BROADCAST AND MULTICAST ROUTING IN AD-HOC NETWORKS WITH PREDETERMINED CLUSTERING

By

Anand S Kashyap

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

August, 2010

Nashville, Tennessee

Approved:

Professor Gabor Karsai

Professor Yuan Xue

Dedicated to my beloved parents

Mrs. Vijayalakshmi S Kashyap and

Mr. Srikrishnan S Kashyap.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Gabor Karsai, for his support, guidance and for providing a great wealth of knowledge during my tenure at the Vanderbilt University in the past two years. I would also like to thank Mr. Himanshu Neema for his help in working with the OMNeT++ network simulator. I am grateful to Dr. Yuan Xue for her insightful comments and direction in this field of research. I also appreciate all my friends and fellow graduate students for all their help and support.

I consider myself extremely fortunate for having a wonderful family. My brother, Dr. Avinash Kashyap, has been a great source of motivation and inspiration throughout my life. I am grateful to my father, Mr. Srikrishnan Kashyap, and mother Mrs. Vijayalakshmi Srikrishnan, for what I am today.

Finally I would like to express my deepest gratitude to the almighty that helps me find peace within myself.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Background

A mobile ad-hoc network (MANET) is a self-configuring network of mobile devices connected by wireless links. MANETs are a kind of wireless ad hoc networks that usually have a routable networking environment on top of a Link Layer. Each device or node in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. Each node must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic [9].

An ad-hoc routing protocol is a convention, or standard, that controls how nodes decide which way to route packets between computing devices in a mobile ad-hoc network [9]. Efficient routing mechanisms in MANETs are a subject of long and deep research. Numerous algorithms have been invented in the past that have been become foundations for MANET routing.

Initially, two popular approaches for designing routing protocols were the *pro-active or table-driven* routing and the *reactive or on-demand* routing.

The first approach maintains fresh lists of destinations and their routes by periodically distributing routing tables throughout the network. Some common examples of pro-active routing protocols are Ad-hoc Wireless Distribution Service (AWDS) [13], Distributed Bellman-Ford Routing Protocol (DBF) [14], Direction forward routing (DFR) [15], Cluster-head Gateway Switching Routing (CGSR) [16], Destination Sequenced Distance Vector Routing (DSDV) [17], Hierarchical State Routing (HSR) [18], Linked Cluster Architecture (LCA) [19], Optimized Link

State Routing (OLSR) [20] and Mobile Mesh Routing Protocol [21]. The main disadvantages of such algorithms are the respective amount of data needed for maintenance and the slow reaction on restructuring and failures.

The reactive on-demand protocols find a route on demand by flooding the network with Route Request packets. Some popular examples of this kind are Multirate Ad-hoc On-demand Distance Vector Routing Protocol [22], Reliable Ad-hoc On-demand Distance Vector Routing Protocol [23], Admission Control enabled On demand Routing (ACOR) [24], Ad-hoc On-demand Multipath Distance Vector [25], Caching and Multipath routing (CHAMP) [26] and Dynamic MANET On-demand Routing (DYMO) [27]. But these protocols also suffer from disadvantages such as high latency time in finding routes and excessive flooding that can lead to network clogging.

Researchers are trying to overcome these disadvantages and have tried to devise smarter algorithms that use the best of both the approaches. This has resulted in a proliferation of several alternate approaches towards developing routing algorithms. Some of these methods are given below:

i. *Flow oriented routing:* Protocols of this type find routes by following present flows. One option is to unicast consecutively when forwarding data while promoting a new link. Disadvantages of these algorithms are that it takes a long time to explore new routes without their prior knowledge and may refer to tentative existing traffic to compensate for missing knowledge on routes. e.g.: Inter-zone Routing Protocol/reactive part of the ZRP (IERP) [28].

ii. *Adaptive or Situation aware routing:* These protocols try to combine the advantages of both proactive and reactive routing. The routing is initially established with some proactively prospected routes and then proceeds to update routes by some method of reactive flooding. Some disadvantages of such algorithms are that the initial determination of routes depends on number of nodes initially activated and the reaction to traffic depends on gradient of traffic flow. e.g.: Temporally-ordered routing algorithm routing protocol (TORA) [29].

iii. *Hybrid routing:* Commonly referred to as balanced-hybrid routing, it is a combination of distance-vector routing, which works by sharing its knowledge of the entire network with its neighbors and link-state routing which works by having the routers tell every router on the network about its closest neighbors. Hybrid routing protocols use distance-vectors for more accurate metrics to determine the best paths to destination networks, and report routing information only when there is a change in the topology of the network. Hybrid routing allows for rapid convergence but requires less processing power and memory as compared to link-state routing.
e.g.: Enhanced Interior Gateway Routing Protocol (EIGRP) [12].

iv. *Hierarchical routing:* In these protocols, the choice of proactive and reactive routing depends on the hierarchic levels where the nodes reside. The routing is initially established with some proactively prospected routes and then serves the demand from additionally activated nodes through reactive flooding on the lower levels. The choice of one or the other method requires proper attribution at respective levels. These algorithms also suffer from certain disadvantages such as high dependence on depth of nesting and addressing scheme, reaction to traffic demand depends on meshing parameters, dependence on quality of administration and addressing scheme and adverse effects caused by rapid changes in topology. e.g.: Distributed Dynamic Routing Algorithm (DDR) [30].

v. *Geographic routing protocols:* These protocols acknowledge the influence of physical distances and distribution of nodes to areas as significant to network performance. Efficiency in such protocols depends on balancing geographic distribution versus occurrence of traffic and these protocols often struggle to maintain the balance. Moreover, under overloaded conditions, performance may get affected with high loads, thwarting the negligence of distance [31].

vi. *Power-aware routing protocols:* Energy required to transmit a signal is approximately proportional to d^{α}, where d is the distance and $\alpha >= 2$ is the attenuation factor or path loss exponent, which depends on the transmission medium. When $\alpha = 2$ (optimal case), transmitting a signal half the distance requires one fourth of the energy and if there is a node in the middle willing to spend another fourth of its energy for the second half, data would be transmitted for half of the energy than through a direct transmission (inverse square law of physics). These algorithms use this concept for routing messages thereby saving power in individual nodes. Some drawbacks of these algorithms are that they tend to induce a minor delay for each transmission [32].

Other routing algorithm classes include Multicast routing protocols and Geographical multicast routing protocols which are used for specific applications depending on the routing needs and the topology of the networks.

Motivation

An obvious conclusion that can be derived from the above algorithms and techniques is that there is no single routing technique that can be applied generically across all applications and MANET systems. All the algorithms have drawbacks which are inherent to the techniques that these algorithms have adopted. It would be thus be prudent to design algorithms by considering several influencing parameters. Some of the factors that govern routing designs are;

a) Size and density of the network (number of nodes)
b) Node mobility
c) Type of application

This work aims to design an efficient routing algorithm for MANETs by considering all the above factors. The goal is to design a routing mechanism for effectively routing broadcast and multicast messages across the network. The design is applicable to systems where nodes are clustered into loose groups. Each cluster has a leader called the cluster-head which coordinates a lot of the message passing activity. In the proposed system these clusters are predefined and the cluster-heads are predetermined. In situations where a leader has to be picked dynamically, one can use 'leader-election' algorithms to do so (though this will lead to performance degradation). A typical real world system that emulates this kind of network is a system with groups of emergency vehicles moving towards a common destination or large military contingents moving towards a target area. The use of such restricted premise allows us to create a customized highly efficient routing mechanism. Moreover, by solving a problem of smaller magnitude allows us to extend the work further in future to larger more complex MANET systems. As stated earlier, the primary goal of this work is to devise an efficient but customized routing technique for a certain kind of application, rather than trying to determine a generic routing technique that can be applied across all application domains. We have also seen from the earlier examples that any effort to devise a universal routing technique has always had its disadvantages related to one or many aspects of its designing technique.

The rest of the Thesis is organized into the following chapters. Chapter 2 contains related work which discusses other clustered routing techniques. Chapter 3 introduces the reader to the author's work. It explains the initial setup, routing mechanism and provides other related information. Chapter 4 discusses the simulation environment and some of its features in detail. OMNeT++ is the simulation engine used in this work [10]. The associated INET library has provided some useful implementations of network entities that have made the work of the author easier. The INET Framework contains models for several Internet protocols: UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF, and several other protocols [11]. INET has also been discussed in detail in Chapter 4. Chapter 5 is about implementation details and how the system was constructed. It explains the various modules developed for this work and how the system was setup for the purpose of simulation. Chapter 6 discusses some results that were obtained from the simulations. We gauge the algorithm's effectiveness from the results we obtain. In Chapter 7 we make some conclusions about this work and also discuss about possible future work in this area.

CHAPTER II

RELATED WORK

Utilizing node topology for devising smart routing techniques has been in use for several years. Especially using the clustered positions of MANET nodes to group them into virtual groups (clusters) for easy coordination has been in vogue in the recent past. Routing techniques discussed in this section utilize the cluster based approach for routing messages in an ad-hoc network. It provides the reader with a good background about cluster based routing and enables the reader to easily relate to the author's work.

Before delving deep into the various clustering approaches, we define two terms relevant to this approach: cluster-head and gateway. A cluster-head is a node which coordinates the routing of messages within a cluster. A gateway is node on the border of two (or more) clusters belonging to both clusters, aiding in the routing of messages across the clusters they are connecting. They act as bridges between these clusters.

In [2], an ad-hoc network is logically represented as a set of clusters. The cluster-heads form a d-hop dominating set. (In a d-hop dominating set each node is at most d hops from a cluster-head). These cluster-heads form a virtual backbone and maybe used to route packets in their cluster. The authors present a heuristic to form d-clusters in a wireless ad-hoc network. The clusters are formed by diffusing node identities along the wireless links. When the heuristic terminates, a node either becomes a cluster-head or is at most d hops away from its cluster-head. The value d is parameter to the heuristic and the heuristic can be run at regular intervals or when the network configuration changes. The heuristic re-elects cluster-heads every time thus reducing the overhead of transition from old cluster-head to new cluster-head. Cluster-head responsibility is also evenly distributed among all the nodes in the cluster making the heuristic fair and stable. The authors perform experiments to show their approach is better than the LCA approach and degree based approach.

In [1], the authors propose an efficient passive clustering technique which does not employ control packets to collect topological information in ad-hoc networks. Repeated election and re-election of cluster-heads and gateways are avoided. Here the routing algorithm is initiated when a node wants to transmit a message. The source node assumes the role of a cluster head. Two nodes that contend to become the cluster-head resolve the conflict through the least id method as each node is assigned a unique id. Gateways are chosen by taking into account the history of competitions a node underwent while deciding its status. The authors call this gateway selection mechanism Intelligent Gateway selection Heuristic.

Authors of [4] propose a routing technique through dynamic group construction. In this called routing groups. A routing group communicates with other routing groups via the boundary mobile hosts acting as forwarding hosts. In a routing group the mobile hosts are divided, by means of the dominating values, into two groups – one positive cluster and several non-positive clusters. The nodes in the positive cluster maintain the topology information of the routing group. Under such a construction environment, intra-group routing performs unicasting and gets multiple paths, while inter-group routing performs on group level by propagating route requests to the boundary clusters, which are called bridge clusters. The authors have claimed that this scheme massively reduces message complexity and also provides more efficient infrastructure updates.

In [3], the authors propose a Scenario-based Clustering distributed algorithm (SCAM) in which a distributed algorithm based on *(k, r) dominating set* is used for the selection of cluster-heads and gateway nodes where *k* is the minimum number of cluster-heads per node in the network and *r* is the maximum number of hops between the node and the cluster-head. The quality of the cluster-head is calculated based on various metrics, which include connectivity, stability and residual battery power. The algorithm periodically calculates the quality of all dominating nodes and if it goes below a threshold level, it resigns the job as cluster-head and sends this message to all other member nodes. Since these nodes have *k* dominating nodes within *r* hop distance, it can choose the best qualified node as its cluster-head. The technique tries to maintain stable clusters with fewer messages.

Another interesting technique proposed in [5] uses mobility patterns to form clusters of variable diameters. This algorithm is also a d-hop clustering algorithm. The authors introduce metrics such as variation of estimated distance between nodes over time and estimated mean distance for cluster, in order to measure the stability of the cluster. The stability of the cluster will determine its size. Nodes with similar moving patterns are grouped into one cluster. The authors conclude stating that this technique can be used to provide an underlying hierarchical routing structure to address the scalability of routing protocols in large MANETs.

In [6], the authors propose a clustering technique based on mobility prediction. The algorithm is a weighted clustering technique based on an on-demand distributed clustering algorithm. The cluster-heads periodically sends information about is position and speed in *Hello* messages. Nodes store this information in prediction lists. Using information from these prediction lists nodes decide whether to stay in the current cluster or not. If the node predicts that its cluster-head is at a distance farther than the transmission it tries to join another cluster if it can find one, otherwise it stays in the same cluster. The non-cluster-head nodes use the most recent information from the prediction lists to predict their cluster-head position and speed. The authors simulate their algorithm to show that their algorithm performs better than the underlying weighted clustering algorithm in terms of updates to the dominating set, handover of nodes between clusters and average number of clusters in a dominant set.

A self-organizing approach to MANET clustering was proposed in [8]. The paper tries to overcome rapid power depletion in mobile nodes caused to the large number of messages being passed. The algorithm uses route messages of a proactive routing protocol for keeping track of nodes in the cluster thus minimizing explicit message passing for cluster maintenance. Moreover the cluster-head in this protocol only acts an identifying label for that cluster. It does not perform any cluster maintenance or hierarchical routing and thus does not become a bottleneck in the cluster functioning. The algorithm boasts of low latency in the cluster formation phase and it also elects gateways during this phase thus avoiding the need to have a separate phase for gateway discovery thereby reducing the transmission overheads. The proposed algorithm cannot re-elect cluster-heads and the authors have deemed it as “work in progress”.

Almost all of the previously discussed algorithms use a single quality measure to distinguish the capabilities of the nodes in the selection of cluster-heads and in the formation of clusters. A single measure bounds the efficiency of the selection process and degrades network performance. Paper [7] discusses a scalable clustering approach that can generate customizable clustering techniques with as many quality measures as needed. The quality measure used for cluster-head selection is a weighted sum of a number of quality measures such as throughput, degree of connectivity and energy level. This allows the algorithm to choose the best cluster-head in every cluster. The authors also show empirical results that their clustering algorithm experiences significant improvements in network performance.

This is not an exhaustive list of all clustering algorithms but simply a list of work that is closely related to the work presented in this thesis. As we can observe from the above algorithms, there is no universal clustering algorithm that can be applied across all application domains. Most of them try to address key issues pertaining to ad-hoc network routing but it is hard to determine the best among all the ones given above. The cluster routing technique proposed in this paper is simply an addition to the above list algorithms that is suitable to given application domain customized to work efficiently under specific conditions. The proposed algorithm minimizes the number of messages generated during the initialization phase by avoiding a leader-election phase. This is achieved by arbitrarily choosing leaders or cluster-heads beforehand. The author of this work does not claim that the algorithm will be most appropriate for all MANET applications but can be modified minimally in certain aspects to work efficiently for other MANET applications.

CHAPTER III

BROADCAST AND MULTICAST ALGORITHM

This work primarily focuses on ad-hoc network systems that aim to communicate in groups rather than among individual nodes. Some examples of this kind of systems are clusters of emergency vehicles, army contingents converging on a target area, robot groups used in reconnaissance and several others.

The initial setup of the system has the following steps:

i) Nodes grouped together in clusters, with all nodes within a given cluster at a single hop distance from the node acting as the cluster-head. (This can be extended to multi-hop networks, which will be explained in the later sections.)

ii) A predetermined cluster-head node. The cluster-head selection could be implemented as an elective process but that would negate the advantage of this algorithm. The aim is to design a simple yet efficient ad-hoc routing system which can establish the routing infrastructure with minimal number of messages. In most ad-hoc routing algorithms, the cluster-head election consumes the most number of control messages. One may question the rationale of not electing a cluster-head in this algorithm but rather selecting it because that may not resemble a real network system with clusters. A closer look at the applications we have targeted would provide us insight into the rationale for selecting a predetermined cluster. Most of these application systems have a leader node in each of the clusters which can be assigned the role of cluster-head at the beginning of the operation. This work uses this to its advantage and avoids cluster-head election at the beginning.

Cluster-head Declaration

Though the cluster-head is predetermined at the beginning all the nodes in the cluster would need to know which node is the cluster-head. Moreover, in adverse physical conditions the role of the cluster-head will have to be taken over if the actual cluster-head cannot perform the role anymore. (The node may die, lose power rapidly, move out of transmission range etc.) In order to manage all the above mentioned situations, a separate cluster-head declaration message is used. Additionally, these messages also serve as "heart-beat" messages for gateway nodes for them to determine if they still belong to a particular cluster.

Establishing Routing Infrastructure

The following steps will explain how the routing is carried out by the proposed algorithm.

1. The cluster-heads send cluster declaration messages to all the nodes within its cluster.
2. Each non-cluster node (which is at a 1-hop distance), upon receiving this message, will set its cluster-head to this node which sent the cluster-head declaration message. Nodes which are at more than 1-hop distance from a cluster-head cannot receive these messages and remain unassociated with any cluster. Once they move closer (1-hop) to any of these cluster-heads they can receive these messages and immediately form a part of that corresponding cluster.
3. Nodes that receive cluster-head declaration messages from more than one cluster-head declare themselves gateway nodes for the respective clusters. By being gateways, these nodes act as bridge points through which the two clusters can communicate. A gateway node can bridge more than two clusters. It maintains the list of clusters it links.

4. Each gateway node sends its list of cluster-heads to all the cluster-heads in its cluster list. This allows each cluster-head to know which nodes are the gateways and what clusters they connect to.
5. The cluster-heads update their gateway list upon receiving the messages from their gateways and send cluster connection messages to each of their gateways. Using these messages, the gateways can update their cluster list with additional clusters information. This allows the gateways to connect to other clusters indirectly through a set of cluster-heads and gateways.

   Thus each cluster-head and gateway has a list of nodes through which messages can be routed.

   The nodes run a UDP application which requires exchange of messages among them. The UDP application simply broadcasts or multicasts messages. In this work, broadcasting is defined as the transmission of a message from one node to all nodes in the network.
6. When a non-cluster-head node produces a message for broadcasting, it forwards it to its cluster-head. The cluster-head processes that message and forwards it to all nodes in the cluster.
7. When a gateway node receives a message from any of its cluster-heads for broadcasting it forwards it to all its other cluster-heads if it has not received the same message previously.
8. Cluster-heads on receiving broadcast messages from gateways forwards it to all the nodes in its cluster if it has not received the same message previously.

A multicast message is defined as a message produced by a node that is intended to only a certain set of nodes (i.e. to a certain cluster).

9. When the application in a node produces a multicast message, it contains the message data and a destination cluster id. It forwards this to its cluster-head which in turn forwards it to all the nodes in the cluster.
10. When gateways receive multicast messages they process the message and read the destination cluster id. If the destination cluster id is found in its list of clusters it forwards the message, else does not forward it to any cluster in its list.

Thus all broadcast and multicast messages can be routed to the appropriate clusters easily.

Due to the highly dynamic nature of ad-hoc networks, gateways can lose connectivity with any of their clusters. In order to check this, the gateways use the cluster-head declaration messages sent by the cluster-heads at periodic intervals.

11. When a gateway does not receive cluster-head declaration messages from any of its cluster-heads within a certain timeout period, it sends out gateway_change messages to all the cluster-heads to inform them that this node will not serve as gateway for that cluster from which it did not receive cluster-head declaration message for the last timeout period.

The timeout period we use should be sufficiently larger than the period of the cluster-head declaration messages interval to account for the loss-high nature of the UDP protocol.

12. When the cluster-heads receive the gateway_change message they delete the gateway from their list of gateways for that corresponding cluster. This allows them to send

cluster connection messages to any new gateway which may bridge this cluster with the other disconnected cluster.

Thus the routing tables and lists are always maintained up-to-date and it is ensured that the messages are never routed through stale routes.

The implementation of this routing technique was done on the OMNeT++ discrete event simulator, details of which are presented in the next section. The exact implementation details and data structures used will be explained in chapter 5 following the introduction to the OMNeT++ simulator.

CHAPTER IV

THE SIMULATION ENVIRONMENT (OMNeT++)

OMNeT++ is a discrete event simulation environment. Its primary application area is the simulation of communication networks, but because of its generic and flexible architecture, is successfully used in other areas like the simulation of complex IT systems, queuing networks or hardware architectures as well [10].

OMNeT++ provides component architecture for models. Components *(modules)* are programmed in C++ and then assembled into larger components and models using a high-level language *(NED)*. Reusability of models comes for free. OMNeT++ has extensive GUI support, and due to its modular architecture, the simulation kernel (and models) can be embedded easily into applications [10].

Although OMNeT++ is not a network simulator itself, it is currently gaining widespread popularity as a network simulation platform in the scientific community as well as in industrial settings, and building up a large user community [10].

This project uses OMNeT++ 4.0 for its simulations. In addition to OMNeT++, for the purpose of developing the various modules this work has also used the INET framework. The INET Framework is an open-source communication networks simulation package for the OMNeT++ simulation environment. The INET Framework contains models for several Internet protocols: UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF, and several other protocols [11].

The INET framework also provides a set of predefined modules that can be configured to operate as network entities such as routers, hosts, switches and so on. The framework provides mobile hosts which are optimally configured to function as wireless nodes in our simulation. The architecture of these wireless nodes can be seen in the following screenshot.

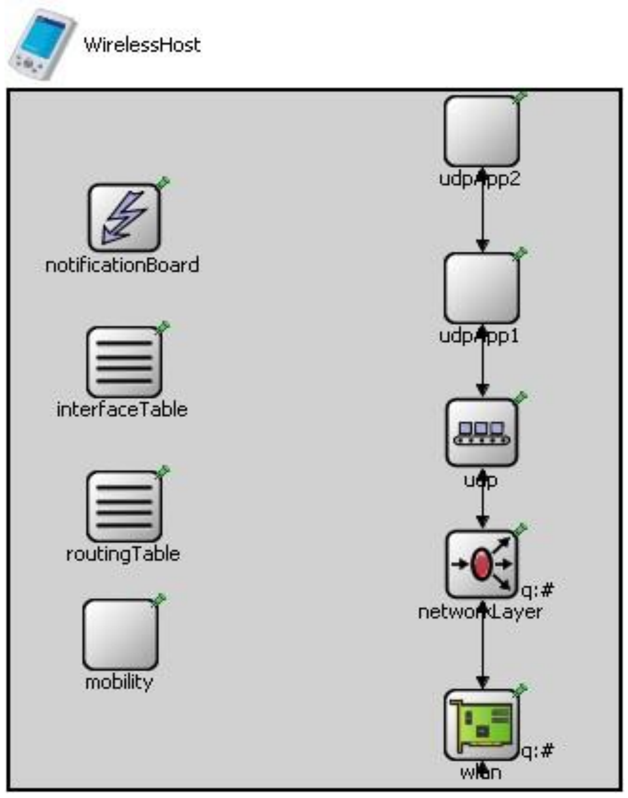

Figure I: Wireless Host

Each wireless host has one external interface, namely, the WLAN interface card. From the diagram above we can observe how packets get routed from the bottom layers to the application layers on top through the network and transport layers (UDP).

The mobility module is responsible for node movement across the field of simulation (Area within which the simulation occurs). The INET framework provides various mobility models which the simulator uses to simulate the movement of nodes on the field of simulation and computes the connectivity between these nodes dynamically during the simulation. Based on the movement of these nodes and their connectivity the network protocols must be able to handle the message passing across these nodes dynamically. This allows users to assess the performance of these protocols under various for a constantly changing node topology. For the purpose of simulation we use the BonnMotionMobility and CircleMobility provided in the INET framework. Nodes with CircleMobility simply move in circles around a fixed center point. Nodes with BonnMotionMobility simply trace the path provided by an external file. The file contains x and y coordinates with corresponding time instances. When the simulation clock reaches a particular time the node will be positioned on the corresponding x and y coordinate. During simulation, over a period of time, the node will trace the path provided by the coordinates in this file.

Other modules such as routingTable and interfaceTable are specific to message routing. The interfaceTable contains information about the various interfaces used in the host. In this case we use only the WLAN card. The routing table contains IP addresses of the various nodes in the network with which this node can communicate.

The algorithm is implemented as an application on the UDP layer. The data application module is also a UDP application. (Any application that uses the UDP protocol for sending messages across nodes is termed as a UDP application). The two modules exchange messages between each other through the AppIn and the AppOut gates. When the data application module wants to broadcast or multicast a message it sends it to the routing module through the AppOut gate which is received by the routing application through the AppIn gate. When the routing module receives an external message for processing it passes it to the data application module through the AppOut gate which is then received by the data application module through the AppIn gate.

Other Modules

a) *Channel Control:* The channel control module is not a real network entity but a module that is required by the simulator to manage other network entities. Channel Control has exactly one instance in every network model that contains wireless or mobile nodes. This module is informed of the location and movement if nodes and determines which nodes are within communication and interference distance. This information is then used by the radio interfaces of nodes during transmissions. Thus the channel control contains all information regarding the node mobility and other network information. It acts as the single backbone through which the simulation event information gets exchanged [11].

b) *Flat Network Configurator:* This module is essential in configuring IP addresses and routing tables for a network in which all hosts and routers have the same network address and will only differ in the host part. This module is not connected to any other module and each model should contain only one instance of it. The module will run only once, at the beginning of the simulation to assign all IP addresses, discover the topology of the network to calculate shortest paths and it will add routes corresponding to the shortest paths to the routing table when the model uses some predefined MANET routing. In order to establish the network topology the module picks all modules that are nodes and builds a graphs from their connections and then runs Dijkstra’s shortest path algorithm for that graph.

CHAPTER V

PROTOCOL IMPLEMENTATION IN OMNeT++

OMNeT++ and the INET framework provide large library network modules that can be used for building our own network applications. The MANET routing is implemented usually on the network layer of the OSI network stack. But due to certain advantages and ease of implementation, the prototype of the proposed routing scheme is implemented on the application layer as a UDP application. No other MANET protocols are used on the underlying network layer to aid in the operation of this routing scheme.

As mentioned, the routing scheme is implemented as a UDP application. This application will be run on every node in the network which is going to participate in the routing. In addition to this application, another simple Source-Sink UDP application is run on these nodes which simply produces and consumes packets, simulating a very simple UDP application. When the simple application wants to broadcast or multicast a message, it sends it to the routing application which then routes it to the other nodes appropriately. When the routing application receives a data message, it validates the message before sending it to the simple application for processing, essentially simulating the behavior of a MANET protocol. Usually MANET routing protocols are implemented on the network layer. Here the proposed MANET routing protocol is implemented as a UDP application for demonstration purposes only.

In the first sub-section a detailed explanation of the types of messages used in the routing protocol is given. The next sub-section the clustering algorithm implementation is discussed followed a small sub-section which provides details about the simple UDP application.

Messages

There are six kinds of messages used in the protocol and one message used by the simple UDP application.

1. *ClusterHdDecMsg*: This is the cluster-head declaration message sent out to all adjacent nodes from the respective cluster-heads. These messages contain two data elements, the IP address of the cluster-head and the cluster id of the respective cluster. This allows all the nodes to know which cluster they belong to and what is the IP address of their respective cluster-heads. A node can establish itself as a gateway if it receives this message from more than one cluster-head and it can also determine to what clusters it is connected to.
2. *GtwyDecMsg*: This is the gateway declaration message sent out by all gateway nodes to their respective cluster-heads. This allows the cluster-heads to know which nodes in the cluster are gateways and to what clusters they are connected to. The message has three data elements. One contains the module id of the gateway, another contains the IP address of the gateway and the third is a list of clusters the gateway is connected to. A gateway can be uniquely identified using these data elements and helps in the easy routing of messages to these gateways.
3. *GtwyChngMsg*: This is the gateway change notification message sent by a gateway node to those cluster-heads to which they are connected when they get disconnected from one of the clusters it was connected to earlier. For example, if a gateway node is connected to three clusters I, II and III and gets disconnected from cluster I, it sends the gateway change message to the other two clusters notifying them of this change. Upon receiving this message the cluster-heads remove this node from their gateway lists to indicate that

this node cannot be used as a gateway anymore to connect to cluster I. This message has two data elements, the IP address of the gateway node and the cluster ID of the cluster from which it was disconnected.

4. *ClusConnMsg*: This is the cluster connection message. Upon receiving the Gateway declaration message, the cluster-head sends this message to the corresponding gateway. This message has the IP address of the cluster-head and the list of gateways and their connecting clusters. This is useful to the gateway because it helps the gateway in deciding whether this cluster-head can indirectly connect to another cluster. This allows the protocol to route messages through multiple clusters and indirectly reaching the destination cluster.
5. *DataMessage*: This is a simple message used in the simple UDP Source-Sink application. This message has two data elements. The first being the actual message which is nothing but a simple string of characters. The other one element is flag which indicates whether the message is for broadcasting or multicasting. Being a simple application the message is also of simple nature. The application and the associated messages can be made arbitrarily complex for real systems. This application is intended to simply demonstrate the underlying protocol.
6. *ClusterHeadWrapper*: When a cluster-head receives a data message that needs to be forwarded, it un-wraps the message from the outer wrapper, processes the message and wraps the message with some additional control information before forwarding it. This wrapped data message is called the ClusterHeadWrapper message. Nodes on receiving this message handle it appropriately and decide whether to process the wrapped data message or not. The control information as part of this message includes the cluster id,

destination cluster id (useful for multicast messages), the IP address of the cluster-head, the id of the node which initially produced the message and the message count of the message produced by that node. Separate message counts are maintained by the nodes for the broadcast and multicast messages. The message is then either cast as a “clusterhead_forward_data_message”, if the message is processed by the cluster-head and then forwarded or as a "clusterhead_to_gateway_forward_data_message” if the message is simply forwarded by the cluster-head to the gateways en-route to its destination (without the cluster-head processing the message, i.e., in case of multicast messages).

7. *Wrapper*: It is similar to the ClusterHeadWrapper but used by non cluster-head nodes. The control information include the IP address of the node, the destination cluster id, the id of the node which initially produced the message and the message count for that kind of message produced by that node. The wrapper messages themselves have two minor variations. These are the wrapper messages forwarded by the non-gateway nodes and the wrapper messages forwarded by the gateway nodes. Depending on the origin of the wrapper messages the cluster-head handles them differently but the two variants have identical structures and related control information. They are distinguished by the names provided to these wrapper messages.

Each of these messages serves a purpose in the routing operation. The routing procedure is discussed in the following sub-section.

Routing Protocol Implementation

In the OMNeT++ paradigm, all modules are implemented by first describing a network descriptor file (NED) followed by the corresponding C++ classes. These NED files contain all relevant parameters for that application and are primarily used for easily interfacing the application with the user. The parameters are read by the corresponding C++ implementation of these modules. In order to build a network with this application, a user will simply need to import the appropriate NED files into the larger application and set the parameters for all the sub applications appropriately. Upon importing the sub applications, the user creates an ini file called the omnet.ini file and initializes the parameter values of all the sub applications in that file. The simulation is triggered by running this omnet.ini file.

The implementation of this routing scheme is separate into two phases, namely, the initialization phase and the operational phase. In the initialization phase the network is setup. In the operational phase the routing infrastructure (routes) is established and messages from the application are routed to other parts of the network.

i) *Initialization*: This phase initializes all the nodes and network elements. Each node is assigned a unique ID which is used later for identifying the nodes. All the parameters for this application are read from the omnet.ini file in this phase and the respective variables are initialized. Each node maintains the number broadcast and multicast messages it has transmitted so far. These are initialized to 0. In the initialization phase the cluster-head of each cluster is determined by reading the parameter used to set this node status as a cluster-head. If a node is initialized as a cluster-head then the cluster count variable is incremented. This is used to determine the number of clusters in the scenario and to assign a unique cluster ID to each cluster.

If the node is a cluster-head it schedules its first cluster-head declaration message to be broadcasted. Since the simulator is simply a discrete event system simulator, all the events occur in a sequence. Hence the scheduling this message even at time zero does not produce a message until all the nodes have been initialized because the message has been scheduled in the discrete event queue after all the initialization. But the first cluster-head declaration message is broadcasted at time zero. (This time is the logical simulation time and not the real time.)

The scheduling of the cluster-head declaration messages in the nodes marks the end of the initialization phase. The transmission of the first cluster-head declaration message marks the beginning of the operational phase which is discussed next.

ii) *Operation*: As mentioned earlier, the first cluster-head declaration message transmitted marks the operational phase of this implementation. This occurs at logical time zero. When the non cluster-head nodes receive this message they add the cluster-head to their list of cluster-heads. Each cluster-head schedules a new cluster-head declaration at regular intervals. The interval is obtained from the update interval parameter set by the user. The default value for this parameter is 2 seconds.

When a node has more than one cluster-head in its list it transmits the gateway declaration message to all the clusters in its list. The structure of the gateway declaration message was declared earlier. Cluster-heads update their list of gateways. This allows cluster-heads route messages to far off clusters. The gateways also check at regular intervals if they are still connected to all the clusters in their cluster list. This interval is once again obtained from a parameter whose default value is set to 5 seconds. When the gateways find that a cluster-head declaration message was not received from one of the cluster-heads from their list within that five second interval they decide that they are not connected to that cluster anymore and transmit gateway change messages to all the remaining clusters in the their list. Hence the users have to ensure that the interval to send cluster-head declaration messages is small compared to the time interval defined to check the reception of the last cluster-head declaration message in the gateways to avoid hasty transmission of gateway change messages. Due to the inherently loss prone nature of the UDP protocol all the messages may not be received by the recipients and the users would need to ensure that there is a sufficiently large time-out period. The values should not be fixed because all networks do not have the same performance. Highly unreliable networks should have a larger timeout period compared to networks that are much more efficient in routing UDP packets.

When the cluster-heads receive a gateway declaration message from a gateway they send the cluster_connection message to that gateway. This message allows the gateway to know what other clusters are connected to this cluster. The gateways can then use this cluster to indirectly route messages to other clusters.

Cluster-heads on receiving a gateway_change message remove the gateway's entry from the list for the cluster to which it was connected earlier. The cluster-heads now wait for the next available gateway for that cluster before they resume routing messages to the disconnected cluster.

The simple Source-Sink application produces data messages that need to be transmitted to other nodes in the network. The routing application receives the data messages from the Source-Sink application and wraps them with additional control information. When the originating node is a cluster-head the wrapped messages are called cluster-head wrapper messages else they are simply called wrapper messages. The control information in these messages is different and serves a different purpose in each case.

If the originating node is a cluster-head it forwards the message to all the nodes in its cluster. If the node is a non cluster-head node, it forwards the wrapped data message only to its cluster-head (cluster-heads if the node is a gateway).

Upon receiving the wrapped data message the nodes process the message. If the node is a gateway, it reads the control information and decides whether to process the data message. The decision is taken based on whether the message was already received earlier. If the message is a new message, the data section is processed. Otherwise the message is simply discarded. Being a gateway it also determines if the message needs to be forwarded to other clusters in its list. If the message to be forwarded is a multicast message with the destination cluster not in the gateway's list, the gateway does not forward the message.

If a cluster-head receives a wrapped message, it reads the control information to determine if the message is to be processed. If the message is new and destined for it (in case of multicast messages), it either processes the message and forwards it to all the nodes in its cluster or simply discards it.

Thus depending on whether the message is a new message and if it is meant for broadcast or multicast the nodes forward the message appropriately. The next section discusses the implementation of the simple Source-Sink UDP application.

Source-Sink Application

The Source-Sink application is a simple UDP application that is run on every node on the network to demonstrate the operation of the proposed protocol. The nodes' application produces data messages at some interval determined by a parameter. The application also has another parameter that defines whether the node wants to broadcast its messages or multicast them. For the sake of simplicity this is fixed at the beginning of the experiment and cannot be changed during the experiment. Thus if a node is defined a broadcast node it always broadcasts messages and if the node is a multicast node it always multicasts its messages to a cluster. The destination cluster for multicasting is not fixed and the node can choose any cluster it wants to multicast the message to. The application forwards the data messages to the routing application which wraps them before forwarding to all the nodes in its cluster. When the routing application receives a wrapped data message, it removes the control information and forwards only the enclosed data message to the Source-Sink application.

On receiving a data message, the application prints its contents and deletes the message. The application also contains data structures to keep track of the number of messages sent and received by the node. These are used recording statistics which can be plotted at the end of the experiment to analyze various performance parameters.

CHAPTER VI

SIMULATION AND RESULTS

To demonstrate the working of the routing operation and its effectiveness, this chapter will describe three simple experiments and the related data that were collected from these experiments. The experiments are simple in nature so as to clearly demonstrate how the routing operation is carried out. The experiments have been designed to demonstrate all the aspects of this protocol but at a scaled down level. Larger or highly complex experiments can be performed using a similar methodology. The protocol itself has been customized to suit the OMNeT++ environment and uses a lot of its built-in libraries. Hence any experimentation has to be carried on this platform for the protocol to work "out of the box".

Experiment I

In the first experiment we have a playground with ten nodes. The nodes are separated into three clusters. The cluster-heads of these three clusters are node7, node9 and node10. The clusters overlap in such a way that they share some nodes between them. Such nodes automatically take up the role of the gateways between these clusters. The gateway nodes for these clusters are node4 and node8. Node2 which lies in cluster2 sends multicast messages at seven second intervals. The cluster to which the multicast message is sent is decided at run time by a random cluster ID generating function. The following screenshot shows the layout of the nodes on the playground. The circles encompass nodes that lie within a particular cluster. Nodes lying within the overlapping regions are the gateways.

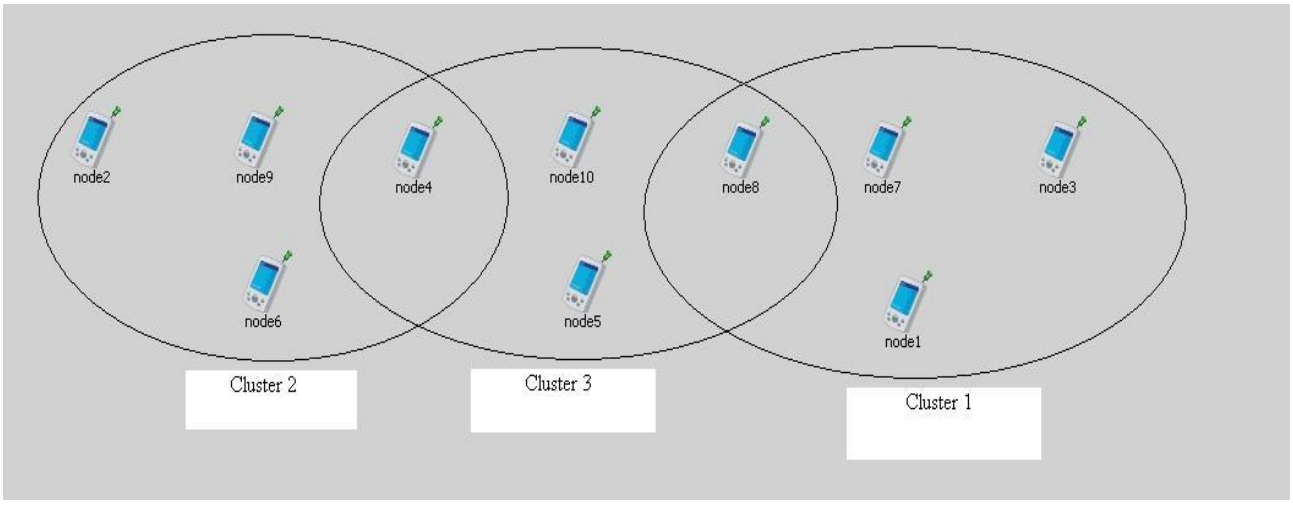

Figure II: Experiment I Node Layout

The cluster numbers are assigned automatically to these clusters during the initialization phase.

The following screenshots show the sequence of steps involved in establishing the routing infrastructure and the routing of data messages through these routes in the MANET. Each red dot in these screenshots represents a message packet being transferred from the source to a destination. The circles surrounding each node is the broadcast range corresponding to that node.

<u>*Note:*</u> *Message broadcasts from a node are represented as multiple individual messages transmitted to each adjacent node. This is specific to OMNeT++'s method of visualization and readers should not confuse this with the actual message transfers that occur in the system.*

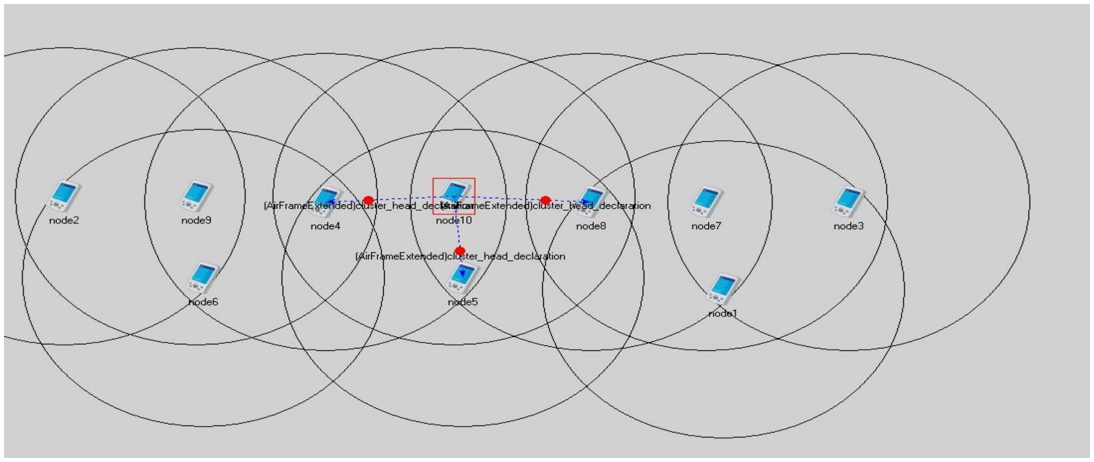
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1
(AirFrameExtended)cluster_head_declaration
(AirFrameExtended)cluster_head_declaration
(AirFrameExtended)cluster_head_declaration

Figure III: Node10 Transmitting Cluster-head Declaration Message

Figure III is the first step in the infrastructure establishment where the cluster-heads transmit the cluster-head declaration messages to the other nodes to announce their leadership. In all our experiments the cluster-head declaration message is sent at five second intervals.

The cluster-heads try to transmit these messages to all the nodes on the field but only the nodes within its transmission range can receive these messages.

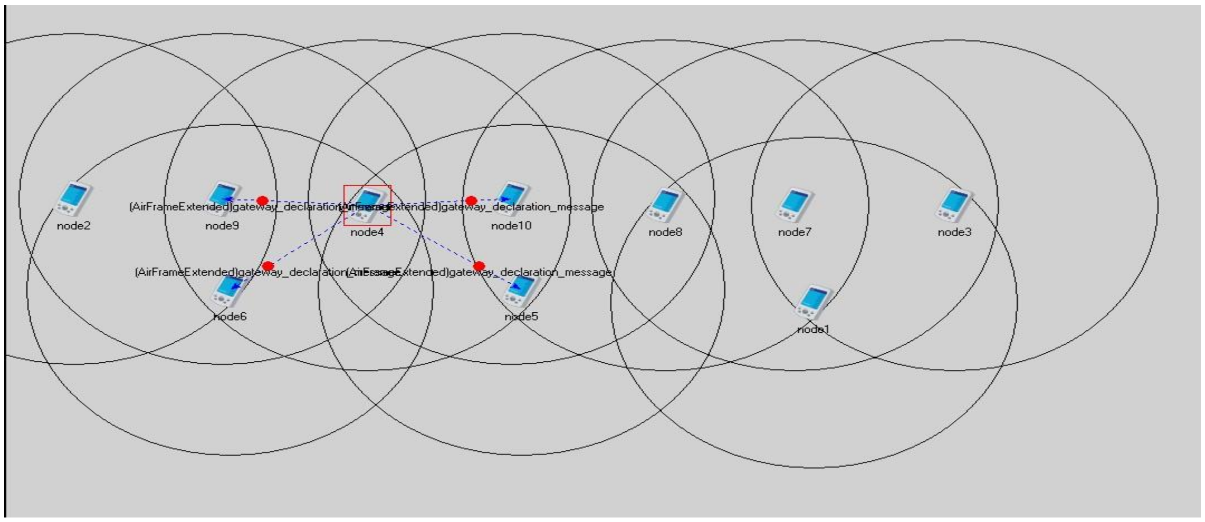
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1
(AirFrameExtended)gateway_declaration_message
(AirFrameExtended)gateway_declaration_message
(AirFrameExtended)gateway_declaration_message
(AirFrameExtended)gateway_declaration_message

Figure IV: Node4 Transmitting Gateway Declaration Message

Figure IV shows node4 transmitting gateway declaration messages to all its cluster-heads in response to receiving cluster-head declaration messages from them. This is because node4 has received cluster-head declaration messages from more than one cluster-head and as result has assumed the role of gateway. This information, along with the cluster-heads it is connected to, is sent as gateway declaration message to all its cluster-heads.

As soon as it transmits this information to the cluster-heads it sets timer for each of the clusters it is connected to. This is to check if it receives the next cluster-head declaration message in time. This helps the gateway to decide it is still connected to a particular cluster.

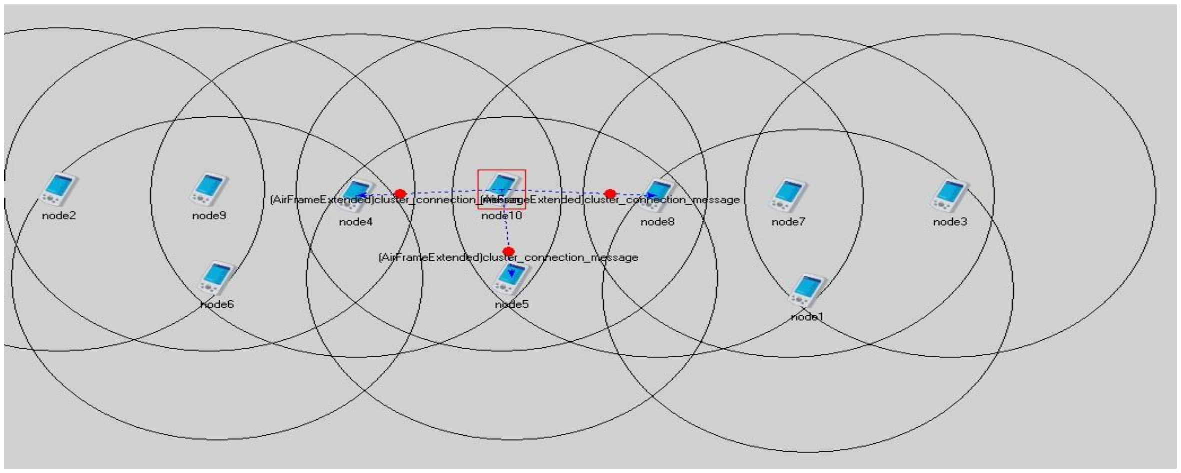
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1
(AirFrameExtended)cluster_connection_message
(AirFrameExtended)cluster_connection_message
(AirFrameExtended)cluster_connection_message

Figure V: Node10 Transmits Cluster Connection Message

Cluster connection messages are sent by cluster-heads to all the gateways in cluster if the cluster is connected to more than one cluster through its gateways. These messages are used by gateways to establish multicast routes. This enables them to know which cluster-head to forward a message to in order to reach the destination cluster.

The cluster connection messages are the last set of control messages for route establishment. When gateways change positions and get disconnected from clusters they send gateway change messages. But this occurs after the initial routing infrastructure establishment and gateways transmit them only in cases of disconnection from their clusters.

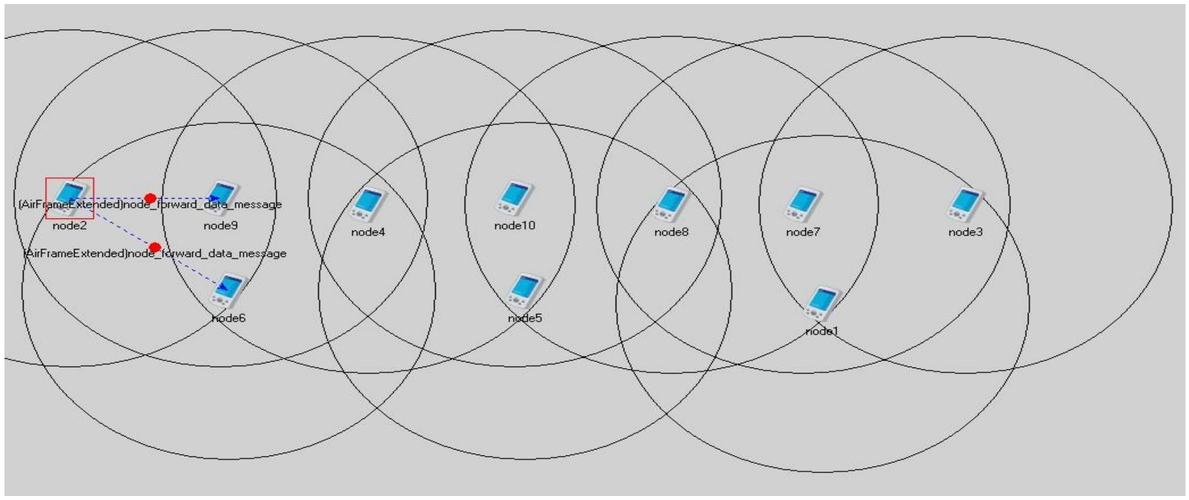
(AirFrameExtended)node_forward_data_message
(AirFrameExtended)node_forward_data_message
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1

Figure VI: Node2 sending Node Forward Data Message

Here node2 sends the first data message which is a multicast message. It simply forwards it to its cluster-head which in this case is node9.

Multicast messages in our routing algorithm are read both by the source cluster and the destination cluster. Thus cluster 2 being the source cluster, node9 forwards it to all the nodes in this cluster. Node9 also reads this message. This multicast message is directed to cluster 1 which is headed by node7.

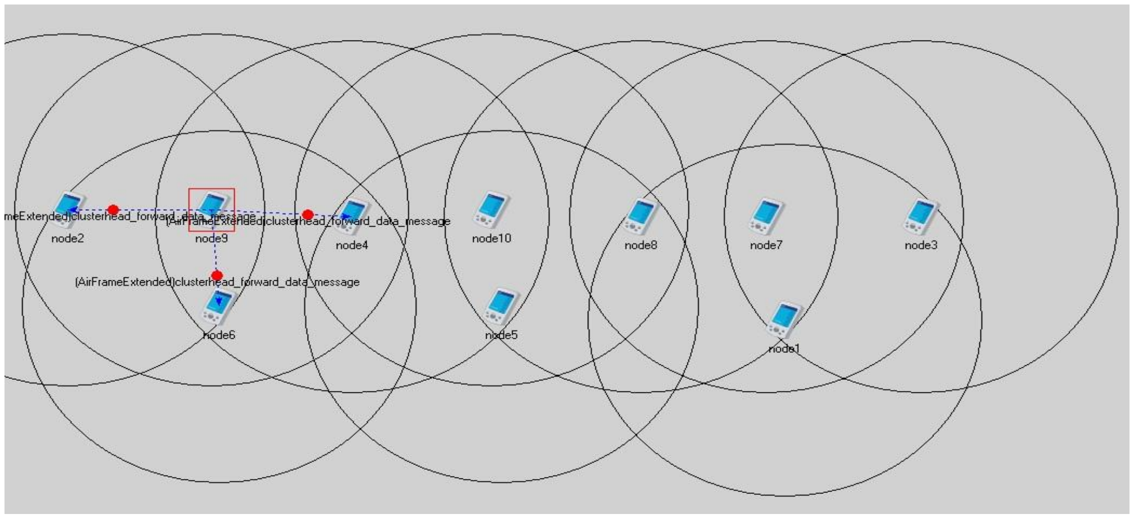
meExtended)clusterhead_forward_data_message
(AirFrameExtended)clusterhead_forward_data_message
(AirFrameExtended)clusterhead_forward_data_message
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1

Figure VII: Node9 Transmits Cluster-head Forward Data Message to Nodes

After node9 reads the multicast message from node2 it forwards it to all the nodes in its cluster. Gateways in this cluster will then re-packet this message with different control information and forward it to appropriate cluster-heads.

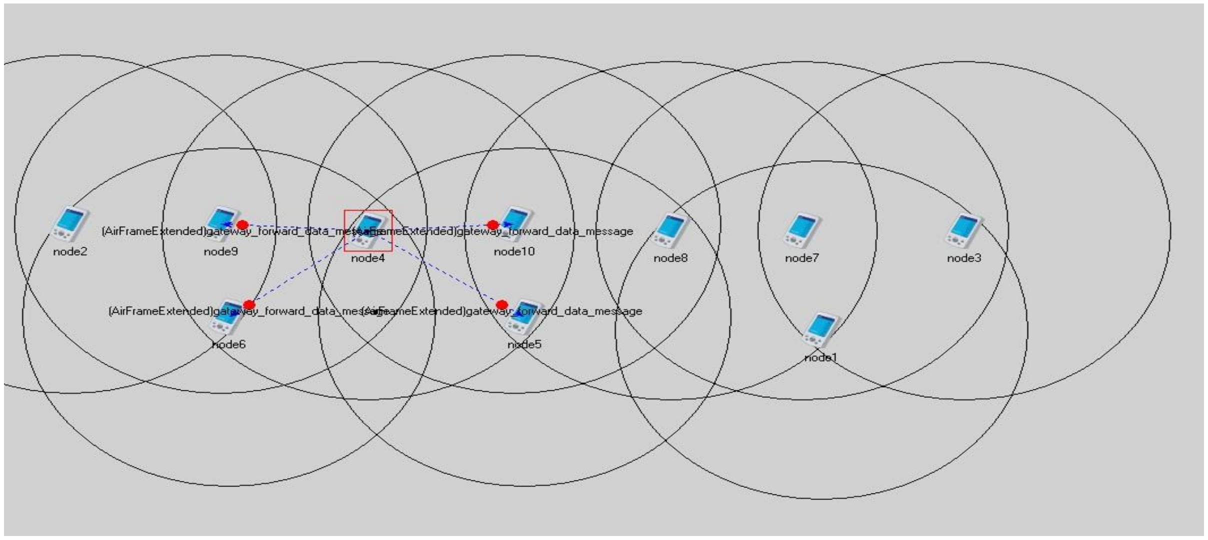
(AirFrameExtended)gateway_forward_data_message
(AirFrameExtended)gateway_forward_data_message
(AirFrameExtended)gateway_forward_data_message
(AirFrameExtended)gateway_forward_data_message
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1

Figure VIII: Node4 Transmits Gateway Forward Data Message to Node10

The multicast message from node2 is destined for cluster 1. Node4 being a gateway forwards it to node10 to be forwarded to the destination. We must note that node4 has two cluster-heads in its list. But it forwards it only to node10 as only node10 can forward it to the correct destination. This is because in node4's list the destination route to cluster1 points to node10 and hence it forwards the packet only to node10.

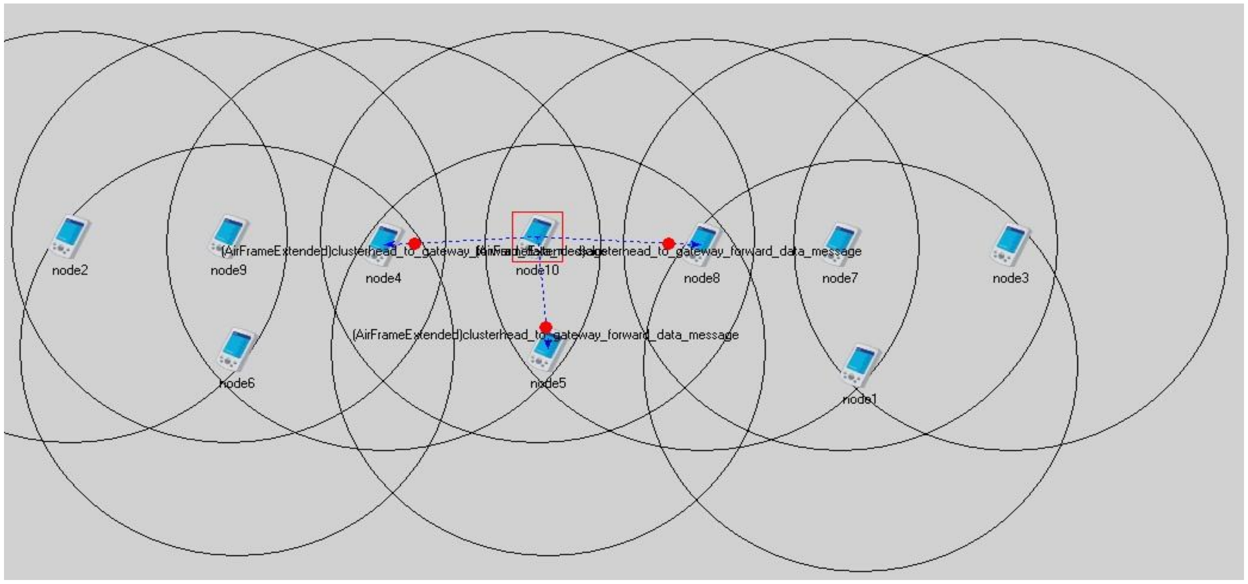
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1
(AirFrameExtended)clusterhead_to_gateway_forward_data_message
(AirFrameExtended)clusterhead_to_gateway_forward_data_message
(AirFrameExtended)clusterhead_to_gateway_forward_data_message

Figure IX: Cluster-head to Gateway Forward Data Message

When a cluster-head receives a message and the cluster is not the intended recipient and it has received the message merely for forwarding purposes, the cluster-head simply forwards this message to its gateways without processing the message. Such a forwarded message from the cluster-head is called Cluster-head to gateway forward data message. When the gateways receive this message they forward it to the appropriate cluster-head if it is available in their list. Gateways do not process (read the contents) the message. They only read the control information. Here node10 forwards the message to gateway node8 to be forwarded to cluster 1.

Node8 receives this message and forwards it to node7 which heads cluster 1. Node7 reads the control information and understands that the message is meant for its cluster and broadcasts to all nodes in its cluster.

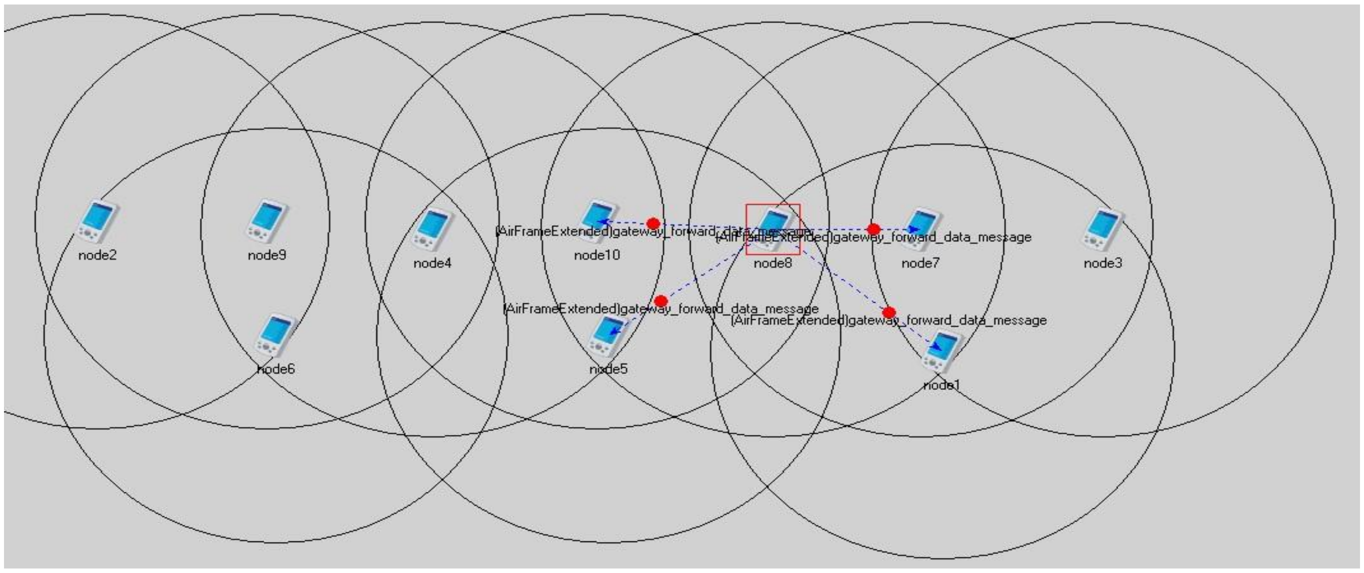
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1
(AirFrameExtended)gateway_forward_data_message
(AirFrameExtended)gateway_forward_data_message
(AirFrameExtended)gateway_forward_data_message
(AirFrameExtended)gateway_forward_data_message

Figure X: Gateway Forward Data Message from Node8

Figure X shows gateway node8 sending the message it received from node10 to node7 which is the destination cluster.

Note: Node8 does not read contents of cluster-head to gateway forward data message because node8 cannot be sure if it belongs to the destination cluster. Only after node7 retransmits this message to all the nodes in its cluster can node8 read the contents of this message. Thus any gateway does not read contents of cluster-head to gateway forward data message but can read the contents of cluster-head forward data message.

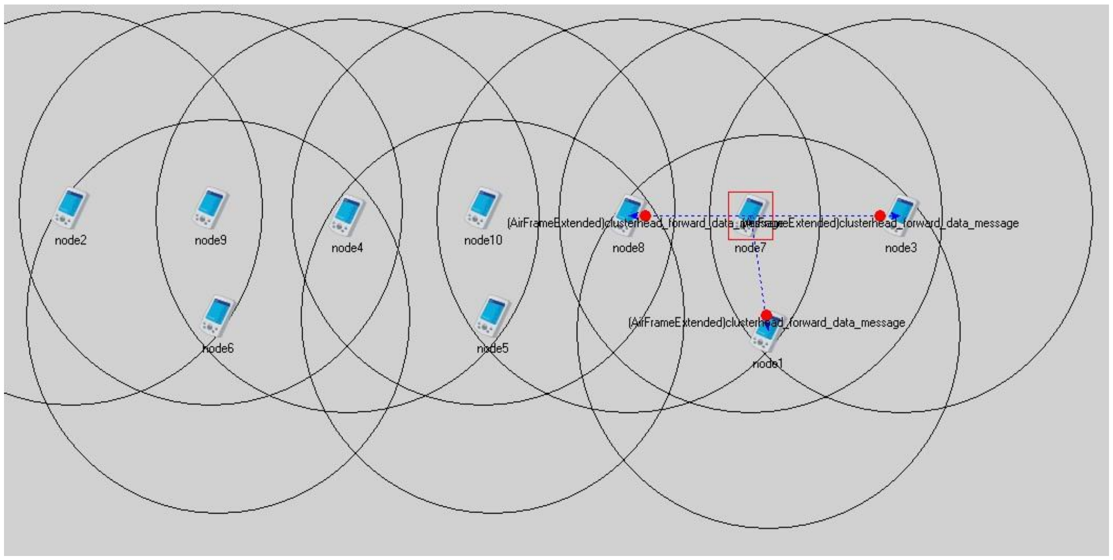
node2
node9
node4
node10
node8
node7
node3
node6
node5
node1
(AirFrameExtended)clusterhead_forward_data_message
(AirFrameExtended)clusterhead_forward_data_message
(AirFrameExtended)clusterhead_forward_data_message

Figure XI: Node7 Transmitting Cluster-head Forward Data Message

Finally node7 forwards this data message to all nodes in its cluster after it reads the control information from the gateway forward data message from node8. Thus all nodes in cluster 1 (destination cluster) receive the destination message. At this point node8 can also process the data message as it is part of the destination cluster.

Thus all nodes in cluster 2 (source cluster) and cluster 1 (destination cluster) receive the data message from node2. Node2 does not receive this message back from its cluster-head as the cluster-head is aware of the sender and does not transmit the message back to node2 avoiding redundant messages in the system.

The steps above show the flow of messages from the source cluster to the destination cluster. After establishing the routing infrastructure the nodes were able to determine the route of a message to the destination cluster. The number of messages required to establish the routes is minimal and the routes are refreshed at regular intervals without flooding the network with too many messages. From these figures we can also observe that the established routes were indeed correct and the messages were processed only by nodes of those clusters which were the intended recipients of these messages.

The following plots show the pattern of message transmission during the simulation.

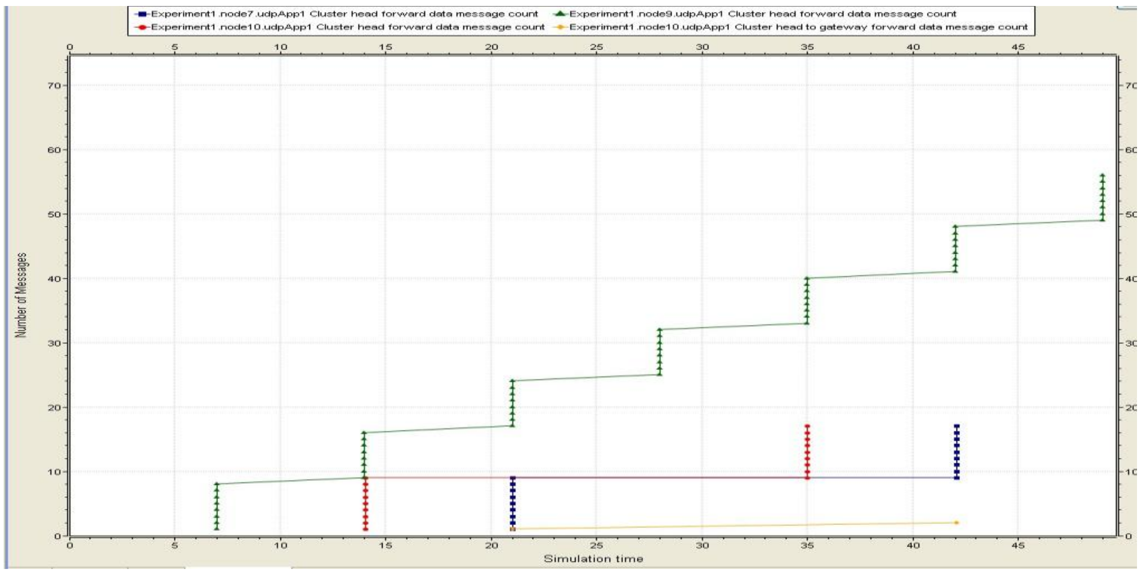

Figure XII: Data Messages Forwarded by Cluster-heads

The green line shows the messages forwarded by node9 (source cluster-head), the red line represents messages forwarded made by node10, the blue line corresponds to node7 and the yellow line at the bottom represents messages forwarded by node7 as an intermediate cluster.

The simulation was carried out for 50 logical seconds. With node2 transmitting every seven seconds we have seven transmissions of data message.

From the graph it quite evident that for transmissions 1, 4 and 7 the destination cluster was cluster 2 itself and the messages were not forwarded to any other cluster. Cluster 3 is the destination cluster for the second and fifth transmissions and cluster 1 is the destination for the third and sixth transmissions.

From the yellow line at the bottom of the chart, we can observe that node10 has simply acted as a forwarding node for the third and sixth transmissions (it does not process the data messages for these transmissions). The yellow line shows the two cluster-head to gateway forward data messages transmitted at around the 21st second and 42nd second.

This plot validates the correctness of this routing technique and shows how the algorithm can forward the data messages to correct destination even with a complex layout of nodes.

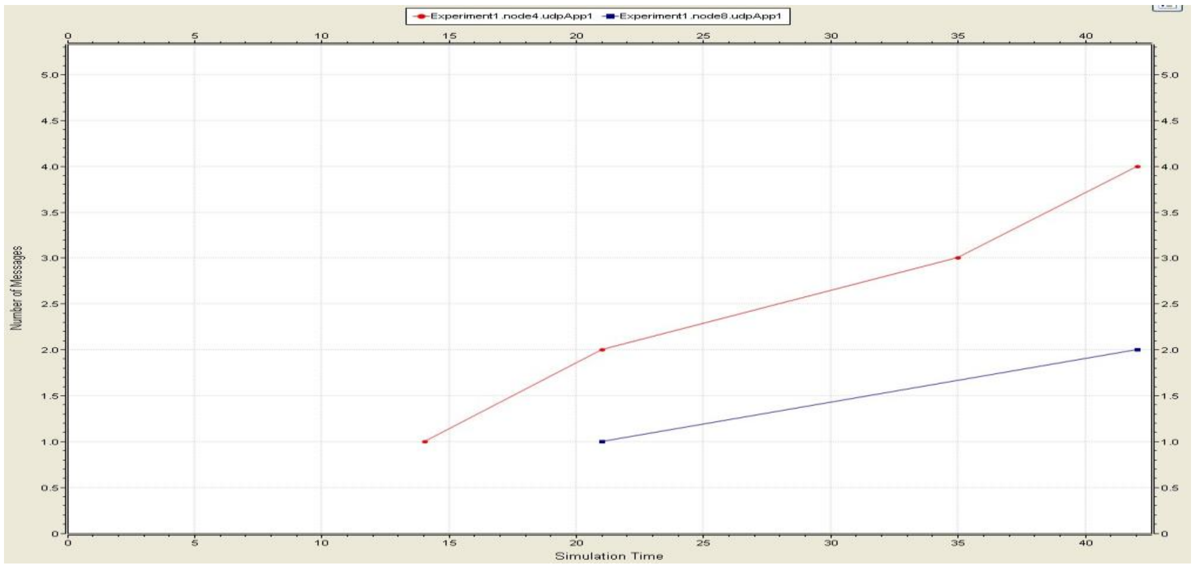

Figure XIII: Gateway Forward Data Messages

Figure XIII shows the number of data messages forwarded by the gateways node4 and node8. Node4 being the only gateway to cluster 2 (source cluster) it forwards all messages to external clusters if the destination cluster is different from source cluster. Since cluster 1 and cluster 3 were destination clusters for two data messages each, node4 forwards all these messages to the other clusters. With node8 being the only gateway for cluster 1 and cluster 1 being the destination cluster for only two data messages, it forwards only two of these data messages. The corresponding forwarding times also confirm that the gateways forwarded the data messages to the destination clusters at the intended times.

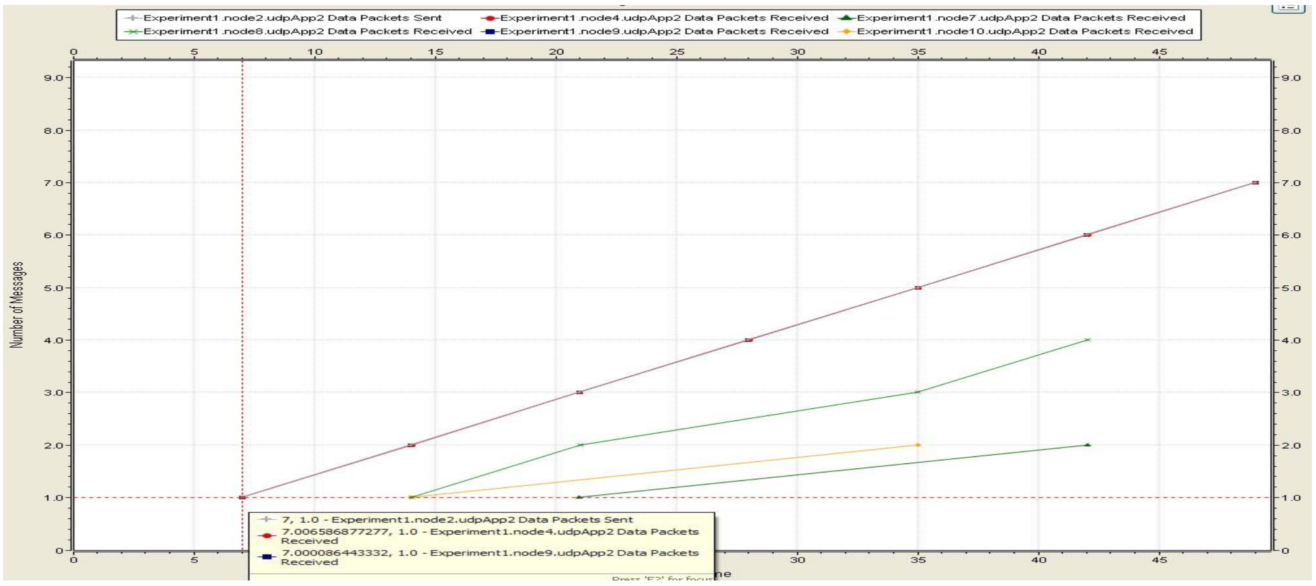

Figure XIV: Data Messages Sent and Received

This graph plots the times at which data packets were sent by the simple UDP application in node2 and received by the simple UDP application in other nodes. Node2 transmits data messages every seven seconds starting from the seventh second. Node9 receives the message from node2 processes it and retransmits it to node4. Since this takes place fairly quickly the times of node2, node9 and node4 are overlapping. The small white box in the screenshot shows the individual times for these nodes. Node7 and node10 receive two messages each, the reason being discussed earlier. Node8 being a gateway between cluster 3 and cluster 1, belongs to both these clusters and hence receives data messages intended for both cluster 3 and cluster 1 and thus received four messages.

This experiment clearly illustrates the effectiveness of the protocol in routing messages across nodes in the MANET. The protocol is able to route messages without any considerable delay. Moreover, when the nodes are stationary, the protocol is able to achieve a 100 percent success rate in routing messages. We could also observe that all messages reached the intended destinations correctly and the protocol also ensured that the messages were never read by nodes in those clusters that were not the intended recipients.

Experiment II

Experiment II is another simple experiment that demonstrates the use of gateway change messages. Gateway change messages are sent by gateway nodes to cluster-heads when they get disconnected from one of the clusters they were already connected to. The decision to send a gateway change message is taken when the gateway does not receive any cluster-head declaration messages within a predefined timeout period.

In this experiment there are six nodes divided into 3 clusters. Three nodes are cluster-heads and are stationary. The other 3 nodes simply circle their respective cluster-heads. When these nodes reach a region where the transmission ranges of two cluster-heads overlap they become gateways and when they leave these regions they send gateway change messages indicating they are not gateways anymore.

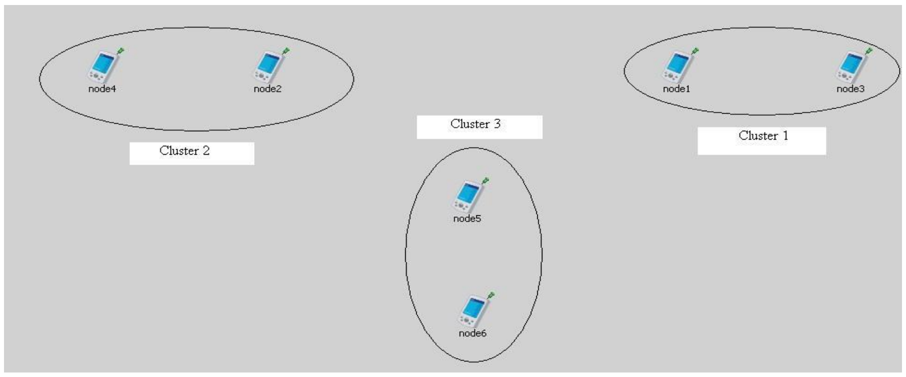

Figure XV: Experiment II Clusters

The gateway nodes move in a counter-clockwise fashion at a speed of 10 m/s. In this experiment node2 sends broadcast messages at seven second intervals. The experiment was simulated for 100 logical seconds.

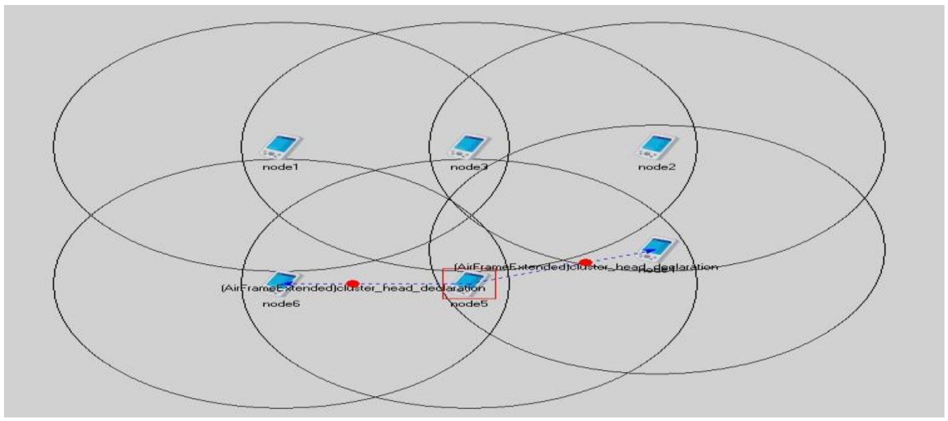
node1
node3
node2
(AirFrameExtended)cluster_head_declaration
(AirFrameExtended)cluster_head_declaration
node6
node5

Figure XVI: Start of Simulation

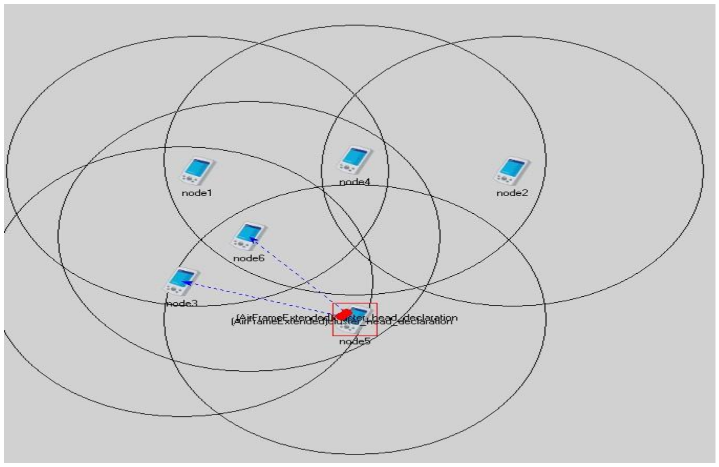
node1
node4
node2
node6
node3
(AirFrameExtended)cluster_head_declaration
(AirFrameExtended)cluster_head_declaration
node5

Figure XVII: Simulation at Time 50 s

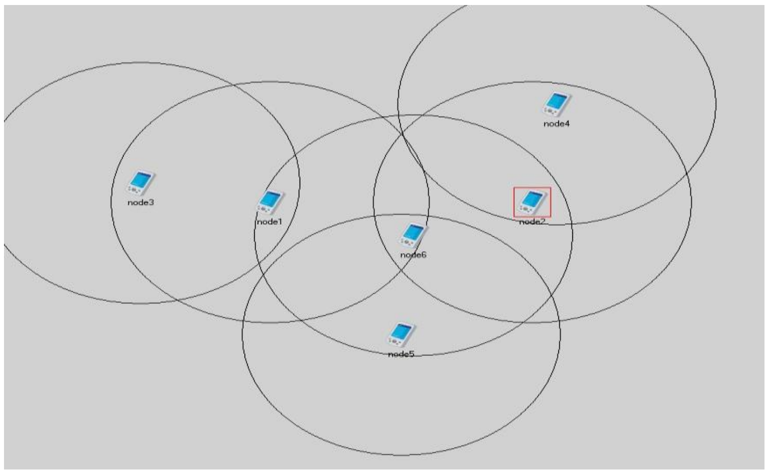

Figure XVIII: End of Simulation

The screenshots above show the movement of nodes during the simulation.

The introduction of node mobility complicates the system several folds. The UDP has a high percentage of message losses and when nodes become mobile this percentage increases rapidly.

The predefined timeout period was increased to 7 seconds, i.e., the gateways wait for 7 seconds to get a cluster-head message before deciding whether or not to send a gateway change message. Even with additional measures to ensure efficiency we do not obtain a 100 percent success rate in this experiment.

The results are summarized below:

a) Node2 sends 14 broadcast messages.
b) Node4 which is in the same cluster as node2 receives all the 14 messages.
c) Node6 which is a circulating gateway receives only 7 messages. Of these only 6 messages are received by node5 which is the cluster-head of node6.
d) Node3 which is another circulating gateway receives only 5 messages of which only 3 reach node1 which is the cluster-head of node3.
e) Node3 also gets disconnected from node1 for a short while even though it never goes out of its transmission range.

These results clearly indicate that the packet loss is very high due to mobility. When the gateways come close to each other it could also cause packet collisions leading to losses. Moreover node positions at different points of time play a major role in determining the successful forwarding of messages. If nodes fall in “dead zones” for long periods of time where they are outside the transmission range of all nodes except their cluster-heads it could lead to several packets being lost.

When the speed of these nodes was reduced to 5 m/s the results did not improve much because the circulating nodes remained as non gateways for a longer time because they did were outside overlapping transmission ranges. But the messages from the gateways to the cluster-heads improved dramatically (100 percent success rate) because they remain connected for a longer time.

Moreover, the OMNeT++ simulation engine cannot always accurately simulate a real-world system due to the discrete event nature of the simulation engine. This causes several nodes to transmit messages at exactly the same logical time, creating a higher probability for packet collisions leading to loss of messages.

But such issues are highly common with any MANET protocol due to the inherent shortcomings of UDP exacerbated by mobility. The aim here is not to completely overcome all these shortcomings but to design a simple and efficient algorithm for broadcasts and multicasts in MANETs by using as few control messages as possible to set up the routing infrastructure. With the use of acknowledgment messages things can be improved dramatically but that would only lead to a message explosion in the system and beat the essence of what we intend to achieve. Users can incorporate some acknowledgment-like system in their application if every message being transmitted is of utmost importance.

Experiment III

The third experiment was performed to compare how the algorithm compared with traditional MANET routing algorithms and with other clustering algorithms. In this experiment, the field has a large number of nodes (26 nodes) divided into seven clusters. Some clusters are disjoint from the field and move towards the main field of nodes and join one large connected set of nodes. Thus the nodes are mobile for about 10 seconds after which they are stationary.

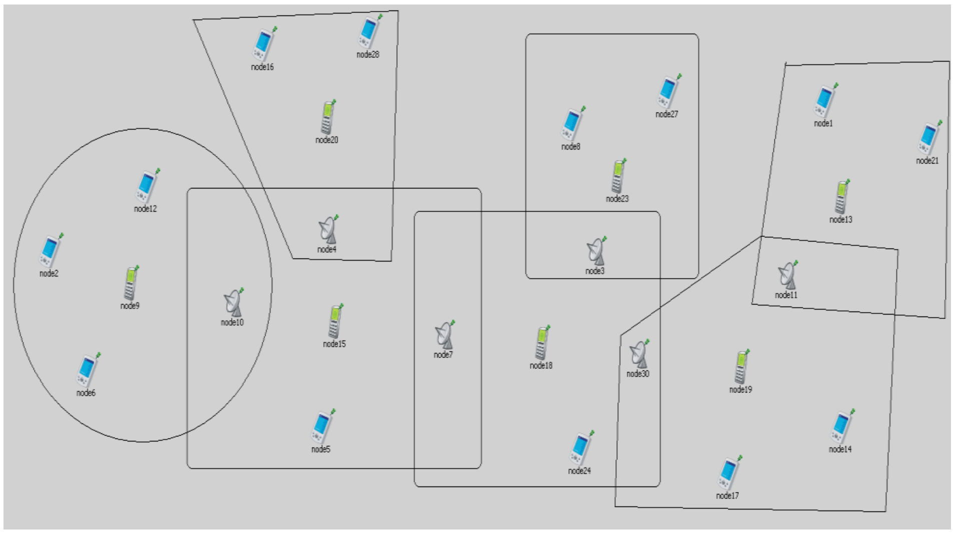

Figure XIX: Experiment III Node Layout

The boxes and circles indicate the various clusters that were formed after the initial motion of some of the nodes.

A graph was then plotted for the total number of control messages that were transmitted and received for route setup. Here each set of message transmissions was considered one message, as we cannot precisely implement broadcasting in OMNeT++. The actual number of packets transmitted will be higher because the nodes try to transmit one packet for each adjacent node. But since this is the case across all the protocols used for this experiment can be ignored.

The total number of messages transmitted for the other protocols was calculated based on their algorithm description. The comparison was made with OLSR which sends “hello” messages, Topology Control messages and Host and Network Control messages as control messages. The number of messages transmitted over a period of time was calculated. Similarly, total number of control messages was calculated for other clustering algorithms (The algorithm described in [1] was used as the sample algorithm). These numbers can be taken in common for all the clustering algorithms as they have a similar methodology of execution during the initial stages.

Here we only compare the number of messages transmitted for the first 50 seconds of simulation which has 10 seconds of mobility.

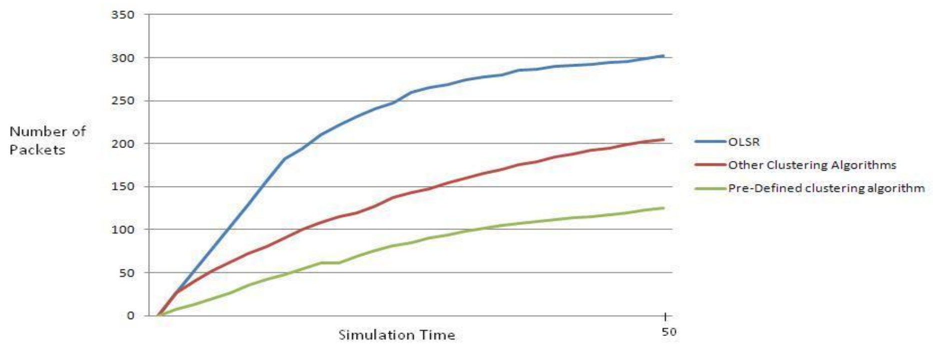

Figure XX: Comparison between Routing Algorithms

We can observe from this plot that OLSR generates a lot of messages (~130 messages) in the initial stages (first ten seconds) when the nodes are mobile.

During this period, the other clustering techniques also generate a lot of messages (~75 messages) during the first 10 seconds. This is primarily because the nodes have to elect cluster-heads and gateways. The election process is further complicated by the mobility of these nodes.

The proposed algorithm produces far lesser number of messages (~35 messages) during this period. The number of messages generated is consistently increasing for this technique but the rate of increase is far lesser compared to the other algorithms.

But after the nodes become stationary we find that number of control messages produced by OLSR is gradually decreasing and the curve almost flattens towards the end the simulation.

For the other clustering algorithms, the messages reduce but at a lesser rate. The rate of control messages produced is almost equal to the number of messages produced by the proposed algorithm (The two curves almost become parallel after the initial few seconds).

The curve for the proposed routing algorithm increases steadily even during this period, but the as mentioned earlier this increase is far slower in nature.

The reason for the change in OLSR is because once the nodes are stationary the established routes do not become stale too often. Hence the need to establish fresh routes does not arise. But when the nodes are mobile the number of hello messages passed around increases dramatically.

For the other clustering algorithms, the initial burst of messages is inevitable because of their in-built election strategy. The movement of nodes worsens the situation and hence the number of messages they produce is almost same as that of OLSR. When the nodes become stationary and routes become established they generate lesser number of messages.

For the proposed algorithm, the cluster-head declaration messages are sent at regular intervals irrespective of whether the nodes are mobile or stationary. When the nodes are mobile it generates a few more gateway_change messages and hence there is a marginal increase in the number of messages generated when the nodes are mobile.

Critics will argue that over a long period of time the proposed algorithm will generate more messages than the other algorithms but one must remember that this time frame will be very large. Also, the curves for the other algorithms flatten mainly because the nodes become stationary. If the nodes continue to be mobile they will continue to generate a large number of messages.

To improve this situation, the proposed algorithm can be configured in such a way that the cluster-heads produce cluster-head declaration messages at longer intervals. This will generate lesser gateway declaration messages as response and the number of messages generated will reduce dramatically. This can adversely affect the gateway selection as cluster-heads may choose gateways which are no longer connected to transmit their messages. This may cause lot of message losses when the nodes are mobile. Thus a fine balance has to be achieved between the transmission interval and the node mobility.

As future work, the proposed algorithm can be optimized in such a way that it can overcome the problem mentioned above. The cluster-head declaration message interval can be increased when the nodes are stationary and the decreased when the nodes become mobile. Thus when the nodes are mobile the cluster-heads send cluster-head declaration messages more often thus keeping their gateway list as fresh as possible and when the nodes become stationary the probability of a gateway disconnecting from a cluster is negligible and the cluster-head declaration messages can be sent as infrequently as possible. The same logic can be used when the nodes operate in adverse conditions where they have a higher probability of getting destroyed or work with varying levels of battery power where they need to regulate their usage judiciously.

CHAPTER VII

CONCLUSION

This work proposed a novel cluster routing technique for mobile ad-hoc network systems exclusively for the transmission of broadcast and multicast messages. The routing technique focuses on minimizing the number of control messages that are passed between nodes in order to setup their routing tables by using some predefined clustering information. Cluster-heads are predefined thus avoiding the need for excessive message passing to elect cluster-heads.

This work also discusses other prominent work in this area, especially in field of cluster routing algorithms. The routing technique and its implementation were discussed in detail and simple experiments were performed to illustrate the actual operation of the algorithm in the OMNeT++ discrete event simulator.

The results of these experiments showed the effectiveness of this routing algorithm and also some of its shortcomings which were common to most other MANET routing techniques. Some improvements that can be made to the algorithm is the use of acknowledgement messages for high priority data messages but this needs to be implemented taking into consideration the criticality of the application and weighing it against the performance of the system.

Currently each cluster-head contains only one gateway for each cluster it can communicate with. Improvements can be made to allow multiple gateways between two clusters to allow better load balancing between these gateway nodes. Since power is a major factor for mobile ad-hoc nodes load balancing can improve the life of a node dramatically. Smart gateway selection heuristics can be used to dynamically choose gateways that allow better load balancing.

At this point communication between nodes within the same cluster is single hop, i.e. nodes need to be located directly within the transmission range of cluster-heads. For very large clusters with hundreds of nodes this may become infeasible. To address this issue, this algorithm could be run on top of traditional MANET routing protocols such as AODV, OLSR or DYMO where nodes within the same cluster can communicate in multiple hops but communication between clusters will occur through the gateways proposed in this paper. This way we can combine the best practices of both the techniques to obtain an optimized algorithm for MANET routing.

Assuming there are 100 nodes in the network with 10 clusters with each cluster containing 10 nodes. Using OLSR directly on all the nodes the number of "hello" message transmissions will become exponential with each node trying to communicate with each other. Using clustering technique discussed here will be better but it would be hard for all the 10 nodes in a cluster to be within a single hop distance from their cluster-heads. By using the clustering technique on say OLSR we can reduce the number "hello" messages because each node in a cluster will try to setup a lower level MANET within each cluster containing only 10 nodes. Some of the nodes in the smaller MANET will be gateways and will pass on messages destined for external clusters appropriately.

Thus an effective and efficient MANET routing technique was proposed in this work that uses the clustering approach for establishing a routing infrastructure. The technique is customized to suit certain kinds of applications and may not be apt for all MANET systems but overcomes the overhead of excessive message passing to establish the MANET routing infrastructure.

REFERENCES

[1] T. Shivaprakash, C. Aravinda, A.P. Deepak, S. Kamal, H.L. Mahantesh, K.R. Venugopal, L.M. Patnaik. Efficient Passive Clustering and Gateway Selection in MANETs. *University Visvesvaraya College of Engineering, Indian Institute of Science, Bangalore, India.*

[2] A.D. Amis, R. Prakash, T.H.P Vuong, D.T. Huynh. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. *Department of Computer Science, University of Texas at Dallas, Richardson, Texas.*

[3] V.S. Anitha, Dr. M.P. Sebastian. SCAM: Scenario-based Clustering Algorithm for Mobile Ad Hoc Networks. *Department of Computer Science, National Institute of Technology, Kerala, India.*

[4] Y.L. Chang, C.C. Hsu. Routing in Wireless/Mobile ad-hoc networks via Dynamic Group Construction. *Department of Computer Science and Information Engineering, National University of Taiwan, Taipei 106, Taiwan.*

[5] I. Inn ER, W.K.G. Seah. Mobility based d-Hop Clustering Algorithm for Mobile Ad Hoc Networks. *Institute of Infocomm Research and national University of Singapore, Singapore.*

[6] V. Bricard-Vieu, N. Mikou. Clustering Algorithm for MANETs based on mobility prediction. *3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications*. March 2005. Tunisia.

[7] A.M. Mahdy, J.S. Deogun, J. Wang. Towards Scalable Clustering of Infrastructured Mobile Ad Hoc Networks. *Computer Science and Engineering Department, University of Nebraska, Lincoln, NE 68588*.

[8] N. Chatterjee, A. Potluri, A. Negi. A Self-Organizing Approach to MANET Clustering. *Department of Computer and Information Sciences, University of Hyderabad, India.*

[9] *en.wikipedia.org/wiki/List_of_ad-hoc_routing_protocols*

[10] *www.omnetpp.org*

[11] *inet.omnetpp.org*

[12] http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml

[13] http://awds.berlios.de/

[14] D. P. Bertsekas and R. G. Gallager. Distributed Asynchronous Bellman-Ford Algorithm. Data Networks, pg. 325-333. Prentice Hall, Engelwood Cliffs, 1987.

[15] Y. Z. Lee, M. Gerla, Jason Chen, Jiwei Chen, B. Zhou and A. Caruso. Ad Hoc & Sensor Wireless Networks, Volume 2, Number 2, 2006.

[16] C. C. Chiang, H. K. Wu, W. Liu and M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. *IEEE Singapore International Conference on Networks, SICON '97, pg. 197-211, Singapore, 16-17, April 1997, IEEE.*

[17] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers. *Proc. of the SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications, Aug 1994, pg. 234-244.*

[18] G. Pei, M. Gerla, X. Hong and C. C. Chiang. A Wireless Hierarchical Routing Protocol with Group Mobility. *IEEE WCNC '99, New Orleans, USA, September 1999.*

[19] M. Gerla and J.T. Tsai. Multicluster, Mobile, Multimedia Radio Network. *ACM Wireless Networks, VOL 1, No. 3, 1995, pg. 255-265*.

[20] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti and L. Viennot, T. Clausen. *Optimized Link State Routing Protocol. RFC 3626. http://www.olsr.org*.

[21] K. Grace. Mobile Mesh Routing Protocol (MMRP). *http://www.mitre.org/work/tech_transfer/mobilemesh/*

[22] R. Guimaraes and Ll. Cerda. Improving reactive routing on wireless Multirate ad-hoc networks. In: Proceedings of 13th European Wireless 2007.

[23] S. Khurana, N. Gupta and N. Aneja. Reliable Ad-hoc On-demand Distance Vector Routing Protocol. *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06).*

[24] N. Kettaf, A. Abouaissa, T. Vudong and P. Lorenz. *http://tools.ietf.org/html/draft-kettaf-manet-acor.*

[25] M. Marina and S. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. *Proceedings of the 2001 IEEE International Conference on Network Protocols (ICNP), pg. 14-23, IEEE Computer Society Press, 2001.*

[26] A. C. Valera, Winston K. G. Seah and S. V. Rao. Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad hoc Networks. *Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), Mar 30- Apr 3, 2003.*

[27] I. Chakeres and C. Perkins. Dynamic MANET On-demand Routing Protocol (DYMO), Internet Draft, *http://tools.ietf.org/html/draft-ietf-manet-dymo.* RFC 4728.

[28] Z. J. Haas, M. R. Pearlman and P. Samar. The Interzone Routing Protocol (IERP) for Ad Hoc Networks, Internet Draft, *http://tools.ietf.org/html/draft-ietf-manet-zone-ierp.*

[29] V. Park and S. Corson. Temporally-Ordered Routing Algorithm (TORA). *Version 1 Functional Specification, Internet Draft, IETFMANET Working Group, June 2001.*

[30] N. Nikaein, H. Labiod and C. Bonnet. Distributed Dynamic Routing Algorithm (DDR) for Mobile Ad Hoc Networks. *In proceedings of the MobiHOC 2000, First Annual Workshop on Mobile Ad Hoc Networking & Computing.*

[31] S. Ruhrup. Theory and Practice of Geographic Routing. *Department of Computer Science, University of Freiburg, Germany.* Feb 2009.

[32] S. Singh, M. Woo and C.S. Raghavendra. Power-Aware Routing in Mobile Ad Hoc Networks. *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking. pg. 181-190. 1998.*