8-8-2002

Fault/configuration management for wireless ad-hoc network

Abhay Doshi
Florida International University

DOI: 10.25148/etd.FI15101219

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

FAULT/CONFIGURATION MANAGEMENT FOR WIRELESS AD-HOC NETWORK

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Abhay Doshi

2003

To: Dean Vish Prasad
College of Engineering

This thesis, written by Abhay Doshi, and entitled Fault/Configuration Management for Wireless Ad-Hoc Networks, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Kia Makki

Subbarao Wunnava

Niki Pissinou, Major Professor

Date of Defense: August 8, 2002

The thesis of Abhay Doshi is approved.

Dean Vish Prasad
College of Engineering

Dean Douglas Wartzok
University Graduate School

Florida International University, 2003

DEDICATION

To my parents and friends.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Niki Pissinou, for having faith and confidence in my abilities. I would also like to thank her for the direction and motivation that she constantly provided.

I would like to thank my committee members Dr. Kia Makki and Dr. Subbarao Wunnava, for guiding me and providing me with constant feedback.

The following people have read draft versions of the thesis: Dr. Gang Feng, Mayra Rosales, Saurabh Agrawal, Nripjeet S Reen and Dr. Niki Pissinou

I would also like to thank Dr. Leonard E. Miller for his help and guidance on the implementation of Ad-hoc networks in OPNET.

Needless to say, I am grateful to all of my colleagues at the Information Technology and Telecommunication Department for their support (and tolerance).

I would like to thank all of my friends, who were always patient, understanding and helpful.

ABSTRACT OF THE THESIS

FAULT/CONFIGURATION MANAGEMENT FOR WIRELESS AD-HOC NETWORKS

by

Abhay Doshi

Florida International University, 2003

Miami, Florida

Professor Niki Pissinou, Major Professor

An ad hoc network is maintained by the combined efforts of all the mobile nodes themselves, who often operate under severe constraints, such as limited battery power, variable link quality, and limited storage capacity. As a result, there is a growing need for enhanced fault and configuration management solutions to help in tracking problems as well as solving them.

Viable network architecture for a wireless ad-hoc environment, which takes advantages of both hierarchical and distributed architectures, has been investigated. A complete design solution is proposed which makes ad-hoc environments less susceptible to faults. Results shows that by applying the proposed power saving technique, network load due to control traffic may be significantly reduced. Based on other gathered statistics, we can set the optimal value of maximum number of nodes allowed in a cluster for efficient performance to be 35 for a specific scenario.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1:

Introduction

1.1 Ad Hoc Networks

Ad-hoc networks are network architectures that can be rapidly (ideally immediately) deployed and that do not need to rely on a pre-existing infrastructure. The salient feature of this breed of networks is that they can operate in different and differing propagation and network operational conditions, which cannot be predicted during the network design stage [1].

These types of networks are used in situations where temporary network connectivity is needed, such as in disaster relief or battle site networks. In the disaster relief scenario, an ad hoc network would enable medics in the field to retrieve patient history from hospital databases (assuming that one or more of the nodes of the ad hoc network are connected to the Internet), or allow insurance companies to file claims from the field. The network is maintained by the combined efforts of all the mobile hosts themselves, often operating under severe constraints, such as limited battery power, variable link quality, and limited storage capacity.

As a result, we need to have an enhanced network management solution, which helps in tracking the problems, as well as solving them. We are targeting our work towards the problem in management of ad hoc networks.

1.1.1 Network Management in Ad Hoc Networks

By definition, "network management" is a process of controlling a complex data network so as to maximize its efficiency and productivity. This process involves data

collection, data processing, data analysis, and problem fixing. Network management can be functionally divided into five areas defined by the International Standards Organization: fault management, configuration management, security management, performance management, and accounting management [2].

Fault management involves discovering, isolating, and fixing problems in the network. This component of network management is responsible for ensuring the smooth and continued operation of the network. Configuration management involves initialization and shutdown of the network. It also involves the maintenance, addition, and updating of new network components. Part of the configuration module's function involves defining the classification. of relationships between network entities.

Security management controls access to network components and information. It is also responsible for implementing encryption and decryption schemes for secure end-to-end communication. Performance management involves collecting network statistics and tuning the network to improve performance. Accounting management tracks network utilization by various users and groups. This information can be very useful in network configuration and the allocation of network resources to various groups in an organization [2].

Managing Ad-hoc networks is more intricate than managing wired or wireless fixed network due to the level of complexity in managing Ad-hoc networks. Some of complexities are as follows

(1) Each and every node of an ad hoc network can range in complexity from simple sensors located in the field, to fully functional computers such as laptops. An

insinuation of this diversity is that not all nodes will be able to contribute equally to the management task. For instance, it is likely that sensors and small personal digital assistant (PDA)-type devices can contribute minimally to the task of management, while more powerful machines will need to take on responsibilities such as collecting data before forwarding it to the management station, tracking other mobiles in the neighborhood as they move, etc. Thus, the management protocol needs to function in very heterogeneous environments [4].

(2) One task of network management is to reveal the topology of the network. In wireline networks, this is a very simple task since changes to the topology are very infrequent. In mobile networks, on the other hand, the topology changes very frequently because the nodes move about constantly. Thus, the management station needs to collect connectivity information from nodes periodically. An implication of this is an increased message overhead in collecting topology information [4].

(3) Most nodes in ad hoc networks run on batteries. We therefore need to ensure that network management overhead is kept to a minimum, so that energy is conserved. For example energy is consumed by a radio when a packet is transmitted or received (the DEC Roam About radio consumes approximately 5.76 W during transmission, 2.88 W during reception [3]). In addition, the CPU expends energy in processing these packets. Thus, we need to reduce the number of packets transmitted/received/processed at each node. This requirement is contradictory to the need for topology update messages previously discussed.

(4) Energy constraints and mobility can result in the network becoming partitioned frequently. For example, nodes may power themselves off to conserve energy

resulting in partitions, or a node may move out of one cell to other. Similarly, a node may die when its battery runs out of power. That node could have been handling the data collection center for its subnetwork. In all these cases, the partitioned subnetworks need to continue running independently, and the management tool must be robust enough to adapt all these changes, while still working efficiently. For example,[4] when the network gets partitioned, the management tool must quickly ascertain that the partition has occurred and re-configure the subnetwork(s) to function independently. In addition, when partitions merge, the management tool must be able to update the network view without too much of an overhead.

(5) Signal quality can fluctuate significantly in wireless environments. Thus, fading and jamming may result in a link going down sporadically. An effect of this is that the network topology from graph theoretic point-of-view changes, but the physical layout of the network may be the same as before. The management tool must be able to distinguish this case from the case when node movement causes topology changes, because a routing table and all the routing decisions are based on the network topologies from a graph theoretic point-of-view. It may not be necessary to exchange topology update messages at all. In order to be able to do this, the management tool (which inhabits the application layer) must be able to query the physical layer. This obviously violates the layering concept of OSI [4], but it results in enormous savings.

(6) Ad hoc networks are frequently set up in hostile environments and are therefore subject to eavesdropping, destruction, and possibly penetration. Thus, the

management protocol needs to incorporate encryption, as well as sophisticated authentication procedures.

An ad hoc network therefore needs a network management solution that can handle all the complexities and does not enforce more overheads on the network. Due to the diverse application, an ad hoc network desires a solution that can make it self-healing.

The aspiration of this thesis is to simulate a tool for network management of wireless ad-hoc networks, and endeavor to make the network self healing. It attempts to tackle all the areas of network management, and consequentially attempts to solve the problems mentioned above, by using different algorithm.

1.2 Scope of the thesis

This thesis designs a complete solution to make ad hoc network self-healing. In the process of designing the scheme to make the network self healing we propose a new network architecture that helps ad hoc networks to improve its performance by categorizing nodes in three different levels, which are node, cluster head, and MOM.

This thesis improves on existing geographical clustering algorithm by finding the threshold values. By considering these values, we can improve the performance of the ad hoc network by reducing the node discovery time and end-to-end delay. The proposed power saving algorithm also reduces the total power consumption by the network and increases the network lifetime.

In this thesis, we have defined a cluster head determination, cluster generation and cluster maintenance algorithm, in detail. Finally, each and every possible case of

faults in ad hoc environment has been discussed, and solutions are proposed individually.

1.3 Overview of thesis

Chapter 2 looks at the theoretical background of different fields in ad hoc networks. The various studies that have been done in the fields of network architectures, ad hoc networks, fault management and power management in ad hoc networks are discussed in this chapter, which provides background information needed to better understand the thesis.

Chapter 3 discusses in depth design of the fault management solution case by case. This chapter also explains proposed network architecture, extensions to the existing clustering algorithm, power saving schemes for ad hoc networks, and case-by-case discussion of possible faults in ad-hoc networks with their solutions

Chapter 4 explains the methodology adopted to simulate the ad hoc environment in OPNET. It also demonstrates the produced results and an analysis of those results.

In chapter 5, the effectiveness of the fault management system is discussed. Possible flaws are discussed and the scope of future work on various parts of the system is explained.

Chapter 2:

Previous Research

A "mobile ad hoc network" (MANET)[11] is an autonomous system of mobile routers (and associated hosts) connected by wireless links, the union of which forms an arbitrary graph. The routers are free to move randomly and organize themselves arbitrarily, allowing the network's wireless topology to change rapidly and unpredictably. This new networking concept defines a simple mechanism, which enables mobile devices to temporarily communicate without any planned infrastructure or any human intercession [12].

The idea is to make a network independent of any preexisting infrastructure or any centralized inspection. This is possible only because a host can act as a router at the same time. Since networks are deployed "on the fly", they find direct application in the field of military, disaster relief, and emergency rescue situations. There are some more useful applications like network sensors, connectivity in campuses, and "on the fly" networks for conferences as mentioned in [13].

While MANET makes many new applications possible, it also poses many challenging problems such as the fault management of the networks that are formed "on the fly".

2.1. Research topics in Ad hoc environment

An ad-hoc network is a very contemporary concept, so there is a lot of work to be done. Consequently, there is work going on in related fields such as ad-hoc routing protocol, clustering algorithms for ad-hoc, multicast and mobility management, QoS

issues in Ad-hoc networks, power control in ad-hoc networks, and mobile agents for ad-hoc network.

The most overlooked topics are management in ad-hoc networks, and security in ad-hoc networks. In the following subsection, we will give a brief introduction of the above research topics.

2.1.1. Routing in Ad-hoc

Development of the solution for network management is based on the assumption that there is a routing mechanism running beneath it. This mechanism takes care of the delivery of packets from one node to another. There are several routing protocol suites specifically designed for ad hoc routing, as well as more traditional protocols, such as link state and distance vector, used for dynamic networks. Some of the key observations are as follows: Proactive, shortest path protocols provide excellent performance in terms of end-to-end delays and packet delivery fraction although at the cost of higher routing load. On-demand protocols suffer from sub-optimal routes as well as decreased packet delivery fraction because of more dropped data packets. However, they are significantly more efficient in terms of routing load.

The multipath protocol TORA did not perform well in past, in spite of maintaining multiple redundant paths. The overhead of finding and maintaining multiple paths seems to outweigh the benefits. Plus, the end-to-end delay performance is poor due to the loss of distance information. The routing load differentials between all routing protocols are reduced with a larger number of peer-to-peer conversations in the network, yet the other performance different differentials

are not affected conclusively. The rate of mobility and network size do not seem to affect the performance beyond what is normally expected, such as higher routing load, increased delay and dropped packets [15].

2.1.2. Clustering algorithm for ad-hoc

There are some routing algorithms that use the clustering approach, but we are not at all considering the routing algorithm here. The clustering algorithm we use assumes that a routing algorithm is already running irrespective of the clustering algorithm. There is substantial work done in this area. For instance, in [3][40], the author discusses two clustering algorithms: the graph-based clustering algorithm and the geographical clustering algorithm. The first algorithm models the ad hoc network as a graph and forms clusters based on the graph topology. Second algorithm uses the global positioning system (GPS) to create clusters. Both algorithms have applicable situations and assumptions. The performance is analyzed for their design and is followed by the issues about the cluster maintenance.

According to this paper [3], even in a graphical clustering algorithm there are two ways to implement the algorithm. First, when we use only ping to measure the topology changes, the number of nodes that remain unmanaged is very high, at almost 50-70%. When we use Mac layer connectivity information along with ping, this number remains below 10%. In geographical clustering, the number of nodes that remain unmanaged is very low and constant, but the message cost increases with the speed.

Finally, this paper concludes that the message cost of maintaining a cluster using the graphical clustering, is comparatively lower than using geographical clustering, but the number of nodes that remains unmanaged is very low in geographical clustering (less then 10%)when compared to graphical clustering (10-20% with MAC and 50-70% without MAC). However in situations where the nodes have widely different transmission ranges, it is very difficult to use geographical clustering [39].

We have made some changes to the algorithm in order to attain a balance for the different architectural approach we are using, and to make it more fault tolerant.

2.1.3. Multicast and Mobility management

Ad hoc networks are deployed in applications such as disaster recovery and distributed collaborative computing, where routes are mostly multihop, and network hosts communicate via packet radios. In a typical ad hoc environment, network hosts work in groups to carry out the given task. Hence, multicast plays an important role in ad hoc networks. Multicast routing protocols used in static networks e.g., Distance Vector Multicast Routing Protocol (DVMRP) [36], Multicast Open Shortest Path First (MOSPF) [33], Core Based Trees Routing Protocol (CBT) [34], and Protocol Independent Multicast (PIM) [35], do not perform well in ad hoc networks.

Multicast tree structures are fragile and must be readjusted continuously as connectivity changes. Furthermore, multicast trees usually require a global routing substructure such as a link state or distance vector. The frequent exchange of routing vectors or link state tables, triggered by continuous topology changes, yields excessive channel and processing overhead. Limited bandwidth, constrained power,

and mobility of network hosts make the multicast protocol design particularly challenging on the effectiveness and efficiency of On-demand Multicast routing Protocol (ODMRP) [32].

2.1.4. Power control in ad-hoc networks

Mobile ad-hoc networking involves peer-to-peer communication in a network with a dynamically changing topology. Achieving energy efficient communication in such a network is more challenging than in cellular networks since there is no centralized arbiter such as a base station that can administer power management. Power control helps combat long term fading effects and interference. When power control is administered, a transmitter will use the minimum transmit power level that is required to communicate.

There are some solutions proposed to deal with this issue, one of which is the COMPOW protocol for power control in ad hoc networks [37]. The solution proposed would simultaneously satisfy the three objectives of maximizing the traffic carrying capacity of the entire network, extending battery life through providing low power routes, and reducing the contention at the MAC layer. A shortcoming of this protocol is that, the common power strategy may settle to an unnecessarily high power level when the nodes in the network are clustered. Even a single node outside a cluster can force all the nodes in the network to use high power levels. In another approach [38], they have proposed a power control loop, analogous to those generally used in CDMA networks. For ad hoc wireless networks, "Their approach shows that

this power control loop reduces energy consumption per transmitted byte by 10%-20%. Furthermore they show that it increases throughput by 15%"[38].

2.1.5. Mobile agents for ad-hoc network

A Mobile agent [21] is an emerging technology attracting interest from the fields of distributed systems, and mobile computing, among others. The mobile agent technology has the potential to provide a convenient, efficient and robust programming paradigm for distributed applications, even when partially connected computers are involved. A mobile agent is a piece of software that can migrate from one computer to another during its execution. Mobile agents can also communicate with each other, clone, merge, and coordinate their computations. Mobile agents are autonomous agents in the sense that they control their relocation behavior in pursuit of the goals with which they are tasked. These properties are additional reasons that make mobile agents good candidates to be used in a loosely coupled network environment [6].

2.2. Design & Architecture Challenges

The three main challenges in the design and operation of the ad-hoc networks are [16][17][19]:

- The lack of a centralized entity.
- The possibility of rapid change in topology.
- The fact that all the communication is carried over wireless, and wireless already has its specific limitations [18].

While considering architectures for setting up an ad-hoc network, there are three basic, possible architectures, which are the centralized approach, and distributed approach, hierarchical approach.

2.2.1. Centralized approach to Network Management

We have learned the requirements and limitations of ad-hoc networks and note that there is no central entity. As discussed in [15][20], a centralized management system has a single management station that collects the management information from all the nodes and controls the entire network. This is an easy implementation that has some potential problems like a single point of failure. In addition, this architecture suffers from the high message overhead, as the data collection is done at a single point. The centralized architecture is not widely used, except in particular situations where centralized control over the network activity is needed.

2.2.2. Hierarchical approach to Network Management

Some prior work in the Ad-hoc network management field considers this architecture to be the best-suited architecture for ad-hoc network management architecture [15]. This architecture uses intermediate managers to distribute the management tasks. All of these intermediate managers have their own domain (cluster). These intermediate mangers themselves are part of that domain and can perform partial management functionality. A shortcoming of this architecture is that it has a lot of centralize control and no intermediate manager-to-manager communication [22][24].

2.2.3. Distributed approach to Network Management

Much of the prior work done in ad-hoc network management reflects that distributed approach performs well in ad hoc environment. Both distributed network management and ad hoc network management approaches do not have a centralized entity, in contention to their counter parts. Author debates that due to the very frequent changes in topology in ad-hoc, the distributed approach is best suited for the network management of an ad-hoc network [17][16].

What we are proposing in this thesis, is a combinational approach in which we are combining the hierarchical approach and distributed approach for network management.

In this amalgamation approach, we are designing the network management solution in such a way that we acquire advantages of both approaches and avoid their shortcomings.

2.3. Network Management

Ad-hoc networks are basically multihop wireless networks where nodes may be mobile. These types of networks are used in situations where temporary network connectivity is needed, and generally, where the network is created on the fly. Since all the functions of network management are not crucial, they depend on the kind of applications being run on it. Generally, ad-hoc networks have very critical applications such as the military application, disaster relief, and emergency situations where reliability is most important. Consequently, we are concentrating our research

toward the fault/configuration management of ad-hoc networks. Due to the dynamic nature of ad hoc networks, there is a higher probability of faults, and a fault may also be triggered when the configuration changes [25][26].

Traditional fault tolerance schemes like checkpointing and message logging would serve the purpose, without any modification [26][27][28][29], if the mobile nodes have restricted their movement within 1 cell and there is a centralized entity to do the logging and checkpointing. Still ad-hoc networks do not require a user to maintain a fixed position in the network. Instead, they allow almost unrestricted user mobility. Due to mobility, the mobile users cross-cells, and handoffs occur. Mobile users die very frequently so transfer of responsibility from one node to another takes place very frequently. For this reason, the notion of 'immobile static storage' cannot be directly applied in the ad-hoc environment. Additionally, the traditional schemes do not consider the disparity in the bandwidth of the static network and the wireless network.

There is not much work done on developing recovery protocol for the ad-hoc network system upon the failure of the mobile node. An algorithm for checkpointing using distributed application on mobile computers is presented in [30]. The focus of the paper is, however, restricted to recording checkpoints, while recovery techniques are not discussed. The paper also talks solely of a mobile environment, not the ad hoc environment, wherein predominantly base station information recovery schemes are presented [31]. The basic approach replicates the information stored at a base station to some "secondary" base station. We have included that concept while designing a fault tolerant system for an ad-hoc network environment.

It has been mentioned that while working on ad hoc in detail, along with the faults that can occur, there is a proposed solution to avoid and solve the faults [15]. Still, these solutions use the default network architecture as the hierarchical, and have many lapses such as not giving details of the initialization stage. Furthermore, overhead in maintaining MIB is very high. In spite of everything the proposed solution in that paper has a partial single point of failure.

2.4. Proposed Approach.

After closely scrutinizing requirements and limitations of an ad hoc network, we are proposing a solution for the management of ad hoc networks. While developing this management solution, we realized that the most crucial but disregarded issues in an ad hoc network are fault/configuration management. The first step is to choose the best-suited network management architecture for an ad hoc environment. We decided on the distributed + hierarchical architecture (which we call a network architecture), through which we can get the minimum overhead and maximum reliability for any ad-hoc network.

We then collect threshold values for the clustering algorithm by simulating the ad hoc environment, as well as study the affect of proposed power saving techniques on ad hoc network performance.

In the following chapters design solutions will be discussed and results will be scrutinized.

Chapter 3:

Fault/Configuration Management for Ad-hoc Network

3.1. Introduction

Mobile wireless environments pose challenging problems in designing fault-tolerant systems due to the dynamic nature of the wireless network, mobility of the nodes, and limited bandwidth available on wireless links. Traditional fault-tolerance schemes cannot be directly applied to wireless systems, since a fault tolerant scheme for the wired network has different concerns than for the wireless network.

However, fault tolerance is much more important in mobile computing systems than in wired networks, as mobile computing systems are more prone to failures. This is because wireless networks have a high error rate, frequent disconnections and mobile devices are also more prone to failure and/or physical damage. Due to limited transmitting power and scarce bandwidth there is also a high probability of fault occurrence.

Traditional implementation of wireless networks is hierarchical, with every node having to report to a corresponding access point (base station). On the other hand an ad-hoc network is "a collection of mobile hosts forming a temporary network without the aid of any centralized administration or standard support services"[7]. In such a network, mobile nodes move very frequently. Thus, we cannot apply the same management schemes as we do to traditional wireless networks.

In this chapter, we address issues with making the traditional wireless network fault tolerant We also discuss the additional problems encountered due to an ad hoc

wireless network environment, and why fault tolerance is much more important in an ad-hoc environment then in conventional wired and wireless environments in section 3.2. We also suggest and justify scalable, reliable, and robust network management architecture for managing ad-hoc networks in section 3.3.2. Finally, we present a detailed design on how to deal with each configuration change and the occurring faults due to ad hoc environments and how to make a system fault tolerant.

3.2. Issues in Fault/Configuration Management of Ad-hoc Networks

By definition, the goal of fault management is to detect, log, notify, and (to the extent possible) automatically fix network problems to keep the network running effectively [5]. The main function of fault management is detecting, isolating and solving the problem in the network [20], but the problems in an ad hoc network are very different from wired networks. For example, a node failing because it has run out of battery power is treated as normal in ad hoc networks, whereas a node failure in wire line networks is treated as a fault that must be corrected [3].

In an ad-hoc network, we have also considered the configuration management problems of a fault tolerant system, since configuration changes can trigger a fault, which can in turn, restrain the system from functioning properly. For example, if a node moves from one cluster to another, this is a configuration change. It is now the responsibility of the configuration management tool to take care of the situation. However, in an ad-hoc scenario, that node could be a cluster head and could leave many unmanaged nodes behind. Because of this situation, the network cannot function properly so we also take into consideration such configuration changes in fault management.

Apart from configuration changes, a major cause of fault in any wireless network is due to the limited battery life. As a result of this all the management functions are performed with battery consideration. In the wired network there is an almost uninterrupted supply of power eliminating battery as a problem thus allowing for a very low probability of fault due to power failure. Therefore, fault management in wired networks and wireless networks have very different issues to be considered. The explanation given above signifies that anything that is of prime importance in a wired network may not be of equal importance in a wireless network.

Many management decisions are made based on battery life statistics collected from the nodes in an ad hoc network, but to collect this information from each and every node causes large overhead on the network. In addition, to collect the statistics, all of the nodes in the network have to participate. That computation consumes a significant amount of battery power. This is not the case with a wired network, where the only thing we have to consider is the network load.

In section 3.3 we discuss a solution, which makes a network management decision not only based on the network load, but also on battery life, and processing capabilities.

3.3. Ad-hoc Network Fault/Configuration Management Design

3.3.1. Design Assumptions

It is necessary to point out that in our design of ad-hoc network fault/configuration management, we focused only on two network management functionalities: fault and configuration management. We did not consider performance management and accounting management because there are many issues in fault/configuration

management, which need to be taken care of in order to implement ad hoc networks in the real world. Unlike traditional networks, power management is very critical for wireless ad hoc environments. We are proposing some power management schemes during our research because in an environment full of thin clients, power is one of the main cause of faults.

We are only considering very limited security methods to prevent unauthorized mobile nodes from joining the ad-hoc network. This is done by a technique called "host authentication" by the manager of manager (MOM), which keeps a list of MAC addresses of the nodes that are allowed to join the network. We believe it's not an overhead to keep a list of MAC addresses in MOM, rather a trade off between security and network traffic overhead.

In order to collect network information like Media Access Control (MAC) addresses, battery life, and position of the nodes that can facilitate the development of an ad-hoc network management protocol, and in turn the development of a supporting tool, our initial task is to come up with a suitable architecture. Once that is done we can look at the other issues associated with configuration management, such as what would be the criteria to form a cluster, how to form clusters, and how to manage configuration changes that can trigger a fault in the network.

We assume the availability of a GPS information system to provide us with the location information about each node. Therefore, we assume each node in the network has some kind of GPS capabilities, which means it can receive and process the information that it obtains from the GPS system. Another assumption would be the pre-existence of the routing protocol, because the network management protocol

is an application-layer protocol, we are assuming that there is AODV routing protocol existing at the network layer.

3.3.2. Ad-hoc Network Management Architecture

Managing an ad-hoc network becomes difficult compared to managing a wired network because of the dynamic topology of such a network, the limited resources on the mobile nodes (such as battery power, processing speed), and unreliable connections [3]. An effort must be made to reduce message overhead during network management while providing a relatively reliable and extensible network.

In an ad-hoc environment we can neither load a Manager of Managers (MOM) with many simultaneous tasks like in a hierarchical architecture [3], nor have a large amount of network traffic overhead like in distributed architecture [12,14]. We are therefore combining the attributes of both architectures. In particular, a distributed architecture is reliable and scalable because the management tasks are not confined to one manager [16,17]. For example, if one or more managers dies or moves out of its domain, the rest of the network is not affected. But, the architecture still suffers from the overhead of large information recollecting. This is because after a new manager is regenerated, it has to broadcast to its neighbors to inform them that a new manager has been selected, and to ask nodes that wish to join in its domain to send it certain information. In addition, in a distributed architecture, there is no single place to get an overall picture of the network. In order to get such information, a message must first be broadcast to all managers, and only then can each manager send the requested information to the requesting node. To alleviate traffic overhead in such situations, it helps if a MOM is introduced as in [3], but keeping all the management information

on the MOM as in [3] will introduce a lot of overhead. To avoid such overhead while keeping the benefit of MOM, we propose a hybrid architecture that combines the advantages of both distributed and hierarchical (DH) approaches (Figure 3.1).

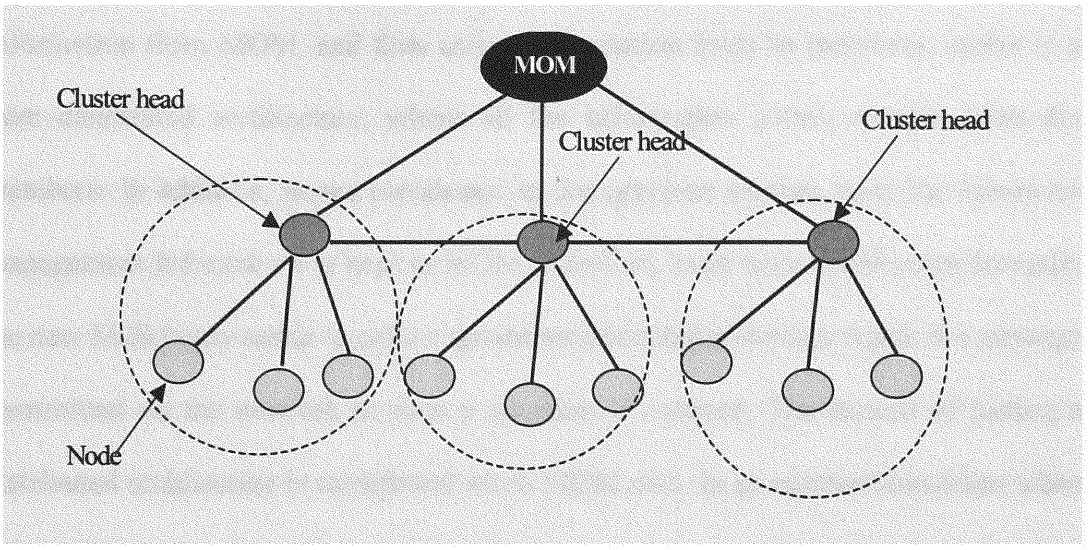


Figure 3.1. Logical Structure of Distributed + Hierarchical Architecture

The proposed Distributed+Hieriarchical(DH) architecture is a combination of distributed architecture and hierarchical architecture. In this architecture, the network is divided into different clusters. In each cluster, there is a cluster head, which is responsible for managing the cluster. Cluster heads can communicate with each other via peer-to-peer communication. Above cluster heads, there is a MOM. However, unlike the MOM in the hierarchical architecture, which collects all the management information and has more management responsibility, in DH architecture, the MOM only keeps some basic information about clusters. This information includes the MAC address of each cluster head, the MAC addresses of nodes under that cluster head, second/third MOM candidates, and coordinates of each cluster. It does not keep information such as coordinates, battery level, and processing speed of each node.

Such information is kept in MIB of the individual cluster head to reduce the message overhead of transmitting and updating such information on MOM.

In case that a cluster head dies, the new cluster head can get membership information from MOM, and then collect information from its members, unlike in a pure distributed architecture, where all the information comes straight from the members. In addition, as we mentioned in the previous section, only the minimum management information is kept in MOM, therefore, even when MOM dies abruptly, the new MOM only needs to collect the information that it desires. Again the message transmitted on the network in such a situation is reduced. The benefit of having a distributed architecture is manifested when MOM dies. In an extreme condition when MOM dies abruptly and the new MOM has not been informed, because of its autonomy and peer-to-peer communication mechanism with other clusters, each cluster still can function independently.

To achieve more fault tolerance efficiency in our design, we divide the network into different clusters so that the burden of network management can be distributed through the whole network. In the following section we will discuss generation, initialization, and maintenance of clusters.

3.3.3. Creating And Maintaining The Clusters

Our cluster generation algorithm is based on the geographical-based clustering algorithm in [3]. We made minor changes to it in order to better serve the requirements in our thesis. This algorithm uses the GPS (global position system)

information to divide the network into clusters**. The goal of this algorithm is to ensure a low message overhead and to form clusters, thus avoiding frequent topology changes.

This algorithm uses the information from a GPS to divide the network into clusters, each of which is rectangular in shape. The algorithm splits the nodes into clusters based on their spatial density. We select one node as the cluster head for each cluster according to the candidate's criteria, such as the location, the power, the processing ability, and the memory etc. Due to the mobility of each node, the spatial density of nodes is constantly changing. Hence, clusters we divided before may not be the optimal choice once the topology changes, so we have to reconfigure the entire network after a period of time, which will depend on the requirements of the system.

There are three entities in our geographical clustering algorithm: the nodes, the cluster head, and the manager of cluster head (MOM). In the life cycle of the entire network system, there are two phases the system has to take: cluster initialization and cluster maintenance. The first of these generates the clusters for the whole system, while the latter changes the topology of the whole system due to the mobility of every node in that system. Before we talk about the cluster initialization phase, we must first introduce the basic algorithms used in this phase. There are three basic algorithms: cluster generation, clusters size adjustment and clusters concentration adjustment. The first two algorithms are used during cluster initialization and the last one is used for cluster maintenance.

** This algorithm is based on GPS, a satellite-based radio-navigation system developed and operated by the U.S. Department of Defense. Any user on the earth knows its three-dimensional position, velocity and time through GPS, 24 hours per day in all weather conditions, anywhere in the world. The accuracy of GPS is in the order of meters.

3.3.3.1. Cluster Generation

The purpose of cluster generation is to divide the whole system area into smaller clusters of equal size so that we can compute the distribution of mobile hosts along the horizontal and the vertical direction.

We split the area into horizontal and vertical strips, and the number of strips in vertical and horizontal direction is determined by the following formula:

$$[\text{Number of strips (vertical or horizontal)}] = \left[\left(\frac{\text{length of the edge (vertical or horizontal) of the box}}{\text{average transmission range}}\right) X \text{ NUM}\right]$$

NUM is an empirical value derived by experience. The transmission range is divided into NUM pieces. If the number is too large, the node density is low. When we decide the scope of each cluster, we should consider the peaks and valleys in the bar graphs. Thin strips tend to have very low frequency count, while wide strips tend to lose the distribution information. In our architecture we set NUM to be 3 but the value of NUM is still an open issue.

After we find the peaks and valleys in the bar graphs, we select the consecutive valleys at least 6 strips in between. If we select very small clusters, the transmission ranges of two cluster heads will overlap, and it will waste the management resource. If we select very large clusters, the communications among the same cluster will need more routing steps. This will affect the message efficiency. The best choice for the size of a cluster is to double the average transmission range (transmit in converse direction) of nodes. Then, we find the valley strip where the density of the cluster is minimum. We use the center of the valley strip to divide the whole area into

rectangular clusters, based on the above criteria. Finally, by calculating the intersections of the valley strips we get the cluster list.

For example, Fig 3.2 is the result after we use the formula to divide the whole system area into horizontal and vertical strips. At the left and bottom of the Fig 3.2, we count the number of nodes located in every strip in both directions. As we can see 2nd and 7th strips are the valley strips on X-axes and the 4th strip is the valley strip on Y-axes. Fig 3.3 is the result after we find the peaks and valleys of the bar graph, and choose the suitable strips as edges of every cluster.

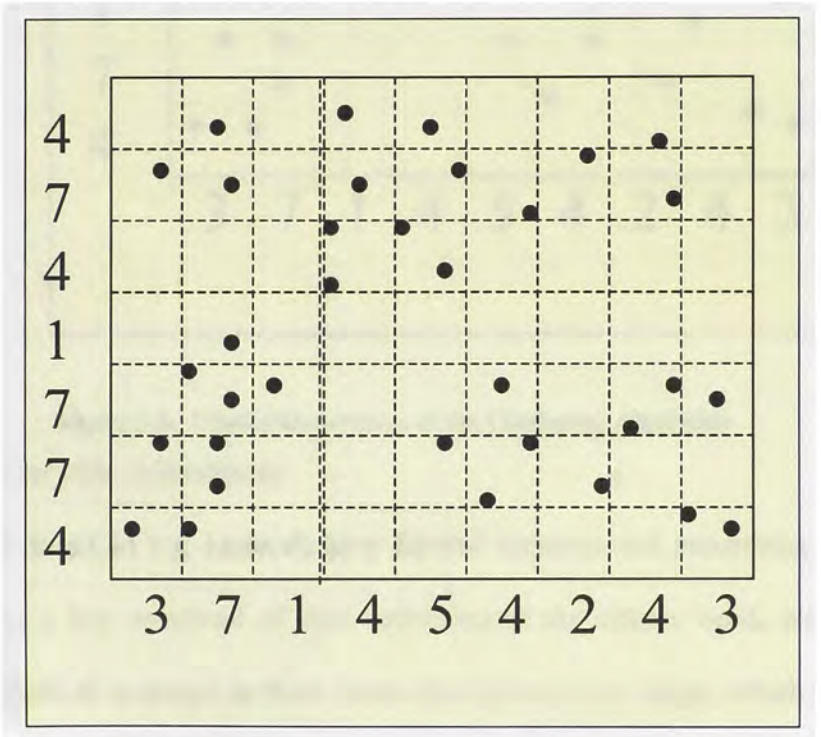


Figure 3.2. Strip Generation of the Clustering Algorithm

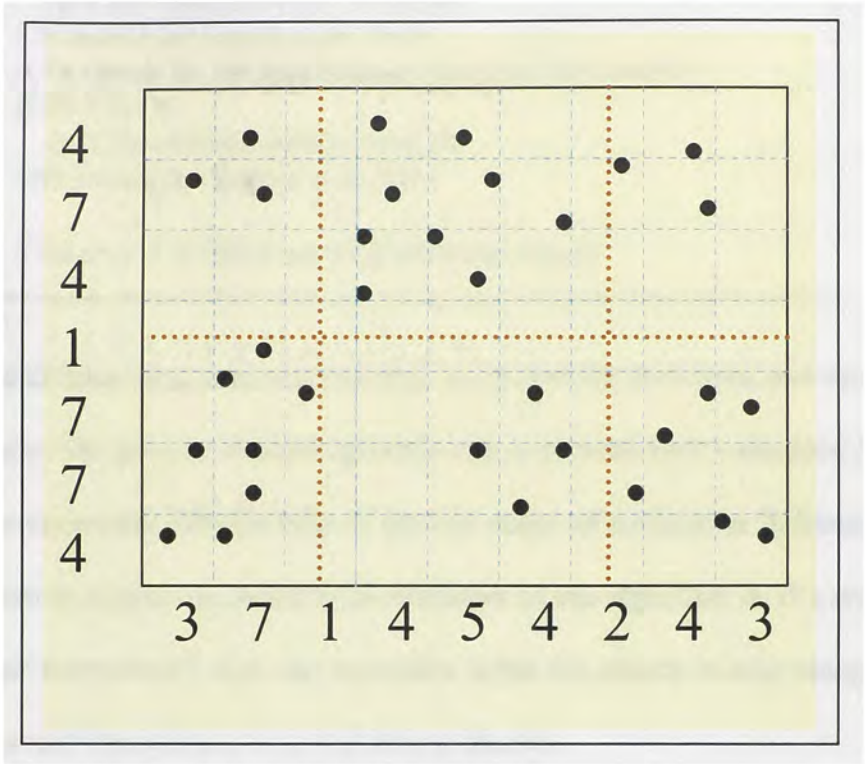


Figure 3.3. Cluster Generation of the Clustering Algorithm

3.3.3.2. Cluster Size Adjustment

The nodes in an ad hoc network have limited memory and processing power. In order to keep a low overhead of data collection at the cluster head, we limit the maximum length of a cluster to three times the transmission range, which means the cluster heads need at least two inter-nodes to access nodes located at the edge of the cluster. In our algorithm, if a cluster is bigger along a dimension, we just split it evenly along that dimension. But as per the results collected in chapter 4, maximum length does depend on the node density in a particular area.

```
Start
        if Bl > Z(Tx)
           split the clusters evenly along Bl
        //Bl stands for length of the box//
        // Tx stands for the transmission range of the device//
        if Bh >Z(Tx)
           split the clusters evenly along Bh
        //Bh stands for length of the box//
end
        //Value of Z is based on the performance data
```

It is possible to form long or skinny clusters, as we find the horizontal and vertical valleys separately. We set up a cluster edge ratio rule to prevent such a situation from happening. The maximum allowed ratio of the two edges of a cluster is X (based on the results shown in chapter 4). Another modification of the algorithm is, if a cluster violates the above-mentioned rule, the algorithm splits the cluster evenly along the longer edge to avoid formation of long and skinny clusters.

```
Start
        if (Bl/ Bh) >X
        split the clusters evenly along Bl
         //Bl stands for length of the box//
        if (Bh/Bl) >X
        split the clusters evenly along Bh
        //Bh stands for length of the box//
end
        Value of X is based on the performance data
```

3.3.3.3. Cluster Concentration Adjustment

Since the clusters in an ad hoc network have limited memory and processing power, we must avoid the high node density or else it would overwhelm the cluster head rendering it incapable of handling other requests. In a converse situation there

could only be a few nodes in a cluster, or in an extreme situation, the cluster head could be the only node in a cluster.

To handle these cases, the cluster concentration adjustment algorithm defines the minimum (MinN) and maximum (MaxN) number of nodes allowed in a cluster. To adjust the node concentration of the cluster, we must first count the number of nodes in each box, and if a box is empty, we delete this cluster from the cluster list. On the other hand, if a cluster B_i has less than MinN nodes, we define the merging objects as: the left, right, top and bottom neighbors, which generates the candidates set S (Bl, Br, Bt, Bb). Whenever S is empty, the merging phase stops. It is possible that when Bi's nodes are added to its neighbor, the number of nodes for its neighbor will exceed the MaxN. If this happens, we get rid of such a neighbor from set S.

It is possible that, when Bi merges with its neighbor, the resultant cluster would violate the cluster edge ratio rule. We will try to merge them by splitting the resultant cluster along the longer edge. Here exists a trade off. We think the merging and splitting method is a good choice from the viewpoint of distributing the management duties. If more than one cluster can merge with Bi, we select the smallest increase in edge length. This is due to the transmission range limit.

If a cluster B has more than MaxN nodes, it should take the splitting steps. If the edges of a cluster are longer than the transmission range, we must consider not violating the cluster edge ratio rule before we split it. We can split Bi into 4 or 2 equal clusters. If any one edge or both edges are located in the transmission range, we can ignore the cluster edge ratio rule to split the cluster. This is because the resulting two clusters are in the same transmission range, and can communicate directly. There

exists an overlap for these two clusters. We have no other choice since one cluster head has limited capability to hold so many nodes. Although it happens rarely, we must consider it. When the previous four steps finish, we get a cluster of a satisfactory node density.

```
Start

get Bi (Count no of Nodes)
        if Bi = 0 (No of Nodes in a Box is 0)
                remove box entry from the MOM database;
        else if Bi < MinN
                {
                find Bl, Br, Bt, Bb and make set S
                // Bl, Br, Bt, Bb stands for the box on the left side, right side, top,
                and bottom of the box Bi respectively//
                if Bi + Bl > MaxN
                        remove Bl from set S
                if Bi + Br >MaxN
                        remove Br from set S
                if Bi + Bt > MaxN
                        remove Bt from set S
                if Bi + Bb > MaxN
                        remove Bb from set S
                Check Bl, Br, Bt, Bb for edge ratio

                // edge ratio = ratio between length and height //

                if edge(Bi + Bl) > edge ratio
                        remove Bl from set S
                if edge(Bi + Br) > edge ratio
                        remove Br from set S
                        if edge(Bi + Bt)ftp://Bt from set S
                if edge(Bi + Bb) > edge ratio
                        remove Bb from set S
                Check S
                        if only one box remaining
                                select that
```

```
            else
                    {
                            compare all for edge ratio
                            select the one with min edge increment from Bi
                    }
            Bj = selected box
            merge Bi + Bj
            remove Bj from database
      else if Bi > MaxN
            if splitting procedure violates the edge rule Stop

            // edge rule is edge ratio = ratio between length and height//

            else
                    P = Bi/MaxN
                    Case 1: P <= 2
                                                    if max (Ex,Ey) > R
                                                            split Bi in 2 equal size clusters
                                                    else no change(break)
                    Case 2: P > 2
                                                    if (Ex and Ey) > R
                                                            split Bi into 4 equal clusters
                                                    if (Ex or Ey) > R
                                                            split Bi into 2 equal clusters
                                                    if (Ex and Ey) < R
                                                            no change(break)
end
```

3.3.4. Network Initialization

In the very beginning, an initialization step has to be run to form a temporary network so that mobile nodes can start collaborating with each other. The initialization process is divided into three functions: First, the MOM is set up (sets up and collects all the information required to setup an ad hoc network). Next, it should setup and collect information from the cluster head, and finally, it should update the MIB's.

To form an ad-hoc network from scratch, there are two possible approaches. For the first, we know in advance that we need an ad hoc network at a particular location. Also, we would like to have a restriction on the people who can join the ad hoc network, for e.g. in disaster relief, we only require federal people to share the resources. The second one is where any one can come in, form a network, share information and depart. In this case we select MOM randomly.

We consider the first case, where the list of nodes that are allowed to participate in forming an ad hoc network is fixed, and a node has all the information regarding the network formation. That node named MOM will trigger the initialization for the whole ad hoc network and will store some registration information. MOM must authenticate every node that wants to join the ad hoc network. This is the limited security our architecture provided. The MOM broadcasts the network set up message to all its neighbors.

Every node, which receives the message, will broadcast to its one-hop neighbors and so on. If it wants to join the system, it should send the joining information to MOM. At the same time, the power, processing ability, and memory information will

be sent too. MOM authenticates every node, and will reply to every node with an identification key.

MOM uses the Cluster Generation algorithm to divide the whole system into clusters, and the Cluster size adjustment algorithm to adjust the edges for clusters, and the Cluster concentration adjustment algorithm to distribute the nodes evenly.

After comparing the collected information from all the nodes that want to join the system, MOM selects the cluster head for every cluster based on the cluster head criteria, such as power, memory, and processing ability. At the same time, MOM selects 2 cluster heads as the backup MOM, that will act as MOM when the current MOM abruptly dies or is out of power. Some security information will be backed up onto this backup MOM and it then sends the assignment information to every cluster informing the boundary of that cluster and giving it the authorization to collect information from the nodes in that cluster.

The second function now begins in which, cluster heads receive the notification and the scope of the cluster, and then broadcast the "join" information with such scopes to its neighbors. Every node that received such "join" information will compare the scope with its own coordination and then reply to the corresponding cluster head's request with the management information.

When the cluster head receives its cluster members' information, it will set up its own MIB. The cluster head then sends a report concerning its cluster to MOM, which contains partial information collected by the cluster head. It continues to send information that is required to assign a task to a new cluster head in case the cluster dies abruptly. Consequently, the MOM sets up its MIB for the whole ad hoc network.

The initialization phase comes to an end here, The next section is about maintaining the network changes and transferring of authorization to the appropriate MOM and cluster heads.

3.3.5. Fault/Configuration Management of Ad-Hoc Networks

As explained previously in ad hoc networks, nodes die, move, or power themselves off to save energy. In all of these situations, network topology changes and the manager station needs to know the exact location and status of the nodes. This information is collected through SNMP, software agent [11] and GPS and management decisions are taken based on this information. SNMP is widely used for network management, for wired networks, and for wireless networks. By using agents to collect certain information, we avoid plunking more loads onto the network.

Before describing the design of the fault/configuration management of ad-hoc networks, it is important to clarify the information that is stored on different types of mobile nodes, i.e., MOM, cluster heads and simple nodes.

We store the following information in the MOM: a list of MAC addresses of cluster heads in the network, a list of MAC addresses of mobile nodes under each cluster head, coordinates of each cluster, and 2nd and 3rd MOM candidates.

The following information is kept in the MIB of a cluster head: a list of MAC addresses of mobile nodes under the same cluster head, coordinates of the cluster, remaining battery, and processing speed of each mobile node within this cluster.

Each mobile node has a MIB, which records the following information: its coordinates, its remaining battery, processing speed, coordinates of its cluster, MAC address of its cluster head, and the MAC address of MOM.

Using MIB we can transmit burst of management information at once. The objective of using MIB is to reduce the load on the network and get more efficient use of battery power.

To achieve the same in our design, we decided not to store the accurate value of remaining battery power in the MIB of each cluster head. Instead, we divide power level into four levels: almost full, reasonably full, less then half, and about to die as shown in figure 3.4. This information can be transferred from one node to another using only 2 bits**. The advantage of doing this is a significant decrease in amount of control traffic on the network as well it significantly reduced battery overhead caused by management information. In traditional networks, SNMP updates are sent every five seconds carrying all the information. During that update, it also carries the amount of battery life remaining.

It requires more then two bits (minimum of 6 bits) to represent battery life information. This sums up to quite a significant amount of traffic, while using the scheme we suggested earlier we could reduce the amount of control traffic significantly. This is because according to that scheme, battery information is only transmitted when the battery life reaches any of the four thresholds, or in the situation when a network node passes through the initialization process (regardless of circumstances).

** 00 to represent threshold level 1, 01, 10 and 11 for threshold lever 2,3 and 4 consecutively

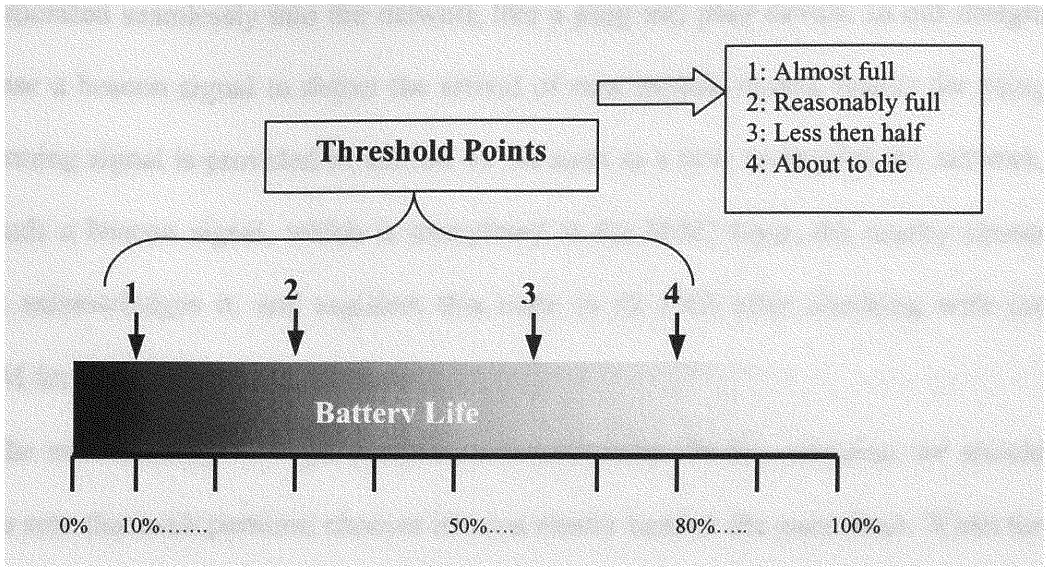


Figure 3.4: Different battery levels

Reduction in amount of control signals on the network greatly reduces the load on the network and also affects battery life significantly. The reason for that is when we talk about ad-hoc networks it is mainly about lap tops and other thin clients which runs on battery using 802.11 or bluetooth technologies. A wireless network card consumes quite a significant amount of power, not only while transmitting, but also while listening to neighbors and even in an idle state. That will be explained in detail in the section 3.3.6 (of power management).

Some other considerations while designing an ad hoc network infrastructure would be in situations when the node is unavailable for of any reason (e.g node is out of range). The manager notes the event in the database. Even if the node is dead, it keeps the entry in the database for a certain amount of time due to the temporary nature of the ad hoc network.

New nodes may join the network periodically and these nodes must be incorporated seamlessly into the network like a plug and play device. In our design, we use a beacon signal to detect the arrival of new mobile nodes(reason for using beaconing signal is provided in section 4). As soon as a new node joins the network, it sends a beacon signal, which is interpreted at the MAC layer, the nearby cluster head acknowledges it, and registers this node in its MIB after checking with the MOM for potential security violence.

The network may also get partitioned occasionally. In this situation we should make sure that each partition chooses its own cluster head at the same time. When the network merges, one common cluster head needs to be chosen and the other one has to give up all rights and be just a node that reports to the cluster head. This decision of who takes over and who gives up is based on the hardware and software capabilities of the node and battery power. For instance, many nodes in the ad hoc network are expected to be thin client, which does not have processing power or decision-making skills. Each and every event triggered by managers consumes power, so the decision is also based on the remaining battery life.

In the following sections, we will explain how we handle different situations of the configuration and fault management in ad-hoc networks. Since there are three types of mobile nodes in the architecture that we proposed, we categorize our explanation based on the type of the mobile node.

3.3.5.1. Managers of Managers (MOM)

Switched off:

When the end user switches off the node, which has MOM responsibilities, that signifies regular termination of the mobile node. This is considered a normal event in an Ad-Hoc wireless environment. In response to this incident the number of events that take place are as follows:

In MOM, there will be a list of 2nd and 3rd best candidates (nodes), which will be chosen from the list of Cluster Heads already defined in MOM's MIB. These 2nd and 3rd best candidates will be updated and informed each time there is a change in the "Cluster Head MIB" in MOM that they are the next best candidates respectively. The 3rd best candidate serves as a surplus in the case that MOM and the 2nd best candidate will simultaneously switch off or abruptly die. As shown in figure 3.5, whenever MOM is preparing to switch off, it fires an agent that transfers the entire MIB and sends a TOC (Transfer of control) message informing the 2nd best candidate that it will be serving as the network's MOM. In the case of the 2nd best candidate not being available, the agent itself goes to the 3rd best candidate.

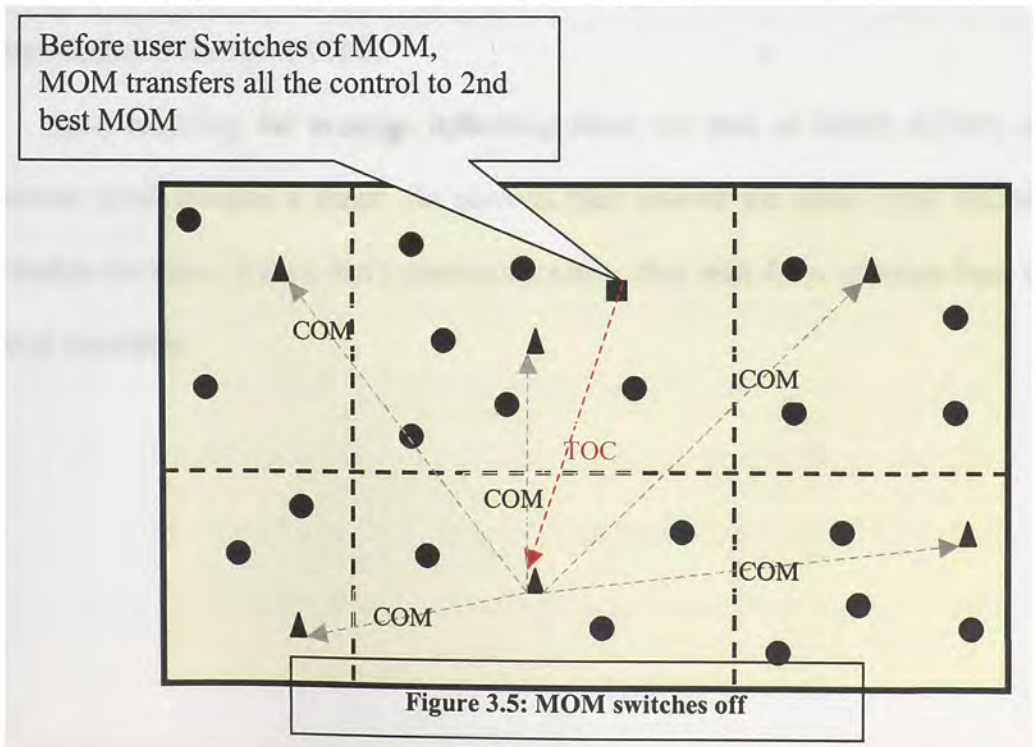


Figure 3.5: MOM switches off

Upon receiving the TOC signal, it informs all the cluster heads about the change in MOM by sending a COM (change of MOM) message. The entire cluster head then updates the MOM address in their database, so that they can direct all the updates to the new MOM instead of the old MOM.

Abruptly dies:

In case the current MOM dies abruptly of unexpected causes, without firing an agent that transfers the MOM's MIB is transfered. As shown in figure 3.6, as soon as any cluster head discovers that there is no MOM, it multicasts a message to all cluster heads informing them about the lack of MOM (LOM) and requesting claim of the next MOM. Cluster heads are one of the most capable nodes in the cluster so, the 2nd and 3rd best MOM should be amongst them. Upon receiving this message, the predetermined 2nd best candidate multicasts a message announcing it is the next MOM and requesting the required data (RFM: Request for MIB) from cluster heads to create MOM's MIB. In reply, required data (MIB) from the cluster head is transferred to the new MOM.

Upon receiving the message informing about the lack of MOM (LOM), all the cluster head invokes a timer. As soon as they receive the claim from MOM they disable the timer. If they don't receive the claim, they wait for a response from the 3rd best candidate.

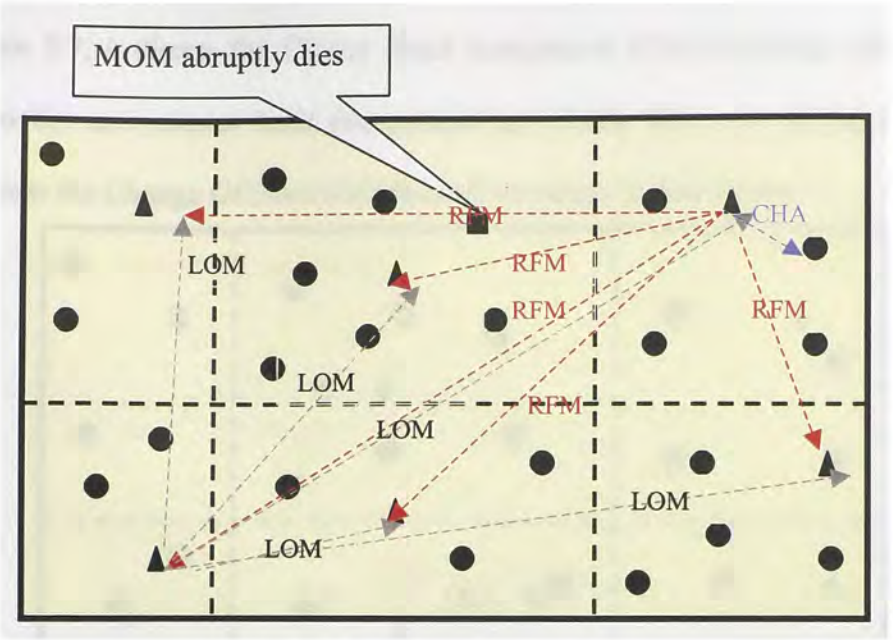


Figure 3.6: MOM abruptly dies

Moves out of it's cluster boundaries to another cluster:

This does not affect the network in any way. Because from cluster management's prospective, it is just another node. Any action taken will be the same as the action taken when any node moves out of it's cluster boundaries to another cluster as is explained later in section 3.3.5.3.

3.3.5.2. Cluster Head

Switched off:

When the node with the cluster head responsibilities is preparing to switch off, it internally runs the Cluster Head Determination (CHD) algorithm to find the best available node in that cluster from the MIB. The CHD algorithm calculates the best node amongst the group of nodes based on predetermined criteria (e.g. battery power, processing speed, node position etc.) stored in the cluster head's MIB. Then as shown

in figure 3.7, it directs the Cluster Head Assignment (CHA) message and then the MIB to the new cluster head (determined by CHD). The new cluster head then broadcasts the Change Of Head (COH) to all the nodes in that cluster.

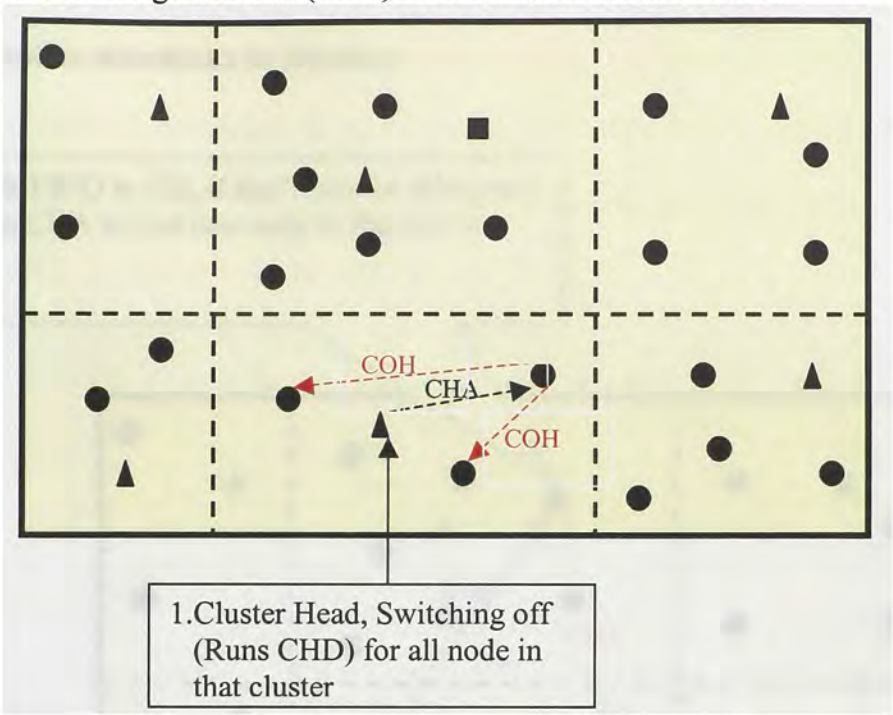


Figure 3.7: Cluster head switches off

Abruptly dies:

It is sometimes possible that a cluster head may be disconnected abruptly from its cluster (either due to fading, jamming, or other catastrophic failure). In such a case, the cluster nodes are left unmanaged, and they do not know that they are unmanaged. The solution is that cluster heads are required to periodically broadcast a Periodical Ping (PP) message.

As shown in figure 3.8, nodes maintain a timer that is reset whenever a ping is received. If the timer exceeds the time limit, it informs MOM of being unmanaged by the "Unmanaged Cluster" (UMC) message. In response MOM sends a verification message to confirm whether the cluster head is definitely dead using an ICMP echo

message. If it does not receive the echo reply according to the database MOM sends that cluster's MIB and Cluster Head Assignment (CHA) message to the next best cluster head in that cluster. If MOM receives an echo reply it sends a message to the cluster head to rebroadcast its presence.

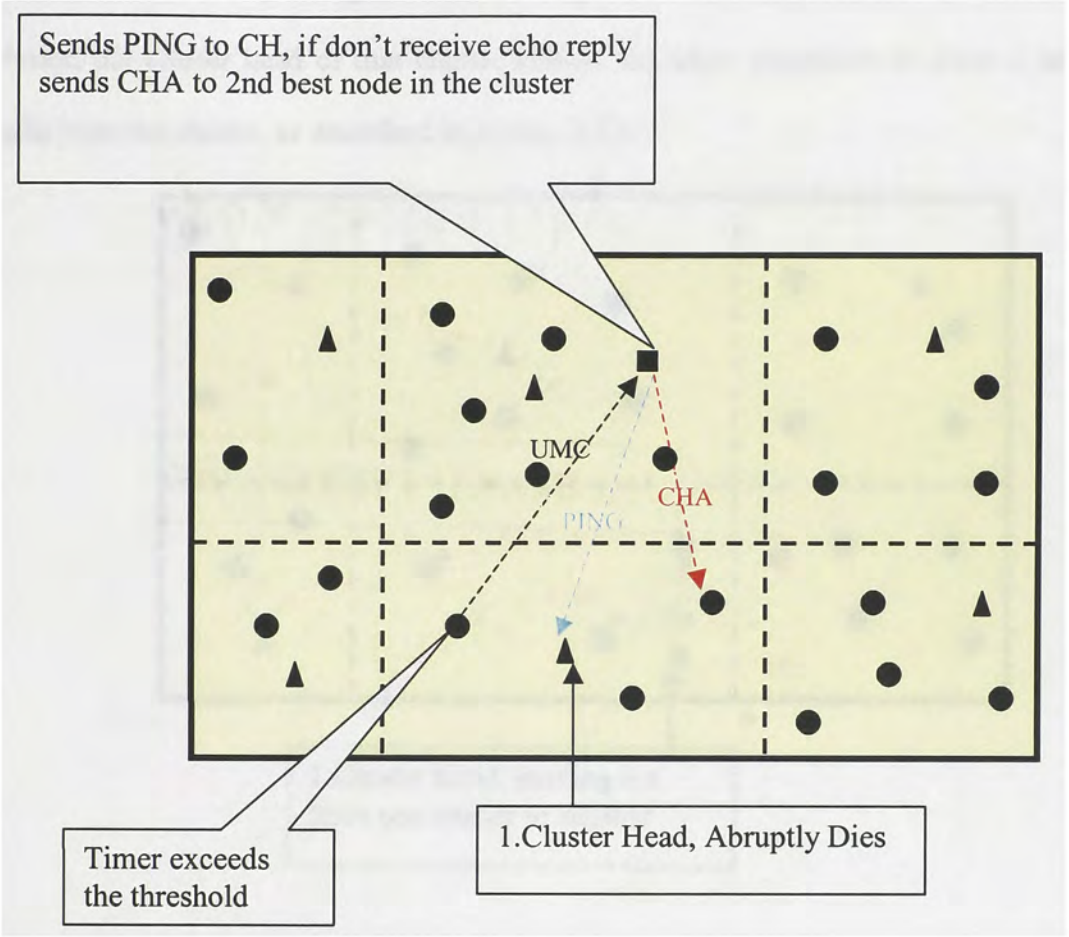


Figure 3.8: Cluster head abruptly dies

Moves out of its cluster boundaries to another cluster:

Every cluster head in our ad hoc network environment continually runs the algorithm, in predetermined time intervals, to find out its own coordinates. To determine if it has

crossed the cluster boundaries, it compares its individual coordinates with the cluster coordinates periodically.

As shown in figure 3.9 the former cluster head sends the cluster head assignment message to the next best candidate, and then the next best candidate informs all the node in the cluster by change of head message.When this cluster head enters a new cluster, the cluster head of that cluster follows the same procedure as when a new node joins the cluster, as described in section 3.3.6.3.

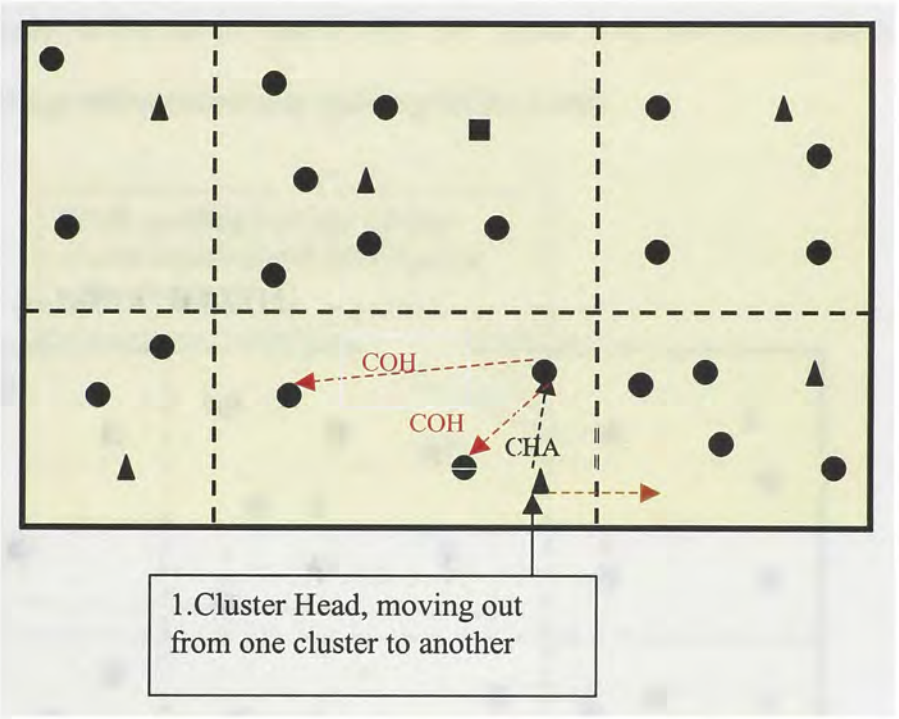


Figure 3.9: Cluster head moves out of its cluster boundaries to another cluster

Moves out of the network boundaries:

A cluster head going out of the network's boundary is treated the same as when it abruptly dies. When this happens there is no indication in advance. Instead, it suddenly moves out of the boundary and can move in the boundary also. To avoid the

overhead of recalculating the entire scenario for that node, the network keeps entry of that node in the cluster head and MOM for some preset amount of time.

3.3.5.3. Simple Node:

Switched off:

When the node is preparing to switch off, as shown in figure 3.10 the node fires a Node Switching Off (NSO) message informing the cluster head that it is about to turn off. In reply to this message the cluster head deletes the record of the node entry from the list of nodes in the cluster head's MIB. The cluster head will inform the MOM about this change while periodically updating MOM's MIB.

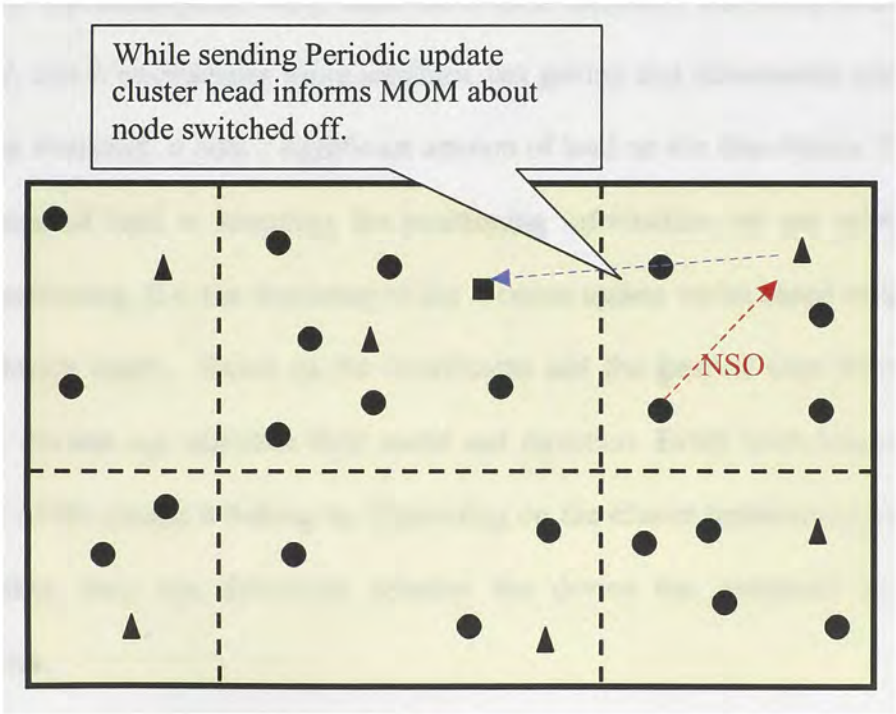


Figure 3.10: Node switches off

Abruptly dies:

It is sometimes possible that a node may be disconnected abruptly from its cluster. Such a case is when a cluster head periodically broadcasts a Periodical Ping (PP) message. The cluster head keep tracks of the echo reply it gets from each and every node. If the cluster head does not receive the echo reply from a particular node within a predetermined time, it marks down the node. If another interval lapses, it will then delete the record of the node entry from the list of nodes in the cluster head's MIB and inform the MOM when it will periodically update MOM's MIB.

Moves out of it's cluster boundaries to another cluster:

As per our assumption, every node has a GPS capability and every node can get it's X, Y, and Z co-ordinates using satellites, but getting that information periodically is also an overhead. It puts a significant amount of load on the thin clients. To reduce the amount of load in receiving the positioning information we use relative time based positioning. (i.e.:the frequency of the location update varies based on the speed of the device itself). Based on the coordinates and the interval time between two updates, devices can calculate their speed and direction. Every node knows the co-ordinate of the cluster it belong to. Depending on the cluster boundary co-ordinate's information, they can determine whether the device has exceeded the cluster boundaries.

As soon as the device realizes that it has crossed the boundaries it sends a node-leaving (NL) message to the previous cluster head and broadcasts a request to join cluster (RJC) message to the new cluster. Upon receiving the leaving message the previous cluster head deletes the record of the node entry from the list of nodes in

cluster head's MIB after a predetermined timer is exceeded. Now the new cluster head that received the join request contacts the MOM for the authorization of the node that has sent the RJC. MOM will check the list of MAC addresses to determine whether this node can be placed in the cluster.

On receiving the authorization, the cluster head will run the Cluster Head Determination (CHD) algorithm to determine which node is now best suitable to become the cluster head. If the node is more powerful then the existing cluster head, the cluster head sends a CHA message to the new node and then new node sends COH to all the nodes in that cluster as described in figure 3.11.

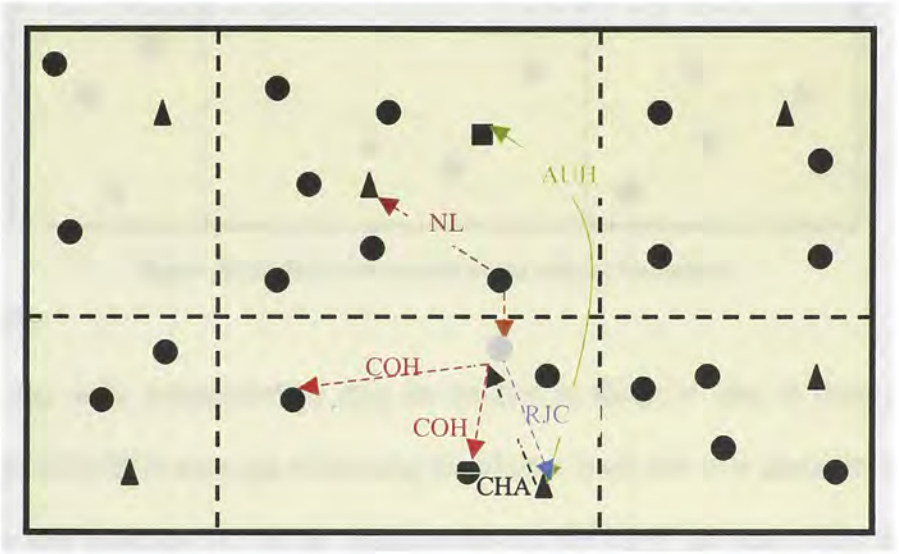


Figure 3.11: Node moves out of it's cluster boundaries to another cluster

Moves out of the network boundary:

As stated in our assumptions and in the previous sections, every node has a GPS and it knows its boundaries. Once the node crosses its boundaries, as in moving from one cluster to another, it will fire the NL message to the old cluster head as shown in figure 3.12 and a NRJ message towards the new cluster, hoping that a new cluster

head will reply to the message after authentication. If no cluster heads receive that message after a predetermined time, depending on the kind of environment the ad-hoc network has been setup in, it realizes that it is out of the network boundary. On the other hand, when the old Cluster Head receives the NL message it will delete the node from its MIB and inform MOM to update its MIB.

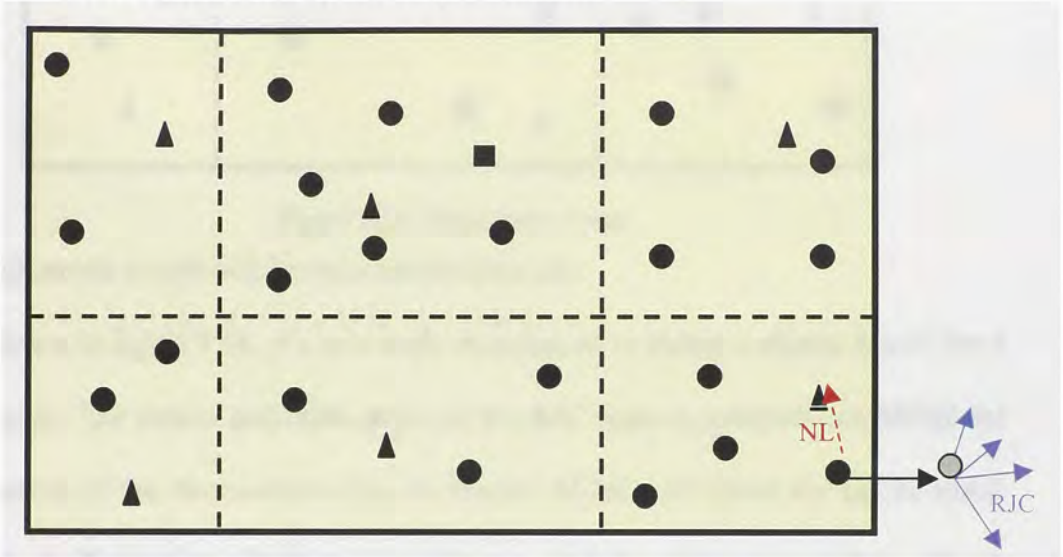


Figure 3.12: Node moves out of the cluster boundary

Battery dies:

When the node acknowledges that its battery is about to die, it fires a Node Switching Off (NSO) message informing the cluster head that it is about to turn off. In reply to this message the cluster head deletes the record of the node entry from the list of nodes in the cluster head's MIB as shown in figure 3.13. The cluster head will inform the MOM when it will periodically updates MOM's MIB.

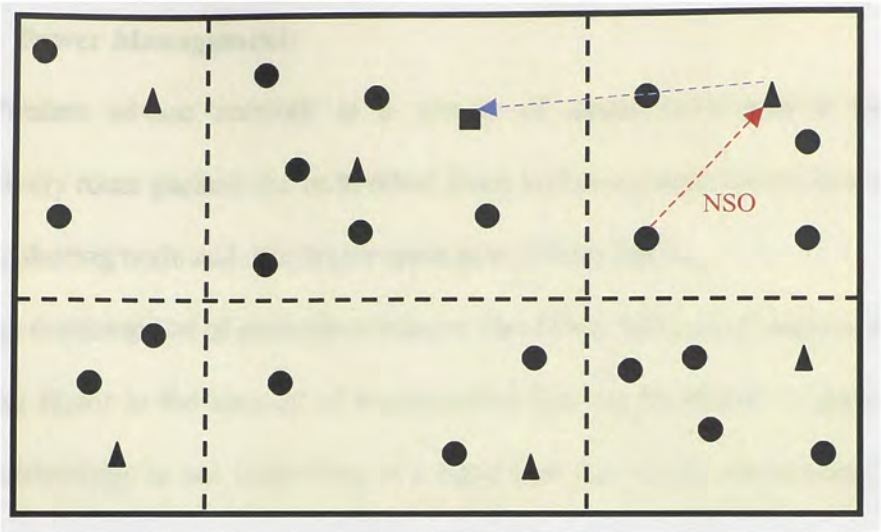


Figure 3.13: When battery dies

New node moves in network boundaries/Switches on:

As shown in figure 3.14, if a new node switches on or enters a cluster it will fire a join request. The cluster head that received the RJC request, contacts the MOM for authorization of the new node to join its cluster. MOM will check the list of MAC addresses to determine whether this node can be placed in that cluster. Upon receiving the authorization, the cluster head will run the Cluster Head Determination (CHD) algorithm to determine the best node that could become the cluster head.

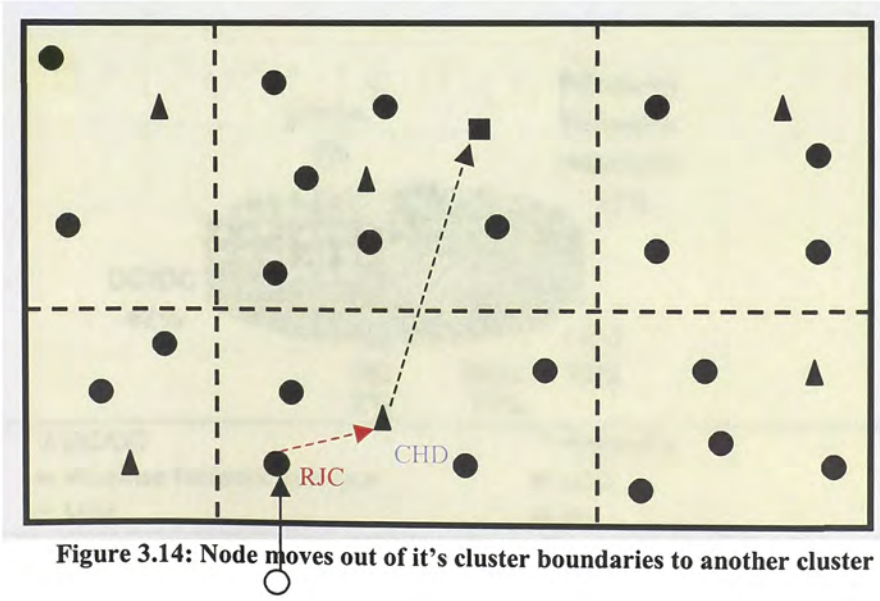


Figure 3.14: Node moves out of it's cluster boundaries to another cluster

3.3.6. Power Management:

A Wireless ad-hoc network is a system of autonomous mobile nodes that cooperatively route packets for each other. Each and every node works as a router for their neighboring node and this draws more power from device.

Energy consumption of portable computer like PDA, Wireless Phones and Laptops is limiting factor in the amount of functionality that can be placed in these devices. Battery technology is not improving at a rapid rate due to the fundamental physical constraints. That's why we are including this issue in our design to make network less prone to failure and more reliable (long lasting).

As shown in figure 3.15, quite a large amount of power in consumed by wireless network interface card. We will discuss different scheme that will help reduce the power consumption. Power consumption due to DC/DC, LCD and I/O device are hardware level and are out of scope of our thesis, so we will mainly concentrate on reducing power consumption due to Wireless Network Interface and Processor.

Average battery Utilization by individual component

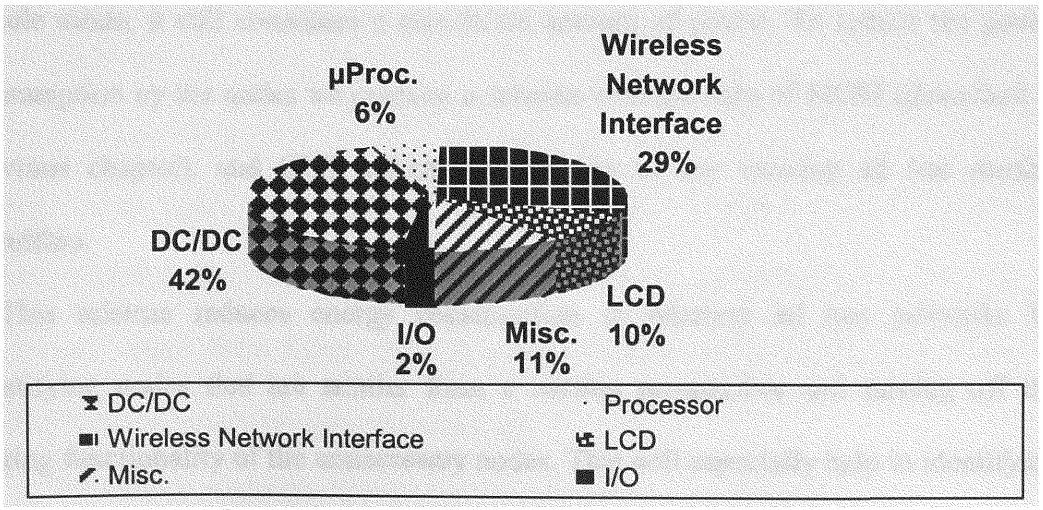


Figure 3.15: Average battery Utilization by individual component

As per wireless radio specifications of wireless devices a wireless card mainly works in 4 different modes, Transmitting, Receiving (listening), Inquiry, and Idle. Figure 3.16 shows the graph for input current verses different card state as explained above for typical 802.11 and bluetooth technology implementation.

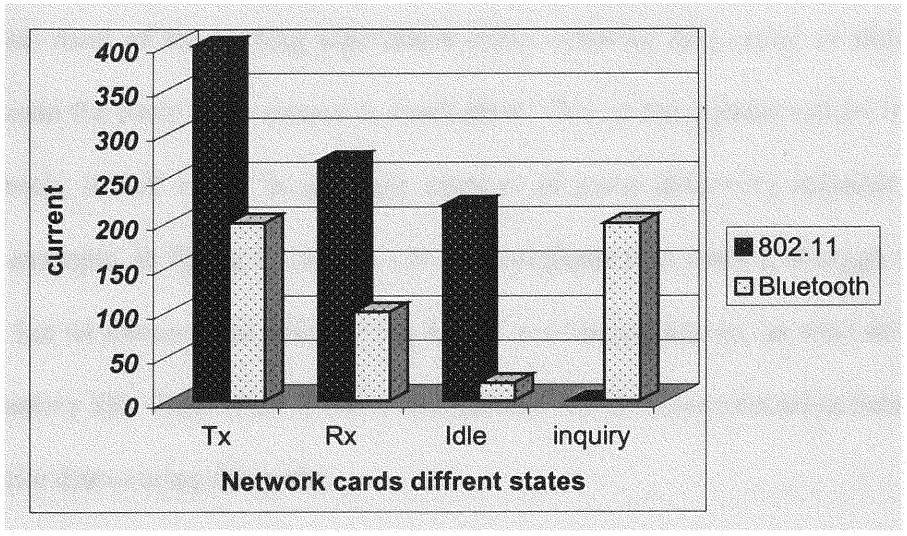


Figure 3.16: Input current verses different network card states

As per the graph shown above, it is clear that even when devices are in receiving or idle mode, it still consumes a significant amount of power. To reduce the power consumption by the nodes we propose a scheme with the help of MOM (described in previous chapter), and GPS, which runs on top of any existing ad hoc routing algorithm.

This scheme reduces energy consumption in wireless ad hoc networks by identifying nodes that are similar from a routing prospective and turning off the routing functionality of the unnecessary nodes. This will especially help in identifying

nodes that are directly connected to the power supply and in taking advantage of these nodes by assigning them comparatively more load and taking off a significant load from other battery operated devices. We will call this algorithm an Energy saving algorithm (ESA)

We have observed that when there is significant node redundancy in an ad-hoc environment, we will see multiple paths exist between the source and destination node because, most of the routing algorithms either consider hop count or distance vector to decide the route from source to destination. Due to the volatile nature of the ad-hoc network, it will result in a higher number of route discovery requests and replies. For example, in figure Y, node A can communicate with node E through node B, C, or D. Let us assume that node D has higher processing power, as well as high remaining battery life. Then node B and C are extraneous for communication between A and E, while discovering the node.

To reduce the number of discovery packets and the node life time we are collecting nodes positioning information through the Global positioning system (GPS), and calculate density and node redundancy at MOM, which will determine which nodes are redundant. MOM can request the cluster heads to send the message to the corresponding node to turn off their routing functionality. Take the same scenario as in figure 3.17, when A wants to transmit a packet to E it will send a discovery packet; only D will forward that packet ahead and reply back to A with the discovered route. This saves lots of energy consumed by individual nodes in transmitting and receiving discovery packets.

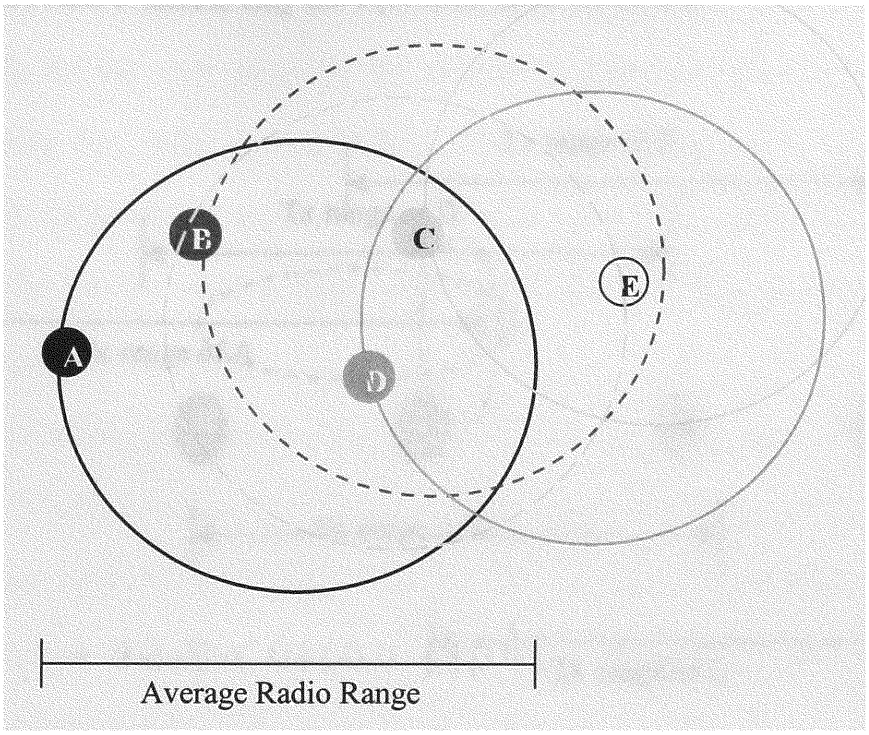


Figure 3.17: Example of Scenario where node redundancy can increase the unnecessary flooding

The main functionality of the Energy saving algorithm will be to determine node equivalence. Node equivalence is basically derived using the GPS information about the transmission range of that device and by creating virtual grids, which are different from the clusters created for the handles that management functionality efficiently.

Even though we have assumed that all nodes have their precise location information, it is not easy to find equivalent nodes in an ad-hoc network. Nodes that are equivalent between some nodes may not be equivalent nodes to another set of nodes. For example, in figure 3.18, the radio range of all nodes is slightly larger than twice the average distance between any node. As shown in figure 3.18, if in the

communication between node A and D there are two equivalent nodes B and C, then between A and E there is only one equivalent node which is C.

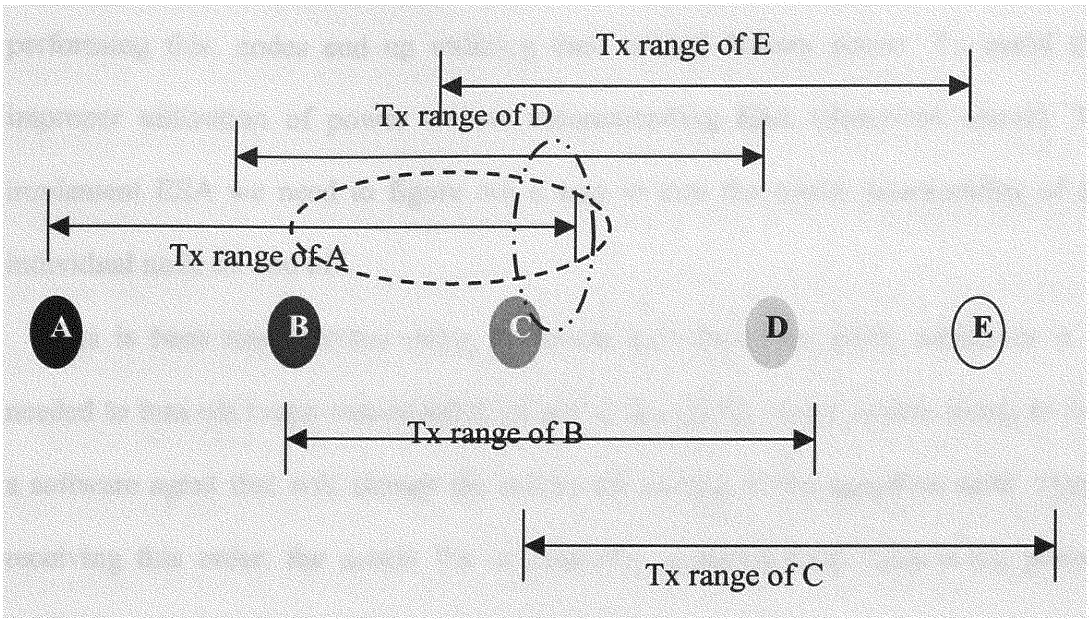


Figure 3.18: Problem of Node equivalence

We are addressing this problem by collecting all the location information at MOM. Then MOM gets network area coordinates from the individual node location information derived from cluster heads. Then MOM now divides this area into small "imaginary grids". The imaginary grid is created such that for two neighboring grids 1 and 2, all the nodes in grid 1 should be able to communicate with all the members in 2, and vice versa. That means all the nodes in the same grid are equivalent. We can select the best node in each grid and turns of the routing functionality of the remaining nodes in that grid.

One of the basic criteria of an ad hoc network is, each and every node works as router as well as a node. This ends up in a node reading each and every packet and checking for the route entry for the destination address of those packets. While performing this, nodes end up utilizing their critical battery power. To avoid the improper utilization of power we are recommending ESA (described above). To implement ESA we need to figure out a way to turn the router functionality of an individual node on and off.

This is been implemented using an access list. Based on ESA, whenever it is needed to turn off router functionality of any node, MOM orders cluster heads to fire a software agent that will change the access list setting of the specified node. Upon receiving this order, the access list is generally modified from "access-list permit any" to "access-list deny any" on the incoming port.

This concludes our entire design chapter. This chapter discusses all critical issues for better implementation of ad hoc networks, as well as issues in making a network fault tolerant. To handle these issues we have proposed some new concepts in addition to which we made some changes to existing algorithms. In the design phase we have discussed all this concepts and extensions in detail.

The next chapter discusses the implementation of these concepts in detail. In it we also scrutinize the results generated by the simulation of these concepts and algorithms to ensure the veracity of our design.

Chapter 4:

Performance evaluation for ad hoc network

As discussed in section 3.3 for ad hoc fault management we are using clustering algorithm. To support our proposed approach for network management of ad hoc network, we need to produce some results. By far we don't have any complete ad hoc implementation so we decided to go with simulation model that resembles complete ad hoc network. Using this model we can run simulations, and by varying different attributes and collect and compare statistics for different scenarios.

In our simulations, we have studied the effect of three different parameters: (1) The mobility parameter: We vary this parameter by varying the speed limit attribute (described in 4.2.3). Mobility is quantified in the form of a mobility factor, which expresses the average relative speed of a node during the simulation time, (2) the offered load parameter: the offered traffic is controlled through the number of flows in the network. For instance, increasing the number of flows increases it, (3) the co-ordinates of the cluster (area): we will discuss this in detail in section 4.2. This parameter can be set to different combination of length and height so that we can collect certain results. This will be helpful in setting up the threshold.

In the following section we will discuss the generalize ad hoc node model which resembles any node that can be a part of ad hoc network then we will discuss about the network scenario in which we are collecting all the results. Finally we will talk about the performance results and corrective measures.

4.1. Ad Hoc Network Node Model

As it is shown in figure 4.1, an ad hoc network node model tries to reproduce the so-called OSI stack. With focus being drawn by the application layer for the management solution, some layers have been legitimately omitted. In fact, the primary idea behind the development of the ad hoc network node model is to provide a test bed with the mac layer implementation for an ad hoc environment and the network layer with ad hoc routing (AODV).

Below is the list of the different modules that compose the ad hoc network's node model:

- *src module:* This is the packet source module, which generates packets according to the specific packet size and inter-arrival distributions. Once generated, packets are sent to the immediate lower layer (app_manager).
- *app_manager module:* The application manager module sets a random destination address to the incoming packet and generates a service request primitive to the routing layer in the form of an internal communication interface[***](ICI). Along with the ICI, the just received packet is sent to the AODV routing module. We are also using this module to collect all the information from the lower layers. It collects and processes this information, and based on that, it takes all the management decisions. The app_manager module helps improve the performance of the ad hoc network.

[***] An ICI is an interface, which allows two processes to exchange a user-defined information.

- *aodv_routing module:* This module receives the protocol data unit PDU from the application layer and executes the AODV routing algorithm as described in 4.2.2.
- *wlan_mac_intf module (provided by OPNET):* This module interfaces the lower layer module; it receives the packet from the aodv routing module, and hands it over to wlan_mac module, and vice versa.
- *wlan_mac module (provided by OPNET):* This module is an implementation of the IEEE 802.11 standard medium access control (MAC) protocol. Some modifications were added to the original model to enable some sort of interaction with the upper layers (especially with the aodv routing process). For instance, upon transmission failure, the current module hands over an ICI to the upper layers indicating the IP address of the unreachable node.

  As the IEEE 802.11 standard is widely used in test beds and simulation platforms, we had practically no choice but to integrate it within the current model.
- *wlan_rx + wlan_tx modules:* These modules are implementations of the IEEE 802.11 standard physical layer specifications.
- *mobility module:* This module performs the movement of the current node by changing its position periodically according to the actual movement scheme.

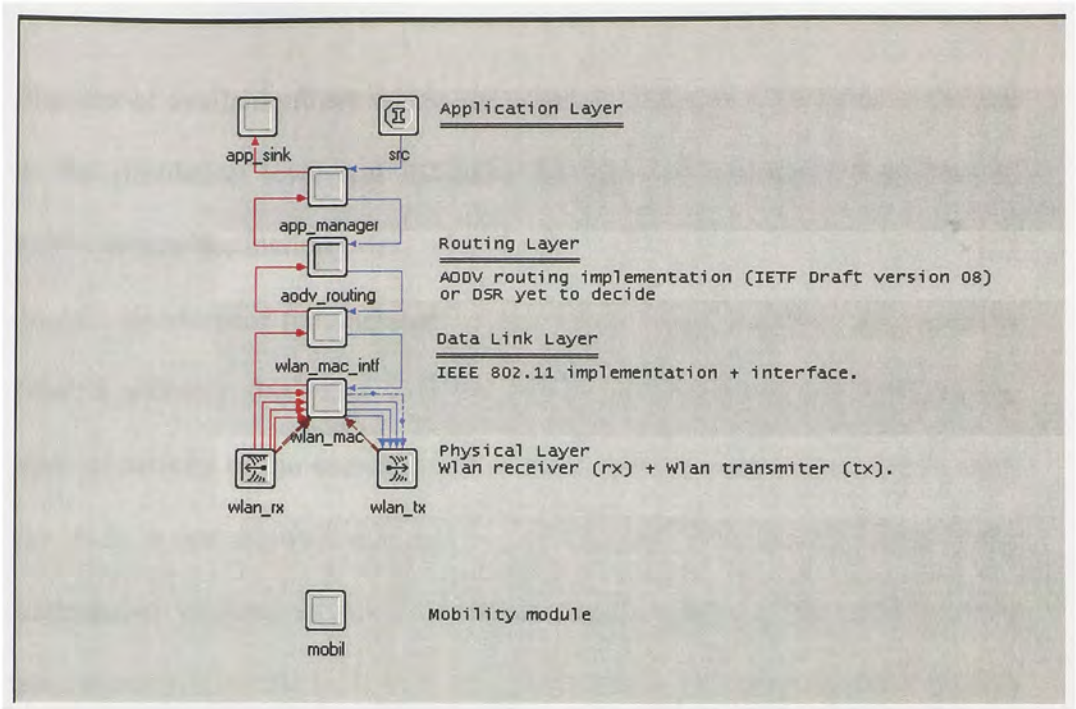


Figure 4.1 AODV Node Model

4.1.1. Ad Hoc Network Application Manager (app_manger) Process Model

As was previously mentioned, the primary function of the application manager process is to assign a destination IP address for each incoming packet. The other function of the app_manger is to "reply" to each received packet by emitting a data packet to the attention of the source node of the received packet. This mechanism was introduced in order to fake a two-way conversation between the members of each pair of source/destination.

Inputs

- Number of available flows within the network (integer): This value is defined at the simulation level and indicates the maximum authorized number of active source/destination pairs.
- Node's interlocutor (enumerated values: either None, Random, or a specific Node's address): this value is defined at the process level and indicates the state of activity of the current node. If the interlocutor attribute is set to zero, the node is not allowed to initiate a conversation with another node. If the interlocutor attribute is set to a specific node's address, then node is only authorized to converse with that same node. Finally, if the interlocutor is set to Random, the current node may (under the condition of flow availability) pick a random destination IP address and initiate a conversation with the corresponding station.

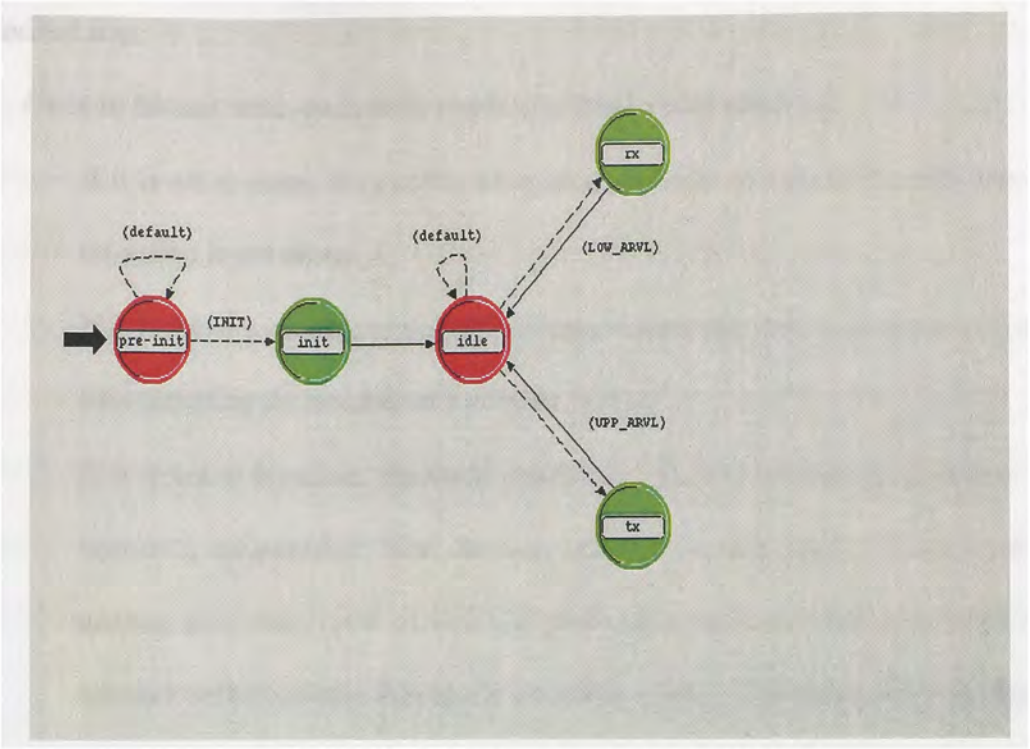


Figure 4.2 Application Manager Process Model

Description

The principle is very basic and can be divided into 3 steps.

First step:

On one hand, the attribution of the flows obeys a FIFO (first in-first served) policy: the first node transiting to the init state has a greater chance to occupy a free spot. On the other hand, each node, at the pre-init state, randomly picks a waiting period before transiting to the init state. The idea is to introduce some sort of discrimination between the existing nodes. Thus, as nodes consecutively transit in the init state, remaining flows are progressively granted to arriving nodes until no more flows are available. Of course, nodes with an interlocutor attribute set to a specific node's IP address automatically transit to the init state (waiting period at the pre-init state equals 0) and are consequently guaranteed to occupy a free flow.

Second step:

Once in the init state, each node checks its interlocutor attribute:

- If it is set to none, the current node automatically transits to the idle state and no action is are taken.
- If it is set to a specific node's IP address, the current node consumes a flow by decrementing the number of available flows.
- If it is set to Random, the node checks the number of remaining flows. Two scenarios are possible. First: At least one flow is unoccupied. In this case, the current node reserves it as in 2 and picks up a random destination node, with which it will converse during the simulation time. The interlocutor attribute is then switched from Random to that specific IP address and the node transits to the idle state. Second and last: All flows are reserved. In this case, the current node did not get lucky enough. It switches its interlocutor value to none and proceeds as in step one.

Final step:

The current node is in the idle state and can transit either to the rx (receiver) (upon packet arrival from the lower layer) or tx (transmitter) (upon packet arrival from the lower layer) state.

When a packet is received from the src (packet generator) module, the current node checks its interlocutor attribute. At this stage, only two values are possible for the interlocutor attribute: None or a specific destination IP address. If the value is none, the current node silently discards the data packet and return to an idle state. In the other case, the current node generates a service request primitive to the AODV

routing module and passes the received packet and the destination IP address (which is stored in the interlocutor attribute) as arguments. In practical, the current process installs an ICI including the destination IP address and sends it along with the data packet to the aodv routing process. After that, the node returns to the idle state.

In the rx state, the current node has just received a packet from the lower layer. As the packet reaches its destination, the current node simply destroys it. Also, if the received packet requires a response, the current node creates a new data packet and generates a primitive service request as described above. This time however, the newly created A-PDU will be destined for the originator of the just received packet.

4.1.2. Ad Hoc Network Routing (aodv routing) Process Model

The AODV routing process (Figure 4.3) implements the AODV routing protocol as specified in the IETF AODV draft version 08.

In our attempt to accomplish the implementation of the present model, we have tried to make the aodv routing process model as independent as possible from the rest of the platform (at the node model level). The only reason for using aodv as the underlying protocol is its ease of implementation and the fact that much work is being done on this protocol. Our concentration is more on the application layer and over-all performance evaluation of ad hoc networks so the routing protocol for our implementation is not that critical.

The remainder of this section presents a general description of the aodv routing process model. For further details, the reader may refer to the technical documentation of AODV [2].

Inputs

- AODV routing constants
- Current node's IP address (integer)
- Repair Parameter (enumerated values: enabled, or disabled): If the Repair attribute is set to Enabled, a node may perform a local repair following a link failure. Otherwise, the same node must generate a RERR packet and broadcast it to the neighboring nodes.
- Data buffer size (integer): This variable indicates the maximum number of data packets waiting for routes that can be stored in the internal queue. If an extra packet is received while the buffer is full, the most recently received packet is ignored (destroyed).
- Max node traversal time (double): Indicates the maximum time that a data packet is allowed to spend in the data buffer of a given node. At this point, the data packet is destroyed.
- Max buffer size (integer): Indicates the maximum capacity (in terms of number of packets) of the send buffer that is maintained by each node.

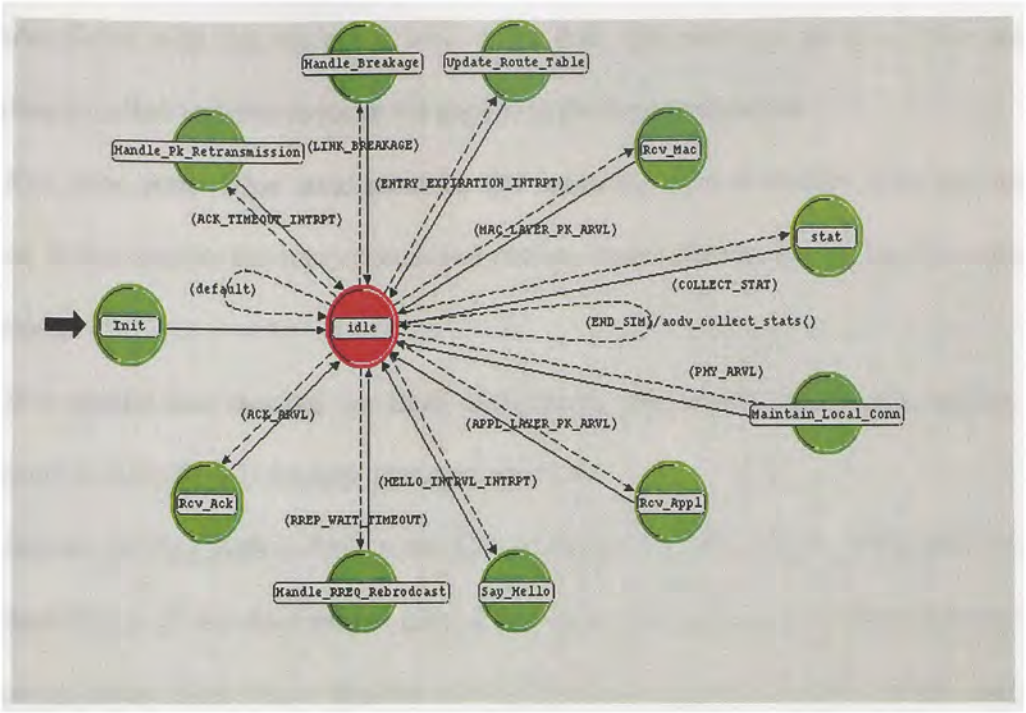


Figure 4.3 AODV Routing Process Model

Description

Init state: This state consists of the initialization of the process model. User defined attributes are loaded and different variables are initialized (routing table, pending request list, etc). In case Hello messages are selected, a self-interrupt is scheduled to initiate the first Hello Interval. Once the initialization step is accomplished, the process transits in the idle state.

Rcv_Appl state: The aodv_routine process transits to this state when a service request is received from the upper layer to transmit a data packet to a given destination. The current state first extracts the ICI which is associated to the just received packet and reads the destination IP address of that same packet. The current state then encapsulates the received data packet into an AODV-PDU and fills its

header fields with the correct values. After that, the aodv_pk_receive_from_appl() routine is called in order to route the packet to its final destination.

Rcv_Mac state: This state receives the incoming packet stream from the lower layer. It first checks the type of received packet, then calls the appropriate function to proceed.

If a packet has reached its final destination, the current state decapsulates its payload and sends it to the app_manager module.

Handle_RREQ_Rebroadcast state: Occurs when a RREP_WAIT_TIMEOUT timer expires for a given destination (the destination IP address is deducted from the interrupt code). This means that the current node has not yet received a route reply to its request. In this case, the current state checks whether a re-broadcast is possible or not (with regards to the number of retries threshold). If the maximum authorized number of retries is reached, the discovery process for that destination is aborted. Consequently, any data packet waiting for this route is dropped from the buffer. In the other case, a RREQ packet is rebroadcast.

Update_Route_Table state: This state occurs when the timer of an entry expires. Three cases are possible.

First: The expired entry is active. The node then invalidates it and schedules a new interruption for its deletion. Also, if the entry is flagged as broken, the node resets the breakage flag before it schedules the deletion interrupt. If the expired entry points toward a neighboring node, the current state either generates a RERR and broadcasts it to its neighboring nodes, or attempts to perform a local repair (according to the Repair attribute value).

Second: The expired entry is under repair. In this case, the current state delays the expiration time and waits till the end of the discovery process.

Third: The expired entry is invalid. In this case, the node simply deletes it from its routing table unless it is under repair.

Handle_Breakage state: This state is called when a failure to transmit occurs at the MAC Layer. In this case, the MAC Layer notifies the upper layer by sending a NACK message containing both final and next hop destinations of the lost data packet. At this point, two scenarios are possible:

Repair is allowed. In this case, the node performs a local repair by calling the aodv_initiate_maintenance() routine.

Repair is not allowed. In this case, the node generates a RERR packet by calling the aodv_rerr_pk_generate() routine.

Rcv_Ack state: Upon Ack reception from a given destination, the current state slides its transmission window and transmits the next waiting packet for that same destination.

Handle_Pk_Retransmission state: The process transits to this state when a node downstream does not acknowledge a data packet within the appropriate time frame. The current state queues a copy of the non-acknowledged data packet in the buffer and retransmits the data packet upon appropriate call for retransmission.

Say_Hello state: It has been hello_interval seconds since the last broadcast. The node must broadcast a hello message in order to advertise its presence in the neighborhood.

Maintain_Local_Conn state: When a packet is received at the physical layer, the latter notifies the routing layer indicating the IP address of the originator node. If the MAC connectivity support is selected, the current state updates its local connectivity map by updating its routing table entry for that same node.

Stat state: The current process periodically transits to this state in order to collect different global statistics. These statistics are written into a Comma Separated Values (CSV) file, which is created at the beginning of each simulation run. Other vector and scalar statistics are also collected through OPNET kernel procedures.

4.1.3. Ad Hoc Network Mobility Process Model

The mobility process model, shown in Figure 4.4, implements a random mobility scheme that is described below.

The general motion of a particular node is simulated through a set of discretized small step intervals. A node in motion updates its position every time step period of time. In our simulations, the duration of each step was set to a value of 0.2 seconds.

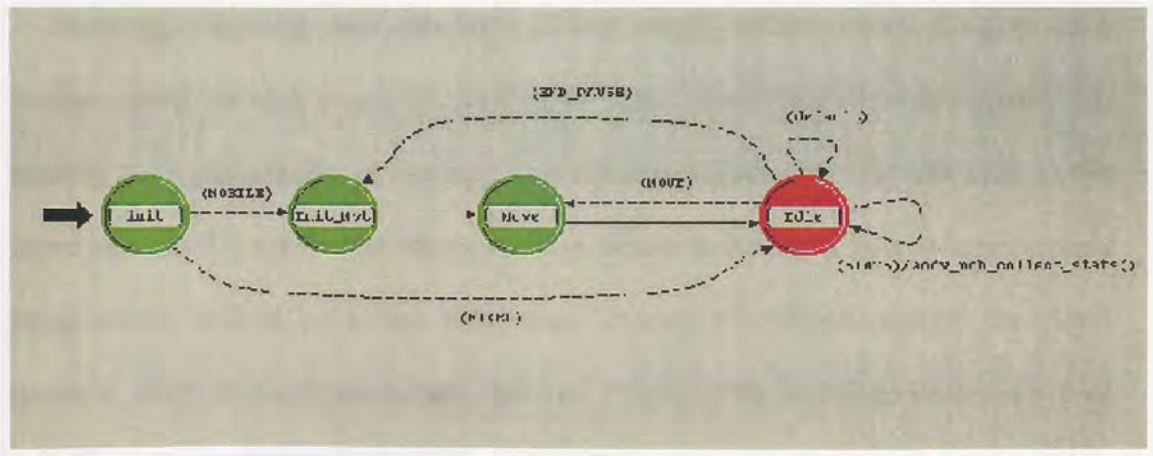


Figure 4.4 Mobility Process Model

Inputs

- Mobility attribute (enumerated values: Enable, or Disabled): Indicates whether the current node is fixed or mobile.
- Grid dimensions: Each mobile node moves around the specified area.
- Speed limit: Maximum speed that a node in motion may reach.
- Pause time: After reaching a target position, a moving node must pause during this period of time before reaching for a new target position.

Description

In the init state, each node picks a random position within the specified grid. After that, each node checks the mobility attribute in order to determine whether it should move or not: If the mobility attribute is set to Disabled, the current node transits immediately to the idle state and remains at the same position throughout the simulation time. In the other case, if the mobility attribute is set to Enabled, the current node transits to the init_mvt state in order to initialize its next movement parameters.

Basically, a moving node chooses a random target position within the grid and a random speed between 0 and the speed limit value. Given these two parameters, the moving node periodically (every step time period) transits from the idle state to the move state until it reaches the target position. While in the move state, a moving node progressively travels by a step time speed amount of distance toward the target position. After each step movement, the node checks if the target has been reached or not. If so, the current node transits to the idle state and enters a pause time phase. When the pause period arrives at its term, the END_PAUSE condition becomes true

and the current node transits to the init_mvt state in order to plan the next trip. On the other hand, if the target position still has not been reached, the current node returns to an idle state and waits for the next step time before returning to the move state.

During the nodes movements, the aodv_mob_collect_stat() routine is periodically (after each step time) called to monitor the actual topology of the network. This provides an instant view of link changes along the simulation time. These statistics are also reported into the CSV file generated at the end of each simulation run.

4.1.4. WLAN-MAC and WLAN-MAC interface Process Model

This simulation model uses 802.11b Wireless LAN module provided by OPNET as a MAC layer, and Wireless MAC interface module for communication between MAC layer and upper layers provided by OPNET. For a detailed explanation of these modules you can refer to OPNET documentation.

4.2. Simulation model

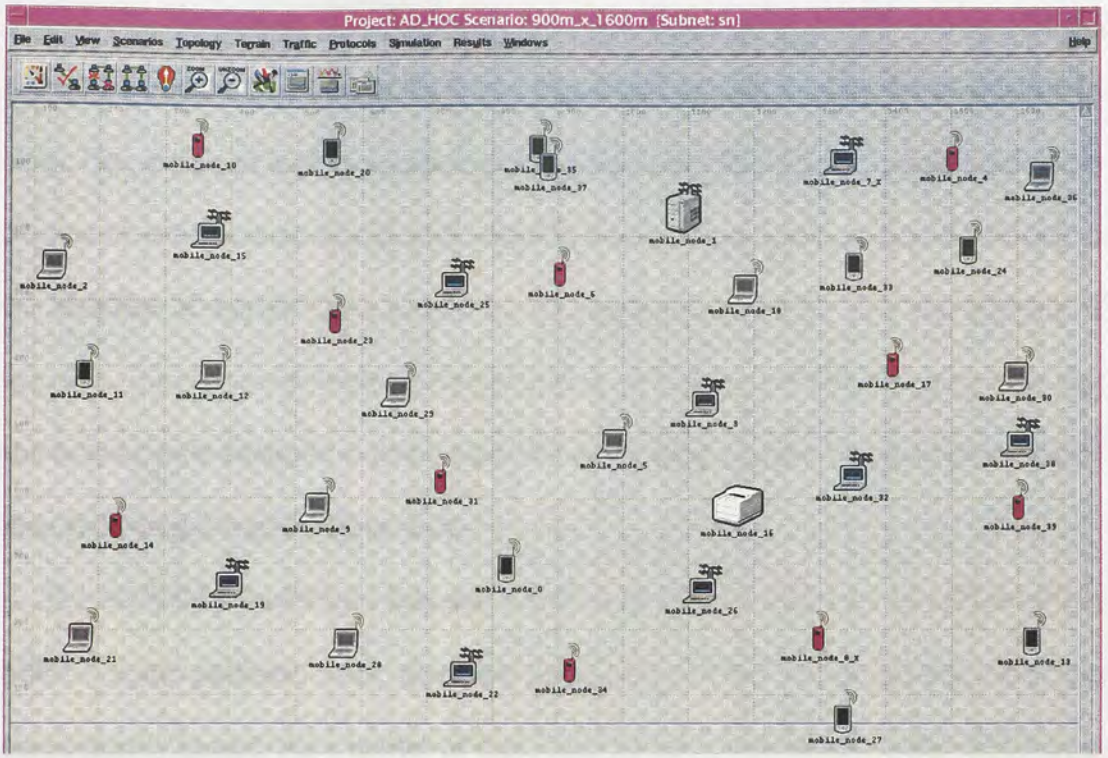


Figure 4.5 Network Model

In all the simulation runs, the same network model that is shown in Figure 4.5 was used. The network contained 40 mobile nodes which can move around a 1000 x1000 meters square wide area. Nodes communicate over wireless links with a transmission range of 250 meters. The AODV layer maintains a send buffer of 64 packets, and the node traversal time is set to 30 seconds (a packet may remain within a send buffer for a period of 30 seconds at most).

As far as the traffic model is concerned, we used Continuous Bit Rate (CBR) sources with a rate of 4 packets per second. Packet size is constant and is equal to 512 bytes, with each run being a simulation of 900 seconds.

The implementation of 802.11 provided by OPNET showed a lot of dysfunctions during our simulations. For instance, the activation of the virtual carrier sensing (RTS/CTS handshake) always leads to a segmentation fault causing the simulation to stop. The source of such a problem is not quite clear. However, it seems that nodes happen to emit packets of size null while deferring. These packets, when received at the interface of a given node, are put in a re-segmentation buffer. The packet being of size null, the re-segmentation routine returns a null packet pointer. The next time that the current node attempts to manipulate this null packet pointer, the simulation would stop following a segmentation fault.

The RTS/CTS handshake scheme, which is supposed to reduce the effect of the hidden node problem, was therefore deactivated. The results shown later are consequently biased. As a matter of fact, each transmission failure at the MAC layer is reported to the AODV layer as a link breakage. In such a situation, the latter undertakes the maintenance procedure (as described in Section 3.3), which leads to new route discoveries.

The RTS/CTS exchange being deactivated, the chances for link failures because of the hidden node problem are greater and the situation described above is likely to happen more often.

4.3. Performance Results

Based on the above node model, we have simulated several scenarios using different criteria, and have collected various statistical data. This will help us in making decisions regarding ad-hoc network management.

In section 3.3.3 we are proposing a clustering algorithm, which takes care of generation, maintenance, merging and splitting of a cluster (geographically). To analize proposed criteria, here are some of the results we have collected using the ad hoc simulation model in OPNET.

Different metrics were collected in order to evaluate the network performance in various situations. The most important metrics are listed below:

- Efficiency – The ratio of delivered data packets to those offered to the network.
- Normalized overhead load — The amount of control traffic generated (in bits) per data traffic delivered (in bits).
- Average end-to-end delay for data packets — This includes all possible delays from the moment the packet is generated to the moment it is received by the destination node. This metric is expressed in seconds.
- Average hop count — Average number of hops traveled by data packets.
- Average discovery period — Average duration of a discovery period in seconds

In the following section, based on these metrics, a number of statistics are collected and discussed to validate our design and assumption mentioned in chapter 3.

4.3.1. Varying clusters height: length ratio

In section 3.3.2, while proposing a solution for maintenance of clusters in section 3.3.3.2, we define some criteria to check before taking any correcting measure. Among them, one of the criteria is that the ratio of height and length should not be more than the threshold value.

To obtain the performance of the network with different combinations of length and height we are using a network model which contains 40 mobile nodes and can move around a X x Y meters square wide area. Nodes communicate over wireless links with a transmission range of 250 meters. The AODV layer maintains a send buffer of 64 packets, and the node traversal time is set to 30 seconds. As far as the traffic model is concerned, we used continuous bit rate (CBR) sources with a rate of 4 packets per second. Packet size is constant and is equal to 512 bytes. We ran each simulation for 30 minutes for all the scenarios.

Each scenario has constant area (14400 m^2.) and different combinations of length and height such as:

- 900m x 1600m
- 1200m x 1200m
- 2400m x 600m

We observed that the discovery time and average end-to-end delay for data packets increases significantly as the ratio between them increases, and as throughput decreases. This signifies that if the box is very narrow, it degrades the performance of

the network as well makes it difficult for the cluster head to maintain that cluster without any node being left out or with minimal delay.

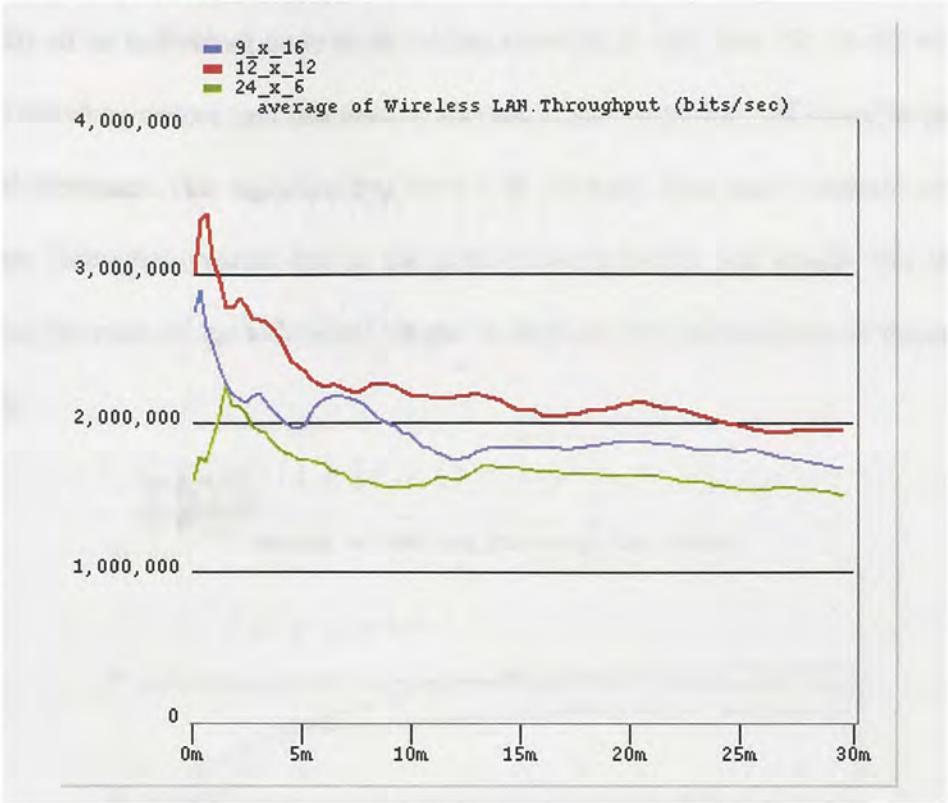


Figure 4.6 Average Throughput V/S Network Life Time

As we increase the ratio between length and heights throughput decreases, while the area in which the network is constrained is constant. As shown in figure 4.6, throughput decreases significantly as the ratio increases. When the ratio between length and height of the cell is 1, throughput is highest. As evidently shown in the figure throughput decrease by more than 25% when the ratio increases from 1 to 4.

We ignore the results collected in the first couple of minutes. They are not valid because as per the mobility algorithm, initially all the nodes are still, and then sequentially they select a random destination and move towards it in a fixed step. A

decrease in throughput with ratio is basically due to the more number of hops it takes for a packet to reach from source to destination. As we have already discussed, reliability of an individual node in an ad hoc network is very low. So, as the number of hops between source and destination increases, the probability of a packet getting dropped increases. This signifies that even with constant area and a constant number of nodes, throughput varies due to the ratio between height and length. We should minimize the ratio of the individual cluster to improve the performance of the ad hoc network.

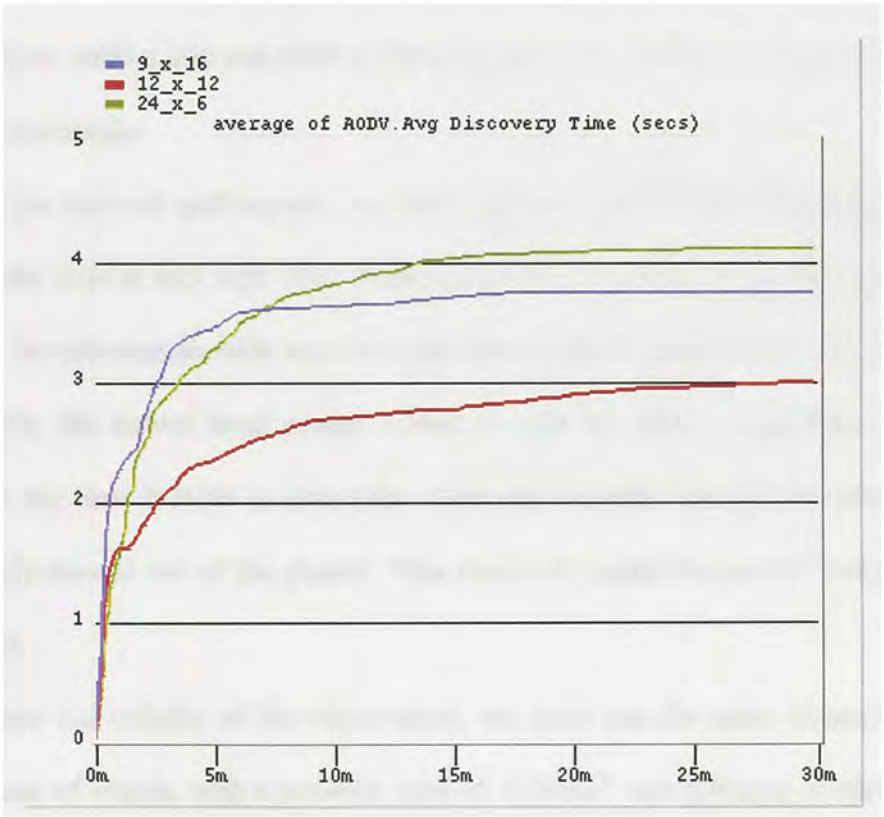


Figure 4.7 Average Discovery time V/S Network Life Time

Average discovery time shown above is an average of the total time taken by nodes to discover another node in the network at a particular time. As we increase the ratio of height and length, average distance between two nodes increases along with

the number of hops a packet has to travel to reach a particular destination. In ad hoc networks, as the number of hops increases, the likelihood of the packets getting lost increases. Our management protocol is connectionless, which raises the chances of the node being unmanaged, or the rediscovery of the node increases the average discovery time.

Figure 4.7 shows that even the discovery time increases more then 25% as the height to length ratio increases. In figure 4.7, discovery time for the same network in the area of 1200 x 1200 meter gradually increases and stabilizes to 3 seconds. As the ratio increases 1600 x 900 and 2400 x 600 avg. discovery time increases to 3.5 and 4 seconds respectively.

As per our network architecture, we cannot afford to have high discovery time. If the discovery time is very high, that means the amount of time it takes for a cluster to collect all the information from nodes is high. Due to the volatile nature of the ad hoc environment, the cluster head cannot afford to wait for such a long time. This is because in the time it takes to detect the node and send the packet, the node might have already moved out of the cluster. This results in raised chances of nodes being unmanaged.

To ensure the validity of the observation, we have run the same simulation for different sets of values, with a network area of $12000m^{2,}$ and different combinations of length and height, 2400 x 500,1200 x 1000 and 1500 x 800 respectively. A collected result confirms the observations by proving the same conclusion.

More performance data is being collected at different speeds and so far it indicates the same results as when the speed increase: throughput decreases, and discovery time increases.

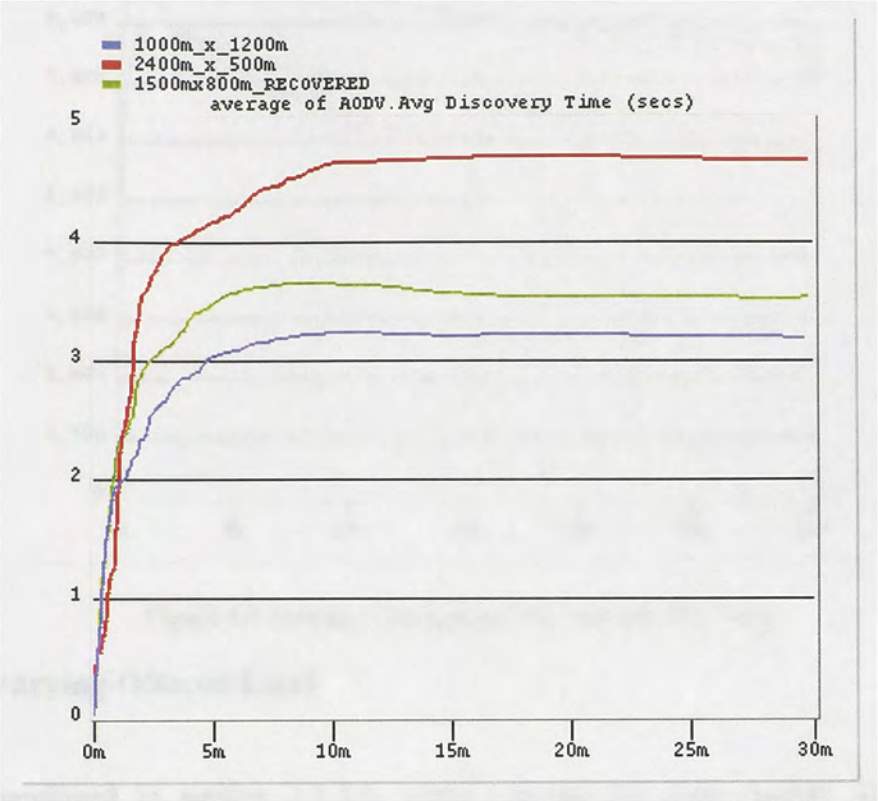


Figure 4.8 Average Discovery time V/S Network Life Time

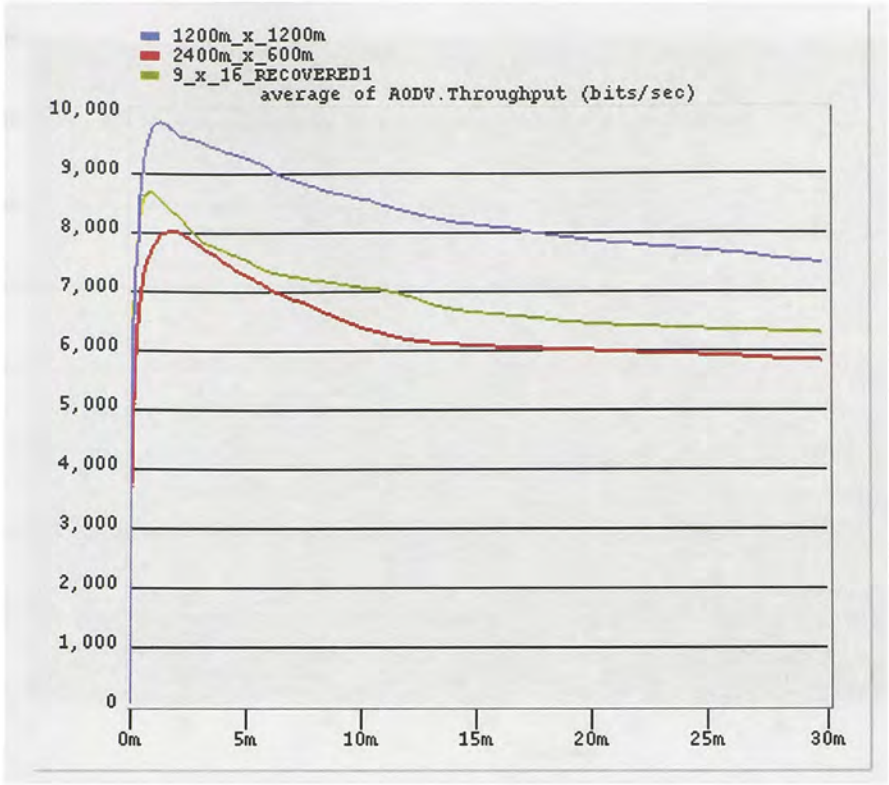
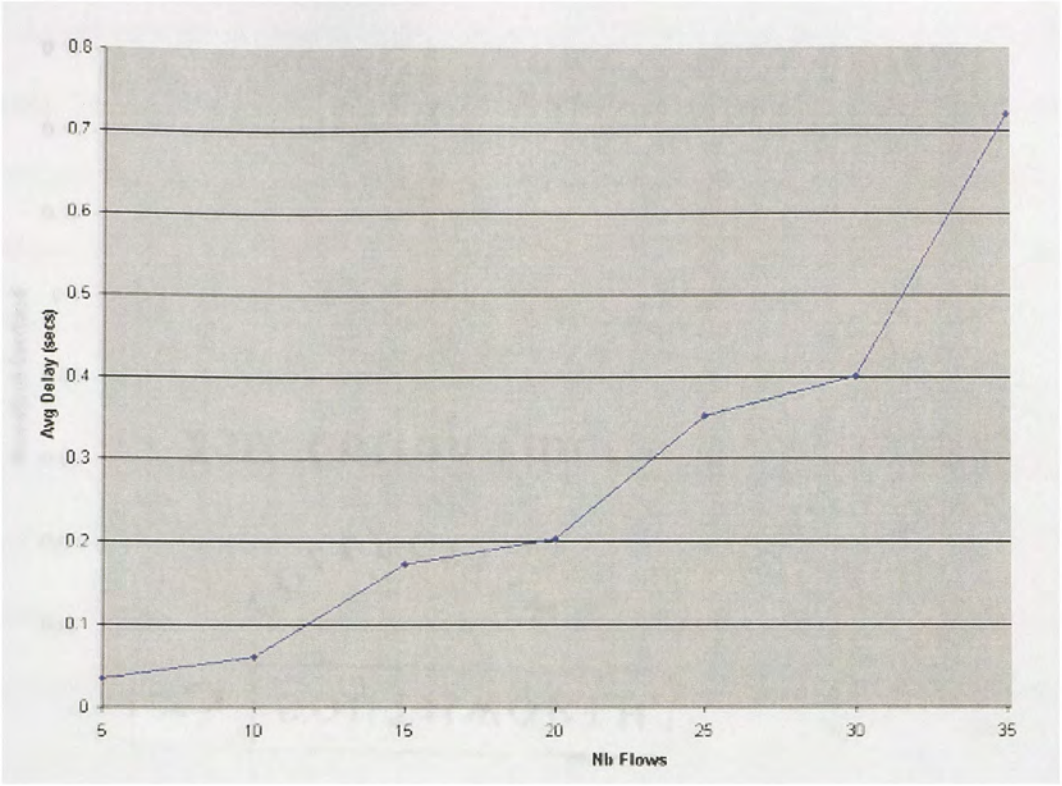


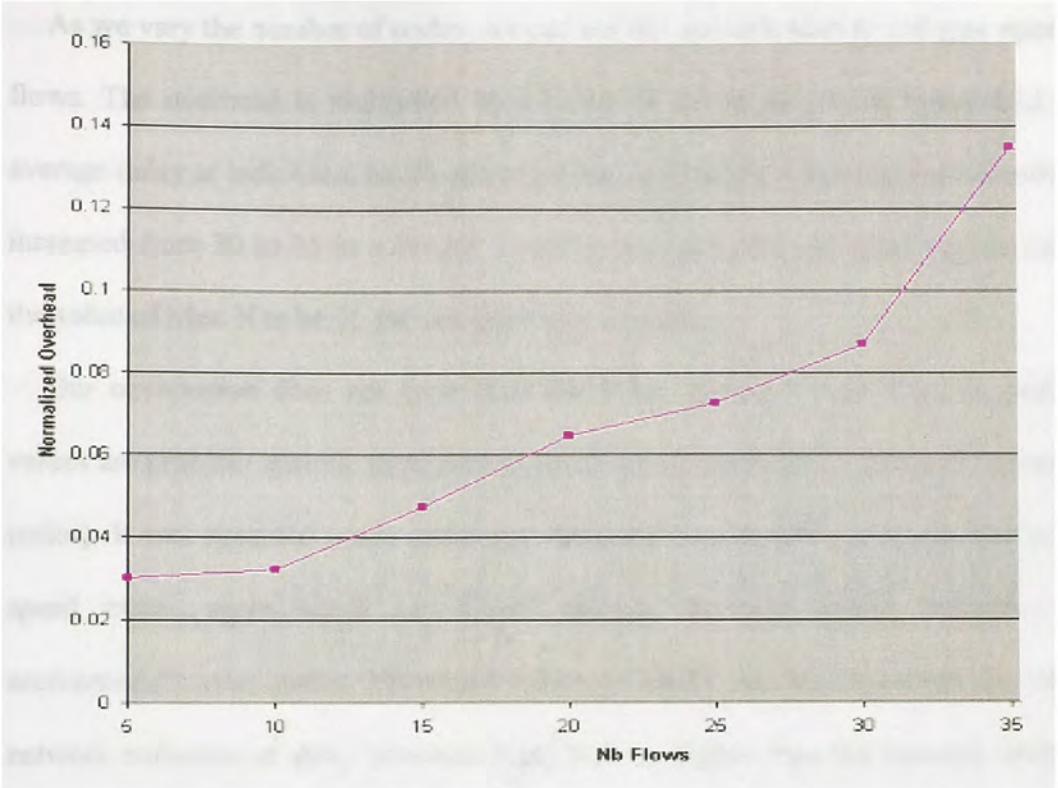
Figure 4.9 Average Throughput V/S Network Life Time

4.3.2. Varying Offered Load

As mentioned in section 3.3.3.3, while running the node density adjustment algorithm, we have to define Min N and Max N. To find those values we have run the simulation for a different number of nodes and collected statistics for a number of nodes v/s average delay and a number of nodes v/s normalized overhead. Defining Min N is an open problem, yet we will assume here that the value of Min N is 5.



4.10 Average Delay v/s Offered Load



4.11 Normalized Overhead v/s Offered Load

As mentioned in section 3.3.3.3, while performing the box refinement procedure we have to define a threshold for the number of nodes allowed in a particular cluster to avoid degradation in network performance and to avoid access load on the cluster head.

In this section while collecting performance data, we vary the number of computers in a fixed area. The maximum speed is set to 0.5 m/s in all runs, which corresponds to a mobility factor of around 0.24, with an interval of 0.01, and transmission range of the individual nodes set to 250m (average range based on the most adopted wireless standards 802.11b).

As we vary the number of nodes, we can see the network start to collapse upon 35 flows. The overhead is multiplied by a factor of 4.5 as shown in figure 4.12 and average delay at individual nodes increases very drastically when number of nodes is increased from 30 to 35 in a cluster. Based on the gathered statistics, we can define the value of Max N to be 35 for this particular scenario.

Our observation does not generalize the value of Min N and Max N. Defined values are problem specific (with assumptions about transmission range of individual nodes). It was observed while collecting statistics for a number of nodes that as the speed varies, again MinN and MaxN change. At high speeds, networks can accommodate more nodes. Threshold values of MinN and MaxN (where the whole network collapses or delay becomes high) will be higher than the network with the node at reduced speed.

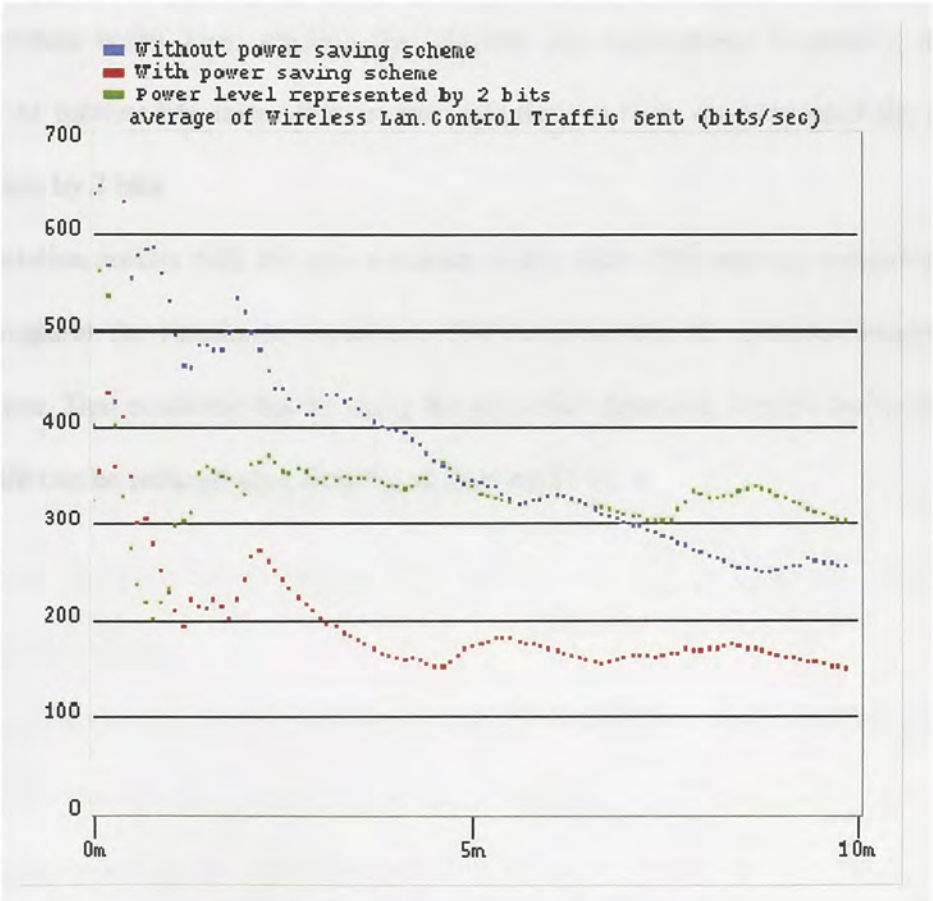
4.3.3. Battery Power

After simulating the technique discussed in 3.3.5 for representing the battery life, we observed that control traffic transmitted and received across the network reduces significantly. Every node has to transfer remaining battery power information to its cluster head and every cluster head shares that data with other cluster heads and MOM. When a node moves from one cluster to another, as it powers on, or goes out of range and comes back into range. time, the battery information is transmitted upwards to the cluster head and then to MOM.

Besides this, when the node moves from one cluster to another, the cluster head of that cluster runs the CHD (cluster head determination) algorithm to determine whether the newly arrived node is superior to the cluster head of that cluster. While

performing CHD if the node determines the new node is superior, it transfers control to the new node, then each node has to transfer their battery life information to the new node. Taken as a whole, it turns out to be a considerable part of control traffic on the network.

To validate our scheme we first ran the simulation with standard technique, then with modified standard technique where battery life representation is performed using only 2 bits instead of actual information. Later simulation was run with the newly proposed approach.



4.12 Average v/s Network Life Time

As revealed in the figure 4.12, received control traffic reduces when battery life information is represented only using 2 bits as discussed in 3.3.5. This is every time the node exchanges battery life information, unlike traditional networks, mobile nodes only need to transfer two bits of information. We ignore the results for the first two minutes of the simulation because of initialization irregularities. As we can see if we use the two-bit representation for battery life, it reduces the average wireless LAN control traffic sent. Results show that after seven-eight minutes control traffic sent by the old method decreases, Due to a lesser number of remaining nodes. In the old scheme where nodes keep sending their battery life information frequently, nodes runs out of battery life faster than in the scheme in which you can send the same information by 2 bits.

Simulation results with the new schemes clearly show that average control traffic sent throughout the simulation reduces to 200 bits/sec/node to represent battery life information. That confirms that by using the proposed approach, control traffic due to battery life can be reduced significantly, at least by 33 %.

Chapter: 5

Conclusion

Ad hoc network management as an emerging research field has attracted lots of attention. We cannot achieve our goal just by applying the existing network management protocol without significant functional enhancement. This is because an ad hoc network has several specific characteristics that were not considered when people made network management protocols. In this thesis, we have presented our solution: a distributed + hierarchical architecture for ad hoc network management. We absorbed the advantages from both the distributed system and the hierarchical system. By using the distributed system, we distribute the management burden through the whole network. Multiple cluster heads in the system help achieve the whole network message efficiency and improve the whole network scalability, reliability and robustness. Still the distributed system has an inherent drawback, which is its lack of the whole network management information. We use the hierarchical method to overcome this, storing the whole network summary information in MOM.

We adopted the idea of geographical clustering from [3] as the basis for the whole system initialization step, but we made some changes, such as the policy for node density adjustment, on the algorithm to satisfy our requirement. We also described in depth distinctive situations that need the fault/configuration management for cluster members, cluster heads, and MOM.

Chapter: 6

Future Work

We provided our solutions for issues concerning power such as switched off, abruptly dies, movement such as moving in/out clusters etc. We also suggested some power conservation scheme that compliments any routing protocol.

There are still lots of issues, which need to be explored about ad-hoc network management. Since the situation for fault/configuration management is so complex, we ignored some details, such as the network transfer delay, in our design. However, it should be considered in future work, because in situations when old faults have not been solved and new faults are generated during the delay interval, it is important to analyze what the critical information is and how to use it to fix the problem. In addition, we used GPS as the basis for a clustering algorithm. The limitation of GPS, that, there is no way to inform whether there is a barrier between objects, affects the generation of clusters. Furthermore, we need implement our design to evaluate the performance, rather than simulating it.

Reference

[1] S. Basagni, I. Chlamtac, and A. Farago. A generalized clustering algorithm for peer-to-peer networks. Workshop on Algorithmic Aspects of Communication July 1997

[2] C. E. Perkins, E. M. Royer and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Internet-Draft of IETF MANET Working Group, Jul. 2000.

[3] Wenli Chen, Nitin Jain, Suresh Singh, ANMP: Ad Hoc Network Management Protocol, IEEE Journal on Selected Areas in Communications, Vol.17, NO. 8, Ausgus 1999.

[4] Ioannis Chatzigiannakis, Sotiris Nikoletseas, Paul Spirakis, Analysis of an Innovative and Efficient Communication Strategy for Hierarchical Ad-hoc Mobile Networks, Proc. 8th Panhellenic Conference on Informatics, Nicosia, Cyprus, Nov. 2001.

[5] Steven M. Dauber, Finding Fault, BYTE Magazine, March 1991.

[6] James Herman, Distributed Network Management, Data Communications Magazine, pp74-84, June 1992.

[7] David B. Johnson, Routing in Ad Hoc Networks of Mobile Hosts, Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Dec. 1994.

[8] Mohsen Kahani, H.W. Peter Beadle, Decentralised Approaches for Network Management, ACM SIGCOMM Computer Communication Review, 27(3), July 1997.

[9] Leinwand, A., Fang, K., "Network Management: A Practical Perspective", Addison Wesley, 1993.

[10] Network Management Presentation, http://www.10.edu/~network/presentations/course/appendix/appendix_h

[11] S. Corson and J. Macker, "Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations (Internet-Draft)," Mobile Ad-hoc Network (MANET) Working Group, IETF, Oct. 1998.

[12] Z.J.Haas and S.Tabrizi. On Some Challenges and Design Choices in Ad-hoc Communications. IEEE Military Communications Conference Proceedings, 1:187-192, 1998. 9

[13] T. Forde and L.E. Doyle and D. O'Mahony, "*An Evaluation System for Wireless **AdHoc Network** Protocols*," Irish Signals and Systems Conference, June 2000, pp 219-224.

[14] IEEE 802.11 Tutorial by Jim Zyren and Al Petrick http://www.wirelessethernet.org/downloads/IEEE_80211_Primer.pdf

[15] S. R. Das, R. Castaneda, and J. Yan. Simulation based performance evaluation of mobile, ad hoc network routing protocols. ACM/Baltzer Mobile Networks and Applications (MONET) Journal, pages 179-189, July 2000.

[16] SAHAI, A., BILLIART, S., AND MORIN, C. *Astrolog: A distributed and dynamic environment for network and system management*. In Proceedings of the 1st European Information Infrastructure User Conference (February 1997).

[17] Kahani, M., Beadle, H.W.P.: Decentralised Approaches for Network Management. Computer Communications Review, ACM SIGCOMM, Vol. 27 N.3 July (1997)

[18] Adler, M. & Scheidler, C. (1998). Efficient Communication Strategies for Ad-Hoc Wireless Networks. In: Proceedings of SPAA 98. ACM Press.

[19] P. Gupta and P. R. Kumar, \The capacity of wireless networks," IEEE Trans. Inform. Theory, vol. IT-46, no. 2, pp. 388-404, March 2000.

[20] Aiko Pras. network management architecture. PhD thesis, Universiteit Twente, Department of Computer Science, February 1995. available from http://wwwsnmp.cs.utwente.nl/~pras/thesis.html.

[21] J. Meng, A. Helal, and Stanley Su, "An Ad-Hoc Workflow Architecture Based on Mobile Agent and Rule-Based Processing," The special session on Software Agent-Oriented Workflows, Proceedings of the International Conference on Parallel and Distributed Computing Techniques and Applications, Las Vegas, Nevada, June 2000.

[22] Y. Xu, J. Heidemann, and D. Estrin, "*Geography-informed Energy Conservation for Ad Hoc Routing*," Proc. of the International Conference on Mobile Computing and Networking, pp. 70--84, 2001.

[23] P. Havinga and G. Smit, "*Energy-efficient Wireless Networking for Multimedia Applications*", in Wireless Communications and Mobile Computing, Wiley, 2000.

[24] F. Stamatelopoulos, N. Roussopoulos, and B. Maglaris. Using a DBMS for hierarchical network management. Engineer Conference, NETWORLD +

INTEROP'95, March 1995. http://www.netmode.ece.ntua.gr/papers/interop e16.eps. 17

[25] S. F. Bush, S. Jagannath, J. B. Evans, V. Frost, G. Minden, and K. S. Shanmugan. *A Control and Management Network for Wireless ATM Systems.* ACM-Baltzer **Wireless** Networks (WINET), 3:267,283, 1997. URL: http://www.ittc.ukans.edu/,sbush.

[26] J.J Garcia-Luna-Acceves,"Aunified Approach to Loop free Routing Algorithm Using Distance Vector or Link States," ACM SIGCOMM Symposium on Communication, Architectures and Protocols, Sep.,1989.

[27] Nitin H. Vaidhya,"A Case of Two-Level Distributed Recovery Schemes," ACM SIGMETRICS, May 1995.

[28] Nitin H. Vaidhyaand D.K.Pradhan, "Roll-Forward Checkpointing Scheme: A Novel Fault-Tolerant Architecture," IEEE Transactions on Computer, pp.1163-1174, Oct.1994

[29] R.Koo and Toueg, "Checkpointing and rollback recovery for distributed system," IEEE Trans. On Software Engineering, Vol.SE-13, No. 1,pp.23-31, Jan 1987

[30] A Acharya and B. R. Badrinath, "Checkpointing Distributed Applications On Mobile Computers, "Proc. of the Third Intl. Conf. on Parallel and Distributed Information System, pp. 73-80, Sep.1994.

[31] S. Alagar et. al., "Tolerating Mobile Support Station Ailure ," Computer Science Technical Report, Univ. of Texas and Dallas, November,1993

[32] S.-J. Lee, W. Su, and M. Gerla, "Ad hoc Wireless Multicast with Mobility Prediction," In Proceedings of IEEE ICCCN'99, Boston, MA, Oct. 1999, pp. 4-9"

[33] J. Moy, "Multicast Routing Extensions for OSPF," Communications of the ACM, vol. 37, no. 8, Aug. 1994, pp. 61-66, 114.

[34] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT) – An Architecture for Scalable Inter-Domain Multicast Routing," In Proceedings of ACM SIGCOMM'93, San Francisco, CA, Oct. 1993, pp. 85-95.

[35] S. Deering, D.L. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "The PIM Architecture for Wide-Area Multicast Routing," IEEE/ACM Transactions on Networking, vol. 4, no. 2, Apr. 1996, pp. 153-162.

[36] S.E. Deering and D.R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," ACM Transactions on Computer Systems, vol. 8, no. 2, May 1990, pp. 85-110.

[37] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar. *Power Control in Ad-Hoc Networks: Theory*, Architecture, Algorithm and Implementation of the **COMPOW** Protocol. In Proceedings of European Wireless Conference, pages 156--162, 2002.

[38] J. Holtzmann, S. Nanda, and D. Goodman, "***CDMA power control*** *for wireless networks*, " in Third Generation of Wireless Information Networks, pp. 299--311, Kluwer Academic Publishers, 1992.

[39] M. Jiang, J. Li, and Y.-C. Tay, "*Cluster based routing protocol (CBRP) functional specification (Internet-Draft)*," Mobile **Ad-hoc Network** (MANET) Working Group, IETF, Aug. 1998.