DESIGNING A SIMULATION BASED APPLICATION FOR LOWEST ID CLUSTERING ALGORITHM IN MOBLE AD HOC NETWORKS

Project Report submitted in partial
fulfillment of the requirement for the degree
of

Bachelor of Technology.

in

Computer Science & Engineering

Under the Supervision of

Mr.Amol Vasudeva

By

Riddhima Pathak(111202)

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "DESIGNING A SIMULATION APPLICATION FOR LOWEST ID CLUSTERING ALGORITHM IN MOBLE AD HOC NETWORKS", submitted by Riddhima Pathak (111202) in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of any other degree or diploma.

Date:

Supervisor's Name: Mr.Amol Vasudeva

Designation: Assistant Professor

Acknowledgement

I would like to express my gratitude to all those who gave me the possibility to complete this project. I want to thank the Department of CSE & IT in JUIT for giving me the permission to commence this project in the first instance, to do the necessary research work.

I am deeply indebted to my project guide Mr Amol Vasudeva, whose help, stimulating suggestions and encouragement helped me in all the time of research on this project. I feel motivated and encouraged every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by him, who gave me his unwavering support.

I am also grateful to Mr.Amit Singh(CSE Project lab) for his practical help and guidance.

Date: Riddhima Pathak

Table of Contents

Glossary

MANET ----------Mobile Adhoc Networks

MOBIC ----------Mobility Based Clustering

ID ----------Identity

DSDV ---------- Destination-Sequenced Distance-Vector Routing

AODV ---------- Ad hoc On-Demand Distance Vector Routing

MM ----------Mobility Metric

MN ----------Mobility Node

RWP ----------Random Way Point Model

CH ----------Cluster Head

CN ----------Cluster Node

RIP ----------Routing Information Protocol

JVM ----------JAVA Virtual Machine

IDE ----------Integrated Development Environment

JRE ----------JAVA RunTime Environment

List of Figures and Tables

Abstract

A wireless ad-hoc network is a self-configuring network that does not depend on any infrastructure for communication. Every node is free to move anywhere in the network and data is exchanged independently across the network. Destruction of one node does not affect the communication of other nodes in the network. Every node in the network can act as both host as well as destination. A wireless ad-hoc network does not rely on fixed infrastructure or predetermined connectivity. It is a self organizing multi-hop wireless network in which all of the nodes can be mobile. Data is exchanged between nodes via wireless communication. Aside from the ability to be rapidly deployed, wireless ad-hoc networks have the ability to exist in highly volatile environments. Unlike traditional networks, if one node is destroyed it will not impact the data exchange between the remaining nodes within the network. The selection of the correct network topology given the network characteristics is extremely important to ensure reliable and efficient communication between nodes. The topology of a network can be either hierarchical or flat. In a hierarchical topology nodes are divided into clusters. Within each cluster, a cluster head is selected via a mathematical formulation or heuristic method. Cluster heads are responsible for keeping track of which nodes are maintained in their respective cluster. Furthermore, they are responsible for transmitting data between clusters. Each of the cluster heads maintains a continuously updated routing table. This table contains specific information detailing which cluster each node belongs to. If a node desires to transfer information to another node, the information is sent to the sender node's cluster head. This cluster head scans its routing table to determine which cluster the recipient is in. If the recipient is in the same cluster, the data is immediately forwarded to the receiving node. If not, the cluster head scans its routing table to determine which cluster the recipient is in and forwards the data to the appropriate cluster head where it is again forwarded to the recipient. The Lowest-ID is considered as a simplest clustering scheme algorithm. In this scheme unique identifier (ID) is assigned to each node. All nodes recognize its neighbors ID and CH is chosen according to minimum ID. Thus, the node IDs of the neighbors of the CH will be higher than that CH. The main drawback with this scheme is there is no limitation to the number of nodes attached to the same CH. Also, CHs are prone to power drainage due to serving as cluster heads.

Literature Survey

Chapter 1: Mobile Adhoc Networks (MANETS)

1.1 Introduction

Over past few years there has been a revolutionary change in the area of telecommunication brought by different kinds of new technology and devices. Networking devices i.e. Laptops, cell phones etc. are getting smaller with increasing mobility and connectivity to internet cloud. In this scenario dependency on hard wired network is no longer of our concern. With The exponential growth of wireless devices the number of wireless internet users is going to exceed than that of the wired line internet users within few years. IEEE 802.11x standards have brought a revolutionary change in the field of wireless networks. To obtain connectivity within a network or to communicate with devices across the network, only a wireless interface or access point is required.

Network devices can get connected across the network or they can obtain mutual connectivity by organising themselves into ad-hoc networks. According to IEEE 802.11 “An ad-hoc network is a network composed of communication devices within mutual transmission range of each other via wireless medium “. In modern communication system where there is a requirement of fast and reliable network, devices may be connected via wireless technologies but in some cases wired backbone structure connects several different wireless networks. There are various kinds of wireless network technologies like WPAN, WLAN and MANET etc.

A mobile ad-hoc network (MANET) is a self-configuring network of mobile devices; i.e. without proper infrastructure and is connected by wireless links. In other words “A MANET is a collection of mobile user that communicate over relatively bandwidth constrained links ". . It is a Random deployable network where nodes are mobile with dynamic topology. In the network topology, each device is termed as a node and the virtual connectivity among each 2

node is termed as the link. There exists a link among the nodes when they are within the transmission range of each other. The nodes are connected in a decentralise fashion where identifying the topology, delivering the message or execution of any kind of operation is done by each node independently. Nodes organise themselves to send any message or packet to other nodes by creating a multi-hop networking structure. It may be a standalone network or can be connected to external networks. Introduction of new technologies like HyperLAN, Bluetooth etc, have facilitated to evolve MANET outside its earlier and constrained domain.

IEEE 802.11 family has got two main versions that deals with Wireless networks

1. IEEE 802.11a, that works in 5GHz band and provides data rate up to 54Mbps and uses ODFM interface.

2. IEEE 802.11b , operates in 2.4GHz band with data rate 11Mbps. That studied in framework of MANET.

The mobile devices can work in two modes under IEEE 802.11 standards . They are the ad-hoc mode and the Infrastructure mode. Infrastructure mode contains wireless interfaces that further leads to connectivity through wired backbone . wireless access point is necessary for this kind of structure. It supports high scalability and centralised security but in ad-hoc wireless network mobile nodes directly communicate with each other without the help of intermediate access point .Ad hoc network is also calledas the peer to peer network where each node acts as a router along with its normal job as a communicating device. An ad-hoc network can be further connected to internet with the help of gateway nodes..

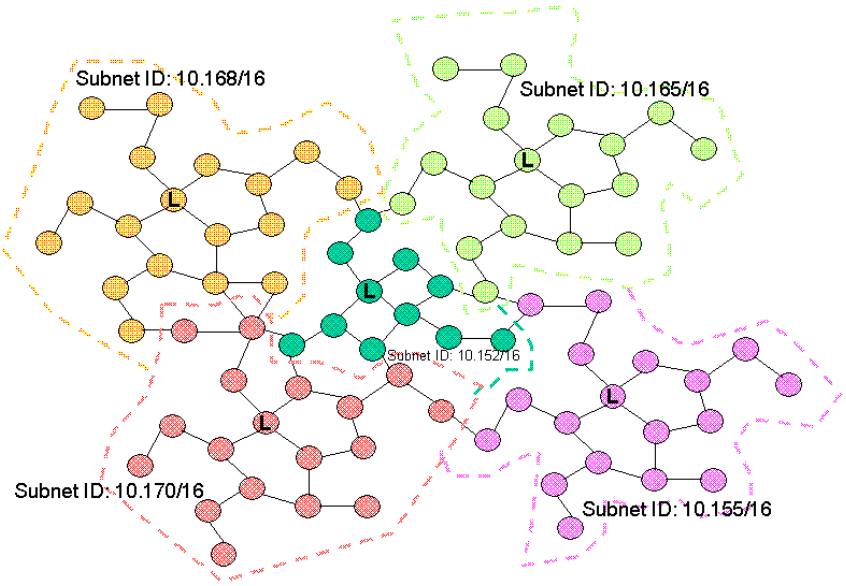


Figure 1.1: Example of MANET

1.2Characteristics

- In MANET, each node acts as both host and router. That is it is autonomous in behavior.
- Multi-hop radio relaying-When a source node and destination node for a message is out of the radio range, the MANETs are capable of multi-hop routing. The nodes can join or leave the network anytime, making the network topology dynamic in nature.
- Network topology dynamic in nature.
- Distributed nature of operation for security, routing and host configuration. A centralized firewall is absent here.
- The nodes can join or leave the network anytime, making the network topology dynamic in nature.
- Nodal connectivity is intermittent.
- Mobile nodes are characterized with less memory, power and light weight features.
- Completely symmetric environment.

- The reliability, efficiency, stability and capacity of wireless links are often inferior when compared with wired links. This shows the fluctuating link bandwidth of wireless links.
- Mobile and spontaneous behavior which demands minimum human intervention to configure the network.
- All nodes have identical features with similar responsibilities and capabilities and hence it forms a completely symmetric environment.
- High user density and large level of user mobility.

1.3 Applications

Mobile Ad-hoc Networks provide self deployed mode of communication but it leads to the problem of scalability, mobility and energy efficiency . the set of applications of MANET vary from small scale ,static and limited power source network to large scale , highly dynamic and mobile networks. MANET can be deployed in any area where there is requirement of cheap ,fast and direct communication ,such as military service, PDAS ,Group communication etc. In these areas the network can be established or removed on demand without affecting the infrastructure.

- **Military Services**:- This is the basic and common application of MANET . In military services there is requirement of highly mobile network where infrastructure is not necessary, because to setup a highly wired and complex static network may create problem in battlefield, besides soldiers are highly mobile so in order to broadcast any message we need a dynamic network. But in military scenario there is requirement of security, high latency, reliability and recovery form failures etc. So military networks are designed to establish low probability of detection so nodes in ad-hoc network must not radiate more power and there should be irregularity in message passing that helps with the problem of detection and interference.

- **Disaster management:-** In Disaster situations use of dynamic and peer to peer communication is highly important. MANET provides network with self configuration and minimum overhead . MANET can be set up on any area with minimum or no infrastructure . In situations like earthquake, flood or any manual or natural destruction this kind of network can be implemented as it provides high mobility and requires no central connectivity .

- **Group Communication:-** Any shared application running on current internet based scenario is typically based on client/server communication for ex- Instant messaging ,server gaming etc. But if a group wants to communicate with another group it has to depend on server then message is passed to another client through that central network but MANET provides P2P communication without centralised approach. It is termed as GCA(Group Communication Application) . Now IM services that earlier used to depend on centralise operator has become distributed and can be implemented on a dynamic network . If two group wants to communicate the just have to come close enough to get connected through multi hop paths.

- **Personal Area Network:-** it is a small range of network that connects mobiles phones, PDAS laptops etc. It can be established as a network to communicate among connected nodes or it can be used to connect a back bone network.

1.4 Limitations

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

- **Scalability:-** Scalability can be broadly defined as whether the network is able to provide an acceptable level of service even in the presence of a large number of nodes. In MANET when the network size increases no. Of packets send by node also increases that leads to drainage of limited battery power and network life time gets reduced thus scalability is major challenging issue.

- **Energy Efficiency:-** Energy efficiency is the major challenging issue as it affects the network life time and stability.

- **Security:-** Mobility implies higher security risks such as peer-to- peer network architecture or a shared wireless medium accessible to both legitimate network users and malicious attackers. Eavesdropping, spoofing and denial-of-service attacks are major vulnerabilities.

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

- **Device discovery:-** Identifying relevant newly moved in nodes and informing about their existence need dynamic update to facilitate automatic optimal route selection.

- **Topology Maintenance:-** Updating information of dynamic links among nodes in MANETs is a major challenge.

- **Transmission quality:-** This is an inherent problem of wireless communication caused by several error sources that result in degradation of the received signal.

1.5 Challenges

- The reliability of wireless transmission is resisted by different factors-The wireless link characteristics are time-varying in nature. There are transmission impediments like fading, path loss, blockage and interference that adds to the susceptible behavior of wireless channels.

- Limited range of wireless transmission-The limited radio band results in reduced data rates compared to the wireless networks. Hence optimal usage of bandwidth is necessary by keeping low overhead as possible

- Packet losses due to errors in transmission – MANETs experience higher packet loss due to factors such as hidden terminals that results in collisions, wireless channel issues (high bit error rate (BER)), interference, and frequent breakage in paths caused by mobility of nodes, increased collisions due to the presence of hidden terminals and uni-directional links.

- Route changes due to mobility- The dynamic nature of network topology results in frequent path breaks.

- Frequent network partitions- The random movement of nodes often leads to partition of the network. This mostly affects the intermediate nodes.

- The application of this wireless network is limited due to the mobile and ad hoc nature. Similarly, the lack of a centralized operation prevents the use of firewall in MANETs. It also faces a multitude of security threats just like wired networks. It includes spoofing, passive eavesdropping, denial of service and many others. The attacks are usually classified on the basis of employed techniques and the consequences.

Chapter 2: Routing Algorithms

2.1 Definition

Routing is the process of selecting best paths in a network. To find and maintain routes between nodes in a dynamic topology with possibly unidirectional /by directional links, using minimum resources

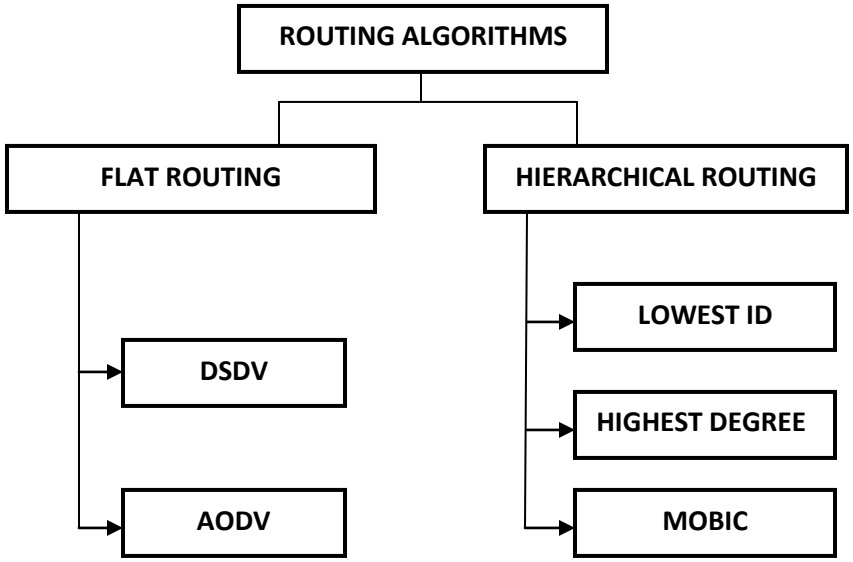
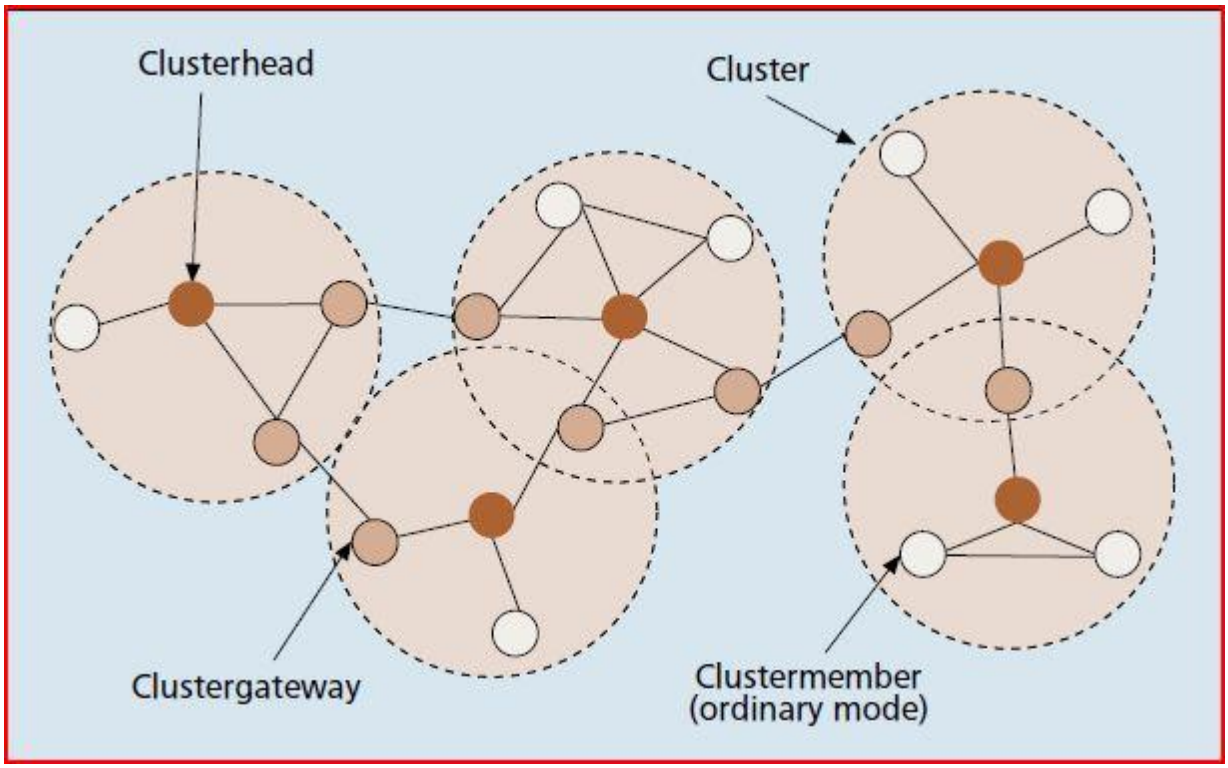


Figure 2.1: Routing Algorithms

Division of network into different virtual groups , based on certain rules in order to differentiate the nodes allocated to other sub-network. Or in simple words we can say that clustering in MANET is defined as virtual partitioning of nodes into sub networks according to geographical area.



A typical cluster structure has shown in figure 1.1. It can be seen that different nodes are grouped to form a structure called as cluster on the basis of closeness and other factors. In any cluster structure there every mobile node is assigned with a status or function .on the basis of there work and status nodes in any structure can be divided into three categories.

Cluster head :- It is a local co-ordinator of a cluster . it performs intra-cluster routing packet forwarding etc. A Cluster head does the resource management for its member nodes and perform inter-cluster and intra-cluster communication. It works similarly as base station. A cluster is shown in the given figure with dark-filled circles.

Gateway node:- A cluster gateway is a non cluster head node with inter-clusters links. it can access neighbouring clusters and exchanges cluster-related information between two clusters. It act as a common or distributed access point between two cluster head. Gateway nodes are of two types

(1) Ordinary gateway node:- When a node lies within the transmission range of two cluster heads i.e. both cluster heads remain its one hop- neighbour and it facilitates the transmission between them then this node is called as ordinary gateway node for those two clusters.

(2) Distributed gateway node:- When a node is a one-hop neighbour of a cluster head and it is connected to other node that is also immediate neighbour of other cluster head so that both cluster head can communicate with each other via 2-hop neighbours then those two nodes are termed as Distributed gateway node.

Ordinary node:- These are the nodes that exist as immediate neighbour of cluster head. They are cluster member but take part in topology and can act as cluster head or cluster gateway when there is a requirement.

Cluster control architecture:- There are two kind of cluster control architecture one-hop and d-hop depending upon the diameter of the cluster in one-hop control architecture every ordinary node remain at maximum of one hop distance to its cluster head . and in d-hop the distribution of ordinary node may be greater than one and at maximum of d-hop distance from central co-ordinator;

2.2 Types

2.2.1 Flat Routing Algorithm:

In a Flat Routing technique, the message to be delivered is directly passed from source node to the destination node without passing the message to any cluster head node. The information directly hops from one node to another and finds the shortest path to reach the destination node. Best example of flat routing is Routing Information Protocol which is a simple hop count protocol where maximum 15 number hops can be taken to reach the destination node.

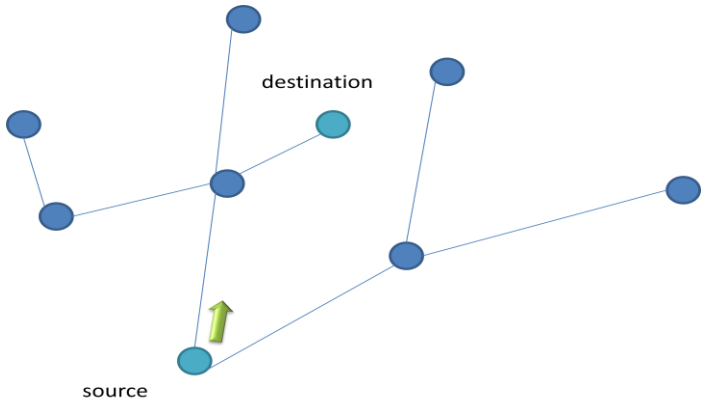


Figure 2.1: Flat Routing

2.2.2 Destination-Sequenced Distance-Vector Routing (DSDV):

DSDV is a table-driven routing scheme for ad hoc mobile networks based on the Bellman–Ford algorithm. The main contribution of the algorithm was to solve the routing loop problem. Each entry in the routing table contains a sequence number, the sequence numbers are generally even if a link is present; else, an odd number is used. The number is generated by the destination, and the emitter needs to send out the next update with this number. Routing information is distributed between nodes by sending full dumps infrequently and smaller incremental updates more frequently.

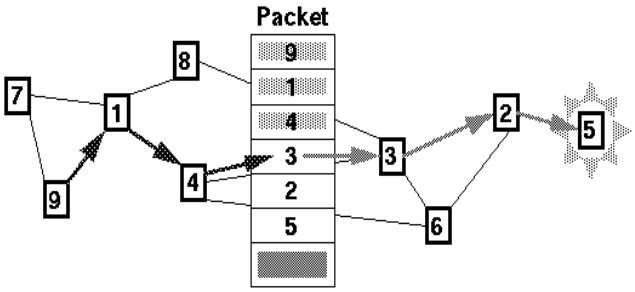


Figure 2.2 Destination-Sequenced Distance-Vector Routing

2.2.3 Ad hoc On-Demand Distance Vector Routing (AODV): In this routing algorithm, the path is drawn on demand by the source node when it wants to send the message that is when the source node desires to send a message and does not have any path so its broadcast the route request message across the network. All the nodes in the network after receiving this message will look for the shortest path through which the message can be sent .The source node will receive the message along with the IP address of the destination node and then can send the message through that path/route.

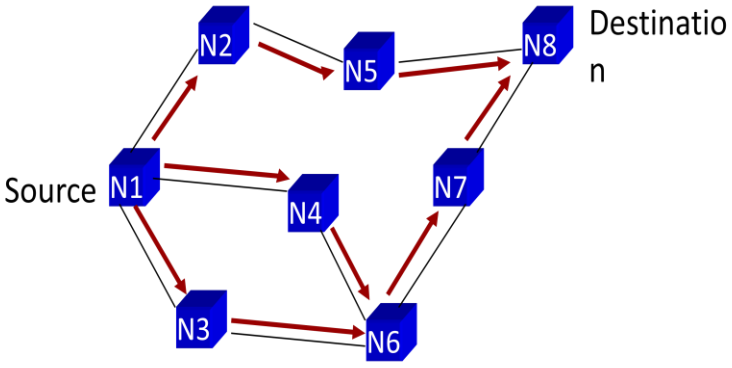


Figure 3.3 (a) Ad hoc On-Demand Distance Vector Routing- source

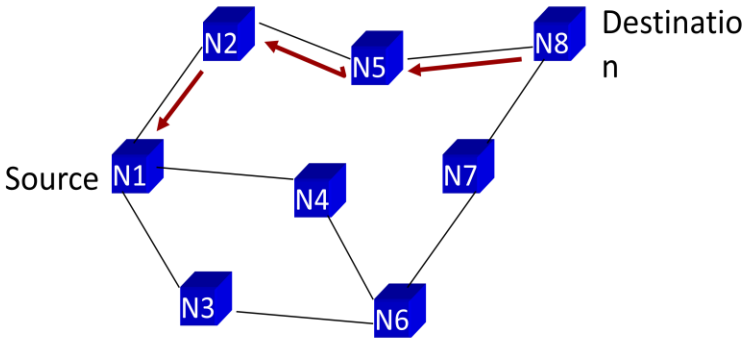


Figure 2.3 (b) Ad hoc On-Demand Distance Vector Routing- Destination

2.2.4 Hierarchical Routing:

In Hierarchical Routing nodes are divided into groups on the basis of the categories of nodes. These groups are known as clusters. A cluster can have three categories of nodes: cluster head, gateway node and ordinary or member nodes. In order to send the information from one node to another node, the message is first passed on to the cluster head and then the information is passed to the destination node after looking at the routing table for the desired cluster. If a node is within the same cluster then the information is directly passed; else first passed to the respective cluster head and then the destination cluster head followed by the destination node. There are numerous methods to form clusters in a simulation environment.

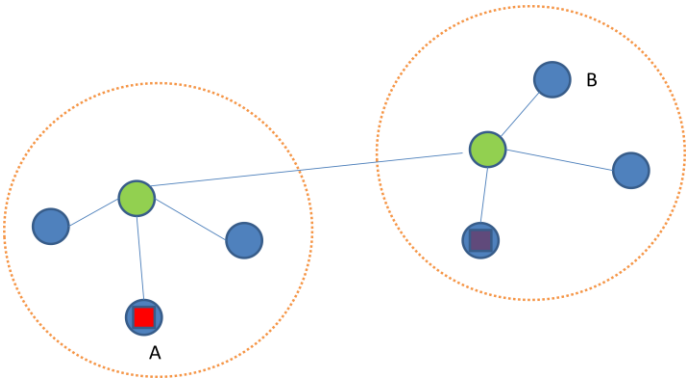


Figure 2.4 Hierarchical Routing

2.2.5 Lowest ID:

The Lowest-ID is considered as a simplest clustering scheme algorithm. In this scheme unique identifier (ID) is assigned to each node. All nodes recognize its neighbors ID and CH is chosen

according to minimum ID. Thus, the nodes IDs of the neighbors of the CH will be higher than that CH. The main drawback with this scheme is there is no limitation to the number of nodes attached to the same CH. Also, CHs are prone to power drainage due to serving as cluster heads.

Disadvantage:-

There is no limit of number of nodes in a cluster so overhead may occur.

No network related parameters are taken into consideration so performance of such network is unpredictable and random.

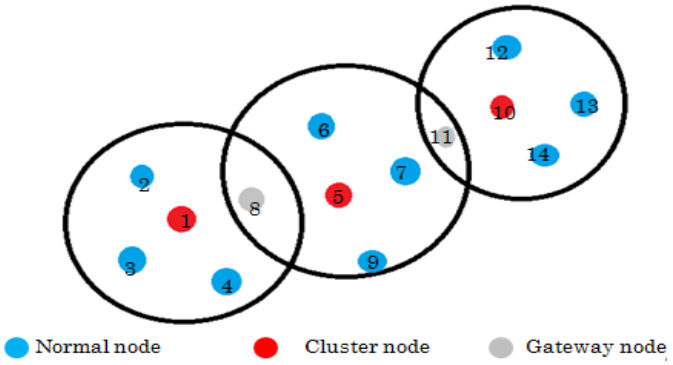


Figure 2.5 Lowest ID

2.2.6 Highest Degree:

In the highest degree clustering, clustering is performed on the basis of its node degree. All nodes flood its connectivity value within their transmission range. Thus, a node decides to become a CH or remain as CN by comparing the connectivity value of its neighbors with its own value. Node with highest connectivity value in its vicinity will become CH. Connectivity-based clustering follows the same circumstances of ID-based regarding to cluster size and performance degradation.

Disadvantage:-

There is no bound on no. Of nodes per cluster head so cluster heads become overloaded.

Does not provide any quantitative measure of stability.

Due to change in network topology, this approach gives high turnover of cluster heads. This is because when the node with the highest connectivity drops, it will fail to be re-elected as a cluster head.

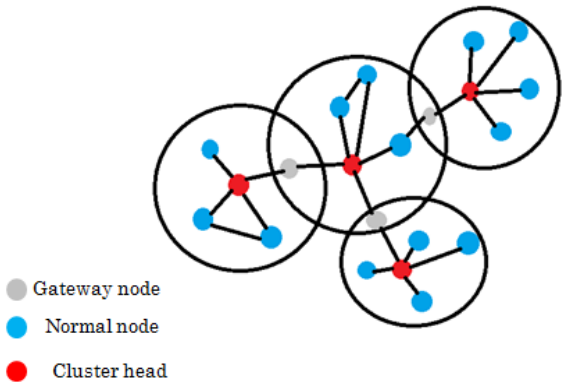


Figure 2.6: Highest Degree

2.2.7 Mobility Based Clustering (MOBIC):

MOBIC uses mobility of the nodes as a feature to form the clusters. Each node in the Mobile Ad hoc Network computes the ratio of two successive "Hello" messages from all its neighbors. This gives the relative mobility metric of the nodes with respect to their respective neighbors. Then by calculating the variance of relative mobility values of all the nodes with respect to their neighbors in a distributed manner, the aggregate speed of all the mobile nodes can be estimated. Finally, the mobile node with lowest variance value in its neighborhood is elected as the cluster head.

Disadvantage:-

It uses signal strength in order to measure mobility but due to noise , obstacles or battery power weight based on signal strength may not be accurate.

Mobility is considered as measurement of stability of cluster head but there are other factors that also need to be taken into account because single parameter will not give desired stability in all scenarios only we are looking at the stability of cluster head alone and not at the entire network. To ensure stability of entire network gateway nodes should be considered.

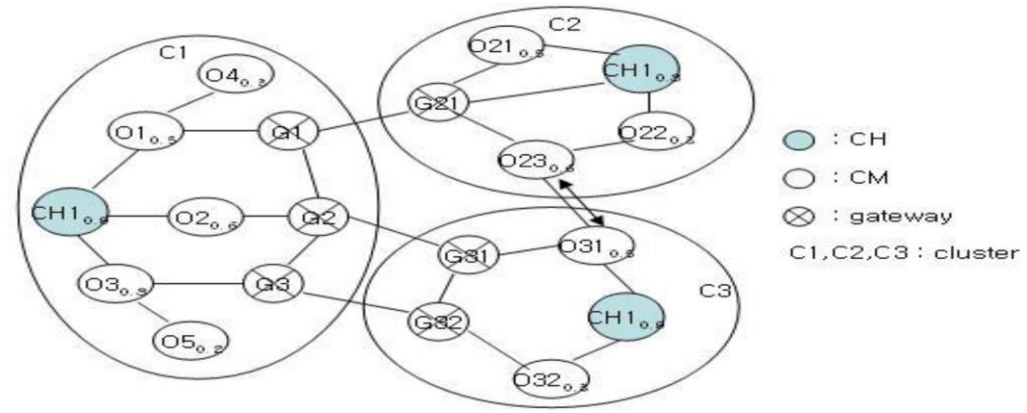


Figure 2.7: MOBIC

Why MANET require Clustering:- It has been already shown that a cluster architecture guarantee basic performance achievement in a network with large no of mobile nodes. There are following reasons for a ad-hoc network that shows the requirement of clustering.

A cluster structure facilitates the reuse of resources to increase the system capacity. With the non-overlapping multi cluster structure two clusters can deploy the same frequency or code set if they are not neighbouring clusters. A cluster can also put the co-ordination in transmission with the help of other special nodes.

Other benefit lies in field of routing as the set of cluster heads and cluster gateways can form a virtual backbone for inter-cluster routing thus the routing information can be restricted in this set of nodes.

A cluster structure makes ad-hoc networks looks smaller and more stable. If a node changes its cluster then only nodes residing in corresponding clusters are need to update the information no need the changes to be seen by entire network.

Chapter 3: Simulation of the Lowest ID Clustering Algorithm

In communication and computer network research, network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/packets, etc.) using mathematical formulas, or actually capturing and playing back observations from a production network. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions. Simulation is a very important modern technology. It can be applied to different science, engineering, or other application fields for different purposes. Computer assisted simulation can model hypothetical and real-life objects or activities on a computer so that it can be studied to see how the system function. Different variables can be used to predict the behavior of the system. Computer simulation can be used to assist the modeling and analysis in many natural systems. Typical application areas include physics, chemistry, biology, and human-involved systems in economics, finance or even social science. Other important applications are in the engineering such as civil engineering, structural engineering, mechanical engineering, and computer engineering. Application of simulation technology into networking area such as network traffic simulation, however, is relatively new.

3.1. Network simulation and simulator

Generally speaking, network simulators try to model the real world networks. The principal idea is that if a system can be modeled, then features of the model can be changed and the corresponding results can be analyzed. As the process of model modification is relatively cheap than the complete real implementation, a wide variety of scenarios can be analyzed at low cost (relative to making changes to a real network). Network simulator always contain the

However, network simulators are not perfect. They cannot perfectly model all the details of the networks. However, if well modeled, they will be close enough so as to give the researcher a meaningful insight into the network under test, and how changes will affect its operation.

3.2.Simulation and emulation

In the research area of computer and communications networks, simulation is a useful technique since the behavior of a network can be modeled by calculating the interaction between the different network components (they can be end-host or network entities such as routers, physical links or packets) using mathematical formulas. They can also be modeled by actually or virtually capturing and playing back experimental observations from a real production networks. After we get the observation data from simulation experiments, the behavior of the network and protocols supported can then be observed and analyzed in a series of offline test experiments. All kinds of environmental attributes can also be modified in a controlled manner to assess how the network can behave under different parameters combinations or different configuration conditions. Another characteristic of network simulation that worth noticing is that the simulation program can be used together with different applications and services in order to observe end-to-end or other point-to-point performance in the networks.

Network emulation, however, means that network under planning is simulated in order to assess its performance or to predict the impact of possible changes, or optimizations. The major difference lying between them is that a network emulator means that end-systems such as

computers can be attached to the emu lator and will act exactly as they are attached to a real network. The major point is that the network emulator's job is to emulate the network which connects end-hosts, but not the end-hosts themsel ves. Typical network emulation tools include NS2 which is a popular network simulator that can also be used as a limited-functionality emulator. In contrast, a typical network emulator such as WANsim [WANsim] is a simple bridged WAN emulator that utilizes some Linux functionality.

3.3 Type of network simulators

Different types of network simulators can be categorized and explained based on some criteria such as if they are commercial or free, or if they are simple ones or complex ones.

3.3.1 Commercial and open source simulators

Some of the network simulators are commercial which means that they would not provide the source code of its software or the affiliated packages to the general users for free. All the user s have to pay to get the license to use their software or pay to order specific packages for their own specific usage requirements. One typical example is the OPNET [OPNET]. Commercial simulator has its advantage and disadvantage. The advantage is that it generally has complete and up-to-date documentations and they can be consistently maintained by some specialized staff in that company. However, the open source network simulator is disadvantageous in this aspect, and generally there are not enough specialized people working on the documentation. This problem can be serious when the different versions come with many new things and it will become difficult to trace or understand the previous codes without appropriate documentations.

On the contrary, the open source network simulator has the advantage that everything is very open and everyone or organization can contribute to it and find bugs in it. The interface is also open for future improvement. It can also be very flexible and reflect the most new recent developments of new technologies in a faster way than commercial network simulators. We can see that some advantages of commercial network simulators, however, are the disadvantage for

the open source network simulators. Lack of enough systematic and complete documentations and lack of version control supports can lead to some serious problem s and can limit the applicability and life-time of the open source network simulators. Typical open source network simulators include NS2 [NS2][NS2-wiki], NS3[NS3]. We will introduce and analyze them in great detail in the following sections.

Network simulators

	Network simulators name
Commercial	OPNET, QualNet
Open source	NS2, NS3, OMNeT++, SSFNet, J-Sim

Note that limited by the length, not all of them will be introduced in detail in this paper. However, we will focus on some typical ones and introduce others briefly.

3.3.2 Simple vs. complex

Currently there are a great variety of network simulators, ranging from the simple ones to the complex ones. Minimally, a network simulator should enable users to represent a network topology, defining the scenarios, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to process network traffic. Graphical applications also allow users to easily visualize the workings of their simulated environment. Some of them may be text-based and can provide a less visual or intuitive interface, but may allow more advanced forms of customization. Others may be programming-oriented and can provide a programming framework that allows the users to customize to create an application that simulates the networking environment for testing.

3.3.3 OPNET

OPN ET [OPNET] is the registered commercial trademark and the name of product presented by OPNET Technologies incorporation. It is one of the most famous and popular commercial network simulators by the end of 2008. Because of it has been used for a long time in the industry, it become mature and has occupied a big market share.

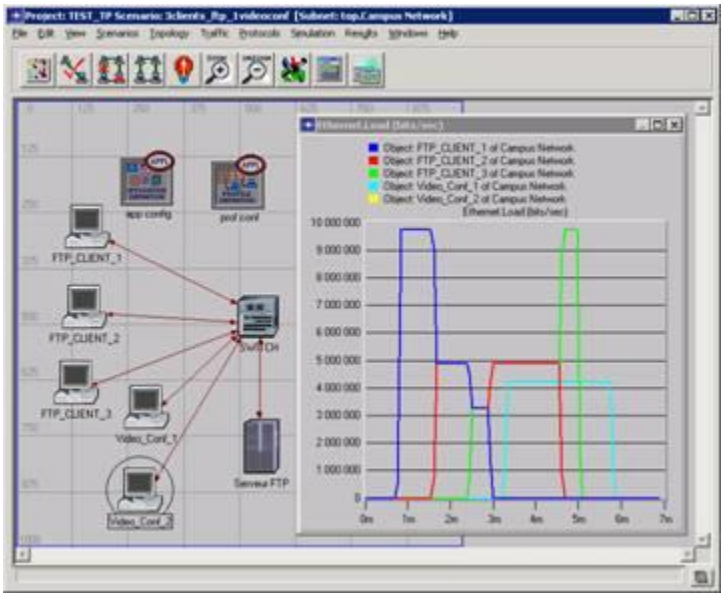


Figure 3.1: OPNET

OPNET is based on a mechanism called discrete event system which means that the system behavior can simulate by modeling the events in the system in the order of the scenarios the user has set up. Hierarchical structure is used to organize the networks. As other network simulators, OPNET also provides programming tools for users to define the packet format of the protocol. The programming tools are also required to accomplish the tasks of defining the state transition machine, defining network model and the process module.

As of all, OPNET is a popular simulator used in industry for network research and development. The GUI interface and the programming tools are also useful to help the user to build the system they want.

3.3.3 Network Simulator 2 (NS2)

NS2 is one of the most popular open source network simulators. The original NS is a discrete event simulator targeted at networking research. In this section, we will give a brief introduction to the NS2 system.

NS2 is the second version of NS (Network Simulator). NS is originally based on REAL network simulator [REAL]. The first version of NS was developed in 1989 and evolved a lot over the past few years. The current NS project is supported through DARPA. The current second version NS2 is widely used in academic research and it has a lot of packages contributed by different non-benefit groups. For NS2 documentation on recent changes, refer to the NS 2 official webpage.

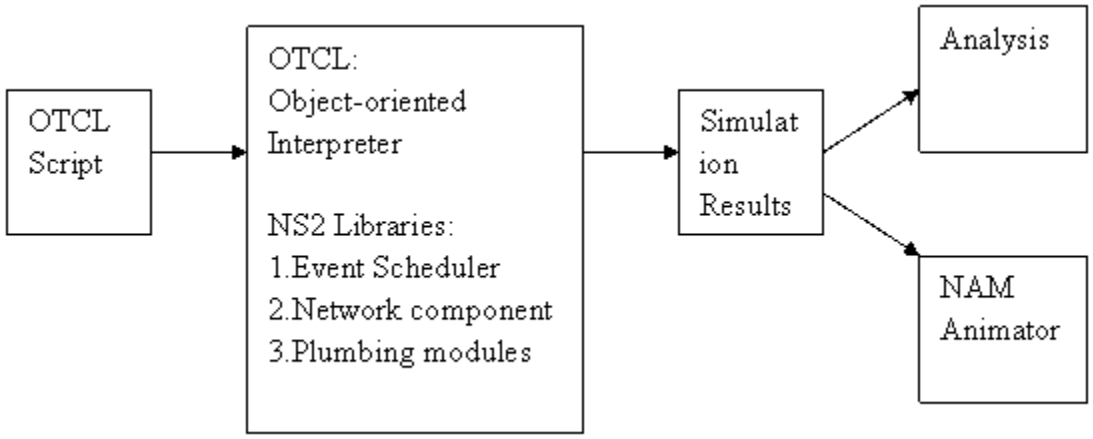


Figure 3.2: NS2

Another feature of NS2 is the event scheduler. In NS2, the event scheduler keeps track of simulation time and release all the events in the event queue by invoking appropriate network components. All the network components use the event scheduler by issuing an event for the packet and waiting for the event to be released before doing further action on the packet.

The network simulator ns-2 is another discrete event simulator targeting at networking research. It is the abbreviation of Network simulator version two, which first been developed by 1989 using as the REAL network simulator. Now, NS-2 is supported by Defence Advanced Research Projects Agency (DARPA) and National Science Foundation. In NS-2 there is no graphical options but it has GUI for visualising the simulation run.

Its modules can be written in Scripting language OTcl or in C++. Besides ns-2 already includes a large library of common modules, which are usually sufficient for simulating 16

common wired- or wireless networks. This simulator is open source and provides online document.

Advantages:-

- NS-2 can support a considerable range of protocols in all layers. For example, the ad-hoc and WSN specific protocols are provided by NS-2.
- Open source model of NS-2 saves the cost of simulation, and online documents allow the users easily to modify and improve the codes.
- It can run on any system having GNU C-compiler gcc as it is written in C++. Thus it can run on UNIX platform as well.

Limitations:-

- People who want to use this simulator need to familiar with writing scripting language and modelling technique, the Tool Command Language(Tcl) is somewhat difficult to understand and write.

- NS-2 is sometimes more complex and time-consuming than other simulators to model a desired job.
- NS-2 provides a poor graphical support, no Graphical User Interface (GUI) ,the users have to directly face to text commands of the electronic devices.
- Due to the continuing changing the code base, the result may not be consistent, and contains bugs.
- NS-2 can not simulate problems of the bandwidth, power consumption or energy saving in WSN., NS-2 has a scalability problem number of nodes can not be exceeded to 100.

3.3.4. Network Simulator 3 (NS3)

Similar to NS2, NS3 is also an open sourced discrete-event network simulator which targets primarily for research and educational use. NS3 is licensed under the GNU GPLv2 license , and is available for research and development.

NS3 is designed to replace the current popular NS2 . However, NS3 is not an updated version of NS2 since that NS3 is a new simulator and it is not backward-compatible with NS2.

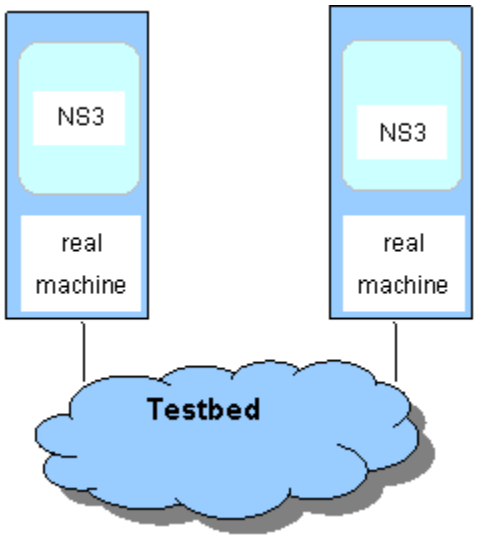


Figure 3.3: NS3

3.3.5 OMNeT++

Similar with NS2 and NS3, OMNeT++ is also a public-source, component-based network simulator with GUI support. Its primary application area is communication networks. OMNeT++ has generic and flexible architecture which makes it successful also in other areas like the IT systems, queuing networks, hardware architectures, or even business processes as well.

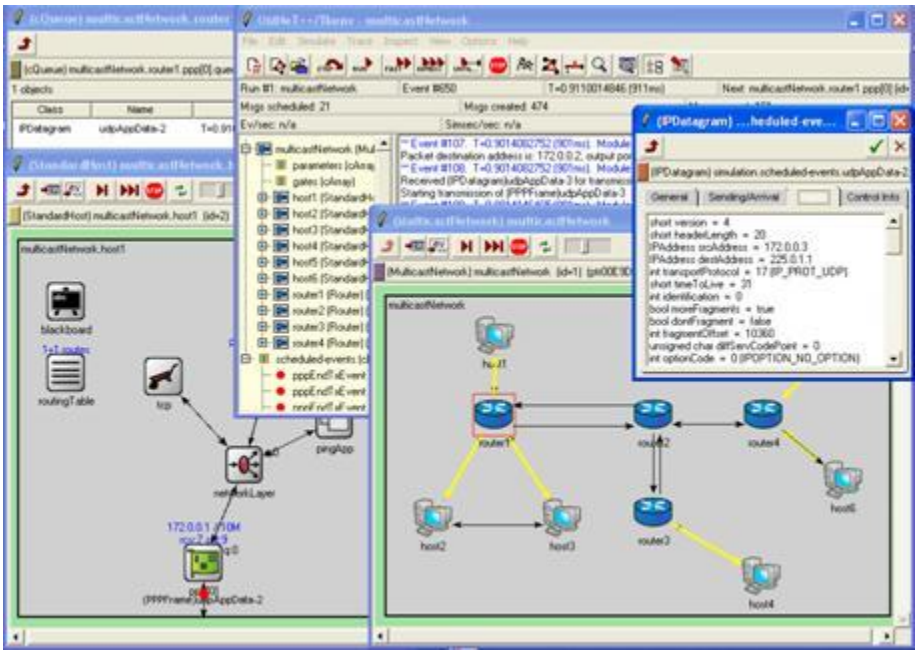


Figure 3.4: OMNET++

3.3.6 Glomo Sim

GloMoSim is a scalable simulation environment for wireless and wired network systems, developed by PCL (Parallel Calculation Laboratory) of UCLA (University of California in Los Angeles). It uses Parallel Simulation Environment for Complex System (PARSEC) that is a C based simulation language for sequential and parallel execution of Discreet event simulation model. Glomosim Supports protocol for purely wireless network.

Advantages:-

- Scalability of network model can be increased up to thousands of nodes.
- Glomo Sim is useful for a network model where nodes are exchanging high amount of information.

Limitations:-

- The updates of GloMoSim are not regular. The last update was in 2003.
- GloMoSim is being designed using the parallel discrete-event simulation capability provided by PARSEC However, the last version of GloMoSim supports only one sequential simulation

3.3.7 Qual Net

Qual Net is a commercial derivative of Glomo sim. Glomo Sim was designed for MANET but Qual Net supports wider range of networks and analysis i.e. MANET, QoS, Wired Network, Cellular, Satellite etc. Its modules are written and designed in C++. Qual Net developer toolkit contains following component .

- Qual Net Animator – Graphical component used as animation tool.
- Qual Net Scenario Designer - Graphical , Finite state machine based custom protocol design tool
- Qual Net Analyzer – Statistical Graphic tool used for built in and custom statistic collection
- Qual Net Packet Tracer –packet level tracing and visualisation.
- Qual net provides a GUI interface where simulation can be seen by the help of its animator component

Advantages:-

- Qual net provides GUI tools for protocol modelling.
- Scalability increases via parallel execution.
- It provides a clear built in measurement for each layer.
- Contains standard API for composition of protocol across each layer.

Limitations:-

- Difficult installation on Linux, as it has been built for older version of Linux (Fedora core 4).
- slow Java-based User Interface.
- very expensive.

3.3.8 NETSIM

Netsim provides complete model creation and modification capabilities along with graphical animation and data output. Netsim uses the event graph method and object-oriented programming for simulation. Netsim is a world wide web based simulation package written entirely in Java. Netsim allows expandability for complex modelling and integration with other Java-based programs, such as graphing and analysis packages.

Any simulation problem modelled in NETSIM can be viewed as an HTML page like applet in JAVA. It provides traditional procedural based simulation package.

Advantages:-

- It is available in all platforms without recompiling. It can be accessed remotely as modelling is JAVA applet based.
- It provides security by password mechanism for individual access , and protect against unauthorized change and delicacy.

- High maintenance as frequent modification is instantly distributed. All modification and implementation are made on-sever.
- Encourage communication and interaction through java and requires Browser capable of running JAVA programs.

Limitations:-

- For Other traditional simulation model loading time depends on system itself not on the internet. In case of web simulation if there is heavy traffic loading time may exceed and thus simulation.
- It does not have copy of simulation over system, so dependency on internet increases and interruptions in network may cause inconsistency.
- It is not efficient for longer run without updating frequently.

3.4 Uses of network simulators

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

nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare

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

3.5Mobility Models

3.5.1 Definition

Mobility models represent the movement of mobile users, and how their location, velocity and acceleration change over time. Such models are frequently used for simulation purposes when new communication or navigation techniques are investigated. For mobility modeling, the behavior or activity of a user's movement can be described using both analytical and simulation models. The input to analytical mobility models are simplifying assumptions regarding the movement behaviors of users. Such models can provide performance parameters for simple cases through mathematical calculations. In contrast, simulation models consider more detailed and realistic mobility scenarios. Such models can derive valuable solutions for more complex cases.

3.5.2 Types

Typical mobility models include

Random Walk Mobility Model

Random Waypoint Mobility Model

Random Direction Mobility Model

3.5.2.1 Random Walk Mobility Model

In this mobility model, a node moves from its current location to a new location by randomly choosing a direction and speed in which to travel. The new speed and direction are both chosen from pre defined ranges, [minspeed; maxspeed] and [0; 2π] respectively. Each movement in the Random Walk Mobility Model occurs in either at constant time interval 't' or a constant distance traveled 'd', at the end of which a new direction and speed are calculated. If any node reaches to the simulation boundary, it bounces off the simulation border with an angle determined by the incoming direction. The node then continues along this new path.

3.5.2.2 Random Waypoint Mobility Model

The random waypoint mobility model contains pause time between changes in direction and/or speed. Once a node begins to move, it stays in one location for a specified pause time. After the specified pause time is elapsed, the randomly selects the next destination in the simulation area and chooses a speed uniformly distributed between the minimum speed and maximum speed and travels with a speed v whose value is uniformly chosen in the interval (0 , Vmax). Vmax is some parameter that can be set to reflect the degree of mobility. Thereafter, it continues its journey toward the newly selected destination at the chosen speed. As soon as it arrives at the destination, it stays again for the indicated pause time before repeating the process.

3.5.2.3Random Direction Mobility Model

In random direction mobility model each node alternates periods of movement (move phase) to periods during which it pauses (pause phase). During the beginning of each move phase, a node independently selects its new direction and speed of movement. Speed and direction are kept constant for the whole duration of the node movement phase.

3.6 Simulation Environment

We will simulate an ad hoc network with n nodes randomly distributed in a 100 ×100 pixel area. The simulator was implemented in Java due to its multithreading feature and collection of numerous container classes. The network simulator has the ability to generate network with any number of nodes. The mobility is based on the Random Way Point model (RWP) in which a mobile node moves from its current location to a new location by randomly choosing a direction and speed in which to travel. The new speed and direction are both chosen randomly from pre-defined ranges, [0, 5 unit/sec] and [0, 2π] respectively. We have set the election process time to be 2 seconds i.e. after each 2 seconds the cluster head selection mechanism will be initiated. Once the election process is over, new directions and speeds are computed for all the nodes in the same manner, as mentioned above. This process was repeated throughout the simulation causing continuous changes in the topology of the underlying network. Once a node reaches the boundary of edge, it returns back with the same direction and speed. The transmission range of all the nodes has been taken to be 40 units, i.e. the two nodes can communicate with each other if the distance between them is shorter than 40 units. In order to complete these objectives, a network simulator was developed using Java, compute the five metrics as previously discussed, apply each of the clustering techniques, and evaluate congestion.

Velocity	0-6 m/s
Angle	0-360
Range	40
Number of nodes	20,30,40
Mobility Model	Random Waypoint Mobility Model

3.7Tools and Technologies:

3.7.1 Java 1.6 Version:

3.7.1.1Characteristics:

JAVA is a programming language, developed by Sun Microsystems and first released in 1995 (release 1.0). Since that time, it gained a large popularity mainly due to two characteristics:

- A JAVA programme is hardware and operating system independant. If well written (!), the same JAVA programme, compiled once, will run identically on a SUN/solaris workstation, a PC/windows computer or a Macintosh computer. Not mentioning other Unix flavors, including Linux, and every Web browser, with some restrictions described below.

  This universal executability is made possible because a JAVA programme is run through a JAVA Virtual Machine.
- it is an object oriented language. This feature is mainly of interest for software developers.

3.7.1.2 JAVA Virtual Machine (JVM):

A JAVA programme is build by a JAVA compiler which generates its own binary code. This binary code is independant from any hardware and operating system. To be executed, it needs a virtual machine, which is a programme analyzing this binary code and executing the instructions it contains.
Of course, this Java Virtual Machine (JVM) is hardware and operating system dependant. Two types of Virtual Machines exist: those included in every Web Browser, and those running as an independent programs, like the Java Runtime Environment (JRE) from Sun Microsystems. These programs need to be downloaded for your particular platform. As seen in the next paragraph, these two types of Virtual Machines do not behave exactly the same.

3.7.1.3 Applet and Standalone Application:

A JVM in a web browser runs a JAVA program as an Applet. The applet is embedded in a web page and downloaded from a web server like any other HTML page or image when requested. An independent JVM runs a JAVA program as a Standalone Application.

3.7.1.4 NetBeans Version 6.9.1:

NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers.[3]

The NetBeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++ and HTML5.[4]

NetBeans is cross-platform and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The NetBeans Team actively support the product and seek feature suggestions from the wider community. Every release is preceded by a time for Community testing and feedback.

Figure 3.5 NetBeans

3.8 Diagrams

3.8.1 Use Case Diagram

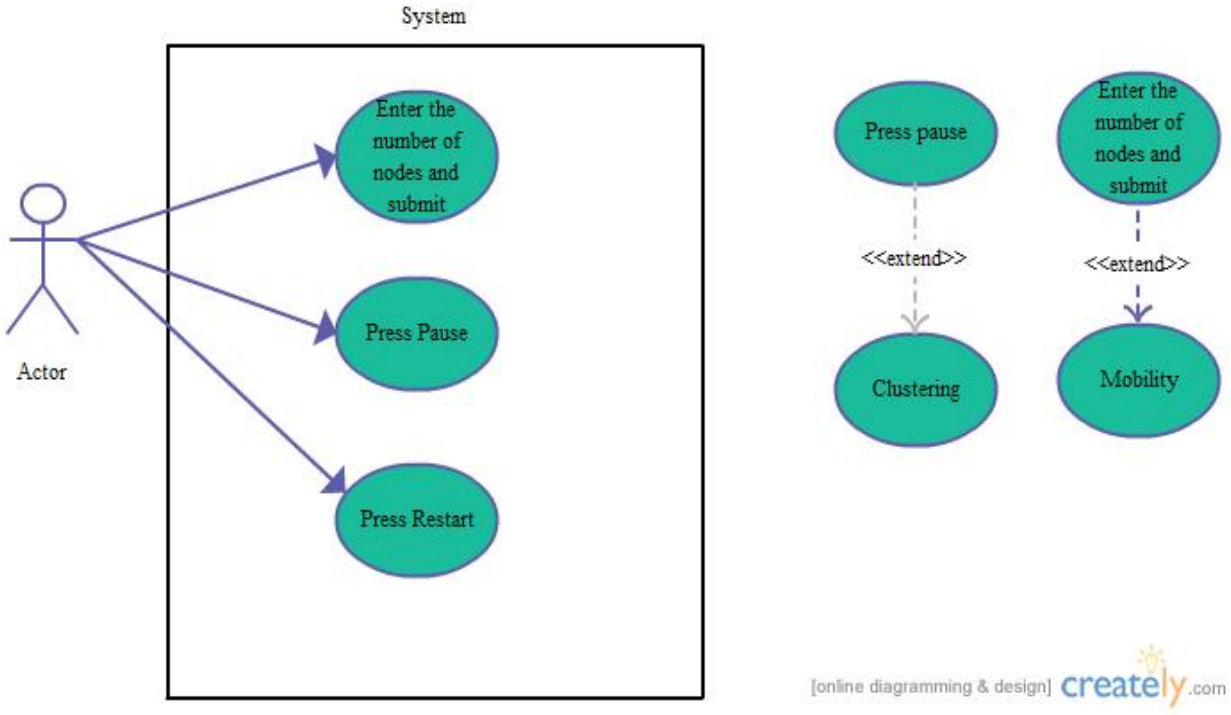


Figure 3.6 Use Case Diagram

Zero Level DFD

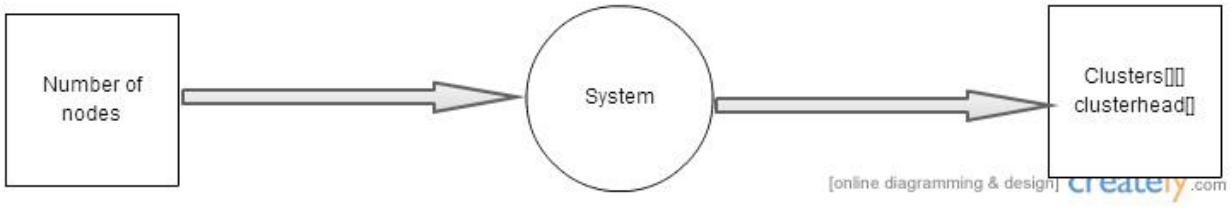


Figure 3.6 (a) Zero Level DFD

First Level DFD

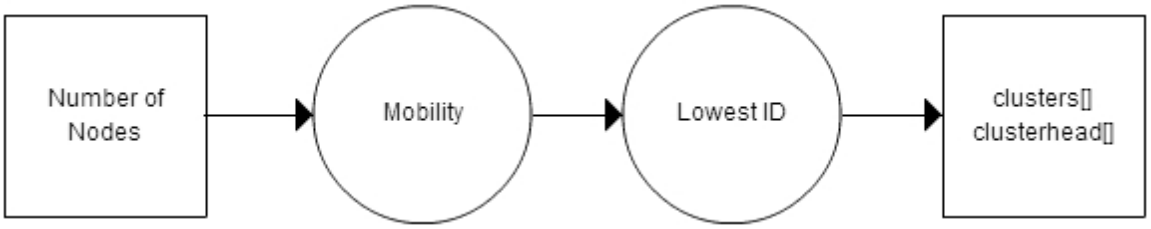


Figure 3.6 (b) First Level DFD

Second Level DFD

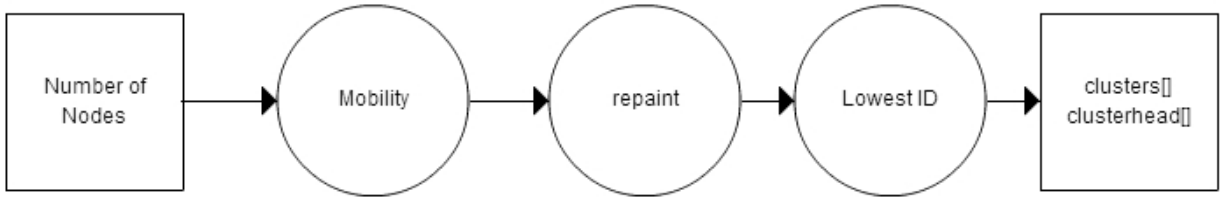


Figure 3.6 (c) Second Level DFD

3.9 Algorithm for Lowest ID Clustering Algorithm:

Find the Distance between nodes,distanceBetweenNodes(Node N[], int numberofnodes) and store in distanceMatrix[i][j]

Find the nodes with an edge between them,getEdgeMatrix(Node N[], int numberofnodes) and store it in edgeMatrix[i][j].

Find the clusterhead[],from the adjacency matrix(edgeMatrix[i][j]) find the node with the lowest id.

Form clusters[],with the lowest id node being the clusterhead[] and other nodes being either the gateway nodes or member nodes.

Compare if a node is already included in a cluster, that node will not participate again in the formation of the cluster.

for every node i

do

If edgeMatrix (i,j) AND edgeMatrix (i,j)!=cluster[k]

Then

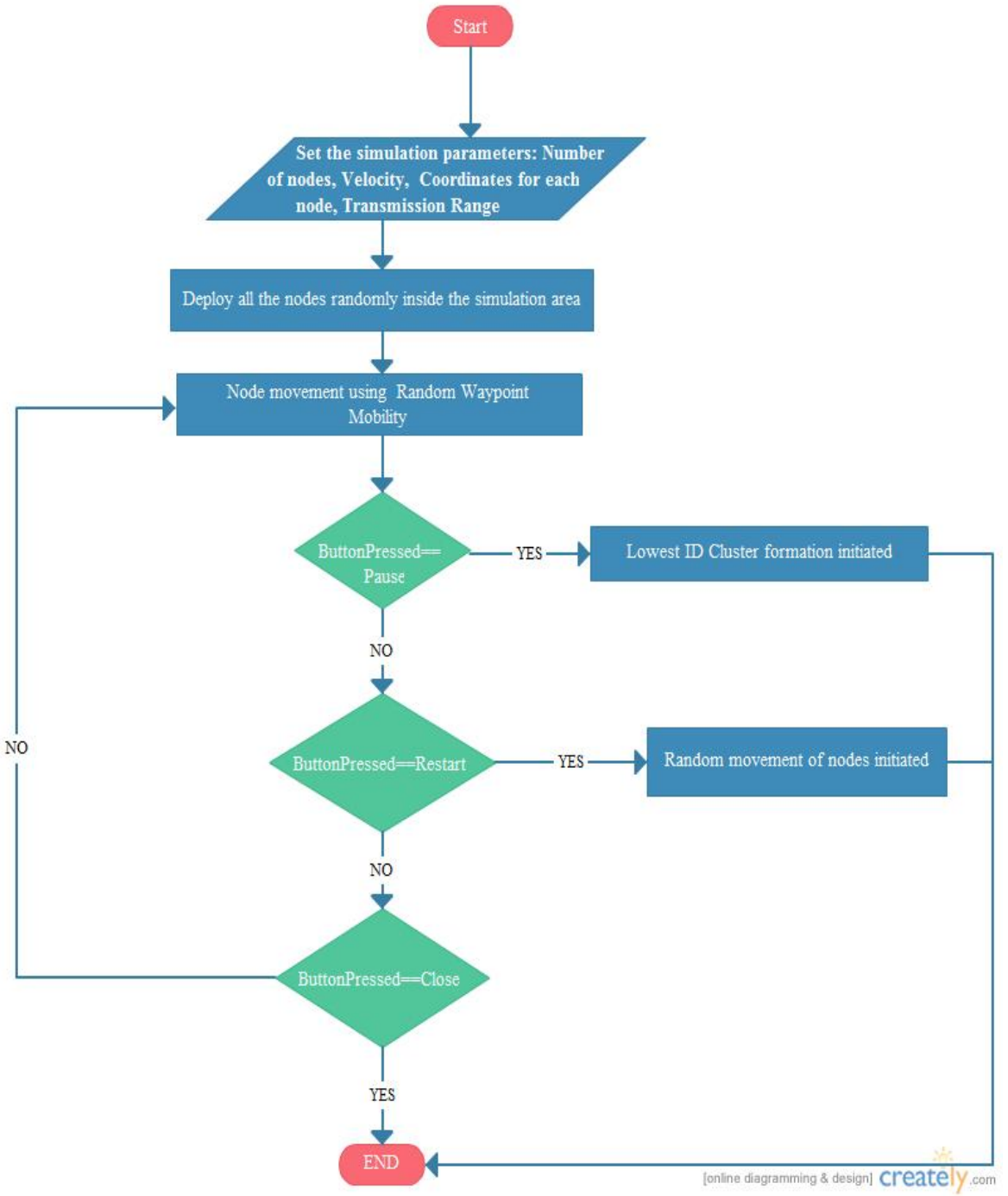
clusterhead[i]=1

cluster[k]=egdeMatrix[i][j];

end if

end for

Flowchart:


Start
Set the simulation parameters: Number of nodes, Velocity, Coordinates for each node, Transmission Range
Deploy all the nodes randomly inside the simulation area
Node movement using Random Waypoint Mobility
ButtonPressed== Pause
YES
Lowest ID Cluster formation initiated
NO
ButtonPressed==Restart
YES
Random movement of nodes initiated
NO
ButtonPressed==Close
NO
YES
END
[online diagramming & design] creately.com

Snapshots:

Figure 3.7 Initial Applet

Figure 3.8 Nodes entered

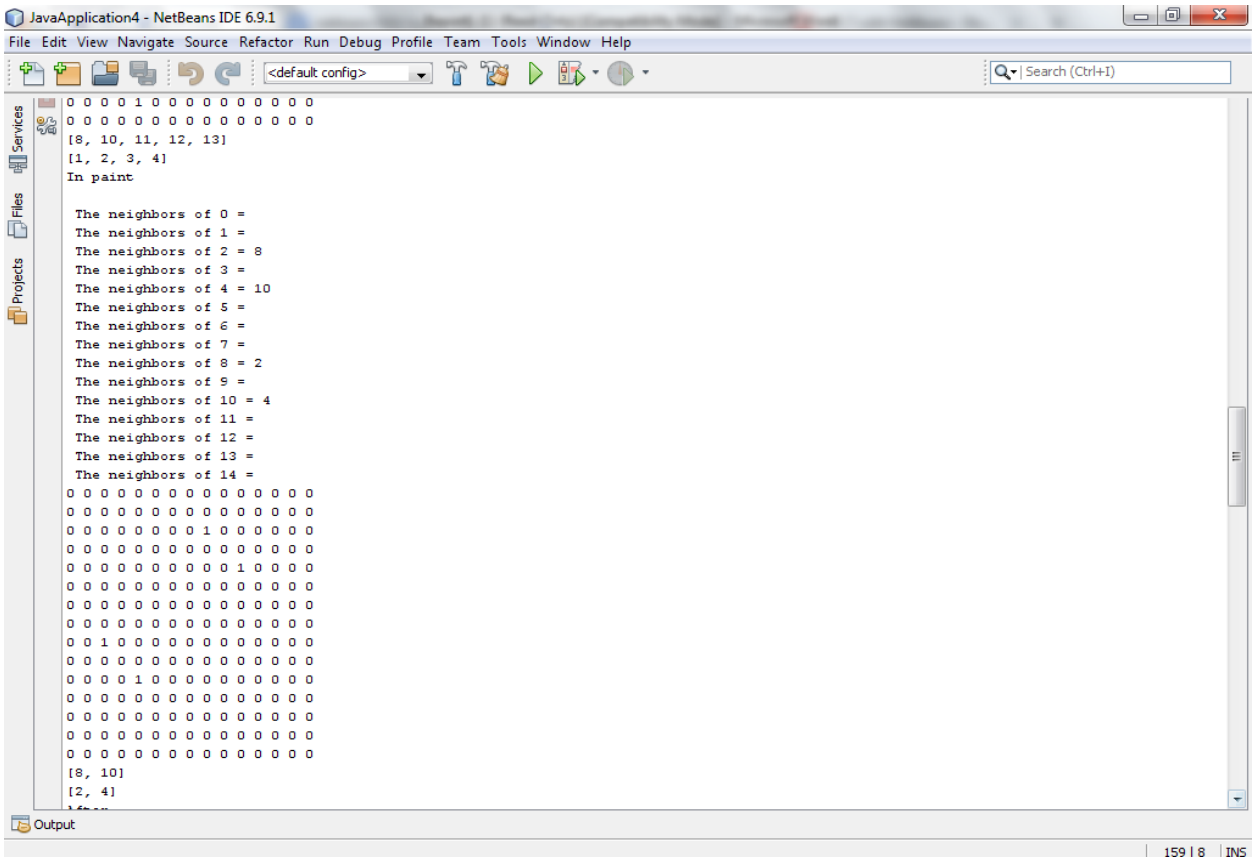


Figure 3.9 Console Output

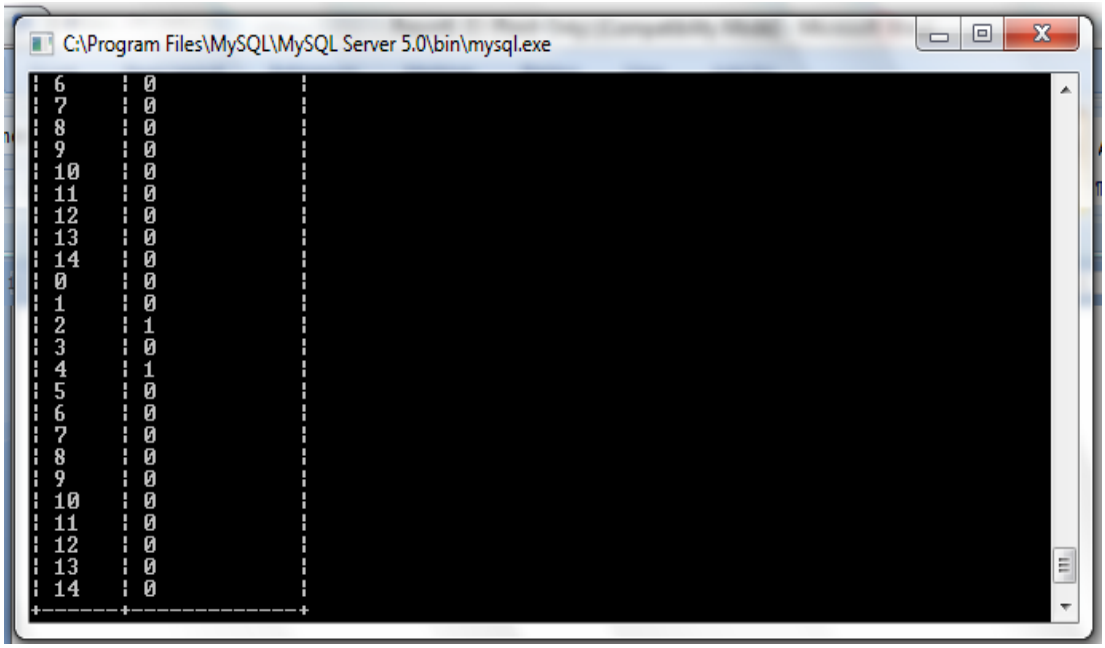


Figure 3.10 Command Prompt

3.10 Implementation

```
package fy;

import java.applet.Applet;

import java.awt.Button;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Label;

import java.awt.TextField;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.*;

import java.util.HashSet;

@SuppressWarnings("serial")

public class lo2 extends Applet implements Runnable, ActionListener {

    Set<Integer> cm;

   Set<Integer> ch;

            Node[] N;

            Node n=new Node();

            Thread th;

       int t;

            int numberOfNodes;

      // double ar;

            TextField nodes;
```

```
    // TextField area;
        Label l1;
    Label l2;
        Button resume, button, pause;
        int flag, suspend;
        int[] heads;
        int[] cluster;
        int c=0;
        double rad = 3.14159 / 180.0;
        Random rand=new Random();
    int xpos[] ;
        int ypos[] ;


  @Override
    public void init()
    {
    cm=new HashSet<Integer>();
ch=new HashSet<Integer>();
            flag = 0;
            suspend=0;
                    //ob = new MobicAlgo();
            N=new Node[numberOfNodes];
```

```
            nodes = new TextField(5);
 //  area=new TextField(5);
            l1 = new Label("Enter the number of nodes:  ");
  // l2 = new Label("Enter the area:  ");
            add(l1);
            add(nodes);
   //add(l2);
//   add(area);
   button = new Button("Submit");
            button.addActionListener((ActionListener) this);
            add(button);
            th=null;
            pause = new Button("Pause");
            pause.addActionListener((ActionListener) this);
            add(pause);
            resume = new Button("restart");
            resume.addActionListener((ActionListener) this);
            add(resume);
   xpos = new int[numberOfNodes];
            ypos = new int[numberOfNodes];
    heads=new int[99];
        cluster=new int[99];
```

```
    }
    public void start()
    {       if (th == null)
            {       th = new Thread(this);
                    th.start();
            }
    }
    public void stop()
    {       if (th != null)
            {       th.stop();
                    th = null;
            }
    }
    public void run()
{
   while (true)
   {
            if (suspend == 0)
        for (int i = 0; i < N.length; i++)
        {
                N[i]=new Node(); //array of references
```

```
                    N[i].currentPosition();
                repaint();

                    N[i].updateSpeedDirection();
                }
        else
        {
            n.clusters(numberOfNodes);
            repaint();
        }
                try
                {
                    Thread.sleep(2500);
                }
                catch(InterruptedException e)
                {   }

            }
    }

@Override
```

```
    public void paint(Graphics g)
{
            g.setColor(Color.black);
    g.fillRect(100,100,500,500);


      if((flag==1 )&&(suspend==0))
      {
                              g.setColor(Color.white);
              for (int i = 0; i < N.length; i++)
               {
                                    g.fillOval((int)N[i].xPosition,(int)N[i].yPosition, 2,2);
               }
          } // make clusters here
                  else
          {
             int index=-1;
             int[]  head= new int[ch.size()];
              int index1=-1;
             int[]  clust= new int[cm.size()];
             for(int i=0;i<heads.length;i++)
             {
```

```
    heads[i]=0;
 }
    for(int i=0;i<cluster.length;i++)
    {
        cluster[i]=0;
   }
    for(Integer i:ch)
 {
    head[++index]=i;
    t=head[index];
    heads[t]=1;
}
 for(Integer i:cm)
 {
  //  j=i.intValue();
    clust[++index1]=i;
  //  System.out.print(clust[index1]+" ");
    t=clust[index1];
    cluster[t]=1;

}
  for(int j=0;j<heads.length;j++)
```

```
        {
        if(heads[j]==1)
           System.out.print(j);
        }
    for (int i = 0; i < N.length; i++) {
                        if (heads[i]==1) {
                                g.setColor(Color.red);
                double angle=N[i].nodeAngle;
               /* int x,y,p,q;
                double r;
                x=(int) N[i].xPosition-1;
                y=(int) N[i].yPosition-1;
                r=x/Math.cos(angle * rad);
                p=(int)(r*Math.sin(angle*rad));
                q=(int)(r*Math.cos(angle*rad));
                g.drawOval(p,q,200,200);*/
                                g.drawOval((int) N[i].xPosition-51,(int) N[i].yPosition-
51,200, 200);
                                g.fillOval((int) N[i].xPosition,(int) N[i].yPosition, 2, 2);
            //   g.fillOval(x,y, 2, 2);
                        }
                    }
```

```
                    g.setColor(Color.green);
                    for (int i = 0; i < N.length; i++) {
                        if (cluster[i]== 1) {
                //  System.out.println("hi");
                        g.fillOval((int) N[i].xPosition,(int) N[i].yPosition,2,2);
                            }

                    }

            //System.out.println(xpos[i]);
                }//end else
           // }//end if

            n.distanceBetweenNodes(N, numberOfNodes);
            n.getEdgeMatrix(N, numberOfNodes);
            n.displayNeighbor(numberOfNodes);
            n.displayEgdeMatrix(numberOfNodes);
            n.clusters(numberOfNodes);
    }
public void update(Graphics g)
        {
            paint(g);
```

```
        }
public void actionPerformed(ActionEvent ae) {
        try {
                if (ae.getSource() == button)
       // if(ae.getActionCommand().equals("button"))

        {
          flag = 1;

                        numberOfNodes = Integer.parseInt(nodes.getText());
                N = new Node[numberOfNodes];
           // System.out.println("bb"+N.length);
           // run();
                        //heads = new double[numberOfNodes];
                        //cluster = new double[numberOfNodes][numberOfNodes];

           for (int i = 0; i < N.length; i++)
                N[i] = new Node();
                        th = new Thread(this);
           th.start();
                n.distanceBetweenNodes(N, numberOfNodes);
                n.getEdgeMatrix(N, numberOfNodes);
```

```
                    n.displayNeighbor(numberOfNodes);
                    n.displayEgdeMatrix(numberOfNodes);
                n.clusters(numberOfNodes);
                    }
                } catch (Exception e) {
                }

                if (ae.getSource() == pause)
              {
            suspend = 1;



                }

                if (ae.getSource() == resume) {
                    suspend = 0;
            //    flag=1;
                }
        }

class Node
        {
```

```
int incr=0;
        public int t;// time interval
        public Random rand = new Random();
        public int nodeId = 0;
        public double xPosition = 0.0; // x coordinate
        public double yPosition = 0.0; // y coordinate
        public double xPositionNew, yPositionNew;
        public double nodeVelocity = 0.0;
        public double xPositionPrev = 0.0, yPositionPrev = 0.0;
        public double nodeAngle; // movement direction
        public double range;
int distanceMatrix[][] = new int[100][100];// Distance between the nodes
    byte edgeMatrix[][] = new byte[100][100] ;
public int malnodeId;
 public int sybnodeId;

        //constructor
        public Node()
        {

       nodeId = incr++;
                xPosition = 100+rand.nextInt(400);
```

```
                yPosition = 100+rand.nextInt(400);
                t = 1;
                nodeVelocity = rand.nextInt(3);//When Velocity is Random
                nodeAngle = rand.nextInt(360);
                range = 200;
        }
  public void distanceBetweenNodes(Node N[], int numberofnodes)
{
        double p1 =0.0;
        double p2 = 0.0 ;
        double p = 0.0 ;
        for(int i = 0; i < numberofnodes; i++)
        {
                for(int j = 0; j < numberofnodes; j++)
                {
                        p1 = Math.pow( (N[i].xPosition - N[j].xPosition), 2);
                        p2 = Math.pow( (N[i].yPosition - N[j].yPosition), 2);
                        p = p1+p2 ;
                        distanceMatrix[i][j] = (int) Math.pow( p, 0.5);
                }
        }
}
```

```
public void getEdgeMatrix(Node N[], int numberofnodes)
{
	int i, j;
	for(i = 0; i < numberofnodes ; i++)
	{
		for(j = 0; j < numberofnodes; j++)
		{
			if(distanceMatrix[i][j] < 40)
			{
				edgeMatrix[i][j] = 1 ;
			}
			if(distanceMatrix[i][j] >40)
			{
				edgeMatrix[i][j] = 0;
			}
			if(i == j)
				edgeMatrix[i][j] = 0 ;
			if(edgeMatrix[i][j] == 1 && distanceMatrix[i][j] > N[i].range)
```

```
                    edgeMatrix[i][j] = 0;
                if(edgeMatrix[i][j] == 1 && distanceMatrix[i][j] > N[j].range)
                    edgeMatrix[i][j] = 0;

            }

        }

}
public void displayNeighbor(int numberofnodes)
{
        for(int i = 0; i < numberofnodes; i++)
        {
            System.out.print("\n The neighbors of " + i + " = " );
            for(int j = 0;j < numberofnodes; j++)
            {
                if(edgeMatrix[i][j] == 1)
                    System.out.print(j + " ");
            }
        }
```

```
                System.out.println();
                }
                public void displayEgdeMatrix(int numberofnodes)
                {
                        for(int i=0;i < numberofnodes;i++)
                        {
                                for(int j=0;j<numberofnodes;j++)
                                {
                                System.out.print(edgeMatrix[i][j]+" ");
                        }
                                System.out.println();
                }}
public void clusters(int numberofnodes)
{  int clusterhead[]=new int[numberofnodes];
   int clustermbr[]=new int[numberofnodes];
  int k=0;
  int count=0;
   for(int i=0;i<numberofnodes;i++)
   {
     clusterhead[0]=0;
  }
 for(int i=0;i<numberofnodes;i++)
```

```
 {
   clustermbr[0]=0;
}


cm.clear();
ch.clear();


  //System.out.println(set);
                    for(int i=0;i<numberofnodes;i++)
                    {
                              for(int j=0;j<numberofnodes;j++)
                              {          if(edgeMatrix[i][j]==1)
                         {
                             if ((cm.contains(i)) == false)
                                 // clusterhead[i]=i;
                                  // clustermbr[k++]=j;
                                 ch.add(i);
                                if(ch.contains(j)==false)
                                 cm.add(j);
                                  // clustermbr[k++]=j;
                             }}
   }
```

```
 if(ch.contains(nodeId))
    count++;
 System.out.println("count"+count);
  System.out.println(cm);
   System.out.println(ch);
   //cm.clear();
   //ch.clear()
     }


                public void updateSpeedDirection()
                {
                        for (int i = 0; i < 5; i++)
                        {
                                nodeVelocity = rand.nextInt(3);
                                nodeAngle = rand.nextInt(360);
                        }


                }
        } // End of inner Node class
} //End of outer class
```

Conclusion

We have reviewed the Mobile Ad-hoc Networks, their characteristics, advantages and disadvantages, the various routing algorithms and different mobility models for the simulation of Lowest ID clustering algorithm which organizes mobile ad hoc networks in a hierarchical way. We also reviewed main characteristics of the Lowest ID clustering algorithm that were applied on MANETs. We have successfully implemented the mobility of the node. All the work mentioned above involved real time data.

References, IEEE Format

Book

[1] Herbert Schildt, Complete Reference JAVA, 5th edition, Tata McGraw Hill.

[2] Charles E.Perkins, AD HOC Networking, Perason Education, 08-Jan-2001.

Research Paper

[3] A Review of Clustering Algorithms as Applied in MANETs by:Dr. Mohammad U. Bokhari, Hatem S. A. Hamatta, Shams Tabrez Siddigui.

[4] Survey of clustering algorithms for MANET By:Ratish Agarwal, Dr. Mahesh Motwani

Web References

[5] MANET Research Summary, http://.hulk.bu.edu/projects/adhoc/summary.html.

[6] www.academia.edu

Software

[7] Creately, for diagrams.