# Performance Comparison of Routing Protocols in Mobile Ad Hoc Networks

Project report submitted in partial fulfilment of the requirement for the

Degree of Bachelor of Technology

In

**Computer Science and Engineering**

By

Prakshal Jain | 131308

Under the supervision of

**Dr. Hemraj Saini**

Assistant Professor (Senior Grade), Department of Computer Science and Engineering

To

Department of Computer Science and Engineering

**Jaypee University of Information Technology, Waknaghat, Solan - 173234, Himachal Pradesh, India.**

# Certificate

## Candidate's Declaration

I hereby declare that the work embodied in this report titled **"Performance comparison of routing protocols in mobile adhoc networks"** in partial fulfilment of the requirements for the award of the degree of "**Bachelor of Technology"** in "**Computer Science and Engineering"** submitted in the department of Computer Science and Engineering**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work Carried out over the period of August 2016 to May 2017 under the esteemed supervision of "**Dr. Hemraj Saini"** Assistant Professor (Senior Grade), Department of Computer Science and Engineering.

The matter presented within this report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Prakshal Jain, 131308

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr. Hemraj Saini

Assistant Professor (Senior Grade)

Department of Computer Science & Engineering

Dated:

# Acknowledgement

There are a lot of people who were indeed very helpful, kind, cooperative and hospitable along the development of this project as well. I express my gratitude to those concerned.

This project and report would have been impossible if it was not for the kind assistance, extreme support and continuous guidance of my learned project guide. The deep sense of heartfelt gratitude that I ow to my learned guide and supervisor **“Dr. Hemraj Saini”** cannot be expressed in words. His insurmountable help, constant encouragement, invaluable guidance, generous nature are among the very few blessing that he bestowed upon me time to time from the inception of this project to this day.

Prakshal Jain, 131308
(Department of Computer Science and Engineering)

# Table of Content

# List of Abbreviation

| | | |
|---|---|---|
| 1. | MANET | Mobile Ad Hoc Network |
| 2. | OLSR | Optimized Link State Routing Protocol |
| 3. | DSDV | Distance Sequenced Distance Vector |
| 4. | AODV | Ad-Hoc On-Demand Distance Vector Routing |
| 5. | DSR | Dynamic Source Routing |
| 6. | Ns-1 | Network Simulator 1 |
| 7. | Ns-2 | Network Simulator 2 |
| 8. | Ns-3 | Network Simulator 3 |
| 9. | NetAnim | Network Animator |

# List of Figures

# List of Graphs

# List of Tables

## Abstract

A Mobile Ad hoc Network (MANET) is basically a collection or group of wireless nodes that can move arbitrarily and can form a dynamic network without the need of an existing infrastructure or we can say a centralized management unit. MANET is configured automatically and the topology of the network changes as the node are mobile, and therefore tend to organize themselves in an arbitrary fashion.

MANET is quite easily deployed and hence, finds a widespread use in Military and Civilian Applications. Nodes interact with each other using multi hop wireless links, which allows this network to have a range of transmissions extended towards infinite. In MANET every node can also act like a router.

Routing Protocols are the rules which basically control the transmission of packets in any network, in MANET too, the transmission of information between the constantly moving nodes is managed and controlled using routing protocols. The main use of these routing protocols is in finding and discovering new routes and paths from source to destination.

It is interesting to note that MANET can easily be installed and deployed in so many situations and areas where it might be impossible to make use of a traditional network. But it should also be noted that this also poses many difficulties in such a network, which may pertain to shared medium, limitations on transmission range, mobility of nodes or energy constraints placed in MANET. Routing’s role plays an important one in management of MANET, it’s because of the mobility of the nodes and topological changes with it. Performance of any routing protocol in such an environment is majorly effected by the mobility and the mobility model used in MANET and its simulation.

# Chapter 1: INTRODUCTION

## 1.1 Introduction

Mobile ad hoc networks (MANET) are increasingly being deployed and evolved as important are of study of mobility of nodes. MANET lacks any kind of infrastructure and is comprised of wireless router and nodes which move arbitrarily and manage themselves. Topology and characteristics of this sort of network changes dynamically and arbitrarily, in which nodes move to and from without any fixed point reference. It supports multi hop routes and paths to allow communication between nodes and over wireless links. To facilitate a packet reaching from a source node to a destination it is required that the nodes are in the communication range of the network or it is facilitated with the use of intermediate nodes. Now a days MANETs have become pretty robust and perform efficiently the operations related to all the mobile networks as it includes functionality in the moving nodes and reduces the overhead, saving energy of the nodes.

Why MANET and its performance is Important?

MANETs are deployed very rapidly and without any prior planning or use of any pre-existing infrastructure hence, have seen exceptional importance in implementing of networks where infrastructure in unavailable, impractical or expensive. MANETs observe their use mostly in military communication, areas in a war zone, areas needing disaster relief etc. Challenges like power and storage constraints are usual in MANET along with wireless communication issues.

## 1.2 Problem Statement

Majorly the research in this field is carried out using simulation techniques, the same we have carried out to measure the performance of the routing protocols with custom traffic profiles. Although, due to lack of consistency in MANET and its application and models, it's observed that the results of different research studies yield different results. This happens with different traffic profiles and networks as well. Studying these facts we understood that the simulation scenarios used in the past studies are not reasonable for all protocols and their conclusions can't be just generalized. Making us consider this for finding the need of measuring their performances using the latest and most stable set of components. This is done in order to find an appropriate protocol for a given MANET scenario or application.

Using this as reference, our problem statement arises from the need of comparing various routing protocols falling under different categories using varied metrics to measure performance and hence, develop a reasonable and generalized conclusion towards the same.

## 1.3 Methodology

Agile method is basically an alternative to the conventional approach of software development. Keeping in mind the rapid and unforeseeable nature of our project we have found it to be fit, making use of incremental and iterative cadences. Apart from opting Agile we also have used a style of researching which is a mix of quantitative and qualitative research models to facilitate our knowledge about MANET and protocols to be used.

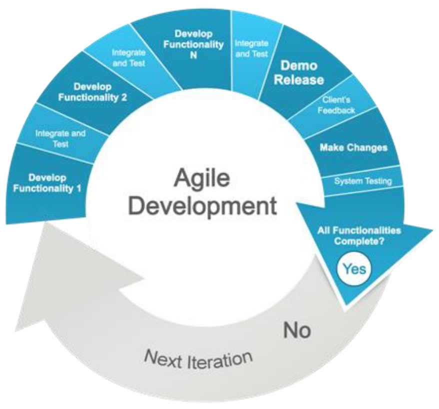


Figure 1. Agile Development

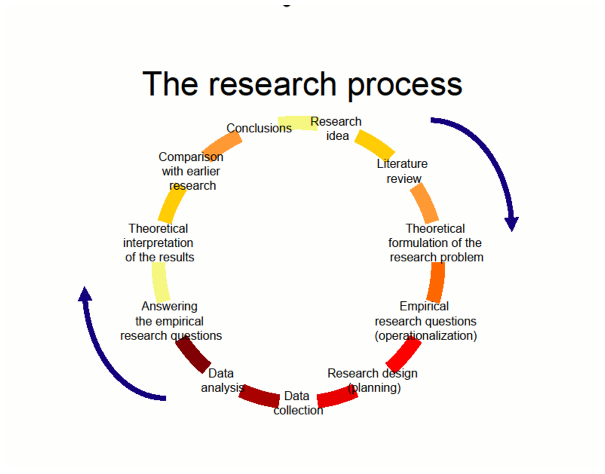

The research process
Research idea
Literature review
Theoretical formulation of the research problem
Empirical research questions (operationalization)
Research design (planning)
Data collection
Data analysis
Answering the empirical research questions
Theoretical interpretation of the results
Comparison with earlier research
Conclusions

Figure 2. Qualitative and quantitative research

## 1.4 Objective

The Objectives covered within this project are listed as:-

i. Study the basics of MANET and existing Routing Protocols.
ii. Survey the existing literature and review the work done.
iii. Study and analyse routing protocols and their categorisation.
iv. Selecting best protocols from conventional categories (proactive and reactive)
v. Implementing the protocols in virtual simulation and analyse their performances
vi. Perform an analysis of the system
vii. Performance Comparison of the Conventional routing protocols and identifying the best in each category.
viii. Concluding and generalising best suiting routing protocols in various situations and conditions

This is the base list of objectives and goals we intend to achieve with the completion of this project.

## 1.5 Organization

This report actually is like shadow of a timeline under which the project has prospered.

It initiates with the introduction to our project titled “Performance comparison of routing protocols in mobile ad hoc networks”. What comes next is an abstract idea of the project and brief details, objectives we intend to achieve followed by the methodology that we have used.

This is then followed by the Literature survey that we have conducted, where we have uniformly put all the information regarding categorization of the routing protocols and applications. It includes:-

i. Categorization of routing protocols
ii. Types of routing protocols
iii. Applications of MANET

The report then leads to the System Development with all the inclusion of analytical, computational and statistical development of the model of our project. This leads to the performance analysis of the routing protocols with the explanation of the simulation and techniques and measurement of the metrics.

The last we conclude incorporating the future scope and application along with some applications where our comparison proves to be useful.

# CHAPTER 2: LITERATURE SURVEY

## 2.1 INTRODUCTION

In MANET routing protocol should be capable to handle a very large number of nodes with limited resources. The main issue associate with the routing protocol involves being appeared and disappeared of nodes in various locations. It is necessary to reduce routing message overhead despite the increasing number of nodes. Another important issue is to keeping the routing table small, reason being increasing the routing table affects the control packets sent in the network and in turn affects large link overheads. Routing protocol needs to have following qualities to be effective: distributed operation, loop freedom, demand based operation, proactive operation, security and unidirectional link support. Distributed operation means that any node can enter or leave whenever they want. Loop-freedom is to prevent overhead created during sending information uselessly. Demand based operation is to decrease traffic and use bandwidth resources more efficiently. Proactive operation is used when they require enough bandwidth and energy resources. Security is the most important factor for any communication.  Routing protocol is categorized on the basis of how and when route are discovered, but both select the shortest path to the destination.

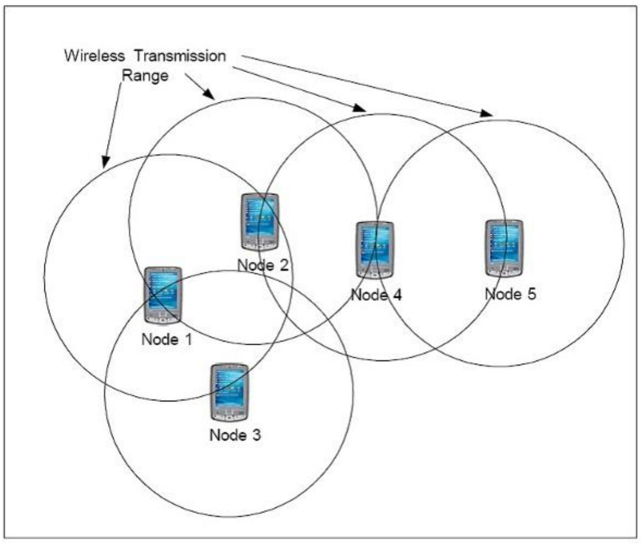


Figure 3. MANET NETWORK

## 2.2 CATEGORIZATION (Conventional)

### 2.2.1 Proactive Routing Protocols (Table Driven)

Proactive routing protocols are also known as Table-driven routing protocol uses link-state routing algorithms which floods link information about its neighbors frequently. This type of protocol keeps and maintains up-to-date routing information between every pair of nodes by sending control message periodically in network. One of the main advantages of this protocol is that routes are ready to use when needed. The major drawback of proactive routing protocols includes the overhead of flooding route. There are various proactive routing protocols present for MANET like DSDV, OLSR, and WRP etc.

### 2.2.2 Reactive Routing Protocols (On-Demand)

Reactive or on-demand routing protocols were designed to reduce overheads present in proactive protocols by maintaining information. It uses distance vector routing algorithm and establishes the route to given destination only when a node request it by initiating route discovery process. This protocols work on route discovery and route maintenance mechanism. Reactive routing protocols have drawback of delay in finding routes to new destination. There are number of reactive routing protocols available in MANET like DSR, AODV, TORA and LMR etc.

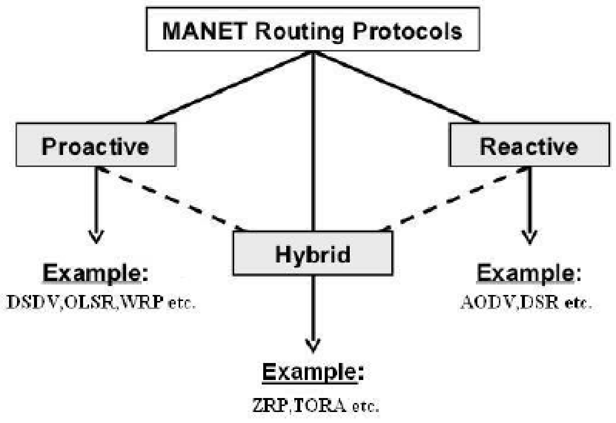


Figure 4. Categorization of Routing Protocols

## 2.3 CATEGORIZATION (Derived)

### 2.3.1 Hybrid Routing Protocols

This type of protocol combines the advantages of proactive and reactive routing. The routing is initially established with some proactively prospected routes and then serves the demand from additionally activated nodes through reactive flooding. The choice of one or the other method requires predetermination for typical cases. The main disadvantages of such algorithms are:

i. Advantage depends on number of other nodes activated.
ii. Reaction to traffic demand depends on gradient of traffic volume.

Examples of hybrid algorithms are ZRP (Zone Routing Protocol). ZRP uses IARP as pro-active and IERP as reactive component.

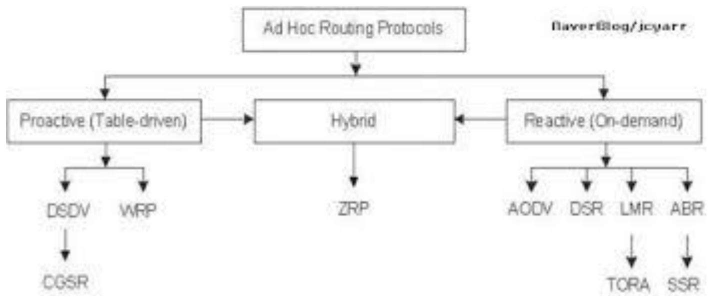


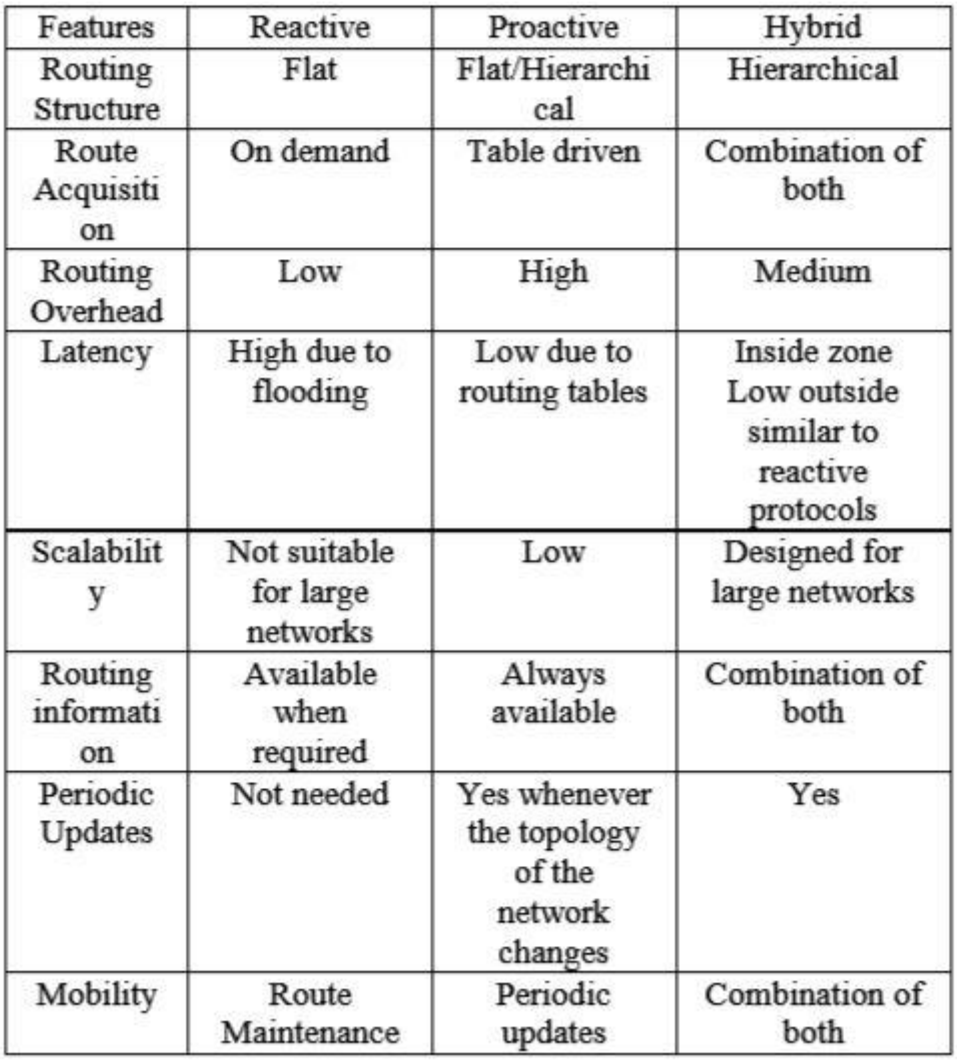
Figure 5. Categorization of Routing Protocols (Derived)

### 2.3.2 Hierarchical Routing Protocols

With this type of protocol the choice of proactive and of reactive routing depends on the hierarchic level in which a node resides. The routing is initially established with some proactively prospected routes and then serves the demand from additionally activated nodes through reactive flooding on the lower levels. The choice for one or the other method requires proper attribution for respective levels. The main disadvantages of such algorithms are:

i. Advantage depends on depth of nesting and addressing scheme.
ii. Reaction to traffic demand depends on meshing parameters.

Some of the Hierarchical routing protocols are Cluster Based Routing Protocol (CBRP) ZHLS (Zone-based Hierarchical Link State Routing Protocol)



| Features | Reactive | Proactive | Hybrid |
|---|---|---|---|
| Routing Structure | Flat | Flat/Hierarchical | Hierarchical |
| Route Acquisition | On demand | Table driven | Combination of both |
| Routing Overhead | Low | High | Medium |
| Latency | High due to flooding | Low due to routing tables | Inside zone Low outside similar to reactive protocols |
| Scalability | Not suitable for large networks | Low | Designed for large networks |
| Routing information | Available when required | Always available | Combination of both |
| Periodic Updates | Not needed | Yes whenever the topology of the network changes | Yes |
| Mobility | Route Maintenance | Periodic updates | Combination of both |

Table 1. Comparison of Categories

## 2.4 TYPES OF ROUTING PROTOCOLS

### 2.4.1 Optimized Link State Routing Protocol (OLSR)

Optimized Link State Routing Protocol is based on link state algorithm. Being a proactive routing protocol, it has an advantage of having the route immediately available within the standard routing table when needed. Due to optimization nature minimum flooding duplication occurs in highly connected network. Each node in the network selects a set of neighboring nodes to retransmit the packets and this set of nodes is called multipoint relays of that node. Instead of pure flooding the OLSR protocol employs Multipoint Relay (MPR) in network to reduce the possible overhead, flooding of broadcast and time interval for control message transmission. Only MPRs forward the control packets in such a way that information should reach entire network and these MPRs are responsible for declaring LS information. Each node periodically broadcasts a list of its one hop neighbors to select the MPRs with the help of hello message. Route calculations are done by MPR from source to destination node. OLSR supports three mechanisms: neighbor sensing, efficient flooding of control traffic and sufficient topology information. OLSR uses two types of control message: Hello and Topology Control (TC). Hello messages are used to find the information about the link status and node's neighbors while TC messages are used for broadcasting information about own advertised neighbors includes at least the MPR selector list.

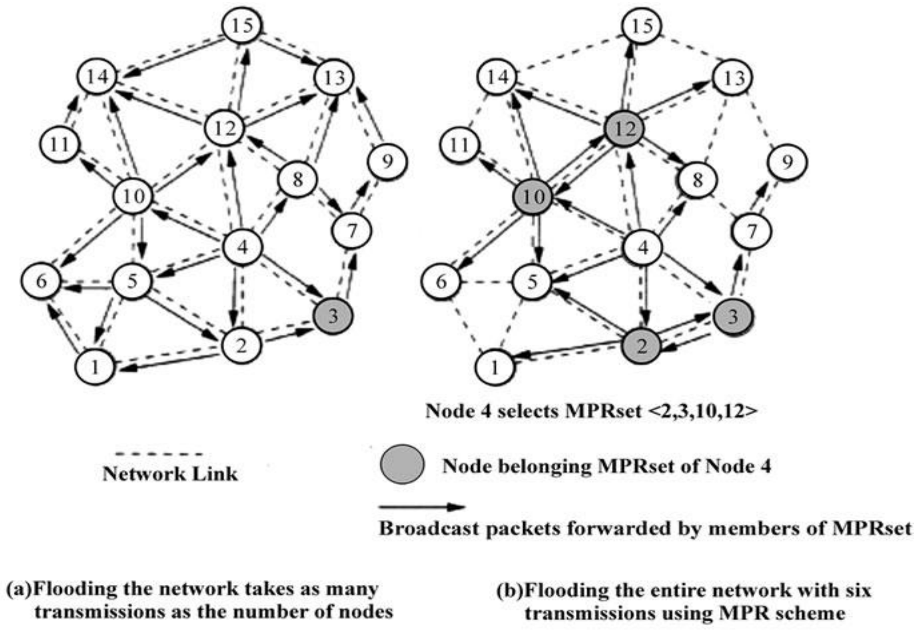
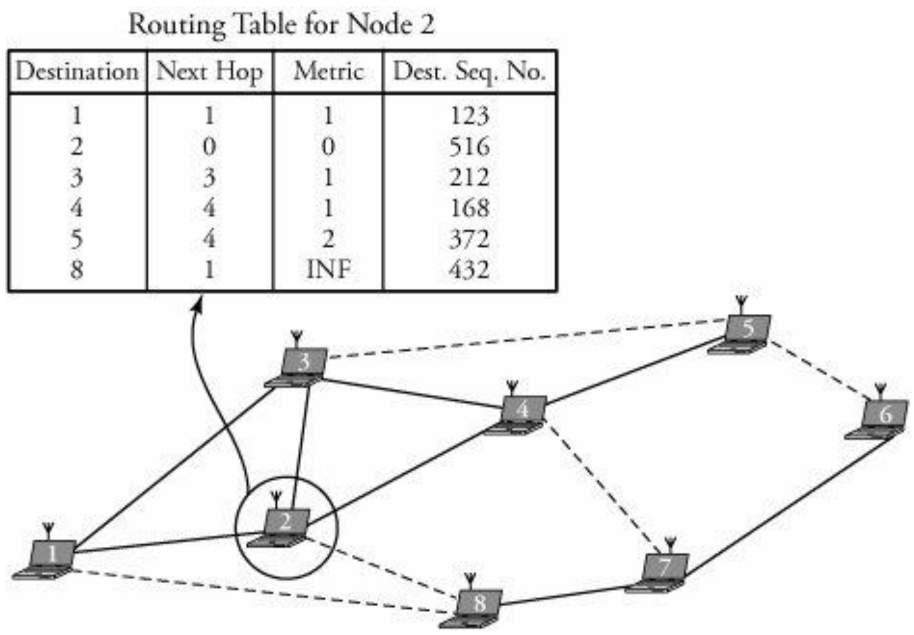

Node 4 selects MPRset <2,3,10,12>
Network Link
Node belonging MPRset of Node 4
Broadcast packets forwarded by members of MPRset
(a)Flooding the network takes as many transmissions as the number of nodes
(b)Flooding the entire network with six transmissions using MPR scheme

Figure 6. OLSR

## 2.4.2 Distance Sequenced Distance Vector (DSDV)

The DSDV routing protocol is a proactive routing protocol based on the Bellman-Ford routing algorithm that provides solution for shortest path between two nodes. In addition it introduces new feature i.e. sequence number for each routing table entry of entire network to avoid the formation of routing loops. Routing table is updated periodically throughout the network to maintain consistency in the table. To maintain the up-to-date view of the network, the tables are exchanged at regular interval of time. In order to reduce the amount of information carried during the broadcasting the routing information packets, two types of message are defined. One carry all the available routing information is called full dump and other types i.e. incremental dump carries information that has changed since the last full dump. A full dump requires multiple Network Protocol Data Units (NPDU) while the incremental dump requires only one to fit in all information. While receiving the information packet from another node, node compares the sequence number with the available sequence number for the entry, and updates the entry with the new sequence number if the sequence number is larger or smaller. If the information arrives with the same sequence number, metric entry will be required. In this protocol the updates lead to high control overhead during high mobility due to broken links. Another drawback is that node has to wait for a table update message initiated by the same destination node in order to obtain information about a particular destination node.



Routing Table for Node 2

| Destination | Next Hop | Metric | Dest. Seq. No. |
|---|---|---|---|
| 1 | 1 | 1 | 123 |
| 2 | 0 | 0 | 516 |
| 3 | 3 | 1 | 212 |
| 4 | 4 | 1 | 168 |
| 5 | 4 | 2 | 372 |
| 8 | 1 | INF | 432 |

Figure 7. DSDV

### 2.4.3 Ad-Hoc On-Demand Distance Vector Routing (AODV)

The AODV routing protocol is a reactive protocol that means routes are established whenever needed. It is based on On-demand mechanism of route discovery and route maintenance, plus the use of hop-by-hop routing and sequence number. Routing table consists of the information about the next hop to the destination and a sequence number received from the destination. This protocol supports two phase: route discovery, route maintenance; and data forwarding. Route discovery is done by broadcasting the RREQ message to its neighbors with the requested destination sequence number, which prevents looping problem. Neighbors reply with the RREP packets while having corresponding route otherwise forward RREQ packets to their neighbors. While noticed the breakage of the route the node sends RERR message to the neighbors. It uses the HELLO message periodically to inform the neighbor that link to the host is alive. While receiving the HELLO message node updates the lifetime of the node information in the routing table. Being a flat routing protocol, AODV protocol does not need any central administrative system to handle the routing process. It also reduces the control traffic message overhead at the cost of increased latency in finding new routes.

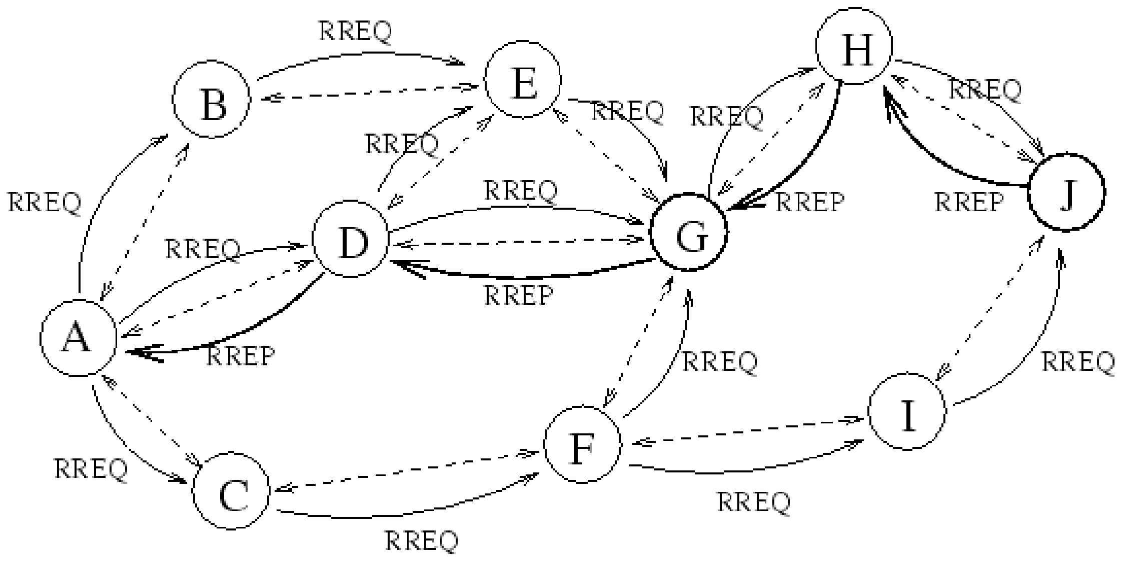


Figure 8. AODV

## 2.4.4 Dynamic Source Routing (DSR)

DSR is a reactive protocol based on source routing concept that requires each packet to carry the full address (every hop in the route) from source to destination. It is based on On-demand mechanism of route discovery and route maintenance. An advantage of DSR protocol is that nodes can store multiple routes in their route cache. Source node can check its route cache for a valid route before initiating route discovery, and if a valid route is found there is no need for route discovery. On the other hand, if a node does not have such a route, it initiates route discovery by broadcasting a RREQ packet. The RREQ packet contains the address of the destination along with address of source, a route record field and a unique identification number. Once the RREQ reaches either the destination or a node that knows a route to destination, it responds with a RREP along with the reverse of the route collected by the RREQ. A failed link is detected by either actively monitoring acknowledgements or passively running in promiscuous mode, overhearing that packet is forwarded by neighboring node. The failed link is notified to the source node with RERR packet. The source node can use other known route to destination node or the process of route discovery is initiated again to find new route to destination. Another thing is to be noted that it does not require hello message exchanges, therefore nodes can enter sleep node to conserve their power. Also saves a considerable amount of bandwidth in the network.

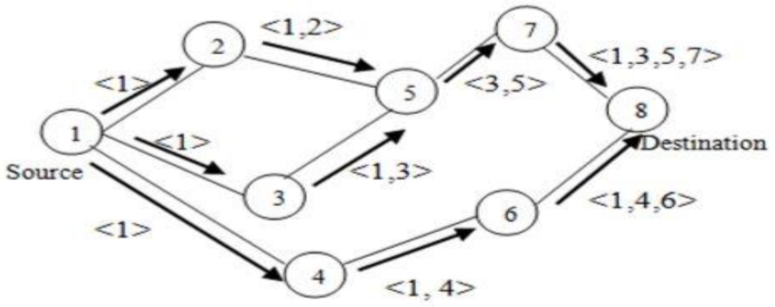


Figure 9. DSR Request

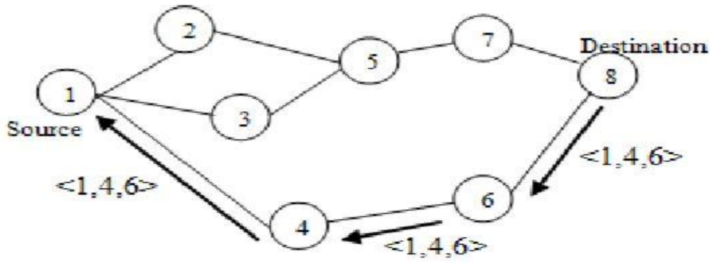
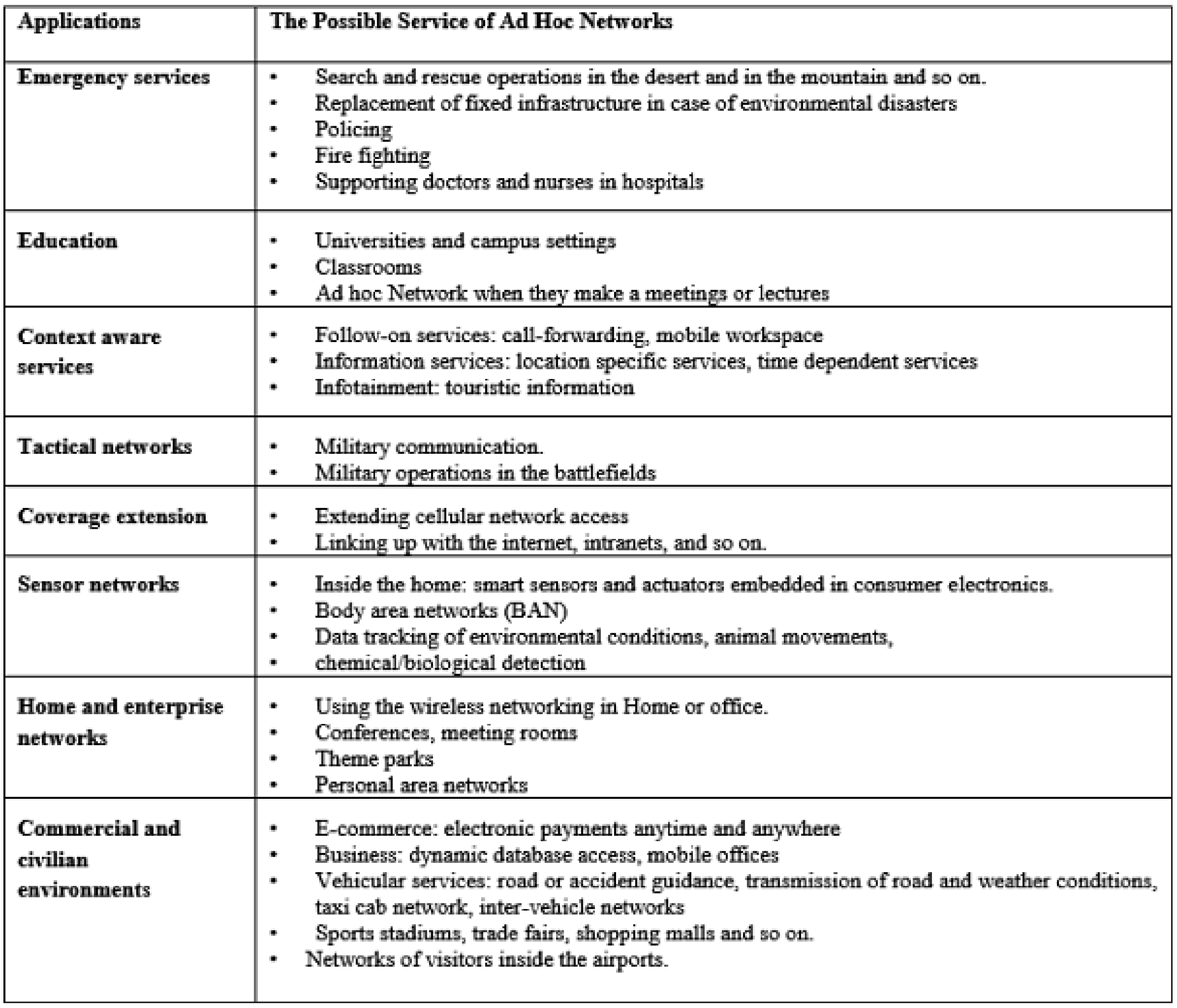


Figure 10. DSR Reply

## 2.5 APPLICATIONS OF MANET

There is a significant increase in the use of portable devices which brings in the very reason of increased rate of progress in the fields like wireless communication, MANET, Ad hoc networks etc. Private communication, Military and commercial communication all are experiencing growth. Ability to exchange information irrespective of demographic locations is a major reason of evolution in Mobile ad hoc networks. Every node in MANET moves dynamically and experiences topological changes unpredictably which is not present in other infrastructure based networks. Decentralized architecture gives MANET an edge by making it more flexible and robust.



| **Applications** | **The Possible Service of Ad Hoc Networks** |
|---|---|
| **Emergency services** | • Search and rescue operations in the desert and in the mountain and so on.<br>• Replacement of fixed infrastructure in case of environmental disasters<br>• Policing<br>• Fire fighting<br>• Supporting doctors and nurses in hospitals |
| **Education** | • Universities and campus settings<br>• Classrooms<br>• Ad hoc Network when they make a meetings or lectures |
| **Context aware services** | • Follow-on services: call-forwarding, mobile workspace<br>• Information services: location specific services, time dependent services<br>• Infotainment: touristic information |
| **Tactical networks** | • Military communication.<br>• Military operations in the battlefields |
| **Coverage extension** | • Extending cellular network access<br>• Linking up with the internet, intranets, and so on. |
| **Sensor networks** | • Inside the home: smart sensors and actuators embedded in consumer electronics.<br>• Body area networks (BAN)<br>• Data tracking of environmental conditions, animal movements,<br>• chemical/biological detection |
| **Home and enterprise networks** | • Using the wireless networking in Home or office.<br>• Conferences, meeting rooms<br>• Theme parks<br>• Personal area networks |
| **Commercial and civilian environments** | • E-commerce: electronic payments anytime and anywhere<br>• Business: dynamic database access, mobile offices<br>• Vehicular services: road or accident guidance, transmission of road and weather conditions, taxi cab network, inter-vehicle networks<br>• Sports stadiums, trade fairs, shopping malls and so on.<br>• Networks of visitors inside the airports. |

Table 2. Applications of MANET

### 2.5.1 Military Sector / Tactical Network

Military equipment now routinely contains some sort of computer equipment. Ad- hoc networking would allow the military to take advantage of commonplace network technology to maintain an information network between the soldiers, vehicles, and military information headquarters. The basic techniques of ad hoc network came from this field.

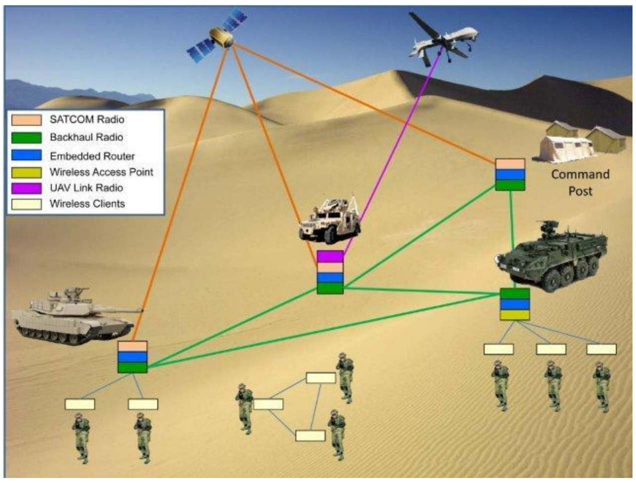


Figure 11. Application of MANET in Military

### 2.5.2 Commercial Sector

Ad hoc can be used in emergency/rescue operations for disaster relief efforts, e.g. in fire, flood, or earthquake. This may be because all of the equipment was destroyed, or perhaps because the region is too remote. Rescuers must be able to communicate in order to make the best use of their energy, but also to maintain safety. By automatically establishing a data network with the communications equipment that the rescuers are already carrying, their job made easier. Other commercial scenarios include e.g. ship-to-ship ad hoc mobile communication, law enforcement, etc.

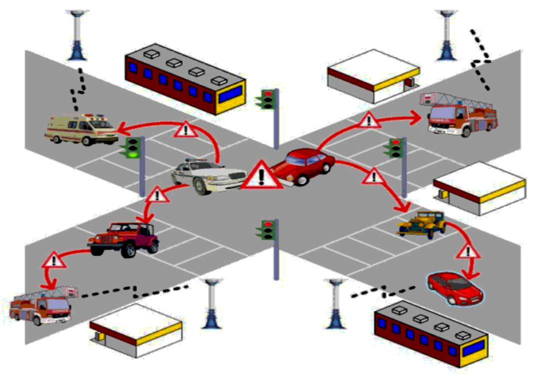


Figure 12. Application of MANET in Commercial Sector

2.5.3 Low Level

Low level application are the ones like present in our home networks where information is being exchanged directly, as the devices communicate without hinderence or levels between them.

2.5.4 Data Networks

Data networks are formed when computing of networks are allowed to transmit data for others. This allows data networks to be extended far beyond a reach that can be measured using a network with installed infrastructure.

2.5.5 Sensor Networks

This technology is a network composed of a very large number of small sensors. These can be used to detect any number of properties of an area. Examples include temperature, pressure, toxins, pollutions, etc. The capabilities of each sensor are very limited, and each must rely on others in order to forward data to a central computer. Individual sensors are limited in their computing capability and are prone to failure and loss. Mobile ad-hoc sensor networks could be the key to future homeland security.

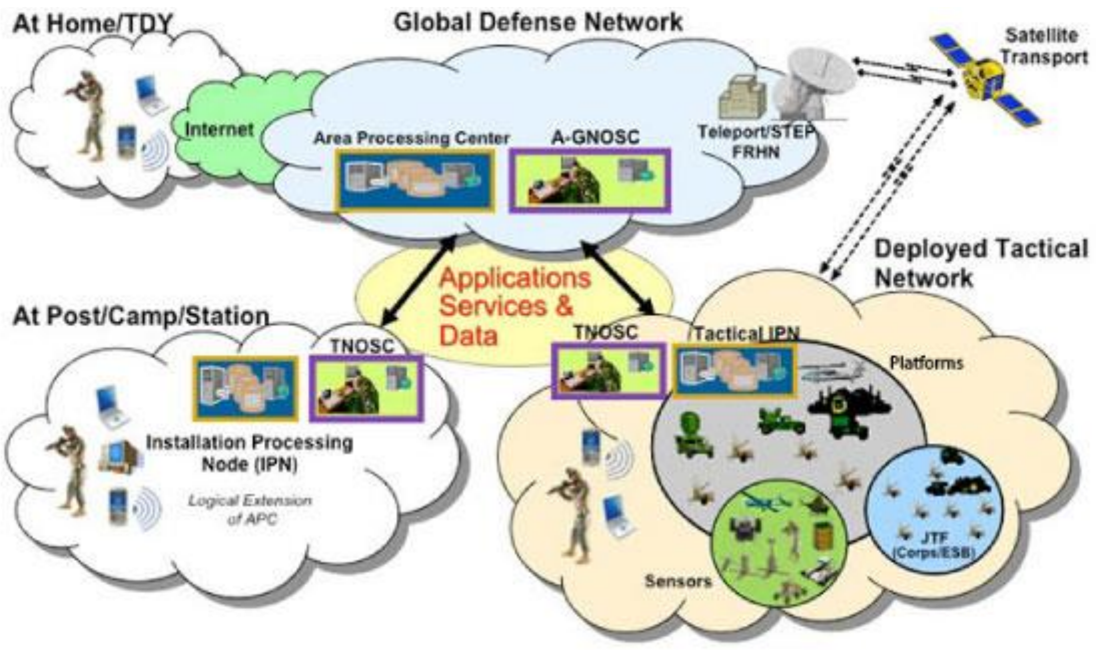


Figure 13. Application MANET

## 2.6 ATTACKS ON MANET

In order to get a grasp on how MANET networks perform in the real world scenarios and also to study the areas where MANETs are vulnerable, we studied about the attacks that make MANETs either difficult to implement or prone to malicious activities. It becomes really challenging to make a mobile network secure. Understanding the fact, for it to be even possible that we make a network secure we need to first understand about the attacks that can harm or disrupt the network. Due to the decentralized administration and structure less nature MANETs are more prone to attacks than a wired network. These attacks are summarized in the following table.

| MANET security Layer | Attacks |
|---|---|
| Application Layer | Malicious code, Repudiation. |
| Transport Layer | Session hijacking, SYN Flooding. |
| Network Layer | Flooding, Black Hole, Grey Hole. Worm Hole, Link Spoofing etc. |
| Data Link Layer | Traffic analysis and monitoring. |
| Physical Layer | Traffic Jamming, Eavesdropping. |

Table 3. Attacks on MANET

After a rigorous round of research we now can broadly categorize attacks on MANETs as Passive Attacks and Active attack based on the nature of intent.

### 2.6.1 Passive Attacks

In Passive attacks, the intruder only performs some kind of monitoring on certain connections to get information about the traffic without injecting any fake information. This type of attack serves the attacker to gain information and makes the footprint of the invaded network in order to apply the attack successfully. The types of passive attacks are eavesdropping, traffic analysis and snooping:

i. Eavesdropping: This is a passive attack. The node simply observes the confidential information. This information can be later used by the malicious node. The secret information like location, public key, private key, password etc. can be fetched by eavesdropper.

ii. Traffic Analysis: In MANETs the data packets as well as traffic pattern both are important for adversaries. For example, confidential information about network topology can be derived by analyzing traffic patterns. Traffic analysis can also be conducted as active attack by destroying nodes, which stimulates self-organization in the network, and valuable data about the topology can be gathered.

iii. Snooping: Snooping is unauthorized access to another person's data. It is similar to eavesdropping but is not necessarily limited to gaining access to data during its transmission. Snooping can include casual observance of an e-mail that appears on another's computer screen or watching what someone else is typing. More sophisticated snooping uses software programs to remotely monitor activity on a computer or network device. Malicious hackers (crackers) frequently use snooping techniques to monitor key strokes, capture passwords and login information and to intercept e-mail and other private communications and data transmissions. Corporations sometimes snoop on employees legitimately to monitor their use of business computers and track Internet usage. Governments may snoop on individuals to collect information and prevent crime and terrorism. Although snooping has a negative aspect in general but in computer technology snooping can refer to any program or utility that performs a monitoring function.

### 2.6.2 Active Attacks

In Active attacks, the intruder performs an effective violation on either the network resources or the data transmitted; this is done by International Journal on New Computer Architectures and Their Applications causing routing disruption, network resource depletion, and node breaking. In the following are the types of active attacks over MANET and how the attacker's threat can be performed

i. Flooding attack: In flooding attack, attacker exhausts the network resources, such as bandwidth and to consume a node's resources, such as computational and battery power or to disrupt the routing operation to cause severe degradation in network performance. For example, in AODV protocol, a malicious node can send a large number of RREQs in a short period to a destination node that does not exist in the network.

ii. Black hole Attack: Route discovery process in AODV is vulnerable to the black hole attack. The mechanism, that is, any intermediate node may respond to the RREQ message if it has a fresh enough route, devised to reduce routing delay, is used by the malicious node to compromise the system. In this attack, when a malicious node listens to a route request packet in the network, it responds with the claim of having the shortest and the freshest route to the destination node even if no such route exists. As a result, the malicious node easily misroute network traffic to it and then drop the packets transitory to it.

iii. Wormhole Attack: In a wormhole attack, an attacker receives packets at one point in the network, "tunnels" them to another point in the network, and then replays them into the network from that point. Routing can be disrupted when routing control message are tunneled. This tunnel between two colluding attacks is known as a wormhole. Wormholes are hard to detect because the path that is used to pass on information is usually not part of the actual network. Wormholes are dangerous because they can do damage without even knowing the network.

iv. Gray-hole attack: This attack is also known as routing misbehavior attack which leads to dropping of messages. Gray-hole attack has two phases. In the first phase the node advertise itself as having a valid route to destination while in second phase, nodes drops intercepted packets with a certain probability.

v. Link spoofing attack: In a link spoofing attack, a malicious node advertises fake links with non-neighbors to disrupt routing operations. For example, in the OLSR protocol, an attacker can advertise a fake link with a target's two hop neighbors.

vi. Malicious code attacks: malicious code attacks include, Viruses, Worms, Spywares, and Trojan horses, can attack both operating system and user application.

vii. Repudiation attacks: Repudiation refers to a denial of participation in all or part of the communications. Many of encryption mechanism and firewalls used at different layer are not sufficient for packet security. Application layer firewalls may take into account in order to provide security to packets against many attacks. For example, spyware detection software has been developed in order to monitor mission critical services.

viii. Session Hijacking: Attacker in session hijacking takes the advantage to exploits the unprotected session after its initial setup. In this attack, the attacker spoofs the victim node's IP address, finds the correct sequence number i.e. expected by the target and then launches various DoS attacks. In Session hijacking, the malicious node tries to collect secure data (passwords, secret keys, logon names etc.) and other information from nodes. Session hijacking attacks are also known as address attack which make effect on OLSR protocol. The TCP-ACK storm problem may occur when malicious node launches a TCP session hijacking attack.

ix. SYN Flooding Attack: The SYN flooding attacks are the type of Denial of Service (DoS) attacks, in which attacker creates a large number of half opened TCP connection with victim node. These half opened connection are never completes the handshake to fully open the connection.

x. Denial of service attack: Denial of service attacks are aimed at complete disruption of routing information and therefore the whole operation of ad-hoc network.

xi. Jamming: Jamming is a special class of DoS attacks which are initiated by malicious node after determining the frequency of communication. In this type of attack, the jammer transmits signals along with security threats. Jamming attacks also prevents the reception of legitimate packets.

xii. Selfish Misbehavior of Nodes: Attacks under this category, are directly affects the self-performance of nodes and does not interfere with the operation of the network. It may include two important factors. Conservation of battery power Gaining unfair share of bandwidth.

xiii. Traffic monitoring and analysis: Traffic monitoring and analysis can be deployed to identify the communication parties and functionalities, which could provide information to launch further attacks. Since these attacks are not specific to the MANET, other wireless networks, such as the cellular network, satellite network, and WLAN also suffer from these potential vulnerabilities. We did not focus on attacks on this layer for the security of MANET.

## 2.7 CHALLENGES IN MANET

MANET comprises of mobile nodes, vulnerability towards compromised nodes etc. this makes MANET more prone to malicious attacks than on a wired network. Also when considering other factors like dynamically changing topology, decentralized administration, and lack of physical security it brings in the picture of the challenges faced in the implementation of MANET. Some of these are cited:

i. Autonomous: No centralized administration entity is available to manage the operation of the different mobile nodes.

ii. Dynamic topology: Nodes are mobile and can be connected dynamically in an arbitrary manner. Links of the network vary timely and are based on the proximity of one node to another node.

iii. Device discovery: Identifying relevant newly moved in nodes and informing about

iv. Bandwidth optimization: Wireless links have significantly lower capacity than the wired links. Routing protocols in wireless networks always use the bandwidth in an optimal manner by keeping the overhead as low as possible. The limited transmission range also imposes a constraint on routing protocols in maintaining the topological information. Especially in MANETS due to frequent changes in topology, maintaining the topological information at all nodes involves more control overhead which, in turn, results in more bandwidth wastage.

v. Limited resources: Mobile nodes rely on battery power, which is a scarce resource. Also storage capacity and power are severely limited.

vi. Scalability: Scalability can be broadly defined as whether the network is able to provide an acceptable level of service even in the presence of a large number of nodes.

vii. Limited physical security: Mobility implies higher security risks such as peer-to-peer network architecture or a shared wireless medium accessible to both legitimate network users and malicious attackers. Eavesdropping, spoofing and denial-of-service attacks should be considered.

viii. Infrastructure-less and self-operated: Self healing feature demands MANET should realign itself to blanket any node moving out of its range.

ix. Poor Transmission Quality: This is an inherent problem of wireless communication caused by several error sources that result in degradation of the received signal.

x. Ad hoc addressing: Challenges in standard addressing scheme to be implemented.

xi. Network configuration: The whole MANET infrastructure is dynamic and is the reason for dynamic connection and disconnection of the variable links.

xii. Topology maintenance: Updating information of dynamic links among nodes in MANETs is a major challenge.

# CHAPTER 3: SYSTEM DEVELOPMENT

## 3.1 Introduction

In communication and computer network research, network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/packets, etc.) using mathematical formulas, or actually capturing and playing back observations from a production network. The behaviour of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions.

A network simulator is software that predicts the behavior of a computer network. Since communication Networks have become too complex for traditional analytical methods to provide an accurate understanding of system behavior, network simulators are used. In simulators, the computer network is typically modeled with devices, links, applications etc. and the performance is analysed. Simulators typically come with support for the most popular technologies and networks in use today.

Ns (network simulator) is a name for a series of discrete event network simulators, specifically ns-1, ns-2 and ns-3. All of them are discrete-event computer network simulators, primarily used in research and teaching. Ns-3(Current) is a free software, publicly available under the GNU GPLv2 license for research, development, and use.

The goal of the ns-3 project is to create an open simulation environment for computer networking research that will be preferred inside the research community:

i. It should be aligned with the simulation needs of modern networking research.
ii. It should encourage community contribution, peer review, and validation of the software.

Since the process of creation of a network simulator that contains a sufficient number of high-quality validated, tested and actively maintained models requires a lot of work, ns-3 project spreads this workload over a large community of users and developers

### 3.1.1 Network Simulator 1

The first version of ns, known as ns-1, was developed at Lawrence Berkeley National Laboratory (LBNL) in the 1995-97 timeframe by Steve McCanne, Sally Floyd, Kevin Fall, and other contributors. This was known as the LBNL Network Simulator, and derived from an earlier simulator known as REAL by S. Keshav. The core of the simulator was written in C++, with Tcl-based scripting of simulation scenarios.

### 3.1.2 Network Simulator 2

Ns began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. Ns has always included substantial contributions from other researchers, including wireless code from the UCB Daedalus and CMU Monarch projects and Sun Microsystems. For documentation on recent changes, see the version 2 change log.

### 3.1.3Network Simulator 3

A team led by Tom Henderson, George Riley, Sally Floyd, and Sumit Roy, applied for and received funding from the U.S. National Science Foundation (NSF) to build a replacement for ns-2, called ns-3. This team collaborated with the Planete project of INRIA at Sophia Antipolis, with Mathieu Lacage as the software lead, and formed a new open source project.

In the process of developing ns-3, it was decided to completely abandon backward-compatibility with ns-2. The new simulator would be written from scratch, using the C++ programming language. Development of ns-3 began in July 2006.

Ns-3 made its twenty sixth release (ns-3.26) in October 2016. (Which we have used for simulation of this Project)

Current status of the three versions is:

i. Ns-1 is no longer developed nor maintained.
ii. Ns-2 is not actively maintained (active development stopped in 2010. Not accepted for publications).
iii. Ns-3 is actively developed (but not compatible for work done on ns-2).

## 3.2 NETWORK SIMULATION

We have used Network Simulator 3(ver. 3.26) for carrying out the simulations of this Project. It is the latest simulator having its last stable version release in October 2016. Most of the studies have seen the use Ns-2 for comparison of Routing protocols in MANET. Keeping in mind the future prospects and also that Ns-2 is not being maintained we chose Ns-3 for carrying out the simulations. NS3 is open source discrete-event network simulator and improves simulation credibility. NS3 is not backward compatible with NS2, built from the scratch to replace NS2. NS2 and NS3 are both written with the help of C++ but NS3 does not support NS2 APIs. Some models have been ported from NS2 to NS3. NS3 is written in C++, with optional python bindings. NetAnim is GUI based network simulator used for NS3. It is stand-alone program that uses XML trace files to display the simulation graphically. It is based on Qt4 GUI toolkit.

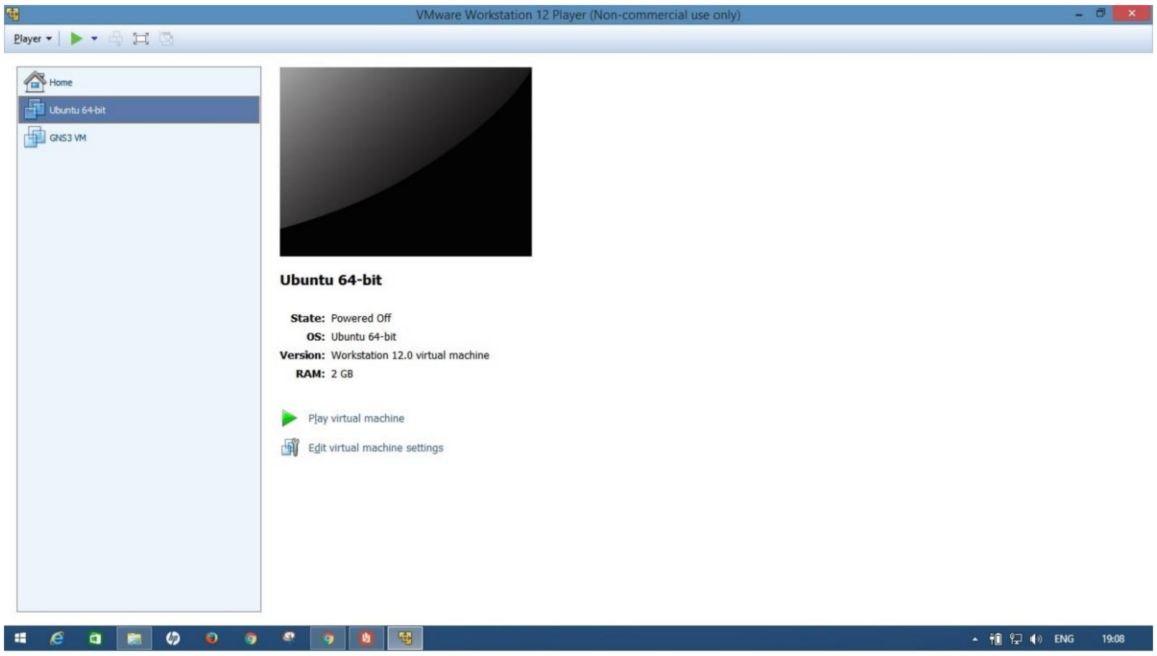
Since Network Simulators are created primarily on Linux platform the minimum requirement to run are a gcc or clang compiler.

Support:

Linux x86 and Linux x86_64 bit

Free BSD x86 and x86_64 bit

Since we intended on using ns-3 as it is the most recent simulator, and were forced by circumstances to use a Windows platform, we used VM workstation 12 player (Latest) for virtualization of a popular Linux platform i.e. Ubuntu 14.10 Desktop LTE. We dedicated 2 GB ram, with 15 GB of Hard disk space and single processor core to the virtual machine.


VMware Workstation 12 Player (Non-commercial use only)
Player
Home
Ubuntu 64-bit
GNS3 VM
Play virtual machine
Edit virtual machine settings
ENG
19:08

Ubuntu 64-bit

State: Powered Off
OS: Ubuntu 64-bit
Version: Workstation 12.0 virtual machine
RAM: 2 GB

Figure 14. Ubuntu 64bit Virtual machine

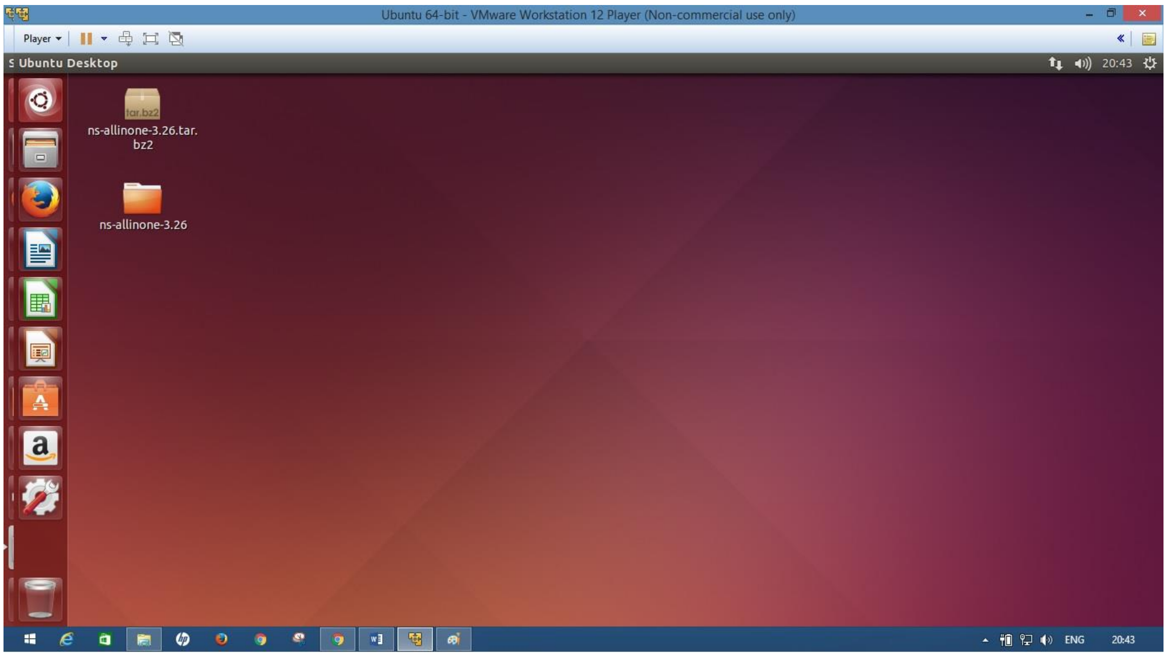


Figure 15. Ubuntu 14.10 (64bit)

### 3.2.1 Pre- Requisites

List of packages we need to install before proceeding with installation of the Network Simulator, (Specific to Ubuntu 14.10).

i. In the tarball version of Ns3, in order to be able to work with cpp we need:

```
“apt-get install gcc g++ python”
```

ii. Since we are using the tarball version, packages must for python usage:

```
“apt-get install gcc g++ python python-dev”
```

iii. Qt tools are a must for using Network Animator:

```
“apt-get install qt4-dev-tools libqt4-dev”
```

iv. To work with Ns3 development repositories, Mercurial would be needed:

```
“apt-get install mercurial”
```

v. To run Python Bindings:

```
“apt-get install bzr”
```

vi. To generate bindings in python:

```
“apt-get install cmake libc6-dev libc6-dev-i386 g++-multilib”
```

vii. When in need of debugging:

```
“apt-get install gdb valgrind”
```

viii. For accurate wifi error modelling a GNU scientific Library:

```
“apt-get install gsl-bin libgsl0-dev libgsl0ldbl”
```

ix. Flex and Bison Analyser as required by Network Simulation Cradle (NSC):

```
“apt-get install flex bison libfl-dev”
```

x. For the reading of traces:

```
"apt-get install tcpdump"
```

xi. Stats Framework's Database support:

```
"apt-get install sqlite sqlite3 libsqlite3-dev"
```

xii. Configuration store based on Xml

```
"apt-get install libxml2 libxml2-dev"
```

xiii. Configuration system based on GTK

```
"apt-get install libgtk2.0-0 libgtk2.0-dev"
```

xiv. In order to be able to experiment with Virtual Machines

```
"apt-get install vtun lxc"
```

xv. Utility Support and Style Check:

```
"apt-get install uncrustify"
```

xvi. Documentation related to Doxygen:

```
"apt-get install doxygen graphviz imagemagick"
"apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils
texlive-lang-portuguese dvipng"
```

xvii. Tutorials and user manual support using Sphinx:

```
"apt-get install python-sphinx dia"
```

**Note:** Sphinx version >= 1.12 required for ns-3.15.

xviii. Visualizer’s Support Package

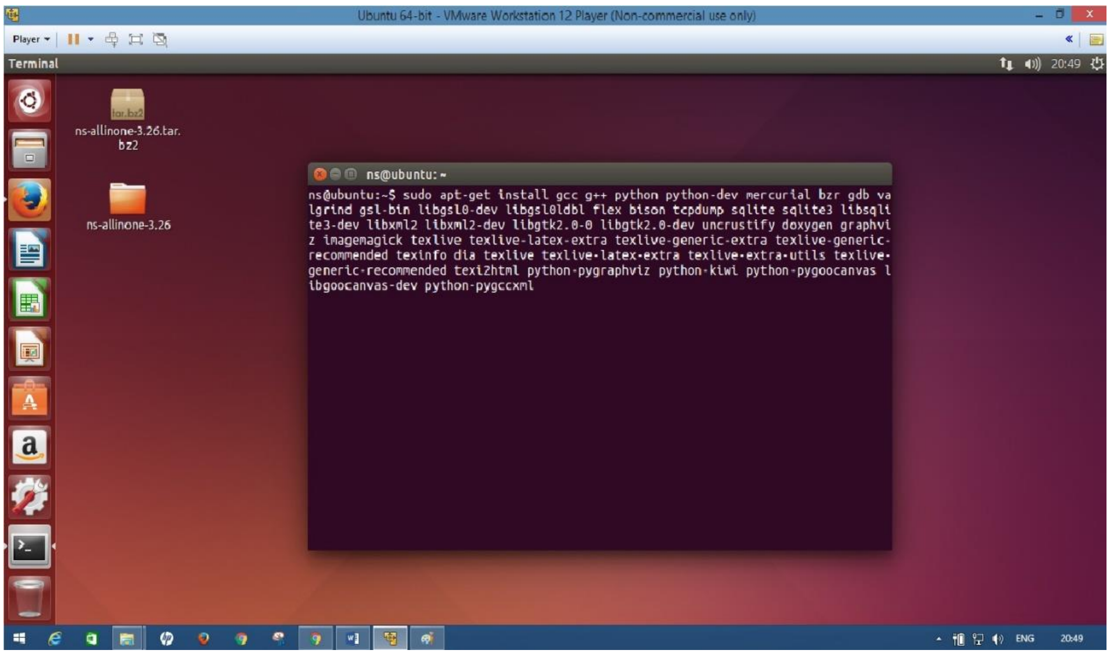
```
“apt-get install python-pygraphviz python-kiwi python-pygoocanvas
libgoocanvas-dev ipython”
```

xix. Openflow module requires support system:

```
“apt-get install libboost-signals-dev libboost-filesystem-dev”
```

xx. Distribution emulation:

```
“apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-de”
```



```
ns@ubuntu:~$ sudo apt-get install gcc g++ python python-dev mercurial bzr gdb va
lgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqli
te3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphvi
z imagemagick texlive texlive-latex-extra texlive-generic-extra texlive-generic-
recommended texinfo dia texlive texlive-latex-extra texlive-extra-utils texlive-
generic-recommended texi2html python-pygraphviz python-kiwi python-pygoocanvas l
ibgoocanvas-dev python-pygccxml
```

Figure 16. Ubuntu Packages Install

### 3.2.2 Installing Ns-3.26

i. How to download Network Simulator on our PC

   We are using a tarball version/edition of the Network Simulator, it is basically a format of software archive, and has a bunch of files and examples bundled together for the ease of users. The procedure to get the tarball version of the Netwok Simulator is pretty easy. All we had to do is pick a release preferable the most recent stable release and download it.

ii. Building of Network Simulator :

   When we use the tarball edition to build our system of Network Simulator, we can actually go and use a convenient way of building and installing of libraries.
   This program referred to as "build.py" can easily be found in the all in one directory, which is present in the tarball edition of Network Simulator. The program build.py would get the simulator configured and we can then use examples and tests to check whether our Simulator has been built and installed successfully or not.

   "./build.py --enable-examples --enable-tests"

   It becomes important since we are working with examples and tests to check successful building of the ns3, the argument "build.py" would by default build the arguments of the examples and the test cases for us. We also have an option of omitting out every example and test cases and make use of only the custom builds that are needed by the project.
   Building of modules that have dependencies outside the scope of our library would not be build using the argument "build.py".

iii. Testing Network simulator 3:

We would be running the script for testing a successful build:

“./test.py -c core”

We run these tests in parallel and look for the report saying:

“92 of 92 tests passed (92 passed, 0 failed, 0 crashed, 0 valgrind errors)”

This is the message we look for.

Figure 17. Ns-3

## 3.3 Development

We have compared all the four protocols (AODV, DSDV, OLSR, DSR) using a random waypoint mobility module which enables the nodes created in our simulation to move arbitrarily. This model is the same for all the protocols. We have made ten sources and sinks pairs, communicating UDP data. To create a real life delay and also giving proper time for initiation of the network, we have set a delay of 100 seconds, and the transmission starts at any random point between 100th and 101th second. UDP data packets are being sent at the rate of 2048 Kbps. The simulation goes up to 200 seconds. During the entire simulation the nodes are moving constantly at a speed of 20m/s with no pause time. The simulation network MANET is a region of 300x1500 m.

The transmission power we have set to be 7.5 dBm, making it be near to the real world. We also have experimented with the mobility and the density of the network with varying speeds of nodes and number of nodes. Increase of power also effects a network majorly as with the increase of power the mobility’s impact is lowered and effects of density are observed rising.

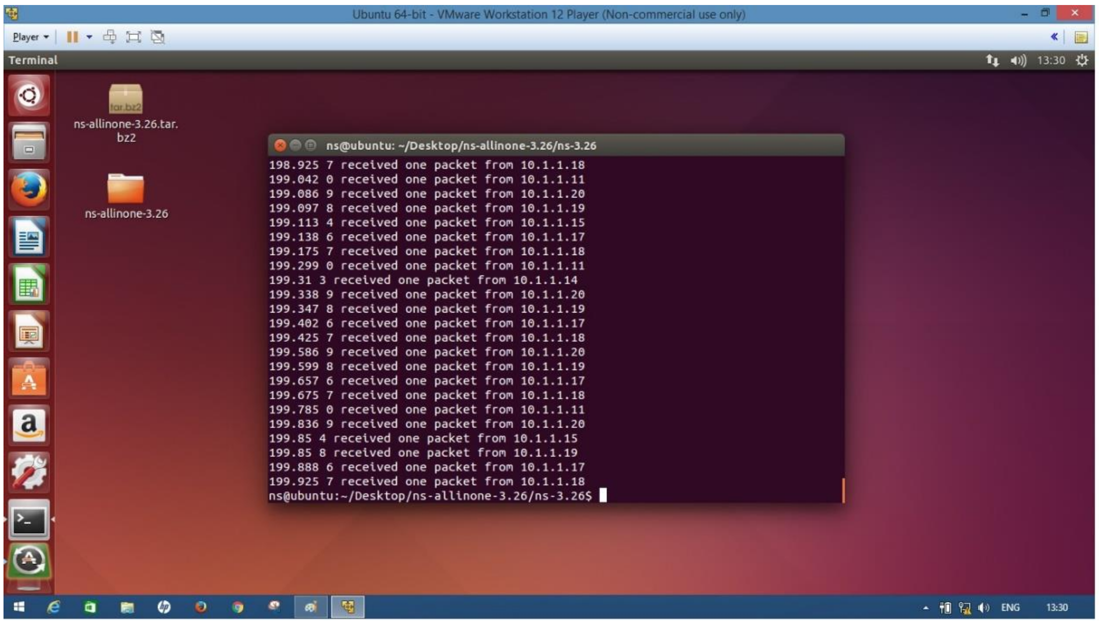
By specifying a value of 1 Optimized link state routing protocol is simulated in the created environment, similarly if we specify a value of 2 in our code Ad hoc distance vector routing protocol is used, with 3 we would have Distance sequence distance vector routing protocol simulated and with a value 4 we will have Distance sequence routing protocol simulated

Output of the program:

The output notifies of the reception of packets in its standard output.

"<timestamp> <node-id> received one packet from <src-address>"

Every second's stats of receiving any packets are tabulated in the output as a coma separated value file (.csv)


Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)
Player
Terminal
13:30
ns-allinone-3.26.tar.bz2
ns-allinone-3.26
ns@ubuntu: ~/Desktop/ns-allinone-3.26/ns-3.26
ENG
13:30

```
198.925 7 received one packet from 10.1.1.18
199.042 0 received one packet from 10.1.1.11
199.086 9 received one packet from 10.1.1.20
199.097 8 received one packet from 10.1.1.19
199.113 4 received one packet from 10.1.1.15
199.138 6 received one packet from 10.1.1.17
199.175 7 received one packet from 10.1.1.18
199.299 0 received one packet from 10.1.1.11
199.31 3 received one packet from 10.1.1.14
199.338 9 received one packet from 10.1.1.20
199.347 8 received one packet from 10.1.1.19
199.402 6 received one packet from 10.1.1.17
199.425 7 received one packet from 10.1.1.18
199.586 9 received one packet from 10.1.1.20
199.599 8 received one packet from 10.1.1.19
199.657 6 received one packet from 10.1.1.17
199.675 7 received one packet from 10.1.1.18
199.785 0 received one packet from 10.1.1.11
199.836 9 received one packet from 10.1.1.20
199.85 4 received one packet from 10.1.1.15
199.85 8 received one packet from 10.1.1.19
199.888 6 received one packet from 10.1.1.17
199.925 7 received one packet from 10.1.1.18
ns@ubuntu:~/Desktop/ns-allinone-3.26/ns-3.26$
```

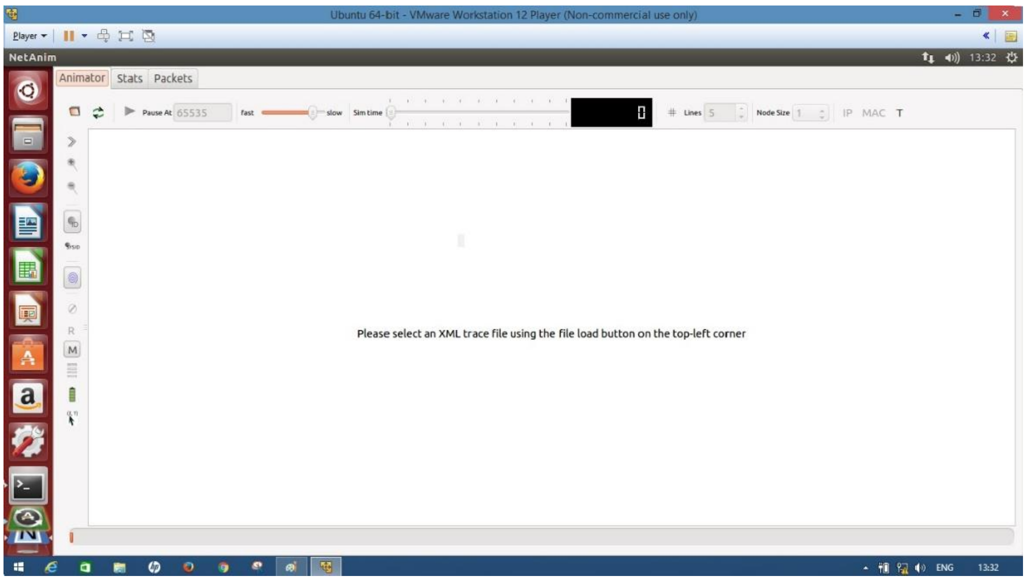
Figure 18. Ns3 Output

## 3.3.1 NetAnim

Network Animator based on qt tools is graphical animator of network simulations created using the network simulators (here Ns 3.26). It makes use of XML trace files for generaton of graphic simulation.

Features of NetAnim:

i. Animate packets over wired-links and wireless-links (Limited support for LTE traces. No support for Ipv6)
ii. Packet timeline with regex filter on packet meta-data.
iii. Node position statistics with node trajectory plotting (path of a mobile node).
iv. Print brief packet-meta data on packets
v. Use custom icons for nodes
vi. Parse flow-monitor XML files and display statistics for each flow.
vii. Show IP and MAC information, including peer IP and MAC for point-to-point links.
viii. Display double or uint32 valued counters vs time for multiple nodes in a chart or a table.
ix. Step through a simulation one event at a time and pause the simulation at a given time
x. Print the routing table at nodes at various points in time



Please select an XML trace file using the file load button on the top-left corner

Figure 19. NetAnim

Pre – Requisite:

Network Animator requires the following packages:

i. Mercurial:
   “apt-get install mercurial”
ii. Qt development tools:
   “apt-get install qt4-dev-tools”

Building Animator and Starting:

Network Animator uses qt tools, “make”.for building.

```
“cd netanim”
“make clean”
“qmake NetAnim.pro”
“make”
“./NetAnim”
```

Using NetAnim:

Using the Animator is a process having two steps:

Step 1: “Generate the animation XML trace file during simulation”
Step 2: “Load the XML trace file generated in Step 1 with the offline animator (NetAnim)”.
For step two we use “AnimationInterface”

The animator NetAnim would need a custom based trace file for the generation of any animation. This is achieved using “AnimationInterface” in the version of Ns we are using.

i. Models are present at “src/netanim/model”
ii. Examples are present at “src/netanim/examples”

Set of steps to be followed

There are a few steps which are recommended for generating any XML trace file, remember to add this before running of any simulation script.

**NOTE:** It is important for a node to have a mobility module associated.

i. Ensuring of wscript: “src/netanim/examples/wscript”
ii. Header “#include "ns3/netanim-module.h"”
iii. "AnimationInterface anim ("animation.xml")"
iv. “anim.SetMobilityPollInterval (Seconds (1))”
v. “anim.SetConstantPosition (Ptr< Node > n, double x, double y)”
vi. “anim.EnablePacketMetadata (true)”

### 3.3.2 Performance Metrics

There are various performance matrices to evaluate the routing protocols for MANET simulation but here we discuss the following metrics.

i. Throughput: It measures how well the network can constantly provide data to the destination. It is derived in Mbps. For achieving better performance it should be high.

$$Throughput = \begin{pmatrix} \text{Re}\,ceivedbytes \\ *8 \end{pmatrix} \div \begin{pmatrix} Simulationtime\,* \\ 1024*1024 \end{pmatrix}$$

ii. Packet Delivery Ratio: The ratio of the number of data packets delivered to the destination nodes and the number of data packets sent by source nodes. The performance would be better when it is high.

$$Packetdeliveryratio = (\text{Re}\,ceivedpacket\,/\,Sentpacket)*100\%$$

iii. End to end delay: The average time interval between the generation of packets in a source node and successfully delivery of it in a destination node. The performance would be better when it is low

$$Endtoenddelay = Delaysum\,/\,\text{Re}\,ceivedpackets$$

iv. Number of Packets dropped: The number of data packets that is not successfully delivered to the destination during transmission.

v. Jitter: It describes standard deviation of packet delay between all nodes

# CHAPTER 4 PERFORMANCE ANALYSIS

## 4.1 Simulation Analysis

We have compared all the four protocols (AODV, DSDV, OLSR, DSR) using a random waypoint mobility module which enables the nodes created in our simulation to move arbitrarily. This model is the same for all the protocols. We have made ten sources and sinks pairs, communicating UDP data. To create a real life delay and also giving proper time for initiation of the network, we have set a delay of 100 seconds, and the transmission starts at any random point between 100th and 101th second. UDP data packets are being sent at the rate of 2048 Kbps. The simulation goes up to 200 seconds. During the entire simulation the nodes are moving constantly at a speed of 20m/s with no pause time. The simulation network MANET is a region of 300x1500 m.

The transmission power we have set to be 7.5 dBm, making it be near to the real world. We also have experimented with the mobility and the density of the network with varying speeds of nodes and number of nodes. Increase of power also effects a network majorly as with the increase of power the mobility’s impact is lowered and effects of density are observed rising.

### 4.1.1 Simulation Using NetAnim

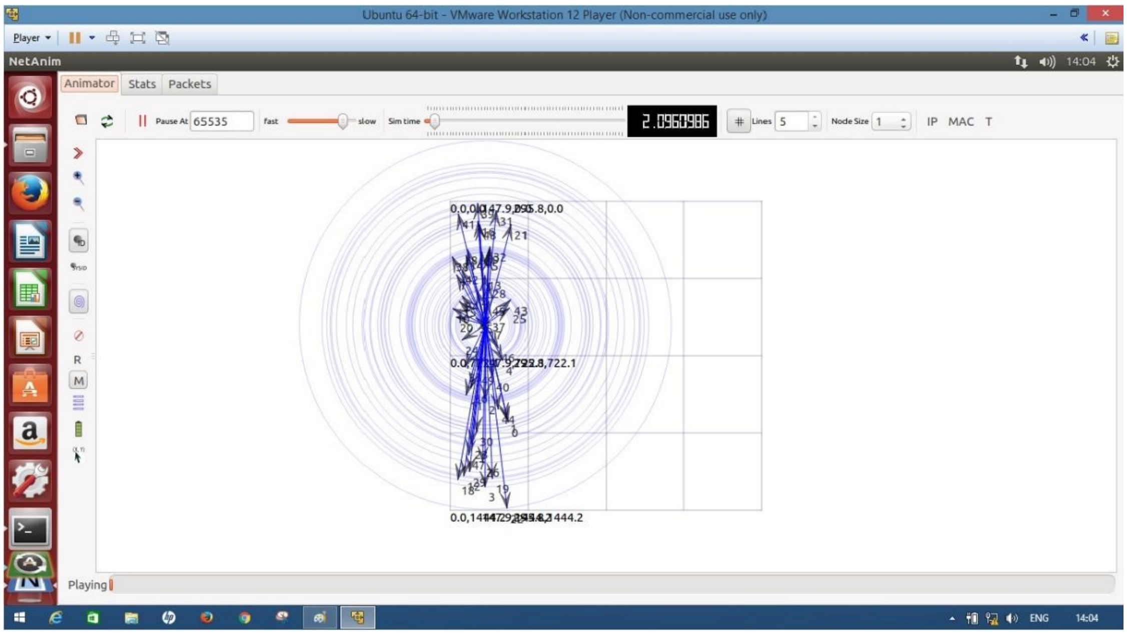


Figure 20. OLSR Simulation

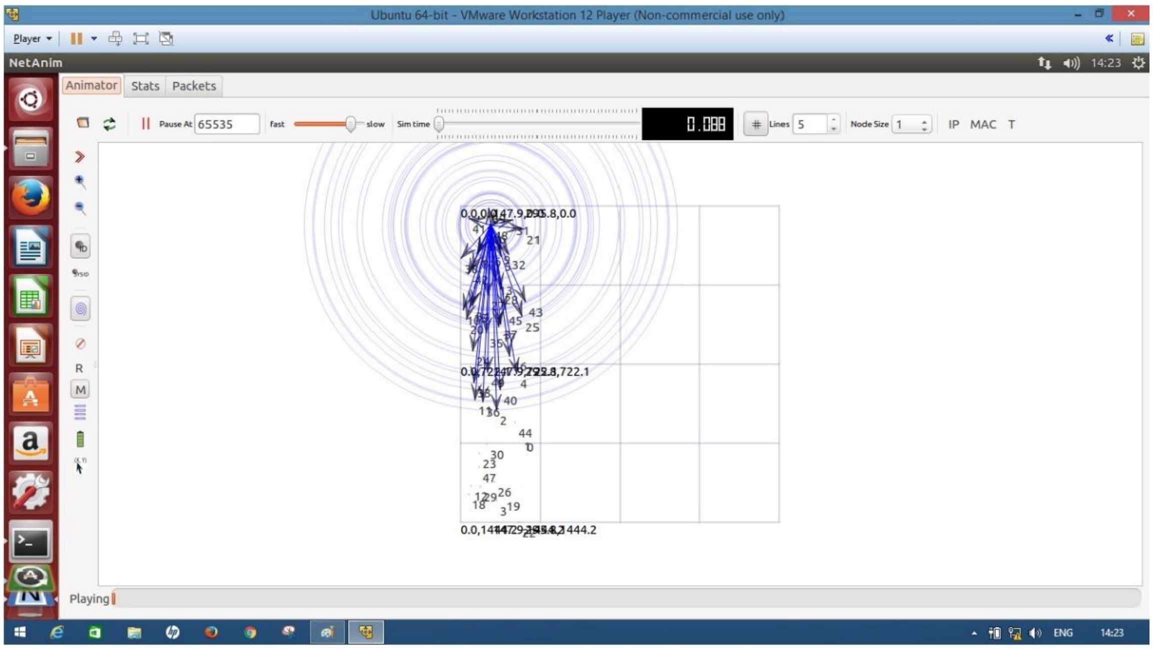


Figure 21. AODV Simulation

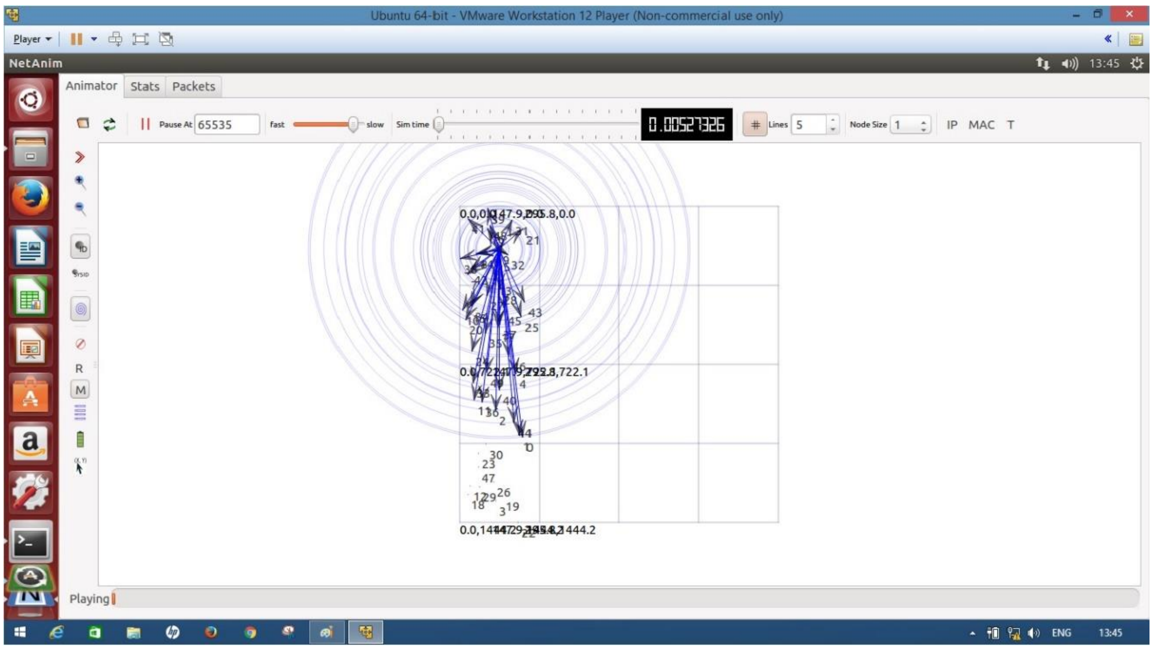


Figure 22. DSR Simulation

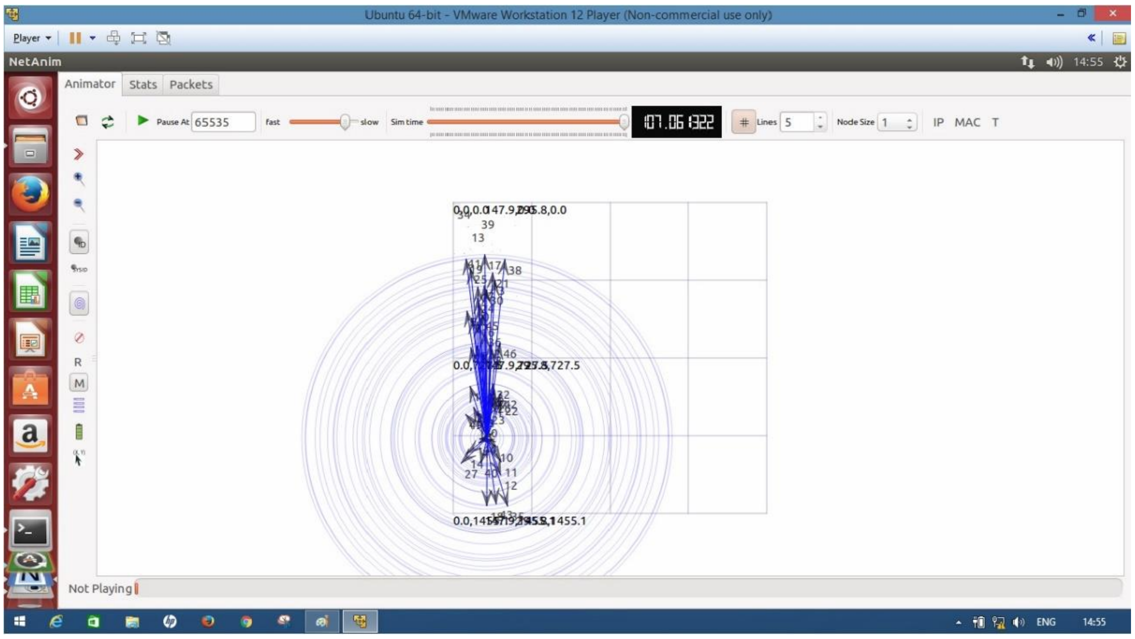


Figure 23. DSDV Simulation

## 4.2 Routing Protocol Analysis

### 4.2.1 OLSR

Optimized Link State Routing protocol is basically an Internet protocol which modified to cater to the needs of a MANET, therefore, it can be used in other wireless networks as well. It is a table driven (proactive) protocol based on link state routing algorithm. This protocol makes use of hello and topology control messages to basically find and discover the route first and then distribute the link state information in the entire MANET. It utilizes the TC to identify the next hop for a packet being sent from source to the destination.

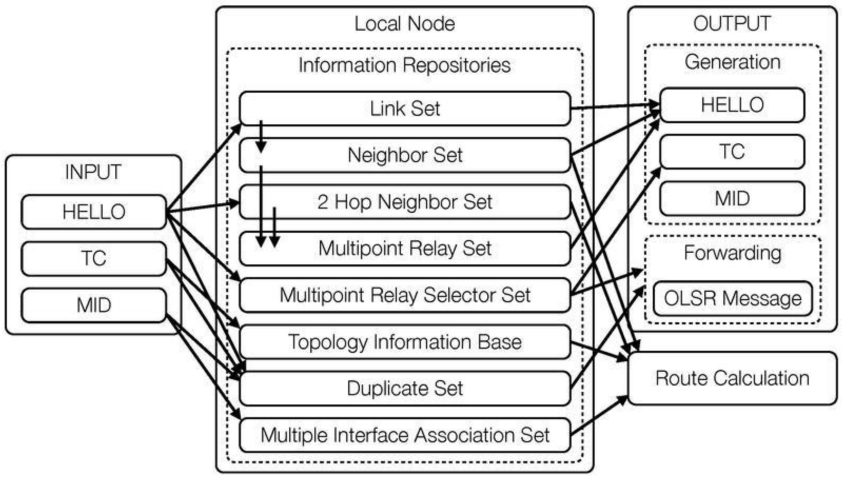
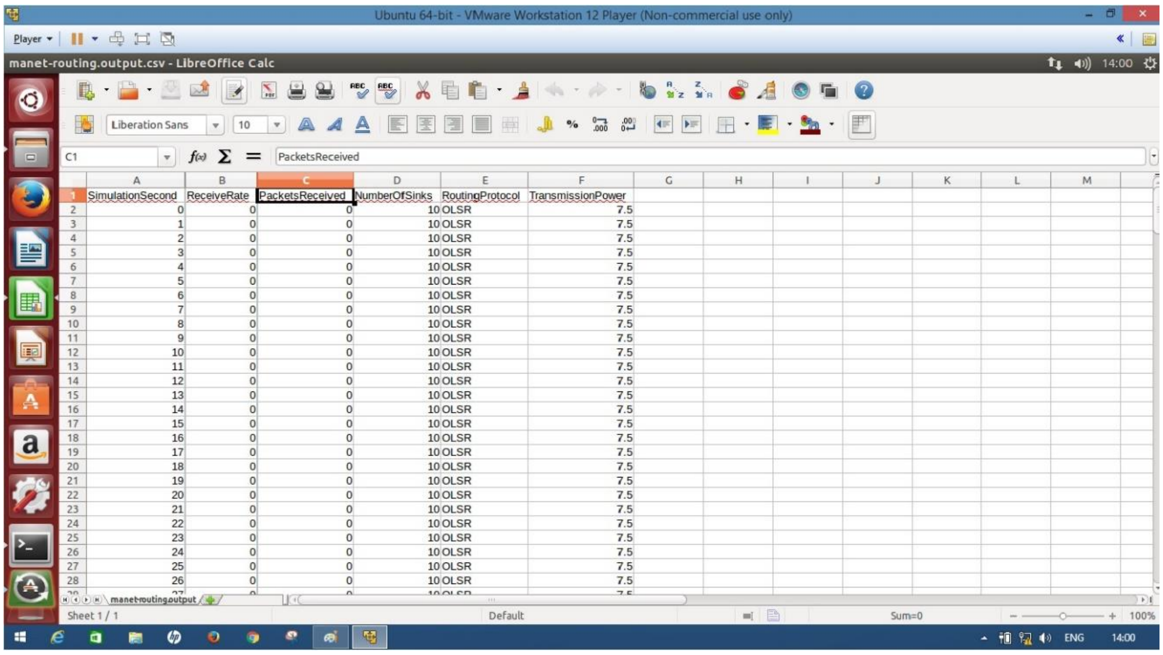


Figure 24. OLSR Characteristic Flow

Output:



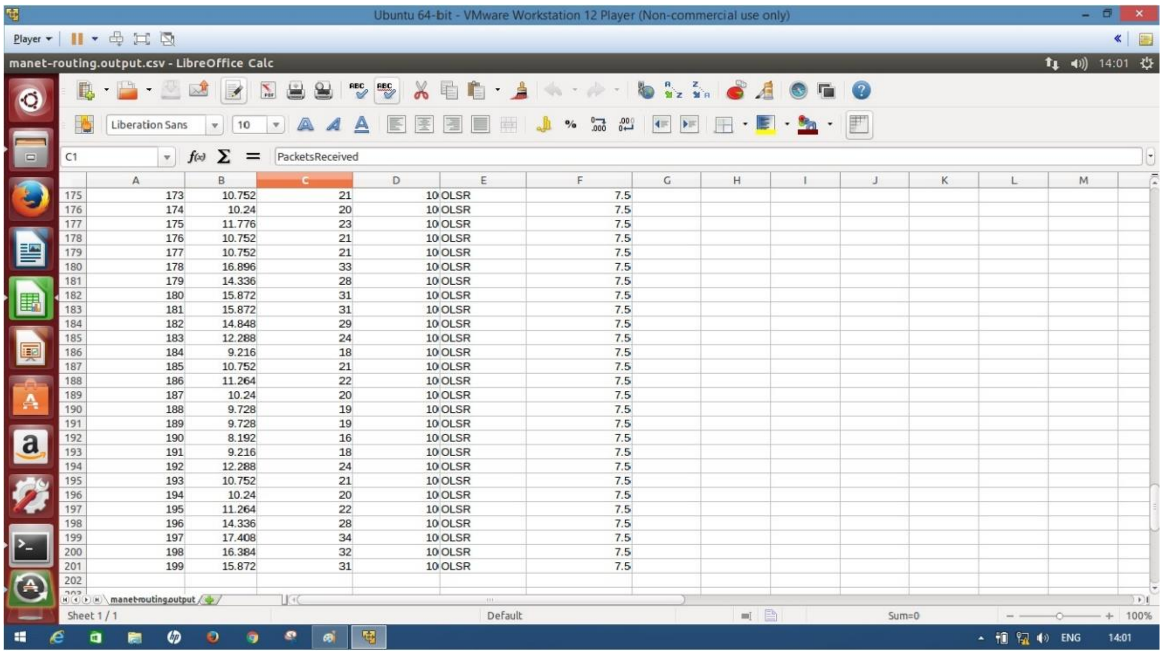
manet-routing.output.csv - LibreOffice Calc

C1 PacketsReceived

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | SimulationSecond | ReceiveRate | PacketsReceived | NumberOfSinks | RoutingProtocol | TransmissionPower |
| 2 | 0 | 0 | 0 | 10 | OLSR | 7.5 |
| 3 | 1 | 0 | 0 | 10 | OLSR | 7.5 |
| 4 | 2 | 0 | 0 | 10 | OLSR | 7.5 |
| 5 | 3 | 0 | 0 | 10 | OLSR | 7.5 |
| 6 | 4 | 0 | 0 | 10 | OLSR | 7.5 |
| 7 | 5 | 0 | 0 | 10 | OLSR | 7.5 |
| 8 | 6 | 0 | 0 | 10 | OLSR | 7.5 |
| 9 | 7 | 0 | 0 | 10 | OLSR | 7.5 |
| 10 | 8 | 0 | 0 | 10 | OLSR | 7.5 |
| 11 | 9 | 0 | 0 | 10 | OLSR | 7.5 |
| 12 | 10 | 0 | 0 | 10 | OLSR | 7.5 |
| 13 | 11 | 0 | 0 | 10 | OLSR | 7.5 |
| 14 | 12 | 0 | 0 | 10 | OLSR | 7.5 |
| 15 | 13 | 0 | 0 | 10 | OLSR | 7.5 |
| 16 | 14 | 0 | 0 | 10 | OLSR | 7.5 |
| 17 | 15 | 0 | 0 | 10 | OLSR | 7.5 |
| 18 | 16 | 0 | 0 | 10 | OLSR | 7.5 |
| 19 | 17 | 0 | 0 | 10 | OLSR | 7.5 |
| 20 | 18 | 0 | 0 | 10 | OLSR | 7.5 |
| 21 | 19 | 0 | 0 | 10 | OLSR | 7.5 |
| 22 | 20 | 0 | 0 | 10 | OLSR | 7.5 |
| 23 | 21 | 0 | 0 | 10 | OLSR | 7.5 |
| 24 | 22 | 0 | 0 | 10 | OLSR | 7.5 |
| 25 | 23 | 0 | 0 | 10 | OLSR | 7.5 |
| 26 | 24 | 0 | 0 | 10 | OLSR | 7.5 |
| 27 | 25 | 0 | 0 | 10 | OLSR | 7.5 |
| 28 | 26 | 0 | 0 | 10 | OLSR | 7.5 |

Sheet 1 / 1 Default Sum=0

Figure 25. OLSR Output 1



manet-routing.output.csv - LibreOffice Calc

C1 PacketsReceived

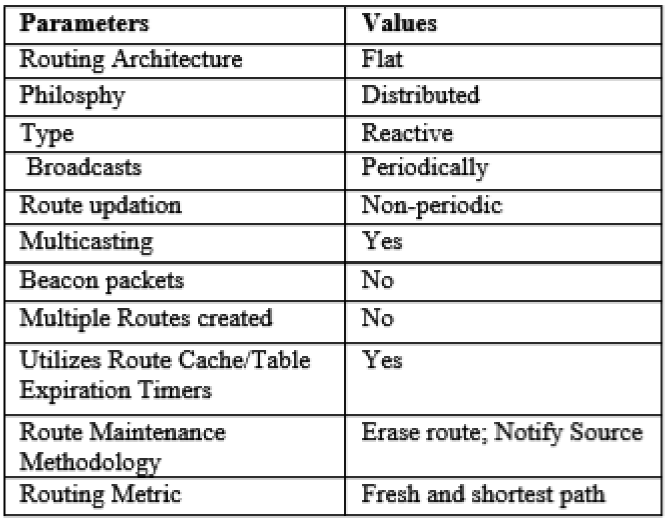
| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 175 | 173 | 10.752 | 21 | 10 | OLSR | 7.5 |
| 176 | 174 | 10.24 | 20 | 10 | OLSR | 7.5 |
| 177 | 175 | 11.776 | 23 | 10 | OLSR | 7.5 |
| 178 | 176 | 10.752 | 21 | 10 | OLSR | 7.5 |
| 179 | 177 | 10.752 | 21 | 10 | OLSR | 7.5 |
| 180 | 178 | 16.896 | 33 | 10 | OLSR | 7.5 |
| 181 | 179 | 14.336 | 28 | 10 | OLSR | 7.5 |
| 182 | 180 | 15.872 | 31 | 10 | OLSR | 7.5 |
| 183 | 181 | 15.872 | 31 | 10 | OLSR | 7.5 |
| 184 | 182 | 14.848 | 29 | 10 | OLSR | 7.5 |
| 185 | 183 | 12.288 | 24 | 10 | OLSR | 7.5 |
| 186 | 184 | 9.216 | 18 | 10 | OLSR | 7.5 |
| 187 | 185 | 10.752 | 21 | 10 | OLSR | 7.5 |
| 188 | 186 | 11.264 | 22 | 10 | OLSR | 7.5 |
| 189 | 187 | 10.24 | 20 | 10 | OLSR | 7.5 |
| 190 | 188 | 9.728 | 19 | 10 | OLSR | 7.5 |
| 191 | 189 | 9.728 | 19 | 10 | OLSR | 7.5 |
| 192 | 190 | 8.192 | 16 | 10 | OLSR | 7.5 |
| 193 | 191 | 9.216 | 18 | 10 | OLSR | 7.5 |
| 194 | 192 | 12.288 | 24 | 10 | OLSR | 7.5 |
| 195 | 193 | 10.752 | 21 | 10 | OLSR | 7.5 |
| 196 | 194 | 10.24 | 20 | 10 | OLSR | 7.5 |
| 197 | 195 | 11.264 | 22 | 10 | OLSR | 7.5 |
| 198 | 196 | 14.336 | 28 | 10 | OLSR | 7.5 |
| 199 | 197 | 17.408 | 34 | 10 | OLSR | 7.5 |
| 200 | 198 | 16.384 | 32 | 10 | OLSR | 7.5 |
| 201 | 199 | 15.872 | 31 | 10 | OLSR | 7.5 |
| 202 | | | | | | |

Sheet 1 / 1 Default Sum=0

Figure 26. OLSR Output 2

### 4.2.2 AODV

The ad hoc on demand distance vector routing (AODV) protocol is designed keeping in mind the impromptu nature of MANETs and provides efficient on demand route discovery. When there is a need of transmission of packets to any destination, the first thing that protocol accomplishes is the process of discovering routes from the source to the required destination. The procedural goes like that the source nodes starts by broadcasting a route request (RREQ) to its neighbours. All RREQ comprise of a unique broadcasting ID, sequence number (usually 2), and the address of the destination (it also includes the source address and hop count possible). The next step is similar in approach to the first one, the neighbours of the source node probably a mediator node if not the destination itself would re-broadcast the RREQ to its neighbours. The same process continues till destination is reached once that happens the destination will cast a unicast message back with route reply (RREP), this would then reach the source node performing the same process in reverse. Therefore at the end of this process usually referred to as the response cycle a bi-directional route is known between the source and the destination. It also adds a certain advantage to the maintainability of the routes, since when there is an error or loss of connectivity, the intermediate node sends a route error request to all the potential nodes where it received a RREP from, hence, a route is maintained as long as it remains active. This is an advantage which makes it more efficient than other routing protocols in a dynamically changing environment and topology.



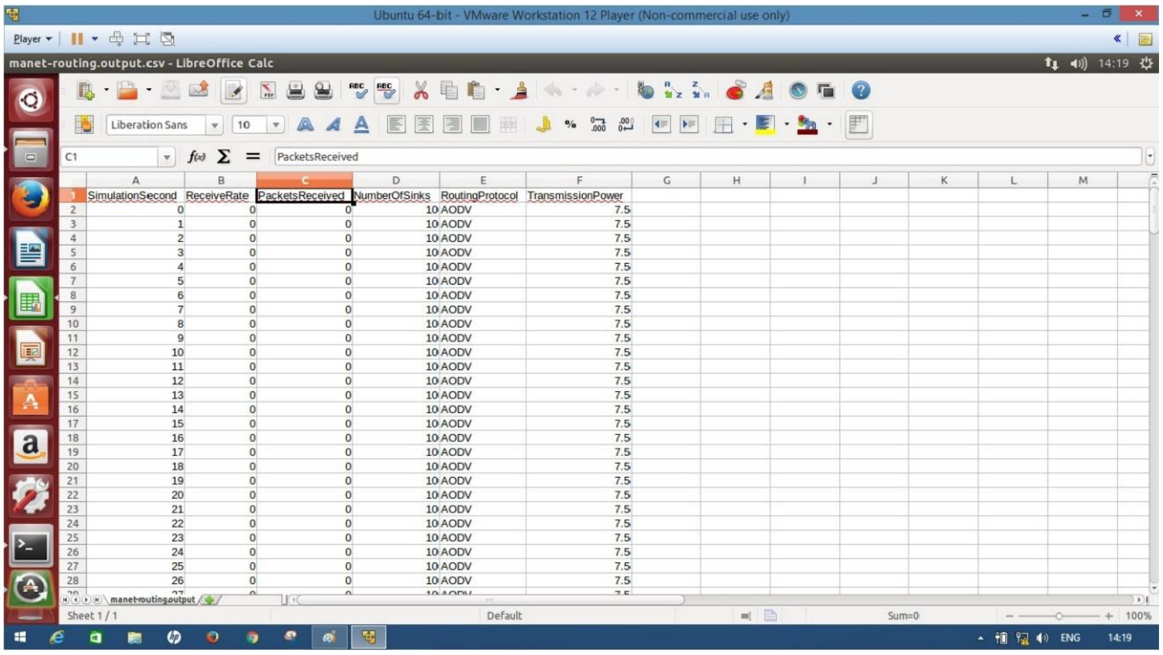
| Parameters | Values |
|---|---|
| Routing Architecture | Flat |
| Philosphy | Distributed |
| Type | Reactive |
| Broadcasts | Periodically |
| Route updation | Non-periodic |
| Multicasting | Yes |
| Beacon packets | No |
| Multiple Routes created | No |
| Utilizes Route Cache/Table Expiration Timers | Yes |
| Route Maintenance Methodology | Erase route; Notify Source |
| Routing Metric | Fresh and shortest path |

Table 4. AODV Characteristics

Snapshot of the routing output:

| Simulation | Delivery Rate | Packets rec. | Sinks | Protocol | Trans. Power |
|---|---|---|---|---|---|
| 101 | 3.072 | 6 | 10 | AODV | 7.5 |
| 102 | 14.848 | 29 | 10 | AODV | 7.5 |
| 103 | 12.288 | 24 | 10 | AODV | 7.5 |
| 104 | 14.336 | 28 | 10 | AODV | 7.5 |
| 105 | 15.36 | 30 | 10 | AODV | 7.5 |
| 106 | 14.848 | 29 | 10 | AODV | 7.5 |
| 107 | 17.92 | 35 | 10 | AODV | 7.5 |
| 108 | 15.36 | 30 | 10 | AODV | 7.5 |
| 109 | 19.456 | 38 | 10 | AODV | 7.5 |
| 110 | 18.432 | 36 | 10 | AODV | 7.5 |
| 111 | 18.432 | 36 | 10 | AODV | 7.5 |
| 112 | 13.312 | 26 | 10 | AODV | 7.5 |
| 113 | 12.8 | 25 | 10 | AODV | 7.5 |
| 114 | 13.312 | 26 | 10 | AODV | 7.5 |

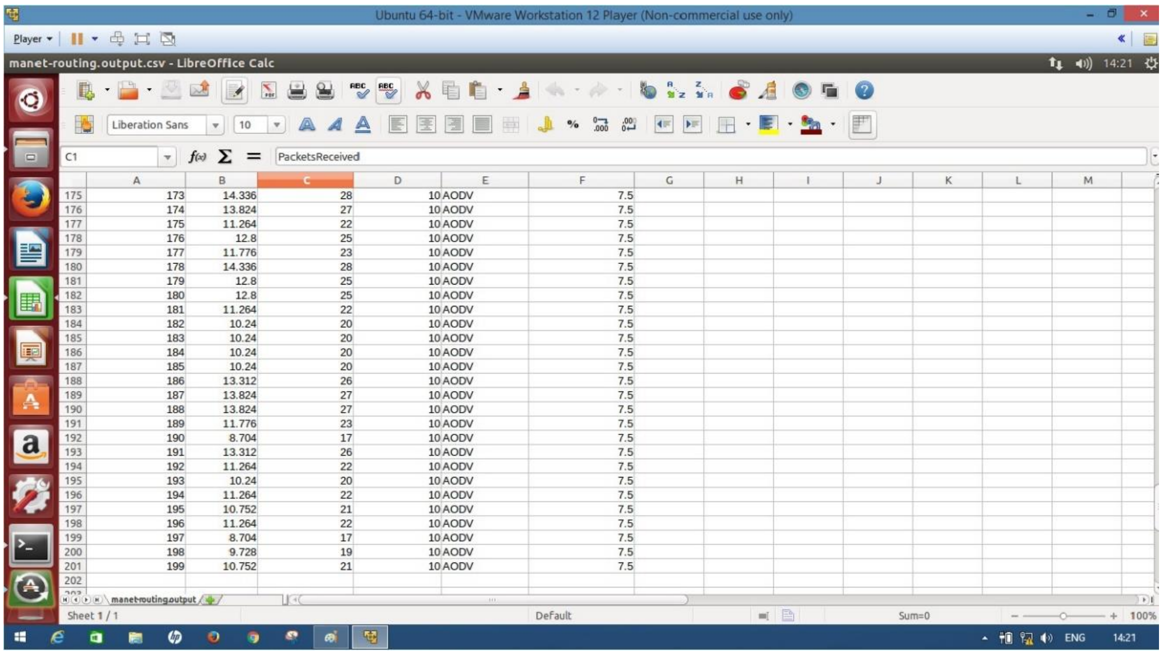
Output:



manet-routing.output.csv - LibreOffice Calc

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | SimulationSecond | ReceiveRate | PacketsReceived | NumberOfSinks | RoutingProtocol | TransmissionPower |
| 2 | 0 | 0 | 0 | 10 | AODV | 7.5 |
| 3 | 1 | 0 | 0 | 10 | AODV | 7.5 |
| 4 | 2 | 0 | 0 | 10 | AODV | 7.5 |
| 5 | 3 | 0 | 0 | 10 | AODV | 7.5 |
| 6 | 4 | 0 | 0 | 10 | AODV | 7.5 |
| 7 | 5 | 0 | 0 | 10 | AODV | 7.5 |
| 8 | 6 | 0 | 0 | 10 | AODV | 7.5 |
| 9 | 7 | 0 | 0 | 10 | AODV | 7.5 |
| 10 | 8 | 0 | 0 | 10 | AODV | 7.5 |
| 11 | 9 | 0 | 0 | 10 | AODV | 7.5 |
| 12 | 10 | 0 | 0 | 10 | AODV | 7.5 |
| 13 | 11 | 0 | 0 | 10 | AODV | 7.5 |
| 14 | 12 | 0 | 0 | 10 | AODV | 7.5 |
| 15 | 13 | 0 | 0 | 10 | AODV | 7.5 |
| 16 | 14 | 0 | 0 | 10 | AODV | 7.5 |
| 17 | 15 | 0 | 0 | 10 | AODV | 7.5 |
| 18 | 16 | 0 | 0 | 10 | AODV | 7.5 |
| 19 | 17 | 0 | 0 | 10 | AODV | 7.5 |
| 20 | 18 | 0 | 0 | 10 | AODV | 7.5 |
| 21 | 19 | 0 | 0 | 10 | AODV | 7.5 |
| 22 | 20 | 0 | 0 | 10 | AODV | 7.5 |
| 23 | 21 | 0 | 0 | 10 | AODV | 7.5 |
| 24 | 22 | 0 | 0 | 10 | AODV | 7.5 |
| 25 | 23 | 0 | 0 | 10 | AODV | 7.5 |
| 26 | 24 | 0 | 0 | 10 | AODV | 7.5 |
| 27 | 25 | 0 | 0 | 10 | AODV | 7.5 |
| 28 | 26 | 0 | 0 | 10 | AODV | 7.5 |

Figure 27. AODV Output 1



manet-routing.output.csv - LibreOffice Calc

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 175 | 173 | 14.336 | 28 | 10 | AODV | 7.5 |
| 176 | 174 | 13.824 | 27 | 10 | AODV | 7.5 |
| 177 | 175 | 11.264 | 22 | 10 | AODV | 7.5 |
| 178 | 176 | 12.8 | 25 | 10 | AODV | 7.5 |
| 179 | 177 | 11.776 | 23 | 10 | AODV | 7.5 |
| 180 | 178 | 14.336 | 28 | 10 | AODV | 7.5 |
| 181 | 179 | 12.8 | 25 | 10 | AODV | 7.5 |
| 182 | 180 | 12.8 | 25 | 10 | AODV | 7.5 |
| 183 | 181 | 11.264 | 22 | 10 | AODV | 7.5 |
| 184 | 182 | 10.24 | 20 | 10 | AODV | 7.5 |
| 185 | 183 | 10.24 | 20 | 10 | AODV | 7.5 |
| 186 | 184 | 10.24 | 20 | 10 | AODV | 7.5 |
| 187 | 185 | 10.24 | 20 | 10 | AODV | 7.5 |
| 188 | 186 | 13.312 | 26 | 10 | AODV | 7.5 |
| 189 | 187 | 13.824 | 27 | 10 | AODV | 7.5 |
| 190 | 188 | 13.824 | 27 | 10 | AODV | 7.5 |
| 191 | 189 | 11.776 | 23 | 10 | AODV | 7.5 |
| 192 | 190 | 8.704 | 17 | 10 | AODV | 7.5 |
| 193 | 191 | 13.312 | 26 | 10 | AODV | 7.5 |
| 194 | 192 | 11.264 | 22 | 10 | AODV | 7.5 |
| 195 | 193 | 10.24 | 20 | 10 | AODV | 7.5 |
| 196 | 194 | 11.264 | 22 | 10 | AODV | 7.5 |
| 197 | 195 | 10.752 | 21 | 10 | AODV | 7.5 |
| 198 | 196 | 11.264 | 22 | 10 | AODV | 7.5 |
| 199 | 197 | 8.704 | 17 | 10 | AODV | 7.5 |
| 200 | 198 | 9.728 | 19 | 10 | AODV | 7.5 |
| 201 | 199 | 10.752 | 21 | 10 | AODV | 7.5 |
| 202 | | | | | | |

Figure 28. AODV Output 2

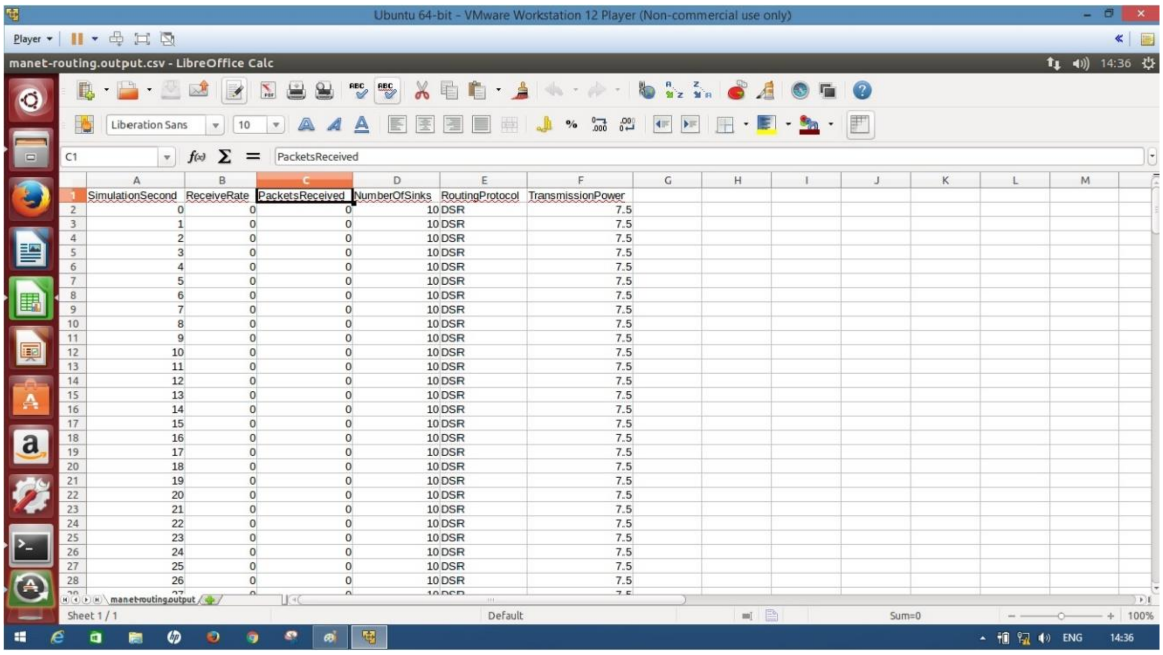
### 4.2.3 DSR

Dynamic Source Routing (DSR) is an on demand reactive protocol specifically designed to incorporate use of multi hop impromptu wireless networks in mobile nodes. Determination of the route to the destination, at every single source node, happens in this protocol. This process is used for transmission of packets to the destination. There are two main stages when using DSR. First one being Route discovery and the other Route Maintenance. Route discovery as the name suggests is the finding of the path by a source node wishing to send a packet to any node (destination). Route maintenance is detection of any breaks in the route which is determined by the source node. The sender knows the complete hop by hop route to the destination. There is a designated cache memory to store the information of these routes. One of the advantage of this protocol is that it allows the use of multiple routes to any destination and also gives the control to the source node or the sending node to choose routes in routing of the packets. “The DSR protocol is designed mainly for mobile ad hoc networks of up to about two hundred nodes, and is designed to work well with even very high rates of mobility”

Snapshot of the routing output:

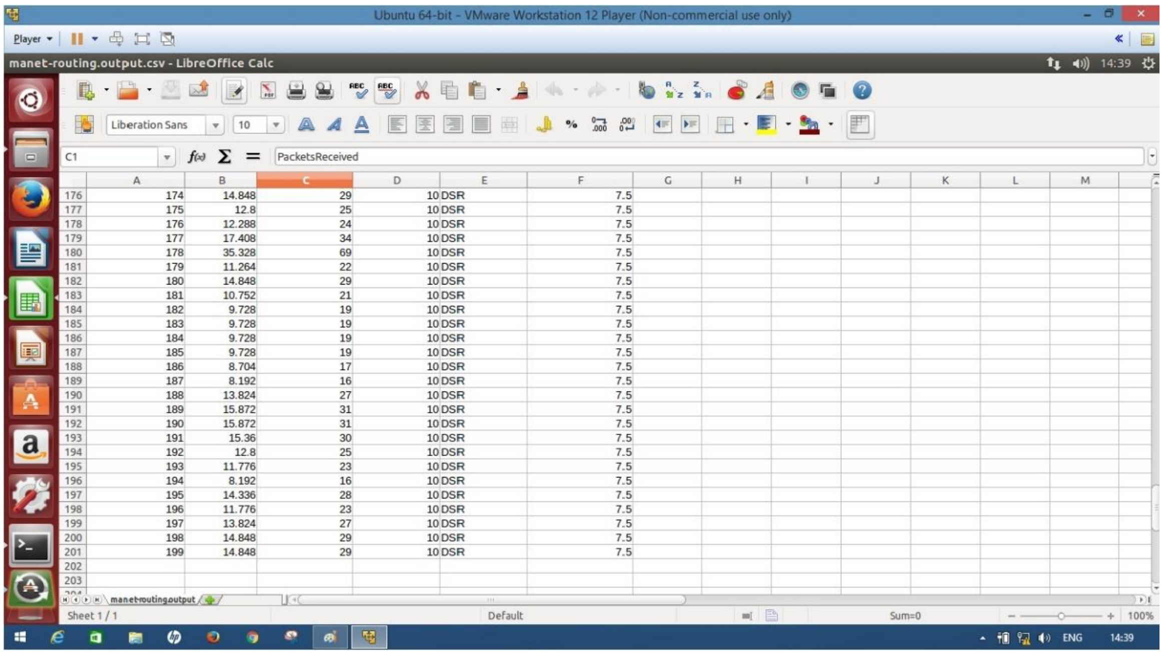
| **Simulation** | **Delivery Rate** | **Packets Rec.** | **Sinks** | **Protocol** | **Trans. Power** |
|---|---|---|---|---|---|
| 101 | 5.12 | 10 | 10 | DSR | 7.5 |
| 102 | 15.36 | 30 | 10 | DSR | 7.5 |
| 103 | 20.48 | 40 | 10 | DSR | 7.5 |
| 104 | 18.432 | 36 | 10 | DSR | 7.5 |
| 105 | 18.432 | 36 | 10 | DSR | 7.5 |
| 106 | 17.92 | 35 | 10 | DSR | 7.5 |
| 107 | 15.872 | 31 | 10 | DSR | 7.5 |
| 108 | 16.896 | 33 | 10 | DSR | 7.5 |
| 109 | 18.944 | 37 | 10 | DSR | 7.5 |
| 110 | 18.432 | 36 | 10 | DSR | 7.5 |
| 111 | 16.896 | 33 | 10 | DSR | 7.5 |
| 112 | 18.944 | 37 | 10 | DSR | 7.5 |

Output:



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | SimulationSecond | ReceiveRate | PacketsReceived | NumberOfSinks | RoutingProtocol | TransmissionPower |
| 2 | 0 | 0 | 0 | 10 | DSR | 7.5 |
| 3 | 1 | 0 | 0 | 10 | DSR | 7.5 |
| 4 | 2 | 0 | 0 | 10 | DSR | 7.5 |
| 5 | 3 | 0 | 0 | 10 | DSR | 7.5 |
| 6 | 4 | 0 | 0 | 10 | DSR | 7.5 |
| 7 | 5 | 0 | 0 | 10 | DSR | 7.5 |
| 8 | 6 | 0 | 0 | 10 | DSR | 7.5 |
| 9 | 7 | 0 | 0 | 10 | DSR | 7.5 |
| 10 | 8 | 0 | 0 | 10 | DSR | 7.5 |
| 11 | 9 | 0 | 0 | 10 | DSR | 7.5 |
| 12 | 10 | 0 | 0 | 10 | DSR | 7.5 |
| 13 | 11 | 0 | 0 | 10 | DSR | 7.5 |
| 14 | 12 | 0 | 0 | 10 | DSR | 7.5 |
| 15 | 13 | 0 | 0 | 10 | DSR | 7.5 |
| 16 | 14 | 0 | 0 | 10 | DSR | 7.5 |
| 17 | 15 | 0 | 0 | 10 | DSR | 7.5 |
| 18 | 16 | 0 | 0 | 10 | DSR | 7.5 |
| 19 | 17 | 0 | 0 | 10 | DSR | 7.5 |
| 20 | 18 | 0 | 0 | 10 | DSR | 7.5 |
| 21 | 19 | 0 | 0 | 10 | DSR | 7.5 |
| 22 | 20 | 0 | 0 | 10 | DSR | 7.5 |
| 23 | 21 | 0 | 0 | 10 | DSR | 7.5 |
| 24 | 22 | 0 | 0 | 10 | DSR | 7.5 |
| 25 | 23 | 0 | 0 | 10 | DSR | 7.5 |
| 26 | 24 | 0 | 0 | 10 | DSR | 7.5 |
| 27 | 25 | 0 | 0 | 10 | DSR | 7.5 |
| 28 | 26 | 0 | 0 | 10 | DSR | 7.5 |

Figure 29. DSR Output 1



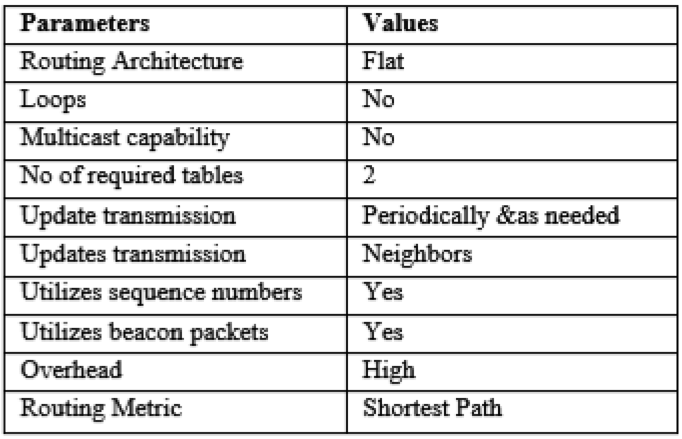
| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 176 | 174 | 14.848 | 29 | 10 | DSR | 7.5 |
| 177 | 175 | 12.8 | 25 | 10 | DSR | 7.5 |
| 178 | 176 | 12.288 | 24 | 10 | DSR | 7.5 |
| 179 | 177 | 17.408 | 34 | 10 | DSR | 7.5 |
| 180 | 178 | 35.328 | 69 | 10 | DSR | 7.5 |
| 181 | 179 | 11.264 | 22 | 10 | DSR | 7.5 |
| 182 | 180 | 14.848 | 29 | 10 | DSR | 7.5 |
| 183 | 181 | 10.752 | 21 | 10 | DSR | 7.5 |
| 184 | 182 | 9.728 | 19 | 10 | DSR | 7.5 |
| 185 | 183 | 9.728 | 19 | 10 | DSR | 7.5 |
| 186 | 184 | 9.728 | 19 | 10 | DSR | 7.5 |
| 187 | 185 | 9.728 | 19 | 10 | DSR | 7.5 |
| 188 | 186 | 8.704 | 17 | 10 | DSR | 7.5 |
| 189 | 187 | 8.192 | 16 | 10 | DSR | 7.5 |
| 190 | 188 | 13.824 | 27 | 10 | DSR | 7.5 |
| 191 | 189 | 15.872 | 31 | 10 | DSR | 7.5 |
| 192 | 190 | 15.872 | 31 | 10 | DSR | 7.5 |
| 193 | 191 | 15.36 | 30 | 10 | DSR | 7.5 |
| 194 | 192 | 12.8 | 25 | 10 | DSR | 7.5 |
| 195 | 193 | 11.776 | 23 | 10 | DSR | 7.5 |
| 196 | 194 | 8.192 | 16 | 10 | DSR | 7.5 |
| 197 | 195 | 14.336 | 28 | 10 | DSR | 7.5 |
| 198 | 196 | 11.776 | 23 | 10 | DSR | 7.5 |
| 199 | 197 | 13.824 | 27 | 10 | DSR | 7.5 |
| 200 | 198 | 14.848 | 29 | 10 | DSR | 7.5 |
| 201 | 199 | 14.848 | 29 | 10 | DSR | 7.5 |
| 202 | | | | | | |
| 203 | | | | | | |

Figure 30. DSR Output 2

### 4.2.4 DSDV

The Destination Sequence Distance Vector Protocol is basically derived from the roots of the Bellman Ford algorithm used for routing purposes. Developed in 1994 by C. Perkins along with P. Bhagwat this protocol makes use of sequence number (added attribute) on every node and every table. Being a table driven protocol it obviously uses and maintains tables having information about every destination which are available and possible to reach, within the changing network, along with the number of hops.

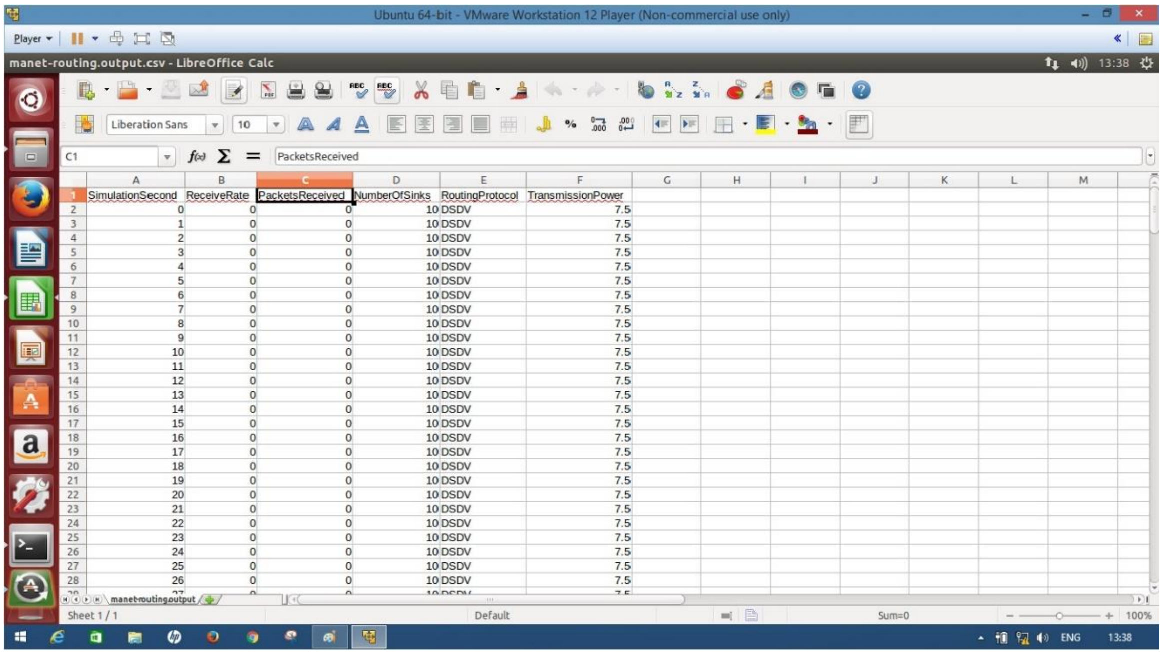
Packets are transmitted and forwarded among nodes using these tables at every node. Since every routing table holds the addresses of each node which can be reached possibly, it directs the packets on the route to reach the destination. It is possible to achieve because along with the addresses of destination node the tables also hold the information about the next hop. This protocol sees its motivated usage due to ease of data exchange along the dynamically changing nodes and randomly changing paths may not be close to the base stations.



| **Parameters** | **Values** |
|---|---|
| Routing Architecture | Flat |
| Loops | No |
| Multicast capability | No |
| No of required tables | 2 |
| Update transmission | Periodically &as needed |
| Updates transmission | Neighbors |
| Utilizes sequence numbers | Yes |
| Utilizes beacon packets | Yes |
| Overhead | High |
| Routing Metric | Shortest Path |

Table 5. DSDV Characteristics

Output:


Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)

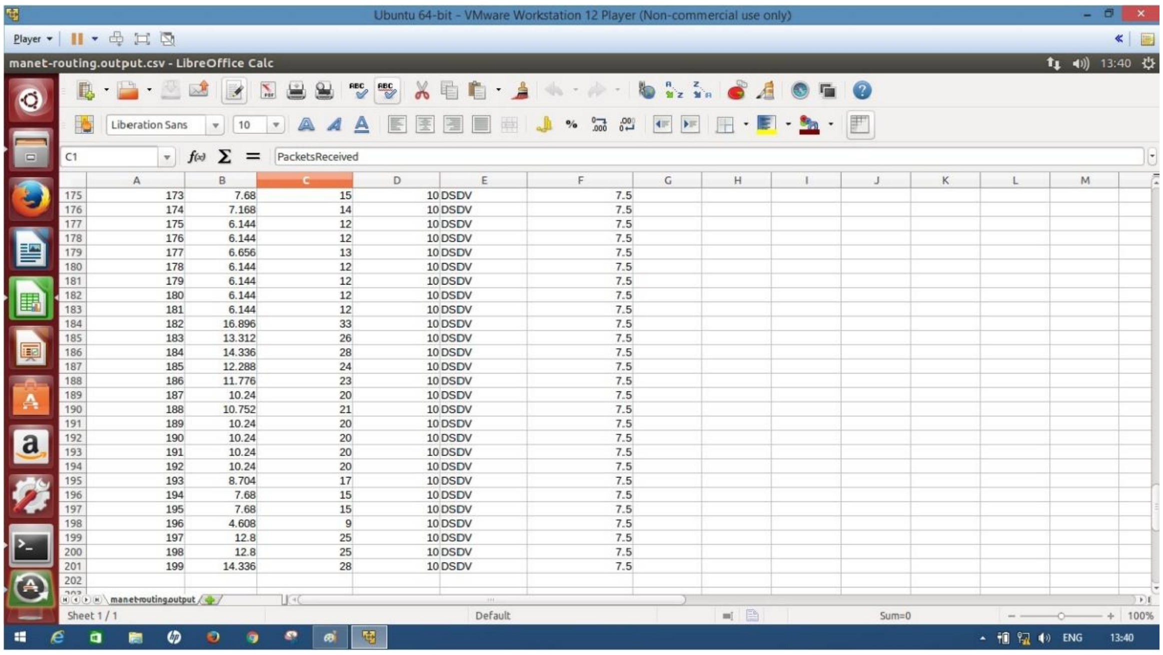
manet-routing.output.csv - LibreOffice Calc

C1 PacketsReceived

|  | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | SimulationSecond | ReceiveRate | PacketsReceived | NumberOfSinks | RoutingProtocol | TransmissionPower |
| 2 | 0 | 0 | 0 | 10 | DSDV | 7.5 |
| 3 | 1 | 0 | 0 | 10 | DSDV | 7.5 |
| 4 | 2 | 0 | 0 | 10 | DSDV | 7.5 |
| 5 | 3 | 0 | 0 | 10 | DSDV | 7.5 |
| 6 | 4 | 0 | 0 | 10 | DSDV | 7.5 |
| 7 | 5 | 0 | 0 | 10 | DSDV | 7.5 |
| 8 | 6 | 0 | 0 | 10 | DSDV | 7.5 |
| 9 | 7 | 0 | 0 | 10 | DSDV | 7.5 |
| 10 | 8 | 0 | 0 | 10 | DSDV | 7.5 |
| 11 | 9 | 0 | 0 | 10 | DSDV | 7.5 |
| 12 | 10 | 0 | 0 | 10 | DSDV | 7.5 |
| 13 | 11 | 0 | 0 | 10 | DSDV | 7.5 |
| 14 | 12 | 0 | 0 | 10 | DSDV | 7.5 |
| 15 | 13 | 0 | 0 | 10 | DSDV | 7.5 |
| 16 | 14 | 0 | 0 | 10 | DSDV | 7.5 |
| 17 | 15 | 0 | 0 | 10 | DSDV | 7.5 |
| 18 | 16 | 0 | 0 | 10 | DSDV | 7.5 |
| 19 | 17 | 0 | 0 | 10 | DSDV | 7.5 |
| 20 | 18 | 0 | 0 | 10 | DSDV | 7.5 |
| 21 | 19 | 0 | 0 | 10 | DSDV | 7.5 |
| 22 | 20 | 0 | 0 | 10 | DSDV | 7.5 |
| 23 | 21 | 0 | 0 | 10 | DSDV | 7.5 |
| 24 | 22 | 0 | 0 | 10 | DSDV | 7.5 |
| 25 | 23 | 0 | 0 | 10 | DSDV | 7.5 |
| 26 | 24 | 0 | 0 | 10 | DSDV | 7.5 |
| 27 | 25 | 0 | 0 | 10 | DSDV | 7.5 |
| 28 | 26 | 0 | 0 | 10 | DSDV | 7.5 |

Sheet 1 / 1 Default Sum=0 100%

Figure 31. DSDV Output 1


Ubuntu 64-bit - VMware Workstation 12 Player (Non-commercial use only)

manet-routing.output.csv - LibreOffice Calc

C1 PacketsReceived

|  | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 175 | 173 | 7.68 | 15 | 10 | DSDV | 7.5 |
| 176 | 174 | 7.168 | 14 | 10 | DSDV | 7.5 |
| 177 | 175 | 6.144 | 12 | 10 | DSDV | 7.5 |
| 178 | 176 | 6.144 | 12 | 10 | DSDV | 7.5 |
| 179 | 177 | 6.656 | 13 | 10 | DSDV | 7.5 |
| 180 | 178 | 6.144 | 12 | 10 | DSDV | 7.5 |
| 181 | 179 | 6.144 | 12 | 10 | DSDV | 7.5 |
| 182 | 180 | 6.144 | 12 | 10 | DSDV | 7.5 |
| 183 | 181 | 6.144 | 12 | 10 | DSDV | 7.5 |
| 184 | 182 | 16.896 | 33 | 10 | DSDV | 7.5 |
| 185 | 183 | 13.312 | 26 | 10 | DSDV | 7.5 |
| 186 | 184 | 14.336 | 28 | 10 | DSDV | 7.5 |
| 187 | 185 | 12.288 | 24 | 10 | DSDV | 7.5 |
| 188 | 186 | 11.776 | 23 | 10 | DSDV | 7.5 |
| 189 | 187 | 10.24 | 20 | 10 | DSDV | 7.5 |
| 190 | 188 | 10.752 | 21 | 10 | DSDV | 7.5 |
| 191 | 189 | 10.24 | 20 | 10 | DSDV | 7.5 |
| 192 | 190 | 10.24 | 20 | 10 | DSDV | 7.5 |
| 193 | 191 | 10.24 | 20 | 10 | DSDV | 7.5 |
| 194 | 192 | 10.24 | 20 | 10 | DSDV | 7.5 |
| 195 | 193 | 8.704 | 17 | 10 | DSDV | 7.5 |
| 196 | 194 | 7.68 | 15 | 10 | DSDV | 7.5 |
| 197 | 195 | 7.68 | 15 | 10 | DSDV | 7.5 |
| 198 | 196 | 4.608 | 9 | 10 | DSDV | 7.5 |
| 199 | 197 | 12.8 | 25 | 10 | DSDV | 7.5 |
| 200 | 198 | 12.8 | 25 | 10 | DSDV | 7.5 |
| 201 | 199 | 14.336 | 28 | 10 | DSDV | 7.5 |
| 202 |  |  |  |  |  |  |

Sheet 1 / 1 Default Sum=0 100%

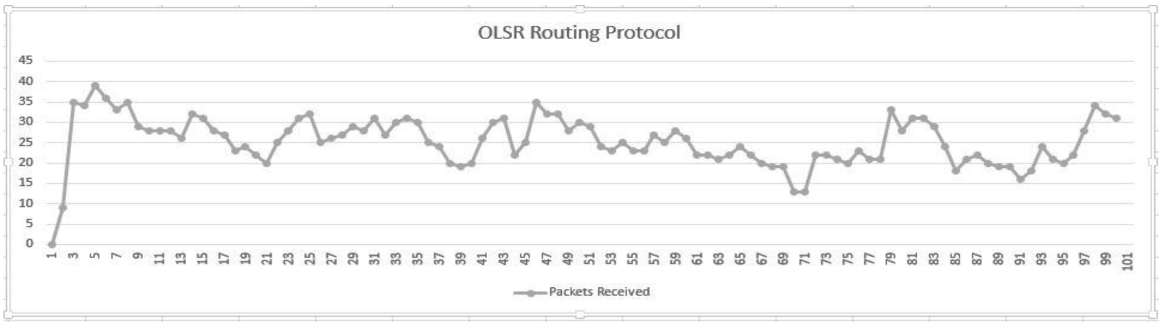
Figure 32. DSDV Output 2

# CHAPTER 5: CONCLUSION

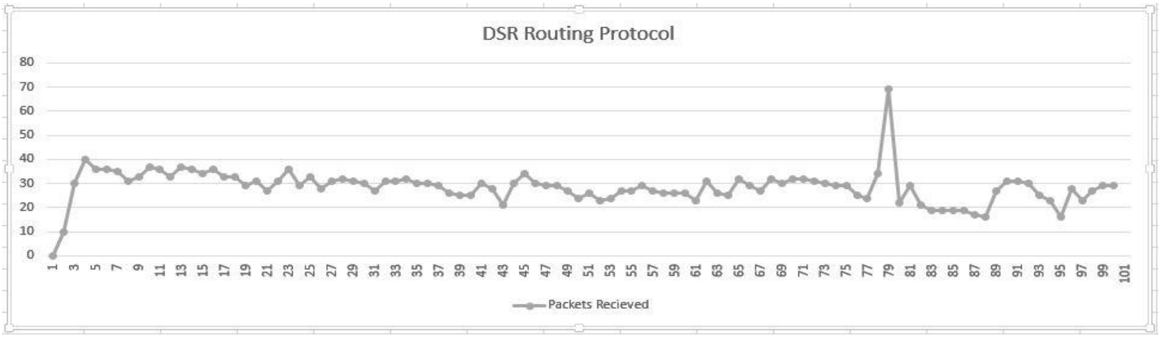
## 5.1 CONCLUSION

There is wide range of different sorts of routing protocols used and practised in MANET. Using a specific routing protocol in MANET can depend on a various of factors which can include size of the network, load that is present on the network, overhead on routing and bandwidth, requirements of the mobility in a network and end to end delay to name a few. Routing protocols in recent times have gained more attention in MANET due to its flexible approach and easy deployment along with efficiency in throughput.

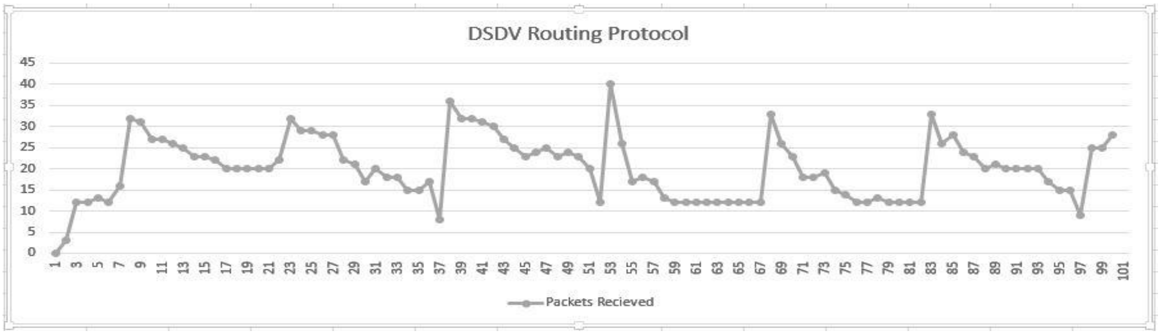
In our project that we have implemented and carried out the comparison under identical traffic and network boundaries. We have found that under given conditions AODV and DSR (Reactive/On-demand) protocols perform better than the counterparts DSDV and OLSR (Proactive/Table driven) protocols with the exception of OLSR performing better than AODV where the use of 50 nodes has been considered in the same environment. It is to be noted we used the latest and most stable version of (Current) of Network Simulator, ver. 3.26 and Network Animator.



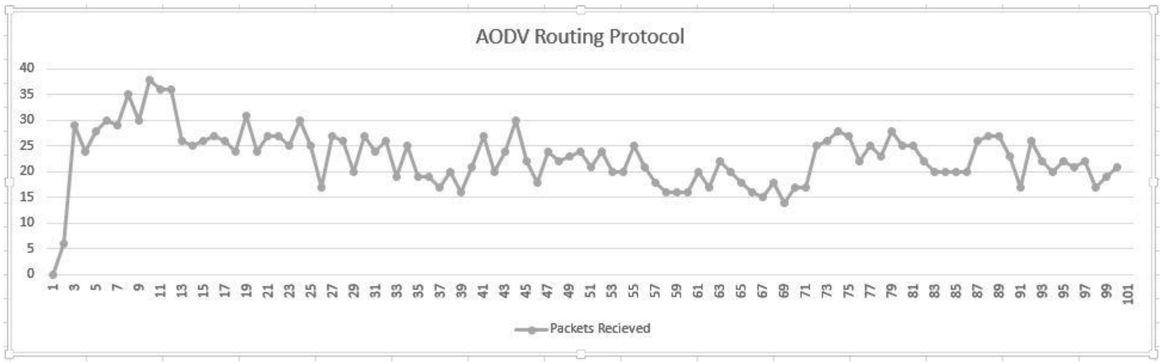
Graph 1. OLSR



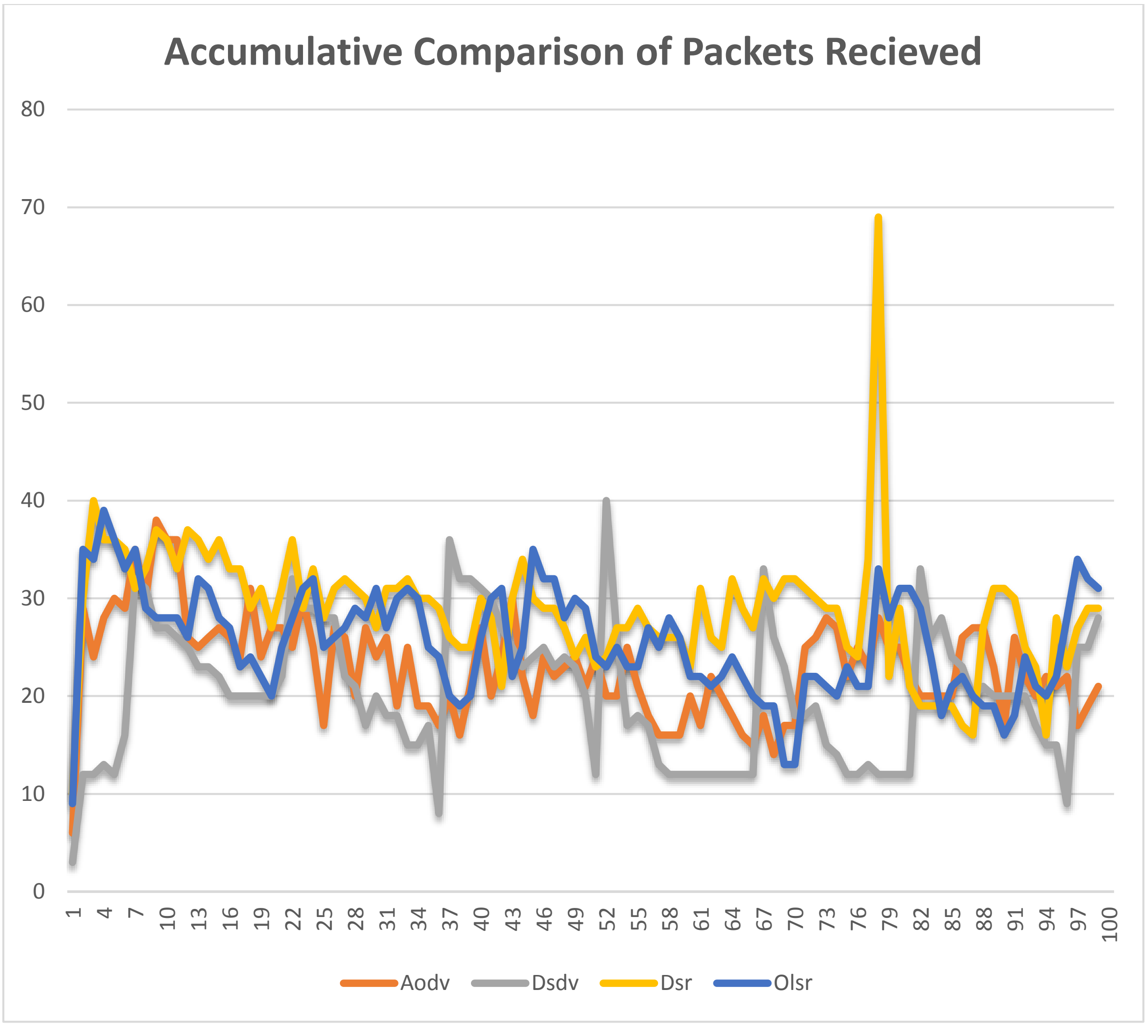
Graph 2. DSR



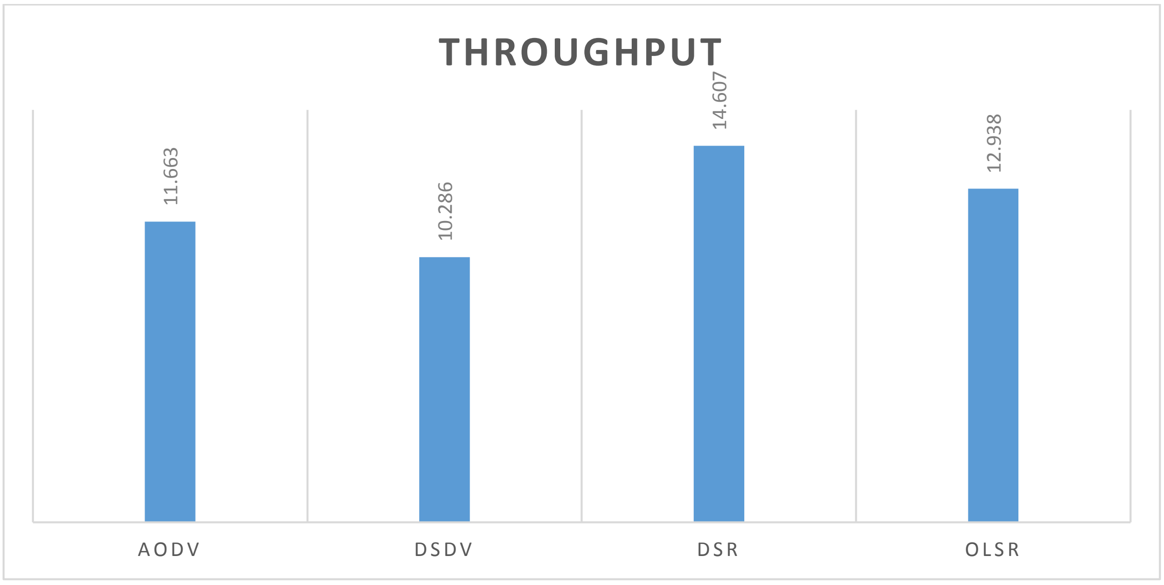
Graph 3. DSDV



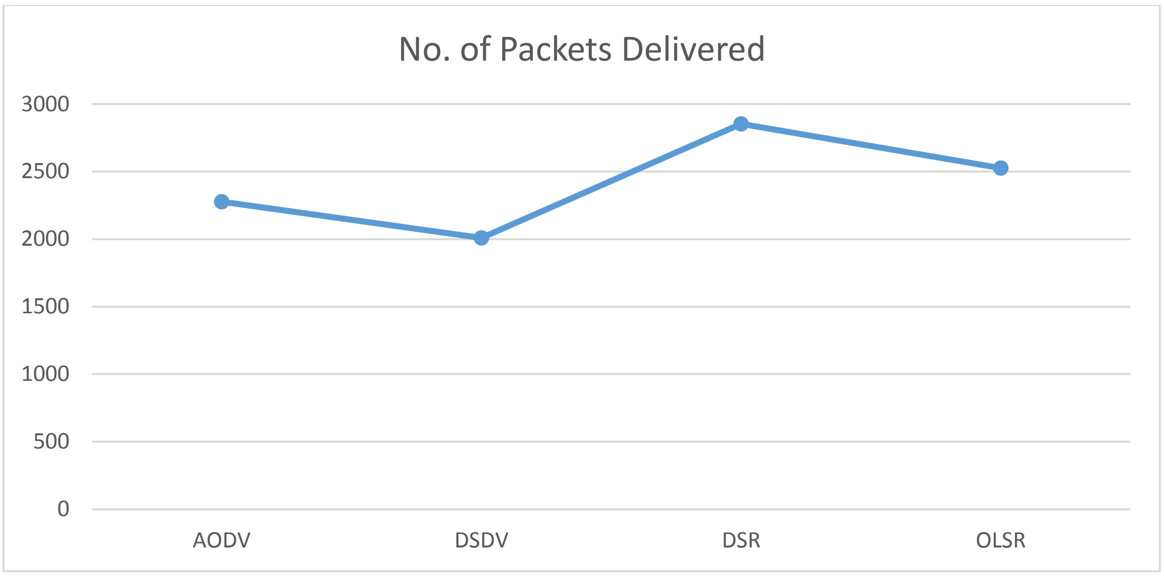
Graph 4. AODV



Graph 5. Accumulative Comparison



Graph 6. Throuput (Mbps)



Graph 7. Number of packets delivered in 200s simulation

## 5.2 Future Scope and Applications

Mobile ad hoc networking is a concept which is increasingly stealing eyes and has now become a boiling concept especially in personal communication. Almost everywhere in the world multiple researches are being conducted to address the issues being faced in MANETs. Both uni-path and multipath routing can be used to take a step towards achieving quality of service. The present research and conclusion can actually be extended to form some of the features which are a must in the future generations of protocols.

Robust Scenario: Any routing must be conducted in a robust scenario and for that there would be need of the most robust routing protocol, if achieved it can make communication a much easier job.

Probabilistic Route Maintenance: There definitely is a need of research in fields from where a protocol which can efficiently identify probabilistic routes, can help in minimizing many failures. But this would first require a robust routing protocol which we have tried to achieve and such results can be used as the foundation of the research for a probabilistic maintenance.

Quality of service (QoS): Any routing protocol must meet a set of requirements of quality to achieve better results pertaining to the performance metrics being considered.

Security: Security is a vital issue in any discipline of computer sciences as it deals with a lot of personal information. In networking it becomes a major issue since communication is taking place and it deals with confidential information. With much of research in the performances of the routing protocols a protocol can be devised having key authorization and security measures, facilitating the usage of MANET in military even better.

Routing Overhead: Routing usually makes use of most of the bandwidth available, with more researches on the performances of the routing protocols, like ours, deployment of a protocol which minimizes he routing overhead can be devised.

Energy Aware Routing: Mobile nodes make use of batteries which are small and portable, with the knowledge on how efficiently a protocol works in certain conditions we can devise protocols which are energy aware protocols, making routing an even more efficient paradigm.

Further, newer operational demands will bring in better trends and ways in designing of a more robust and efficient protocols of and related to mobile and wireless networks. And lastly communication keeps on evolving forever making research a perpetual paradigm.

### 5.2.1 Quality of service (QoS)

We recommend expansion of further knowledge based on our conclusions of comparison of routing protocols in different scenarios and under different conditions and environments in order to achieve better quality of services in networks with mobile nodes and dynamically changing topologies.

# REFERENCES

[1] Roberto Alejandro, Larrea-Luzuriaga, Jose Miguel Jimenez, Sandra Sendra, Jaime Lloret, “Comparative Study of Routing Protocols in Ring Topologies using GNS3”, ICIMP 2016 : The Eleventh International Conference on Internet Monitoring and Protection

[2] Rakesh Kumar Jha, Pooja Kharga, “A Comparative Performance Analysis of Routing Protocols in MANET using NS3 Simulator”, I. J. Computer Network and Information Security, 2015, 4, 62-68

[3] Mohamed Elboukhari, Mostafa Azizi and Abdelmalek Azizi, “Performance Comparison Of Routing Protocols In Mobile Ad Hoc Networks”, International Journal of UbiComp (IJU), Vol.6, No.2, April 2015

[4] Rakesh k. Jha, Pooja Kharga, Idris Z. Bholebawa, Sangeet Satyarthi , Anuradha, Shashi Kumari, “OpenFlow Technology: A Journey of Simulation Tools ”, I.J. Computer Network and Information Security, 2014, 11, 49-55

[5] Supriya, Sushma Jain, “Performance Comparison of Routing Protocols of MANET in Real World Scenario using NS3”, International Journal of Computer Applications (0975 – 8887) Volume 99 – No.14, August 2014

[6] Upneet Singh, Mohinder Singh, Shanu Malhotra, “Performance evaluation of routing protocols under different mobility models over MANETs”, International Journal of Application or Innovation in Engineering & Management (IJAIEM) Volume 3, Issue 5, May 2014

[7] Anne Aaron, Jie Weng, “Performance Comparison of Ad-hoc Routing Protocols for Networks with Node Energy Constraints”, EE 360 Class Project, Spring 2000-2001

[8] Preeti Gaharwar, Mr. Sunil R. Gupta, “Performance Comparison of Routing Protocols”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 4, April 2013

[9] P. Manickam, T. Guru Baskar, M.Girija, Dr.D.Manimegalai, “Performance Comparison of Routing Protocols in Mobile Ad Hoc Networks”, International Journal of Wireless & Mobile Networks (IJWMN) Vol. 3, No. 1, February 2011

[10] Arun Kumar B. R., Lokanatha C. Reddy, Prakash S. Hiremath, "Performance Comparison of Wireless Mobile Ad-Hoc Network Routing Protocols", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.6, June 2008

[11] Abolhasan, M., Wysocki, T. & Dutkiewicz, E., "A review of routing protocols for mobile ad hoc networks", Faculty of Engineering and Information Sciences Papers, 2004, 1-22

[12] Azzedine Boukerche, "A Performance Comparison of Routing Protocols for Ad Hoc Networks", Parallel Simulations and Distributed Systems(PARADISE), Research Laboratory, University of North of Texas

[13] Geetha Jayakumar, Gopinath Ganapathy, "Performance Comparison of Mobile Ad-hoc Network Routing Protocol", IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.11, November 2007

[14] Shiva Prakash, J. P.  Saini, S. C. Gupta, "Methodologies and Applications of Wireless Mobile Ad-hoc Networks Routing Protocols", International Journal of Applied Information Systems (IJAIS), Foundation of Computer Science FCS, New York, USA Volume 1– No.6, February

[15] Amandeep Verma, "A Study of Performance Comparisons of Simulated Ad hoc Network Routing Protocols", Int. J. Comp. Tech. Appl., Vol 2 (3), 565-569

[16] Juan Carlos Cano, Pietro Manzoni, "A Performance Comparison of Energy Consumption for Mobile Ad Hoc Networks Routing Protocols",Departamento de Informática de Sistemas y Computadores (DISCA), Escuela de Informática – Universidad Politécnica de Valencia

[17] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, Jorjeta Jetcheva, "A Performance Comparison of Multi-Hop wireless Ad-Hoc Network Routing Protocols", Carnegie Mellon University, Pittsburgh, P.A

[18] Sushil Kumar, Dinesh Singh, Mridul Chawla, "Performance Comparison of Routing Protocols in MANET Varying Network Size", DCRUST, Haryana, India

[19] Ioannis Broustis Gentian, Jakllari Thomas, Repantis Mart Moll, “A Performance Comparison of Routing Protocols for Large-Scale Wireless Mobile Ad Hoc Networks”, Department of Computer Science & Engineering University of California, Riverside.

[20] K. Prabu1, Dr. A. Subramani, “Performance Comparison of Routing Protocols in MANET”, International Journal of Advanced Research in Computer Science and Software Engineering , Volume 2, Issue 9, September 2012

[21] Aarti, Dr. S. S. Tyagi, “Study of MANET: Characteristics, Challenges, Application and Security Attacks”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013

[22] Mr. L Raja, Capt. Dr. S Santhosh Baboo, “An Overview of MANET: Applications, Attacks and Challenges”, IJCSMC, Vol. 3, Issue. 1, January 2014, pg.408 – 417

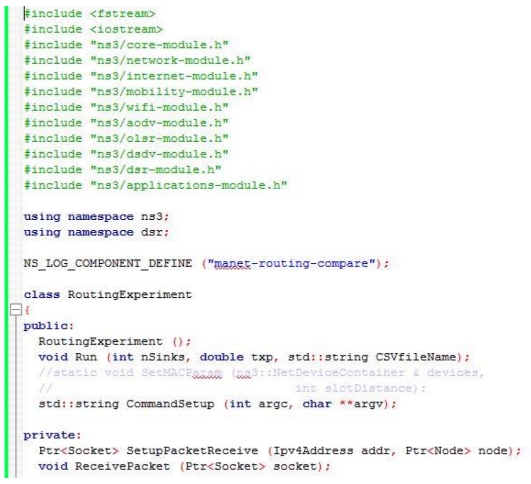
[23] Nadia Qasim, Fatin Said, Hamid Aghvami, “Mobile Ad Hoc Networks Simulations Using Routing Protocols for Performance Comparisons”, Proceedings of the World Congress on Engineering 2008 Vol I WCE 2008, July 2 - 4, 2008, London, U.K.

[24] Ahmad Anzaar, Husain Shahnawaz, Chand Mukesh, SC Gupta, R Gowri, H.L.Mandoria, “Simulation Study for Performance Comparison of Routing Protocols in Mobile Adhoc Network”, World Academy of Science, Engineering and Technology International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:4, No:10, 2010

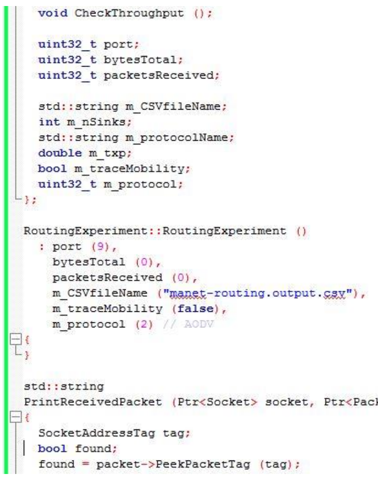
# Appendices

## Program Code:

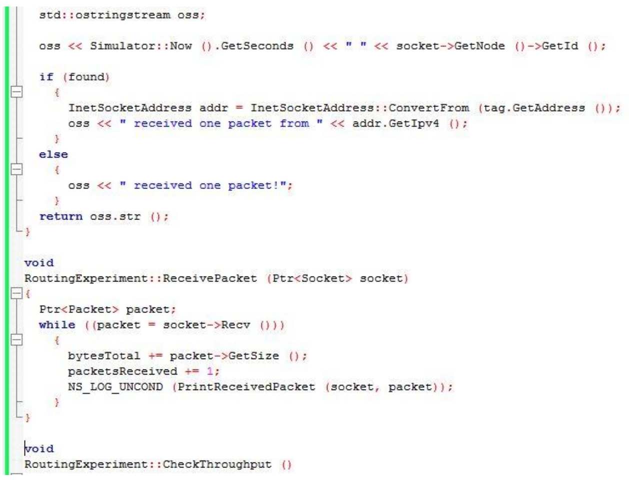
### Screenshots:



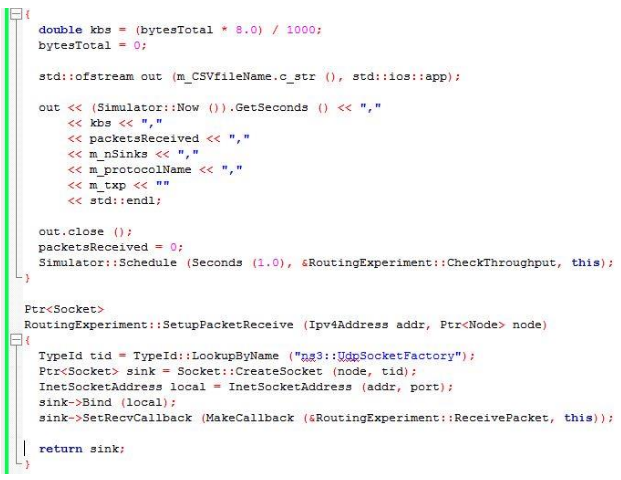
```cpp
#include <fstream>
#include <iostream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
#include "ns3/dsdv-module.h"
#include "ns3/dsr-module.h"
#include "ns3/applications-module.h"

using namespace ns3;
using namespace dsr;

NS_LOG_COMPONENT_DEFINE ("manet-routing-compare");

class RoutingExperiment
{
public:
  RoutingExperiment ();
  void Run (int nSinks, double txp, std::string CSVfileName);
  //static void SetMACParam (ns3::NetDeviceContainer & devices,
  //                                 int slotDistance);
  std::string CommandSetup (int argc, char **argv);

private:
  Ptr<Socket> SetupPacketReceive (Ipv4Address addr, Ptr<Node> node);
  void ReceivePacket (Ptr<Socket> socket);
```



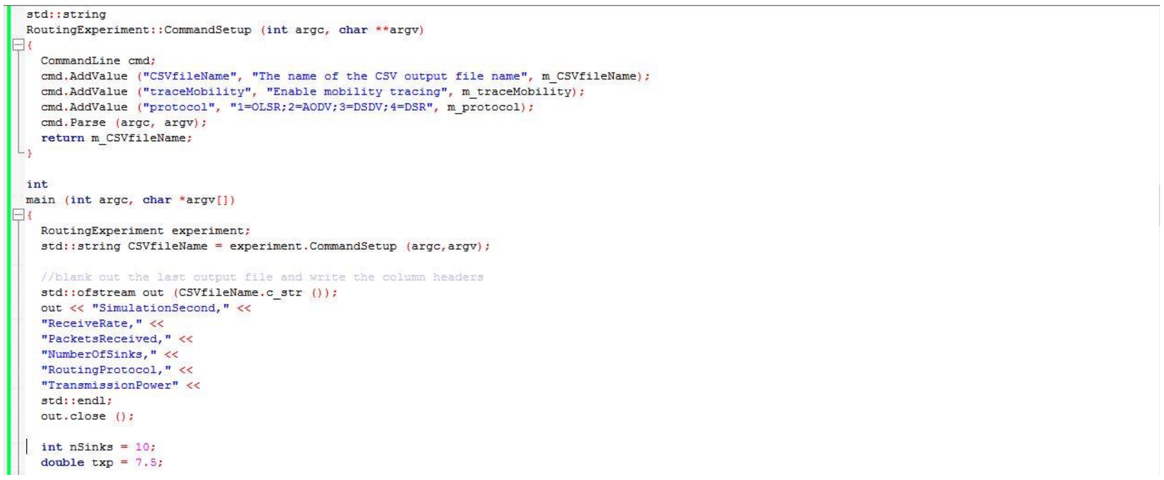
```cpp
  void CheckThroughput ();

  uint32_t port;
  uint32_t bytesTotal;
  uint32_t packetsReceived;

  std::string m_CSVfileName;
  int m_nSinks;
  std::string m_protocolName;
  double m_txp;
  bool m_traceMobility;
  uint32_t m_protocol;
};

RoutingExperiment::RoutingExperiment ()
  : port (9),
    bytesTotal (0),
    packetsReceived (0),
    m_CSVfileName ("manet-routing.output.csv"),
    m_traceMobility (false),
    m_protocol (2) // AODV
{
}

std::string
PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet)
{
  SocketAddressTag tag;
  bool found;
  found = packet->PeekPacketTag (tag);
```



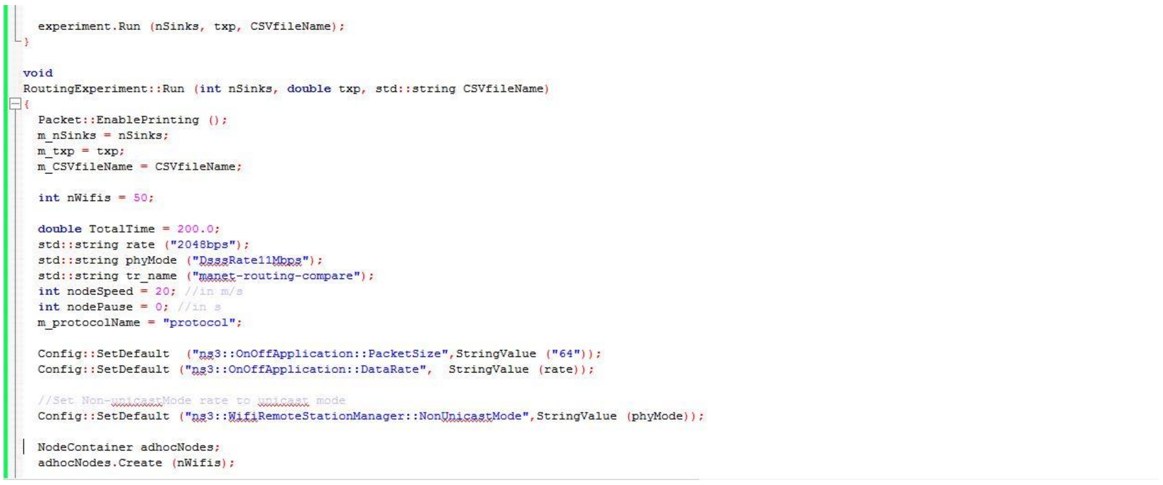
```cpp
  std::ostringstream oss;

  oss << Simulator::Now ().GetSeconds () << " " << socket->GetNode ()->GetId ();

  if (found)
    {
      InetSocketAddress addr = InetSocketAddress::ConvertFrom (tag.GetAddress ());
      oss << " received one packet from " << addr.GetIpv4 ();
    }
  else
    {
      oss << " received one packet!";
    }
  return oss.str ();
}

void
RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
{
  Ptr<Packet> packet;
  while ((packet = socket->Recv ()))
    {
      bytesTotal += packet->GetSize ();
      packetsReceived += 1;
      NS_LOG_UNCOND (PrintReceivedPacket (socket, packet));
    }
}

void
RoutingExperiment::CheckThroughput ()
```



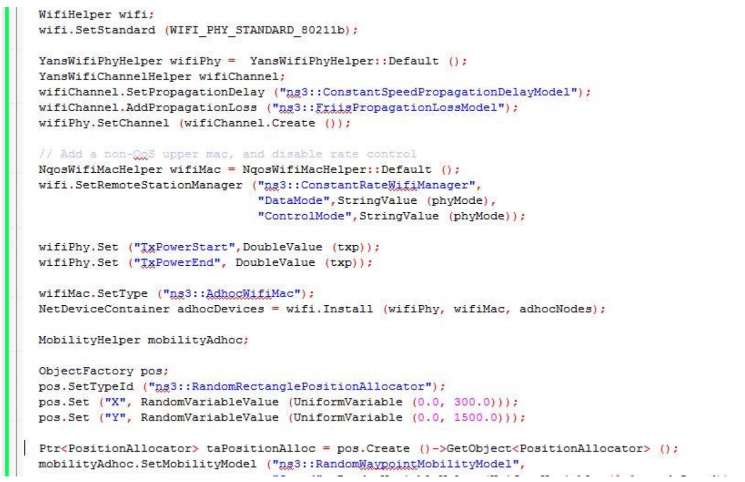
```
{
  double kbs = (bytesTotal * 8.0) / 1000;
  bytesTotal = 0;

  std::ofstream out (m_CSVfileName.c_str (), std::ios::app);

  out << (Simulator::Now ()).GetSeconds () << ","
      << kbs << ","
      << packetsReceived << ","
      << m_nSinks << ","
      << m_protocolName << ","
      << m_txp << ""
      << std::endl;

  out.close ();
  packetsReceived = 0;
  Simulator::Schedule (Seconds (1.0), &RoutingExperiment::CheckThroughput, this);
}

Ptr<Socket>
RoutingExperiment::SetupPacketReceive (Ipv4Address addr, Ptr<Node> node)
{
  TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
  Ptr<Socket> sink = Socket::CreateSocket (node, tid);
  InetSocketAddress local = InetSocketAddress (addr, port);
  sink->Bind (local);
  sink->SetRecvCallback (MakeCallback (&RoutingExperiment::ReceivePacket, this));

  return sink;
}
```



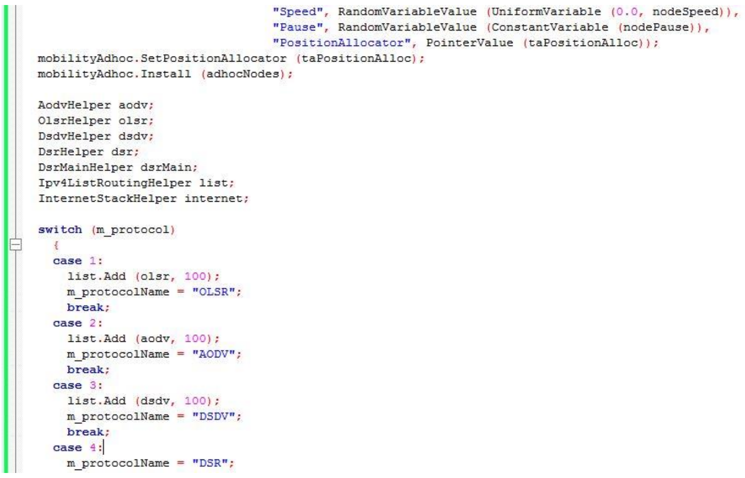
```
std::string
RoutingExperiment::CommandSetup (int argc, char **argv)
{
  CommandLine cmd;
  cmd.AddValue ("CSVfileName", "The name of the CSV output file name", m_CSVfileName);
  cmd.AddValue ("traceMobility", "Enable mobility tracing", m_traceMobility);
  cmd.AddValue ("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR", m_protocol);
  cmd.Parse (argc, argv);
  return m_CSVfileName;
}

int
main (int argc, char *argv[])
{
  RoutingExperiment experiment;
  std::string CSVfileName = experiment.CommandSetup (argc,argv);

  //blank out the last output file and write the column headers
  std::ofstream out (CSVfileName.c_str ());
  out << "SimulationSecond," <<
  "ReceiveRate," <<
  "PacketsReceived," <<
  "NumberOfSinks," <<
  "RoutingProtocol," <<
  "TransmissionPower" <<
  std::endl;
  out.close ();

  int nSinks = 10;
  double txp = 7.5;
```



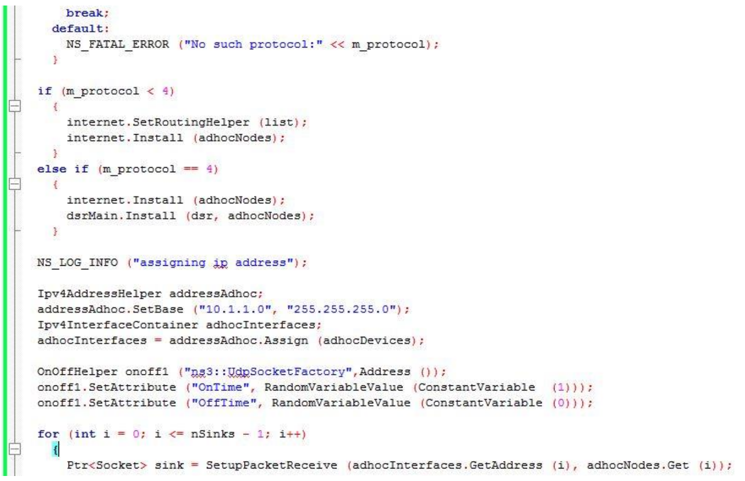
```
  experiment.Run (nSinks, txp, CSVfileName);
}

void
RoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
{
  Packet::EnablePrinting ();
  m_nSinks = nSinks;
  m_txp = txp;
  m_CSVfileName = CSVfileName;

  int nWifis = 50;

  double TotalTime = 200.0;
  std::string rate ("2048bps");
  std::string phyMode ("DsssRate11Mbps");
  std::string tr_name ("manet-routing-compare");
  int nodeSpeed = 20; //in m/s
  int nodePause = 0; //in s
  m_protocolName = "protocol";

  Config::SetDefault  ("ns3::OnOffApplication::PacketSize",StringValue ("64"));
  Config::SetDefault ("ns3::OnOffApplication::DataRate",  StringValue (rate));

  //Set Non-unicastMode rate to unicast mode
  Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",StringValue (phyMode));

  NodeContainer adhocNodes;
  adhocNodes.Create (nWifis);
```



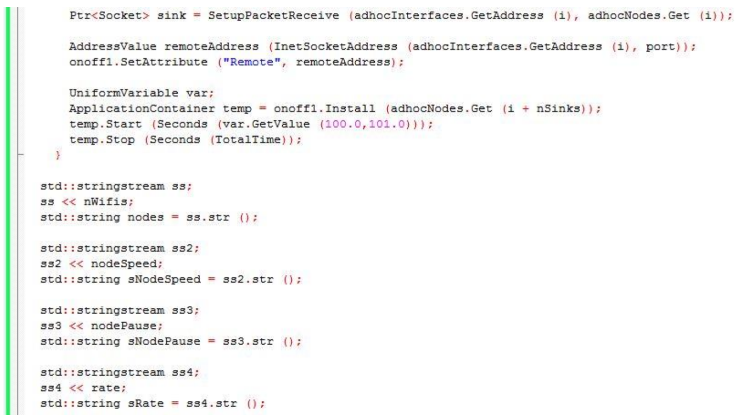
```
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());

// Add a non-QoS upper mac, and disable rate control
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));

wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));

wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer adhocDevices = wifi.Install (wifiPhy, wifiMac, adhocNodes);

MobilityHelper mobilityAdhoc;

ObjectFactory pos;
pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
pos.Set ("X", RandomVariableValue (UniformVariable (0.0, 300.0)));
pos.Set ("Y", RandomVariableValue (UniformVariable (0.0, 1500.0)));

Ptr<PositionAllocator> taPositionAlloc = pos.Create ()->GetObject<PositionAllocator> ();
mobilityAdhoc.SetMobilityModel ("ns3::RandomWaypointMobilityModel",
```



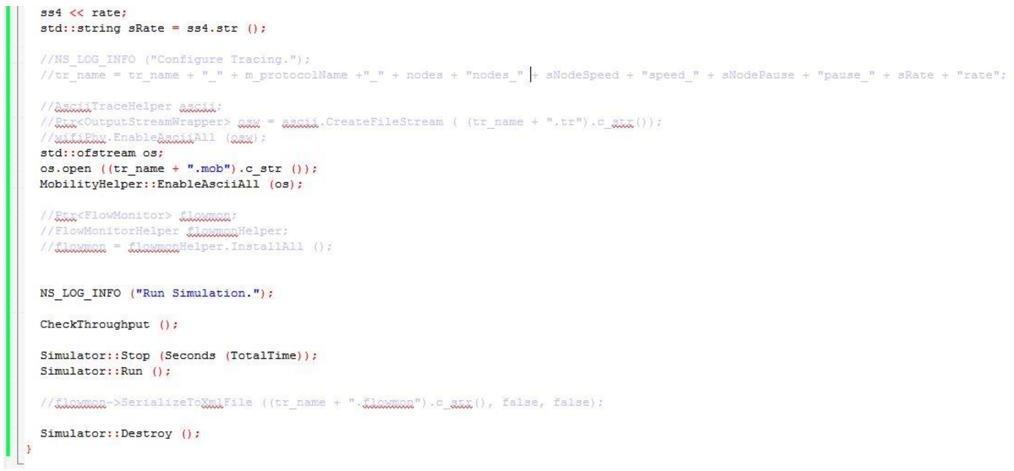
```
                                "Speed", RandomVariableValue (UniformVariable (0.0, nodeSpeed)),
                                "Pause", RandomVariableValue (ConstantVariable (nodePause)),
                                "PositionAllocator", PointerValue (taPositionAlloc));
mobilityAdhoc.SetPositionAllocator (taPositionAlloc);
mobilityAdhoc.Install (adhocNodes);

AodvHelper aodv;
OlsrHelper olsr;
DsdvHelper dsdv;
DsrHelper dsr;
DsrMainHelper dsrMain;
Ipv4ListRoutingHelper list;
InternetStackHelper internet;

switch (m_protocol)
  {
  case 1:
    list.Add (olsr, 100);
    m_protocolName = "OLSR";
    break;
  case 2:
    list.Add (aodv, 100);
    m_protocolName = "AODV";
    break;
  case 3:
    list.Add (dsdv, 100);
    m_protocolName = "DSDV";
    break;
  case 4:
    m_protocolName = "DSR";
```



```
    break;
  default:
    NS_FATAL_ERROR ("No such protocol:" << m_protocol);
  }

if (m_protocol < 4)
  {
    internet.SetRoutingHelper (list);
    internet.Install (adhocNodes);
  }
else if (m_protocol == 4)
  {
    internet.Install (adhocNodes);
    dsrMain.Install (dsr, adhocNodes);
  }

NS_LOG_INFO ("assigning ip address");

Ipv4AddressHelper addressAdhoc;
addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer adhocInterfaces;
adhocInterfaces = addressAdhoc.Assign (adhocDevices);

OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
onoff1.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable  (1)));
onoff1.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));

for (int i = 0; i <= nSinks - 1; i++)
  {
    Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (i), adhocNodes.Get (i));
```



```
        Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (i), adhocNodes.Get (i));

        AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.GetAddress (i), port));
        onoff1.SetAttribute ("Remote", remoteAddress);

        UniformVariable var;
        ApplicationContainer temp = onoff1.Install (adhocNodes.Get (i + nSinks));
        temp.Start (Seconds (var.GetValue (100.0,101.0)));
        temp.Stop (Seconds (TotalTime));
    }

    std::stringstream ss;
    ss << nWifis;
    std::string nodes = ss.str ();

    std::stringstream ss2;
    ss2 << nodeSpeed;
    std::string sNodeSpeed = ss2.str ();

    std::stringstream ss3;
    ss3 << nodePause;
    std::string sNodePause = ss3.str ();

    std::stringstream ss4;
    ss4 << rate;
    std::string sRate = ss4.str ();
```



```
    ss4 << rate;
    std::string sRate = ss4.str ();

    //NS_LOG_INFO ("Configure Tracing.");
    //tr_name = tr_name + "_" + m_protocolName +"_" + nodes + "nodes_" + sNodeSpeed + "speed_" + sNodePause + "pause_" + sRate + "rate";

    //AsciiTraceHelper ascii;
    //Ptr<OutputStreamWrapper> osw = ascii.CreateFileStream ( (tr_name + ".tr").c_str());
    //wifiPhy.EnableAsciiAll (osw);
    std::ofstream os;
    os.open ((tr_name + ".mob").c_str ());
    MobilityHelper::EnableAsciiAll (os);

    //Ptr<FlowMonitor> flowmon;
    //FlowMonitorHelper flowmonHelper;
    //flowmon = flowmonHelper.InstallAll ();


    NS_LOG_INFO ("Run Simulation.");

    CheckThroughput ();

    Simulator::Stop (Seconds (TotalTime));
    Simulator::Run ();

    //flowmon->SerializeToXmlFile ((tr_name + ".flowmon").c_str(), false, false);

    Simulator::Destroy ();
}
```