Design Issues and Implementation of Quality of Service Aware Routing and Resource Allocation in Mobile Ad-Hoc Networks

THESIS

Submitted in partial fulfillment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

MURALI P

Under the Supervision of
Prof. Chittaranjan Hota

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN) INDIA

July 2014

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI, RAJASTHAN

CERTIFICATE

This is to certify that the thesis entitled "**Design Issues and Implementation of Quality of Service Aware Routing and Resource Allocation in Mobile Ad-Hoc Networks**", submitted by **Murali P** ID.No **2006PHXF018P** for the award of Ph.D. degree of the Institute, embodies original work done by him under my supervision.

Signature of the supervisor : _____

Name : **DR. CHITTARANJAN HOTA**
Designation : Associate Professor,
Dept. of Comp. Sc. & Info. Sys.,
Associate Dean, Admissions,
BITS Pilani, Hyderabad Campus,
Hyderabad, A.P.

Date : _____

Dedicated To

my hero, my late father

Acknowledgements

I would like to express my gratitude to several personalities for their constant guidance, help, support and well-wishes.

I acknowledge with great pleasure, my deepest sense of gratitude to my supervisor Dr. Chittaranjan Hota for his constant encouragement, valuable guidance, and inspiring suggestions throughout the course of this work. His mere presence and timely suggestions encouraged me to keep the work rolling. He gave the direction, and clues that made me go deeper into the work. He never let me stagnate but kept me on the move.

I would like to thank my DAC members, Prof. Sundar Shan Balasubramaniam and Prof. G Raghurama for reviewing my proposal and for their valuable inputs throughout the thesis work. I feel indebted to colleagues Prof. Anil Agrawal, Prof. Biju Raveendran, Prof. Rahul Banerjee and Prof. Hari Babu for sharing their views on some of my work and encouraging me. I would like to thank Prof. J.P Mishra, Prof. Navneet Goyal, Prof. TSB Sudarshan and Prof. Sudeept Mohan for their constant encouragement and help. I would like to thank Prof. Poonam Goyal, Prof. Mukesh Rohil, and Prof. Yashwardhan Sharma for encouraging me to complete my thesis.

I would like to thank my students Prasoon, Rakesh Kumar, Rusheel Jain, Kashyap, Himanshu, Shrainik, Karthik, Raj Kiran and Vatsal for their sincere project work. They helped me advance the work by acting as bouncing boards for refining my understanding of the problems. I would like to thank Prof. Anti-Ylaa Jaaski and Mr. Andreas Arjona for their valuable comments on some of the work. I thank all the anonymous reviewers who spent their valuable time to read in detail and suggested several ways of improvement. I

thank Abhishek Varshney for helping me in converting word documents into Latex.

Thanks are due to Prof. B N Jain, Vice-chancellor and Prof. G. Raghurama, Director for the constant support and concern. I would like to gratefully acknowledge Prof. J.P Mishra, Unit Chief, IPC and many other administrators for providing me with a suitable working atmosphere. I thank all my colleagues in our department and those known to me in other departments for their good wishes.

Words fail to express my gratitude to my parents Shri (Late) B Parameswaran and Smt. Jayalekshmi for their love, care and patience. Without their love and constant support I would not have reached so far. I thank my sister Seetha and my nephew Manav and niece Mansi for their affection.

Abstract

Mobile Ad-hoc Networks(MANETs) consist of mobile nodes interconnected by multihop-communication paths with no fixed network infrastructure or administrative support. Due to ease of deployment and their self-organizing behaviour, they are used in various application domains such as battlefields, search-and-rescue operations, and collaborative computing. Tactical scalable MANETs have found larger acceptance in the armed forces, especially for deployment in large cargo ships, rugged terrains, urban warfare and large coastal areas. Due to their capability for self-configuration with minimal overhead, infrastructure-less nature and flexibility; domains such as search-and-rescue, crowd control and commando operations use MANETs for deploying their computational resources on the field. Due to node mobility, frequent transient link failures and resource constraints associated with mobile nodes, the problem of routing in such a network faces several challenges. For better communication among nodes having specific traffic requirements like streaming video, Quality-of-Service(QoS) is to be guaranteed for the network communication. Due to the naturally distributed manner of operation of nodes in a MANET, any resource allocation mechanism shall require a distributed algorithm. Within the ambit of resource allocation, the specific problem of distributed mutual exclusion for accessing critical resources in the network raises interesting challenges. In this thesis, we have focussed on the QoS-aware multicast routing problem and the resource allocation problem, specifically the distributed mutual exclusion problem, in MANETs.

In multicast communication, a single source node communicates to a subset of nodes in the network. While providing QoS for such a multicast com-

munication, we have focussed on energy-based QoS parameters in our work, which is relevant for mobile devices. A mobile node expends energy while it is communicating, idling or performing computations. While attempting to provide QoS aware multicast routing in MANETs, we encountered three sub-problems. The first sub-problem related to finding an appropriate cost metric for energy-aware routing in MANETs. Transient link failures due to mobility of nodes is one of the key challenges in routing in MANETs. This brings to focus the necessity of an appropriate route maintenance technique in any routing algorithm for MANETs. Maintenance of routes should consider energy consumption while re-establishing routes in the wireless scenario. The second problem we explored is related to the impact of energy in route maintenance in such a mobile environment. With the focus shifting towards incorporating mobile nodes with energy-harvesting devices, any algorithm that looks at energy as a QoS parameter must be able to handle this type of device. Energy budgeting involves observing the manner in which the wireless devices are charged and the avenues of energy consumption. This necessitates an appropriate model for improving energy efficiency in the presence of energy-harvesting devices. We have addressed these three sub-problems for QoS-aware routing in MANETs. As a result, we proposed a new cost metric and a new route maintenance mechanism using a Bayesian approach in two new routing algorithms. We then proposed a new energy model for a simplified version of MANETs, namely wireless sensor networks, to incorporate energy budgeting in the presence of energy-harvesting devices.

While addressing the QoS-aware multicast routing problem in MANETs, we focussed on the geographic routing approach that uses the location of nodes for performing the routing operation. Within the geographic approach, we found that a specific technique called the virtual force technique to solve the routing problem had been used to address problems such as unicast routing and node deployment. However, the multicast routing problem has never

been addressed using this technique due to the presenece of multiple destination nodes. We adapted the virtual force technique to enable multicast routing in MANETs, by using the notion of dampening forces to include QoS parameters in the system model for the multicast routing problem. While observing the trends in multicast routing, we found that research were limited to using quadrants or sectors of angle $\frac{\pi}{2}$. We used adaptable variable sector angles for multicast routing in mobile ad-hoc networks. We found further scope for improving routing algorithm by dividing the network into smaller regions and applying virtual force technqiue for communication among the regions. We thus proposed and analyzed three new virtual force based QoS-aware multicast routing algorithms in this thesis.

In the final part of this thesis, we explored resource allocation in MANETs, specifically the distributed mutual exclusion problem for MANETs. We focussed on the permission-based approach for solving the distributed mutual exclusion problem. Within the permission-based approach, we explored the "look-ahead" technique. Due to the presence of transient link failures associated with mobility, faut-tolerance becomes an essential part of a distributed mutual exclusion problem. We addressed the problem of fault-tolerance with the introduction of a new message type and adaptive timeout handling mechanisms. We addressed the problem by dividing the network into regions and using the notion of an arbitrator node to solve the distributed mutual exclusion problem. We thus proposed two permission-based distributed mutual exclusion algorithms in this thesis.

As part of this work, our main contributions towards the routing problem are as follows: We introduced a new cost metric for energy-aware routing in MANETs. The Energy-Aware Routing algorithm proposed using this metric was able to improve network lifetime by 10%-65% when compared to other techniques. We developed an efficient on-demand routing protocol for

MANETs using the Bayesian approach to address the route maintanence and route establishment problem. This algorithm improved key parameters such as delivery ratio and the number of control packets broadcasted under varying mobility conditions. We then developed a new model, termed as Generalized Energy Consumption Model, to support energy harvesting devices in wireless sensor networks.

Our major contributions towards the multicast routing problem are as follows: We analyzed the use of virtual force technique for multicast routing problem and adapted the technique for solving the multicast routing problem in our work. We introduced the notion of dampening forces in multicast routing algorithms for MANETs. We then introduced the notion of virtual force based on regions in multicast routing algorithms for MANETs. We also developed the Multicast Routing Algorithm using Virtual-force, the sector-based Virtual Force Multicast routing algorithm and the Virtual Multicast Tree routing algorithm for for QoS-aware multicast routing in MANETs. These algorithms generated relatively minimal Steiner trees with better length of the multicast tree than those reported in recent works. This part of our research has opened the doors for the use of the virtual force technique to solve multicast routing problem and other allied problems for ad-hoc networks.

The other major contribution of this thesis is towards solving distributed mutual exclusion problem in MANETs. We developed a novel permission-based distributed mutual exclusion algorithm for MANETs that used adaptive time-out mechanisms for handling fault-tolerance. We introduced the notion of an arbitrator node in the "look ahead" technique to handle distributed mutual exclusion among nodes in multiple regions. Based on this concept, we further developed a reliable arbitrator-based distributed mutual exclusion algorithm for MANETs. The notion of using arbitrators among regions can be applied to solve other problems similar to the distributed mutual exclusion problem in MANETs.

Table of Contents

List of Figures **xv**

List of Tables **xviii**

List of Abbreviations/Symbols **xix**

1 Introduction **1**

1.1 Wireless Ad hoc Networks: An Introduction 1

1.2 Issues in Wireless Ad Hoc Networks 5

1.3 Characteristics of Mobile Ad hoc Networks 6

1.4 Applications of Mobile Ad Hoc Networks 8

1.5 Routing in MANETs 10

1.5.1 Introduction 10

1.5.2 Issues related to design of routing protocols for MANETs 11

1.5.3 Route maintenance 14

1.6 Classification of Routing Algorithms for MANETs 14

1.6.1 Classification based on Mechanism for Updating Routing Information 15

1.6.2 Classification based on Temporal Information 16

1.6.3 Classification based on Utilization of Specific Resources 16

1.6.3.1 Power-Aware Routing Protocols 16

1.6.3.2 Geographical Routing Protocols 17

1.7 Graph Models used in MANETs 17

1.7.1 Unit Disk Graph 18

1.7.2 Unit Ball Graph 18

1.7.3 Relative Neighbourhood Graphs 20
1.8 QoS Aware routing in MANETs 20
1.8.1 Hard QoS and Soft QoS 21
1.8.2 Design Considerations 22
1.9 Multicast Routing in MANETs 24
1.10 Resource Allocation in MANETs 25
1.10.1 Mutual Exclusion 25
1.10.2 Distributed mutual Exclusion 27
1.10.3 Classification of Distributed Mutual Exclusion Algorithms in MANETs 28
1.10.3.1 Token-based Approach 28
1.10.3.2 Permission-based Approach 28
1.10.3.3 Consensus-based approach 29
1.11 Summary of Contributions 29
1.12 Thesis Organization 30

2 Literature Survey **31**
2.1 Routing techniques in MANETs 32
2.1.1 Destination Sequenced Distance-Vector Protocol(DSDV) 32
2.1.2 Dynamic Source Routing (DSR) 33
2.1.3 Ad Hoc On-Demand Distance-Vector (AODV) Routing Protocol . . 35
2.1.4 Location Aided Routing (LAR) 35
2.1.5 Other traditional algorithms 37
2.2 Geographic or Geometric routing algorithms 38
2.2.1 Strategies and approaches in position aware routing algorithms . . 40
2.2.2 Simple position based schemes 42
2.2.3 Greedy Perimeter Stateless Routing (GPSR) 44
2.2.4 Spine Routing 45
2.2.5 Virtual force technique 46
2.2.6 Minimum Connected Dominating Set(MCDS)/Maximal Independent Set(MIS) based algorithms 47

2.2.6.1 A simple Maximal Independent Set(MIS) construction algorithm . . . 48
2.2.6.2 A Maximal Independent Set(MIS) algorithm over two-hop neighbourhood . . . 51
2.2.7 Well-Separated Pair Decomposition . . . 51
2.3 Quality-of-Service(QoS) aware routing protocols in MANETs . . . 53
2.3.1 Reservation-less approach for QoS aware routing in MANETs . . . 54
2.3.2 Reservation-oriented QoS aware routing algorithms for MANETs . . . 55
2.3.3 Energy aware routing protocols in MANETs . . . 55
2.4 Multicast routing protocols . . . 57
2.4.1 Some of the earlier Multicast routing algorithms . . . 58
2.4.2 Graph based approach in Multicast routing algorithms . . . 59
2.4.3 Contemporary approaches used in Multicast routing algorithms . . . 60
2.5 Distributed Mutual Exclusion . . . 61
2.5.1 Classical token based algorithms . . . 62
2.5.2 Classical Permission based Algorithms . . . 65
2.5.3 Recent algorithms for DME . . . 66
2.6 Conclusion . . . 67

3 Energy Efficiency in Mobile Ad-hoc Networks 69

3.1 Energy-Aware Routing in Mobile Ad-Hoc Networks . . . 70
3.1.1 Background and Motivation . . . 70
3.1.2 Assumptions . . . 71
3.1.3 Basic Mechanism . . . 71
3.1.4 Algorithm . . . 74
3.1.5 Simulation Results . . . 76
3.1.6 Conclusion . . . 83
3.2 An Efficient On-Demand Routing Protocol(EORP) for MANETs based on Bayesian Approach . . . 83
3.2.1 Introduction and Motivation . . . 84
3.2.1.1 Basic Working of EORP . . . 84

3.2.1.2 Calculating Affinity 85
3.2.1.3 An Example 86
3.2.2 Algorithm for Efficient On-demand Routing Protocol 89
3.2.3 Performance Results 90
3.2.4 Conclusion 92
3.3 A Generalized Energy Consumption Model for Wireless Sensor Networks . 95
3.3.1 Background 95
3.3.2 Network Topology 96
3.3.3 System Model and Assumptions 97
3.3.4 Energy Budget 97
3.3.5 Simulation Results 99
3.3.6 Conclusion 99
3.4 Conclusion 101

4 Quality of Service (QoS) Aware Multicast Routing in Mobile Ad-hoc Networks 103
4.1 Introduction and Motivation 104
4.2 Background 106
4.3 Multicast Routing Using the Virtual Force Technique 107
4.3.1 Assumptions 108
4.3.2 Basic Idea 109
4.3.3 System Model 109
4.4 Multicast Routing Algorithm using Virtual-force(MRAV) 111
4.4.1 An illustration of MRAV 111
4.4.2 The MRAV Algorithm 114
4.4.2.1 Data structures 114
4.4.2.2 The Algorithm 114
4.4.3 Analysis 114
4.5 Sector-based Virtual-force-based Multicast Routing Algorithm 117
4.5.1 Motivation 117
4.5.2 Basic Working 117
4.5.3 Algorithm 120

4.5.4 Analysis 120
4.6 Virtual Multicast Tree 122
4.6.1 Basic concepts 123
4.6.1.1 Well Separated Pair Decomposition 123
4.6.1.2 Basic idea 124
4.6.2 2-MIS Region creation 124
4.6.3 Basic Working 125
4.6.4 Virtual Multicast Tree Algorithm 126
4.6.5 Analysis 126
4.6.6 Optimizations of the algorithm 128
4.7 Simulation Results 128
4.8 Conclusion 132

5 Distributed Mutual Exclusion in Mobile Ad-hoc Networks 134
5.1 Introduction 135
5.1.1 The Mutual Exclusion Problem 135
5.1.2 Background 137
5.2 The Dynamic Look-Ahead Technique 139
5.2.1 Notation 139
5.2.2 Rules 139
5.2.3 Wu, *et al.*'s improvements to Dynamic Information Set 141
5.3 Initialization of nodes for the DME algorithm 141
5.3.1 Awasti's Initialization Scheme 142
5.4 A Novel Permission-based Reliable Distributed Mutual Exclusion Algorithm for MANETs 143
5.4.1 Introduction and Motivation 143
5.4.2 System Model and Assumptions 144
5.4.3 Algorithm Overview 145
5.4.4 Adaptive Timeout Mechanism 147
5.4.5 Data Structures Used 148
5.4.6 The Algorithm 149

5.4.6.1 Requesting for CS 149
5.4.6.2 Receiving *REQUEST* message 150
5.4.6.3 Receiving *HOLD* message 151
5.4.6.4 Handling of Timeouts 151
5.4.6.5 Entering the Critical Section 152
5.4.6.6 Exiting the Critical Section 152
5.4.7 Correctness of the algorithm 152
5.4.7.1 Mutual Exclusion 152
5.4.7.2 Freedom from Deadlocks 153
5.4.7.3 Safety 153
5.4.7.4 Fault Tolerance 154
5.4.8 Performance 155
5.5 Reliable Arbitration-based DME Algorithm 155
5.5.1 Assumptions 157
5.5.2 Working of Arbitrator 157
5.5.3 Data Structures Used 162
5.5.4 The Proposed Algorithm 163
5.5.4.1 Requesting for CS 163
5.5.4.2 Receiving *REQUEST* message from site S_j 163
5.5.4.3 Receiving *HOLD* message 165
5.5.4.4 Receiving *REPLY* message 165
5.5.5 Correctness 166
5.5.5.1 Mutual Exclusion 166
5.5.5.2 Freedom from Deadlocks 167
5.5.5.3 Freedom from Starvation 167
5.5.5.4 Fault tolerance 167
5.5.6 Performance 170
5.6 Conclusion 177

6 Conclusions & Future Work **178**

6.1 Summary and Conclusions 179

6.1.1 QoS-aware Routing Algorithms 179
6.1.2 Distributed Mutual Exclusion Algorithms for MANETs 181
6.2 Scope for Future Research 183

References **185**

Publications **201**

Biographies **203**

List of Figures

1.1 Infrastructure Based Wireless Network 3
1.2 Ad hoc Wireless Network 3
1.3 A simple ad hoc network 7
1.4 A sampe MANET containing ten nodes 10
1.5 Hidden Terminal Problem and Exposed Terminal Problem 13
1.6 A Unit Disk Graph 19
1.7 A Unit Ball Graph 19

2.1 Dynamic Source Routing 34
2.2 Ad Hoc On-Demand Distance-Vector Routing Protocol 36
2.3 Greedy Perimeter Stateless Routing(GPSR)(1) 44
2.4 Greedy Perimeter Stateless Routing(GPSR)(2) 45
2.5 Spine Routing 46
2.6 Virtual force technique 48
2.7 Maximal Independent Set construction(1) 50
2.8 Maximal Independent Set construction(2) 51
2.9 2-hop Maximal Independent Set construction 52
2.10 Well-seperated pair decomposition 53
2.11 A sample token-based DME algortihm 62
2.12 A sample permission-based DME algortihm 63

3.1 A sample MANET 72
3.2 Procedure for Collecting Power 75
3.3 Procedure for generating route 76

3.4 Procedure for computing cost for node i 76
3.5 A sample Network 78
3.6 A sample run of EAR 78
3.7 Another sample run of EAR 79
3.8 Normalized Residual Energy for each node 80
3.9 Average Residual Energy over time 80
3.10 Variance of energy over time 81
3.11 Average Normalized Cost per packet per hop 82
3.12 Example for showing forwarding of $RREQ$ 87
3.13 Example showing forwarding of $RREP$ and path for transferring data . . . 87
3.14 Procedure for sending route request (RREQ) packet at node u 89
3.15 Procedure for forwarding route request (RREQ) packet at node u 89
3.16 Procedure for handling Route Reply (RREP) packet at node u 90
3.17 Procedure for Route Repair at node u 91
3.18 A sample simulation run 92
3.19 Delivery ratio over mobility 93
3.20 Number of control packets broadcast over time(no mobility) 93
3.21 Number of control packets broadcast over time(low mobility) 94
3.22 Number of control packets broadcast over time(high mobility) 94
3.23 Structure of a sensor node 95
3.24 A sample sensor network 96
3.25 Power of a node in the i^{th} cluster v/s A_i 100
3.26 Power of a node in the i^{th} cluster v/s X_i 100

4.1 Example for failure of classic virtual force technique 108
4.2 Working of MRAV(a simple example) 111
4.3 Working of MRAV(impact of forces) 112
4.4 MRAV routing algorithm: sending the Query message 115
4.5 Virtual Force-based Multicast(VFM) routing algorithm 118
4.6 Virtual Force acting on node u. 119
4.7 A sample scenario for $\alpha = 4$ 121

4.8 Virtual Multicast Tree(VMT) algorithm 127
4.9 Average normalized residual energy over time 129
4.10 Normalized cost per hop per packet over time 130
4.11 Average length of multicast tree with respect to number of nodes 131

5.1 Novel DME Algorithm 146
5.2 Working of Novel DME Algorithm 146
5.3 Growth and decay phases for the adaptive timeout mechanism 148
5.4 Receiving *REQUEST* Message from site S_j 150
5.5 Receiving *HOLD* Message from site S_j with timeout value τ 151
5.6 Basic operation with an arbitrator in Holding. 158
5.7 A sample network with multiple regions. 159
5.8 Working of an Arbitrator. 160
5.9 Reliable Arbitrator-based DME Algorithm 161
5.10 Receiving *REQUEST* Message from site S_j 164
5.11 Receiving *HOLD* Message from site S_j with timeout value τ 166
5.12 Normalized Average Synchronization Delay vs Number of nodes. 174
5.13 Normalized Average Response Time vs Number of nodes. 174
5.14 Average Hops / Critical Section vs Number of nodes. 175
5.15 Normalized Average Time per Critical Section vs Number of nodes. 175
5.16 Effect of number of arbitrator nodes 176

List of Tables

1.1 Comparison of Infrastructure-based networks and Ad hoc networks 4

3.1 History information at different nodes for network in Figure 3.12 88

5.1 Performance comparison 172

List of Abbreviations/Symbols

Term	**Definition**
UDG	Unit Disk Graph
MANET	Mobile Ad-Hoc Networks
DME	Distributed Mutual Exclusion
RTT	Round-trip Time
IETF	Internet Engineering Task Force
QoS	Quality of Service

Chapter 1

Introduction

1.1 Wireless Ad hoc Networks: An Introduction

The advent of laptops, tablets and mobile phones in the past couple of decades has changed the face of information sharing by leaps and bounds. One of the crucial factors for this drastic increase in Internet usage is due to the mobile and ubiquitous means of accessing network resources by the end users. It has become common place for the end users to rely on wireless means of communication for last mile connectivity. Users now-a-days use their mobile devices to do all sorts of activities, from booking tickets for flights and/or trains to coordinate assaults in rain forests. Also, users don't want to be constrained to sit in one particular seat for accessing the network. They now expect to be reachable in all sorts of places including coffee shops, inside aircrafts and in cruise ships.

Wireless networking had mostly used physical infrastructure like towers for establishing network connectivity. However, the ability to establish a quick network on-demand, without waiting for installation of towers and other infrastructure, looks very appealing, especially for rescue workers and military tacticians. In the current usage vectors, it is not uncommon to find such an ad hoc network to be established by common users. For example, students sometimes establish such a simple ad hoc network of laptops in parks or hostels for sharing files or playing multi-player games. In fact, wireless ad hoc networking enables the establishment of a wireless network in the environments where there is lack of infrastructure for wired or wireless communications; or where it is cheaper or

more effective to establish such an ad hoc network instead of using existing links.

A typical wireless ad hoc network is a network of nodes established for special applications offering customised services. Such a network is set up for a limited duration. For example, in a search-and-rescue environment, the ad hoc network needs to be established only for the duration of the operation. Once the missing person has been found, the network shall be "dismantled". Even the protocols used in this network have to be customised as per the requirements (e.g., provide location and other intelligence from soldiers deployed in the battlefield, share video of a disaster zone). These protocols must be capable of handling issues arising due to changing environment or application characteristics. They must be extremely flexible, so that they can provide services under all circumstances. This demands such a protocol to be capable of self-organization and self-configuration, while still maintaining robustness.

Thus, wireless ad hoc network may be defined as the category of wireless networks consisting of autonomous nodes capable of operating without the support of any fixed infrastructure, utilizing multi-hop radio relaying for communication among such nodes. When the nodes forming the network are mobile users with tight restrictions on the capacity of the wireless links, such a network is termed as a Mobile Ad hoc Network (MANET). Figure 1.1-1.2 show examples of infrastructure based and Ad hoc wireless networks respectively. It can be seen in Figure 1.1 that the access points(cellular towers) need to be connected separately through a wired network. The difference between an infrastructure-based network and a wireless ad hoc network is listed in Table 1.1.

For appreciating a MANET architecture, it is essential to understand the issues pertaining to such networks which we describe in the following section.

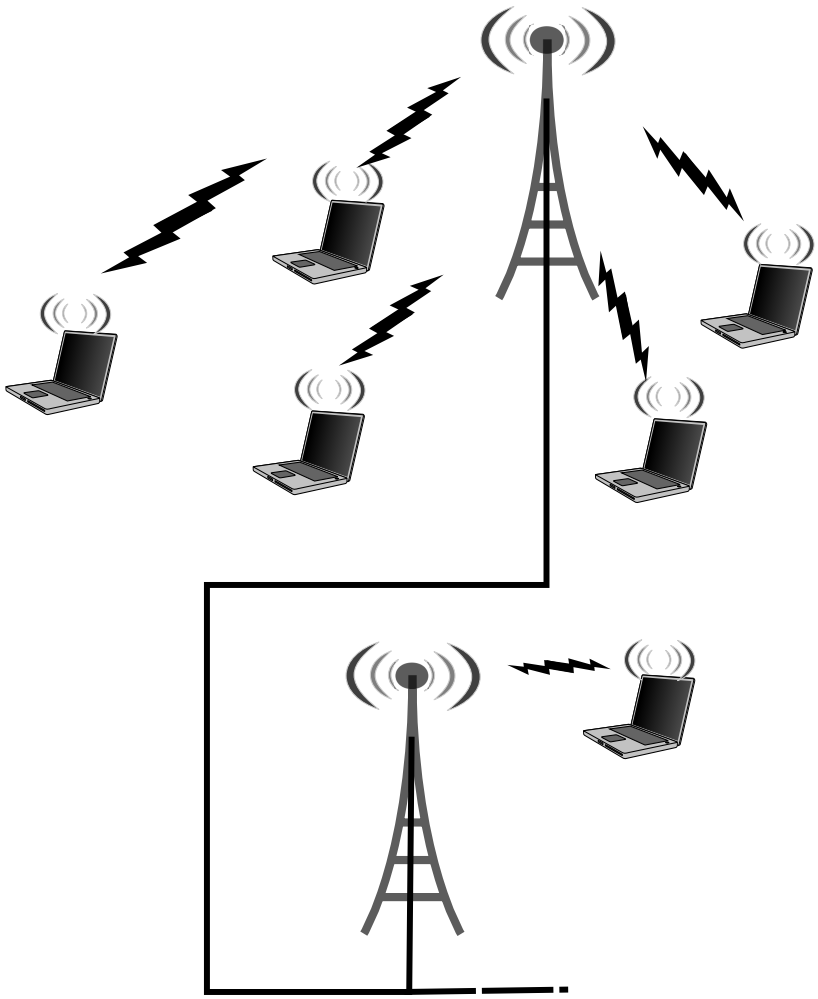

Figure 1.1: Infrastructure Based Wireless Network - a sample network with six nodes

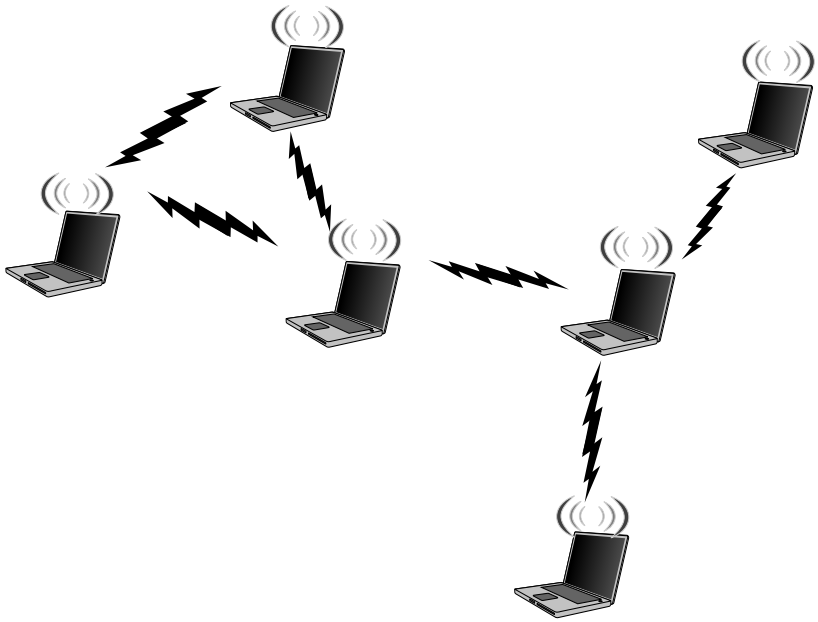

Figure 1.2: Ad hoc Wireless Network - a sample network with six nodes

Table 1.1: Difference between the two categories of wireless networks

Sl No	Infrastructure-based Networks	Ad Hoc Wireless Networks
1.	Fixed, Infrastructure-based	Infrastructure-less
2.	Single-hop wireless links	Multi-hop wireless links
3.	Centralized routing	Distributed routing
4.	Circuit-switched (evolving towards packet-switched)	Packet-switched (evolving towards emulation of circuit-switched)
5.	Seamless connectivity with low call drop rate	Frequent path breaks due to mobility
6.	Designed for voice traffic	Designed for best-effort traffic
7.	Deployment is costly	Quick and easy deployment
8.	Costly to maintain (power source, extra staff, etc.)	Self-organization and maintenance are built into the network
9.	Easy bandwidth reservation	Bandwidth reservation involves complex medium access control protocols
10.	Mobile hosts of relatively low complexity (have transceiver)	Intelligent mobile hosts (have transceiver and routing capability)
11.	Easier to synchronize time	Time synchronization is difficult, with heavy bandwidth consumption
12.	Major goals of routing and call admission are to reduce call drop rate and maximize call acceptance rate	Major goal of routing is to find paths and to recover from any broken links/paths
13.	Application domain is typically civilian and commercial	Application domain includes battlefields, search-and-rescue operations, and collaborative computing
14.	Widely deployed and currently moving towards fourth generation	Several issues are yet to be addressed for successful commercial deployment, even though widespread use exists in defence

1.2 Issues in Wireless Ad Hoc Networks

Following are the commonly faced issues in wireless ad hoc networks.

Mobility Model

The most interesting characteristic of a wireless ad hoc network is the nature of mobility of the nodes forming the network. There can be networks in which most of the nodes are static, after the initial establishment phase, like the networks forming a wireless mesh network. There are wireless sensor networks that are deployed in floating buoys to map sea-levels or under-water seismic vibrations, in which the mobility of nodes is limited to a particular region or is predefined. Even the nature of mobility varies from one application domain to another. For example, a search-and-rescue operation in an earth quake affected region will typically have teams searching in groups. This leads to a mobility model in which groups of nodes move independent of one another. In a rescue operation in a forest, the nodes may move in a fixed pattern. If we look at a vehicular ad hoc network, which is a wireless ad hoc network made up of vehicles, the mobility pattern closely follows the vehicular traffic flow in which the speed and the direction of the nodes can be predicted to a fair degree. Guaranteeing packet delivery and optimizing various network parameters depend heavily on the nature of the application and the mobility model.

Self-organization

The nodes forming an ad hoc network are going to be deployed almost randomly in a non-cooperating environment. Even at the time of deployment, mandating the nature of topology may not be possible for the entire network. It is crucial for any node interested in being part of the network to autonomously be capable of handling addressing, routing, clustering, location services, power control, and other necessary parameters.

Multihopping

The nodes forming the network need to be able to communicate to one another. However, it is not physically possible to ensure that all the nodes can have a physical channel with one another. There will be nodes in the network that are not within the communication range of a host node. To ensure that packets can be transferred to the relevant destination node, it will be necessary to forward the packets through one or more inter-

mediate nodes that are part of the network. Also, choosing shorter hops improve power control of nodes.

Energy Efficiency

A wireless ad hoc network typically consists of nodes that work using battery source. They typically have finitely limited power supply and do not usually have the capability to generate own power. (Exceptions are special wireless sensor nodes deployed in remote areas, in which a solar cell or other power source is specially added.) These nodes typically consume power for performing their computational operations and for wireless communication. The nodes that are currently deployed come with a facility to put the node to sleep, if the node's power drops below a particular threshold. A node that was previously part of the network may disassociate itself from the network, if its power level goes down. This may even result in network partitioning, if a crucial node that links two halves of the network fails. Thus energy efficiency is crucial for both the longevity of the ad hoc network as well as its robustness.

Scalability

In an infrastructure-based wireless network, adding extra nodes simply involves adding extra towers and related infrastructure. The hierarchical nature of such a network makes dealing with scalability easy, by using a combination of Mobile IP and local hand-off techniques. Such techniques common in infrastructure-based wireless network cannot be used in ad hoc networks because of the extensive mobility of participating nodes. The large number of intermediate hops necessary for transmitting packets makes it difficult to ensure robust delivery of packets across the network.

1.3 Characteristics of Mobile Ad hoc Networks

The following characteristics become relevant while designing any protocol/algorithm for mobile ad hoc networks (MANETs).

Node mobility:

MANETs get their name because of the variable mobility of the participating nodes. Unlike other types of ad hoc networks, a MANET does not have a typical mobility pattern.

Due to the random mobility pattern (both in terms of speed and direction), any algorithm or protocol intended for MANETs needs to be highly flexible.

Dynamic network:

A node may enter or leave the network at any time. MANETs are prone to higher churn rate than their static equivalents. The main reason for this dynamic behaviour is mobility. Nodes forming the network may drift away and return back at any point in the future.

Link failure:

A communication link can exist in a MANET if and only if the receiver node is within transmission range of the sender node. In the sample mobile ad hoc network shown in Figure 1.3, it can be noted that only neighbours of a node within its transmission range (denoted with dashed lines) have links to the node. If a node drifts out of this transmission range, then the link will get broken. In a MANET, it is common for some of the links to get broken like this due to the frequent mobility of the nodes involved in the network. Also, it is possible for a previously broken link to be re-established if the drifting node returns to the transmission range of the node.

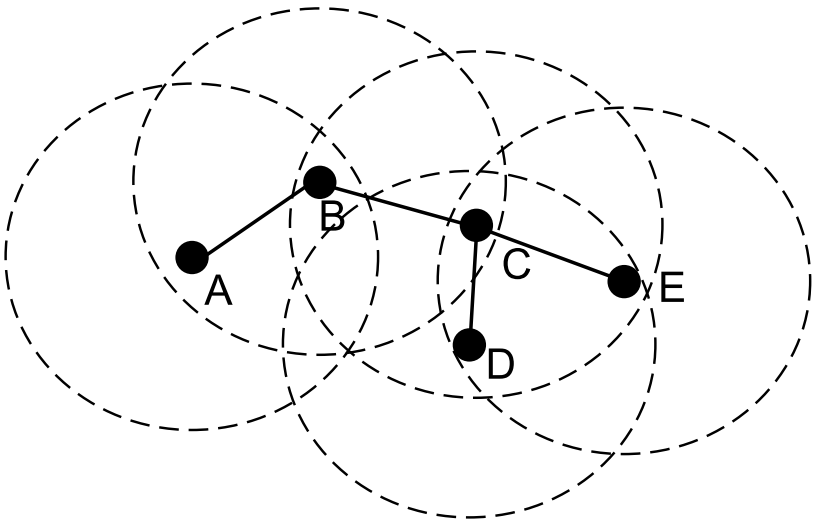

Figure 1.3: A simple ad hoc network - a six node network showing transmission ranges and links interconnecting nodes

Combinatorial Stability:

Due to the mobility of nodes, the global topology of the ad hoc network gets altered quite frequently. Let τ be the time interval between two consecutive topology change

events. Let T_c be the time required for an algorithm to detect the change, perform relevant processing to incorporate the changes, and propagate this topology change information to other relevant nodes. A network is considered to be combinatorially stable if $\tau > T_c$. Thus, combinatorial stability indicates that the algorithms/protocols running in the system can detect any change, and take necessary actions fast enough so that the correctness of the algorithm is not affected by such a change in topology. As the difference between τ and T_c becomes smaller, the probability for the algorithm to fail to meet its designated goal increases. This characteristic is dictated by the random changes in topology and link traffic intensity, making it difficult to predict the probability for a fault to occur. Hence, the robustness of the protocol/algorithm cannot be guaranteed if a network lacks combinatorial stability.

Energy Restrictions:

Mobile nodes do not have a power source attached to them. A node forming the network spends energy for its communication needs as well as its processing requirements. The energy spent on communication depends on the transmission range of a node. Also, a certain amount of energy is expended while listening to packets transmitted within its transmission range. Any algorithm or protocol intended for MANET must ensure that they are energy-efficient.

1.4 Applications of Mobile Ad Hoc Networks

MANETs are often used in application domains which require a fast and efficient deployment, especially in places where it is either practically infeasible to deploy an infrastructure-based network, or it is economically not viable to do so. A few of the application domains that use MANETs are described in this section.

Battlefields

One of the most important application domains for MANETs is in military battlefields. Even during the nascent stages of the networking domain, during the time that ALOHAnet [Mohapatra & Krishnamurthy 2010] was used to try to connect various uni-

versities in Hawaii, the utility of a quickly deployable network had captured the attention of military tacticians. Recently, with the advent of smart soldiers carrying wearable devices, having effective communication between the soldiers in battlefields and the military/intelligence command has become more valuable. With the focus shifting towards urban warfare, as has been observed in Iraq and Syria, military personnel demand much greater flow of information and greater network connectivity in hostile infrastructure-less environment. Ad hoc networks find great acceptance in such hostile environments.[Trellisware 2013] Some nodes forming the part of such a network may not have energy constraints, such as battle tanks or submarines. Some nodes, such as foot soldiers with wearable computers, need to be highly energy-efficient. Tactical Scalable MANETs have found larger acceptance in the armed forces, especially for deployment in large cargo ships, rugged terrains, across urban downtown areas and large coastal areas[TSM 2013].

Emergency Operations

Search-and-rescue, crowd control and commando operations occur in unexpected areas affecting large number of people and in most cases are unavoidable. MANETs are capable of self-configuration with minimal overhead, infrastructure-less and flexible of mobility. The unavailability of conventional infrastructure due to the nature of terrain is not an impediment for deploying a MANET. These features make the use of MANETs an ideal solution for such scenarios. For search-and-rescue operation in disaster environments such as earth-quakes or tsunami, the existing infrastructure might have got destroyed or may not be reliable. The ability to deploy MANETs immediately makes their use highly desirable in such environments. The main design challenges in these environments include fault-tolerance, real-time communication capability for both data and voice, and scalability.

Collaborative computing

MANETs can be deployed in classrooms or in conference rooms for sharing files or video. In these environments, there is an immediate requirement to create an instantaneous network for a short duration. Setting up infrastructure for such short duration is not economically viable, especially since the duration for which such a network is going to be active is very small.

Civilian Applications

MANETs have also found interest in certain civilian applications as well. For communication among people inside a ship, the use of MANET makes more sense. Another area where this class of networks can be used is for connecting cab drivers in a city environment[Trellisware 2013]. For such an application, full coverage can be ensured by minimally placing infrastructure points at strategic locations for traffic distribution, while banking predominantly on mobile nodes for relaying traffic.

1.5 Routing in MANETs

1.5.1 Introduction

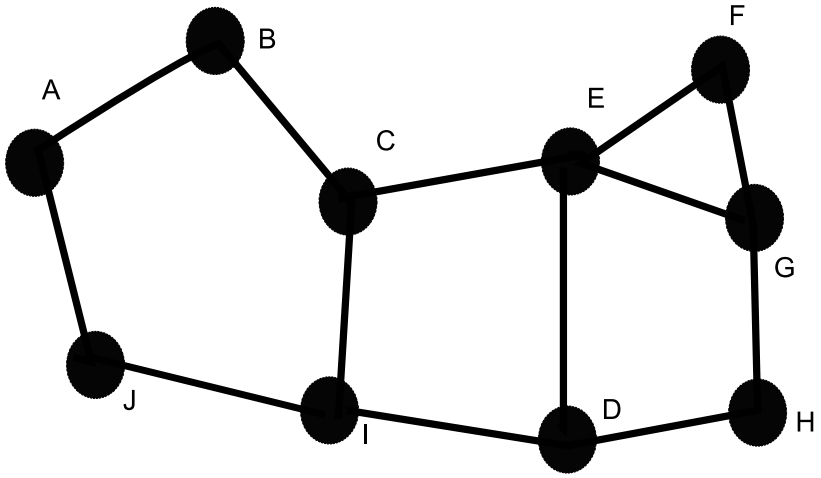

Figure 1.4: A sampe MANET containing ten nodes -

If the source and destination are not directly reachable in a MANET, a packet needs to be routed by intermediate nodes between the source node and the destination node. Each intermediate node acts as a router for this particular traffic flow. Along with issues faced by mobile nodes in an ad hoc network, the lack of global knowledge of the network topology makes it difficult to address the problem of identifying the best next node to forward a packet. In Figure 1.4, say the node A wants to communicate to the node D. The source node, A, may forward the packet to an intermediate node B, which in turn can forward to node C. From C, the packet may be routed via E to reach the final destination D. Apart from the node B (whose forwarding choice is trivial), every node in the routing path needs to decide whether the packet is going to get forwarded to the next best route.

A simple tactic used for routing involves asking all the neighbours of the source node regarding whereabouts of the destination node by means of flooding. The information passed by neighbours, and their neighbours' neighbours can be collated to determine the best path to reach the destination. Though flooding looks like an inefficient approach and hence unappealing, there are scenarios where flooding may be the only way to forward a packet. Another technique that may be explored involves sending packets to the destination using the geometric information of all the nodes in the network. By using this information, the route $A \rightarrow J \rightarrow I \rightarrow D$ might appear to be the best path. Getting accurate geometric information of all nodes that belong to a large dynamically changing topology is still not easy.

While deciding the best routing path, hop count is typically used in traditional routing approaches in many computer networks. However, in an ad hoc environment, other factors such as dynamic topology and energy-efficiency are equally important. While designing routing algorithms/protocols for MANETs, other routing metrics like average residual energy of nodes, network lifetime, packet delivery rate, etc. need to be considered as well. The next two sections describe issues and various types of routing algorithms used for MANETs.

1.5.2 Issues related to design of routing protocols for MANETs

Due to the constraints imposed by the underlying network, the following issues are typically faced while designing a routing protocol for MANETs.

Mobility

The network topology is highly dynamic in a wireless ad hoc network due to the movement of nodes and the resulting disruption to routing paths. Such a disruption can occur either due to the mobility of the source and destination nodes themselves, or due to the wandering of intermediate nodes that form the path from source to destination. In such dynamic environments, the processing time required for establishing and main-

taining routing paths needs to be small. To ensure that the consistency of the global state of the network, the protocol will eventually end up sending a lot of control packets. If appropriate measures are not taken, the global state may not converge in time for making the relevant routing decision. An efficient and effective mobility management mechanism must be incorporated in any protocol intended for use in this type of ad hoc networks.

Bandwidth constraint

High bandwidth links can be deployed in a wired network with the help of fibre optics and other related technologies. However, in a wireless ad hoc environment, the data rates are much lower due to the limited radio band. On top of that, frequent topology changes will require transmission of control messages to communicate the changes in the location of intermediate and end nodes. A routing protocol written for MANET must be efficient, so that the wastage of bandwidth in the form of control overhead in limited.

Error-prone broadcast medium

In a wireless environment, typical signals are broadcast over available radio channel. The link capacity and probability for error in this wireless physical link varies widely over time due to radio interference and other similar factors. While routing packets, the mobile node must interact with the Medium Access Control (MAC) layer and determine the best-quality link to transmit the packet. Apart from the nature of the link, there is also the possibility of collisions of packets transmitted over the broadcast medium. For example, say node E in the Figure 1.4 initiates a communication to F, while G transmits data to H. Since both E and G are within transmission range of each other, this will result in collision, and the packets have to be retransmitted. While routing packets, it is desirable to choose a path that is least congested.

Hidden and Exposed terminal problems

Hidden terminal problem occurs when a node is unaware of the existence of other interfering transmissions to its intended recipient(s). Figure 1.5 can be used to illustrate the problem. When node A transmits a signal, the nodes in its transmission range, viz. B and C, will receive the signal from A. Node A is unaware of the existence of other nodes (D and E), as they are outside its transmission range. Suppose A listens to the medium and observes that no one is using currently the medium. It transmits a signal to B. At the same time, suppose D is also transmitting a packet. Note that both A and D are unaware

of the other's transmission as they are not within transmission range. C can receive the transmission from A and E can receive the transmission from D without any problems. However, the node B will get the transmission from both A and D, resulting in collision at B. The nodes outside the range of the transmitting node (A) are hidden from sender, and hence the problem is called as hidden terminal problem. One simple mechanism involves the use of RTS(Request to send)-CTS(clear to send) handshaking. The initiating node, A, can transmit an RTS signal to its neighbours. B, after receiving the RTS signal, can respond with a CTS signal. This can ensure that no hidden nodes in the vicinity of B attempts to initiate communication. However, consider the case where D failed to listen to the CTS being transmitted by B due to an incoming transmission from E. Since E is hidden from both A and B, this will still result in collisions at B, when D responds to E.

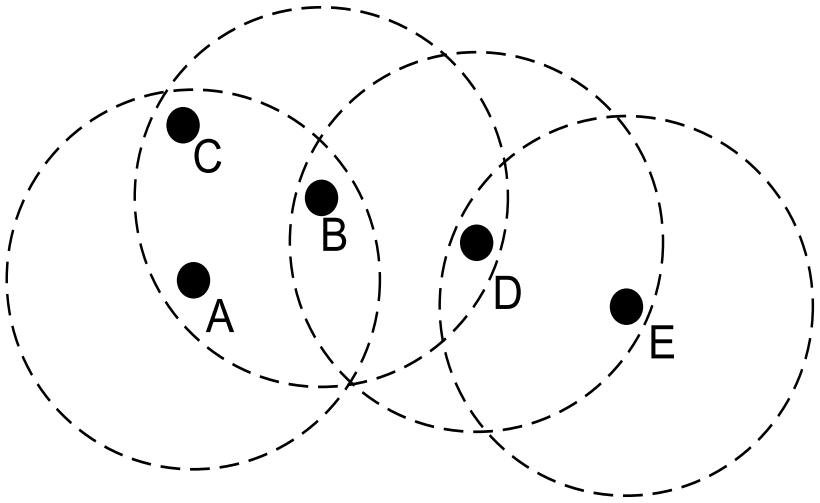

Figure 1.5: Hidden Terminal Problem and Exposed Terminal Problem - A sampe MANET containing five nodes.

While hidden terminal problem refers to the problem of unsuccessful transmission being initiated because of the collisions resulting from nodes that are hidden to the sender, the exposed terminal problem refers to the problem of inability of a node to transmit data as it believes that someone else is currently transmitting. The exposed terminal problem is a consequence of the RTS-CTS handshaking protocol that was intended to remove the hidden terminal problem. Consider the network in the Figure 1.5. Let A transmit the RTS signal to B. At the same time, E is interested in initiating communication with D, and transmits an RTS signal to D. After receiving RTS from E, D will respond with a CTS signal, which is heard by both B and E. B will assume that there is an ongoing com-

munication, and will backoff from sending the CTS to A. A thus becomes incapable of transmitting data. Here, the node E becomes the exposed terminal preventing the communication between A and B. The exposed terminal problem thus affects the reusability of radio spectrum.

Resource constraints

As the mobile nodes forming part of a network require portability in most cases, there are serious limitations on the size and weight of such devices. This translates to restrictions on the power source used as well as the processing power of such devices. Efficient use of battery life and processing power needs to be considered while making routing decisions.

1.5.3 Route maintenance

Route maintenance is part of any good routing protocol in MANETs[Broch *et al.* 1998]. The nature of MANETs is such that there are going to be frequent failures of links. Mobility of the nodes may also result in drifting of the nodes away from the network, resulting in network partitioning or transient node failures. While computing efficient nodes to route the packets, the algorithms need to be aware of links that get established at a future time as such links may result in a more efficient path from source to destination.

The task of route maintenance may be performed as part of the normal protocol operations or as a separate part of the routing infrastructure. For example, in algorithms such as AODV(Ad Hoc On-Demand Distance-Vector) [Perkins & Royer 1999], the route maintenance is achieved with the help of $RERR$ packets as part of the normal protocol operations. DSR(Dynamic Source Routing) [Johnson & Maltz 1996], on the other hand, uses a separate route maintenance phase for identifying changes in topology and incorporating the changes into the network.

1.6 Classification of Routing Algorithms for MANETs

Various routing techniques have been proposed to take into consideration the set of constraints imposed on a MANET. Routing protocols can be classified on the basis of choices

made for routing information update mechanism, use of temporal information, and utilization of specific resources. In this section, the various approaches used for routing protocols in MANETS are discussed.

1.6.1 Classification based on Mechanism for Updating Routing Information

Based on the mechanism used to update the routing information, routing protocols can be broadly classified as proactive, reactive and hybrid protocols.

Proactive Protocols

Proactive protocols are table-driven protocols.[Murthy & Manoj 2004] In such protocols, every node typically maintains a routing table consisting of topology information. This topology information is updated periodically using flooding. Path selection is based on the routing information stored in an individual node. Generally, proactive protocols do not tend to be combinatorially stable in the event of frequent changes in topology.

Reactive Protocols

Reactive protocols are also called as on-demand protocols.[Murthy & Manoj 2004] Unlike table-driven routing protocols, on-demand routing protocols determine the path to a node during the connection establishment process, alleviating the need to store the entire topology in each node. These protocols don't need to send periodic beacon messages to exchange route information. A typical reactive protocol will transmit a RouteRequest packet to its neighbouring nodes in the network. The intermediate nodes will forward these request packets till the destination is reached. The destination node sends a RouteReply message through the path traversed by the RouteRequest message back to the source node. Reactive protocols are considered more scalable, as these protocols do not need to wait for the global state to converge. For smaller networks though, proactive protocols perform much better than reactive protocols.

Hybrid Protocols

Hybrid protocols are a combination of proactive and reactive methods. These protocols rely on a table-driven approach for nodes within a short range or within a particular geographic region. They use a reactive approach to deal with the source-destination pairs

that are farther apart.[Murthy & Manoj 2004]

1.6.2 Classification based on Temporal Information

Using past state information

This type of routing protocols store the previous state information pertaining to links for computation of the routing path. The current recorded state information regarding the availability of wireless links along with a simple shortest path algorithm may be used for determining the path towards the destination. In this method, it is believed that the stored state information corresponds to the actual state of the network. So, in the event of topology changes, the paths may become invalid and a fresh path may need to be recomputed.

Using future state information

In this type of protocol, the expected state of the wireless links in the immediate future is computed and an approximate solution is provided for determining the best path from source to destination. Apart from the state of the link, information regarding lifetime of the node, location and link availability in the near future may also be predicted.

1.6.3 Classification based on Utilization of Specific Resources

Apart from the previous modes of characterization, there are certain classes of protocols that focus on one or more specific resources like power, location information, etc. These other classes of protocols are discussed in this sub section.

1.6.3.1 Power-Aware Routing Protocols

This category of routing protocols is designed to optimize power consumption of the network. Routing decisions are based purely on minimizing energy related metrics. Some routing protocols may strive for globally minimizing the metric. Typical energy related metrics like average residual energy or variance of residual energy might be considered. Some protocols might focus on minimizing a specific power metric locally using other energy metrics like minimizing the normalized per hop per flow energy consumption.

1.6.3.2 Geographical Routing Protocols

With the reducing costs of GPS devices, most of the current mobile devices come equipped with location support. There are also techniques available to estimate the distance between transmitter and receiver based on the signal strength. This additional information may be used by routing protocols to efficiently forward packets to the destination. Location aware routing protocols require an additional support for sharing the location information of all the nodes that form part of the network. There is also the requirement of updating this location database in the event of topology changes. The source node and all the intermediate nodes choose the next node to forward the packet by observing the location of the next node and the destination. Another typical approach is to model the underlying network as a graph and utilize the geometric information to compute the best route to destination. Due to the use of location information, geographic position, and typical modelling as a Euclidean graph, this category of routing protocols is also known variously as location aware routing protocols, position-based routing protocols, geometric routing protocols or graph-based routing protocols.

Typical geographic routing protocols are localized in nature. They use simple greedy techniques to choose the best neighbour to forward the packet. To add robustness to the protocol, additional steps may be taken to avoid loops and dead ends or voids.

1.7 Graph Models used in MANETs

Graphs have been traditionally used to model computer networks. By modelling the network using appropriate graph models, problems found in MANETs can be reduced to graph problems for which there are known solutions or problems whose solutions can be determined. A common problem like determining the best route from a source node to other nodes in the network reduces to finding all pair shortest path from the node. The problem of multicasting reduces to computing minimal Steiner tree. Frequency assignment problem reduces to colouring of the graph.[Stojmenovic 2003] Many problems in MANETs in their purest form are typically NP-Hard or NP-Complete. Quite often, the graph model will aid in identifying the complexity class of the given problem.

1.7.1 Unit Disk Graph

One of the simplest ways for modelling a MANET is with the help of a unit disk graph (UDG)[Clark *et al.* 1990].

Definition 1. A unit disk graph is defined as a graph $G(V, E)$, in which an edge e from a vertex u to another vertex v exists if and only if $|\overrightarrow{uv}| < \delta$, where δ is a threshold value.

A UDG can appropriately model a MANET, if we assume that all nodes are in the same horizontal 2-D plane, and the antennas used are omni-directional. For a mobile node,u, to communicate with any of its neighbours, the distance between u and its neighbors has to be less than the transmission range, Tx. The threshold value, δ, will then correspond to the transmission range, Tx, of the node in the graph.

Figure 1.6 shows a sample UDG consisting of three nodes. Here, as nodes A and B are within δ distance from each other, there is an edge connecting them. As can be seen from the figure, the distance between the nodes A and C, $d(A, C) > \delta$. Similarly, $d(B, C) > \delta$. Hence, there is no edge between the nodes A and C, nor between B and C.

Once the network has been modelled as a UDG, the operation of finding the shortest path between source and destination translates to finding shortest path between the corresponding vertices in the UDG. Weights can be suitably added to the edges of this graph to allow for additional link-based parameters while making routing decisions.

Most of the position-based routing algorithms assume that the network is in a two dimensional plane and is modelled as a UDG.

1.7.2 Unit Ball Graph

Unit Ball Graph (UBG) is an extension of a UDG to three dimensions. UBG provides for a rough approximation of an omni-directional antenna capable of transmitting a distance of in δ all directions around the node. Figure 1.7 shows a node in a UBG.

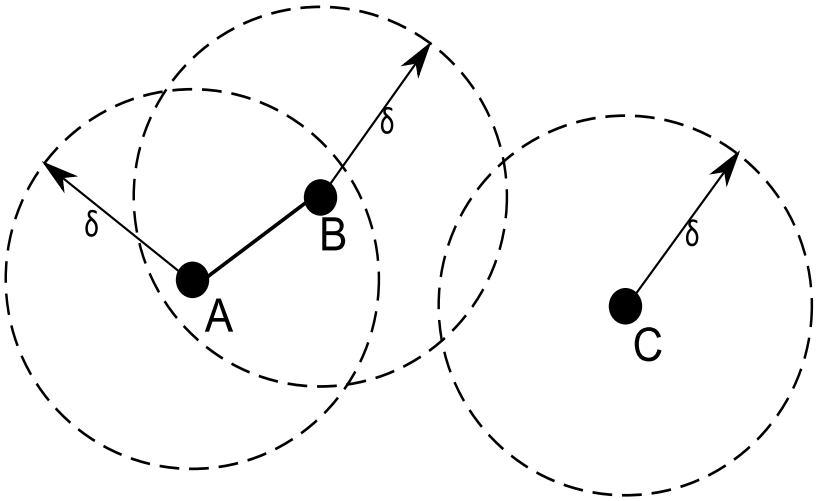

Figure 1.6: A Unit Disk Graph - with 3 nodes

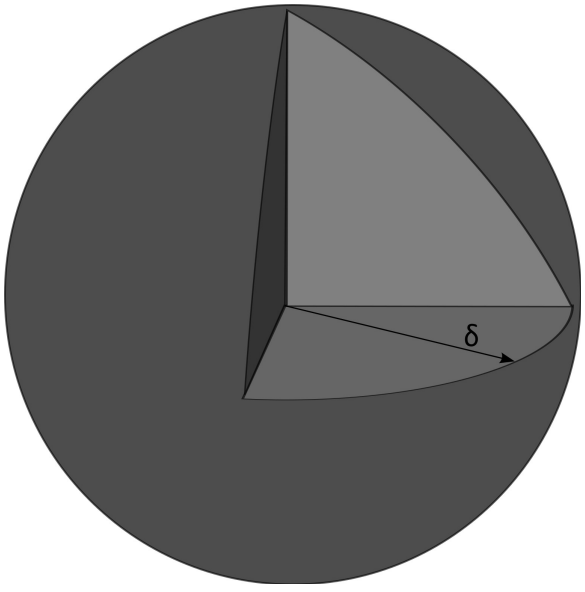

Figure 1.7: A Unit Ball Graph -

1.7.3 Relative Neighbourhood Graphs

Another class of graphs that can be used to model MANETs is relative neighbourhood graph (RNG).

Definition 2. For a set of n points belonging to a set V in a real space , the relative neighbourhood graph of V, $RNG(V)$, is a graph with vertex set V and set of edges defined as , where $\delta(p, q)$ denotes the distance between p and q.

The set of neighbours, $N(v)$, for a given node v in an RNG corresponds to the set of neighbours that are closest to a given node. By additionally checking that the distance between the nodes having an edge between them is less than the transmission range, the graph representation can be ensured to mimic a MANET with edges representing the closest link in any particular direction. Such a model can be used to ensure that the next hop in any particular direction is always to a nearby node. Such a property is useful while searching for paths with lowest energy or other similar metric.

1.8 QoS Aware routing in MANETs

Quality-of-Service(QoS) aware routing involves determining the best path that can satisfy the QoS requirements of the user. Intrinsic to the notion of QoS routing is an agreement or contract on behalf of the network to provide for measurable guarantees for the expected quality of service, in terms of parameters such as end-to-end delay, bandwidth, jitter, etc.

In typical wired networks, the intermediate node that acts as the router is only aware of packets passing through it, but is unaware of the particular source-destination pair that it is serving. This is because of the best-effort service that is provided in the Internet. QoS is meaningful only if the nodes become aware of the traffic flow corresponding to the source-destination pair. A logical connection is desired for determining whether QoS guarantees are met or not. Each intermediate node must be aware of the traffic flows and duration of each flow so that it can attain and preserve the relevant QoS parameters for such a logical connection. There must be appropriate resource reservation mechanisms

implemented, so that QoS guarantees can be attained.

1.8.1 Hard QoS and Soft QoS

In typical wired networks, the source-destination pair remains fixed. It is possible to reserve resources required to guarantee QoS in such static topologies. Since there are no issues of mobility of nodes or frequently breaking links, QoS guarantees can be ensured by the resource reservation scheme and the QoS routing protocol. Such a definitive guarantee of QoS is known as Hard QoS.

Hard QoS is typically very tough to guarantee in the case of MANETs due to the underlying network characteristics and the requirement for combinatorial stability. Nodes in MANET can wander from their current location leading to frequent joining and leaving of nodes forming the network. The links are unreliable due to mobility and due to drifting of nodes away from the transmission range of its neighbours. There is also the issue of hidden terminal and exposed terminal problems, and the resultant interference. Apart from these reasons, there is also the distributed nature of the MANET architecture. Resource reservation cannot be centralized as is the case with wired networks. Because of all these reasons, we cannot really build a virtual circuit that has guaranteed resources allocated to it for the entire end-to-end path. In MANETs, resource allocation and sharing happen as a result of competition for resources among the nodes in the vicinity and coordination among these nodes. Soft QoS, on the other hand, seems to be a better alternative for mobile environments.

In Soft QoS, a better-than-best-effort service is provided to support the QoS requirements. Whenever possible, QoS guarantees are provided in soft QoS. However, in the event of network partitioning or route breaks, QoS guarantee cannot be provided and hence an attempt to provide them is not made. Due to the fast changing state of the underlying network, a soft reservation of resources may be performed. For a traffic flow, it may be possible to notionally allocate the required resource. Based on the feedback from the network piggy-backed with acknowledgement packets or from control packets,

this reservation is updated accordingly. If a packet is not received within a pre-defined interval, the entire reservation is released. The sender may ask for a fresh reservation, if it wants to continue transmitting data with the same requirements.

1.8.2 Design Considerations

QoS requirements are usually demanded by real-time applications. In such systems, the performance can be optimized based on the feedback received from the system. By having efficient mechanisms to inform the nodes regarding the current network state including the available resources, the application can be suitably altered to provide the best quality for the currently available resource.

While routing the packets, apart from looking at traditional metrics like hop count, the protocol/algorithm must also take into consideration the metrics relevant to the QoS requirements. Apart from computing a path that minimally satisfies the QoS parameter, the routing algorithm can also be written to provide additional route optimizations such as provide lowest cost for the transmission, stability of route or computation of alternate path in the event of failure. The basic design considerations relevant for QoS aware routing protocol to satisfy the various design goals are listed below.

Bandwidth estimation

A typical QoS parameter that is commonly requested is the minimum bandwidth for the end-to-end route. To guarantee a resource like bandwidth, it is necessary to gather information regarding the available bandwidth for the entire path. The minimum bandwidth that can be guaranteed is reflected by the most congested link in the entire path. Also, a mobile node is going to be sharing its bandwidth with its neighbouring nodes. As part of the regular neighbour identification mechanism (HELLO protocol [Perkins & Royer 1999]), it is easy to gain information regarding the current bandwidth consumption from the neighbours of a given node. The link quality details can be piggybacked with the Hello packet, or the acknowledgement to the Hello packet. Even such a simple estimation strategy also consumes certain amount of bandwidth. Hence we not only require a good

bandwidth estimation strategy, but also need to control the frequency of the estimation related communication. Here, the trade-off is between accuracy of the estimation and the cost in terms of packet overhead and computational overhead required for the estimation mechanism.

Resource reservation

The wireless channel is a shared resource for mobile nodes within a particular region. This poses the challenge of identifying the best mechanism to implement any form of resource reservation and maintaining such a reservation.

Route discovery

The route discovery strategy can determine the efficacy of the QoS aware routing protocol. A proactive protocol might look appealing as every node following such a protocol is aware of the global state. However, maintaining the accuracy and the consistency of the global state across all nodes in a large MANET is a major concern. Instead of a proactive approach, if a reactive protocol is chosen, the overhead associated with propagating topology change information across the network may be reduced. But, such an approach still suffers from delay involved in determining the suitable route. Route maintenance in the event of route breaks remains a major concern with both approaches.

Route maintenance:

It is difficult to meet QoS constraints due to the frequently changing topology imposed by the mobility of the nodes. Any route maintenance mechanism must be fast enough so that the underlying topology doesn't change during the course of the maintenance. The traditional approach of initiating a maintenance algorithm only after the route break may not work in the case of MANETs. A better alternative is to come up with some prediction scheme or redundant routing strategy that can be made ready in the event of a route break, so that regular service can be resumed as fast as possible.

Route selection:

Route stability is one of the major requirements of QoS aware routing. Task of route stability will affect the end-to-end QoS in the event of route failures. While choosing a path for routing the packets, a path with largest available resources may not therefore be the best option. While choosing among alternate paths, the reliability of the path needs to be considered as well.

1.9 Multicast Routing in MANETs

Unlike unicast routing, where the focus is only to provide an efficient path from a single source to a single destination node, in multicast routing the single source node may want to communicate a packet to zero or more , but not all, nodes in the network. The multicast group comprises of a set of hosts that want to participate in the group communication. These hosts may join or leave the multicast group at any point of time. When the same data needs to be transmitted to a group of nodes, usage of multicast routing is going to be more efficient than multiple unicast transmissions. The multicast path is a tree rooted at the sending node. Data is transmitted through the tree, with packets getting forwarded through the tree alone. If the path from source to any two destinations in the tree shares a set of edges, then a single packet is passed through the shared edges, and the packet will be split into two different paths only at the node where the tree branches/splits into the two different paths. The multicast routing problem can be modelled as minimum Steiner tree problem of graphs[Gilbert & Pollak 1968]. The Steiner tree problem is one of the original NP-Complete problems defined in [Karp 1975].

The Steiner tree problem[Gilbert & Pollak 1968, Bern & Plassmann 1989] and the length of a Steiner tree are defined below:

Definition 3. Steiner tree problem for a graph $G(V, E)$ is the problem of finding a tree spanning all nodes in Q, where $Q \subseteq V$, such that the length of the resultant tree is minimized. The set of nodes S, where $S \subseteq V, S \cap Q = \emptyset$, are known as Steiner nodes.

Definition 4. Length of a tree $T(V, E)$ is defined as $len(T) = \sum_{e \in E} w(e)$, where w is the weight of an edge e.

The task of a multicast routing problem is to determine the correct set of Steiner nodes S so that the tree can reach all terminal nodes in Q. A Steiner minimal tree constructed for a local sub-graph of a G need not be part of the minimal tree constructed for the entire graph[Gilbert & Pollak 1968]. Steiner tree problem is known to be NP-complete for planar graphs[Karp 1975]. Approximation algorithms like the one given in [Borradaile *et al.* 2009] have been suggested for the Steiner tree problem. A related notion is to classify a tree as relatively minimal Steiner tree[Gilbert & Pollak 1968].

Definition 5. A Steiner tree $T(V', E')$ of a graph $G(V, E)$ is termed to be relatively minimal, if the length of the tree T, spanning the nodes Q, where $Q \subseteq V$, and a set of chosen nodes S, where $S \subseteq V, S \cap Q = \emptyset$, is minimum for the given topology.

For efficient communication, any multicast tree created must at least be relatively minimal. In this thesis, we attempt to determine a relatively minimal Steiner tree with the help of virtual force approach.

1.10 Resource Allocation in MANETs

Resource allocation is one of the challenging problems in a dynamically changing environment such as a MANET. Typical solutions tend to become inadequate for this particular domain. The problem of mutual exclusion of multiple tasks interested in sharing a common critical resource becomes especially challenging to solve, considering the restrictions implied by the underlying network. Over the years, attempts have been made to tackle the problem by using a centralised approach or a distributed approach.[Benchaïba *et al.* 2004] In the centralised approach, an assigned coordinator takes the responsibility of guaranteeing that permission for entering the critical section (CS) is limited to a single node. In the distributed approach, the permission to enter the CS is decided by some consensus mechanism amongst all nodes to determine the right to enter into CS.

1.10.1 Mutual Exclusion

For the mutual exclusion problem, we need to guarantee that no pair of operations in a critical section is concurrent. Apart from the usual requirement of absence of deadlocks, any solution for the mutual exclusion problem must ensure that the fairness property is satisfied. By fairness, it is meant that any task wishing to enter CS shall not wait indefinitely to enter into CS. A stronger version of fairness property is to mandate that any process P_i, that had attempted to enter CS before any other process P_j, must enter its CS before P_j does.

The stricter fairness requirement mandating absolute First Come First Serve (FCFS) behaviour may not be feasible in all distributed environments, especially when faced with

synchronizing a global clock in a resource constrained distributed system[Wang 1992]. In traditional mutual exclusion, the *REQUEST* for resources is serviced in the order of their arrival. Although FCFS is a natural policy for many applications, there may be circumstances where serving out of order might be sufficient with respect to the constraints of the network. It may be mandated for MANETs that a certain logical ordering be maintained as soon as a participating node becomes aware of any communication pertaining to the desire to utilise the common shared resource.

In most implementations, it is usually assumed that a process running in a particular site is going to cycle between non-critical sections of code and critical sections of code till it reaches its conclusion. It is also safe to assume that every process will have at least a small non-critical section code prior to entering the CS for the first time, and there shall be some continuation after leaving the CS. Thus, we mean that no participating process shall abruptly halt while it is executing in its CS[Lamport 1986]. A process may, however, decide to abort from the execution of the mutual exclusion algorithm at any point of time. Any such abort operation will bring its execution state outside the CS, and the process shall reset the related variables of the algorithms appropriately.

During execution in CS, there is a possibility of the following faults. Firstly, there is a possibility of unannounced death. An unannounced death happens when a process halts abruptly and without being detected. Secondly, a process may malfunction, in which it keeps setting its variables to arbitrary values. A malfunctioning process may enter CS even while another process is in CS. In the event of a transient fault, we may need to ensure that the process will execute normally after the issues leading to its malfunction are rectified. While these two are the standard faults that can be expected in any environment, we need to consider transient loss of communication in the case of MANETs and probability for a node to force itself to abort, say if its residual energy drops below a particular threshold. It is safe to assume that a node will know apriori when it is forced to abort an execution due to power constraints. However, a transient failure due to loss of communication link to its neighbouring nodes may not be predictable. Any such fault during the execution in CS may lead to a scenario where the other processes indefinitely

waiting for the faulty node to indicate that it had come out of the CS. Guaranteeing reliability in a mobile environment is hence a challenging problem. The most we may require a solution to the DME problem in the MANET environment is that whenever a node is reconnected to the network after a disconnection, or when a node wakes up after forcing itself to go into doze mode; the system shall continue its correct operation within a finite period of time.

1.10.2 Distributed mutual Exclusion

The problem of mutual exclusion is to ensure that only one of the concurrent processes is allowed to access the common shared resource at any given point of time. In the context of distributed systems, where processes can reside in multiple sites, the problem is known as distributed mutual exclusion (DME). The problem of DME has been studied extensively and a number of solutions have been proposed in the recent years. Any algorithm dealing with DME must consider the following requirements:

- **Safety :** No two sites must be allowed to enter CS simultaneously.
- **Freedom from deadlocks :** Any site interested in entering CS must be able to do so within a finite amount of time.
- **Freedom from starvation :** If a site has requested for entering CS, then it must not be kept indefinitely waiting for entering CS, while other sites enter the CS repeatedly.
- **Fairness :** The requests for entering CS must be executed in the order in which the requests are made.
- **Fault-tolerance :** In the wake of a failure, it is desirable that the algorithm reorganizes itself so that it can continue to function without any prolonged disruptions.

In the case of mobile ad hoc networks (MANET), the different nodes forming the network can move around at will. Thus, because of the movement of the nodes, there can be frequent link failures. Moreover, since these nodes are often running on battery supply, the node may fail due to exhaustion of batteries. The nodes may also go to a $SLEEP$

mode, shutting down all non-essential components to save power, if the battery charge level drops below some threshold. The DME problem is exasperated by the fact that node failures and link failures happen more frequently in MANETs than static networks.

1.10.3 Classification of Distributed Mutual Exclusion Algorithms in MANETs

Depending on the manner in which mutual exclusion is guaranteed, the solutions to the DME problem can be classified into two broad categories: token-based algorithms and permission-based algorithms. Apart from these two categories some other miscellaneous approaches have been attempted to solve this problems such as consensus-based approach. The various categories of DME algorithms are discussed below.

1.10.3.1 Token-based Approach

As the name suggests, the token-based approach relies on the possession of a token to enter the CS. As the token can only be possessed by one node at any given point of time, this category of algorithm prevents any two nodes from simultaneously accessing the critical resource. Token-based algorithms may be further classified into the circulating token method and the requesting token method. In the former method, a token is passed among all the participating nodes in a logical sequence. A node may enter CS only when it receives the token. After it completes its CS, the node will forward the token to its successor in the logical structure irrespective of whether the successor is the next node that requested for entering CS or not. In the requesting token method, the onus of retrieving the token from the current holder falls on the node requesting to enter the CS.

1.10.3.2 Permission-based Approach

A permission-based algorithm relies on getting permission from all the participating nodes by explicitly asking for permission to enter the CS from each of the participating nodes. In this approach, all nodes are involved in determining the next node that can enter CS. A node interested in entering critical section will ask permission from other nodes participating in the network. Any node receiving such a request will grant permission to the requesting node unless it itself is asking for entering CS. Once the requesting

node has received permission from all the other nodes, it enters CS. After it has completed its execution in CS, the node will release the resource for others to use. In this thesis, we are exploring this approach further.

1.10.3.3 Consensus-based approach

Another approach that can be used for entering CS is by consensus or election. In this class of DME algorithms, the requesting node asks other nodes for permission, similar to the permission-based approach. But unlike the permission-based approach, the node doesn't wait for all nodes to grant permission. Instead of asking for a grant of permission, this approach only asks for a vote. If enough nodes have voted in favour of the requesting node, then it can enter CS.

1.11 Summary of Contributions

The following are the contributions of the research carried out as a part of this thesis work.

- Developed an energy-aware routing algorithm for MANETs that used a novel cost metric.
- Developed an Efficient On-demand Routing Protocol for MANETs using the Bayesian approach.
- Developed a new model, termed as Generalized Energy Consumption Model, to support energy harvesting devices in wireless sensor networks.
- Analyzed the use of virtual force technique for multi-cast routing in MANETs.
- Introduced the notion of dampening forces for the first time in multicast routing algorithms for MANETs.
- Introduced the notion of virtual force based on regions for the first time in multicast routing algorithms for MANETs.
- Developed the Multicast Routing Algorithm using Virtual-force for QoS-aware multicast routing in MANETs.

- Developed the sector-based Virtual Force Multicast routing algorithm for QoS-aware multicast routing in MANETs.
- Developed the Virtual Multicast Tree routing algorithm for for QoS-aware multicast routing in MANETs.
- Developed a novel permission-based distributed mutual exclusion algorithm for MANETs that used adaptive timeout for handling fault-tolerance.
- Introduced the notion of arbitrator node in the "look ahead" technique to handle distributed mutual exclusion among nodes in multiple regions.
- Developed a Reliable Arbitrator-based Distributed mutual exclusion algorithm for MANETs.

1.12 Thesis Organization

Rest of the thesis is organized as follows. Chapter 2 discusses the previous approaches available in the literature and their significance. Chapter 3 discusses our attempts on energy-efficient protocols and algorithms for wireless ad hoc networks. Chapter 4 discusses three QoS aware multicast routing algorithms and protocols proposed by us. Chapter 5 discusses two permission-based algorithms for solving the DME problem. Chapter 6 provides conclusions for our work.

Chapter 2

Literature Survey

Mobile Ad-hoc Networks(MANET) are useful in application domains such as battlefields, search-and-rescue operations, and collaborative computing where fast deployment is of utmost importance. During the recent past, users expect to have seamless data transfer of both textual and audio/video data in these environments. The users currently expect their devices to be active for larger duration while engaging in various forms of data communications. In this thesis, the focus is on providing energy efficient and QoS-aware data communication for transmitting streaming and non-streaming data to unicast and multicast destinations. Later, we focus on ensuring mutual exclusion for accessing critical resources in the network, even in the event of changes in the topology. We restrict ourselves to the distributed mutual exclusion(DME) problem for MANETs. The DME problem can be visualized with an objective of minimizing the number of messages used and the number of hops traversed by the messages. In this chapter, various approaches used in the literature to address routing and resource allocation issues are presented. Then we discuss multi-casting challenges for routing in MANETs. We explore the various trends in routing and quality-of-service (QoS) issues including network lifetime and other energy-related metrics in the beginning. The last part of this chapter discusses the various techniques used in the literature to deal with the DME problem.

This chapter is organized as follows: Section 2.1 discusses the various traditional approaches for unicast routing in MANETs. In Section 2.2, geographic or geometric routing algorithms are discussed. We have used position based approaches in a major part of our

thesis. Section 2.3 discussed some of the background details for the Quality-of-Service (QoS) aware routing algorithms. We discuss multicast routing algorithms for MANETs in Section 2.4. In Section 2.5, we discuss the distributed mutual exclusion (DME) problem in MANETs. We conclude the chapter in Section 2.6.

2.1 Routing techniques in MANETs

Mobility of nodes is one of the key challenges for routing in MANETs. Centralized approaches for routing do not provide good results in MANETs. Ensuring combinatorial stability in a mobile and scalable environment is not achievable, if the global state has to be accurately maintained at a single site. There is also the problem of the central node going out of range, i.e. transient failure of the central node or a link connecting the central node to a sub-network. Considering the autonomous nature of the mobile nodes, distributed algorithms are employed for enabling routing in MANETs. In this section, we review a few traditional routing protocols/algorithms used in MANETs. Destination Sequenced Distance-Vector algorithm [Perkins & Bhagwat 1994] is a representative of a proactive routing protocol. Dynamic Source Routing (DSR) [Johnson & Maltz 1996] is an example for source routing approach that uses on-demand routing technique. Ad hoc On-demand Distance-Vector(AODV) [Perkins & Royer 1999] routing protocol is a very good representative of a reactive approach for routing in MANETs. We also discuss Location-Aided Routing (LAR)[Ko & Vaidya 2000] as a representative of the geographic routing approach in MANETs.

2.1.1 Destination Sequenced Distance-Vector Protocol(DSDV)

The Destination Sequenced Distance-Vector protocol (DSDV) [Perkins & Bhagwat 1994] is a classic example for a table-driven routing protocol for MANETs. This protocol is an extension of wired routing protocols to the wireless environment based on an extension to Bellman-Ford algorithm [Goodrich & Tamassia 1998]. Each node maintains a table that contains the shortest path to every other node in the network. Consistency of this routing scheme is effected by frequently updating the local table. Sequence numbers are used to prevent loops.

DSDV is one of the classical proactive routing protocols. The global state is captured in each and every node of the network. The state information is shared in both incremental and whole table formats. As the updates have unique increasing sequence numbers, each node can recognize the latest update message and use them for the update operations of the routing table.

Combinatorial stability is crucial for this proactive protocol. As the protocol relies on the correctness of the table data, the frequency of the update operations becomes crucial. To maintain the latest network topology, it is desired to send as many update messages as possible. However, if the number of update messages are increased, the effective bandwidth of the links drop down. The additional overhead associated with the update messages is thus a major challenge with this type of protocols.

2.1.2 Dynamic Source Routing (DSR)

One of the early on-demand routing protocol for routing MANETs is the Dynamic Source Routing(DSR) [Johnson & Maltz 1996]. The bandwidth constraint associated with table driven approaches are addressed well in this type of protocol. One of the main contributions of DSR is its lack of special beacon messages (or hello packets) for capturing the local state information. This protocol uses source routing principle [Peterson 2011], storing the entire route from the source to destination in the route construction phase.

Being one of the classical on-demand routing protocol, DSR uses a RouteRequest (RREQ) packet to construct the route from the source node to the destination node. The destination, on receiving the RREQ packet, responds with a RouteReply (RREP) packet. The RREQ packets are flooded, while the RREP packet returns through the exact reverse path that was used by the RREQ packet. Any node receiving a RREQ packet will flood it to its neighbours. Since a single request from a source could be flooded by many nodes in the network, this approach will result in a formation of loops. To avoid a packet to be forwarded indefinitely through a loop, each request is uniquely identified by a sequence number. The sequence number is also useful in restricting multiple transmission of the same RREQ packet by an intermediate node. Every RREQ packet is also equipped with a Time to Live (TTL) counter to limit a request to be forwarded indefinitely. The nodes

can be run in a promiscuous mode, listening to even those communications that are not addressed to the node, to capture extra details regarding the local topology. Route cache can be used to optimize the route construction phase. These features have become almost de facto for most of the on-demand routing protocols, like [Singh *et al.* 1998], [Jacquet *et al.* 2001], [Basagni *et al.* 1998], [Lee *et al.* 2002] [Shah & Rabaey 2002].Figure 2.1 shows an example route discovery using DSR. Source nodes and destination nodes are labelled accordingly as shown in the figure. Arrows indicate the flow of request messages, and dashed arrows indicate the flow of reply messages.

The major advantage of reactive protocols, as demonstrated by DSR, is the lack of periodic table update messages. Route to a destination node is calculated whenever data needs to be sent to it. Therefore, unlike table-driven approaches, there is no need to maintain routes to all nodes that need not be contacted at all. The control overhead is further reduced by the use of route caches.

On the downside, route maintenance mechanism in DSR is not capable of repairing locally broken links. The routing cache used may contain stale topology information long after the link was broken. The routing overhead is directly proportional to the length of the path. Compared to static and low-mobility scenarios, the performance of this protocol drops in high-mobility scenarios. [Murthy & Manoj 2004]

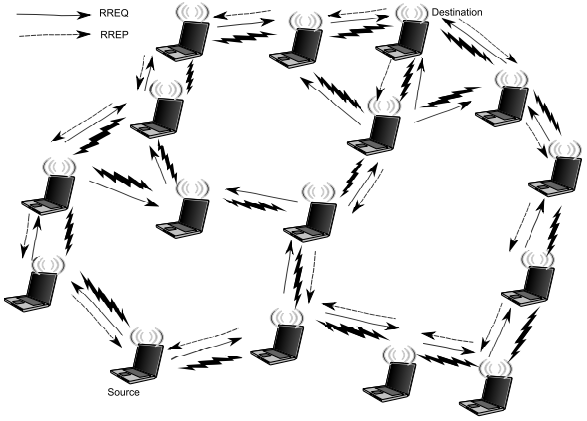

Figure 2.1: Dynamic Source Routing - DSR

2.1.3 Ad Hoc On-Demand Distance-Vector (AODV) Routing Protocol

Ad hoc On-demand Distance-Vector (AODV) routing protocol [Perkins & Royer 1999] is one of the most popular routing protocols for MANETs. AODV, unlike DSR [Johnson & Maltz 1996], uses periodic Hello packets to capture the local state information (one hop neighbourhood). The routing table in the source node and the intermediate nodes store the next-hop information, instead of the complete path to the source/destination. Destination sequence numbers are used to identify the desired destination. Each intermediate node updates information related to the path to destination node whenever it receives a packet with a newer destination sequence number. When a node receives duplicate route request packets which are identified by the source id-broadcast id pair, it discards these extra packets.

AODV allows an intermediate node containing a direct path to the destination, or a node whose neighbour is the destination to transmit RREP packet back to the source node. Timers are used to ensure that old or stale information is not maintained in the routing cache.

[Perkins & Royer 1999] had designed the AODV protocol without taking into consideration the restrictions imposed by the physical medium. This allowed the routing protocol to be constructed with the basic assumption that all mobile nodes that are neighbours (all nodes that respond to the hello packet) can broadcast the RREQ packets to determine the destination. Since Hello packets are used as the proof of neighbourhood, any node whose packet is received is considered as a neighbour. AODV does not verify bidirectional communication. Figure 2.2 shows an example route discovery using DSR. Source nodes and destination nodes are labelled accordingly in the figure. Arrows indicate the flow of request messages, the dashed arrows indicate the flow of reply messages. The dashed arrows indicate the flow of reply messages through which also reply messages could have been sent back to the source node.

2.1.4 Location Aided Routing (LAR)

Ko and Vaidya suggested the use of location information already available with the help of GPS receivers for routing in their protocol, Location-Aided Routing(LAR) [Ko

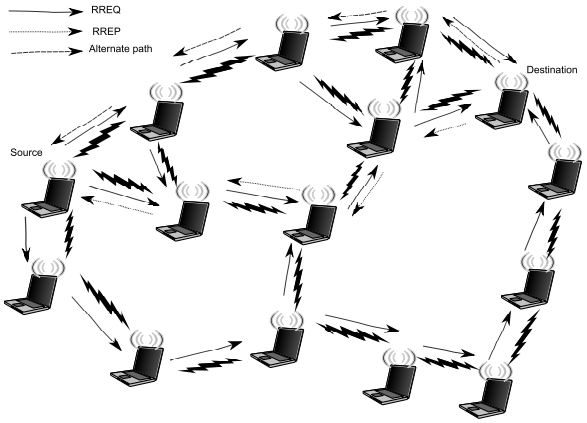

Figure 2.2: Ad Hoc On-Demand Distance-Vector Routing Protocol - AODV

& Vaidya 2000]. The use of location information is used to reduce control overhead during the route discovery phase. This is unlike Zone Based Routing (ZBR) [Haas *et al.* 2002], where the proactive phase is limited to the source's zone, while following a reactive approach for the rest of the network. LAR restricts the control packets to *RequestZone*, the region through which the path-finding control packets can propagate. The *ExpectedZone* indicates the region in which the destination is likely to be found. The diameter of the *ExpectedZone* can be altered to allow varying mobility of the destination node. Though two algorithms are suggested as part of [Ko & Vaidya 2000], it is the second algorithm that is significant in our context, which we will henceforth refer to as LAR.

In the second algorithm of LAR, the RREQ packet contains the location information including the coordinates of both source(S) and destination (D) nodes, and the current distance between them. It is to be noted here that like most early geographic/geometric routing protocols, LAR inherently assumes that the nodes lie on a 2-D plane. The intermediate node decides to forward the packet based on this Euclidean distance stored in the RREQ packet. Let the distance between two nodes u and v be denoted as $d_{u,v}$. If the intermediate node is closer to destination than the source node, i.e., the distance between the intermediate node(u) and destination(D), d_u,D is less than the distance $\alpha \dot{d}_{s,D} + \beta$, where α and β are parameters based on error in location estimation and mobility, then the intermediate node will forward the RREQ packet. The forwarding node replaces the value of $d_{s,D}$ in the RREQ message with $d_{u,D}$.

The main advantage of LAR is the reduction of control overhead. In this protocol, more bandwidth is available for data transfer compared to proactive or reactive protocols.

Another advantage of protocols using location information is that only the nodes that are likely to be used for communication are involved in processing.

2.1.5 Other traditional algorithms

The four algorithms described in the previous section are a subset of protocols that formed part of the early work related to routing problem in MANETs. Other approaches for routing include link state routing paradigm proposed in Optimized Link State Routing (OLSR) protocol [Jacquet *et al.* 2001]. They defined selected nodes termed as multi-point relays to broadcast the messages while performing flooding. Link state information is generated only by these multi-point relay(MPR) nodes. It also allowed for partial link information sharing by broadcasting only information related to the MPR nodes alone. Neighbourhood detection is performed by means of the HELLO protocol. The HELLO messages are used to identify the neighbouring MPR nodes. The set of MPRs are changed whenever there is a link failure in the network. All the information related to these nodes form the global state of the overall topology. Routing tables are used to reflect this topology information. It is to be noted here that the MPR set is a subset of the neighbour set of a node. The choice of MPR nodes is such that the overall connectivity of the network is not challenged. Being a table-driven protocol, OLSR comes with the advantages and disadvantages of such an approach. The protocol works efficiently when the stored network information is used frequently.

Traffic-Aware and Low-Overhead Routing Protocol (TALORP) [Alsheakhali & Awad 2008] used a different approach to route maintenance. The low-overhead routing protocol (LORP)[Yu *et al.* 2007] is used to generate the efficient route from source to destination. Along with this best route, TALORP also computes the relative significance of the rest of the nodes that may form the next best path. They use internal data structures to maintain a measure of significance of the nearby nodes in the event of a link failure by observing both hop counts and traffic intensity. This information is used at the time of link failure to compute the next best path from any point of failure.

[Kang & Ko 2010] suggested a route maintenance and restoration scheme to reduce control overhead in location-based hybrid routing protocol for MANETs. They guide the

route maintenance by tracking the destination node's location information while performing the routing operation. Their protocol is designed in such a manner that the nodes that could be part of the back-up paths would definitely receive the beacon messages from the main nodes. These are the nodes that participate in route restoration in the event of a link failure. By using the location information from the beacon messages, they were able to restore broken routes effectively.

2.2 Geographic or Geometric routing algorithms

In this section, we explore in detail the geographic or geometric routing algorithms for dealing with routing protocols. Geographic approaches or position based approaches typically rely on position information to determine the set of nodes through which route request will be forwarded. In geometric algorithms, the underlying network is typically modelled as an appropriate graph thereby reducing the routing problem of a communication network to a geometric problem with known or easily computable solutions. The set of algorithms classified in literature as geometric, geographic, position-aware or graph-based is described under a single approach in this thesis as these approaches can trivially be reduced to a graph problem, and the names are used interchangeably.

In the early days of research in routing problems of MANETs, there was significant reluctance in the use of geometric algorithms due to the high cost for capturing the location information. In the late 1990s, the Global Positioning System (GPS) had gained significant traction in consumer electronics industry. [Ko & Vaidya 1999], [Mauve *et al.* 2001], [Mauve *et al.* 2003] [Liao *et al.* 2001] [Stojmenovic 2002] The sudden availability of cheap, low cost GPS devices, and suitable techniques for determining relative positions combined with the demand for energy efficient routing resulted in acceptability of position-aware routing approaches in MANETs. Distance between two neighbours can be estimated based on the strength of the received signal. The relative distance between nodes can be shared among neighbours. With a combination of hardware and software, the position aware routing protocols have become practical. There are scalable location services like geographic location service (GLS) [Li *et al.* 2000] that can provide location information to the participating nodes. These services scale well, as the increase in storage and bandwidth

overhead is gradual with respect to increase in number of nodes.

Localized routing algorithms are characterized by the use of local information alone by the participating nodes to determine the next hop for forwarding a packet from the source to the destination node. These algorithms are distributed in nature, usually using a greedy strategy to compute the best path towards the destination. In routing table-driven networks, the routing table update operations need to account for route maintenance due to frequent node mobility and change in network topology. Each such change in topology needs to be propagated to the rest of the network, resulting in increased overhead. On the other hand, the position-aware localized routing algorithms require accurate location information of the current node and a good estimate of position of the destination node. The only overhead for these algorithms is due to the underlying position service like GLS. A change in topology or change in position of the destination does not affect the routing strategy to be made by the intermediate nodes as long as the rough location of the destination is available. As there is no need for communicating the routing tables, the communication overhead for table maintenance can be averted.

We discuss various strategies and approaches used by various routing algorithms in Section 2.2.1. We briefly discuss various metrics used in the greedy choice for computing the source-destination path in the sub section. We then discuss a set of geography-based routing algorithms in Section 2.2.2, to provide an overview of the various strategies that had been used in this class of algorithms. In Section 2.2.3 and 2.2.4, we discuss algorithms representing two approaches we had explored in this thesis. In Section 2.2.5, we describe the details of the virtual force technique for performing routing in MANETs. The virtual force technique was one of the major approaches we had explored further in our thesis. The other significant approach of dividing the network into regions was derived from the notion of minimum connected dominating set or its dual problem of maximal independent set. The algorithms related to these two approaches are discussed in Section 2.2.6. Another related notion of well-separated planar decomposition, which forms the basis of one of our multicast routing algorithms is provided in Section 2.2.7.

2.2.1 Strategies and approaches in position aware routing algorithms

In a realistic network, the routing algorithm is preferred to work in a distributed manner. The resultant algorithm must guarantee loop-free routes to the destination to avoid a packet to cycle around the network in infinite loop. It needs to be demand-based, and capable of handling temporary failure of nodes due to nodes temporarily going to power-saving mode (SLEEP mode) when the residual energy of the node falls below a threshold. [Giordano *et al.* 2001] [Stojmenovic 2002]

The position-based algorithms adopt three strategies to compute the route; namely single-path, flooding, and multi-path.[Stojmenovic 2002] An ideal localized algorithm will just generate a single path from source to destination. In such a single-path strategy, there is only one active route request packet getting forwarded in the network. Flooding strategies on the other hand transmit the request message through the whole network or through all the nodes in a subnet, thereby computing all the paths to the destination. The multipath strategy uses a balanced approach, computing a few single-path routes from the source to the destination. The algorithms that use the single-path strategy are typically localized in nature, relying only on the local state information. The algorithms advance purely on the basis of the greedy choice. The single path strategy may provide the best path from the source to the destination, but route maintenance phase will include re-computation of the entire path, leading to significant overheads. Some amount of memorization of path may be required. In the event of intermediate node failure or link failure, this strategy either needs to re-compute an alternate path from source to destination, or it needs to compute a path to forward packets around the point of failure. The flooding strategy has a higher overhead for computing the paths similar to the proactive protocols discussed earlier in section 2.1.1, however there is less overhead in switching to an alternate viable path to the destination in the event of a failure. The multipath strategy includes lesser overhead as compared to flooding strategy while constructing the initial path. It also suffers lesser degradation in performance with respect to a single path strategy. Thus multipath strategy is a compromise solution when compared with the other two strategies; sharing the advantages as well as disadvantages of the two approaches.

The graph-based approaches model the network as a planar graph, adding sufficient

restrictions on the model based on their goals. The localized algorithms use metrics to reflect the design goal, which in turn reflect the greedy choice made by the algorithm. Unit Disk Graph (UDG)[Clark *et al.* 1990] is used to model most of the networks discussed under this section. Wherever the model is different, it shall be explicitly mentioned in this chapter.

One of the simplest metric to use is the hop count [Stojmenovic 2002]. Hop count for a source-destination pair is defined as the length of the path corresponding to the route assuming that weight of each edge is exactly one. When the Euclidean distance between source and destination nodes are to be measured, they are typically named as distance [Liu & Wu 2006]. Hop count reflects the UDG model of the underlying network, including the notion of unit cost for communicating between any two nodes within a transmission range, TxR. The transmitter, in this case, is assumed to be capable of transmitting only at the high power setting. In other words, this metric is sufficient to reflect the cases where it is assumed that the transmitting power of the nodes cannot be adjusted for each communication.

A power metric that depends on the Euclidean distance between nodes can be used, if we assume that the transmitter's power can be adjusted based on the distance between the two nodes.[Stojmenovic 2003] The greedy choice in this case shall correspond to an optimization problem, with the stated goal as minimization of the measure reflecting the cumulative sum of power values for the path. When we consider energy as a parameter, the transmission cost cannot be visualized entirely as a per path measure like the way it can be done for the hop count metric. If a node forms part of more than one flow from multiple source-destination pairs, then it might get drained faster in a simple greedy approach. While considering a power metric, the residual energy becomes as important, if not more important, parameter in an energy efficient algorithm. A suitable cost metric may be used when the goal is to maximize the number of network flows that can be maintained for finite duration of communication.

Delivery ratio is the measure of the number of packets received by destination and the number of packets actually transmitted[Stojmenovic 2002]. The routing layer must be capable of guaranteed delivery, with a certain assurance that the message will eventually be delivered using the retransmission scheme of the underlying medium access layer in

the event of collisions. Due to the dynamic nature of MANETs and possibility of obstacles and voids in the underlying network, a certain degree of deviation in the computed positions is likely to occur. A robust routing strategy needs to be capable of handling any such deviations from the graph model.

Though it has been mentioned in this section that the algorithms use a greedy choice to construct the path from source to destination, relying only on the greedy mode may not be sufficient to guarantee delivery of packets from source to destination. In the greedy mode, the optimality measure may result in the choice of a neighbour as the best choice from the given node. However, as the choices are made locally, the algorithm may lead the packet to a node from which there is no further route to reach the destination using the greedy choice. Thus, though a greedy choice may always lead to the local optima. If the routing algorithm gets stuck in a local optimum, an appropriate recovery mode of operation is required to ensure that the routing algorithm will eventually converge.

2.2.2 Simple position based schemes

In the previous section, various strategies, metrics and concerns for using position-based routing algorithms were discussed. In this section, a set of simple greedy schemes for position-based localized routing algorithms are discussed.

Takagi and Kleinrock[Takagi & Kleinrock 1984] suggested the Most Forward within Radius (MFR) scheme for forwarding a packet. In this greedy scheme, the progress of an intermediate node is measured as the projection of the node on the line connecting the transmitting node S and the destination node D. The next hop is the node with maximal projection. The essence of this routing scheme is that the progress as mentioned here reflects the best neighbour that can effectively forward the packet with the least cost. Instead of the MFR strategy, the neighbour with the nearest progress can be used, termed as the Nearest Forward Progress(NFP)[Stojmenovic 2002] technique.

Instead of computing projection to determine the progress, the Euclidean distance between the current node U and destination node D can be used, as was suggested by Finn in [Finn 1987]. In this scheme, only all the neighbours closer to D are considered for the greedy choice. If there is no such node, then it indicates that the greedy choice is

not advancing, and we may have to enter into the recovery mode. In Geographic Distance Routing(GEDIR) [Stojmenovic & Lin 2001b], instead of restricting to the neighbours closer to D, all neighbours are considered while making the greedy choice. Lack of advance can be identified when the message is forwarded to the node that sent the message.

Compass routing technique as proposed by Kranakis, et al, in [Kranakis *et al.* 1999] uses the angle between the neighbour, source and destination for the greedy choice. It is assumed here that the node closest to the general direction of the destination is the best node to forward the packet.

Stojmenovic and Lin suggested a power metric to compute the next hop neighbour that mimics the best node closest to the destination in [Stojmenovic & Lin 2001b]. In this algorithm, the hypothetical position of the best imaginary neighbour to forward the packet is determined. The neighbour of the current node that is closest to this hypothetical best node is used to forward the packet. The advancement of this algorithm will halt when no such node can be determined.

By altering the underlying model to a k-disk graph, Yeh [Yeh 2001] proposed varying radius algorithms to improve throughput, to reduce latency and to lower power consumption in MANETs.

Greedy-Face-Greedy (GFG) protocol was suggested by [Bose *et al.* 2001] as a simple alternative protocol that switches between greedy mode and recovery mode to eventually deliver the packet to the destination. This algorithm uses the notion of Gabriel Graph.

Definition: Gabriel graph, GG(S) is a geometric graph in which the edge (u,v) is present if and only if disk(u,v), the disk containing the diameter (u,v), contains no other point of S.

A connected planar sub graph is first extracted by modelling the network as a Gabriel graph. The algorithm proceeds by identifying the face, determined as the polygon bounded by the edges of the graph, around which the packet needs to be forwarded based on the right hand rule. The intersection of the face and the line connecting the current node and the destination node is used as the guide to reach the destination. This scheme alternates between directional routing and face routing to forward the packet to the destination.

2.2.3 Greedy Perimeter Stateless Routing (GPSR)

The Greedy Perimeter Stateless Routing(GPSR) [Karp & Kung 2000] algorithm uses greedy mode for predominantly forwarding the packet towards the destination, and perimeter node to tackle situations where local optima does not lead to a path to the destination. The greedy choice involves forwarding the packet to the neighbour with the least geographical distance to the destination. This activity only requires local topology information that is captured with the help of a simple beaconing algorithm like the HELLO protocol [Perkins & Royer 1999][Murthy & Manoj 2004] . Figure 2.3 illustrates the working of the GPSR algorithm in the greedy mode. The source node, S, looks at the location of the destination node, D1. The shortest route to D1 shall pass through its neighbour node C. So, the data is first forwarded to the node C. Then C determines that the shortest route to D1 shall be through the node B. Node B identifies that D1 is its neighbour, and forwards the data packet to D1. The path taken by the packet is shown using thick lines in the figure.

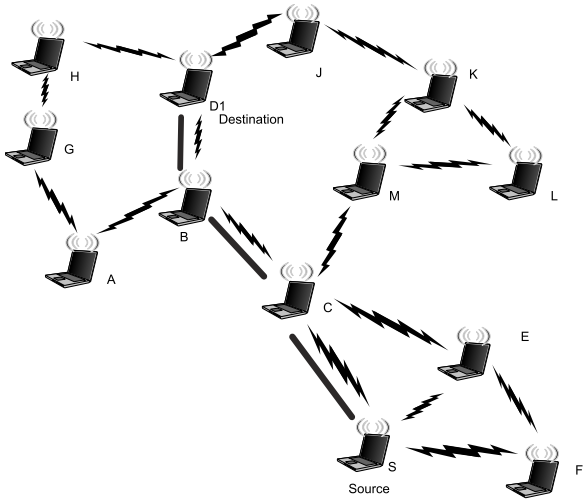

Figure 2.3: Greedy Perimeter Stateless Routing(GPSR)(1) - Greedy mode

The perimeter mode of GPSR is based on the right hand rule. GPSR allowed for shifting to perimeter mode whenever a void is encountered on the forward path. Once the perimeter mode had taken the packet away from the void and there is a viable greedy path towards destination, it reverts to the greedy mode to forward the packet to destination. For the purpose of illustration, the same network is used as in Figure 2.3, but with the change in location of node D1. If the node D1 had drifted away from the node B, the network would look as in Figure 2.4. As can be seen from the figure, GPSR uses the

greedy method to forward the packet till node B. But when the packet reaches B, it would not be able to forward the packet further as the packet had encountered a void. Voids are said to occur in geographic routing when the packet cannot be forwarded in the intended greedy direction due to lack of any neighbouring nodes towards the destination. When the packet encounters a void, GPSR shifts to the perimeter mode. The packet, in this example, shall be forwarded via the nodes $B \rightarrow C \rightarrow M \rightarrow K \rightarrow J \rightarrow D1$ as illustrated in Figure 2.4.

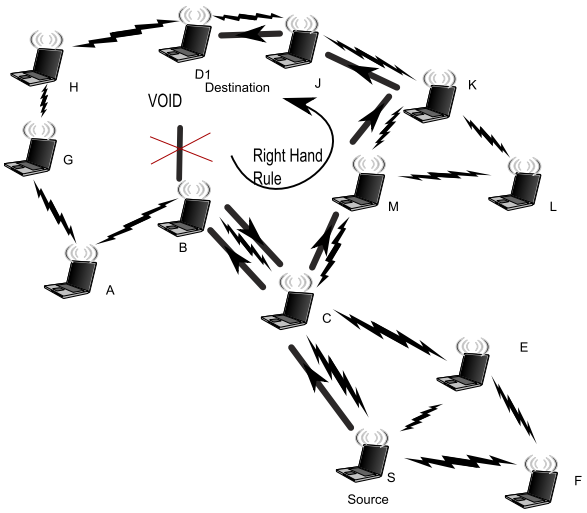

Figure 2.4: Greedy Perimeter Stateless Routing(GPSR)(2) - Perimeter mode

2.2.4 Spine Routing

[Sivakumar *et al.* 1998] introduced the notion of spine for routing in a self organizing network. Spine forms the backbone for the network through which all routing is carried out. They suggested two variants: Optimal Spine Routing(OSR) and Partial-knowledge spine routing(PSR). The spine nodes, once formed are capable of forwarding the routing requests appropriately through the backbone. They used an approximation Minimum Connected Dominating Set(MCDS) algorithm to identify spine nodes. In OSR, a non-spine node will pass the routing request to the nearby spine node. The spine node, as it has the information about all other nodes in the network, shall provide the route to the requesting node. It is to be observed here that the spine is used only to update the global information, while the routing decision is made locally. OSR assumed that the up-to-date topology information for the entire network is already available. Based on the assumptions made

by it, OSR is not a practical version of routing. However, the notion of spine forms the basis for other routing algorithms like GCRP [Fotopoulou-Prigipa & McDonald 2004], Proactive Cluster-Based Distance Vector (PCDV)[Hoon & Seok-Yeol 2007], Swing+ [Liu & Wu 2006] [Liu & Wu 2009]. PSR suggested in [Sivakumar *et al.* 1998] was constructed to be more practical. The link state information stored in each node is primarily used to perform routing as done in OSR. A notion of add and delete waves is used to share the link state information among the spine nodes. If the mechanism fails, then PSR shifts to a probe-based routing strategy to compute the source-destination route. Figure 2.5 illustrates a simple network. The spine constructed for the network is shown using thick lines in the figure. It can be inferred from the figure that the spine nodes B, C, E, M and K are capable of reaching all the non-spine nodes directly.

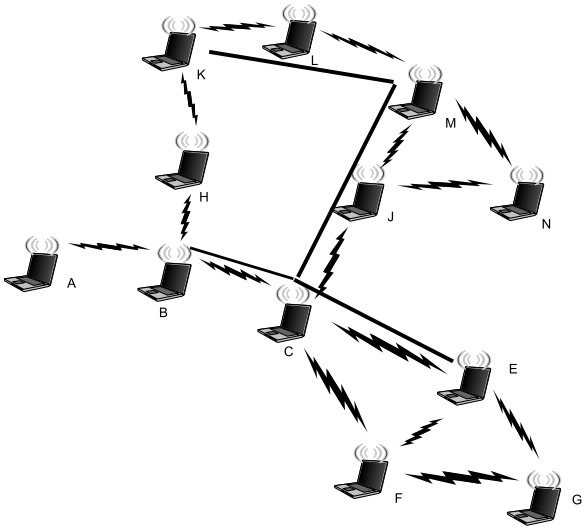

Figure 2.5: Spine Routing - a sample network

2.2.5 Virtual force technique

[Fotopoulou-Prigipa & McDonald 2004] suggested the use of geography-based virtual circuits, termed as geo-circuits, for unicast routing in the Geo-Circuit based Routing Protocol(GCRP). They computed the unicast path from source to destination based on greedy position-based strategy and assigned a virtual circuit number to such paths. Once a geo-circuit was established, they re-used this geo-circuit to send future packets to the destination till the end of traffic or till the path is updated due to node mobility. This notion was expanded by adding the concept of virtual force by Liu, **et al.** in [Liu & Wu 2006, Liu &

Wu 2009].

The notion of virtual force had been explored in deployment problems in wireless sensor networks [Poduri & Sukhatme 2004],[Zou & Chakrabarty 2003], and in routing in MANETs [Liu & Wu 2009]. In the virtual force approach (VFA), the participating nodes and/or the packet in transit are applied with some electric charge. The electrostatic forces are computed and the routing decision is made based on the magnitude and direction of the resultant force. [Poduri & Sukhatme 2004] demonstrated the use of a combination of attractive and repulsive forces for solving sensor deployment problem in wireless sensor networks. Liu, et al. [Liu & Wu 2006], [Liu & Wu 2009] used the concept of virtual force in unicast routing for MANETs. In the routing algorithm for MANETs, the resultant force is used only to make a decision to move the packet in forward direction.

Liu, et al. suggested two routing algorithms, SWING [Liu & Wu 2006] and SWING+ [Liu & Wu 2009], for unicast routing in MANETs based on the notion of small world networks. These algorithms expanded the notion of virtual circuits as described by Fotopoulou, et al. [Fotopoulou-Prigipa & McDonald 2004] to include long *virtual logical links* to the surrounding neighbourhood of the current node with the help of virtual force. These two protocols were the first unicast routing protocols to use virtual force to compute the path towards destination. By computing the repulsive force exerted on the neighbours of the current node, the next hop neighbour was chosen as the node with the maximum repulsive force from the source node towards the destination node. They demonstrated that their approach works efficiently in a 3-D environment, as well. Figure 2.6 illustrates a simple scenario where data needs to be transmitted from source to destination, D1. The thick line indicates the path through which the packet will be forwarded. The arrow indicates the direction of force on the packet near the source node.

2.2.6 Minimum Connected Dominating Set(MCDS)/Maximal Independent Set(MIS) based algorithms

We have already mentioned about one of the algorithms that use the MCDS approach for the routing problem when we introduced the spine routing concept by [Sivakumar *et al.* 1998]. The essence of the MCDS approach is to find C, the set of dominator nodes

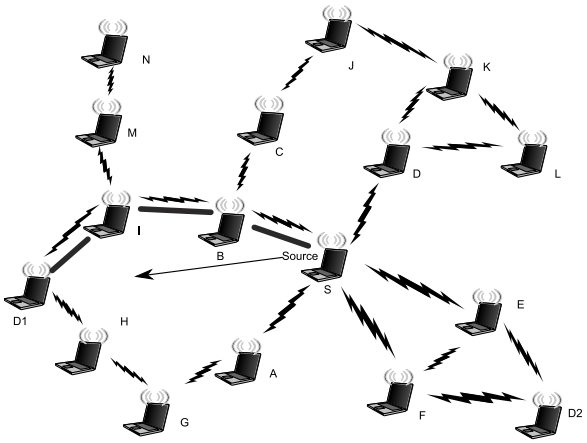

Figure 2.6: Virtual force technique - an illustration

in the graph $G(V, E)$ such that $\forall v \notin C$, $C \subset V$, $v \in N(u)$, where u belongs to C. Here, $N(u)$ is the set of neighbouring nodes of u. The dominator nodes are then used in the routing process. The problem of computing the MCDS is known to be NP-Complete [Amis *et al.* 2000]. All algorithms referred in this thesis for computing MCDS are only approximation algorithms.

The team of Shivakumar, Das and Bhargavan had suggested a set of routing algorithms in [Das & Bharghavan 1997], [Sivakumar *et al.* 1998], [Sivakumar & Bharghavan 1998] based on MCDS. [Wu 2002] suggested a routing algorithm that used a marking phase to compute the dominator nodes. One of the simplest algorithms for computing MCDS was suggested by [Alzoubi *et al.* 2002c] that provides an approximation factor of 8.

2.2.6.1 A simple Maximal Independent Set(MIS) construction algorithm

In this subsection, we discuss the MIS construction algorithm suggested by [Alzoubi *et al.* 2002a]. For the purpose of construction, they use the property of MIS nodes: any two MIS nodes are separated by at least two hops. A two-phase distributed algorithm was used by Alzoubi, et al. The first phase constructed the MIS S. The second phase was used by the non-MIS nodes to identify the MIS nodes that are at most two hops away from them, and to broadcast this information to the other nodes. They defined the term *dominator* to refer to a node belonging to the set S, and the term *dominatee* to refer to a node that was not in S. It is easy to note that each dominator node can now be aware of the path to its MIS neighbours, i.e. the subset of MIS nodes that are at most three

hops away from the current node. The dominatee node that is in the path between two dominator nodes is referred to as a *connector* node.

The algorithm suggested by them is reproduced below for clarity. [Alzoubi *et al.* 2002a]

All nodes are candidate nodes in the beginning. A candidate assigns itself as a dominator node, updates its local variables, and broadcasts the DOMINATOR message. When a node recieves a DOMINATOR message, a node which cannot be a dominator marks itself as a dominatee and broadcasts the DOMINATEE message after updating its local variables. If the node has no more candidate neighbours, it broadcasts the list of dominators one hop away (neighbors) in LIST1 message. If all its neighbours are dominators, it also broadcasts a LIST2 message containing the list of dominators two hops away (even if it is empty). When a node receives a DOMINATEE message, a candidate node verifies whether the sender of the message was its last neighbour having an ID lesser than itself. If so, the node marks itself as a dominator node and broadcasts DOMINATOR message to its neighbours. When a dominatee node receives the DOMINATEE message and there are no more neighbours which are candidate nodes, it broadcasts the LIST1 message.

When a dominatee or a candidate node recieves a LIST1 message, it checks its own lists of dominators that are one hop and two hops away. If any dominator node in the LIST1 message is not in its lists, then it marks the sender of the message as the connector node to that dominator node in its internal list. If the dominatee/connector node has received LIST1 messages from all its neighbours that are not dominators, it broadcasts the LIST2 message. When the node receiving the LIST1 message is a dominator, for each dominator in LIST1 that has a higher ID than itself and that dominator node is not already marked as reachable in two hops, the node makes an entry indicating that the sender of the message can act as a connector node to that dominator node. The node behaves in a similar manner when it receives a LIST2 message, with the only addition that it would maintain one extra neighbour in its corresponding list. If a dominator node has received LIST1 and LIST2 messages from all the relevant nodes, it broadcasts the LIST3 message. When a node (which is either a dominatee or a connector node) receives the LIST3 message, it checks whether it appears in the list of IDs. If so, it marks itself as a connector node.It then updates its internal table to reflect the dominator nodes for which it is a connector.It then broadcasts CONNECTOR1 message that contains the pair of dominators for whom

it acts as a connector node along with the list of other connector nodes that form the path between the two connectors.

When a node that is not a dominator receives a CONNECTOR1 message, it verifies whether it appears in the list of nodes in the message. If so, it updates its internal tables to reflect the fact that it is a connector between the corresponding pair of dominators and also stores the path between these two dominators. Then it broadcasts CONNECTOR2 message. All nodes that receive the CONNECTOR1 and CONNECTOR2 messages to update its internal list of nodes that are connectors. At this stage, all the nodes are either a dominator, a dominatee or a connector.

Figure 2.7 shows a sample network for MIS construction. Figure 2.8 illustrates the same network after the construction algorithm discussed above has completed. The blue circles denote the MIS nodes, while the rest of nodes are non-MIS nodes.

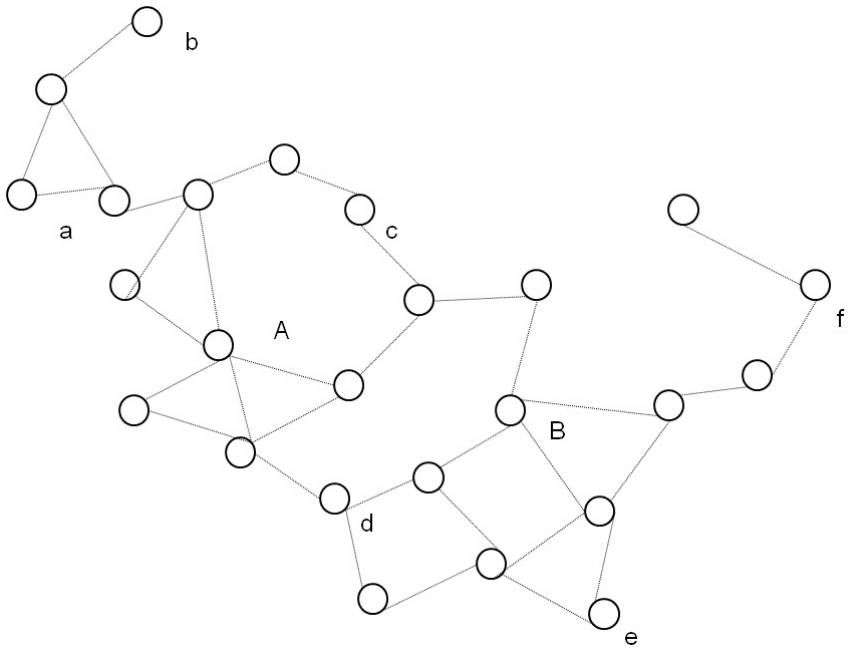

Figure 2.7: Maximal Independent Set construction(1) - Initial Network

This algorithm has time and message complexity of $O(n)$, where n is the number of nodes in the network. We have used the basic structure of this MIS creation algorithm in our work described in Chapter 4.

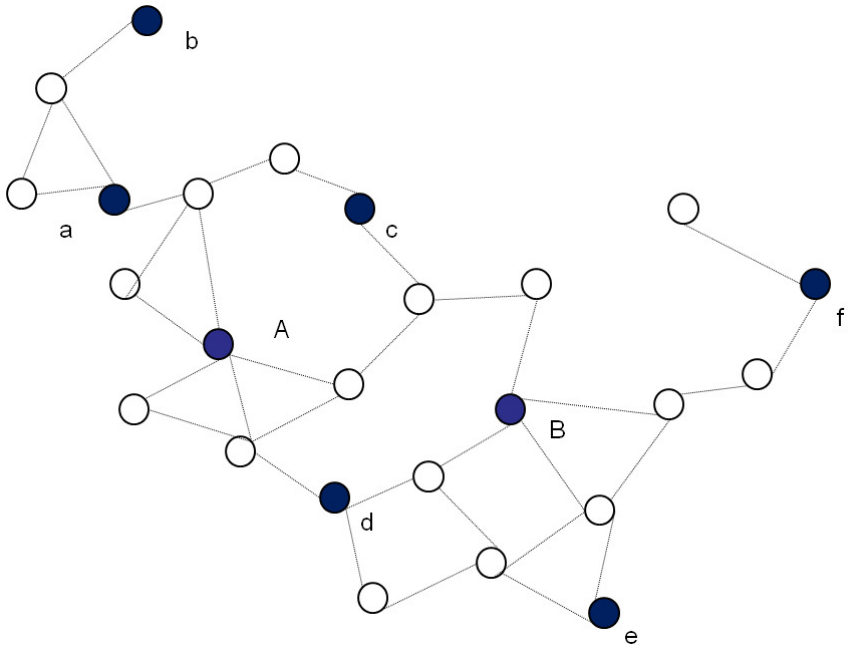

Figure 2.8: Maximal Independent Set construction(2) - After construction

2.2.6.2 A Maximal Independent Set(MIS) algorithm over two-hop neighbourhood

[Calinescu 2003] suggested a routing algorithm using two-hop neighborhood, instead of the usual one hop neighbourhood, for MANETs. In their algorithm, they first construct the MIS for the given network. From this information, they found out the 2-hop neighbourhood information for each node. The nodes formed the virtual backbone used by them for their route computations. Figure 2.9 illustrates the 2-hop construction suggested by them. It is interesting to observe that in the same network, by constructing MIS for the nodes belonging to the backbone alone, the number of nodes drop to 2, as is depicted with the red nodes in the graph.This is reflective of the results of [Theorems 1 and 2] proved by them.

2.2.7 Well-Separated Pair Decomposition

Well-Separated Pair Decomposition (WSPD) is a well known phenomenon used to deal with reaching long distance points in t-optimal paths. [Gao 2004] When the source and the destination points are far apart, it is sufficient to get the path from the centre of the region containing source to the centre of the region containing destination. The finer details can either be taken care by a recursive usage of WSPD, or by a direct route from the centre

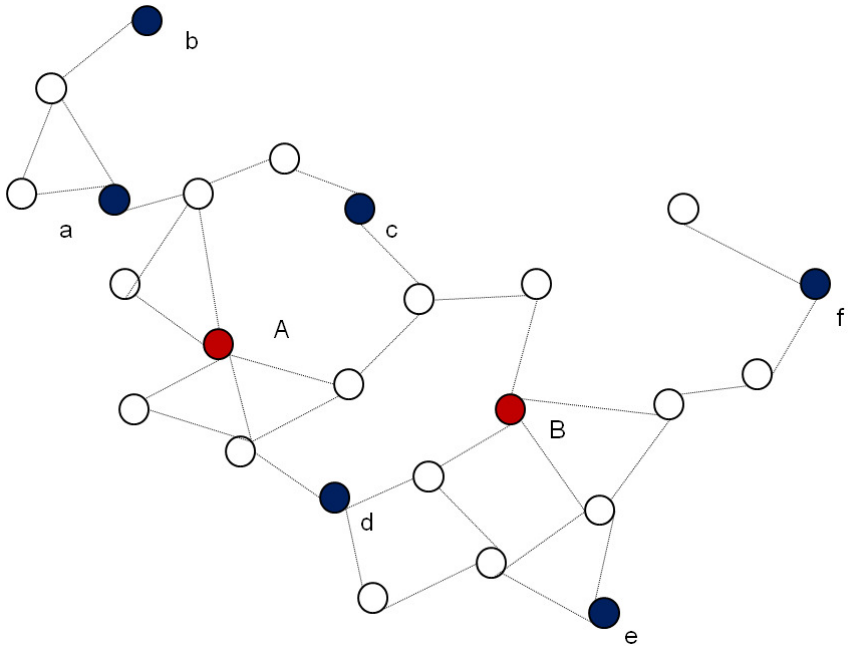

Figure 2.9: 2-hop Maximal Independent Set construction - [Calinescu 2003]

of the region having the actual node of interest. The concept of WSPD guarantees that the path derived using the method is going to be very close to the shortest path between source and destination, if the region containing the source and the region containing the destination are comparatively farther apart.[Callahan & Kosaraju 1995]. We use the notion of WSPD is used in an algorithm presented in Chapter 4. For better understanding of our approach later, we describe the notion of WSPD below.

Suppose a given graph G is divided into point sets, A and B, then to find out the distance between a and b, where $a \in A$ and $b \in B$, it is enough to find the distance between a' and b', where a' is the centre of the point set A and b' is the centre of the point set B, and then the individual distances of (a, a') and (b,b'). Even the points a and a' may be parts of two point sets A1 and A2 respectively and the distance may further be approximated in a similar way. The major advantage of this method is that once all the pair distances are pre-computed, then time needed for distance query is O(1). Figure 2.10 illustrates the concept of WSPD.

[Gao & Zhang 2005] described the use of WSPD in unit disk graphs. They pre-computed the distance of all the pairs. They came up with the result that for any set S of n points in the plane and any $c \geq 1$, a c-WSPD can be constructed under the unit

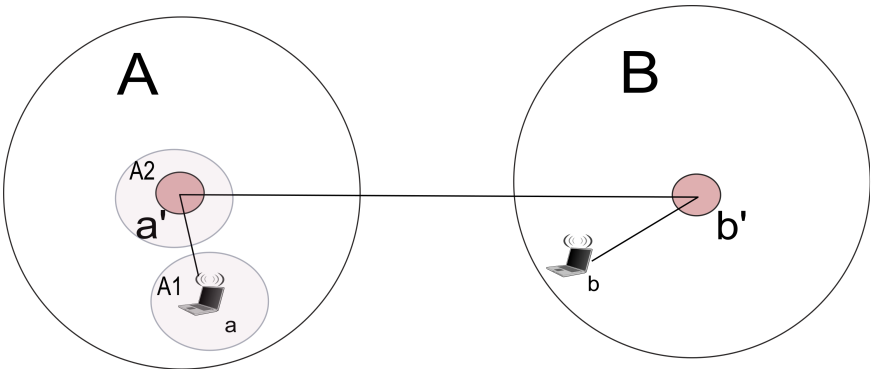

Figure 2.10: Well-seperated pair decomposition - an illustration

disk graph metric with $O(c^4 \dot{n} \log n)$ pairs and $O(c^4 \dot{n} \log n)$ time.

[Funke *et al.* 2004] performed experimental analysis of the effects of various computational geometry based data structures for querying an energy efficient path in a multi-hop wireless network. Though their approach had assumed that the positions and hence the paths can be precomputed, their analysis throws light into the efficiency of the WSPD method over all other method in terms of time. In their experiments, it was found that the maximum relative error peaked at about 50%, and the average query time was faster by atleast 100 times compared to the closest competing technique employed by them.

2.3 Quality-of-Service(QoS) aware routing protocols in MANETs

Quality of Service (QoS) is provided in wired networks with the help of over-provisioning or network traffic engineering. In over-provisioning, the network provides more resources to accommodate for any QoS requirements that may arise.[Guimaraes *et al.* 2004] In network traffic engineering, the traffic is classified on the basis of established rules and handled as per the requirements. Integrated Services (IntServ) and Differentiated Services (DiffServ) are two proposals for handling QoS[Peterson 2011]. In IntServ, users request for specific QoS parameters, and a reservation-based approach is used to provide for the required guarantees. DiffServ approach offers a set of differentiated classes of services and the user may chose from them. In mobile environments, over-provisioning cannot be done, as the network is already having severe restrictions imposed by the underlying dynamic topology and resource constraints. IntServ requires identification and reservation of QoS parameters, which means that global state needs to be collected and control

messages need to be propagated to provide specific QoS requirements. This approach will result in significant control overhead. As DiffServ uses a reservation-less approach and has low overhead, such an approach may be more suitable for MANETs.

In the remainder of this section, we discuss some reservation-less approaches in section 2.3.1 and some reservation-oriented approaches in section 2.3.2. We concentrate ourselves on energy as QoS parameter and discuss a few energy-aware routing protocols in section 2.3.3.

2.3.1 Reservation-less approach for QoS aware routing in MANETs

One of the simplest ways to handle QoS aware routing in MANETs is to use a load-balancing approach. In this approach, the mobile node estimates the available bandwidth. This estimation is easy to perform by computing the number of packets transmitted over a period of time. From the quantum of data communicated over a unit period of time, the mobile node can estimate the current available bandwidth. This information is shared with other mobile nodes. [Kazantzidis *et al.* 2001] suggested an improvement of AODV to handle QoS requirements using this technique. [Badis *et al.* 2003] suggested a modification of the OLSR [Jacquet *et al.* 2001] protocol to provide QoS. Once the bandwidth related information is shared with the neighbors, the routing algorithm can just add residual bandwidth as an extra parameter while making the routing decisions.

If there are both high-priority and low priority traffic to be served, another approach named as courtesy piggybacking can be used [Liu *et al.* 2004]. In this approach, the high-priority packets are checked to find out whether there is some unused/underused frames. Whenever such a frame is found, data from the low priority traffic is added to the remainder of the packet. Thus the low-priority data is transmitted without compromising on the high priority traffic flows.

[Ahn *et al.* 2002] suggested a different approach to provide QoS in their SWAN project. They measured the current state of the network by using active feedback. This allowed the algorithm to identify the nodes where some form of admission control is to be used. Bandwidth is measured by counting the number of packets passing through the current node. MAC delays are measured to identify whether rate control mechanisms need to be

introduced. This approach is sufficient to provide soft QoS guarantees.

2.3.2 Reservation-oriented QoS aware routing algorithms for MANETs

One of the simplest ways to perform QoS reservation in routing algorithms for MANETs is to piggyback reservation requests along with the route request packets. An example for such an implementation is the INSIGNIA protocol proposed by [Lee *et al.* 2000].

Flexible QoS Model for MANETs(FQMM) suggested a hybrid approach to provide per-flow and per-class granularities as provided by IntServ and DiffServ approaches [Xiao *et al.* 2000]. Traffic is classified at the source node to identify whether a per-flow or a per-class provisioning needs to be done. High priority traffic flows are provided the per-flow granularity. Other traffic is provided a per-class provisioning. The part for QoS provisioning is done in each of the intermediate nodes till the destination node is reached.

2.3.3 Energy aware routing protocols in MANETs

Apart from the QoS parameters like bandwidth, delay and jitter in MANET environment, energy related parameters like per-flow energy cost, network life time and residual energy are also important to the users. In this sub section, we shall discuss a few algorithms that provide energy aware routing in MANETs.

[Singh *et al.* 1998] proposed protocols that established routes ensuring all nodes in the route equally drain out their battery power reducing the energy consumption at a node, and hence increasing the network lifetime. But their approach has a significant communication delay as every node has to have the entire topology knowledge. [Stojmenovic & Lin 2001a] described several localized routing algorithms (power, cost, and power-cost) that tried to minimize the total energy per packet and lifetime of each node. Their algorithms depended on location information of all nodes affected while routing, which they collected using GPS receivers integrated with each mobile node. Power efficient algorithms selected well-positioned neighboring nodes in forwarding the message, while the cost efficient algorithms favored nodes with more remaining battery power. [Lewin-Eytan *et al.* 2007] solved unicast and multicast routing problems in wireless networks by building a Steiner tree spanning the terminals with maximum lifetime. [Tarique

et al. 2005] proposed an approach to improve upon DSR [Johnson & Maltz 1996] by using load sharing for making routing decisions. The link by link power adjustment was performed on a per packet basis using the transmit power control approach. [Srinivasan *et al.* 2004] formulated the problem of energy efficiency as one of maximizing the sum of the source utilities subject to a required constraint on the network lifetime. They defined the network lifetime as the time until the first node in the network drains its battery completely. They proposed a flow control algorithm to solve this problem that provides optimal source rates. Rodoplu *et al*. proposed a minimum energy routing algorithm which chooses minimum energy consumption per packet as the metric for optimization. They performed local optimization on the metric from knowing the neighboring nodes' power values thereby attaining a globally optimized minimum energy solution.

Merely considering current energy related parameters for improving certain metrics may seem sufficient if we assume that the mobile node is only going to dissipate power once it is in the field. However, it is impossible to use a mobile network where the power is not replenished from time to time. This raises serious practical difficulties. The underlying assumption in the algorithms discussed so far was that whenever a mobile node loses too much power, then the node will be taken out of the network, recharged and then reconnected to the network. There is, however, a different way that nodes can be kept in the field. An energy harvesting device can be used as a simple alternative to allow a node to continue functioning in the field.

For discussing the design of routing algorithms in the presence of energy-harvesting scenarios, we can consider a special case of a MANET that uses sensor nodes instead of mobile nodes. An energy harvesting device, like solar cell, converts solar energy into electricity to be supplied to a sensor node. However, such a device cannot deterministically supply power to the system, due to its inherent limitations [Kansal *et al.* 2007]. Also, an energy harvesting device can produce energy only at a limited rate [Niyato *et al.* 2007]. In traditional power management schemes used in battery-operated nodes, the emphasis is usually on optimizing the residual energy of each node. When it comes to energy harvesting scenario, we need to optimize the rate at which the energy shall be utilized by the nodes, instead of trying to optimize the residual energy alone. Ideally, such energy harvesting mechanisms must lead to an energy-neutral system [Kansal *et al.* 2007]. Under

these circumstances, appropriate energy saving mechanisms is more relevant to reduce energy consumption at a sensor node, and by extension, the effective drainage of residual energy in the node. These energy saving mechanisms can be deployed at different layers of protocol stack as suggested by [Niyato *et al.* 2007]. Energy saving protocols at network layer considers energy constraints, path length, survivability, etc. for sensor centric routing [Kannan & Iyengar 2004]. In the MAC layer, energy saving techniques like limiting the idle listening time, avoiding overhearing, avoiding packet collisions, etc. are used [Ye *et al.* 2004]. Another mechanism used to save energy in the MAC layer is the use of energy efficient packet scheduling [Yu & Prasanna 2003].

2.4 Multicast routing protocols

Multicast communication forms one of the key factors for group-oriented communication scenarios, like collaborative computing.[Stojmenovic 2003] The two popular approaches used for multicast routing in wired networks are core-based tree approach and shortest path multi-cast tree approach[Junhai *et al.* 2009]. In the shortest path multicast tree approach, each source node builds a tree to include the other nodes in the multicast tree. The number of trees to be constructed will become vast if the number of participant nodes is large. This approach suffers from high control overhead in the event of frequent link failures. In the core-based tree approach, only one tree is constructed for the multicast communication. When we are interested in multicast tree construction for MANETs, the characteristics of the underlying network pose serious challenges to the multicasting problem. Mobility of nodes causes frequent node/link failures. The topology of the underlying network will change when a node moves from one position to another. Typical mobile nodes have resource constraints, including residual energy and available bandwidth. While choosing the multicast algorithms, extra care needs to be taken to reduce the overhead of the algorithm/protocol. When it comes to multicasting in MANETs, one of the approaches that can be used is to flood the request to the node's neighbors. Flooding will result in a large set of packets to be generated during the tree construction phase. Another approach is to follow the lines of proactive unicast routing protocols. The paths to all the multicast destination nodes can be pre-computed and stored in the routing table.

Just as in unicast routing, the main challenge in this approach is to ensure that the global state of the network is captured accurately by each node before every multicast communication. Combinatorial stability has to be ensured in this approach; otherwise the stored multicast tree may not be able to deliver the message to all the hosts within the multicast group. A third alternative is to use an on-demand algorithm for creating the paths. A query phase can be used to explore the network, and a response phase may be used to establish the paths to the hosts belonging to the multicast group.

In the rest of this section, we discuss some of the earlier multicast routing algorithms in Section 2.4.1. We discuss the graph-based approach used for multicast routing in Section 2.4.2. Finally, we discuss a set of contemprary multicast routing algorithms in Section 2.4.3.

2.4.1 Some of the earlier Multicast routing algorithms

In this sub-section, we shall discuss some of the Multicast routing algorithms that were discussed in the past.

Wu and Tay proposed a shared-tree based routing protocol that used the underlying unicast routing protocol named Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) [Wu & Tay 1999]. Every participating node dynamically assigned an id-number for a multicast session. The multicast flow is determined by the ordering of the nodes based on their id-numbers. A special node termed as Sid node is used as the root of the shared tree. The protocol differentiates between the multicast nodes (termed as I-Nodes) and other nodes which may be a Steiner node (termed as U-Node).

Multicast Ad hoc On-demand Distance Vector(MAODV) protocol, which is an extension of the unicast AODV protocol, to generate the multicast tree whenever a multicast communication is to be initiated was proposed by [Royer & Perkins 1999]. This protocol relies on the correctness of the AODV protocol to prove that loop-free multicast paths can be generated.

Ad hoc Multicast Routing protocol (AMRoute)[Xie *et al.* 2002] exploits user-multicast trees to handle multicast communication. Unicast tunnels are used to neighbors on the user-multicast tree. This protocol relies on the underlying unicast routing protocol to deal

with node/link failures. The protocol assigns some nodes as logical cores. These nodes are responsible for managing the multicast group.

Garcia-Luna-Aceves and Madruga proposed a multicast protocol that constructs a group shared-tree to handle multicasts in [Garcia-Luna-Aceves & Madruga 1999] [Madruga & Garcia-Luna-Aceves 2005]. They use a shared multicast mesh, that is defined for each multicast group.

ODMRP [Yi *et al.* 2002] is a source-initiated multicast routing protocol which uses the concept of forward groups, a set of nodes responsible for forwarding multicast data on shortest paths between any member pairs, to build a forwarding mesh for each multicast group.

In CEDAR [Sivakumar *et al.* 1999], core groups are formed among the mobile nodes belonging to the multi-cast group. Each multicast group contains a core node that performs computations required for determining the routes to all the nodes in the group.

ReMHoc [Sobeih *et al.* 2004] is a distributed receiver-initiated NACK-based algorithm and it makes use of feedback suppression in order to avoid negative acknowledgement (NACK) and retransmission implosion.

Neighbor-Supporting Multicast Protocol (NSMP) [Lee & Kim 2000] is based on multicast meshes and designed to minimize data transmissions and control overhead in maintaining the meshes. NSMP also attempts to improve route efficiency by giving preference to forwarding nodes in establishing a route thereby reducing data packet transmissions and contention in a network.

2.4.2 Graph based approach in Multicast routing algorithms

From a graph-based approach, the multicast routing problem is equivalent to Steiner tree problem that was defined in Section 1.9.

[Gilbert & Pollak 1968] had put forth the criteria that the angle between out-going edges at a branch node for a Steiner node is 120°. A relatively minimal Steiner tree is a Steiner tree when this angle is greater than or equal to 120° for all internal nodes. In their paper, they have suggested an inductive property for constructing relatively minimal Steiner trees with the help of unit tension force from the group nodes. However, in a real-

world network, we do not have control over the location of various nodes in the network. Thus, we should ideally be choosing branching nodes such that the angle between the outgoing edges are exactly 120°, but still be able to handle cases where appropriate nodes are not available in those positions.

2.4.3 Contemporary approaches used in Multicast routing algorithms

The previous sub-section had a set of broadly discussed attempts for multicast routing in MANETs. In this sub-section, we discuss a set of the contemporary multi-cast routing algorithms.

Mauve, *et al.* suggested a multicast routing algorithm using geometric information in [Mauve *et al.* 2003]. Their greedy algorithm uses a heuristic dependent on the normalized number of next hop neighbors to determine whether a branching node in multicast tree has been reached. The parameter λ used in their algorithm determines how late the split of the multicast forwarding will take place, with $\lambda = 0$ indicating an early split and $\lambda \approx 1$ for a very late split.

Rahman *et al.* [Rahman & Gregory 2011b] divided the region around the current node into quadrants, and considered four closest nodes in each quadrant for determining the next hop in the multicast tree. Their contention was that a split in multicast packet will occur when the destination nodes belong to multiple quadrants around the current node. They later expanded their algorithm [Rahman & Gregory 2011a] by using an intelligent energy matrix to increase the average life of the nodes in the network. Their algorithm provides a very reasonable length of the multicast tree. However, we found that by statically fixing the quadrants, their algorithm was generating slightly longer multicast trees.

Liu, *et al.* suggested two routing algorithms, SWING [Liu & Wu 2006] and SWING+ [Liu & Wu 2009], for unicast routing in MANETs based on the notion of small world networks. These algorithms expanded the notion of virtual circuits as described by Fotopoulou, *et al.*[Fotopoulou-Prigipa & McDonald 2004] to include long virtual logical links to the surrounding neighborhood of the current node with the help of virtual force. By computing the repulsive force exerted on the neighbors of the current node, the next hop neighbor was chosen as the node with the maximum repulsive force from the source node

towards the destination node. They demonstrated that their approach works efficiently in a 3-D environment, as well. We extend the notion of virtual force suggested by Liu, *et al*. for multicast routing in this paper. Though in [Liu & Wu 2009], Liu, *et al*. suggest that their approach can be used for multicast routing, they did not actually suggest any algorithm or protocol towards the end. We have explored their approach further and developed multicast routing algorithms, which are presented in Chapter 4.

2.5 Distributed Mutual Exclusion

Depending on the manner in which mutual exclusion is guaranteed, the solutions to the DME problem can be classified into two broad categories[Murthy & Manoj 2004]: token-based algorithms and permission-based algorithms [Singhal & Manivannan 1997, Wu *et al.* 2008, Benchaïba *et al.* 2004, Ricart & Agrawala 1981, Erciyes 2004, Maekawa 1985]. In short, a token-based approach [Raymond 1989, Walter *et al.* 2001, Suzuki & Kasami 1985, Derhab & Badache 2008, Benchaïba *et al.* 2004, Chen & Welch 2002] relies on the possession of a token to enter the critical section(CS). The token-based algorithms may be further classified into the circulating token method and the requesting token method. In the first method, a token is passed among all the participating nodes in a logical sequence. A node may enter CS only when it receives the token. After it completes its CS, the node will forward the token to its successor in the logical structure irrespective of whether the successor is the next node that requested for entering CS or not. In the requesting token method, the onus of retrieving the token from the current holder falls on the node requesting to enter the CS. Figure 2.11 illustrates a simple token-based algorithm for a MANET containing seven participating nodes interested in entering the critical section that works on the basis of circulating token.

A permission-based algorithm relies on getting permission from all the participating nodes by explicitly asking for permission to enter the CS from each of the participating nodes.[Murthy & Manoj 2004] Figure 2.12 illustrates a simple permission-based algorithm for a MANET containing seven participating nodes interested in entering the CS. Unlike the previous approach, here the requesting node sends the request message to other interested nodes in the network. When the permission is explicitly granted by other nodes,

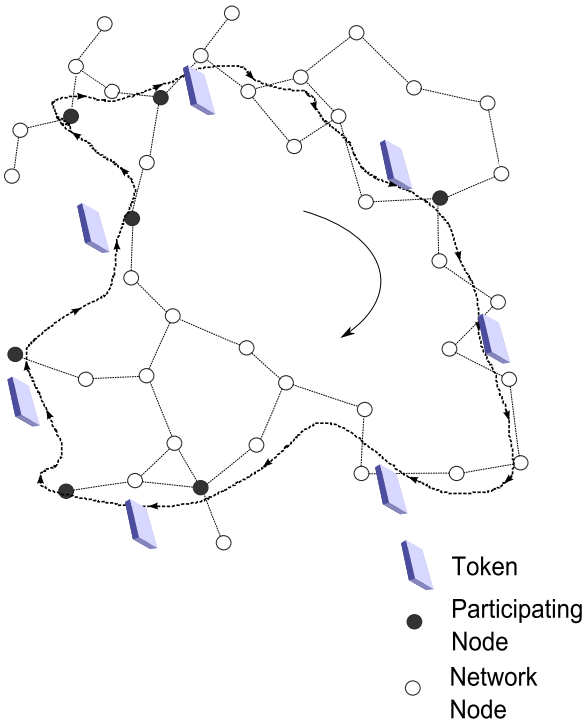

Figure 2.11: A sample token-based DME algortihm - Token circulated in the loop

the requesting node may enter the CS.

2.5.1 Classical token based algorithms

In this subsection, we discuss some of the token based algorithms available in literature. Token-based algorithms are broadly classified into two approaches on the basis of how tokens are handled as circulating token approach and requesting token approach. [Murthy & Manoj 2004]

Le Lann [Le Lann 1977] proposed an algorithm in which a unidirectional logical ring is first formed. The token circulates through this ring. When the token reaches a node, it can enter critical section(CS) or it forwards the token to the next node in the ring. The next node to forward the token is denoted as successor in each participating node. Each node contains a FIFO queue which contains the list of requesting hosts attached to it. A counter is used to ensure fairness property, so that a site enters CS only once in a round.

[Ricart & Agrawala 1983] suggested a token based algorithm that sends at most N messages to determine the token holder. The requesting node sends request to all N-1 sites for the token. If the current token holder has not received any request, it keeps

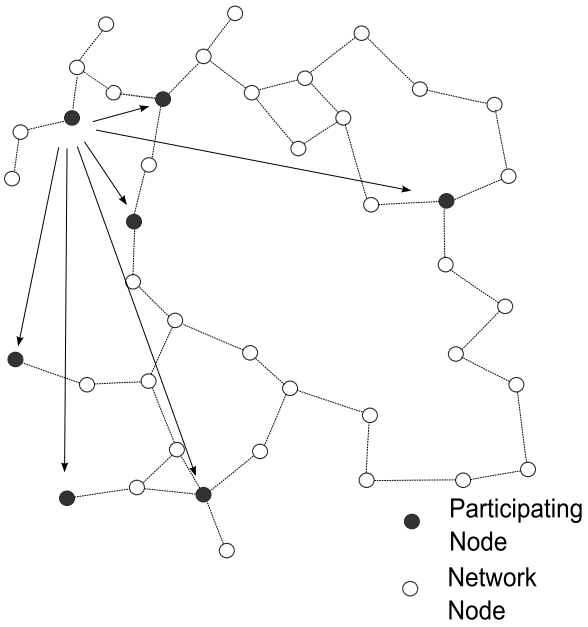

Figure 2.12: A sample permission-based DME algortihm - Arrows indicate request for permission

the token with itself, and marks the token as idle. If a site has requested, then token is circulated to the next node in a circular manner.

Suzuki-Kasami [Suzuki & Kasami 1985] proposed a token based algorithm in which the request queue is maintained as part of the token itself. The node number is used to order the requests, so that liveness property can be assured. The request queue in the token is updated by each node depending on the node's local queue.

Singhal [Singhal 1989] suggested a modification of the Suzuki-Kasami algorithm with the help of a heuristics. This algorithm sends at most N messages under heavy load. The reuest message is sent to only those participating nodes that currently hold the token, or are likely to hold the token. The knowledge of each node is piggybacked with the token.

Raymond [Raymond 1989] suggested a token based algorithm based on a logical tree with token-holder as the root of the tree. This approach requires at most O(log N) messages to be transmitted for a node to enter CS. Each node maintains a logical pointer, pointing to the root of the tree (token holder). The reverse path to the requesting node is used to send the token from the current token holder(root of the tree) to that node.

Raymond's algorithm was improved by [Chang *et al.* 1990] to tolerate node and link failures. This approach maintains multiple paths for the token to traverse. This approach also avoids cycles on the way back to the requesting node. This algorithm was extended by

[Dhamdhere & Kulkarni 1994] to handle some remaining cycles, and to make it resilient to k faults.

[Badrinath *et al.* 1994] suggested DME algorithms in cellular networks. The requests are forwarded to the base station. The base station forwards the request to the base station of the mobile host. In this algorithm, the computations are delegated to the static part of the network.

[Walter & Kini 1997] proposed a Directed Acyclic Graph (DAG) based algorithm for the token based DME problem. Token-oriented pointers maintain multiple paths to the token holder. Each node only keeps information about the immediate neighbours to whom the token needs to be forwarded. Under static environment, this algorithm reduces to the Raymond's algorithm [Raymond 1989]. Each node also keeps track of elevation information, indicating the number of hops to the token holder. This information needs to be updated in the event of link failures. This algorithm assumes that the token is never lost, and all communication links are bidirectional. There is a constraint on upper limit of mobility, so that combinatorial stability is maintained.

The nodes in the DAG are characterized by pointers using relative elevation to the token holder. The neighbours of a node are classified as set of nodes connected to incoming links and set of nodes connected to outgoing links. All requests are generated within the node and requests from incoming neighbours are forwarded exactly once. A request queue is used to maintain the order of requests. The backward path to reach the requesting node is also maintained. When a token is received by a node, it checks whether its own id is on top of requesting queue. If the node id is on top of the queue, it becomes the sink by modifying its elevation, and finally enters the CS. The token holder is always the node with the lowest elevation. A partial rearrangement of the DAG is required whenever the token holder changes.

A node exiting the CS sends the token to the node on top of its request queue. If no such node exists, then it keeps the token with itself. In the event of one or more link failures, a search message is generated by the token holder to fix the DAG. When a node receives a search message it forwards to all the neighbors in the incoming link. Once the next node is found for the node which suffered a link failure, the token can be forwarded using the updated link. The nodes adjacent to a new link exchange messages and update

the incoming and outgoing set of neighbors.

[Walter *et al.* 2001] suggested a token-based mutual exclusion algorithm that uses DAG with reverse links for its operation. Each node is aware of only the local topology. The token is delivered to the requesting node using the reverse path from the token holder. Just as in [40 of benchaiba], the token holder shall have the lowest height.

[Baldoni *et al.* 2002] combined circulating token and requesting token in the dynamic logical ring based DME algorithm. The requests are sent to the closest neighbor in the ring, thus allowing for node mobility. This algorithm attempts to reduce power consumption by reducing the average cost per CS. It also avoids transmitting any messages when no site is requesting for entering CS.

2.5.2 Classical Permission based Algorithms

One of the earlier permission-based algorithms that could be extended for use in MANET was suggested by [Ricart & Agrawala 1981]. This algorithm uses only two messages $REQUEST$ and $REPLY$ for handling mutual exclusion. Here, the $REPLY$ message is used to indicate that a site has been granted permission to enter CS. A $REPLY$ message from the site in CS was equivalent to sending release messages to all the nodes. Also, once a site S_i has received a $REPLY$ message from site S_j, site S_i can enter its CS any number of times without requesting permission from site S_j until S_i recieves a $REQUEST$ message from S_j. This algorithm has a message complexity of $2(N-1)$ messages, where N is the number of sites. Current versions of this algorithm are discussed in the next sub section.

[Maekawa 1985] suggested a distributed mutual exclusion algorithm that has a message complexity of $O(\sqrt{N})$. In this algorithm, the nodes participating in the distributed system are grouped together into quorums. The set of nodes in these quorums are designed in such a way that the intersection of any two sets is non-null. Thus a node needs to $REQUEST$ only to all the nodes in its quorum. Just communicating to these nodes directly is sufficient for finding out whether any other node is also requesting for CS. Each of the nodes to which $REQUEST$ is sent will verify that all nodes in the quorums it is participating are not requesting for CS. When it comes to MANETs, because of the characteristics of the underlying network, the additional overhead involved in the cre-

ation of quorums will weigh too much, compared to the benefits that can be reaped. The restrictions on creating the quorums typically limits the use of this algorithm in MANETs. As far as we know, there is no known algorithm that uses or mimics this approach in MANETs.

[Singhal & Manivannan 1997] observed that a site need not consult other sites that are not currently contending for CS. In fact, the number of sites who are currently participating in the CS(Φ) is going to be smaller than N in most cases. They introduced the "look-ahead" technique to further reduce the message complexity to $2(\Phi - 1)$. They used Dynamic Information Sets, comprising of $Info\_set$ and $Status\_set$, to keep track of sites that are currently involved in CS. In our approach, we had explored the use of Dynamic Information Sets further in MANETs.

2.5.3 Recent algorithms for DME

In this section, we discuss some of the contemporary permission-based DME algorithms in MANETs.

[Wu *et al.* 2008] extended Singhal, *et al.*'s algorithm to accommodate mobile nodes taking into consideration the dynamic behavior of a MANET. They introduced three additional messages namely, $DOZE$, $DISCONNECT$ and $RECONNECT$. $DOZE$ message is used to indicate that a mobile node is going to a sleep mode to save power. $DISCONNECT$ and $RECONNECT$ messages were used to allow a mobile node to move out of communication range and return to the range. To deal with fault-tolerance, they suggested the use of a timeout vector for $REQUEST$ messages, TO_{REQ}, which is maintained for the $REQUEST$ messages sent from the requesting site. If the TO^i_{REQ} expires, then the $REQUEST$ message is resent to the site S_i.

In [Bouillageut *et al.* 2009], a timer-free fault tolerant k-mutual exclusion algorithm is being introduced by Bouillageut, *et al.*. The method proposed by them could detect and tolerate f number of failures, and it integrates the resiliency within the k-mutual exclusion algorithm itself. As this algorithm uses a static mesh network, it cannot be directly used in MANETs.

[Erciyes 2004] had attempted in extending the Ricart-Agrawala algorithm for MANETs.

Instead of directly using the Ricart-Agrawala algorithm on all the nodes of the MANET, the RA_Ring algorithm divides nodes into a set of logical coordinators that form a ring of nodes. These coordinators then use a modified Ricart-Agrawala technique to deal with the DME problem. Here, the message complexity is $O(k+3)$, where k is the number of coordinators. Synchronization delay varies from $2T$ to $(k+1)T$, where T is the communication delay. This appears to be large when the number of clusters to be formed (k) becomes large. Erciyes, *et al.* [Erciyes & Dagdeviren 2012] have used a weighted partitioning scheme described in [Dagdeviren *et al.* 2005] for creating clusters. However, the algorithm in [Dagdeviren *et al.* 2005] expects the number of partitions to be created as a parameter, which is not tenable in practical applications. In the process of trying to extend the Ricart-Agrawala algorithm with the help of clustering, unfortunately, variants of the $RELEASE$ message had to be brought back both in intra-cluster and inter-cluster communication.

[Gupta *et al.* 2012] suggested a cluster-based DME algorithm using a consensus based voting scheme. The number of votes for a cluster-head depends on the number of nodes within its cluster. The cluster heads coordinate among one another by using a version of representative voting. Each cluster head keeps track of whether it has allowed a node to enter CS or not. When a $REQUEST$ is received, the cluster-head checks whether it has sufficient votes to let a node enter CS. If it doesn't have sufficient votes, then it asks for the votes of other cluster-heads. The cluster-heads maintain a $REQUEST$ queue to keep track of the oldest $REQUEST$ received by it to ensure fairness. This algorithm uses two release messages, as is used in [Erciyes & Dagdeviren 2012].

2.6 Conclusion

In this chapter, we discussed related work for the various approaches for routing and DME in MANETs that are being explored in this thesis. We discussed the routing problem, with the classical approaches, in the beginning. We expanded our discussion into one specific approach, namely the geography-based approach for routing. Then we discussed the QoS aware routing algorithms. The general approaches in multicast routing were explored next. We proceeded then to discuss the distributed mutual exclusion algorithm, with the

various approaches.

When we analyzed the energy-aware routing algorithms, we found that choosing a good metric is crucial to ensure that QoS parameters like network life time can be achieved. In terms of unicast routing, route maintenance is another aspect where the choice of an appropriate metric along with awareness of the network is crucial. In the energy-aware routing techniques mentioned for ad hoc networks in general, the special case of energy harvesting needs to be considered as well. We introduce three algorithms that use energy as a QoS parameter in unicast routing in Chapter 3. In terms of multicasting, we observed that the virtual force technique has not been applied for multicast routing in literature. That gave us opportunity to explore the use of virtual force for multicast routing in MANETs. We describe three virtual force based multicast routing approaches in Chapter 4. Permission-based distributed mutual exclusion (DME) algorithms had seen a slight resurgence in the recent past. However, we observed that not enough work has been carried out to make the algorithms scalable. We found that arbitration as a notion is not explored in solving the DME problem. Chapter 5 discusses the two DME algorithms explored by us.

Chapter 3

Energy Efficiency in Mobile Ad-hoc Networks

Energy forms one of the key Quality-of-Service (QoS) parameters to be considered in the routing problem for a Mobile Ad-hoc Network(MANET). A mobile node expends energy while performing communications, idling, and doing relevant computations. Use of a suitable energy metric is thus a necessity in mobile environments. Additionally, we need a stable mechanism to ensure that a routing algorithm can still provide energy-efficient routes in the presence of node mobility. In short, route maintenance in a routing algorithm should consider energy consumption while re-establishing routes in the wireless scenario. While energy consumption is a major driving force during design of routing algorithm, another key aspect to consider is to observe how energy is replenished in these wireless devices. Energy budgeting involves observing the manner in which the wireless devices are charged, and the avenues of energy consumption.

In this chapter, we first propose an energy-aware unicast routing protocol for MANETs that proposes a new energy metric in Section 3.1. We then propose an energy-aware routing algorithm that uses Bayesian approach for route discovery and maintenance in Section 3.2. In Section 3.3, we propose a generalized energy consumption model for wireless networks while using hierarchical routing. We conclude the chapter in Section 3.4.

3.1 Energy-Aware Routing in Mobile Ad-Hoc Networks

In this section, we explore an energy metric in a MANET that can improve key energy parameters like average residual energy. We have used a simple proactive routing protocol for looking at the effectiveness of this metric. Section 3.1.1 introduces the background and motivation for this algorithm. The assumptions used are presented in 3.1.2. The basic mechanism is presented in Section 3.1.3, followed by the pseudocode used in Section 3.1.4. The simulation results for this algorithm is presented in Section 3.1.5. Section 3.1.6 provides concluding remarks for this algorithm.

3.1.1 Background and Motivation

Mobile nodes that form part of a MANET typically are battery operated. Residual energy in the device becomes a key constraint due to the device's operational constraints.[Singh *et al.* 1998] There is a direct correlation between the number of hops in the route to destination and power consumed for transmitting the packet. While making routing decisions, the routing algorithm must take into consideration power constraints. Typical power constraints include average residual energy, minimum energy consumption per packet, maximum network life time, minimum variance in the node power levels, etc. A simple approach for designing power aware routing algorithms is to model the MANET as a disk graph and assign weights to the edges depending on energy factors. The task of finding energy aware path then reduces to finding the path with the least cost, where cost is modelled in terms of battery power of a node. We had presented most of the background for this algorithm in Section 2.3.3.

Now-a-days devices come with dual channel capability, allowing separation of the data and the control channels. Control signals are high power and data signals are low power transmissions. Wake-up channels are used to wake-up a sleeping node. The control signals have to be transmitted at high power, so that all nodes in the neighbourhood can become aware of the state. However, the transmission power of data can be adjusted such that just enough power is used to reach the destination. This enables sender device to spend less energy and reduces overhearing by the neighbouring nodes which unneces-

sarily waste power on hearing the signals not meant for them.[Tarique *et al.* 2005] Thus, we can model the communication over control channel as Unit Disk Graphs(UDG), and the communication over data as k-disk graphs. We had described some of the related work for this algorithm in Section 2.3.3.

In this section, we use the two channel approach to come up with a proactive routing algorithm for energy efficiency. We discuss a novel cost metric formulation that is used to optimize key metrics in the network ensuring proper utilization of the network and prolonged life time of nodes.

3.1.2 Assumptions

We assume that the devices operating in the MANET have dual channels, i.e. data channel and control channel as has been done in [Singh *et al.* 1998]. Data channel is used for the transmission of data packets and the control channel is for medium access, i.e. for transmitting Request to Send (RTS) or Clear to send (CTS) packets, and for wake-up calls in the end-to-end path. The devices are assumed to be aware of the remaining battery power in them. This is a fair assumption as most of the modern wireless devices, like laptops, have this information. We also assume that we will not be having a high mobility scenario, and therefore it is reasonable to use a proactive routing protocol. The down power for all nodes is assumed to be the same for the sake of simplicity. We can incorporate the notion of separate down powers by slightly modifying the algorithm.

3.1.3 Basic Mechanism

When a MANET node joins the network, its data channel is switched off. This is to save battery resource. Only the control channel is kept active and ready to transmit. The device is now in partial active state. At the beginning, all the nodes have their data channels closed. When a node needs to send some data packet, it gives a wake up call over its control channel. As it is a high power signal, the receiving nodes simply forward it to their neighbours. Effectively, this is a broadcast communication using multiple hops. The

nodes receiving this wake up signal not only forward it to their neighbours but also open their data channel. In this state, the device is said to be in fully active mode. Now the initiating node will broadcast GREQs, i.e. Graph Request Packets, over the data channel. The GREQs are used to generate the graph corresponding to MANET and to get the residual battery power of each node in the network. When a node receives a GREQ packet, it responds with a Graph Reply packet (GREP) with its identity and power values. We use Breadth first search (BFS) [Goodrich & Tamassia 2008] for collecting the global topology.

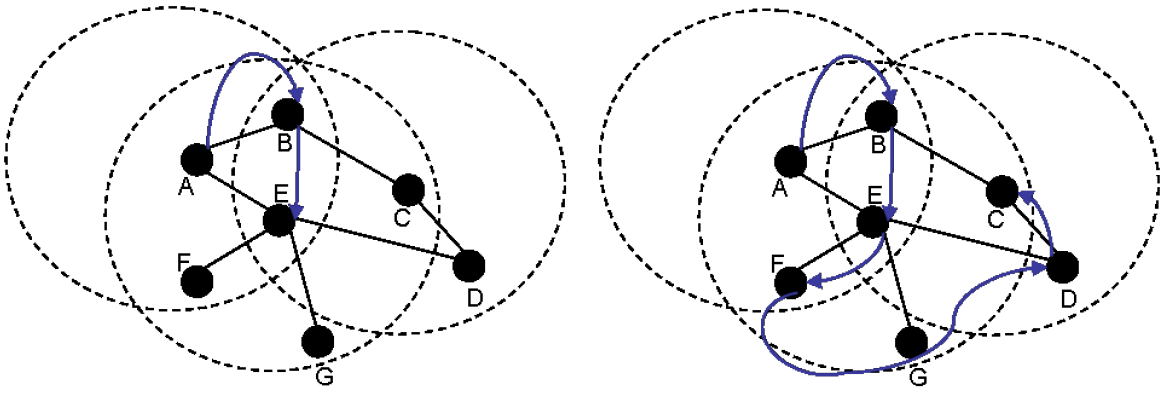

Figure 3.1: A sample MANET - BFS traversal

An example of BFS traversal is shown in Figure 3.1 for a sample MANET containing seven nodes. The BFS traversal on the MANET results in the node order visited as ABEFCDG. During the BFS traversal, the topology information and the energy levels associated with the nodes are collected.

A data structure, *String*, is used to store the relative location of the nodes (about topology) and also its residual battery power. After visiting every node in the topology, we wind up the *String* and generate the entire graph of the network starting at vertex A with the data structure just obtained using GREPs.

After obtaining the global topology with the residual battery power at every node using BFS, the cost of each node is calculated as shown in Equation 3.1. In this equation, P_{down} is tentatively set to 10% or 5% i.e. residual battery power when a device either hibernates or switches off, respectively.

$$A = \frac{100 - P_{down}}{S} \tag{3.1}$$
$$B = P_n - P_{down}$$
$$cost = S - \frac{B}{A}$$

At this point, a node is said to be isolated from the MANET or is dead. The variable P_n is the value obtained from the *GREP*s about each node. For getting proper values of the cost function, a scaling factor, denoted as S, $S = 10^k$ such that $10^{k-1} \leq N < 10^k$, is employed. i.e. S is taken as the next higher power of 10 of the number of nodes in the network. Each node calculates this cost before replying to GREQs. The cost values are updated on to an adjacency list for route computations. These values are used to check the results during our simulation.

After collecting the global topology and the cost associated with each node, Dijkstra's shortest path algorithm [Goodrich & Tamassia 2008] is used to find out the least cost path to the desired destination. Now we have the path to the destination and the information about the nodes participating in the data transfer. These nodes will be known as pool of active nodes. Now, we send a RFP(route finalize packet) with the end-to-end path information to tell the pool of active nodes not to shut their data channels on hearing the next *SLEEP* control signal. If the sender node receives ACK signal, it initiates *SLEEP* signal over the control channel. All nodes, except pool of active nodes, switch off their data channel keeping the control channel alive. However, after this the sender should acquire the medium using RTS/CTS signals over the control channels.

After generating the energy aware path, the source and destination can exchange data packets using the data channel in the usual way. To minimize the variance in the energy of nodes in the network, the notion of refresh intervals is used. By recalculating the path at refresh intervals, variance of the network can be reduced. If huge chunk of data are transmitted over the same nodes, the battery of these nodes will get used up faster. The

use of refresh interval ensures that same nodes don't get used quite often, thereby saving their battery resources.

3.1.4 Algorithm

When a source node wants to send some data to a destination node, it checks whether it has the current global topology. If it sees that its last topology is older than the current topology, it will start the Collect_Power routine, shown in the algorithm in Figure 3.2. This routine performs a BFS with the help of GREQ and GREP packets. First, the source node broadcasts the GREQ packet to its neighbours. When a node receives a GREQ packet, it responds with the identity of its parent node, its own identity and its power values back to the sender in the GREP packets. For ease of implementation, power is expressed in terms of percentage for calculation. For each of the nodes and links, the power values are updated.

The algorithm inherently calls our main metric calculation routine Calculate_Cost for each node value shown in the algorithm in Figure 3.3.. This routine calculates the cost metric based on the current residual power of a node and returns the cost. These two routines are called further when there are any updates in the topology.

When source node wants to compute the route to destination, it calls the Route_Generate routine, shown in the algorithm in Figure 3.4, to bring up the best path for sending the data packets. This routine is a modified Dijkstra's algorithm, taking into account the cost of the nodes and the link cost of the edge connecting the neighbouring nodes. Each node will initialize its own distance metric as 0. This algorithm is periodically called as per the refresh interval set. The route generated using this routine is used for sending the RFP packet to all the nodes that are part of the route. Each active node stores this route information in its route cache and routing table.

Figure 3.2: Procedure for Collecting Power

```
/* Collect_Power : retrieves the current residual power of
the nodes in the MANET                                          */
```

Input: source node s, the MANET modeled as graph $G(V, E)$, current residual power of nodes stored $crp_u, \forall u \in V$

Output: Graph G with discovery of connected edges, Residual power P_v for all nodes

```
/* N_u is the set of neighboring nodes of node u                  */
```

1 **Procedure Collect_Power(S, G, crp_u)**
2 **begin**
3 $C \leftarrow \emptyset$
4 $k \leftarrow 0$
5 $C \leftarrow C \cup \{s\}$
6 **while** $C_k \neq \emptyset$ **do**
7 $C_{k+1} \leftarrow \emptyset$
8 **foreach** $v \in C_k$ **do**
9 Broadcast $GREQ$ to N_v
10 **foreach** *GREP packet p received* **do**
11 $w \leftarrow p.getSource()$
/* $p.getSource()$ returns ip of source node from packet p */
12 Let edge e be the edge $\overrightarrow{vw}$
13 **if** $visited(e) =$ ***false*** **then**
14 **if** $visited(w) =$ ***false*** **then**
15 $link\_cost_e \leftarrow p.getLinkCost(v, w)$
/* $p.getLickCost(v, w)$ returns the link cost of the link from v to w piggy-backed in packet p */
16 $e.type \leftarrow$ discovery
17 $P_w \leftarrow Calculate\_Cost(crp_w)$
18 $C_{k+1} \leftarrow C_{k+1} \cup \{w\}$
19 $v.type \leftarrow$ Visited
20 $e.type \leftarrow$ cross edge
21 **end if**
22 **end if**
23 **end for**
24 **end for**
25 $k \leftarrow k + 1$
26 **end while**
27 **end**
28 **end**

Figure 3.3: Procedure for generating route

```
/* Calculate_Cost(crp_i) : calculates the cost metric of node i
based on the power values                                    */
```

Input: Current residual power of node i crp_i
Output: Cost value of node i

Calculate_Cost(crp_i)
begin
  $A \leftarrow \frac{100-P_{down}}{N}$
  $B \leftarrow crp_i - P_{down}$
  $cost_i \leftarrow N - \frac{B}{A}$
  return $cost_i$
end

Figure 3.4: Procedure for computing cost for node i

```
/* Route_Generate : generates the best path based on the cost
metric.                                                      */
```

Input: Graph G, cost values computed in $cost_u$
/* using a priority queue Q for all the vertices of G */
Output: a list which contains the shortest path
/* d_i is the costto node i initalized to ∞ for all nodes */
/* Q is a priority queue */

Route_Generate
begin
  $Q \leftarrow V$
  $d_{startnode} \leftarrow 0$
  while $Q \neq \emptyset$ **do**
    $u \leftarrow Q.getMin()$ **forall** v *s.t.* $v \in N_u$ *and* $v \in Q$ **do**
      $e \leftarrow edge(u, v)$
      if $d_u + link\_cost_e + cost_v < d_v$ **then**
        $d_v \leftarrow d_u + cost_v$
      end if
      $route.add(v)$
    end for
  end while
end

3.1.5 Simulation Results

In our work, we have compared the performance of our algorithm with two popular approaches for path selection; modification of DSDV protocol, which uses hop-based route determination mechanism, and the least cost approach as mentioned by Rodoplu and Meng in [Rodoplu & Meng 1999]. We compared key metrics like average residual energy

of the network, variance of energy over the network and average cost per packet for various data transfer. We also measured the network life time, defined as the time for one of the nodes to fail.

We have used an event-driven simulator in C-language based on the ns2 simulator code for our simulations. We generated nodes in Box configuration over a 200×200 region. We have taken initial node spacing as 10 units. We chose ten source-destination pairs in each generated network. For the simulations, we randomly generated the graph and adjusted the node positions so that we could get a connected graph with two or more paths between any two nodes. We have stressed on the condition of connectivity and the requirement for more than two paths since there will be no difference on power consumption in those paths for the various algorithms otherwise. The individual nodes may drift over time as the traffic is flowing through the network. The data generated were within 90% confidence interval. All values were within 85% confidence interval.

We have used constant bit rate (CBR) traffic generator for generating traffic from designated source and destination pairs. CBR traffic closely mimics multimedia communication. While choosing the source and destination pairs, scenarios where the source and destination are direct neighbours were avoided. The traffic rate for the CBR was randomly chosen for different execution cycles. For each sample execution, the node's power values were updated after transmission of every 1000 data packets, termed as epoch in our analyses.

In Figure 3.5, a sample mobile ad hoc network that was generated in our simulation run is shown. In Figure 3.6, the path for transferring data from node 1 to node 24 is shown. Initially, the best path is from $1 \rightarrow 2 \rightarrow 6 \rightarrow 9 \rightarrow 13 \rightarrow 14 \rightarrow 17 \rightarrow 20 \rightarrow 24$. But after some time, the residual energy of the nodes reduced. The proposed algorithm then selected automatically the alternate best path of $1 \rightarrow 5 \rightarrow 8 \rightarrow 11 \rightarrow 12 \rightarrow 16 \rightarrow 19 \rightarrow 23 \rightarrow 24$ as shown in Figure 3.7.

Figure 3.8 shows the Normalized energy levels after a sample run in our simulator

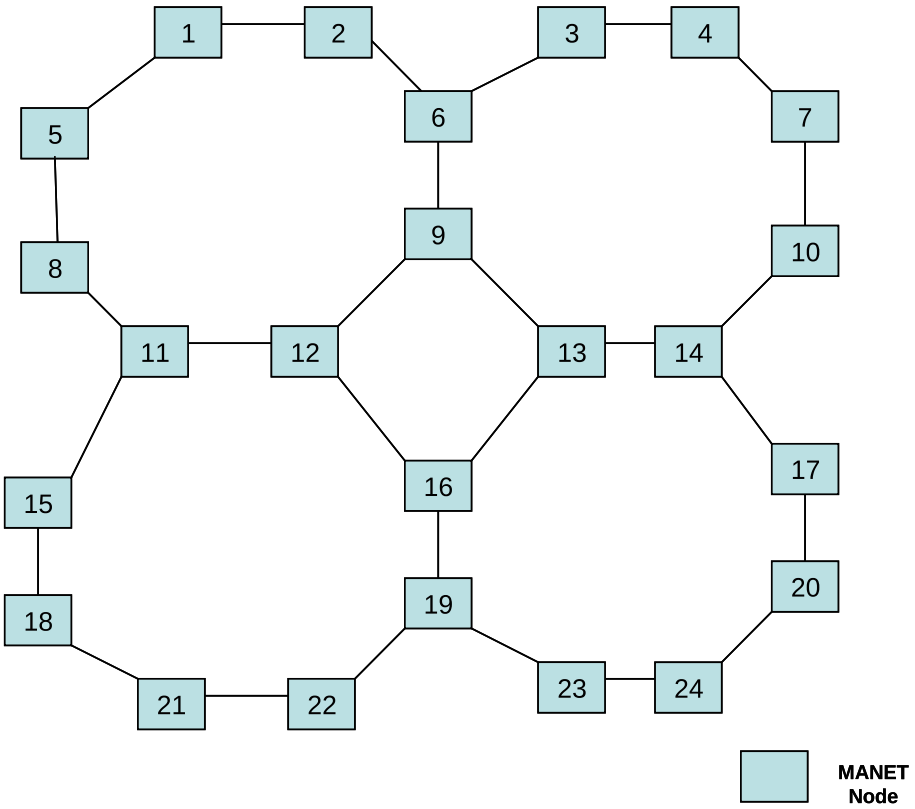

Figure 3.5: A sample Network - 24 nodes

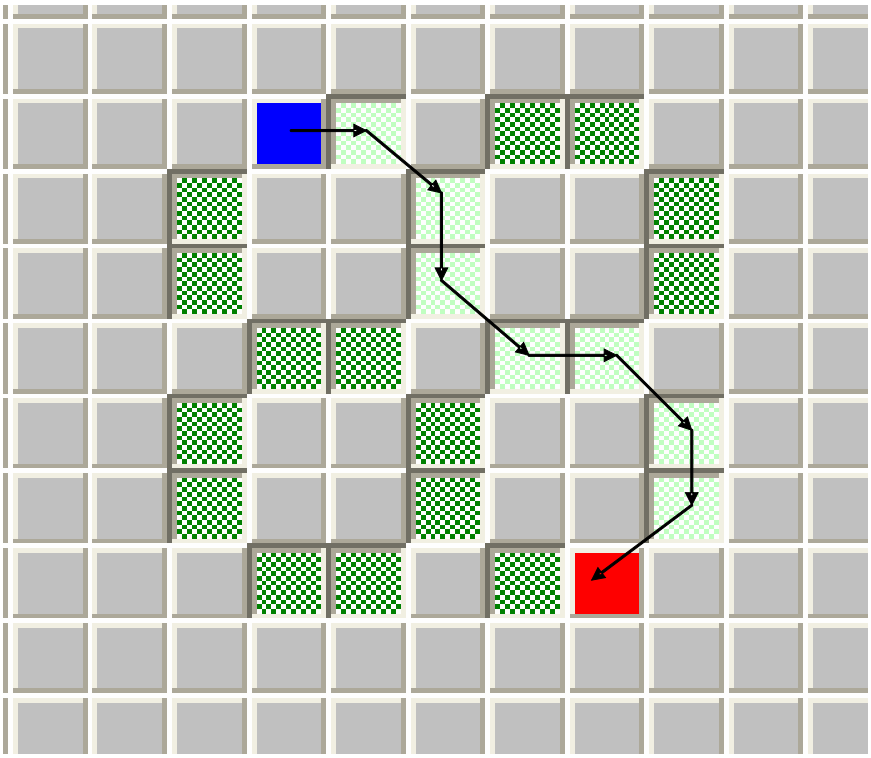

Figure 3.6: A sample run of EAR - Initial run of the network

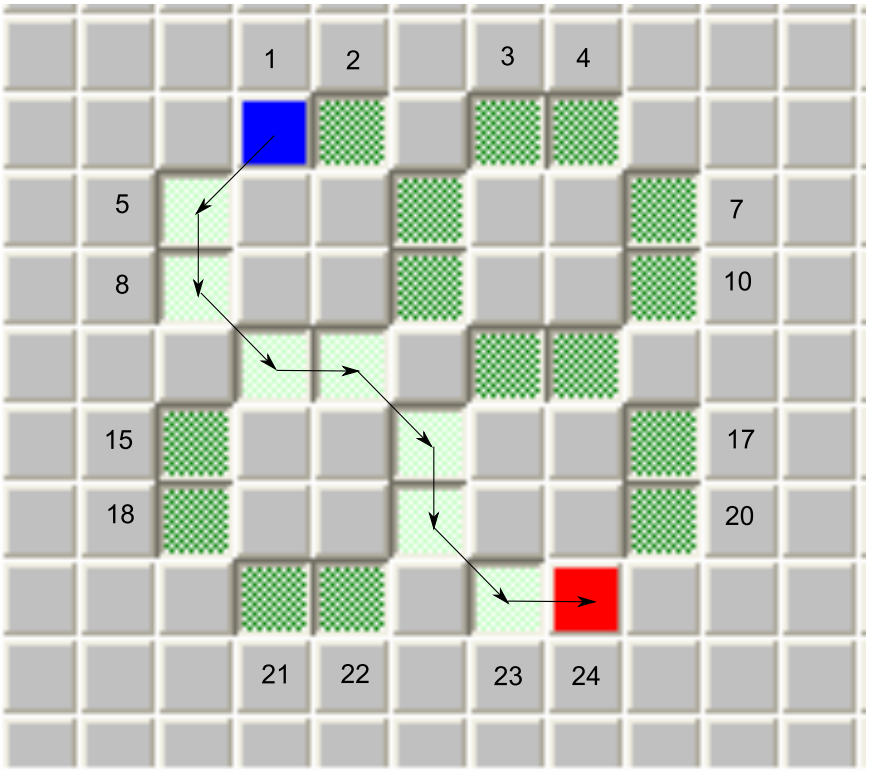

Figure 3.7: Another sample run of EAR - After refresh interval

using the above network. The residual energy values of the nodes are usually better as compared with the other protocols. In a few instances, where the cost metric results in a longer route, the corresponding nodes' power values are much lower than for other algorithms. This is expected behavior, as we are trying not to choose the shortest paths, but the routes with the best power characteristics. In the example topology, the nodes 17 and 20 were used more often in the set of sample runs over the topology, since that path was found to be better in terms of our cost metric for a set of data transfers.

After every epoch consisting of data transfer between a source-destination pair, the average Residual energy was evaluated as the average residual energy value of all nodes. The simulation results with respect to average residual energy are shown in Figure 3.9. As shown in the figure, the proposed algorithm results in better average residual energy for the network as a whole. For some rare sample source-destination pairs, there was not much variation in the average residual energy amongst shortest, least cost and our approach. All battery levels are expressed in terms of normalized form as derived from the cost function.

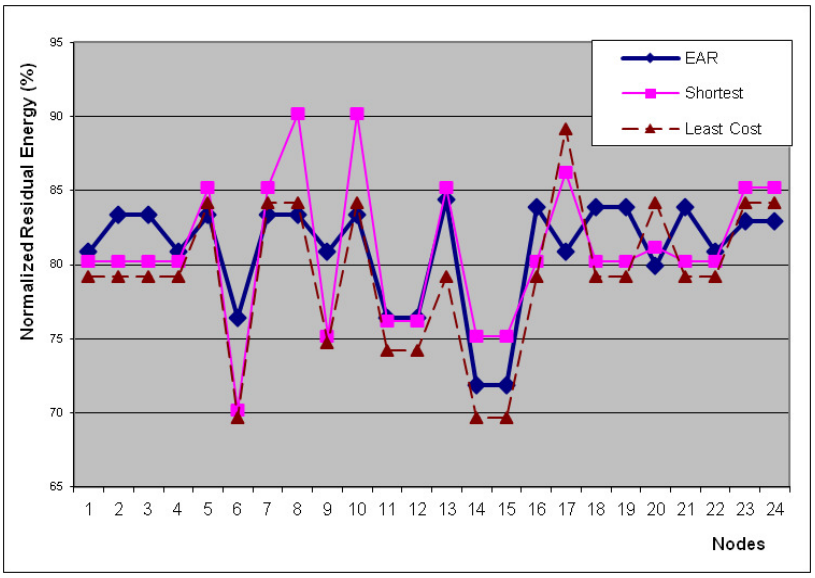

Figure 3.8: Normalized Residual Energy for each node - for EAR

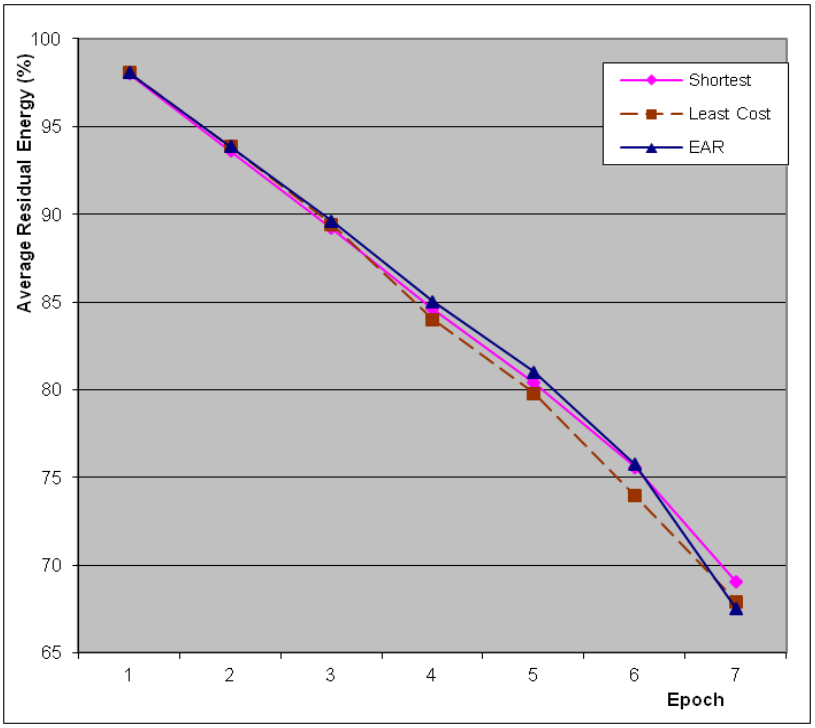

Figure 3.9: Average Residual Energy over time - for EAR

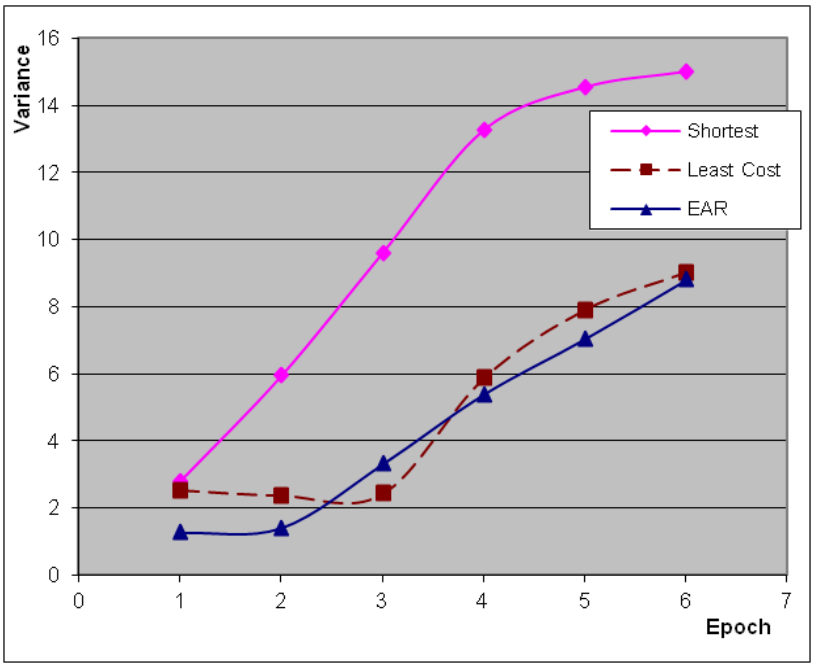

Figure 3.10: Variance of energy over time - for EAR

Figure 3.10 shows the variance of the residual energy of all the nodes in the network over the various epochs. Our algorithm has a better value for variance compared to the shortest path algorithm. Compared to the least cost approach, we observed that our approach has an improvement in terms of variance of residual power of nodes as seen from the graph. There were instances where the least cost approach was performing better than our algorithm as can be seen in epoch 3. This was observed just before the a change of path was computed. The variance of our metric improved in the next epoch. Our cost metric chooses the best available path in the refresh intervals, there by minimizing variance of the network.

The average cost of transmitting a packet over the network was computed from the simulation data. Instead of using the actual power values, normalized power values spent over the network for the entire transmission of each individual packet was used for uniformity. The results are shown in Figure 3.11. The average cost per packet is lesser in most instances as shown in the figure. When the power values of almost all the small paths were way too low and circuitous route were taken to reach the destination from

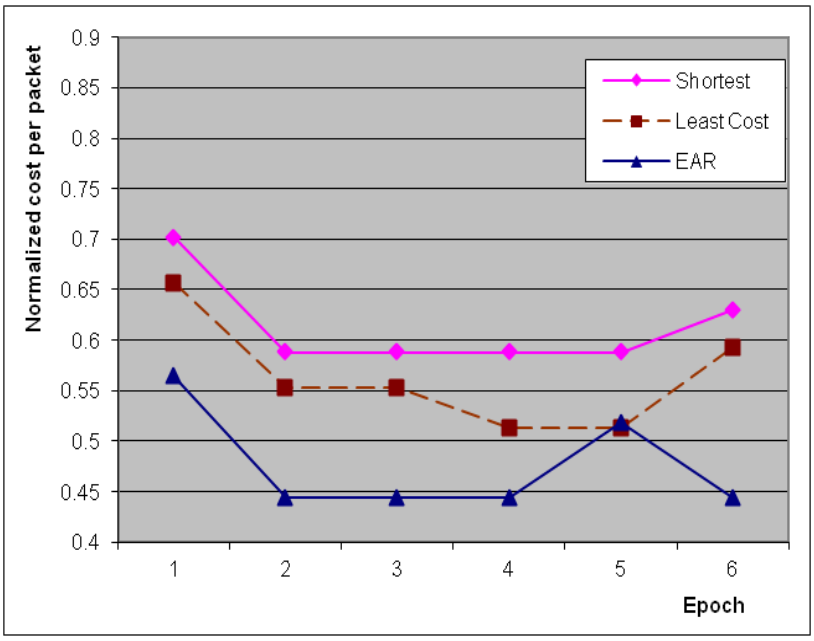

Figure 3.11: Average Normalized Cost per packet per hop - for another network

source, there were instances where the cost was slightly worse than the other algorithms as shown in the graph at epoch 5.

We also computed the network life time as defined in [Rodoplu & Meng 1999]. In one of our sample runs using bipartite graphs for input, we observed that the number of time units needed for the first node to fail in shortest path approach was 100 time units, in least cost approach 151 time units, while in our algorithm the first node to fail took 168 time units. We are getting a maximum of up to 68% improvement over shortest path approach, and 11% improvement, in the particular run. Averaging over all the sample runs, we observed 65% improvement over shortest path approach, and 10% improvement compared with least cost approach. The significant improvement with respect to shortest path approach stems from the fact that in shortest path approach, the same path will be used for communication between the source-destination pair. This led to faster depletion of residual energy unlike our approach where we change paths to redistribute the resource consumption.

3.1.6 Conclusion

From the simulation runs, we can conclude that use of our power-metric improves in finding the best path such that the difference in battery consumption between various nodes is reduced.This resulted in the improvement of network life time. The specific conclusions from the experiments are:

- As the network size increases, with more paths between the nodes, the cost savings of the network increases.
- The cost function introduced significantly improves the performance of the network.
- The overall cost savings are best in low mobility environments, where the nodes slightly drift from their locations only as the cost due to control packets is reduced.

In this section, we introduced a method for computing cost function for maximizing the average residual energy of the network, minimizing the variance of the power of the nodes in a MANET and reduce the cost for each packet to transmit data on the mobile wireless network. Our simulation results demonstrate that the cost metric used significantly improves these power-aware metrics. We have observed that our approach works fairly well when deployed in multimedia application environment, where a large amount of data has to be streamed between the source and destination nodes.

This algorithm cannot handle medium and high mobility, as can be expected in a proactive algorithm. Having looked into an efficient cost metric, we addressed the need for an energy-aware routing algorithm that can do route maintenance even in the presence of mobility. In the next section, we present such a routing algorithm for MANETs.

3.2 An Efficient On-Demand Routing Protocol(EORP) for MANETs based on Bayesian Approach

In this section, we present an Efficient On-demand Routing Protocol(EORP) that uses Bayesian approach for route discovery and maintenance in a MANET. We look at affinity

of a node towards the destination path to compute the best route to the destination. The motivation and relevant concepts are introduced in Section 3.2.1. The details of the algorithm along with relevant pseudocode is presented in Section 3.2.2. Performance results are presented in Section 3.2.3, and concluding remarks in Section 3.2.4.

3.2.1 Introduction and Motivation

The proposed Efficient On-demand Routing Protocol for MANET, using Bayesian Approach (EORP) is a novel way of finding route to the destination based on summing up the probabilities (affinity) of each node towards the particular destination, which is calculated using Bayes Theorem[Ahmed & Kanhere 2010][Luger 2001]. The protocol also makes sure that the data travels through shortest route only, thereby minimizing the time delays between sending and receiving of data packet. Another important feature of EORP is that it sends data through two disjoint paths, and the data is sent through both the paths alternatively. This reduces traffic through one path and avoids the loss of battery power that may occur as a result of standby mode of nodes in second path, as these nodes might only be waiting for the data packet to be received.

3.2.1.1 Basic Working of EORP

In EORP, each node maintains a history table, which contains the destination node's id and the value of the attributes used for calculating the affinity, along with the status whether a route reply (*RREP*) was received or not for every route request (*RREQ*) sent. This history is used in calculating own affinity while sending or rejecting a *RREQ*.

Each intermediate node upon receiving *RREQ* checks their routing table (*RT*) to find if the path to the destination is known or not. If known, a *RREP* is generated back to the node which generated the *RREQ* otherwise, node first compares the hop count in *RREQ* with last known hop count for same destination. If hop count in *RREQ* is higher; the *RREQ* is discarded (to ensure minimum hops route). Then node compares the stored

affinity value for that particular destination in their routing table (RT) with the affinity contained in $RREQ$. If affinity in RT is greater, $RREQ$ is rejected (to ensure only highest affinity requests are forwarded and stale routes are avoided) otherwise node will add its own affinity in the $RREQ$ and will broadcast the $RREQ$. Since the affinity of destination for itself will be 1(highest), upon receiving a $RREQ$, destination replies back by adding 1 to the affinity contained in the $RREQ$. Upon receiving $RREP$, intermediate nodes will again check their RT to see if route affinity in $RREP$ is higher than affinity in RT. If it is less, $RREP$ is rejected, otherwise it is sent to the node from which it received $RREQ$ with highest affinity. When a route reply is received by the source, it accepts best two $RREP$s based upon the affinity contained in them.

Upon route failure, intermediate node first tries to repair route locally. If route couldn't be repaired locally, route error ($RERR$) is generated and sent back to the previous node, which forwards it to the source node. The source upon receiving $RERR$ will start a fresh route discovery if it has lost both the paths to destination (ensuring reduction in control packet overhead). In case if other path still exists, it will only mark its backup path as INVALID, and will start sending data through only one path.

3.2.1.2 Calculating Affinity

Affinity Index (AI) is a probability of packet delivery based upon historical data. Through this we can find out how much likely it is for a particular node to transfer the data packet to the desired destination. It is calculated by using the Bayes Theorem [Ahmed & Kanhere 2010][Luger 2001], which is given in Equation 3.2. Here, C_i is the class indicating whether reply was received for the RREQ sent; and $X = \{X_1, X_2, \ldots X_n\}$ corresponds to the various attributes that the probability will depend. The term $P(X)$ does not depend on C_i and is being used for the purpose of normalization. $P(C_i|X)$ will be maximum when $\{P(X|C_i)P(C_i)\}$ is maximum [Luger 2001]. The equation for Affinity Index, AI, can thus be written as in Equation 3.3[Ahmed & Kanhere 2010].

$$P(C_i|X) = \frac{\{P(X|C_i)P(C_i)\}}{P(X)} \tag{3.2}$$

$$AI = P(C_{yes} \cdot \prod_{\forall i} P(attribute_i / C_{yes}) \tag{3.3}$$

Assuming that every attribute x_i is independent of attribute x_j, for $j \neq i$, AI can be computed as shown in Equation 3.4.

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ AI &= P(C_i) \cdot \prod_{\forall k} P(x_k|C_i) \end{aligned} \tag{3.4}$$

Since we are multiplying the probabilities for each attribute; even if one of the attributes has a probability value of 0, the whole index will become zero. To avoid this, we use a very low probability value (e.g. 0.0001) whenever we encounter such a scenario. In the next section, we use an example scenario to illustrate the working of the algorithm.

3.2.1.3 An Example

In the example below, source S broadcasts a request for destination D. It calculates its own affinity and puts it in the Route Request (RREQ). Upon receiving RREQ, nodes A , B and C compare the affinity in RREQ with affinity in their Routing Table (RT). Since affinity in RT was less, they calculate their own affinity and add it in RREQ, and then they broadcast RREQ as shown in Figure 3.12.

Now the destination upon receiving the route requests, replies by adding 1 to the affinity in *RREQ* received. Now each intermediate node (i.e. A, B, C) will forward route reply (*RREP*) to previous node if *RREP* contains higher affinity than that in their RT as shown in Figure 3.13.

Source S upon receiving *RREP*, chooses route through C and A , while discarding *RREP* from B since it had affinity lower than best two replies received by S. Now S starts sending data through both paths alternatively.

The history information for the example here is shown in Table3.1.

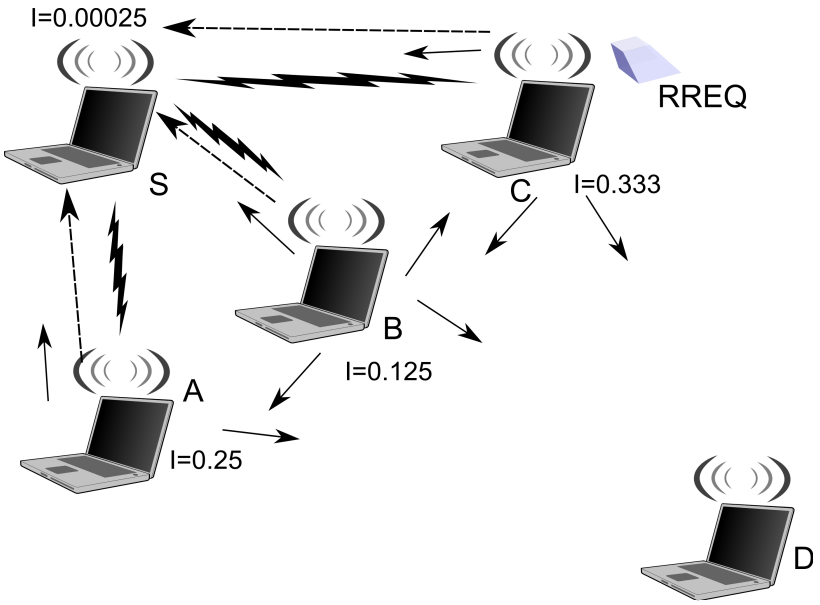

Figure 3.12: Example for showing forwarding of *RREQ* - for sample network

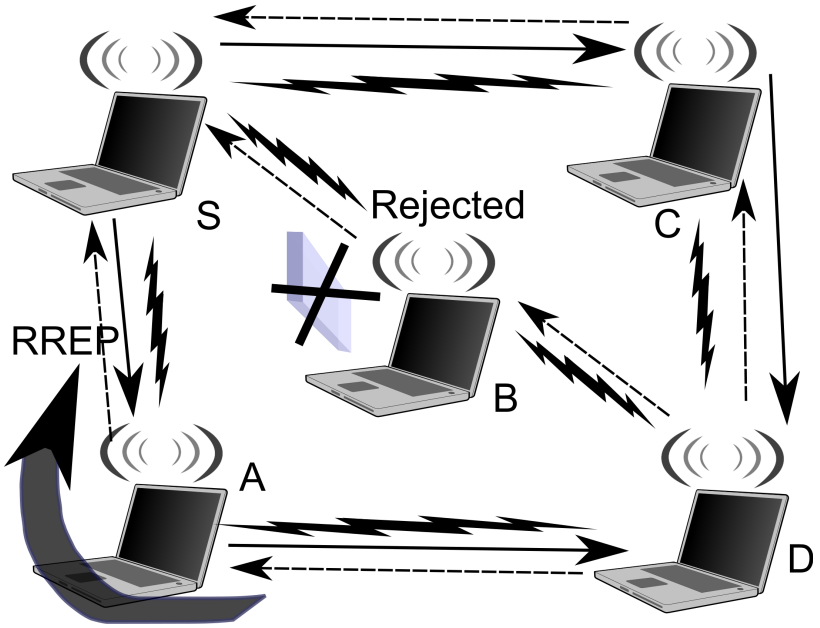

Figure 3.13: Example showing forwarding of *RREP* **and path for transferring data** -
**Solid line means the route used for sending the data packets.
**Dotted lines mean path to previous node (to which reply must be sent)
** I is each node's affinity for destination D

Table 3.1: History information at different nodes for network in Figure 3.12

Destination	Region (Attr1)	Time Slot (Attr 2)	Class
D	R2	T2	Yes
A	R1	T1	Yes
B	R1	T2	No
C	R1	T1	No
D	R2	T1	No

For Destination D from node A
$P(X_{R1}|C_y) = 0.001$
$P(X_{T1}|C_y) = 0.001$
$P(C_y) = 0.5$
I = 0.0000005

Destination	Region (Attr1)	Time Slot (Attr 2)	Class
D	R2	T2	Yes
A	R1	T1	Yes
B	R1	T2	No
C	R1	T1	No
D	R2	T1	Yes

For Destination D from node S
$P(X_{R1}|C_y) = 0.001$
$P(X_{T1}|C_y) = 0.5$
$P(C_y) = 0.5$
I = 0.00025

Destination	Region (Attr1)	Time Slot (Attr 2)	Class
D	R1	T2	Yes
C	R1	T2	No
B	R2	T2	No
A	R1	T1	No
D	R2	T1	Yes

For Destination D from node B
$P(X_{R1}|C_y) = \frac{1}{2}$
$P(X_{T1}|C_y) = \frac{1}{2}$
$P(C_y) = 0.5$
I = 0.125

Destination	Region (Attr1)	Time Slot (Attr 2)	Class
D	R1	T1	Yes
A	R1	T1	Yes
B	R2	T2	No
C	R1	T1	Yes
D	R2	T1	Yes

For Destination D from node C
$P(X_{R1}|C_y) = \frac{2}{3}$
$P(X_{T1}|C_y) = \frac{3}{3}$
$P(C_y) = 0.5$
I = 0.3333

3.2.2 Algorithm for Efficient On-demand Routing Protocol

In this section, we discuss the algorithm for EORP. Route generation is performed by the handling of $RREQ$ and $RREP$ packets. Algorithm 3.14 describes the steps taken while sending a $RREQ$ packet. The algorithm used for forwarding a RREQ packet is described in Algorithm 3.15. The handling of $RREP$ packet is described in Algorithm 3.16.

Figure 3.14: Procedure for sending route request (RREQ) packet at node u

```
/* sendRREQ : Sending the RREQ packet from node u                    */
/* History H maintians the historic information for computing
affinity                                                             */
1 begin
2     foreach v ∈ N_u do
3         if received(data_pkt) ∧ Route_flag = INVALID then no route to destination
4             affinity(rreq) ← affinity(rreq) + affinity(u)
5             Broadcast RREQ(destination_id, affinity(rreq))
6             History H ← H ∪ {< desitnation_id, region, time, status>}
7         send(data_packet)
8 end
```

Figure 3.15: Procedure for forwarding route request (RREQ) packet at node u

```
/* fwdRREQ : Forwarding the RREQ packet at node u                    */
1 begin
2     foreach v ∈ N_u do
3         if Route_flag = VALID then Route exists
4             send(RREP)
5         if destination_id(rreq) = id(u) then I am the destination
6             affinity(rrep) ← affinity(rreq) + 1
7             send( RREP <destination_id, affinity(rrep)> )
8         else if affinity(rtable) > affinity(rreq) then
9             affinity(rtable) ← affinity(rreq)
10            affinity(rreq) ← affinity(rreq) + affinity(u)
11            Broadcast RREQ <destination_id, affinity(rreq)>
12            History H ← H ∪ {< desitnation_id, region, time, status>}
13        Discard RREQ
14 end
```

In our algorithm, local repair works similar to AODV, except that we use Bayesian Approach discussed in Section 3.2.1 to find routes instead of destination sequence number, as shown in Algorithm 3.17.

Figure 3.16: Procedure for handling Route Reply (RREP) packet at node u

```
/* recvRREP : handles RREP packet                                    */
1  begin
2      foreach v ∈ N_u do
3          if source_id(rrep) ≠ u then
4              if affinity(rrep) > affinity(rtable) then
5                  Route_flag ← VALID
6                  affinity(rtable) ← affinity(rrep)
7                  Forward (RREP <destination_id, affinity(rtable)>)
8                  update( history, destination_id )
9              Discard RREP
10         else current node is the source
11             if affinity(rrep) > affinity(rtable) then
12                 Route_flag ← VALID
13                 backup_hop(rtalbe) ← nexthop(rtable)
14                 backup_affinity(rtable) ← affinity(rtable)
15                 nexthop(rtable) ← sender_ip affinity(rtable) ← affinity(rrep)
16             else if (affinity(rrep) ≤ affinity(rtable)) ∧ (affinity(rrep) >
                   backup_affinity(rtable)) then only source needs to maintain backup
                   routes
17                 Route_flag ← VALID
18                 backup_hop(rtalbe) ← sender_ip
19                 backup_affinity(rtalbe) ← affinity(rrep)
20
21 end
```

3.2.3 Performance Results

We have done all the simulations using the network simulator NCTUNs (version 6.0) using 802.11 wireless network [Wang *et al.* 2003]. The paths of all moving nodes were generated randomly, and the payload used was 1400 bytes. A sample simulation run is shown in Figure 3.18. For calculating delivery ratio, network size used was 10 nodes, as shown in Figure 3.19. For computing control overhead, we had used a network containing 32 nodes. All values were within 95% confidence interval.

Delivery Ratio: Delivery ratio is the number of data packets received by destination upon number of packets sent by the source. Any protocol aims to have a higher delivery ratio so that there is minimum loss of data packets. Figure 3.19 shows our result for delivery ratio with respect to varying mobility.

With zero mobility i.e. static network, both protocols have nearly 100% transfer of

Figure 3.17: Procedure for Route Repair at node u

```
/* recvRREP : Sending and Forwarding the RERR packet          */
1  begin
2      foreach v ∈ N_u do
3          if received(data_packet) ∧ Route_flag = VALID then
4              previous_node(rtable) ← send(rerr)
5          else if received(rerr) then
6              if backup_hop(rtable) ≠ Φ then
7                  if backup_hop(rtable) ≠ sender_ip then
8                      affinity(rtable) ← backup_affinity(rtable)
9                      nexthop(rtable) ← backup_hop(rtable)
10                 backup_hop(rtable) ← Φ
11                 backup_affinity(rtable) ← 0
12                 Route_flag ← VALID
13             else
14                 Route_flag ← INVALID
15                 previous_node(rtable) ← send(rerr)
16 end
```

data, but with the increase in mobility, EORP is able to maintain a higher delivery ratio as compared to AODV.

Effect of Mobility on Packet Overhead: The number of control packets flooded into the network is the major reason of having unnecessary congestion and collision of data packets. Unnecessary flooding of control packets can cause collision of data packets, causing extensive increase in the number of data packets being dropped because of which any genuine packet may also not be able to reach the destination. Hence we present experimental analysis of the packet overhead between EORP and AODV.

Initially the overhead is more in both protocols, which decreases once the initial route is established. Later on, whenever a route breaks, control packets are again flooded into the network for finding the new path. Figure 3.20, 3.21 and 3.22 shows the number of control packets broadcasted by our protocol over a sample run under varying mobility of 0 m/s(static network), 5 m/s(low mobility) and 20 m/s(high mobility) respectively.

From the results it can be seen that, with the increase in mobility, number of control packets that are broadcast increase manifolds in AODV whereas in EORP the increase is very less. In EORP there is an increase of only 13.68% in total control packet overhead,

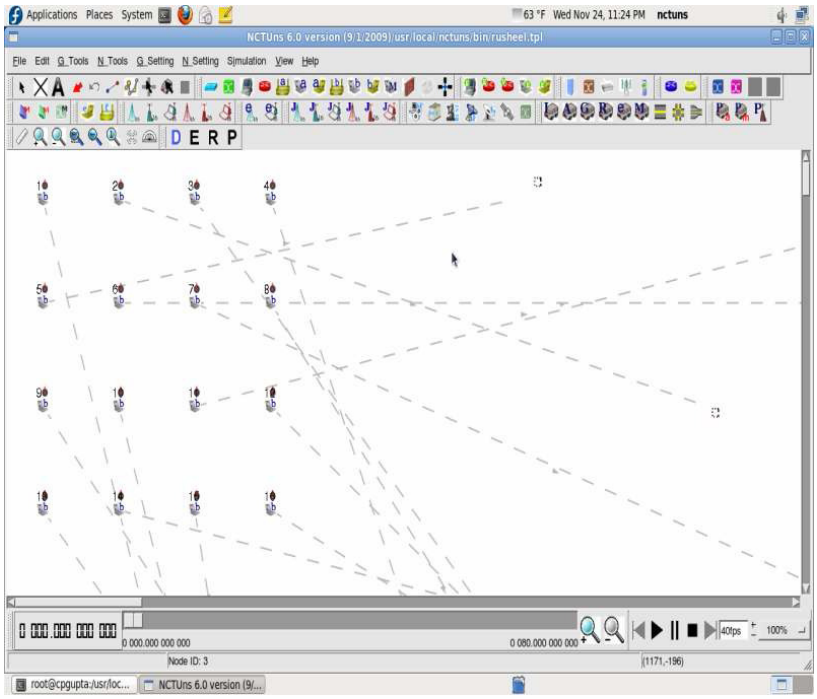

Figure 3.18: A sample simulation run -

when mobility was increased from 5 m/s to 20 m/s whereas AODV showed a jump of 18.78%. The reason why there is very high number of control packets in EORP initially as compared to AODV is that, in the very beginning of data transfer, we need to access secondary memory, so that the stored history can be loaded into the primary memory. This produces an additional time delay in the beginning, which causes the nodes to broadcast more number of control packets, because at that time they have no paths to destination. But after history is loaded into the secondary memory, for the remaining part of the data transfer, control packet overhead reduces drastically as compared to AODV. This clearly shows that, by using the Bayesian approach in finding the route, EORP is able to find such routes to destination which have better life time.

3.2.4 Conclusion

In this algorithm, we are using both time and space information to compute the route from source to destination. We have maintained the historic traffic information in each node along with the details on relative region from which the requests had come from by just expanding the current broadcast cache used in AODV. By using a Bayesian method, we have limited the flooding of broadcast requests. Though EORP, as a protocol, was built

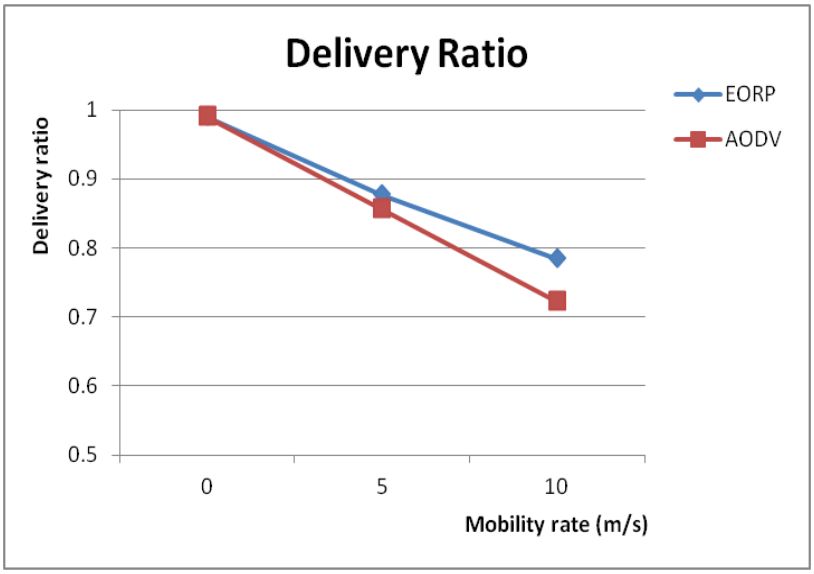

Figure 3.19: Delivery ratio over mobility -

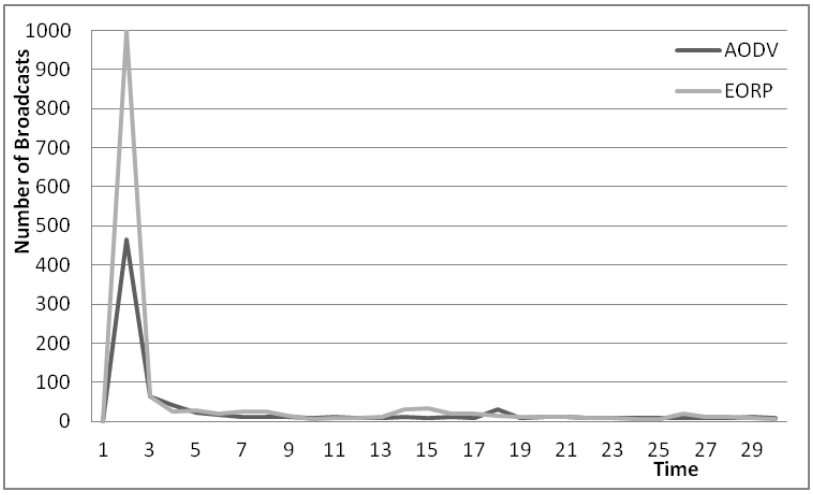

Figure 3.20: Number of control packets broadcast over time(no mobility) - static network

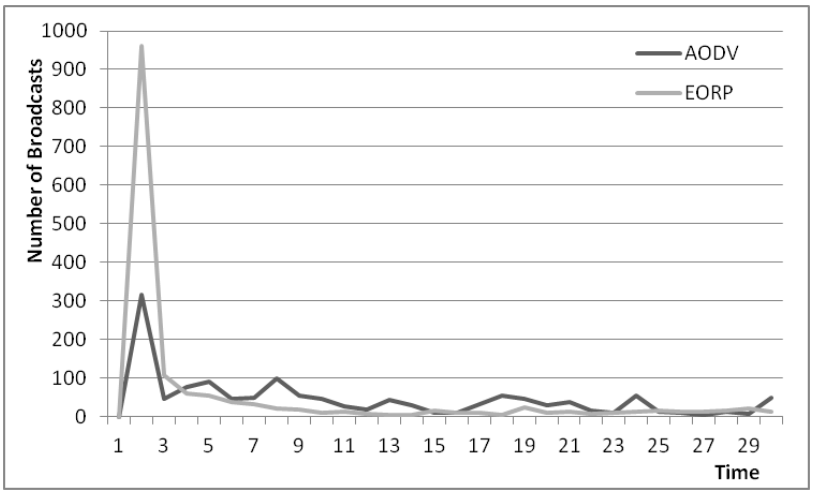

Figure 3.21: Number of control packets broadcast over time(low mobility) - mobility of 5 m/s

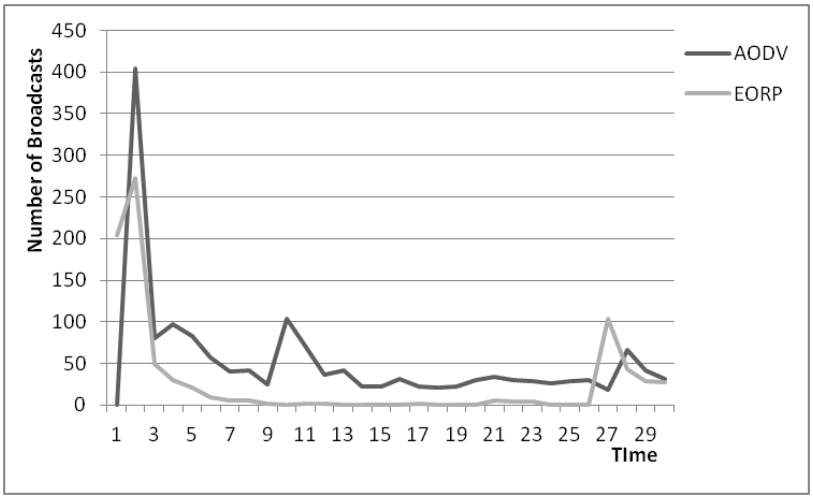

Figure 3.22: Number of control packets broadcast over time(high mobility) - mobility of 20 m/s

on top of the on-demand routing approach, it can easily be extended to other routing approaches as well.

3.3 A Generalized Energy Consumption Model for Wireless Sensor Networks

In this section, we discuss a Generalized Energy Consumption Model(GECM) for wireless sensor networks. Though this model was written with a wireless sensor network as the background network, it can be extended to any wireless network including MANETs with suitable modifications. This model takes into account the high energy consumption processes in nodes, derived using the two parameters of relative hop number and a certain usage pattern ratio. The energy consumed by the network as a whole is compared vis-a-vis the solar energy that may be harvested. In this work, we consider a wireless sensor network implementing a hierarchical routing strategy towards sinks and examine the energy constraints and the plausibility of achieving an energy neutral operation. Section 3.3.1 provides necessary background for this work. Section 3.3.2 shows the basic network topology. Section 3.3.3 presents the system model used. Section 3.3.4 details the energy budget system used. Section 3.3.5 shows the results of our simulation. Section 3.3.6 provides the concluding remarks.

3.3.1 Background

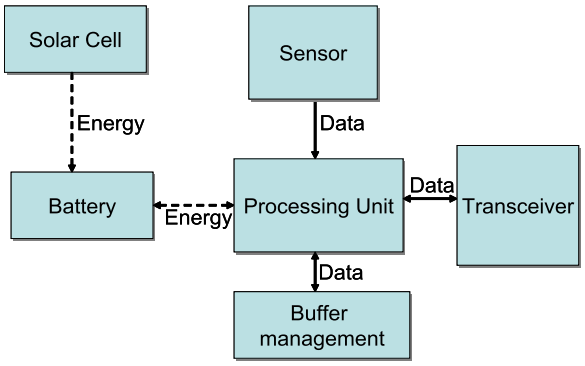

Figure 3.23: Structure of a sensor node -

The general structure of a wireless sensor node that supports energy harvesting is shown in Figure 3.23. The solar cell supplies energy to battery that powers the processing

unit. The processing element processes the incoming data and transmits them in the form of packets via the transceiver. The task of buffer management is to hold the incoming data waiting for channel access and to help in packet scheduling. The relevant literature was introduced in Section 2.3.3. In our work, we are mainly dealing with energy saving at routing layer.

The problem that needs to be addressed here is essentially two-fold. The first being the distribution of battery capacity among the different clusters and the rate at which the cluster head ought to switch from one node to another. The other being the feasibility of an Energy Neutral Operation(ENO) [Vigorito *et al.* 2007] while assuming the nodes are capable of harvesting solar energy.

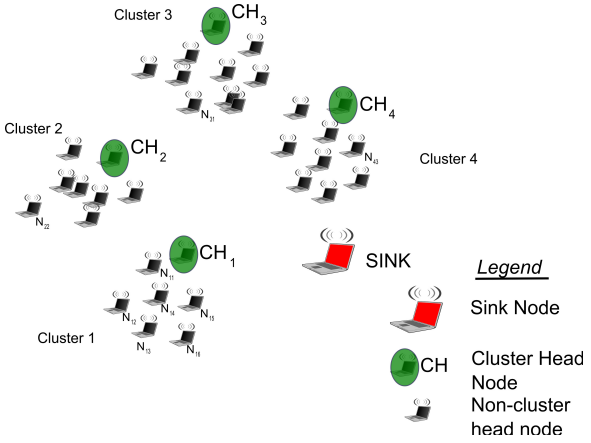

Figure 3.24: A sample sensor network -
The nodes within green circle represent the cluster heads of the clusters that are constituted by the nearby nodes.
The node with red screen represents the sink.

3.3.2 Network Topology

We have considered a sensor network running hierarchical routing for our implementation as shown in Figure 3.24. The entire network is considered to have nodes that form clusters with other nodes that are geographically *close*. *Closeness* is decided so that the inclusion of another node does not degrade the energy efficiency of the cluster as a whole. The nodes belonging to any particular cluster are homogeneous in all aspects. Further, every cluster has one of its nodes acting as a cluster head (CH) at any given instant of time. All the non-cluster head (nCH) nodes in a given cluster transmit their data to its CH node, which further routes the data through other CH nodes until the *Sink* is reached. In this

network, only a single level of clustering is assumed. Further the role of the CH is donned by different nodes and it keeps switching from one node to another in order to maintain nearly the same level of energy among all the nodes of a cluster. For any given number of nodes, the number of CHs is around 5% of the total number of nodes. Every nCH node is assumed to be sampling at two different frequencies; one for normal scenario and another for alert scenarios.

3.3.3 System Model and Assumptions

For modeling the network, the following notation is followed: N_{ij} is used to refer to the j^{th} node in the i^{th} cluster. Hop number as a parameter assumes significance given that routing takes place through the CH nodes until the sink and that there is a strong correlation between the number of packets routed through a CH node and its hop number. We assume that the hop number of every node in a cluster is the same when it acts as the CH node for the sake of simplicity. For the i^{th} cluster, we denote the relative hop number by X_i, which is the ratio of the hop number of the i^{th} node to the maximum hop number of a CH.

We consider two states of operations. State a is used for normal sampling of data. State b is used for alert scenarios. The number of packets is a function of the relative number of nodes in state a that route their packets through a particular CH node. We define another critical parameter A_i, which is the ratio of the number of nodes in state a that route through the i^{th} CH to the total number of nodes (which also includes nodes that are in the state b) that route through it.

3.3.4 Energy Budget

We take a look at the operations performed by both the CH and the nCH nodes in order to be able to account for their energy expenditure. For a CH node, the energy is spent primarily on the routing of data. While the CH node is not routing or similar active particpiation in the WSN, it will go to the sleep state where the energy consumed is lesser than energy spent during data transmission. For example, while INTEL's StrongARM microcontroller consumes 400mW executing instructions, it takes up a meager 0.16mW

in the sleep state. Thus, we may ignore the energy consumed in the sleep state in our calculations. Also even if the sensing activity were to occur concomitantly with the other operations of the CH node, it would account for far lesser than that for transmission.

The derivation of a more exact relation for the consumption of power in CH nodes is based on the following:

- Energy, being a weighted mean of the nodes in the states a and b, is directly proportional to A_i.
- It is inversely proportional to the relative hop number as nodes closer to the sink have to route a far greater number of packets. For the degree to which it was inversely proportional, the quantity must depend on the exact distribution of nodes. As in most cases it would turn out to be of a degree of more than 1 and is close to exponential, we choose 2.

Thus, the power consumption of the CH Node, N_{ij}, is computed as per Equation 3.5. Here re, α, m and β are constants. While α and β relate to the nature of dependence of the power consumed on X_i, m relates to the ratio of the sampling frequency in the state b to the sampling frequency in the state a.

$$P(N_{ij}) = re\left\{\frac{A_i + m(1 - A_i)}{\alpha \cdot X_i^2 + \beta}\right\} \tag{3.5}$$

An nCH node consumes energy when it is sensing data from the environment, as well as when it is transmitting data. The power consumed by an nCH node is given in Equation 3.6. Here, se and te are constants corresponding to sensing and transmission. The overall energy consumed by any node is a weighted sum of the time spent as a CH node and as a non-CH node.

$$P_s = se + te \tag{3.6}$$

3.3.5 Simulation Results

For the above model, we choose plausible values of constants in order to be able to better appreciate it. It is known that the constants P_s and re have nearly the same value. Using the fact that the energy required for transmitter/receiver is approximately $50nJ/bit$ and for the amplifier it is $0.01nJ/bit/m^2$ [Gupta & Younis 2003], and assuming a sampling frequency of 20Hz in the state a and the frequency in the state b to be five times that value, we obtain plots of power of a node versus both A_i and X_i. Note that here we take α as 1 and β as 0.

From Figure 3.25 and Figure 3.26, it can be inferred that the difference in the power consumption between CH nodes with different X_i is high at small values of A_i, thereby making energy management among homogenous nodes difficult and also rendering it impossible to keep the network alive for long. Thus, it is essential that A_i is greater than 0.8 and also that X_i is not less than 0.1 for any CH, as having too many hops causes bottlenecks in the CH's closer to the sink as well causes a great energy drain in them.

From the methods suggested by Kansal, *et al.* in [Kansal *et al.* 2007] with a further assumption of uniform solar energy harvesting of nodes and from the relationship between relative hop number and the parameter A_i as shown in Figure 3.25, it can be inferred that ENO can be achieved if the deployment of nodes is such that A_i is higher. Instead of focusing on A_i, it may be easier to achieve ENO and also prolong the life of the network by routing the packets such that X_i value becomes higher as seen in Figure 3.26.

3.3.6 Conclusion

In this work, we developed an energy model to observe power consumption patterns in a hierarchical wireless sensor network based on two parameters; namely, relative hop number(X_i) and the relative activity A_i. While the energy-relative hop number relation is a function of topology and routing approach, the activity is application dependent. Though the analysis presented here is valid for the particular case of hierarchical routing, the relationship the relative hop number and the relative activity of nodes routing through a particular node is clearly established in our model. There is further scope of this work in the context of heterogeneous networks. This model can hence be generalized

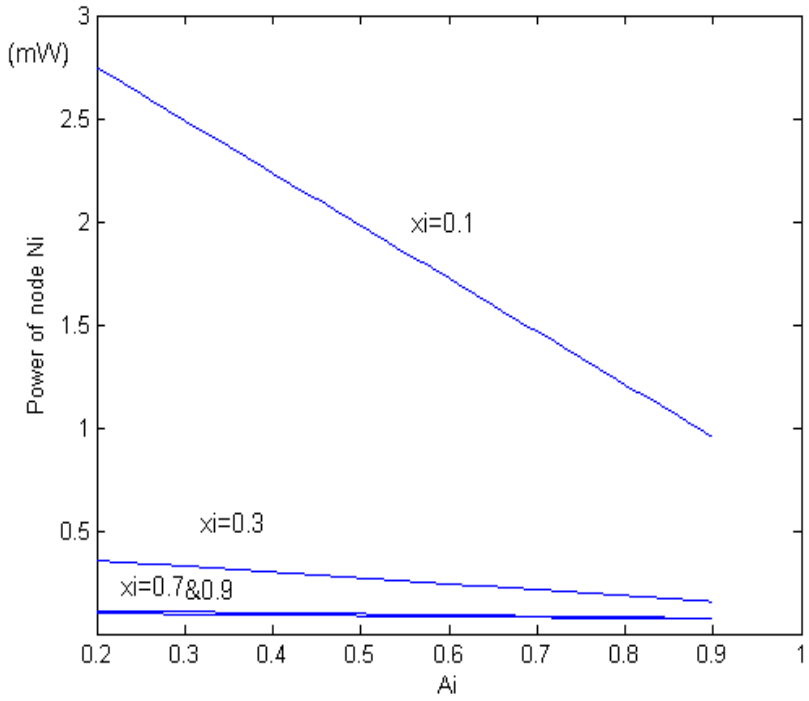

Figure 3.25: Power of a node in the i^{th} cluster v/s A_i -

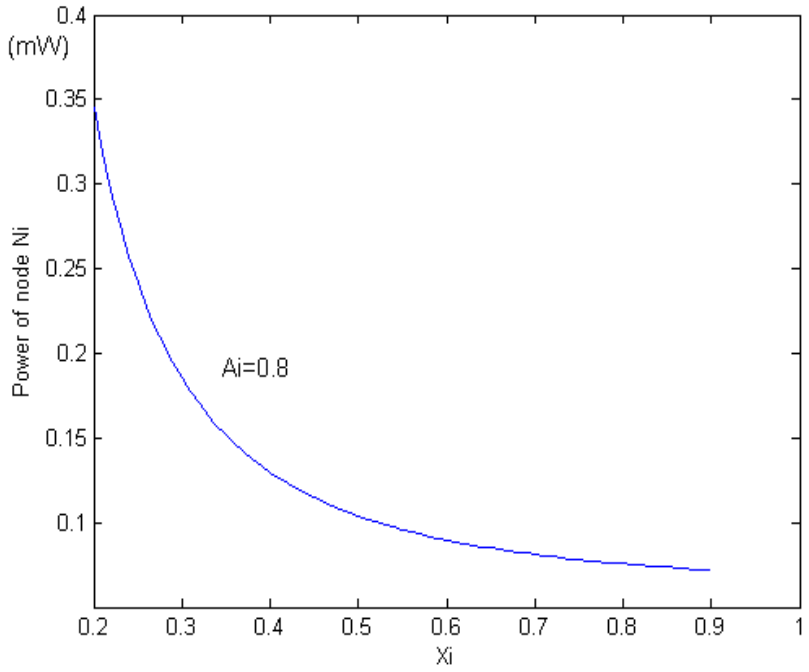

Figure 3.26: Power of a node in the i^{th} cluster v/s X_i -

for application in MANETs, especially while designing routing algorithms in networks where the nodes can logically be organized into clusters/regions.

3.4 Conclusion

In this chapter, we approached three specific sub-problems in energy aware routing. For the purpose of solving these sub-problems, we reduced our scope to uni-casting to evaluate our options.

We started by exploring whether there could be a better metric that could optimize network life time in Section 3.1. We proposed a new metric for identifying energy cost, and demonstrated its efficiency in the section. Our cost function was capable of maximizing the average residual energy of the network, minimizing the variance of the power of the nodes in a MANET, and still reduce the cost for each packet over transmission on the mobile network. Our simulation results demonstrated that the cost metric used significantly improved these power-aware metrics. This approach worked fairly well when deployed in multimedia application environment, where a large amount of data has to be streamed between the source and destination nodes. We could hence use the metric for energy computation in algorithms/protocols explained in Chapter 4.

The next problem we explored is related to impact of energy in route maintenance in Section 3.2. When we explored this sub-problem, we realized that there is a certain affinity that can be attached to nodes, and this affinity could be leveraged for energy efficiency in routing in general. Historic information was used along with details regarding the relative regions from which the requests were initiated to compute efficient paths in this approach. The Bayesian approach used could leverage the affinity of nodes to efficiently forward routing requests to the rest of the nodes. We had used the on-demand routing approach to prove that the method is viable. The same method can be expanded fairly easily in the algorithms discussed in Chapter 4 as simple modifications.

The final sub-problem, discussed in Section 3.3, explored the possibility of including energy harvesting techniques in MANETs. We reduced the problem to the specific constraints of a wireless sensor network, and explored the available options for improving energy efficiency. The model suggested in Section 3.3 observed the relative hop number

and the relative activity of nodes routing through a particular node to explore the changes needed in the basic energy metric when the device is capable of energy harvesting.

In this chapter, we thus looked at factors that need to be addressed while looking at energy-efficient routing in MANETs. We refined these factors and formulated sub problems to reflect the additional aspects that ought to be covered while looking at routing in MANETs. These factors by themselves do not form the basis of multicast routing in MANETs. However, the results of these algorithms can be used to augment the effectiveness of the algorithms and methods explored in the next chapter.

Chapter 4

Quality of Service (QoS) Aware Multicast Routing in Mobile Ad-hoc Networks

Multicast routing algorithms for mobile ad-hoc networks have been extensively researched in the recent past. In this chapter, we present three algorithms for dealing with multicast routing problem using the notion of virtual forces. We look at the effective force exerted on a packet and determine whether a node could be considered as a Steiner node. The nodes' location information is used to generate virtual circuits corresponding to the multicast route. QoS parameters are taken into consideration in the form of virtual dampening force. The first algorithm produces relatively minimal multicast trees under the set of constraints. We present a second algorithm that provides improvement in average residual energy in the network as well as effective cost per data packet transmitted. We observe that we could improve further on the energy consumed for multicast communication. The third algorithm uses the virtual force technique to generate multicast trees rooted on geographically distant nodes. This enabled us to improve the performance in terms of minimizing the branch nodes in the multicast trees provided by the virtual force technique, while still be able to improve on the effective number of hops per multicast flow. The three algorithms suggested in this chapter are the first ones to use the virtual force technique for multicast routing in MANETs.

This chapter is organized as follows. In Section 4.1, we provide the basic introduction and motivation for our work explained in this chapter. In Section 4.2, we discuss the relevant back ground work and the reason why virtual force technique is challenging in multicast communication. In Section 4.3, we discuss the virtual force technique proposed by us. In Section 4.4, we discuss the first algorithm named as Multicast Routing Algorithm using Virtual-force(MRAV) proposed by us. In Section 4.5, we discuss the second multicast routing algorithm, named as Sector-based Virtual-force-based Multicast (VFM) Routing Algorithm. In Section 4.6, we discuss the third algorithm named as Virtual Multicast Tree (VMT) routing algorithm proposed by us. In Section 4.7, we discuss the results observed by us using simulation. We conclude the chapter in Section 4.8.

4.1 Introduction and Motivation

A mobile ad-hoc network (MANET) involves independent nodes that are rapidly deployed in an environment. These mobile nodes must make the correct routing decisions in the presence of node mobility, transient link failures and energy and other constraints. In the event of sending multiple copies of the same information to multiple nodes, use of multicast routing is the preferred option. Depending on how the distribution paths among multicast group members are created, multicast routing protocols can be classified into tree-based, mesh-based and hybrid multicast routing protocols [Junhai *et al.* 2009]. Tree-based protocols can further be sub-divided into source-tree and core-tree protocols. In our algorithms, we take the middle path by letting the multicast tree to be created by the initial packet in transit.

The creation of multicast tree can be equated with the Steiner tree problem. The Steiner tree problem was defined in Section 1.9. A Steiner minimal tree constructed for a local sub-graph of a G need not be part of the minimal tree constructed for the entire graph[Gilbert & Pollak 1968]. Steiner tree problem is known to be NP-complete for planar graphs[Karp 1975]. Approximation algorithms like the one given in [Borradaile *et al.* 2009]

have been suggested for the Steiner tree problem. A related notion is to classify a tree as relatively minimal Steiner tree. In a relatively minimal Steiner tree, the length of the tree is minimal for the specific set of nodes chosen to be part of the tree. The problem of multicast communication reduces to finding a minimal Steiner tree in a graph. The task of a multicast routing problem is to determine the correct set of Steiner nodes S so that the tree can reach all terminal nodes in Q. For efficient communication, any multicast tree created must at least be relatively minimal. In our approach, we attempt to determine a relatively minimal Steiner tree with the help of virtual force approach.

The notion of virtual force had been explored in the context of the deployment problem in wireless sensor networks[Zou & Chakrabarty 2003, Poduri & Sukhatme 2004], and in routing in MANETs [Liu & Wu 2009]. In the virtual force approach(VFA), the participating nodes and/or the packet in transit are applied with some electric charge. The electrostatic forces are computed and the routing decision is made based on the magnitude and direction of the resultant force. Poduri, *et al*. [Poduri & Sukhatme 2004] demonstrated the use of a combination of attractive and repulsive forces for solving sensor deployment problem in wireless sensor networks. Liu, *et al*. [Liu & Wu 2006, Liu & Wu 2009] used the concept of virtual force in unicast routing for MANETs. In the routing algorithm for MANETs, the resultant force is used only to make a decision to move the packet forward.

Our main contribution described in this chapter is the use of virtual force in multicast routing for MANETs. We use the notion of virtual force to guide the packet towards the destinations. The effective force on the packet is a sum of contributing force values from the destinations, the source node and the packet itself. In addition to the forward guiding force, we use dampeners to limit choice of the next node in the path to accommodate QoS parameters. The combined result of these forces will let the packet know whether it is currently in a Steiner node. The resultant force will be towards the node that will be in the path to the majority of the destination nodes in one direction of a branch in the multicast tree.

While attempting to use the virtual force technique for multicast routing, we identified several new issues that were not present in unicast routing algorithms. The traditional virtual force technique could not handle multiple destinations as the packet was frequently forced to move towards the force equilibrium. Also, there was no means for including the

QoS parameters in the traditional virtual force technique as described in [Liu & Wu 2009]. The first algorithm proposed in this chapter deals with directly imposing the virtual force technique for MANETs. We provide a refined algorithm that uses virtual force in a different manner from the way it is done by [Liu & Wu 2009]. We divide the region around the current node into sectors and perform a virtual-force based multicast routing on each of the sectors. Multicast routing using virtual force technique is attempted for the first time in this chapter. As far as we know, we were the first to suggest the use of sectors of variable arc-lengths for performing multicast routing, in conjunction with the virtual force technique.

4.2 Background

We present some of background work that formed the basis of the research presented in this chapter. A broader literature review was presented in Sections 2.2 and 2.4.

Gilbert, *et al.* in [Gilbert & Pollak 1968] had put forth the criteria that the angle between out-going edges at a branch node for a Steiner node is 120 *degrees* for a relatively minimal Steiner tree. However, in a real-world MANET, it is not possible to choose branching nodes such that the angle between the out-going edges are exactly 120 *degrees*. The key challenge in a MANET is to be able to handle cases where such an ideal branching node cannot be found.

[Mauve *et al.* 2003] suggested a greedy multicast routing algorithm that used a heuristic dependent on the normalized number of next hop neighbours to determine whether a branching node in multicast tree has been reached. The parameter λ used in their algorithm determines how late the split of the multicast forwarding will take place, with $\lambda = 0$ indicating an early split and $\lambda \approx 1$ for a very late split.

Liu, *et al.* suggested two routing algorithms, SWING [Liu & Wu 2006] and SWING+ [Liu & Wu 2009], for unicast routing in MANETs based on the notion of small world networks. These algorithms expanded the notion of virtual circuits as described in Geographic virtual Circuit Routing Protocol (GCRP) [Fotopoulou-Prigipa & McDonald 2004] to include long virtual logical links to the surrounding neighbourhood of the current node with the help of virtual force. Liu, *et al.* chose the node with the maximum repulsive force

from the source node towards the destination node as the next hop for forwarding the packet. However, if we apply electrical charges in only the source node and the packet, as was described in Liu, *et al*.[Liu & Wu 2009], we found during our research that we could not really accommodate for multiple destinations typical in a multicasting environment. Figure 4.1 shows a sample network where the force values of all the four destination nodes, $\{D1, D2, D3, D4\}$ cancel each other. Initially, when we attempted to apply the electrical charges in destination nodes and the packet in transit, the packet was moving towards a force-equilibrium position over and over again, increasing the likelihood of loops in the multicast tree. In the Figure 4.1 the force equilibrium is at node u itself. It is trivial to argue that a force equilibrium centred between two neighbouring nodes will lead to the packet being forwarded to the two neighbours. Also, supporting QoS parameters in a multicast route is slightly different than adding it in a simple virtual circuit because of the nature of path establishment.

Rahman *et al*. in [Rahman & Gregory 2011b] divided the region around the current node into quadrants, and considered four closest nodes in each quadrant for determining the next hop in the multicast tree. They later expanded their algorithm [Rahman & Gregory 2011a] by using an intelligent energy matrix to increase the average life of the nodes in the network. Their algorithm provides a reasonable length of the multicast tree as was determined during our simulations. However, we found that by statically fixing the quadrants, their algorithm was generating slightly longer multicast trees. Instead of limiting the decision into fixed quadrants around the current node, we dynamically divide the region around the current node into α sectors based on the direction of the effective force due to the nodes in the multicast group. In this chapter, we have devised simple mechanisms to adapt the concept of virtual circuits into the multicast environment.

4.3 Multicast Routing Using the Virtual Force Technique

In this section, we start with a discussion on the assumptions and basic working of our multicast routing algorithms with the help of the virtual force technique. We then introduce the system model that we have used.

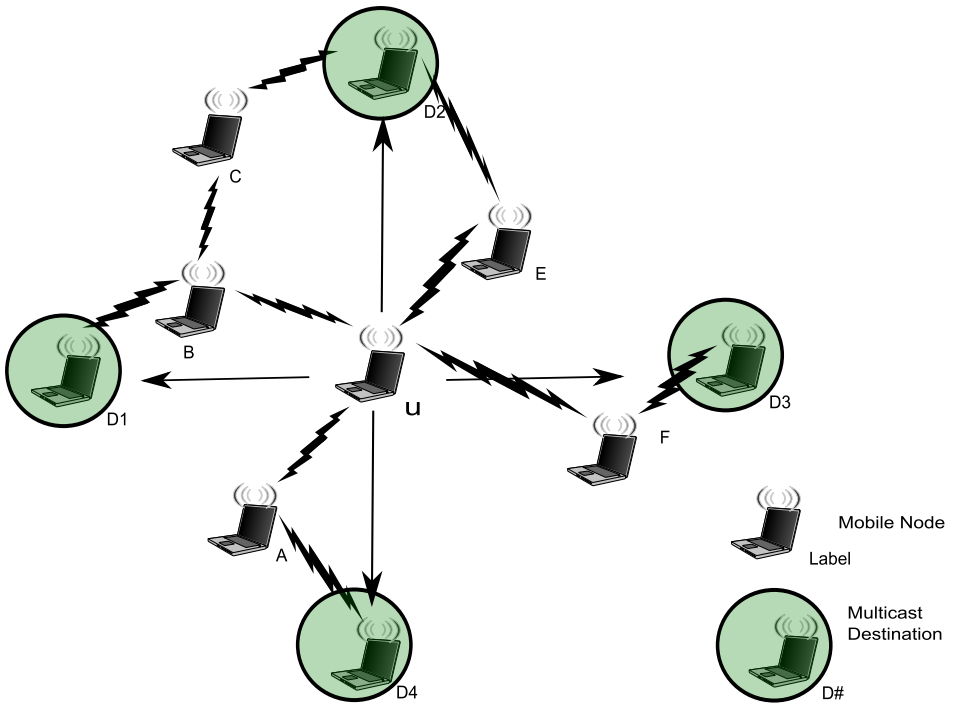

Figure 4.1: Example for failure of classic virtual force technique - four multicast destinations

4.3.1 Assumptions

We assume that location information is available in all nodes in the network and that all nodes in the network will become eventually aware of location and mobility information of other nodes in the network. Most off-the-shelf products currently come with GPS facility. Our algorithm assumes that there is an appropriate data structure that can store this location information. Such a data structure will allow look-up of node location from the node's address. As far as proper working of the algorithm is concerned, the location information in the immediate vicinity of the current node needs to be accurate. There can be a slight imprecision in the location values of farther nodes, as the algorithm only computes the best likely members of the multicast route for the farther nodes. As the query for computing the multicast route reaches closer to the actual destination(s), the intermediate node can divert the query appropriately as it is assumed that the closer nodes will have more accurate information.

All mobile nodes are assumed to be having omni-directional antennas, as is the case with most of the off-the-shelf devices. As our algorithm relies on the neighbourhood information derived from the Hello protocol[Perkins & Royer 1999], all mobile nodes

need not be in the same plane.

4.3.2 Basic Idea

In this algorithm, we assign positive charges to sender, the multicast destinations, and the packet. The packet is placed at a virtual point near the sender node, and will be guided by the effective repulsive force to move towards the nodes in the multicast group. All the other nodes are given a tentative charge based on QoS parameters.

When a packet reaches node u, we need to decide whether u is a branching node(node at which the multicast tree branches) or not. If u is a branching node, we need to determine the number of branches to be taken, and the specific neighbours that will become part of the branch. If u is not a branching node, then we need to determine which of the neighbors in the forwarding direction has to be chosen for the next hop.

Let M be the nodes that are part of the multicast group. At the node u, we compute the effective force exerted on the packet due to nodes belonging to M, $\vec{F_M}$. We take the direction of $\hat{F_M}$ as the forward direction. Unlike [Rahman & Gregory 2011a], we divide the nodes around u into α sectors, with each sector covering an angle of $\theta = \frac{2 \cdot \pi}{\alpha}$. Here, α varies from 3 to 6 depending on the density of the region containing u. For $\alpha = 4$, we have $\theta = \frac{\pi}{2}$. For each of the α sectors containing some destination nodes, we select the appropriate neighbor in that sector to forward the packet. We compute the effective force on each of the k neighboring nodes $v_1, \ldots, v_k$ in the sector s, by taking into account the cumulative effect of dampening force($\overrightarrow{E_{v_i}}$), u and M_s, where $M_s \subseteq M$, M_s contains all the destination nodes in sector s. Once the appropriate forwarding node v_s for the sector s is determined, the packet is communicated to v_s along with M_s, which is the set of destination nodes to be handled in sector s.

4.3.3 System Model

[Liu & Wu 2009] defined Equation 4.1 for computing the virtual force the current node v to the destination v_d, where d_{max} is the maximum distance measure, and $d(v, v_d)$ is a measure of distance between v and v_d. This equation is sufficient for dealing with a unicast transmission. For determining simple, point-to-point force, we may use the same

equation in our algorithm. However, for a multicast transmission, we need to expand the notion to incorporate all the multicast group nodes. We use the alternate definition of destination force as given in Equation 4.2 in our work. Here, Q is a large constant charge value assigned for ease of computation. This is a minor modification of Equation 4 in [Liu & Wu 2009].

$$F_{dest}(v, v_d) = d_{max} - d(v, v_d) \tag{4.1}$$

$$F_{dest}(v, v_d) = \frac{2 \times Q}{(d(v, v_d))}, v \neq v_d \tag{4.2}$$

At any node in the network, we compute the effective force on the node u due to the current set of destinations, M, as shown in Equation 4.3. Here, $\vec{F_{u,M}}$ is effective force on the node u exerted by all the nodes in the set of current multicast destination, M.

$$\overrightarrow{F_{u,M}} = \sum_{d \in M} \overrightarrow{F_{u,d}} \tag{4.3}$$

Apart from this force acting on the packet, there is a force component from the source node s and a dampening force caused by other parameters determining the choice of the next node. The effective force on the packet p at node u is given in Equation 4.4.

$$\overrightarrow{F_p} = \overrightarrow{F_{s,u}} + \overrightarrow{F_{u,M}} - \overrightarrow{E_u} \tag{4.4}$$

The dampening force E_u produced by the node u depends on QoS parameters used. If the number of QoS parameters to be considered is m, the weightage of the i^{th} parameter is α_i, the value of the parameter i in the node u is δ_i^u, the requested value for the QoS parameter is δ_{req} and the maximum value allowed for that parameter is λ_{max}, then the value of the dampening force at node u is given by Equation 4.5. Here, for each parameter i, $0 \leq \alpha_i \leq 1$ and $\sum_i^m \alpha_i = 1$.

$$\vec{E_u} = (\sum_i^m \frac{\alpha_i \cdot (\delta_{req} - \delta_i^u)}{\lambda_{max}}) \cdot \hat{\mathbf{F}_{\mathbf{s,u}}} \tag{4.5}$$

4.4 Multicast Routing Algorithm using Virtual-force(MRAV)

In this section, we discuss the design of the Multicast Routing Algorithm using Virtual-force (MRAV).

4.4.1 An illustration of MRAV

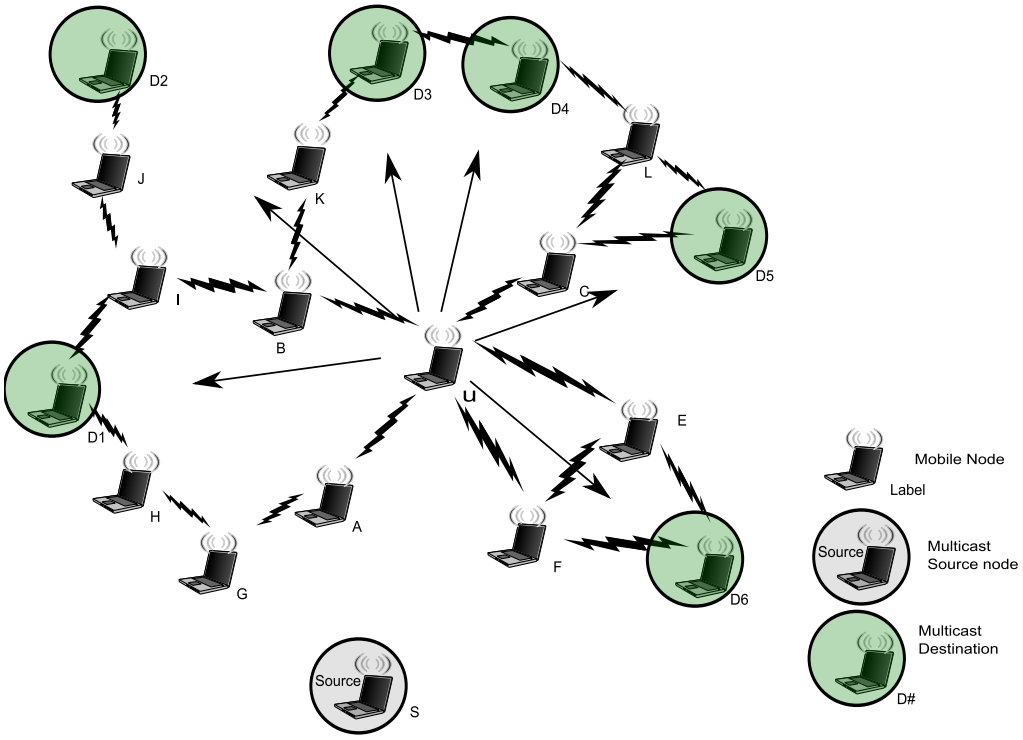

Figure 4.2: Working of MRAV(a simple example) - six destination nodes

Figure 4.2 shows the node u that needs to multicast to six members of the multicast group $M = D1, D2, D3, D4, D5, D6$ represented as nodes. Viable communication links are indicated as lightning bolts. The node u has five neighbouring nodes, namely A, B, C, E and F.

Let us consider the example of a multicast from node u to the set of destination nodes depicted in Figure 4.2. The packet at node u experiences virtual force as a result of interactions with all the six destination nodes. The direction of the force is indicated with the help of arrows in the figure. We first compute the effective force on the packet that is currently at node u to each of the destination nodes. The arrows in the figure indicate the direction of the force vectors due to the six destinations. By computing the effective force

on the node, we can compute the general forward direction of the node.

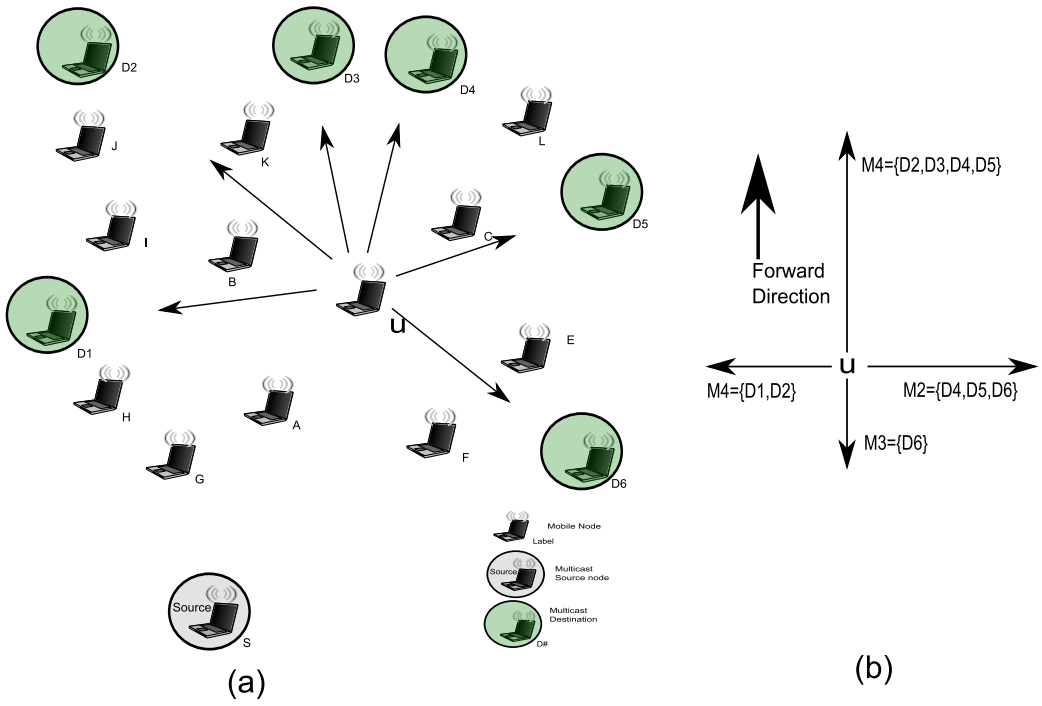

Figure 4.3: Working of MRAV(impact of forces) - (a) The virtual forces exerted by the destinations on node u. (b) Effective force along the axes.

Figure 4.3(a) shows the various force values on the current node, u, of the network shown in Figure 4.2. Here the resultant force at the node u is computed as vector addition of all forces at the point corresponding to node u. For the purpose of illustration, let us assume that the forward direction, which is the direction of the resultant force, is towards $D3$. Figure 4.3 (b) shows the cumulative sum of forces along the axes at node u. Since there are destination forces affecting more than two directions in Figure 4.3 (b), it can be inferred that the node u is a split node. It now has to decide which among the nodes in its neighbourhood set (N_u) will form part of the multicast route.

Let us have four subsets(assuming $\alpha = 4$) of the destination set M, named as M_1, M_2, M_3 and M_4, indicating the set of destination nodes that had contributed to the force vector upwards, rightwards, downwards and leftwards respectively. In Figure 4.3 (b), the elements belonging to the subsets are indicated along with the force component to which they contribute. It can be seen that $D6$ belongs to the destination subset M_3, M_2 contains $D4, D5$ and M_4 contains $D1, D2$. M_1 contains $D2$, $D3$, $D4$, and $D5$.

From Figure 4.3 (b), it can be inferred that only the destination $D6$ is contributing to

the downward direction, which is the direction opposite to the direction of the effective force at u. Also, in a clockwise direction starting from the left-side, it can be observed that the angle between $D1$ and $D6$ is greater than $180degree$. As it is the only node in subset M_3 as well as the angle being more than $120degree$, $D6$ is considered separate from the remaining set of destination nodes. To reach $D6$, the relevant neighbours of u are the nodes E and F. If E is chosen as a multicast forwarding node for u, then both destination $D5$ and $D6$ can be served through it. In our algorithm, all nodes that are within $\pm 60degree$ from the direction of the chosen force vector are considered to be part of the same general path towards multicast set. The *Query* message is forwarded to node E, with destination set as $D5, D6$.

When the multicast route query reaches the node E, it identifies that $D6$ is its neighbour. So the node E directly communicates the query message to $D6$. To communicate to $D5$, however, there is no path in the general direction towards it. Thus choice of E as the next node neighbour for communicating to $D5$ is a wrong one.

To circumvent this issue of wrongly identifying a neighbour to forward the multicast query, we first evaluate all possibilities, before deciding on the list of most suitable forwarding nodes. If the node still encounters a void, we choose perimeter routing as discussed in Greedy Perimeter Stateless Routing (GPSR)[Karp & Kung 2000].

In this case, we first try to combine $D5$ and $D6$ as was discussed earlier, and then go on to determine the rest of the nodes to forward the query. The nodes $D1$ and $D2$ can be considered together, as they form part of the set M_4. The effective force that acts on the node B is the strongest among the neighbours of u. Hence, the forwarding node for these two multicast destinations will be B. The remaining nodes $D3$ and $D4$ can now be considered. The closest neighbour according to our force metric is node C. For the destination node $D5$, while comparing between the node C and the node E, it can be observed that C feels a stronger force than E. The destination $D5$ is hence removed from the *Query* message to node E and then attached to node C. The *Query* message is going to be forwarded to the nodes B, C and E with the destination sets marked as $D1, D2, D3, D4, D5$ and $D6$ respectively. Note that the node F can be used in the place of E, as the effective force value on both E and F due to node $D6$ is the same.

4.4.2 The MRAV Algorithm

In this section, we discuss the multicast routing algorithm using virtual-force.

4.4.2.1 Data structures

Data structures relevant for the algorithm are mentioned in this sub section.

N_u : This is typically implemented as a linked list to store the list of neighbours of the current node. The list is updated with the help responses from the periodic Hello packets transmitted as per the Hello Protocol.

$P(M)$: This is the power set of M, such that all nodes within each of the subset created are reachable in increasing values of the angles. For example, in the sample network in Figure 4.2, $P(M)$ may contain $D1, D2, D3$, but may not contain $D1, D3$ as there exists a node $D2$ direction is in between $D1$ and $D3$.

Π : This is a priority queue that stores the subset of nodes with highest force value on the front of the queue. It is assumed that this queue also has separate lookup function implemented that can retrieve the force value for any subset given as input to the function. This can be achieved by including an auxiliary data structure in the form of a hash table along with the queue Π.

v_s: The set of possible neighbours to choose from, while marking a split node. This list stores the next node as well as the list of destinations that are reachable from it.

4.4.2.2 The Algorithm

The algorithm for sending a *Query* message is shown in Algorithm 4.4. Once the *Query* message has reached the destination or a split node, a virtual circuit linking the destination and the previous split node or the source node shall be constructed as the acknowledgement to the *Query* travels back to the source node.

4.4.3 Analysis

We analyse the MRAV algorithm in this section. The results of simulations are discussed later.

Figure 4.4: MRAV routing algorithm: sending the Query message

```
1  Procedure sendQuery(u, M)
2  begin
3      if u ∈ M then in multicast set
4          M ← M − {u}
5      foreach v ∈ N_u ∧ v ∈ M do neighbor is part of multicast set
6          Call sendQuery(v,M)
7      Let M′ ← M
8      repeat
9          foreach s | s ∈ P(M) do
10             F⃗_s ← calculateVirtualForce(u, s)
11             Π(M) ← Π(M) ∪ (F⃗_s, s)
12         Let M_s be front(Π(M))
13         Compute w ∈ N_u, such that ∠uvw, M̂_s ≤ 60° ∨ ∠M̂_s, uvw ≤ 60°
14         if M_s = M then all nodes are in the same general direction
15             Call sendQuery(w, M) and exit
16         v_s ← v_s ∪ {(w, M_s)}
           /* Remove all entries from P(M) which contain the
              destination nodes M_s                                  */
17         P(M) ← P(M) − {X | X ∈ P(M) ∧ X ∩ M_s ≠ ∅}
18         M′ ← M − M_s
19     until M′ = ∅ ;
20     Mark u as split node
21     foreach v ∈ v_s do
22         if v_s ≠ ∅ then
23             select w ∈ v_s | F⃗_max = max_{w∈v_s} F⃗_{w,M_s} − E⃗_w
24             Check for other destination nodes in the direction of uw and verify
               whether they can also be addressed by the same neighbour, w
25             sendQuery(w, M_s)
26 end
```

Lemma 4.4.1. *Given a graph $G(V, E)$ that models the underlying network and a set of multicast destination nodes given in M, the subgraph T of G through which the multicast packet is transmitted using Algorithm 4.4 is a relatively minimal Steiner tree.*

Proof. By the definition of the virtual force in Equation 4.2 and Lemma 1 of [Liu & Wu 2009], the path between any two split nodes, the path between the split node and the destination node, and the path between the current node and the split node or the destination node, as the case may be, is the shortest.

The following two cases have to be considered for the proof.

Case 1. Branching does not happen in the current node.

From Line 13 to Line 15 of Algorithm 4.4 and wedge property defined in Section 8.2 of [Gilbert & Pollak 1968], the algorithm clearly tackles the case where no branching needs to take place. Hence, it is trivial to prove that Line 20 shall be reached only if there is a need for branching of the multicast tree.

Case 2. The current node is a branching node.

In Line 23, the algorithm chooses a neighbouring node $w \in N_u$ such that the virtual force of the corresponding multicast subset M_s exerted on it is the maximum. Hence, by definition, w is the closest neighbour towards the destination set, M_s. To address the case where more than one neighbour of the branch node can to be used to reach a multicast destination, Line 24 guarantees that only the closest neighbour with maximum force is used for the purpose.

Hence, T generated is relatively minimal. □

Theorem 4.4.1. *MRAV Algorithm (in Figure 4.4) is loop-free as per the underlying assumptions.*

Proof. There is a loop-free path between any two nodes that employ the virtual force technique[Liu & Wu 2009]. The only way, hence, for a loop to be formed is if two branch nodes forward the packet to a node in the network. That is, a path from a branch node u to a multicast destination d_1 and a path from a branch node v to a multicast destination d_2 intersected at some node in the graph. Such an intersection implies that there was a shorter path to d_2 from some $w_1 \in N_u$ and a shorter path from some $w_2 \in N_v$ to d_1. By definition of *P(M)* and by Lemma 4.4.1, this leads to a contradiction. Hence, proved. □

Theorem 4.4.2. *The time complexity for the Algorithm 4.4 is* $O(|T| \cdot \max\{2^m, \Delta\})$*, where* $|T|$ *is the total number of nodes in the resultant multicast tree,* $m = |M|$*,* Δ *is the maximum degree of a node in the network.*

Proof. The proof of this theorem is not presented here as it follows directly from Algorithm 4.4. □

In each node in the multicast tree, Algorithm 4.4 needs to verify whether branching must be done or not. For this, it must at least verify whether every subset of $P(M)$ can form a branch or not at each branching node, leading to an exponential algorithm. This is not suitable for large values of m.

4.5 Sector-based Virtual-force-based Multicast Routing Algorithm

In this section, we discuss the sector-based virtual-force-based multi-cast(VFM) routing algorithm that we have proposed in this work.

4.5.1 Motivation

Though MRAV uses virtual force technique to correctly identify the set of neighbours to whom the query message needs to be forwarded, the message complexity for making the computations was higher as we found during simulation in Section 4.7. One of the reasons for higher overhead and energy consumption derives from the fact that the algorithm compares all the nodes in the neighbourhood for making the routing decision. However, most of the nodes queried are not in the general direction of communication. The number of nodes that need to be enquired about the effective force values can hence be reduced. At each node u, we need to identify the subset of N_u that are least likely to be part of the multicast route.

The primary challenge while trying to decide which nodes can be excluded for path computation comes from the uncertainty regarding the rest of the network.

4.5.2 Basic Working

Unlike MRAV and [Rahman & Gregory 2011a], we divide the nodes around u into α sectors, with each sector covering an angle of $\theta = \frac{2 \cdot \pi}{\alpha}$. Here α varies from 3 to 6 depending on the density of the region containing u. For $\alpha = 4$, we have $\theta = \frac{\pi}{2}$. For each of the α sectors containing some destination nodes, we select the appropriate neighbour in that sector to forward the packet. We compute the effective force on each of the k neighbouring nodes $v_1, v_2, \ldots, v_k$ in the sector s, by taking into account the cumulative effect of dampening force (E_{v_i}), u and M_s, where $M_s \subseteq M$, M_s contains all the destination

nodes in sector s. Once the appropriate forwarding node v_s for the sector s is determined, the packet is communicated to v_s along with M_s, which is the set of destination nodes to be handled in sector s.

When a source node s is interested in transmitting a packet to a multicast group M, it first sends a *Query* message to establish the multicast route. This *Query* message contains a pointer to indicate the multicast group and the set of nodes in M represented using bit array. The algorithm for sending the *Query* message is given in Algorithm 4.5.

Figure 4.5: Virtual Force-based Multicast(VFM) routing algorithm

```
1  Procedure sendQuery(u, M)
2  begin
3      if u ∈ M then We are part of multicast group
4          M ← M − {u}
5      foreach v ∈ N_u ∧ v ∈ M do Our neighbor is part of M
6          Call sendQuery(v,M)
7      F_{u,M}^{→} ← calculateVirtualForce(u, M)
8      Divide region into α sectors centred at u, each of angle θ = 2·π/α such that first
       sector lies within ±θ/2 of F̂_{u,M}
9      foreach sector s do
10         v_s ← ∅
11         foreach v ∈ N_u do
12             if s = sector(v) then
13                 v_s ← v_s ∪ {v}
14
15     foreach sector s do
16         M_s ← ∅
17         foreach d ∈ M do
18             if s = sector(d) then
19                 M_s ← M_s ∪ {d}
20     foreach sector s such that M_s ≠ ∅ do
21         if v_s ≠ ∅ then
22             select w ∈ v_s | F_max^{⟶} = max_{w∈v_s} F_{w,M_s}^{⟶} − E_w^{→}
23             sendQuery(v, M_s)
24 end
```

When a node u decides on forwarding a packet to the destinations, it determines whether it has to split the multicast packet to different paths or not. The node determines whether all destinations are in the same general direction or not by computing the ef-

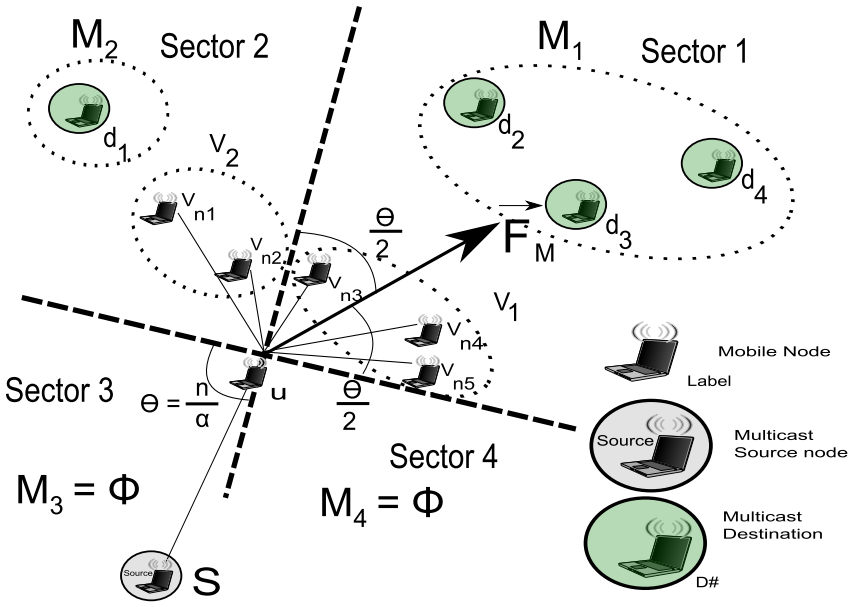

Figure 4.6: Virtual Force acting on node u. - $\alpha = 4$

fective force from destinations, $\overrightarrow{F_{u,M}}$. If $\overrightarrow{F_{u,M}}$ and $\overrightarrow{F_{s,u}}$ are in opposing directions, then the node determines the appropriate neighbour in the forward direction to forward the *Query* packet. The best way to determine whether all nodes are in the same general direction is to check whether the direction of force from each destination,$\overrightarrow{F_{d_i}}$, where $d_i \in M$, are within $\pm\frac{\theta}{2}$ angle from $\overrightarrow{F_{u,M}}$ as is indicated in Figure 4.6.

Figure 4.6 shows a simple illustration on how the force vectors are used in our algorithm. First, by using Equation 4.3, the effective repulsive force on the node u due to the destinations in multicast group M given as $\overrightarrow{F_{u,M}}$ is computed. The next step is to divide the region around the node u into α sectors. Here α is the split parameter that we use. For this purpose, first we look at the unit vector corresponding to the effective destination force, $\hat{\mathbf{F}_{\mathbf{u,M}}}$. From this direction, we take all nodes within angle $\pm\frac{\theta}{2}$ as the first sector, where $\theta = \frac{2\cdot\pi}{\alpha}$. In the example in Figure 4.6, $\alpha = 4$ and $\theta = \frac{\pi}{2}$. The nodes within the next θ radians form part of the second sector. Thus, a sectors will be marked around the current node u. The four sectors in Figure 4.6 are divided by the dotted lines as illustrated.

Now, we divide the neighbours of u into α sets $(V_1, \ldots V_\alpha)$, such that all neighbouring nodes in sector s are put inside the set v_s. We apply the same procedure to divide the nodes in M into $(M_1, \ldots, M_\alpha)$, where each set M_s indicates the set of destination nodes in the current destination set M. In the example in Figure 4.6, the set M is divided into

two sets $M_1 = d_1$ and $M_2 = \{d_2, d_3, d_4\}$ corresponding to sectors 1 and 2. Similarly, the neighbours of u are divided into two sets; $V_1 = \{v_{n1}, v_{n2}\}$ and $V_2 = \{v_{n3}, v_{n4}, v_{n5}\}$. Here, $\bigcup_i V_i = N_u$, where N_u is the set of neighbouring nodes of the current node u. Also, $\bigcup_i M_i = M$.

Once the subsets of M i.e., M_s has been computed, we can easily determine whether any sector is having destination or not by checking for the condition: $M_s = \emptyset$. If the condition holds true, then we don't have to explore that sector. If $M_s \neq \emptyset$, then we have to determine appropriate neighbor to forward the packet in that sector s.

To choose the appropriate neighbor in sector s, first we have to determine whether $v_s = \emptyset$. If so, then we have encountered a void in that sector. The simplest way to overcome void in the network is to follow the approach taken by Liu, *et al.* in [Liu & Wu 2009]. Virtual force is used to transmit across the void if possible, or else the perimeter rule of greedy perimeter stateless routing(GPSR) algorithm [Karp & Kung 2000] is applied.

If $v_s \neq \emptyset$, then the effective force on the packet as given in Equation 4.4 is computed for each neighbor in v_s. The neighbor $w \mid w \in v_s$ with the maximum force on the packet is chosen as the next hop neighbor.

4.5.3 Algorithm

The algorithm that uses the notion of sectors and virtual force, sector-based virtual force-based multicast routing algorithm (VFM), is shown in Algorithm in Figure 4.5.

4.5.4 Analysis

Lemma 4.5.1. *Given a graph $G(V, E)$ that models the underlying network and a set of multicast destination nodes given in M, the subgraph T of G through which the multicast packet is transmitted in VFM Algorithm(Figure 4.5) is a relatively minimal Steiner tree for the given value of α that is $(1 + \alpha)$-competitive with respect to the ideal solution. The ideal solution is a relatively minimal Steiner tree of length $len(T)$.*

Proof. The proof is similar to the one used in Lemma 4.4.1. When $\alpha = 3$, the algorithm mimics the wedge property described in Gilbert, *et al.* [Gilbert & Pollak 1968]. For $\alpha = 3$, the angle $\theta = 120^\circ$ which corresponds to a wedge described by Gilbert, *et al.*. Hence a

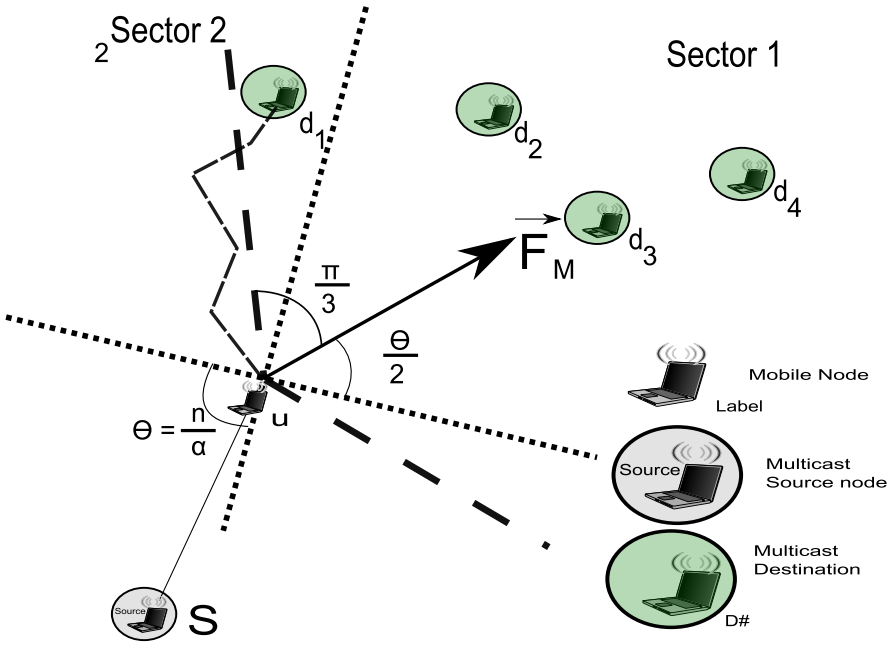

Figure 4.7: A sample scenario for $\alpha = 4$

relatively minimal Steiner tree is formed in this case. We now need to consider the proof for higher values of α.

For $\alpha = 4$, the forward sector is of angle $\pm\frac{\theta}{2}$ of the forward direction of $\overrightarrow{F_{u,M}}$. This sector captures the scenario of majority of nodes in the forward direction, in the event of low branching(i.e. for no branches, one branch or two branches). This scenario mimics the wedge property[Gilbert & Pollak 1968] closely, and hence the resultant graph will be relatively minimal. For the scenario where more than two branches are formed excluding the case of the source node, this implies that the sector in the front, right and left are having branches. Let us consider the scenario shown in Figure 4.7:

The destination node d_1 is just outside the forward sector, while all the other nodes are in the forward sector. As per the wedge property of [Gilbert & Pollak 1968], the node u should not be a branch node. However, as per the Algorithm 4.5, this scenario will result in branching for destination node d_1, with the path ud_1 being additionally added to the path for the multicast operation. If length of the relatively minimal Steiner Tree T for this graph is $len(T)$, it is trivial to observe that length of the path ud_1 is at most $len(T)$. Such a destination node can appear in all α sectors.

The argument can be generalized to higher values of α as well, with the number of

erroneous extra branches limited by α. Therefore the resultant tree will have a cost of at most $O((1+\alpha) \cdot len(T))$. Hence, proved. □

Theorem 4.5.1. *The VFM Algorithm 4.5 is loop-free as per the underlying assumptions.*

Proof. By Theorem 1 of Liu, *et al.* [Liu & Wu 2009] and Lemma 4.5.1, all paths between any two nodes in the graph is loop-free. To prove that the algorithm is loop-free, we need to prove that no two paths can intersect between the pairs of nodes (u, d_i) and (v, d_j), where u and v are branch nodes, and d_i and d_j are multicast destination nodes. If two such paths were to intersect, the only way it can happen is for d_i to be in v's sector and for d_j to be in u's sector in a branch node that is ancestor of u and v in the multicast tree. Since this cannot happen as per the loop in Lines 20 to 23 of the Algorithm in Figure 4.5, the intersection of the paths leads to contradiction. Hence, proved. □

Theorem 4.5.2. *The time complexity for the VFM Algorithm (in Figure 4.5) is* $O(|T| \cdot \alpha \cdot \max\{m, \frac{\Delta}{\alpha}\})$, *where* $|T|$ *is the total number of nodes in the resultant multicast tree,* $m = |M|$, Δ *is the maximum degree of a node in the network, and* α *is the number of sectors representing the branching factor.*

Proof. The proof of this theorem is not presented here as it follows directly from Algorithmin Figure 4.5. The dominating statements for the complexity computation are Lines 15 to 19 and Lines 20 to 23. □

From Theorem 4.5.2, if we consider the scenario of a graph with very low density, where the number of destination nodes is far grater than the maximum degree of a node in the network, then we get the Corollary 1.

Corollary 1. *The time complexity of Algorithm 4.5 for fixed value of* α *and for a sparse network is* $O(|T| \cdot m)$, *where* $|T|$ *is the total number of nodes in the resultant multicast tree,* $m = |M|$.

4.6 Virtual Multicast Tree

The Virtual Multicast Tree (VMT) routing algorithm uses a different approach to use the virtual force technique for multicasting. In this algorithm, we compute multicast trees among distant nodes using the virtual force technique. We use these pre-computed paths

to make most of the multicast routing decisions. In the remainder of this section, we introduce the basic concepts used in this algorithm in Section 4.6.1. Then we briefly describe the 2-MIS region creation mechanism in Section 4.6.2. The basic working of the algorithm is discussed in Section 4.6.3. The details of the algorithm, explaining the various scenarios to be considered is presented in Section 4.6.4. Possible optimizations for the algorithm in presented in Section 4.6.6.

4.6.1 Basic concepts

We had introduced the basic concepts in Sections 2.2, 2.4 and 4.3. In this section, we discuss the notion of Well Separated Planar Decomposition, and some related basics for the VMT algorithm.

4.6.1.1 Well Separated Pair Decomposition

In this algorithm, we use the concept of well-separated pair decomposition (WSPD) [Callahan & Kosaraju 1995]. Well-separated pair decomposition (WSPD) is a well known phenomenon used to deal with reaching long distance in t-optimal paths. When the source and destination is far apart, it is sufficient to get the path from the centre of the region containing source to the centre of the region containing the destination. The finer details can either be taken care by a recursive usage of WSPD, or by a direct route from the centre of the region having the actual node of interest. The concept of WSPD guarantees that the path derived using the method is going to be very close to the shortest path between source and destination, if the region containing the source and the region containing the destination are comparatively farther apart.

The notion of pre-computed paths have already been explored in two unicast routing variants, GCRP[Fotopoulou-Prigipa & McDonald 2004], and SWING [Liu & Wu 2006] and SWING+ [Liu & Wu 2009]. In GCRP, geo-circuits used are pre-computed paths handled using geo-tables, an extension of routing tables in which the geo-circuit details such as circuit-id and next hop are stored. In SWING and SWING+, virtual logical links (VLLs) are stored in a similar fashion. VLLs are also pre-computed paths towards specific directions from the current node. In our VMT algorithm, we use virtual force technique

to compute the paths between various geographically distant, connected sub-networks.

4.6.1.2 Basic idea

We use the notion of maximal independent sets (MIS) to generate well separated regions. To recap, MIS is a set of dominators such that every node in the network is either an MIS node or has at least one MIS node in its neighbourhood set, N_u[Calinescu 2003]. Instead of using directly the MIS nodes, where each MIS node may cover a maximum diameter of 2 hops, we extend the notion of MIS to generate a second level of MIS nodes based on the initial set of MIS nodes determined. Thus, 2-MIS regions are an extension of 1-MIS regions. [Alzoubi *et al.* 2002a] has suggested a simple algorithm to compute MIS nodes. The MIS creation algorithm was explained in Section 2.2.6.1.

For transmitting data from a source node to a destination node, it is sufficient to communicate to the nearest 2-MIS node. That 2-MIS node in turn will route the packet to the distant node using the concepts of WSPD. The communication between source node and its related 2-MIS node as well as the communication between destination node and its related 2-MIS node, can be done directly. By using the concept similar to geocircuits, we can have pre-computed paths among the centre points of these big regions. From the set of random motion graphs we generated, we had observed that there are usually a few regions where the density of nodes tends to be higher than the average density of the network.

4.6.2 2-MIS Region creation

In our algorithm, we are broadly dividing the nodes into regions corresponding to MIS and its dominatees. After computing the 1-MIS nodes, we model an auxiliary graph, H, that is induced by 1-MIS nodes. Edges in H indicate that two 1-MIS nodes are logically connected with one another. We then construct the 2-MIS nodes based on the vertices and edges in this auxiliary graph H to generate the 2-MIS dominators, 2-MIS dominatees and 2-MIS connectors. It is to be noted here that all 2-MIS connector nodes and 2-MIS dominatee nodes are themselves 1-MIS nodes.

4.6.3 Basic Working

In this algorithm, we first construct the 2-MIS as mentioned in the previous section. While constructing the 2-MIS, the 1-MIS nodes shall include their two-hop neighbourhood lists. Thus the 2-MIS dominator node is aware of four-hop neighbourhood information. We thus have three types of nodes, 1-MIS dominatee, 1-MIS dominator and 2-MIS dominator. The routing decisions made by these three types of nodes are different.

As far as a 1-MIS dominatee node is concerned, the task of routing is very simple. It checks whether it itself is one of the multicast destination. If so, it removes itself from the multicast destinations list. It then forwards the multicast query to its 1-MIS dominator.

A 1-MIS dominator node could be a 2-MIS dominatee or 2-MIS connector. (2-MIS dominator forms the third type to be discussed.) A simple manner for routing in the 1-MIS dominator is to forward the request to its 2-MIS dominator node. Since the 1-MIS dominator is aware of its 2-hop neighbourhood information, it shall forward the query directly to the destinations within its 2-hop neighbourhood. All remaining destinations in the multicast destination list shall be handled by the 2-MIS dominator. Hence, it forwards the remaining multicast set to its 2-MIS dominator.

A 2-MIS dominator node is aware of four hop neighbourhood information. All the nodes in the multicast set that are in its four-hop neighbourhood are marked for sending the packet directly. It then applies the VFM algorithm, that was discussed in Section 4.5, on the auxiliary graph H of the network, and forwards the query to other 2-MIS dominators.

When a 2-MIS dominator recieves a query request, it decides whether a node is within its 4-hop neighbourhood region or not. Based on that it decides whether to further apply VFM or to forward the request to the node. If a destination node in the multicast group is within its region, it forwards the query either directly or via a 2-MIS dominatee or 2-MIS connector node as the case may be.

In this basic working of the algorithm explained, we have not mentioned any of the optimizations to be performed to deliver the packet more efficiently. The optimizations are discussed later in Section 4.6.6.

4.6.4 Virtual Multicast Tree Algorithm

Apart from the data structures used in VFM algorithm as mentioned in Section 4.4.2, we maintain the following additional lists:

N_u^1 : List of 1-hop neighbours of the current node. Note that this list was named as N_u in the previous algorithms.

N_u^2 : List of 2-hop neighbours of the current node.

N_u^3 : List of 3-hop neighbours of the current node.

N_u^4 : List of 4-hop neighbours of the current node.

The simple algorithm for VMT, without any optimizations, is presented in Algorithm 4.8. In the algorithm presented, we are assuming that the following methods are already available:

state(u) : This function returns the state of a node as 1-MIS dominatee, 1-MIS dominator and 2-MIS dominatee.

dominator(u): This returns the 1-MIS dominator for a 1-MIS dominatee or a 1-MIS connector node. For a 2-MIS dominatee or a 2-MIS dominator node, this function returns the 2-MIS dominator for the node u.

sendVFMQuery(M): This applies the VFM algorithm on the 2-MIS nodes alone and forwards the query accordingly.

4.6.5 Analysis

In the Virtual-force Multicast Tree(VMT) algorithm presented in 4.8, once the multicast packet reaches the 2-MIS dominator of the source node, the packet is forwarded to the destination nodes' 2-MIS dominator using the VFM algorithm discussed in Section 4.5.

Lemma 4.6.1. *Given a graph $G(V, E)$ that models the underlying network and a set of multicast destination nodes given in M with m nodes, the subgraph T of G through which the multicast packet is transmitted in Algorithm 4.8 is a relatively minimal Steiner tree for the given value of α that is $(1 + \alpha + \frac{m}{len(T)})$-competitive with respect to the ideal solution.*

Proof. By Lemma 4.5.1, the multicast tree formed between the 2-MIS dominator nodes of the source and the multicast destination nodes is relatively minimal with an approximation factor of $(\alpha \cdot len(T))$. Hence for proving this lemma, it is sufficient to prove that the

Figure 4.8: Virtual Multicast Tree(VMT) algorithm

```
/* sendQuery :  Sending the query packet from node u          */
1  begin
2      if u ∈ M then //We are part of multicast group
3          M ← M − {u}
4      if state(u) = 1 − MISdominatee then
5          sendQuery( dominator(u), M)
6          exit
7      foreach v ∈ N_u^1 ∧ v ∈ M do //Our neighbor is part of M
8          Call sendQuery(v,M)
9          M ← M − {v}
10     if state(u) = 1 − MISdominator then
11         sendQuery( dominator(u), M)
12         exit
13     if state(u) = 2 − MISdominator then
14         foreach (v ∈ N_u^2 ∪ N_u^3 ∪ N_u^4) ∧ (v ∈ M) do
15             Call sendQuery(v,M)
16             M ← M − {v}
17         sendVFMQuery(M)
18 end
```

additional hops required for the source node to multicast the destination does not take more than $O(m)$. □

Theorem 4.6.1. *Algorithm 4.5 is loop-free as per the underlying assumptions.*

Proof. From Theorem 4.5.1 and the concept of WSPD explained in Section 4.6.1.1, the algorithm is loop-free. □

Theorem 4.6.2. *The time complexity for the Algorithm 4.8 is* $O(|T| \cdot \alpha \cdot \max\{m, \frac{\Delta}{\alpha}\} + len(T) \cdot m)$*, where* $|T|$ *is the total number of nodes in the resultant multicast tree,* $m = |M|$*,* Δ *is the maximum degree of a node in the network, and* α *is the number of sectors representing the branching factor.*

Proof. The proof uses Theorem 4.5.1 and the concept of WSPD explained in Section 4.6.1.1. The proof is trivial and is omitted. □

With path optimizations suggested in Section 4.6.6 and for a fixed value of α and by assuming a dense network, the time complexity mentioned in Theorem 4.6.1 can be brought down to $O(|T| \cdot \Delta)$.

4.6.6 Optimizations of the algorithm

A number of optimizations can be performed to streamline the operations and forward the query to the destination.

Proactive Cluster-Based Distance Vector (PCDV) Protocol [Hoon & Seok-Yeol 2007] had suggested path optimizations for routing packets, whenever a path computed is wider than the shortest path available. In PCDV, when an intermediate node is aware of a shorter path towards the destination, it shall alter the path accordingly for future traffic from the source node to the destination. In our algorithm, we use a similar strategy to optimize the path whenever an intermediate node is aware of a better path towards the destination or the next split node. The algorithm remains loop-free after this optimization.

It is also possible to perform other optimizations. We are listing a few of them here.

In the algorithm presented, we stated that the 1-MIS dominator node forwards all queries that it cannot service itself to the 2-MIS dominator node. The 1-MIS node can apply VFM at this juncture itself by considering only its 2-MIS neighbours (neighbours in the auxiliary graph) and forward the packets accordingly.

In a 2-MIS region, when a query is received by a 2-MIS connector that is part of the region, the 2-MIS connector node can decide for itself whether the query needs to pass through the 2-MIS dominator node, or it can be directly transmitted to another 2-MIS connector within the same region.

4.7 Simulation Results

We have performed simulation of our algorithm using a simulator written in Python language that is based on the code base of ns-2 [Issariyakul & Hossain 2011]. The simulator uses the same energy model and Two Ray propagation model as implemented in ns-2[Issariyakul & Hossain 2011]. For the simulation runs, we have placed nodes in a fixed area of $2000m \times 2000m$ with maximum transmission range set as $250m$. We have compared MRAV, VFM and VMT with the QBIECRA algorithm proposed by [Rahman & Gregory 2011b]. While comparing with the sector-based VFM algorithm, we have taken α values as 3 and 4. We didn't consider higher values of α as it was counter-intuitive to the principle proposed by [Gilbert & Pollak 1968]. We have chosen QBIECRA since it

uses a notion of quadrants for computing the multicast tree which is comparable with our approach of using sectors. Though Rahman, *et al.* also proposed another version of their quadrant based algorithm in [Rahman & Gregory 2011a], while performing simulations we didn't perceive a major difference between the results of QBIECRA and 4-N Intelligent routing. This may be due to the fact that these algorithms only differ on the basis of which four neighbours are going to be chosen for computing the next forwarding node in the multicast path. The simulation results obtained are discussed in the rest of this section. All values were within 95% confidence interval.

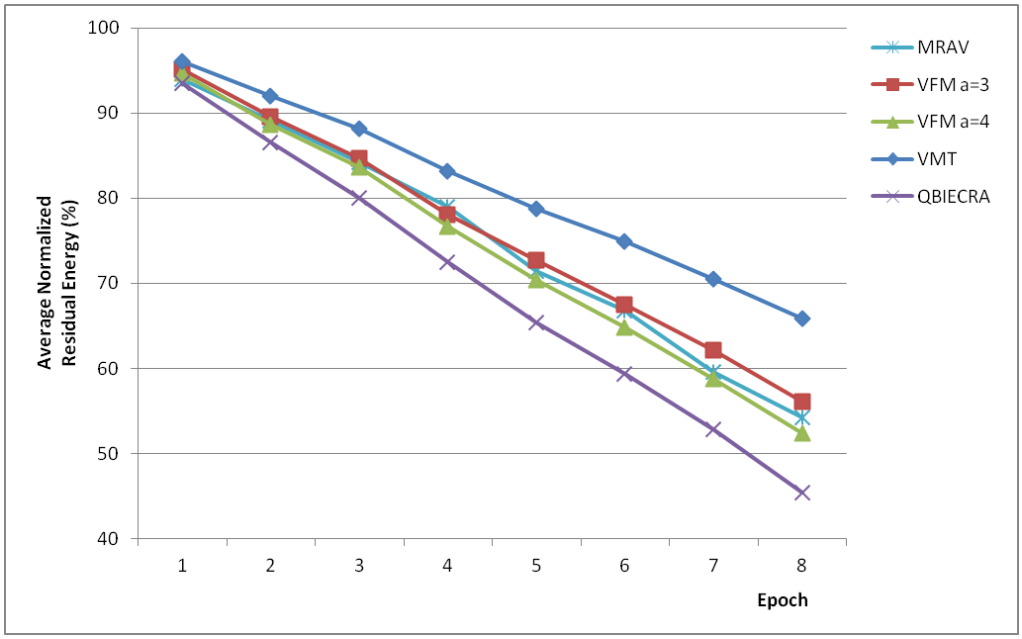

Figure 4.9: Average normalized residual energy over time -

Average normalized residual energy is computed as the average residual energy of all nodes in the network normalized in percentage terms to account for variations in initial energy values of various nodes. We have computed the average of residual energy in each of the nodes after completing one cycle of packet transmissions for a sender node in each epoch, and are showing the normalized value in percentage terms after each epoch. The results for normalized average residual energy for MRAV, VFM with $\alpha = 3$ and $\alpha = 4$, VMT and QBIECRA are shown in Figure 4.9. We observed that VFM for $\alpha = 3$ and $\alpha = 4$, MRAV and VMT performed better than QBIECRA. In seven rounds(epochs) itself, the

residual energy for QBIECRA had dropped below 50% of battery level. The results of VFM and MRAV were almost similar as far as average residual energy is concerned. The data indicated that the effective number of nodes that were participating in the decision making process was almost similar. These two algorithms performed better than QBIECRA as per the figure. This was because of the difference in the choices of split nodes. VMT performed much better than other algorithms. This is because of the fact that the decision making nodes were much limited with VMT, with most of the nodes only forwarding the packets through pre-determined virtual circuits. Data used to pass through the nodes with more energy level more often due to the use of virtual circuits. Hence, VMT appears better than MRAV and VFM ($\alpha = 3$ and $\alpha = 4$), which in turn performed better than QBIECRA.

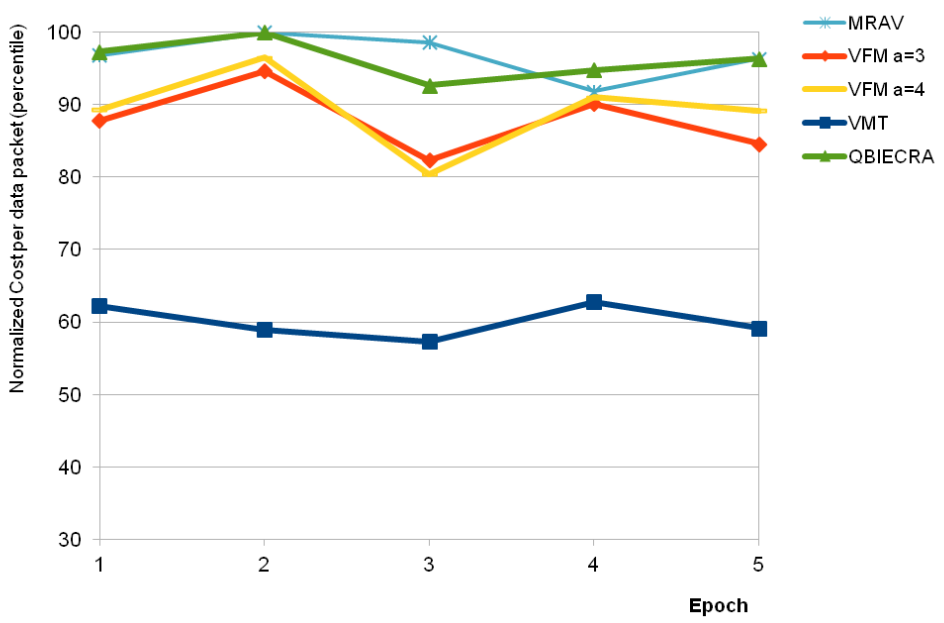

Figure 4.10: Normalized cost per hop per packet over time -

Normalized cost per data packet is computed as the overall cost for sending a data packet from source node to destinations that is normalized in terms of percentile energy cost for the sake of comparison across multiple simulation runs with varying network sizes. The results for this metric for MRAV, VFM with $\alpha = 3$ and $\alpha = 4$, VMT and QBIECRA are shown in Figure 4.10. We observed that performance of MRAV and

QBIECRA were almost similar. The effective number of data retransmissions for both the algorithms turned out to be approximately similar. The normalized cost per data packet for the two versions of VFM performed better than MRAV and QBIECRA primarily by optimizing the effective energy spent on optimal path computation. VMT performed much better than other algorithms. Though the effective length of the multicast path was a bit larger than other algorithms, VMT was optimally forwarding the data packet through well-established paths thereby reducing the path computation penalties that were incurred by the other algorithms. Also, the split nodes were determined by the 2-MIS nodes which were having a more holistic view of the network topology. Thus, on an average, the effective number of data forwards on a per data basis was smaller for VMT as compared to the other three algorithms.

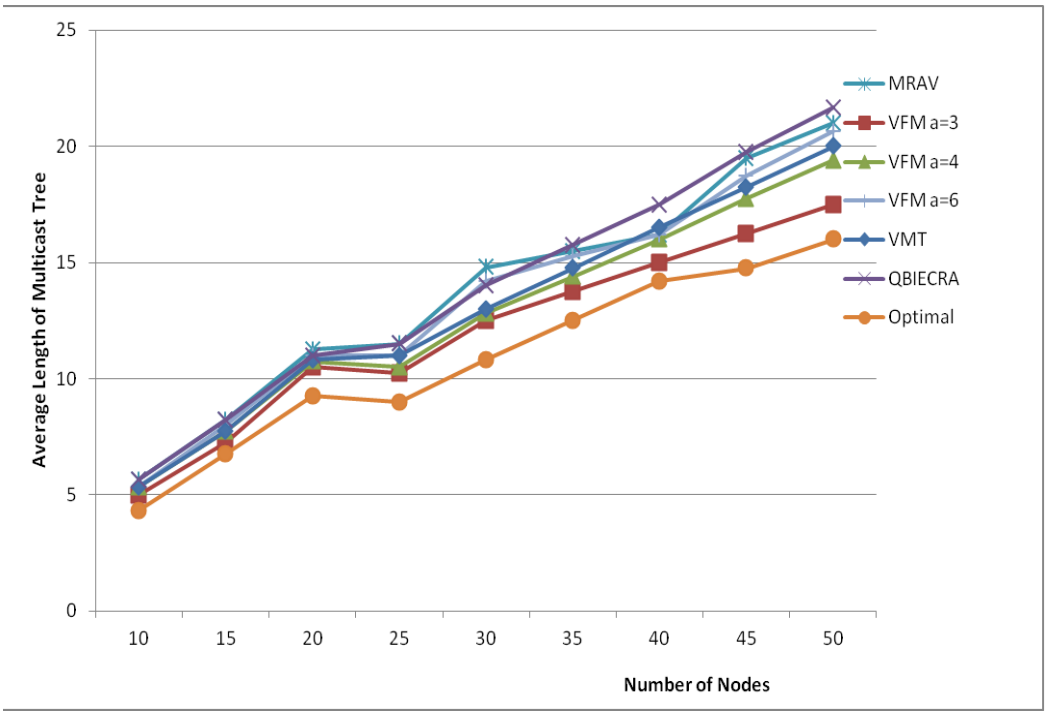

Figure 4.11: Average length of multicast tree with respect to number of nodes -

We also observed the effective length of the multicast tree generated as shown in Figure 4.11 by comparing the results of QBIECRA, MRAV, VFM ($\alpha = 3, 4$ and 6), VMT and optimal algorithm. Here, the optimal algorithm was a branch and bound algorithm that looked at all possibilities for determining the best Steiner nodes for the given graph. For evaluating this metric, we had computed the paths after the 10^{th} data packet was sent

for all the algorithms. We had also disabled mobility for this experiment. We observed that 120*degree* is the best angle between the branches for getting relatively minimal Steiner trees as defined by [Gilbert & Pollak 1968]. As expected, the length of the multicast tree was nearing the optimal values for VFM with $\alpha = 3$. We surprisingly found that we were getting good results for VFM with $\alpha = 4$ as well. This was primarily due to the fact that we were not having many out-going edges in the opposite direction of the forward path. Due to the static manner in which quadrants are established, we observed a lot of unnecessary splits in QBIECRA as compared to VFM with $\alpha = 4$. When it came to $a = 6$, VFM algorithm performed worse as expected, as we were generating too many out-going edges in some nodes resulting in a sub-optimal multicast tree. However, the performance of the algorithm was still better than QBIECRA as seen from the figure. VMT performed worse than VFM for $\alpha = 3$ and $\alpha = 4$. This is because VMT relies on WSPD for computing the source to destination path. So even if there is a shorter path from source to destination, the data will first move towards the 2-MIS dominator node, which in turn would forward the packet to the destination 2-MIS node, which used to then forward the packet to individual destinations. We observed many instances where the packet was moving in opposite direction to the destination node for one or two hops before moving towards the destination node. We observed a variation in the general trend of length of the multicast tree for number of nodes $= 20$. We attribute this variation to the choice of multicast set, presence of voids and other aspects specific to the test graphs used for simulation. Even with this variation, our algorithms provided shorter length for multicast tree as compared to QBIECRA.

4.8 Conclusion

We have applied the notion of virtual force for energy efficient multicast routing in MANETs. We have presented three algorithms centred on the virtual-force technique. Our algorithms generate relatively minimal Steiner trees for use in multicast routing. The simulation results indicate that our algorithms perform better than other quadrant/sector based multicast routing algorithm. In the first algorithm, we applied our new definition for virtual force. This algorithm proved that the virtual force technique can be used for

multicast routing. In the second algorithm, we used the notion of sectors to the virtual force technique. We have evaluated appropriate choice for the number of sectors to be used, and have found that $\alpha = 3$ or $\alpha = 4$ can be used to generate energy efficient multicast paths. We observed that the sector-based virtual force approach as a whole provides good results for multicast routing. We haven't yet explored the relationship between the value of α and network density and spread. However, we believe that there is still scope in fine tuning the force model used. Though we had only used life of the network for QoS, the dampening force can easily be extended to include other parameters such as bandwidth as well. In the third approach, we have used the notion of regions and applied the virtual force touring algorithms amongst the regions. We defined regions with four-hop radius for the algorithm. We adapted the ideas of spine routing, well separated planar decomposition and virtual geographic circuits to enable routing among the regions using the virtual force technique. The proposed algorithms have shown that the virtual force approach can be successfully used in MANETs. To our knowledge, we are the first to have worked on multicast routing algorithm using the virtual force approach in mobile ad-hoc networks. This is also the first time virtual-force computation is performed on the basis of sectors and regions.

Chapter 5

Distributed Mutual Exclusion in Mobile Ad-hoc Networks

Resource allocation is one of the crucial aspects of any large network. Due to the naturally distributed manner of operation of the nodes in a Mobile Ad-hoc Network(MANET), any such mechanism shall use a distributed algorithm for its operations. Within the ambit of resource allocation, we address the specific problem of distributed mutual exclusion(DME) for accessing critical resources in the network, even in the event of changes in topology. In the DME problem the objective could be to minimize the number of messages used and the number of hops traversed by the messages. Apart from the primary constraint of holding the mutual exclusion property, the objective of making the DME algorithm more resource efficient need to be maintained as well. In this chapter, we have explored the permission-based approach to solve the DME problem. We present a node number initialization mechanism for the DME algorithms that follow the look-ahead technique of [Singhal & Manivannan 1997]. We follow it up with two permission-based DME algorithms for MANETs.

In the first algorithm presented in this chapter, we present a novel permission-based algorithm for solving the DME problem that can handle site failures. Our algorithm introduces a new message called *HOLD* message to ensure that the requesting nodes are aware of the node currently executing in the critical section(CS). Unlike the earlier algorithms, that use predetermined static timeouts for handling fault tolerance, we use an

adaptable timeout mechanism so that we can deal with critical sections having varying execution times. The algorithm could handle situations where the node in CS itself can fail, with the help of the *HOLD* message and the adaptive timeout mechanism. In the second algorithm presented in this chapter, we divide the network into regions of fixed diameter. By suitably manipulating the behaviour of arbitrator nodes, that form the bridge between two neighbouring regions, we have ensured that permission is granted by every participating node, irrespective of the size of the network. We have used a single additional message, the *HOLD* message, to ensure that correctness for the DME algorithm is achieved for both inter-region and intra-region communications. Fault tolerance arguments for the proposed algorithm are also presented. To our knowledge, this is the first distributed mutual exclusion algorithm that uses the notion of regions and fault tolerance in MANETs.

The remainder of this chapter is organized as follows: We provide an introduction to the DME problem in Section 5.1. We discuss the mutual exclusion problem and background relevant for this chapter in that section. In Section 5.2, we describe the Dynamic Information Set used in the "look-ahead" technique. We discuss the notations used, and the rules defined for handling the Dynamic Information Set. In Section 5.3, we present an initialization mechanism for the Dynamic Information Set. We provide proof for the new initialization mechanism in that section. In Sections 5.4 and 5.5, we present two different permission-based DME algorithms proposed by us. We conclude the chapter in Section 5.6.

5.1 Introduction

We had defined the basic terminology of the DME problem in Section 1.10.2. We had discussed some of the relevant literature in Section 2.5.

5.1.1 The Mutual Exclusion Problem

For the mutual exclusion problem, we need to guarantee that no two operations in critical section are concurrent. Apart from the usual requirement of absence of deadlocks, any solution for the mutual exclusion problem must ensure that the fairness property is satis-

fied. By fairness, it is meant that any task wishing to enter CS shall not wait indefinitely to enter into CS. A stronger version of fairness property is to mandate that any process P_i, that had attempted to enter CS before any other process P_j, must enter its CS before P_j does.

The stricter fairness requirement mandating absolute First Come First Serve(FCFS) behaviour may not be feasible in all distributed environments, especially when faced with synchronizing a global clock in a resource constrained distributed system[Wang 1992]. In traditional mutual exclusion, requests for resources are serviced in the order of their arrival. Although FCFS is a natural policy for many applications, there may be circumstances where serving out of order might be sufficient with respect to the constraints of the network. It may be mandated for MANETs that from the moment a node becomes aware of any shared-resource-related communication, a certain logical ordering be maintained.

In most implementations, it is usually assumed that a process running in a particular site is going to cycle between non-critical sections of code and critical sections of code till it reaches its conclusion. It is also safe to assume that every process will have at least a small non-critical section code prior to entering the CS for the first time, and there shall be some continuation after leaving the CS. Thus, we mean that no participating process shall abruptly halt while it is executing in its CS[Lamport 1986]. A process may, however, decide to *abort* from the execution of the mutual exclusion algorithm at any point of time. Any such abort operation will bring its execution state outside the CS, and the process shall reset the related variables of the algorithm appropriately. Our algorithm can handle such a voluntary discontinuation of a process in CS.

While executing, there is a possibility of the following faults. Firstly, there is a possibility of an unannounced death, which happens when a process halts abruptly and without being detected. Secondly, a process may malfunction, in which case it keeps setting its variables to arbitrary values. A malfunctioning process may enter CS even while another process is in CS. In the event of a transient fault, we may need to ensure that the process will execute normally after the issues leading to its malfunction are rectified. While these two are the standard faults that can be expected in any environment, we need to consider transient loss of communication in the case of MANETs and the likelihood of a node forcing itself to abort, say if its residual energy drops below a particular threshold.

It is safe to assume that a node will know a priori when it is forced to abort an execution due to power constraints. However, a transient failure due to loss of communication to its neighbouring nodes may not be predictable. Any such fault during the execution in CS may lead to a scenario where other processes are indefinitely waiting for the faulty node to indicate that it had come out of the CS. Guaranteeing reliability in a mobile environment is hence a challenging problem. The most we may require from a solution to the DME problem in the MANET environment is that whenever a node is reconnected to the network after a disconnection, or when a node wakes up after forcing itself to go into doze mode, the system shall resume its correct operation within a finite period of time.

In a mobile environment, there is one more constraint that must be satisfied: combinatorial stability. While a node is waiting to decide whether it can enter CS or not, the underlying topology of the network may change due to high churn in the network. The running time of the algorithm must be fast enough to guarantee that the change in network state does not affect the current objectives of the algorithm. The notion of combinatorial stability is especially important for algorithms like token-based DME algorithms ([Walter *et al.* 2001, Chen & Welch 2002]) which include initialization routines that need to communicate to the entire network. Even permission-based algorithms like the one in [Wu *et al.* 2008] have self-stabilizing initialization steps. In this chapter, we discuss an initialization routine that satisfies combinatorial stability.

5.1.2 Background

Wu, *et al.* in [Wu *et al.* 2008] extended the algorithm in [Singhal & Manivannan 1997] to accommodate mobile nodes. They introduced a randomized initialization procedure to initialize the node numbers of the nodes participating in DME. They also introduced three additional messages namely, *DOZE*, *DISCONNECT* and *RECONNECT*. *DOZE* message is used to indicate that a mobile node is going to a sleep mode to save power. *DISCONNECT* and *RECONNECT* messages were used to allow a mobile node to move out of communication range and return to the range. To deal with fault-tolerance, they suggested the use of a timeout vector for *REQUEST* messages, TO_{REQ}, which is maintained for the *REQUEST* messages sent from the requesting site. If the TO^i_{REQ} expires,

then the $REQUEST$ message is resent to the site S_i.

[Erciyes 2004] had mapped the Ricart-Agrawala algorithm onto MANETs. Instead of directly using the Ricart-Agrawala algorithm on all the nodes of the MANET, the *RA_Ring* algorithm divides nodes into a set of logical coordinators that form a ring of nodes. These coordinators then use a modified Ricart-Agrawala technique to deal with the DME problem. Here, the message complexity is $O(k+3)$, where k is the number of coordinators. Synchronization delay varies from $2T$ to $(k+1)T$, where T is the communication delay. [Erciyes & Dagdeviren 2012] have used a weighted partitioning scheme described in [Dagdeviren *et al.* 2005] for creating clusters. However, the algorithm in [Dagdeviren *et al.* 2005] expects the number of partitions to be created as a parameter, which is not tenable in practical applications. In the process of trying to extend the Ricart-Agrawala algorithm with the help of clustering, unfortunately, variants of the $RELEASE$ message had to be brought back both in intra-cluster and inter-cluster communications.

The current set of approaches for permission based algorithms thus focus on reducing message complexity either by focusing on trying to reduce the effective number of participating nodes or by resorting to clustering. The clustering solution, however, would still perform worse than the classic Maekawa's algorithm due to the fact that communication overhead between cluster-heads is still relatively higher, especially in the case of large networks in which the participating nodes are spread out. While trying to divide the current node into clusters, the current set of algorithms are using only one-hop neighbourhood for cluster creation. The cluster's periphery nodes transmit many of the messages, but they never make any decisions. This leads to overburdening of the cluster-heads. Since such hierarchical approaches look at inter-cluster and intra-cluster communication as two different sub-problems, they resort to a distributed version for handling DME among clusters, while resorting to a centralized algorithm with the cluster-head acting as the lone decision-making node within the cluster. There is a high computational as well as communication load on the cluster-heads, as they have to handle both cluster management and DME algorithm. In our proposed algorithm, we relax the roles assigned by the nodes for region management and handling DME, thereby distributing the load to nodes in the periphery of the region as well.

5.2 The Dynamic Look-Ahead Technique

A brief description of dynamic look-ahead technique defined in [Singhal & Manivannan 1997] is described in this section. We are reproducing the major contributions of [Singhal & Manivannan 1997] here, as the algorithm for the basis of [Wu *et al.* 2008] and the algorithms presented in this chapter.

Singhal, *et al.*'s approach relied on the fact that not all sites(mobile nodes forming the MANET, in our case) are actually interested in participating in the CS decision making. A concurrency set involving a small subset of the nodes in the network are interested in entering CS at any given point of time. Only these nodes need to be consulted for requesting and getting granted permission to enter CS. We first discuss the notations used in [Singhal & Manivannan 1997], and then reiterate the rules presented by them.

5.2.1 Notation

We shall use the following notations in the rest of this chapter.

$S = S_1, S_2, S_3, \ldots, S_{n-1}, S_n$: The set of all sites.

$Info\_set_i$: Set of ids of the nodes to whom S_i must send the $REQUEST$ message to enter CS.

$Status\_set_i$: Set of ids of the nodes from whom $REQUEST$ message shall be recieved before they could enter CS, and to whom S_i must send the $REPLY$ message to allow them to enter CS.

5.2.2 Rules

Rule 5.2.1. ***(Construction of Sets):***

I. $(\forall S_i :: Info\_set_i \cup Status\_set_i = S)$

II. $(\forall S_i \forall S_j :: S_i \in Info\_set_j \Rightarrow S_j \in Status\_set_i)$

For sending minimum number of messages, the condition $Info\_set_i \cap Status\_set_i = \emptyset$ needs to be satisfied. If $Info\_set_i \cap Status\_set_i = T \neq \emptyset$, then there shall be redundant messages transmitted to the nodes $S_j \in T$. In such a case, S_j informs S_i as well as gets informed by S_i. In their effort to minimize the number of messages communicated, DME algorithms presented in both [Singhal & Manivannan 1997] and [Wu *et al.* 2008] ensured that the minimality condition was satisfied. We had deviated from these algorithms in this basic condition as shall be evident in the later part of this chapter.

Corollary 2. $(\forall S_i \forall S_j :: \neg(S_i \in Info\_set_j) \Rightarrow S_j \in Info\_set_i)$

If the node S_j need not inform S_i when it enters CS, then S_j is certain that S_i will ask its permission before entering CS. That means that $S_j \in Info\_set_i$.

Rule 5.2.2. ***(Handling Requests):***

When S_i receives a REQUEST message from S_j, it sends a REQUEST message to S_j provided S_i itself is requesting CS at that time and $S_j \in Status\_set_i$

As per Rule 5.2.2, S_i moves the entry of S_j from $Status\_set_i$ to $Info\_set_i$.

Rules 5.2.1 and 5.2.2 state how the data structures related to the lookahead technique get manipulated. The reminder of the rules(5.2.3 to 5.2.5) relate to the operation of the algorithm by Singhal, *et al.*

Rule 5.2.3. ***(Handling Requests):***

A site S_i sends a REPLY message in response to a REQUEST message from S_j, if S_i itself is not requesting CS or if S_i's request has a lower priority than S_j's request.

To determine priority, the priority rule stated by [Ricart & Agrawala 1981] is used. Logical clocks are used to keep track of age of request messages. Lower logical clock value implies an older message and a higher logical clock value implies a newer one. If the logical clock values of both the requests are same, then the tie is resolved by assigning a higher priority to the node with the lower node id.

Rule 5.2.4. ***(Executing CS):***

A site S_i executes CS only after it has received a REPLY message for every REQUEST message it sent out.

From this rule and following Rule 5.2.2, we can infer the following Corollary at the moment the *REPLY* message is received.

Corollary 3. $(\exists S_i \exists S_j :: (S_i \in Info\_set_j) \wedge (S_j \in Info\_set_i)$

This Corollary leads us to the next rule.

Rule 5.2.5. ***(Handling Reply message):***

After a site S_i receives a REPLY message from S_j, S_i moves S_j's entry from $Info\_set_i$ to $Status\_set_i$.

This is to ensure that S_j does not enter CS without getting permission from S_i which is on the verge of entering CS.

Rule 5.2.6. ***(Exiting CS):***

On exiting its CS, a site S_i sends REPLY message to all sites in $Info\_set_i$.

5.2.3 Wu, *et al.*'s improvements to Dynamic Information Set

While the algorithm by [Singhal & Manivannan 1997] is meant for a cellular network, [Wu *et al.* 2008] proposed the first algorithm to use "look-ahead" technique in MANETs. For this, they came up with an initialization algorithm to ensure that the concurrency set converges. To incorporate fault-tolerance in the event of transient node or link failures, they added three new messages: *DOZE*, *DISCONNECT* and *RECONNECT*. A voluntary disconnection, defined as the node intentionally disconnects as it's battery level is below a threshold, leads a node to transmission of *DOZE* message, while involuntary disconnection, characterized by transient link failures, lead to transmission of *DISCONNECT* message. In either case, the node S_i sets $Info\_set_i = S$ and $Status\_set_i = \emptyset$.

5.3 Initialization of nodes for the DME algorithm

The initial work on "lookahead technique" by [Singhal & Manivannan 1997] provided certain suggestions for initialization of their sets. However, these were made under the context of a cellular network, where the sets could converge with relative ease guided by the mobile support stations.

[Wu *et al.* 2008] suggested that an initiator node elected by the participating nodes interested in entering CS can broadcast a table indicating whether a node will be part of $Status\_set$ or $Info\_set$.

5.3.1 Awasti's Initialization Scheme

In [Awasthi 2006], a different scheme for initialization was suggested. Whenever a site S_i needs to initialize, it follows this rule:

Rule 5.3.1. $Info\_set_i = \{S_j \mid 1 \leq j \leq i-1 and S_j \in S\}$
$Status\_set_i = \{S_i\}$

For example, the $Info\_set_4$ for the site S_4 shall have $\{S_1, S_2, S_3\}$. After initialization, a site S_i needs to send $REQUEST$ message to sites in $Info\_set_i$, i.e. all sites with a lower id than i.

To prove the validity of Rule 5.3.1, it is sufficient to prove that the invariant in Corollary 2 is satisfied.

Lemma 5.3.1. *Initializing the dynamic sets using Rule 5.3.1 satisfies the condition in Corollary 2.*

Proof. The proof is by construction.
When a site S_i sends a $REQUEST$ to a site S_j ($i > j$ by definition) and S_j is not participating in CS or not requesting to enter CS, by Rule 5.2.2, S_i will be added to $Inform\_set_j$.

When a site S_i sends a $REQUEST$ to a site S_j ($i > j$ by definition) and S_j has requested to enter CS, S_j's request cannot be older than S_i's request as that would mean that S_i had communicated before leading to a contradiction. Hence by Rule 5.2.3, S_j notices that a site with higher priority has requested to enter CS. S_i will be added to $Inform\_set_j$.

When a site S_i sends a $REQUEST$ to a site S_j ($i > j$ by definition) and S_j is in CS, S_i will be added to $Inform\_set_j$ so that it can be informed when S_j exits CS.

In all possible scenarios, an appropriate entry for S_i shall be made in all the sites which didn't have its entry. Once all sites have made their requests, the dynamic information sets converge, and Corollary 2 gets satisfied. □

Theorem 5.3.1. *The initialization mechanism suggested satisfies combinatorial stability.*

Proof. The proof for the theorem is trivial as there is no requirement of a centralized node in this initialization routine. □

5.4 A Novel Permission-based Reliable Distributed Mutual Exclusion Algorithm for MANETs

In this section, we discuss a permission-based algorithm for solving the DME problem that can handle site failures. Our approach introduces a new message called *HOLD* message to ensure that the requesting nodes are aware of the node currently executing in the critical section. This algorithm doesn't predetermine the timeouts as static values as done in [Wu *et al.* 2008]. We use an adaptable timeout mechanism so that we can deal with critical sections having varying execution times. This algorithm that can handle situations where the node in critical section itself can fail, with the help of the *HOLD* message and the adaptive timeout mechanism.

5.4.1 Introduction and Motivation

In permission-based approach, the most common way to take care of fault-tolerance is to use timeouts for the messages send, as is done in [Wu *et al.* 2008]. Such a mechanism can handle link failures well. But when it comes to node failures, especially those involving the failure of the nodes containing the CS, then such a mechanism does not prove much useful. One of the key challenges is to identify whether the site is not responding because of failure or it is just taking too much time to execute CS. Another issue that is not addressed so far is how a node determines when a site has completed CS. That is, there is a need to determine whether a node has completed its CS in expected time, or did it crash while performing CS routine, or whether it is simply taking too much time in executing the CS. We introduce the *HOLD* message to intimate genuine requesting nodes that the current site is in CS. By mentioning the remaining time needed for exiting CS in the *HOLD* message, a requesting node can be informed about the expected time it needs to wait before permission can be granted for it to enter CS.

The choice of timeout values is crucial for fault-tolerant algorithms. In [Wu *et al.* 2008], the timeout value is statically assigned. The major issue with using a predetermined value is that if the value is not properly chosen, then we may end up resending messages too many times, or we may end up reacting to failure too slow. For all practical applications, it is simply not possible to predetermine values. It may be more appropriate to choose an approximate value, and to adapt it over time to meet our requirements while executing the algorithm. This led us to explore the possibility of using an adaptive timeout handling mechanism for dealing with this problem.

5.4.2 System Model and Assumptions

We consider a distributed system consisting of N sites $(S_0, S_1, \ldots, S_{N-1})$ participating concurrently for a shared resource. In this algorithm, we assume that there is one process per site that is involved in accessing the shared resource for the sake of convenience. Even if there is more than one process, our algorithm will still work as the processes in the same site will send the messages using the loop-back interface of the local machine.

In our work, we are not differentiating the failure of a node and the failure of the process which is running the code containing CS. In both the cases, the actions to be taken to continue execution are the same. The crashed or faulty node will take necessary steps to recover the data structures of the process, either by resetting the values or by using an older consistent set of values as has been suggested in [Masum *et al.* 2010]. The changes to the shared resource shall be committed only just before exiting the CS. Upon restart of the application of the crashed system, we assume all that things will continue to work as they were just before the node entered the CS.

We do not consider network partitions in this work. If the shared resource exists in only one of the partitions, then the nodes in the other partitions will eventually realize that the shared resource cannot be accessed. The partition with the shared resource will

continue to function normally as per the fault-tolerance mechanism. When the other partitions join later on, they will be considered as new nodes joining the system. If the shared common resource is still accessible in both partitions, then in effect we will have two distinct distributed systems in which the DME algorithm will be executed independently. The question of what has to be done when the two partitions merge is beyond the scope of this work. The issues related to network partitioning are already discussed in [Wu *et al.* 2008].

We do not place any restrictions on the number of code fragments for the CS. We also assume that all sites requesting for CS, can continue their execution only after going to CS. Also, a site in CS will not remain there forever, but will come out of CS within a finite time.

We assume that the nodes are already initialized using the initialization routines mentioned in [Singhal & Manivannan 1997] or [Wu *et al.* 2008]. Alternatively, the initialization method mentioned in Section 5.3.1 can also be used in our algorithm.

5.4.3 Algorithm Overview

In Figure 5.1, a simple operation of the algorithm is illustrated. The site S_0 is initially interested in entering into CS, and it sends $REQUEST$ message (in solid arrow) to all other sites in its $Info\_Set$, S_1 and S_2. Since both the sites are not currently interested in entering into CS, they immediately respond with $REPLY$ message (in dashed arrow). Since S_0 got $REPLY$ from all the requested sites, it enters into CS. While it is in CS, S_2 is interested in entering CS, and sends $REQUEST$ message to S_0 and S_1. Since S_1 is still not in CS, it simply sends back a response. S_0 is already in CS, so it sends back a $HOLD$ message. As soon as S_0 exits CS, it sends back $REPLY$ message to S_2. S_2 can now enter into CS, as it has received $REPLY$ message from all sites.

Figure 5.2 shows the state diagram corresponding to the operation of a participating node in the network. In the state diagram, *askCS* and *exitCS* are events that trigger change

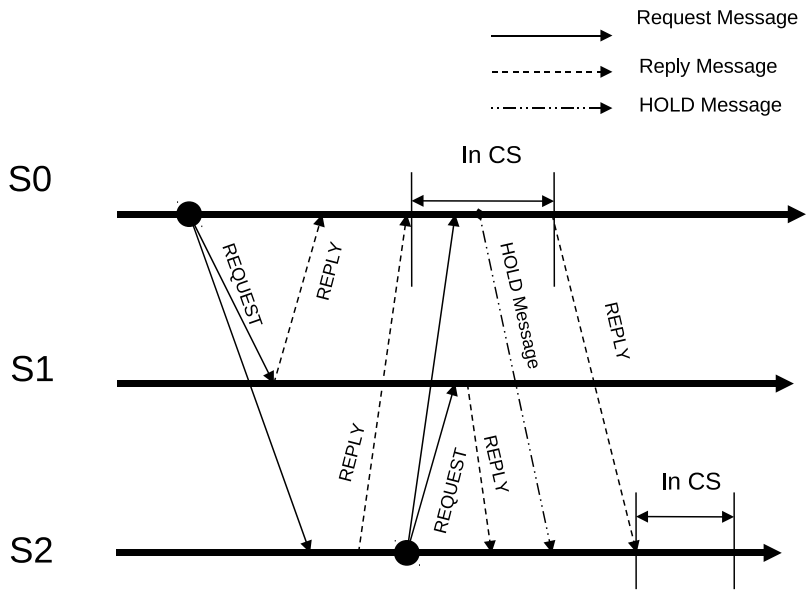

Figure 5.1: Novel DME Algorithm - an illustration

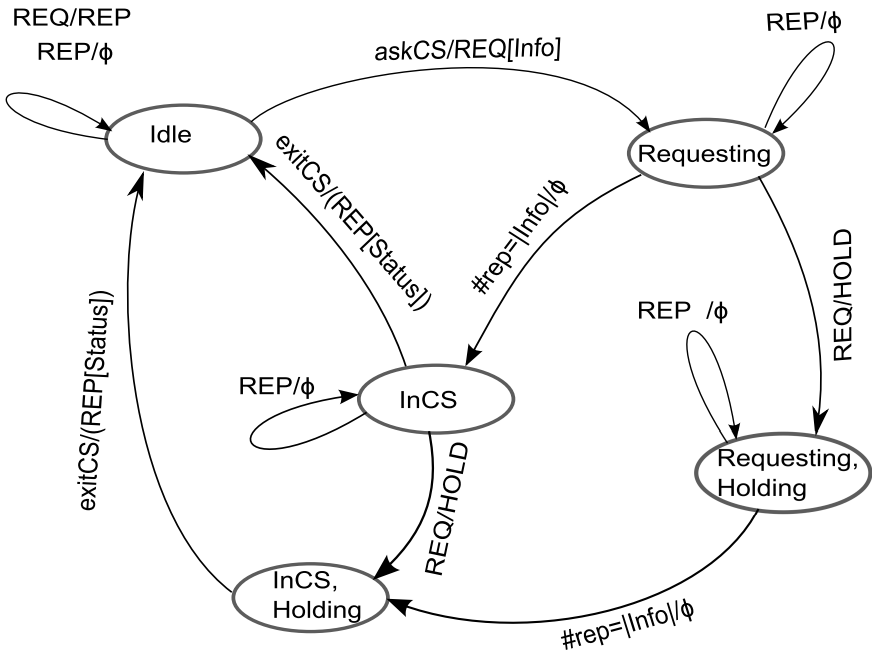

Figure 5.2: Working of Novel DME Algorithm - State Diagram

of state. There are two state changes triggered when a condition is met, as can be seen from the Figure 5.2. The events *REQ*, *REP* and *HOLD* indicate send/receive event of the $REQUEST$, $REPLY$ and $HOLD$ messages respectively. Wherever there is no output of a state transition, Φ is used to denote the lack of such an event.

Every node starts as an *Idle* node. When it asks for entering CS, it sends $REQUEST$ message to all nodes in its $Info\_set_i$ (represented in the figure as *REQ[Info]*) before moving to the *Requesting*. The node transitions from *Requesting* state to *Requesting-Holding* state when it receives $REQUEST$ message with a lower priority. From this state, the transition happens to *InCS-Holding* state when the node has received $REPLY$ message from all the nodes in its $Info\_Set_i$. While in *Requesting*, the node receives all pending replies, it transitions to the *InCS* state. If a $REQUEST$ message is received while a node is in CS, the node replies with a $HOLD$ message and transitions to *InCS-Holding* state. When a node exits from its CS (*exitCS* event), it transitions back to the *Idle* state.

5.4.4 Adaptive Timeout Mechanism

The algorithm depends on timeouts for handling fault-tolerance. Apart from link failures, there can also be node failures occurring in a MANET. Even the node that is currently in CS can also fail. To determine whether a node in CS (or the process running the code for CS in the node) has failed, we need to keep track of whether the node is still active or not. In this section, we discuss the adaptive timeout mechanism for maintaining the timeout value used to determine how long the site may be inside the critical section.

In the event of timeouts happening, we use Equation 5.1 to update the timeout value of T_{CS_DONE}.

$$T_{CS\_DONE} = T_{CS\_DONE} + (2 \cdot T - T_{CS\_DONE}) \cdot g\% \tag{5.1}$$

In Equation 5.1, g is the growth metric in percentage terms and T is the communication delay. The growth is mentioned in percentage terms instead of a fraction for convenience. The growth metric ensures that in the event of repeated timeouts, we will be progressively increasing the timeout values, but there is a clear upper bound of $2 \cdot T$ beyond which we

will not be raising the timeout values. $2 \cdot T$ is a reasonable upper limit that has been set in network protocols[Peterson 2011]. If we are leaving CS before the timer T_{CS_DONE} expires, then we should reduce the timeout value as shown in Equation 5.2.

$$T_{CS\_DONE} = T_{CS\_DONE} - (T_{CS\_DONE} - 0.5 \cdot T) \cdot g\% \tag{5.2}$$

The value $0.5 \cdot T$ is used as a lower bound during the decay phase of T_{CS_DONE}. Figure 5.3 shows the trends for the growth phase and the decay phase with $g = 50\%$. The value of g chosen should neighter be too high nor too low. If the value is too high and there is more than one critical section in the same process with varying execution time, then it will force the value of T_{CS_DONE} to be updated quite often. If the value is too low, then we may not adapt fast enough for the varying situations resulting in the transmission of extra *HOLD* messages.

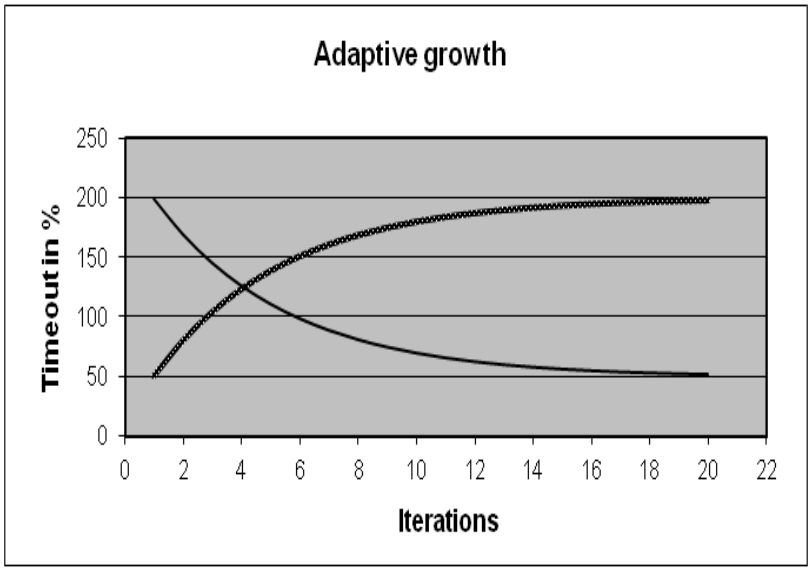

Figure 5.3: Growth and decay phases for the adaptive timeout mechanism - $g = 50\%$

5.4.5 Data Structures Used

Apart from the data structures used in [Wu *et al.* 2008] for maintianing the dynamic information sets and request queue, we use the following data structures.

TO_{CS_DONE}: A timeout meant to keep track of time left for the current node to exit its

CS.

TO_{REQ}: The vector TO_{REQ} is used to maintain timeout values to sites to which the current node had sent a $REQUEST$ message..

TO_{HOLD}: A timeout vector to keep track of the timeout values sent by sites from which $HOLD$ message was received.

Q_{HOLD}: This queue is used by the site in CS to keep track of the set of nodes to which it has to send $HOLD$ messages when TO_{CS_DONE} expires.

T_{CS_DONE}: This variable is used to set the initial value for TO_{CS_DONE}. The default value for this variable is the average time, T, needed for CS as estimated by the process.

5.4.6 The Algorithm

In this section, we describe how the algorithm behaves when it is entering CS, requesting for CS, receiving a $REQUEST$ message, receiving a $HOLD$ message, and handling the timeout mechanism.

5.4.6.1 Requesting for CS

As in [Singhal & Manivannan 1997], when a site S_i wants to enter CS, it will send a $REQUEST$ message to all the nodes in the $Info\_set_i$. Unlike waiting for a $REPLY$ message as is being done in [Wu *et al.* 2008] that may come immediately or with a significant delay (if the other site is in CS), the requesting site in this algorithm waits for either a $REPLY$ message or a $HOLD$ message to be sent back in response to its $REQUEST$. For every site S_j to which S_i had sent the $REQUEST$ message, it will set TO^j_{REQ} as the expected round-trip time between S_i and S_j.

5.4.6.2 Receiving $REQUEST$ message

When a site S_i receives a $REQUEST$ message from another site S_j, S_i first checks whether it itself is in CS. If S_i is not in CS or it is not contending for entering into CS, then it immediately sends a $REPLY$ message back to S_j as specified in Rule 5.2.2. If S_i is contending for CS, and it determines that S_j's request has a higher priority than its own request, then it sends back a $REPLY$ message as specified in Rule 5.2.3.

If S_i is having a higher priority than S_j or S_i itself is in CS, it sends a $HOLD$ message with the expected time for completing CS and the timestamp of its $REQUEST$ message ts_{req} back to S_j. The expected time includes the propagation delay for sending the message back to S_j and the value of T_{CS_DONE} . The dynamic information sets are updated as in Rule 5.2.3. S_i also adds the site S_j to Q_{HOLD}. Algorithm 5.4 shows the algorithm used for receiving $REQUEST$ message.

Figure 5.4: Receiving $REQUEST$ Message from site S_j

```
1  Procedure recvRequest
2  begin
3      if InCS(S_i) = true then
4          Info_set_i ← Info_set_i ∪ S_j
5          Send HOLD(T_CS_DONE − δ_prop^expected) to S_j
6          TO_CS_DONE ← T_CS_DONE (which is initialized to default value)
7          Q_HOLD ← Q_HOLD ∪ {S_j}
8      else if Requesting(S_i) = true then
9          if getPriority(S_i) > getPriority(S_j) then
10             Send HOLD(ts_req) to S_j
11             Q_HOLD ← Q_HOLD ∪ {S_j}
12         else
13             Send REPLY message to S_j
14         end if
15     else if ¬InCS(S_i)) then
16         Send REPLY message to S_j
17     end if
18     Wait till all sites have replied.
19 end
```

5.4.6.3 Receiving *HOLD* message

When a site S_i receives a *HOLD* message from site S_j, it understands that either S_j is in CS or it is having a lower priority than S_j for entering into CS. So it will update its *HOLD* message timeout TO_{HOLD}, as the maximum of the timeout value, τ, from the message and the current value of TO_{HOLD}. Algorithm 5.5 shows the algorithm used for receiving *HOLD* message.

Figure 5.5: Receiving *HOLD* Message from site S_j with timeout value τ

```
1 Procedure recvHold
2 begin
3 |   TO^j_HOLD ← max(τ, TO^j_HOLD)
4 |   Wait for REPLY from S_j
5 end
```

5.4.6.4 Handling of Timeouts

In this algorithm, we maintain timers for dealing with TO_{REQ}, TO_{HOLD} and TO_{CS_DONE}. The actions of their timeout handlers are described below.

TO_{REQ} : When a current node sends a *REQUEST* message, it will also be setting TO_{REQ} to twice the round trip time from the current node to the node being requested permission as is being done in [Wu *et al.* 2008]. If a *REPLY* or *HOLD* message is not received back within TO_{REQ} , *REQUEST* message is retransmitted. If a site doesn't respond with some message despite resending the message three times, we assume that the node has disconnected from the network and take necessary steps to deal with disconnection of a node.

TO_{HOLD} : is the vector used to keep track of nodes that are in CS or have higher precedence than the current node in CS. If in the current site S_i, the timer for TO^j_{HOLD} expires, then the site will again send a *REQUEST* message to S_j to check whether the site S_j is still active. If it didn't receive back a *REPLY* or *HOLD* message within TO_{REQ} then the S_i will assume that S_j has got disconnected and take necessary steps as mentioned in the TO_{REQ} timeout handler.

TO_{CS_DONE} : When this timer expires, it will send $HOLD$ message to all the sites in Q_{HOLD}, after updating the value using adaptive mechanism mentioned in Equation 5.1.

5.4.6.5 Entering the Critical Section

Just before a site S_i enters its critical section, it sets the timer for TO_{CS_DONE} as T_{CS_DONE}.

5.4.6.6 Exiting the Critical Section

When a site S_i exits its critical section, it sends $REPLY$ messages as in Rule 5.2.6. It then deletes all the entries in Q_{HOLD}. It then disables TO_{CS_DONE}. If the site exits its CS before T_{CS_DONE} time units, then it will compute the value of T_{CS_DONE} as derived by applying Equation 5.2. It updates the value of T_{CS_DONE}, if derived value has changed.

5.4.7 Correctness of the algorithm

Lemma 5.4.1. *Any site requesting for CS is aware of all the nodes, if any, waiting for CS with a higher precedence than its own.*

Proof. As soon as the $REQUEST$ message is received by another site S_j from a site S_i, S_j will immediately send back a $HOLD$ message if it itself is in CS as per Lines 3-7 of Algorithm 5.4. Thus S_i is aware of the site which is currently executing in CS, and it also knows for how long the site S_j is expected to be present in CS. Suppose Site S_k is also contending for CS, and it has a higher precedence (as in [Singhal & Manivannan 1997]) than S_j, then S_k is going to send back a $HOLD$ message with additional information on timestamp of its request as per Lines 9-11 of Algorithm 5.4. In the meantime, S_i would also have received the $HOLD$ message from S_j which is already in CS, along with its estimation on when it will release the CS. Thus S_i now has the information of both S_j which is in CS and any other node S_k with higher precedence than it's own. In fact, now S_i can compute the estimated time it has to wait for entering into CS. □

5.4.7.1 Mutual Exclusion

Theorem 5.4.1. *Algorithm 5.4 ensures that no two sites can execute CS simultaneously.*

Proof. To prove that the condition of mutual exclusion is satisfied, it is sufficient to prove that no two sites S_i and S_j can enter the critical section simultaneously. This can be done by the method of proof by contradiction.

Suppose two sites S_i and S_j are in CS. From the state diagram shown in Figure 5.2, it follows that both S_i and S_j have received *REPLY* message from all other sites including from one another before they could enter the *InCS* state or the *InCS-Holding* state. That means that S_i had received a *REQUEST* from S_j and determined that it is having a lower priority than S_j (Line 9 of Algorithm 5.4) and sent a *REPLY* back to S_j (Line 12 of Algorithm 5.4). Similarly S_j had also sent a *REPLY* message back to S_i after determining that it is having a lower priority than S_i. But according to the rule for finding priority, only one of the sites can have lower priority. Hence, a contradiction. □

5.4.7.2 Freedom from Deadlocks

Theorem 5.4.2. *Algorithm 5.4 is free from deadlocks.*

Proof. The proof is by contradiction. One of the constraints for deadlock to occur is circular wait. Let sites $S_i, S_j, S_k, \ldots S_l$ be part of circular wait, i.e., S_i is waiting for a *REPLY* to be sent by S_j, which in turn is waiting for S_k to send a *REPLY*, and so on and so forth, and S_l is waiting for a *REPLY* to be sent by S_i. This is only possible if the priority of S_i is less than S_j, which is less than S_k, which must be less than S_l, which is less than S_i. Clearly, this is not possible, and hence is a contradiction. □

5.4.7.3 Safety

Theorem 5.4.3. *Algorithm 5.4 is starvation-free.*

From Figure 5.2, any node in *Requesting* or *Requesting-Holding* state will transition to *InCS* or *InCS-Holding* state respectively, when the node has received replies from all sites in its $Info\_set_i$. To prove freedom from starvation (the safety property), we can use the same method employed by [Wu *et al.* 2008] and Corollary 2. since there is not much difference between the arguments, the proof is omitted.

5.4.7.4 Fault Tolerance

Theorem 5.4.4. *Based on assumptions, our algorithm effectively handles failure of both nodes and links.*

Proof. For proving this theorem, we need to consider two separate scenarios. The first one is the impact of link failure in the working of the algorithm. The second one is the impact of failure of node in three separate cases; viz. site not interested in getting into CS, site interested in getting into CS but not itself in CS, and a site in CS.

1. Link failure between Sites S_i and S_j: If link between S_i and S_j fails before S_i's request reaches S_j, then there are two cases to look at.

Case 1. *The failing link is the only link that connects S_i to S_j.*

If the failing link is the only link connecting the two sites, then this results in network partition, and is anyway dealt in assumptions.

Case 2. *There is an alternative path from S_i to S_j.*

In this case, the underlying routing algorithm in the MANET will take care of routing the message from S_i to S_j and back. If the timeout TO_{REQ} or TO_{HOLD} expires by the time the message is being rerouted, the resending mechanism described in handling of timeouts in Section 5.4.6.4 will ensure that the message is sent through the new available path.

Hence, in the event of link failures, the messages will be sent across between S_i and S_j.

2. Site S_i fails: Here there are three cases to consider.

Case 1. *S_i is not contending for CS.*

If S_i is not contending for CS, then it is as good as it had disconnected from the network, and the algorithm handles accordingly.

Case 2. *S_i is contending for CS, but is not in CS.*

In this case, there are two things to consider. If a site S_j has sent a $REQUEST$ message to S_i with a higher priority than S_i's $REQUEST$, then if S_i did not reply with a $REPLY$ message, it will resend the $REQUEST$ message for three more times following the mechanism for TO_{REQ} described in Section 5.4.6.4. After that it comes to the conclusion that S_i has failed

and removes it as if S_i has been disconnected from the network, which is anyway the case. If a site S_j has sent a *REQUEST* message to S_i with a lower priority than S_i's *REQUEST*, then it depends whether S_i had replied a *HOLD* message before it failed. If S_i had not sent a *HOLD* message back, then the mechanism for TO_{REQ} will handle the failure. If S_i had sent a *HOLD* message to S_j before failing, then the TO^i_{HOLD} would have been set. This timer will expire, since the S_i is no longer going to send the *REPLY* message as it had failed. Following the mechanism for the expiry of TO_{HOLD}, S_J will eventually find out that S_i had failed.

Case 3. *S_i is in CS.*
If the site S_i was in CS when it failed, then in each site S_j in Q_{HOLD} the timer corresponding to TO^i_{HOLD} will expire. Following the mechanism for the expiry of TO_{HOLD} described in Section 5.4.6.4, *REQUEST* message shall be resend and S_j will eventually find out that S_i had failed.

In all three cases, the algorithm eventually identifies that the site has failed.

□

5.4.8 Performance

Since we are using an additional message, the message complexity of this algorithm is more than $2 \cdot (\Phi - 1)$ described in [Singhal & Manivannan 1997]. The message complexity will depend on the number of times the *HOLD* message has to be sent. If we assume that the node in CS miscalculates the timeout m times, and there are w nodes in Q_{HOLD}, then the message complexity of the algorithm will tend to be $2 \cdot (\Phi - 1) + m \cdot w$. We could attempt an optimization to reduce the number of messages. Instead of immediately responding with a *HOLD* message, a site could delay sending the *HOLD* message depending on the round-trip time, so that one message could be saved. But our ability to recover from any node failure offsets the additional message overhead incurred.

5.5 Reliable Arbitration-based DME Algorithm

In this section, we discuss the second DME algorithm that was proposed by us. In this work, we have focused on permission based approach for the DME problem. Within the

ambit of permission-based algorithms, we are using arbitration as the basis for handling DME among multiple regions. An arbitrator node that is aware of more than one region can easily look at the two regions and judge which node has the right to enter its CS as is being done in [Maekawa 1985]. By using regions spanning multiple hops in diameter, the task of granting permissions among the nodes is better distributed by leveraging the available geographic information. We use arbitrators to decide which of the nodes in neighbouring regions can be granted to enter CS without requiring a requesting node to ask for permission from a far away node. Our main contribution in this algorithm is the use of arbitration among multiple regions and providing reliability using appropriate fault-tolerance within the algorithm. The related works for this algorithm was presented in Section 2.5.

The current set of approaches for permission based algorithms focus on reducing message complexity either by focusing on trying to reduce the effective number of participating nodes or by resorting to clustering. The clustering solution, however, would still perform worse than the classic Maekawa's algorithm due to the fact that communication overhead between cluster-heads is still relatively higher, especially in the case of large networks in which the participating nodes are spread out. While trying to divide the current node into clusters, the current set of algorithms are using only one-hop neighbourhood for cluster creation. The cluster's periphery nodes transmit many of the messages, but they never make any decisions. This leads to overburdening of the cluster-heads. Since such hierarchical approaches look at inter-cluster and intra-cluster communication as two different sub-problems, they resort to a distributed version for handling DME among clusters, while resorting to a centralized algorithm with the cluster-head acting as the lone decision-making node within the cluster. There is a high computational as well as communication load on the cluster-heads, as they have to handle both cluster management and DME algorithm. Combinatorial stability is another factor that is not guaranteed by these algorithms. In the algorithm presented in this section, we relax the roles assigned by the nodes for region management and handling DME, thereby distributing the load to nodes in the periphery of the region as well.

5.5.1 Assumptions

Apart from the assumptions made in Section 5.4.2, we assume that every participating node in CS is aware of the location of every other participating node in the network. This is a fair assumption as most of the current nodes do come with in-built GPS system just like the nodes using geographic routing algorithm.

We assume that the regions have been created a priori. For a simple region creation mechanism, we could use a two-level Maximal Independent Set (MIS) as can be derived from [Alzoubi *et al.* 2002b] with a region radius of approximately 5 hops. The construction of MIS was introduced in Section 2.2.6.1. We assume that there is also an appropriate region maintenance algorithm available for maintaining the network in the event of node mobility or node failure. For the sake of simplicity of the discussion, we assume that the network contains no voids. As long as the region maintenance algorithm is capable of handling addition and drifting of nodes between regions, and of appropriately identifying arbitrator nodes between two neighbouring regions, our algorithm will work.

We assume that the region maintenance algorithm will take care of initialization of the $Info\_set$ and $Status\_set$ appropriately. The addition of initialization steps is a trivial assignment of internal variables to the appropriate steps. The list of arbitrators in a region is available with its region head and can easily be communicated to other nodes. We assume that the initialization of the relevant data structure are being done by the region maintenance algorithm. For the purpose of initialization, the simplified version presented in Section 5.3.1 can be used.

We do not place any restrictions on the size of code fragment pertaining to the CS. We also assume that all sites requesting for CS, can continue their execution only after going to CS. Also, a site in CS will not remain there forever, but will come out of CS within a finite time. We also assume that the nodes are capable of differentiating between the actual sender of a particular message and the message initiator.

5.5.2 Working of Arbitrator

In this section, we discuss the overall working of the algorithm and the working of an arbitrator node.

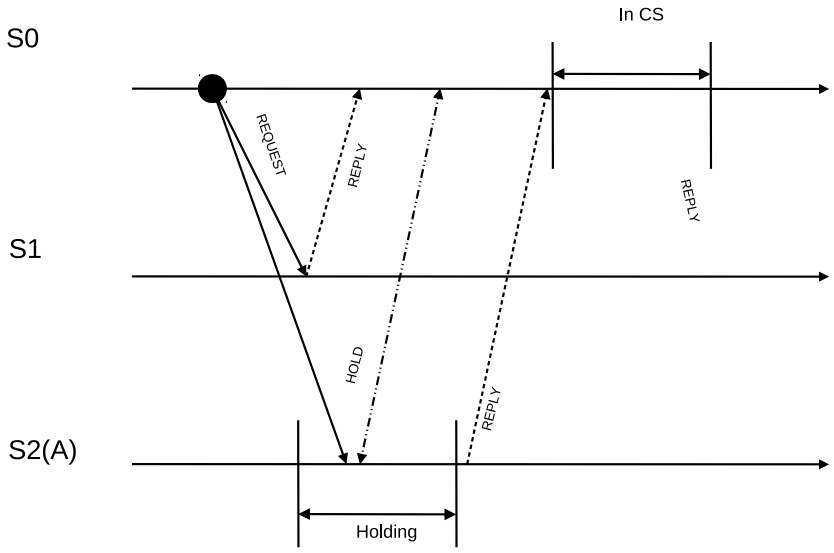

Figure 5.6: Basic operation with an arbitrator in Holding. -

In Figure 5.6, the site S_0 initially is interested in entering CS. So, it sends a *REQUEST* message to all the nodes in its Info_Set, S_1 and S_2 (illustrated with solid arrow). As S_1 within S_0's region is not involved in CS, they send back a *REPLY* message as in [Singhal & Manivannan 1997](illustrated with dashed arrows). The site S_2, which is an arbitrator, is aware that some other site has already made a request earlier. So it has assumed the responsibility of ensuring that all the interested nodes within its region could quickly be informed about this. S_2 sends *HOLD* message to S_0, indicating that S_0 has to wait for its turn to enter CS, along with an estimated waiting time. As soon as S_2 is aware that the other site has come out of CS (by means of a *REPLY* message not shown in the figure), it sends a *REPLY* message to S_0. The intervention of S_2 reduces the number of messages that needs to be transmitted across the network.

Figure 5.7 shows a sample network to indicate how regions, region heads and arbitrators appear. This network is drawn with the extended 2-level MIS creation algorithm based on the algorithm mentioned in Section 2.2.6.1and Section 4.6.2 is chosen for region maintenance. Seven regions are shown in the figure, identified by the irregular dotted shapes. The region heads are marked as *RH* followed by the region number in subscript. Arbitrators are marked with the letter *a*. The nodes in the network are marked as cir-

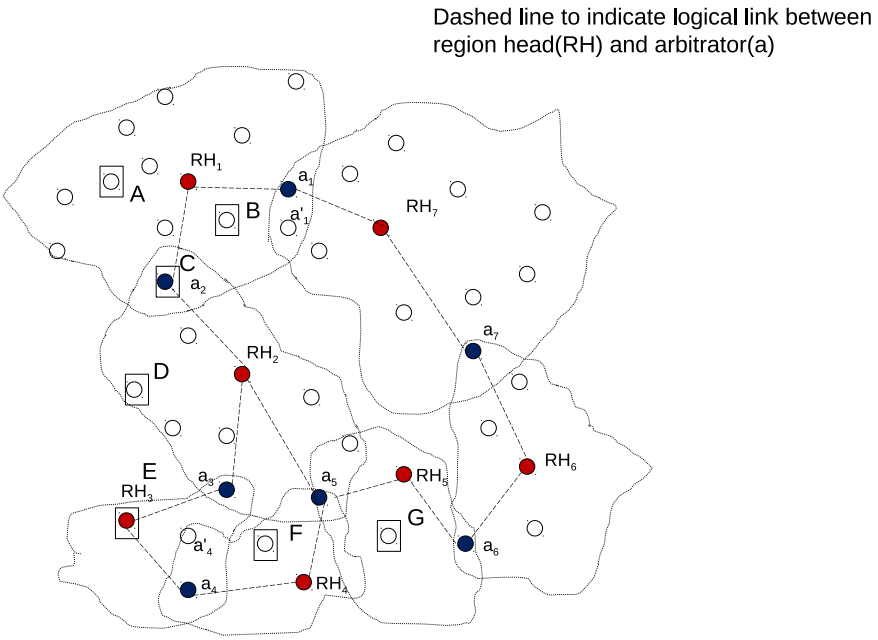

Figure 5.7: A sample network with multiple regions. - seven regions

cles. The participating nodes, lettered A through G, that are part of the group of nodes competing for CS, are indicated as circle within a rectangle.

Two regions may share more than one candidate arbitrator. For example, region 1 and region 7 have two possible arbitrators, a_1 and a_1'. Here we have arbitrarily chosen one of the nodes, a_1 as the arbitrator. If we observe the arbitrator node a_2(also marked as C) connecting region 1 and region 2, we can see that the node is a participating node as well. All arbitrator nodes need not be participating nodes, as can be confirmed with nodes a_1, a_3, a_4, a_5, a_6, and a_7. It is also mandatory for the region heads to be participating nodes as well. In Figure 5.7, only RH_3 is a participating node.

It can also be observed that, unlike [Maekawa 1985] which stated that an arbitrator is meant to arbitrate only two sets, in our algorithm, an arbitrator may be connecting more than two regions. In Figure 5.7, the arbitrator a_5 will be aware of three regions; namely, region 2, region 4 and region 5.

The overall functioning of an arbitrator is illustrated in Figure 5.8. Here, S_1 is an arbitrator node that is common to region 1 and region 2. The site S_0, belonging to region 1, initiates a request for entering CS. As S_1 is aware that it is an arbitrator and there are no current requests pending with it, it will act as a proxy for S_0 in region 2, and send

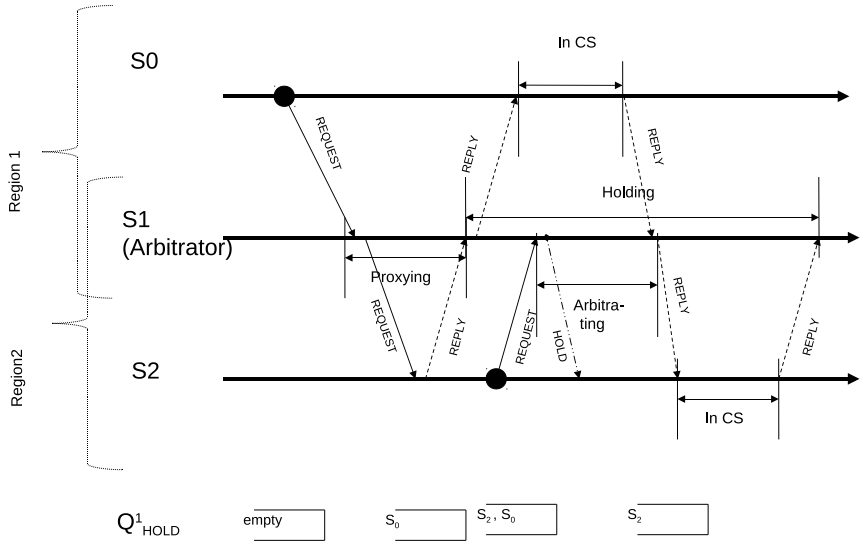

Figure 5.8: Working of an Arbitrator. -

request message to nodes in its Info_Set in region 2. As soon as S_1 has received requisite permissions from region 2, it sends *REPLY* to S_0 and moves from *proxying* state to *holding* state. When S_2's *REQUEST* message arrives in S_1, S_1 knows that it is currently holding for the nodes in its Q_{HOLD} (listed in the bottom of the figure), it sends a *HOLD* message back to S_2. As S_1 is now dealing with two (or more) regions, it will also mark itself as *arbitrating*. When the *REPLY* message of S_0 reaches S_1, it identifies that only nodes in one region are associated with it. S_1 resets the *arbitrating* flag. It now sends *REPLY* message to S_2 and waits for S_2 to exit from the CS. S_1 will come out of *holding* state as soon as it identifies that all nodes in all regions associated with it are neither requesting to enter CS nor executing in CS.

Figure 5.9 depicts the state diagram of an arbitrator node that can also be participating node in DME. Apart from the state diagram mentioned in Section 5.4.3, the additional states used are *Proxying*, *Holding*, *Arbitrating* and *Requesting-Arbitrating*. We describe here only the state transitions of the node where it differs from Figure 5.2.

The node transitions to the *Proxying* state when it receives a *REQUEST* message while it is in the *Idle* state. If the node receives *REPLY* message from the node that had requested earlier(refereed in the figure as *REP(CS node)* event), it transitions back to the *Idle* state. While in the *Proxying* state, if it receives a *REQUEST* message from a different region than the requesting node's region (refereed in the figure as *REQ(diff region)* event), the

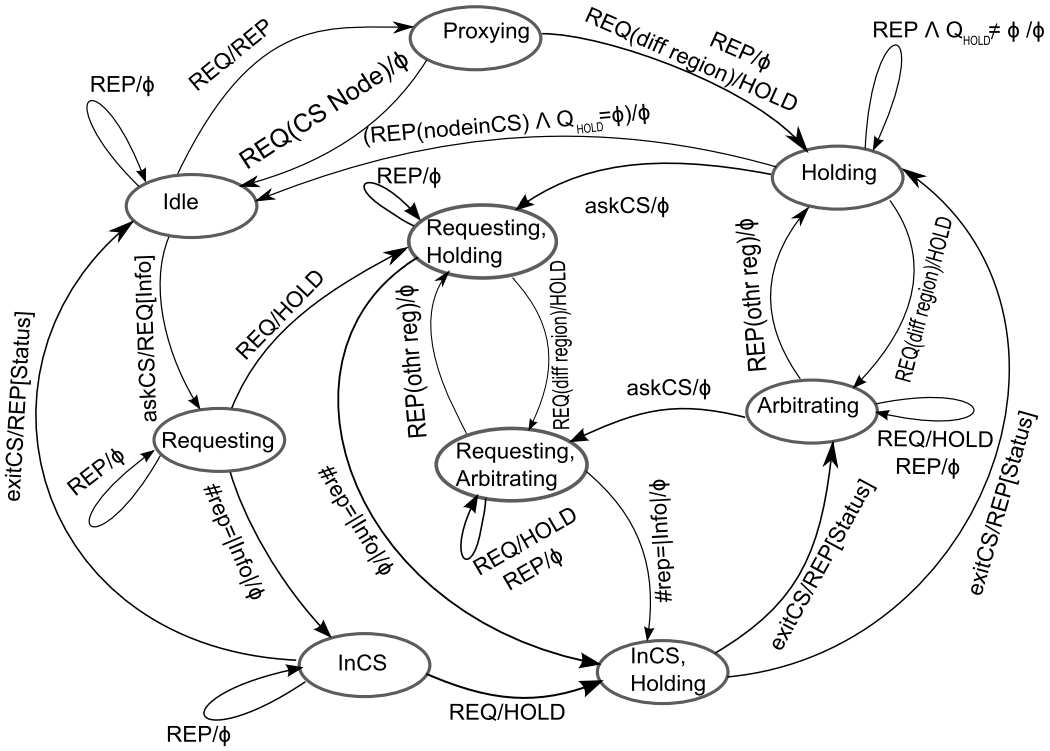

Figure 5.9: Reliable Arbitrator-based DME Algorithm - State diagram of a participating Arbitrator node

node transitions to the *Holding* state.

If a node in the *Holding* state receives a $REQUEST$ message, it moves to the *Arbitrating* state. While in *Holding* state, the node receives $REPLY$ message and no more nodes are present in Q_{HOLD}, then it transitions back to the *Idle* state. While in the *Holding* state, if a node receives $REQUEST$ message from a different region other than the region to which the nodes in Q_{HOLD} belong to, it transitions to *Arbitrating* state. While a node is in *Arbitrating* state, the node transitions back to the *Holding* state if it had received $REPLY$ message and all nodes in Q_{HOLD} belong to the same region.

The state of the arbitrator transitions from the *Holding* state to the *Requesting-Holding* state and from the *Arbitrating* state to the *Requesting-Arbitrating* state happen when the arbitrator node itself is asking for entering CS. The transitions between *Requesting-Holding* state and *Requesting-Arbitrating* state are similar to *Holding* state and *Arbitrating* state, and hence not elaborated. As is evident from the state diagram shown in Figure 5.9, the arbitrator transitions back to the *Idle* state from *Proxying* and *Holding* states once its job as an arbitrator is completed, or when the arbitrator is in *InCS* state and it had completed its

CS.

5.5.3 Data Structures Used

Apart from the data structures used for look-ahead technique in [Wu *et al.* 2008], we use the following data structures.

Reg: This list keeps track the different regions that the site belongs to, as well as the full list of all nodes in each of the associated regions. This list is mainly used by arbitrator nodes.

$Info\_set$: This set is the inform set defined in [Singhal & Manivannan 1997]. After the regions are decided by the region maintenance algorithm, all the nodes within a region that are participating will get included in this set. Apart from that, the set will also get the list of arbitrators into this list. When a new arbitrator node is established, it will get added into this set.

Status_set: This set is the status set defined in [Singhal & Manivannan 1997]. Apart from the usual operations on this set, whenever the failure of a node in this list is identified, it will get automatically dealt with as if it is a non-participating node.

Q_{REQ}: This queue is used to maintain the list of nodes to whom we had sent request for the sake of fault-tolerance. When there is a change in the arbitrator due to arbitrator node failure, the entry of the failed node will be purged from this queue.

TO_{CS_DONE}: A timeout meant to keep track of time left for the current node to exit its CS. This is the estimated time that the current node will be informing via the $HOLD$ message.

TO_{REQ}: The vector TO_{REQ} is used to maintain timeout values to sites to which the current node had sent a $REQUEST$ message.

TO_{HOLD}: A timeout vector to keep track of the timeout values sent by sites from which HOLD message was received.

T_{CS_DONE}: This variable is used to set the initial value for TO_{CS_DONE}. The default value for this variable is the average time, τ, needed for CS as estimated by the process.

Q_{HOLD}: This priority queue is used by the site in CS and arbitrators to keep track of the set of nodes to which it has to send $HOLD$ messages.

proxying: This flag is used by an arbitrator to indicate that it is working on behalf of another node for getting permissions from other regions.

holding: This flag is used to identify whether the current node is aware that some other node is in CS. It is used primarily by arbitrators. This is an optional flag, and could be replaced by a check on the number of entries in Q_{HOLD}.

arbitrating: This flag is used by an arbitrator to indicate that it is currently acting on behalf of one region to another region. This flag is set only to enable a site to easily identify whether it is dealing with multiple regions. This is an optional flag, and could be replaced by a less efficient method of searching the contents of Q_{HOLD} for sites in all regions.

5.5.4 The Proposed Algorithm

We only describe the modifications done in Section 5.4 to support the notion of arbitrators in this section. The handling of $DISCONNECT$ and $RECONNECT$ messages are not elaborated here, as their implementation is same as in [Wu *et al.* 2008] and Section 5.4.

5.5.4.1 Requesting for CS

As in Section 5.4, when a site S_i wants to enter CS, it will send a $REQUEST$ message to all the nodes in the $Info\_set_i$. The requesting site is either sent with a $REPLY$ message or a $HOLD$ message in response to its $REQUEST$. For every site S_j to which S_i has sent the $REQUEST$ message, it will set TO^j_{REQ} as the expected round-trip time between S_i and S_j.

5.5.4.2 Receiving $REQUEST$ message from site S_j

Apart from the procedure for dealing with $REQUEST$ message followed in Section 5.4, the site performs the following to take into account the multiple regions involved. If the site S_i is an arbitrator, then it will mark the *holding* flag when it itself is in CS or when it has a higher priority than the requesting site S_j.

If S_i identifies that it is holding for one region, while the request is coming from another region, then it sets the *arbitrating* flag. If S_i is already arbitrating, then the site simply adds the incoming request to the Q_{HOLD}.

Figure 5.10: Receiving *REQUEST* Message from site S_j

```
1  begin
2      if S_i = S_j then
3          Discard the message
4          Exit procedure
5      end
6      if InCS(S_i) = true then
7          Info_set_i ← Info_set_i ∪ {S_j}
8          Send HOLD(T_CS_DONE − δ_prop^expected) to S_j
9          TO_CS_DONE ← T_CS_DONE
10         Q_HOLD ← Q_HOLD ∪ {S_j}
11         if Arbitrator(S_i) = true then
12             holding_i ← true
13     else if Requesting(S_i) = true ∨ proxying_i then
14         if getPriority(S_i) > getPriority(S_j) then
15             Send HOLD(ts_req) to S_j
16             Q_HOLD ← Q_HOLD ∪ {S_j}
17             if Arbitrator(S_i) = true then
18                 holding_i ← true
19         else
20             Send REPLY message to S_j
21         end if
22     else if ¬InCS(S_i) then
23         if arbitrating_i then
24             Send HOLD(ts_req) to S_j
25             Q_HOLD ← Q_HOLD ∪ {S_j}
26         else if holding_i then
27             arbitrating_i ← true
28             Info_set_i ← Info_set_i ∪ {S_j}
29             Send HOLD(T_CS_DONE − δ_prop^expected) to S_j
30             Set TO_CS_DONE ← T_CS_DONE
31             Q_HOLD ← Q_HOLD ∪ {S_j}
32         else if Arbitrator(S_i) = true then
33             proxying_i ← true
34             foreach site S_k ∈ Info_set_i do
35                 Send REQUEST message to S_k
36         else
37             Send REPLY message to S_j
38         end if
39     end
40     Wait till all sites have replied.
41 end
```

If S_i is not currently holding for some other node and it receives a request from S_j, then it identifies that it needs to act as a proxy for S_j. S_i transmits $REQUEST$ message to all nodes in the other region, making a note of the $REQUEST$ message in its request queue, just like a normal request initiated by it. While S_i is acting as a proxy, if it receives further requests, it responds with a $HOLD$ message.

Algorithm 5.10 shows the algorithm used for this event.

5.5.4.3 Receiving $HOLD$ message

When a site S_i receives a $HOLD$ message from a site S_j, it means that S_i's request could not be served because some other site is in CS or has a higher priority than itself. S_i acts the way it is done in Section 5.4. If the site S_i is an arbitrator, it will check whether *proxying* flag is set. If so, then it marks the *holding* flag to indicate that some other site is in CS and puts the node in its request queue to Q_{HOLD}. It then sends a $HOLD$ message with updated estimated time inclusive of the time in the incoming $HOLD$ message to this site in Q_{HOLD}. If S_i has the *holding* flag set, it updates the holding timeout value, TO_{HOLD}, so that it can convey the new holding timeout value for the future. Algorithm 5.11 shows the steps taken by the site receiving the $HOLD$ message.

5.5.4.4 Receiving $REPLY$ message

When a site S_i receives a $REPLY$ message from a site S_j, apart from the way it is treated in [Wu *et al.* 2008], S_i checks the following in case it is an arbitrator. If *proxying* flag is set and the $REPLY$ received is the last $REPLY$ message it needs from that region, then it marks the *holding* flag and adds the site on behalf of which S_i had requested (stored in front of its request queue) to Q_{HOLD}. If the *holding* flag is set and S_i has received all requisite $REPLY$ messages (i.e. S_j was the last site not in its Q_{HOLD} to send the reply), it sends a $REPLY$ message to the node in front of Q_{HOLD}. We check *holding* flag separately and use $front(Q_{HOLD})$ to ensure that the site with the highest priority will get the right to continue.

Figure 5.11: Receiving *HOLD* Message from site S_j with timeout value τ

```
1  begin
2      TO^j_HOLD ← max(τ, TO^j_HOLD)
3      if proxying_i then
4          holding_i ← true
5          S_proxy ← front(Q_REQ)
6          Q_HOLD ← Q_HOLD ∪ {S_proxy}
7          Send HOLD(TO_HOLD + δ_prop(S_proxy)) to S_proxy
8      else if holding_i then
9          foreach site S_k in Q_HOLD do
10             TO^k_HOLD ← max(τ + δ_prop(S_k), TO^k_HOLD)
11     end if
12     Wait for REPLY from S_j
13 end
```

5.5.5 Correctness

5.5.5.1 Mutual Exclusion

Lemma 5.5.1. *Mutual exclusion is guaranteed if all sites are within a single region.*

Proof. The proof for Lemma 5.5.1 comes from the correctness argument in Section 5.4 as the proposed algorithm reduces to the algorithm in Section 5.4 if there is only a single region. □

Lemma 5.5.2. *Any site requesting for CS is aware of all the nodes, if any, waiting for CS with higher precedence than its own.*

Proof. We follow the same argument as Lemma 5.4.1 for sites requesting within a region. We need to consider the case of a *REQUEST* sent to an arbitrator node as can be seen from the state diagram given in Figure 5.9. If an arbitrator node S_j receives a *REQUEST* message from the site S_i, and it had responded with a *HOLD* message, then the only reason why S_j had sent the *HOLD* message is because it's *holding* flag is set. This can happen only when S_j is aware of a site in CS or is aware of a site with a higher priority than S_i. Hence S_i becomes aware of another site in CS or with a higher precedence than itself in another region as well. □

Theorem 5.5.1. *Distributed mutual exclusion property holds for the Algorithm 5.10.*

Proof. Given Lemma 5.5.1, we only need to prove that mutual exclusion holds in the event of simultaneous requests from multiple regions. The proof is by contradiction. Let S_i be the site with highest priority in the network that is contending for CS. If there exist a site S_j, whose priority is lower than S_i, but still has entered CS, then it means that S_j has entered CS despite knowing that S_i has a higher priority than itself. From Lemma 5.5.2, every site S_j is aware of all sites S_i with a higher priority than itself. From Algorithm 5.10, a site will not enter CS in the presence of another site with a higher priority than itself. This is a contradiction. □

5.5.5.2 Freedom from Deadlocks

Theorem 5.5.2. *The algorithm is free from deadlocks.*

Proof. The proof is the same as that in Theorem 5.4.7.2, as the modifications in the algorithm do not change the dependence on receiving $REPLY$ messages for entering CS. □

5.5.5.3 Freedom from Starvation

Theorem 5.5.3. *The algorithm is free from starvation.*

Proof. The proof is along the same lines as that in Section 5.4.7.3. Though we have multiple regions, no site will be denied the right to enter critical section, if it has the highest priority in the network. From Lemma 5.5.2 and Algorithm 5.10, we can infer that every node will allow all nodes with a higher precedence than itself to enter critical section, thereby allowing every node to enter its CS when its turn arrives. In the case of failure of an arbitrator node, once the system has stabilized after assigning new arbitrator(s), a node with a lower priority would still have to acquire permission from the arbitrator who will eventually become aware of the presence of a node with higher priority. □

5.5.5.4 Fault tolerance

Lemma 5.5.3. *Fault tolerance is guaranteed if all sites are within a single region.*

Proof. The proof for Lemma 5.5.3 comes from the correctness argument in Section 5.4 as the proposed algorithm reduces to the algorithm in Section 5.4 if there is only a single region. □

Theorem 5.5.4. *Given the assumptions, our algorithm effectively handles failure of both nodes and links.*

Proof. From Lemma 5.5.3, proving that fault tolerance is ensured in the case of inter-region communication is sufficient to prove that our algorithm is fault-tolerant. In inter-region communication, we need to prove that link failure and node failure are correctly dealt with.

1. *Link failure between Sites S_i and S_j*: The proof argument is the same as that given in Section 5.4, as the behaviour and assumptions are exactly the same two cases to look at.

2. *Node failure at site S_i*: Depending upon what S_i was maintaining, there are the following six cases to consider.

Case 1. *S_i is not contending for CS and it is not an arbitrator.*

If S_i is not contending for CS, then it is as good as it had disconnected from the network, and our algorithm handles accordingly.

Case 2. *S_i is not contending for CS, but it is an arbitrator whose proxying flag is set.*

The failure of S_i will be treated as disconnection of the node from the network. The region maintenance algorithm will immediately assign some other node(/s) as the arbitrator(/s) connecting the regions affected. In the site S_j, on whose behalf S_i was acting as a proxy, the timeout TO_{REQ} will expire and will eventually be dealt with as a disconnected node. In the mean time, it would also observe that a new node has been included in its $Info\_set$, for whom it had not sent a request. So, it sends a $REQUEST$ message to this arbitrator, which will in turn take care of proxying the request. From this point on, the algorithm would continue with the normal mode of operation.

Case 3. *S_i is not contending for CS, but it is an arbitrator whose holding flag is set.*

This case again results in addition of new arbitrator(s), and the affected nodes' TO_{HOLD} or TO_{REQ} will expire, as the case may be, while new non-requested nodes will appear in their Info_sets. This will make the affected node(s) to send $REQUEST$ message to the newly created arbitrator(s) and the algorithm would continue with the normal mode of operation.

Case 4. *S_i is contending for CS, but not in CS and it is not an arbitrator.*

There are four situations to consider here.

- If a site S_j had send a $REQUEST$ message to S_i with a higher priority than S_i, and S_i had not replied; then the timeout mechanism will eventually inform S_j that S_i had disconnected and S_j will act accordingly.
- If a site S_j had send a $REQUEST$ message to S_i with a higher priority than S_i, and S_i had replied; then the failure of S_i is inconsequential to the working of the algorithm.
- If a site S_j had send a $REQUEST$ message to S_i with a lower priority than S_i, and S_i had responded with a $HOLD$ message; then the timeout TO_{HOLD} will eventually expire and S_j will know that the S_i had failed. S_j will now act as if S_i had disconnected and act accordingly.
- If a site S_j had send a $REQUEST$ message to S_i with a lower priority than S_i, and S_i had not responded with a $HOLD$ message; then the timeout TO_{REQ} will eventually expire and this treated as if S_i had disconnected.

Case 5. *S_i is contending for CS, but not in CS and it is an arbitrator with Holding flag set.*

This is handled in the same way as case 4, as the algorithm doesn't differentiate whether the incoming $REQUEST$ is from an arbitrator or not.

Case 6. *S_i is in CS*

If the site S_i was in CS when it failed, then in each site S_j in Q_{HOLD} the timer corresponding to TO^i_{HOLD} will expire. Following the mechanism for the expiry of TO_{HOLD}, $REQUEST$ message shall be resend and will find that S_i had failed. This mechanism works exactly the same way whether S_i was an arbitrator or not. Following the identification of the failure of a node in CS, the appropriate mechanism for failure recovery of the application will kick in, as was assumed by our algorithm.

From the above six cases, it is clear that a failing node will be detected correctly. Note that as *arbitrating* flag is used only to keep track of whether S_i is having pending requests from more than one region, it is not going to affect the working of the algorithm in the event of node failure. □

5.5.6 Performance

Theorem 5.5.5. *If there are r regions with a maximum of Φ_{max} participating nodes within each region; m number of wrong timeout computations; and w sites in all the Q_{HOLD}; then the message complexity of the algorithm is $2 \cdot r \cdot (\Phi_{max} - 1) + m \cdot w$ messages.*

Proof. If there is no failure, all the sites are within the same region, and if we assume that time taken in CS by all nodes are less than τ defined in Algorithm 5.10, then the message complexity will be $2(\Phi - 1)$, same as in [Singhal & Manivannan 1997]. Within a region, the algorithm will be sending the *REQUEST* and *REPLY* messages for gaining permission. Thus, following [Singhal & Manivannan 1997], a total of $2 \cdot (\Phi_{max} - 1)$ messages are needed within each region. For every timeout (except when timeout limit is reached), the algorithm will resend *REQUEST* or *HOLD* message as the case maybe. With a single region containing Φ participating nodes, if we assume that the node in CS miscalculates the timeout m' times, and there are w' nodes in Q_{HOLD}, then the message complexity of the algorithm will tend to be $2(\Phi - 1) + m' \cdot w'$. Thus, for the potential w sites that may cause a timeout, a total of $m \cdot w$ messages would be required to be transmitted in the worst case. Since we need to consider r possible regions, we get the total number of messages to be $2 \cdot r \cdot (\Phi_{max} - 1) + m \cdot w$ in the worst case. □

Corollary 4. *With r equal regions and optimal node placement, message complexity is $O(\sqrt{\Phi})$ messages, for Φ participating nodes.*

Proof. If there is no failure; there are exactly r regions arranged in hexagonal circular mapping (the best possible way to pack the circular regions in 2-D with maximum density [Wells 1991]); there are exactly 6 arbitrators in each region. The total number of participating nodes in each of the r regions is then Φ', where $\Phi = r \cdot \Phi'$). Thus, in every region Φ' *REQUEST* and *REPLY* messages will be sent, and 6 *HOLD* messages will be sent. If all the regions are equally spaced, then $\Phi' \approx \sqrt{\Phi}$, as derived in [Maekawa 1985]. Let d be the number of regions from the centre of the packing to the periphery of the network along the diameter. The total number of messages needed to be sent through these d rings will be $2 \cdot d \cdot (\sqrt{\Phi'} + 6) \preceq O(\sqrt{r} \cdot \Phi^{\frac{1}{4}})$ which tends to $O(\sqrt{\Phi})$. □

Theorem 5.5.6. *The synchronization delay for the algorithm is* $2 \cdot \delta_{comm}^{max}$*, where* δ_{comm}^{max} *is the communication delay for the longest path in the network, in the absence of failures.*

Proof. Let τ be the tree created by connecting all pair of nodes which need to communicate with one another for gaining permissions, as given in their respective Info_sets. It is trivial to prove that τ forms a multicast tree for the list of participating nodes and arbitrator nodes within their regions. Let ρ_{max} take the longest path in this tree, τ. The communication delay of ρ_{max} is bounded by the communication delay for the longest path in the network to get the permission, δ_{comm}^{max}. In the worst case, a node may have to send a $REQUEST$ and wait for a $REPLY$ along ρ_{max}. Since all other paths along τ is going to be smaller than ρ_{max}, and since the delay is bounded by delay of the longest path in τ, synchronization delay will be the time for $REQUEST$ message to reach the other end, and the $REPLY$ message to return, which is $2 \cdot \delta_{comm}^{max}$. □

Corollary 5. *In the absence of any failures, the synchronization delay for the algorithm in the worst case is* $2 \cdot r \cdot T_{reg}^{max}$*, where* T_{reg}^{max} *is the maximum round-trip time for transmitting a message within a region.*

Proof. The worst case occurs when all the regions are strung together in a long line. This can easily be explained by taking a network where only arbitrators are participating for the CS, and each region is having exactly two arbitrators in diametrically opposite ends, except for the two regions in the extremes containing just one arbitrator. So, instead of getting a tree, τ, as mentioned in the proof for theorem 5.5.6, all the participating nodes requesting for permission will form a long chain. As there are r regions, the length of the chain will be $r \cdot d$, where d is the maximum diameter of a region as defined by the region maintenance algorithm. As we have defined the delay for traversing the distance d as T_{reg}^{max}, the maximum synchronization delay is $2 \cdot r \cdot T_{reg}^{max}$. □

Performance comparison of various approaches are detailed in Table. 5.1. Apart from the usual metrics, we have also included the total number of message types used in the algorithm for comparison. This inclusion has been primarily made, because we observed that there is a significant difference in the number of message types that have been added to the classic algorithms while using clustering. Though the number of types of messages

Table 5.1: Performance comparison

Algorithm	**Synchronization Delay**	**Message count**	**Number of message types (ignoring cluster maintenance)**	**Fault tolerance Support**
Singhal *et al.* [Singhal & Manivannan 1997]	ΦT	$2(\Phi)$	2	No
Ricart-Agrawala [Ricart & Agrawala 1981]	NT	$2(N-1)$	3	No
Maekawa [Maekawa 1985]	$\sqrt{N} \cdot T$	$3 \cdot \sqrt{N}$	3	No
Wu, *et al.* [Wu *et al.* 2008]	ΦT	$2(\Phi)$ (without failure)	2 + 4(for node failure)	Yes (Timeout)
Erciyes, *et al.* [Erciyes & Dagdeviren 2012]	$(k+2d-1)T$	$k+3d$	4	No
Gupta, *et al.* [Gupta *et al.* 2012]	$2T \cdot (m+1)$	$O(N)$	6	No
Parameswaran *et al.* [Parameswaran & Hota 2010]	ΦT	$2 \cdot (\Phi - 1) + m \cdot w$	3 + 2(for node failure)	Yes
RAD [Parameswaran & Hota 2013]	$2 \cdot \delta_{comm}^{max}$	$2 \cdot r \cdot (\Phi_{max} - 1) + m \cdot w$	3 + 2(for node failure)	Yes

is independent of message complexity, we observed that addition of extra types of messages generally tend to increase the size of code and the number of event triggers to be handled without any significant benefit in terms of performance improvement. Reducing the number of message types also improves ease of implementation.

We performed a comparative evaluation of the proposed DME algorithm, which we call as RAD, and algorithms in [Gupta *et al.* 2012, Erciyes & Dagdeviren 2012] with the help of a simulator. All the three algorithms were tested by creating networks randomly for different number of nodes. While testing, we have deliberately inserted voids in the

middle of the graph for half of the simulations to bring out worst case performance in our algorithm. A random subset of nodes were chosen to be participating nodes in CS. The entry time and duration of CS for each participating node was randomly chosen.

For the purpose of comparison, we have used four performance metrics. Normalized average synchronization delay is computed as the average synchronization delay divided by the average per-hop communication delay. The next parameter used, normalized average response time, is taken as average response time computed as the difference between the time that a site requested for entering CS and the time at which the site exited CS, upon average per-hop communication delay. Normalized average time per CS is computed as the difference between the end of N CS computations and the start of the first among the N requests for CS, divided by the number of CS requests and average per-hop communication delay. The fourth parameter that we compared with in average number of hops per critical section. While the earlier three metrics provide an intuitive feeling of the performance of a DME algorithm in most cases, we observed that a typical algorithm that spans multiple regions would suffer a little slack in performance for the serving requests while no other node is contending for critical section. The normalized average time per critical section is a better measure for observing the performance of an algorithm over a large number of CS requests. Unlike synchronization delay or response time, this metric considers the time needed for all messages related to the critical section, including the messages needed after completing CS.

Figure 5.12 shows the comparison between the three algorithms ([Erciyes & Dagdeviren 2012, Gupta *et al.* 2012] and RAD). As can be observed, the three algorithms perform well with increase in the size of the networks with respect to synchronization delay. As the size of the network increases we observed that the excess delay in reaching to the next node requesting for CS via the intermediate node is less pronounced in [Erciyes & Dagdeviren 2012] and in RAD. [Gupta *et al.* 2012] were effectively sending the cluster release messages to the cluster head of the node with the next higher priority, once the requests were getting queued.

The average response time for the algorithms are maintained in almost the same levels for different sizes of the network in Figure 5.13. The results of the truly permission-based approaches in RAD and [Erciyes & Dagdeviren 2012] follow almost the same trajectory.

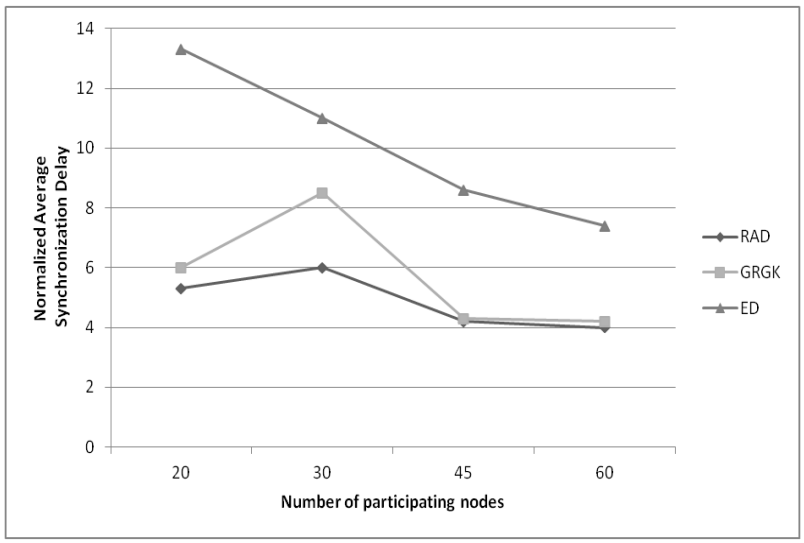

Figure 5.12: Normalized Average Synchronization Delay vs Number of nodes. -

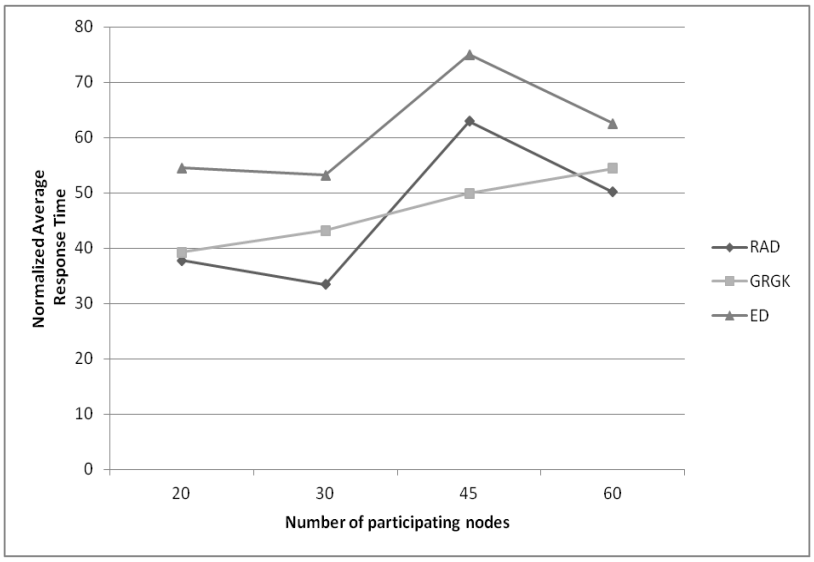

Figure 5.13: Normalized Average Response Time vs Number of nodes. -

Our algorithm performs slightly better in small to medium sizes of the network as seen from Figure 5.13.

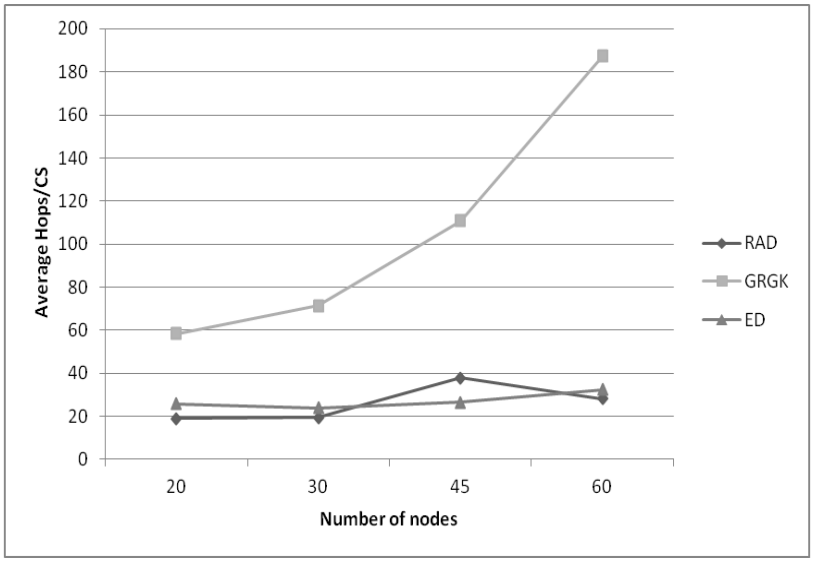

Figure 5.14: Average Hops / Critical Section vs Number of nodes. -

With respect to average hops per critical section shown in Figure 5.14, our algorithm seems to lag slightly behind [Erciyes & Dagdeviren 2012]. This is due to the extra set of packets that will get transmitted for maintaining fault tolerance. Our algorithm still performs much better than [Gupta *et al.* 2012].

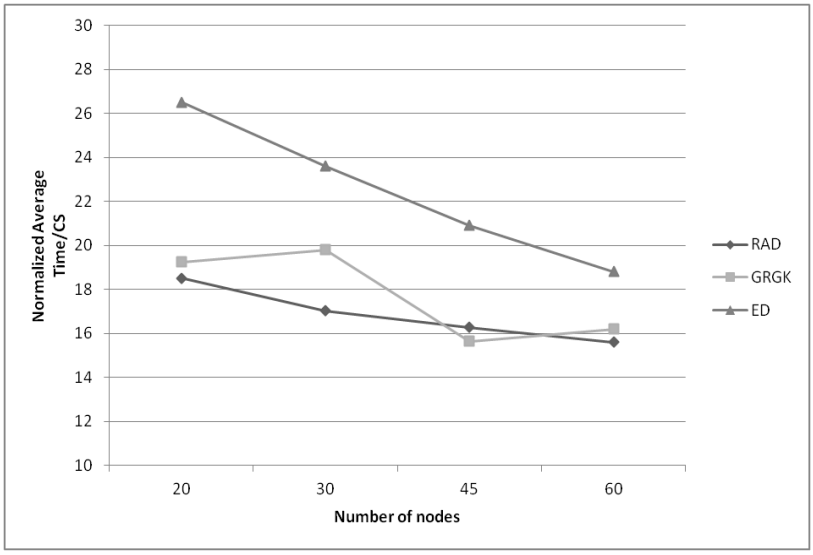

Figure 5.15: Normalized Average Time per Critical Section vs Number of nodes. -

In Figure 5.15, the normalized average time per critical section is shown. In [Gupta

et al. 2012] and [Erciyes & Dagdeviren 2012], the node only needs to inform its cluster head after exiting from CS. In RAD, the the node that exits will eventually inform all affected arbitrators. But this has not affected the performance of the algorithm as compared to the other contemporary non-fault-tolerant algorithms. As can be observed, the time that a node would typically need to complete its CS is steadily reducing as the size of the network increases. Our algorithm performs much better than [Erciyes & Dagdeviren 2012], while it is almost same for larger size of the network. The slight degradation is due to the voids that we had introduced in our test cases.

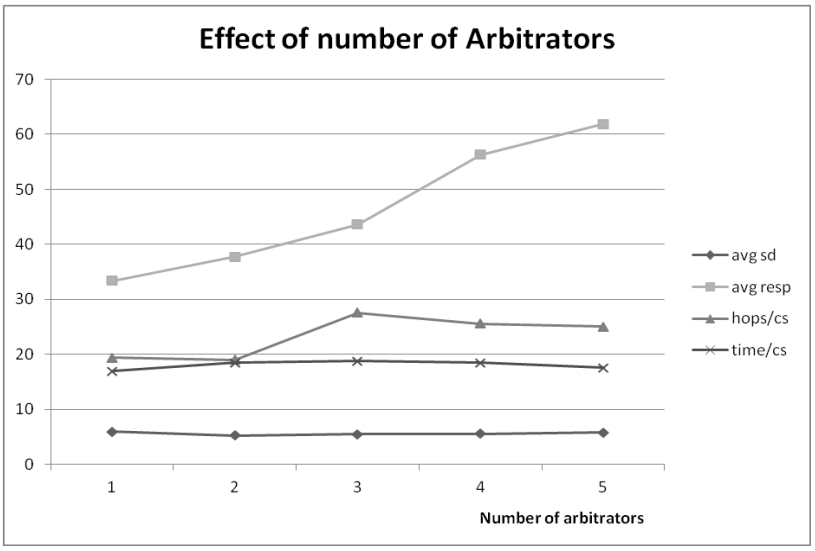

Figure 5.16: Effect of number of arbitrator nodes - impact on major parameters

In Figure 5.16, the effect of the number of arbitrator nodes with regards to the various parameters mentioned so far is shown. The comparison for this chart was done among networks having the same number of total nodes, but with different number of arbitrator nodes used. As can be seen from the figure, the average response time grows lineraly wuth respect to the number of arbitrator nodes used. If the number of arbitrator nodes are more in the network, then the requests would be sent to more nodes than are necessary. This figure also indicates the necessity of having a large region, so that the number of arbitrator nodes can be suitably reduced. Figure 5.16 also indicates that other parameters such as the average synchronization delay, average time per CS and average number of hops per CS were mostly flat even with varying number of arbitrators. The results thus indicate that having multiple regions and multiple number of arbitrator nodes does not

have any detrimental effect with respect to these key parameters.

5.6 Conclusion

In this chapter, we presented an initialization algorithm for dynamic information sets of "look-ahead" technique, and two permission-based DME algorithms. The initialization algorithm presented is simple one, without the need for a centralized initiator node. We proved that the initialization method will lead to convergence of the dynamic information set across the nodes in the MANET.

In the first algorithm, we introduced the new message called *HOLD* message to handle arbitrary length execution in CS. This algorithm is a fault-tolerant DME algorithm that can handle crashing of node in CS. The *HOLD* message also enabled us to handle transient node and link failures. Unlike token-based algorithms, there is no additional overhead involved in maintaining a logical structure. The use of "look-ahead" technique ensures that only the currently contending nodes need to be sent the additional *HOLD* message. To achieve better fault-tolerance, our algorithm makes a slight trade-off with message complexity. However, it is desirable to reduce the number of sites that need to be sent the *HOLD* message if possible, so that we can improve the message complexity of the algorithm. We present a different approach for permission based DME using *HOLD* messages in the next section.

In the second algorithm, we have used *HOLD* messages to handle DME across multiple regions as well as to provide fault-tolerance. Unlike other permission-based algorithms using multiple clusters, we use only three message types to handle the DME problem in a MANET spanning multiple regions. Our algorithm provides the same message complexity as [Maekawa 1985] in the best case. As far as we are aware of, this is also the first arbitration-based algorithm that has attempted to split the nodes into near-equal regions augmented with the look-ahead technique for MANETs. The algorithm proposed is the first fault-tolerant region-based distributed mutual exclusion algorithm for mobile ad-hoc networks.

Chapter 6

Conclusions & Future Work

In this chapter, we summarize our work, provide conclusions and discuss some possible directions for future research.

In this thesis, we addressed the QoS-aware multicast routing problem for mobile ad-hoc networks(MANETs). Multicast routing forms the basis for group-oriented communication scenarios, like collaborative computing and multimedia streaming. In a MANET, the characteristics of the underlying network like mobility, frequent transient link failures and resource constraints, pose serious challenges to the multicast routing problem. Due to the growing demand for energy efficiency while using mobile devices, focus of this thesis is restricted to energy-based QoS parameters. We looked into the problem of reliable QoS-aware routing for multicast communication in MANETs. We found the scope for using virtual-force technique in multicast communication for MANETs. We added the notion of dampening forces and the ability to handle multiple destinations to the virtual force technique for solving the multicast routing problem in MANETs. We explored this technique further and presented three QoS-aware multicast routing algorithms in this thesis.

Due to the innate behaviour of the MANETs, the resource allocation strategies for fixed networks cannot be applied on these networks. The objective of the Distributed Mutual Exclusion(DME) problem is to provide access to shared critical resources amongst different mobile nodes in a MANET. Among the approaches available for dealing with this problem, we looked into the use of permission-based DME algorithms. One of the most promising techniques used in the permission-based approach is the "lookahead"

technique using dynamic information sets. While applying this technique in the mobile ad-hoc environment, the key challenges include effectively dealing with link failures and scalability. We introduced a new message type(*HOLD* message), an adaptive timeout handling mechanism, and the notion of an arbitrator node for multiple regions to improve the "look ahead" technique. We presented two new algorithms to solve the DME problem in MANETs.

In the remainder of this chapter, we summarize the contributions of this thesis and conclude.

6.1 Summary and Conclusions

6.1.1 QoS-aware Routing Algorithms

While looking into the problem of multicast routing in MANETs, we explored a few interesting questions before we delved into the problem of QoS-aware multicast routing.

The effectiveness of any QoS-aware routing algorithm depends on the utility of the underlying energy metric used. We proposed a new cost metric for improving network life time. We proved the concept by using a proactive routing protocol. In terms of network lifetime, the new metric resulted in $10\% - 65\%$ improvement over two other techniques. Apart from improving network life time, the algorithm proposed was capable of maximizing the average residual energy of the network and minimizing the variance of the power of the nodes in a MANET.

Due to the nature of the underlying network, apart from the use of an effective cost metric, we also looked at the route maintenance problem. We used the information of previous packet flow in conjunction with a Bayesian approach to enable appropriate route maintenance over regions of the MANET. The Bayesian approach used could leverage the affinity of nodes to efficiently forward routing requests to the rest of the nodes. We showed that our energy-efficient on-demand routing protocol improved the delivery ratio for higher mobility rates and effective number of control packets used through simulations.

Another issue for QoS-aware routing is the inclusion of energy-harvesting devices in ad-hoc networks. We focus on the problem for the domain of wireless sensor networks,

and proposed a new model that uses energy budgeting to enable energy efficient communication in mobile nodes having energy-harvesting devices. The model captured the impact of the relative hop number and the relative activity of nodes routing through a particular node. This model led to identification of the changes needed in the basic energy metric in the presence of energy harvesting devices in the nodes of the network.

Having briefly explored these questions, we addressed the QoS-aware multicast routing problem in MANETs. The virtual force technique was applied only for unicast routing problem. We adapted the technique to handle multiple destinations in a multicast communication. We used the notion of dampening forces to handle QoS parameters in multicast routing for MANETs. In our initial attempt, we looked at adapting virtual force directly on the multicast destinations that formed part of the multicast communication. This algorithm successfully created a relatively minimal multicast tree of length $len(T)$. Our simulation results indicated that the technique fielded better results as compared to its peers while looking at energy-related QoS parameters. However, the algorithm was having exponential complexity. To improve upon this algorithm, we approached the virtual force technique from a different angle.

We divided the vicinity(an arbitrary k-hop neighbourhood of the current node) of every node into sectors and applied virtual force to channel the multicast path through these sectors. For dividing the vicinity into sectors, our algorithm used an adaptive variable, α, that can be tuned based on the network. α is typically chosen to be a small constant. Instead of a time complexity of $O(|T| \cdot \max\{2^m, \Delta\})$ for the previous algorithm, this algorithm has a time complexity of $O(|T| \cdot \alpha \cdot \max\{m, \frac{\Delta}{\alpha}\})$, where $|T|$ is the total number of nodes in the resultant multicast tree, $m = |M|$, Δ is the maximum degree of a node in the network. The trade-off was in terms of the relatively minimal Steiner tree generated. The variable angle sectors used in this $(1 + \alpha)$-competitive multicast routing algorithm allowed for improving performance parameters like average normalized residual energy and normalized cost per hop per data packet.

We looked at the multicast routing problem from a different perspective so that we could effectively handle the case of *voids* in the network. We divided the network into regions of at most four-hop radius. We applied the virtual force technique to route

packets to multicast destinations that are spread among these regions. We adapted the ideas of spine routing, well separated planar decomposition(WSPD) and virtual geographic circuits to enable routing among the regions using the virtual force technique. Though this $(1 + \alpha + \frac{m}{len(T)})$-competitive algorithm has a slightly higher time complexity of $O(|T| \cdot \alpha \cdot \max\{m, \frac{\Delta}{\alpha}\} + len(T) \cdot m)$, the simulation results indicate that the algorithm was able to leverage the positive influences of spine routing and WSPD to improve parameters such as average normalized residual energy and normalized cost per hop per data packet.

The multicast routing algorithms proposed by us generated relatively minimal Steiner trees for QoS-aware multicast communication. Our MRAV algorithm was creating $15\% - 25\%$ longer multicast trees with respect to the optimal case, VFM algorithm(for $\alpha = 3$) creating $6\% - 15\%$ longer and VMT algorithm $15\% - 25\%$ longer. In terms of normalized cost per hop per data packet, VFM algorithm(for $\alpha = 3$) was providing $5\% - 12\%$ improvement over other comparable algorithms, while VMT algorithm was providing $30\% - 45\%$ improvement. For another key parameter, average normalized residual energy, we compared with other algorithms that used the approach of sectors/quadrants for multicast routing. When we conducted experiments until the first peer algorithm fell below a 50% residual energy threshold, our algorithms provided an improvement in average normalized residual energy of approximately 20%, 24% and 44% respectively over the peer algorithm.

6.1.2 Distributed Mutual Exclusion Algorithms for MANETs

We explored resource allocation problem in MANETs. In the resource allocation problem, we focussed on Distributed Mutual Exclusion(DME) problem in MANETs. We introduced a mechanism to initiate dynamic information sets used in the "lookahead" technique. We proposed two new algorithms that used the permission-based approach for solving the DME problem. We used the notion of regions and the notion of arbitrator nodes in the "look-ahead" technique to handle DME problem in MANETs.

In the first algorithm, we addressed the problem of fault-tolerance with the introduc-

tion of a new message type($HOLD$ message) and adaptive timeout handling mechanism. We theoretically proved that fault-tolerance is guaranteed in the event of node or link failures. In terms of ability to handle faults, our algorithm was equally good as the peer group. With the help of the timeout mechanism introduced in this algorithm, our algorithm avoids the need for the implicit assumption that all nodes take the same time to execute critical section(CS). If we assume that the node in CS miscalculates the timeout m times, and there are w nodes in Q_{HOLD}, then the message complexity of the algorithm will tend to be $2 \cdot (\Phi - 1) + m \cdot w$, where Φ is the number of nodes participating in CS. The synchronization delay for this algorithm is ΦT, which is the same as other algorithms using the "look ahead" technique.

In the second algorithm, we divided the network into regions of at most four-hop radius, and used arbitrator nodes to handle DME across the regions. If there are r regions with a maximum of Φ_{max} participating nodes within each region; m number of wrong timeout computations; and w sites in all the Q_{HOLD}; then the message complexity of the algorithm is $2 \cdot r \cdot (\Phi_{max} - 1) + m \cdot w$ messages. The average case synchronization delay for this algorithm is $2 \cdot \delta_{comm}^{max}$, where δ_{comm}^{max} is the maximum communication delay between any two nodes in the network. In terms of normalized synchronization delay, this algorithm performs $25\% - 100\%$ better than the peer group. As the related algorithms did not have fault-tolerance as well, we were not able to get a conclusive result with respect to average number of hops per CS. We observed that the parameters such as average synchronization delay, average time per CS and average hops per critical section were independent of the number of arbitrator nodes used among networks of equal number of nodes. As the number of arbitrator nodes are increased, the average response time increased linearly.

We theoretically proved the correctness of the two DME algorithms for MANETs. We observed from our simulation results that our second DME algorithm performed better than its peers with respect to key parameters such as normalized average synchronization delay, normalized average time per CS and average number of hops per CS. Our algorithms used just 5 message types in total to deal with both DME and fault-tolerance, which is smaller than the peer group. To include fault-tolerance, our algorithms incurred an extra cost of $m \cdot w$, where m is the number of wrong timeout computations; and w is the number of sites in all the Q_{HOLD}, as a trade-off. In other related parameters like

average response time, average hops per CS and normalized average time per CS, the second DME algorithm performed $20\% - 27\%$ better in terms of average response time and $20\% - 30\%$ better in terms of average time per CS with respect to some of the recently proposed peer algorithms.

6.2 Scope for Future Research

In the virtual-force based multicast routing algorithms presented by us, there is further need to explore the impact of the adaptive parameter α. The relationship between network density and α is to be explored. We believe that an adaptive mechanism in which the value of α is changed according to the local network density might reduce the time complexity of the algorithm further. However, we have not explored this aspect in our work.

The notion of virtual force can be applied to solve any problem provided there is an appropriate means to model the parameters of the problem into equivalent force values. The multi-radio multi-channel routing problem is one interesting problem where it might be possible to apply the virtual force technique in conjunction with dampening forces. Though we haven't explored this problem further, we believe that a suitable combination of per-radio link dampening forces and a suitable application of inertia of the packet could be adapted to solve multi-radio multi-channel routing problem. There is also scope for employing this technique in related domains like Vehicular ad-hoc networks, delay tolerant networks and inter-planetary networks to solve similar routing problems.

Till now, we have assumed that all mobile nodes are using omni-directional antenna. With the increasing research in directional antennas with lower signal-to-noise ratios and higher transmission range, we need to explore the changes required to enable our algorithm in the presence of other antenna technologies.

Another aspect we have not considered is the impact of security on our routing algorithms. We believe that by suitably modelling trust relationships among the mobile nodes, a private secure overlay network on top of the underlying MANET could be established. This could allow establishment of secure private networks within a wider MANET especially in military environments. As tactical networks forms a crucial application domain for the deployment of MANETs, we would like to explore trust-based communication in

conjunction with message security.

All our algorithms have been validated using suitable simulation environments. We would like to explore the use of our algorithms in real-life rural networks or other similar environments where solar powered energy-harvesting devices are typically used.

As part of our work related to the distributed mutual exclusion problem, we had introduced the notion of arbitration among fixed-diameter regions. Though we have used arbitration in permission-based approach, we would also like to explore the possibility of arbitration in token-based approach to solve the stated problem. The principle of arbitration fitted well with this problem primarily because there is a need for granting permission to proceed with the execution of critical section. We would like to explore whether this technique can also be applied on allied problems like distributed locking, distributed shared memory in ad-hoc environments. We would also like to expand the notions of arbitration approach, 2-MIS regions and virtual force with or without the inclusion of dampening forces to other distributed problems in similar self-organizing networks.

References

[Ahmed & Kanhere 2010] S. Ahmed and S.S. Kanhere. *A Bayesian Routing Framework for Delay Tolerant Networks*. In Wireless Communications and Networking Conference (WCNC), 2010 IEEE, pages 1 –6, april 2010. 84, 85

[Ahn *et al.* 2002] Gahng-Seop Ahn, A.T. Campbell, A. Veres and Li-Hsiang Sun. *SWAN: service differentiation in stateless wireless ad hoc networks*. In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, pages 457–466 vol.2, 2002. 54

[Alsheakhali & Awad 2008] M. Alsheakhali and F. Awad. *Traffic-Aware and Low-Overhead Routing Protocol for MANETs*. In Wireless Days, 2008. WD '08. 1st IFIP, pages 1–5, 2008. 37

[Alzoubi *et al.* 2002a] Khaled M. Alzoubi, Peng-Jun Wan and Ophir Frieder. *Message-optimal Connected Dominating Sets in Mobile Ad Hoc Networks*. In Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing, MobiHoc '02, pages 157–164, New York, NY, USA, 2002. ACM. 48, 49, 124

[Alzoubi *et al.* 2002b] Khaled M. Alzoubi, Peng-Jun Wan and Ophir Frieder. *Message-optimal connected dominating sets in mobile ad hoc networks*. In Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, MobiHoc '02, pages 157–164, New York, NY, USA, 2002. ACM. 157

[Alzoubi *et al.* 2002c] K.M. Alzoubi, Peng-Jun Wan and O. Frieder. *New distributed algorithm for connected dominating set in wireless ad hoc networks*. In System Sciences,

2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on, pages 3849–3855, 2002. 48

[Amis *et al.* 2000] A.D. Amis, R. Prakash, T.H.P. Vuong and D.T. Huynh. *Max-min d-cluster formation in wireless ad hoc networks*. In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 1, pages 32–41 vol.1, 2000. 48

[Awasthi 2006] Prasoon Awasthi. *Approaches to Distributed Mutual Exclusion in Mobile Ad Hoc Networks*. Rapport technique, Department of Computer Science and Information Systems, BITS-Pilani, Raj, IND, 2006. 142

[Badis *et al.* 2003] Hakim Badis, Anelise Munaretto, Khaldoun Al Agha and Guy Pujolle. *QoS for ad hoc networking based on multiple-metric: Bandwidth and delay*. In IFIP MWCN: International Workshop On Mobile and Wireless Communications Networks, pages 55–59, 2003. 54

[Badrinath *et al.* 1994] B. R. Badrinath, A. Acharya and T. Imielinski. *Structuring distributed algorithms for mobile hosts*. In Distributed Computing Systems, 1994., Proceedings of the 14th International Conference on, pages 21–28, 1994. 64

[Baldoni *et al.* 2002] R. Baldoni, A. Virgillito and R. Petrassi. *A distributed mutual exclusion algorithm for mobile ad-hoc networks*. In Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on, pages 539–544, 2002. 65

[Basagni *et al.* 1998] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk and Barry A. Woodward. *A Distance Routing Effect Algorithm for Mobility (DREAM)*. In Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom '98, pages 76–84, New York, NY, USA, 1998. ACM. 34

[Benchaïba *et al.* 2004] M. Benchaïba, A. Bouabdallah, N. Badache and M. Ahmed-Nacer. *Distributed mutual exclusion algorithms in mobile ad hoc networks: an overview*. SIGOPS Oper. Syst. Rev., vol. 38, no. 1, pages 74–89, January 2004. 25, 61

[Bern & Plassmann 1989] Marshall Bern and Paul Plassmann. *The Steiner problem with edge lengths 1 and 2*. Information Processing Letters, vol. 32, no. 4, pages 171 – 176, 1989. 24

[Borradaile *et al.* 2009] Glencora Borradaile, Philip Klein and Claire Mathieu. *An O(n log n) approximation scheme for Steiner tree in planar graphs*. ACM Trans. Algorithms, vol. 5, no. 3, pages 31:1–31:31, July 2009. 24, 104

[Bose *et al.* 2001] Prosenjit Bose, Pat Morin, Ivan Stojmenović and Jorge Urrutia. *Routing with Guaranteed Delivery in Ad Hoc Wireless Networks*. Wirel. Netw., vol. 7, no. 6, pages 609–616, November 2001. 43

[Bouillageut *et al.* 2009] M. Bouillageut, L. Arantes and P. Sens. *A Timer-Free Fault Tolerant K-Mutual Exclusion Algorithm*. In Dependable Computing, 2009. LADC '09. Fourth Latin-American Symposium on, pages 41 –48, sept. 2009. 66

[Broch *et al.* 1998] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu and Jorjeta Jetcheva. *A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols*. In Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom '98, pages 85–97, New York, NY, USA, 1998. ACM. 14

[Calinescu 2003] Gruia Calinescu. *Computing 2-Hop Neighborhoods in Ad Hoc Wireless Networks*. In Samuel Pierre, Michel Barbeau and Evangelos Kranakis, editeurs, Ad-Hoc, Mobile, and Wireless Networks, volume 2865 of *Lecture Notes in Computer Science*, pages 175–186. Springer Berlin Heidelberg, 2003. 51, 52, 124

[Callahan & Kosaraju 1995] Paul B. Callahan and S. Rao Kosaraju. *A Decomposition of Multidimensional Point Sets with Applications to K-nearest-neighbors and N-body Potential Fields*. J. ACM, vol. 42, no. 1, pages 67–90, January 1995. 52, 123

[Chang *et al.* 1990] Y.-I. Chang, M. Singhal and M.T. Liu. *A fault tolerant algorithm for distributed mutual exclusion*. In Reliable Distributed Systems, 1990. Proceedings., Ninth Symposium on, pages 146–154, 1990. 63

[Chen & Welch 2002] Yu Chen and Jennifer L. Welch. *Self-stabilizing mutual exclusion using tokens in mobile ad hoc networks*. In Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications, DIALM '02, pages 34–42, New York, NY, USA, 2002. ACM. 61, 137

[Clark *et al.* 1990] Brent N. Clark, Charles J. Colbourn and David S. Johnson. *Unit disk graphs*. Discrete Mathematics, vol. 86, no. 1âĂŞ3, pages 165 – 177, 1990. 18, 41

[Dagdeviren *et al.* 2005] Orhan Dagdeviren, Kayhan Erciyes and Deniz Cokuslu. *Merging Clustering Algorithms in Mobile Ad Hoc Networks*. In Goutam Chakraborty, editeur, Distributed Computing and Internet Technology, volume 3816 of *Lecture Notes in Computer Science*, pages 56–61. Springer Berlin Heidelberg, 2005. 67, 138

[Das & Bharghavan 1997] B. Das and V. Bharghavan. *Routing in ad-hoc networks using minimum connected dominating sets*. In Communications, 1997. ICC '97 Montreal, Towards the Knowledge Millennium. 1997 IEEE International Conference on, volume 1, pages 376–380 vol.1, 1997. 48

[Derhab & Badache 2008] A. Derhab and N. Badache. *A distributed mutual exclusion algorithm over multi-routing protocol for mobile ad hoc networks*. The International Journal of Parallel, Emergent and Distributed Systems, vol. 23, no. 3, pages 197–218, 2008. 61

[Dhamdhere & Kulkarni 1994] Dhananjay M. Dhamdhere and Sandeep S. Kulkarni. *A token based k-resilient mutual exclusion algorithm for distributed systems*. Information Processing Letters, vol. 50, no. 3, pages 151 – 157, 1994. 64

[Erciyes & Dagdeviren 2012] K. Erciyes and O. Dagdeviren. *A Distributed MUTUAL EXCLUSION ALGORITHM FOR MOBILE AD HOC NETWORKS*. "International Journal of Computer Networks and Communications", vol. 4, no. 2, pages 129–148, March 2012. 67, 138, 172, 173, 175, 176

[Erciyes 2004] Kayhan Erciyes. *Distributed Mutual Exclusion Algorithms on a Ring of Clusters*. In Antonio LaganÃą, MarinaL. Gavrilova, Vipin Kumar, Youngsong Mun, C.J.Kenneth Tan and Osvaldo Gervasi, editeurs, Computational Science and Its

Applications âĂŞ ICCSA 2004, volume 3045 of *Lecture Notes in Computer Science*, pages 518–527. Springer Berlin Heidelberg, 2004. 61, 66, 138

[Finn 1987] G. G. Finn. *Routing and Addressing Problems in Large Metropolitan-Scale Internetworks*. Rapport technique ISI/RR-87-180, Information Sciences Institute, March 1987. 42

[Fotopoulou-Prigipa & McDonald 2004] S. Fotopoulou-Prigipa and A.B. McDonald. *GCRP: geographic virtual circuit routing protocol for ad hoc networks*. In Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on, pages 416 – 425, oct. 2004. 46, 47, 60, 106, 123

[Funke *et al.* 2004] Stefan Funke, Domagoj Matijevic and Peter Sanders. *Constant time queries for energy efficient paths in multi-hop wireless networks*. In University of Bologna, pages 97–111, 2004. 53

[Gao & Zhang 2005] J. Gao and L. Zhang. *Well-Separated Pair Decomposition for the Unit-Disk Graph Metric and Its Applications*. SIAM Journal on Computing, vol. 35, no. 1, pages 151–169, 2005. 52

[Gao 2004] Jie Gao. *Hierarchical data structures for mobile networks*. PhD thesis, stanford university, 2004. 51

[Garcia-Luna-Aceves & Madruga 1999] J.J. Garcia-Luna-Aceves and E.L. Madruga. *The core-assisted mesh protocol*. Selected Areas in Communications, IEEE Journal on, vol. 17, no. 8, pages 1380–1394, 1999. 59

[Gilbert & Pollak 1968] E. Gilbert and H. Pollak. *Steiner Minimal Trees*. SIAM Journal on Applied Mathematics, vol. 16, no. 1, pages 1–29, 1968. 24, 59, 104, 106, 116, 120, 121, 128, 132

[Giordano *et al.* 2001] Silvia Giordano, Ivan Stojmenovic and Ljubica Blazevic. *Position Based Routing Algorithms For Ad Hoc Networks: A Taxonomy*. In Ad Hoc Wireless Networking, pages 103–136. Kluwer, 2001. 40

[Goodrich & Tamassia 1998] Michael T. Goodrich and Roberto Tamassia. Data structures and algorithms in java with cdrom. John Wiley & Sons, Inc., New York, NY, USA, 1st édition, 1998. 32

[Goodrich & Tamassia 2008] Michael T. Goodrich and Roberto Tamassia. Data structures and algorithms in java. Wiley Publishing, New York, NY, USA, 5th édition, 2008. 72, 73

[Guimaraes *et al.* 2004] Rafael Guimaraes, Julián Morillo, Llorenç Cerda, J Barceló and Jorge Garcıa. *Quality of service for mobile ad-hoc networks: an overview*. 2004. 53

[Gupta & Younis 2003] G. Gupta and M. Younis. *Load-balanced clustering of wireless sensor networks*. In Communications, 2003. ICC 2003. IEEE International Conference on, volume 3, pages 1848 – 1852 vol.3, may 2003. 99

[Gupta *et al.* 2012] A. Gupta, BVR Reddy, U. Ghosh and A. Khanna. *A Permission-based Clustering Mutual Exclusion Algorithm for Mobile Ad-Hoc Networks*. "International Journal of Engineering Research and Applications", vol. 2, no. 4, pages 019–026, July 2012. 67, 172, 173, 175, 176

[Haas *et al.* 2002] Zygmunt J. Haas, Marc R. Pearlman and Prince Samar. *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. IETF Internet Draft, July 2002. 36

[Hoon & Seok-Yeol 2007] OH Hoon and YUN Seok-Yeol. *Proactive Cluster-Based Distance Vector (PCDV) Routing Protocol in Mobile Ad Hoc Networks*. IEICE transactions on communications, vol. 90, no. 6, pages 1390–1399, 2007. 46, 128

[Issariyakul & Hossain 2011] Teerawat Issariyakul and Ekram Hossain. Introduction to network simulator ns2. Springer Publishing Company, Incorporated, 2nd édition, 2011. 128

[Jacquet *et al.* 2001] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum and L. Viennot. *Optimized link state routing protocol for ad hoc networks*. In Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International, pages 62 – 68, 2001. 34, 37, 54

[Johnson & Maltz 1996] DavidB. Johnson and DavidA. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. In Tomasz Imielinski and HenryF. Korth, editeurs, Mobile Computing, volume 353 of *The Kluwer International Series in Engineering and Computer Science*, pages 153–181. Springer US, 1996. 14, 32, 33, 35, 56

[Junhai *et al.* 2009] Luo Junhai, Ye Danxia, Xue Liu and Fan Mingyu. *A survey of multicast routing protocols for mobile Ad-Hoc networks*. Communications Surveys Tutorials, IEEE, vol. 11, no. 1, pages 78 –91, quarter 2009. 57, 104

[Kang & Ko 2010] Byung-Seok Kang and In-Young Ko. *Effective route maintenance and restoration schemes in mobile ad hoc networks*. Sensors, vol. 10, no. 1, pages 808–821, 2010. 37

[Kannan & Iyengar 2004] R. Kannan and S.S. Iyengar. *Game-theoretic models for reliable path-length and energy-constrained routing with data aggregation in wireless sensor networks*. Selected Areas in Communications, IEEE Journal on, vol. 22, no. 6, pages 1141 – 1150, aug. 2004. 57

[Kansal *et al.* 2007] Aman Kansal, Jason Hsu, Sadaf Zahedi and Mani B. Srivastava. *Power management in energy harvesting sensor networks*. ACM Trans. Embed. Comput. Syst., vol. 6, no. 4, September 2007. 56, 99

[Karp & Kung 2000] Brad Karp and H. T. Kung. *GPSR: greedy perimeter stateless routing for wireless networks*. In Proceedings of the 6th annual international conference on Mobile computing and networking, MobiCom '00, pages 243–254, New York, NY, USA, 2000. ACM. 44, 113, 120

[Karp 1975] R. Karp. *On the computational complexity of combinatorial problems.* Networks, vol. 5, no. 1, pages 45–68, 1975. 24, 104

[Kazantzidis *et al.* 2001] M. Kazantzidis, M. Gerla and Sung-Ju Lee. *Permissible throughput network feedback for adaptive multimedia in AODV MANETs*. In Communications, 2001. ICC 2001. IEEE International Conference on, volume 5, pages 1352–1356 vol.5, 2001. 54

[Ko & Vaidya 1999] Young-Bae Ko and N.F. Vaidya. *Geocasting in mobile ad hoc networks: location-based multicast algorithms*. In Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on, pages 101–110, 1999. 38

[Ko & Vaidya 2000] YoungâĂŘBae Ko and NitinH. Vaidya. *LocationâĂŘAided Routing (LAR) in mobile ad hoc networks*. Wireless Networks, vol. 6, no. 4, pages 307–321, 2000. 32, 36

[Kranakis *et al.* 1999] Evangelos Kranakis, Harvinder Singh and Jorge Urrutia. *Compass Routing on Geometric Networks*. 1999. 43

[Lamport 1986] Leslie Lamport. *"The mutual exclusion problem: partII statement and solutions"*. J. ACM, vol. 33, no. 2, pages 327–348, April 1986. 26, 136

[Le Lann 1977] Gérard Le Lann. *Distributed Systems-Towards a Formal Approach.* In IFIP Congress, volume 7, pages 155–160. Toronto, 1977. 62

[Lee & Kim 2000] Seungjoon Lee and Chongkwon Kim. *Neighbor supporting ad hoc multicast routing protocol*. In Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on, pages 37–44, 2000. 59

[Lee *et al.* 2000] Seoung-Bum Lee, Gahng-Seop Ahn, Xiaowei Zhang and Andrew T. Campbell. *INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks*. Journal of Parallel and Distributed Computing, vol. 60, no. 4, pages 374 – 406, 2000. 55

[Lee *et al.* 2002] Sung-Ju Lee, William Su and Mario Gerla. *On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks*. Mobile Networks and Applications, vol. 7, no. 6, pages 441–453, 2002. 34

[Lewin-Eytan *et al.* 2007] Liane Lewin-Eytan, Joseph (Seffi) Naor and Ariel Orda. *Maximum-lifetime routing: system optimization & game-theoretic perspectives*. In Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '07, pages 160–169, New York, NY, USA, 2007. ACM. 55

[Li *et al.* 2000] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger and Robert Morris. *A Scalable Location Service for Geographic Ad Hoc Routing*. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom '00, pages 120–130, New York, NY, USA, 2000. ACM. 38

[Liao *et al.* 2001] Wen-Hwa Liao, Jang-Ping Sheu and Yu-Chee Tseng. *GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks*. Telecommunication Systems, vol. 18, no. 1-3, pages 37–60, 2001. 38

[Liu & Wu 2006] Cong Liu and Jie Wu. *SWING: Small World Iterative Navigation Greedy Routing Protocol in MANETs*. In Computer Communications and Networks, 2006. ICCCN 2006. Proceedings.15th International Conference on, pages 339 –350, oct. 2006. 41, 46, 47, 60, 105, 106, 123

[Liu & Wu 2009] Cong Liu and Jie Wu. *Virtual-Force-Based Geometric Routing Protocol in MANETs*. Parallel and Distributed Systems, IEEE Transactions on, vol. 20, no. 4, pages 433 –445, april 2009. 46, 47, 60, 61, 105, 106, 107, 109, 110, 115, 116, 120, 122, 123

[Liu *et al.* 2004] Wei Liu, X. Chen, Yuguang Fang and J.M. Shea. *Courtesy piggybacking: supporting differentiated services in multihop mobile ad hoc networks*. Mobile Computing, IEEE Transactions on, vol. 3, no. 4, pages 380–393, 2004. 54

[Luger 2001] George F. Luger. Artificial intelligence: Structures and strategies for complex problem solving. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 4th édition, 2001. 84, 85

[Madruga & Garcia-Luna-Aceves 2005] Ewerton L Madruga and Joaquin J Garcia-Luna-Aceves. *Core assisted mesh protocol for multicast routing in ad-hoc Networks*, July 12 2005. US Patent 6,917,985. 59

[Maekawa 1985] Mamoru Maekawa. *A N algorithm for mutual exclusion in decentralized systems*. ACM Trans. Comput. Syst., vol. 3, no. 2, pages 145–159, May 1985. 61, 65, 156, 159, 170, 172, 177

[Masum *et al.* 2010] Salahuddin Mohammad Masum, Mohammad Mostofa Akbar, Amin Ahsan Ali and Mohammad Ashiqur Rahman. *A consensus-based ℓ-Exclusion algorithm for mobile ad hoc networks*. Ad Hoc Networks, vol. 8, no. 1, pages 30 – 45, 2010. 144

[Mauve *et al.* 2001] M. Mauve, J. Widmer and H. Hartenstein. *A survey on position-based routing in mobile ad hoc networks*. Network, IEEE, vol. 15, no. 6, pages 30–39, 2001. 38

[Mauve *et al.* 2003] Martin Mauve, Holger Füssler, Jörg Widmer and Thomas Lang. *Position-based multicast routing for mobile Ad-hoc networks*. SIGMOBILE Mob. Comput. Commun. Rev., vol. 7, no. 3, pages 53–55, July 2003. 38, 60, 106

[Mohapatra & Krishnamurthy 2010] Prasant Mohapatra and Srikanth Krishnamurthy. Ad hoc networks: Technologies and protocols. Springer Publishing Company, Incorporated, 1st édition, 2010. 8

[Murthy & Manoj 2004] C.S.R. Murthy and BS Manoj. Ad hoc wireless networks: Architectures and protocols. Prentice Hall, 2004. 15, 16, 34, 44, 61, 62

[Niyato *et al.* 2007] D. Niyato, E. Hossain, M.M. Rashid and V.K. Bhargava. *Wireless sensor networks with energy harvesting technologies: a game-theoretic approach to optimal energy management*. Wireless Communications, IEEE, vol. 14, no. 4, pages 90 –96, august 2007. 56, 57

[Parameswaran & Hota 2010] M. Parameswaran and C. Hota. *A novel permission-based reliable distributed mutual exclusion algorithm for MANETs*. In Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On, pages 1 –6, sept. 2010. 172

[Parameswaran & Hota 2013] M. Parameswaran and C. Hota. *Arbitration-based reliable distributed mutual exclusion for Mobile Ad-hoc Networks*. In Modeling Optimization in Mobile, Ad Hoc Wireless Networks (WiOpt), 2013 11th International Symposium on, pages 380–387, May 2013. 172

[Perkins & Bhagwat 1994] Charles E. Perkins and Pravin Bhagwat. *Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers*. SIGCOMM Comput. Commun. Rev., vol. 24, no. 4, pages 234–244, October 1994. 32

[Perkins & Royer 1999] C.E. Perkins and E.M. Royer. *Ad-hoc on-demand distance vector routing*. In Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on, pages 90 –100, feb 1999. 14, 22, 32, 35, 44, 108

[Peterson 2011] Larry L. Peterson. Computer networks ise. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th édition, 2011. 33, 53, 148

[Poduri & Sukhatme 2004] S. Poduri and G.S. Sukhatme. *Constrained coverage for mobile sensor networks*. In Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, volume 1, pages 165 – 171 Vol.1, april-1 may 2004. 47, 105

[Rahman & Gregory 2011a] F.M. Rahman and M.A. Gregory. 4-*N intelligent MANET routing algorithm*. In Australasian Telecommunication Networks and Applications Conference (ATNAC), 2011, pages 1 –6, nov. 2011. 60, 107, 109, 117, 129

[Rahman & Gregory 2011b] F.M. Rahman and M.A. Gregory. *Quadrant based intelligent energy controlled multicast algorithm for Mobile Ad Hoc Networks*. In Advanced Communication Technology (ICACT), 2011 13th International Conference on, pages 1298 –1303, feb. 2011. 60, 107, 128

[Raymond 1989] Kerry Raymond. *A tree-based algorithm for distributed mutual exclusion*. ACM Trans. Comput. Syst., vol. 7, no. 1, pages 61–77, January 1989. 61, 63, 64

[Ricart & Agrawala 1981] Glenn Ricart and Ashok K. Agrawala. *An optimal algorithm for mutual exclusion in computer networks*. Commun. ACM, vol. 24, no. 1, pages 9–17, January 1981. 61, 65, 140, 172

[Ricart & Agrawala 1983] Glenn Ricart and Ashok K. Agrawala. *Technical Correspondence*. Author's response to Commun. ACM, vol. 26, no. 2, pages 146–149, February 1983. 62

[Rodoplu & Meng 1999] V. Rodoplu and T.H. Meng. *Minimum energy mobile wireless networks*. Selected Areas in Communications, IEEE Journal on, vol. 17, no. 8, pages 1333 –1344, aug 1999. 76, 82

[Royer & Perkins 1999] Elizabeth M. Royer and Charles E. Perkins. *Multicast Operation of the Ad-hoc On-demand Distance Vector Routing Protocol*. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom '99, pages 207–218, New York, NY, USA, 1999. ACM. 58

[Shah & Rabaey 2002] R.C. Shah and J.M. Rabaey. *Energy aware routing for low energy ad hoc sensor networks*. In Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE, volume 1, pages 350–355 vol.1, 2002. 34

[Singh *et al.* 1998] Suresh Singh, Mike Woo and C. S. Raghavendra. *Power-aware routing in mobile ad hoc networks*. In Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, MobiCom '98, pages 181–190, New York, NY, USA, 1998. ACM. 34, 55, 70, 71

[Singhal & Manivannan 1997] M. Singhal and D. Manivannan. *A distributed mutual exclusion algorithm for mobile computing environments*. In Intelligent Information Systems, 1997. IIS '97. Proceedings, pages 557 –561, 8-10 1997. 61, 66, 134, 137, 139, 140, 141, 145, 149, 152, 155, 158, 162, 170, 172

[Singhal 1989] M. Singhal. *A heuristically-aided algorithm for mutual exclusion in distributed systems*. Computers, IEEE Transactions on, vol. 38, no. 5, pages 651–662, 1989. 63

[Sivakumar & Bharghavan 1998] B. Sivakumar R. Das and V. Bharghavan. *An Improved Spine-based Infrastructure for Routing in Ad Hoc Networks*. In Computers and Communications, 1998. ISCC '98. Proceedings. Third IEEE Symposium on, 1998. 48

[Sivakumar *et al.* 1998] Raghupathy Sivakumar, Bevan Das and Vaduvur Bharghavan. *Spine routing in ad hoc networks*. Cluster Computing, vol. 1, no. 2, pages 237–248, 1998. 45, 46, 47, 48

[Sivakumar *et al.* 1999] R. Sivakumar, P. Sinha and V. Bharghavan. *CEDAR: a core-extraction distributed ad hoc routing algorithm*. Selected Areas in Communications, IEEE Journal on, vol. 17, no. 8, pages 1454–1465, 1999. 59

[Sobeih *et al.* 2004] A. Sobeih, H. Baraka and A. Fahmy. *ReMHoc: a reliable multicast protocol for wireless mobile multihop ad hoc networks*. In Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, pages 146–151, 2004. 59

[Srinivasan *et al.* 2004] V. Srinivasan, C.-F. Chiasserini, P.S. Nuggehalli and R.R. Rao. *Optimal rate allocation for energy-efficient multipath routing in wireless ad hoc networks*. Wireless Communications, IEEE Transactions on, vol. 3, no. 3, pages 891 – 899, may 2004. 56

[Stojmenovic & Lin 2001a] I. Stojmenovic and X. Lin. *Power-aware localized routing in wireless networks*. Parallel and Distributed Systems, IEEE Transactions on, vol. 12, no. 11, pages 1122 –1133, nov 2001. 55

[Stojmenovic & Lin 2001b] I. Stojmenovic and Xu Lin. *Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks*. Parallel and Distributed Systems, IEEE Transactions on, vol. 12, no. 10, pages 1023–1032, 2001. 43

[Stojmenovic 2002] I. Stojmenovic. *Position-based routing in ad hoc networks*. Communications Magazine, IEEE, vol. 40, no. 7, pages 128–134, 2002. 38, 40, 41, 42

[Stojmenovic 2003] Ivan Stojmenovic. Handbook of wireless networks and mobile computing, volume 27. John Wiley & Sons, 2003. 17, 41, 57

[Suzuki & Kasami 1985] Ichiro Suzuki and Tadao Kasami. *A distributed mutual exclusion algorithm*. ACM Trans. Comput. Syst., vol. 3, no. 4, pages 344–349, November 1985. 61, 63

[Takagi & Kleinrock 1984] H. Takagi and L. Kleinrock. *Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals*. Communications, IEEE Transactions on, vol. 32, no. 3, pages 246–257, 1984. 42

[Tarique *et al.* 2005] M. Tarique, K.E. Tepe and M. Naserian. *Energy saving dynamic source routing for ad hoc wireless networks*. In Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2005. WIOPT 2005. Third International Symposium on, pages 305 – 310, april 2005. 56, 71

[Trellisware 2013] Trellisware. *Trellisware*. http://www.trellisware.com/apps/manet-applications/, 2013. Retrieved on 16th Jun 2013. 9, 10

[TSM 2013] Trellisware TSM. *Trellisware TSM*. http://www.trellisware.com/tactical-scalable-manet-tsm/, 2013. Retrieved on 16th Jun 2013. 9

[Vigorito *et al.* 2007] C.M. Vigorito, D. Ganesan and A.G. Barto. *Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks*. In Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on, pages 21–30, 2007. 96

[Walter & Kini 1997] Jennifer Walter and Savita Kini. *Mutual Exclusion on Multihop, Mobile Wireless Networks*. Rapport technique, College Station, TX, USA, 1997. 64

[Walter *et al.* 2001] Jennifer E. Walter, Jennifer L. Welch and Nitin H. Vaidya. *A mutual exclusion algorithm for ad hoc mobile networks*. Wirel. Netw., vol. 7, no. 6, pages 585–600, November 2001. 61, 65, 137

[Wang *et al.* 2003] S.Y. Wang, C.L. Chou, C.H. Huang, C.C. Hwang, Z.M. Yang, C.C. Chiou and C.C. Lin. *The design and implementation of the NCTUns 1.0 network simulator*. Computer Networks, vol. 42, no. 2, pages 175 – 197, 2003. 90

[Wang 1992] J. Wang. *Distributed mutual exclusion based on dynamic costs*. In Intelligent Control, 1992., Proceedings of the 1992 IEEE International Symposium on, pages 109 –115, aug 1992. 26, 136

[Wells 1991] D.G. Wells. The penguin dictionary of curious and interesting geometry. Penguin Mathematics Series. Penguin Books, 1991. 170

[Wu & Tay 1999] C.W. Wu and Y. C. Tay. *AMRIS: a multicast protocol for ad hoc wireless networks*. In Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE, volume 1, pages 25–29 vol.1, 1999. 58

[Wu *et al.* 2008] Weigang Wu, Jiannong Cao and Jin Yang. *A fault tolerant mutual exclusion algorithm for mobile ad hoc networks*. Pervasive and Mobile Computing, vol. 4, no. 1, pages 139 – 160, 2008. 61, 66, 137, 139, 140, 141, 142, 143, 144, 145, 148, 149, 151, 153, 162, 163, 165, 172

[Wu 2002] Jie Wu. *Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links*. Parallel and Distributed Systems, IEEE Transactions on, vol. 13, no. 9, pages 866–881, 2002. 48

[Xiao *et al.* 2000] Hannan Xiao, W.K.-G. Seah, A. Lo and Kee Chaing Chua. *A flexible quality of service model for mobile ad-hoc networks*. In Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st, volume 1, pages 445–449 vol.1, 2000. 55

[Xie *et al.* 2002] Jason Xie, Rajesh R. Talpade, Anthony Mcauley and Mingyan Liu. *AMRoute: Ad Hoc Multicast Routing Protocol*. Mob. Netw. Appl., vol. 7, no. 6, pages 429–439, December 2002. 58

[Ye *et al.* 2004] Wei Ye, J. Heidemann and D. Estrin. *Medium access control with coordinated adaptive sleeping for wireless sensor networks*. Networking, IEEE/ACM Transactions on, vol. 12, no. 3, pages 493 – 506, june 2004. 57

[Yeh 2001] Chi-Hsiang Yeh. *Variable-radius routing protocols for high throughput, low power, and small latency in ad hoc wireless networks*. In Proc. IEEE IntâĂŹl conf. Wireless LANs and Home Networks, pages 215–227, 2001. 43

[Yi *et al.* 2002] Yunjung Yi, Sung-Ju Lee, William Su and Mario Gerla. *On-demand multicast routing protocol (ODMRP) for ad hoc networks*. 2002. 59

[Yu & Prasanna 2003] Yang Yu and V.K. Prasanna. *Energy-balanced multi-hop packet transmission in wireless sensor networks*. In Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE, volume 1, pages 480 – 486 Vol.1, dec. 2003. 57

[Yu *et al.* 2007] Chang Wu Yu, Tung-Kuang Wu and Rei Heng Cheng. *A low overhead dynamic route repairing mechanism for mobile ad hoc networks*. Computer Communi-

cations, vol. 30, no. 5, pages 1152 – 1163, 2007. <ce:title>Advances in Computer Communications Networks</ce:title>. 37

[Zou & Chakrabarty 2003] Y. Zou and Krishnendu Chakrabarty. *Sensor deployment and target localization based on virtual forces*. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 2, pages 1293 – 1303 vol.2, march-3 april 2003. 47, 105

Publications

Conference Papers

1. Murali Parameswaran, Rakesh Kumar, Chittaranjan Hota, Antti Yla- Jaski; "Energy-Aware Routing in Mobile Ad-Hoc Networks", IFIP/IEEE Wireless Days 2008, Nov 24-27, 2008, Dubai. doi: 10.1109/WD.2008.4812884

2. Parameswaran, Murali; Hota, Chittaranjan; , "A novel permission-based reliable distributed mutual exclusion algorithm for MANETs," Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On , vol., no., pp.1-6, 6-8 Sept. 2010, Colombo. doi: 10.1109/WOCN.2010.5587356

3. P. Murali, A. Challa, M.R. Kasyap, Chittaranjan Hota, "A Generalized Energy Consumption Model for Wireless Sensor Networks," Computational Intelligence and Communication Networks, International Conference on, pp. 210-213,November 2010, Bhopal. doi: 10.1109/CICN.2010.51

4. Jain, R.; Parameswaran, M.; Hota, C.; , "An efficient on-demand routing protocol for MANETs using Bayesian approach," Communication Systems and Networks (COMSNETS), 2011 Third International Conference on , vol., no., pp.1-4, 4-8 Jan. 2011, Banglore. doi: 10.1109/COMSNETS.2011.5716496

5. Parameswaran, M.; Hota, C.; ,"Arbitration-based Reliable Distributed Mutual Exclusion for Mobile Ad-hoc Networks", Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2013 11th International Symposium and Workshops on , vol., no., pp.380-387, May 13-17, 2013, Tsukuba.
   URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6576458

6. Parameswaran, M.; Rastogi, V.; Hota, C.; , "A Virtual-Force Based Multicast Routing Algorithm for Mobile Ad-hoc Networks," Ubiquitous and Future Networks 2013 (ICUFN), 2013 The Fifth International Conference on , vol., no., pp.696-700, 2-5 Jul. 2013, Da Nang. doi: 10.1109/ICUFN.2013.6614910

Journal Papers

1. Parameswaran, Murali, and Chittaranjan Hota, "SECTOR BASED MULTICAST ROUTING ALGORITHM FOR MOBILE AD-HOC NETWORKS." International Journal of Wireless & Mobile Networks (IJWMN), vol.5 (5), pp 49–63, October, 2013. doi : 10.5121/ijwmn.2013.5504

Biographies

Brief Biography of the Candidate

Murali P is currently Lecturer in the Department of Computer Science and Informatoin Systems at Birla Institute of Technology & Science, Pilani, Rajasthan, India. He has completed Bachelors of Engineering(Computer Science & Engineering) degree in 2002 from Regional Engineering College, Rourkela(currently NIT Rourkela) and Masters of Technology degree in Computer & Information Systems in 2004. He is a nucleus member of Information & Processing Centre and carried out several computerization tasks. He was a reviewer for IEEE WoWMoM 2014. He teaches courses like Computer Programming, Data Strucures & Algorithms, Network Security, etc. His research interests are in the areas of Mobile Wireless Networks, Quality of Service, and Distributed Systems.

Brief Biography of the Supervisor

Chittaranjan Hota is currently Associate Professor of Computer Science & Engineering and the Associate Dean of Admissions at Birla Institute of Technology and Science-Pilani, Hyderabad Campus, Hyderabad. He served as the founding Head in the Dept. of Computer Science and Engineering at BITS, Hyderabad. He has a PhD in Networks and Information Security from the Dept. of Computer Science and Engineering, BITS, Pilani. Prior to this he has a Masters in Engineering (Computer Sc. & Engineering) from Thapar with First class (honors), and a Bachelors in Engineering (Computer Engineering) from Amravati (MS) with First class. He has been at BITS-Pilani since past fifteen years, and overall since past twenty-five years at Indian universities. He has been a visiting researcher and visiting professor over two months to a semester at University of New South Wales, Sydney; University of Cagliari, Italy; Aalto University, Finland; and City University, London in the past. His research has been funded by University Grants Commission, New Delhi; Department of Information Technology, New Delhi; and Tata Consultancy Services, India. He has guided three PhD students and currently guiding six PhD students in the areas of

Wireless networks, Information security, and Healthcare informatics. He is the recipient of Australian Vice Chancellors' Committee award, recipient of Erasmus Mundus fellowship from European commission, Italian ministry of education fellowship, and recipient of Certificate of Excellence from K.R. Faculty Excellence Award at BITS Pilani. He has published extensively in peer-reviewed journals and conferences. He has edited LNCS volumes. He is a member of IEEE, ACM, IE, and ISTE.