Assessing the Effect of Packet Size on Proactive Mobile Ad Hoc Network Routing Protocols

A Thesis Submitted to the School of

Electrical and Computer Engineering

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In Computer Engineering Addis Ababa Institute of Technology

Addis Ababa, Ethiopia

By

Iyasu Daniel

October 2014

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

COMPUTER ENGINEERING STREAM

Assessing the Effect of Packet Size on Proactive Mobile Ad Hoc Network Routing Protocols

By

Iyasu Daniel

Advisor: Dr. Yalemzewd Negash

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

Assessing the Effect of Packet Size on Proactive Mobile Ad Hoc Network Routing Protocols

By

Iyasu Daniel

APROVAL BY BOARD EXAMINERS

Prof.Kwon Ho Yeol _____

Head of the Dept. of Electrical Signature

and Computer Engineering

Dr.YalemzewdNegash____ _____

Advisor Signature

_____ _____

External Examiner Signature

_____ _____

Internal Examiner Signature

Declaration

I, the undersigned, declare that this thesis work, to the best of my knowledge and belief, is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

NAME: Iyasu Daniel Signature: _____

Place: Addis Ababa

Date of submission: _____

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Yalemzewd Negash Signature: _____

Advisor Name

Acknowledgement

In First place I would like to thank God for guiding me in every stage of my life.

I am greatful for Aksum University for sponsoring my exepenses while working my Thesis

I would like to thank Dr.Yalemzewed Negash for his guidance in this thesis work and introroducing me to research world.Dr.Sirnavas Nunne for his encouraging advises during the work

Finally on Personal basis I would like to thank Ato Daniel Geleta, W/ro Altayech Seyoum and Lidya Daniel for your moral support.

A special word of thanks goes to Lina Felleke for her encouraging me to accomplish my work

Tabel of Content

List of Figures

List of Acronym

AODV	Ad Hoc On-demand Distance Vector Routing
AP	Access Point
BS	Base Station
BSC	Base Switching Center
DSDV	Destination Sequenced Distance Vector Routing
DSR	Dynamic Source Routing
IDE	Integrated Development Enviroment
IEEE	Institiute of Electrical and Electronic Engineers
IP	Internet Protocol
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
MID	Multiple Interface Declaration
MIG	Mobile Inetrnet Gateway
MPR	Multi Point Relaying
MS	Mobile Station
MSC	Mobile Switching Center
NED	Network Description
OLSR	Optimized Link State Routing
OMNET++	Objective Modular Network Testbed in C++
PDR	Packet Delivery Ratio
QoS	Quality of Service

RFC	Request for Commnet
RIP	Routing Information Protocol
TC	Topology Control
TCP	Transmission Control Protocol
TIB	Topology Information Base
UDP	User datagram Protocol
VANET	Vehicular Ad-Ho Network
WLAN	Wireless local area network

Abstract

The existence of different kinds of wireless technologies enabled formation of two types of Mobile Ad hoc Networks (MANET). Homogeneous MANETwhere communication is entirely infrastructurless and Heterogeneous MANET where the network is composed of infrastructured and non-infrastructured Wireless Local Area Networks (WLANs). MANET use particular routing protocols designed to cope with the absence of central coordinator to route packets and can be classified as proactive and reactive.The control messages generated by each routing protocol vary which affects the size of the payload in a single transmission.

This Masters Thesisimplements two proactive routing protocols in homogeneous and heterogeneous environment and assesses the effect of packet size on the performance of the two environments and uses the reserved bits to signal that a node can be used as a gate way From the tests we showed the performance degradation for Destination Sequenced Distance Vector(DSDV) routing protocol interms of throughput is reduced about (8.7% - 10.5%) and for Optimized Link State Routing(OLSR) protocol (3.51% - 4.08%) as compared to to Ad hoc On Demand Distance Vector(AODV) routing protocol .Considering PDR as a performance metrics for DSDV the reduction is about (11.7% - 28.6%) and for OLSR(6.1% -10.4%) as compared to AODV.The environments have been developed using omnet++ simulator.

The findings show network performances increase as the packet size increase .In addition the presence of multigateway has positive correlation with network performance.

Keywords: DSDV, Packetsize, gateway, Proactive MANET, HetrogeneousMANET, OLSR, Throughput, PDR, OLSR

Chapter 1

Introduction

Advances in computing and communication technologies resulted in pervasive computing environment where future mobile communication will include both infrastructured wireless and Mobile ad hoc networks. A MANET is a collection of wireless nodes that can dynamically form a network to exchange information without existing fixed network infrastructures. The absence of fixed network infrastructure brings these technologies a great opportunities with severe challenges .This can be routing due to the dynamic nature of the environment and lead to the existence of different routing protocols to cope with this dynamic environment, Inter networking where coexistence of routing protocols of various environment is a challenge for the harmonious mobility management .Moreover,the amount of data in a packet and the composition of the header can change depending on the communications protocol and this networks use a store and forward mechanism the impact of the packet size must be carefully analyzed.

In this thesis we

- Observe the performance of proactive routing protocol degrades in heterogeneous environment as compared to the homogeneous environment
- Proposed a way of connecting mobile ad hoc networks with infrastructured wireless network and our method improved performance

1.1 Back ground

In 2020 it is estimated the number of interconnected devices will be more than 6-7 times than the number of people. This shows that the wireless arena has been experiencing exponential growth in the past decades and lead to advances in computing and communication technologies. This has created a pervasive computing environment and stared to change the way we live .To mention only a few examples Mobile users can rely on their mobile devices to check e-mail and browse the internet from different public locations and files or other information can be exchanged by connecting portable computers via wireless LANs while attending conferences.

Most of the connections among wireless devices occur over fixed-infrastructure-based service providersor private networks; for example, connections between two cell phones set up by BSC andMSC in cellular networks, or laptops connected to the Internet via wireless access points.

Although infrastructure-based networks provide a great way for mobile devices to get network Services, it takes time to set up the infrastructure network, and the costs associated with installing infrastructure can be quite high. There are, furthermore, situations in which user-required infrastructure is not available, can not be installed, or can not be installed in time in a given geographic area. Providing the needed connectivity and network services in these situations requires a MANET.

To perform all this operation in the absence of infrastructures different routing algorithms have been proposed and broadly categorized in to proactive and on demand routing protocols. The on-demand routing algorithms initiate to find out the suitable route when a route is requested the pro-active routing algorithm exchanges routing information periodically and generates the routing table inadvance of route request [1]

In proactive routing, each node has one or more tables that contain the latest information of the routes to any node in the network. Each node has the next hop for reaching to a node/subnet and the cost of this route. Various table-driven protocols

differ in the way the information about change in topology is propagated through all nodes in the network.

Based up on the infrastructures used we can categorize MANET as homogeneous MANET and a heterogeneous MANET.In homogeneous MANET there is an absence of Acess Point (AP) where as in a heterogeneous MANET there exists an AP where the nodes will have access to other kinds of wireless technology.

In this thesis, the effect of packet size on existing proactive routing protocols is studied in a Heterogeneous environment. Where communication is between MANET and wireless LAN [2].This type of environment needs to be considered due to their vital role in internetwork operations.

1.2 Problem definition

T he demands of users nowadays make it difficult to find a closed MANET since users needs to get connected to other networks to satisfy their daily needs like Internet banking, online booking, e-marketing and others which could be non MANET.

Many researchers propose a number of routing protocols but so far not all problems are solved due to the nature of the environment and the solutions are limited to closed MANET thus, it is important to re-evaluate the existing protocols under different scenarios.

Network performance is hugely affected by the size of the packet, especially in a heterogeneous environment where the network is made of both infrastructure and adhoc WLAN and researches have shown the performance of reactive protocols degrades in a heterogeneous environment hence we need to re evaluate the proactive categories for such kinds of environment. More over, architecture of Integration contributes to the degradation in performance of mobile adhoc networks.

1.3 Objectives

General Objective

The general objective of this thesis it to assess the effect of packet size on DSDV and OLSR routing algorithms in a heterogeneous environment

Specific objectives:

The specific objectives of this thesis include:

- To make experiment about the proactive routing protocols of MANET and their metrics response towards packet size.

- Examining the effect in homogeneous and heterogeneous environment
- Comparing the performance of the improved routing protocols with the reactive counterpart

1.4 Scope

As the number of nodes in the MANET increases, the size of the table will increase; this can become a problem in and of itself. In addition, it needs control traffic for continually update stale route entries. Thus, makes proactive routing algorithm less efficient and this thesis is implemented in such routing protocols.

The outcome of this research shall be based on the simulation of the proposed techniques using a simulation tool (OMNeT++).

1.5 Document Organization

The rest of this document is organized as follows the second chapter surveys related researchesin Mobile Adhoc networks.Chapter three describes in detail the design implementation and the simulation software used.Finally concluding remarks are madewith future work.

Chapter 2

Literature Review

Before the deployment of Mobile Ad hoc network technologies major challenges need to be solved this can be justified with the lack of killer application [6].Particularly the fundamental research issue is focuses on packet routing due to the dynamic topology of MANETs [6].

Among main focus of research on routing protocols in MANETs has been network performance.A number of routing protocols have been proposed to provide multi-hop communication in wireless Ad Hoc networks [2] [7] [8] [9] [10] [11] [12] [13] in each studies various parameters where used. However. Author *Aleksi Penttinen* points Active research has produceda wide range of proposals, but so far not many problems are not solved.

In this chapter we will review researches focusing on performance evaluation of routing protocols, issues and techniques of integrating MANETs with other networking technologies.

There exists a number of proactive routing protocols among this we assess the effect of packet size on DSDV and OLSR.

Destination Sequenced Distance vector Routing Protocols: [3] Destination sequenced distance vector routing (DSDV) is adapted from the conventional Routing Information Protocol (RIP) to ad hoc networks routing. It adds a new attribute, sequence number, to each route table entry of the conventional RIP. Using the newly added sequence number, the mobile nodes can distinguish stale route information from the new and thus prevent the formation of routing loops.

In DSDV, each mobile node of an ad hoc network maintains a routing table, which lists all available destinations, the metric and next hop to each destination and a sequence number generated by the destination node. Using such routing table stored in each mobile node, the packets are transmitted between the nodes of an ad hoc network. Each node of the ad hoc network updates the routing table with advertisement periodically or when significant new information is available to

maintain the consistency of the routing table with the dynamically changing topology of the ad hoc network.

Periodically or immediately when network topology changes are detected, each mobile node advertises routing information using broadcasting or multicasting a routing table update packet. The update packet starts out with a metric of one to direct connected nodes. This indicates that each receiving neighbor is one metric (hop) away from the node. It is different from that of the conventional routing algorithms. After receiving the update packet, the neighbors update their routing table with incrementing the metric by one and retransmit the update packet to the corresponding neighbors of each of them. The process will be repeated until all the nodes in the ad hoc network have received a copy of the update packet with a corresponding metric. The update data is also kept for a while to wait for the arrival of the best route for each particular destination node in each node before updating its routing table and retransmitting the update packet. If a node receives multiple update packets for a same destination during the waiting time period, the routes with more recent sequence numbers are always preferred as the basis for packet forwarding decisions, but the routing information is not necessarily advertised immediately, if only the sequence numbers have been changed. If the update packets have the same sequence number with the same node, the update packet with the smallest metric will be used and the existing route will be discarded or stored as a less preferable route. In this case, the update packet will be propagated with the sequence number to all mobile nodes in the ad hoc network. The advertisements of routes that are about to change may be delayed until the best routes have been found. Delaying the advertisement of possibly unstable route can damp the fluctuations of the routing table and reduce the number of rebroadcasts of possible route entries that arrive with the same sequence number.

The elements in the routing table of each mobile node change dynamically to keep consistency with dynamically changing topology of an ad hoc network. To reach this consistency, the routing information advertisement must be frequent or quick enough to ensure that each mobile node can almost always locate all the other mobile nodes in the dynamic ad hoc network. Upon the updated routing information, each node has to relay data packet to other nodes upon request in the dynamically created ad hoc network.

Optimized Link state Routing Protocol :[4]The optimized link state routing protocol is a table driven proactive protocol that uses the link state scheme in an optimized manner to diffuse topology information. In a classic link-state algorithm, link-state information is flooded throughout the network.OLSR uses this approach as well, but since the protocol runs in wireless multi-hop scenarios the messageflooding in OLSR is optimized to preserve bandwidth. The optimization is based on a technique calledMultiPoint Relaying.

Being a table-driven protocol, OLSR operation mainly consists of updating and maintaining informationin a variety of tables. The data in these tables is based on received control traffic, and control traffic is generated based on information retrieved from these tables. The route calculation itself is also driven bythe tables.

OLSR defines three basic types of control messages

HELLO - HELLO messages are transmitted to all neighbors. These messages are used for neighborsensing and MPR calculation.

TC - Topology Control messages are the link state signaling done by OLSR. This messaging is optimizedin several ways using MPRs.

MID - Multiple Interface Declaration messages are transmitted by nodes running OLSR on more thanone interface. These messages list all IP addresses used by a node.

[5]WLANs can operate in infrastructure (single-hop) mode, where connectivity is provided by an AP, or in MANET mode, where devices can communicate with each other through multi-hop routing. These two operation modes have different characteristics and particular issues that need to be addressed separately in an integrated architecture. In the hotspot area, multiple APs may overlap to some extent; also, a BS and anAP may be collocated. MSs can arbitrarily move, and at a given instant a particular MS can either be within or outside the coverage of BSs and/or APs. A connection from an MS to a BS/AP canbe established by a single hop or using multihop when the MS is out of the coverage of the corresponding BS/AP, as shown in Fig. 1. Factors influencing the design of such a heterogeneous architecture include multi-interface MSs, trans-mission power and co-channel interference,

topology and routing, mobility and handoff, load balance, interoperability, and quality of service(QoS) provisioning.

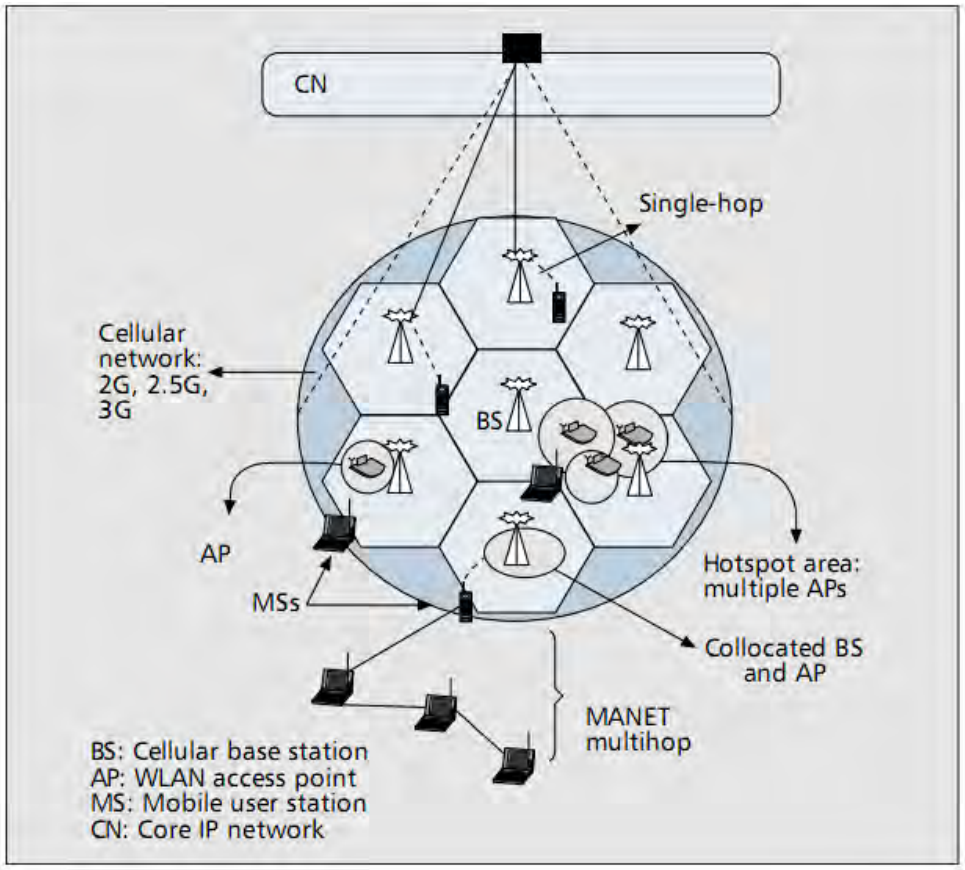


Figure 1 Hetrogeneous network"Hetrogeneous network Architecture" IEEE Wireless Communication June2005

After reviewing the operation of the two routing protocols in the subsequent section we will review the response as well as the comparison of this two routing protocols with other types of routing protocols.

Performance evaluation of MANETS: [14] evaluates the performance of protocols interms of energy efficiency and also [15] compares routing protocols interms of number of nodes, node speed and simulation time. The paper [16] concludes a more reliable and efficient routing protocol is necessary in a mobile ad hoc network which considers mobility and longevity of a route as its main concern. Thus, we can conclude that there is no well suited routing protocol for every network condition.

[2] Evaluates the effects of packet size on AODV routing protocol and implements in homogeneous and heterogeneous MANET enviromnt. And shows that the increase of

through put and PDR performance was parallel with the increase of packet size in addition the performance due to the packet size effect in homogeneous MANET is better than in heterogeneous MANET. In the simulation, the size of the packets transmit has a large impact on throughput and PDR in wireless environment and shows shows that the performance of the AODV routing protocol in homogeneous MANET is better compared to heterogeneous MANET this has an implication that the existing AODV routing protocol needs an optimal enhancement to cope with such environment.

[10] Points MANETs are a wireless network in which packets are transmitted in store and forward manner from source to destination. When processing the packets from one node to another node packet size is very important parameter because changing the packet size effect the performance of the MANET network and uses AODV, OLSR & DSR routing protocols in homogeneous environment.It concludes even though AODV have higher throughput for the same packet size than OLSR when we compare OLSR interms of delay it shows superior performance for real time applications.

[7] Studies three adhoc routing protocols STAR, DSR and AODV based on IEEE 802.11 and evaluates the performance by varying the packet size and uses throughput, End to End delay as performance metrics and concludes DSR is more suitable protocol for routing purposes in MANET.From this we can see that there is a variation in performance of routing protocols.On one end reactive protocols show great performance and on the otherhand source initiated routing protocols have better performance thus, its better to see the proactive routing protocols.

Comparison of reactive and proactive routing protocols:[17]Varying parameters of VANET shows that in the real traffic scenarios proactive protocol performs more efficiently for energy conservation. On the other hand [18] DSR/AODV performs better than DSDV with large number of nodes. Hence for real time traffic AODV is preferred over DSR and DSDV. For less number of nodes and less mobility, DSDV's performance is superior [2]. This also shows each category of routing protocols have different behavior for different enviromnet.

Integration with other types of networks: [19] presents an extended DSDV protocol, named as Efficient DSDV (Eff-DSDV) protocol, has been used to provide

bidirectional connectivity between ad hoc nodes and the hosts in the Infrastructure-based networks. The proposed framework uses one of the ad hoc hosts known as Mobile Internet Gateway (MIG) to act as a bridge between the two networks.

[20] Propose and evaluate a practical architecture to build multi-hop hybrid adhoc networks used to extend the coverage of traditional wired LANs, providing mobility support for mobile/portablesdevicesinthe local area environment. And points out a light weight solution is needed for the practical realization of this type of networks.[21] points The key challenge in providing connectivity is to minimize the overhead of mobile IP and Ad Hoc routing protocol between Internet and Ad Hoc networks.

Chapter 3

Simulation Software and Design Implementation

The simulation software used in this research work is OMNet++. OMNeT++ is an object-oriented modular discrete event network simulation framework. To simulate the mobile wireless radio environment, this research has used inet framework which is Written for the OMNEST/OMNeT++ simulation system. The INET framework contains models for several Internet protocols. This chapter discusses the simulation software used and the experimental design.

3.1 OMNeT++

[22] OMNeT++ refers to Objective modular network test bed in C++ which is an object-oriented modular discrete event simulation framework. In itself it is not a simulator instead it provides infrastructure and tools for writing simulations. Specific problem domains are modeled using specialized component libraries built on top of the OMNeT++ simulation framework. As an example it can be used in Modeling of wired and wireless communication networks Protocol modeling.

Models are assembled from reusable components termed as modules and can be assembled to build a complex system each module communicates with each other with message passing. The top level module is called the system module which contains sub modules that can also contain sub modules themselves. Depth of module nesting is unlimited, allowing the user to reflect the logical structure of the actual system in the model structure. The active modules are termed simple modules and are written in C++ using the simulation class library Simple modules can be grouped into compound modules and so forth; the number of hierarchies is unlimited.

An OMNeT++ model consists of the following parts:

- NED language topology description(s)(.ned files) : - that describe the module structure with parameters,gates,etc.NED files can be written using any text editor,but the OMNeT++ IDE provides excellent support for two-way graphical and text editing.

- Message dentitions (.msg files): - We can define various message types and add data fields to them.OMNeT++ will translate message definitions in to full-fledged C++classes.
- Simplemodule sources: - bConsists of C++files, with .h/.cc suffix.

The simulation system provides the following components:

- Simulationkernel.This contains the code that manages the simulation and the simulation class library. It is written in C++, compiled in to a shared or static library.
- Userinterfaces.OMNeT++ user interfaces are used in simulation execution, to facilitate debugging, demonstration, or batch execution of simulations.They is written in C++, compiled in to libraries.

Simulation programs are built from the above components.First, .msg files are translated in to C++ code using the opp_msgc.program.Then all C++ sources are compiled and linked with the simulation kernel and a user interface library to form a simulation executable orshared library.NED files are loaded dynamically in their original text forms when the simulation program starts.

The simulation may be compiled as a standalone program executable; thus it can be run on other machines without OMNeT++ being present, or it can be created as a shared library. In This case the OMNeT++ shared libraries must be present on that system.When the program Is started, it first reads all NED files containing your model topology, then it reads a configuration file (usually called omnetpp.ini).This file contains settings that control how the simulation is executed, values for model parameters, etc.The configuration file can also prescribes everal simulation runs; in the simplest case, they will be executed by the simulation program one afteranother.

The output of the simulation is written in to result files: outpu tvectorfiles, output scalar files, and possibly the user's own output files.OMNeT++ contains an IntegratedDevelopment Environment (IDE) that provides rich environment for analyzing these files.Output files are line-oriented text files which makes it possible to process them with a variety of tools and programming languages as well, includingMatlab, GNUR, Perl, Python, and spread sheet programs.

3.2 System Modeling

The systems in the models are built from the following modules and in this section we will describe the components .

The NED Language

The NED language describes the structure of the simulation of model.NED enables the user to create simple modules, and connect and assemble them in to compound modules.The user can label some compound modules as networks; that is,self-containedsimulationmodels. As shown below:

Figure 2: The network Model

The above network consists of different modules and is made of different sub modules and we can create different types of networks from this network depending on the configuration.

Simple modules are the basic building blocks for other (compound) modules, denoted by the simple keyword. All active behavior in the model is encapsulated in simple modules. Behavior is defined with a C++ class; NED files only declare the externally visible interface of the module (gates, parameters).

In Figure 2 the functionalities of each component is complex thus it is better to implement it with several smaller simple module types which we are going to assemble in to a compound module.

In the subsequent sections we will see each component in detail

3.2.1 The Ad Hoc Host Compound Module

The adhoc Host is made of the following sub modules as shown below and its role is to act as a member of the mobile adhoc network .In addition Each adhoc Host have the capability of acting as a gate way.

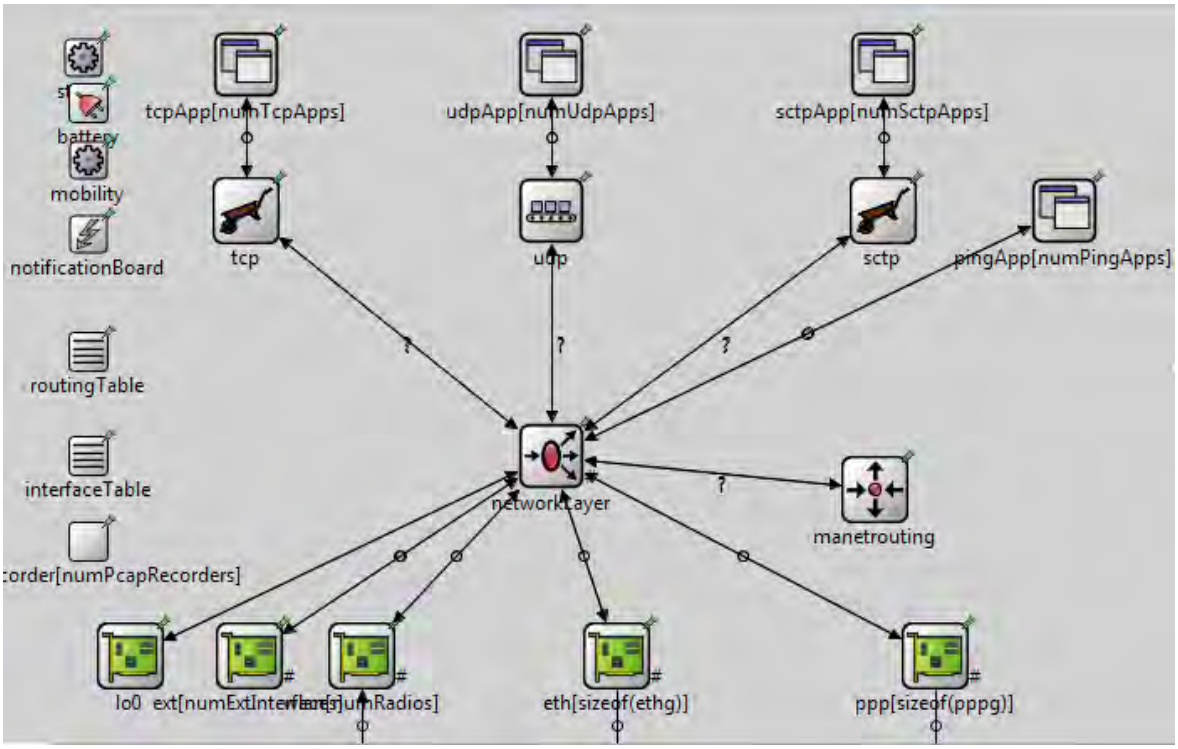


Figure 3: The Ad hoc Host Compound Module

Figure 3 shows the sub modules in ad hoc Host are organized as the layers of TCP/IP protocol stack and some sub modules by themselves are compound modules .The components on the left part of the figure are used to facilitate communication between sub modules as well as communication with other modules.

3.2.1.1Componenets of the Ad hoc Host Compound module

1. InterfaceTable (InterfaceTable):-This module contains the table of network interfaces in the host. Interfaces are registered dynamically during the initialIzation phase by modules tha trepresent network interface cards (NICs).Other modules can access interface information via a C++ class interface.

2. Routing Table:-This module contains the IPv4 routing table.It is also accessed from other via a C++ interface.The interface contains member functions for adding,

removing ,enumerating and looking up routes,and finding the best matching route for a given destination IP address.

3. Notification Board:-This module makes it possible for several modules to communicate in a publish-subscribe manner.Notifications include change notifications (“routing table changed”) and state changes (“radio became idle”).

4. Mobility module:-In simulations involving node mobility, this module is responsible formoving around the node in the simulated“play ground.”A mobility module is also neededfor wireless simulations even if the node is stationery,because the mobility module storesthe node’s location,needed to compute wireless transmissions.Different mobility models are supported via different module types,and many host models define their mobility sub modules with parametric type so that the mobility model can be changed in the configuration ("mobility: <mobility Type> like IMobility").

5. NICs :-Network interfaces are usually compound modules themselves,composed of a queue and a MAC module (and in the case of wireless NICs,a radio module or mod-ules).The queue sub module stores packets waiting for transmission,and it is usually defined as having parametric type as it has multiple implementations to accommodate different needs.

6. Networklayer:-Modules that represent protocols of the network layer are usuallygrouped in to a compound module: Network Layer contains the modules IP, ARP, ICMP and Error Handling.The IP module performs IP encapsulation /decapsulation and routing of datagrams;for the latter it accesses the C++ function call interface of the RoutingTable.Packet forwarding can be turned on/off via a module parameter.The ARP moduleisputintothepathofpackets leaving the network layer to wards the NICs,and performs address resolution for inter-faces that needit(e.g.Ethernet).ICMP deals with sending and receiving ICMP packets.The Error Handling module receives and logs ICMP error replies.

7. Transport layer protocols:-Transport protocols are represented by modules connected to the network layer. UDP is implemented by UDP module.

8. Applications:-Application modules typically connect to TCP and/ or UDP,and model the user behavior as well as the application programand application layer protocol .

9. Routing protocols:-Router models typically contain modules that implement routing Protocols.These modules are connected to the TCP and/or theUDP module, and manipulate routes in the RoutingTable module via C++ function calls.

3.2.2 The WirelessHost Compound Module

The wirelessHost is made of the following sub modules is shown below and its role is to act as a member of the wireless network which uses the access point to communicate with the adhoc hosts.

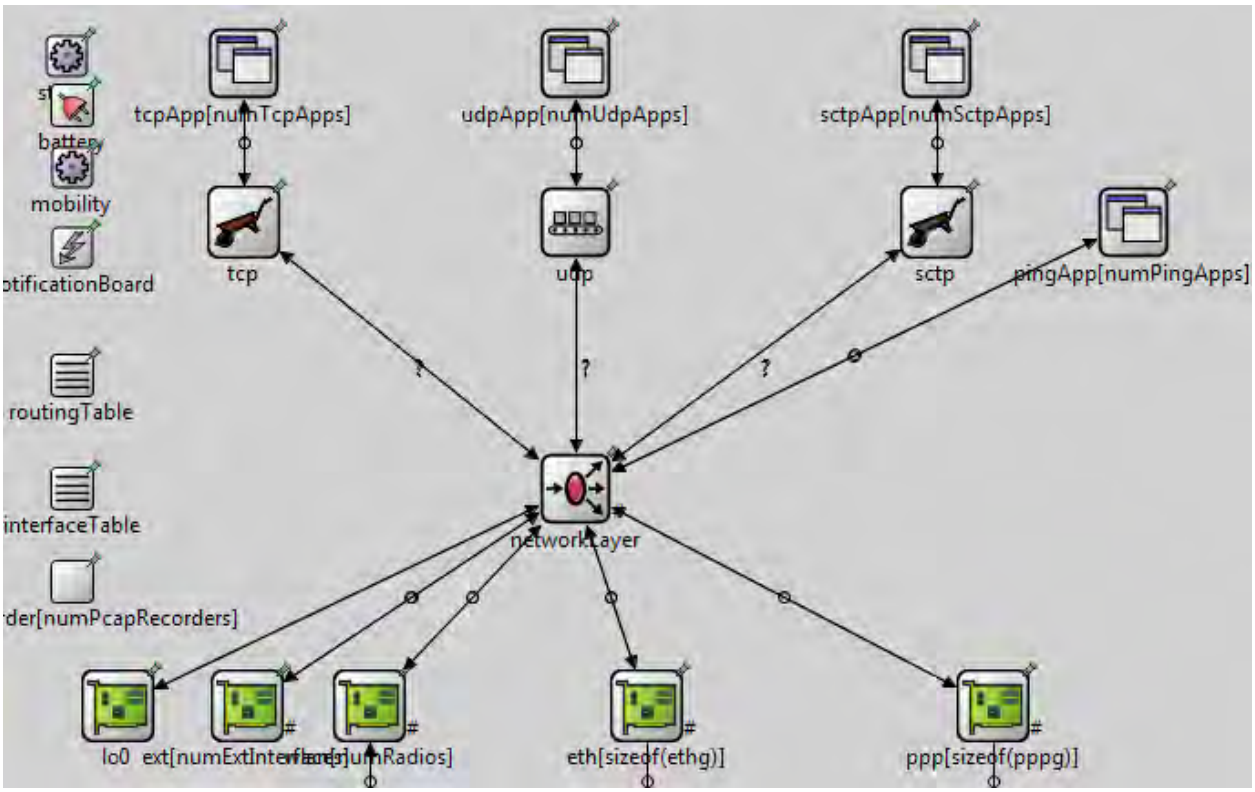


Figure 4:The wirelessHost compound module

Figure 4 can be shows the sub modules in wireless Host are organized as the layers of TCP/IP protocol stack and some sub modules by themselves are compound modules The components on the left part of the figure are used to facilitate communication between sub modules the difference with the adhocHost submodule it does not implement any mobile adhoc routing protocol.

3.2.3 The Access point Compound Module

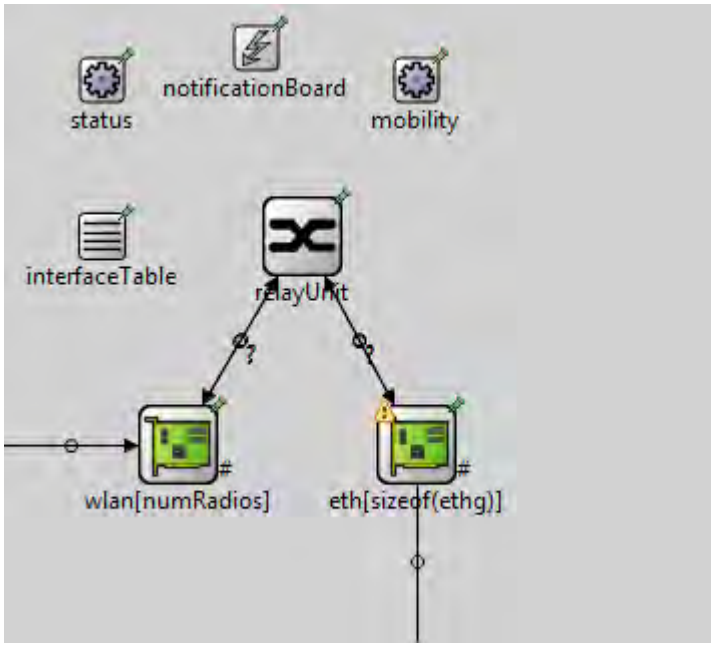


Figure 5: The access point compound module

Figure 5 depicts a generic access point supporting multiple wireless radios, and multiple Ethernet ports. The type of the Ethernet MAC, relay unit and wireless card can be specified as parameters. Ethernet (and possibly other) switch models may contain a relay unit, whichSwitches frame among Ethernet (and other IEEE802) NICs.

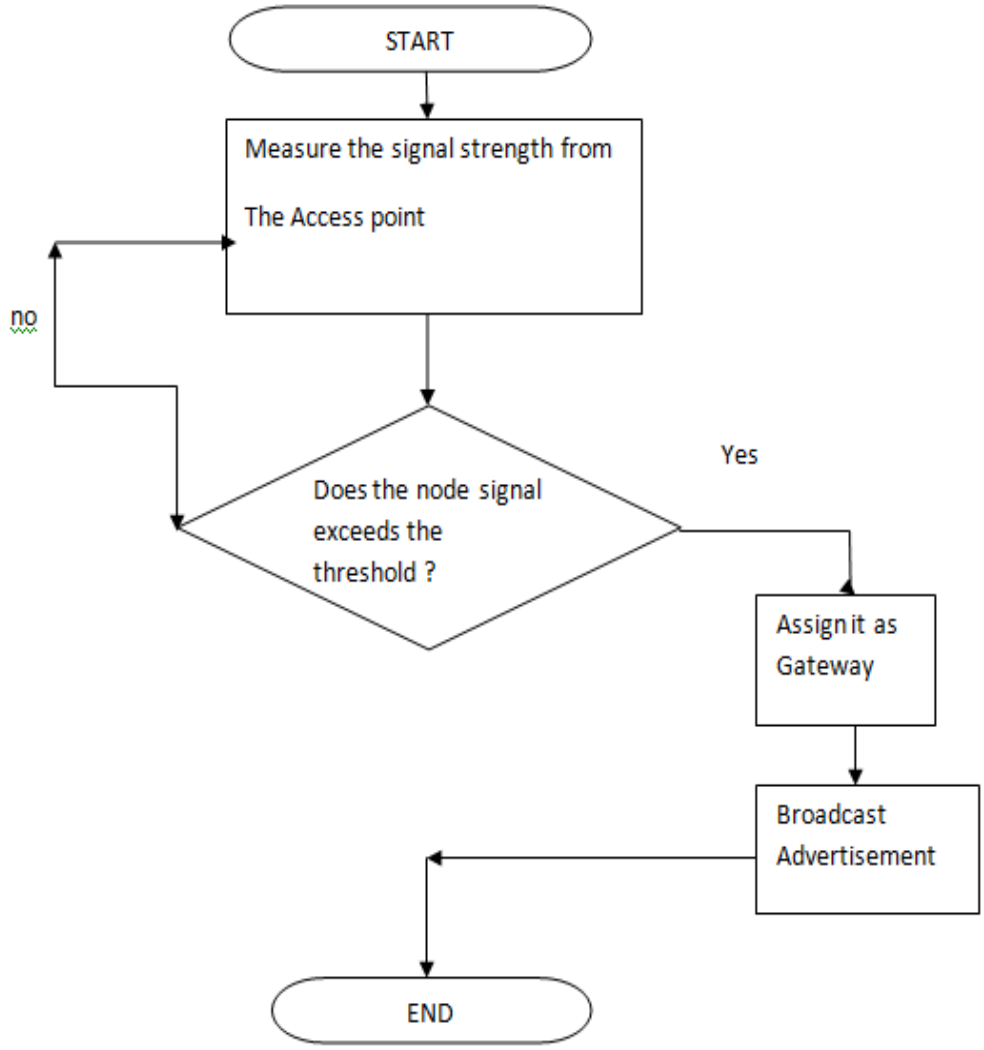
3.2.4 The configuration

This module assigns IP addresses. The configuration supports both manual and automatic address assignment, and their combinations. The configuration also does routing table optimization that significantly decreases the size of routing tables in large networks.

3.2.5 Channel control

Channel Control has exactly one instance in every network model that contains mobile or wireless nodes. This module gets informed about the location and movement of nodes, and determines which nodes are within communication or interference distance. This information is then used by the radio interfaces of nodes at transmissions.

The channel control simple module is modified in such a way that it can be used to indicate the mobile adhoc host that can be used as a gate way according to the flow chat shown below: .



Figue 6 Flow chart of Modified Channel control Simplemodule

Each host has a transmission power that determines the range ,with in which a communication isfeasible. The signal power degradation is modeled by the Free Space Propagation Model in which the receivedsignal power is inversely proportional to the node distancesquare ($1/ r^2$).

The mininum power transmission can be calculated using the formula

```
minReceivePower = pow(10.0, sat / 10.0)
```

The power control module keeps tracking the nodes and selects the node which is capable of acting as a gate way by setting the lags which are found on the headers of each routing protocol control packet

3.3 Simple module Implementation

Simple modules are the active components in the model.Simple modules are programmed in C++, using the OMNeT++ class library.Components are represented with the C++ classc Component.The abstract module class cModule and the abstract channel class cChannel both subclass from cComponent.cModule has two subclasses: cSimpleModule and cCompound Module.The user defines simple module types by subclassing cSimpleModulecComponent has the following member functions meant for redefining in subclasses: initialize().This method is invoked after OMNeT++ has setup the network (i.e. created modules and connected them according to the dentitions),and provides a place for initialization code;finish() is called when the simulation has terminated successfully,and its recommended use is the recording of summary statistics.

To define the dynamic behavior of a simple module, you need to redefine one of the following Member functions: handle Message (cMessage *msg).It is invoked with the message as parameter when ever the module receives a message. HandleMessage () is expected to process the message, and then return.Simulation time never elapses inside handleMessage () calls, only between them.

Models of protocol layers in a communication network tend to have a common structure on a High level because fundamentally they all have to react to three types of events :to messages Arriving from higher layer protocols(or apps),to messages arriving from lower layer protocols(From the network), and to various timers and timeouts (that is,self-messages).

Using the above mentioned classes and function the two routing protocols have been implemented. The full souce code found at the Appendix.

The packet fomat modification is made in the.msg extension files and discussed in the following two subsections.

3.3.1 DSDV Routing protocol Packet Modification

Every mobile node in the network maintains a routing table in which all of the possible destinations with in the network and the number of hops to each destination are recorded. Each entry is marked with asequence number assigned by the destination node. Thesequence numbers enable the mobile nodes to distinguishstale routes from newer ones, avoiding the formation of routing loops.

Absence of standardized RFC enables us to modify the structure and change the utilization of the bits as shown below:

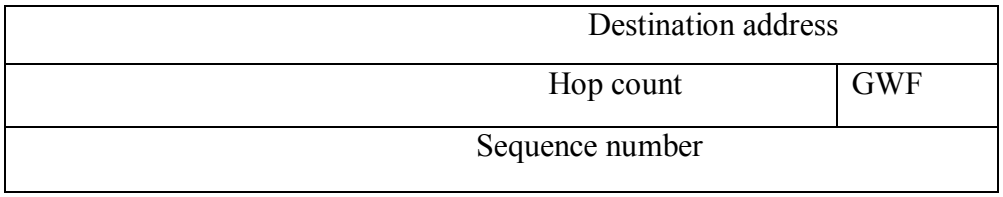


Figure 7 Modified packet headers for DSDV

The DSDV header is a 32 bit wide with the total header size of 12 bytes an shown in Figure 7.Destination address holds the Ip address of the destination. We have Modified the hop count field so that it can be used to signal the node can act as a gateway using the least significant bits:

0- cannot act as a gate way

1- act as a gateway

This modification is done on the DSDV.msg file by adding a new identifier on the exsisting library as shown below:

```
bool advertizeflag;/*A new field which  will be used to signal anode can  act as a gateway*/
```

DSDV.msg files are translated in to C++ code using the opp_msgc.program.The source code can be found in the Appendix section.

The generated DSDV_m.cc file wIll be incorporated with the existing DSDV routing protocol implementation with the following modification.

On the handle message method we have added the following code fragment

```
void DSDV_2::handleMessage(cMessage *msg)

{

    DSDV_HelloMessage * recHello = NULL;

    // When DSDV module receives selfmessage (scheduled
event)

    // it means that it's time for Hello message
broadcast event

    // i.e. Brodcast Hello messages to other nodes when
selfmessage=event

    // But if selmessage!=event it means that it is time
to forward useful Hello message to othert nodes

    if (msg->isSelfMessage())

    {

        if (msg==event)

        {

            //new hello message

            DSDV_HelloMessage * Hello = new
DSDV_HelloMessage("Hello");          //pointer to
interface and routing table

            if (!ift)

                ift = InterfaceTableAccess().get();

            if (!rt)

                rt = RoutingTableAccess().get();

            rt->purge();

            // count non-loopback interfaces
```

```
            // int numIntf = 0;

            // InterfaceEntry *ie = NULL;

            //for (int k=0; k<ift->getNumInterfaces();
k++)

            //  if (!ift->getInterface(k)->isLoopback())

            //  {ie = ift->getInterface(k); numIntf++;}

            // Filling the DSDV_HelloMessage fields

            // IPv4Address source = (ie->ipv4()-
>getIPAddress());

            IPv4Address source = (interface80211ptr-
>ipv4Data()->getIPAddress());

            Hello->setBitLength(128); ///size of Hello
message in bits

            Hello->setSrcIPAddress(source);

            sequencenumber += 2;

            Hello->setSequencenumber(sequencenumber);

            Hello->setNextIPAddress(source);

            Hello->setHopdistance(1);
if(advertizeflag==1) //advetizement will be added

Hello-->setGWF(1);// Set the gateway flag in to 1 thus
the node can act as gateway*/

else

Hello--> setGWF(0);

…

}
```

3.3.2 OLSR Routing Protocol Packet modification

OLSR maintains a set of repositories which are updated upon processing control messages which includes Multiple Interface association information set, Link set which is used to calculate the status of the interfaces, Neighbor set-uses information from the link set and registers both symmetric and asymmetric neighbors,2 hop neighbor set list of nodes which ae two hop away fom the souce,MultiPoint Relay(MPR) set consists of set of multi point , Multi point selectof this are nodes which select it as a multi point Topology Information Base (TIB) consists of the topolog information of each link and duplicate set list of ecentl pocessd messages.

Among this we have used the reserved bits of the link sensing message protocol header to signal the node can act as a gateway by assigning 0-can not act as a gateway and 1-can act as a gateway and modified the OLSR message file as shown below.

```
bool advertizeflag;/*A new field which  will be used to signal anode can  act as a gateway*/
```

The generated OLSR_m.cc file wIll be incorporated with the existing DSDV routing protocol implementation with the following modification.

On the handle message method we have added the following code fragment all the routing management tables and timers are implemented in the existing library and we will show the skeleton of the handle message method below:

```
void OLSR::handleMessage(cMessage *msg)

{

…

        if (duplicated == NULL)

        {

            // Process the message according to its type

            if (msg.msg_type() == OLSR_HELLO_MSG)

                process_hello(msg, receiverIfaceAddr,
src_addr, index);

if(advertizeflag==1)
```

```
            Hello-->setGWF(1);/* Set the gateway flag in
to 1 thus the node can act as a gateway*/

else

LinkSense--> setGWF(0);

            else if (msg.msg_type() == OLSR_TC_MSG)

                process_tc(msg, src_addr, index);

if(advertizeflag==1)

            Hello-->setGWF(1);/* Set the gateway flag in
to 1 thus the node can act as a gateway*/

            else if (msg.msg_type() == OLSR_MID_MSG)

                process_mid(msg, src_addr, index);

            else

            {

                debug("%f: Node %s can not process OLSR
packet because does not "

                      "implement OLSR type (%x)\n",

                      CURRENT_TIME,

                      getNodeId(ra_addr()),

                      msg.msg_type());

            }

        …

        }
```

3.4 configuring simulation

Configuration and input data for the simulation are in a configuration file usually called omnetpp.ini. In this section we will describe the initialization file used in the simulations and our experimental design.

For the homogeneous environment simulation we have choosen the number of nodes to be 5 and for heterogeneous environment we added two nodes which are accessed via access point. .We have simulated for 3000 sec in a 500 m^2 play ground area.

Routing and MAC protocols

The routing protocol used where the improved DSDV and OLSR. And the MAC protocol was IEEE 82.11 g which already exist in omnet++.

Mobility pattern

There exist a number of mobility models to simulate mobile ad-hoc networks among this the most commonly used mobility models the Random waypoint mobility model and the following seed values were used.Speed: 2m/s, Mobility wait time: 0.1s and Maximum Hop limit: 7.

Communication Model

We have used UDP protocol due at the transport layer and used various packet size starting from 100 bytes and increment it till it reaches 1000 bytes.

Radio characteristics

We used nodes with a transmitter power of 2mW with 250m transmission range and 54Mbps bit rate.

3.5 Performance metrics used

There are several quantitative metrics that might be used to analyze the performance of a routing protocol. The RFC 2501[23] defines the following measures. Basically they can be used to analyze the performance of any routing protocol.Among the metrics we used the following:

Throughput: is the number of successfully received bits in each second.

Packet delivery ratio: is the ratio of packets received over the total number of sent packet.

3.5 Scenarios used

3.5.1 Open Manet with multiple gate way

n this scenario any Adhoc compound module can act as a gate way if the channel control module allows it to act as a gate way as shown in figure 10.

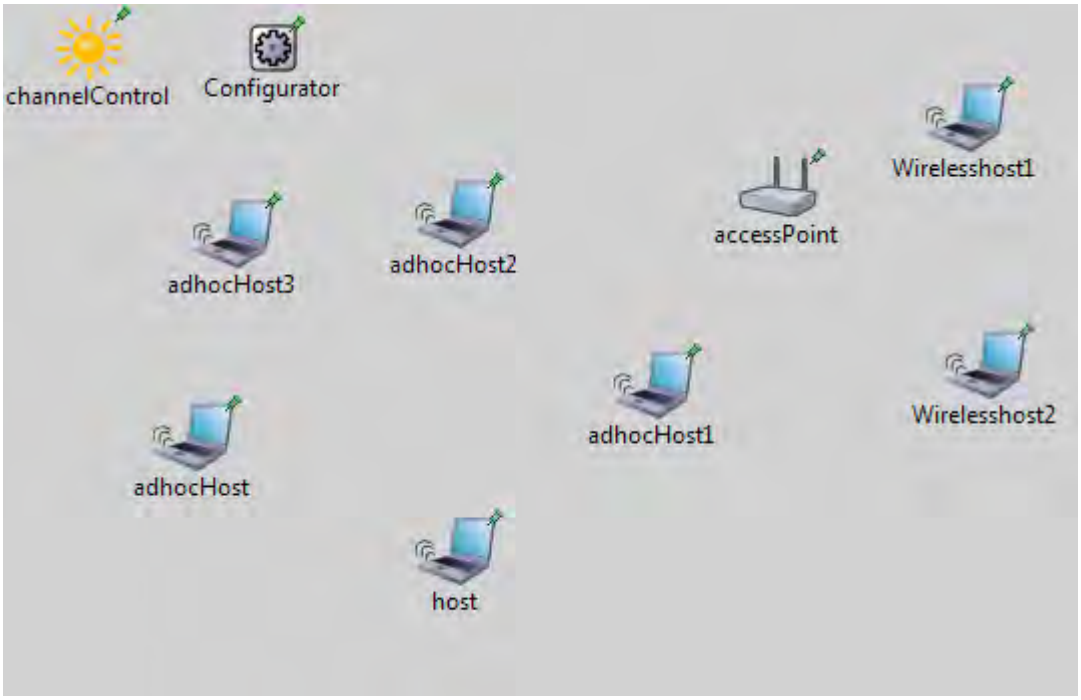


Figure 8: Open Manet with Multiple Gateways

A s depicted in the above figure each host can act as a gateway and selection of the gateway is controlled by the channel controller the source and the destination can be configured from the configuration file.

3.5.2 Open Manet with single Gate way

In this scenario one of the node was selected as a gate way and nodes used it to route their packets as shown in figure 9.

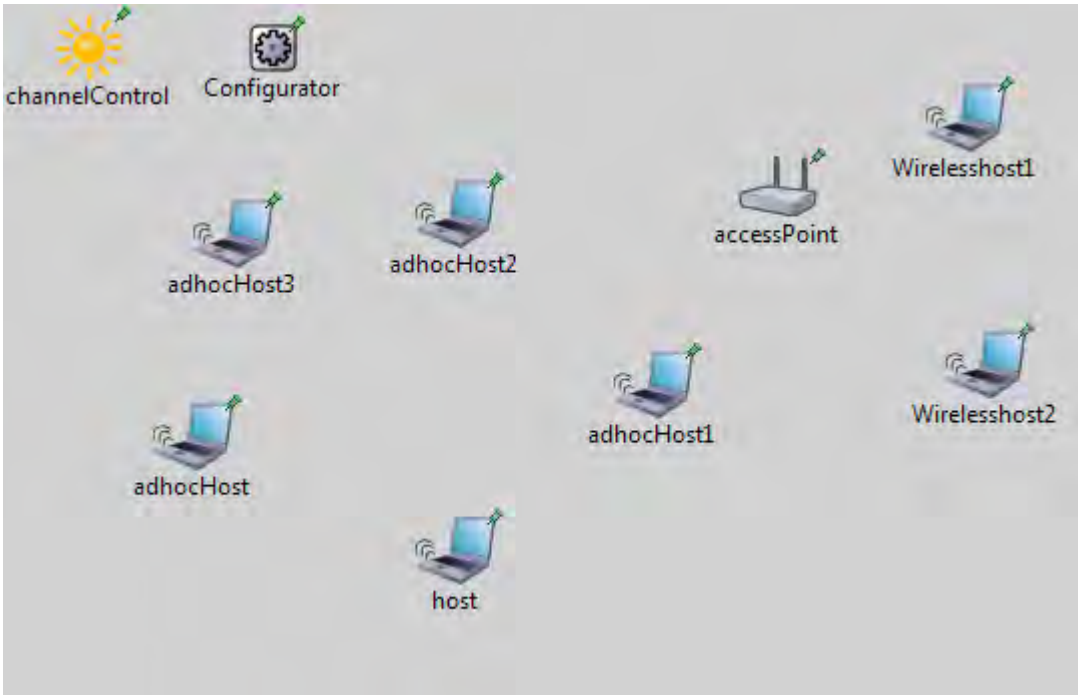


Figure 9: Open Manet with single gate way

In Figure 9 adhoc Host1 acts as a single gate way and adhocHost3 sends packets to wirelesshost1 and wirelesshost2 and the results are explained in chapter4 .

3.5.3 Closed MANET

In this Scenario the Adhoc compound module was used and simulation was performed as per the configuration file and a result shown in chapter 4 was obtained.

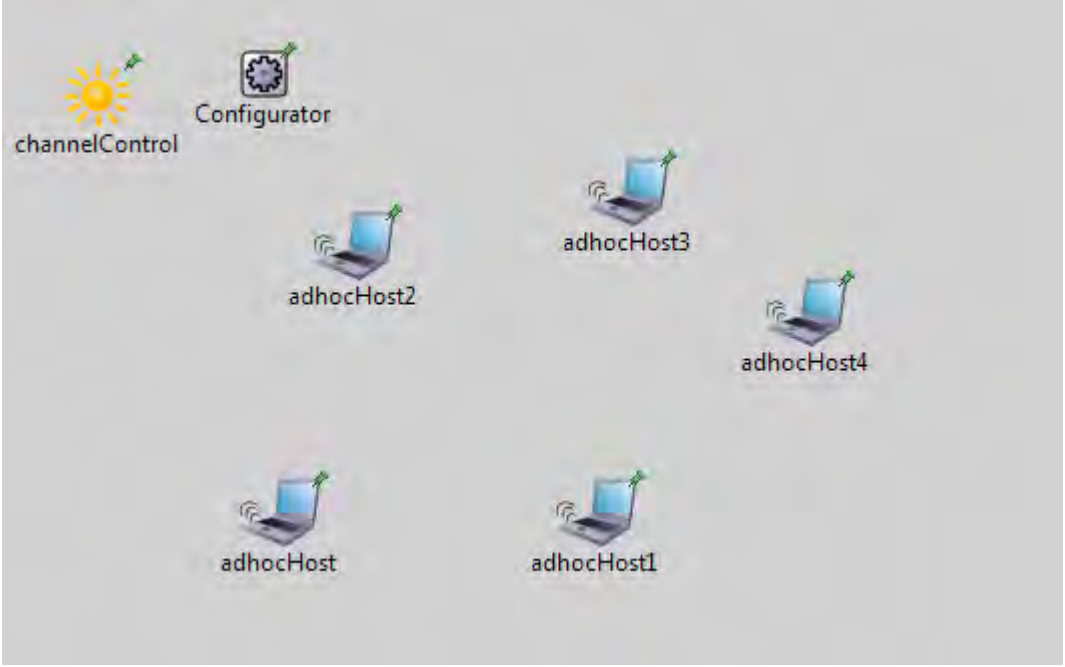


Figure 10: Closed MANET

In figure 8 adhoc host1 broadcasts message and the rest will receive various sized packets and represents a homogeneous MANET.

Chapter 4

Result Analysis

In this chapte we will discuss the experimental results obtained from the simulation runs.

4.1 Performance of DSDV using Throughput

Troughput:-indicates the fraction of the channel capacity usedused for useful transmission selection of a destination at the beginning of the simulation.Throughput is defined as the average rate of successful message delivery over a communication channel. This data may be delivered over a physical or logical link, or pass through a certain network node [22].

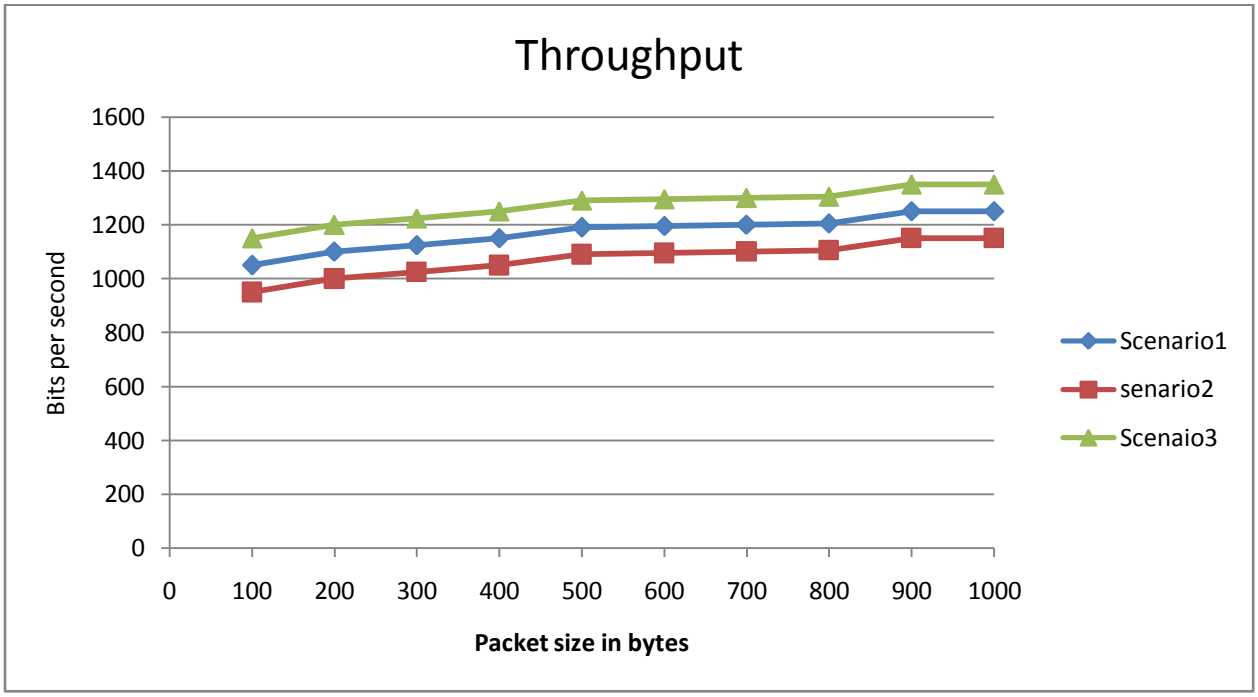


Figure 11 Performance evaluation of DSDV using throughput

As depicted in the figure above closed Manet (Scenario 3) out performs the heterogeneous environment due to the absence of control messages and this will result in the establishement of route cost thus the throughput is reduced.In Scenario 1 where the nodes use multiple gateway and Scenario 2 where the nodes use fixed gateway the difference is due to the presence of multiple gateway made the hosts to choose a gate way thus a network with multi gate way will have a better throughput.For packet size

up to 400 bytes in heterogeneous environment performance in throughput is increased by (9.5% - 10.5%) compared to the fixed gateway and the performance become stable between 800 and 1000 btes.

4.2 Performance of OLSR using Throughput

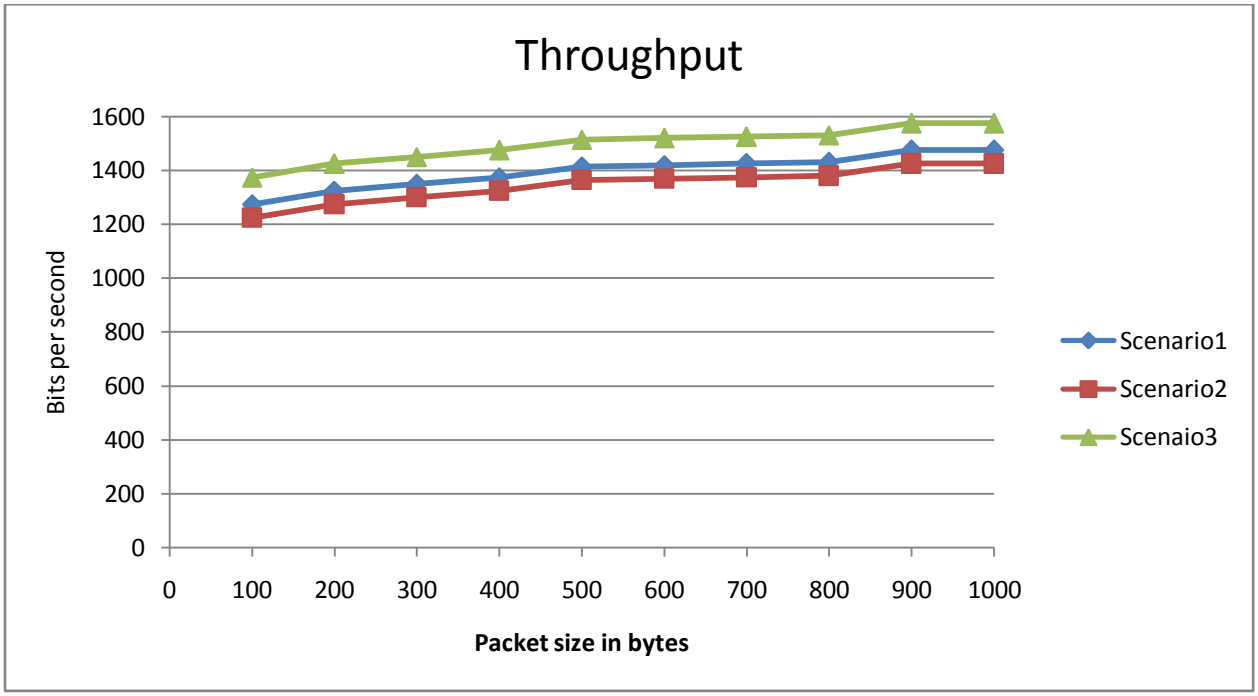


Figure 12 Performance evaluation of OLSR using throughput

As compared to the performance of DSDV OLSRperforms better than DSDV this is due to the usage of MPR to disscominate information and we have observed the presence of multiplegateway boosted the performance of the entire network.For packet size up to 400 bytes in heterogeneous environment performance in throughput is increased by (3.7% - 4.08%) compared to the fixed gateway and the performance become stable between 800 and 1000 btes.

4.3 Performance of DSDV using PDR as a performance metrics

Packet Delivery Ratio (PDR) is the ratio between the numbers of packets delivered to the receiver and the number of packets sent by the source [23]. Packet delivery ratio can be defined as:

$$PDR = \frac{\text{Successfully delivered packets}}{\text{Total number of transmitted packets}}$$

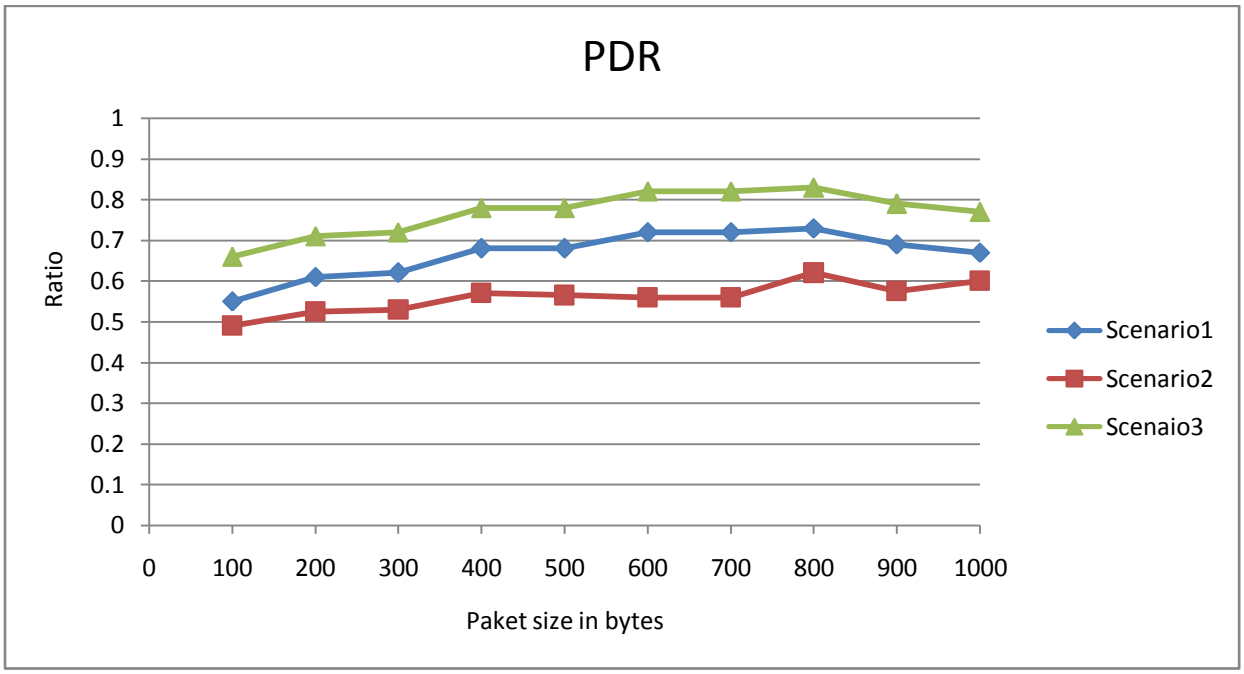


Figure 13 Performance of DSDV using PDR as a metrics

The maximum throughput is achivede when the packet size is 800 and the performance of heterogeneous environment degrads due to the addition of overhead.

4.3 Performance of OLSR using PDR as a performance metrics

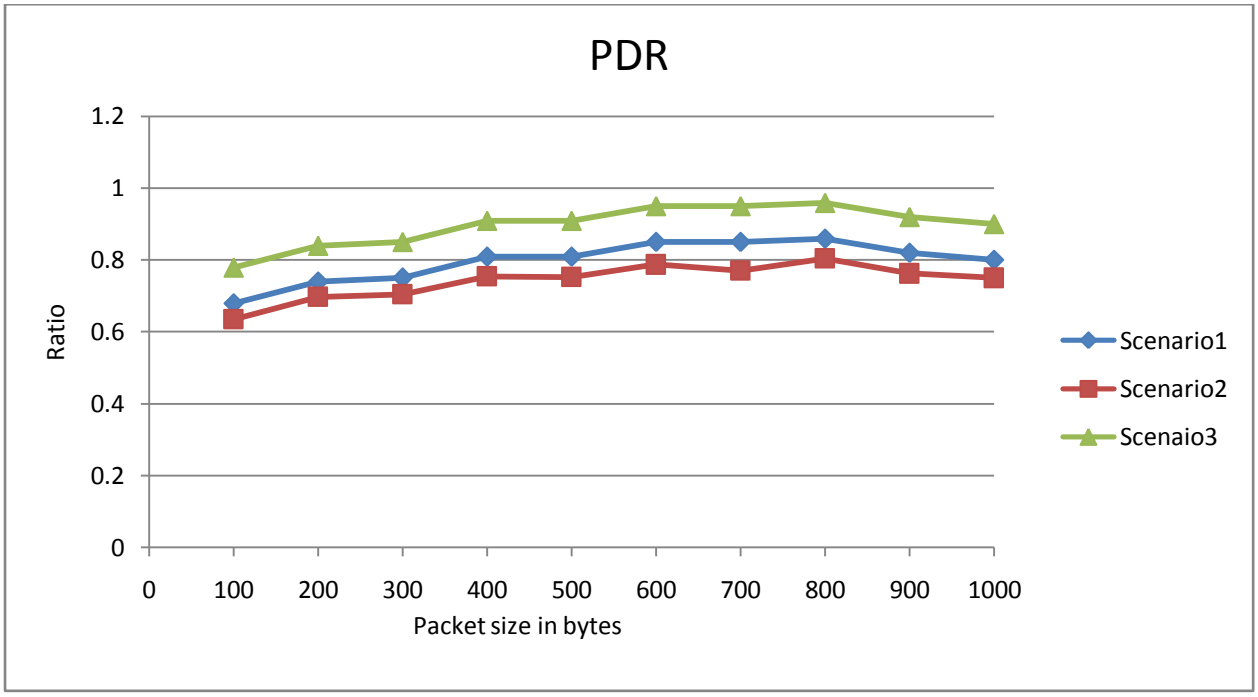


Figure 14 Performance of OLSR using PDR as a metrics

As compared to the performance of DSDV OLSR performs better than DSDV this is due to the usage of MPR to disscominate information and we have observed the presence of multiple gateway boosted the performance of the entire network.However after 800 bytes the PDR starts to decline .

Chapter 5

Conclusion and Recommendation

5.1 Conclusion

Mobile adhoc networks are multi hop wireless networks characterized by the absence of cenral coordinator, dynamic topology, connection link fluctuation and in such environments the loss of packet is norm. Due to the variation in the transmission range of mobile devices each device must cooperate in delivering messages across the network,to perform such tasks routing protocols are needed.Routing protocols use various computational algorithms to find shortest route to a destination.However,this protocols consumes time and space of this limited bandwidth of the network.

Among the factors that affect the size of the payload that can be efficiently transmitted in a single transaction are routing protocol overheads.Using throughput and packet delivery ratio as a performance metrics we have evaluated the performance of two proactive routing protocols and observed in addition to node density,node mobility packet size influence the performance of Mobile adhoc networks. We can conclude that there is no well suited routing protocol for every network condition and the choice of the routing protocol is dependent to the networking environment we will use.

In heterogeneous MANET where nodes use fixed gate way and adaptive gate way to communicate with other type of networks and have performance difference in the latter the performance increases due to the presence of multigate way and in addition to that selction of architecture of inter connection influences the performance of heterogeneous mobile adhoc networks.

Compared to the work of [2] the performance degradation between heterogeneous Mobile adhoc enviromnet implementing AODV and homogeneous AODV the performance difference is hugely reduced due to the usage of multiple gateways in both DSDV and OLSR implementations.

5.2 Recommendation

In this thesis we modeled a heterogeneous environment consisiting of closed MANET and another type of wireless network using access point and experimented the performance.This work can be extended to include wirless wide area networks like cellular networks and WIMAX technologies and see the performance effect.

A number of architectures have been proposed [5] using this architectures mobile adhoc network various routing protocols can be compared in performance considering this architectures..

MANETS use store and forward mechanism to route packets thus processing such packets consume the limited energy resource. This research can be exteneded by measuring the energy consumed per packet and see the effect of the life time of the network.

References

[1] Rupak Sharma, Rajeev Kr. Sharma and Manoj Kumar, "MANET: THE ESSENTIAL TECHNOLOGY FOR PERVASIVE COMPUTING", 2012.

[2] Zahian Ismail and Rosilah Hassan " Effects of Packet Size on AODV Routing Protocol Implementation in Homogeneous and Heterogeneous MANET", Kolej University Insaniah, University Kebangsaan, Malaysia, Third International Conference on Computational Intelligence, Modeling & Simulation,IEEE 2011.

[3] Guoyou He,"Destination-Sequenced Distance Vector (DSDV) Protocol", 2002.

[4] Andreas Tønnesen, "Impementing andextending the Optimized LinkState RoutingProtocol "Network Work Group, October2003.

[5] Dave cavalcanti, Dharma Agrawal, Carlos Cordeiro,Bin Xieand Anup Kumar,"Issues in integrating cellular networks,wlans, and manets: a futuristic heterogeneous wireless network",IEEE Wireless Communications,June 2005.

[6] Carlos DeMorias Cordeiro and Dharma p Agrawal," Mobile Adhoc Networking", Brazilian Symposium on Computer Networks,2002; 125–186.

[7] Tilak Raj,Himanshu Sharma,Vikas Gahlot,"Simulation Based Analysis of Proactive and Reactive Protocols in MANET with Varying Packet Size",Nature and Science, 2011;9(9).

[8] Ammar Odeh, Eman Abdel Fattah and Muneer Alshowkan,"PERFORMANCE EVALUATION OF AODV AND DSR ROUTING PROTOCOLS IN MANET NETWORKS",International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.4, July 2012.

[9] Patil V.P."Effect of Traffic Pattern on Packet Delivery Ratio in Reactive Routing Protocol of Manet", IOSR Journal of Electronics and Communication Engineering, Volume 2, Issue 2 (July-Aug 2012).

[10] Manoj Tolani and Rajan Mishra“Effect of Packet Size on Various MANET”, International Journal of Applied Information SystemsFoundation of Computer Science FCS, New York, USA, Volume 4– No.9, December 2012.

[11] Basu Dev Shivahare,Charu Wahiand Shalini Shivhare“Comparison Of Proactive And Reactive Routing Protocols InMobile Adhoc Network Using Routing Protocol PropertyRouting Protocols”,International Journal of Emerging Technology and Advanced EngineeringVolume 2, Issue 3, March 2012.

[12] Tony Larsson and Nicklas Hedman“Routing protocols in Mobile Adhoc Networks-A simulation study”, Lulea University of Technology, Stockholm 1998.

[13] Md Foyzer Mondal and Akshai Aggarwal,“A Report on Study of MANET Routing Protocols by Glomosim Simulator”,International Journal of Network Management, 15: 393–410;2005.

[14] Shivani Dua,Tanu Preet and Prof R.K singh ”Energy-Efficient Routing Protocols In Mobile Ad-Hoc Networks”,International Journal of Advanced Research in Computer Science and Software Engineering,Volume 2, Issue 1, January 2012.

[15] Yasser Kamal Hassan, Mohamed Hashim Abd El-Aziz and Safwat Abd El-Radi, “Performance Evaluation of Mobility Speed over MANET Routing Protocols”, International Journal of Network Security, Vol.11 ,No.3, PP. 128-138, Nov.2010.

[16] Manish Sharma and Gurpadam Singh“Evaluation of Proactive, Reactive and Hybrid Ad hoc Routing Protocol for various Battery models in VANET using Qualnet”,International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN) ISSN No. 2248-9738 (Print) Volume-1, Issue-2, 2011 .

[17] Basu Dev Shivahare and Shalini Shivhare”Comparisonof Proactive and Reactive Routing Protocols inMobile Adhoc bbcNetwork Using Routing Protocol Property: “, International Journal of Emerging Technology and Advanced EngineeringVolume 2, Issue 3, March 2012.

[18] Khaleel Ur Rahman Khan, Prof. A Venugopal Reddy and Rafi U Zaman”An Efficient Integrated Routing Protocol for Interconnecting Mobile Ad Hoc Network and the Internet”,International Journal of Computer and Electrical Engineering, Vol. 1, No. 1, April 2009.

[19] Susana Sargento, Tânia Calçada, João Paulo Barraca e"Mobile Ad-Hoc Networks Integration in the Daidalos Architecture",(FP6-2002-IST-1-506997).

[20] E.Ancillott, R.Bruno, M.Conti, E.Gregori and A.Pinizzotto," A Layer-2 Architecture for Interconnecting Multi-hop Hybrid AdHoc Networks to the Internet", University of Pisa and IIT InstituteNational Research Council (CNR), January (2006).

[21] Rakesh Kumar, Anil K. Sarje and Manoj Misra," Review Strategies and Analysis of Mobile Ad Hoc Network Internet Integration Solutions", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 6, July 2010.

[22] The OMNET++ user manual,http://www.omnet.org.

[23] S.Corson, J.Macker. MANET: Routing Protocol Performance Issues and Evaluation Considerations RFC 2501, IETF Network Working Group January1999.

Appendix

1) Source code for constructing Generic Mobile adhoc network

```
package dsdvhomogene;
import inet.networklayer.autorouting.ipv4.IPv4NetworkConfigurator;
import inet.world.radio.ChannelControl;
import inet.nodes.inet.AdhocHost;
network Net80211_DSDV
{
parameters:
int numHosts;
// int numFixHosts;
submodules:

    host[numHosts]: AdhocHost {
parameters:
@display("i=device/pocketpc_s;r=,,#707070");
    }
    channelControl: ChannelControl {
parameters:
@display("p=60,50;i=misc/sun");
    }
    configurator: IPv4NetworkConfigurator {
parameters:
        config=xml("<config><interface hosts='*' address='145.236.x.x'
netmask='255.255.0.0'/></config>");
@display("p=140,50;i=block/cogwheel_s");
    }
connectionsallowunconnected:
}
```

2) Modified Channel control module

2.1 Ned file

```
simple ChannelControl
{
parameters:
bool coreDebug = default(false); // debug switch for core framework
```

```
sdouble pMax @unit("mW") = default(20mW); // maximum sending power used for this network (in mW)
double sat @unit("dBm") = default(-110dBm); // signal attenuation threshold (in dBm)
double alpha = default(2); // path loss coefficient
double transmissionRange @unit("m")= default(250.0m) ;
double carrierFrequency @unit("Hz") = default(2.4GHz); // base carrier frequency of all the channels (in Hz)
int numChannels = default(1); // number of radio channels (frequencies)
string propagationModel
@enum("FreeSpaceModel","TwoRayGroundModel","RiceModel","RayleighModel","NakagamiModel","LogNormalShadowingModel") = default("FreeSpaceModel");
@display("i=misc/sun");
@labels(node);
}
```

2.2 Modified c++ file

```
#include"ChannelControl.h"
#include"FWMath.h"
#include<cassert>

#include"AirFrame_m.h"

#define coreEV (ev.isDisabled()||!coreDebug) ? ev : ev <<"ChannelControl: "

Define_Module(ChannelControl);
```

```
std::ostream&operator<<(std::ostream& os, constChannelControl::RadioEntry& radio)
{
   os << radio.radioModule->getFullPath() <<" (x="<< radio.pos.x<<",y="<< radio.pos.y<<"), "
<< radio.neighbors.size() <<" neighbor(s)";
return os;
}

std::ostream&operator<<(std::ostream& os, constChannelControl::TransmissionList& tl)
{
for (ChannelControl::TransmissionList::const_iterator it = tl.begin(); it != tl.end(); ++it)
      os <<endl<< *it;
return os;
}

ChannelControl::ChannelControl()
{
}

ChannelControl::~ChannelControl()
{
for (unsignedint i = 0; i <transmissions.size(); i++)
for (TransmissionList::iterator it = transmissions[i].begin(); it != transmissions[i].end(); it++)
delete *it;
}

/**
 * Calculates maxInterferenceDistance.
 *
 * @ref calcInterfDist
 */
voidChannelControl::initialize()
{
coreDebug = hasPar("coreDebug") ? (bool) par("coreDebug") : false;

   coreEV <<"initializing ChannelControl\n";

numChannels = par("numChannels");
transmissions.resize(numChannels);
```

```
lastOngoingTransmissionsUpdate = 0;

maxInterferenceDistance = calcInterfDist();

    WATCH(maxInterferenceDistance);
    WATCH_LIST(radios);
    WATCH_VECTOR(transmissions);
}

/**
 * Calculation of the interference distance based on the transmitter
 * power, wavelength, pathloss coefficient and a threshold for the
 * minimal receive Power
 *
 * You may want to overwrite this function in order to do your own
 * interference calculation
 */
doubleChannelControl::calcInterfDist()
{
double interfDistance;

//the carrier frequency used
double carrierFrequency = par("carrierFrequency");
//maximum transmission power possible
double pMax = par("pMax");
//signal attenuation threshold
double sat = par("sat");
//path loss coefficient
double alpha = par("alpha");

double waveLength = (SPEED_OF_LIGHT / carrierFrequency);
//minimum power  level to be able to physically receive a signal
double minReceivePower = pow(10.0, sat / 10.0);

    interfDistance = pow(waveLength * waveLength * pMax /
                    (16.0 * M_PI * M_PI * minReceivePower), 1.0 / alpha);

    coreEV <<"max interference distance:"<< interfDistance <<endl;
```

```
return interfDistance;
}

ChannelControl::RadioRefChannelControl::registerRadio(cModule *radio, cGate *radioInGate)
{
   Enter_Method_Silent();

RadioRef radioRef = lookupRadio(radio);

if (radioRef)
throwcRuntimeError("Radio %s already registered", radio->getFullPath().c_str());

if (!radioInGate)
      radioInGate = radio->gate("radioIn");

RadioEntry re;
   re.radioModule = radio;
   re.radioInGate = radioInGate->getPathStartGate();
   re.isNeighborListValid = false;
   re.channel = 0;  // for now
   re.isActive = true;
radios.push_back(re);
return&radios.back(); // last element
}

voidChannelControl::unregisterRadio(RadioRef r)
{
   Enter_Method_Silent();
for (RadioList::iterator it = radios.begin(); it != radios.end(); it++)
   {
if (it->radioModule == r->radioModule)
      {
RadioRef radioToRemove = &*it;
// erase radio from all registered radios' neighbor list
for (RadioList::iterator i2 = radios.begin(); i2 != radios.end(); ++i2)
         {
RadioRef otherRadio = &*i2;
            otherRadio->neighbors.erase(radioToRemove);
            otherRadio->isNeighborListValid = false;
            radioToRemove->isNeighborListValid = false;
```

```
        }

// erase radio from registered radios
radios.erase(it);
return;
      }
   }

error("unregisterRadio failed: no such radio");
}

ChannelControl::RadioRefChannelControl::lookupRadio(cModule *radio)
{
   Enter_Method_Silent();
for (RadioList::iterator it = radios.begin(); it != radios.end(); it++)
if (it->radioModule == radio)
return&(*it);
return 0;
}

constChannelControl::RadioRefVector&ChannelControl::getNeighbors(RadioRef h)
{
   Enter_Method_Silent();
if (!h->isNeighborListValid)
   {
      h->neighborList.clear();
for (std::set<RadioRef,RadioEntry::Compare>::iterator it = h->neighbors.begin(); it != h-
>neighbors.end(); it++)
         h->neighborList.push_back(*it);
      h->isNeighborListValid = true;
   }
return h->neighborList;
}

voidChannelControl::updateConnections(RadioRef h)
{
Coord& hpos = h->pos;
double maxDistSquared = maxInterferenceDistance * maxInterferenceDistance;
for (RadioList::iterator it = radios.begin(); it != radios.end(); ++it)
   {
```

```
RadioEntry *hi = &(*it);
if (hi == h)
continue;

// get the distance between the two radios.
// (omitting the square root (calling sqrdist() instead of distance()) saves about 5% CPU)
bool inRange = hpos.sqrdist(hi->pos) < maxDistSquared;

if (inRange)
      {
// nodes within communication range: connect
if (h->neighbors.insert(hi).second == true)
         {
            hi->neighbors.insert(h);
            h->isNeighborListValid = hi->isNeighborListValid = false;
         }
      }
else
      {
// out of range: disconnect
if (h->neighbors.erase(hi))
         {
            hi->neighbors.erase(h);
            h->isNeighborListValid = hi->isNeighborListValid = false;
         }
      }
   }
}

voidChannelControl::checkChannel(int channel)
{
if (channel >= numChannels || channel < 0)
error("Invalid channel, must above 0 and below %d", numChannels);
}

voidChannelControl::setRadioPosition(RadioRef r, constCoord& pos)
{
   Enter_Method_Silent();
   r->pos = pos;
   updateConnections(r);
```

```
}

voidChannelControl::setRadioChannel(RadioRef r, int channel)
{
   Enter_Method_Silent();
   checkChannel(channel);

   r->channel = channel;
}

constChannelControl::TransmissionList&ChannelControl::getOngoingTransmissions(int channel)
{
   Enter_Method_Silent();

   checkChannel(channel);
   purgeOngoingTransmissions();
returntransmissions[channel];
}

voidChannelControl::addOngoingTransmission(RadioRef h, AirFrame *frame)
{
   Enter_Method_Silent();

// we only keep track of ongoing transmissions so that we can support
// NICs switching channels -- so there's no point doing it if there's only
// one channel
if (numChannels == 1)
   {
delete frame;
return;
   }

// purge old transmissions from time to time
if (simTime() - lastOngoingTransmissionsUpdate> TRANSMISSION_PURGE_INTERVAL)
   {
      purgeOngoingTransmissions();
lastOngoingTransmissionsUpdate = simTime();
   }

// register ongoing transmission
```

```
take(frame);
   frame->setTimestamp(); // store time of transmission start
transmissions[frame->getChannelNumber()].push_back(frame);
}

voidChannelControl::purgeOngoingTransmissions()
{
for (int i = 0; i <numChannels; i++)
   {
for (TransmissionList::iterator it = transmissions[i].begin(); it != transmissions[i].end();)
      {
TransmissionList::iterator curr = it;
AirFrame *frame = *it;
         it++;

if (frame->getTimestamp() + frame->getDuration() + TRANSMISSION_PURGE_INTERVAL <
simTime())
         {
delete frame;
transmissions[i].erase(curr);
         }
      }
   }
}

voidChannelControl::sendToChannel(RadioRef srcRadio, AirFrame *airFrame)
{
// NOTE: no Enter_Method()! We pretend this method is part of ChannelAccess

// loop through all radios in range
constRadioRefVector& neighbors = getNeighbors(srcRadio);
int n = neighbors.size();
int channel = airFrame->getChannelNumber();
for (int i=0; i<n; i++)
   {
RadioRef r = neighbors[i];
if (!r->isActive)
      {
         coreEV <<"skipping disabled radio interface \n";
continue;
```

```
        }
if (r->channel == channel)
        {
            coreEV <<"sending message to radio listening on the same channel\n";
// account for propagation delay, based on distance in meters
// Over 300m, dt=1us=10 bit times @ 10Mbps
simtime_t delay = srcRadio->pos.distance(r->pos) / SPEED_OF_LIGHT;
            check_and_cast<cSimpleModule*>(srcRadio->radioModule)->sendDirect(airFrame->dup(),
delay, airFrame->getDuration(), r->radioInGate);
        }
else
            coreEV <<"skipping radio listening on a different channel\n";
    }

// register transmission
    addOngoingTransmission(srcRadio, airFrame);
}
```

1)Source code for DSDV routing protocol

1.1 Modified DSDV.msg

```
cplusplus {{
#include "IPv4Address.h"

}}
classnoncobject IPv4Address;
packet DSDV_HelloMessage
{
    IPv4Address srcIPAddress;
unsignedint sequencenumber;
    IPv4Address nextIPAddress;
int hopdistance;
bool advertizeflag;
}
```

2) Source code for OLSR routing protocol

2.1 Modified OLSR.msg

```
cplusplus {{
#include "OLSR_ETX_parameter.h"
#include "ManetAddress.h"
#include <string.h>

#ifndef nsaddr_t
typedef ManetAddress nsaddr_t;
#endif

/********** Message types **********/
/// %OLSR HELLO message type.
#define OLSR_HELLO_MSG      1
/// %OLSR TC message type.
#define OLSR_TC_MSG     2
/// %OLSR MID message type.
#define OLSR_MID_MSG        3


/// %OLSR HELLO message type.
#define OLSR_ETX_HELLO_MSG  OLSR_HELLO_MSG
/// %OLSR TC message type.
#define OLSR_ETX_TC_MSG     OLSR_TC_MSG
/// %OLSR MID message type.
#define OLSR_ETX_MID_MSG    OLSR_MID_MSG

/********** Packets stuff **********/

///
/// \brief Size of the addresses we are using.
///
/// You should always use this macro when interested in calculate addresses'
/// sizes. By default IPv4 addresses are supposed, but you can change to
/// IPv6 ones by recompiling with OLSR_IPv6 constant defined.
///

#ifdef OLSR_IPv6
#define ADDR_SIZE_DEFAULT   16
```

```
#else
#define ADDR_SIZE_DEFAULT   4
#endif


/// Maximum number of messages per packet.
#define OLSR_MAX_MSGS       4
#define OLSR_ETX_MAX_MSGS OLSR_MAX_MSGS
/// Maximum number of hellos per message (4 possible link types * 3 possible nb types).
#define OLSR_MAX_HELLOS     12
#define OLSR_ETX_MAX_HELLOS OLSR_MAX_HELLOS
/// Maximum number of addresses advertised on a message.
#define OLSR_MAX_ADDRS      64
#define OLSR_ETX_MAX_ADDRS OLSR_MAX_ADDRS

/// Size (in bytes) of packet header.
#define OLSR_PKT_HDR_SIZE   4
#define OLSR_ETX_PKT_HDR_SIZE OLSR_PKT_HDR_SIZE

/// Size (in bytes) of message header.
#define OLSR_MSG_HDR_SIZE   12
#define OLSR_ETX_MSG_HDR_SIZE OLSR_MSG_HDR_SIZE

/// Size (in bytes) of hello header.
#define OLSR_HELLO_HDR_SIZE 4
#define OLSR_ETX_HELLO_HDR_SIZE OLSR_HELLO_HDR_SIZE

/// Size (in bytes) of hello_msg header.
#define OLSR_HELLO_MSG_HDR_SIZE 4
#define OLSR_ETX_HELLO_MSG_HDR_SIZE OLSR_HELLO_MSG_HDR_SIZE

/// Size (in bytes) of tc header.
#define OLSR_TC_HDR_SIZE    4
#define OLSR_ETX_TC_HDR_SIZE OLSR_TC_HDR_SIZE


class OlsrAddressSize
{
  public:
  static uint32_t ADDR_SIZE;
```

```
};

typedef struct OLSR_ETX_iface_address {

/// Interface Address
 nsaddr_t  iface_address_;

/// Link quality extension
double  link_quality_;
double  nb_link_quality_;

/// Link delay extension
double  link_delay_;
double  nb_link_delay_;
int parameter_qos_;

 inline nsaddr_t& iface_address() { return iface_address_; }

/// Link quality extension
 inline double& link_quality() { return link_quality_; }
 inline double& nb_link_quality() { return nb_link_quality_; }

 inline double  etx()  {
double mult = (double) (link_quality() * nb_link_quality());
   switch (parameter_qos_) {
   case OLSR_ETX_BEHAVIOR_ETX:
    return (mult <0.01) ? 100.0 : (double) 1.0 / (double) mult;
    break;

   case OLSR_ETX_BEHAVIOR_ML:
    return mult;
    break;

   case OLSR_ETX_BEHAVIOR_NONE:
   default:
    return 0.0;
    break;
   }
 }
```

```
/// Link delay extension
  inline double& link_delay() { return link_delay_; }
  inline double& nb_link_delay() { return nb_link_delay_; }

} OLSR_ETX_iface_address;


/// Auxiliary struct which is part of the %OLSR_ETX HELLO message (struct OLSR_ETX_hello).
typedef struct OLSR_hello_msg {

/// Link code.
  uint8_t  link_code_;
/// Reserved.
  uint8_t  reserved_;
/// Size of this link message.
  uint16_t  link_msg_size_;
/// List of interface addresses of neighbor nodes.
  OLSR_ETX_iface_address  nb_iface_addrs_[OLSR_MAX_ADDRS];
/// Number of interface addresses contained in nb_iface_addrs_.
int    count;

  inline uint8_t&  link_code()    { return link_code_; }
  inline uint8_t&  reserved()    { return reserved_; }
  inline uint16_t&  link_msg_size()    { return link_msg_size_; }
  inline OLSR_ETX_iface_address&  nb_etx_iface_addr(int i)  { return nb_iface_addrs_[i]; }
  inline nsaddr_t  & nb_iface_addr(int i)  { return nb_iface_addrs_[i].iface_address_;}
  inline void set_qos_behaviour(int bh) {for(int i=0;i<OLSR_MAX_ADDRS;i++)
nb_iface_addrs_[i].parameter_qos_=bh;}

  inline uint32_t size() { return OLSR_HELLO_MSG_HDR_SIZE +
count*OlsrAddressSize::ADDR_SIZE; }

} OLSR_ETX_hello_msg;

#if 0
/// Auxiliary struct which is part of the %OLSR HELLO message (struct OLSR_hello).
typedef struct OLSR_hello_msg {

/// Link code.
    uint8_t link_code_;
```

```
/// Reserved.
   uint8_t reserved_;
/// Size of this link message.
   uint16_t    link_msg_size_;
/// List of interface addresses of neighbor nodes.
   nsaddr_t    nb_iface_addrs_[OLSR_MAX_ADDRS];
/// Number of interface addresses contained in nb_iface_addrs_.
int     count;

   inline uint8_t& link_code()     { return link_code_; }
   inline uint8_t& reserved()      { return reserved_; }
   inline uint16_t&    link_msg_size()     { return link_msg_size_; }
   inline nsaddr_t&    nb_iface_addr(int i)    { return nb_iface_addrs_[i]; }

   inline uint32_t size() { return OLSR_HELLO_MSG_HDR_SIZE +
count*OlsrAddressSize::ADDR_SIZE; }

} OLSR_hello_msg;
#endif

/// %OLSR HELLO message.

typedef struct OLSR_hello :cObject {

/// Reserved.
   uint16_t    reserved_;
/// HELLO emission interval in mantissa/exponent format.
   uint8_t htime_;
/// Willingness of a node for forwarding packets on behalf of other nodes.
   uint8_t willingness_;
/// List of OLSR_hello_msg.
   OLSR_hello_msg  hello_body_[OLSR_MAX_HELLOS];
/// Number of OLSR_hello_msg contained in hello_body_.
int     count;

   inline uint16_t&    reserved()      { return reserved_; }
   inline uint8_t& htime()         { return htime_; }
   inline uint8_t& willingness()       { return willingness_; }
   inline OLSR_hello_msg&  hello_msg(int i)    { return hello_body_[i]; }
```

```
    inline uint32_t size() {
        uint32_t sz = OLSR_HELLO_HDR_SIZE;
        for (int i = 0; i < count; i++)
            sz += hello_msg(i).size();
        return sz;
    }
} OLSR_hello;


/// %OLSR TC message.
typedef struct OLSR_tc :cObject{

/// Advertised Neighbor Sequence Number.
    uint16_t    ansn_;
/// Reserved.
    uint16_t    reserved_;
/// List of neighbors' main addresses.
    OLSR_ETX_iface_address  nb_main_addrs_[OLSR_MAX_ADDRS];
/// Number of neighbors' main addresses contained in nb_main_addrs_.
int     count;

    inline  uint16_t&   ansn()          { return ansn_; }
    inline  uint16_t&   reserved()      { return reserved_; }
    inline  nsaddr_t&   nb_main_addr(int i) { return nb_main_addrs_[i].iface_address_; }
    inline  OLSR_ETX_iface_address& nb_etx_main_addr(int i) { return nb_main_addrs_[i]; }
    inline void set_qos_behaviour(int bh) {for(int i=0;i<OLSR_MAX_ADDRS;i++)
nb_main_addrs_[i].parameter_qos_=bh;}
    inline  uint32_t size() { return OLSR_TC_HDR_SIZE + count*OlsrAddressSize::ADDR_SIZE; }
} OLSR_tc;

#if 0
/// %OLSR TC message.
typedef struct OLSR_tc :cObject{

/// Advertised Neighbor Sequence Number.
    uint16_t    ansn_;
/// Reserved.
    uint16_t    reserved_;
/// List of neighbors' main addresses.
    nsaddr_t    nb_main_addrs_[OLSR_MAX_ADDRS];
```

```
/// Number of neighbors' main addresses contained in nb_main_addrs_.
int     count;

   inline  uint16_t&   ansn()          { return ansn_; }
   inline  uint16_t&   reserved()      { return reserved_; }
   inline  nsaddr_t&   nb_main_addr(int i) { return nb_main_addrs_[i]; }

   inline  uint32_t size() { return OLSR_TC_HDR_SIZE + count*OlsrAddressSize::ADDR_SIZE; }

} OLSR_tc;
#endif

/// %OLSR MID message.
typedef struct OLSR_mid :cObject{

/// List of interface addresses.
   nsaddr_t    iface_addrs_[OLSR_MAX_ADDRS];
/// Number of interface addresses contained in iface_addrs_.
int     count;

   inline nsaddr_t & iface_addr(int i) { return iface_addrs_[i]; }
   void    setIface_addr(int i,nsaddr_t a) {iface_addrs_[i]=a; }

   inline uint32_t size()          { return count*OlsrAddressSize::ADDR_SIZE; }

} OLSR_mid;

#define MidSize sizeof (OLSR_mid)
#define HelloSize sizeof (OLSR_hello)
#define TcSize sizeof(OLSR_tc)

#define MAXBODY (HelloSize > TcSize ?  (HelloSize > MidSize? HelloSize: MidSize):(TcSize >
MidSize? TcSize:MidSize))


class   MsgBody {
public:
   OLSR_hello  hello_;
   OLSR_tc     tc_;
   OLSR_mid    mid_;
```

```
    MsgBody(MsgBody &other){
        memcpy(&hello_,&other.hello_,sizeof(OLSR_hello));
        memcpy(&tc_,&other.tc_,sizeof(OLSR_tc));
        memcpy(&mid_,&other.mid_,sizeof(OLSR_mid));
      }
    MsgBody(){
        memset(&hello_,0,sizeof(OLSR_hello));
        memset(&tc_,0,sizeof(OLSR_tc));
        memset(&mid_,0,sizeof(OLSR_mid));
       }
    ~MsgBody(){}
    MsgBody & operator =  (const MsgBody &other){
        if (this==&other) return *this;
                memcpy(&hello_,&other.hello_,sizeof(OLSR_hello));
                memcpy(&tc_,&other.tc_,sizeof(OLSR_tc));
                memcpy(&mid_,&other.mid_,sizeof(OLSR_mid));
          return *this;
        }
    OLSR_hello  * hello(){return &hello_;}
    OLSR_tc * tc(){return &tc_;}
    OLSR_mid* mid(){return &mid_;}


};

/// %OLSR message.
typedef struct OLSR_msg {

    uint8_t msg_type_;  ///< Message type.
    uint8_t vtime_;     ///< Validity time.
    uint16_t    msg_size_;  ///< Message size (in bytes).
    nsaddr_t    orig_addr_; ///< Main address of the node which generated this message.
    uint8_t ttl_;       ///< Time to live (in hops).
    uint8_t hop_count_; ///< Number of hops which the message has taken.
    uint16_t    msg_seq_num_;   ///< Message sequence number.

    MsgBody msg_body_;          ///< Message body.
    inline  uint8_t&    msg_type()  { return msg_type_; }
    inline  uint8_t&    vtime()     { return vtime_; }
```

```
inline  uint16_t&   msg_size()  { return msg_size_; }
inline  nsaddr_t    & orig_addr()   { return orig_addr_; }
inline  void    setOrig_addr(nsaddr_t a)    {orig_addr_=a; }
inline  uint8_t&    ttl()       { return ttl_; }
inline  uint8_t&    hop_count() { return hop_count_; }
inline  uint16_t&   msg_seq_num()   { return msg_seq_num_; }
inline  OLSR_hello& hello()     { return *(msg_body_.hello()); }
inline  OLSR_tc&    tc()        { return *(msg_body_.tc()); }
inline  OLSR_mid&   mid()       { return *(msg_body_.mid()); }


inline uint32_t size() {
   uint32_t sz = OLSR_MSG_HDR_SIZE;
   if (msg_type() == OLSR_HELLO_MSG)
      sz += hello().size();
   else if (msg_type() == OLSR_TC_MSG)
      sz += tc().size();
   else if (msg_type() == OLSR_MID_MSG)
      sz += mid().size();
   return sz;
}
OLSR_msg(){}
OLSR_msg(const OLSR_msg &other)
{
   msg_type_=other.msg_type_;  ///< Message type.
   vtime_=other.vtime_;        ///< Validity time.
   msg_size_=other.msg_size_;  ///< Message size (in bytes).
   orig_addr_=other.orig_addr_;    ///< Main address of the node which generated this message.
   ttl_=other.ttl_;        ///< Time to live (in hops).
   hop_count_=other.hop_count_;    ///< Number of hops which the message has taken.
   msg_seq_num_=other.msg_seq_num_;    ///< Message sequence number.
   msg_body_=other.msg_body_;          ///< Message body.
}

OLSR_msg & operator = (const OLSR_msg &other)
{
   if (this==&other) return *this;
   msg_type_=other.msg_type_;  ///< Message type.
   vtime_=other.vtime_;        ///< Validity time.
   msg_size_=other.msg_size_;  ///< Message size (in bytes).
```

```
        orig_addr_=other.orig_addr_;    ///< Main address of the node which generated this message.
        ttl_=other.ttl_;        ///< Time to live (in hops).
        hop_count_=other.hop_count_;    ///< Number of hops which the message has taken.
        msg_seq_num_=other.msg_seq_num_;    ///< Message sequence number.
        msg_body_=other.msg_body_;          ///< Message body.
        return *this;
    }

} OLSR_msg;

}}


struct OLSR_msg;

packet OLSR_pkt
{
@omitGetVerb(true);
bool reduceFuncionality=false;
short pkt_seq_num;          ///< Packet sequence number.
long sn;                    /// CapProbe packet sequence number
double send_time;           /// Packet sending timestamp
    OLSR_msg msg[];             ///< Packet body.
}
```

3) Configuration File

```
[General]
network = Network
#record-eventlog = true
#eventlog-message-detail-pattern = *:(not declaredOn(cMessage) and not declaredOn(cNamedObject)
and not declaredOn(cObject))

sim-time-limit = 3000s
*.numnodes =5
*.numwireless = 2
#description = "DSDV Simple test"
#**.drawCoverage=false
**.constraintAreaMinX = 0m
**.constraintAreaMinY = 0m
**.constraintAreaMinZ = 0m
**.constraintAreaMaxX = 500m
**.constraintAreaMaxY = 100m
**.constraintAreaMaxZ = 0m

**.arp.globalARP = true
**.adhocHost[*].mobilityType = "RandomWPMobility"
**.wirelessHost[*].mobilityType="RandomWPMobility"
**.speed = 2mps
**.waitTime = 0.1s

# nic settings
**.wlan*.bitrate = 54Mbps

**.wlan*.typename="Ieee80211Nic"
**.wlan*.opMode="g"
**.wlan*.mac.EDCA = false
**.wlan*.mgmt.frameCapacity = 10
**.wlan*.mac.maxQueueSize = 14
**.wlan*.mac.rtsThresholdBytes = 3000B
**.wlan*.mac.basicBitrate = 6Mbps  # 24Mbps
**.wlan*.mac.retryLimit = 7
**.wlan*.mac.cwMinData = 31
**.wlan*.mac.cwMinBroadcast = 31

# channel physical parameters
```

```
*.channelControl.pMax = 2.0mW

**.wlan*.radio.transmitterPower=2.0mW

**.routingProtocol="DSDV_2"
**.manetrouting.RNGseed_DSDV = 0
**.adhocHost[0].numUdpApps = 1
**.adhocHost[*].udpApp[*].typename = "UDPBasicBurst"
**.udpApp[0].destAddresses = "adhocHost[2]"
**.udpApp[0].localPort = 1234
**.udpApp[0].destPort = 1234
**.udpApp[0].messageLength = 100B
**.udpApp[0].sendInterval = 0.5s + uniform(-0.001s,0.001s)
**.udpApp[0].burstDuration = 0
**.udpApp[0].chooseDestAddrMode = "perBurst"
**.udpApp[0].sleepDuration = 1s
**.adhocHost[2].udpApp[*].typename = "UDPSink"
**.adhocHost[2].numUdpApps = 1
**.adhocHost[2].udpApp[0].localPort = 1234
```