IMPLEMENTING AND COMPARING DSR AND DSDV ROUTING PROTOCOLS FOR MOBILE AD HOC NETWORKING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Computer Science and Engineering

By

Bikash Rath

Roll Number 10506041

Department of Computer Science and Engineering

National Institute of Technology

Rourkela

2009

IMPLEMENTING AND COMPARING DSR AND DSDV ROUTING PROTOCOLS FOR MOBILE AD HOC NETWORKING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology
In
Computer Science and Engineering

By:

Bikash Rath

Roll Number 10506041

Under the guidance of:

Prof. S. Chinara

Department of Computer Science and Engineering
National Institute of Technology
Rourkela

2009

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled "IMPLEMENTING AND COMPARING DSR AND DSDV ROUTING PROTOCOLS FOR MOBILE AD HOC NETWORKING" submitted by Bikash Rath in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. S. Chinara
Dept .of Computer Science and Engineering
National Institute of Technology
Rourkela 769008

ACKNOWLEDGEMENT

I express my sincere gratitude to Prof. S . Chinara, Department of Computer Science and Engineering, National Institute of Technology, Rourkela, for her valuable guidance and timely suggestions during the entire duration of my project work, without which this work would not have been possible. I would also like to convey my deep regards to all other faculty members and staff of Department of Computer Science and Engineering, NIT Rourkela, who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish this project work. I would like to thank my senior Ramana sir (Master of Technology Computer science and engineering) for his valuable suggestions. Finally I would also like to thank my friends for their advice and pointing out my mistakes.

Bikash Rath

Roll Number: 10506041

CONTENTS

ABSTRACT

Ad hoc networking allows portable devices to establish communication independent of a central infrastructure. However, the fact that there is no central Infrastructure and that the devices can move randomly gives rise to various kind of problems, such as routing and security. In this thesis the problem of routing is considered. This thesis addresses issues pertaining to **Destination Sequenced Distance vector (DSDV)** and **Dynamic Source Routing (DSR)** protocols, which are used for efficient routing under different scenarios in Mobile Ad-hoc Network **(MANET)**, which plays a critical role in places where wired network are neither available nor economical to deploy. My objective was to implement the two routing protocols using Network Simulators and run it for different number of nodes. Then I compared the two routing protocols for different network parameters and studied the efficient protocol under a particular scenario on the basis of two metrics.

(1) Packet delivery ratio

(2) Routing load

DSDV is a **Proactive gateway discovery** algorithm where the gateway periodically broadcasts a gateway advertisement message which is transmitted after expiration of the gateways timer.

DSR is a **Reactive gateway discovery** algorithm where a mobile device of MANET connects by gateway only when it is needed.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Wireless cellular systems have been in use since 1980s. We have seen their evolutions to first, second and third generation's wireless systems. Wireless systems operate with the aid of a centralized supporting structure such as an access point. These access points assist the wireless users to keep connected with the wireless system, when they roam from one place to the other.

The presence of a fixed supporting structure limits the adaptability of wireless systems. In other words, the technology cannot work effectively in places where there is no fixed infrastructure. Future generation wireless systems will require easy and quick deployment of wireless networks. This quick network deployment is not possible with the existing structure of current wireless systems.

Recent advancements such as Bluetooth introduced a new type of wireless systems known as mobile ad-hoc networks. Mobile ad-hoc networks or "short live" networks operate in the absence of fixed infrastructure. They offer quick and easy network deployment in situations where it is not possible otherwise. Ad-hoc is a Latin word, which means "for this or for this only." Mobile ad-hoc network is an autonomous system of mobile nodes connected by wireless links; each node operates as an end system and a router for all other nodes in the network.

Nodes in mobile ad-hoc network are free to move and organize themselves in an arbitrary fashion. Each user is free to roam about while communication with others. The path between each pair of the users may have multiple links and the radio between them can be heterogeneous. This allows an association of various links to be a part of the same network. [3]

A mobile ad-hoc network is a collection of mobile nodes forming an ad-hoc network without the assistance of any centralized structures. These networks introduced a new art of network establishment and can be well suited for an environment where either the infrastructure is lost or where deploy an infrastructure is not very cost effective.

The popular IEEE 802.11 "WI-FI" protocol is capable of providing ad-hoc network facilities at low level, when no access point is available. However in this case, the nodes are limited to send and receive information but do not route anything across the network. Mobile ad-hoc networks can operate in a standalone fashion or could possibly be connected to a larger network such as the Internet. [3]

Mobile ad-hoc networks can turn the dream of getting connected "anywhere and at any time" into reality. Typical application examples include a disaster recovery or a military operation. Not bound to specific situations, these networks may equally show better performance in other places. As an example, we can imagine a group of peoples with laptops, in a business meeting at a place where no network services is present. They can easily network their machines by forming an ad-hoc network. This is one of the many examples where these networks may possibly be used.

1.2 PROJECT DESCRIPTION

The ad hoc routing protocols DSDV and DSR are two of the promising routing protocols. They can be used in mobile ad hoc networks to rout packets between mobile nodes. The main objectives of this thesis project are:

(1) Implementing the existing DSDV and DSR routing protocols in ns2

(2) Comparing the performance of two protocols under following metrics

- (i) Packet delivery ratio
- (ii) End-to-end delay

CHAPTER 2

MOBILE AD HOC NETWORKING

This chapter gives an overview of Mobile Ad Hoc Networking. Section 2.1 introduces the protocol stacks used in the Internet and MANET and compares them with the Open Systems Interconnection (OSI) model. Section 2.2 and 2.3 describes the proactive and reactive gateway discovery. Then section 2.4 and 2.5 describes the different routing concepts of DSDV and DSR.

2.1 THE PROTOCOL STACK

In this section the **protocol stack** for mobile ad hoc networks is described. This gives a comprehensive picture of, and helps to better understand, mobile ad hoc networks. Figure 2.1, shows the protocol stack which consists of five layers: physical layer, data link layer, network layer, transport layer and application layer. It has similarities to the TCP/IP protocol suite. As can be seen the OSI layers for session, presentation and application are merged into one section, the application layer.

On the left of Figure 2.1, the OSI model is shown. It is a layered framework for the design of network systems that allows for communication across all types of computer systems.

In the middle of the figure, the TCP/IP suite is illustrated. Because it was designed before the OSI model, the layers in the TCP/IP suite do not correspond exactly to the OSI layers. The lower four layers are the same but the fifth layer in the TCP/IP suite (the application layer) is equivalent to the combined session, presentation and application layers of the OSI model.

On the right, the MANET protocol stack -which is similar to the TCP/IP suite -is shown. The main difference between these two protocol stacks lies in the network layer. Mobile nodes (which are both hosts and routers) use an ad hoc routing protocol to route packets. In the physical and data link layer, mobile nodes run protocols that have been designed for wireless channels. Some options are the IEEE standard for wireless LANs, IEEE 802.11, the European ETSI standard for a high-speed wireless LAN, HIPERLAN 2, and finally an industry approach toward wireless personal area networks, i.e. wireless LANs at an even smaller range, Bluetooth. In the simulation tool used in this project, the standard IEEE 802.11 is used in these layers. [7]

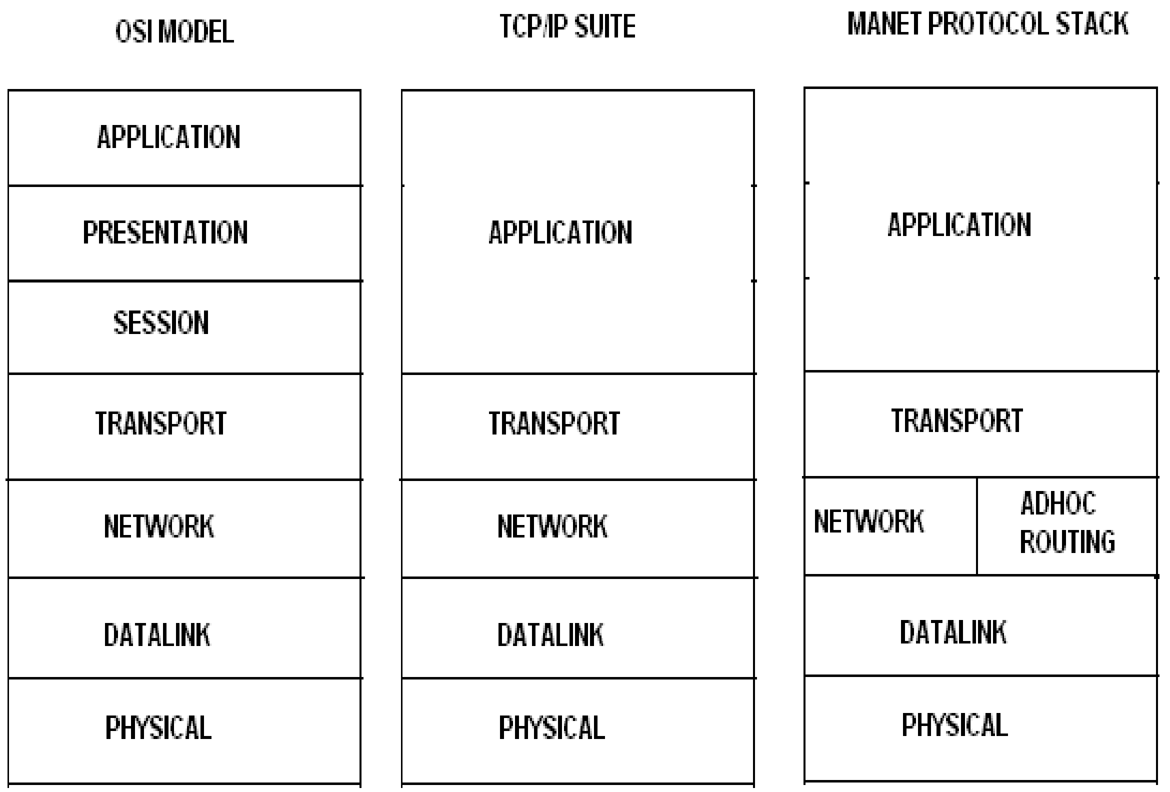


Figure 2.1: Three models

This thesis focuses on ad hoc routing which is handled by the network layer. The network layer is divided into two parts: Network and Ad Hoc Routing. The protocol used in the network part is Internet Protocol (IP) and the protocols which can be used in the ad hoc routing part are Destination Sequenced Distance Vector (DSDV), or Dynamic Source Routing (DSR), which are described in section 2.4.

2.2 PROACTIVE GATEWAY DISCOVERY

All the proactive approach algorithms are based on traditional distance vector and link state protocols developed for use in wireless approach. The primary characteristic of proactive approach is that each node in the maintenance of network is to maintain a route to every other node in the network all the times regardless of whether or not these routes are needed. In order to maintain correct route information, a node must periodically send control messages. Updates to route table are triggered or by certain events which caused in manipulation of other nodes (neighboring) route table. Link addition and removal can trigger an event triggered updation of routing table. In proactive approach the main advantage is that the rout to each node is instantly found because the table contains all the nodal address. Source only need to check the routing table and transfer a packet. The major disadvantage of proactive approach is that each node is prone to rapid movement. So the overhead of maintaining a rout table is very high, and amount of routing state maintained at each node scales as order of o[n] where n is the number of nodes in the network. It becomes inefficient for a large network.

GSR introduced below is a proactive routing protocol
Global State Routing (GSR) is based on the Link State (LS) routing method. In the LS Routing method, each node floods the link state information into the whole network (global flooding) once it realizes that links change between itself and its neighbors. The link state information includes the delay to each of its neighbors. A node will know the whole topology when it obtains all link information. LS routing works well in networks with static topologies. When links change quickly, however, frequent global flooding will inevitably lead to huge control overhead. [4]

Unlike the traditional LS method, GSR does not flood the link state packets. Instead, every node maintains the link state table based on up-to-date LS information received from neighboring nodes, and periodically exchanges its LS information with its neighbors only (no global flooding). Before sending an LS packet, a node assigns the LS packet a unique sequence number to identify the newest LS information. LS information is disseminated as the LS packets with larger sequence numbers replace the ones with smaller sequence numbers.

The convergence time required to detect a link change in GSR is shorter than in the Distributed Bellman-Ford (DBF) protocol. The convergence time in GSR is O (D*I) where D is the diameter of the network and I is the link state update interval. The convergence time is normally smaller than O(N*I) in DBF, where N is the number of nodes in the networks and I is the update interval. Since the global topology is maintained in every node, preventing routing loops is simple and easy. [3][4]

The drawbacks of GSR are the large size of the update messages, which consume a considerable amount of bandwidth, and the latency of the LS information propagation, which depends on the LS information update interval time. ``Fisheye" technology can be used to reduce the size of update messages. In this case, every node maintains highly accurate network information about the immediate neighboring nodes, with progressively fewer details about farther nodes.

2.3 REACTIVE GATEWAY DISCOVERY

Reactive routing technique is also known as **on-demand routing.** It takes a different approach of routing which overcomes the disadvantages of proactive routing. In reactive approaches those nodes which require connectivity to the Internet reactively find Internet gateways by means of broadcasting some kind of solicitation within the entire ad hoc network. This approach reduces the overhead of maintaining the route table as that of proactive. The node dynamically checks the route table, and if it does not find an entry for its destination or it finds an outdated entry it performs route discovery to find the path to its destination. [5]

The signaling overhead is reduced in this method, particularly in networks with low to moderate traffic loads. However it has a drawback of route acquisition latency. That is when corresponding entry is not found the route discovery mechanism occurs which takes a very large amount of time, and for that time the packet waits for updation of the table.

2.4 DSDV

This protocol is based on classical Bellman-Ford routing algorithm designed for MANETS. Each node maintains a list of all destinations and number of hops to each destination. Each entry is marked with a sequence number. It uses full dump or incremental update to reduce network traffic generated by rout updates. The broadcast of route updates is delayed by settling time. The only improvement made here is avoidance of routing loops in a mobile network of routers. With this improvement, routing information can always be readily available, regardless of whether the source node requires the information or not. DSDV solve the problem of routing loops and count to infinity by associating each route entry with a sequence number indicating its freshness. In DSDV, a sequence number is linked to a destination node, and usually is originated by that node (the owner). The only case that a non-owner node updates a sequence number of a route is when it detects a link break on that route. An owner node always uses even-numbers as sequence numbers, and a non-owner node always uses odd-numbers. With the addition of sequence numbers, routes for the same destination are selected based on the following rules: 1) a route with a newer sequence number is preferred; 2) in the case that two routes have a same sequence number, the one with a better cost metric is preferred. [4]

The list which is maintained is called routing table. The routing table contains the following:

(1) All available destinations' IP address
(2) Next hop IP address
(3) Number of hops to reach the destination
(4) Sequence number assigned by the destination node
(5) Install time

The sequence number is used to distinguish stale routes from new ones and thus avoid the formation of loops. The stations periodically transmit their routing tables to their immediate neighbors. A station also transmits its routing table if a significant change has occurred in its table from the last update sent. So, the update is both time-driven and event-driven.

As stated above one of “full dump" or an incremental update is used to send routing table updates for reducing network traffic. A full dump sends the full routing table to the neighbors and could span many packets whereas in an incremental update only those entries from the routing table are sent that has a metric change since the last update and it must fit in a packet. If there is space in the incremental update packet then those entries may be included whose sequence number has changed. When the network is relatively stable, incremental updates are sent to avoid extra traffic and full dump are relatively infrequent. In a fast-changing network, incremental packets can grow big so full dumps will be more frequent. [4]

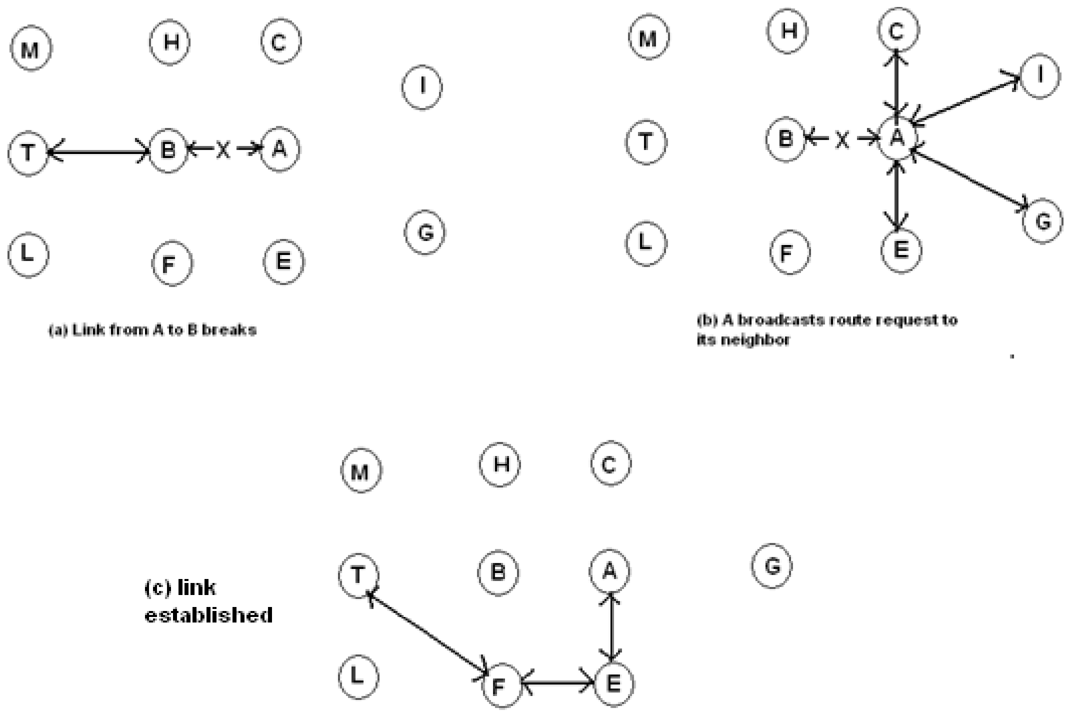
Each route update packet, in addition to the routing table information, also contains a unique sequence number assigned by the transmitter. The route labeled with the highest (i.e. most recent) sequence number is used. If two routes have the same sequence number then the route with the best metric (i.e. shortest route) is used. Based on the past history, the stations estimate the settling time of routes. The stations delay the transmission of a routing update by settling time so as to eliminate those updates that would occur if a better route were found very soon.

Each row of the update send is of the following form:

<Destination IP address, Destination sequence number, Hop count>

After receiving an update neighboring nodes utilizes it to compute the routing table entries.

To damp the routing fluctuations due to unsynchronized nature of periodic updates, routing updates for a given destination can propagate along different paths at different rates. To prevent a node from announcing a routing path change for a given destination while another better update for that destination is still in route, DSDV requires node to wait a settling time before announcing a new route with higher metric for a destination.



2.2 Resolving failed links in DSDV

2.5 DSR

The Dynamic Source Routing protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. Dynamic Source Routing, DSR, is a reactive routing protocol that uses source routing to send packets. It uses source routing which means that the source must know the complete hop sequence to the destination.

Each node maintains a route cache, where all routes it knows are stored. The route discovery process is initiated only if the desired route cannot be found in the route cache.

To limit the number of route requests propagated, a node processes the route request message only if it has not already received the message and its address is not present in the route record of the message.

As mentioned before, DSR uses source routing, i.e. the source determines the complete sequence of hops that each packet should traverse. This requires that the sequence of hops is included in each packet's header. A negative consequence of this is the routing overhead every packet has to carry. However, one big advantage is that intermediate nodes can learn routes from the source routes in the packets they receive. Since finding a route is generally a costly operation in terms of time, bandwidth and energy, this is a strong argument for using source routing. Another advantage of source routing is that it avoids the need for up-to-date routing information in the intermediate nodes through which the packets are forwarded since all necessary routing information is included in the packets. Finally, it avoids routing loops easily because the complete route is determined by a single node instead of making the decision hop-by-hop. [5][6]

The protocol is composed of the two main mechanisms of "Route Discovery" and "Route Maintenance", which work together to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. All aspects of the protocol operate entirely on demand, allowing the routing packet overhead of DSR to scale automatically to only what is needed to react to changes in the routes currently in use. The protocol allows multiple routes to any destination and allows each sender to select and control the routes used in routing its packets, for example, for use in load balancing or for increased robustness.

Route Discovery

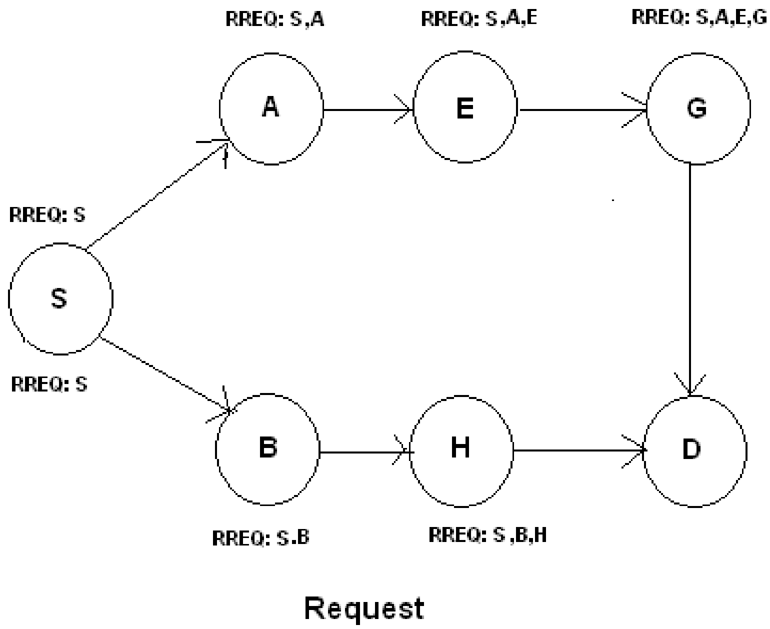
Route Discovery is used whenever a source node desires a route to a destination node. First, the source node looks up its route cache to determine if it already contains a route to the destination. If the source finds a valid route to the destination, it uses this route to send its data packets. If the node does not have a valid route to the destination, it initiates the route discovery process by broadcasting a route request message. The route request message contains the address of the source and the destination, and a unique identification number.

An intermediate node that receives a route request message searches its route cache for a route to the destination. If no route is found, it appends its address to the route record of the message and forwards the message to its neighbors. The message propagates through the network until it reaches either the destination or an intermediate node with a route to the destination. Then a route reply message, containing the proper hop sequence for reaching the destination, is generated and unicast back to the source node. [6]

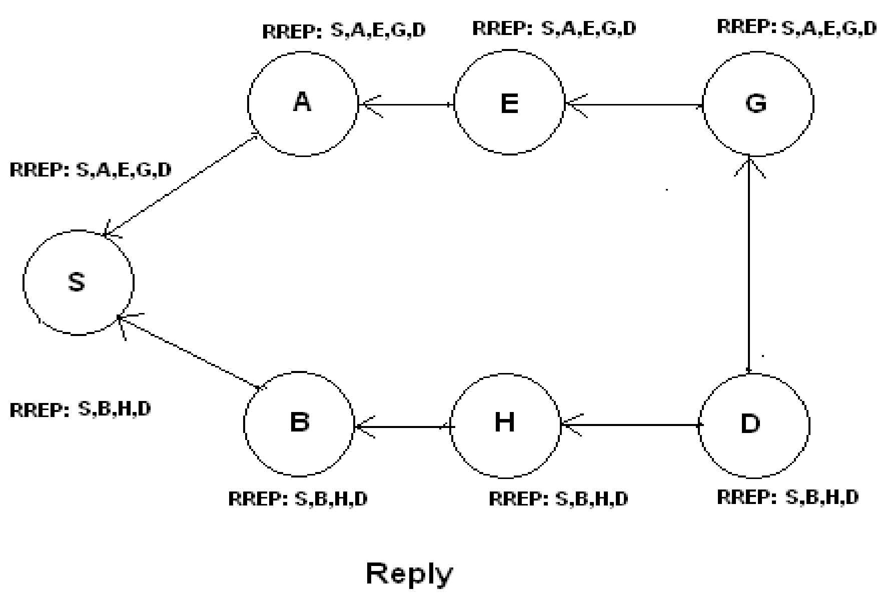
Route maintenance

Route Maintenance is used to handle route breaks. When a node encounters a fatal transmission problem at its data link layer, it removes the route from its route cache and generates a route error message. The route error message is sent to each node that has sent a packet routed over the broken link. When a node receives a route error message, it removes the hop in error from its route cache.

Acknowledgment messages are used to verify the correct operation of the route links. In wireless networks acknowledgments are often provided as e.g. an existing standard part of the MAC protocol in use, such as the link-layer acknowledgment frame defined by IEEE 802.11. If a built-in acknowledgment mechanism is not available, the node transmitting the message can explicitly request a DSR-specific software acknowledgment to be returned by the next node along the route. [6]



2.3 DSR request and Reply



CHAPTER 3

NETWORK SIMULATOR

DSDV and DSR routing protocols can be implemented using Network Simulator 2.31. NS is a discrete event simulator targeted at networking research. It provides substantial support for TCP routing and multicast protocols over wired and wireless networks. Using Xgraph (A plotting program) we can create graphical representation of simulation results. All the work is done under linux platform, preferably ubuntu.

3.1 ABOUT NS 2

ns is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. ns uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols require a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important.

On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important.

ns meets both of these needs with two languages, C++ and OTcl .C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration.

In NS-2, the frontend of the program is written in TCL(Tool Command Language). The backend of NS-2 simulator is written in C++ and when the tcl program is compiled, a tracefile and namfile are created which define the movement pattern of the nodes and keeps track of the number of packets sent, number of hops between 2 nodes, connection type etc at each instance of time. In addition to these, a scenario file defining the destination of mobile nodes along with their speeds and a connection pattern file(CBR file) defining the connection pattern, topology and packet type are also used to create the trace files and nam files which are then used by the simulator to simulate the network. [1][2]

Also the network parameters can be explicitly mentioned during the creation of the scenario and connection-pattern files using the library functions of the simulator.

3.2 BASIC INSTALLATION STEPS

(1) NS-2.31 is extracted in the home folder or folder in which it is to be installed

(2) Then TCL and TK packages are installed using the following command.

(i) Sudo apt-get install tcl 8.4-dev tk 8.4-dev

(ii) sudo apt-get install build-essential autoconf automake libxmu-dev

(3) Now NS-2.31 is ready to be installed

(4) "bashrc" file is edited by adding the following variables and their attributes.

```
# LD_LIBRARY_PATH
OTCL_LIB=/your/path/ns-allinone-2.31/otcl-1.13
NS2_LIB=/your/path/ns-allinone-2.31/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$
USR_LOCAL_LIB


# TCL_LIBRARY
TCL_LIB=/your/path/ns-allinone-2.31/tcl8.4.14/library
USR_LIB=/usr/lib
```

export TCL_LIBRARY=$TCL_LIB:$USR_LIB

PATH

XGRAPH=/your/path/ns-allinone-2.31/bin:/your/path/ns-allinone-2.31/tcl8.4.14/unix:/your/path/ns-allinone-2.31/tk8.4.14/unix

NS=/your/path/ns-allinone-2.31/ns-2.31/

NAM=/your/path/ns-allinone-2.31/nam-1.13/

PATH=$PATH:$XGRAPH:$NS:$NAM

(5) It is immediately made effective by the command " $ source ~/.bashrc"

(6) To confirm NS is installed ns is typed on konsole which gives a "%" symbol

(7) It is validated by the command

$ cd ns-2.31

$ ./validate

[2]

3.3 DEFINING GLOBAL VARIABLES

set ns_ [new Simulator] #creates a new simulator instance

set topo [new Topography] #creates a new topology

$topo load_flatgrid 670 670 #defines it in 670X670 area

Here set command is used to create a global variable. The first argument is the variable name (ns_, topo, etc.). the second argument is used to get the value of the variable. It may be a constant or a function whose return value is assigned to the variable. To access a variable we use $var_name, where var_name is the name of the variable.[2]

3.4 DEFINING STANDARD NS/NAM TRACE

To run the output of the program in an animator we need a nam file, and to analyze the output we need trace file. So the program must output certain files called nam file and trace file. We can do so by the following commands:

```
Set tracefd [open demo.tr  w]

$ns_  trace-all $tracefd

Set namtrace [open demo.nam w]

$ns_  namtrace-all-wireless $namtrace 670 670
```

The above commands opens two files called demo.tr and demo.nam and initialize them. [2]

3.5 MOBILE NODE CONFIGURATION

We can configure a mobile node by following codes.

```
$ns_node-config -adhocRouting DSDV\

                 -llType LL \

             -macType Mac/802_11\

             -ifqLen 50 \

             -ifqType Queue/DropTail/PriQueue \

             -antType Antenna/OmniAntenna \

             -propType Propagation/TwoRayGround \

             -phyType Phy/WirelessPhy \
```

-channelType Channel/WirelessChannel \

-topoInstance $topo

-agentTrace ON \

-routerTrace OFF \

-macTrace OFF

[1][2]

3.6 TRAFFIC AND MOVEMENT

We can also define the traffic and movement pattern in separate files called CBR file and **scenario file** respectively**. Cbr file** can be created by using a tcl program called cbrgen.tcl which is present in the directory ”ns-2/indep-utils/cmu-scen-gen/”. To define the movement we use an exe file called setdest present in the folder “ns-2/indep-utils/cmu-scen-gen/setdest/”.

The scenario and cbr files are generated by using the following commands in the appropriate directory respectively.

./setdest -n <num_of_nodes> -p pausetime -s <maxspeed> -t <simtime> -x <maxx> -y <maxy>

ns cbrgen.tcl [-type cbf|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate rate]

[1][2]

CHAPTER 4

SIMULATION

To be able to implement the Destination Sequenced Distance Vector and Dynamic Source Routing protocols certain simulation scenario must be run. This chapter describes the details of the simulation which has been done and the results of the simulations done for the protocols. The simulations were conducted under UBUNTU(linux) platform.

CBR file

Manually giving traffic connections for a large number of nodes would be cumbersome. So random traffic connections of TCP and CBR can be setup between mobile nodes using a traffic scenario generator script. The generator script is available under "/indep-utils/cmu-scen-gen" directory, and the file name is cbrgen.tcl. Using this script we can generate random traffic connections between any number of nodes. We need to define the following to generate random traffic connections: [1][2]

(1) The type of traffic connection (CBR or TCP)

(2) The number of nodes for which simulation is being done

(3) A random seed value

(4) Maximum number of connections

(5) Rate, whose inverse is used to compute the interval time between CBR packets

Scenario file

As cbr file is used to store the traffic connections, similarly scenario file is used to store the initial position of the nodes and movement of nodes at different times and their speed, etc. Since it will be difficult to manually give initial position, movement of the nodes and their speed for each movement at different times we use a random file generator here also. The node movement generator is available under /indep-utils/cmu-scen-gen/setdest/ directory. It is available under the name "setdest", which is an exe file. This file is run with certain arguments to create the scenario file. The arguments are:

(1)	Number of nodes
(2)	Pause time
(3)	Maximum speed
(4)	Simulation time
(5)	X-axis dimension
(6)	Y-axis dimension

The cbr and scenario files are loaded in the tcl program instead of creating traffic and movement of the nodes manually and the program is executed. [1][2]

NAM file and TRACE file

After simulating the program using cbr and scenario files we can get the output in form of two files. One is called as the network animator file (NAM) and the other is called the trace file. These two files are created in the due course of running the program. Basically the two files stores the same things but in different format. NAM file stores the output in such a way that it can be used by the animator to show an animated result, and the trace file stores the output so that it can be analyzed. [2]

4.1 SIMULATION OF DSDV AND DSR

My aim here was to implement DSDV and DSR routing protocol for 10 nodes sending cbr packets with random speed. First the cbr files and scenario files are generated and then using dsdv protocol simulation is done which gives the nam file and trace file. Then another nam and Trace files are created dsr protocol. [2]

The following figures are the execution of the nam files instances created. For each execution of the same program different nam files are created and we can view the output on the network simulator.

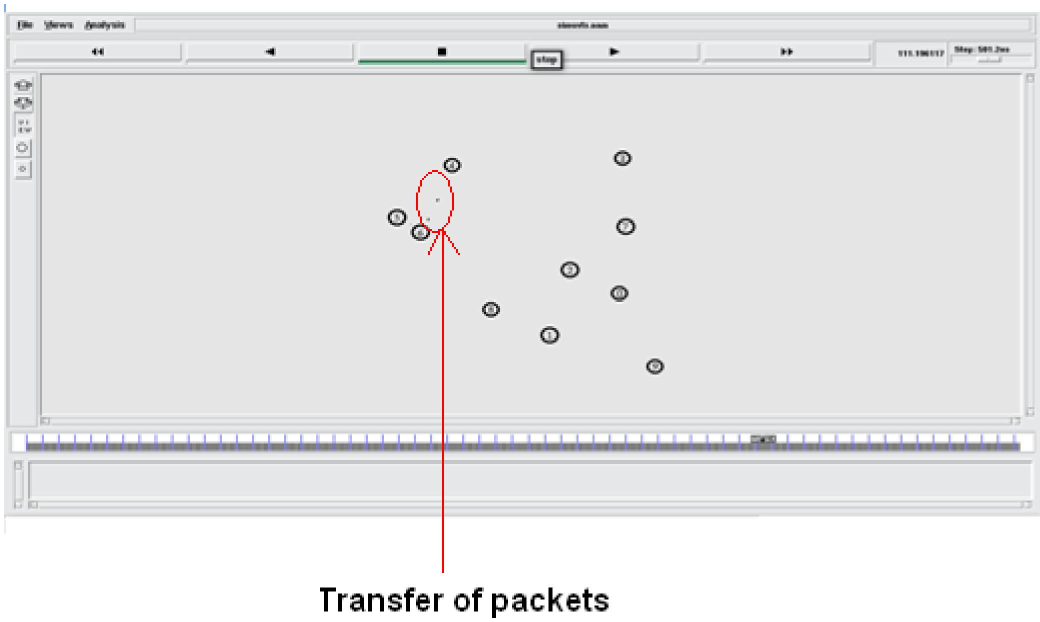
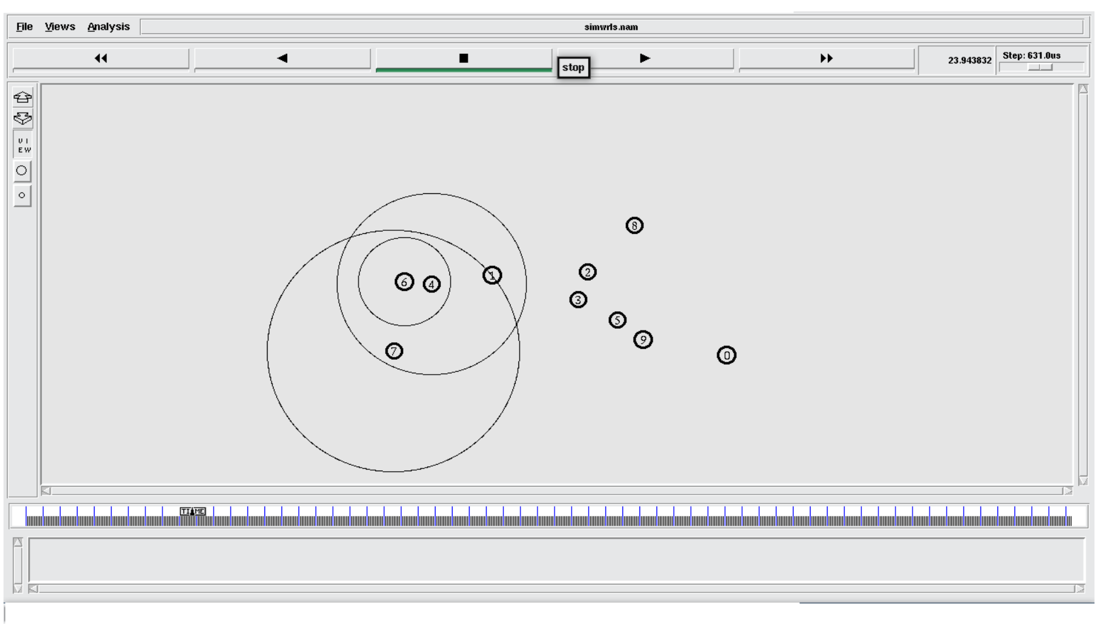


Fig 4.1 transfer of packet



`

Fig 4.2 route discovery

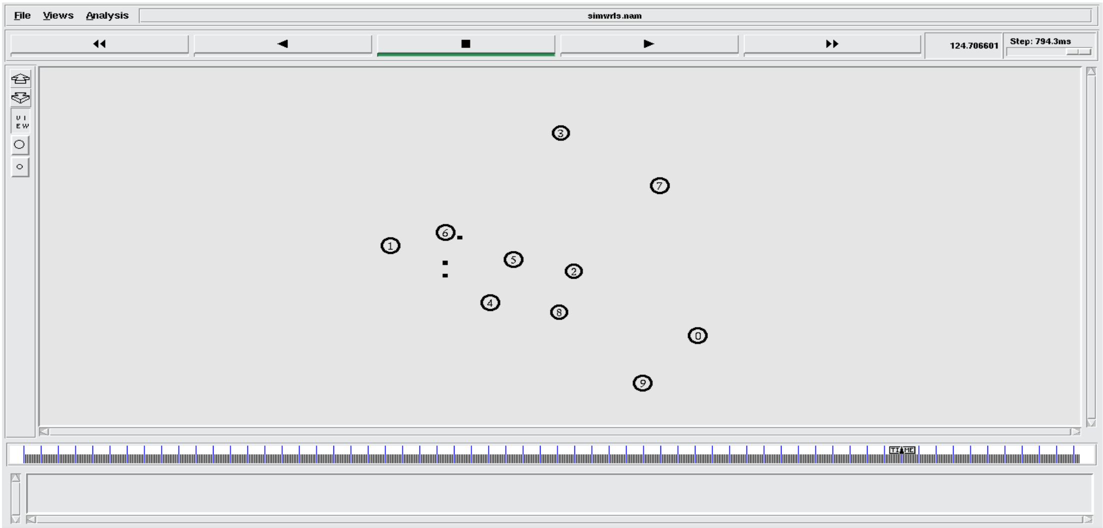


Fig 4.3 dropping of packets

Movement Model

The mobile nodes move according to the \random waypoint" model. Each mobile node begins the simulation by remaining stationary for pause time seconds. It then selects a random destination in the defined topology area and moves to that destination at a random speed. The random speed is distributed uniformly between zero (zero not included) and some maximum speed. Upon reaching the destination, the mobile node pauses again for pause time seconds, selects another destination, and proceeds there as previously described. This movement pattern is repeated for the duration of the simulation. [1]

The movement patterns are generated by CPU's movement generator (setdest). The chosen values for pause time and maximum speed are shown in Table 5.1.

Communication Model

In the scenario used in this study, five mobile nodes communicate with one of two fixed nodes (hosts) located on the Internet through a gateway. As the goal of the simulations was to compare the different approaches for gateway discovery, the traffic source was chosen to be a constant bit rate (CBR) source. Each source mobile node generates packets every 0.2 seconds in this study. In other words, each source generates 5 packets per second. Since each packet contain 512 bytes of data, the amount of generated data is 5*512*8 bit/s = 20 kbit/s, for each source. The traffic connection pattern is generated by CMUs traffic generator (cbr-gen.tcl). The main parameters in cbrgen.tcl are \connections" (number of sources) and \rate" (packet rate); see Table 4.1 [1]

.

Parameter	Value
Transmission range	250 m
Simulation time	110 s
Topology size	800m x 500m
Number of mobile nodes	14
number of sources	4
Number of gateways	2
Traffic type	constant bit rate
Packet rate	5 packets/s
Packet size	512 bytes
Maximum speed	10 m/s

Table 4.1: General parameters used in all simulations.

4.2 PERFORMANCE METRICS

The second goal of this project is to compare the performance of the two protocols under different scenario. Comparing the different methods is done by simulating them and examining their behavior.

In comparing the two protocols, the evaluation could be done in the following three metrics:

(1) **The packet delivery ratio** defined as the number of received data packets divided by the number of generated data packets

(2) **The end to end delay** is defined as the time a data packet is received by the destination minus the time the data packet is generated by the source.

[7]

CHAPTER 5

RESULTS

5.1 PACKET DELIVERY RATIO

Figure 4.4 shows packet delivery ratio with pause time varying from 2 to 10 for DSDV and DSR routing protocol. The red line shows graph for DSDV and the green line shows the graph for DSR protocol. The delivery ratio for both the protocols is always greater than 90 percent. The basic difference between the two protocols is very less. But generally the graph for the DSR protocol lies above than that of DSDV for most cases. However in certain cases the DSDV protocols is also better.

It is more likely for the mobile nodes to have fresher and shorter routes to a gateway and thereby minimizing the risk for link breaks. Link breaks can result in lost data packets since the source continues to send data packets until it receives a RERR message from the mobile node that has a broken link. The longer the route is (in number of hops), the longer time it can take before the source receive a RERR and hence, more data packets can be lost.

When the pause time interval increases, a mobile node receives less gateway information and consequently it does not update the route to the gateway as often as for short advertisement intervals. Therefore, the positive effect of periodic gateway information is decreased as the advertisement interval increases.

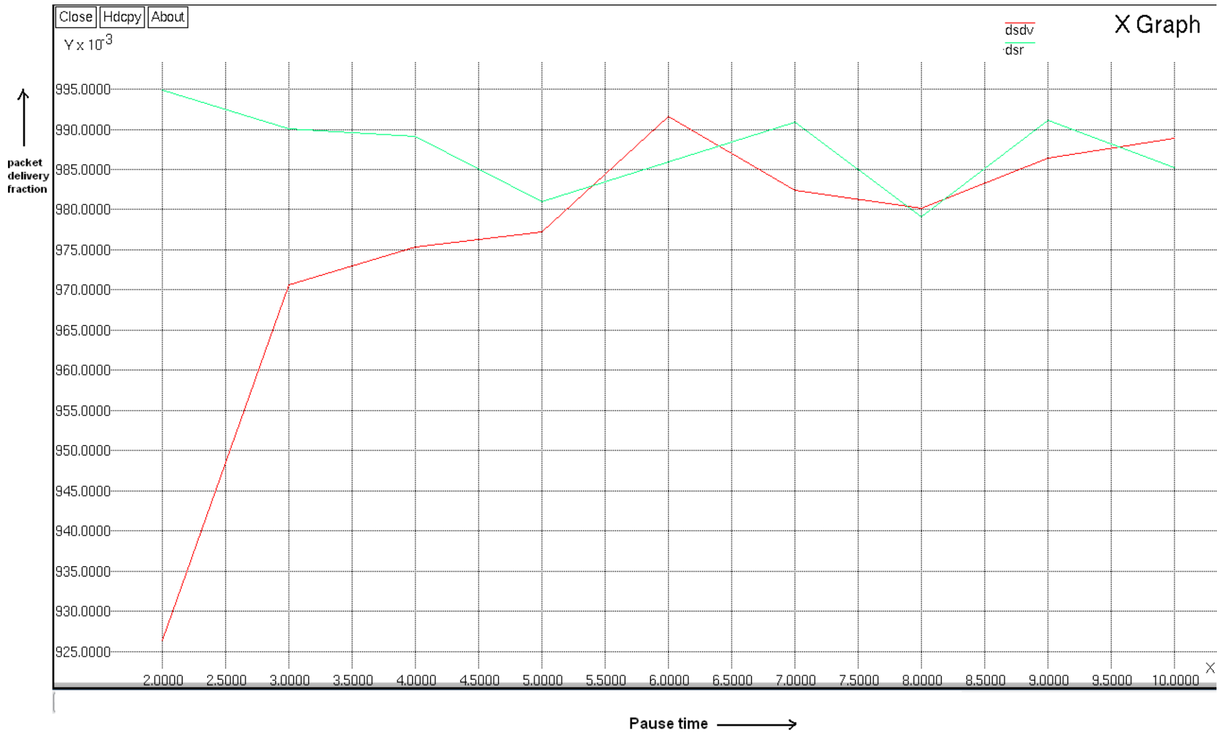


Fig 5.1 Packet delivery ratio verses pause time

5.2 AVERAGE END TO END DELAY

The average end-to-end delay is less for the DSDV approach than for the DSR approach. The reason is that the periodic gateway information sent by the gateways allows the mobile nodes to update their route entries for the gateways more often, resulting in fresher and shorter routes. With the DSR (reactive approach) a mobile node continues to use a route to a gateway until it is broken. In some cases this route can be pretty long (in number of hops) and even if the mobile node is much closer to another gateway it does not use this gateway, but continues to send the data packets along the long route to the gateway further away until the route is broken. Therefore, the end-to-end delay increases for these data packets, resulting in increased average end-to-end delay for all data packets.

The average end-to-end delay is decreased slightly for short pause time intervals when the advertisement interval is increased. At the first thought this might seem unexpected. However, it can be explained by the fact that very short advertisement intervals result in a lot of control traffic which lead to higher processing times for data packets at each node

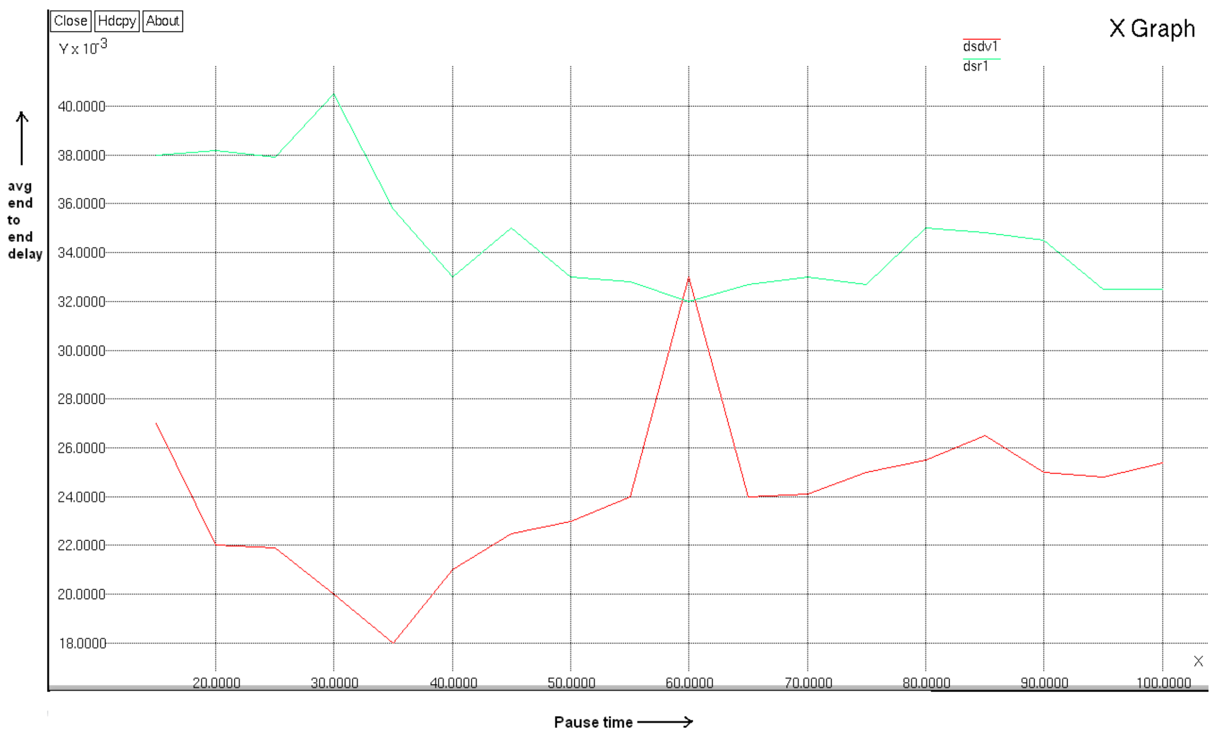


Fig 5.2 end-to-end delay verses pause time

CHAPTER 6

CONCLUSION

CONCLUSION

In this paper I have implemented the Destination Sequenced Distance Vector and Dynamic Source Routing protocols in Tool command language and integrated the module in the ns-2 Simulator. The performance of the protocols were measured with respect to metrics like Packet delivery ratio, end – end delay etc. I have made the performance comparison of the protocols. Simulations were carried out with identical topologies and running different protocols on the mobile node.

The results of the simulation indicate that performance of the DSR protocol is superior to standard DSDV. It is also observed that the performance is better especially when the pause time is low. For higher pause time although DSR is better for most cases but their delivery ratio remains close to each other.

It is also true that any of the single protocol does not supersede the other one. There performance depends upon the different scenarios.

REFERENCES

(1) www.isi.edu/nsnam/ns/tutorial Marc Greis tutorial on ns2

(2) Matthias Transier " Ns2 tutorial running simulations "

(3) D. Kim, J. Garcia and K. Obraczka, "Routing Mechanisms for Mobile Ad Hoc Networks based on the Energy Drain Rate", IEEE Transactions on Mobile Computing. Vol 2, no 2, 2003, pp.161-173

(4) C.E. Perkins & P. Bhagwat, "Highly Dynamic Destination Sequence-Vector Routing (DSDV) for Mobile Computers", Computer Communication Review, vol. 24, no.4, 1994, pp. 234-244.

(5) C.E. Perkins and E.M. Royer, "Ad-Hoc on-Demand Distance Vector Routing," Proc. Workshop Mobile Computing Systems and Applications (WMCSA '99), Feb. 1999 pp. 90-100

(6) David B. Johnson and David A. Maltz. "Dynamic source routing in ad hoc wireless networks", Mobile Computing, Kluwer Academic Publishers. 1996 pp.153–181, 1996.

(7) M. S. Corson, J. P. Maker and G. H. Cirincione , "Internet-Based Mobile Ad Hoc Networking," IEEE Internet Computing, Vol. 3, no. 4, July-August 1999, pp. 63-70.