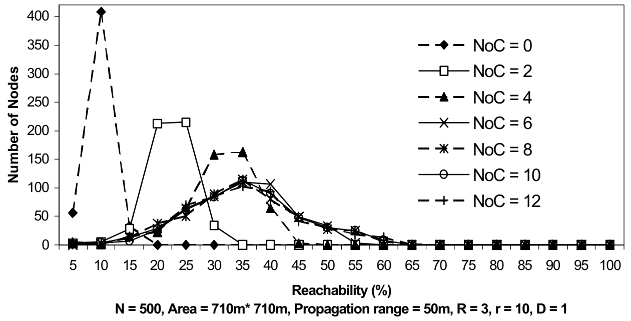


Fig. 7 Effect of Number of Contacts (*NoC*) on Reachability

4.1.4. Varying Depth Of Search (*D*). *D* specifies the levels of contacts that are queried in a breadth first manner. When *D=1*, a source node looking for a resource beyond its vicinity, queries its first level contacts only. When *D=2*, if none of the first level contacts contain the resource in its vicinity, second level contacts (contacts of the first level contacts) are queried through the first level contacts. As can be seen from the Fig 8, reachability increases sharply as the depth of search *D* is increased. Hence, depth of search results in a tree-like structure of contacts, improving the scalability of *CARD*.

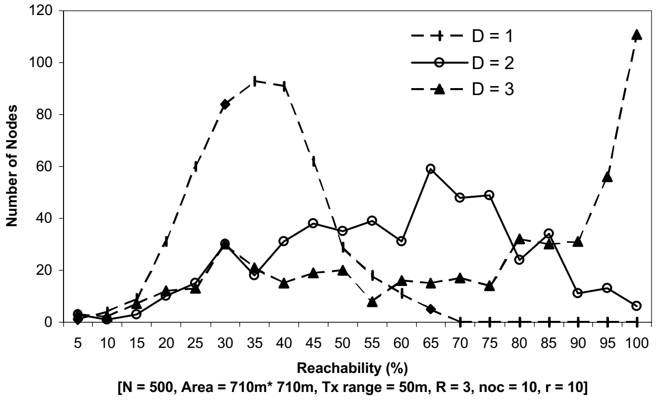


Fig 8. Effect of Depth of Search (*D*) on Reachability

4.1.5. Varying Network Size. Fig. 9 shows a variation of reachability distribution for three different network sizes, *N*. The area of the three networks has been chosen so that the node density is almost same across the three networks. Fig. 9 shows that for any given network (specified by the values of *N* and the area), the values of *R* and *r* can be configured to provide a desirable reachability distribution in which most of the nodes have a high value of reachability.

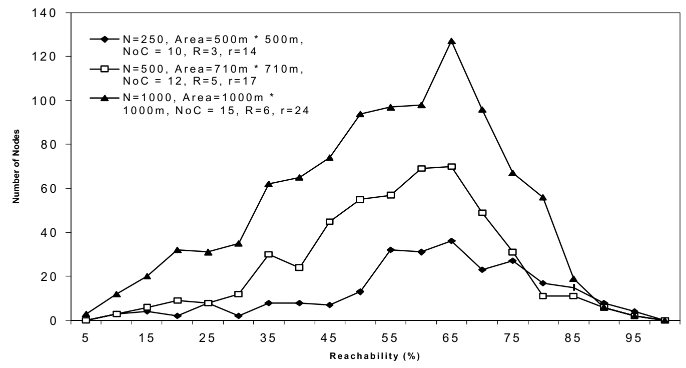


Fig. 9 Reachability for different network sizes

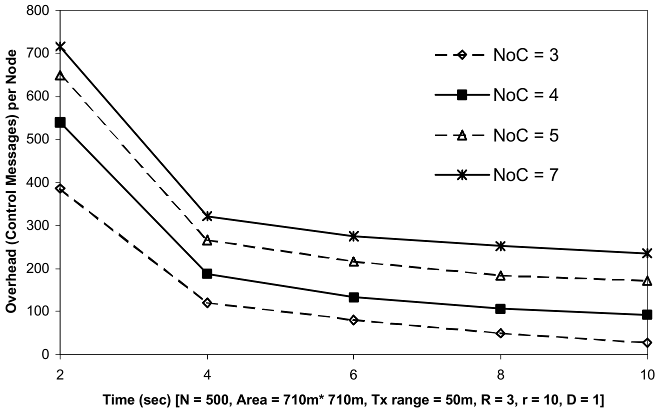


Fig. 10 Effect of Number of Contacts (*NoC*) on Overhead

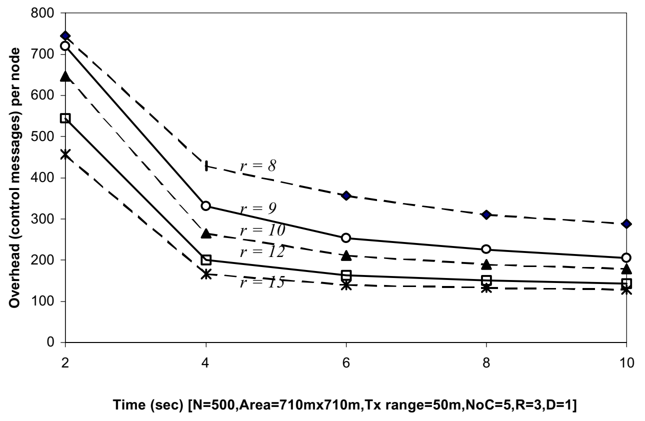


Fig. 11 Effect of Maximum Contact Distance (*r*) on Overhead

4.2. Overhead Analysis

Overhead analysis is done in terms of number of control messages required for contact selection and maintenance. Query overhead is considered in the next section. The overhead considered in this section includes:

1. Contact selection overhead: This is the amount of *CS* traffic generated for selecting new contacts. This includes overhead due to Backtracking as described earlier.
2. Contact maintenance overhead: This is the traffic generated by the contact path validation messages. *Local recovery*, as described earlier, helps in reducing this part of the total overhead.

Results are shown for scenario number 5 in Table 1. Similar results were obtained for other scenarios.

4.2.1. Varying Number Of Contacts (*NoC*). As shown in Fig. 10, as the number of contacts increases the maintenance overhead increases sharply as more nodes are validated.

4.2.2. Varying Maximum Contact Distance (*r*). As *r* increases the number of selected contacts increases. The increase in the number of contacts is due to the availability of a wider area for choosing contacts. Moreover, with higher values of *r*, contacts may lie at greater distances from the source. That is, the contact path length is expected to be higher for larger values of *r*. This suggests that the maintenance overhead should increase with increase in *r*. However, as shown in Fig. 11, the overhead actually decreases with increase in *r*. Fig. 12 explains this decrease in maintenance overhead. Fig.12 shows that as the value of *r* increases the backtracking overhead decreases significantly. Recall that backtracking occurs when a node receiving a *CS* cannot become a contact due to overlap with already existing contacts. As *r* increases, the possibility of this overlap decreases due to availability of a wider area for contact selection. This decrease in back-tracking overhead is significantly more than the increase in overhead due to increased number of contacts and contact path length. Therefore, the total maintenance overhead decreases.

4.2.3. Maintenance Overhead Over Time. Fig. 13 shows the maintenance overhead per node over a 20sec period for Vmax=20m/s. The maintenance overhead decreases steadily with time. However, the number of contacts increases slightly. This suggests that the source nodes find more and more stable contacts. Stable contacts may be defined as those nodes that have low velocity relative to the source node. Therefore, a node moving in the same direction as source node with similar velocity could prove to be a stable contact. Hence, *CARD* leads to source nodes finding more such nodes in the vicinity[2].

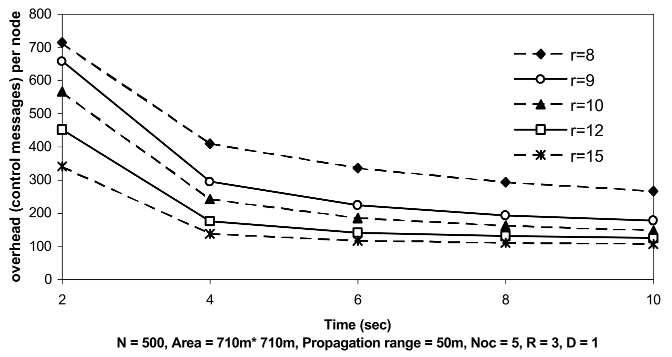


Fig. 12 Effect of max contact distance (*r*) on backtracking overhead

[2] This was observed for the random way-point (RWP) mobility model. We plan to investigate this problem further and expect that different mobility models may have different effects on performance of *CARD*.

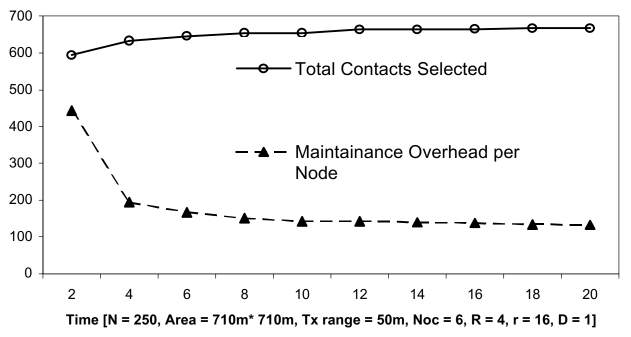


Fig. 13 Variation of overhead with time

4.3. Comparison with Other Approaches

We compare the performance of *CARD* to that of flooding and bordercasting [8], in terms of average query overhead and overall overhead. Simulations were repeated several times with various random seeds to filter out the noise.

Fig. 15 shows the average traffic generated per query for the three protocols. We select random source-destination pairs in the network (the same pairs were used for all the three protocols). The graph shows the average overhead for random queries with different network sizes, for each protocol. The overhead includes transmission as well as reception. Therefore the overhead for flooding is (as expected) about twice the number of links. Bordercasting is implemented as described in [8]. We implemented query detection (*QD1* and *QD2*) and early termination (*ET*) as described in [8] to improve the performance. For *CARD* the values of *R* and *r* used were chosen as the values that gave maximum reachability for that particular network size. This information was obtained from previous results shown under the analysis of *CARD* with respect to various parameters (Fig 9. reachability for different network sizes). Flooding and bordercasting result in 100% success in queries, *CARD* showed a 95% success rate with *D=3. CARD's* success rate can be increased by increasing *D,* or with resource replication. As can be seen from Fig. 15, *CARD* leads to significant savings in communication overhead over the other two approaches. *CARD* incurs, on average, around 5% of the query overhead for flooding, and around 10% of the query overhead of bordercasting.

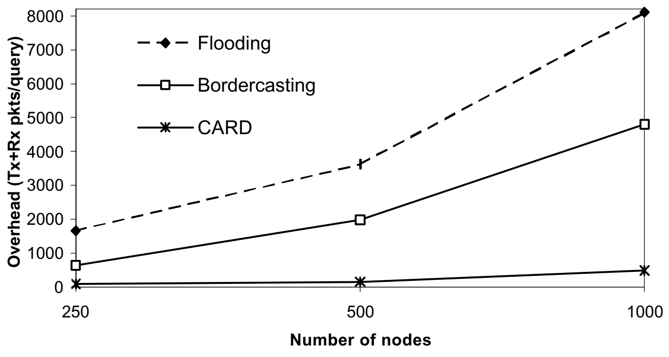
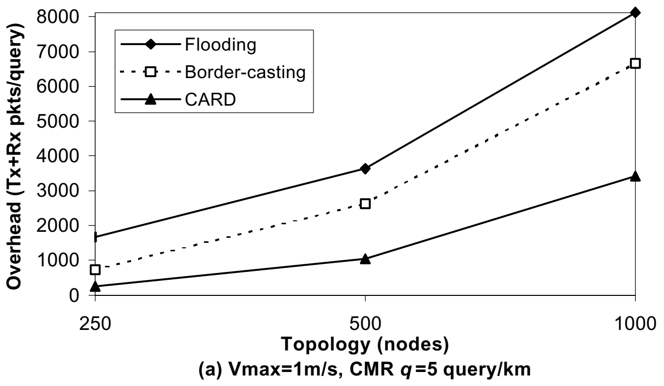


Fig. 15 Query overhead for CARD, flooding and bordercasting



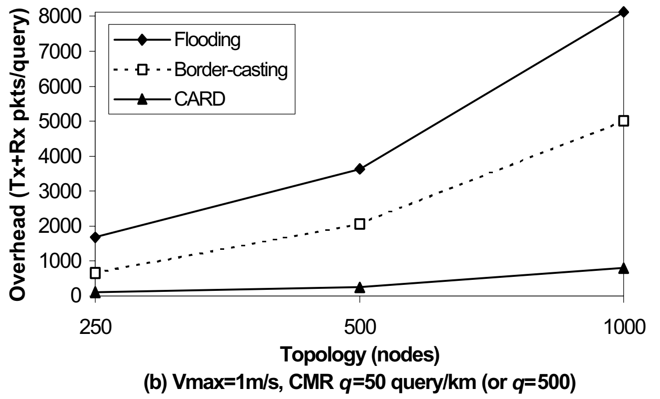
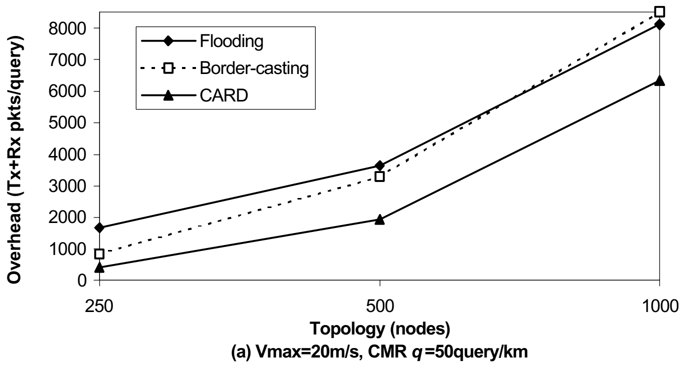


Fig. 16 Total overhead for low mobility and different query rates

What is not shown in Fig. 15, however is the effect of contact and vicinity maintenance. For that we show the following 'total overhead' comparison results. Maintenance overhead (for contacts and vicinity) is a function of mobility and simulation time. Its cost is amortized over the number of queries performed during that period. Hence, we present our results as function of the query rate per mobility per node (i.e., query/sec/(m/s) or query/km); this is referred to as call-to-mobility ratio (or CMR for short) for which we use the symbol *q*. We show results for 20m/s and 1m/s simulations, for various query rates, *q,* for 20 seconds of simulated time. These results take into consideration the contact selection and maintenance overhead, the vicinity establishment and maintenance overhead and the query overhead. As can be seen from Figures 16, 17, the advantage of using contacts becomes clearer for higher query rates, where the cost of maintenance is amortized over a large number of queries. For low mobility, in Figure 16 (a) and (b), the maintenance overhead is low and the advantages of using contacts are the clearest (46-85% savings for low query rates *q=5query/km*, and 86-94% savings for high query rates *q=50 to 500query/km*).

For high mobility, in Figure 17 (a), (b) the savings are less than low mobility scenarios, nonetheless they are still significant for moderate to high query rates (22-75% savings for *q=50*query/km, 79-93% savings for *q=500*query/km). For low query rates and high mobility however, e.g., for 20m/s and *q=5*, CARD and bordercasting perform worse than flooding, where maintenance overhead dominates and only very few queries are triggered (an unlikely scenario in mobile ad hoc networks). For high mobility, large-scale, high query rates (1000 nodes, 20m/s, 500 query/km), we get savings between 79% (vs. bordercasting) and 87% (vs. flooding).



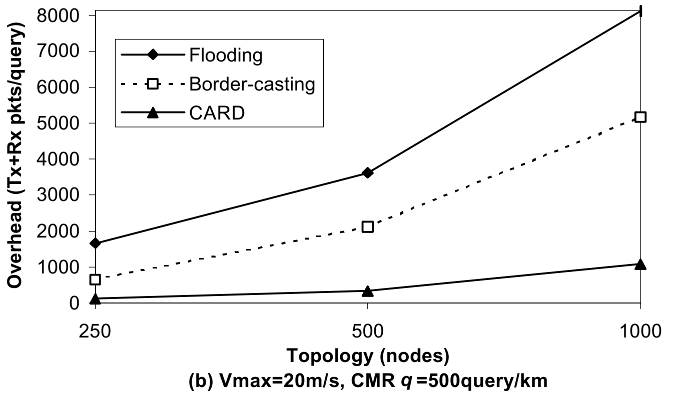


Fig. 17 Total overhead for high mobility and different query rates

5. Conclusions

In this paper we presented the *CARD* architecture for resource discovery in large-scale ad hoc networks. Salient features of our architecture include its ability to operate without requiring any location information or any complex coordination. In our architecture, each node proactively discovers resources within its vicinity. Based on small world concepts, we have introduced the notion of *contacts* to serve as short cuts that increase reachability beyond the vicinity. Two protocols for contact selection were introduced and evaluated: (a) probabilistic method and (b) edge method. The edge method was found to result in more reachability and less overhead during selection due to reduced backtracking, and was thoroughly analyzed over the various dimensions of the parameter space (including *R, r, D, NoC,* and network size). We further compared our approach to flooding and bordercasting. The overall overhead experienced by *CARD* was found to be significantly lower than the other approaches. Overhead savings are function of the query rate, reaching 87% (vs. flooding) and 79% (vs. bordercasting) in communication savings for high query rates during high mobility in large-scale networks; a drastic improvement in performance.

These results show a lot of promise for the contact-based approach and we are encouraged to further investigate this direction. One possible direction is to integrate *CARD* with other routing protocols (e.g., ZRP), where *CARD* may be used as the resource discovery (and transaction routing) protocol. Similarly, we plan to investigate the integration of *CARD* in other data dissemination protocols for sensor networks, such as directed diffusion[19]. Instead of using flooding, *CARD* maybe use for efficient resource discovery. We shall also pursue other heuristics for contact selection mechanisms.

References

[1] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", Comp. Comm. Rev., Oct. 1994, pp.234-244.
[2] Tsu-Wei Chen and Mario Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks" Proc. IEEE ICC'98.
[3] S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks", ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Comm. Networks, Oct. 1996.
[4] David B. Johnson, Davis A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks" Oct 99 IETF Draft.
[5] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, "Ad Hoc On-demand Distance Vector Routing", October 99 IETF Draft.
[6] C.-C. Chiang, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel" Proc. IEEE SICON'97, Apr.1997.
[7] J. Li, J. Jannotti, D. Couto, D. Karger, R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing", Mobicom 2000.
[8] M. Pearlman, Z. Haas, "Determining the optimal configuration for the zone routing protocol", IEEE JSAC, p. 1395-1414, 8, Aug 99.
[9] Z. Haas, M. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", IETF Internet draft for Manet group, June '99.
[10] D. Watts, S. Strogatz, "Collective dynamics of 'small-world' networks",Nature, Vol. 393, June 4, 1998.
[11] D.J.Watts. In Small Worlds, The dynamics of networks between order and randomness. Princeton University Press, 1999.
[12] A. Helmy, "Architectural Framework for Large-Scale Multicast in Mobile Ad Hoc Networks", *IEEE ICC '02.*
[13] A. Helmy, "*Small* Large-Large Scale Wireless Networks: Mobility-Assisted Resource Discovery", TRN Journal, August 2002.
[14] 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,* May 2000
[15] J. Liu, Q. Zhang, W. Zhu, J. Zhang, B. Li, "A Novel Framework for QoS-Aware Resource Discovery in MANets", *IEEE ICC '02.*
[16] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, a. Qayyum et L. Viennot, Optimized Link State Routing Protocol, IEEE INMIC 01.
[17] W. Heinzelman, J. Kulik, and H. Balakrishnan, ``Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *MobiCom '99*, Seattle, WA, August, 1999.
[18] M. Mitzenmacher, "Compressed Bloom Filters", PODC 2001.
[19] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, *MobiCOM 2000.*