Scalable Computing: Practice and Experience
Volume 9, Number 1, pp. 21–28. http://www.scpe.org
ISSN 1895-1767

A LINK-CLUSTER ROUTE DISCOVERY PROTOCOL FOR AD HOC NETWORKS

DOINA BEIN*, AJOY K. DATTA†, AND SHASHIREKHA YELLENKI†

Abstract. In MANETS, node mobility induces structural changes for routing. We propose a route discovery algorithm for MANET based on the link-cluster architecture. The algorithm selects the clusterheads and gateway nodes, and then builds routing tables for nodes both inside and outside the cluster. The algorithm attempts to minimize the number of clusterheads and gateway nodes to avoid storing redundant data. For intra-cluster routing, the shortest paths are maintained. For inter-cluster routing, we implement routing on-demand (the shortest paths are maintained only for the nodes that need to send packets). The proposed algorithm adapts to arbitrary movement of nodes, and joining and/or leaving of existent nodes.

Key words. ad hoc networks, clustering, location management, agent mobility

1. Introduction. A mobile ad-hoc network (MANET) is a self-configuring network of mobile hosts connected by wireless links, with an arbitrary topology. The mobility management of such networks is important since a minimal configuration and quick deployment make ad hoc networks suitable for emergency situations like natural or human-induced disasters, military conflicts, emergency medical situations, etc. Beginning as a military application, MANETs had become largely used for personal use, e.g. personal area network (PAN) (for short-range communication of user devices), wireless local area network (WLAN) and in-house digital network (IHDN) (for video and audio data exchange).

The first self-configuring (self-organizing) protocols for a MANET (protocol LCA [1, 2], protocol DEA [15], protocol Layer Net [3]) periodically discard the network topology information and rebuild the network from scratch. Later protocols consider a gradual approach to self-configure a MANET (for example, the protocol SWAN by Scott and Bambos [17]).

The first self-configuring wireless network, proposed by Baker and Ephremides [1, 2], is a two-tier hierarchical model. The nodes, classified as ordinary, clusterheads, and gateways, have the restriction that a node belongs to a single cluster (clusterhead) and it is one hop away from it. Since selecting the minimum number of such clusterheads is NP hard, they proposed a link cluster algorithm (LCA) for categorizing the nodes and a link activation algorithm (LAA) to schedule (activate) the links between nodes. LCA algorithm is a dominating set partitioning of the network based on node ID and works as follows. The node with the highest identity number among a group of nodes without a clusterhead within one hop declares itself as a clusterhead. The other nodes become either gateways (if there are connected to two or more clusterheads) or ordinary nodes.

Variations of the LCA algorithm are to either consider the lowest ID or the highest connected node instead of the highest ID node.

The distributed evolutionary algorithm (DEA) proposed by Post *et al.* [15] is based on a clique partitioning of the network (also an NP hard problem) and is uniform (it is the same for each node in the network). It works as follows. A starter node activates all its neighbors that are part of some clique as itself (so called *clique neighbors*) to begin communication based on a schedule decided by itself. Then these nodes become starter nodes for the rest of the network.

In protocol SWAN proposed by Scott and Bambos [17], new connections are sought during random access periods. After a timeout, the connections that do not respond to a control call are declared unusable.

In spite of the various applications served by the ad-hoc networks, they still have to overcome aspects as the limited transmission range, interference due to its broadcast nature, route changes and packet losses due to the node mobility, battery constraints, and potentially frequent network partitions. A major challenge faced in MANETs is locating the devices for communication, especially with high node mobility and sparse node density. Present solutions provided by the ad hoc routing protocols range from flooding [11] the entire network with route requests, to deploying a separate location management scheme [14] to maintain a device location database. Kawadia et al. [10] had given a general framework to support the implementation of ad-hoc routing protocols in Unix-like operating systems.

*Department of Computer Science, Erik Jonsson School of Engineering, University of Texas at Dallas, 2601 North Floyd Road, Richardson, TX 75083-0688 (siona@utdallas.edu).

†School of Computer Science, University of Nevada, Las Vegas, 4505 Maryland Parkway, Las Vegas, NV 89154-4019 ({datta, rekha}@cs.unlv.edu)

Contributions. We present a protocol for routing in ad hoc networks that adapts fast to frequent node movement, yet requires little or no overhead during periods in which hosts move less frequently. Moreover, the protocol routes packets through a dynamically established and nearly optimal path between two wireless nodes. It also achieves higher reliability—if a node in a cluster fails, the data is still accessible via other cluster nodes.

In a network with link-cluster architecture we propose a protocol that discovers an optimal route for the nodes to communicate. We use the concept of *proactive* protocols to route the packets within the cluster and the concept of *reactive* protocols to route the packets between the clusters. (A combination of proactive and reactive protocols used for routing the packets is called a *hybrid* protocol [16]). When a node leaves a cluster we update the routing tables (location management, [9]).

Outline of the paper. In Section 2 we present the architectural model and the variables used by the algorithm. The cluster-based route discovery algorithm is presented in Section 3, together with a proof of correctness in Section 4. We finish with concluding remarks in Section 5.

2. Preliminaries. Clustering is a scheme designed for large dynamic networks to build a control structure that increases network availability, reduces the delay in responding to changes in network state, and improves data security. Clustering is crucial for scalability as the performance can be improved by adding more nodes to the cluster.

Link-cluster architecture [1, 2, 8] is a network control structure in which nodes are partitioned into clusters that are interconnected. The union of the members of all the clusters covers all the nodes in the network. Nodes are classified into clusterheads, gateways, and ordinary nodes. A *clusterhead* schedules the transmissions and allocates resources within clusters. *Gateways* connect adjacent clusters. An *ordinary node* belongs to a single cluster (has a unique clusterhead). We will consider only *disjoint* clusters. Specifically, a gateway node is a member of exactly one cluster and forms links to members of other clusters.

A non-clusterhead node is within two hops from its clusterhead. Since there are no adjacent clusterheads, the clusterheads form an independent set of nodes.

For example, in Figure 2.1, an ad hoc network is divided into five clusters.

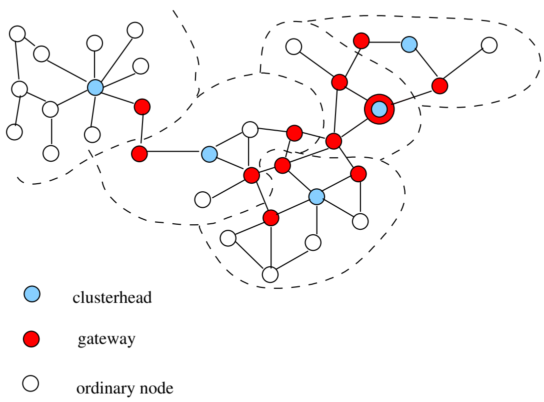

Fig. 2.1. *Ad hoc network divided into 5 clusters*

There are couple of advantages to such an architecture. Scalability is improved since a reduced number of mobile nodes participate in some routing algorithm, hence a low routing-related control overhead. Also, the chance of interference via coordination of data transmissions is lower.

Cluster maintenance schemes are designed to minimize the number of changes in the set of existing clusters. They do not re-cluster after every movement, but instead make small adjustment to the cluster membership as necessary, as in only when the most highly connected node in a cluster moves. All the gateway nodes and the clusterhead node which are present in the cluster region C_u of the clusterhead node u act as location servers for all the nodes in the cluster region C_u. When a node moves across two clusterhead regions, the node updates its home region C_u of the movement by a location update or by sending a leave message. A source node x from outside the cluster that needs to communicate with another node y in the cluster region C can use the clusterhead and gateway tables to identify the location of y and send a location query packet towards region C to obtain the current location of y. The first location server to receive the query for u responds with the current location of y to which data packets are routed.

3. Cluster-Based Route Discovery Algorithm. Our algorithm is based on the fact that every mobile node has a unique identifier [7] that differentiates every single node in the network from others. The node with the highest identifier within a geographical region becomes the clusterhead [5].

The change in clusterhead status occurs only if two clusterheads move within the range of each other—in that case one of them relinquishes its role as clusterhead—or if an ordinary node moves out of range of all other nodes—in that case it becomes the clusterhead of its own cluster.

The algorithm uses a variable N_i^1 representing the one-hop neighborhood set of node i and a variable N_i^2 representing the two-hop neighborhood set of node i. These two sets are maintained by a local topology maintenance protocol that adjusts its value in case of topological changes in the network due to failures of nodes or links.

Node i has a unique ID, $ID.i$. Variable $n.i$ is used to identify the neighbor of the shortest path to the clusterhead (for a non-clusterhead node).

Every node in a network has a sequence table that keeps track of the messages already received by the node and makes the routing messages loop-free [4, 13]. Only gateways and clusterheads maintain the routing tables used for routing [6]. A clusterhead has another table that is used to route messages outside the cluster. This table has entries of all the destination and boundary gateway pairs. The gateway tables contain all the entries of the destination-clusterhead pairs of all the clusters they connect to. The routing table is updated whenever a new clusterhead is elected or some changes occur related to paths in the routing table. The ordinary nodes have only a variable indicating the neighbor on the shortest path towards their clusterhead.

The proposed protocol consists of three main modules: Clusterhead Election, Gateway Election, and Route Discovery.

3.1. Clusterhead Selection Module. The clusterhead selection protocol must satisfy two conditions.
Condition 1: Each non-clusterhead is within two hops from its clusterhead.
Condition 2: There are no adjacent clusterheads [12].

A node can act as a clusterhead as well as a gateway at the same time.

A clusterhead will periodically do the following:

- It checks the consistency of each variable.
- It broadcasts CL_ANN messages to all its neighbors within its two hop distance.
- It checks whether any other clusterhead is within its transmission range and if it finds one whose ID is bigger than itself, then it gives up its clusterhead status by broadcasting CL_REJ messages.

Algorithm 3.1 Clusterhead Selection Module

Actions of some node i

E.01 Timeout $\longrightarrow$
    if i is a clusterhead then sends CL_ANN to immediate neighbors
    else if i finds itself with faulty values or is "orphan" (has no CH),
        then elects itself as a clusterhead
        else i sends CL_REQ message to $n.i$

E.02 Receive CL_ANN from node nb $\longrightarrow$
    if i is an ordinary node and either the sender was its own CH or
        i has no current clusterhead, then updates its variable with respect
        to the sender as a clusterhead and forwards the message
    else if i is a clusterhead and the sender is a clusterhead with
        a lower ID and within 2 hops, then i accepts the sender as
        a clusterhead and sends CL_REJ to all neighbors

E.03 Receive CL_REJ from node nb $\longrightarrow$
    if i is a clusterhead, then drops the message
    else if the sender is i's CH, then mark itself as "orphan" and forward it

---

Algorithm 3.2 Clusterhead Selection Module (continued)

E.04 Receive CL_REQ from node nb $\longrightarrow$
if i is a clusterhead then
if the sender belongs to its cluster, then send CL_ANN to sender
else send CL_CHG to the sender
else if the message is addressed to i then reply with CL_CHG
else if the addressee is within two hops, then forward it to addressee
else drop the message

E.05 Receive CL_CHG from node nb $\longrightarrow$
if the message is regarding is i's clusterhead, then i updates
its variables accordingly and forwards the message to neighbors

E.06 Receive CL_ACCEPT from node nb $\longrightarrow$
if i is a clusterhead and the addressee, then updates its routing table
and sends the updated message to the bordering gateway nodes
else if the message is not addressed to the node, then it forwards
the message to its neighbors if the hop count < 2,
but drops the message if the hop count ≥ 2

E.07 Receive *leave* from node nb $\longrightarrow$
if i is a clusterhead and the addressee, then updates its routing table
and sends the updated message to the bordering gateway nodes
else if the message is not addressed to the node, then it forwards
the message to its neighbors if the hop count < 2,
but drops the message if the hop count ≥ 2

E.08 Receive *ctable_copy* from node nb $\longrightarrow$
if i is a clusterhead and the message is addressed to it, then the row
contained in the message is copied into the routing table if
the destination node is within 2 hop distance

E.09 Receive CL_CHG from node nb $\longrightarrow$
if i is gateway and the sender is clusterhead of one of its neighbors, then
updates its GC_TABLE

An ordinary node periodically checks whether its clusterhead is still alive or not, by sending a CL_REQ message through $n.i$. In case it finds out that it has no clusterhead within two hop distance, then it sets its variables accordingly and waits for a CL_ANN message from a clusterhead node within two hops distance. The ordinary node becomes a clusterhead if there is no clusterhead within two hops distance.

A CL_REQ message travels at most two hops from the sender. Once the CL_REQ message reaches the right destination but finds that the clusterhead moved from that location, the node in that particular location or the node which was supposed to be the one hop neighbor on the shortest path from the sender to the supposed-to-be clusterhead's location sends a CL_CHG message indicating that the previous clusterhead no longer exists in that location.

3.2. Gateway Selection Module. In the gateway selection protocol, a gateway node periodically does the following. It checks whether there exists another gateway in two hop distance that connects the same clusters. If it finds one, it compares its own ID with it. If it has a smaller ID, then it relinquishes its role as a gateway by updating its $g.i$ variable and sending a GW_REJ message.

3.3. Route Discovery Module. For route discovery, we have intra-cluster (routing within the cluster) and inter-cluster routing (routing between the clusters).

For intra-cluster routing, each clusterhead keeps in its routing table data about the nodes that belong to its own cluster, collected in the clusterhead election module using CL_REQ messages. These messages

Algorithm 3.3 Gateway Selection Module

Actions of some node i

G.01 Timeout $\longrightarrow$ if i is a gateway and there is another gateway within
2 hops with a lower ID that connects at least the clusters, then sends
GL_REJ to all neighbors

G.02 Receive GL_ANN from node nb $\longrightarrow$
if i is a clusterhead and the message is addressed to it, then updates
its inter-cluster table
else if i is a gateway and there is another gateway within 2 hops with
a lower ID that connects at least the clusters, then i sends
GL_REJ to all neighbors
else it forwards the message to its neighbors if the hop count < 2,
but drops the message if the hop count ≥ 2

G.03 Receive GW_REJ from node nb $\longrightarrow$
if i is a clusterhead and the message regards one of its bordering
gateway node, it removes all such rows containing the sender's ID
in the GW field of its tables
else it forwards the message to its neighbors if the hop count < 2,
but drops the message if the hop count ≥ 2

are periodically sent by a non-clusterhead node to check the status of its own clusterhead and the path towards it.

For inter-cluster routing, the clusterheads as well as the gateway nodes keep track of the gateway-destination and clusterhead-destination pairs, respectively, to reach the temporary destination, which is a milestone in reaching the actual destination. This data is collected only when there is a need to communicate with the node and stored in the inter-cluster tables. The tables are purged by the routes that are unused for a long time, and their entries are kept up-to-date.

The following steps are repeated until the route is found.

1. Sender checks with its clusterhead if its routing table has an entry for the destination node that it wants to communicate with. If the cluster-head has an entry, the sender gets the path from the clusterhead and uses it to communicate.
2. If the clusterhead's routing table does not have an entry, it checks with the clusterhead's gateway table. If it finds an entry, then it uses that route to communicate.
3. If the clusterhead's gateway table does not have an entry, then it checks with the gateway's cluster tables of all the bordering gateways for the route. If it finds the route, it uses that to communicate.

4. Proof of Correctness. LEMMA 4.1. *The maximum number of hops between a clusterhead and a member of its own cluster is two.*

Proof. In clusterhead election module, Actions E.02 and E.06 ensure that any clusterhead announcement (CL_ANN) message or the clusterhead accept (CL_ACCEPT) message can travel at most a distance of two hops. For a node to be a member of a cluster it has to receive the clusterhead announcement message from a clusterhead and send the clusterhead accept message back to the clusterhead, which is possible only if the node is at a two-hop distance from its clusterhead. □

LEMMA 4.2. *No two clusterheads can be neighbors of each other.*

Proof. We prove this lemma by contradiction. Suppose there are two clusterheads that are neighbors.

Action E.02 ensures that the clusterhead announcement message (CL_ANN) of one clusterhead reaches the other that is at one or two-hop distance from it (Lemma 4.1). When a clusterhead receives a clusterhead announcement message, it compares its own ID with the sender's ID. If its ID is less than the sender's ID, it relinquishes its role as a clusterhead and sends a clusterhead reject message (CL_REJ) message to all its two-hop neighbors.

Algorithm 3.4 Route Discovery Module

Actions of some node i

A.01 Receive *Routedisc* from node nb $\longrightarrow$
 if the same message was received before, then drop it
 if i is a clusterhead
  if the message was addressed to i then sends back an *ack* message
  else if the destination node belongs to its cluster, it sends
   the *shortestpath* message to the sender
   else it updates its inter-cluster table and sends the
    updated message to the bordering gateway nodes
 else if i is a gateway
  if the message was addressed to i then sends back an *ack* message
  else if the destination node belongs to its inter-cluster table, it
   forwards it to all the clusterheads in its inter-cluster table
   else it updates its inter-cluster table and sends the
    updated message to the bordering gateway nodes
 else if i is an ordinary node
  if the message was addressed to i then it sends back an *ack* message
  else forwards the message to its neighbors

A.02 Receive *me_dest* from node nb $\longrightarrow$
 if i is the clusterhead of the destination and the sender does not belong to
  its inter-cluster routing table, it updates the table and sends the updated
  message to all its bordering clusterheads
 else if i is a gateway
  if the clusterhead of the destination is at one hop distance,
   it forwards the message
  if the sender does not belong to the inter-cluster routing table,
   it updates the table and sends the updated message to
   all its bordering clusterheads
 else if i is an ordinary node and the clusterhead of the destination is
  at one hop distance, it forwards the message

A.03 Receive *me_dest* from node nb $\longrightarrow$
 if i is a clusterhead or a gateway
  if the sender does not belong to its inter-cluster routing table,
   it updates its table and sends the updated message to all its
   bordering gateway nodes
  if it is not the destination, then it forwards the message to all
   the nodes in the specified in the field *route* of the message
 else if i is an ordinary node and it is not the destination, then it forwards
  the message to all nodes in the specified in the field *route* of the message

A.04 Receive *ack* from node nb $\longrightarrow$
 if i is a clusterhead, it updates its table and sends the updated message
  to the bordering gateways
 else if i is a gateway, it updates its table and sends the updated message
  to the bordering clusterheads
 else if i is an ordinary node, if the clusterhead of the destination is at
  one hop distance, it forwards the message to that particular neighbor

Action E.03 ensures that a clusterhead reject message reaches all the two-hop neighbors. So, the clusterhead with lower ID no longer remains a clusterhead. This contradicts our assumption that there can be two clusterheads that can be one-hop neighbors. □

Lemma 4.3. *The minimum number of hops between two clusterheads is three.*

Algorithm 3.5 Route Discovery Module (continued)

A.05 Receive *Ctable_update* from node *nb* $\longrightarrow$
    if i is a gateway
        if the message is from a neighboring clusterhead, it updates its
            inter-cluster routing table, else forwards it to its neighbors
    else if i is an ordinary node, not the addressee, but the addressee is
        a neighbor then it forwards the message to it

A.06 Receive *Gtable_update* from node *nb* $\longrightarrow$
    if i is a clusterhead
        if the message is from a gateway node that is present in its inter-cluster
            routing table, it updates its inter-cluster routing table
        else forwards it to its neighbors
    else if i is an ordinary node, not the addressee, but the addressee is
        a neighbor then it forwards the message to it

Proof. From Lemma 4.2, no two clusterheads can be neighbors of each other. Assume that the distance between two clusterheads is two hops. But because the node between them becomes a gateway and acts as a common node for both clusters, that cancels one of the two clusterheads with lower ID. □

LEMMA 4.4. *The maximum number of hops between the clusterheads of two neighboring clusters is five.*

Proof. Let us assume that the distance between two given clusterheads is six. According to *Clusterhead Selection* Module, Action E.02 ensures that any clusterhead announcement message travels at most a distance of two hops. Then, there is at least one node situated in between the two clusterheads that does not receive any clusterhead announcement message. This node waits for a timeout period (Action E.01) and then, at timeout, it sets itself a clusterhead forming its own cluster. Then the distance between the two original clusterheads reduces to three. □

LEMMA 4.5. *If there exists only one link connecting two neighboring clusters then the eligible gateway node(s) of the link will be selected as gateway nodes.*

Proof. We prove this lemma by contradiction. Suppose the nodes connecting the clusters are not gateway nodes. By the definition of a gateway, both nodes are eligible gateway nodes because both of them have at least one neighbor that does not belong to its own cluster. In *Gateway Selection* Module, we eliminate the eligible gateway nodes becoming the gateway nodes only if they belong to the same cluster. So, both the nodes become the gateway nodes that contradict the assumption that they are not gateway nodes. □

LEMMA 4.6. *If both the sender and destination are in the same cluster, a route discovery message is always acknowledged.*

Proof. When a node generates a route discovery message (*Routedisc*), it first sends it to its own clusterhead. Route discovery within a cluster means that the sender and destination belong to the same cluster. If the message reaches the destination before reaching the clusterhead, the destination node directly sends the acknowledgment (*ack*) message to the sender following the reverse path followed by the route discovery message. If the message reaches the clusterhead, all the clusterheads have entries for all the nodes in their intra-cluster table (routing table as named in *Route Discovery* Module) that belong to its own cluster. Once the clusterhead receives the message, it looks in its routing table, attaches the route from itself to the destination to the path followed by the route discovery message, and sends an acknowledgment message to the sender using a *shortestpath* message on the reverse path followed by the route discovery message.□

LEMMA 4.7. *If a node moves to another cluster, the route discovery algorithm will be able to find the node in finite time upon a request.*

Proof. When a node is part of a cluster, it periodically acknowledges a clusterhead that it is still part of the cluster.

When the node moves out of the cluster, the clusterhead waits for a timeout interval, then removes all the rows with this node as destination from its intra- and inter-cluster routing tables, and updates the same to its boundary gateway nodes so that they can remove the rows from their inter-cluster routing tables.

If the node joins another cluster, it acknowledges the new clusterhead's CL_ANN message with a CH_ACCEPT message, to acknowledge that it has joined the new cluster. The new clusterhead updates its entry in its intra-cluster routing table.

If the node itself becomes the clusterhead because it is not in two-hop distance from any clusterhead, then it broadcasts CL_ANN messages to all the other nodes. Eventually gateways nodes adjacent to that cluster will receive the message and the route is thus discovered. □

5. Conclusion. We have presented a route discovery algorithm for MANET based on link-cluster architecture. The algorithm selects the clusterheads and gate-way nodes, and then builds routing tables for nodes both inside and outside the cluster. The proposed protocol guarantees that in finite number of steps, the network is divided into clusters. The algorithm attempts to minimize the number of clusterheads and gateway nodes to avoid storing redundant data. For intra-cluster routing, the shortest paths are maintained. For inter-cluster routing, we implement routing on-demand (the shortest paths are maintained only for the nodes that need to send packets). For both inter- and intra-cluster routing, the paths are loop free.

The proposed algorithm adapts to arbitrary movement of nodes, and joining and/or leaving of existent nodes.

As future work, we currently explore the possibility of a self-stabilizing cluster-based route discovery, in which, starting from an arbitrary configuration of the network, a correct configuration is reached in finite time without human intervention.

Shortest paths are guaranteed only for intra-cluster routing. Another direction for future research is to study the degree of sub-optimal paths for inter-cluster routing by varying various parameters.

REFERENCES

[1] D. J. Baker and A. Ephremides: A Distributed Algorithm for Organizing Mobile Radio Telecommunication Networks. *Proceedings of the Second International Conference on Distributed Computer Systems*, pages 476–483, April 1981.

[2] D. J. Baker and A. Ephremides: The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, 29(11), pages 1694–1701, November 1981.

[3] A. Bhatnagar and T. G. Robertazzi: Layer Net: a New Self-organizing Network Protocol. *Military Communications Conference (Milcom)*, vol. 2, pages 845–849, 1990.

[4] G. G. Chen, J. W. Branch, B. K. Szymanski: Self-selective routing for wireless ad hoc networks. *IEEE International Conference on Wireless And Mobile Computing, (WiMob'2005)*, August 2005.

[5] C. C. Chiang: Routing in Clustered Multihop, Mobile Wireless Networks. *Proceedings of the ICOIN*, 1996.

[6] C. C. Chiang, H-K Wu, W. Liu, and M. Gerla: Routing in Clustered Multihop, Mobile Wireless Networks. *IEEE Singapore International Conference on Networks*, pages 197–211, 1997.

[7] S. R. Das, C. E. Perkins, and E. M. Royer: Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. *Proceedings of INFOCOM*, March 2000.

[8] A. Ephremides, J. E. Wieselthier, and D. J. Baker: A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling. *Proceedings of the IEEE*, 75(1), pages 56–73, January 1987.

[9] Z. Kai, W. Neng, and L. Ai-Fang: A new AODV based clustering routing protocol. *International Conference on Wireless Communications, Networking, and Mobile Computing*, September 2005.

[10] V. Kawadia, Y. Zhang, and B. Gupta: System Services for Implementing Ah-hoc Routing Protocols. *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)*, pages 135–142, 2002.

[11] Y. B. Ko and N. H. Vaidya: Location-aided routing in mobile ad hoc networks. *Technical report 98-012, Texas A&M University*, 1998.

[12] T. Johansson, L. Carr-Motyckova: Bandwidth-constrained Clustering in Ad Hoc Networks *The Third Annual Mediterranean Ad Hoc Networking Workshop*, pages 379–385, June 2004.

[13] C. E. Perkins and E. M. Royer: Ad-Hoc On-Demand Distance Vector Routing. *Second Annual IEEE Workshop on Mobile Computing Systems and Applications*, pages 99–100, February 1999.

[14] S. J. Philip, J. Ghosh, S. Khedekar, and C. Qiao: Scalability analysis of location management protocols for mobile ad hoc networks. *Wireless Communications and Networking Conference*, March 2004.

[15] M. J. Post, S. Kershenbaum, and P. E. Sarachik: A Distributed Evolutionary Algorithm for Reorganizing Network Communication. *Military Communications Conference (Milcom)*, 1985.

[16] E. M. Royer and C. K. Toh: A Review of Current Routing Protocols for Ad hoc Mobile Networks. *IEEE Personal Communications*, 6(2), pages 46–55, April 1999.

[17] K. Scott and N. Bambos: Formation and Maintenance of Self-Organizing Wireless Networks. *Thirty-First Asilomar Conference on Signals, Systems, and Computers*, vol. 1, pages 31–35, 1997.

Edited by: Maria Ganzha, Marcin Paprzycki
Received: Jan 30, 2008
Accepted: Feb 9, 2008