

A New Adaptive Distributed Routing Protocol using d-hop Dominating Sets for Mobile Ad Hoc Networks*

Zhengen Shi and Pradip K Srimani

Dept of Computer Science
Clemson University
Clemson, South Carolina 29634
{cshi,srimani}@cs.clemson.edu

Abstract

In this paper, we propose an adaptive self-stabilizing algorithm for producing a d-hop connected d-hop dominating set. In the algorithm, the set is cumulatively built with communication requests between the nodes in the network. The set changes as the network topology changes. It contains redundancy nodes and can be used as a backbone of an ad hoc mobile network.

Keywords: Mobile ad hoc network, routing algorithm, shortest path routing protocol, distributed computing, fault tolerance

1 Introduction

A *mobile ad hoc network* is an interconnection of mobile computing devices (mobile clients) where the link between two neighboring nodes is established via radio propagation; two nodes are neighbors in the network and can communicate directly when they are within transmission range of each other and radio propagation condition in the vicinity of these nodes is adequate. Communication between non-neighboring nodes requires a multi-hop routing protocol [1, 2]. Although mobile ad hoc networks have been mainly used in military applications, they are being increasingly used for civilian applications such as virtual class rooms, wireless local area networks, and law enforcement. It is predicted that in the future such ad hoc networks in collaboration with cellular and overlay networks will provide a ubiquitous computing environment [3]. A connected dominating set in an undirected graph is defined to be a set of vertices such that every vertex not belonging to the set is adjacent to some vertex in the set, and the subgraph induced by the vertices in that

set is connected. Construction of a connected dominating set in an ad hoc network is useful since it is necessary to maintain information about this set (in stead of the entire network) only to be used as a backbone for routing. In this paper, we propose a self-stabilizing distributed algorithm to compute a set of vertices which could serve as the network backbone and which are known as d-hop connected and d-hop dominating. Please see [4] for all the related definitions and properties of those sets. The main feature of our algorithm is that the protocol can adapt to a mobile network when the underlying network has a dynamic topology.

2 Distributed Routing Protocol

Each node in the network has a unique id $1, \dots, n$. When a node transmits a message, the nodes in the *coverage area* of the sender can simultaneously receive the message. If node i is in the coverage area of node j , node i is called a *neighbor* of node j . In such networks, all of the nodes are mobile and so the infrastructure for message routing must be self-stabilizing and adaptive.

Let $G = (V, E)$ be a connected graph with vertex set V and edge set E . Let u and $v \in V(G)$ be two vertices. Let $\delta(u, v)$ be the distance between vertex u and v . Let G_d denote the *d-closure* of G .

Definition 1 A *d-closure* of a graph G has the same vertex set as G . The edge set of G_d is: $\forall u, v \in V(G)$, there is an edge connect u and v in G_d iff $0 < \delta(u, v) \leq d$ in G .

Definition 2 A set $S \subset V$ is a *d-hop dominating set* of G if it is a dominating set of G_d . We say S is *d-hope connected* if it is connected in G_d .

A distributed algorithm was proposed in [5] for constructing a connected dominating set in a connected graph of radius at least two. The paper also introduced the "shortest path property".

*The research reported in this paper is supported by NSF grants # ANI-0073409 and # ANI-0218495.

Definition 3 Let set $S \subset V$ be a connected dominating set. If any two vertices in G can be connected by a shortest path consisting solely of vertices from S , then S has the shortest path property.

Authors in [4] further introduced a d-CDS algorithm. They proved that a d-hop connected d-hop dominating set has shortest path property if the radius of the graph is at least $d+1$.

The above algorithms calculate the whole set of nodes at the beginning and permit only limited perturbation. When the mobile network constantly drastically changes, repeated calculation overhead is incurred.

In this paper, we present an adaptive distributed self-stabilizing algorithm to construct a dynamic set of nodes to be used as routers in the network. The set is dynamic and serves the purpose for mobile network. We show that the protocol stabilizes to produce a d-hop connected d-hop dominating set that covers all the need of communication in the network. We include redundancy factor to further tolerate network mobility.

3 System Model

1. A data link layer protocol at each node P_i maintains the identities of its neighbors [6, 7] in some list $\mathbf{Neighbors}(P_i)$; this data link protocol also supports logical links between neighbors and ensures correct message transmit. The protocol informs the upper layer of any creation or deletion of logical links as the node moves.
2. The topology of the ad hoc network is modeled by an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of logical links between neighboring nodes. We assume for simplicity that all links are bidirectional. Nodes in the network may leave or join the network. We assume such changes are detected and handled by link layer protocol of the neighboring nodes by updating their lists of $\mathbf{Neighbors}(P_i)$. We assume the transient link failures are also handled by the link layer protocol. The network graph is dynamic. Both the node set and the edge set may change over time.
3. The change in topology is arbitrary but satisfies the Liveness property [7] : if there are pending messages for a processor p_i , then processor p_i remains alive and the network topology stays constant long enough to guarantee that p_i receives at least one of the messages and acknowledges the receipt before any further change of the network.

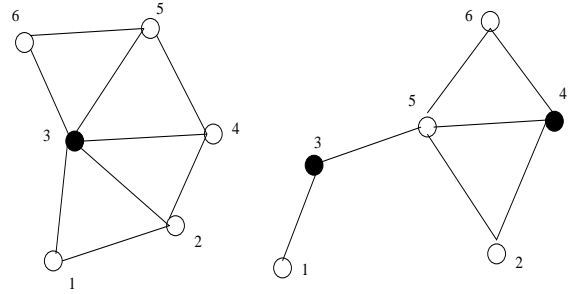


Figure 1: An Example Network

4. Assuming there is a data transmit protocol to send and receive data messages after a set of selected router nodes is identified by this algorithm. Our presentation of the algorithm concentrates on the routing section.

3.1 Proposed Approach

1. The basic idea of the algorithm is to adoptively produce the desired d-hop connected d-hop dominating set. The selected node set does not adjust to network changes by floods of messages and calculations. Instead, the algorithm adjusts to a suitable d-hop connected d-hop dominating set step by step with the communication requests.
2. Assuming at the beginning of the algorithm, there is some random data stored at each node. If a node requests to communicate to another node and there is no route between them in its routing table, it will send a new request. If this requested path is not yet covered by the existing d-hop connected d-hop dominating set, those nodes lying in between the path will modify their local routing table and some of them will change their status as selected. Thus, after the request the pair of nodes is covered by the current set until the network changes to invalidate the route.

We show an example that the d-hop connected d-hop dominating set changes as the network changes in Figure 1. At the beginning, the left network has one selected node 3. Any node in the network may communicate through node 3 with another node that is not immediately in its neighborhood. If the other node is in the neighborhood, there is no need to seek a route. The changed network at a later time is shown on the right side. Node 4 is selected in the new network because node 3 no longer covers all the shortest paths of the pairs of nodes.

4 Proposed protocol

4.1 Data Structure

We assume that there is a maximum of n nodes over time.

1. $DCDS[1 : n]$ is a Boolean array. The value $DCDS_i[j]$ is **true** if node j is a member of the router node set from node i 's knowledge.
2. $PATH[1 : n]$ is an array of node IDs. The first and last entries are the unique IDs of the starting and ending nodes. The entries in between represent the nodes in the path.
3. $R[1 : n]$ is an array of two-tuples of source node and destination node. This array indicates the paths that are covered by the sending node without the detailed *path*. This array reflects and is based on the $ROUT[1 : n]$ table of the node.
4. $ROUT[1 : n]$ is a routing table. Each entry of the array is a three-tuple of a path length, the shortest *PATH* from a source node to a destination node and a list of router set nodes through which the network may route the path. We denote the list CNT which is a list of node IDs.
5. $Neighbors[1 : n]$ is a Boolean array. The value $Neighbors_i[j]$ is true if node j is within one hop communication distance from node i in the mobile ad hoc network. This array is maintained by a data link layer protocol.

4.2 Message Types

This protocol involves only the routing procedure of the ad hoc mobile network. Thus the message type here refers to those related control messages.

(1) $INIT(i, j, path)$ The message indicates that the sender node i requests a new rout to the destination node j . The path parameter records the path between node i and the receiving node. For example, when node i sends the message to a neighbor k , the parameters are $INIT(i, j, \{i, k\})$.

(2) $ACK(i, j, path)$ An $INIT$ message arrives at the destination node, the destination node responses with this message. The definitions of the parameters are the same as in the $INIT$ message. The *path* is ended with i and j .

(3) $FAIL(i, j, path)$ When a message cannot be transmitted following the supplied path, the node sends this message. The definitions of the parameters are the same as in the $INIT$ message. The *path* is ended with i and j .

(4) $ANNOUNCE(i, R[1 : n])$ This message broadcasts the routing table of a node. This is used to redefine the router node set in the network.

(5) $ANNOUNCE(i, DCDS[1 : n])$ This message broadcasts the modified router node set information.

4.3 System Primitives

The following primitives are used in the pseudo-code given in the next section. They are used to simplify the presentation of the algorithm.

1. **send**(*source, destination*)

There are two types of send primitives in the presentation, unicast and broadcast. We assume there is some underlying mechanism to broadcast messages throughout the network. The unicasting sends a message to another node either by an underlying unicast protocol or recursively by this routing algorithm.

2. **insert**(*entry, array*) This operation insert the *entry* into the *array*.
3. **delete**(*entry, array*) This operation delete all occurrences of the *entry* from the *array*.
4. **update**(*path, array*) The update operation removes all entries with the same sender and destination nodes as *path* from the *array*. Then the *path* is inserted.

4.4 Algorithm Description

The procedure Send-To-Neighbors assumes the functionality of sending a message to the neighbors. For convenience of explanation, we use the following notations for parameters. The sender-node and dest-node are IDs of these nodes. The parameter *type* is an integer uniquely identifying one of the five message types. The parameter *path* is a *PATH* data structure. The parameter *d* is the path length parameter for the d-hope connected d-hop dominating set. Assuming the ID of this node is i .

Procedure Send-To-Neighbors(sender-node, dest-node, *type, path, d*)

begin

if *type* is *init* **and** dest-node \notin *path*

for $k = 1$ to n

do

if $Neighbors[k] = \text{true}$ **and** $k \notin path$

 send $INIT(\text{sender-node}, \text{dest-node}, path$ appended with $k)$ to k ;

return;

done

if $path[0] = \text{sender-node}$

then $x = path[i - 1], y = path[i + 1]$;

else $x = path[i + 1], y = path[i - 1]$;

if *type* = *ack* **and** $Neighbors[x] = \text{true}$

then send $ACK(\text{sender-node}, \text{dest-node}, path)$ to x ;

```

else if  $type = INIT$  and  $Neighbors[y] = true$ 
  then send INIT(sender-node, dest-node,  $path$ ) to  $y$ ;
else
  send FAIL(sender-node, dest-node,  $path$ ) to  $x$ ;
end

```

The procedure *route* attempts to route a message based on local routing table *ROUT*. Assuming this router node is *i*. The definitions of the parameters are the same as the previous procedure.

Procedure route(sender-node, dest-node, $type$, $path$, d)

```

begin
  if  $\forall route \in ROUT$ , sender- node  $\notin route$  or dest-
  node  $\notin route$ 
  or  $type = FAIL$ 
    then delete( $path, ROUT$ );
    send ANNOUNCE( $i, ROUT_i[1 : ]$ ) message;
    call Send-To-Neighbors(sender-node, dest-node,
   $type, path, d$ );
  return;
   $path$  is the route for sender-node and dest-node in
   $ROUT$ ;
  if  $type = INIT$ 
    then call Send-To-Neighbors(sender- node, dest-
  node,  $init, path, d$ );
  if  $type = ACK$ 
    then send ACK(sender-node, dest-node,  $path$ ) to
  sender-node;
end

```

The procedure *Insert-Info* adds the new information into the $ROUT[1 : n]$ table, and prunes longer paths. Assuming the procedure is running on node *i*.

Procedure Insert-Info($path$)

```

begin
  if  $path \notin ROUT$  or length of  $path \leq$  length of
   $route \in ROUT$ 
    then insert( $path, ROUT$ ), delete( $route, ROUT$ );
     $x = path[0]$ ,  $y = path[length - 1]$ ;
     $\forall$  entries  $e \in ROUT[1 : ]$ ,  $e[0] = x$  or  $e[length - 1] =$ 
   $y$ 
    if length of  $path + e \leq$  length
    then insert or update( $path, ROUT$ );
  if  $|pathlength \leq d| \neq$  previous cardinality
  then  $\forall path \in ROUT$ 
     $R =$  new Array;
    insert( $(path[0], path[length - 1]), R$ );
    send ANNOUNCE( $i, R[1 : ]$ ) message;
end

```

The procedure *Is-Privileged* adjusts local routing table and decides whether the node is a member of the router node set. The parameter *k* identifies the sender node of the $R[1 : n]$ array which is the second parameter of the function. *d* is the parameter for d-hop connected d-hop dominating set. *m* is a redundancy factor. Each path in

the network is covered by no more than *m* nodes from the d-hop dominating set.

Procedure Is-Privileged($k, R[1 : n], d, m$) : Boolean

```

begin
   $\forall tuple \in R_k[1 : ]$ 
   $\forall path \in ROUT_i[1 : ]$ 
    if  $tuple$  equivalents to  $path$ 
      then insert( $k, CNT$  of  $path$ )
      if  $k \in CNT$  of  $path$  and the corresponding
   $tuple \notin R_k$ 
        then delete( $k, CNT$  of  $path$ )
      if  $DCDS_i[i] = false$  and  $\exists entry \in ROUT_i$ , where
  length of  $CNT < m$ 
        then  $DCDS_i[i] = true$ ;
      else if  $DCDS_i[i] = true$  and  $\forall entry \in ROUT_i$ ,
  length of  $CNT \geq m$ 
        then  $DCDS_i[i] = false$ ;
      else return  $DCDS_i[i]$ ;
end

```

A node *i* in the network runs the following procedure for routing control periodically, assuming the messages are stored in incoming message buffer.

Procedure dCDS-daemon(d, m)

```

begin
  if receive an INIT/ACK/FAIL ( $j, k, path$ ) message
    if message type is ACK and this node is  $k$ 
      then return to the calling send( $k, j$ ) Procedure;
    if message type is FAIL and this node is  $j$ 
      then return ;
    if  $DCDS_i[i] = true$  and  $k \in path$ 
      then call route( $j, k, type, path, d$ );
    else if message type is INIT and this node is  $k$ 
      then call Send-To-Neighbors( $k, j, ack, path, d$ );
    else
      call Insert-Info( $path$ );
      call Send-To-Neighbors( $j, k, type, path, d$ );
  else if receive an ANNOUNCE( $k, R[1 : ]$ ) message
    boolean  $tmp = DCDS_i[i]$ ;
    Is-Privileged ( $k, R[1 : ], d, m$ );
    if  $DCDS_i[i] \neq tmp$ 
      then Send ANNOUNCE( $i, DCDS[1 : ]$ );
  else if receive an ANNOUNCE( $i, DCDS_i[1 : n]$ )
    then for  $index = 1$  to  $n$ 
      do
        if  $DCDS_i[index] \neq DCDS_i[index]$ 
          then  $DCDS_i[index] = DCDS_i[index]$  ;
      done
end

```

If a node *i* intends to start a link with node *j*, it first search its *ROUT* table. If the entry is not valid, it runs the following procedure. The parameters *i* and *j* denote

the IDs of the sender and destination nodes.

```

Procedure dCDS-send( $i, j, d$ )
begin
  for  $index = 1$  to  $n$ 
    do
      if  $DCDS_i[index] = \text{true}$ 
        then send INIT( $i, j, \{i, index\}$ ) to node  $index$ .
      done
    while (true)
      wait( $t_a$ );
      if received ACK( $j, i, -$ )
        then proceeding transfer message;
        return;
      else if received ANNOUNCE( $k, DCDS[1 :]$ )
        then for  $index = 1$  to  $n$ 
          do
            if  $DCDS_i[index] = \text{false}$  and
               $DCDS_k[index] = \text{true}$ 
              then  $DCDS_i[index] = DCDS_k[index]$ ;
              send INIT( $i, j, \{i, index\}$ ) to node  $index$ .
            done
          else call Send-To-Neighbors( $i, j, init, i, d$ );
        end

```

5 Correctness Proof

Lemma 1 *If the set of the selected nodes does not cover a path from node x to y , then the selected node set covers it after a request from node x to y .*

Proof : If there is no known route from a node x to another node y , then the node x has to incur extra overhead to find the route. This may be resulted from the mobility property of the network. A previously known path is invalidated. When the node x requests a route to node y , the procedure dCDS-send will send INIT requests to all the nodes that are in the dCDS set. Because none of the selected nodes knows this route, this will trigger a broadcast in the network. After the broadcast message reaches the destination node y , the path is added to the routing tables of the nodes along the path. Some node lying in the path between x and y will call Is-Privileged function and select themselves into the d-hop connected d-hop dominating set. Therefore, the selected nodes now cover the communication pair x and y . \square

Lemma 2 *If the source and destination nodes are in a connected mobile ad hoc network which satisfies the liveness property, the protocol will produce a route with length at most d - the diameter of the network.*

Proof : This follows from the procedure dCDS-send and lemma 1. That is: if the path is covered by the existing selected node set, the communication request can be accomplished without any overhead. If such a path is not or no longer covered by the set, by lemma 1, the communication request will trigger a round of broadcast and the path will be covered afterwards. In both cases, the route is shorter than d because the algorithm will only add those paths that are shorter than d into routing table. \square

Theorem 1 *The protocol creates and maintains a router set with redundancy in the mobile ad hoc network.*

Proof : By lemma 2, the proposed algorithm is self-stabilizing. If a requested route is covered by the existing selected node set, then a communication can be carried out without additional overhead. As the network changes, the previous valid routes may become invalid. However, from lemma 1, once a path for this route is requested, the selected d-hop connected d-hop dominating set will cover it afterwards. Such operations trigger a round of broadcast thus incur extra overhead. However, this overhead is distributed over time to accommodate the changing nature of the underlying mobile network. Conceptually, over a period of time the algorithm adjusts the selected d-hop dominating set to satisfy communication request and reflect the dynamic underlying mobile network. \square

6 Performance Analysis

We show an example of this protocol in Figure 2. Assume in topology (a), the 3CDS set contains some random nodes initially. Notice the parameter d is 3. Without loss of generality, we assume the set is 2, 3. Then node 1 requests to communicate with node 7 and 8, by calling procedure dCDS-send. Node 1 sends INIT messages to both node 2 and 3. The dCDS-daemon programs at node 2 and 3 call Send-To-Neighbors and trigger a round of broadcast. This is because when the nodes in the previous dCDS set receive the request, they do not know such route and will call the procedure route. This will trigger the reconfiguration of the dCDS set in the network. After the destination nodes 7 responds to the broadcast message, the Is-Privileged functions self-elected the new 3CDS set to include node 6. Then the path between node 1 and 7/8 are in the 3CDS set, shown in Figure 2 (a). Thus node 1 communicates with node 8 can proceed without any overhead. After this, node 2 attempts to communicate with node 4 which causes node 5 to be added into the set. Node 3 then communicates with node 7 and 8 without any 3CDS set configuration overhead. This is because such route is stored in node 6.

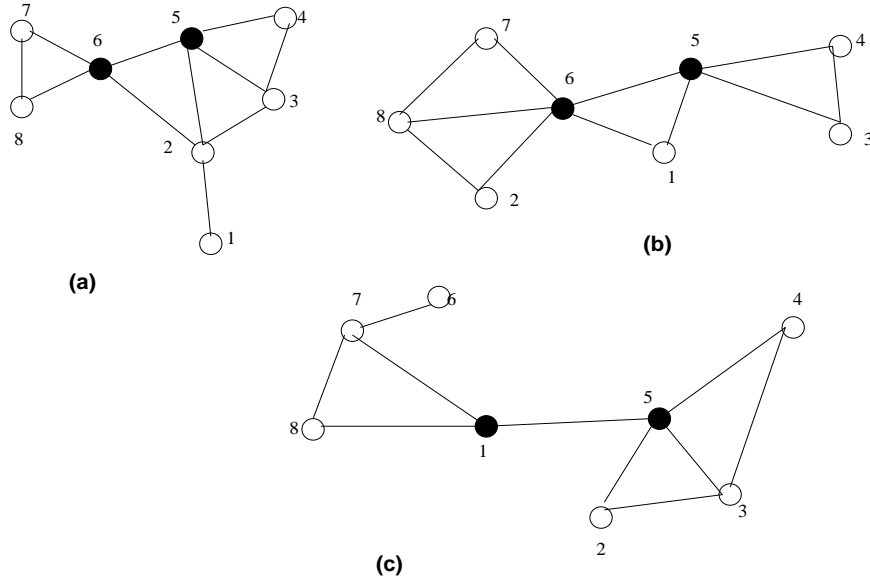


Figure 2: Protocol Execution

The topology then changes to (b) in our example, show in Figure 2. The 3CDS set is still valid in the new network.

After the network changes to (c), notice that the new d parameter is 4 now. The algorithm will realize this new d value over time as the routes in the routing tables in the nodes adjust. Then node 2 requests to communicate with node 4 which can be handled by the existing 4CDS set. Thus no extra overhead is incurred. When node 2 requests to communicate with node 7, node 1 is added to the new 4CDS set as a result.

Obviously, such a scheme avoided the repeated overhead of recalculating the complete dCDS set, which is not always suitable for a constantly changing mobile network. The parameter d is the diameter of the current mobile network.

Definition 4 *Message cost is the total number of messages sent between all pairs of mobile nodes that are within direct communication distance.*

Lemma 3 *If two nodes that wish to communicate are covered by the current dCDS set in the network, then no extra overhead is incurred. If the request of communication is not yet covered by the current dCDS set in the network, the message cost for routing is of order $O(n)$.*

Proof : Let m be the redundancy factor and d be the diameter of the network. In the protocol, each path is covered by no more than m nodes in the dCDS set. The INIT message is passed to n nodes in the network. The corresponding broadcasting messages are sent $O(n)$ times. The broadcasting for resetting the dCDS set has a cost $O(mn)$.

Notice the redundancy factor m is a constant. Thus the total number of message cost for such a routing overhead is no more than $O(n)$. \square

Remark 1 *The resulting d -hop connected d -hop dominating set size is small over time because it covers a subset of nodes that requested links. If the network is fixed and the redundancy factor is set to be 1, then the size of the dCDS set is of the same size of existing protocol producing such set such as GDCDS algorithm [4].*

Remark 2 *The bigger the redundancy factor, the bigger the resulting dCDS set size. However, at the same time, with a bigger dCDS set, it is less likely to incur overhead to reconfigure the dCDS set when the network changes (perturbation). The optimal redundancy factor varies depending on the real situation of the mobile network.*

Remark 3 *The cumulative routing path length is less than parameter d (usually set to the diameter of a network) of the dCDS set. As shown in the paper, this scheme satisfies the d -shortest path property.*

7 Experimental Study

We analyze the efficiency of the proposed adaptive routing algorithm by comparing it to GDCDS algorithm [4]. Let n be the number of nodes in the testing ad hoc mobile network. We use the term *combined cost* to compare the overall efficiency of an algorithm. The combined cost is a weighted sum of cost factors in the process of achieving a router set. These costs include the number of packets sent,

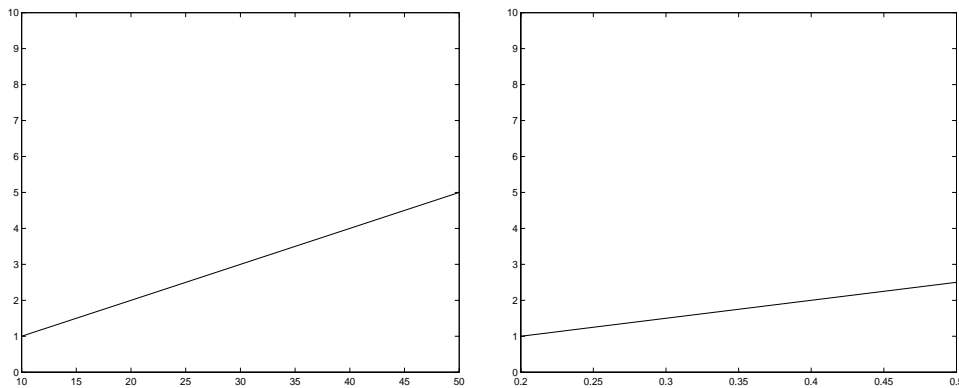


Figure 3: $R - R_1$ plot, $R_2 = 0.10$ and $R - R_2$ plot, $R_1 = 5$

the overall data transmission needed and the overall time overhead incurred.

In GDCDS algorithm, when the network topology changes, several rounds of broadcasts are triggered to achieve a new configuration of the routing nodes. Let $cost_{GDCDS}$ be the combined cost of a complete reconfiguration of the d -hop connected d -hop dominating set. Let p_{GDCDS} be the probability of a communication request that triggers a new round of complete reconfiguration. Let m_{GDCDS} be the mean cost: $m_{GDCDS} = cost_{GDCDS}p_{GDCDS}$. Note that the equation assumes that no extra overhead is incurred if the existing router set can resolve the routing request.

For our approach, let $cost_A$ be the combined cost of an adaptive reconfiguration of the d -hop connected d -hop dominating set. Let p_A be the probability of a communication request that triggers a new round of adaptive reconfiguration. Let m_A be the mean cost: $m_A = cost_A p_A$.

We observe that $p_{GDCDS} \leq p_A$ and $cost_{GDCDS} \geq cost_A$. This is because the completeness of reconfiguration by the GDCDS algorithm. Let rate $R_1 = cost_{GDCDS}/cost_A$. Let rate $R_2 = p_{GDCDS}/p_A$. Rate R_1 is the ratio of the combined cost of the two algorithms. It is greater than 1. Rate R_2 is the ratio of the probability of incurring overhead in the two algorithms. It is less than 1. Let rate $R = m_{GDCDS}/m_A$. Clearly, $R = R_1 R_2$. As an indicator for evaluation, the bigger R , the better the adaptive approach over GDCDS algorithm. The results are shown in figure 3.

8 Conclusion

We have proposed a new distributed algorithm to compute the dominating sets that provide the backbone of the routing in a mobile ad hoc network. We have provided very preliminary results about the performance of the proto-

col; extensive simulation studies are being currently carried out.

References

- [1] D. B. Johnson. Routing in adhoc networks of mobile hosts. In *Proceedings of Workshop on Mobile Computing and Applications*, 1994.
- [2] C. Perkins and P. Bhagwat. Routing over multi-hop wireless network of mobile computers. In T. Imielinski and H. F. Korth, editors, *Mobile Computing*, pages 182–205. Kluwer Academic Publisher, 1996.
- [3] Mark Weiser. Some computer science issues in ubiquitous computing. *Comm. ACM*, 36(7):75–84, July 1993.
- [4] S. Pai M. Q. Rieck and S. Dhar. Distributed routing algorithms for wireless ad hoc networks using d-hop connected d-hop dominating sets. In *Proceedings of the 6th Int. Conf. on High Performance Computing: Asia Pacific Region*, volume 2, pages 443–450, 2003.
- [5] J. Wu and H. Li. A dominating-set-based routing scheme in ad hoc wireless networks. *Special Issue on Wireless Networks in the Telecommunication Systems Journal*, 3(1):63–84, 2003.
- [6] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM/Baltzer Mobile Networks and Applications*, 1(1):183–197, 1996.
- [7] E. Pagani and G. P. Rossi. Providing reliable and fault tolerant broadcast delivery in mobile ad hoc networks. *Mobile Networks and Applications*, 4(1):175–192, 1999.