Fish Eye State Routing in Mobile Ad-Hoc Networks

Submitted in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

by

Jaidev Ramakrishna

Raghu Varier

Kartik Chawla

Shashank Alva

Supervisor

Manjusha Deshmukh

Department of Computer Engineering

MES's Pillai Institute of Information Technology, Engineering, Media Studies and Research,

New Panvel, Navi Mumbai - 410 206

2014-15

Department of Computer Engineering

MES's Pillai Institute of Information Technology, Engineering, Media Studies & Research

New Panvel – 410 206.

CERTIFICATE

This project report entitled **"Fish Eye Routing in Mobile Ad-Hoc Networks"** by

Jaidev Ramakrishna

Raghu Varier

Kartik Chawla

Shashank Alva

is certified for the submission for the degree of Bachelor of Engineering in Computer Engineering.

SUPERVISOR	PROJECT COORDINATOR
(Prof. Manjusha Deshmukh)	**(Prof. Varunakshi Bhojane)**
Department of Computer Engineering	Department of Computer Engineering

HEAD OF DEPARTMENT

(Dr. Madhumita Chatterjee)

Department of Computer Engineering

PRINCIPAL

(Dr. R. I. K. Moorthy)

M E S's Pillai Institute of Information Technology

Date:

Place:

Department of Computer Engineering

MES's Pillai Institute of Information Technology,

Engineering, Media Studies & Research

New Panvel – 410 206.

<u>Project Report Approval</u>

This project report entitled **"Fish Eye Routing in Mobile Ad-Hoc Networks"** by

Jaidev Ramakrishna

Raghu Varier

Kartik Chawla

Shashank Alva

is approved for the degree of Bachelor of Engineering in Computer Engineering.

Examiners

1.---

2.---

Supervisors

1.---

2.---

Chairman

Date:

Place:

Declaration

We declare that this written submission for our B.E project entitled **“Fish Eye State Routing in Mobile Ad-Hoc Networks”** represents our original ideas in our own words, and that where the ideas or words of others have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity, and have not misrepresented, fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute, and can also evoke penal action from the sources which have thus not been properly cited, or from whom proper permission has not been taken when needed.

Jaidev Ramakrishna ----------------------------------

Raghu Varier ----------------------------------

Kartik Chawla ----------------------------------

Shashank Alva ----------------------------------

Date:

Place:

Acknowledgement

We would like to express our profound gratitude towards our project guide, **Prof. Manjusha Deshmukh,** whose constant encouragement enabled us to work constructively and achieve our goals. Her deep expertise in our area of research helped us immensely. We are indebted to her for taking time out of her busy schedule, and placing her knowledge at our disposal.

We are thankful to **Dr. Madhumita Chatterjee, H.O.D** of the Computer Engineering Department and **Prof. Varunakshi Bhojane, Project Coordinator**, for their guidance, encouragement and support during the development of our project. They were instrumental in making all required resources available to us.

We would also like to thank **Dr. R. I. K Moorthy, Principal**, Pillai Institute of Information Technology, New Panvel, for providing an extremely supportive academic environment, and for providing excellent infrastructure and facilities. We acknowledge the support and encouragement offered by all the staff members of the Computer Engineering Department

Jaidev Ramakrishna

Raghu Varier

Kartik Chawla

Shashank Alva

INDEX

Sr. no	Chapter Name	Page No.
i.	Abstract……………………………………………………………..	i
ii	List of Figures…………………………………………………………	ii
1.	Introduction……………………………………………………….....	1
	1.1 Project Overview……………………………………………….	1
	1.2 Fish Eye Routing………………………………………………	2
	1.3 Advantages of FSR……………………………………………..	4
2.	Literature Survey………………………………………………………	5
3.	Fish Eye Routing………………………………………………………	8
	3.1 Algorithm……………………………………………………	8
	3.2 Neighbor Discovery……………………………………………	9
	3.3 Information Dissemination……………………………………..	9
	3.3.1 LSP Integrity………………………………………..	10
	3.3.2 Update Interval……………………………………..	11
	3.4 Route Computation…………………………………………….	12
4.	Virtual Grid System……………………………………………………	14
	4.1 Grid System…………………………………………………….	14
	4.2 Location Awareness……………………………………………	15
	4.3 Gateway Election………………………………………………	16
	4.4 Algorithm…..………………………………………………......	17
	4.4.1 Strong Form………………………………………....	18
	4.4.2 Weak Form…………………………………………	19
5.	Requirement Analysis…………………………………………………	22

5.1 Software………………………………………………………….. 22

5.2 Packages & Dependencies…………………………………….. 23

5.3 Hardware…………………………………………………………. 24

6. Simulation…………………………………………………………... 25

6.1 20 Node Network……………………………………………… 25

6.2 60 Node Network……………………………………………… 28

7. Conclusion…………………………………………………………. 30

8. Appendices………………………………………………………….. 31

I 20 Nodes – Sparse Topology………………………………….. 31

II 40 Nodes – Medium Topology………………………………… 35

III 60 Nodes – Dense Topology…………………………………... 40

9. References…………………………………………………………... 45

ABSTRACT

In mobile ad hoc networks, all nodes are mobile and can be connected dynamically in an arbitrary manner. All nodes of these networks behave as routers and take part in discovery and maintenance of routes to other nodes in the network. This feature presents a great challenge to the design of a routing scheme, since link bandwidth is very limited, and the network topology changes as users roam.

This project investigates and implements a routing approach for ad hoc wireless networks: Fisheye State Routing (FSR), a simple, efficient and scalable routing solution in MANETs. It is based on Link State routing, and introduces the notion of multi-level Fish Eye Scope, to reduce the routing update overhead in large networks. Nodes exchange Link State Entries with their neighbors with a frequency which depends on distance to destination. From Link State Entries, nodes construct a Topology Map of the entire network, and compute optimal routes.

Additionally, this project explores the possibility of introducing the concept of a Virtual GRID into the FSR algorithm. The grid is based upon the geographical location of the nodes. This location is calculated using GPS or some other triangulation system. The nodes organize themselves into grids, and elect a single gateway node. This election takes place based on many desirable parameters, such as mobility, transmitting power, a large enough range to communicate with the gateways of adjacent grids and available bandwidth.

Once a gateway has been elected, all inter-grid packets will be routed through the gateways. All nodes will forward the packet to their respective gateway, which maintains a complete network map and grid information. Once the gateway of the grid where the destination node is located receives the packet, it passes it to that node. Intra grid packets are routed directly, without the involvement of the gateway nodes. In case a gateway node fails, or moves out of the grid square, the remaining nodes conduct an election once again and choose a new gateway node.

The classic FSR algorithm reduces the overheads caused by flooding of Link State Packets, which frees bandwidth for data transmission and reduces power consumption. The virtual grid system does require some processing overhead, but the main advantage of it is that routing paths are more stable, and need to be recalculated less frequently. Another advantage is that only gateway nodes need to maintain a complete network map. The other nodes can afford to assume that the gateway will guarantee transmission. Throughput and end to end latency also increase, and net power consumption reduces.

List of Figures

Sr. no	Name	Page No.
1	Figure 1.1 - Fish Eye Scope………………………………………......	3
2	Figure 4.1 - Virtual Grid………………………………………………	15
3	Figure 4.2 - Location Awareness……………………………………...	15
4	Figure 4.3 - Gateway Election & Routing Table……………………...	16
5	Figure 4.4 - Routing using Gateway Nodes…………………………...	17
6	Figure 4.5 - Strong Form………………………………………………	19
7	Figure 4.6 – Weak Form………………………………………………	21
8	Figure 5.1 – Package Installation…………….................................	24
9	Figure 5.2 – Compiler Configuration………………………………….	24
10	Figure 6.1 – Gateway Election and Grid Creation…………………….	26
11	Figure 6.2 – Inter Grid Transmission………………………………….	27
12	Figure 6.3 – Intra Grid Transmission………………………………….	27
13	Figure 6.4 – Gateway Election and Grid Creation…………………….	28
14	Figure 6.5 – Inter Grid Transmission………………………………….	28
15	Figure 6.6 – Inter Grid Transmission………………………………….	29
16	Figure 6.7 – Intra Grid Transmission………………………………….	29

Chapter 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Internet and Mobile Communication systems are integral components of the modern world, and have become central parts of our everyday lives. As more and more of our systems and amenities move online, it has become more necessary to ensure that the infrastructure upon which these systems rest is robust, extensible, flexible and widely available. For this purpose, it is necessary to rethink the theoretical and technological paradigms that form the base of our communication systems, to ensure that they can provide reliable services to end users. Wireless terminals like Mobile Phones are now the most common methods of leveraging services. Hence, it is necessary to optimize wireless networks to ensure that the Quality of Service they provide is compatible with the scope of their intended purposes.

Since Wireless terminals are power and bandwidth limited, it is obvious that making the methods of connectivity they use more efficient is the logical way to enhance the services they provide. To this end, we have set out to rethink the paradigms that lie at the very lowest level of wireless networks – Ad Hoc routing protocols. There is a pressing need for a system that is more power efficient, and yet provides reliable services.

1.2 FISH EYE ROUTING

Fisheye State Routing (FSR) is an implicit hierarchical routing protocol. Also considered a proactive protocol and is a link state based routing protocol that has been adapted to the wireless ad hoc environment. Relays on link state protocol as a base, and it has the ability to provide route information instantly by maintaining a topology map at each node. Thus will maintain updated information from the neighbor node through a link state table. In each node the network, a full topology map is stored then utilized (List of ad hoc routing protocols).

According to Kleinrock and Stevens, FSR uses the "fisheye" technique where the technique was used to reduce the size of information required to represent graphical data. The eye of a fish captures with high detail the pixels near the focal point. The detail decreases as the distance from the focal point increases. In routing, the fisheye approach translates to maintaining accurate distance and path quality information about the immediate neighborhood of a node, with progressively less detail as the distance increases. One use of the FSR is reducing overhead control traffic. It has also shown a good performance in terms of successful packet delivery in the presence of low mobility. During the case of high mobility, in order to get a successful packet delivery, the update interval time must be properly selected.

Fisheye Routing determines routing decisions using a table-driven routing mechanism similar to link state. The table-driven ad hoc routing approach uses a connectionless approach of forwarding packets, with no regard to when and how frequently such routes are desired. It relies on an underlying routing table update mechanism that involves the constant propagation of routing information. A table-driven mechanism was selected over an on-demand mechanism based on the following properties:

1. On-Demand routing protocols on the average create longer routes than table driven routing protocols.
2. On-Demand routing protocols are more sensitive to traffic load than Table-Driven in that routing overhead traffic and latency increase as data traffic source/destination pairs increase.

3. On-Demand Routing incurs higher average packet delay than Table Driving routing which results from latency caused by route discovery from new destinations and less optimal routes.
4. Table-Driven routing accuracy is less sensitive to topology changes. Since every node has a ‘view’ of the entire network, routes are less disrupted when there is link breakage (route reconstruction can be resolved locally).
5. Table-Driven protocols are easier to debug and to account for routes since the entire network topology and route tables are stored at each node, whereas On-Demand routing only contain routes that are source initiated and these routes are difficult to track over time.

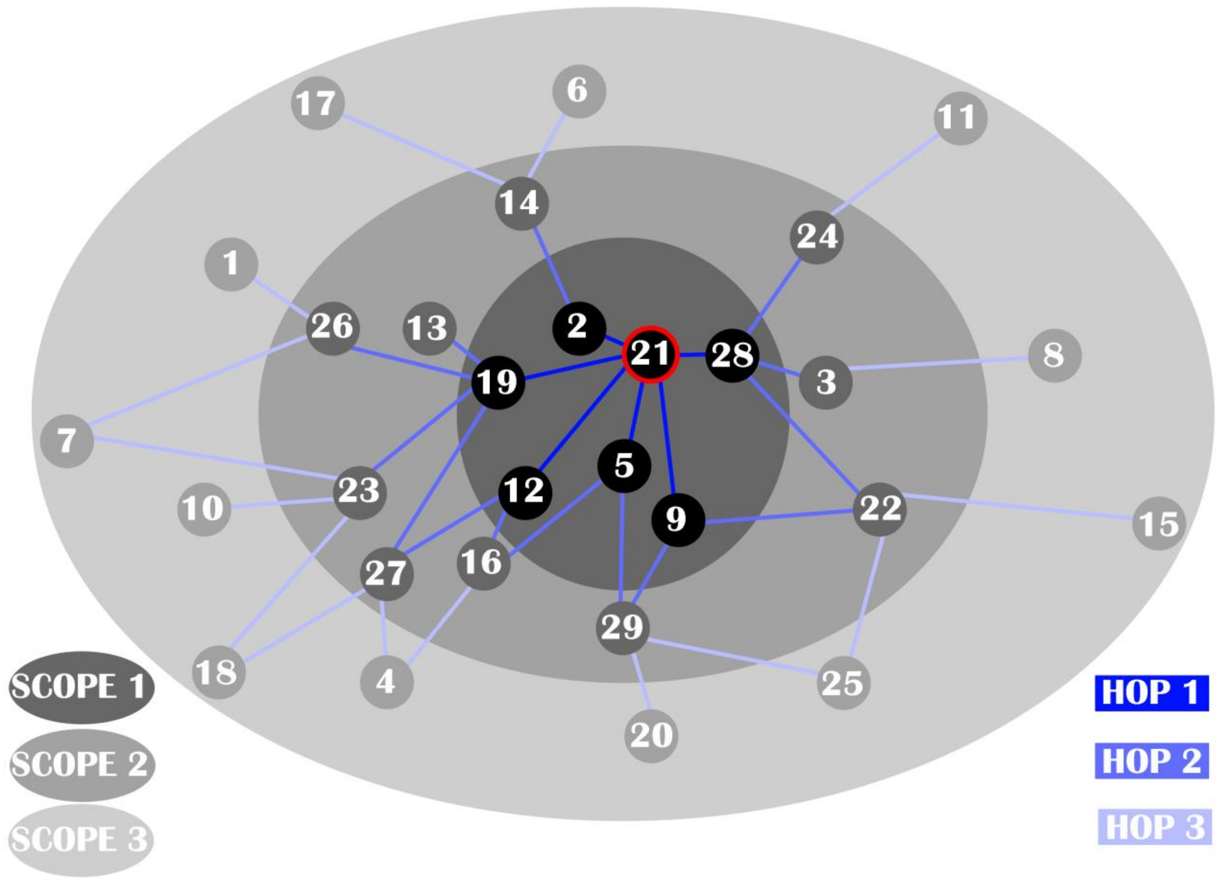


Figure 1.1 - Fish Eye Scope

1.3 ADVANTAGES OF FSR

1. **Distributed operation:** The protocol should be distributed, meaning that it should not be dependent on a centralized controlling node. This makes the system more robust to failure and growth.
2. **Fast convergence:** Routes should be quickly determined in the presence of network changes. This means that when topology changes occur, the protocol should be able to quickly determine optimal new routes.
3. **Loop free:** To have good overall performance, the routing protocol should supply routes that are loop-free. This avoids wasting bandwidth and CPU resources.
4. **Optimal routes:** It is important for the protocol to find routes with the least number of hops. This reduces bandwidth and CPU consumption. In addition, it leads to lower overall routing delay.
5. **Low overhead control traffic:** Bandwidth in a wireless network is a limited resource. The protocol should minimize the amount of overhead control messages for routing.

Chapter 2

LITERATURE SURVERY

“FSR resembles Link State routing in that it propagates Link State updates. However, the updates are propagated as aggregates, periodically (with period dependent on distance) instead of being flooded individually from each source. FSR leads to major reduction in link O/H caused by routing table updates. It enhances scalability of large, mobile ad hoc networks.” **- IETF MANET Working Group Internet Draft.**

“DSR shows best performance than AODV, FSR and ZRP in terms of packet delivery ratio and throughput as a function of pause time. FSR show lowest end-to-end delay and ZRP has less average jittering than DSR, AODV and FSR. DSR and AODV performed the worst in case of average jitter and ZRP performed the worst in case of throughput.” - **Performance Analysis of DSR, AODV, FSR and ZRP Routing Protocols in MANET.**

“Finally, the reduced flooding in routing traffic overhead and periodical propagation of link state in-formation makes Grid FSR suitable for the high mobile dynamic changing network topology and thus the throughput is good with the high mobility of nodes, and therefore the average end-to-end delay is also very low." - **A Reduced Flooding Algorithm and Comparative Study of Grid Fisheye State Routing Protocol for MANET.**

"From our simulations, we conclude that the fisheye technique is efficient in the mobile ad hoc networks studied, sine that maximum user capacity is achieved when the routing control traffic attenuation is prominent. WE can also gather from simulations that the FSR will suffice in a highly mobile network." - **The Fisheye Routing Technique in Highly Mobile Ad Hoc Networks.**

"At the cost of a little longer routing path, GFSR provides a highly efficient solution with lower control message overhead and fewer routing nodes which are responsible for exchanging routing information. In high-density environments, our method exhibits low interference and fewer collisions and it leads to more efficient communication among mobile nodes." - **Efficient Fisheye State Routing Protocol using Virtual Grid in High-Density Ad-Hoc Networks.**

"Moreover, as the number of nodes increases, the FSR protocol trends to be zero delay as shown in Figure 8. The reason for this due to the behavior of the FSR protocol, because the routes are available the moment they are needed. Also, each node consistently maintains an up-to-date route to every other node in the network, a source can simply check its routing table when it has data packets to send to some destination and begin packet transmission so, no delay occur." - **Performance Measurement of Some Mobile Ad Hoc Network Routing Protocols.**

"Simulation show that FSR is more desirable for large mobile networks where the mobility is high and the bandwidth is low. By choosing the proper number of scope levels and radius size, FSR proves to be a flexible solution to the challenge of maintain accurate routes in ad hoc networks." - **Fisheye State Routing, A Routing Scheme for Ad hoc Wireless Networks.**

“Evolutionary Fisheye State Routing Protocol [which] applies the genetic algorithmic approach for finding the optimal path to the existing Fisheye State Routing Protocol This gives the better delivery of packets to the destination and throughput and reduces overhead and delays on the network. The EFSR will be definitely a promising protocol of future. Extensive simulation results reveal that the proposed scheme features better transmission delay, route convergence time, system efficiency and system throughput.” **- EFSR: Evolutionary Fisheye State Routing Protocol in Mobile Ad Hoc Networks.**

“Through simulation, Fisheye routing has exhibited good performance in reducing overhead control traffic. It also performs well in terms of successful packet delivery in the presence of low mobility. Proper selection of the update interval time is necessary for good successful packet delivery in the presence of high mobility.” “The table-driven approach makes the protocol insensitive to traffic source/destination pair density. The fisheye update mechanism significantly reduces the overhead traffic that plagues conventional and proposed table driven schemes. This results in good scalability to network size.” **- Design and Implementation of a Fisheye Routing Protocol for Mobile Wireless Ad Hoc Networks.**

Chapter 3

FISH EYE ROUTING

3.1 ALGORITHM [9]

FSR is based on a link-state foundation updating mechanism. Which has the following feature: maintaining a topology map at each node. This mechanism reduces the control overhead by disseminating topology information using the fisheye technique, where routing information is updated at different rates depending on the distance from the source, and it can be broken down into:

1. Node stores the Link State for every destination in the network
2. Node periodically broadcast update messages to its neighbors
3. Updates correspond to closer nodes propagate more frequently

There are 3 main tasks in the routing protocol:

1. **Neighbor Discovery:** responsible for establishing and maintaining neighbor relationships.
2. **Information Dissemination:** responsible for disseminating Link State Packets (LSP), which contain neighbor link information, to other nodes in the network.
3. **Route Computation:** responsible for computing routes to each destination using the information of the LSPs.

Initially every node starts has an empty topology table and an empty neighbor list. Invoking the Neighbor discovery mechanism in order to acquire neighbors and to maintain current neighbor relationships after its local variables are initialized. By using the Information Dissemination mechanism, the distribution of LSP in the network is produced. Each node has a database consisting of the collection of LSPs originated by each node in the network. From this database, the node uses the Route Computation mechanism to yield a routing table for the protocol. This process is periodically repeated.

3.2 NEIGHBOR DISCOVERY [9]

This mechanism is responsible for establishing and maintaining neighbor relationships. Neighbors can meet each other simply by transmitting a special packet (a HELLO packet) over the broadcast medium. In the wireless network, HELLO packets are periodically broadcasted and nodes within the transmission range of the sending node will hear these special packets and record them as neighbors. Each node associates a TIMEOUT value in the node's database for each neighbor. When it does not hear a HELLO packet from a particular neighbor within the TIMEOUT period, it will remove that neighbor from the neighbor list. TIMEOUT values are reset when a HELLO message is heard.

HELLO Packets also contain the list of routers whose HELLO Packets have been seen recently. Nodes can use this information to detect the presence of unidirectional or bidirectional links by checking if it sees itself listed in the neighbor's HELLO Packets.

3.3 INFORMATION DISSEMENATION [8]

This mechanism is responsible for distributing LSPs to the nodes in the network. Its two main functions are to handle the LSP integrity and updating interval.

3.3.1 LSP INTEGRITY

After the router generates a new LSP, the new LSP must be transmitted to all the other routers. A simple scheme is flooding, in which each packet received is transmitted to each neighbor except the one from which the packet was received. Because each router retains the most recently generated LSP from other nodes, the router can recognize when it is receiving a duplicate LSP and refrain from flooding the packet more than once.

The problem with this flooding is that a router cannot assume that the LSP most recently received is the one most recently generated by that node. Two LSPs could travel along different paths and might not be received in the order in which they were generated. A solution to this is to use a scheme involving a combination of a sequence number and an estimated age for each LSP.

A sequence number is a counter. Each router keeps track of the sequence number it used the last time it generated an LSP and uses the next sequence number when it needs to generate a new LSP. When a router receives a LSP, it compares the sequence number of the received LSP with the one stored in memory (for that originating node) and only accepts the LSP if it has a higher sequence number. The higher the sequence number, the more recently generated.

However, a sequence number alone is not sufficient. The sequence number approach has various problems:

1. The sequence number field is of finite size. A problem arises when a node creates a LSP to case the field to reach the maximum value. Making the sequence number field wrap around is not a good idea because it causes ambiguity on the relation of the sequence numbers.
2. Sequence number on an LSP becomes corrupt. If the sequence field is corrupted to a very large sequence number, it will prevent valid, newer LSPs (with smaller sequence numbers) to be accepted.

3. Sequence number is reset. When a router goes down or forgets the sequence number it was using, newer LSPs cannot be distinguished from older LSPs

To solve the preceding problems, an age field is added to each LSP. It starts at some value and is decremented by routers as it is held in memory. When an LSP's age reaches 0, the LSP can be considered too old and an LSP with a nonzero age is accepted as new, regardless of its sequence number.

3.3.2 UPDATE INTERVAL [8]

The key difference between fisheye and traditional Link-state is the interval in which the routing information is disseminated. In Link State, the link state packets are generated and flooded into the network whenever a node detects topology changes. Fisheye uses a new approach to reduce the number of LSP messages. Kleinroch and Stevens proposed the fisheye technique to reduce the size of information required to represent graphical data. The original idea of fisheye was to maintain high resolution information within a range of a certain point of interest and lower resolution further away from the point of interest. For routing, this fisheye approach can be interpreted as maintaining a highly accurate network information about the immediate neighborhood of a node and becomes progressively less detailed as it moves away from the node.

The scope is defined in terms of the nodes that can be reached in a certain number of hops. The center node has most accurate information about all nodes in the first circle, and becomes less accurate with each outer circle. Even though a node does not have accurate information about distance nodes, the packets are routed correctly because the route information becomes more and more accurate as the packet moves closer to the destination.

The reduction of routing messages is achieved by updating the network information for nearby nodes at a higher frequency and remote nodes at a lower frequency. As a result, considerable amount of LSPs are suppressed. When a node receives a LSP, it calculates a time to wait before sending out the LSP from the following equation:

$$\mathbf{UpdateInterval} = \mathbf{ConstantTime} * \mathbf{hopcount}^{\propto}$$

ConstantTime is the user defined default refresh period to send out LSPs (in the first scope), hopcount is the number of hops the LSP has traversed, alpha is a parameter that determines how much effect each scope has on the UpdateInterval. Values for alpha are zero (same as no fisheye) and greater than or equal to one (fisheye). A maximum value of UpdateInterval is established to prevent an effective complete suppression of LSP messages (when calculated UpdateInterval is too large).

When a router accepts a LSP from a faraway node, and has not yet sent out the LSP in memory, the next time it will send out the LSP will be the minimum of the time left to wait in memory and the new calculated UpdateInterval based on the new LSP. This is to prevent a router from waiting indefinitely to send out a LSP when a new LSP arrives before the one in memory is sent out for that node.

$$\mathbf{UpdateInterval}(\mathbf{new}) = \mathbf{MIN}\big(\mathbf{UpdateInterval}(\mathbf{memory}), \mathbf{UpdateInterval}(\mathbf{LSP})\big)$$

3.4 ROUTE COMPUTATION [11]

Once the router has a database of LSPs, it computes the routes based on the Djikstra's algorithm which computes all shortest paths from a single vertex. The link metric used for path cost is the hop count. The algorithm uses 3 databases:

1. Link State Database- Contains the LSPs the node received.
2. PATH- contains ID, path cost, forwarding direction tuples. Holds the best path found.
3. TENT- contains ID, path cost, forwarding direction tuples. Holds possible best paths.

The Djikstra algorithm is as follows:

1. Start with “self” as the root of a tree by putting (myID, 0, 0) in PATH.
2. For node N just place in PATH, examine N’s LSP. For each of N’s neighbors, add the total path cost at N to the cost path of each neighbor. If the new total path of the node is better than the value for that node in PATH or TENT, put into TENT.
3. If TENT is empty, terminate the algorithm. Otherwise, find the minimal cost in TENT, move into PATH, and go to Step 2.

One the algorithm completes, PATH now contains the shortest next-hop information for each destination. The protocol can now use the PATH database as a routing table to forward packets toward their destinations.

Chapter 4

VIRTUAL GRID SYSTEM

4.1 GRID SYSTEM [4]

A hierarchical architecture is used in GFSR. A gateway is elected in each grid and is the only node in the grid to exchange control messages and data packets with other grids. Substantial bandwidth can be saved in this way. GFSR provides the advantage of less control message exchange and more bandwidth to transmit data.

The main idea of GRID is that a geographic area is partitioned into several logic grids and the gateway election is held in each grid. A number of mobile nodes may exist in each grid. However, when gateway election has finished, a single mobile node called gateway in each grid will take responsibility for route discovery and forwarding messages. Its advantage is that significantly less routing cost is incurred in GRID than in other reactive routing protocols. Only gateways will relay messages for the route discovery. If a gateway node crashes or leaves the network, a new gateway election will be held and a newly elected gateway will take over. Although GRID needs some messages to keep correctness of grid operations, it still brings the advantages of less control message overhead and more stable routing path.

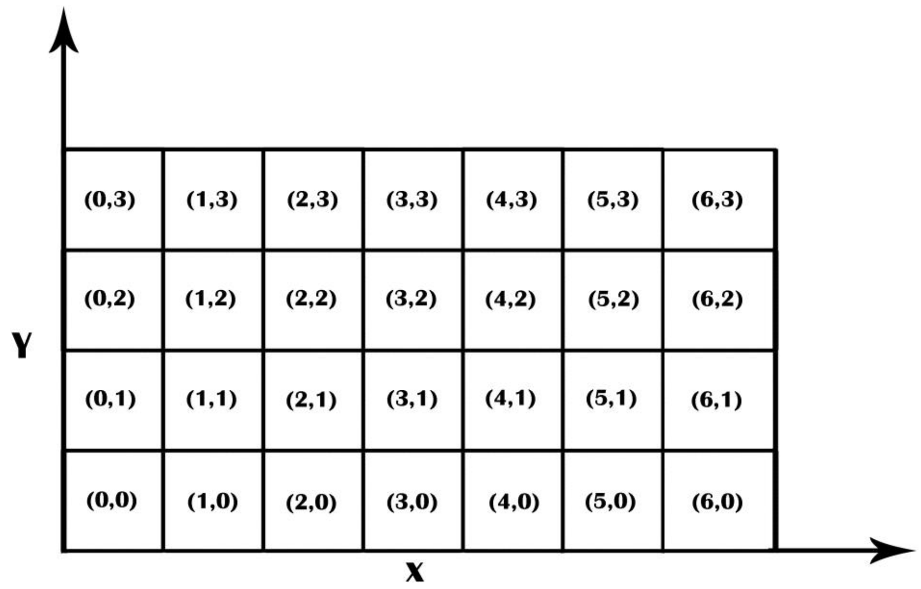


Figure 4.1 – Virtual Grid

4.2 LOCATION AWARENESS [4]

Through GPS devices, nodes can easily derive their location information. The geographic area of an ad hoc network is partitioned into two dimensional virtual grids and each grid has its unique coordinate number (x,y). As illustrated in Fig. 1, each node can calculate in which grid it currently dwells based on the physical location information derived from GPS.

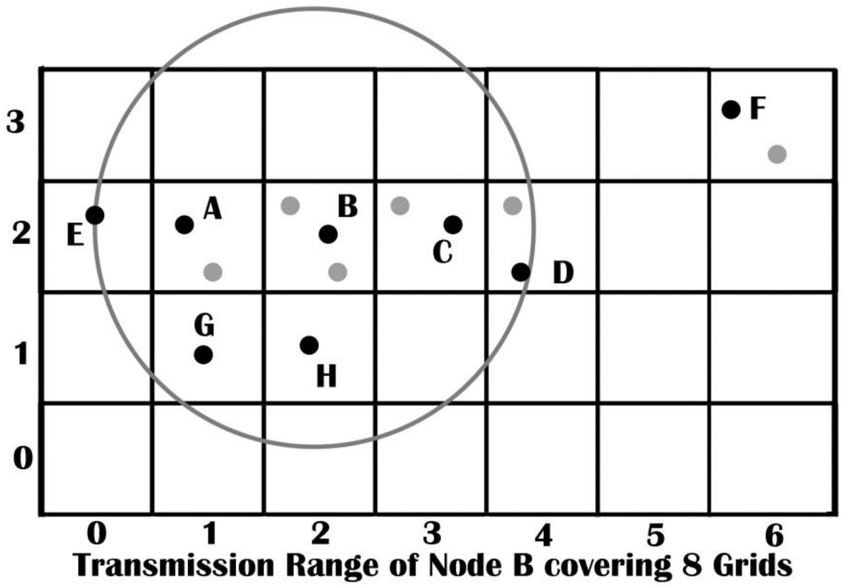
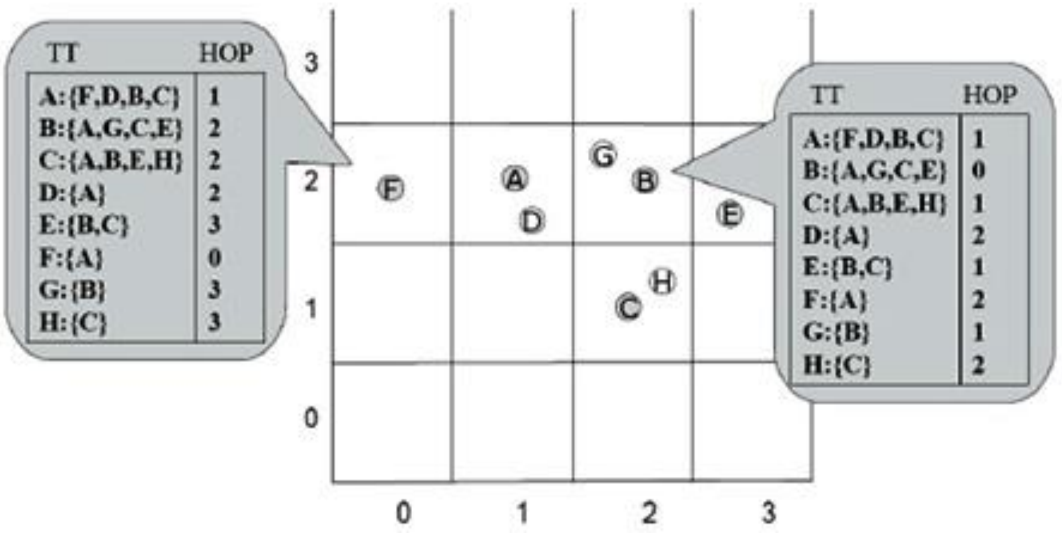


Figure 4.2 – Location Awareness

4.3 GATEWAY ELECTION [4]

Initially, a gateway election is held in each grid, where a gateway is the node responsible for maintaining the routing table in its grid, and for exchanging routing information using fisheye state routing scheme. The node nearest to the physical center of a grid is a good candidate as a gateway. The grid length in GRID architecture can be set in such a way that the transmission range of each gateway can effectively cover its eight neighbor grids. Therefore, any mobile node knows the gateways of its eight neighbor grids. As illustrated in Fig. 2, the black-dotted nodes denote gateways and the large gray circle represents the transmission range of node B.



TT	HOP
A:{F,D,B,C}	1
B:{A,G,C,E}	2
C:{A,B,E,H}	2
D:{A}	2
E:{B,C}	3
F:{A}	0
G:{B}	3
H:{C}	3

TT	HOP
A:{F,D,B,C}	1
B:{A,G,C,E}	0
C:{A,B,E,H}	1
D:{A}	2
E:{B,C}	1
F:{A}	2
G:{B}	1
H:{C}	2

Figure 4.3 – Gateway Election & Routing Table

Using the routing information which is exchanged periodically, each gateway maintains a routing table as in Fig. 3. When a node needs to transmit data, it checks whether it is a gateway or not. If it is a gateway, it transmits data to next hop by checking its routing table. If it is a non-gateway node, it just sends data to its gateway, and then the gateway will take over to forward the message to next hop. The gateways along the routing path will check the destination and determine who the next hop should be. When a packet approaches nearer to its destination, the routing information stored in the gateways becomes progressively more accurate. Packets finally arrive at the grid of destination node. If destination is a gateway, the packet is received by that gateway. Otherwise, the gateway forwards the packet to the destination node in its grid.

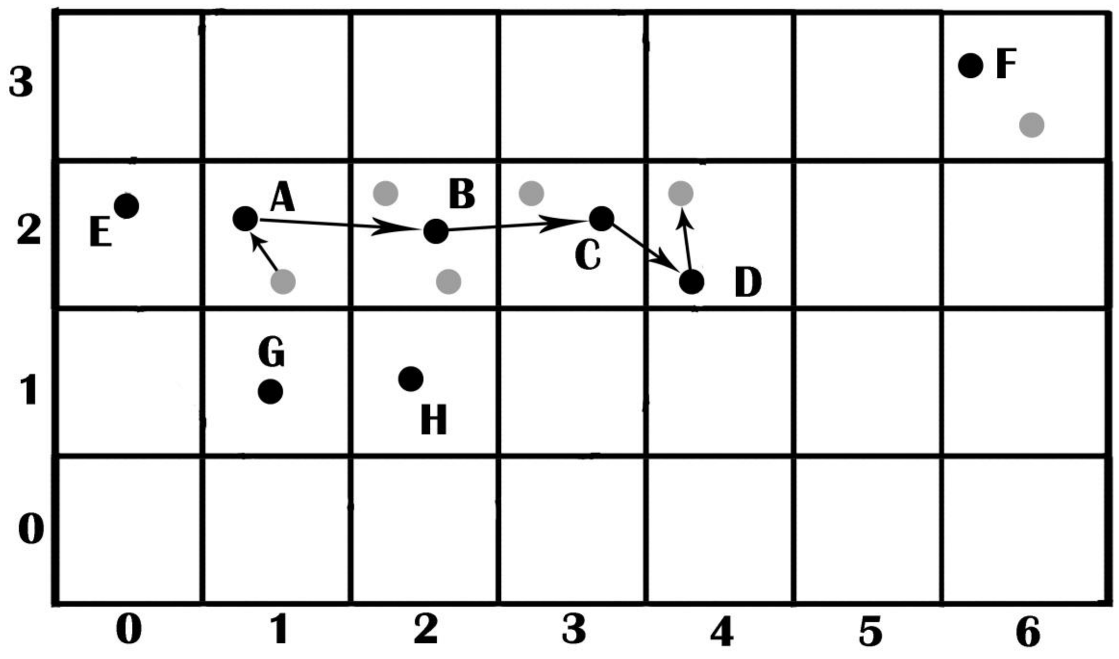


Figure 4.4 – Routing using Gateway Nodes

The gateway election can be performed using various techniques and algorithms. The most common gateway election algorithm used in mesh networks is LEACH, which uses the available node power to select a gateway. Other algorithms are based on existing routing algorithms, like AODV-UU. We have used the mobility of the nodes to select gateways. This principle states that the ideal gateway node is one which has the least mobility. This increases the possibility of it remaining in the same grid square for an extended period of time, thereby ensuring that the path will remain static for a greater period of time. Applied across all gateways, it is found that path stability is improved, causing a reduction in end to end latency and packet loss, and a net increase in throughput.

4.4 ALGORITHM

The algorithm will have to be heavily modified to implement the grid system. Instead of simply maintaining a list of first hop nodes for any given route, each node will have to maintain grid awareness. The selection of the node which will handle routing in each square will be done upon the basis of available bandwidth and latency. We shall research 2 forms of grid routing – the strong form and the weak form. Depending on feasibility and performance considerations, we shall choose one, or a hybrid, to implement.

4.4.1 STRONG FORM

- Case 1: When a node wishes to send a packet to a node within its own grid square, it shall do so directly.
- Case 2: When a node wishes to send a packet to a node outside its own grid square, it will always use the routing node of its own grid square.

The strong form forces all inter-grid routing to be performed through the routing nodes. The advantage of this form of grid routing is that the system architecture is greatly simplified, as only the routing nodes need to maintain complete grid awareness. The network is broken up into areas, similar to OSPF. Non fisheye capable nodes can be configured to connect to the network by using the routing node as an access point in infrastructure mode. Monitoring the network and measuring performance becomes easier. In case a node changes grid squares, it does not have to update its entire routing table, but only the routing node information of the current grid.

The disadvantages are that the system becomes static and rigid. If one routing node fails, all nodes within that grid square are cut off from the network until network convergence is complete. Also, since all routing is done via routing nodes exclusively, there is a high probability of congestion at these points, which may result in poor network performance and dropped packets.

This algorithm offers the best advantages as far as route stability is concerned. It manages to eliminate many of the problems inherent in most ad-hoc routing algorithms, and its disadvantages are not great enough to render it impractical. An especial advantage that it has is the fast convergence time. This is a necessity, given the random and fluid nature of Mobile Ad-Hoc Networks.

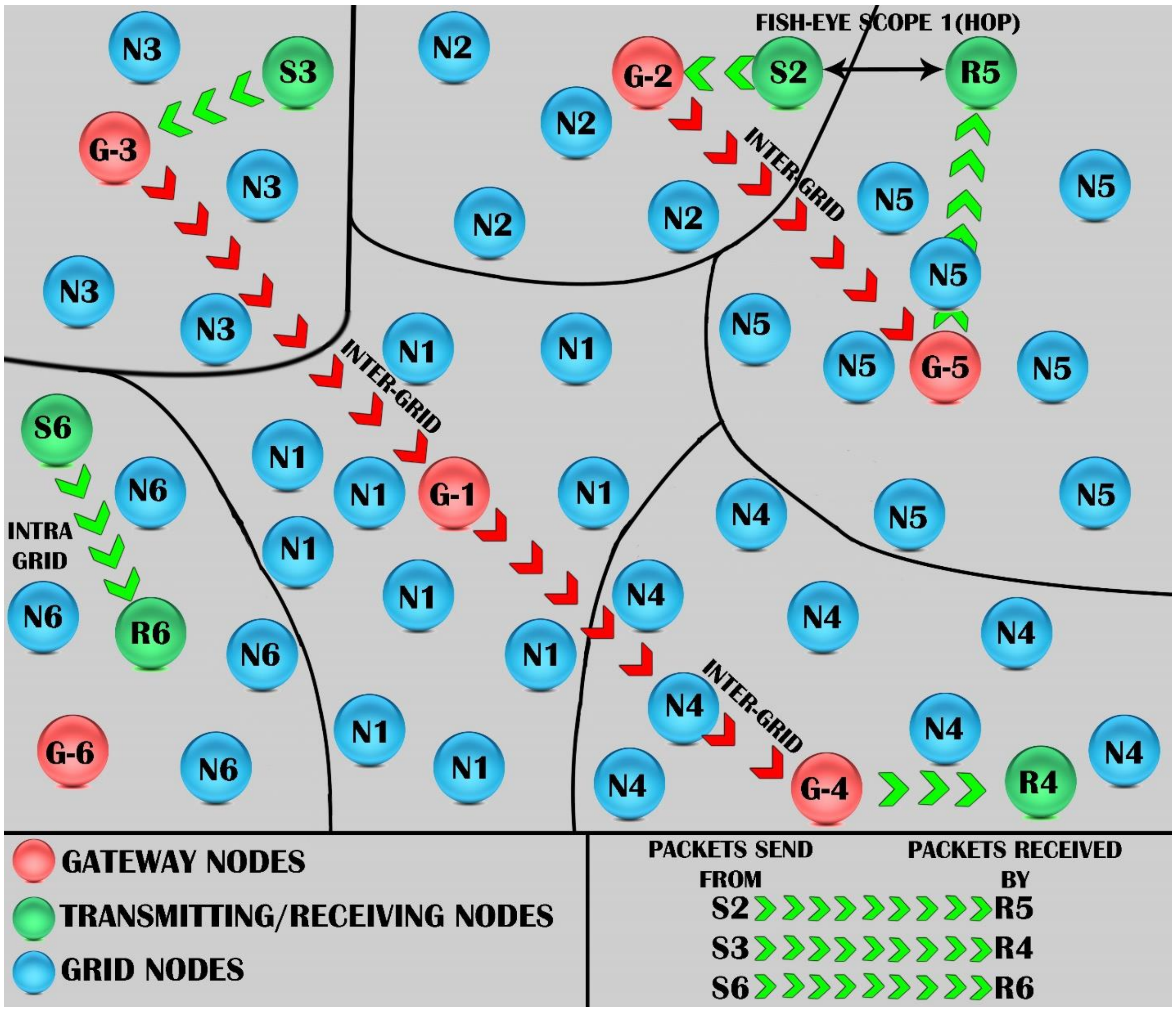


Figure 4.5 – Strong Form

4.4.2 WEAK FORM

- <u>Case 1:</u> When a node wishes to send a packet to a node within its own grid square, it shall do so directly.
- <u>Case 2:</u> When a node wishes to send a packet to a node outside its own grid square, but within its fisheye scope, it shall do so directly.
- <u>Case 3:</u> When a node wishes to send a packet to a node outside its own grid square, and outside its fisheye scope, it shall do so via the routing node of the first hop square.

- Case 4: When a node wishes to send a packet to a node outside its own grid square, and outside its fisheye scope, and the routing node is congested, it shall choose another node in the first hop grid square.

The weak form is much more permissive, only requiring nodes to use the routing nodes in case the target node is both outside the grid and outside its fisheye scope. The advantage of this is that the network becomes much more dynamic and flexible. The network is one single entity, with minimal segmentation and fragmentation. Convergence is faster, in case of node failure.

The disadvantage is that all nodes in the network must be location aware, and maintain complete routing information and a grid map. No fisheye incapable nodes can be part of this network. In case a node changes grid squares, it has to update its entire routing table. In case of network congestion, time and resources are wasted in choosing an alternate node through which to route packets.

While this algorithm manages to eliminate many of the problems inherent in most ad-hoc routing algorithms, its disadvantages are great enough to render it impractical. Since nodes are only forced to use gateway nodes in selected situations, routing paths remain relatively random. While this does reduce the possibility of congestion at any particular locality, it also induces a penalty as far as throughput and latency are concerned.

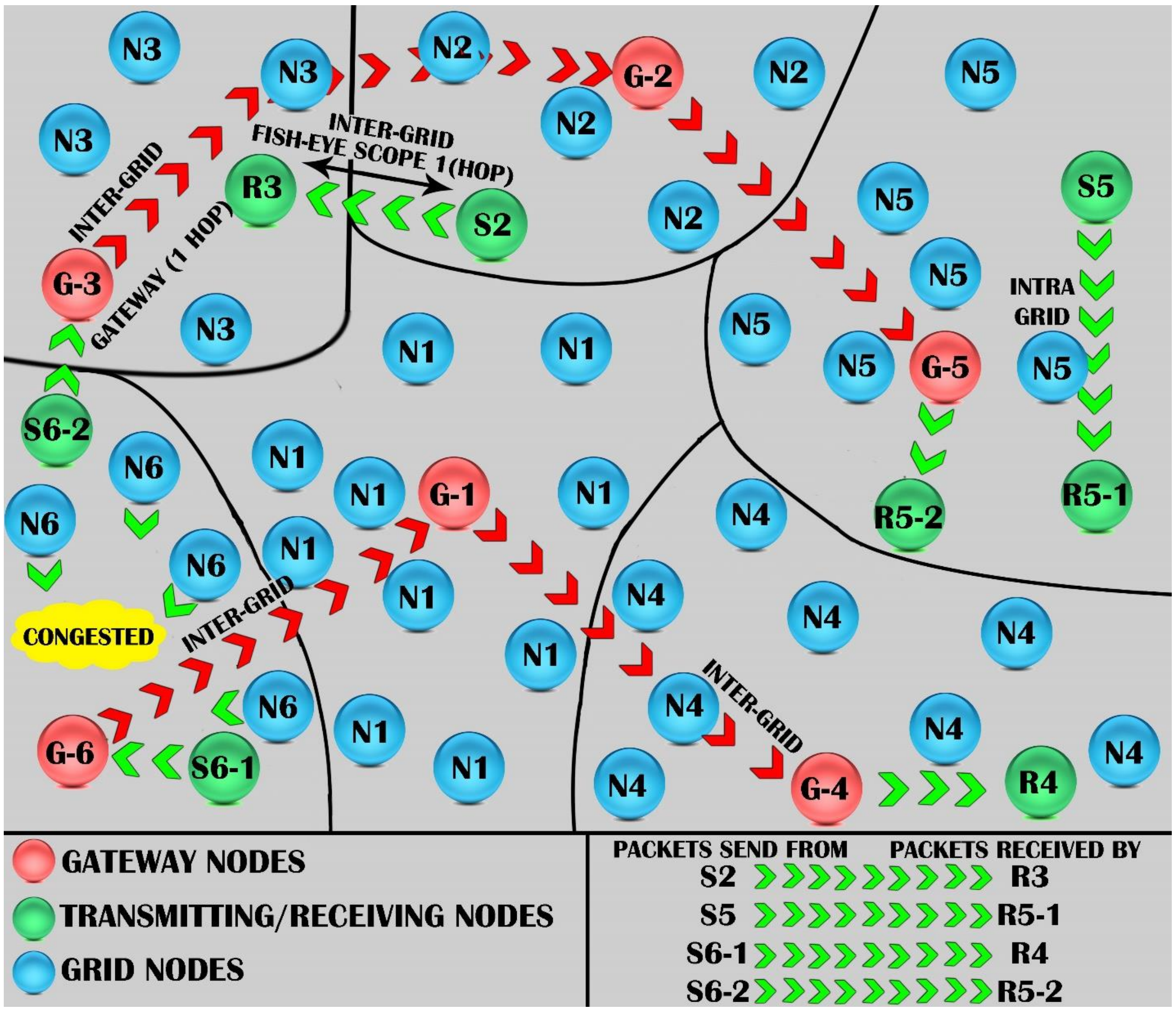


Figure 4.6 – Weak Form

Chapter 5

REQUIREMENT ANALYSIS

5.1 SOFTWARES

1. **NS2 -** The MANET upon which analysis is to be performed will be simulated using NS2. NS2 is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. It is written in C++ and an Object Oriented version of TCL called OTCL. NS2 scripts are also written in TCL. NS2 was chosen as the simulator for this project due to its powerful features, large library of extensions and tools, detailed documentation, and active community, which make it the industry standard for network simulation. The version of NS2 used is 2.35.

2. **TCL -** Tool Command Language is a scripting language used for rapid prototyping, scripted application, GUI testing, and embedded systems. TCL interfaces natively with the C language. This is because it was originally written to be a framework for providing a syntactic front-end to commands written in C, and all commands in the language (including things that might otherwise be keywords, such as if or while) are implemented this way. Each command implementation function is passed an array of values that describe the

(already substituted) arguments to the command, and is free to interpret those values as it sees fit. Environment Modules provides a set of extensions to the "standard" TCL package, which provide a rich environment for handling setting defaults and initializing into an environment. The version of TCL being used is 4.5.10.

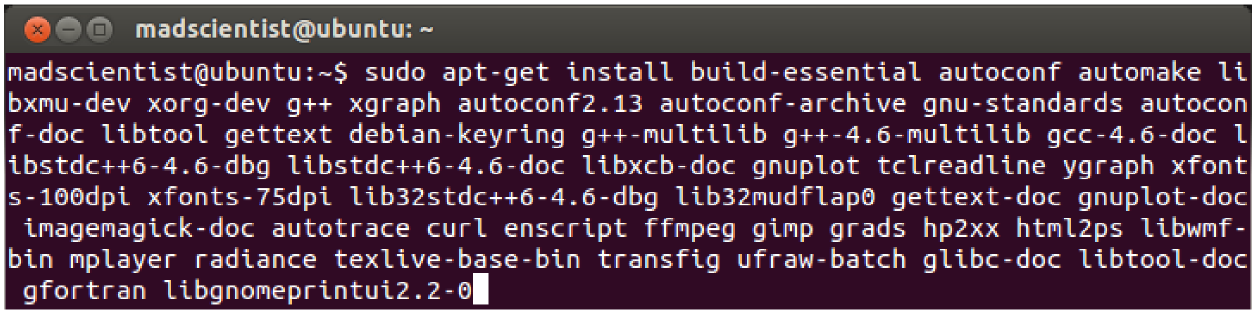
3. **NAM -** Nam is a TCL/TK based animation tool for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation, and various data inspection tools. It is mainly intended as a companion animator to the NS family of simulators. The version of NAM being used is 1.15.

4. **UBUNTU -** Ubuntu is a Debian based Linux operating system. After Android, it is the most popular Linux distribution in the world. It is extremely stable and light on resources, being user friendly, while still retaining features necessary for power users. Another advantage is the massive amount of free and open source software present in the repositories. All the applications needed for this project are present in the stable repositories, and can be easily installed and configured from terminal. The version of Ubuntu being used is 12.04.

5.2 PACKAGES AND DEPENDENCIES

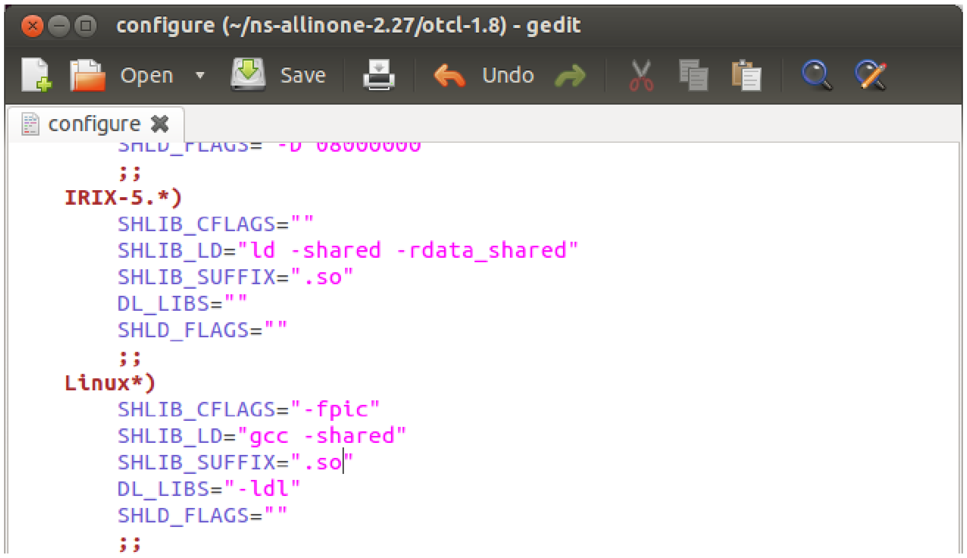
1	libx11-dev	X11 client-side library (development headers)
2	mesa-common-dev	Developer documentation for Mesa
3	libglu1-mesa-dev	Mesa OpenGL utility library
4	libxrandr-dev	X11 RandR extension library
5	libxi-dev	X11 Input extension library
6	xorg-dev	X.Org X Window System development libraries
7	gcc	GNU C compiler
8	g++	GNU C++ compiler
9	build-essential	Informational list of build-essential packages

10	autoconf	Automatic configure script builder
11	Automake	Tool for generating GNU Standards-compliant Makefiles
12	libxmu-dev	X11 miscellaneous utility library
13	tcl8.5-dev	Tcl (the Tool Command Language) v8.5
14	tk8.5-dev	Tk toolkit for Tcl and X11, v8.5
15	tclcl-dev	Transitional dummy package to libtclcl-dev
16	Nam	Network Animator for network simulation
17	Xgraph	Plotting program, reads stdin, allows interactive zooming
18	Perl	Larry Wall's Practical Extraction and Report Language
19	libxt-dev	X11 toolkit intrinsics library



```
madscientist@ubuntu:~$ sudo apt-get install build-essential autoconf automake li
bxmu-dev xorg-dev g++ xgraph autoconf2.13 autoconf-archive gnu-standards autocon
f-doc libtool gettext debian-keyring g++-multilib g++-4.6-multilib gcc-4.6-doc l
ibstdc++6-4.6-dbg libstdc++6-4.6-doc libxcb-doc gnuplot tclreadline ygraph xfont
s-100dpi xfonts-75dpi lib32stdc++6-4.6-dbg lib32mudflap0 gettext-doc gnuplot-doc
 imagemagick-doc autotrace curl enscript ffmpeg gimp grads hp2xx html2ps libwmf-
bin mplayer radiance texlive-base-bin transfig ufraw-batch glibc-doc libtool-doc
 gfortran libgnomeprintui2.2-0
```

Figure 5.1 – Package Installation



```
            SHLD_FLAGS="-D 08000000"
            ;;
    IRIX-5.*)
            SHLIB_CFLAGS=""
            SHLIB_LD="ld -shared -rdata_shared"
            SHLIB_SUFFIX=".so"
            DL_LIBS=""
            SHLD_FLAGS=""
            ;;
    Linux*)
            SHLIB_CFLAGS="-fpic"
            SHLIB_LD="gcc -shared"
            SHLIB_SUFFIX=".so"
            DL_LIBS="-ldl"
            SHLD_FLAGS=""
            ;;
```

Figure 5.2 – Compiler Configuration (flags -lGL -lGLU -lXrandr -lXext -lX11)

5.2 HARDWARE

An advantage of the softwares used for this project is that they are either supported on a wide range of hardware configurations, or they are platform independent. For the purpose of this project, the following machines have been chosen.

1. **Dell Inspiron 15R 5520** – Intel ™ Core i5 3210M CPU (2.5 – 3.1 GHz), 8 GB DDR3 RAM (1600 MHz), 500 GB HDD (5400 rpm), AMD Radeon 7670M GPU (1 GB GDDR3 VRAM), HM77 Chipset.
2. **Acer Aspire E-571G -** Intel ™ Core i3 2320M CPU (2.3 – 2.8 GHz), 4 GB DDR3 RAM (1333 MHz), 500 GB HDD (5400 rpm), NVidia GeForce GT 620M GPU (1 GB GDDR3 VRAM), HM77 Chipset.

Chapter 6

SIMULATION

6.1 20 - Node Network

Scenario for Gateway Election

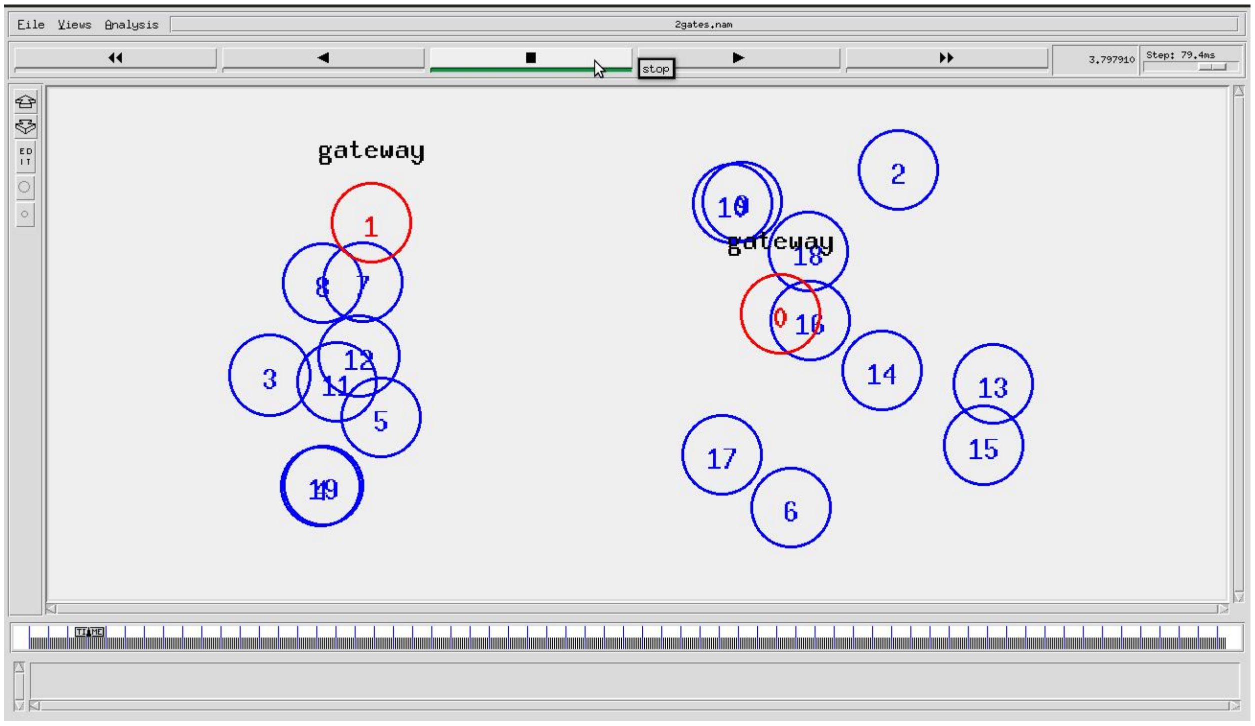


Figure 6.1 – Gateway Election and Grid Creation

Scenario for transmission between nodes in different grids.

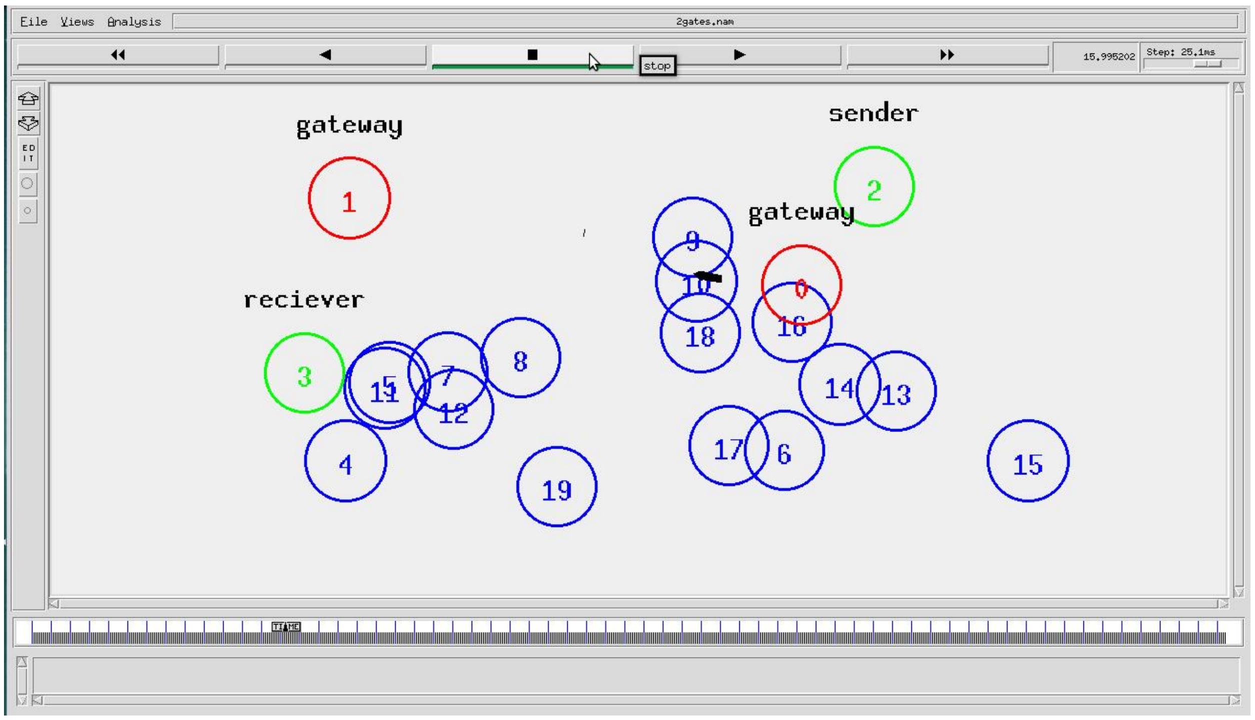


Figure 6.2 – Inter Grid Transmission

Scenario for transmission between nodes in same grid.

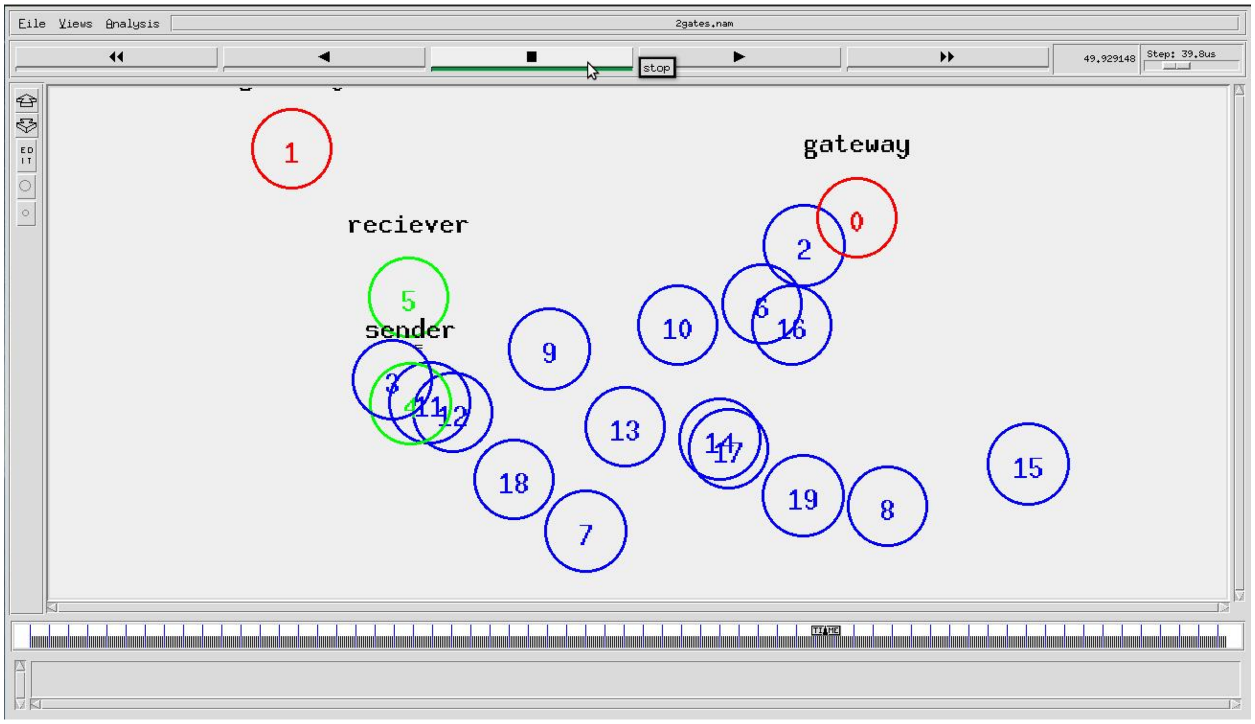


Figure 6.3 – Intra Grid Transmission

6.2 60 - Node Network

Scenario for Gateway Election

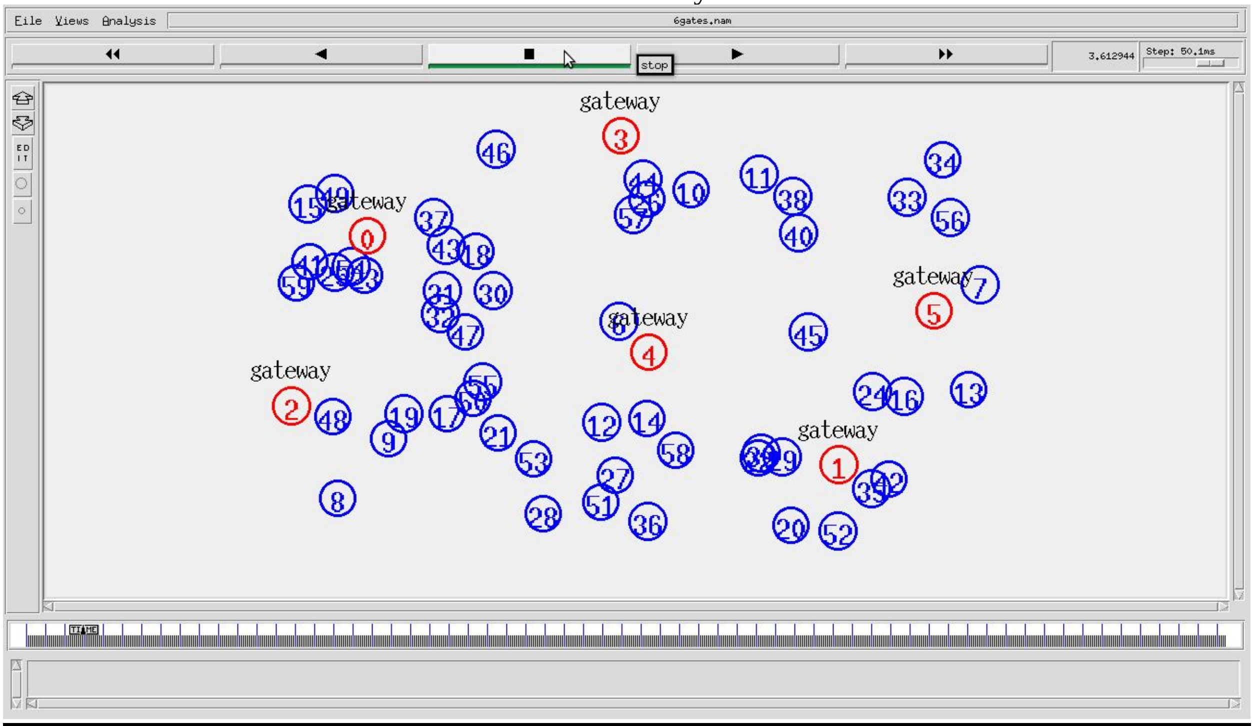


Figure 6.4 – Gateway Election and Grid Creation

Scenario for transmission between nodes in different grids.

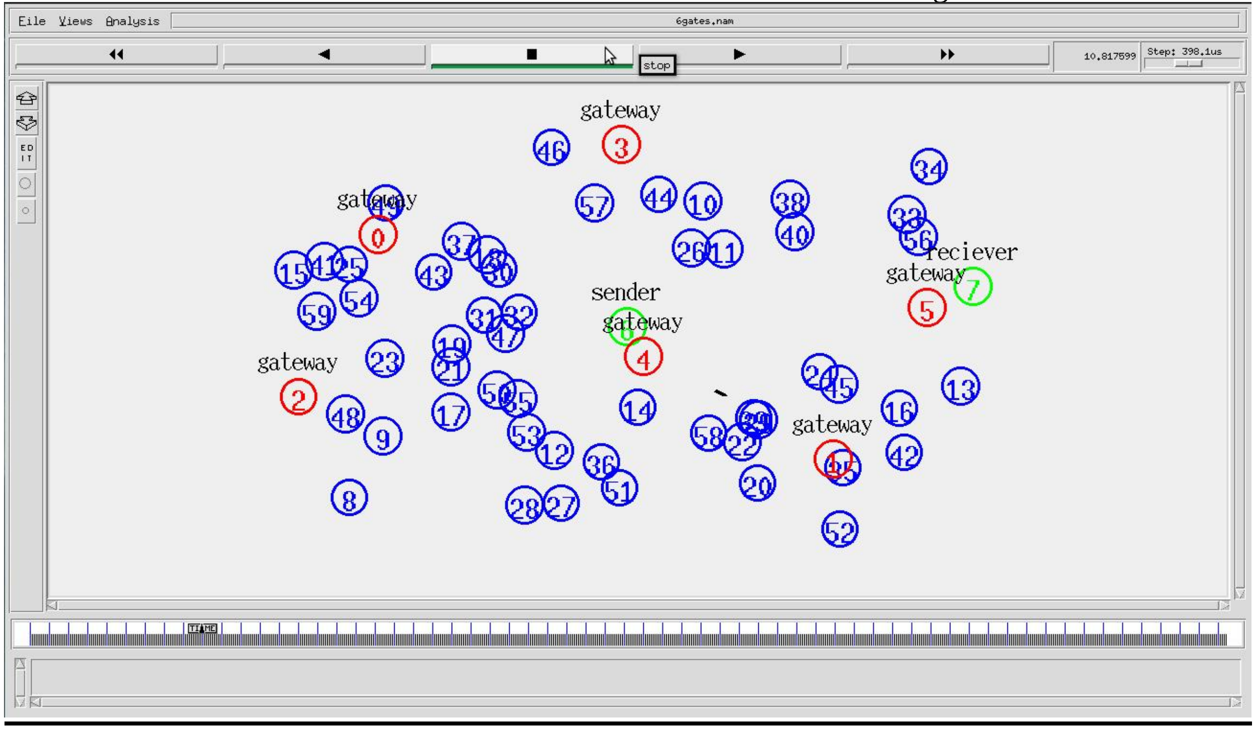


Figure 6.5 – Inter Grid Transmission

Scenario for transmission between nodes in different grids.

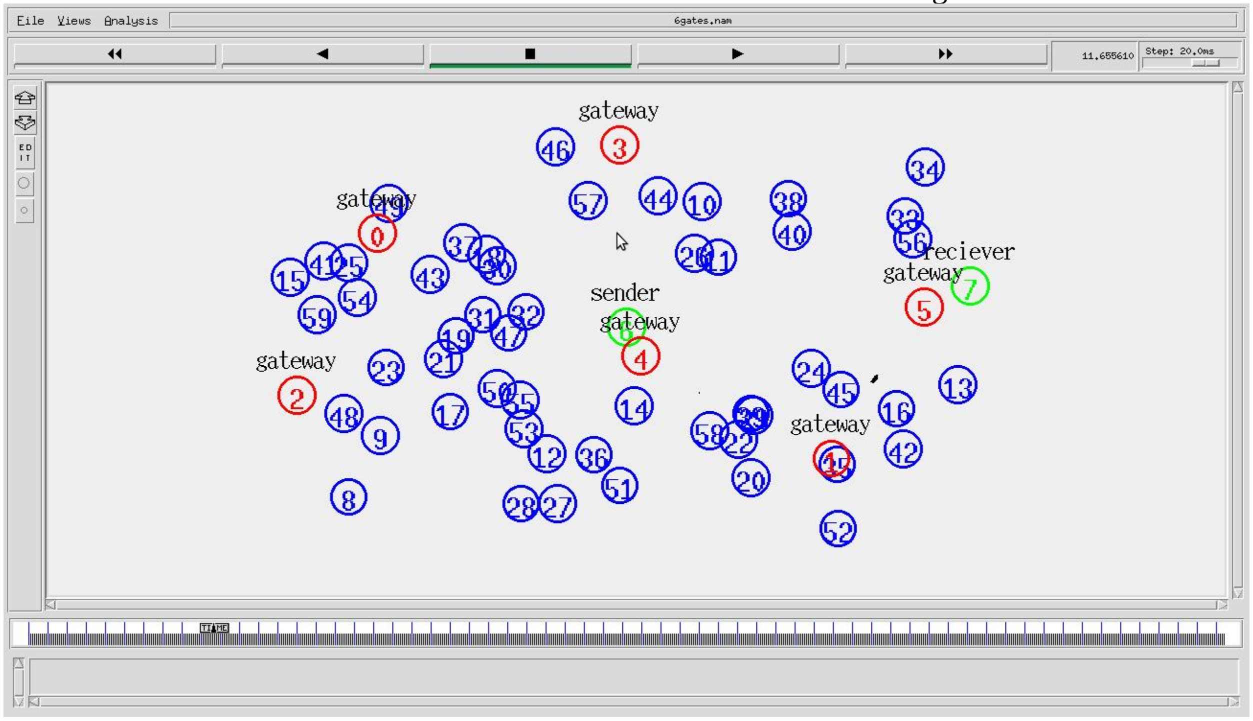


Figure 6.6 – Inter Grid Transmission

Scenario for transmission between nodes in same grid.

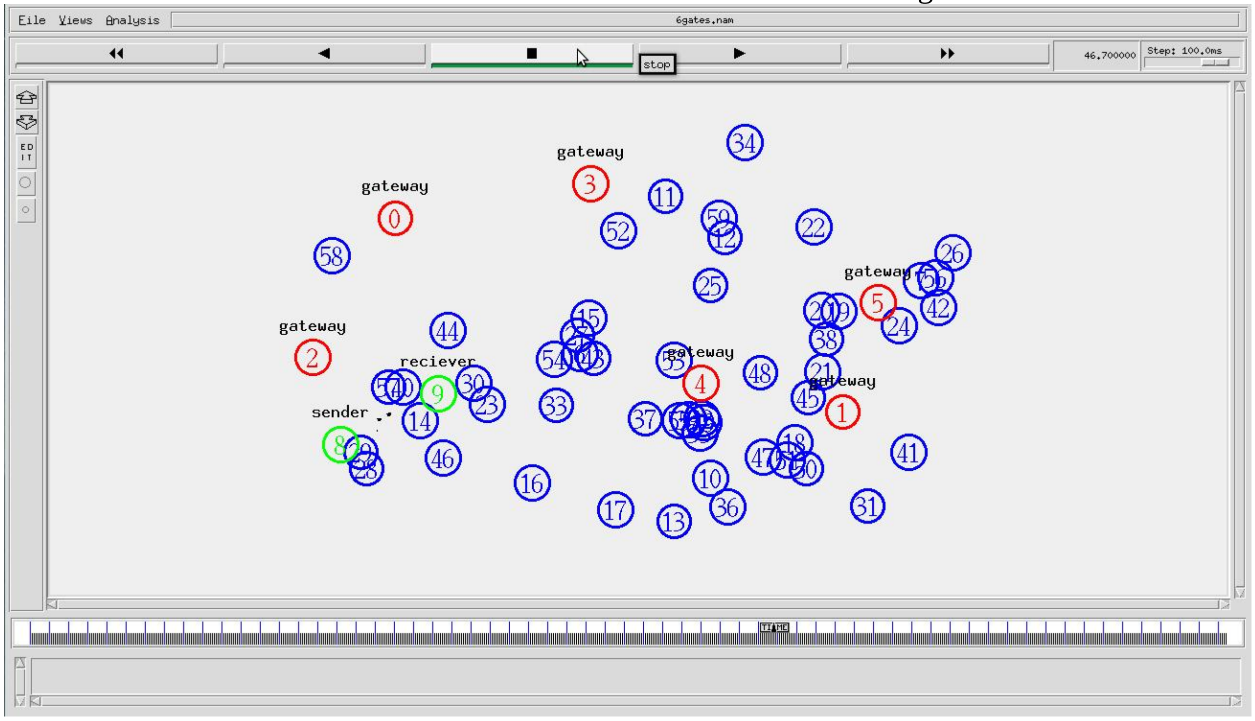


Figure 6.7 – Intra Grid Transmission

Chapter 7

CONCLUSION

Through the running of simulations, it can be seen that the strong form of the GRID algorithm offers significant advantages over classic ad-hoc routing algorithms. The reduction in end to end latency and packet loss, which result from the stabilization of routing paths, are the most significant advantages. This results in an improvement in net throughput across the network.

The main disadvantage of the system is the fact that a parallel algorithm and supporting architecture have to be set up alongside the main routing mechanism. This causes a processing overhead which can be an issue for less powerful nodes. The main consequences of this are increased power consumption, which is a definite problem in the typical low energy devices that are present in MANETs.

However, new wireless standards offering greater bandwidth, and high capacity power storage technologies, will alleviate the urgency of some of the imperatives contingent upon nodes in mobile environments. Additionally, the increased processing power available to modern chipsets and SOCs will reduce the effect of processing overhead penalties greatly.

The Virtual Grid System offers a method of building more realistic topology maps, which can reduce the effects of obstacles that block or reflect signals. Paired with the appropriate Ad-Hoc routing algorithm, it can reduce the time required to perform network convergence, in case of a drastic change in topology, or failure of critical nodes. Ultimately, more research into this area of networking is sure to yield promising new methods to leverage the fruits of the ongoing mobile revolution.

APPENDIX I

20 NODES - SPARSE TOPOLOGY

```
# Define options
        set val(chan) Channel/WirelessChannel  ;# channel type
        set val(prop) Propagation/TwoRayGround        ;# radio-propagation model
        set val(netif) Phy/WirelessPhy         ;# network interface type
        set val(mac) Mac/802_11                       ;# MAC type
        set val(ifq) Queue/DropTail/PriQueue   ;# interface queue type
        set val(ll) LL                                ;# link layer type
        set val(ant) Antenna/OmniAntenna              ;# antenna model
        set val(ifqlen) 50                            ;# max packet in ifq
        set val(nn) 20                                ;# number of mobilenodes
        set val(rp) AODV                              ;# routing protocol
        set val(x) 300                                ;# X dimension of topography
        set val(y) 160                                ;# Y dimension of topography
        set val(stop) 75                       ;# time of simulation end

#-------Event scheduler object creation--------#
        set ns          [new Simulator]

# Creating trace file and nam file
        set tracefd       [open 2gates.tr w]
        set windowVsTime2 [open win.tr w]
        set namtrace      [open 2gates.nam w]
        $ns trace-all $tracefd
        $ns namtrace-all-wireless $namtrace $val(x) $val(y)

# Set up topography object
        set topo       [new Topography]
        $topo load_flatgrid $val(x) $val(y)
        create-god $val(nn)

# Configure gateway node
        $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 $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON
        for {set i 0} {$i < 2 } { incr i } {
        set node_($i) [$ns node];
        $node_($i) color "red";
        $ns at 3.0 "$node_($i) color red"               }

        for {set i 0} {$i < 2 } { incr i } {
        $ns at 3.0 "$node_($i) label gateway"     }

        for {set i 2} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node];
        $node_($i) color "blue";
        $ns at 3.0 "$node_($i) color blue"           }

#Change node color based on role
        $node_(4) color green
        $ns at 7.0 "$node_(2) color green"
        $ns at 36.0 "$node_(2) color blue"

        $node_(5) color green
        $ns at 8.0 "$node_(3) color green"
        $ns at 39.0 "$node_(3) color blue"

        $node_(6) color green
        $ns at 44.0 "$node_(4) color green"
        $ns at 65.0 "$node_(4) color blue"

        $node_(9) color green
        $ns at 45.0 "$node_(5) color green"
        $ns at 66.0 "$node_(5) color blue"

#Change node label based on role
        $ns at 7.0 "$node_(2) label sender"
        $ns at 36.0 "$node_(2) label \"\""
        $ns at 8.0 "$node_(3) label reciever"
        $ns at 39.0 "$node_(3) label \"\""

        $ns at 44.0 "$node_(4) label sender"
        $ns at 65.0 "$node_(4) label \"\""

        $ns at 45.0 "$node_(5) label reciever"
        $ns at 66.0 "$node_(5) label \"\""

# Provide location of gateways
        $node_(0) set X_ 200.0
        $node_(0) set Y_ 90.0
        $node_(0) set Z_ 0.0
        $node_(1) set X_ 50.0
        $node_(1) set Y_ 125.0
        $node_(1) set Z_ 0.0
        $node_(2) set X_ 250.0
        $node_(2) set Y_ 150.0
        $node_(2) set Z_ 0.0
        $node_(3) set X_ 5.0
        $node_(3) set Y_ 70.0
        $node_(3) set Z_ 0.0
        $node_(4) set X_ 25.0
        $node_(4) set Y_ 25.0
        $node_(4) set Z_ 0.0
        $node_(5) set X_ 50.0
        $node_(5) set Y_ 50.0
        $node_(5) set Z_ 0.0

# Provide random initial position of mobilenodes
        for {set i 6} {$i < $val(nn)} {incr i} {
        $node_($i) set X_ [expr rand()*300]
        $node_($i) set Y_ [expr rand()*160]
        $node_($i) set Z_ 0                              }

# Generation of random movement and speed
        for {set i 0} {$i < 6} {incr i} {
        $ns at 0.0 "$node_($i) setdest [expr rand()*300] [expr rand()*160] 1.0"    }

        for {set i 6} {$i < $val(nn)} {incr i} {
        $ns at 0.0 "$node_($i) setdest [expr rand()*300] [expr rand()*160] [expr rand()*10]"        }

# Set up TCP connections
        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(2) $tcp
        $ns attach-agent $node_(0) $sink
        $ns connect $tcp $sink
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ns at 9.0 "$ftp start"
        $ns at 35.0 "$ftp stop"

        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(0) $tcp
        $ns attach-agent $node_(1) $sink
        $ns connect $tcp $sink
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ns at 10.0 "$ftp start"
        $ns at 36.0 "$ftp stop"

        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(1) $tcp
```

```
          $ns attach-agent $node_(3) $sink
          $ns connect $tcp $sink
          set ftp [new Application/FTP]
          $ftp attach-agent $tcp
          $ns at 11.0 "$ftp start"
          $ns at 37.0 "$ftp stop"

          set tcp [new Agent/TCP/Newreno]
          $tcp set class_ 2
          set sink [new Agent/TCPSink]
          $ns attach-agent $node_(4) $tcp
          $ns attach-agent $node_(5) $sink
          $ns connect $tcp $sink
          set ftp [new Application/FTP]
          $ftp attach-agent $tcp
          $ns at 45.0 "$ftp start"
          $ns at 65.0 "$ftp stop"

# Printing the window size
          proc plotWindow {tcpSource file} {
          global ns
          set time 0.01
          set now [$ns now]
          set cwnd [$tcpSource set cwnd_]
          puts $file "$now $cwnd"
          $ns at [expr $now+$time] "plotWindow $tcpSource $file"          }
          $ns at 10.0 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam
          for {set i 0} {$i < $val(nn)} { incr i } {
          $ns initial_node_pos $node_($i) 30        }

# Telling nodes when the simulation ends

          for {set i 0} {$i < $val(nn) } { incr i } {
          $ns at $val(stop) "$node_($i) reset";       }

# Ending nam and the simulation
          $ns at $val(stop) "$ns nam-end-wireless $val(stop)"
          $ns at $val(stop) "stop"
          $ns at 150.01 "puts \"end simulation\" ; $ns halt"
          proc stop {} {
                    global ns tracefd namtrace
                    $ns flush-trace
                    close $tracefd
                    close $namtrace
                    exec nam 2gates.nam & }
          $ns run
```

APPENDIX II

40 NODES - MEDIUM TOPOLOGY

```
# Define options
        set val(chan) Channel/WirelessChannel  ;# channel type
        set val(prop) Propagation/TwoRayGround         ;# radio-propagation model
        set val(netif) Phy/WirelessPhy                 ;# network interface type
        set val(mac) Mac/802_11                        ;# MAC type
        set val(ifq) Queue/DropTail/PriQueue    ;# interface queue type
        set val(ll) LL                                 ;# link layer type
        set val(ant) Antenna/OmniAntenna               ;# antenna model
        set val(ifqlen) 50                             ;# max packet in ifq
        set val(nn) 40                                 ;# number of mobilenodes
        set val(rp) AODV                               ;# routing protocol
        set val(x) 500                                 ;# X dimension of topography
        set val(y) 275                                 ;# Y dimension of topography
        set val(stop) 75                       ;# time of simulation end

#-------Event scheduler object creation--------#
        set ns          [new Simulator]

# Creating trace file and nam file
        set tracefd      [open 4gates.tr w]
        set windowVsTime2 [open win.tr w]
        set namtrace      [open 4gates.nam w]
        $ns trace-all $tracefd
        $ns namtrace-all-wireless $namtrace $val(x) $val(y)

# Set up topography object
        set topo      [new Topography]
        $topo load_flatgrid $val(x) $val(y)
        create-god $val(nn)

# Configure gateway node
            $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 $topo \
                    -agentTrace ON \
                    -routerTrace ON \
                    -macTrace OFF \
                    -movementTrace ON
                    for {set i 0} {$i < 4 } { incr i } {
                    set node_($i) [$ns node];
                    $node_($i) color "red";
                    $ns at 3.0 "$node_($i) color red"          }
                    for {set i 0} {$i < 4 } { incr i } {
                    $ns at 3.0 "$node_($i) label gateway"     }
                    for {set i 4} {$i < $val(nn) } { incr i } {
                    set node_($i) [$ns node];
                    $node_($i) color "blue";
                    $ns at 3.0 "$node_($i) color blue"         }

#Change node color based on role
          $node_(4) color green
          $ns at 7.0 "$node_(4) color green"
          $ns at 36.0 "$node_(4) color blue"

          $node_(5) color green
          $ns at 8.0 "$node_(5) color green"
          $ns at 39.0 "$node_(5) color blue"

          $node_(6) color green
          $ns at 44.0 "$node_(6) color green"
          $ns at 65.0 "$node_(6) color blue"

          $node_(9) color green
          $ns at 45.0 "$node_(7) color green"
          $ns at 66.0 "$node_(7) color blue"


# Change node label based on role
          $ns at 7.0 "$node_(4) label sender"
          $ns at 36.0 "$node_(4) label \"\""
          $ns at 8.0 "$node_(5) label reciever"
          $ns at 39.0 "$node_(5) label \"\""
          $ns at 44.0 "$node_(6) label sender"
          $ns at 65.0 "$node_(6) label \"\""
          $ns at 45.0 "$node_(7) label reciever"
          $ns at 66.0 "$node_(7) label \"\""

# Provide location of gateways
          $node_(0) set X_ 340.0
          $node_(0) set Y_ 90.0
          $node_(0) set Z_ 0.0
          $node_(1) set X_ 50.0
          $node_(1) set Y_ 125.0
          $node_(1) set Z_ 0.0
          $node_(2) set X_ 200.0
          $node_(2) set Y_ 175.0
          $node_(2) set Z_ 0.0
          $node_(3) set X_ 425.0
          $node_(3) set Y_ 180.0
          $node_(3) set Z_ 0.0
          $node_(4) set X_ 5.0
          $node_(4) set Y_ 70.0
          $node_(4) set Z_ 0.0
          $node_(5) set X_ 480.0
          $node_(5) set Y_ 205.0
          $node_(5) set Z_ 0.0
          $node_(6) set X_ 50.0
          $node_(6) set Y_ 50.0
          $node_(6) set Z_ 0.0
          $node_(7) set X_ 100.0
          $node_(7) set Y_ 100.0
          $node_(7) set Z_ 0.0

# Provide random initial position of mobilenodes
          for {set i 8} {$i < $val(nn)} {incr i} {
                    $node_($i) set X_ [expr rand()*500]
                    $node_($i) set Y_ [expr rand()*275]
                    $node_($i) set Z_ 0                          }

# Generation of random movement and speed

          for {set i 0} {$i < 8} {incr i} {
          $ns at 0.0 "$node_($i) setdest [expr rand()*500] [expr rand()*275] 1.0"    }

          for {set i 8} {$i < $val(nn)} {incr i} {
          $ns at 0.0 "$node_($i) setdest [expr rand()*500] [expr rand()*275] [expr rand()*10]"}
# Set up TCP connections
          set tcp [new Agent/TCP/Newreno]
          $tcp set class_ 2
          set sink [new Agent/TCPSink]
          $ns attach-agent $node_(4) $tcp
          $ns attach-agent $node_(1) $sink
          $ns connect $tcp $sink
          set ftp [new Application/FTP]
          $ftp attach-agent $tcp
          $ns at 9.0 "$ftp start"
          $ns at 35.0 "$ftp stop"

          set tcp [new Agent/TCP/Newreno]
          $tcp set class_ 2
          set sink [new Agent/TCPSink]
          $ns attach-agent $node_(1) $tcp
          $ns attach-agent $node_(2) $sink
          $ns connect $tcp $sink
          set ftp [new Application/FTP]
          $ftp attach-agent $tcp
          $ns at 10.0 "$ftp start"
          $ns at 36.0 "$ftp stop"
```

```
        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(2) $tcp
        $ns attach-agent $node_(0) $sink
        $ns connect $tcp $sink
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ns at 11.0 "$ftp start"
        $ns at 37.0 "$ftp stop"

        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(0) $tcp
        $ns attach-agent $node_(3) $sink
        $ns connect $tcp $sink
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ns at 12.0 "$ftp start"
        $ns at 38.0 "$ftp stop"

        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(3) $tcp
        $ns attach-agent $node_(5) $sink
        $ns connect $tcp $sink
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ns at 13.0 "$ftp start"
        $ns at 39.0 "$ftp stop"

        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(6) $tcp
        $ns attach-agent $node_(7) $sink
        $ns connect $tcp $sink
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ns at 45.0 "$ftp start"
        $ns at 65.0 "$ftp stop"

# Printing the window size
        proc plotWindow {tcpSource file} {
        global ns
        set time 0.01
        set now [$ns now]
        set cwnd [$tcpSource set cwnd_]
        puts $file "$now $cwnd"
        $ns at [expr $now+$time] "plotWindow $tcpSource $file"          }
        $ns at 10.0 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam
        for {set i 0} {$i < $val(nn)} { incr i } {
        $ns initial_node_pos $node_($i) 30        }

# Telling nodes when the simulation ends
        for {set i 0} {$i < $val(nn) } { incr i } {
        $ns at $val(stop) "$node_($i) reset";       }

# Ending nam and the simulation
        $ns at $val(stop) "$ns nam-end-wireless $val(stop)"
        $ns at $val(stop) "stop"
        $ns at 150.01 "puts \"end simulation\" ; $ns halt"
        proc stop {} {
                global ns tracefd namtrace
                $ns flush-trace
                close $tracefd
                close $namtrace
                exec nam 4gates.nam & }
        $ns run
```

APPENDIX III

60 NODES - DENSE TOPOLOGY

```
# Define options
        set val(chan) Channel/WirelessChannel      ;# channel type
        set val(prop) Propagation/TwoRayGround     ;# radio-propagation model
        set val(netif) Phy/WirelessPhy             ;# network interface type
        set val(mac) Mac/802_11                    ;# MAC type
        set val(ifq) Queue/DropTail/PriQueue       ;# interface queue type
        set val(ll) LL                             ;# link layer type
        set val(ant) Antenna/OmniAntenna           ;# antenna model
        set val(ifqlen) 50                         ;# max packet in ifq
        set val(nn) 60                             ;# number of mobilenodes
        set val(rp) AODV                           ;# routing protocol
        set val(x) 600                             ;# X dimension of topography
        set val(y) 375                             ;# Y dimension of topography
        set val(stop) 75                           ;# time of simulation end

#-------Event scheduler object creation--------#
        set ns          [new Simulator]

# Creating trace file and nam file
        set tracefd     [open 6gates.tr w]
        set windowVsTime2 [open win.tr w]
        set namtrace    [open 6gates.nam w]
        $ns trace-all $tracefd
        $ns namtrace-all-wireless $namtrace $val(x) $val(y)

# Set up topography object
        set topo    [new Topography]
        $topo load_flatgrid $val(x) $val(y)
        create-god $val(nn)

# Configure gateway node
        $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

        for {set i 0} {$i < 6 } { incr i } {
        set node_($i) [$ns node];
        $node_($i) color "red";
        $ns at 3.0 "$node_($i) color red"       }

        for {set i 0} {$i < 6 } { incr i } {
        $ns at 3.0 "$node_($i) label gateway"           }

        for {set i 6} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node];
        $node_($i) color "blue";
        $ns at 3.0 "$node_($i) color blue"      }

# Change node color based on role
        $node_(6) color green
        $ns at 7.0 "$node_(6) color green"
        $ns at 36.0 "$node_(6) color blue"
        $node_(7) color green
        $ns at 7.0 "$node_(7) color green"
        $ns at 39.0 "$node_(7) color blue"
        $node_(8) color green
        $ns at 44.0 "$node_(8) color green"
        $ns at 65.0 "$node_(8) color blue"
        $node_(9) color green
        $ns at 45.0 "$node_(9) color green"
        $ns at 66.0 "$node_(9) color blue"

# Change node label based on role
        $ns at 7.0 "$node_(6) label sender"
        $ns at 36.0 "$node_(6) label \"\""
        $ns at 7.0 "$node_(7) label reciever"
        $ns at 39.0 "$node_(7) label \"\""
        $ns at 44.0 "$node_(8) label sender"
        $ns at 65.0 "$node_(8) label \"\""
        $ns at 45.0 "$node_(9) label reciever"
        $ns at 66.0 "$node_(9) label \"\""

# Provide location of gateways
        $node_(0) set X_ 75.0
        $node_(0) set Y_ 270.0
        $node_(0) set Z_ 0.0
        $node_(1) set X_ 478.0
        $node_(1) set Y_ 77.0
        $node_(1) set Z_ 0.0
        $node_(2) set X_ 14.0
        $node_(2) set Y_ 125.0
        $node_(2) set Z_ 0.0
        $node_(3) set X_ 292.0
        $node_(3) set Y_ 358.0
        $node_(3) set Z_ 0.0
        $node_(4) set X_ 318.0
        $node_(4) set Y_ 175.0
        $node_(4) set Z_ 0.0
        $node_(5) set X_ 558.0
        $node_(5) set Y_ 207.0
        $node_(5) set Z_ 0.0
        $node_(6) set X_ 288.0
        $node_(6) set Y_ 202.0
        $node_(6) set Z_ 0.0
        $node_(7) set X_ 597.0
        $node_(7) set Y_ 230.0
        $node_(7) set Z_ 0.0
        $node_(8) set X_ 50.0
        $node_(8) set Y_ 50.0
        $node_(8) set Z_ 0.0
        $node_(9) set X_ 100.0
        $node_(9) set Y_ 100.0
        $node_(9) set Z_ 0.0

# Provide random initial position of mobilenodes
        for {set i 10} {$i < $val(nn)} {incr i} {
        $node_($i) set X_ [expr rand()*600]
        $node_($i) set Y_ [expr rand()*375]
        $node_($i) set Z_ 0                              }

# Generation of random movement and speed
        for {set i 0} {$i < 10} {incr i} {
        $ns at 0.0 "$node_($i) setdest [expr rand()*600] [expr rand()*375] 1.0"      }

        for {set i 10} {$i < $val(nn)} {incr i} {
        $ns at 0.0 "$node_($i) setdest [expr rand()*600] [expr rand()*375] [expr rand()*10]"}

# Set up TCP connections
        set tcp [new Agent/TCP/Newreno]
        $tcp set class_ 2
        set sink [new Agent/TCPSink]
        $ns attach-agent $node_(6) $tcp
        $ns attach-agent $node_(4) $sink
```

```
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 9.0 "$ftp start"
$ns at 35.0 "$ftp stop"

set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(4) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
$ns at 36.0 "$ftp stop"

set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(5) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 11.0 "$ftp start"
$ns at 37.0 "$ftp stop"

set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(5) $tcp
$ns attach-agent $node_(7) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 15.0 "$ftp start"
$ns at 38.0 "$ftp stop"

set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(8) $tcp
$ns attach-agent $node_(9) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
```

```
        $ftp attach-agent $tcp
        $ns at 45.0 "$ftp start"
        $ns at 65.0 "$ftp stop"

# Printing the window size
        proc plotWindow {tcpSource file} {
        global ns
        set time 0.01
        set now [$ns now]
        set cwnd [$tcpSource set cwnd_]
        puts $file "$now $cwnd"
        $ns at [expr $now+$time] "plotWindow $tcpSource $file"   }
        $ns at 10.0 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam
        for {set i 0} {$i < $val(nn)} { incr i } {
        $ns initial_node_pos $node_($i) 30   }

# Telling nodes when the simulation ends
        for {set i 0} {$i < $val(nn) } { incr i } {
        $ns at $val(stop) "$node_($i) reset";  }

# Ending nam and the simulation
        $ns at $val(stop) "$ns nam-end-wireless $val(stop)"
        $ns at $val(stop) "stop"
        $ns at 150.01 "puts \"end simulation\" ; $ns halt"
        proc stop {} {
                global ns tracefd namtrace
                $ns flush-trace
                close $tracefd
                close $namtrace
                exec nam 6gates.nam &         }
        $ns run
```

REFERENCES

[1] S. Sathish, K. Thangavel, S. Boopathi, Department of Computer Science, Periyar University - ***Performance Analysis of DSR, AODV, FSR and ZRP Routing Protocols in MANET***, in MES Journal of Technology and Management.

[2] S. Nithya Rekha, Dr. C.Chandrasekar, ***A Reduced Flooding Algorithm and Comparative Study of Grid Fisheye State Routing Protocol for MANET***, in International Journal of Scientific & Engineering Research, Volume 3, Issue 4, April-2012.

[3] Katarina Persson, Erika Johansson, Ulf Sterner, Mattias Skold, ***The Fisheye Routing Technique in Highly Mobile Ad Hoc Networks***, in Swedish Defence Research Agency Methodology Report FOI-R-1058-SE, December 2003.

[4] Ting-Hung Chiu, Shyh-In Hwang, Dept. of Computer Science and Engineering, Yuan Ze University, ***Efficient Fisheye State Routing Protocol using Virtual Grid in High-Density Ad-Hoc Networks.***

[5] Ahmed A. Radwan, Tarek M. Mahmoud, Essam H. Houssein, Computer Science Department, Faculty of Science, Minia University El Minia, Egypt, ***Performance Measurement of Some Mobile Ad Hoc Network Routing Protocols,*** in IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011.

[6] A. Samuel Chellathurai, E.George Dharma Prakash Raj, James College of Engineering and Technology, Nagercoil, ***EFSR: Evolutionary Fisheye State Routing Protocol in Mobile Ad Hoc Networks,*** in International Journal of Computer Trends and Technology (IJCTT) – volume 14 number 2 – Aug 2014.

[7] Guangyu Pei, Mario Gerla, Tsu-Wei Vhen Computer Science Department, UCLA, Los Angeles, California, ***Fisheye State Routing, A Routing Scheme for Ad hoc Wireless Networks.***

[8] Allen C. Sun, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 14, 2000, ***Design and Implementation of a Fisheye Routing Protocol For Mobile Wireless Ad Hoc Networks.***

[9] Mario Gerla, Xiaoyan Hong, Guangyu Pei, UCLA, ***Fisheye State Routing Protocol (FSR) for Ad Hoc Networks,*** IETF MANET Working Group Internet Draft, June 17, 2002.

[10] T. Clausen, P. Jacquet, Project Hipercom, ***Optimized Link State Routing Protocol (OLSR),*** Network Working Group, October 2003.

[11] E. W. Dijkstra, ***A Note on Two Problems in Connexion with Graphs,*** Numerische Mathematlk l, 269 - 27 I (l 959).

[12] Defense Advanced Research Projects Agency, Information Processing Techniques Office, ***DARPA Internet Program Protocol Specification,*** September 1981.