SIMULATION OF A MULTIPATH ROUTING PROTOCOL FOR ADHOC NETWORKS

A PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

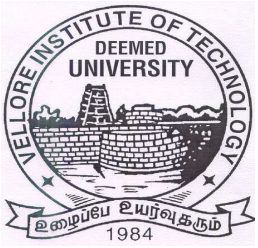
Bachelor of Engineering
in
Computer Science and Engineering

to the
University of Madras

by

KRISHNAVENI.S **8903885**
VIDHYASAGARI.S **8903909**

Under the Guidance of
Mrs.S.SUMATHY M.E.,



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VELLORE INSTITUTE OF TECHNOLOGY
VELLORE - 632014
TAMIL NADU, INDIA

MARCH 2003

VELLORE INSTITUTE OF TECHNOLOGY
VELLORE - 632014

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project report titled **Simulation of a Multipath routing protocol for Adhoc networks** is an approved record of work done by Ms. **VIDHYA SAGARI.S** Register number **8903885** in the partial fulfillment for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering branch under the University of Madras**, during the year 2002-2003.**

Guide **Head of the Department**

(Mrs.S.Sumathy) **(Dr.K.Ganesan)**

Date

Submitted for the university examination held on_____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

First of all, we are indebted to the almighty for having showered upon us His blessings throughout. The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without thanking the people who have all made this possible. We consider it our privilege to express our sincere gratitude and respect to all those who guided and inspired us in the completion of the project.

We are very thankful to Mr.G.Viswanathan, chancellor of VIT for giving us this wonderful opportunity to study in his college and utilize all the facilities to the fullest. We express our thanks to the prochancellor Mr.P.Radhakrishnan,the vice chancellor Mr.P.Selvam and the principal Mr.Anand.A.Samuel. Our sincere thanks to Dr.K.Ganesan, HOD, Computer Science and engineering department, VIT.

We express our deep sense of gratitude to our guide Mrs.S.Sumathy, lecturer, Computer Science and engineering department, VIT for constantly motivating, guiding and encouraging us throughout the project tenure. We would have been nowhere without her valuable advice in times of need.

Our heartfelt sincere thanks to Dr.S.Saravanan, senior lecturer, Department of information technology, VIT for having given us the crux idea of the entire project and guided us from the beginning.

Most importantly, we are grateful to our beloved parents and friends who have been a mental support throughout this project.

ABSTRACT

Mobile Communication is the present trend in the world of networking. Adhoc Networks is a specialised area of research in this field. This network does not rely on a fixed infrastructure and works in a shared wireless media .The topology of the adhoc network keeps on changing rapidly and so any routing decision has to be taken “on the fly”.

The protocols used in the Adhoc Networks address certain specialised issues like node mobility, potentially very large number of nodes and limited communication resources (bandwidth and energy).

Adhoc routing protocols can generally be divided into 3 types. They are flat routing, hierarchical routing and a geographical position based routing. Flat routing protocols can be further classified as pro-active and reactive types. In pro-active protocols details of all other nodes are maintained in every other node in its routing table which is modified frequently and this takes a considerable amount of time to converge. In reactive protocols the path is obtained only when the source needs to transfer the data to any other node like in protocols like. AODV and DSR.

In DSR ,eventhough multiple paths are discovered, only a single path is used for the whole data transmission until a link break occurs on that line. In the multipath routing protocol, all the multiple paths are used for the entire process of data transmission. The benefit is the effective utilization of the limited resource – Bandwidth. GloMoSim is the network simulator used to simulate this .It is a scalable simulation environment for wireless network system designed using parallel discrete event simulation capability provided by Parsec.

CONTENTS

INTRODUCTION

Efficient communication through networking has become the order of the day. Networking, till a few decades back was confined to the wires that imposed a limitation on the positioning of users. With the advancement of technology in strides, networking has evolved into the wireless mode, wherein it is not required for the users to be stationary.

In the next generation of wireless communication systems, there will be a need for the rapid deployment of independent mobile users. Significant examples include establishing survivable, efficient, dynamic communication for emergency/rescue operations, disaster relief efforts, and military networks. Such network scenarios cannot rely on centralized and organized connectivity, and can be conceived as applications of Mobile Ad Hoc Networks. A MANET is an autonomous collection of mobile users that communicate over relatively bandwidth constrained wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. Here, routing functionality will be incorporated into mobile nodes.

The design of network protocols for these networks is a complex issue. The network should be able to adaptively alter the routing paths.This has to be designed taking into account the scarcity of the resources. (Bandwidth, energy of nodes). A protocol that concentrates on these criteria has been designed.

1.1 Problem Definition

To effectively balance the load of a mobile adhoc network by transmission of data in multiple node disjoint paths.This balancing is done to utilize the resources of the entire network properly.

1.2 Aim

To simulate a routing protocol for adhoc networks that incorporates a multipath strategy in the existing library of GloMoSim (Global mobile information system simulation library) .

1.3 Objective

To effectively simulate the multipath routing protocol that balances the network load over multiple node disjoint paths .

1.4 Application

GloMoSim is a library based sequential and parallel simulator for wireless networks. The library of GloMoSim has been built based on the parallel discrete-event simulation capability provided by Parsec.This project aims to contribute a multipath routing protocol to the already existing library. We term it as the Multipath DSR.The actual application would be in areas of research, which intend to concentrate on networks with energy and bandwidth constraints. This protocol is applicable in Adhoc networks which is a sort of a network built “ on the fly “ according to

the geographical positioning of the nodes. This find wide application in areas like Defence (Military application), Emergency search and And rescue operations and data acquisition systems.

1.5 Software configuration

1.5.1 GloMoSim

It is a scalable simulation environment for wireless network systems. It is designed using the parallel discrete even simulation capability provided by Parsec. It is designed as a set of library modules, each of which simulates a specific wireless simulation protocol in the protocol stack.

1.5.2 Parsec

PARSEC is a C-based discrete-event simulation language. It adopts the process interaction approach to discrete-event simulation. An object (also referred to as a physical process) or set of objects in the physical system is represented by a logical process. Interactions among physical processes (events) are modeled by time stamped message exchanges among the corresponding logical processes.

One of the important distinguishing features of PARSEC is its ability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures. PARSEC is designed to clearly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute it. Thus, with few modifications, a PARSEC program may be executed using the traditional sequential (Global Event

List) simulation protocol or one of many parallel optimistic or conservative protocols.

In addition, PARSEC provides powerful message receiving constructs that result in shorter and more natural simulation programs. Useful debugging facilities are available.

Note: - GloMoSim is executable only in the following operating systems Win2000, WinNT, winxp and the parsec for the appropriate OS has to be chosen.

2. DYNAMIC SOURCE ROUTING PROTOCOL

2.1 Introduction

The DSR is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. Using DSR, the network is completely self-organized and self-configured, requiring no existing network infrastructure or administration. Network nodes cooperate to forward packets for each other to allow communication over multiple “hops” between nodes not directly within wireless transmission range of one another. As nodes in the network move about or join or leave the network, and as wireless transmission conditions such as sources of interference change, all routing is automatically determined and maintained by the DSR routing protocol.

Since the number or sequence of intermediate hops needed to reach any destination may change at any time, the resulting network topology may be quite rich and rapidly changing.

The DSR protocol allows nodes to dynamically discover a source route across multiple network hops to any destination in the ad hoc network. Each data packet sent then carries in its header the complete, ordered list of nodes through which the packet will pass, allowing packet routing to be trivially loop-free and avoiding the need for up-to-date routing

information in the intermediate nodes through which the packet is forwarded.

By including this source route in the header of each data packet, other nodes forwarding or overhearing any of these packets can also easily cache this routing information for future use.

The DSR protocol provides highly reactive service in order to help ensure successful delivery of data packets in spite of node movement or other changes in network conditions.

The DSR protocol is composed of two main mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network:

2 Route Discovery is the mechanism by which a node S wishing to send a packet to a destination node D obtains a source route to D. Route Discovery is used only when S attempts to send a packet to D and does not already know a route to D.

3 Route Maintenance is the mechanism by which node S is able to detect, while using a source route to D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works. When Route Maintenance indicates a source route is broken, S can attempt to use any other route it happens to know to D, or can invoke Route Discovery again to find a new route for subsequent packets to D.

In DSR, Route Discovery and Route Maintenance each operate entirely “on demand”. In particular, unlike other protocols, DSR requires no periodic packets of any kind at any layer within the network.

2.1.1 Basic Route Discovery

When some source node originates a new packet addressed to some destination node, the source node places source route in the header which gives the sequence of hops that the packet is to follow on its way to the destination. Normally, the sender will obtain a suitable source route by searching its “Route Cache” of routes previously learned; if no route is found in its cache, it will initiate the Route Discovery protocol to dynamically find a new route to this destination node.

To initiate the Route Discovery, node A (the initiator) transmits a “Route Request” as a single local broadcast packet, which is received by (approximately) all nodes currently within wireless transmission range of A. Each Route Request identifies the initiator and target of the Route Discovery, and also contains a unique request identification 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 has been forwarded.

When another node receives this Route Request, if it is the target of the Route Discovery, it returns a “Route Reply” to the initiator of the Route Discovery, giving a copy of the accumulated route record from the Route Request; when the initiator receives this Route Reply, it caches this route in its Route Cache for use in sending subsequent packets to this destination. Otherwise, if this node receiving the Route Request has recently seen another Route Request message from this initiator bearing

this same request identification and target address, or if this node's own address is already listed in the route record in the Route Request, this node discards the Request. Otherwise, this node appends its own address to the route record in the Route Request and propagates it by transmitting it as a local broadcast packet (with the same request identification).

In returning the Route Reply to the initiator of the Route Discovery, the intermediate node will typically examine its own Route Cache for a route back to the initiator and if found, will use it for the source route for delivery of the packet containing the Route Reply. Otherwise, it should perform its own Route Discovery for target node A, but to avoid possible infinite recursion of Route Discoveries, it must piggyback this Route Reply on the packet containing its own Route Request for A. The intermediate node could instead simply reverse the sequence of hops in the route record that it is trying to send in the Route Reply, and use this as the source route on the packet carrying the Route Reply itself.

When initiating a Route Discovery, the sending node saves a copy of the original packet (that triggered the Discovery) in a local buffer called the"Send Buffer". The Send Buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet's destination. Each packet in the Send Buffer is logically associated with the time that it was placed into the Send Buffer and is discarded after residing in the Send Buffer for some timeout period. If necessary for preventing the Send Buffer from overflowing, a FIFO or other replacement strategy may also be used to evict packets even before they expire. While a packet remains in the Send Buffer, the node should occasionally initiate a new Route Discovery for the packet's destination address. However, the node must limit the rate at which such new Route Discoveries for the same address are initiated, since it is possible that the destination node is not currently reachable.

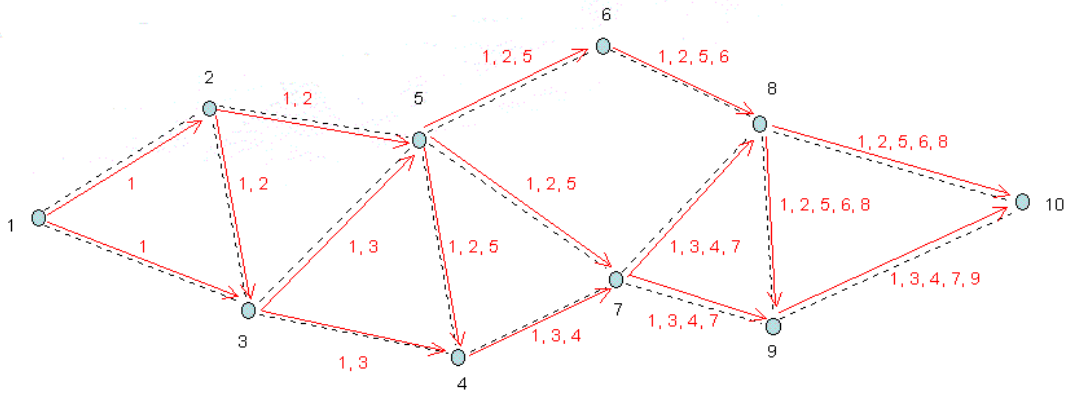
Illustration of route discovery :-

Source :1

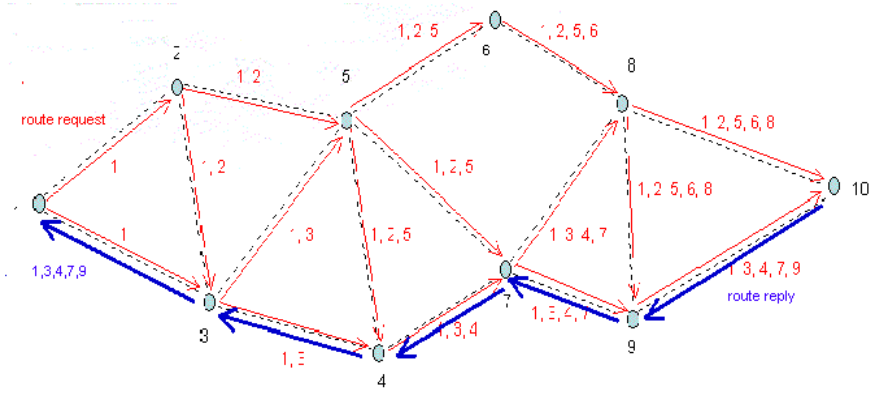
Destination :10

Route request path: Refer figure 2.1.1.b

Route reply path: Refer figure 2.1.1.c



ROUTE REQUEST


route request
1
1,3,4,7,9
1 2
1, 2, 5
1, 2, 5, 6
1, 2, 5, 6, 8
1, 2, 5
1, 2, 5, 6, 8
1 3 4, 7
1, 3, 4, 7, 9
route reply
1, 3, 4
10
9
8
6
5
4
3

ROUTE REPLY

2.1.2 Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that data can flow over the link from that node to the next hop.

An acknowledgment can provide confirmation that a link is capable of carrying data, and in wireless networks, acknowledgments are often provided at no cost, either as an existing standard part of the MAC protocol in use or as a software acknowledgement. When a software acknowledgment is used, the acknowledgment request should be retransmitted up to a maximum number of times. A retransmission of the acknowledgment request can be sent as a separate packet, piggybacked on a retransmission of the original data packet, or piggybacked on any packet with the same next-hop destination that does not also contain a software acknowledgment.

After the acknowledgment request has been retransmitted the maximum number of times, if no acknowledgment has been received, then the sender treats the link to this next-hop destination as currently “broken”. It should remove this link from its Route Cache and should return a “Route Error” to each node that has sent a packet routed over that link since an acknowledgment was last received.

2.2Additional Route Discovery Features :

2.2.1 Caching Overheard Routing Information

A node forwarding or otherwise overhearing any packet SHOULD add all usable routing information from that packet to its own Route Cache.

2.2.2 Replying to Route Requests using Cached Routes

A node receiving a Route Request, for which it is not the target, searches its own Route Cache for a route to the target of the Request. If found, the node generally returns a Route Reply to the initiator itself rather than forwarding the Route Request.

2.2.3. Preventing Route Reply storms

In order to reduce multiple route replies to be simultaneously transmitted back to the source, if a node can put its network interface into promiscuous receive mode, it MAY delay sending its own Route Reply for a short period, while listening to see if the initiating node begins using a shorter route first. Specifically,
This node MAY delay sending its own Route Reply for a random period

$d = H * (h - 1 + r)$

Where

h - the length in number of network hops for the route to be returned in this node's Route Reply.
r - a random floating point number between 0 and 1 .
H- a small constant delay (at least twice the maximum wireless link propagation delay) to be introduced per hop.

2.3 Additional Route Maintenance features :

2.3.1. Packet salvaging

When an intermediate node finds the route to the destination to be broken, instead of discarding the packet, it " salvages " it by sending it in an alternate route that it retrieves from its route cache (if present).

2.3.2. Queuing packets destined for a broken link

2.3.3. Automatic route shortening

Source routes in use MAY be automatically shortened if one or more Intermediate nodes in the route become no longer necessary.

3. THE NETWORK SIMULATOR

GLOMOSIM

In an adhoc networks, communication among the nodes in established " on the fly " without using an existing infrastructure. Adhoc networking supports transmission of data packets over multihop transmissions in a network of mobile computing devices.

When protocols are designed for such an environment, it becomes very difficult to evaluate them analytically due to certain factors like node mobility, channel propagation characteristics and radio characteristics. So, in this project we have used a network simulator called GloMoSim to reduce the execution time for detailed simulation model of wireless networks.

GloMoSim (Global Mobile Information system simulation library) is a library based sequential and parallel simulator for wireless networks. The library of GloMoSim has been built based on the parallel discrete-event simulation capability provided by Parsec. GloMoSim has been built on the basis of the layered approach provided by OSI. Standard APIs are used between the simulation layers. Models of protocols at one layer interact with the models in the other layers only by means of using these APIs. The proposed protocol stack in GloMoSim will include models for the channel, radio, MAC, network, transport and the application layer. GloMoSim by itself provides a scalable simulation environment for wireless networks.

In GloMoSim, each network node is initialized as a separate parsec entity. These are considered to be separate logical processes in the system. The memory requirement in such cases would increase very rapidly as GloMoSim is a scalable simulation environment and as such one can imagine the stack space required for thousands of nodes. To solve this problem, a concept called Network Gridding is used. With this , a single entity can simulate several nodes in the system when it contains a data structure, which contains sufficient information about that node.

This is in short to say that we can increase the number of nodes in the system while maintaining the same number of parsec entities.

In GloMoSim each entity represents a geographical area of the simulation. Hence the network nodes which a particular entity represents are determined by the physical position of the nodes.

3.1 Structure of GloMoSim

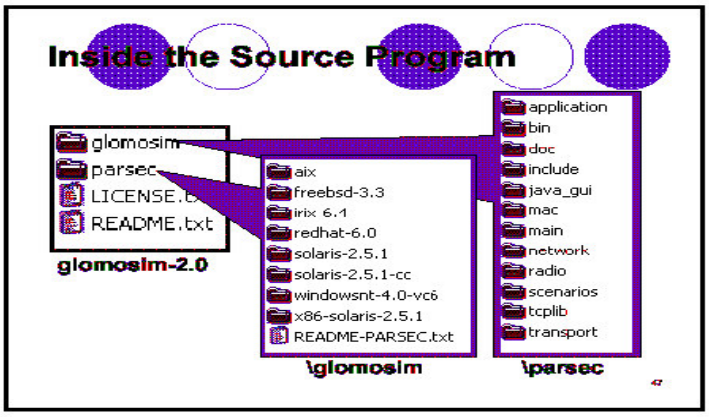
The layered approach in GloMoSim is built with the aim of rapidly integrating models at each layer developed by different people. For example, in this project, a multipath routing protocol for the network layer has only been given. This makes the development process more focused to a single layer and easier.

When each layer is represented by a single parsec entity, the memory problems above stated seems to arise again. For this, the different layers of GloMoSim are integrated into a single parsec entity. Each layer is implemented by functions. The initialization function is called for each

node at the beginning of simulation. Functions are provided to send messages between layers. when a layer receives a particular message, appropriate functions to handle it are invoked and necessary instructions are executed. At the end of simulation the finalize function is also called for each of the nodes. This can be used by layers to collect the necessary statistics.

3.2 Basic Structure of the source directory

The following figure depicts the structure of the directories that can be found inside the GloMoSim folder.



/doc – contains the documentation

/scenarios- contains directories of various sample configuration topologies

/main - contains the basic framework design

/bin- for executable and input/output files

/include- contains common include files

/application- contains code for the application layer

/transport -contains the code for the transport layer
/network -contains the code for the network layer
/Mac -contains the code for the Mac layer
/radio- contains the code for the physical layer
The multipath routing protocol has been designed for the network layer.

3.3 Layers In GloMoSim

LAYERS	**PROTOCOLS**
Mobility	Random waypoint,Random drunken,Trace based
Radio Propagation	Two ray and Free space
Radio Model	Noise Accumulating
Packet Reception Models	SNR bounded,BER based with BPSK/QPSK modulation
Data Link (MAC)	CSMA,IEEE 802.11 and MACA
Network(Routing)	IP with AODV,Bellman-Ford,DSR,Fisheye,LAR scheme1,ODMRP,WRP
Transport	TCP and UDP
Application	CBR,FTTP,HTTP,TELNET

Each and every layer in GloMoSim is represented by C structures in implementation. For example the structure for the radio layer could be represented by

```
typedef struct glomo_radio_str
{
RADIO_TYPE radioType;
/* general radio layer variable */
void * radioVar;
} Radiolayer;
```

Based on the radio type, different model specific structures are accessed through ' radioVar ' .

From the developer point of view of developing a new protocol for a particular layer ,a .pc (parsec) file and the corresponding header file are the ones to be written.

For the start of simulation, the input configuration file (config.in) is to be written or rather defined properly. The details regarding the source, destination, number of packets to be transmitted is all defined in a file called app.conf (application.config). If the placement of the nodes is to be decided by the user, a file called nodes.input comes in to picture and this defines the node co ordinates in terms of the X, Y and the Z axes.

3.4 Details of the Input Files used

3.4.1 APPLICATION.CONFIG FILE

Application (config) file is used to generate the traffic for the simulation (i.e.) to specify the number of packets, source, destination and the duration of transmission.

The version of GloMoSim, which we are using now, provides the facility to use the following traffic generators.

FTP,FTP/GENERIC,TELNET,CBR AND HTTP

FTP

FTP uses 'tcplib' to simulate the file transfer protocol. In order to use FTP, the following format is needed:
FTP <src> <dest> <items to send> <start time>
where
<src> is the client node.
<dest> is the server node.
<items to send> is how many application layer items to send.
<start time> is when to start FTP during the simulation.

If <items to send> is set to 0, FTP will use tcplib to randomly determine the amount of application layer items to send. The size of each item will always be randomly determined by tcplib. Note that the term "item" in the application layer is equivalent to the term "packet" at the network layer and "frame" at the MAC layer.

e.g.:

a) FTP 0 1 10 0S

Node 0 sends node 1 ten items at the start of the simulation, with the size of each item randomly determined by tcplib.

FTP/GENERIC

FTP/GENERIC does not use tcplib to simulate file transfer. Instead, the client simply sends the data items to the server without the server sending any control information back to the client. In order to use FTP/GENERIC, the following format is needed:

FTP/GENERIC <src> <dest> <items to send> <item size> <start time> <endtime>

where
<src> is the client node.
<dest> is the server node.
<items to send> is how many application layer items to send.
<item size> is size of each application layer item.
<start time> is when to start FTP/GENERIC during the simulation.
<end time> is when to terminate FTP/GENERIC during the simulation.

If <items to send> is set to 0, FTP/GENERIC will run until the specified <end time> or until the end of the simulation, which ever comes first.
If <end time> is set to 0, FTP/GENERIC will run until all <items to send> is transmitted or until the end of simulation, which ever comes first.

If <items to send> and <end time> are both greater than 0,
FTP/GENERIC
will run until either <items to send> is done, <end time> is reached, or the simulation ends, which ever comes first.

Eg :FTP/GENERIC 0 1 10 1460 0S 600S

Node 0 sends node 1 ten items of 1460B each at the start of the simulation up to 600 seconds into the simulation. If the ten items are sent before 600 seconds elapsed, no other items are sent.

TELNET

TELNET uses tcplib to simulate the telnet protocol. In order to use TELNET, the following format is needed

TELNET <src> dest> <session duration> <start time>
where
<src> is the client node.
<dest> is the server node.
<session duration> is how long the telnet session will last.
<start time> is when to start TELNET during the simulation.
If <session duration> is set to 0, FTP will use tcplib to randomly determine how long the telnet session will last. The interval between telnet items is determined by tcplib.

e.g.:TELNET 0 1 100S 0S

Node 0 sends node 1 telnet traffic for a duration of 100 seconds at the start of the simulation.

CBR

CBR simulates a constant bit rate generator. In order to use CBR, the following format is needed:

CBR <src> <dest> <items to send> <item size>
<interval> <start time> <end time>

where
<src> is the client node.
<dest> is the server node.
<items to send> is how many application layer items to send.
<item size> is size of each application layer item.
<interval> is the interdeparture time between the application layer items.
<start time> is when to start CBR during the simulation.
<end time> is when to terminate CBR during the simulation.

If <items to send> is set to 0, CBR will run until the specified <end time> or until the end of the simulation, which ever comes first. If <end time> is set to 0, CBR will run until all <items to send> is transmitted or until the end of simulation, which ever comes first.
If <items to send> and <end time> are both greater than 0, CBR will will run until either <items to send> is done, <end time> is reached, or the simulation ends, which ever comes first.
e.g.: CBR 0 1 10 1460 1S 0S 600S

Node 0 sends node 1 ten items of 1460B each at the start of the simulation up to 600 seconds into the simulation. The interdeparture time for each item is 1 second. If the ten items are sent before 600 seconds elapsed, no other items are sent.

HTTP

HTTP simulates single-TCP connection web servers and clients. Bruce following format describes its use for servers:

HTTP <address>
where
<address> is the node address of a node, which will be serving Web pages.
For HTTP clients, the following format is used:
HTTP <address> <num_of_server> <server_1> ... <server_n> <start> <thresh>
where
<address> --- node address of the node on which this client resides
<num_of_server> --- number of server addresses which will follow
<server_1>…<server_n> --- node addresses of the servers which this client will choose between when requesting pages.
<start> is the start time for when the client will begin requesting pages
<thresh> is a ceiling (specified in units of time) on the amount of “think time” that will be allowed for a client.
e.g.:
HTTPD 2
HTTP 1 3 2 5 11 10S 120S

There are HTTP servers on nodes 2, 5, 8, and 11. There is an HTTP client on node 1. This client chooses between servers {2, 5, and 11} only when requesting web pages. It begins browsing after 10S of simulation time have passed.

3.4.2 CONFIG.IN FILE

During the simulation takes the input from the file called config.in, which specifies various parameters to be used.

The details that can be obtained from this input files are as follows.

1. simulation time
2. a random number seed used to initialize part of the seed of various randomly generated numbers in the simulation. This can be used to vary the seed of the simulation to see the consistency of the results of the simulation.

3.parameters stand for the physical terrain in which the nodes are being simulated. For example, the following represents an area of size 100meters by 100 meters. All range parameters are in terms of meters.

4. the number of nodes being simulated.

5. parameter representing the node placement strategy.

RANDOM: Nodes are placed randomly within the physical terrain.

UNIFORM: Based on the number of nodes in the simulation,the physical terrain is divided into a number of cells. Within each cell, a node is placed randomly.

GRID: Node placement starts at (0, 0) and are placed in grid format with each node GRID-UNIT away from its neighbors .The number of nodes has to be square of an Integer.

FILE: Position of nodes is read from NODE-PLACEMENT-FILE. On each line of the file, the x and y position of a single node is separated by a space.

6.Parameters for mobility.

If MOBILITY is set to NO, than there is no movement of nodes in the model. RANDOM-DRUNKEN model, if a node is currently at position (x, y), it can possibly move to (x-1, y),(x+1, y), (x, y-1), and (x, y+1); as long as the new position is within the physical terrain.

RANDOM WAYPOINT, a node randomly selects a destination from the physical terrain. It moves in the direction of the destination in a speed uniformly chosen between MOBILITY-WP-MIN-SPEED and MOBILITY-WP-MAX-SPEED (meter/sec). After it reaches its destination, the node stays there for MOBILITY-WP-PAUSE time period.

The MOBILITY-INTERVAL is used in some models that a node updates its position every MOBILITY-INTERVAL time period. The MOBILITY-D-UPDATE is used that a node updates its position based on the distance(in meters).

7. propagation-limit:

Signals with powers below PROPAGATION-LIMIT (in dB) are not delivered. This value must be smaller than RADIO-RX-SENSITIVITY + RADIO-ANTENNA-GAIN of any node in the model. Otherwise, simulation results may be incorrect. Lower value should make the simulation more precise, but it also make the execution time longer.

8 . propagation-pathloss: pathloss model
FREE-SPACE: Friss free space model.
(path loss exponent, sigma) = (2.0, 0.0)
TWO-RAY:. It uses free space path loss(2.0, 0.0) for near sight and plane earth path loss (4.0, 0.0) for far sight.
9. NOISE-FIGURE: noise figure

10.TEMPARATURE of the environment

11. RADIO-TYPE:

It represents the radio model to transmit and receive packets. it may be one of the following

RADIO-ACCNOISE: standard radio model

RADIO-NONOISE: abstract radio model

12.RADIO-FREQUENCY: frequency (in hertz)

13.RADIO-BANDWIDTH: bandwidth (in bits per second)

14. RADIO-RX-TYPE: packet reception model

15. RADIO-TX-POWER: radio transmission power (in dBm)

16. RADIO-ANTENNA-GAIN: antenna gain (in dB)

17. RADIO-RX-SENSITIVITY: sensitivity of the radio (in dBm)

18. RADIO-RX-THRESHOLD: Minimum power for received packet (in dBm)

19. protocol to be used in MAC layer

20. parameter to enable (Or) disable the PROMISCOUS mode

21. protocol to be used in NETWORK layer

22. ROUTING-PROTOCOL to be used

23. input file to setup applications such as FTP and Telnet.

The file will need to contain parameters that will be use to determine connections and other characteristics of the particular application.

24. Parameters specifying which layer's statistics to be collected.

4. THE PARSEC PROGRAMMING ENVIRONMENT

The PARSEC compiler, called pcc, accepts all the options supported by the C compiler, and also supports separate compilation. C programs (files with .csuffix) and object files (files with .o suffix) can also be compiled and linked with PARSEC programs. PARSEC programs are usually given a .pc extension.

4.1 Options

PARSEC compiler also supports the following options:

-sync Specify one of the synchronization algorithms:
-mpc Message-passing C: ignores message timestamps
cons Conservative
opt Optimistic (not implemented yet)
-c Generate ".o" and ".pi" files.
-E Generate ".c" and ".pi" files.
-P Inhibit line number translation. (Normally, the PARSEC compiler inserts line numbers into the intermediate C file so that compiler and runtime errors report the correct line in the PARSEC file.)
-env Show environment names set for pcc.
-pcc_cc_options û use these options for compiling.
-pcc_linker_options û use these options for linking.
-ini Save the auto-generated initialization file.
-ff Enable compilation of friend functions.
-clock Set the default representation for clocktype.

The following examples illustrate how to compile PARSEC programs on a sequential architecture.

% pcc -o example example.pc
This generates an executable file example in the current working directory.

4.2 Separate Compilation

PARSEC supports separate compilation of entities. Entities defined in one file and used in a second file must be declared extern in the second file.

4.3 Common PARSEC Runtime Errors

Runtime errors/warnings are detected by PARSEC and appropriate messages are sent to stderr. The most common errors are related to stack allocation. PARSEC adds a finalize block at the end of the entity body

"Run out of memory for a message." Memory has been exhausted, possibly due
to a deadlock, an infinite loop, or an imbalance in message processing.

"Error: Trying to send messages to remote entity." A sequential program is attempting to send a message to a remote processor, probably indicating that an ename variable is uninitialized or has become corrupted.

"Error: Trying to receive messages from remote entity." Same as above, but less common.

"Thread Local Storage Key Create Error." Error in creating Windows NT threads.
"Fail to create NT thread." Ditto.
"*** PARSEC error. Failed in à" Pthread setup error.
"*** PARSEC error. Failed to create threads." Pthread setup error.

"Too many different types of messages. Recompile the runtime to support more
message types." By default, the maximum number of message types the Parsec
system can support is 64. The number can be increased, but it has a detrimental impact on performance.

"Run out of memory in setting entity parameters." Apparently, the parameters
for this entity are HUGE.

"Wrong NEW_ENTITY_ACK." An error in creating a new entity has occurred.

"Unrecognized Remote Message." Somehow the message has become corrupted,
possibly due to memory mismanagement in the program.

5.2 MODULE WISE ALGORITHMS

5.2.1 MODULE 1 (EXTRACTION OF DISJOINT PATHS)

5.2.1.1 Extraction of disjoint paths

Route Discovery phase

1. when any node wants to send a message to a destination, it originates a RREQ packet . This is broadcast locally.
2. The algorithm for handling RREQ is as below

```
If  RREQ  has been received already
discard
Else
{
    if the node's address is already in the route record
    discard
    else
{
    if RREQ's destination address is same as the node's address ,
    send RREP packet
    else
    {
    if there is any route from this node to the destination
    send RREP packet
    else
    {
    append this nodes address to the RREQ,broadcast
    store RREQ packet's id in the seen table.
}
}
```

}
}

5.2.1.2 Methodology to find node disjoint path

1.Use a " deleted " bit in each route cache entry to indicate whether the path is feasible i.e. it is one of the node disjoint multiple paths

2.whenever an entry is deleted from the route cache table
all the entry's deleted bit is reset and the variable FIRST (to indicate t hat the node disjoint path discovery must be performed again) is set.

5.2.1.3 Specific algorithm

function " getroute "---to return the route from the cache table
If the value of the variable FIRST = 1

1. Store the primary path in the temporary array of disjoint paths
2. For all the paths in the route cache table

if the path is disjoint with paths already stored in the temporary array of disjoint paths

Add the disjoint path in to the array
else
set "deleted bit" of that route cache entry

5.2.2 Module 2(LOAD DISTRIBUTION)

Calculation of load distribution:

In GloMoSim the links between the nodes is assumed to have uniform bandwidth. But to simulate a real time constraint , we have randomly assigned the bandwidth between the various links non-uniformly.

For the maximum utilization of bandwidth we calculate the number of packets to be transmitted in each path as below.

1.Find minimum reserve bandwidth (mrbw) for all the disjoint paths discovered so far.

mrbw[i]= min(R kl / k,l E path i)

where

Rkl- bandwidth between the kth and the lth node in the path i

2. Find the number of packets to be transmitted in each path as

nop[i] = (bwth[i] / sumofbw) * totalnop

where

nop[i] - number of packets to be transmitted in the ith node disjoint path mrbw[i] - minimum reserve bandwidth of the ith node disjoint path. sumofbw - sum of minimum bandwidth of all node disjoint paths.

totalnop - total number of packets to be transmitted from the source to the destination.

5.2.3 MODULE 3 (PERFORMANCE ANALYSIS)

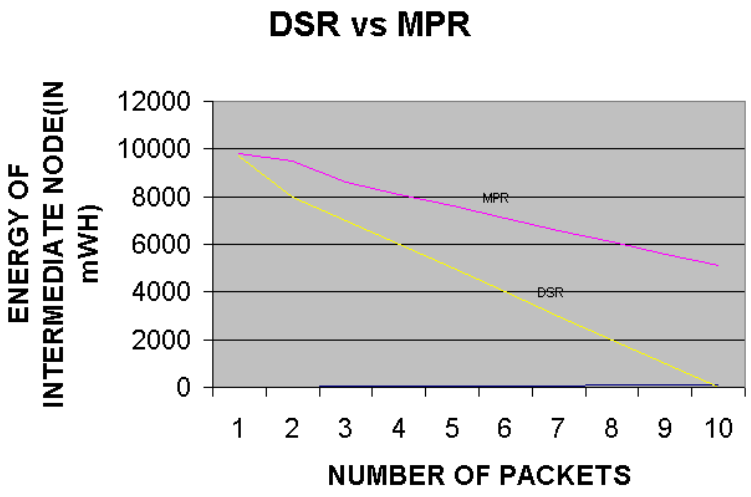
The third module is concerned with the analysis of performance of our multipath routing protocol with the DSR protocol that has been already inbuilt in GloMoSim.This is just a comparative study to prove how efficient this multipath routing protocol is.

For the purpose of analysis, several simulation runs were made,first on our multipath routing protocol.During these runs, the statistics that were studied is as below
1.Number of packets
2.Energy in the intermediate nodes after transmitting the number of packets.

Then the DSR simulation was run on the basis of same inputs and the above said statistics were observed .Several runs were made for different number of packets and the conclusions derived on their basis are as under

1. The intermediate nodes in the multipath routing protocol have more residual energy after transmitting the specified number of packets than the ones in the DSR.
2. The mean time to node failure is increased significantly.
3. Bandwidth of the entire network is used in an efficient manner.
4. The number of routes lying idle at any point of time is minimal when compared to normal DSR.

The above said benefits are illustrated by the graph shown in figure 3.2.5.a.It can be seen that the nodes stand out much longer in terms of energy.



6 IMPLEMENTATION DETAILS

6.1 Simulation Procedure in GloMoSim

First , we need to specify the necessary input parameters in the Config.in file as said above.. For our simulation procedure, we have been specific about certain parameters as mentioned below to enable hassle free simulation

Terrain range – (2000,2000)
Number of nodes – 49 (This is a scalable simulator. Hence number of nodes can be increased at will.
Routing protocol – DSR

These parameters were adhered to for the whole process of experimentation with the new protocol .

A copy of the config.in file used for the simulation is given below for reference.

CONFIG.IN

```
SIMULATION-TIME     15M
SEED                1
TERRAIN-DIMENSIONS  (2000, 2000)
```

```
NODE-PLACEMENT      UNIFORM
MOBILITY  NONE
MOBILITY-POSITION-GRANULARITY 0.5
PROPAGATION-LIMIT      -111.0
PROPAGATION-PATHLOSS   TWO-RAY
NOISE-FIGURE    10.0
TEMPARATURE     290.0
RADIO-TYPE              RADIO-ACCNOISE
RADIO-BANDWIDTH     2000000
RADIO-RX-TYPE          SNR-BOUNDED
RADIO-RX-SNR-THRESHOLD  10.0
RADIO-TX-POWER        15.0
RADIO-ANTENNA-GAIN  0.0
RADIO-RX-SENSITIVITY -91.0
RADIO-RX-THRESHOLD -81.0
MAC-PROTOCOL        802.11
NETWORK-PROTOCOL    IP
NETWORK-OUTPUT-QUEUE-SIZE-PER-PRIORITY 100
ROUTING-PROTOCOL    DSR
APP-CONFIG-FILE   ./app.conf
APPLICATION-STATISTICS          YES
TCP-STATISTICS                YES
UDP-STATISTICS                YES
ROUTING-STATISTICS            YES
NETWORK-LAYER-STATISTICS        YES
MAC-LAYER-STATISTICS           YES
RADIO-LAYER-STATISTICS          YES
CHANNEL-LAYER-STATISTICS        YES
MOBILITY-STATISTICS            YES
GUI-OPTION   YES
GUI-RADIO    YES
```

GUI-ROUTING YES

After specifying the configuration parameters, we need to define the source , destination and other details in the application.config file. We have used the CBR (constant bit rate) traffic generator. The other traffic generators could be used as well.
Strings specified during the process of testing individually are given below

CBR 10 2 10 512 15S 0S 110S
CBR 7 0 35 512 50S 0S 100S
CBR 17 11 40 512 25S 0S 0S
CBR 14 10 20 512 20S 0S 0S
CBR 34 48 10 512 45 0S 75S
CBR 16 30 73 512 20 0S 25S
CBR 12 6 50 512 33S 0S 0S
TELNET 16 30 10S 150S

After specifying these two input files , if we mention that the node placement should be according to a placement file, then we need to mention the co ordinates in a separate file called nodes.input.

A sample of how that file would look like is given below

NODE-PLACEMENT-FILE

Format: nodeAddr 0 (x, y, z)
The second parameter is for the consistency with the mobility trace format.
0 0 (20.2, 0.9, 0.11)
1 0 (20.3, 30.8, 0.01)

2 0 (20.4, 60.7, 0.12)
3 0 (20.5, 90.6, 0.05)
4 0 (50.6, 0.5, 0.09)
5 0 (50.7, 30.4, 0.10)
6 0 (50.8, 60.3, 0.12)
7 0 (50.9, 90.2, 0.21)
8 0 (80.1, 0.1, 0.30)
9 0 (80.2, 30.0, 0.16)

The list of commands to be specified in the appropriate locations is given below

E:\GloMoSim\main\makent

- This is a batch file (makent.bat). This contains appropriate calls to the parsec compiler. This results in creation of the GloMoSim.exe file which is to be used for simulation

E:\GloMoSim\bin\GloMoSim config.in

- This is to start the process of simulation. The exe file takes in input from the config.in file.
- When the above command is executed, the process of simulation starts and the screen shot is as shown in figure 3.4.1.a

The appearance on the screen shows a series of numbers which denote the current simulation time , the node involved in the message switch and other details.

This indicates the end of the simulation run. GloMoSim internally writes the statistics required (as specified in the finalize function of the protocol.pc file)

The statistics file generated in one of our simulation runs is enclosed

6.2 General statistics format

Node: 0, Layer: RadioAccnoise, Signals transmitted: 1
Node: 0, Layer: RadioAccnoise, Signals arrived with power above RX Sensitivity: 7
Node: 0, Layer: RadioAccnoise, Signals arrived with power above RX threshold: 2
Node: 0, Layer: RadioAccnoise, Signals received and forwarded to MAC: 2
Node: 0, Layer: RadioAccnoise, Collisions: 0
Node: 0, Layer: RadioAccnoise, Energy consumption (in mWhr): 225.000
Node: 0, Layer: 802.11, pkts from network: 0
Node: 0, Layer: 802.11, UCAST (non-frag) pkts sent to chanl: 0
Node: 0, Layer: 802.11, BCAST pkts sent to chanl: 1
Node: 0, Layer: 802.11, UCAST pkts rcvd clearly: 0
Node: 0, Layer: 802.11, BCAST pkts rcvd clearly: 2
Node: 0, Layer: 802.11, retx pkts due to CTS timeout: 0
Node: 0, Layer: 802.11, retx pkts due to ACK timeout: 0
Node: 0, Layer: 802.11, pkt drops due to retx limit: 0
Node: 0, Layer: 802.11, RTS Packets ignored due to Busy Channel 0
Node: 0, Layer: 802.11, RTS Packets ignored due to NAV 0
Node: 0, Layer: RoutingMpr, Number of Requests Txed = 1
Node: 0, Layer: RoutingMpr, Number of Replies Txed = 0

Node: 0, Layer: RoutingMpr, Number of Errors Txed = 0
Node:0, Layer: RoutingMpr, Number of CTRL Packets Txed = 1
Node: 0, Layer: RoutingMpr, Number of Routes Selected = 0
Node: 0, Layer: RoutingMpr, Number of Hop Counts = 0
Node: 0, Layer: RoutingMpr, Number of Data Txed = 0
Node: 0, Layer: RoutingMpr, Number of Data Originated = 0
Node: 0, Layer: RoutingMpr, Number of Data Received = 0
Node: 0, Layer: RoutingMpr, Number of Link Breaks = 0
Node: 0, Layer: RoutingMpr, Number of Salvaged Packets = 0
Node: 0, Layer: RoutingMpr, Number of Dropped Packets = 0
Node: 0, Layer: NetworkIp, Number of Packet Attempted to be Sent to MAC: 1
Node: 0, Layer: NetworkIp, Number of Packets Routed For Another Node: 0
Node: 0, Layer: NetworkIp, Number of Packets Delivered To this Node: 0
Node: 0, Layer: NetworkIp, Total of the TTL's of Delivered Packets: 0
Node:0, Layer: NetworkIp, Number Fragments dropped because Node was Unreachable: 0
Node: 0, Layer: NetworkIp, Number Fragments dropped because TTL expired: 0
Node:0, Layer: TransportUdp, Number of pkts from application 0.
Node:0, Layer: TransportUdp, Number of pkts to application 0.

Our simulation involved analysis of the statistics for the nodes numbered 16,23 , 30 in our protocol. We did a study on the energy spent in these nodes for the evaluation of performance of our protocol.

GUI options

It is possible to view the simulation in the GUI which is a front end inbuilt facility based on java

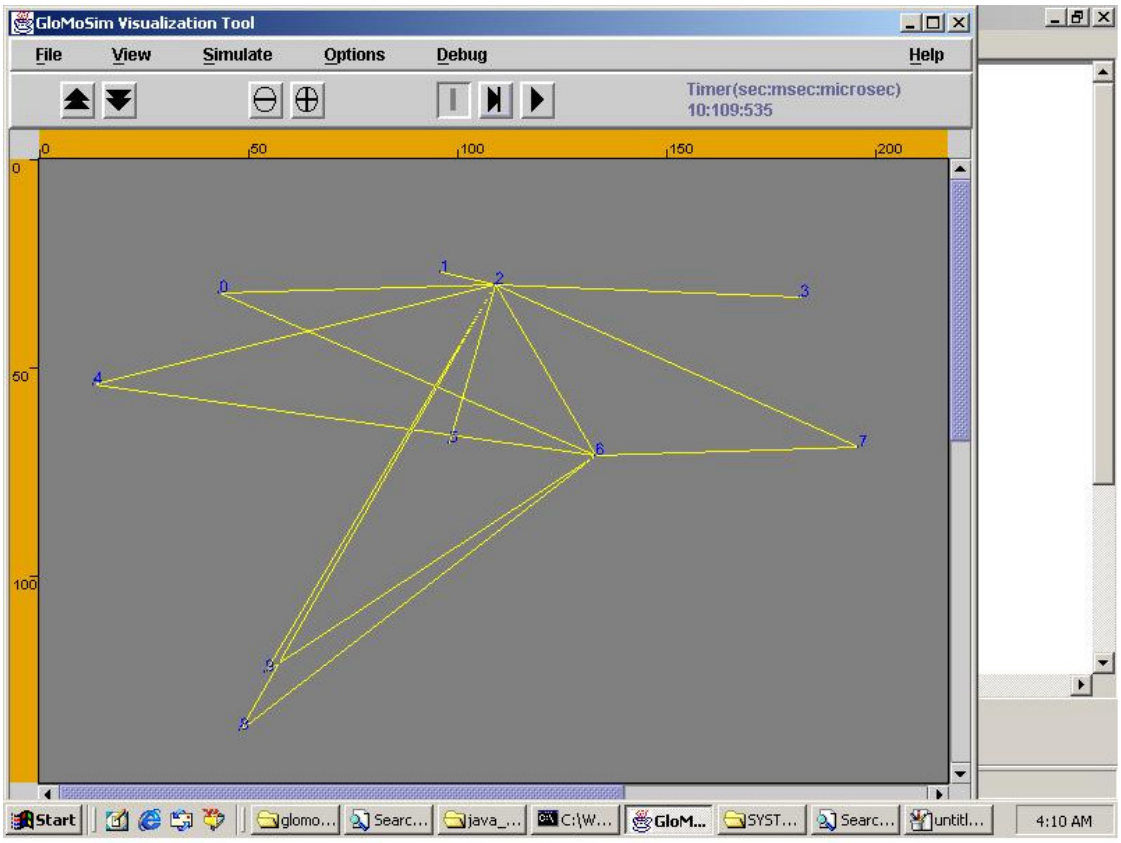
For this , the configuration file needs to be converted in to a trace file .
This requires the execution of the command

E:\GloMoSim\bin\GloMoSim config.in>a.trace
Where a.trace is the name of the trace file.

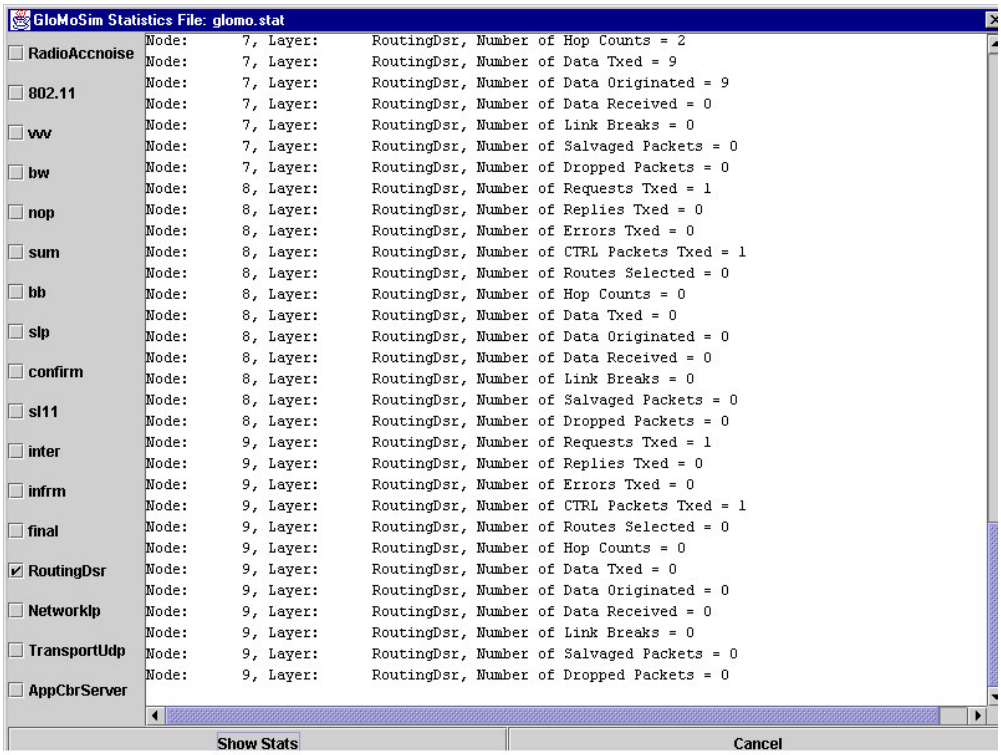
The trace file needs to be copied in to the java_gui folder.
And the following command is to be run

E:\GloMoSim\java_gui\javac Glomomain.java
- this is java compilation of the source code for the front end facility.
E:\GloMoSim\java_gui\java Glomomain

This will display the java interface where in the nodes and the communication between them can be seen on the screen .
An illustration is given in the figure.



AFTER PLAYBACK



STATISTICS VIEW

FUTURE ENHANCEMENTS

1. To make this multipath routing protocol work for multiple sources and multiple destinations
2. An enhancement in the protocol to handle a type of a situation described below :

Till now, we have chosen only completely node disjoint paths.
There may be network conditions wherein there exists many node disjoint multiple paths to the intermediate node only .In such cases this node will act a limiting node inspite of the fact that multiple paths exist halfway.

The solution lies in transmission of packets in multiple paths till the intermediate node, buffering it there and then transmitting it sequentially to the destination.

REFERENCES

1. “Adhoc Networks “ By Charles.E.Perkins.

2.. “draft-ietf-manet-dsr-07.txt” released by Internet Engineering Task

Force (MANET GROUP).

3. Paper on “ On-Demand Multipath Routing for Mobile Ad Hoc Networks”

Kui Wu and Janelle Harms .

4. Paper on “ Load Balancing of Multipath Source Routing in Adhoc Networks “ by Liang Fang Zang , Zhegua Zao.

CONCLUSION

So far, we have attempted to design an efficient multipath routing protocol for an adhoc network. We concentrated on the constraints that arise in data transfer and used the multipaths accordingly.

This is just a beginning of how good an existing protocol could be modified to suit the network needs. Additional works that can be made in this regard are presented in the future enhancement section.

Thus, in this project, we have designed a multipath routing protocol, which could be added to GloMoSim as an inbuilt code to facilitate further research in the area of adhoc networks.

.