A

PROJECT REPORT

on

ENHANCED DSR ALGORITHM WITH REDUCED DELAY AND OVERHEAD

Submitted in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

N.NIHARIKA (160110737014)

R.SINDHUJA (160110737025)

Under the esteemed guidance of

Dr. K. Radhika

Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(Affiliated to Osmania University; accredited by NBA (AICTE), ISO certified 9001:2000)

GANDIPET – 500 075

2014-2015

Chaitanya Bharathi Institute of Technology
(Affiliated to Osmania University : accredited by NBA (AICTE), ISO Certified 9001:2000)
Chaitanya Bharathi P.O., Gandipet, Hyderabad - 500 075.
e.mail: ircrcbit@yahoo.com; Website : www.cbit.ac.in
☎ (O) 08413-233356; 958413-233356; 958413-233865; Fax : 958413 - 232099

CERTIFICATE

This is to certify that the project work entitled **ENHANCED DSR ALGORITHM WITH REDUCED DELAY AND OVERHEAD** submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY,** in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology, during the academic year 2014-15, is a record of original work done by **N.NIHARIKA (160110737014), R.SINDHUJA (160110737025)** during the period of study in the Dept. of IT, CBIT, HYDERABAD, under our supervision and guidance.

.

Project Guide	Head of the Department
Dr. K. Radhika	**Dr. Suresh Pabboju**
Associate Professor,	Professor and Head,
Department of IT,	Department of IT,
CBIT, Hyderabad.	CBIT, Hyderabad.

DECLARATION

This is to certify that the work reported in the present report titled **ENHANCED DSR ALGORITHM WITH REDUCED DELAY AND OVERHEAD** is a record of work done by us in the Department of Information Technology, Chaitanya Bharathi Institute of Technology, Osmania University.

No part of this is copied from books / journals / Internet and wherever the portion is taken the same have been duly referred in the text. This report is based on the project work done entirely by us and not copied from any other source.

N.NIHARIKA (160110737014)
R.SINDHUJA (160110737025)

ACKNOWLEDGEMENT

I take this opportunity to thank all who have rendered their full support to our work. The pleasure, the achievement, the glory, the satisfaction, the reward, the appreciation and the construction of our project seminar cannot be thought without a few, how apart from their regular schedule, spared a valuable time for us. This acknowledgement is not just a position of words but also an account of the indictment. They have been a guiding light and source of inspiration towards the completion of the project seminar work.

We are grateful to our project guide **Dr. K. Radhika** Asst. Prof., Dept of IT, CBIT for her kind and timely help offered to us in this project report.

We take the opportunity to express our thanks to **Dr. Suresh Pabboju**, Professor and Head of the dept, Dept of I.T. CBIT for his valuable suggestion and moral support.

Our respects and regards to **Dr. B. Chennakesava Rao,** Principal, Chaitanya Bharathi Institute of Technology, for his cooperation and encouragement.

Finally, we thank the staff members, faculty of Dept. of IT, CBIT, our friends, and all our family members who with their valuable suggestions and support, directly or indirectly helped us to accomplish this project.

N.NIHARIKA (160110737014)
R.SINDHUJA (160110737025)

ABSTRACT

MANET is a collection of wireless mobile devices that communicate with each other without the use of any wired network. MANET have some issues like flooding and broadcast storm problem. These problems arise because of its dynamic topology, broadcasting characteristic and mobility. The most commonly used routing algorithm in

MANETS is DSR algorithm. The Dynamic Source Routing (DSR) protocol is an on-demand routing protocol. Mobile nodes are required to maintain route caches that contain unexpired routes and are continually updated as new routes are learned. The major problem with this algorithm is the storm problem.

In this project, solution for “storm problem” in reactive routing protocol have been proposed by reducing rebroadcast messages. The above mentioned problem have been catered by modifying the flow of route request message and have included route selection field. A modified algorithm M-DSR have been proposed which helps in minimizing the control overhead in reactive routing protocol .This project will further limit the number of control messages in the network to increase the protocol performances.

This protocol is implemented over the MANET network and simulated using NS2. NS2 is a discrete event packet level simulator. It is implemented over cygwin emulator in windows environment. Ns provides substantial support for simulation, routing, and multicast protocols over wired and wireless networks. Performance is evaluated from the parameters such as end-to-end delay and routing overhead. The main objective of the project is reducing the overhead which is found in reactive routing protocol and improving the energy efficiency.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

MANET	Mobile Ad hoc Network
DSR	Dynamic source routing
TCL	Tool Command Language
M-DSR	Modified - Dynamic source routing
NS2	Network simulator 2
WAN	Wide Area Network
LAN	Local Area Network
TCP	Transmission Control Protocol
NAM	Network animator
WLAN	Wireless Local Area Networks
MAN	Metropolitan Area Networks
CAN	Campus Area Networks
SAN	Storage or System Area Network
PAN	Personal Area Network
DAN	Desk Area Network
InVANET	Intelligent vehicular ad hoc networks
VANETs	Vehicular ad hoc networks
WMN	wireless mesh network
WSN	wireless sensor network
AODV	Ad Hoc on-Demand Distance Vector Routing
DSDV	Destination-Sequenced Distance Vector
AP	Access point

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Mobile Ad hoc Network (MANET) is an emerging, exciting and important technology in these days because of the fast growth and enhancement of wireless devices. A MANET is a collection or group of wireless mobile nodes and these nodes cooperate by forwarding packets each other to allow them to communicate beyond their range of direct wireless transmission. Such networks extend the limited wireless transmission range of each node by multi-hop packet forwarding, and thus they ideally suited for scenarios in which pre-deployed infrastructure support is not available. MANETs have some special characteristic features such as unreliable wireless links used for communication between hosts, constantly changing network topologies, limited bandwidth, battery power, low computation power etc.

The primary concern is became security in order to provide a secure and protected communication between mobile nodes in an open hostile environment. The main unique characteristics of MANETS are open peer-to-peer network architecture, shared wireless medium, stringent resource constraints, and highly dynamic network topology unlike the wire line networks. These characteristics cause a number of challenges to security design. The challenges that are nontrivial make a case for building multifence security solutions that is able to achieve both broad protection and desirable network performance.

MANETs are vulnerable to various types of attacks including passive eavesdropping, active interfering, impersonation, and denial-of-service. Intrusion prevention measures such as strong authentication and redundant transmission should be complemented by detection techniques to monitor security status of these networks and identify malicious behaviour of any participating node(s). One of the most critical problems in MANETs is the security vulnerabilities of the routing protocols. A set of nodes may be compromised in such a way that it may not be possible to detect their malicious behaviour easily. Such nodes can generate new routing messages to advertise non-existent links, provide incorrect link state information, and flood other nodes with routing traffic, thus inflicting Byzantine failure in the network.

Dynamic source routing algorithm is most frequently used by wireless nodes. In Dynamic Source Routing algorithm (DSR), the mobile nodes are required to maintain the route through route maintenance and route discovery mechanism. In network simulator, the dynamic source routing algorithm can be implemented by setting up a couple of nodes. The dynamic source routing algorithm can be modified by proposing a solution for “storm problem” in reactive routing protocol where the rebroadcast messages are reduced.

In this project, the focus is on implementing the basic DSR and the Enhanced DSR. The improvements made in enhanced DSR are depicted using a graph where the delay and overhead are compared. This project is developed with NS 2.35 on windows 7 environment. The languages used are TCL script and AWK script.

1.2 APPLICATIONS

With the increase of portable devices as well as progress in wireless communication, ad-hoc networking is gaining importance with the increasing number of widespread applications. Ad-hoc networking can be applied anywhere where there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use. Ad hoc networking allows the devices to maintain connections to the network as well as easily adding and removing devices to and from the network. The set of applications for MANET is diverse, ranging from large-scale, mobile, highly dynamic networks, to small, static networks that are constrained by power sources. Besides the legacy applications that move from traditional infra structured environment into the ad hoc context, a great deal of new services can and will be generated for the new environment.

Manets are majorly used in military battlefield where the military equipment 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. It is also applied in commercial sector where ad hoc can be used in emergency/rescue operations for disaster relief efforts, e.g. in fire, flood, or earth-quake. Emergency rescue operations must take place where non-existing or damaged communications infrastructure and rapid deployment of a communication network is needed. Information is relayed from one rescue team member to another over a small hand held. Other commercial scenarios include e.g. ship-to-ship ad hoc mobile communication, law enforcement, etc.

At local Level, ad hoc networks can autonomously link an instant and temporary multimedia network using notebook computers or palmtop computers to spread and share information among participants at e.g. conference or classroom. Another appropriate local level application might be in home networks where devices can communicate directly to exchange information. Similarly in other civilian environments like taxicab, sports stadium, boat and small aircraft, mobile ad hoc communications will have many applications.

Short-range MANET can simplify the intercommunication between various mobile devices (such as a PDA, a laptop, and a cellular phone). Tedious wired cables are replaced with wireless connections. Such an ad hoc net-work can also extend the access to the Internet or other networks by mechanisms e.g. Wireless LAN (WLAN), GPRS, and UMTS. The PAN is potentially a promising application field of MANET in the future pervasive computing context.

The sensor network 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.

1.3 LITERATURE SURVEY

Increased use of laptop computers within the enterprise, and increase in worker mobility have fuelled the demand for wireless networks. Up until recently of vendors. The technology was slow, expensive and reserved for mobile situations or hostile environments where cabling was impractical or impossible. With the maturing of industry standards and the deployment of lightweight wireless networking, wireless technology was a patchwork of incompatible systems from variety hardware across a broad market section, wireless technology has come of age.

1.3.1 Computer networks and their types:

A computer network is basically a telecommunications network which allows computers to exchange data. A telecommunications network is a collection of terminal nodes, links and any intermediate nodes which are connected so as to enable telecommunication between the terminals. The transmission links connect the nodes together. The connections (network links) between nodes are established using either cable media or wireless media. The best-known computer network is the Internet.

Network computer devices that originate, route and terminate the data are called network nodes .

Nodes can include hosts such as personal computers, phones, servers as well as networking hardware. The nodes use circuit switching, message switching or packet switching to pass the signal through the correct links and nodes to reach the correct destination terminal. Types of networks include LAN(Local Area Networking), WLAN(Wireless Local Area Networks), WAN(Wide Area Networks), MAN(Metropolitan Area Networks), CAN(Campus Area Networks),SAN(Storage or System Area Network),PAN(Personal Area Network) and Dan(Desk Area Network).

1.3.1.1 Ad-Hoc Network

A wireless ad hoc network is a decentralized wireless network. The network is ad hoc because it does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity.

The decentralized nature of wireless ad hoc networks makes them suitable for a variety of applications where central nodes can't be relied on, and may improve the scalability of wireless ad hoc networks compared to wireless managed networks, though theoretical and practical limits to the overall capacity of such networks have been identified. Minimal conFiguration and quick deployment make ad hoc networks suitable for emergency situations like natural disasters or military conflicts. In most wireless ad hoc networks the nodes compete to access the shared wireless medium, often resulting in collisions. Using cooperative wireless communications improves immunity to interference by having the destination node combine self-interference and other-node interference to improve decoding of the desired signal. Wireless ad hoc networks can be further classified by their application as namely Mobile ad hoc networks (MANETs), Wireless mesh networks and Wireless sensor networks.

1.3.1.2 Mobile ad hoc Networks:

A mobile ad hoc network (MANET), sometimes called a mobile mesh network, is a self-conFiguring network of mobile devices connected by wireless links. Each device in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic. Such networks may operate by themselves or may be connected to larger Internet.

MANETs are a kind of wireless ad hoc networks that usually has a routeable networking environment on top of a Link Layer ad hoc network. They are also a type of mesh network, but many mesh networks are not mobile or not wireless. The growth of laptops and 802.11/Wi-Fi wireless networking have made MANETs a popular research topic since the mid- to late 1990s. Different protocols are then evaluated based on the packet drop rate, the overhead introduced by the routing protocol, and other measures.

The different types of MANETs have been described as follows: Firstly, Vehicular Ad Hoc Networks (VANETs) are used for communication among vehicles. Intelligent vehicular ad hoc networks (InVANETs) are a second kind of MANETS where artificial intelligence helps vehicles to behave in intelligent manners during vehicle-to-vehicle during collisions, accidents and drunk and drive cases.

A wireless mesh network (WMN) is a communications network made up of radio nodes organized in a mesh topology. Wireless mesh networks often consist of mesh clients, mesh routers and gateways. The mesh clients are often laptops, cell phones and other wireless devices while the mesh routers forward traffic to and from the gateways which may but need not connect to the Internet. The coverage area of the radio nodes working as a single network is sometimes called a mesh cloud. Access to this mesh cloud is dependent on the radio nodes working in harmony with each other to create a radio network. A mesh network is reliable and offers redundancy. When one node can no longer operate, the rest of the nodes can still communicate with each other, directly or through one or more intermediate nodes.

Wireless mesh networks can be implemented with various wireless technology including 802.11, 802.16, cellular technologies or combinations of more than one type. A wireless mesh network can be seen as a special type of wireless ad-hoc network. It is often assumed that all nodes in a wireless mesh network are immobile but this need not be so. The mesh routers may be highly mobile. Often the mesh routers are not limited in terms of resources compared to other nodes in the network and thus can be exploited to perform more resource intensive functions. In this way, the wireless mesh network differs from an ad-hoc network since all of these nodes are often constrained by resources.

Wireless mesh architecture is a first step towards providing high-bandwidth network over a specific coverage area. Wireless mesh architecture’s infrastructure is, in effect, a router network minus the cabling between nodes. It's built of peer radio devices that don't have to be cabled to a wired port like traditional WLAN access points (AP) do. Mesh architecture sustains signal strength by breaking long distances into a series of shorter hops. The intermediate nodes, not only boost the signal but cooperatively make forwarding decisions based on their knowledge of the network, i.e. performs routing. Such an architecture may, with careful design, provide high bandwidth, spectral efficiency, and economic advantage over the coverage area.

Example of three types of wireless mesh network are as follows Infrastructure wireless mesh networks: Mesh routers form an infrastructure for clients. Client wireless mesh networks: In this type of networks, the client nodes constitute the actual network to perform routing and conFiguration functionalities. Hybrid wireless mesh networks: Mesh clients can perform mesh functions with other mesh clients as well as accessing the network.

Wireless mesh network have a relatively stable topology except for the occasional failure of nodes or addition of new nodes. The traffic, being aggregated from a large number of end users, changes infrequently. Practically all the traffic in an infrastructure mesh network is either forwarded to or from a gateway, while in ad hoc networks or client mesh networks the traffic flows between arbitrary pairs of nodes.

This type of infrastructure can be decentralized (with no central server) or centrally managed (with a central server), both are relatively inexpensive, and very reliable and resilient, as each node needs only transmit as far as the next node. The topology of a mesh network is also more reliable, as each node is connected to several other nodes. If one node drops out of the network, due to hardware failure or any other reason, its neighbours can find another route using a routing protocol.

1.3.1.4 Wireless Sensor Networks:

A wireless sensor network (WSN) which is shown in Fig 1.1, consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. They are now used in many industrial and civilian application areas, including industrial process monitoring and control, machine health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control.

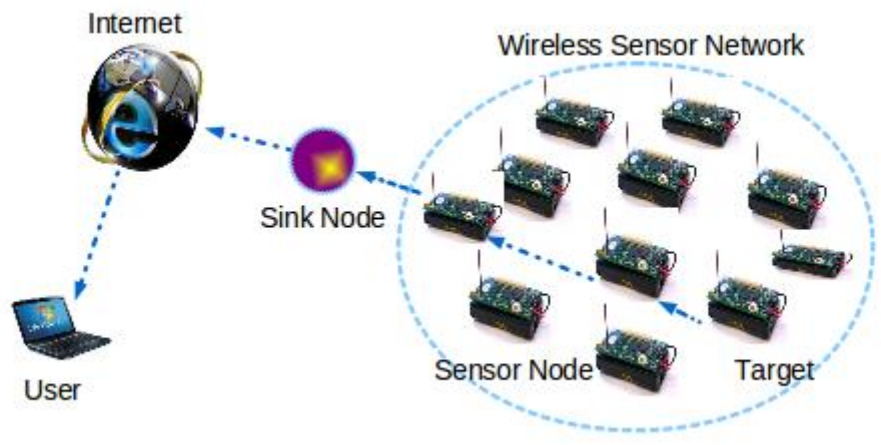


Fig 1.1 Wireless Sensor Network

In addition to one or more sensors, each node in a sensor network is typically equipped with a radio transceiver or other wireless communications device, a small microcontroller, and an energy source, usually a battery. A sensor node, as shown in the Figure, might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from hundreds of dollars to a few pennies, depending on the size of the sensor network and the complexity required of individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory and computational speed.

A sensor network normally constitutes a wireless ad-hoc network, meaning that each sensor supports a multi-hop routing algorithm (several nodes may forward data packets to the base station). Sensor nodes can be imagined as small computers, extremely basic in terms of their interfaces and their components. They usually consist of a processing unit with limited computational power and limited memory, sensors (including specific conditioning circuitry), a communication device (usually radio transceivers or alternatively optical), and a power source usually in the form of a battery. Other possible inclusions are energy harvesting modules and secondary ASICs.

The base stations are one or more distinguished components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user.

1.3.1.5 Wireless Networks

The term wireless networking refers to technology that enables two or more computers to communicate using standard network protocols, but without network cabling. Strictly speaking, any technology that does this could be called wireless networking. The current buzzword however generally refers to wireless LANs. This technology, fuelled by the emergence of cross-vendor industry standards such as IEEE 802.11, has produced a number of affordable wireless solutions that are growing in popularity with business and schools as well as sophisticated applications where network wiring is impossible, such as in warehousing or point-of-sale handheld equipment.

There are two kinds of wireless networks:

An ad-hoc, or peer-to-peer wireless network (shown in Fig 1.2) consists of a number of computers each equipped with a wireless networking interface card. Each computer can communicate directly with all of the other wireless enabled computers. They can share files and printers this way, but may not be able to access wired LAN resources, unless one of the computers acts as a bridge to the wired LAN using special software. (This is called "bridging").

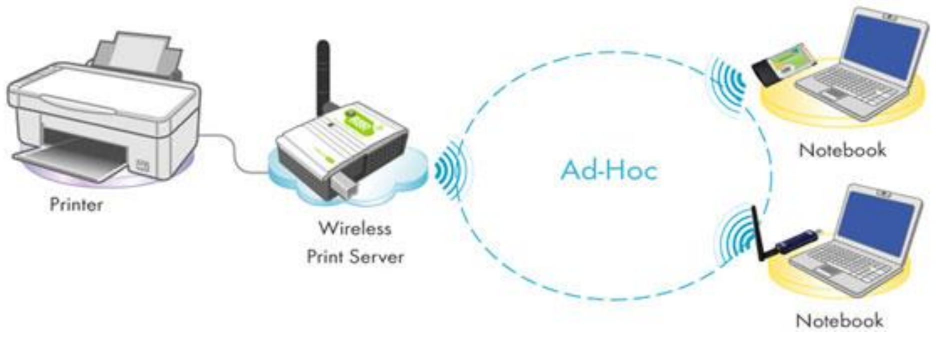


Fig 1.2 Ad-Hoc or Peer-to Peer Networking.

A wireless network can also use an access point, or base station. In this type of network the access point acts like a hub, providing connectivity for the wireless computers. It can connect (or "bridge") the wireless LAN to a wired LAN, allowing wireless computer access to LAN resources, such as file servers or existing Internet Connectivity.

There are two types of access points:

Hardware access points (which is shown in Fig 1.3) offer comprehensive support of most wireless features, but check your requirements carefully.

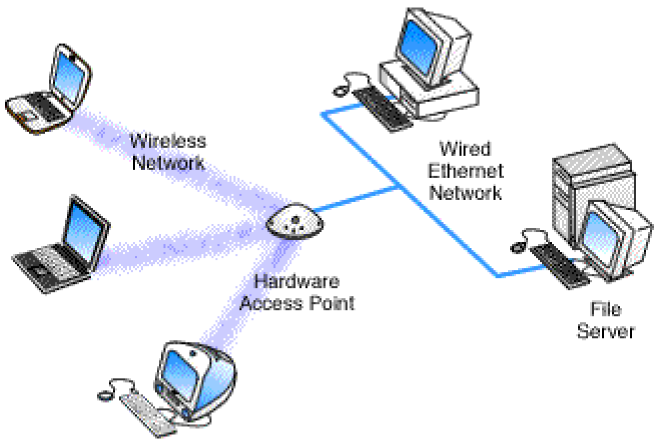


Fig 1.3 Wireless connected computers using a Hardware Access Point.

Software Access Points (shown in Fig 1.4) which run on a computer equipped with a wireless network interface card as used in an ad-hoc wireless network.

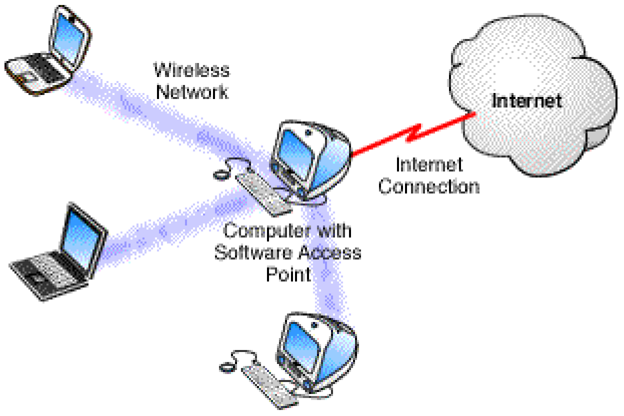


Fig 1.4 Software Access Point

1.3.2 Mobile Ad-Hoc Networks (MANET)

A mobile ad hoc network (MANET), as shown in Fig 1.5, is a continuously self-conFiguring, infrastructure-less network of mobile devices connected without wires. Ad hoc is Latin and means "for this purpose".

Each device in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic. Such networks may operate by themselves or may be connected to the larger Internet. They may contain one or multiple and different transceivers between nodes. This results in a highly dynamic, autonomous topology.

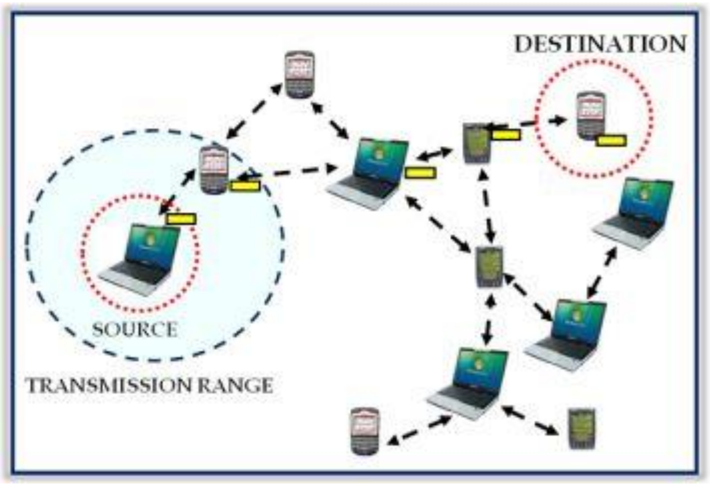


Fig 1.5 Mobile Ad-Hoc Network

MANETs are a kind of Wireless ad hoc network that usually has a routable networking environment on top of a Link Layer ad hoc network. MANETs consist of a peer-to-peer, self-forming, self-healing network in contrast to a mesh network has a central controller (to determine, optimize, and distribute the routing table).

The growth of laptops and 802.11/Wi-Fi wireless networking have made MANETs a popular research topic since the mid-1990s. Many academic papers evaluate protocols and their abilities, assuming varying degrees of mobility within a bounded space, usually with all nodes within a few hops of each other. Different protocols are then evaluated based on measures such as the packet drop rate, the overhead introduced by the routing protocol, end-to-end packet delays, network throughput, ability to scale, etc.

Some salient characteristics of MANETs are:-

- Dynamic topologies,
- Bandwidth constrained, variable capacity links,
- Energy constrained operation
- Limited physical security.

1.3.3 Routing in Networks

Routing is the process of selecting best path in the network for packet to reach from source to destination. It is a part of network layer software. There are two classifications in routing algorithms namely adaptive and non-adaptive.

1.3.3.1 Non Adaptive Routing

Non adaptive refers to static routing algorithms in which router to be used for routing packets from source to destination is decided in advance. In these algorithms routes change very slowly. This can be further classified as:

Flooding: Flooding adapts the technique in which every incoming packet is sent on every outgoing line except the one on which it arrived. One problem with this method is that packets may go in a loop. As a result of this a node may receive several copies of a particular packet which is undesirable. Flooding is not practical for general kinds of applications. But in cases where high degree of robustness is desired such as in military applications, flooding is of great help. Some techniques adapted to overcome these problems are as follows:

Sequence Numbers: Every packet is given a sequence number. When a node receives the packet it sees its source address and sequence number. If the node finds that it has sent the same packet earlier then it will not transmit the packet and will just discard it.

Hop Count: Every packet has a hop count associated with it. This is decremented (or incremented) by one by each node which sees it. When the hop count becomes zero (or a maximum possible value) the packet is dropped.

Spanning Tree: The packet is sent only on those links that lead to the destination by constructing a spanning tree routed at the source. This avoids loops in transmission but is possible only when all the intermediate nodes have knowledge of the network topology.

Random Walk: In this method a packet is sent by the node to one of its neighbors randomly. This algorithm is highly robust. When the network is highly interconnected, this algorithm has the property of making excellent use of alternative routes. It is usually implemented by sending the packet onto the least queued link.

1.3.3.2 Adaptive Routing

In adaptive routing algorithms which are known as dynamic algorithms, the routing decisions change depending on the topology and traffic load. This can be further classified as:

Centralized: In this type some central node in the network gets entire information about the network topology, about the traffic and about other nodes. This then transmits this information to the respective routers. The advantage of this is that only one node is required to keep the information. The disadvantage is that if the central node goes down the entire network is down, i.e. single point of failure.

Isolated: In this method the node decides the routing without seeking information from other nodes. The sending node does not know about the status of a particular link. The disadvantage is that the packet may be send through a congested route resulting in a delay. Some examples of this type of algorithm for routing are:

Hot Potato: When a packet comes to a node, it tries to get rid of it as fast as it can, by putting it on the shortest output queue without regard to where that link leads. A variation of this algorithm is to combine static routing with the hot potato algorithm. When a packet arrives, the routing algorithm takes into account both the static weights of the links and the queue lengths.

Backward Learning: In this method the routing tables at each node gets modified by information from the incoming packets. One way to implement backward learning is to include the identity of the source node in each packet, together with a hop counter that is incremented on each hop. When a node receives a packet in a particular line, it notes down the number of hops it has taken to reach it from the source node. If the previous value of hop count stored in the node is better than the current one then nothing is done but if the current value is better than the value is updated for future use. The problem with this is that when the best route goes down then it cannot recall the second best route to a particular node. Hence all the nodes have to forget the stored information periodically and start all over again.

Distributed: In this the node receives information from its neighboring nodes and then takes the decision about which way to send the packet. The disadvantage is that if in between the interval it receives information and sends the packet something changes then the packet may be delayed.

1.3.4 Routing Protocols in MANET

Routing protocols for MANETs can be broadly classified into three main categories:-

Proactive routing protocols: - Every node in the network has one or more routes to any possible destination in its routing table at any given time.

Reactive routing protocols: - Every node in the network obtains a route to a destination on a demand fashion. Reactive protocols do not maintain up-to-date routes to any destination in the network and do not generally exchange any periodic control messages.

Hybrid routing protocols: - Every node acts reactively in the region close to its proximity and proactively outside of that region, or zone.

1.3.4.1 Ad Hoc on-Demand Distance Vector Routing (AODV)

The AODV Routing Protocol uses an on-demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. The major difference between AODV and Dynamic Source Routing (DSR) stems from the fact that DSR uses source routing in which a data packet carries the complete path to be traversed. However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an on-demand routing protocol, the source node floods the Route Request packet in the network when a route is not available for the desired destination. It may obtain multiple routes to different destinations from a single Route Request. The major difference between AODV and other on-demand routing protocols is that it uses a destination sequence number (DestSeqNum) to determine an up-to-date path to the destination. A node updates its path information only if the DestSeqNum of the current packet received is greater than or equal to the last DestSeqNum stored at the node with smaller hop count.

A RouteRequest carries the source identifier (SrcID), the destination identifier (DestID), the source sequence number (SrcSeqNum), the destination sequence number (DestSeqNum), the broadcast identifier (BcastID), and the time to live (TTL) field. DestSeqNum indicates the freshness of the route that is accepted by the source. When an intermediate node receives a RouteRequest, it either forwards it or prepares a RouteReply if it has a valid route to the destination. The validity of a route at the intermediate node is determined by comparing the sequence number at the intermediate node with the destination sequence number in the RouteRequest packet. If a RouteRequest is received multiple times, which is indicated by the BcastID-SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid routes to the destination, or the destination node itself, are allowed to send RouteReply packets to the source. Every intermediate node, while forwarding a RouteRequest, enters the previous node address and its BcastID. A timer is used to delete this entry in case a RouteReply is not received before the timer expires. This helps in storing an active path at the intermediate node as AODV does not employ source routing of data packets. When a node receives a RouteReply packet, information about the previous node from which the packet was received is also stored in order to forward the data packet to this next node as the next hop toward the destination.

1.3.4.2 Destination-Sequenced Distance Vector (DSDV)

It is a proactive routing algorithm. The DSDV algorithm is a Distance Vector (DV) based routing algorithm designed for use in MANETs, which are defined as the cooperative engagement of a collection of Mobile Hosts without the required intervention of any centralised Access Point (AP).

It operates each node as a specialised router which periodically advertises its knowledge of the network with the other nodes in the network. It makes modifications to the basic Bellman-Ford routing algorithms, thereby doing away with the count-to-infinity problem. The algorithm is designed for portable computing devices such as laptops who have energy and processing capabilities far beyond that of a typical WSN node.

In routing tables of DSDV, an entry stores the next hop towards a destination, the cost metric for the routing path to the destination and a destination sequence number that is created by the destination. Sequence numbers are used in DSDV to distinguish stale routes from fresh ones and avoid formation of route loops. The route updates of DSDV can be either time-driven or event-driven. Every node periodically transmits updates including its routing information to its immediate neighbors. While a significant change occurs from the last update, a node can transmit its changed routing table in an event-triggered style.

1.3.4.3 Dynamic Source Routing (DSR)

It is a routing protocol for wireless mesh networks. It is similar to AODV in that it forms a route on-demand when a transmitting computer requests one. However, it uses source routing instead of relying on the routing table at each intermediate device. This protocol is truly based on source routing whereby all the routing information is maintained (continually updated) at mobile nodes. It has only two major phases, which are Route Discovery and Route Maintenance. Route Reply would only be generated if the message has reached the intended destination node (route record which is initially contained in Route Request would be inserted into the Route Reply). Therefore, it is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The major difference between this and the other on-demand routing protocols is that it is beacon-less and hence does not require periodic hello packet (beacon) transmissions, which are used by a node to inform its neighbours of its presence. The basic approach of this protocol (and all other on-demand routing protocols) during the route construction phase is to establish a route by flooding RouteRequest packets in the network. The destination node, on receiving a RouteRequest packet, responds by sending a RouteReply packet back to the source, which carries the route traversed by the RouteRequest packet received.

1.3.4.4 Broadcast Storm Problem in MANET

A broadcast storm occurs when a network system is overwhelmed by continuous multicast or broadcast traffic. When different nodes are sending/broadcasting data over a network link, and the other network devices are rebroadcasting the data back to the network link in response, this eventually causes the whole network to melt down and lead to the failure of network communication.

There are many reasons a broadcast storm occurs, including poor technology, low port rate switches and improper network conFigurations. A broadcast storm is also known as a network storm. Although computer networks and network devices are very intelligent and efficient, networks and network devices sometimes fail to provide 100% efficiency. The broadcast storm is one of the major deficiencies in computer network systems. For example, suppose there is a small LAN network consisting of three switches (Switch A, Switch B and Switch C), and three network segments (Segment A, Segment B and Segment C). Two nodes are attached within this network. Node A is attached to Segment B, while Node B is directly attached to Switch A. Now, if Node B wants to transmit a data packet to Node A, then traffic is broadcast from Switch A over to Segment C; if this fails, then Switch A also broadcasts traffic over Segment A. Because Node A neither attaches to Segment C, nor Segment A, these switches would further create a flood to Segment B. If neither device/switch has learned the Node A address, then traffic is sent back to Switch A. Hence, all devices/switches keep sending and resending the traffic, eventually

resulting in a flood loop or broadcast loop. The final result is that the network melts down, causing failure in all network links, which is referred to as a broadcast storm. The following elements play an active role in the creation of a broadcast storm:
Poor network management
Poor monitoring of the network
The use of cheap devices, including hubs, switches, routers, cables, connectors, etc.
Improperly maintained network conFiguration and inexperienced network engineers
The lack of a network diagram design, which is needed for proper management and to provide guidelines for all network traffic routes. This can be done on paper and with the help of application software that creates an automated network diagram.

1.4 PROBLEM STATEMENT

MANET have some issues like flooding or broadcast storm problem. These problems arise because of its dynamic topology, broadcasting characteristic and mobility. Broadcast storm problem is considered as a huge problem because of the overhead it imposes on the source node of the wireless network.

The algorithm which is used by DSR is found to be inefficient by certain researchers due to the broadcast storm problem. To avoid this problem, the nodes are selected using the route selection algorithm. In this project, the basic concentration was on the route selection field where the nodes are selected by using one-hop mechanism and two hop mechanism in order to reduce the rebroadcast messages and hence reduce the storm broadcast problem.

To check the efficiency of the algorithm proposed, the graph is generated between DSR and MDSR for the overhead and end-to-end delay.

1.5 EXISTING DSR

The Dynamic Source Routing Protocol is a source-routed on-demand routing protocol. A node maintains route caches containing the source routes that it is aware of. The node updates entries in the route cache as and when it learns about new routes. DSR does not rely on functions like periodic routing advertisement, link status sensing or neighbour detection packets and because of the entirely on demand behaviour, the number of overhead packets caused by DSR scales down to zero.

Route Discovery and Route Maintenance, which are the main mechanisms of the DSR protocol, allows the discovery and maintenance of source routes in the ad hoc network's works entirely on an on-demand basis. As DSR works entirely on demand and as nodes begin to move continuously, the Routing packet overhead automatically scales to only that needed to react to changes in the route currently in use. In response to a single Route Discovery if a node learns and caches multiple routes to a destination, it can try another route if the one it uses fails. The overhead incurred by performing a new Route Discovery can be avoided when the caching of multiple routes to a destination occurs and Mobile IP routing and supports internetworking between different types of wireless networks

DSR route discovery

The header of the packet, which originates from a source node S to a destination node D, contains the source route, which gives the sequence of hops that the packet should traverse. A suitable source route is found normally when searching the Route Cache of routes obtained previously but if no route is found then the Route Discovery protocol is initiated to find a new route to D. Here S is the initiator and D the target. Node A transmits a ROUTE REQUEST message, which is received by all the nodes in the transmission range of A.

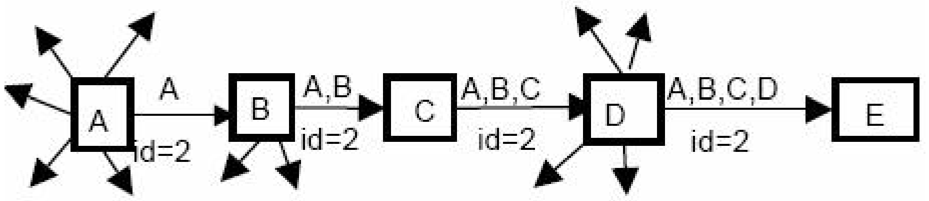


Fig 1.6 Route discovery using dsr

Each ROUTE REQUEST message identifies the initiator and target of the

Route Discovery as shown in Fig 1.6 and also contains a unique request ID, determined by the initiator of the REQUEST. Each ROUTE REQUEST also contains a record listing the address of each intermediate node through which this particular copy of the ROUTE REQUEST message has been forwarded. The initiator of the Route Discovery initializes the route record to an empty list. When the target node receives the ROUTE REQUEST message, it returns a ROUTE REPLY message to the ROUTE Discovery initiator with a copy of the accumulated route record from the ROUTE REQUEST. This route is cached in the Route Cache when the initiator receives the ROUTE REPLY and is used in sending subsequent packets to this destination. When the target node finds a ROUTE REQUEST message from the same initiator bearing the same request ID or if it finds its own address is already listed in the route record of the ROUTE REQUEST message, it discards the REQUEST. If the target node does not find the ROUTE REQUEST message from the initiator, then it appends its address to the route record in the ROUTE REQUEST message and propagates it by transmitting it as a local broadcast packet.

When Route Discovery is initiated the copy of the original packet is saved in a local buffer called Send Buffer. The Send Buffer contains copies of each packet that cannot be transmitted by the sending node. The packets are kept until a source route is available or a timeout or Send Buffer overflow occurs.

As long as a packet is in the Send Buffer, the node should initiate new Route Discovery until time out occurs or overflow of Buffer occurs. An exponential Back off algorithm is designed to limit the rate at which new ROUTE Discoveries may be initiated by any node for the sametarget.

DSR Route Maintenance

When a packet with a source route is forwarded, each node in the source route makes sure that the packet has been received by the next hop in the source route. The confirmation of receipt will be received only by re-transmitting the packet for a number of times. Node A is the originator of a packet to the desired destination E. The packet has a source route through intermediate nodes B, C and D. Node A is responsible for receipt of the packet at B, node B at C, node C at D and node D at E.

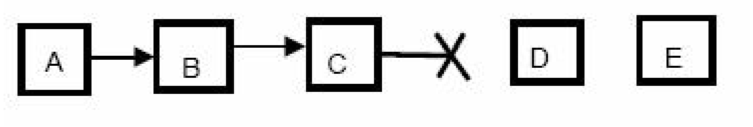


Fig 1.7 Disturbance in route maintenance

Node B confirms receipt of packet at C, as seen in Fig 1.7, by overhearing C transmit the packet to forward it to D. The confirmation of acknowledgement is done by passive acknowledgements or as link-layer mechanisms such as option in MAC protocol. The node receiving the packet can return a DSR specific software acknowledgement if neither of the acknowledgements is available. This is done by setting up a bit in the packet's header and then requesting a DSR specific software acknowledgement by the node transmitting the packet. When a node is unable to deliver a packet to the next node then the node sends a ROUTE ERROR message to the original sender of the packet. The broken link is then removed from the cache by the originator of the packet and retransmissions to the same destination are done by upper layer protocols like TCP.

Route maintenance is also carried out also by both ROUTE REQUEST and ROUTE REPLY packets, when they traverse form each node the data from the option header of these packets which contain the link information of the nodes are updated in the nodes route cache. The simulation consist of five nodes, the source is Node which generates a cbr traffic using udp data packet every .2 seconds. The sink is the Node4, which constantly has to receive the packets generated form the source Node4 moves in a linier manner form the range of node0 to range of node3 as shown in the Fig 1.7. So every time when a data packet arrive without the route to destination the ROUTE REQUEST packet is created and broadcasted and when the ROUTE REPLY arrives, if the route is broken in between, then a ROUTE ERROR is generated to the source node that generated send the data packet

1.5.1 Limitations of existing system

It does not reduce the overhead of source.

It is not energy efficient.
It does not reduce network failure by analysing the energy level of the nodes.

1.6 MODIFIED DSR

In this proposed system solution for “storm problem” in reactive routing protocol by reducing rebroadcast messages have been proposed. The mentioned problem have been catered by modifying the data structure of route request message and have included route selection field .A modified algorithm M-DSR have been proposed that helps in minimizing the control overhead in reactive routing protocol. The advantages of this system are it minimizes the overhead and improves the packet delivery ratio.

One hop neighbour selection

The input in the one hope neighbour selection is route request packet and the output is one hop neighbour. The source node floods the route request over neighbour nodes and it selects the one hop nodes.

Two hop neighbour selection

In two hop neighbour selection, the input is route request packet and the output is two hop neighbour. Source node route request floods the neighbour nodes and it selects the two hop neighbour through the one hop node.

Route selection

The route selection starts from source node keeping the initial set, empty. Calculate the one hop neighbour set and name it as N1. Calculate the two hop neighbour set and name it as N2. Add to RS the nodes in N1 which are the only nodes to provide reachability to a node in N2 or the nodes with degree one. Remove the nodes from N2 which are now covered by a node in RS. When N2 is not NULL calculate the reachability for each node in N1. Add to RS the node that provides highest reachability. In case of multiple possibilities select the node which has highest degree. Remove the nodes from N2 that are covered by RS.

1.6.1 Performance Metrics

In order to measure the performance, we consider Modified DSR Algorithm with Route Selection Criteria as input and the output is the measurement of the end to end delay and overhead

Routing Overhead

Overhead is defined as total number of route update messages involved in the communication.

End-to-end delay (or average delay)

Delay is the average time difference between the time a data packet is sent by the source node and the time it is successfully received by the destination node.

1.7 ORGANISATION OF REPORT

This section outlines the organisation of the report

Chapter 1 discusses the overview of the project, existing system and its disadvantages, proposed system and also explains about survey on the computer networks

Chapter 2 explains about the requirements needed to implement the project. It also explains about the features of the languages used to implement the project.

Chapter 3 discusses about the approach to implement the project

Chapter 4 discusses the implementation part which is discussed in chapter 3

Chapter 5 discusses about the assumptions which are considered to test the project and the screenshots are displayed.

Chapter 6 concludes the project and gives suggestions for future work.

CHAPTER 2
SYSTEM REQUIREMENT SPECIFICATIONS

2.1 INTRODUCTION

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

2.1.1 Purpose of the document

This SRS describes all the requirements elicited for “Enhanced DSR algorithm with reduced delay and overhead” is intended to be used by the members examining the project and implementing and verifying the application. Unless otherwise noted all requirements are of high priority and are committed.

2.2 FUNCTIONAL REQUIREMENTS

While executing the project the user has to specify the number of nodes in the network. The user has to execute the DSR algorithm and modified DSR algorithm.

The graph can be generated by using the trace files which are generated as part of executing the DSR and MDSR algorithm. By doing so, one can check the performance of the MDSR against DSR and hence measure the efficiency. The user can also change the source and destination nodes as per the requirements.

2.3 NON FUNCTIONAL REQUIREMENTS

Performance requirements: The system should be fast and must produce accurate results.

Reliability: The system should be reliable enough such that the increase in nodes doesn’t affect the performance.

Maintainability: The system will be designed as a closed system. New methods can be added easily with little or no changes in the existing architecture.

System Integrity: The power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

2.4 SOFTWARE AND HARDWARE REQUIREMENTS

The Operating System required is Windows7 / Ubuntu and ns-all-in-one 2.35 must be installed along with Cygwin. TCL and AWK scripts are used for coding.

For installing ns2 on windows, free disk space of 5GB along with a minimum of 256MB RAM is required.

2.4.1 NS2

In communication and computer network research, network simulation is a technique where a program models the behaviour 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 behaviour of a computer network. In simulators, the computer network is typically modelled with devices, links, applications etc. and the performance is analysed. Typically, users can then customize the simulator to fulfil their specific analysis needs. Simulators typically come with support for the most popular protocols and networks in use today, such as WLAN, Wi-Max, TCP, WSN, cognitive radio

Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / conFiguration describes the state of the network (nodes, routers, switches, links) and the events (data transmissions, packet error etc.). Important outputs of simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events—such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyse various standard results apart from devising some novel protocol or strategy for routing etc.

There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization.

Ns-2 is a discrete event simulator targeted at networking research. Ns-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Ns-2 is written in C++ and an Object oriented version of Tcl called OTcl. It is developed by UC Berkeley. It is a discrete event simulator for networking research and works at packet level. It provides substantial support to protocols like TCP, UDP, FTP, HTTP, etc. It simulates wired and wireless networks and is UNIX based. It uses TCL as its scripting language.

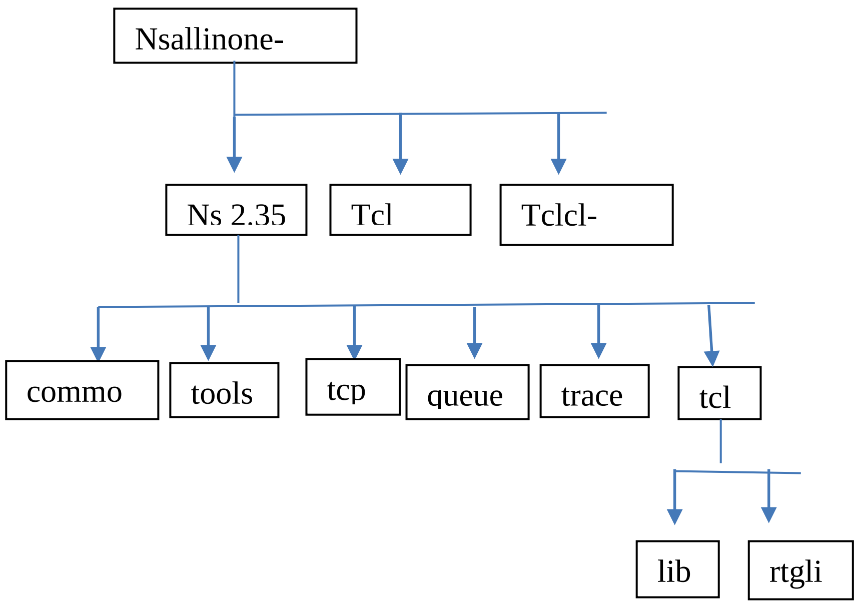


Fig 2.1 Directory Structure of NS2

In Fig 2.1 the directory structure of ns2 can be seen where directory nsallinone-2.35 is on the Level 1. On the Level 2, directory tclcl-1.18 contains classes in TclCL (e.g., Tcl, TclObject, TclClass). All NS2 simulation modules are in directory ns-2.35 on the Level 2. Here after, we will refer to directories ns-2.30 and tclcl-1.18 as ˜ns/ and ˜tclcl / respectively.

On Level 3, the modules in the interpreted hierarchy are under the directory tcl. Among the directories, the frequently used ones (ns-lib.tcl,ns-node.tcl,ns-node.tcl) are stored under the directory lib on Level 4.Simulation modules in compiled hierarchy are classified in the directories on level 3.For example, directory tools contains various helper classes such as random variable generators. Directory common contains basic modules related to packet forwarding such as simulator, scheduler, packet and connectors. Directories queue, tcp and trace contain modules for queue, TCP [Transmission Control Protocol and tracing respectively.

Network Simulator is mainly based on two languages. They are;
C++.
OTCL.

The process of simulation is shown in Fig 2.2. Initially tcl simulation script is sent to ns2 shell executable command which consists of simulation objects and tcl acts as communication interface between two objects. The simulation trace file is generated as output.

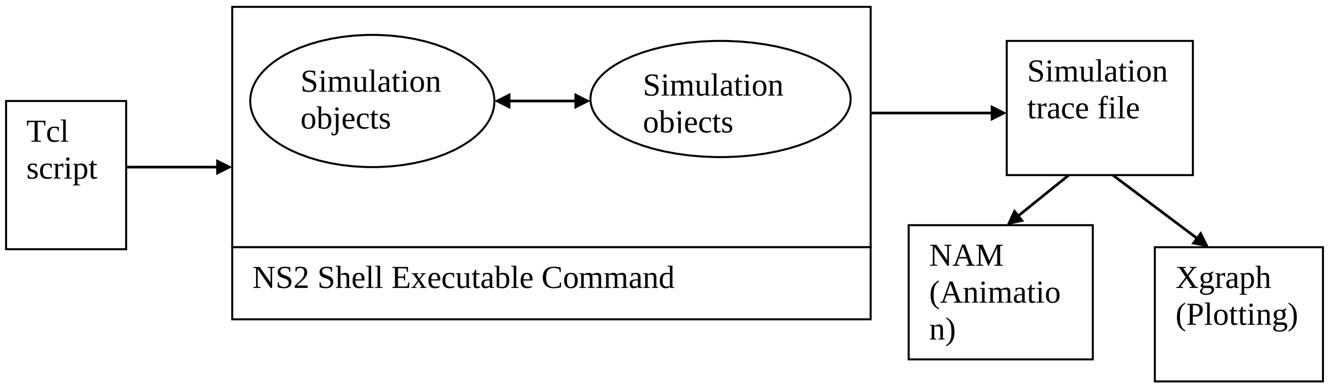


Fig 2.2 Simulation Flow in ns2

The installation of network simulator can be done in two ways. One is using the Oracle virtual VM box which is used to set up the virtualization of Linux environment in windows (works like VMware) and next is by downloading Cygwin. CYGWIN is a Unix-type environment for teaching and developing code in a Unix-like environment for the Windows world. The features of Cygwin is that it shares files with windows and the setup program of Cygwin remembers all the software's which are downloaded previously; hence there is no need of starting the installation of packages from the beginning if there is any disturbance.

The Cygwin packages which are needed to install ns2 are gcc4-g++,gzip, make, patch, perl, w32api, libmpfr4, and X11. NS2 Program comprises four types of scripts: TCL (Tool Command Language) script (.tcl), Trace file (.tr), NAM (Network Animator) file (.nam) and AWK script (.awk)

Tcl is shortened form of Tool Command Language. Tcl is a scripting language that aims at providing the ability for applications to communicate with each other. Tcl was developed initially for UNIX. It was then ported to Windows, DOS, OS/2 and Mac OSX. Tcl is much similar to other Unix shell languages like Bourne Shell (Sh), the C Shell (csh), the Korn Shell (sh), and Perl. The features of tcl script includes powerful set of networking functions; reduced development time and write once run anywhere feature.

AWK is an interpreted programming language. It is very powerful and specially designed for the text processing. The features of awk are text processing; producing formatted text reports; perform arithmetic operations and string operations.

2.4.2 Usage of the scripts:

.tcl and .awk files are created by the programmer. .tr and .nam files are automatically generated while .tcl and .awk files are executed.

Tcl scripts have the code for Node creation, Node conFiguration, Communication establishment between nodes and for Creating of .tr and .nam files. The tcl script os also used for executing .nam and .awk files.

Events in the tcl script are traced and entered into .nam file. This generated code will produce the animation in Network Animator Window (NAM). Events in the tcl script are also traced and entered into .tr file. It contains many lines. Each line comprises number of fields, each separated by the space. Each line specifies an event.

AWK script is used to extract the required information from the code available in .tr file as the .tr file has the many fields. Output of awk script will give the performance measures and also the x and y parameters to be plotted in a graph. Fig 2.3 describes the flow of the code

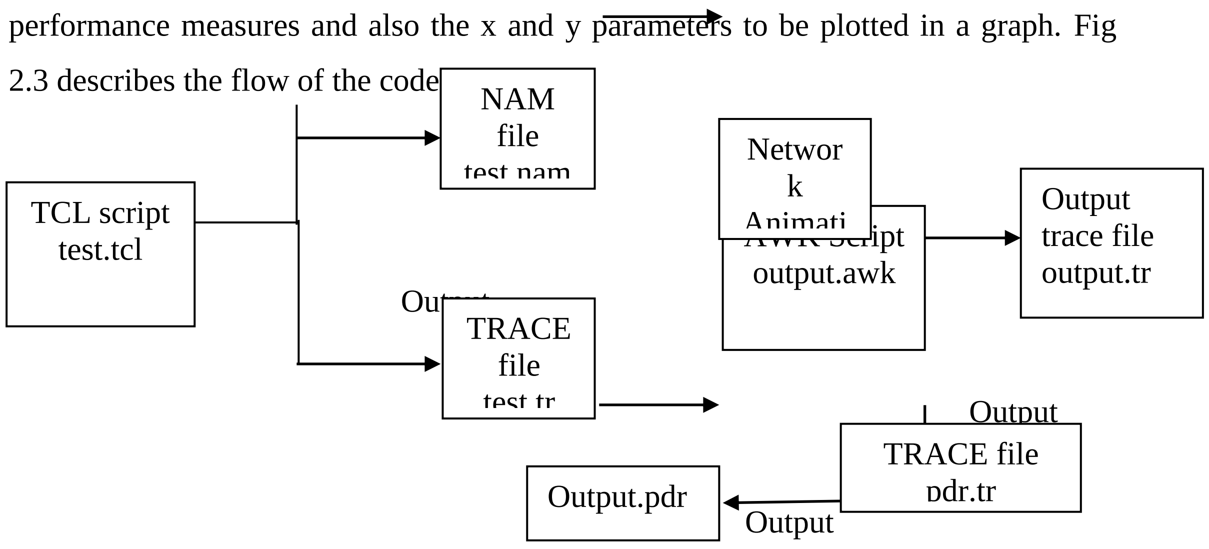


Fig 2.3 Flow of the code

2.4.2.1 Overview of TCL Programming

Tcl (Tool Command Language) is a dynamic programming/scripting language based on concepts of Lisp, C, and Unix shells. It can be used interactively, or by running scripts (programs) which can use a package system for structuring, hence allowing to do much with little code. Tcl is available for Linux, Windows, Mac OS X, as well as other platforms, as open-source software under BSD-like license, or as pre-built binaries.

The following rules define the syntax and semantics of the Tcl language:

Commands

A Tcl script is a string containing one or more commands. Semi-colons and newlines are command separators unless quoted as described below. Close brackets are command terminators during command substitution (see below) unless quoted.

Evaluation

A command is evaluated in two steps. First, the Tcl interpreter breaks the command into words and performs substitutions as described below. These substitutions are performed in the same way for all commands. The first word is used to locate a command procedure to carry out the command, then all of the words of the command are passed to the command procedure. The command procedure is free to interpret each of its words in any way it likes, such as an integer, variable name, list, or Tcl script. Different commands interpret their words differently.

Words

Words of a command are separated by white space (except for newlines, which are command separators).

Double quotes

If the first character of a word is double-quote (") then the word is terminated by the next double-quote character. If semi-colons, close brackets, or white space characters (including newlines) appear between the quotes then they are treated as ordinary characters and included in the word. Command substitution, variable substitution, and backslash substitution are performed on the characters between the quotes as described below. The double-quotes are not retained as part of the word.

Braces

If the first character of a word is an open brace ({) then the word is terminated by the matching close brace (}). Braces nest within the word: for each additional open brace there must be an additional close brace (however, if an open brace or close brace within the word is quoted with a backslash then it is not counted in locating the matching close brace). No substitutions are performed on the characters between the braces except for backslash-newline substitutions described below, nor do semi-colons, newlines, close brackets, or white space receive any special interpretation. The word will consist of exactly the characters between the outer braces, not including the braces themselves.

Command substitution

If a word contains an open bracket ([) then Tcl performs command substitution. To do this it invokes the Tcl interpreter recursively to process the characters following the open bracket as a Tcl script. The script may contain any number of commands and must be terminated by a close bracket (``]). The result of the script (i.e. the result of its last command) is substituted into the word in place of the brackets and all of the characters between them. There may be any number of command substitutions in a single word. Command substitution is not performed on words enclosed in braces.

Variable substitution

If a word contains a dollar-sign ($) then Tcl performs variable substitution: the dollar-sign and the following characters are replaced in the word by the value of a variable. Variable substitution may take any of the following forms:

$name

Name is the name of a scalar variable; the name is a sequence of one or more characters that are a letter, digit, underscore, or namespace separators (two or more colons).

$name(index)

Name gives the name of an array variable and index gives the name of an element within that array. Name must contain only letters, digits, underscores, and namespace separators, and may be an empty string. Command substitutions, variable substitutions, and backslash substitutions are performed on the characters of index.

${name}

Name is the name of a scalar variable. It may contain any characters whatsoever except for close braces. There may be any number of variable substitutions in a single word. Variable substitution is not performed on words enclosed in braces.

Backslash substitution

If a backslash (\) appears within a word then backslash substitution occurs. In all cases but those described below the backslash is dropped and the following character is treated as an ordinary character and included in the word. This allows characters such as double quotes, close brackets, and dollar signs to be included in words without triggering special processing. The following table lists the backslash sequences that are handled specially, along with the value that replaces each sequence.

\a - Audible alert (bell) (0x7).

\b - Backspace (0x8).

\f - Form feed (0xc).

\n - Newline (0xa).

\r - Carriage-return (0xd).

\t - Tab (0x9).

\v - Vertical tab (0xb).

\<newline>whitespace

A single space character replaces the backslash, newline, and all spaces and tabs after the newline. This backslash sequence is unique in that it is replaced in a separate pre-pass before the command is actually parsed. This means that it will be replaced even when it occurs between braces, and the resulting space will be treated as a word separator if it isn't in braces or quotes.

\\ **-** Literal backslash (\), no special effect.

\ooo

The digits ooo (one, two, or three of them) give an eight-bit octal value for the Unicode character that will be inserted. The upper bits of the Unicode character will be 0.

\xhh

The hexadecimal digits hh give an eight-bit hexadecimal value for the Unicode character that will be inserted. Any number of hexadecimal digits may be present; however, all but the last two are ignored (the result is always a one-byte quantity). The upper bits of the Unicode character will be 0.

\uhhhh

The hexadecimal digits hhhh (one, two, three, or four of them) give a sixteen-bit hexadecimal value for the Unicode character that will be inserted. Backslash substitution is not performed on words enclosed in braces, except for backslash-newline as described above.

Comments

If a hash character (#) appears at a point where Tcl is expecting the first character of the first word of a command, then the hash character and the characters that follow it, up through the next newline, are treated as a comment and ignored. The comment character only has significance when it appears at the beginning of a command.

Order of substitution

Each character is processed exactly once by the Tcl interpreter as part of creating the words of a command. For example, if variable substitution occurs then no further substitutions are performed on the value of the variable; the value is inserted into the word verbatim. If command substitution occurs then the nested command is processed entirely by the recursive call to the Tcl interpreter; no substitutions are performed before making the recursive call and no additional substitutions are performed on the result of the nested script. Substitutions take place from left to right, and each substitution is evaluated completely before attempting to evaluate the next. Thus, a sequence like

set y [set x 0][incr x][incr x]

will always set the variable y to the value, 012.

Comments

The first rule for comments is simple: comments start with # where the first word of a command is expected, and continue to the end of line (which can be extended, by a trailing backslash, to the following line)

2.4.2.2 AWK SCRIPTS

AWK is a full-featured text processing language with syntax reminiscent of C. AWK breaks each line of input passed to it into fields. By default, a field is a string of consecutive characters delimited by whitespace, though there are options for changing this. AWK parses and operates on each separate field. This makes it ideal for handling structured text files. In this project AWK script is used to calculate the delay and overhead.

In its simplest usage AWK is meant for processing column-oriented text data, such as tables, presented to it on standard input. The variables $1, $2, and so forth are the contents of the first, second, etc. column of the current input line. For example, to print the second column of a file use the following simple AWK script:

```
AWK < file '{ print $2 }'
```

This means "on every line, print the second field".

To print the second and third columns use

```
AWK < file '{ print $2, $3 }'
```

AWK is a weakly typed language; variables can be either strings or numbers, depending on how they're referenced. All numbers are floating-point. It has some built-in variables that are automatically set; $1 and so on are examples of these. The other built-in variables that are useful for beginners are generally NF, which holds the number of fields in the current input line ($NF gives the last field), and $0, which holds the entire current input line.

You can make your own variables, with whatever names you like (except for reserved words in the AWK language) just by using them. You do not have to declare variables. Variables that haven't been explicitly set to anything have the value "" as strings and 0 as numbers.

AWK includes a printf statement that works essentially like C printf. This can be used when you want to format output neatly or combine things onto one line in more complex ways.

The essential organization of an AWK program follows the form:

pattern { action }

The pattern specifies when the action is performed. Like most UNIX utilities, AWK is line oriented. That is, the pattern specifies a test that is performed with each line read as input. If the condition is true, then the action is taken.

The default pattern is something that matches every line. This is the blank or null pattern. Two other important patterns are specified by the keywords "BEGIN" and "END".

These two words specify actions to be taken before any lines are read, and after the last line is read. The AWK program below:

```
BEGIN
{ print "START" }
{ print         }
END   { print "STOP"  }
```

adds one line b

The expression is reevaluated each time the rule is tested against a new input record. If the expression uses fields such as $1, the value depends directly on the new input record's text; otherwise, it depends only on what has happened so far in the execution of the awk program, but that may still be useful.

There are only a few commands in AWK. The list and syntax follows:

```
if ( conditional ) statement [
else statement ]
while ( conditional ) statement
for
( expression ; conditional ; expression ) sta
tement
for ( variable in array ) statement
break
continue
{ [ statement ] ...}
variable=expression
```

2.4.3 Wireless Network Simulation using NS2

The general process of creating a simulation can be divided into several steps:-

Topology definition:- To ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process.

Model usage:- Models are added to simulation (for example, UDP, IPv4, point-to-point devices and links, applications); most of the time this is done using helpers.

Node and link conFiguration:- Models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system.

Execution:- Simulation facilities generate events, data requested by the user is logged.

Performance analysis:- After the simulation is finished and data is available as a time-stamped event trace. This data can then be statistically analysed with tools like R to draw conclusions.

Graphical Visualization:- Raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or Xgraph. Xgraph is the plotting tool bundled with many of the installation pack

2.4.3.1 Code snippet to create a wireless network

```
# Simulator Instance Creation
set ns [new Simulator]
# Fixing the co-ordinate of
simulation area
set val(x) 500
set val(y) 500
```

```
# Define options
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround;# radio-propagationmodel
set val(netif) Phy/WirelessPhy            ;# networkinterface type
set val(mac)   Mac/802_11                 ;# MAC type
set val(ifq)  CMUPriQueue              ;# interface queue type
set val(ll) LL                         ;# link layer type
set val(ant)  Antenna/OmniAntenna     ;# antenna model
```

Configuring the network

```
    $ns node-conFig -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF \
                 -movementTrace ON \
```

Defining communication between the nodes

```
# Defining a transport agent for sending
set udp [new Agent/UDP]
# Attaching transport agent to sender node
$ns attach-agent $node_($sur($i)) $udp
# Defining a transport agent for receiving
set null [new Agent/Null]
# Attaching transport agent to receiver node1
$ns attach-agent $node_($des($i)) $null
#Connecting sending and receiving transport agents
$ns connect $udp $null
#Defining Application instance
set cbr [new Application/Traffic/CBR]
# Attaching transport agent to application agent
$cbr attach-agent $udp
#Packet size in bytes and interval in seconds definition
$cbr set packetSize_ 512
$cbr set interval_ 0.1
# data packet generation starting time
$ns at $now "$cbr start"
# data packet generation ending time
$ns at [expr $now + 10 ] "$cbr stop"
```

operating context.

It consists of two parts: a dynamic link library (DLL) as an API compatibility layer providing a substantial part of the posix API functionality, and an extensive collection of software tools and applications that provide a Unix-like look and feel.

Cygwin is not a full operating system, and so must rely on Windows for accomplishing some tasks. For example, Cygwin provides a POSIX view of the Windows file system, but does not provide file system drivers of its own. Therefore part of using Cygwin effectively is learning to use Windows effectively. Many Windows utilities provide a good way to interact with Cygwin's predominately command-line environment. For example, ipconFig.exe provides information about network conFiguration, and net.exe views and conFigures network file and printer resources. Most of these tools support the /? switch to display usage information.

Windows programs do not understand POSIX pathnames, so any arguments that reference the file system must be in Windows (or DOS) format or translated. Cygwin provides the cygpath utility for converting between Windows and POSIX paths. The same format works for most Windows programs, for example

```
notepad.exe "$(cygpath -aw "Desktop/Phone Numbers.txt")"
```

Once Cygwin is installed, ns-2 can be installed from its all-in-one package. This file must be extracted from the location using commands like the following:

```
gzip –d ns-allinone-2.27.tar.gz
tar –xvf ns-allinone-2.27.tar
```

The files will be extracted into a new directory with name “ns-all-in-one”, “./install” can be used to install NS2

CHAPTER 3
METHODOLOGY

3.1 APPROACHES TOWARDS THE PROJECT

The enhancement in dynamic source routing algorithm can be done by implementing the mechanism through which the overhead on the source node gets disturbed. To implement this mechanism we considered one hop neighbours, then identified the two hop neighbours through which the route selection algorithm was established. The approach can be seen in the Figure

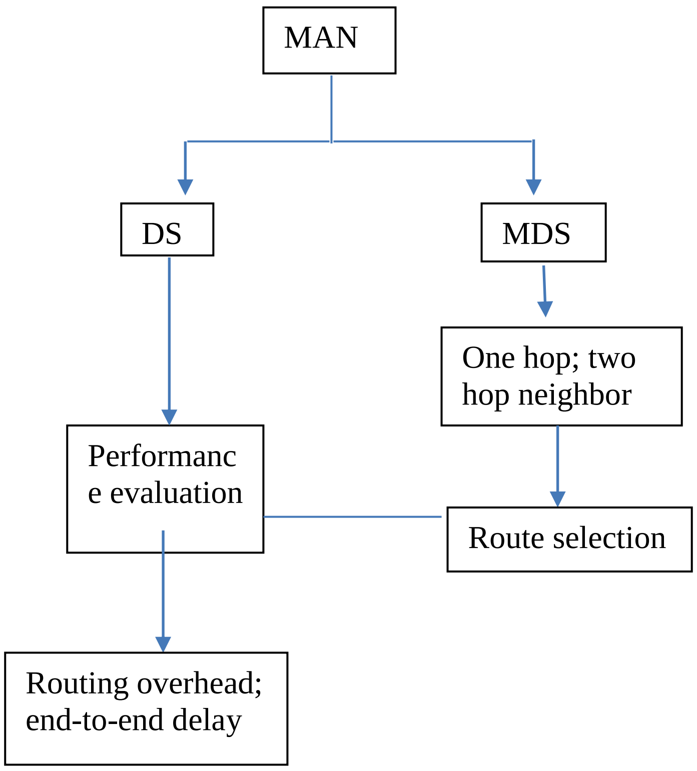


Fig 3.1 Approach towards the project

3.2 Route selection algorithm

One hop neighbours are selected based upon the distance of a particular node from source node. For example if the nodes are expanded till 500mm then the one hop neighbours are assumed to lie within 200mm. the same selection criteria is followed for selecting the two hop neighbours even. Thus the two hop neighbours are selected with the help of one hop neighbour.

The following are the steps to implement RS:

```
Let one hop neighbor=N1; two hop neighbor set =N2
Let RS= NULL
Add to RS the nodes in N1 which are the only nodes to provide reachability
to a node in N2 or the nodes with degree one.
Remove the nodes from N2 which are now covered by a node in RS
Calculate the reachability for each node in N1 if n2 is not null
Add to RS the node that provides highest reachability
Remove the nodes from N2 that are covered by RS
```

There is other feature included in this project where we consider the bandwidth of the node. Source node selects the neighbour based on bandwidth. If the node is not having sufficient bandwidth to broadcast the packet, then the information will not be reached to destination.

3.3 Performance measurement terms

Routing Overhead:

Overhead is defined as total number of beacon update messages involved in the communication.

Overhead = Number of messages involved in beacon update process.

End-to-end delay (or average delay):

Delay is the average time difference between the time a data packet is sent by the source node and the time it is successfully received by the destination node.

CHAPTER 4
IMPLEMENTATION

The implementation phase of any project development is the most important phase as it yields the final solution, which solves the problem at hand. The different modules that comprise the project are detailed here. The software used to develop these modules is NS 2.35 and Cygwin. The scripts used are TCL and AWK

4.1 SAMPLE IMPLEMENTATION CODES

4.1.1 Implementation code in tcl script for setting up the mobile nodes

Two nodes are created with the conFiguration mentioned in 2.4.3.1

```
set node_(0) [$ns node]

set node_(1) [$ns node]

# Initialisation of nodes with x and y coordinates
$node_(0) set X_ 200
$node_(0) set Y_ 250
$node_(0) set Z_ 0.0
$node_(1) set X_ 300
$node_(1) set Y_ 250
$node_(1) set Z_ 0.0
# defining the initial position of nodes in NAM
$ns initial_node_pos $node_(0) 50
$ns initial_node_pos $node_(1) 30
```

```
  # Initial node and labelling the node with colour
"magneta" at the start of simulation
  for {set i 0} {$i < $val(nn)} { incr i } {
  $node_($i) color magenta
  $ns at 0.0 "$node_($i) color Cyan"
  }
   specifying the nodes when the simulation ends
   {set i 0} {$i < $val(nn) } { incr i } {
     $ns at $val(stop) "$node_($i) reset"; }
    #files can be attached to the source code by using the
keyword "source"
     source ./label.tcl
     source ./color.tcl
     source ./comm.tcl
     source ./mobi.tcl
     source ./traceannotate.tcl
# Ending NAM and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
#stop procedure will be called at the time $val(stop)
$ns at $val(stop) "stop"
$ns at 10.00 "puts \"end simulation\"";
```

```
#stop procedure:
proc stop {} { global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
    exec nam test.nam &
}
```

If we ve have to

decla s declared

outside the procedure. The simulator method "flush-trace will dump the traces on the respective files defined before. This command flushes the trace buffer and is typically called before the simulation run ends. exec executes the nam program for visualization.

In following execution statement .awk file process the trace file and display the result in command prompt or send the result to another file.

```
exec awk -f output.awk test.tr &
exec awk -f output.awk test.tr > output.tr &
exec awk -f pdr.awk test.tr > pdr.tr &
```

Following execution statement generates the xgraph having the coordinates available in trace file.

```
exec xgraph -bg white -t OVERHEAD_FILE vs TIME -x TIME -y
OHD(%)  ohd.tr &
```

The command exit will ends the application and return number 0 as status to the system.

```
exit 0
}
```

#simulation begin using the command "ns" : `$ns run`

4.1.2 Sample Awk Script to Calculate Overhead

```
BEGIN {
    #initialization
generated_packets = 0;
}
{
   # Pattern and action
if ($1 == "s" && $7 == "DSR") {
generated_packets++;
}
}
END {# Calculation and result printing
print "generated packets = "generated_packets;
}
```

```
BEGIN {# Calculation of Delay# initialization
     Total_Delay = 0;
     Count = 0; }
{    # Pattern and action
     if($1 == "s" && $4 == "AGT") {
     packet_sequence_number = $6;
     }
     else if($1 == "r" && $4 == "AGT") {
     receiving_time[$6] = $2;
     }
     else if ($1 == "D" && $7 == "cbr") {
     receiving_time[$6] = -1;
     }
}
END {# Calculation and result printing
for (i=0; i<=packet_sequence_number; i++) {
if (receiving_time[i] > 0) {
delay[i] = receiving_time[i] - sending_time[i];
Total_Delay = Total_Delay + delay[i];
count++;
```

4.1.4 Sample code for one hop neighbors

```
        for {set k 0} {$k < $count($source) } { incr k } {

          $ns at 0.0 "$node_($n($source-$k)) color
        darkviolet"

          set onehopneighbors($k) $n($source-$k)

          puts $dsr $onehopneighbors($k)

          set tmark($onehopneighbors($k)) 1

          $ns at 0.0 "$node_($onehopneighbors($k)) label
        onehopneighbor"
}
```

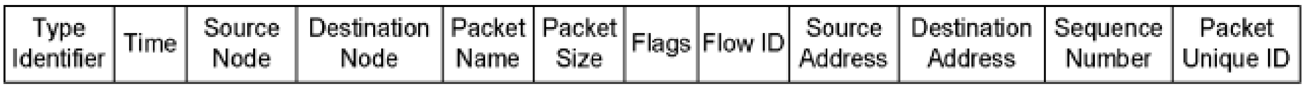
4.2.1 Syntax for Creating Trace File

```
# Open the trace file
set nf [open out.tr w]
$ns trace-all $nf
```

The above line means that we are opening a new trace file named as "out" and also telling that data must be stored in .tr [trace] format.
"nf" is the file handler that we are used here to handle the trace file.
"w" means write i.e the file out.tr is opened for writing.
>> "r" means reading and "a" means appending

The second line tells the simulator to trace each packet on every link in the topology and for that we give file handler nf for the simulator ns.

4.2.2 Trace File Format



Type Identifier	Time	Source Node	Destination Node	Packet Name	Packet Size	Flags	Flow ID	Source Address	Destination Address	Sequence Number	Packet Unique ID

Fig 4.1 Trace file format

For example consider the following trace string:
s 21.500275000 _0_ MAC --- 0 AODV 106 [0 ffffffff 0 800] ------ [0:255 -1:255 30 0] [0x2 1 4 [1 0] [0 10]] (REQUEST)
Here is the interpretation
The node “_0_” sends (i.e., “s”) at time “21.500275” second.
The trace level is at the “MAC” layer.
The packet has the unique ID of “0”, contains an “AODV” payload type, and is “106” bytes in size.
The MAC protocol assumes that the delay over the underlying wireless channel is zero “0”.
The source and destination MAC addresses are “0” and “ffffffff”, respectively.
This is an IP packet running over an Ethernet network (i.e., “800”).
The IP source and destination addresses of “0” and “1”, respectively.
The ports for both source and destination are “255”.
The time to live and the address of the next hop node are “30” hops and “0”, respectively.
This is an RREQ packet tagged with the ID “0x2”.
The number of hop counts is “1” and the broadcast ID is “4”.
The destination IP address and sequence number are 1 and 0, respectively.

The source IP address and sequence number are 0 and 10, respectively.
The string “(REQUEST)” confirms that this is the RREQ packet.

CHAPTER 5
TESTING AND RESULTS

5.1 TESTING

Testing is the practice of making objective judgments regarding the extent to which the project meets, exceeds or fails to meet stated objectives. Projects are tested to know the efficiency and the complete functionality.

5.1.1 Assumptions

- The queue length in this project is restricted to 50 nodes so we can have nodes in the network from 0 to 49. The number of nodes in each simulation is mentioned in the command line arguments
- The source node and the destination node can be fixed in the code as per the requirement
- The dimensions of the network can be modified as per the requirement.

5.1.2 Testing the Project

The project is tested based on all the assumptions. The first assumption we are considering here would be the number of nodes.

5.1.2.1 Testing Based on Number of Nodes

For testing the algorithm based on this particular criterion, vary the number of nodes for checking the variance in output:

Initially we are considering the number of nodes as 30 and later increasing the nodes to 40 and 50:

Number of nodes: 30

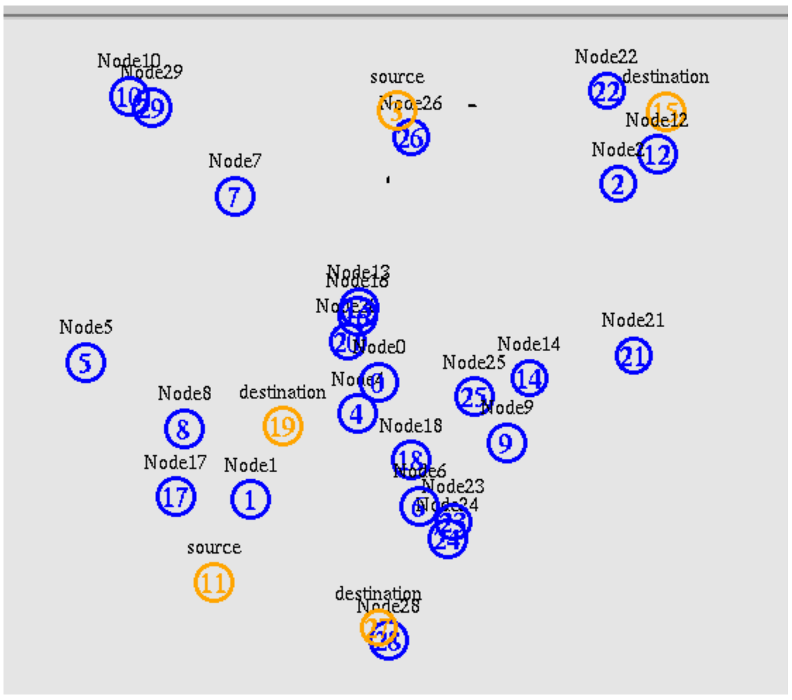


Fig 5.1 DSR with 30 nodes

Number of nodes: 40

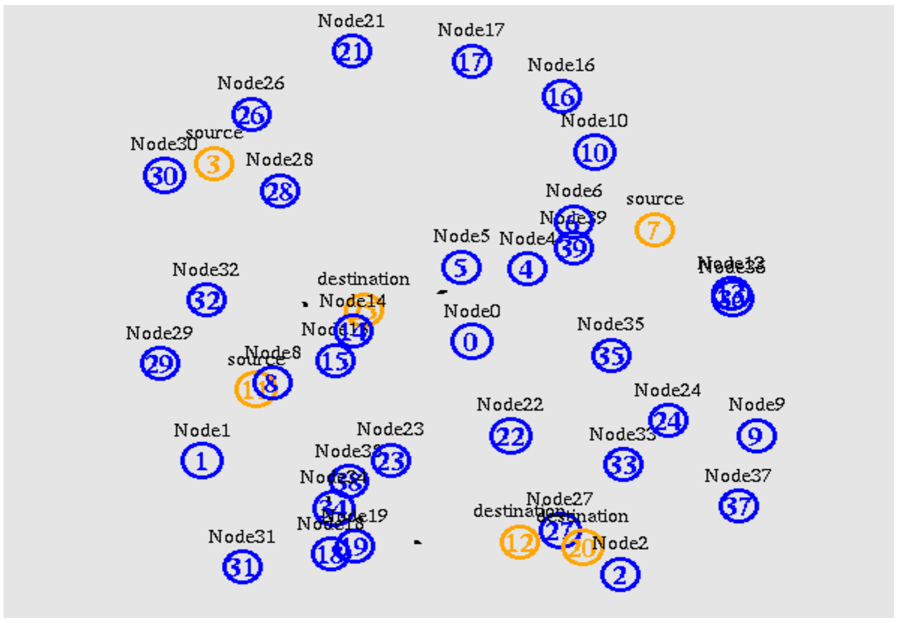


Fig 5.2 DSR with 40 nodes

Number of nodes: 50

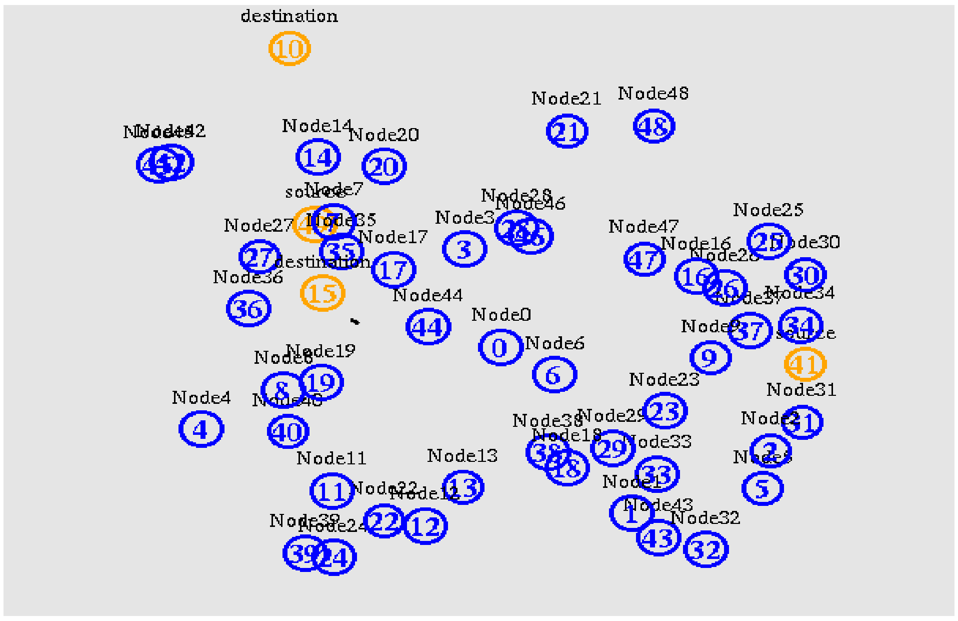


Fig 5.3 DSR with 50 nodes

5.1.2.2 Testing based on the dimensions of topology

In this section, we are testing the project based upon the dimensions of the network topology.

Dimension: 500x500 scale: mm

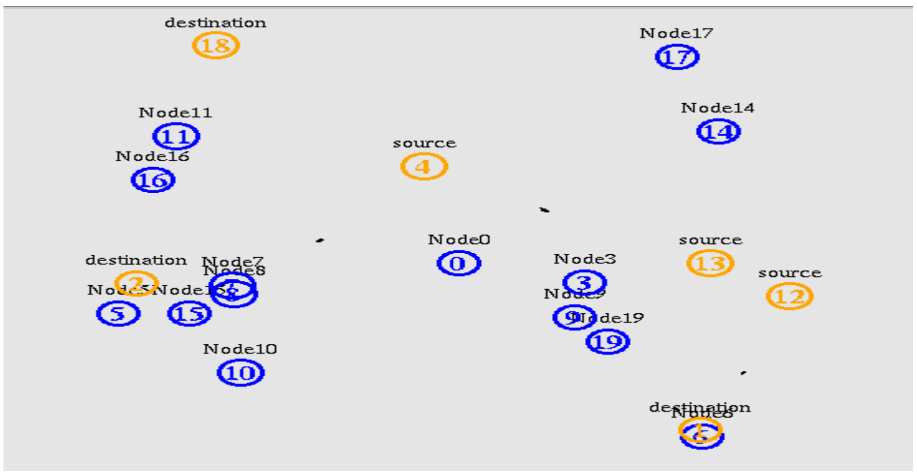


Fig 5.3 dimensions 500 x 500

Dimension: 1000x1000 scale: mm

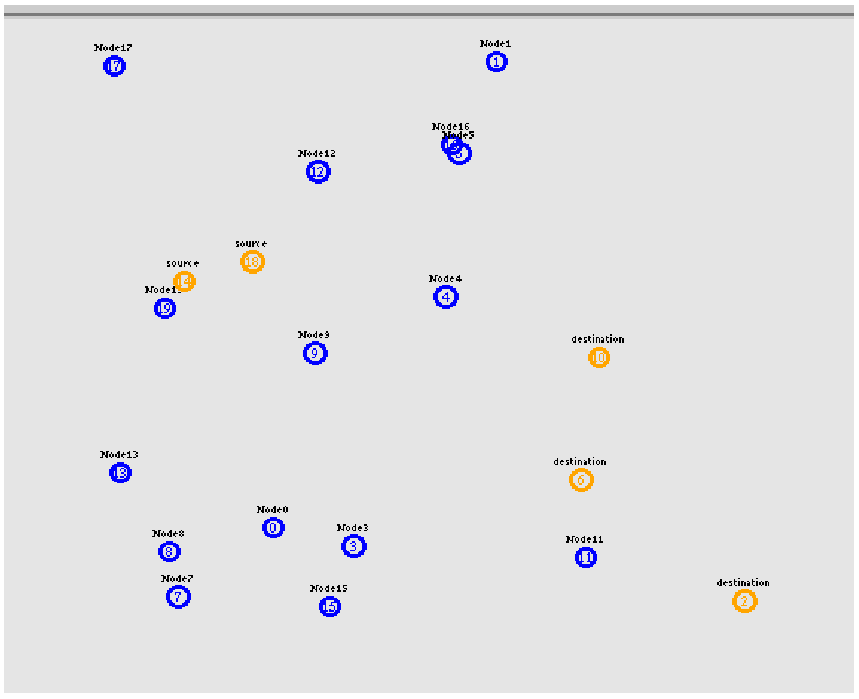


Fig 5.4 dimensions 1000 x 1000

5.1.2.3 Testing based on source node and destination node

Source node=7;   Destination node=17

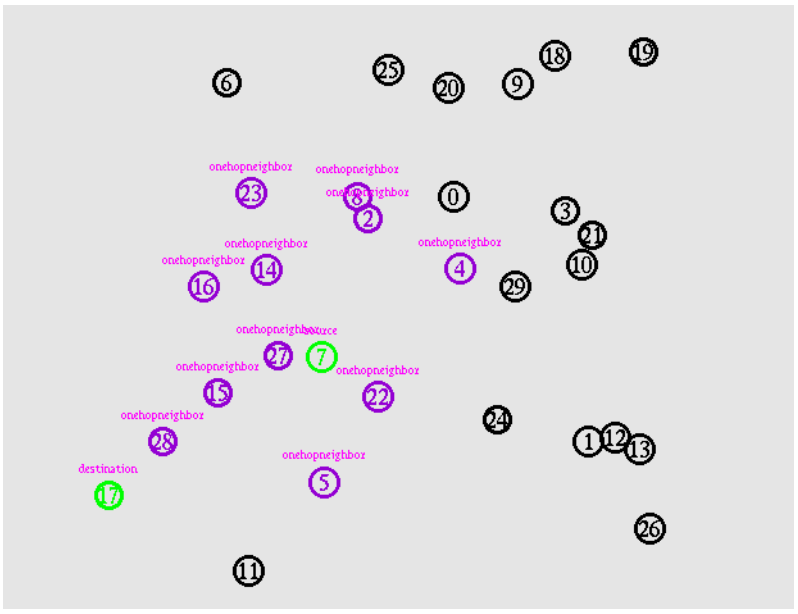


Fig 5.6 source 7 destination 17

Source node=12; Destination node=36

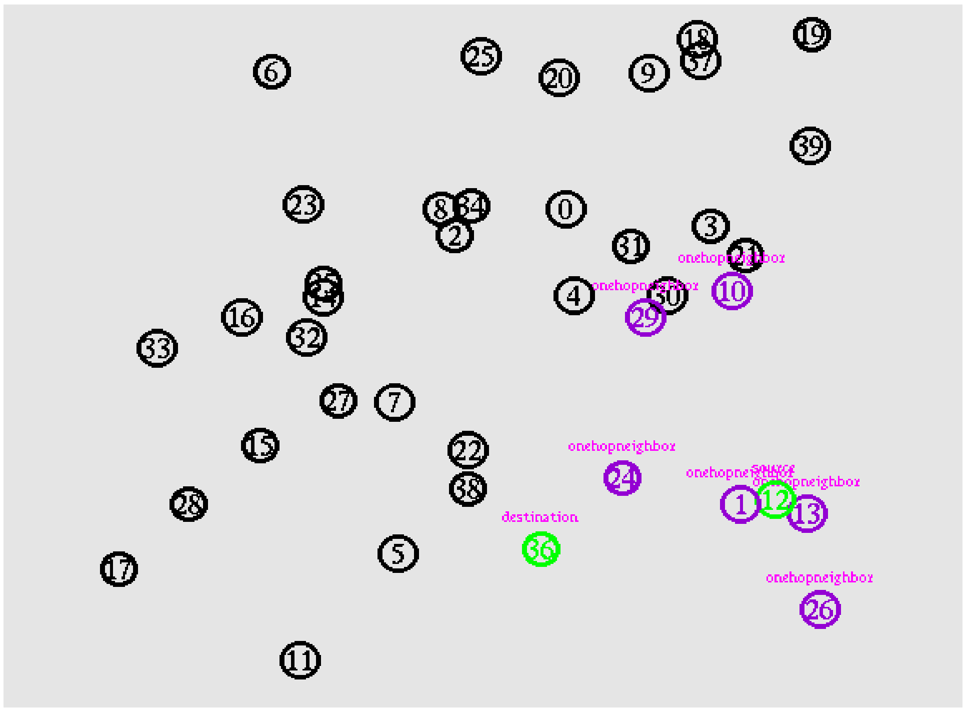


Fig 5.7 source 12 destination 36

5.2 RESULTS

The project is tested according to the assumptions mentions in the previous section and the results obtained are as follows.

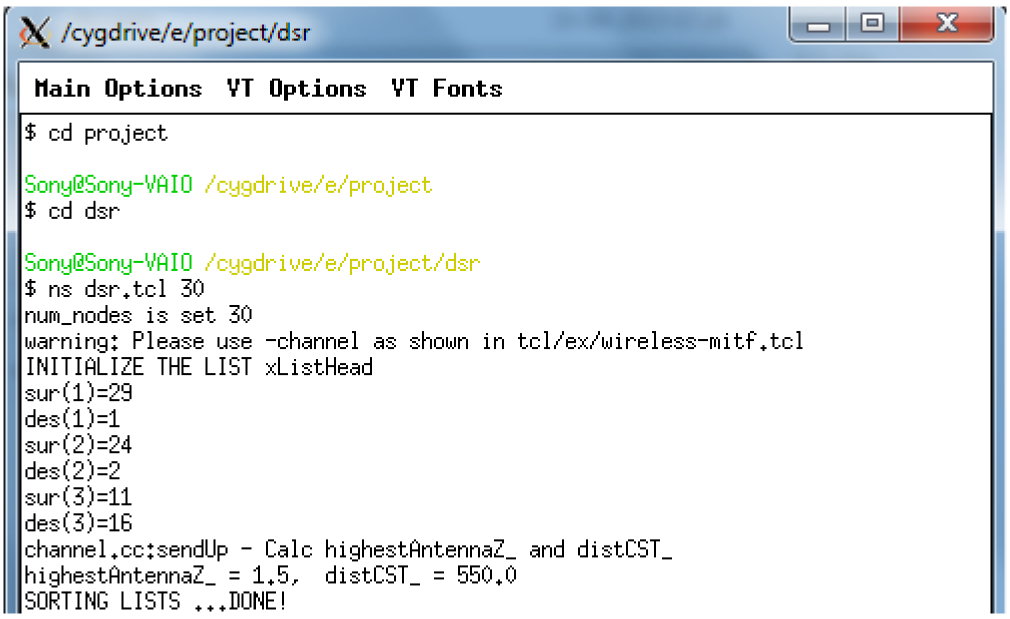
5.2.1 DSR Algorithm

This is of the form

ns dsr.tcl 30

Here 30 specifies the number of nodes

After the dsr.tcl file is executed, two files are generated. One is dsr.tr and dsr.nam. dsr.tr is the trace file which stores all the events that have occurred during the simulation and dsr.nam serves as an input file to the network animator. The simulation can be visualized using the network animator



```
$ cd project

Sony@Sony-VAIO /cygdrive/e/project
$ cd dsr

Sony@Sony-VAIO /cygdrive/e/project/dsr
$ ns dsr.tcl 30
num_nodes is set 30
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
sur(1)=29
des(1)=1
sur(2)=24
des(2)=2
sur(3)=11
des(3)=16
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
```

Fig 5.8 tcl file execution command

As NS2 is a packet level simulator, Network Animator shows only the transfer of packets. There are two kind of packets which are being transferred in DSR algorithm. CBR packet represents the constant bit rate data packet and DSR packet represents the route request packet of the DSR algorithm.

The frequency of packet generation can also be controlled. The animation time between two events can be controlled manually using NAM. The below Figure shows the working of the basic DSR algorithm:

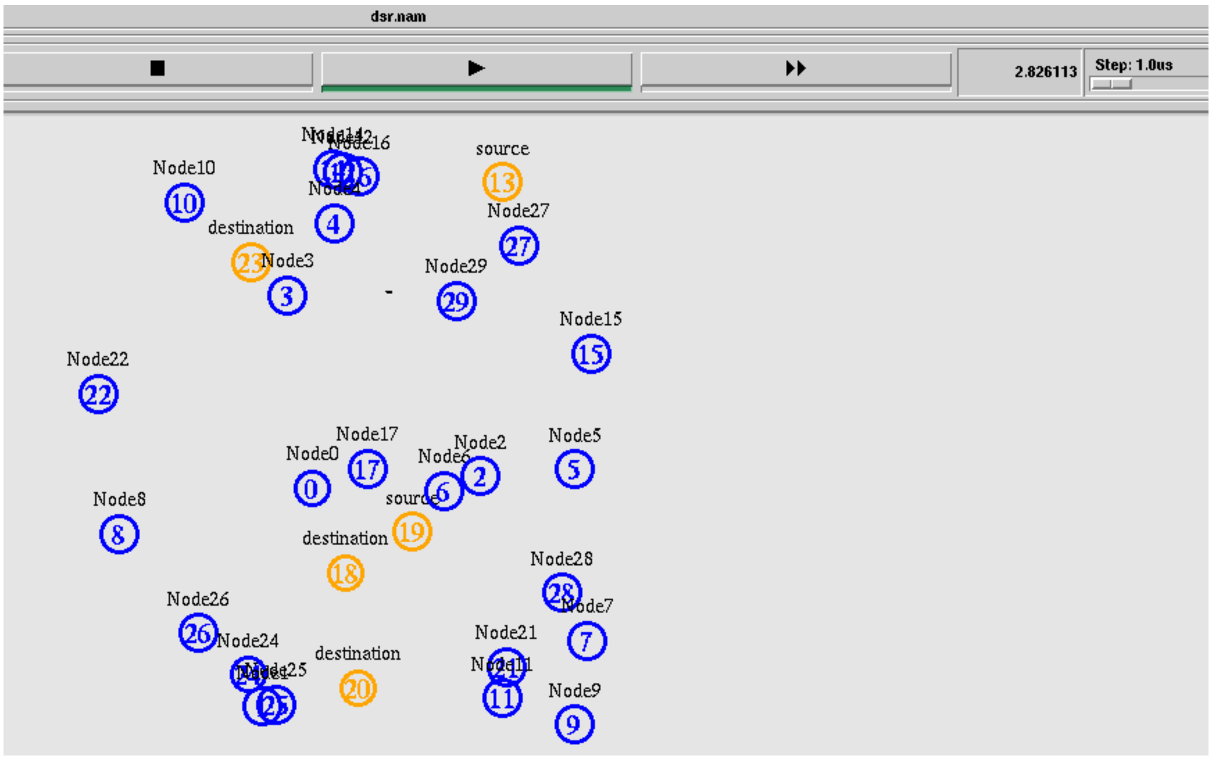


Fig 5.9 DSR algorithm

5.2.2 One Hop Neighbor

The source node and destination node can be fixed. Based on the source node the on hop neighbors of the source node are selected.

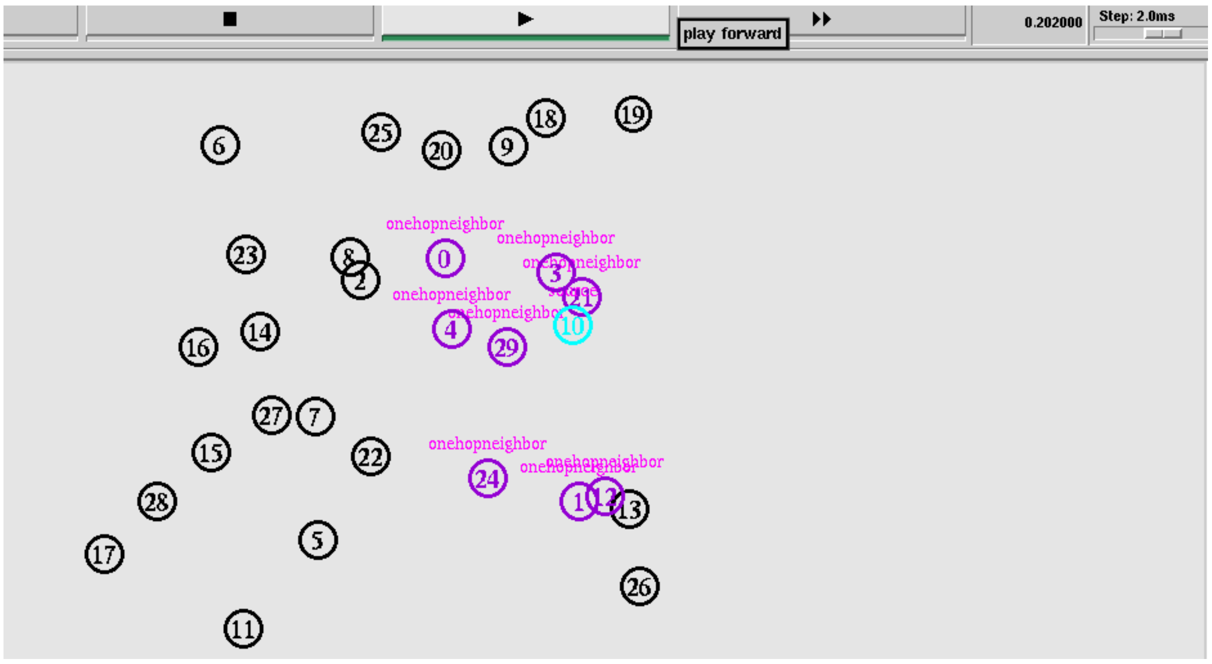


Fig 5.10 one hop neighbors

All those nodes that are at a distance of less than or equal to 200 mm are selected as the one hop neighbors. The topology of the nodes is configured as a flat grid with dimensions 500 x 500 mm. The position of the initial node is fixed and the positions of the remaining nodes are calculated by performing some arithmetic between the initial node position and the rand function.

5.2.3 Two Hop Neighbors

From the one hop neighbor set the two hop neighbors are calculated.

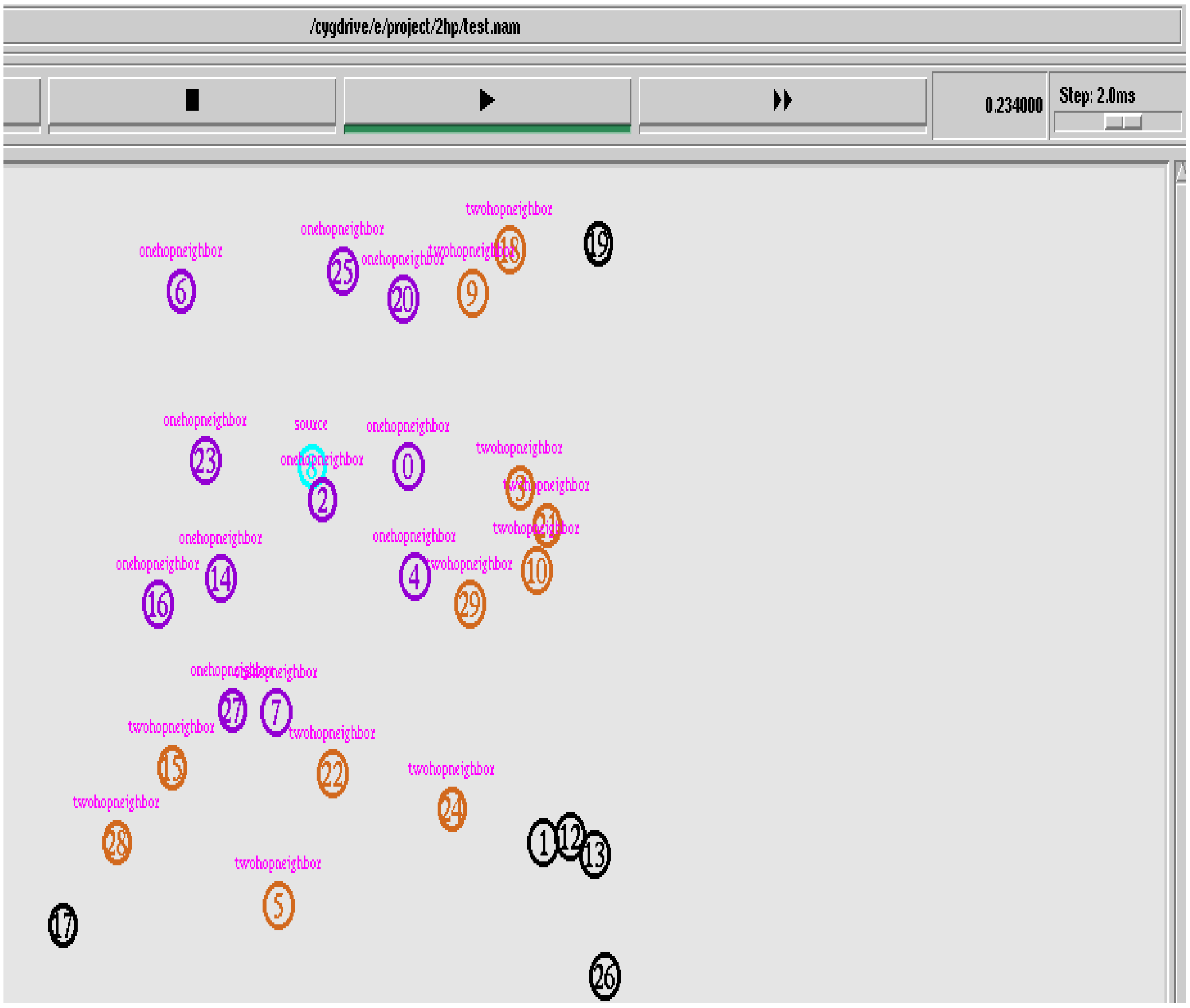


Fig 5.11 two hop neighbor

5.2.4 Modified DSR

The RS node set is calculated and the node with the highest reachability is taken to be the RS node

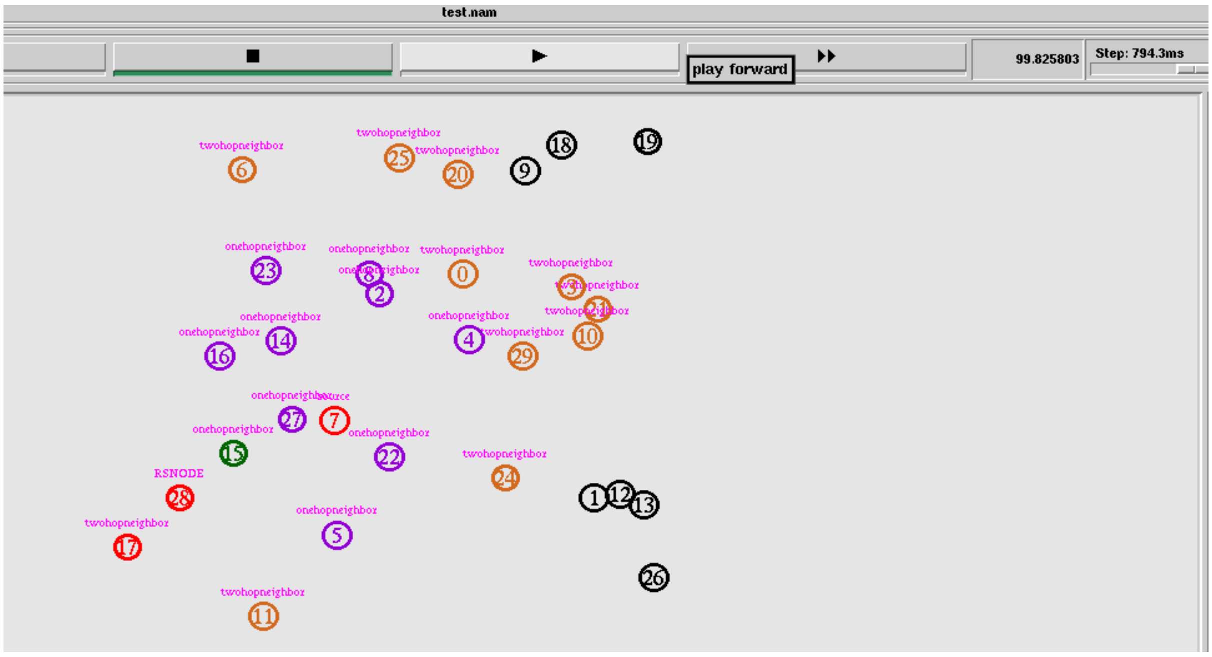


Fig 5.12 Modified DSR

5.2.5 Graph

Graph is plotted using the xgraph.

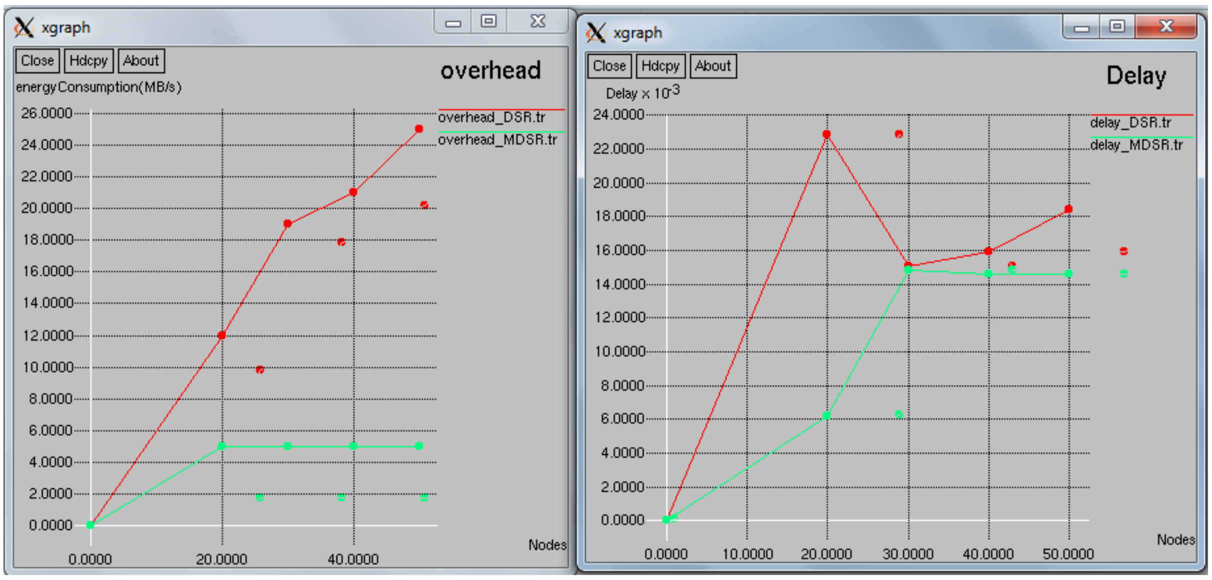


Fig 5.13 Comparative graph

CHAPTER 6

CONCLUSION AND FUTURE WORK

Increase in the growth of wireless devices all around the world has led to the emergence of MANET. The routing protocols are important because they specify the way in which routers communicate with each other. Dynamic

Routing protocols play an important role in recent times as most of the networks are Ad-hoc.

DSR protocol uses a reactive approach which eliminates the need to periodically flood the network with table update messages which are required in a table-driven approach. In a reactive approach such as this, a route is established only when it is required and hence the need to find routes to all other nodes in the network as required by the table-driven approach is eliminated. The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead

In this project, the existing DSR protocol and the MDSR protocol which uses the route selection mechanism have been implemented. The control overhead in reacting routing protocol can be minimized by implementing MDSR. The existing DSR and MDSR are compared to check for the efficiency. The results of comparison are obtained in the form of a graph .MDSR helps to reduce the number of packets in DSR routing protocol and also reduce the rebroadcast messages. It reduces the total overhead in the network.

FUTURE WORK:

Future work aims at limiting the replies sent by destination, keeping only one route per destination and preferring fresher routes over shorter ones. While multiple routes may benefit at higher traffic loads, keeping only one route per destination helps sender nodes gather routes when the topology changes.

BIBLIOGRAPHY

[1] Changling Liu, Jorg Kaiser, "A Survey of Mobile Ad Hoc network Routing Protocols", the University of Magdeburg, October 2005

[2] Chandra .R, V. Ramasubramanian, and K.P. Birman, “Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks,” Proc. of IEEE Int’l conf. on Distributed Computer System (ICDCS’2001), pp. 275– 283, Apr. 2001.

[3] PENG Wei and LU Xichengand LU Xicheng, “AHBP: An Efficient Broadcast Protocol for Mobile Ad Hoc Networks” Department of Computer Science, Changsha Institute of Technology, Vol.16 No.2 March 2001

[4] B.Williams and T. Camp, “Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks” Proc.of3rdACMInt, Symposium on Mobile AdHoc Networking and Computing(MOBIHOC’2002),pp.194-205,Jun.2002

[5] Neighbour Coverage: A Dynamic Probabilistic Route Discovery for Mobile Ad Hoc Networks

Web resources

1. http://ns2blogger.blogspot.in/p/the-easiest-way-to-install-ns-2-on_5.html?m=1
2. http://csis.bits-pilani.ac.in/faculty/murali/resources/tutorials/ns2.htm
3. http://ciemcal.org/manet-and-routing-techniques/
4. http://ns2tutor.weebly.com/throughput-calculation-using-ns2.html

Text Books

1. Introduction to network simulator by Teerawat Issariyakul, Ekram Hossain
2. TCL and TK programming for the Absolute Beginner by Kurt wall

A

PROJECT REPORT

on

ENHANCED DSR ALGORITHM WITH REDUCED DELAY AND OVERHEAD

Submitted in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

N.NIHARIKA	**(160110737014)**
R.SINDHUJA	**(160110737025)**

Under the esteemed guidance of

Dr. K. Radhika

Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

(Affiliated to Osmania University; accredited by NBA (AICTE), ISO certified 9001:2000)

GANDIPET – 500 075

2014-2015

Chaitanya Bharathi Institute of Technology

(Affiliated to Osmania University : accredited by NBA (AICTE), ISO Certified 9001:2000)

Chaitanya Bharathi P.O., Gandipet, Hyderabad - 500 075.

e.mail: ircrcbit@yahoo.com; Website : www.cbit.ac.in

☎ (O) 08413-233356; 958413-233356; 958413-233865; Fax : 958413 - 232099

CERTIFICATE

This is to certify that the project work entitled **ENHANCED DSR ALGORITHM WITH REDUCED DELAY AND OVERHEAD** submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY,** in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology, during the academic year 2014-15, is a record of original work done by **N.NIHARIKA (160110737014), R.SINDHUJA (160110737025)** during the period of study in the Dept. of IT, CBIT, HYDERABAD, under our supervision and guidance.

.

Project Guide

Dr. K. Radhika
Associate Professor,
Department of IT,
CBIT, Hyderabad.

Head of the Department

Dr. Suresh Pabboju
Professor and Head,
Department of IT,
CBIT, Hyderabad.

DECLARATION

This is to certify that the work reported in the present report titled **ENHANCED DSR ALGORITHM WITH REDUCED DELAY AND OVERHEAD** is a record of work done by us in the Department of Information Technology, Chaitanya Bharathi Institute of Technology, Osmania University.

No part of this is copied from books / journals / Internet and wherever the portion is taken the same have been duly referred in the text. This report is based on the project work done entirely by us and not copied from any other source.

N.NIHARIKA (160110737014)
R.SINDHUJA (160110737025)

ACKNOWLEDGEMENT

I take this opportunity to thank all who have rendered their full support to our work. The pleasure, the achievement, the glory, the satisfaction, the reward, the appreciation and the construction of our project seminar cannot be thought without a few, how apart from their regular schedule, spared a valuable time for us. This acknowledgement is not just a position of words but also an account of the indictment. They have been a guiding light and source of inspiration towards the completion of the project seminar work.

We are grateful to our project guide **Dr. K. Radhika** Asst. Prof., Dept of IT, CBIT for her kind and timely help offered to us in this project report.

We take the opportunity to express our thanks to **Dr. Suresh Pabboju**, Professor and Head of the dept, Dept of I.T. CBIT for his valuable suggestion and moral support.

Our respects and regards to **Dr. B. Chennakesava Rao,** Principal, Chaitanya Bharathi Institute of Technology, for his cooperation and encouragement.

Finally, we thank the staff members, faculty of Dept. of IT, CBIT, our friends, and all our family members who with their valuable suggestions and support, directly or indirectly helped us to accomplish this project.

N.NIHARIKA (160110737014)

R.SINDHUJA (160110737025)

ABSTRACT

MANET is a collection of wireless mobile devices that communicate with each other without the use of any wired network. MANET have some issues like flooding and broadcast storm problem. These problems arise because of its dynamic topology, broadcasting characteristic and mobility. The most commonly used routing algorithm in

MANETS is DSR algorithm. The Dynamic Source Routing (DSR) protocol is an on-demand routing protocol. Mobile nodes are required to maintain route caches that contain unexpired routes and are continually updated as new routes are learned. The major problem with this algorithm is the storm problem.

In this project, solution for “storm problem” in reactive routing protocol have been proposed by reducing rebroadcast messages. The above mentioned problem have been catered by modifying the flow of route request message and have included route selection field. A modified algorithm M-DSR have been proposed which helps in minimizing the control overhead in reactive routing protocol .This project will further limit the number of control messages in the network to increase the protocol performances.

This protocol is implemented over the MANET network and simulated using NS2. NS2 is a discrete event packet level simulator. It is implemented over cygwin emulator in windows environment. Ns provides substantial support for simulation, routing, and multicast protocols over wired and wireless networks. Performance is evaluated from the parameters such as end-to-end delay and routing overhead. The main objective of the project is reducing the overhead which is found in reactive routing protocol and improving the energy efficiency.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

MANET	Mobile Ad hoc Network
DSR	Dynamic source routing
TCL	Tool Command Language
M-DSR	Modified - Dynamic source routing
NS2	Network simulator 2
WAN	Wide Area Network
LAN	Local Area Network
TCP	Transmission Control Protocol
NAM	Network animator
WLAN	Wireless Local Area Networks
MAN	Metropolitan Area Networks
CAN	Campus Area Networks
SAN	Storage or System Area Network
PAN	Personal Area Network
DAN	Desk Area Network
InVANET	Intelligent vehicular ad hoc networks
VANETs	Vehicular ad hoc networks
WMN	wireless mesh network
WSN	wireless sensor network
AODV	Ad Hoc on-Demand Distance Vector Routing
DSDV	Destination-Sequenced Distance Vector
AP	Access point