IMPLEMENTATION AND PERFORMANCE ANALYSIS OF A MULTI-MESHED TREE ROUTING PROTOCOL FOR MANETS

BY ROHIT THAREJA

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. Ivan Marsic

and approved by

New Brunswick, New Jersey

January, 2011

ABSTRACT OF THE THESIS

Implementation and Performance Analysis of a Multi-Meshed Tree Routing Protocol for MANETs

by Rohit Thareja

Thesis Director: Prof. Ivan Marsic

As the importance of mobile data connectivity in our daily lives increases, there has been an accompanying increase in the demand for rapidly deployable, short-lived networks which operate without any base infrastructure. This is where Mobile-ad-hoc Networks step in to turn the dream of getting connected anywhere and at any time into a reality and are fast being adopted into civilian applications like vehicular networks and location-based advertising. They have several advantages over infrastructure based wireless networks (like cellular networks) in terms of cost of deployment, ability to use unlicensed bands, etc. However, they are still plagued with several constraints mainly brought about by their dynamic nature. The main concern here is the use of an efficient routing protocol which must be adaptive and able to maintain routes in spite of the changing network connectivity.

In this work we investigate a recently-proposed cluster-based routing protocol which utilizes both mesh and tree concepts and aims at minimizing the control overhead required in the setup and maintenance of the network while maintaining robust connectivity. This hybrid protocol, called Multi-Meshed Tree routing protocol, has a unique way of addressing the issues of dynamic adaptation and enhanced connectivity through

a hierarchical cluster-based approach along with Virtual Identifiers. Its main advantage over the various other MANET protocols developed is its algorithm simplicity and low message complexity suited for these limited bandwidth, limited power, resource constrained networks.

The focus of this work is implementing this protocol (to work with the ns2 simulator), understanding its advantages and conducting some preliminary simulation based comparisons with prevalent Mobile ad-hoc network protocols. We looked at comparison with AODV, DSDV and DSR Protocols and observed MMT to exhibit more resilient connectivity due to redundant routes, particularly for larger sized networks. Further, there was roughly a 1/6th reduction in routing overhead when compared to proactive protocols. We also studied the effects of configurability of clustersize and inherent non-optimality of routes associated with MMT. We foresee the increased use of MANETS in our daily lives, and hope that the advantages of this routing protocol are utilized for future applications.

Acknowledgements

I would like to thank my adviser Prof. Ivan Marsic for his constant guidance and support throughout the entire duration of my MS research. I am very grateful for the freedom he gave me to work in the field of my interest while making sure that I was always on the right path. This work would not have been possible without him.

I would like to thank my Manager at Research In Motion for holding my Job while I finished my Thesis work.

I would also like to thank my parents and my family for their continuous support throughout this journey. I would like to dedicate this work to them.

Table of Contents

List of Figures

Chapter 1

Introduction

A Mobile Ad-Hoc Network (MANET) is an autonomous, self-configuring system of mobile nodes connected by wireless links and capable of communication without any static infrastructure like base stations. Nodes in MANETs are free to move and organize themselves in an arbitrary fashion. Each node has the freedom to roam about while communicating with others. The communication path between pairs of nodes may have multiple links and the radio between them can be heterogeneous. This permits an association of different links to be a part of the same network.

If two hosts are not within radio range of each other, all message communication between them must pass through one or more intermediate nodes that serve as routers. As an example, we can think of a group of people with laptops, at a location where there is no network serice present. They can easily form an ad-hoc network between their machines to facilitate data communication. This is one of the many examples where these networks may possibly be used.

Interest in such networks has recently grown due to the plethora of mobile communication devices hitting the market, like mobile phones, palmtops, laptops, tablets, etc, that can operate in license-free radio frequency bands. Technological advancements in wireless communication devices have also led to lower prices and higher data rates. Interest is also partly fueled by increased enthusiasm for running common network protocols in dynamic wireless environments without the need of specific infrastructures.

The main challenge in the design of ad hoc networks is the development of dynamic routing protocols that can efficiently calculate routes between two communicating nodes. The routing protocol must be able to keep up with the high degree of node mobility that often changes the network topology drastically and unpredictably.

Some of the inherent features of MANETS are:

1. They are rapidly deployable and self configuring.
2. They have no requirement for existing infrastructure.
3. They utilize Wireless links, usually operating in unlicensed bands.
4. Nodes are mobile and hence the topology can be very dynamic.
5. Nodes must be able to relay traffic since communicating nodes might be out of range.
6. It can be a standalone network or it can be connected to external networks(Internet).

The popularity of these networks has grown, and while typical, traditional applications of MANETs include Military operation, disaster recovery, etc these networks are also fast being adapted into civilian applications. These range from simple scenarios such as people at a conference where their laptops constitute a temporary MANET to more complicated scenarios such as highly mobile vehicles on the highway which form a Vehicular ad hoc network to provide traffic management.

Some popular applications for which these MANETS are suited are listed below:

1. Military environments: soldiers, tanks, planes
2. Emergency operations: Disaster recovery, Search-and-rescue, Policing and fire fighting
3. Civilian environments: Taxi cab network, Conference venues, Meeting rooms, Sports stadiums , Boats and small aircraft
4. Vehicular Ad Hoc Networks (VANET)
5. Sensor Networks Eg: Disposable sensors which are dropped from high altitudes and dispersed on the ground for hazardous materials detection. Underwater Sensors.

A MANET Protocol Stack ideally consists of

- Physical Layer: 2.4/5.8 Ghz, FHSS/DSSS, OFDM, OFDMA, MIMO, Directional Antenna, etc
- MAC Layer: CSMA, CSMA/CA, RTS/CTS, TDMA with Scheduling Algorithm
- Routing Layer: Addressing; DSR, AODV, OLSR, TORA, ZRP, LAR, etc. (Unicast, Broadcast, Multicast, Reliable Multicast, Geocast)
- Transport Layer: UDP, TCP, RTP, etc.
- Cross-cutting functions:

  Power-awareness, energy-conservation

  Security of control and data packets

There has been a lot of research work on all these Layers of the Protocol Stack, as well as several cross-layer solutions. The emphasis of this work is on developing a novel protocol for the Routing Layer. Such networks allow the wireless mobile nodes to dynamically enter the network as well as leave the network. Due to the inherent limits in the transmission range of wireless network nodes, multiple hops are usually needed for a node to exchange information with any other node in the network. Nodes in these networks act as hosts as well as routers, and hence the most important part of these networks is having an efficient Routing Protocol capable of supporting these mobile, multihop networks. The broad requirements of these protocols include:

1. Self starting and self organizing
2. Multi-hop, loop-free paths
3. Dynamic topology maintenance
4. Rapid convergence
5. Minimal network traffic overhead
6. Efficient Address allocation

7. Scalable to large networks

There are numerous issues to consider when deploying MANETs. The following are some of the main issues.

1. Unpredictability of environment: Ad hoc networks may be deployed in unknown terrains, hazardous conditions, and even hostile environments where tampering or the actual destruction of a node may be imminent. Depending on the environment, node failures may occur frequently.

2. Unreliability of wireless medium: Communication through the wireless medium is unreliable and subject to errors. Also, due to varying environmental conditions such as high levels of electro-magnetic interference (EMI) or inclement weather, the quality of the wireless link may be unpredictable. Furthermore, in some applications, nodes may be resource-constrained and thus would not be able to support transport protocols necessary to ensure reliable communication on a lossy link. Thus, link quality may fluctuate in a MANET.

3. Resource-constrained nodes: Nodes in a MANET are typically batterypowered as well as limited in storage and processing capabilities compared to fixed nodes of Infrastructure based networks. Moreover, they may be situated in areas where it is not possible to re-charge and thus have limited lifetimes. Because of these limitations, they must have algorithms which are energy-efficient as well as operating with limited processing and memory resources. The available bandwidth of the wireless medium may also be limited because nodes may not be able to sacrifice the energy consumed by operating at full link speed.

4. Dynamic topology: The topology in an ad hoc network may change constantly due to the mobility of nodes. As nodes move in and out of range of each other, some links break while new links between nodes are created.

As a result of these issues, MANETs are prone to numerous types of faults including,

1. Transmission errors: The unreliability of the wireless medium and the unpredictability of the environment may lead to transmitted packets being garbled and

thus received in error.

2. Node failures: Nodes may fail at any time due to different types of hazardous conditions in the environment. They may also drop out of the network when their energy supply is depleted.

3. Link failures: Node failures as well as changing environmental conditions (e.g., increased levels of EMI) may cause links between nodes to break.

4. Route breakages: When the network topology changes due to node/link failures and/or node/link additions to the network, routes become out of date and thus incorrect. Depending upon the network transport protocol, packets forwarded through stale routes may either eventually be dropped or be delayed; packets may take a circuitous route before eventually arriving to the destination node.

5. Congested nodes or links: Due to the topology of the network and the nature of the routing protocol, certain nodes or links may become overutilized, i.e., congested. This will lead to either larger delays or packet loss.

Routing protocols for MANETs must deal with these issues to be effective. There are a large number of Routing Protocols that have been proposed, each having their own pros and cons, some of which we will discuss in the next chapter. The Multi Meshed Tree Routing Protocol,which is the focus of this work, addresses the current major issues facing MANET Routing protocols, which include Robust Connectivity, Scalalbility and Dynamic Adaptation to node mobility. These concerns and their proposed solutions are discussed in more details in the following chapters.

Now that we have an understanding of the important applications of MANETS, as well as their characteristics and requirements, the next chapter delves further into the Routing Protocols currently in use, as well as their broad classifications. It also explains the functioning of the AODV, DSDV and DSR protocols which we use as benchmarks for performance comparison in Chapter4.

Chapter 3 then explains the detailed working of a novel Multi-meshed Tree Routing protocol, and discusses its features and advantages.

Chapter 4 explains the Simulation framework setup to test the implemented MMT Protocol and provides a comparative analysis of the Protocol with AODV, DSDV and DSR protocols.

It is the goal of this work to explain the detailed operation, associated implementation and algorithms of the Multi-Meshed Tree Routing Protocol. To discuss what advantages it has over prevelant MANET routing protocols and to conduct simulation based experiments for performance comparisons in key areas such as Robustness, Overhead and Delay.

Chapter 2

Survey of MANET Routing Protocols

The aim of this section is to describe a classification of the different types of Mobile Ad-hoc Network Routing Protocols with a few examples.

Routing protocols in conventional wired networks are usually based upon either distance vector or link state routing algorithms. To keep up with topology changes, both of these algorithms require periodic routing advertisements to be broadcast by each router. In distance vector routing, each router broadcasts to all of its neighboring routers its view of the distance to all other nodes; the neighboring routers then compute the shortest path to each node. In link-state routing, each router broadcasts to its neighboring nodes its view of the status of each of its adjacent links; the neighboring routers then compute the shortest distance to each node based upon the complete knowledge of the network. These conventional routing algorithms are not efficient for the type of dynamic changes which may occur in an ad-hoc network. This is because, in conventional networks, routers do not generally move around and only rarely leave or join the network. In an environment with mobile nodes, the changing topology will not only trigger frequent re-computation of routes but the overall convergence to stable routes may be infeasible due to the high-level of mobility. Routing in MANETs must take into consideration their important characteristics such as node mobility, scalability as well as resource constraints.

In the remainder of this section, we present an overview of some of the key routing protocols for MANETs.

2.1 Classification of MANET Routing Protocols

There are a large variety of Routing Protocols, which implement vastly different Routing strategies. It is important for the routing and topology information to be kept up to date in such a dynamic environment. The means by which the information is updated is a major characteristic for classification. Figure 2.1 shows the general broad classification of Routing Protocols. Routing information can be updated:

- Proactively
- On-Demand
- Hybrid: Both proactive and On-demand

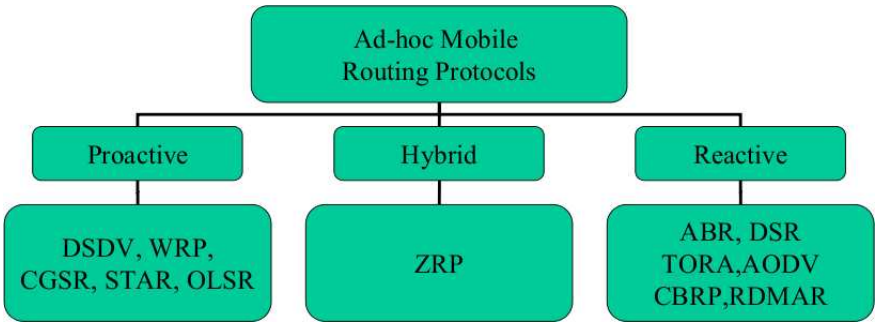

Figure 2.1: Categorization of ad hoc routing protocols

A routing protocol can maintain routing information either on-demand (only when required) or proactively (at all times). Further proactive protocols can be divided into protocols that update routing information in regular intervals and protocols that update on certain events. Finally, there are routing protocols that are hybrid and make use of both methods. These different types of protcols are explained in detail in the coming sections.

2.1.1 On-Demand or Reactive Protocols

A network using an on-demand protocol will not maintain routing information on all nodes for all times, but instead will be obtained on demand. If a node wants to transmit a message, and does not have enough routing information to send the message to the

destination, the required information has to be obtained. For Reactive Protocols, this routing information is not consistently maintained, but instead gathered when data transfer between nodes is required. The node needs to know, at least the next hop (among its neighbors) for the packet. Although the node could just broadcast(flood) the packet to all neighbors this leads to serious congestion in many cases, and usually adversely affects throughput. However, such broadcasts can be used in a route discovery process, if there is no next-hop information available, yet. This usually is made up of a broadcast message from the source node, indicating the desired destination. Nodes which have the desired routing information will respond to the originating node, which will decide a route from the replies it receives. The broadcast may be limited to propagate to only a few hops first, before a net-wide broadcast is issued (which would flood the whole network). The route request and selection process must complete, before the message can be sent. This leads to initial setup delays for messages, if their route is not known to the node. This is one of the drawbacks of Reactive Protocols as their delay in data propagation is generally higher, attributed to the initial route discovery process. To limit the impact of this delay, most protocols will use a route cache for routes which are already established. The information in this cache will time out, because in a mobile environment, the routes will be invalid after some time, due to node mobility and topology changes. This is done to avoid stale route cache data. Clearly, applications that are used over an on-demand routing protocol need to be tolerant to the initial setup delay. The advantage of on-demand routing protocols is that the wireless channel is not required to carry a lot of routing overhead data for routes,a large number of which are generally not even used. In high traffic, high mobility scenarios, having low route-setup overhead can be a great advantage in increasing performance such as Throughput or Latency.

Examples for on-demand protocols are the following:

ABR :Associativity Based Routing [5], AODV: Ad Hoc On Demand Distance Vector Routing Protocol [10] , CEDAR: Core-Extraction Distributed Ad Hoc Routing [4] , DREAM: Distance Routing Effect Algorithm for Mobility [6] , DSR: Dynamic Source Routing Protocol[19].

2.1.2 Proactive Protocols

Proactive routing protocols will try to maintain correct routing information on all nodes in the network at all times. This can be achieved in different ways, and thus divides the protocols into two subclasses: event driven and periodically updated protocols.

Event driven protocols will not send any routing update packets, if no change in topology occurs. Only if a node detects a change of the topology (usually a change in the neighbor set, or the reception of a message indicating a change in some other nodes neighbor set, or a link/node failure), this will be reported to other nodes, according to the updating strategy of the routing protocol. Protocols that are updated in regular intervals will always send their topology information to other nodes at periodic intervals, irrespective of any change in topology. Many link state protocols work in such a manner. They may vary the maximum distance of an update message with the length of the interval, so that nodes farther away get updates less frequently than close nodes, thus balancing the load imposed on the network. This is based on the assumption that there is higher traffic on closer nodes, and there is no need to flood the entire network everytime the topology changes. Proactive protocols of either subclass impose a fixed overhead to maintain the routing tables, even if many of the entries are not used at all. Their primary advantage is, that the routes can be used at once and there is no setup delay. Proactive protocols tend to perform best in networks with low to moderate mobility, fewer nodes, and many data sessions.

Event driven proactive routing protocols are the following: CBRP: Cluster Based Routing Protocol [12], CGSR:Clusterhead Gateway Switch Routing [7] , DSDV: Destination Sequenced Distance Vector Routing Protocol [18]

Regular updated protocols are: ,FSR: Fisheye State Routing [14], OLSR:Optimized Link State Routing [13].

2.1.3 Hybrid Protocols

These are the protocols that utilize both proactive and on-demand routing and attempt to leverage the advantages of both. Routes are maintained proactively, but only to

certain nodes (active receivers), and the size and frequency of the updates is controlled, usually by dividing the network into zones, such as in ZRP[15]. For nodes not served by the proactive protocol, there is reactive route discovery.

Some examples are: HWMP (Hybrid Wireless Mesh Protocol): which is the default mandatory routing protocol for 802.11s. HWMP is inspired by a combination of AODV (RFC 3561[2]) and tree-based proactive routing. [16] Terminode Routing:Terminode Routing consists of an on-demand location based component: AGPF (Anchored Path Geodesic Packet Forwarding) and a proactive local routing component (Terminode Local Routing, TLR).[17] ZRP - Zone Routing Protocol The Zone Routing Protocol also consists of a proactive Intra Zone Routing Protocol (IARP) and an on-demand Inter Zone Routing Protocol (IERP)[15].

2.2 Benchmark Routing Protocols

In this section, we explain the working of 3 common MANET protocols, which we have used as benchmarks for Performance comparison with MMT in Chapter 4. These are: Ad Hoc On Demand Distance Vector Routing Protocol(AODV), Destination Sequenced Distance Vector Routing Protocol (DSDV) and Dynamic Source Routing(DSR)

2.2.1 Ad Hoc On Demand Distance Vector Routing Protocol

This is one of the most discussed and most advanced routing protocols. It is an important part of the work of the MANET IETF working group and is probably the most mature suggestion for an ad hoc routing protocol. Its main developers are Charles E. Perkins (Nokia Research) Elizabeth Belding-Royer (UCSB) and Samir Das (University of Cincinnati)[10]. AODV is discussed in lots of studies and is often used as a reference to compare other routing protocols such as in [8],[9] It is for this reason that we have chosen to compare the developed MMT protocol with AODV.

AODV does not need to exchange periodic messages proactively, but works in an on-demand fashion, instead. If a route to a destination is unknown, a route discovery process is initiated. Figure 2.2 shows the Route Discovery process in AODV. This

consists of broadcasting a Route Request (RREQ) packet throughout the network. To limit the impact of a net-wide broadcast, these request should be sent with an expanding ring search technique: the TTL of the packets starts with a small value; if no route has been found, the TTL will be increased and the request will be resent. Each node that rebroadcasts this request, adds its address into a list in the packet. If the destination sees the request, it will reply with a unicast Route Reply (RREP) to the source. Each intermediate node may cache the learned routes. The routing table entries consist of a destination, the next hop toward this destination and a sequence number. Routes are only updated if the sequence number of the updating message is larger than the existing one. Thus routing loops and updates with stale information are prevented. The amount of information, which needs to be present at each node, is rather limited: The node is aware of its neighbors (via link-layer-notification, or explicit HELLO messages). The node knows route destinations and the next hop. The node has a precursor list for each destination. This list consists of all nodes, which use the current node as a relay for the destination in the routing table entry. In case of a route failure to the destination, the node knows exactly which other nodes to notify, and does so proactively. Each routing entry also has a lifetime. The authoritative description of AODV is RFC 3561 in [10]

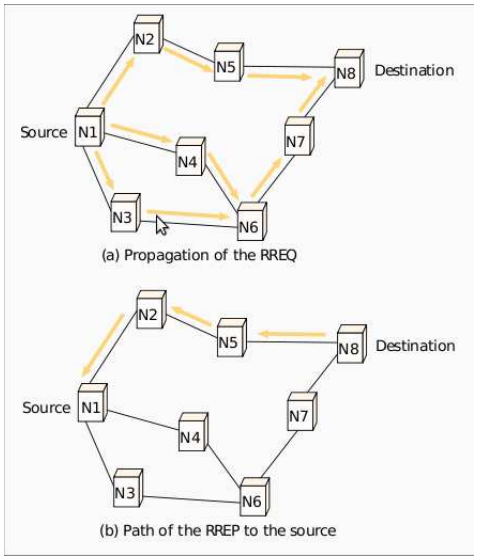

Figure 2.2: AODV Route Discovery
[22]

Some of the major disdavantages of AODV are Connection setup delay due to reactive nature. This is the main disadvantage as it leads to high route discovery latency. Also multiple RouteReply packets in response to a single RouteRequest packet can lead to heavy control overhead, when compared to other reactive protocols like DSR. AODV is designed to support the shortest hop count metric. This metric favors long, low- bandwidth links over short, high-bandwidth links [11]

2.2.2 Destination Sequenced Distance Vector Routing Protocol

This protocol is the result to adapt an existing distance vector routing algorithm (Distributed Bellman Ford), as used in RIP, to an ad hoc networking environment. This is a proactive protocol, that updates routing information on a regular basis.

DSDV is one of the first attempts to incorporate an established routing mechanism to work with mobile ad hoc networks. To avoid routing loops, the concept of destination sequence numbers has been introduced. Each entry in the routing table contains a sequence number which are generally even if a link is present, else odd. Every mobile station maintains a routing table that lists all available destinations, the number of hops to reach the destination and the sequence number assigned by the destination node. The number is generated by the destination and sends out the next update with this number. The sequence number is used to distinguish stale routes from new ones and thus avoid the formation of loops. Routes with more recent sequence numbers obsolete older routes. The stations periodically transmit their routing tables to their immediate neighbors. A station also transmits its routing table if there has been a significant change in its table from the last update sent. Hence, the update is both time-driven and event-driven. The routing table updates can be sent in two ways:- a "full dump" or an incremental update. A full dump sends the entire routing table to the neighbors and could occupy many packets whereas in an incremental update only those entries from the routing table are sent which have a metric change since the last update and it must fit in a packet. If there is space in the incremental update packet then those entries may be included whose sequence number has changed since the last update(This is kept track of internally). This is a method of providing relevant

updated routing data in the packet when there is space leftover. When the network is relatively stable, incremental updates are sent to avoid extra traffic and full dumps are infrequent, since there would have been little change and the data transmitted would be redundant. However,in a fast-changing network, incremental packets can grow big, and since the incremental updates must be limited to one packet, so full dumps will be more frequent. As mentioned previously, each route update packet, in addition to the routing table information, also contains a unique sequence number assigned by the transmitter. The route labeled with the highest (i.e. most recent) sequence number is used. This mechanism is used to provide freedom from loops and prevents stale routes. If two routes have the same sequence number, then the route with the best metric (i.e. shortest route) is used. Based on the past history, the stations estimate the settling time of routes. It is calculated by maintaining a running weighted average over the most recent updates of the routes for each destination [18]. A route settling time table is kept in each node with a time to wait for a route with a better metric before advertising. This is used to dampen fluctuations caused by hosts with irregular updates, varying propagation speeds, etc. The stations delay the transmission of a routing update by settling time so as to eliminate those updates that would occur if a better route were found very soon.

The routing information is transmitted every time a change in the topology has been detected (i.e. a change in the set of neighbors of a node). DSDV works only with bidirectional links. Mobiles also keep track of the settling time of routes, or the weighted average time that routes to a destination will fluctuate before the route with the best metric is received. By delaying the broadcast of a routing update by the length of the settling time, mobiles can reduce network traffic and optimize routes by eliminating those broadcasts that would occur if a better route was discovered in the very near future.

Some Disadvantages of DSDV:

DSDV requires a regular update of its routing tables, which uses up battery power and a small amount of bandwidth even when the network is idle. Network throughput is affected by this high overhead. Also, whenever the topology of the network changes,

a new sequence number is necessary before the network re-converges. Thus, DSDV is not suitable for highly dynamic networks.

DSDV was presented in [18] in 1994

2.2.3 Dynamic Source Routing

The Dynamic Source Routing (DSR) protocol as presented in [19] and [22] is an on-demand routing protocol that is based on the concept of source routing. Mobile nodes are required to maintain route caches that contain the source routes of which the mobile is aware. Entries in the route cache are updated as new routes are learned. There are two major phases in this protocol: route discovery and route maintenance. When a mobile node needs to send a packet to a particular node, it first checks its route cache to determine if it already has a route to the destination. If there is an unexpired route to the destination, it will use this route to send the packet. However, if the node does not have such a route, it initiates the process of route discovery by broadcasting a route request packet(RREQ). This route request is made up of the address of the destination, along with the source nodes address and a unique identification number. Each node which recieves the packet checks whether it is aware of any route to the destination. If it is not, it adds its own address to the route record of the packet and then forwards the packet along its outgoing links. To limit the flooding of route requests propagated on the outgoing links of a node, the node will onl forward the route request if the request has not yet been seen by the mobile and if the mobiles address does not already appear in the route record. A route reply (RREP) is generated when the route request reaches either the destination itself, or an intermediate node which contains in its route cache an unexpired route to the destination. By the time the packet reaches either the destination or such an intermediate node, it contains a route record yielding the sequence of hops taken. Figure 2.3 illustrates the formation of the route record as the route request travels through the network.

If the destination generates the route reply, it places the route record present in the route request into the route reply. If the route reply is generated by an intermediate node, it will append its cached route to the route record and then generate the route

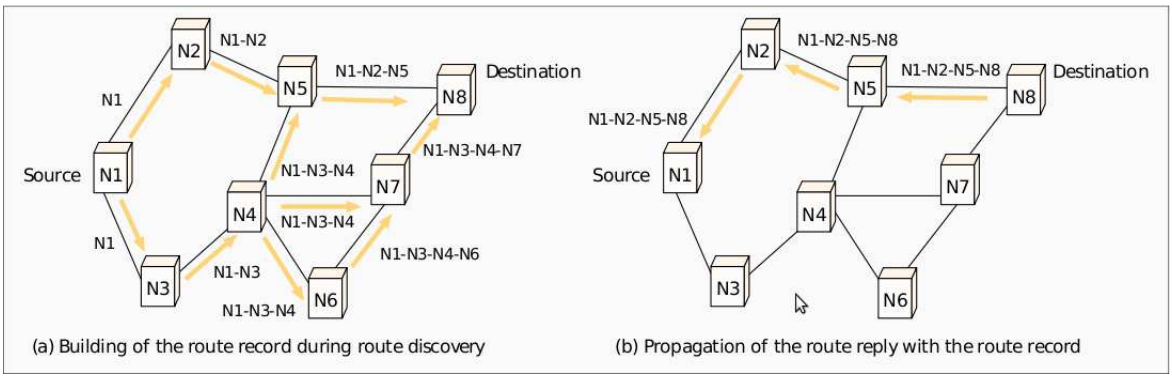

Figure 2.3: Creation of the route record in DSR
[22]

reply. To return the route reply, the responding node should have a route to the source. If it has a route to the source in its route cache, it may use that route. Otherwise, if symmetric links are supported, the node may reverse the route in the route record. However, if symmetric links are not supported, the node may initiate its own route discovery and piggyback the route reply on the new route request. 2.3b shows the transmission of the route reply with its associated route record back to the source node. Route maintenance is accomplished through the use of route error packets and acknowledgments. Route error packets are generated at a node when the data link layer encounters a fatal transmission problem. When a route error packet is received, the hop in error is removed from the nodes route cache and all routes containing the hop are truncated at that point. In addition to route error messages, acknowledgments are used to verify the correct operation of the route links. Such acknowledgments include passive acknowledgments, where a mobile is able to hear the next hop forwarding the packet along the route.

Both AODV and DSR are reactive protocols using RREQ and RREP messages. The main difference between them is that in DSR, a source routing option is used; i.e. when a node wants to send something to a destination it sets the whole route for that packet, indicating the addresses of the terminals it has to pass through. In this sense all packets have a DSR header included, and it is required that all nodes within the ad hoc network know the whole network topology.

On the other hand, AODV does not perform source routing at all. When a terminal

wants to send something to a destination, it checks its routing table, looking for the next hop towards that destination, and sends the packet to it, and so on. In this sense, data packets travel through the ad hoc network without any AODV specific information. Also, DSR uses routing cache aggressively, and maintains multiple routes per destination, while AODV uses one route per destination. For application-oriented metris like delay and throughput, DSR outperforms AODV in less stressful situations (smaller number of nodes and lower load and/or mobility). AODV outperforms DSR in more stress situations (more load, higher mobility)[9]

2.3 Need for a new Protocol

So far, we have discussed the general classification of MANET Routing Protocols, seen some examples and then explained three popular Protocols, AODV, DSDV and DSR in detail. Although some of these protocols have become well defined and even have commercial applications, there is still a lot of scope for improvement. The development of MANET routing protocols is still a subject of constant research interest, and rightly so, given that there are several inadequacies in current protocols. More importantly there are several application-specific requirements of MANET's, each differing in their Network Performance requirements ranging from Qualtiy of Service to Throughput to Latency requirements. Some of the inadequacies of Proactive protocols, like DSDV are that they have large Routing Overheads and do not deal well with route maintenance when faced with high mobility. Reactive Protocols like DSR and AODV also have several inadequacies particularly their large setup delays. There has been work done on Hybrid protocols [43] but the complexity of the algorithms has generally been a deterring factor. We find the Multi-Meshed Tree Routing protocol to have some unique features which attempt to bridge this gap of inadequacies of current Protocols. The advanatges of the MMT protocol are stated clearly in the following chapter, and it is our goal here to confirm the need for a simple yet robust algorithm capable of delivering high throughput at a relatively low control overhead. MMT steps in to attempt to bridge this requirement by providing multiple proactive routes, which is uncommon in current routing protocols, as well as avoiding expensive routing overhead to maintain

connectivity in a dynamic environment.

Chapter 3

The Multi-Meshed Tree Routing Protocol

In this section, we describe the operation of the MMT Routing Protocol, the details of the algorithm used, and some principal advantages over the previously described Ad-Hoc Routing Protocols.

3.1 The MMT Framework

The Multi-Meshed Tree routing Protocol, as described in [1], is a recently proposed, unique MANET routing protocol. It has several advantages as explained below and is the main focus of this work. The protocol deals with solving the challenges of robust connectivity and increased configurability in data delivery, while addressing scalability and adaptability to highly dynamic network conditions (due to node mobility and the wireless environment).

MMT is a single algorithm that facilitates the formation of multiple clusters of definable size and max hops from an elected cluster head and simultaneously sets up proactive routes within the clusters without flooding, or routing tables and states maintenance. For inter cluster routing, MMT adapts the reactive route discovery process used in most current reactive routing protocols but based on the meshed tree principles which avoids flooding discovery messages and reduces the route dependency between distant communicating nodes to the number of clusters between them, which is very much less than the number of actual mobile nodes that forward packets between the communicating nodes.

The hybrid approach hence consists of the proactive MMT Algorithm, used for intra cluster routing while a reactive MMT (RMMT) is used for inter cluster routing. The route discovery and route recording scheme using route request and route response

messages but has low flooding overheads and exhibits high route stability under high node mobility conditions.

The algorithm is developed as a Hybrid routing scheme to facilitate high route robustness with an efficient forwarding approach based on virtual IDs (VID). This novel approach leverages the combined features of a tree(nodes are able to connect to multiple tree branches) and a mesh (the tree branches are overlapped for enhanced redundancy) to facilitate efficient routing.

The MMT based framework has inherent capabilities that provision for:

1. Dynamic adaptation to node mobility and unreliable wireless environment as nodes connect to multiple tree branches and to better ones as they move. This results in the availability of multiple non-stale routes and localized repairing of routes. It facilitates quick joining of mobile nodes to a cluster. In contrast, proactive Protocols like DSDV dont cater for multiple routes, hence their performance is greatly affected by Link/Node failures and high mobility.

2. Robust connectivity among devices that wish to communicate and the reduction in dependency on the number of forwarding nodes between distant communicating nodes. This is achieved through route redundancy and non-fixed routes. Having non-fixed routes is an important feature missing in the protocols previously discussed as it provides more flexibility in the choice of routes to avoid regions affected by link failures.

3. Scalability to increase network size and number of communicating nodes. This is because of configurable cluster size, hybrid routing approach (proactive within a cluster and reactive across clusters) and low overhead due to no flooding. Proactive protocols like DSDV are greatly stunted by their inability to scale, as well as their large routing overhead. For AODV and DSR, particularly DSR, scaling leads to exponential delay in setting up routes.

4. Configurable operational aspects to specific application requirements.This is attributed to clustered approach and route selection and traffic prioritization capability. Configurability is a feature lacking in protocols discussed previously to

allow apllication specific flexibility.

5. Simple and robust mechanisms to self organize and self heal.

3.2 Routing Algorithm

The fundamentals of the MMT algorithm build on the knowledge that hierarchy helps address scalability in large wireless ad hoc networks and hybrid routing with proactive and reactive routing components helps in reducing routing overheads.

The functionality of the algorithm can be split up into 3 main components:

1. Cluster Formation: This involves the mobile nodes organizing themselves into clusters with an elected clusterhead. All data traffic must pass through the clusterhead, even if the source and destination nodes are within the same cluster.

2. Intra-Cluster Proactive Routing: Algorithm for routing packets within a cluster.It is a proactive algorithm which constantly checks and updates its state.

3. Inter-Cluster Reactive Routing: Algorithm for routing packets across clusters. It is a reactive algorithm which kicks in only when a clusterhead acknowledges that the destination is not within its cluster.

3.2.1 Cluster Head Election

The first aspect of the MMT algorithm is multi-hop multiple overlapped clusters formation, which involves 1) Cluster head election and 2) Cluster formation. Cluster Head Election involves determining a suitable cluster head in a locality using criteria like node IDs, credentials, power level, MAC address and the number of neighbors (i.e. the most connected and central node) or combinations thereof. We shall be adopting one such algorithm as explained in [44] that is based on the number of neighbors and use IDs to resolve a tie. This election process is used only when all nodes are deployed at the same time or when there is need for resolution. Once elected, a cluster head continues for a predefined period or till it is disabled or dies. The cluster-head election algorithm is currently an ongoing work, and will be later integrated into the routing

logic. For the purpose of this work, we assume that we have efficiently elected Cluster heads.

Once a Cluster Head (CH) is elected, the MMT algorithm is used to create the cluster around it. The CH hence, is the root of the meshed tree and multihop client feature is inherent in the proactive route set up.

3.2.2 Initial Cluster formation

Cluster Formation is the process by which nodes decide to join a cluster head for reasons such as better signal and so on. The MMT cluster formation algorithm is different as it is the cluster head that decides which of the requesting cluster clients will be in its cluster, based on a defined cluster size and a threshold on the number of hops from the cluster head. The multi hop cluster formation is achieved in a simple way and built in a localized fashion without flooding messages in the cluster. This is possible through the use of Virtual Identifiers (VID). Each node in the network has:

1. A unique Identifier (UID): This is a single identifier unique to each node, and has a one-to-one relationship (similar to a MAC address) to identify a praticular node at any moment of time or network state.

2. Virtual Identifiers (VID's): Each node can have one or multiple (upto a configurable limit) number of VID's. these are assigned by the clusterhead, as explained below. They are non-unique and change for a particular node depending on which cluster(s) it is a part of at a snapshot of time.

In figure 3.1, for sake of clarity, only a few nodes are shown. Node B can hear Node A, and Node C can hear Node B. The figure 3.1(a) shows that node A with a unique ID (UID) = 100 is elected as a cluster head. The double circle indicates A as a clusterhead in this and other figures in the text. A advertises its UID as a cluster head VID. Node B hears this advertisement and sends a request to A to join its cluster. A will allocate a VID =1001 to B. A parent node is allowed to limit the maximum number of children in its cluster, to reduce traffic bottleneck at the parent node. Whenever node A accepts a

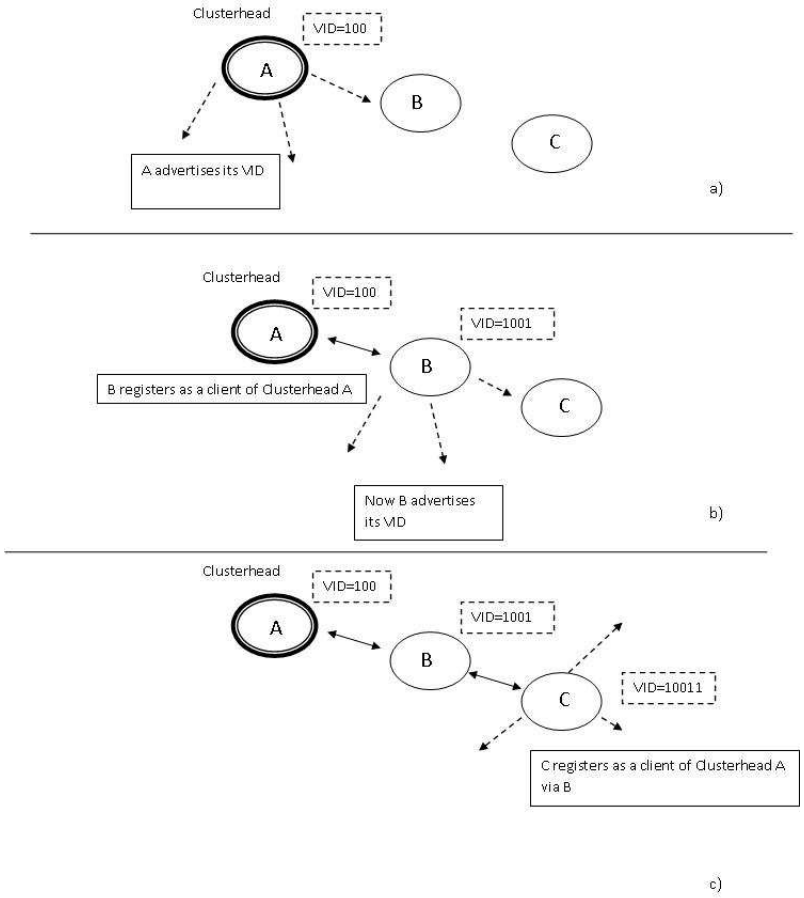

Figure 3.1: Initial Link Assignment and VID Establishment

child, it will allocate the child a VID that is its own VID, appended with a single digit integer . If another nodes wants to join as a first hop child of cluster head A it will be given a VID 1002, the following one will be given a VID = 1003 and so on.

In figure 3.1(b), B has acquired its VID from A and now advertises this VID. Node C hears B's advertisement and sends a join request to B. B follows a similar procedure as A in allocating a VID to C and C gets VID = 10011. As part of the protocol, C must register with A the cluster head. In the registration request C provides its UID and its newly acquired VID. The path taken by the registration request will be C to B to A, so that the parent is aware of the registration. Acceptance into cluster is completed by cluster head A sending registration accept.

Figure 3.1(c) shows that the link has been established between A-B and B-C. C can now advertise its VID to its own neighbors. The cluster head advertises the maximum

clustersize, as explained below, in the advertisement messages, so that when the time comes its clients will not accept any more new children into the cluster. This is just as courtesy to reduce extra overhead, since anyway nodes must register with the clusterhead, and if the cluster has reached it configurable max size, the clusterhead will stop registering new nodes. NOTE: The VIDs directly provide the number of hops from the cluster head. Limiting the max VID size defines the max hops from the cluster head. This is achieved by simply controlling the maximum number of allowed digits in the assigned VID. This max size is what is sent in the advertisement message, so that a node can check whether it has reached the max size and make the decision of whether to broadcast its presence or not. Now, it is important to keep in mind that this is completely VID dependent. A node can have multiple VID's and if even one of them has less digits than the advertised max size, then it may broadcast that particular VID, for other nodes to link to. The cluster head ID is inherent in all VIDs used by the cluster clients and hence special cluster head advertisements indicating the cluster ID are not required. The use of VIDs hence simplifies the process of multi hop cluster creation to simple arithmetic processing on the nodes and low message overhead both for cluster formation and maintenance. In [3], which is based on tethered MANET's(where the clusterheads are gateways to the internet), but uses a similar clustering approach, it was observed that the joining time for new mobiles was as low as 28 milliseconds for mobiles 7 hops away, and dropped down for mobiles closer to the clusterhead. Clustering addresses scalability of the ad-hoc network, as it caters for the organization of an increasing number of nodes. Within the cluster the MMT proactive routing scheme will be used. For inter cluster routing, a reactive scheme which extends MMT for route discovery and inter cluster routing/forwarding is used.

3.2.3 Intra-Cluster Proactive Routing

Using Figure 3.2, the multiple proactive route establishment process is explained as the clusters are formed. The one hop children of node A for example B gets a VID = 1003, C gets a VID = 1001, D gets VID = 1002. The second hop children are K, G, H and J, which have respectively VIDs 10012, 10031, 10043 and 10053, which have been derived

from their parents namely C, B, F and E respectively. Note that the VIDs carry the route information from the cluster clients to the cluster heads. As we can see in Figure 3.2, certain nodes like K, B, G, etc have multiple VIDs. This is where the concept of Secondary VID's comes in. Nodes which are assigned more than one VID classify their VID's into one Primary VID (Which has the least digits, and hence the shortest hops to reach the clusterhead), and the remaining as Secondary VID's. The secondary VIDs were acquired by these nodes by overhearing the advertisements from their neighbors and joining as their children. The multiple VIDs thus result in multiple routes (also known as multiple branches). The dynamic multiple proactive routes establishment provides robust connectivity with low overhead.

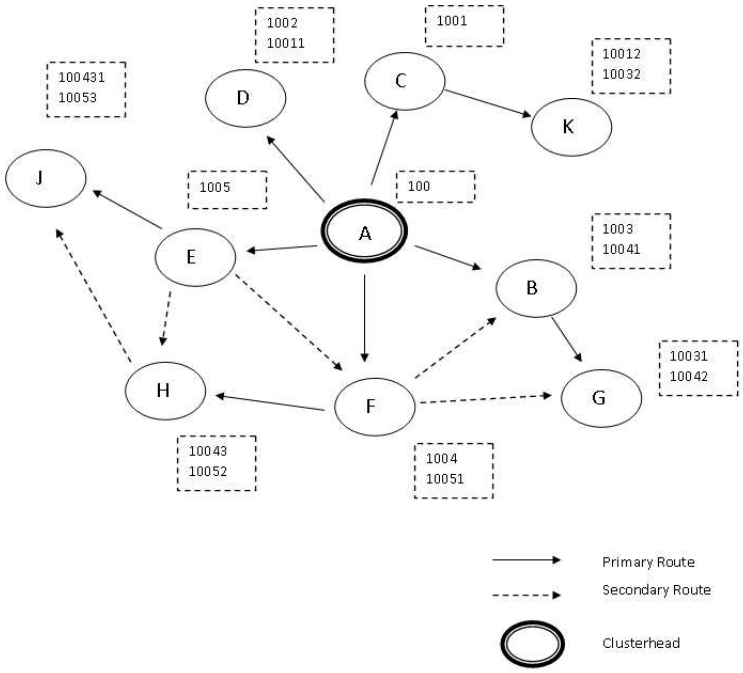

Figure 3.2: Sample topology with VID's assigned

NOTE: Due to mobility, if a node looses one VID, it can fallback on its Secondary VID's for other backup routes. For example in Figure 3.2, assume J moves away from E and towards H. Although it may lose connectivity with E, it still has connectivity to the cluster via H. Nodes continually overhear activity by their neighbors and acquire new and better VIDs (least hops, with least number of digits in the VID), thus eliminating the possibility for stale routes. The process has been very much simplified and is an

integral part of the cluster formation. Setting aside a single digit i.e. 4 bits for every hop and restricting the maximum hops to 4, the max VID size is limited to length of UID + 16 bits, which is a very low overhead. No routing tables or states are required at cluster clients as route information is carried in the VIDs. Though a node has multiple routes to the cluster head, one VID is associated with the node as a primary VID and the other VIDs used as fallback or secondary VIDs. This does not preclude the use of the multiple routes for multiple concurrent paths for forwarding. A mobile can have M maximum number of routes and hence M VIDs. (The value of M here is a configurable parameter, and can be changed in accordance with the topology or application). A mobile uses one of the nodeIDs as the primary - that with the least number of hops to the clusterhead, the rest are stored in order of preference. These routes are continually updated. The VIDs indicate a branch from the cluster head in the cluster. The multiple VIDs help in meshing the tree branches. A mesh and tree topology have thus been successfully unified to leverage the advantages of both. The topology will consist of individual tree branches from the clusterheads which are inter-meshed leading to a larger number of possible routes from node to clusterhead, and thus improved redundancy. No complex computations are required to avoid loops in the mesh; a node simply checks its VIDs and compares the integers after the cluster head VID to determine if a loop will be formed if it were to request for a particular VID. For example in 3.2 once G has acquired VID 10042 it may broadcast this VID which will be recieved by F. Before requesting for a VID with a link 10042x, F will do a check to see if a loop is being formed. Since the digits after the clusterhead match with its own VID, it will not request G for a VID. This simple arithmetic check avoids routing loops in the topology.

OVERLAPPING CLUSTERS OF MMT: To further improve route robustness in the scheme, the clusters are allowed to overlap i.e. cluster clients can be members under different clusters. In figure 3.3, there are two clusters one with A as the cluster head and the other with L as the cluster head. The client VIDs under A start with 100, whereas the clients VIDs under L start with 200, the cluster head portion of the VID is shown underlined. In the figure, nodes B, K and G with the double circle are

members of both clusters as they have VIDs that start with 100 and 200. If the mobility of these nodes causes them to move towards cluster head L, they may lose their VIDs with A but will be still connected to cluster L and vice versa. Other nodes that move in between the two clusters will provide the inter cluster connectivity. The branches of a tree mesh to provide multiple routes or route redundancy to a mobile. The overall connectivity in the network is thus enhanced, where the branches of the meshed trees (under different clusters) further mesh across the trees (clusters) leading to the multi meshed tree concept.

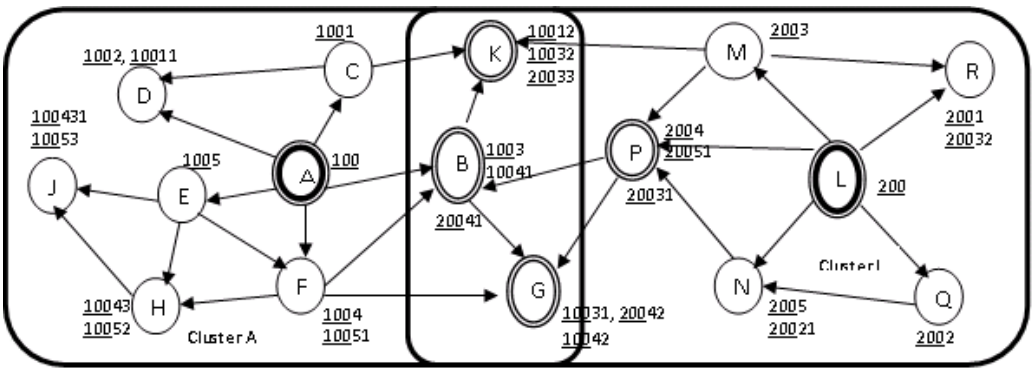

Figure 3.3: Overlapping Clusters

NOTE: Overlapping clusters have been investigated earlier, but the ease with which it is achieved in this algorithm is worth noting. Nodes that belong to more than one cluster register their multiple VIDs and UID with all involved cluster heads. What this is means is that as soon as a node gets a VID (even from a different cluster) it must register that VID with all the involved clusterhead(s). For example in Figure 3.3, when B gets VID 20041, it registers that VID with clusterhead A as well.

This leads to some very useful knowledge since the cluster heads are now aware of the neighboring cluster heads and their VIDs. The border node knowledge is available at the cluster heads and can be used when a cluster wants to communicate with nodes in other clusters. When a clusterhead needs to find border nodes, it does a simple check(of the first 3 digits) through its registered VID's for nodes also registered on a different cluster. These are flagged as border nodes. This is very useful in reducing flooding when route discovery is required for inter cluster communications, as clusterheads only

forward to discovered border nodes as opposed to the entire cluster.

MAINTAINING PROACTIVE ROUTES: The maintainance of proactive routes is an integral part of the efficiency of this algorithm and the procedure as explained in [2], has been followed consisting of:

Advertisements: Data frames going upstream or downstream could serve as advertisements for mobiles/gateway forwarding them, to indicate the links are alive. If there are no data frames to forward for an advertisement interval , then nodes will send out a short advertisement frame periodically at a configurable periodicity. In our implementation, we have ommitted the use of advertisments piggybacking on data frames and have used a hardcoded value of 10 seconds as the time period for proactive broadcasts. Using piggybacking will be an added advantage, to reduce periodic routing overhead.

Implicit Acknowledgements: Another feature could be that the node that has forwarded a frame listens to mobiles downstream or upstream, which is supposed to forward it further. If it does not hear the frame being forwarded for a certain ACK interval, it will resend the frame, assuming the frame was lost or faced collisions. It will repeat this retry times, at the end of which it reports a route failure to the clusterhead.

Route failures and repairs: A mobile that does not hear from its predecessor or successor for an interval of 3 (3*advertisement interval) reports a failure of that route of the mobile to clusterhead. The clusterhead then removes that failed route (i.e. the corresponding VID) for the mobile and for all successor mobiles whose VIDs were derived from the failed VID. On failing to hear any activity from their predecessor or successor, a mobile reaches one of following conclusions; that either it has moved or its predecessor/successor has moved or failed.

3.2.4 Inter-Cluster Reactive Routing

The MMT reactive routing scheme, termed RMMT, for inter cluster forwarding is an extension of the above explained meshed tree approach. Figure 3.4 shows four overlapping clusters and will be used to explain the reactive routing using the proposed scheme when node J wants to communicate with node Q1, in cluster L1. Node J sends a route discovery message (with Q1s UID) to its cluster head A. Cluster head

A first checks its client list and noting that the destination is not in its cluster it will record its VID in the route record field of the discovery message. As explained in the previous section, the clusterhead has knowledge of its neighboring clusters and the shared border node VIDs (in this case 109 and 105). It will forward a copy of the route discovery message to B (border node between 100 and 109) and to H (border node between 100 and 105). Border nodes B and H will forward to the cluster heads A1 and L respectively. Cluster heads A1 and L, check their list of registered clients and as the destination is not in their cluster they will forward the discovery message to their neighboring clusters after recording their VIDs in the route record field. As A was recorded in the route record field no copy is sent to A. In the current scenario, two route discovery messages may finally reach cluster head L1. L1 forwards the route discovery message to Q1 using its primary VID from its clients list. The recorded route from J to Q1 in the discovery messages will be A, A1, L1 and A, L, L1 i.e. only the cluster head VIDs. Route reply from Q1 to J, follows the recorded path in reverse as identified by the clusters.

NOTE: The length of the recorded route is that of the clusters along the way and not all the nodes. A typical path taken by the route discovery message may be J-E-**A**-B-P-**L**-Q-R1- **L1**-Q1 while the route response message may take the path Q1-**L1**-K1-N-**L**-P-G-F-**A**-E-J (cluster heads are noted in bold). So, the route between J and Q1 depends on only the change in 3 cluster heads and not on ten or more of the intermediate nodes that will be actually forwarding the data packets. Hence, the probability of route failure during data transfer is reduced considerably, as there is dependence on only the clusterheads, and the other intermediate nodes could change depending on the topology conditions at the time of transmission. This should result in a significant reduction in route rediscovery and route maintenance. The routes however are not optimal from the perspective of the number of hops; this research study will reveal how well the advantages are able to offset the disadvantages. A detailed investigation regarding non-optimality is conducted and results are documented in Chapter 5.

Redirection capability of VIDs: In Figure 3.4 after receiving the route discovery message from A, let node P move away and loose its VID 1094, before forwarding the

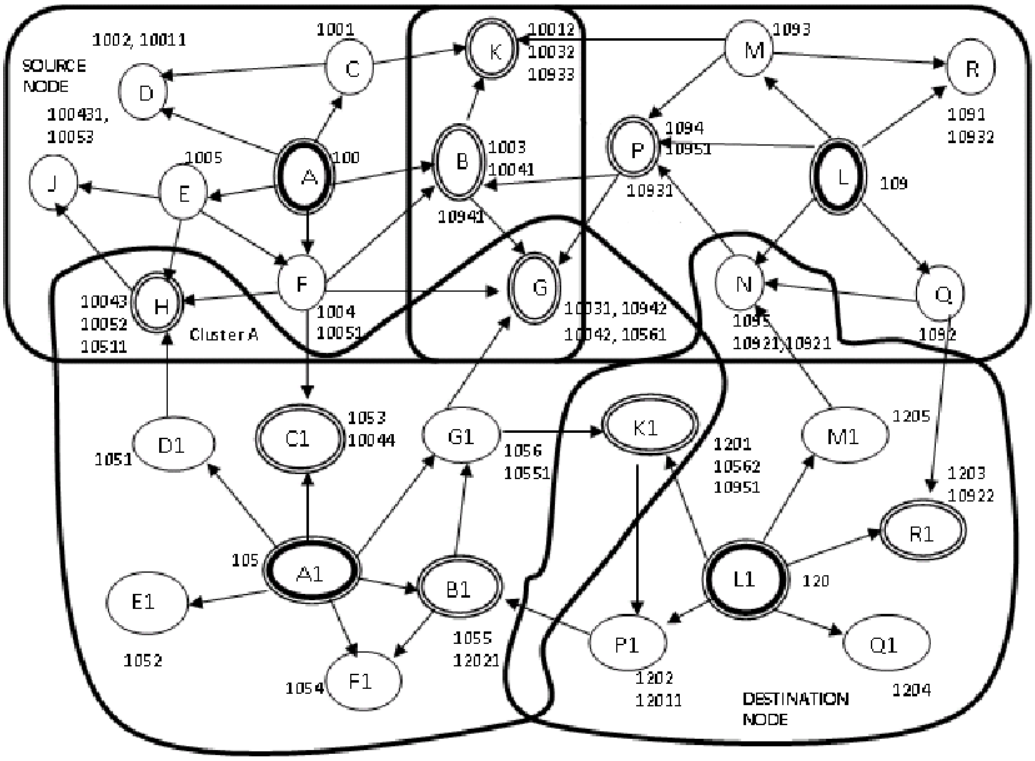

Figure 3.4: Inter-Cluster Organization

message. However it is aware that the route discovery packet is to be delivered to the cluster head L. It will use its VID 10931 to deliver the packet to the cluster head L. This redirection capability can be used while forwarding data packets too and thus the scheme is highly resilient to node movement and varying link conditions. This may seem similar to the cached route approach used in several reactive routing schemes, such as DSR explained in Section 2.2.3. However, the routes in the case of MMT have a high probability of not being stale as they are updated locally based on the neighborhood activity.

The reactive route discovery and setup for MMT uses source routing principles similar to DSR, however the reactive routes are concatenations of the proactive routes in a cluster based on bidirectional links. As the proactive routes are updated constantly on changes in link conditions the reactive routes will mostly be updated, which is a distinct advantage for MMT based reactive routes. The RMMT (Reactive protocol of MMT) route discovery process involves the standard transmission of 'route discovery' messages

and storage of 'recorded route' for the distant destination node. However MMT route discovery messages are transmitted to selected border nodes in the overlapping regions of the clusters and the recorded routes maintain only the Cluster Head information. This would lead to reduction of route discovery messages while also reducing route failure probability to the set of intermediate Cluster Heads and not all the intermediate nodes that were used for either forwarding the discovery query messages or returning the route response messages.

3.3 Advantages of the MMT Framework

Now that we have explained the operation of the MMT routing algorithm, we can get a better understanding of some of the advantages that this framework offers. This list complements the list provided in Section 3.1 which outlined the inherent features of the protocol.

- It uses a single algorithm to perform clustering and proactive routing; reactive routes are concatenation of the proactive routes.
- Reactive routes use a loose source routing concept and are impacted by only the local changes in a cluster, (which are resolved local to the cluster) and do not depend on the dynamics of all the nodes that are forwarding.
- The scheme supports the maintenance of multiple proactive routes, which is uncommon in proactive routing as algorithms used until now determine and maintain one best route and on the failure of this route (which is known through topology dissemination), recalculate another.
- Unification of tree and mesh topologies is new and helps leverage the advantages of the two.
- Multi hop multiple overlapped cluster formation is achieved in a simple way with low overhead.
- Cluster formation and setting up proactive routes are both achieved through one operation using the concept of VID's.

- Cluster clients do not maintain routing tables and states, except when communicating with nodes outside their cluster.
- No complex algorithms are involved; the simple VID scheme is novel as it facilitates all the above.

CAVEAT: One disadvantage is the non optimal routes, which was discussed earlier. The effects of this are shown quantitatively in Chapter 4.

As mentioned in Chapter 1, MANET routing protocols face a lot of challenges. This list shows how MMT provides effective solutions for some of them:

- Challenge of Scalability: Addressed by logical clustering.
- Challenge of Low overhead: Addressed by Hybrid routing (proactive and reactive), to avoid flooding.
- Challenge of Robust connectivity: Addressed by Dynamic routes with non stale backup.
- Challenge of Robust Routes: Addressed by reducing dependency on the number of forwarding nodes.

Chapter 4

Implementation of MMT

So far we have discussed the detailed operation of the Protocol and seen some of the advantages that it offers. However, so far to the knowledge of this author, there has been little work done on simulating the MMT Protocol and analyzing its Performance with other Ad-hoc Protocols in a Simulation Environment.

The main purpose and achievement of this work is the implementation, from scratch, of the above described MMT routing algorithm, followed by experiments to evaluate its performance with AODV, DSDV and DSR Protocols.

4.1 Protocol Design in ns2

We chose ns2 as the simulation tool for the implementation of the MMT Protocol. Ns2 is a powerful, open-source discrete event simulator written in C++, with an OTcl interpreter as a frontend [56].The simulator supports a class hierarchy in C++ (compiled hierarchy), and a similar class hierarchy within the OTcl interpreter (interpreted hierarchy). From the user's perspective, the two hierarchies are closely related to each other: there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class TclObject.

Ns2 was chosen as it provides a comprehensive support platform for simulation of TCP, routing and MAC protocols along with Physical layer configurations. The simulator encompasses the implementation of a large number of protocols, network types, traffic models, network elements, etc. known as Simulated Objects.

The implementation of the MMT protocol inherits and utilizes some of the generic interfaces provided by the simulator, like Agent and node implementations. The simulator is based on 2 languages: an object oriented simulator, in C++ and an OTcl

Interpreter used to execute configuration scripts. NS has a rich library of network and protocol objects which have been used extensively in the overall development and performance analysis of the MMT protocol. There are 2 class hierarchies: The compiled C++ hierarchy and the interpreted OTcl one. The former caters for efficiency in the simulation and faster execution time. This is particularly useful for the detailed definition and operation of protocols. Hence, the implementation of the MMT protocol has been done almost entirely in C++. C++ is fast to run in the sense of code execution efficiency, but slower to change(compared to an Interpreted langauge), as it needs to be recompiled making it unsuitable for varying parameter values. Hence C++ is suitable for the detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. Ns2 (via tclcl) provides glue to make objects and variables appear on both languages. The Otcl scripts are linked with the C++ components and can be used to configure simulation parameters like network topology, specific layer protocols as well as applications (The behavior of these is already defined in the compiled simulator hierarchy). The OTcl can make use of the objects compiled in C++ through OTcl linkages that creates a matching of OTcl objects for each of the C++. The class Tcl encapsulates the actual instance of the OTcl interpreter, and provides the methods to access and communicate with that interpreter. We create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy. The interpreted class hierarchy is automatically established through methods defined in the class TclClass. Instantiated objects are mirrored through methods defined in the class TclObject. As explained in [56], NS2 is a discrete event simulator, where the advance of time depends on timing of events maintained by a scheduler. An event is an object in the C++ hierarchy with a unique ID, a scheduled time and the pointer to an object that handles the event.

The ns2 wireless model essentially consists of the MobileNode at the core, shown in Figure 4.1, with additional supporting features that allows simulations of multi-hop ad-hoc networks. The MobileNode object is a split object and the C++ class MobileNode is derived from parent class Node. It is thus the basic Node object with

added functionalities of a wireless and mobile node like ability to move within a given topology, ability to receive and transmit signals to and from a wireless channel etc.

The mobility features including node movement, periodic position updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within MobileNode itself (like classifiers, dmux , LL, Mac, Channel etc) have been implemented in Otcl.

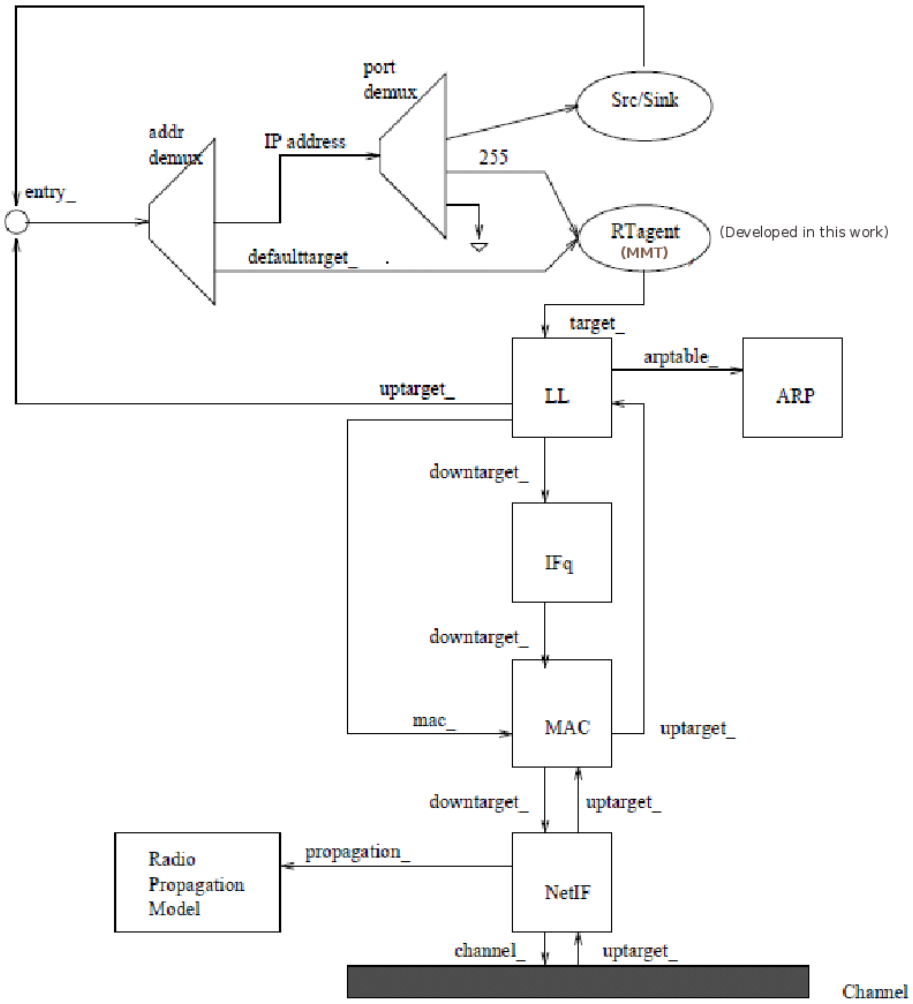

Figure 4.1: Components of the Mobile Node

Network Components in a MobileNode

As explained in [56] and shown in 4.1, the network stack for a MobileNode consists of a link layer(LL), an ARP module connected to LL, an interface priority queue(IFq),

a mac layer(MAC), a network interface(netIF), all connected to the channel. These network components are created and plumbed together in OTcl

- **Link Layer**: The link layer for MobileNode, has an ARP module connected to it which resolves all IP to hardware (Mac) address conversions. Normally for all outgoing (into the channel) packets, the packets are handed down to the LL by the MMT Routing Agent. The LL hands down packets to the interface queue. For all incoming packets (out of the channel), the mac layer hands up packets to the LL which is then handed off at the node_entry_point.

- **ARP**: The Address Resolution Protocol (implemented in BSD style) module receives queries from Link layer. If ARP has the hardware address for destination, it writes it into the MAC header of the packet. Otherwise it broadcasts an ARP query, and caches the packet temporarily. For each unknown destination hardware address, there is a buffer for a single packet. In case additional packets to the same destination is sent to ARP, the earlier buffered packet is dropped. Once the hardware address of a packet's next hop is known, the packet is inserted into the interface queue.

- **Interface Queue**: The class PriQueue is implemented as a priority queue which gives priority to routing protocol packets, inserting them at the head of the queue. It supports running a filter over all packets in the queue and removes those with a specified destination address.

- **Network Interfaces**: The Network Interface layer serves as a hardware interface which is used by MobileNode to access the channel. The wireless shared media interface is implemented as class Phy/WirelessPhy. This interface subject to collisions and the radio propagation model receives packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength etc. This meta-data in pkt header is used by the propagation model in receiving network interface to determine if the packet has minimum power

to be received and/or captured and/or detected (carrier sense) by the receiving node. The model approximates the DSSS radio interface (LucentWaveLan direct-sequence spread-spectrum).

- **Radio PropagationModel**: It uses Friss-space attenuation (1/r2) at near distances and an approximation to Two ray Ground (1/r4) at far distances. The approximation assumes specular reflection off a flat ground plane.
- **Antenna**: An omni-directional antenna having unit gain is used by MobileNodes.

Implementation Structure: As mentioned above, the Implementation of the MMT protocol was done almost entirely in C++ to facilitate reduced packet and event processing time. Following is the outline file structure used:

- mmt.h This is the header file where will be defined all necessary timers and the declaration of the routing agent which performs protocol's functionality.
- mmt.cc This file contains the important method implementations.
  - It provides the important Tcl hooks and handles the split objects(C++/OTcl). Overloads the command methods which interact with the Tcl Scripts, for network layered configuration.
  - Also overloads Recv() method which control interaction with the other necessary network layers.
  - Implementation of methods to deal with sending and receiving of the protocol packets, as mentioned above.
  - Definition of various timers.
  - Initialization of nodes (Constructors):
- mmtpkt.h Here are declared all the packets MMT protocol needs to exchange among nodes in the MANET. The structures used to represent these packets are defined here as well as header information and how they behave with the other network layers.

Now that we have our physical structure (files), let us continue with the logical one (classes). To implement a routing protocol in NS2 you must create an agent by inheriting from the Agent class, which has an implementation partly in OTcl and partly in C++. It includes enough internal state to assign various fields to a simulated packet before it is sent and supports packet generation and reception, through states and methods. Agents may be created within OTcl and an agent's internal state can be modified by use of Tcl's set function and any Tcl functions an Agent (or its base classes) implements. Some of an Agent's internal state may exist only within OTcl, and thus is not directly accessible from C++ [56].

Agents essentially represent endpoints where network-layer packets are constructed or consumed, and are used in the implementation of protocols at various layers. Class Mmt inherits from the Agent class. This is the main class representing the implementation of the MMT routing protocol. In addition, this class offers a linkage with Tcl interface, to control the MMT routing protocol parameters through simulation scripts written in Tcl. This scripting control is explained in more detail in Section 4.3.

4.2 The Routing Agent

Agents are used in the implementation of protocols at various layers. For agents used in the implementation of lower-layer protocols (e.g. routing agents), size and departure timing is generally dictated by the agent's own processing of protocol messages.

We define our Routing Agent Inside mmt/mmt.h through a new class called Mmt containing the attributes and functions needed to assist the protocol in doing its job. Mmt inherits from the Agent base class two main functions which need to be implemented: recv() and command().

- The recv() function is called whenever the agent receives a packet. This may occur when the node itself (actually an upper layer agent such as UDP or TCP) is generating a packet or when it is receiving one from another node.

- The command() function is invoked from Tcl and provides a way to ask the C++ object to do some task from Tcl code.

Another function is forwarddata(): So far we have been mainly focused on control packets, but it is time to deal with data packets. The forwarddata() function decides whether a packet has to be delivered to the upper-layer agents or to be forwarded to other node. When it is an incoming packet and destination address is the node itself or broadcast, then we use the node's dmux to accept the incoming packet. Otherwise, we must forward the packet. This is accomplished by properly setting the common header. Every Packet has a common header called hdrcmn defined in common/packet.h. There is a macro defined internally in ns2 to access this header. Our implementation returns IP BROADCAST when there is no route to destination address. In such a case we print a debug message and drop the packet. If everything goes fine then the packet is sent.

A node consists of an address classifier and a port classifier. The address classifier is used to guide incoming packets to a suitable link or to pass them to the port classifier, which will in turn carry them to appropriate upper layer agent. When it receives data packets destined to itself it will use dmux in order to give them to the corresponding agent. The Packet class inherits from the Connector class, which has a reference to a TclObject called target . This is the handler which will process the above event, and is passed as an argument to the schedule() function.

The network stack for a MobileNode consists of a link layer(LL), an ARP module connected to LL, an interface priority queue(IFq), a mac layer(MAC), a network interface(netIF), all connected to the channel. These network components are created and layered together in Otcl. One of the aims is to let Mmt to be instantiated from Tcl. To do so we must inherit from the class TclClass, which contains the methods that C++ code will use to access the interpreter. After implementing these methods, we were able to write Tcl Scripts to run the implemented MMT Protocol Front End, and provide for the Layer vise configurations.

Private Members of the Routing Agent:

*nsaddr_t ra_addr_*; // my node address(UID)

long int vid_[5]; // my VID array

*bool clusterhead_*; // Clusterhead or not

int assignedvidcounter; // used for assigning new vid's to my children

long int clustertable_[][2]; // if I am clusterhead, maintain table of UID to VID mapping.

MobileNode node_*;

PortClassifier dmux_*; // For passing packets up to agents.

Trace logtarget_*; // For logging.

The current implementation also has also been developed with NAM and trace logging for each node. NAM is the Network Animator and gives a graphical representation of the network and its activities. The initial setup of development environment was undertaken to make the necessary changes in the ns2 project to allow for creating a new protocol. Necessary code links and inheritance hierarchies were setup in order to integrate the MMT code inside the simulator.

Packet Types:

1. *Broadcast VID packet*: If I have VID assigned then announce my VID.

2. *Request for VID packet*: Requests for a VID from the node sending above packet and provides its UID

3. *Assigning VID packet*: Contains VID to be assigned.

4. *Register with Clusterhead packet* :Contains sourceuid and assignedvid.

5. *Registration accept packet*: From clusterhead.

6. *Data Packet.*

Packet headers:

Objects in the class Packet are the fundamental unit of exchange between objects in the simulation. The class Packet provides enough information to link a packet on to a list (i.e., in a PacketQueue or on a free list of packets), refer to a buffer containing packet headers, and to refer to a buffer of packet data. The new packet header for MMT was introduced into the simulator by defining a C++ structure with the needed fields for the different packet types mentioned above. This was followed by defining a static class to provide OTcl linkage, and then modifying some of the simulator initialization code to assign a byte offset in each packet where the new header is to be located relative

to others. The static class variable offset_ is used to find the byte offset at which the header is located in an arbitrary nspacket. We used the method access() to utilize this variable to access this header in any packet. We also defined a member function which creates a new packet to send by calling allocpkt(), which handles assignment of all the network-layer packet header fields. There were essentially four steps to create new Packet header types:

- Create a new structure defining the raw fields for the different MMT packet types, define offset_ and access methods.
- Define member functions for needed fields.
- Create a static class to perform OTcl linkage, do bind_offset() in its constructor.
- Make appropriate changes to ñs/tcl/lib/ns-packet.tcl to enable the new MMT packet format

4.3 Sample Scenario and Network Configuration

This section provides a walkthrough of how the protocol is run by defining its configuration parameters in Tcl. Also, through a sample network topology, the setup of the network and the flow of data is explained in accordance with the above mentioned implementation details. The algorithm pseudocode is provided at appropriate places.

The configuration of the network is done through a Tcl script. For our sample network topology, we chose 16 nodes in a scattered topology. They were configured to have unity gain omni directional antenna's and used the 802.11 MAC. The wireless propagation model was the Two Ray Ground Propagation Model.

The sample network is shown in Figure 4.2. Here nodes 0 and 15 marked in blue are chosen as the clusterheads and two independent clusters are formed around them. At this point there is no gateway node.

The clusterheads register their Unique Id's (UID) equal to their Virtual ID's (VID) and start broadcasting their VID's periodically through *Broadcast VID packets*. The periodicity of this broadcast is a configurable parameter and was kept as 5 seconds in

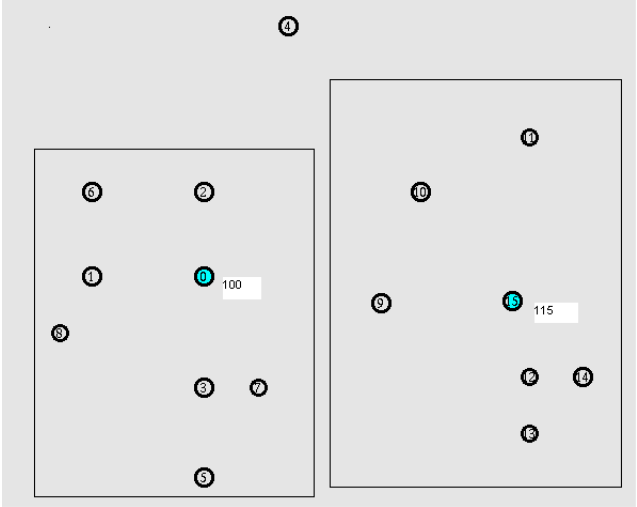

Figure 4.2: Sample Network Topology(NAM Screenshot)

our simulations. Also, clusterheads have only one VID. Here, nodes 0 and 15 broadcast their VID's 100 and 115 as shown in Figure 4.3.

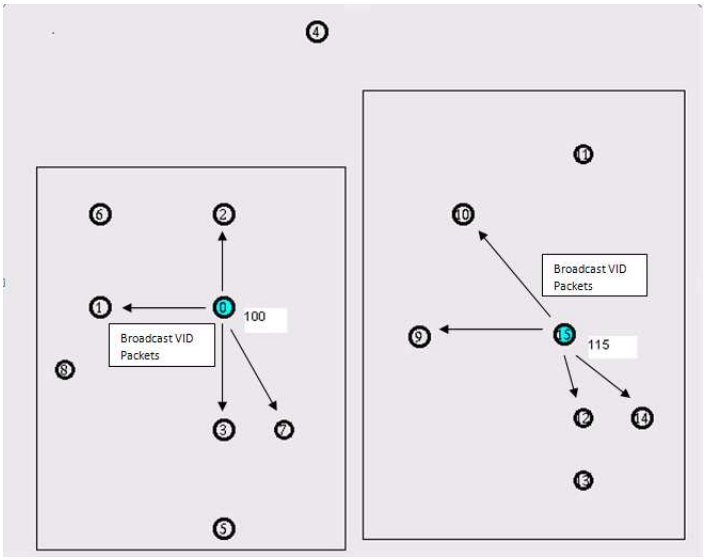

Figure 4.3: Clusterheads Broadcasting their VID's

These *Broadcast VID packets* are received by nodes which are within range. In this case nodes 1,2,3 and 7 receive the broadcast from 0 and carry out the following actions.

Pseudocode:

For *Broadcast VID packet* received :

1. Check if I have a vid

- no: send *Request for VID packet* back to node from where it was received.
- yes: check if any my VIDs is a child of broadcast VID
  - yes: Update state flag. Send Confirmation packet to clusterhead.
  - no: Check for Loops: If none then send *Request for VID packet* back to node from where packet was received.

Nodes 1,2 ,3 and 7 hence send *Request for VID packets* back to the clusterhead(node0), as shown in Figure 4.4. The clusterhead then checks to see if it can allocate any more VID's. This is where the size of the cluster is defined and controlled(configurable parameter).

Pseudocode:

For *Request for VID packet* received:

1. Check if I can allocate more VID's

- no : do nothing. (Maximum size of the cluster has been reached)
- yes : send *Assigning VID packet* back to node which sent this packet.

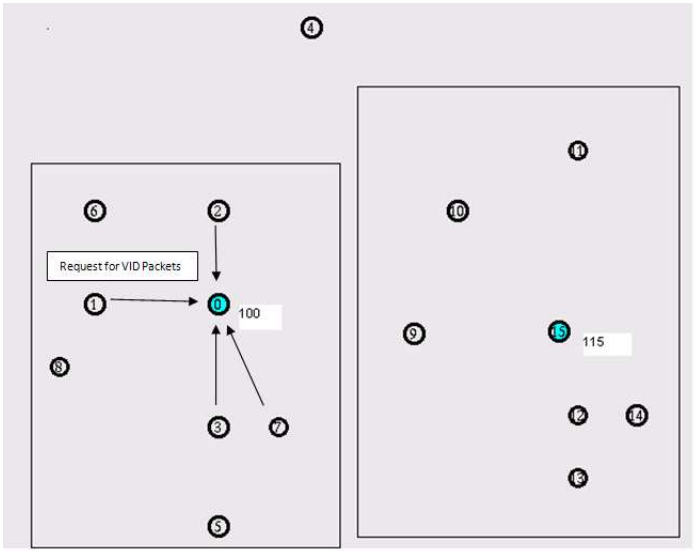

Figure 4.4: Nodes request the clusterhead to assign them a VID

If the maximum size of the cluster is not reached, the clusterhead then sends *Assigning VID packet's* back to nodes 1,2,3 and 7 which are assigned VID's 1001, 1002,1003

and 1004 respectively, as shown in Figure 4.5

Pseudocode:

For *Assigning VID packet* received

1. Assign my new received VID to my VID array.
2. Register with clusterhead by sending *Register with Clusterhead packet.*
3. Broadcast my newly acquired VID sending *Broadcast VID packet.*

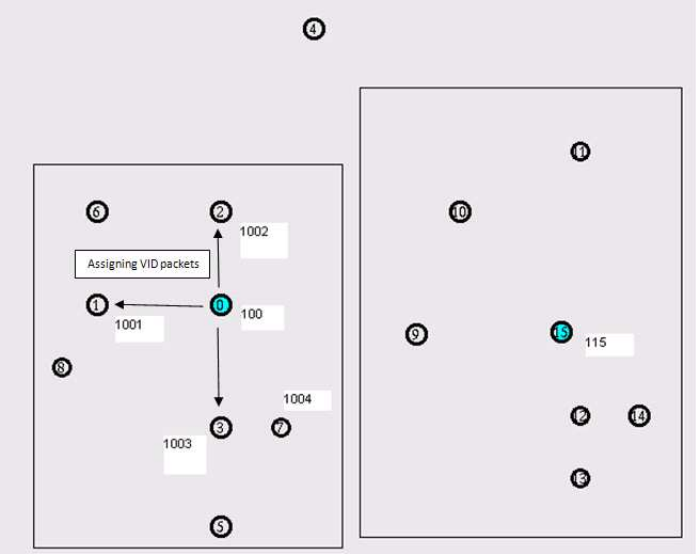

Figure 4.5: Clusterhead assigns VID's

So the nodes are assigned their VID's following which they register with the clusterhead (If they are border nodes they register with multiple clusterheads). They then broadcast their newly acquired VID's as shown in Figure 4.6. This process occurs everytime a node receives a new VID.

Pseudocode:

For *Register with Clusterhead packet* packet received

1. Am i clusterhead?

- yes: update my clusterhead Table.
- no: forward to clusterhead.

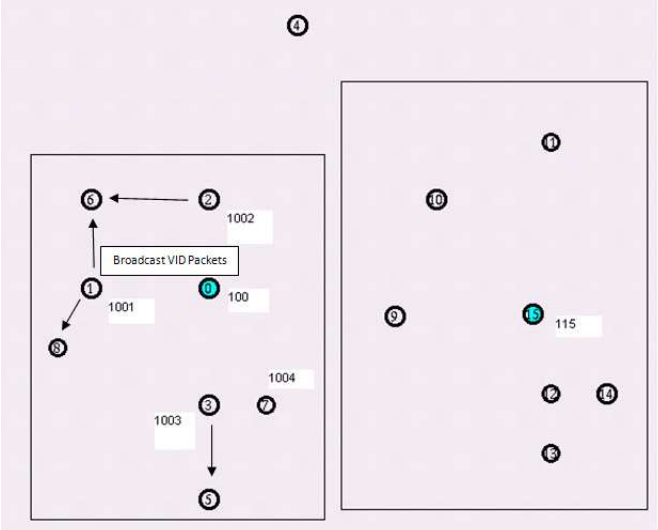

Figure 4.6: Nodes broadcast their newly acquired VID's

This informs the clusterheads of which nodes are associated to its cluster through a table maintained at each clusterhead. The process then continues as VID's are broadcast and new nodes at further hops away, register with the clusterhead. This is shown in Figure 4.6where, for example node 8 receives a Broadcast VID packet from node 1, following which it is assigned VID 10011 by node 1 and then registers this VID with the clusterhead. The final stable network with asssigned VID's is shown in the Figure 4.7.

Intra-Cluster Proactive Routing:

Once the network initially stabilizes, ie. appropriate allocation of VID's to the respective nodes has completed, we can now send Data from one node to another. This process is outlined below. Data traffic was generated in ns2 through a Constant Bit rate Traffic application working over a UDP Agent (Best effort service).

Pseudocode:

For *Data Packet* received

1. Check if I am final destination

- yes : Display Payload contents
- no : Check if I am clusterhead
  - yes : Update destination vid by looking at table. Find nexthop which is first

4 digits of dvid. Update direction bit.

- no : Check direction
  - if 0 (towards clusterhead) update nexthop by /10 and resend.
  - if 1 (away from clusterhead) count no. of digits in my VID. put first (count+1) digits of destination vid in nexthop.

In our network topology let us supposse that a data packet is sent from node 8 to node 5. Node 8 does not know where in the network node 5 is and hence forwards the payload packet to its clusterhead (node 0). This happens through node 1 as node 8 knows that the VID assigned to it is through node 1 which is the next hop enroute to the clusterhead. A *direction bit* is used in the packet to indicate whether it is travelling towards the clusterhead or away from it. This is required, so that the appropriate VID arithmetic is carried out at the receiving node to calculate the next hop. This is explained in the above pseudocode and depends on the direction bit. When the clusterhead receives the Payload packet it checks its clusterhead table to see if the destination UID is present. If it is, then the destination node's VID is obtained and the next hop is calculated based on VID arithmetic as shown above. In our example, node with UID 5 is found in the clustertable at node 0. It hence knows that destination has primary VID 10041. It then calculates the next hop as first 4 digits (1004), updates the direction bit and sends the Payload packet to node 7. Node 7 then notes that the direction is away from clusterhead, hence calculates nexthop (as described above) to be 10041, which is the destination node 5.The path followed by the data for this Intra-cluster transfer is shown in Figure 4.7

Inter-Cluster Reactive Routing:

Now, let us suppose Node 4 moves down in the topology and comes into range of both nodes 2 and 10 as shown in Figure 4.8. Hence it is assigned VID's from both these nodes which are associated with different clusterheads. Consequently, node 4 now acts a gateway node to connect the previously disjoint networks. Also, as it registers with the two clusterheads, the clusterheads consequently are aware that it is a border node and make note of this in their clustertables. This is done by simply checking the VID's of

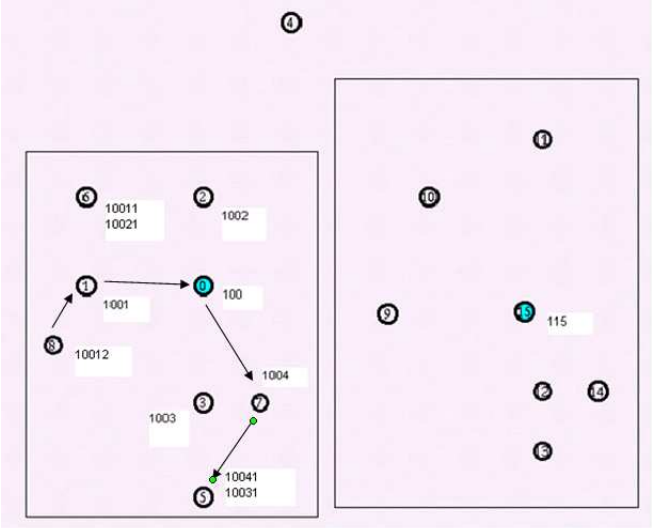

Figure 4.7: Path followed for flow of data from node 8 to node 5(Intra-cluster)

the new registered node for a VID starting with a different number indicating a different cluster. For example, as node 4 moves within range of 2 and 10, it hears their periodic *Broadcast VID packet's* and is assigned a VID of 10022 from node 2 and VID of 11532 from node 10 (node 10 was registered as 1153). So when these VID's are registered with the clusterheads, the clusterhead 0 becomes aware that node 4 is a border node as it has a VID starting with 115 of a different cluster, and so does clusterhead 15. This is the way by which clusterheads are made aware of which nodes in their clustertables are border nodes. This is the precursor to the reactive routing process by which nodes communicate across clusters. Let us take an example to explain this further.

In our sample topology, we have so far looked at the process by which nodes can communicate with other nodes within the same cluster through the clusterhead with minimal overhead and simple VID arithmetic. Let us now look at how nodes communicate with other nodes outside their cluster through the reactive routing process as described in the previous section. If node 8 wishes to transfer data to node 13, the payload packets first travel from node 8 to its clusterhead node 0. This is done in the same way as for the Intra-cluster routing algorithm as explained in the previous section on the basis of simple VID arithmetic. Now, when the payload packet reaches node 0, it realizes that the destination UID (13) is not in its clustertable as it is not

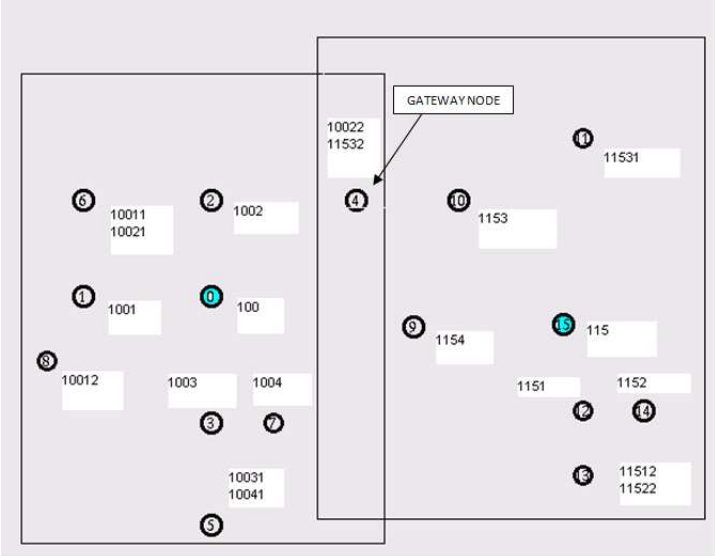

Figure 4.8: Introduction of gateway node 4

part of its cluster. This is when it forwards the packet out to its bordernodes which are distinguished from other nodes in the cluster by the method explained above. The payload packet is hence forwarded to the border nodes which in this case is node 4.

Pseudocode:

At clusterhead if there is no entry for destination UID in cluster table:

- Search clustertable for border nodes: loop through clustertable, if first 3 digits don't match my UID, then that is a border node VID
- Send payload packet to the acquired bordernode VID through the approach similar to the proactive part.
- At Border node, switch direction and then forward to the other clusterhead(as specified by the non-matching acquired VID).

So when the packet reaches Border Node 4 it will change the direction and forward it to clusterhead 15. Clusterhead 15 will then search through its clustertable and finds the Destination UID 13. It will then forward it to the destination similar to the proactive process. If it doesn't find the destination UID, then it must find other border nodes repeating the process mentioned above. The path followed by the data for this Inter-cluster transfer is shown in Figure 4.9

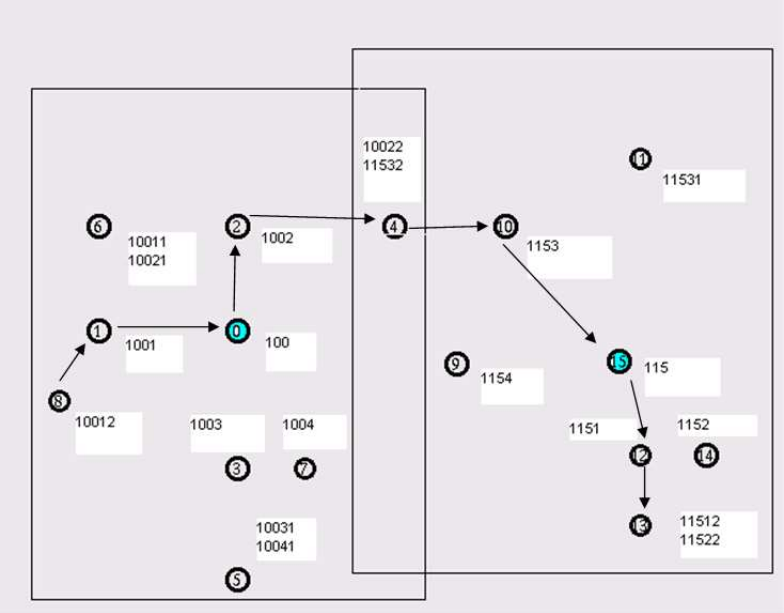

Figure 4.9: Path followed for flow of data from node 8 to node 13(Inter-cluster)

This is how inter-cluster packet communication through a border node takes place. Again, this is a reactive process and routes are discovered only when the clusterhead does not find the destination in its clustertable.

Summary: This chapter has explored in detail the working and implementation (in ns2) of the Multi Meshed Tree routing algorithm through the use of sample network topology covering Network setup as well as both Inter and Intra cluster data transfer protocols.

Chapter 5

Simulation Setup and Results

The previous chapter has discussed the core implementation of the MMT Protocol in ns2. In this section, we describe the test-bench and simulations used to analyze the performance of the protocol.

5.1 Simulation Setup

The core operation of the MMT Protocol was implemented in C++ as described in the previous chapter. In order to use and test the operation of the protocol in ns2, we create configuration scripts in Tcl.

5.1.1 Tcl Configuration Script

In ns2 Tcl simulation scripts are used to define the network topology, the agents used, output tracing configurations as well as NAM configurations.

Let us take an example Tcl script to explain the various configuration parameters:

- Configurations

  set opt(chan) Channel/WirelessChannel // Channel type

  set opt(prop) Propagation/TwoRayGround // Radio Propagation Model

  set opt(netif) Phy/WirelessPhy //Network Interface Type

  set opt(mac) Mac/802_11 // MAC Protocol to be used

  set opt(ifq) Queue/DropTail/PriQueue // Drop Tail Queue

  set opt(ll) LL

  set opt(ant) Antenna/OmniAntenna // Antenna Type

set opt(x) 670 ;// X dimension of the topography

set opt(y) 670 ;// Y dimension of the topography

set opt(ifqlen) 50 ;// max packet in ifq

set opt(tr) "" ;// trace file location

set opt(nam) "" ;// nam trace file location

set opt(adhocRouting) MMT // Routing Protocol used

set opt(nn) 48 ;// how many nodes are simulated

set opt(cp) "" ; //Connection Pattern

set opt(sc) ""; //Scenario File

set opt(stop) 500.0 ;// simulation time

- Link Layer settings:

  LL set mindelay_ 50us

  LL set delay_ 25us

- Configure unity gain, omni-directional antennas centered in the node and 1.5 meters above it

  Antenna/OmniAntenna set X_ 0

  Antenna/OmniAntenna set Y_ 0

  Antenna/OmniAntenna set Z_ 1.5

  Antenna/OmniAntenna set Gt_ 1.0

  Antenna/OmniAntenna set Gr_ 1.0

- Initialize the SharedMedia interface with parameters to make it work like the 914MHz Lucent WaveLAN DSSS radio interface

  Phy/WirelessPhy set CPThresh_ 10.0

  Phy/WirelessPhy set CSThresh_ 1.559e-11

  Phy/WirelessPhy set RXThresh_ 3.652e-10

Phy/WirelessPhy set Rb_ 2*1e6

Phy/WirelessPhy set Pt_ 0.2818

Phy/WirelessPhy set freq_ 914e+6

Phy/WirelessPhy set L_ 1.0

As explained in the previous chapter, the MobileNode is at the core of the ns2 wireless model simulations. It is a split object inherited from Node Class with added functionalities like mobility and interfacing with the wireless channel.Hence for our simulations we instantiate MobileNode objects:

```
for{ set j 0 } { $j < $opt(nn)} {incr j} {
set node_($j) [ $ns_ node ]
$node_($i) random-motion 1
}
```

The above steps create a mobilenode (split)object, an adhoc-routing agent as specified, as well as the network stack consisting of a link layer, interface queue, mac layer, and a network interface with an antenna, using the defined propagation model. It then interconnects these components and connects the stack to the channel. Each mobilenode is set with a random position and has routine updates to change the direction and speed of the node, through setdest as explained below. The destination position and speed values are generated in a random fashion.

Irrespective of the methods used to generate node movement, the topography for mobilenodes needs to be defined. It should be defined before creating mobilenodes. Normally flat topology is created by specifying the length and width of the topography using the following primitive:

set topo [new Topography] $topo load_flatgrid $opt(x) $opt(y)

where opt(x) and opt(y) are the boundaries used in simulation.

These mobilenodes move about within an area whose boundary is defined as 670mX670m. They use Two Ray ground Propagation model and 802.11 MAC.

Following these configurations, the Simulator is instantiated and trace objects for ns tracing and NAM (Network Animator) are created to write logging data to files specified in the configuration. A topology object is also created that keeps track of movements of mobilenodes within the topological boundary, as well as a GOD (General Operations Director) object. This is the object that is used to store global information about the state of the environment, network or nodes that an omniscent observer would have, but that should not be made known to any participant in the simulation. God object stores the total number of mobilenodes and a table of shortest number of hops required to reach from one node to another. The next hop information is normally loaded into god object from movement pattern files, before simulation begins, since calculating this on the fly during simulation runs can be quite time consuming. For our simulations we enable random motion of the nodes. The setdest program generates movement pattern files using the random waypoint algorithm. We use this to create 5 different topologies to run in the simulations. A pause time of 10 seconds and Max speed of 5m/s was chosen and for MMT, 5 appropriate clusterheads were hardcoded. The node-movement files generated using setdest already include lines to load the god object with the appropriate information at the appropriate time.

The connection pattern file specifies the Traffic Agents created. The cbrgen.tcl script provides an easy way of setting up source/destination pairs. For our simulations we used Constant bit Rate (CBR) Traffic application working over a UDP Agent. Best effort delivery of packets is suited for a mobile ad-hoc network, which led us to chose UDP instead of TCP. The overhead of setting up a TCP connection is usually very high and unsuitable for this dynamic environment.

- Traffic Generator Configurations

  set udp_(0) [new Agent/UDP] //Instantiate UDP Agent

  $ns_ attach-agent $node_(43) $udp_(0)

  set null_(0) [new Agent/Null] // Traffic Sink

  $ns_ attach-agent $node_(28) $null_(0)

set cbr_(0) [new Application/Traffic/CBR] // Constant Bit Rate Traffic Application

$cbr_(0) set packetSize_ 512

$cbr_(0) set interval_ 4.0

$cbr_(0) set random_ 1

$cbr_(0) set maxpkts_ 10000

$cbr_(0) attach-agent $udp_(0)

$ns_ connect $udp_(0) $null_(0)

$ns_ at x "$cbr_(0) start"

Simulation Stack:

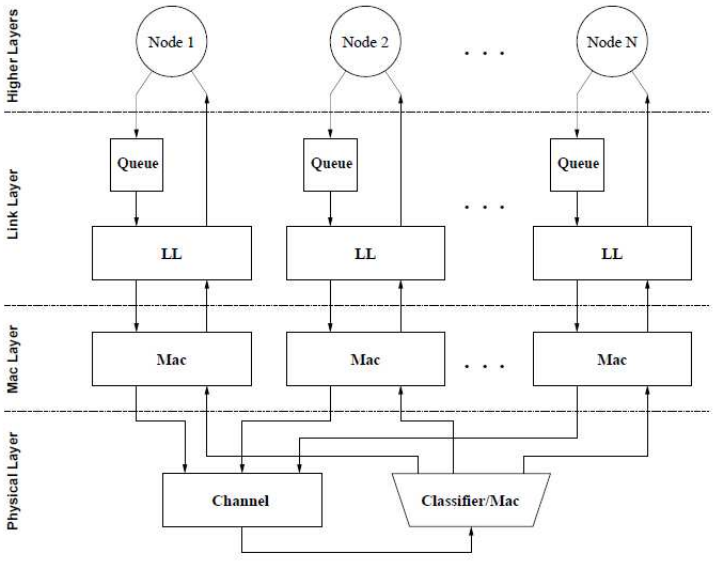

Figure 5.1: Simulation Stack

Figure 5.1 illustrates the extended network stack that makes simulations of local area network possible in ns. A packet sent down the stack flows through the link layer (Queue and LL), the MAC layer (Mac), and the physical layer (Channel to Classifier/Mac). The packet then makes its way up the stack through the Mac, and the LL.

At the bottom of the stack, the physical layer is composed of two simulation objects: the Channel and Classifier/Mac. The Channel object simulates the shared medium and

supports the medium access mechanisms of the MAC objects on the sending side of the transmission. On the receiving side, the Classifier/Mac is responsible for delivering and optionally replicating packets to the receiving MAC objects. Depending on the type of physical layer, the MAC layer must contain a certain set of functionalities such as: carrier sense, collision detection, collision avoidance, etc. Since these functionalities affect both the sending and receiving sides, they are implemented in a single Mac object. For sending, the Mac object must follow a certain medium access protocol before transmitting the packet on the channel. For receiving, the MAC layer is responsible for delivering the packet to the link layer. The IEEE 802.11 MAC, that is used in our MMT simulations, uses the distributed coordination function (DCF) with both physical and virtual carrier sense.

Above the MAC layer, the link layer can potentially have many functionalities such as queuing and link-level retransmission. The need of having a wide variety of link-level schemes leads to the division of functionality into two components: Queue and LL (link-layer).

The LL object implements a particular data link protocol, in this case ARQ. By combining both the sending and receiving functionalities into one module, the LL object can also support other mechanisms such as piggybacking.

The Channel class simulates the actual transmission of the packet at the physical layer. The basic Channel implements a shared medium with support for contention mechanisms. It allows the MAC to carry out carrier sense, contention, and collision detection. If more than one transmissions overlaps in time, a channel raises the collision flag. By checking this flag, the MAC object can implement collision detection and handling. Since the transmission time is a function of the number of bits in the packet and the modulation speed of each individual interface (MAC), the Channel object only sets its busy signal for the duration requested by the MAC object. It also schedules the packets to be delivered to the destination MAC objects after the transmission time plus the propagation delay. The C++ class Channel includes enough internal state to schedule packet delivery and detect collisions. OTcl configuration parameter: delay_ propagation delay on the channel

5.1.2 Trace Support

Tracing support was also provided for the implemented protocol through relevant changes in the CMU trace classes. The trace support for wireless simulations uses cmu-trace objects which are of three types - CMUTrace/Drop, CMUTrace/Recv and CMUTrace/Send. These are used for tracing packets that are dropped, received and sent by agents, routers, mac layers or interface queues in ns. The type field (described in Trace class definition) is used to differentiate among different types of traces. For cmu-trace this is s for sending, r for receiving or D for dropping a packet. A fourth type f is used to denote forwarding of a packet (When the node is not the originator of the packet). Similar to the method Trace::format(), the CMUTrace::format() defines and dictates the trace file format

An example of a trace for a tcp packet is as follows:

```
r 160.093884945 _6_ RTR --- 5 tcp 1492 [a2 4 6 800] -------
s[65536:0 16777984:0 31 16777984] [1 0] 2 0
```

Here we see a TCP data packet being received by a node with id of 6. UID of this pkt is 5 with a common header size of 1492. The MAC details shows an IP pkt (ETHERTYPE_IP is defined as 0x0800, ETHERTYPE_ARP is 0x0806), mac-id of this receiving node is 4. That of the sending node is 6 and expected time to send this data pkt over the wireless channel is a2 (hex2dec conversion: 160+2 sec). Additionally, IP traces information about IP src and destination addresses. The src translates (using a 3 level hier-address of 8/8/8) to a address string of 0.1.0 with port of 0. The dest address is 1.0.3 with port address of 0. The TTL value is 31 and the destination was a hop away from the src.

A Trace object is used to write wanted information of a packet everytime it is received, sent or dropped. To log information regarding our packet type we implement the format protoname() function inside the CMUTrace class. Appropriate changes are made in cmu-trace.h, cmu-trace.cc to print the necessary MMT protocol information like Sorce UID, Destination UID, Nexthop VID and Packet Type.

There is also a separate log of changes made to the clusterheads cluster tables. This helps keep track of the snapshot of a cluster at a particular instant of time. It also logs any changes in the VID list of a node, including adding a VID, removing a VID and change in VID priority. Example of a clustertable log:

```
Displaying Cluster Table for VID: 115
13 1151 0 0 0 0
23 1154 0 0 0 0
14 1156 0 0 0 0
9 1152 0 0 0 0
21 1153 11522 0 0 0
12 1157 0 0 0 0
27 1155 0 0 0 0
```

The first column represents the UID (particular node). This is followed by the VID list of that node in order of preference. Multiple VID's associated with a node represent the redundant routes responsible for robust connectivity.

5.2 Simulation Results

Once the implementation of the protocol was in place, we looked to evaluate its performance compared to other MANET Protocols. We chose the widely popular DSDV (Proactive), AODV and DSR (Reactive) protocols as reference. The below results were averaged over the 5 topologies as created by the setdest function explained earlier. The following sections cover the comparative simulation tests run, as per the TCl specifications mentioned above.

5.2.1 Packet Delivery Ratio

The first test was with regard to the route robustness of the MMT Protocol compared to the other MANET protocols. This is an indication of how resilient a route is to change in topology. Packet delivery ratio is the ratio between the number of packets originated

by the application layer Constant Bit Rate (CBR) sources and the number of packets received by the CBR sink at the final destination. A CBR traffic generator worked over UDP (Best effort) in a topology with random motion. The number of packets transmitted to the number of packets received was calculated as the Packet delivery ratio. This is plotted as a function of optimal number of hops taken from source to destination. Figure 5.2 shows the comparison of the protocols. We see that the Packet delivery Ratio decreases as the number of hops increases. The effect of this is worse on DSDV, and we see that MMT fares much better than the proactive protocol, especially for large number of hops. This shows the effects of route robustness of the MMT protocol where redundant routes allow it to drop fewer packets. This is more evident for the larger number of hops, where the Reactive protocol kicks in, which depends only on the clusterheads traversed. This reduced route dependency, shows improved Delivery Ratio. The packet delivery ratio of the proactive protocol is seen to be lower than the others. This can be attributed to the fact that the routes are pre-calculated in these scenarios, and as a result of node mobility, the routes become relatively stale at the time of packet delivery. This reduces the delivery ratio as compared to reactive protocols wherin the routes are relatively fresher due to their recent discovery. The MMT protocol is seen to have better results than DSDV owing to the increased route robustness inherent in the algorithm. The initial packet delivery for 1 hop is seen to be the same for all protocols, however at 3 hops MMT tends to perform inferior to AODV. This can be attributed to the fact that nodes at 3 hops from the source will typically be at the edge of a cluster (as the max cluster size is kept as 3 hops from clusterhead). Nodes at this position are the most vulnerable to changes in motion as that would specify which cluster they belong to and are at the transition between the reactive and proactive componenets. There is also an inherent relative staleness of routes which depends on the periodicity of updates in the proactive componenet of MMT (a configurable parameter which was kept as 10 seconds for this simulation). These factors contribute to the lower Packet delivery ratio at small number of hops.

As the number of hops increases, the MMT results look closer to AODV as the network stabilizes and the packet delivery enters the reactive part. The routes are

robust and more fresh due to their on-demand discovery. As we can see, for higher number of hops, the MMT packet delivery ratio surpasses that of AODV. This can be attributed to the more robust nature of MMT brought about by the non-dependence on individual nodes, but instead on just the clusterheads. This better avoids the posibility of stale routes due to less dependenceon intermediate nodes keeping routes alive longer.

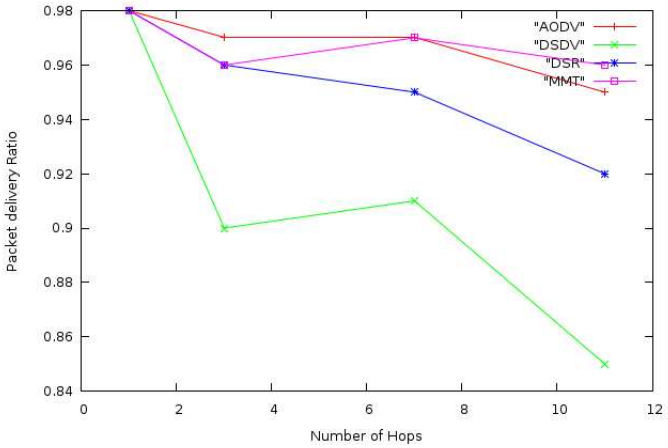

Figure 5.2: Packet Delivery Ratio

5.2.2 Routing Overhead

Routing overhead is an important metric for comparing these protocols, as it measures the scalability of a protocol, the degree to which it will function in congested or low-bandwidth environments, and its efficiency in terms of consuming node battery power. It is calculated as the total number of routing packets transmitted during the simulation. For packets sent over multiple hops, each transmission of the packet (each hop) counts as one transmission. Protocols that send large numbers of routing packets can also increase the probability of packet collisions and may delay data packets in network interface transmission queues. We did not include IEEE 802.11 MAC packets or ARP packets in routing overhead, since the routing protocols could be run over a variety of different medium access or address resolution protocols, each of which would have different overhead. Our goal was to compare the routing protocols to each other, not to find the optimal performance possible in our scenarios. Unlike some other proactive protocols, the MMT protcol does not engage in flooding of packets which reduces its

control overhead.

As shown in Figure 5.3, we see that the routing Overhead of DSDV is the highest and is relatively constant as expected of a Proactive Protocol. We also see that the overhead of the MMT protocol is considerably lower than that of DSDV. It increases as the number of hops increases as a result of Reactive Request packets for the Inter-cluster communication along with the Intra-cluster Periodic Overhead.

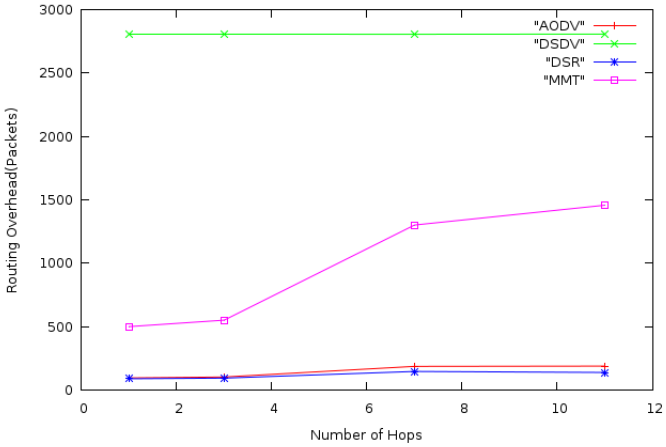

Figure 5.3: Routing Overhead

5.2.3 End-to-End Delay

End-to-end delay is the time taken for a packet to be transferred across a network from source to destination. It includes transmission delays, propagation delays as well as processing delays. For this experiment, we varied the number of hops between the source node and destination and averaged the end-to-end delay over all packets transmitted. The results of the experiment are shown in Figure 5.4. We see the DSDV, DSR and AODV protocols having roughly the same end-to-end delay for adjascent nodes, but the DSDV shows the least end-to-end delay as the number of hops increases. This can be attributed to the proactive nature of the protocol, wherin routes are pre-calculated, so the extra time taken to calculate a route is absent. That said it must be kept in mind that although DSDV exhibits a low delay, it is because only packets belonging to valid routes at the sending instant get through (The Traffic generator is over UDP, which is a best effort service). Several packets do get lost until new routing table

updates propagate through the network. For the MMT protocol, there is a significantly larger delay even for data transmission between adjascent nodes. This is because the protocol is designed so that all traffic must pass through the cluster-head. This leads to the extra delay for traffic to propagate through to the cluster head, and then to the destination node. Further, we can see very slight difference for delay between 0 hop transmissions and 1 hop transmissions. This is because transmission of data to any node within the cluster will take roughly the same amount of time. This is a useful feature for scenarios where we require frequent data transfer between a subset of nodes, with occasional transfer outside the subset. We can control the cluster-size to that of the subset requiring frequent communication, and have relatively deterministic delay estimates provided the nodes are not moving out of the cluster too frequently. We see an expected increase in the end-to-end delay for all protocols as the number of hops between source and destination increases. It must be noted that in MMT, for higher number of hops, when the reactive protocol is dominant, the increase in the delay is more gradual as seen from the line plateauing towards the end.

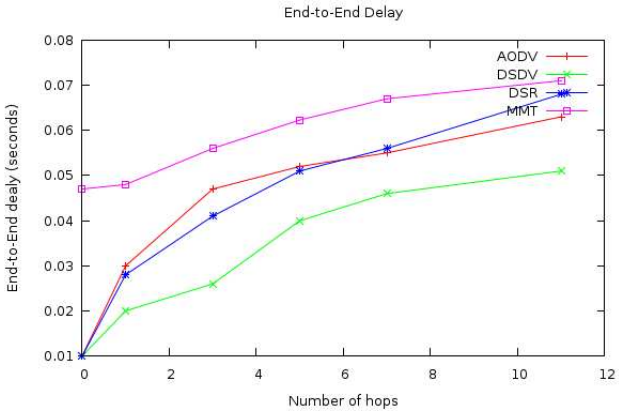

Figure 5.4: End-to-End Delay

5.2.4 Effects of Non-optimality

It must be kept in mind that although the MMT protocol is highly robust in nature, provides multiple routes, and has realtively low overhead compared to proactive protocols, there is an important factor of non-optimality which must be considered. Data

transmission is not along the shortest path since all traffic must pass through the cluster head. The discussion on end-to-end delay showed this effect of non-optimality as one of the factors increasing the delay in data transfer.

We conducted an experiment to see the effects of Route non-optimality by comparing the number of hops taken for data to reach from source to destination by MMT versus AODV as a benchmark. For this experiment, we first identified source-destination pairs which used a certain number of hops on AODV at a particular snapshot of the topology. We then ran MMT for the same source-destination pair and noted the change in the number of hops. We carried this out over 5 different topologies created using the setdest function described earlier. We varied the number of hops and noted the average divergence from the AODV benchmark. A max clustersize of 4 was chosen in a 30 node stationary network.

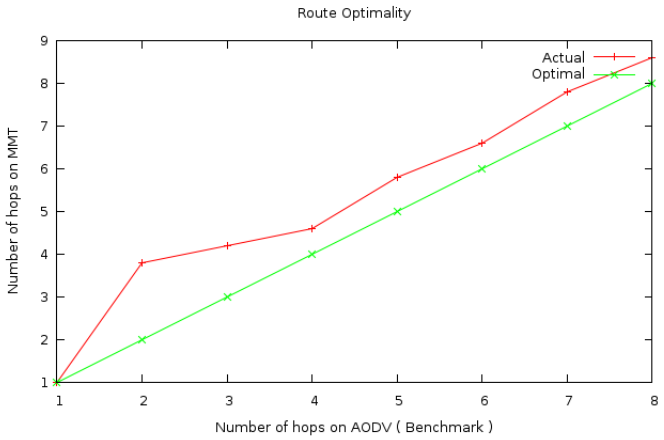

Figure 5.5: Route Optimality compared to AODV (Benchmark)

The results are shown in Figure 5.5. As we can see, the divergence from AODV is larger for shorter number of hops and tends to decrease as the number of hops increases. This is expected, as for a small number of hops, the Intra-cluster proactive protocol is predominant and all traffic must pass through the clusterhead. This restriction leads to longer routes and the effects of this would be dependent on the size of the cluster. AODV does not have this restriction of data having to pass through certain nodes in

the network and hence finds more optimal routes. Even at a higher number of hops we do notice around a 0.6-0.8 increase in number of hops for MMT. This can be attributed to the overall restriction of including clusterheads in all reactive routes as well.

The effects of non-optimality are more visible for data transfer over shorter hops, and is dependent on the cluster size. As the cluster-size increases, data will have to propagate over a larger number of nodes increasing the delay. That said decreasing the clustersize too much will have the adverse effect of too many reactive protocol route setups, which also contibute towards delay. A trade-off must be drawn, and the MMT structure provides that capability by allowing the cluster-size to be changed depending on the requirements of the network. Different networks are setup for different applications, and MMT provides the Network deployer with the virtual knobs of controlling clustersize and maximum number of hops, to tune the networks for tradeoffs in efficiency of certain application specific parameters, such as delay, throughput or robustness. The next section describes the aspect of configurability in the MMT protocol.

5.2.5 Configurability

A key feature of the MMT Protocol is its configurability, which includes controlling the size of the cluster. In this section we explain the effects of changing the cluster size on the Routing Overhead. This is done by controlling the Maximum number of hops from the clusterhead at which a node can join that particular cluster. We consider a 50 node network with 3 data sources transmitting packets through a Constant-Bit-Rate generator. To accomodate all nodes, clusterheads were added as the clustersize reduced. Effects of varying clustersize is shown in Figure 5.6

As we can see the increase in Routing Overhead is exponential with respect to increasing clustersize. This is attributed to an increase in Proactive routing traffic within the cluster. There are a larger number of redundant routes setup requiring more Request for VID packets, Assigning VID packets, and VID maintainance. This overhead grows as the clustersize increases, and hence will effect the throughput of the network. Another important factor to keep in mind is congestion at the clusterheads. Since all traffic associated with a cluster passes through the clusterhead, as the clustersize

increases, there is an increased load through the clusterhead. Bandwidth restrictions, Queuing limitations and congestion at the clusterheads become important factors to consider. Figure 5.7 indicates the Average total traffic passing through the clusterheads for varying size of the cluster. We see a large increase in Traffic for the larger clusters as the clusterheads are handling a larger number of data sources. The load is transferred from several clusterheads to a single clusterhead, causing an increase in traffic. These are important factors to be kept in mind while choosing cluster size and are dependent on the requirements of the network.

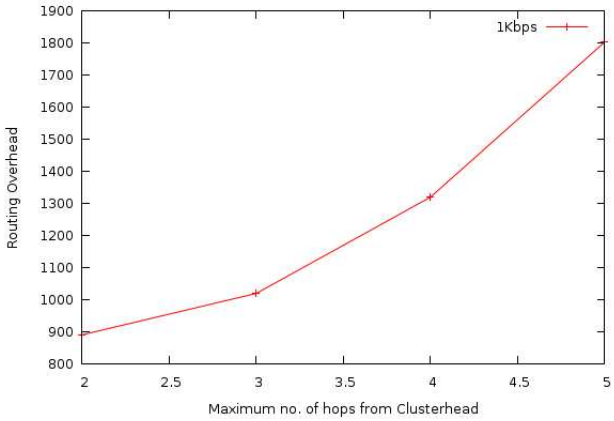

Figure 5.6: Cluster-Size Configurability

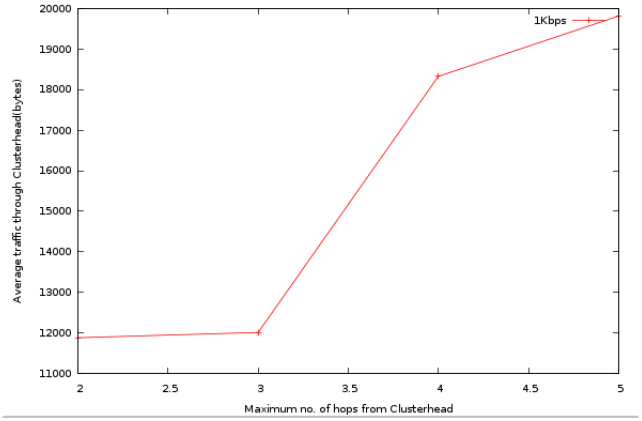

Figure 5.7: Cluster-Size Configurability 2

These results show that the Robustness of the MMT protocol is high due to its route redundancy and low node dependence. It also shows the relatively low overhead involved due to its algorithm simplicity and non-flooding approach. It must be kept in

mind however that the effects of non-optimal routes do play a significant role in end-to-end delay and the number of hops involved. Hence this protocol should be ideally chosen for delay-tolerant applications in which route robustness is an important factor. We also discussed the advantage of cluster size configurability in the MMT protocol.

Chapter 6

Conclusions and Future Work

It has been the focus of this work to explore the current challenges that MANET Routing Protocols face, and propose a novel algorithm to address some of these challenges. We explained the detailed operation of the Multi-Meshed Tree based Routing protocol and discussed its main characteristics and how it attempts to solve the issues associated with dynamic environments through redundant routes and reduced route dependency. The unique aspects of the algorithm are outlined here:

1. It uses a single algorithm to perform clustering and proactive routing; reactive routes are concatenation of the proactive routes.

2. Reactive routes use a loose source routing concept and are impacted by only the local changes in a cluster, (which are resolved local to the cluster) and do not depend on the dynamics of all the nodes that are forwarding.

3. The scheme supports the maintenance of multiple proactive routes, which is uncommon in proactive routing as algorithms used until now determine and maintain one best route and on the failure of this route (which is known through topology dissemination), recalculate another.

4. Unification of tree and mesh topologies is new and helps leverage the advantages of the two.

5. Multi hop multiple overlapped cluster formation is achieved in a simple way with low overhead. Proactive routes are setup as clusters are formed i.e. both functions are achieved in one operation.

Cluster clients do not maintain routing tables and states, except when communicating with nodes outside their cluster.

6. No complex algorithms are involved; the simple VID scheme is novel as it facilitates all the above.

Through Simulation results and comparative analysis with benchmark MANET Protocols, certain advantages of the Multi-Meshed tree routing protocol have been demonstarted such as its low overhead and robust connectivity.

We hope this work is used a base for understanding the advantages of the MMT protocol, its unique implementation challenges, and its outline performance analysis. There is scope for Future Work on more advanced Simulation analysis, including effects of non-optimality of routes. There is also scope for work on a better understanding of the effects of Cluster-head election algorithms on the MMT Protocol

We foresee the increased use of MANETS is our daily lives and we feel there is a large untapped potential for these networks in civilian applications as well. The outcomes of this project can impact several critical civilian MANET applications such as disaster recovery networks, industry control networks, using of mobile units in a locality to provide a backbone network facility, all of which are currently constrained due to the connectivity and reliable data delivery. The outcomes of this study will clearly help identify civilian MANET application, which desire certain performance thresholds in terms of connectivity and reliable data delivery. Due to the low complexity of the proposed scheme the solution should be acceptable to the industry and vendor community.

References

[1] Nirmala Shenoy, Yin Pan, Darren Narayan, David Ross, and Carl Lutzer. Mobile ad hoc and sensor systems 2008. In *MASS 2008 5th IEEE International Conference*, 2008. Atlanta, GA.

[2] Pudlewski S., Shenoy N., Al-Mousa Y., Yin Pan, and Fischer J. Route robustness of a multi-meshed tree routing scheme for internet manets. In *IEEE GLOBECOM 2005*, 2005. Atlanta, GA.

[3] Nirmala Shenoy and Yin Pan. Multi meshed tree routing for internet manet's. In *The 18th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '99*, 2005. Atlanta, GA.

[4] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. Cedar: a core-extraction distributed ad hoc routing algorithm. In *2nd International Symposium on Wireless Communication Systems 2005*, 1999. New York, NY.

[5] Chai Keong Toh. A novel distributed routing protocol to support ad hoc mobile computing. In *IEEE 15th Annual International Phoenix Conference on Computers and Communications*, 1996. Phoenix, AZ.

[6] Basagni Stefano, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (dream). In *International Conference on Mobile Computing and Networking Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, 1998. New York, NY.

[7] CHING-CHUAN CHIANG, HSIAO-KUANG WU, WINSTON LIU, and MARIO GERLA. Routing in clustered multihop, mobile wireless networks with fading channel. In *IEEE Singapore International Conference on Networks*, 1997. Singapore.

[8] C.E. Perkins, E.M. Royer, S.R. Das, and M.K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Personal Communications, IEEE Feb 2001*, 2001. Singapore.

[9] Rajiv Misra C. R. Mandal. Performance comparison of aodv/dsr on-demand routing protocols for ad hoc networks. In *in Constrained Situation, Proceedings of 7th IEEE International Conference on Personal Wireless Communications (ICPWC-2005*, pages 86–89.

[10] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc on-demand distance vector (aodv) routing. `http://www.ietf.org/rfc/rfc3561.txt`.

[11] Krishna Ramachandran. Aodv-st. technical report, university of california, santa barbara, usa. http://www.cs.ucsb.edu/krishna/aodv-st/.

[12] M. JIANG, J LI, and Y. C. TAY. Cluster based routing protocol. http://tools.ietf.org/html/draft-ietf-manet-cbrp-spec.

[13] PHILIPPE JACQUET, PAUL MUHLETHALER, AMIR QAYYUM, ANIS LAOUITI, LAURENT VIENNOT, and THOMAS CLAUSEN. Optimized link state routing protocol. ,http://www.olsr.net/.

[14] MARIO GERLA, GUANGYU PEI, XIAOYAN HONG, and TSU-WEI CHEN. Fisheye state routing protocol (fsr) for ad hoc networks. ,http://tools.ietf.org/html/draft-ietf-manet-fsr.

[15] ZYGMUNT J. HAAS, MARC R. PEARLMAN, and PRINCE SAMAR. The zone routing protocol (zrp) for ad hoc networks. ,http://tools.ietf.org/html/draft-ietf-manet-zone-zrp.

[16] Guenael Strutt. Hwmp specification update. the working group for wlan standards of the institute of electrical and electronics engineers. 14 november 2006. ,https://mentor.ieee.org/.../11-06-1778-01-000s-hwmp-specification.doc.

[17] L Blazevic. A location-based routing method for mobile ad hoc networks. ,http://icawww1.epfl.ch/TNRouting/.

[18] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination- sequenced distance-vector routing (dsdv) for mobile computers. In *ACM SIGCOMM94 Conference on Communications Architectures, Protocols and Applications*, 1994.

[19] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. In *Mobile Computing, T. Imielinski and H. Korth, Eds., Kluwer, 1996*, 1996.

[20] Geetha Jayakumar and G. Gopinath. Ad hoc mobile wireless networks routing protocols a review. Journal of Computer science, 2007.

[21] Shima Mohseni, Rosilah Hassan, Ahmed Patel, and Rozilawati Razali. A comparative review study of reactive and proactive routing protocols in manets. 4th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2010).

[22] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. IEEE Personal Communications April 1999.

[23] Qifa Ke, David A. Maltz, and David B. Johnson. Emulation of multi-hop wirless ad hoc networks. Proceedings of the 7th International Workshop on Mobile Multimedia Communications (MoMuC 2000).

[24] C.-C. Chiang. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proc. IEEE SICON 97*, 1997.

[25] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. In *ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks*, 1996.

[26] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. INFOCOM 97*, 1997.

[27] Z.J. Haas and S. Tabrizi. On some challenges and design choices in ad-hoc communications. In *Military Communications Conference, 1998. MILCOM 98. Proceedings., IEEE , vol.1,*, 1998.

[28] R. Ramanathan and J. Redi. A brief overview of ad hoc networks: challenges and directions. In *Communications Magazine, IEEE , vol.40,*, 2002.

[29] Bandyopadhyay S. and E.J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE , vol.3*, 2003.

[30] Pudlewski S., Shenoy N., and Al Mousa Y. A hybrid multi meshed tree routing protocol wireless ad hoc networks. In *Second IEEE International Workshop on Enabling Technologies and Standards for Wireless Mesh Networking*, 2008.

[31] Qin L. and Kunz T. Technical report systems and computer engineering. In *Survey on Mobile Ad Hoc Network Routing Protocols and Cross-Layer Design*, 2004.

[32] QAbolhasan M., Wysocki T., and Dutkiewicz E. A review of routing protocols for mobile ad hoc networks. In *Journal of ad hoc networks*, 2004.

[33] Daniel L. Technical report, department of computer science, technische universitat, munchen, germany. In *A comprehensive overview about selected Ad hoc networking routing protocols*, 2004.

[34] Royer E. M. and C.K. Toh. Ieee personal communications magazine. In *A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks*, 1999.

[35] S. Basagni. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. In *Vehicular Technology Conference, 1999. VTC 1999 - Fall. IEEE VTS 50th , vol.2,*, 1999.

[36] C.R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks,. In *Selected Areas in Communications, IEEE Journal on , vol.15, no.7,*, 1997.

[37] Chatterjee M., Das S., and Turgut D. Wca: A weighted clustering algorithm for mobile ad hoc networks. In *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks), vol. 5*, 2002.

[38] Basagni S., Chlamtac I., and Farago A. Workshop on algorithmic aspects of communication. In *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks), vol. 5*, 1997.

[39] Chen G. and Stojmenovic I. Clustering and routing in mobile wireless networks. In *Technical Report TR- 99-05, SITE, June 1999*, 1999.

[40] Chen Y., Liestman A., and Liu J. Clustering algorithms for ad hoc wireless networks. In *Ad Hoc and Sensor Networks*, 2004.

[41] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies , vol.1,*, 2004.

[42] Belding-Royer E. M. Multi-level hierarchies for scalable ad hoc routing. In *Wireless Networking (WINET), Vol. 9, No. 5,*, 2003.

[43] Ramasubramanian V., Haas Z. J., and Emin G un Sirer. Sharp: A hybrid adaptive routing protocol for mobile ad hoc networks. In *MobiHoc 03 Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and Computing.*

[44] Chuan ming Liu and Chuan hsiu Lee. Conf. on wireless networks (icwn'05) 405 distributed algorithms for energy-efficient cluster-head election in wireless mobile sensor networks.

[45] Li J. et al. A scalable location service for geographic ad hoc routing. In *ACM Mobicom , Boston, MA.*, 2000.

[46] Pei G. and M. Gerla. Mobility management in hierarchical multi-hop mobile wireless networks. In *Proceedings of IEEE ICCCN99, Boston, MA,*, 1999.

[47] Pei G., M. Gerla, X. Hong, and C. C. Chiang. A wireless hierarchical routing protocol with group mobility. In *Proceedings of IEEE WCNC99, New Orleans, LA,*, 1999.

[48] Chiang C. and M. Gerla. Routing and multicast in multihop, mobile wireless networks. In *IEEE ICUPC97, San Diego, CA,*, 1997.

[49] Das S.R., C.E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *IEEE INFOCOM 2000, Tel Aviv, Israel*, 2000.

[50] Hong X., Kaixin Xu, and Mario Gerla. Scalable routing protocols for mobile ad hoc networks. In *IEEE Network Journal, July/Aug 2002, Vol 16, issue 4,*, 2002.

[51] Vincent D. Park and M. Scott Corson. A performance comparision of tora and ideal link state routing. In *IEEE Symposium on Computers and Communication,*, 1998.

[52] Vincent D. Park and M. Scott Corson. Proceedings of infocom97. In *A highly adaptive distributed routing algorithm for mobile wireless networks*, 1997.

[53] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

[54] Bernd Freisleben and Ralph Jansen. Analysis of routing protocols for ad hoc networks of mobile comuters. In *Proceedings of the 15th IASTED International Conference on Applied Informatics,*, 1997.

[55] Marc Greis. Tutorial for the network simulator ns. http://www.isi.edu/nsnam/ns/tutorial/index.html.

[56] Kevin Fall and Kannan Varadhan. The ns manual, the vint project. http://www.isi.edu/nsnam/ns/ns-documentation.html.

[57] Kevin Fall. Network emulation in the vint/ns simulator. Proceedings of the Fourth IEEE Symposium on Computers and Communications (ISCC99), July 1999.