**Chapter 1**

**INTRODUCTION**

* 1. **General Introduction**

Research on Wireless Ad Hoc Networks has been ongoing for decades. The history of wireless ad hoc networks can be traced back to the Defense Advanced Research Project Agency (DAPRPA) packet radio networks (PRNet), which evolved into the survivable adaptive radio networks (SURAD) program. Ad hoc networks have played an important role in military applications and related research efforts, for example, the global mobile information systems (GloMo) program and the near-term digital radio (NTDR) program. Recent years have seen a new spate of industrial and commercial applications for wireless ad hoc networks, as viable communication equipment and portable computers become more compact and available.

Since their emergence in 1970’s, wireless networks have become increasingly popular in the communication industry. These networks provide mobile users with ubiquitous computing capability and information access regardless of the users’ location. There are currently two variations of mobile wireless networks: infrastructured and infrastructureless networks. The infrastructured networks have fixed and wired gateways or the fixed Base-Stations which are connected to other Base-Stations through wires. Each node is within the range of a Base-Station. A “Hand-off” occurs as mobile host travels out of range of one Base-Station and into the range of another and thus, mobile host is able to continue communication seamlessly throughout the network. Example applications of this type include wireless local area networks and Mobile Phone [1].

An example of other type of wireless network, (infrastructureless networks) is the Mobile Ad-hoc Network (MANET). These networks have no fixed routers, every node could be router. All nodes are capable of movement and can be connected dynamically in arbitrary manner. The responsibilities for organizing and controlling the network are distributed among the terminals themselves. The entire network is mobile, and the individual terminals are allowed to move freely. In this type of networks, some pairs of terminals may not be able to communicate directly with each other and have to rely on some terminals so that the messasges are delivered to their destinations. Such networks are often referred to as multi-hop or store-and forward networks. The nodes of these networks function as routers, which discover and maintain routes to other nodes in the networks. The nodes may be located in or on airplanes, ships, trucks, cars, perhaps even on people or very small devices. Mobile Ad-hoc Networks are used for disaster recovery, battlefield communications, and rescue operations when the wired network is not available. It can provide a feasible means for ground communications and information access [1].

Self-organization mechanisms can be found in every stretch of our every-day life. From an academic point of view, self-organization was first analyzed in biological systems. This research was soon extended to technical systems and engineering in general. Self-organization can be summarized as the interaction of multiple components on a common global objective. This collaborative work is done without centralized or decentralized control. Instead, the interaction is done using a local context, e.g. the direct environment that can be changed and adapted by each individual and, therefore, affects the behaviour of other individuals. The primary objectives of self-organization are scalability, reliability, and availability of systems composed of a huge number of subsystems.

In computer networks, self-organization is especially important in ad hoc networking because of the spontaneous interaction of multiple heterogeneous components over wireless radio connections without human interaction. This is especially the case in the areas of pervasive and ubiquitous computing. Eventually, self-organization is the only possible solution for many issues in this area but it definitely is not the universal remedy [2].

**1.2 Problem Statement**

MANETs are one of the most usable and reliable networks for communication with distinct applications. Hence it is necessary to evaluate the performance of the chosen MANET routing protocols: Energy Efficient Self-Organizing Algorithm (EESOA) and AODV. These protocols are being evaluated by simulating them in NS2. Network lifetime, average end to end delay, delivery ratio and convergence time are the chosen performance metrics.

**1.3 Objectives of Study**

The objectives of the study are as follows:

* Obtain an understanding of MANETs and the algorithms that facilitate routing in MANETs.
* Understand the working of AODV and EESOA routing algorithms and find a way to better the EESOA algorithm.
* Obtain a method to measure network lifetime, average end to end delay, delivery ratio and convergence time of algorithms simulated using ns2.
* Compare the performance of the proposed SO\_AODV (Self-Organizing AODV) with the Traditional AODV algorithm in terms of the above performance metrics through graphical representation plotted with the help of Microsoft Excel.
* Compare SO\_AODV against Traditional AODV to ascertain if the application of self-organizing techniques can improve upon Traditional AODV with respect to the metrics mentioned.
  1. **Scope of Study**

To implement a Self-Organizing Routing Protocol using Traditional AODV and to evaluate its performance against basic EESOA and Traditional AODV. This is done by simulating them using the discrete event network simulator ns2 to calculate network lifetime, average end to end delay, delivery ratio and convergence time.

* 1. **Organization of the Work**

This report consists of nine individual chapters. *Chapter one* contains a brief introduction of Mobile Ad-hoc Networks, routing issues in MANET and problem statement. *Chapter two* gives an overview of MANETs, the chosen algorithms, and tools used in the study. *Chapter three* gives the software requirement specification, consisting of the project perspective, functions, end users, etc. Chapter *four* deals with the proposed design of the system. *Chapter five* gives details of the implementation of the algorithm and performance comparison. *Chapter six* discusses the test cases on the basis of which evaluation is done. *Chapter seven* gives the conclusion of the project. *Chapter eight* provides the references used in the entire project. *Chapter nine* contains the user manual.

**Chapter 2**

# LITERATURE SURVEY

**2.1 Introduction**

**2.1.1 Mobile Ad Hoc Networks (MANETs)**

In the past few years, a rapid expansion in the field of mobile computing is seen due to the proliferation of inexpensive, widely available wireless devices. However, current devices, applications and protocols are solely focused on cellular or wireless local area networks (WLANs), not taking into account the great potential offered by mobile ad hoc networking. A mobile ad hoc network is an autonomous collection of mobile devices (laptops, smart phones, sensors, etc.) that communicate with each other over wireless links and cooperate in a distributed manner in order to provide the necessary network functionality in the absence of a fixed infrastructure. This type of network, operating as a stand-alone network or with one or multiple points of attachment to cellular networks or the Internet, paves the way for numerous new and exciting applications. Application scenarios include, but are not limited to: emergency and rescue operations, conference or campus settings, car networks, personal networking, etc. [3].

**2.1.2 Open Issues in MANETS**

The requirements of MANETS represent a spectrum of network challenges. During the last few years, almost every aspect of MANET has been explored to some level of detail. Yet, more questions have arisen than been answered. The major open problems are listed as:

* **Autonomous**- No centralized administration entity is available to manage the operation of the different mobile nodes.
* **Dynamic topology**- Nodes are mobile and can be connected dynamically in an arbitrary manner. Links of the network vary periodically and are based on the proximity of one node to another node.
* **Device discovery**- Identifying relevant newly moved in nodes and informing about their existence need dynamic update to facilitate automatic optimal route selection.
* **Bandwidth optimization**- Wireless links have significantly lower capacity than the wired links.
* **Limited resources** -Mobile nodes rely on battery power, which is a scarce resource. Also storage capacity and power are severely limited.
* **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.
* **Limited physical 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 should be considered.
* **Infrastructure-less and self-operated**- Self healing feature demands MANET should realign itself to blanket any node moving out of its range.
* **Poor Transmission Quality**- This is an inherent problem of wireless communication caused by several error sources that result in degradation of the received signal.
* **Ad hoc addressing**- Challenges in standard addressing scheme to be implemented.
* **Network configuration**- The whole MANET infrastructure is dynamic and is the reason for dynamic connection and disconnection of the variable links.
* **Topology maintenance**- Updating information of dynamic links among nodes in MANETs is a major challenge [4].

**2.1.3 Self-Organization**

Designers of distributed systems have recently turned to decentralized mechanisms to build large-scale systems for dynamic network environments, such as Mobile Ad Hoc Networks (MANETs), as traditional approaches to system design based on global knowledge or strict consensus among entities are no longer viable. However, the complexity of decentralized control increases rapidly with system size and dynamism, and researchers have turned to self-organization as a guiding principle to construct such systems. Self-organization offers the promise of providing new ways to build distributed systems from massive numbers of low-cost hosts that interact and adapt to produce properties such as self-management capabilities, robustness and adaptability to a dynamic environment.

The properties and constraints on mechanisms that are required for a system to be described as self-organizing with emergent properties have been, most clearly defined in the field of biology by Camazine et al. [2001]as “a process in which a pattern at the global level of a system emerges from the numerous interactions among lower-level components of the system.

Moreover, the rules specifying the interactions between the system’s components are executed using only local information, without reference to the global pattern.” This definition captures important aspects of self-organization such as autonomous components that take decisions using only local information, and how interactions between components cause system properties to emerge.

However, this definition does not say anything about how the system or its constituent components interact with the system’s environment, or how interaction with an external environment can be used to guide self-organization to produce macro-level structures [Prigogine and Stengers 1984]. There has been extensive research on self-organization in biological, social and physical systems [Camazine et al. 2001; Prigogine and Stengers 1984; Heylighen 2001; Albert and Barab´ asi 2002], producing a common set of concepts with which these systems can be discussed. These concepts are used by the project to propose an energy efficient self-organizing routing algorithm [5].

**2.2 Background**

**2.2.1 Energy Issues in MANETs**

The problem of energy efficiency in MANETs can be addressed at different layers. In recent years, many researchers have focused on the optimization of energy consumption of mobile nodes, from different points of view. Some of the proposed solutions try to adjust the transmission power of wireless nodes, other proposals tend to efficiently manage sleep state for the nodes (these solutions range from pure MAC-layer solutions (as the power management of 802.11) to solutions combining MAC and routing functionality). Finally, there are many proposals which try to define an energy efficient routing protocol, capable of routing data over the network and of saving the battery power of mobile nodes. Such proposals are often completely new, while others aim to add energy-aware functionalities to existing protocols (like AODV, DSR and OLSR).

The aim of energy-aware routing protocols is to reduce energy consumption in transmission of packets between a source and a destination, to avoid routing of packets through nodes with low residual energy, to optimize flooding of routing information over the network and to avoid interference and medium collisions. Many energy efficient routing protocol proposals were originally studied for sensor networks, where the limited energy of nodes is a strong constraint; in MANETs, however, the requirements are different: a node has generally more hardware resources (capable of better performance, but consuming more energy) and the protocol must preserve the resources of every node in the network (not only a subset of them, because each node can be, at any time, source or destination of data). A single node failure in sensor networks is usually unimportant if it does not lead to a loss of sensing and communication coverage; ad-hoc networks, instead, are oriented towards personal communication and the loss of connectivity to any node is significant[6].

**2.2.2 Simulation Environment**

Network Simulation (NS)-2 is used in as an evaluation tool. The NS-2 is a discrete event driven simulator [7] developedat UC Berkeley. The current version of the simulator is NS-2.34. The goal of ns2 is to support networking research and education. It is suitable for designing new protocols, comparing different protocols and traffic evaluations [8]. It is an object oriented simulation written in C++, with an OTcl interpreter as a frontend. NS uses two languages because simulator got to deal with two things:

* Detailed simulation of protocols which require a system programming language which can efficiently manipulate bytes, packet headers and implement algorithms,
* Research involving slightly varying parameters or quickly exploring a number of scenarios.

**2.2.2.1 Energy Model**

Energy Model, as implemented in ns, is a node attribute. The energy model represents level of energy in a mobile host. The energy model in a node has an initial value which is the level of energy the node has at the beginning of the simulation. This is known as initialEnergy\_. It also has a given energy usage for every packet it transmits and receives. Thes eare called txPower\_ and rxPower\_ [8].

NS2 uses the wireless interface which works like the 914MHZ Lucent WaveLAN DSSS radio interface [9]. Nodes use omni-directional antenna, and the transmission range is 250 meters.

**2.2.2.2 Fixed Transmission and Receiving Power**

Energy consumption only counts receiving and transmission. Thus, idle nodes do not consume energy. The power for transmission and receiving are *fixed* values, X Watt and Y Watt, respectively. Assume a packet p with time length t(p); when a node transmits p, its energy capacity will be decreased by Etx (p), where Etx (p)= X × t(p); when    a node receives p, its energy capacity will be decreased by Erx(p), where Erx(p) = Y × t(p). Thus, under this model, MTPR is the same as the minimum hop routing.

**2.3 Related Work**

Wireless ad-hoc networks require a special management because of their hardware and energy limitations compared with wired networks. The problem of constructing a backbone structure over wireless ad-hoc networks has been widely researched. The basic problem is to minimize the wireless backbone size by taking into consideration the node’s capabilities. Therefore, an efficient, self-organized, scalable, and fault-tolerant algorithm is proposed, where node connection in the backbone is minimal. The proposed algorithm groups the node elements in clusters by means of a self-organization strategy based in four possible roles for each node in the network: leader, gateway, member and bridge [12]. In order to show the performance of the algorithm, an implementation in the NS2 simulator is used. This simulation allows evaluating the structure built by the proposed algorithm. This allows showing that the proposed algorithm can obtain better results, in wireless backbone size and energy consumption when comparing with MWAC algorithm.

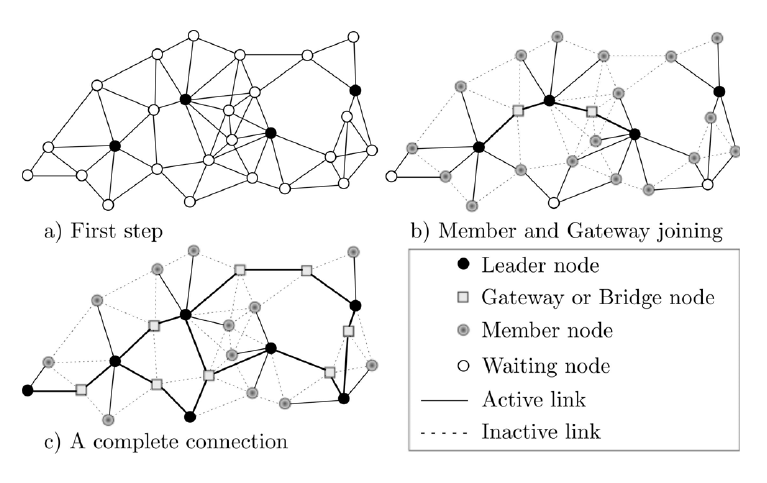


Figure 2.1 Virtual Backbone [12]

There exist many applications for wireless sensor networks. For example: surveillance tasks, widespread environmental sampling, monitoring the physical parameters of different farming zones [13], healthcare intelligent monitoring systems [14], security applications, etc. These applications have a need of wireless systems because the terrain is inhospitable; the cost of wired systems is high, or simply where physical placement is difficult. Examples of these applications are described in [15], in this survey different protocols and algorithms for sensor networks are analysed.

Wireless ad-hoc networks require a special management because of their hardware and energy limitations compared with wired networks. For example, they often are powered by batteries; they have limited memory size, computation power, etc. Therefore, memory usage and energy conservation are critical issues in these networks.

A big problem in these networks, is building the communication backbone. It is clear that, a sensor network connected in a wireless manner can only achieve a good coverage if all the nodes can communicate. A simple solution to this problem is the technique of pure flooding [16]. This approach guarantees with a high probability, that each non isolated node will receive the broadcast packet. The main disadvantage of this technique is that it consumes a large amount of bandwidth because of many redundant retransmissions are needed, which conveys to diminish the lifetime of the network [17]. Actually, not all the nodes need to retransmit the packet after receiving it.

The problem of constructing a backbone structure over wireless ad-hoc networks has been widely investigated. A great amount of existing research is devoted to backbone formation based on Connected Dominating Set (CDS) [18] [19].

A Common feature of previous approach is the high computational complexity requiring high energy consumption, and the lower capacity to maintain the connectivity of the network faced with node’s mobility. That is why it is desirable to design algorithms that are able to operate in a distributed manner; due to ad-hoc wireless networks are highly dynamic and autonomous. In addition, it is also desirable for these algorithms to implement an efficient use of its energy without jeopardizing communication between the nodes.

Jia-Liang et al. propose Low-Energy self-orGanizatiOn (LEGOS), an event-driven self-organization scheme without the use of periodical *Hello* messages [20]. The authors also provide a quick join procedure for newly arrived nodes as well as a solution for handling partition splitting and partition merge in the network. In order to solve partitions, a member must know the number of leaders in partitions, but it needs exchange of messages and consumes energy. LEGOS also only works for the case of sporadic nodes arrival.

An approximation by using the Steiner Tree Problem with Minimum number of Steiner Points (STP-MSP) can be found in [21]. The first step is to select a dominating set of the nodes and the second step is to put some relay nodes into the network to provide the connectivity of selected backbone nodes. It is a good work, but the global approach is disadvantageous and quite complex.

For ad-hoc networks, this project shows interest in prolonging the lifetime of the network which requires to conserve energy as much as possible, because overall performance becomes highly dependent on the energy efficiency of the algorithm.

In this paper, an Energy-Efficient Self-Organized Algorithm for wireless ad-hoc networks (EESOA) has been presented. The algorithm takes into account the nodes capabilities in order to build a minimum wireless backbone based in the concepts of inhibition and node roles.

Because the proposed algorithm is fully-distributed, no node has the whole knowledge of the virtual backbone, i.e., each node is only aware of its neighbours within one hop. A way to overcome the problem of lack of global knowledge is the use of group-based algorithms [22].

The advantages of a group-based algorithm are:

* reducing routing table size,
* reducing the redundancy of exchanged messages,
* reducing the energy consumption, and
* extending the network lifetime

In the group-based strategy, deployed sensor nodes are partitioned into a number of single-hop groups and each leader performs intra-group local communication.

*A. Network Model*

The wireless network is described by an undirected communication graph G (V, E), where the set V represents the wireless devices in the network and the set of edges E represents bidirectional communication channels operating between neighbour nodes. Between any two nodes u and v, there will be an edge (u, v) if the transmission from node u is received by the node v with a signal strength greater than RXthreshold.

In classical approaches for choosing the leader in the group, the lowest identifier is often used. However, such a scheme is inefficient since this criterion cannot reflect the aptitude of a node to act as a backbone member during a long time, creating a stable structure. Different metrics can be used to determine the more suitable nodes. In this scheme, the characteristics of nodes are taken into account when constructing a backbone, where the characteristics could be namely the node’s degree, the remaining energy, the link quality, the communication capacity, the processing power, etc. In this case for simplicity’s sake, only the node’s degree and the remaining energy will be used to calculate the weight of the node.

Under this model, it is assumed that:

* Each node has a unique identifier ID e.g. its IP address.
* Each node only knows the information of neighbours to one-hop.
* The nodes can move, arrive, or leave the network.
* Each node can adjust its transmission power.
* The agent can use overhearing to obtain important information for reducing the message transmission.
* Nodes keep: a neighbours table and a weight that is equal to the product between the number of neighbours and the residual energy units (wu = Nu∗eu).
* The nodes do not know their geographical positions.
* The wireless nodes are placed in the 2-dimensional Euclidean plane (it also works correctly if agents are located in three-dimensional space).

*B. Radio Propagation Model*

There are factors that influence the communication in the network. For example, the background noise produced by signals naturally present in the environment. In order to make a more realistic environment the two-ray ground reflection and free space propagation models are used [23]. These models are used to predict the received signal power of each packet. The two-ray ground reflection model considers both the direct path and a ground reflection path. This model gives more accurate prediction at a long distance than the free space model. However, the two-ray model does not give a good result for a short distance due to the oscillation caused by the constructive and destructive combination of the two rays. Instead, the free space model is used when distance is small [24].

Each node can determine the specific power levels it needs to reach each of its neighbours, by measuring the receiving power of *hello* messages. This approach can be applied to any propagation channel model. In this manner, a node uses just the necessary energy. In order to calculate the optimal transmission power, the following equation is used [25]:

RTXmin = RXthreshold∗PTX (1)

PRX

Where,

* RTXmin is the necessary transmission power to communicate with a leader agent.
* RXthreshold is the minimum signal strength to receive the packet. If one packet is received whose signal strength is stronger than RXthreshold, the packet is received correctly, otherwise it is discarded.
* PTX is the transmitted power.
* PRX is the received power.

AN ENERGY-EFFICIENT SELF-ORGANIZED ALGORITHM

*A. Self-Organization Strategy*

Cluster-based control structures allow a more efficient use of resources. A hierarchical view of the created network through clustering decreases the computational complexity of the underlying network, especially in sensors networks which are expected to consist of large amounts of individual nodes. In this way, clustered structures can make a highly mobile topology appears more static, and thus mitigate the effects of mobility. Therefore, efficient, self-adaptive, scalable, fault-tolerant and robust algorithms must have these important characteristics, which should be taken into account for the design goal. To meet these requirements, self-organization algorithms appear to be the most promising approach where the network nodes are self-organized in a distributed or a localized way as virtual network structure. Self-organization can be defined as a process in which the pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the systems components are executed using only local information, without reference to the global pattern [26].

This does not assume any synchronization in the network; each node independently decides its state using only local information. Self-organization is an emerging paradigm which can be used in either mobile ad-hoc or wireless sensor networking problems [27].

The number of nodes in the backbone should be the smallest possible and, at the same time, the backbone should be node-failure tolerant. Backbone nodes are in charge of relaying packets and thus are more likely to drain their battery; in that case, less backbone nodes means that fewer nodes are intensively used.

*B. Algorithm Description*

In the proposed algorithm, the generated structure is based in groups; it is composed by four different roles (leader, gateway, member and bridge); there is only one leader per group. Leaders are in charge of doing all the communication inside the group they coordinate. Thus it is necessary to choose the more suitable nodes to play the leader role. The gateway node makes the communication between the leaders; it can communicate more than two leaders. In this way the virtual backbone is formed with leaders and gateways. Then, the communication is carried out by mean of these nodes. The member node only takes care of its own tasks, there can be zero or more members connected to one leader.

Because the algorithm is localized, it is impossible to determine a whole connection by using only leaders and gateways as backbone. When a backbone does not connect the whole network, because of communication between members is not defined; this situation is called segmentation. By using bridge nodes, the different network segments can be connected.

A *hello* message sent by a leader will inhibit all nodes that receive it. Whenever a new node arrives to the network, if it receives a *hello* message from a leader, the node will get inhibition and become member. If no *hello* message from leader is received, the node starts to discover the neighbourhood and follows the algorithm.

A node that is not inhibited by a neighbour will get the leader role as long as it has the greatest weight in the group, with ties broken by the node *id*. A node will wait if it has an uninhibited neighbour either with a greater weight or with the same weight but with a greater *id*. After a node u takes the role of leader, it broadcast an inhibiting message. After receiving this message, all the u's neighbours will be inhibited. A node will switch to gateway as long as it is inhibited by more than one leader. If a node is inhibited by only one leader, then it takes the member role. This behaviour is described in Algorithm 1 [12].

The segmentation problem can be solved by using bridge nodes. For the purpose of connecting different segments of the network, a member node u will get the bridge role if it fulfils the following conditions:

* u has a neighbour v with the role of member or bridge and the v's group-*id* is different of u's group-*id*.
* there is not a bridge node in the neighbourhood of u.
* there are not gateway nodes w connecting a leader node with the *id* equal to the v's group-id.

Bridge nodes actually connect the different segments of the network because member nodes are all the time in the edge of the segments due to the lower weight; member nodes are those that turn to bridge. In this way, bridge nodes connect segments where possible. These steps are described in Algorithm 2[12].

*C. Neighbours Table Management*

Neighbourhood management essentially has three operations: insertion, erasure, and updating. Each node broadcast *hello* messages to discover the neighbourhood. Since all these nodes wake up at the same time, EESOA attempts to schedule the times at which nodes send their broadcast messages, so that not all nodes send their messages at once.

This scheduling helps avoid collisions. In order to do this, a random number between 0 and ξ is computed when the node starts the algorithm. Afterwards, the node will send *hello* messages at intervals of T.

Each node keeps a frequency count for each entry in its own table. On insertion, a node is updated by establishing its count at ρ. As the messages of the local node neighbourhood reach the node, new discovered nodes will be inserted in the table as a neighbour, if there is bidirectional communication between them. This means that the insertion is done when the node receives an acknowledgement from another node, which received a *hello* message. In every cycle, the count of all entries is decremented by one. Each time that the node receives a message from a node in the neighbour table, it will reset the neighbour value count to ρ. As time passes, if the count of some neighbour reach zero, it will be dropped by erasing its entry in the neighbour node table.

At the beginning, each node starts the algorithm when the neighbours table has not inserted a new node for a period of σ ∗T rounds.

*D. Increasing Network lifetime*

*1) Redundant Gateways:* Most of the time more than one gateway connects the same leaders. It is obvious that duplicate gateways will affect the overall energy consumption, and consequently the network lifetime is reduced. In order to make the communication among a set of leaders it is necessary only one gateway. When the gateway node depletes its energy, it is no longer able to communicate its leaders. In this way, another gateway must be available to do the communication.

Let w be a gateway, if the leader set of w is subset of other gateway’s leader set, inside the neighbourhood, will switch to member node. If the w's set of leaders is exactly the same as other gateway’s leader set, and idw is lower than the other one, then will switch to member node as well (see Algorithm 3 [12]).

Gateway nodes that left its role, i.e., a gateway that turned to member due to the redundancy, will join to the leader with the stronger signal strength max {PRXv|v ∈Nl (w)}.

Gateway nodes that switch to member, cannot take the role of bridge.

*2) Topology Control:* In order to generate a network with the desired properties, topology control is used. It is about tuning the range assignment to optimize the energy consumption without changing the network structure.

The node w having the gateway role will adjust the transmission power. It will compute the necessary power by using the Equation 1, and it will choose the maximum value among the leaders (max {RTXv|v ∈Nl (w)}). In the same way, members can compute the optimal power transmission by using the reception threshold of its leader.

In order to change the transmission power, the nodes must wait, because if the nodes’ mobility is too high, a bad structure can be formed. Whenever a gateway or member node keeps the role for an elapsed time κ, it will execute the procedure of adjusting the retransmission power. In this way, when a node decide to reduce the transmission power the network structure is not affected.

*3) Broadcast:* Since communications dominates the energy consumption, it is necessary to avoid send messages as much as possible. The time interval T between two broadcasts of *hello* messages will depend on the environment. After the neighbours table does not insert a new node for a while σ ∗T, the broadcast transmission period will be increased by a constant value of Δ once. Therefore, the node will transmit a *hello* message in the interval T +Δ. Thus, the energy consumption is minimized. It is worth mentioning that it does not affect the finding of new neighbours since a node that joins the network ask for neighbours.

Different events can change the neighbours table such as: joining of a new node, a node leave the network, failure of nodes or simply the node dies because its energy is over; even a more likely situation is the movement of nodes inside the network. If the neighbour table changes the parameters, the broadcast transmission must be restarted.

When the node changes the role, it restarts the transmission power and broadcast transmission interval, and gets the initial values. [12]

*4) Overhearing:* Communication through overhearing comes at a cost, since the node listens for packets that are not necessarily addressed to it, but in many cases the packets are received anyway [28]. By taking advantage of overhearing, for each incoming packet the source is considered for updating. If the source is already in the neighbour table, an updating operation may be performed to keep up-to-date the neighbours’ information. In this way, the number of messages is reduced. As a consequence the algorithm obtains energy saving [29].

*5) Reorganization:* Always some changes are happened in the environment due to new obstacles, movement of nodes, starvation or failure of a node, an increase in the loss rate of a link, etc. Therefore, when a node dies, reorganization must be carried out. A complex situation appears when leaders die or move because they are in charge of doing all the communication in the group.

**Chapter 3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 Introduction**

**3.1.1 Purpose**

The purpose of this project is to enable self-organizing behaviour in MANETs by proposing an algorithm which groups the node elements in clusters by means of a self-organization strategy based on three possible roles for each node in the network: leader, member and bridge by assigning weights to the nodes based on the node’s characteristics, specifically degree of the node, remaining energy and hence achieving energy conservation of the nodes.

**3.1.2Scope of the Project**

The scope of the project includes performance analysis of SO\_AODV routing protocol, to enable self-organizing behaviour among the nodes in MANETs and achieve the objectives of coordination and collaboration on a global level.

Packet delivery ratio, network lifetime, convergence time and average end to end delay are taken as performance metrics to evaluate and compare SO\_AODV, EESOA and AODV routing protocols.

**3.1.3 Definitions, Acronyms and Abbreviations**

**MANETs:** Mobile Ad Hoc Networks are a type of wireless ad hoc networks, and is a self-configuring network of mobile devices connected by any number of wireless links. Every device in a MANET is also a router because it is required to forward traffic unrelated to its own use.

**AODV :** Ad-hoc On-Demand Distance Vector (AODV) is a reactive protocol: the routes are created only when they are needed. It uses traditional routing tables, one entry per destination, and sequence numbers to determine whether routing information is up-to-date and to prevent routing loops.

**EESOA:** Energy-Efficient Self-Organizing Algorithm (EESOA) takes into account the nodes capabilities in order to build a minimum wireless backbone based in the concepts of inhibition and node roles.

**Self-Organization:** can be summarized as the interaction of multiple components on a common global objective.

**ns2:** NS or the Network Simulator is a discrete event simulator targeted at networking research. NS provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks.

**Packet delivery ratio:** This is the fraction of correctly delivered data packets versus sent packets.

Packet delivery ratio = Packets received

Packets sent

**Convergence Time:** Has been defined as the time between detection of an interface being down, and the time when the new routing information is available.

**Network Lifetime:** It is defined as the time from the start of the simulation till when the first node in the network dies.

**Average end to end delay:** Time taken for packet to be transmitted from source to destination.

Dend – end = N [dtrans + dprop + dproc]

Where,

dend – end is average end to end delay

dtrans is transmission delay

dprop is propagation delay

dproc is processing delay

(Queuing delays are neglected)

**3.2 General Description**

**3.2.1 Project Perspective**

The project consists of the ns2 network simulator which includes generators for creating randomized network scenarios and a Perl script to measure packet delivery ratio, network lifetime, convergence time and average end to end delay using the output of the simulation. It also includes a user interface created using Perl, which can invoke NAM a network animator provided with ns2, to visualize the simulations.

This project deals with the development of the SO\_AODV routing algorithm and evaluation of its performance for MANETs, using packet delivery ratio, network lifetime, convergence time and average end to end delay as metrics. The result of the evaluation can be used to state which algorithm is more energy efficient compared to traditional AODV algorithm.

**3.2.2 Project Functions**

**AODV**

**EESOA**

**Network Scenario (generated using Ns2)**

**SO\_AODV**

**Calculate average end to end delay, network lifetime, convergence time, and delivery ratio**

Figure 3.1 Project Functions Overview

**3.2.3 End Users**

The possible end user is a researcher trying to implement the routing algorithms for MANETs using NS2 simulator. Such a developer will want to analyze how efficient the algorithms are and improve their performance.

**3.2.4 General constraints**

The developed system is constrained to work only on MANET algorithms implemented using the Ns2 network simulator or Glomosim or Qualnet. It depends on the Linux system environment, and needs the Perl compiler to be present.

**3.2.5 Assumptions and Dependencies**

The system assumes that it is being run in a Linux environment which contains the Perl complier. It also assumes that the network simulator Ns2 or Glomosim or Qualnet has been used to simulate the algorithm, and the trace file generation by Ns2 is in the new format.

Packet delivery ratio, network lifetime, convergence time and average end to end delay of the nodes has a strong dependency on the output of the Ns2 simulation, as certain assumptions are made about the format of the output. As mentioned earlier, the new trace format needs to be used.

This project, choses to use wireless channel, TwoRayGround as radio-propagation model, WirelessPhy for network interface type, MAC 802.11, DropTail and Priority Queue as interface queue type, OmiAnteena for antenna model, maximum number of packets is set to 50, over link layer.

**3.3. Specific Requirements**

**3.3.1 Functional Requirements**

First, the project presents the functional capabilities of the system through use cases. Then, it outlines the possible inputs to the system and describe the outputs generated by the system in each use case.

Design of SO\_AODV Routing Protocol

<<include>>

<<include>>

<<Researcher>>

Figure 3.2 Use Case Diagram for the Development of the SO\_AODV Routing Protocol

Evaluate Performance of Modified EESOA Routing Protocol

<<include>>

<<include>>

<<include>>

<<include>>

<<include>>

<<include>>>

Figure 3.3 Use Case Diagram to Evaluate the Performance of SO\_AODV Routing Protocol

***Use Cases related to the simulation:***

**Use Case 1: Run Simulation**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Protocol is simulated using Ns2, it extends the generate scenario and specify routing protocol use cases.

**Use Case 2: Generate Scenario**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Here the scenario for the simulation is generated such as number of nodes and pause time to be given.

**Use Case 3: Routing Protocol**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Here the routing protocol is specified.

***Use cases related to the measurements:***

**Use Case 4: Parse Trace File**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Parses the trace file.

**Use Case 5: Network Lifetime**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Measures the network lifetime from trace file.

**Use Case 6: Packet Delivery Ratio**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Measures the packet delivery ratio from the trace file.

**Use Case 7: Convergence Time**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Measures the Convergence time from trace file.

**Use Case 8: End to end delay**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Measures the end to end delay from trace file

**Use Case 9: Compare**

**Primary Actor:** Researcher

**Precondition:** User has decided on the scenarios to test SO\_AODV on.

**Specification:** Use to compare the routing protocols using metrics such as packet delivery ratio, network lifetime, average end to end delay and convergence time.

Which output should be produced from which inputs is described with the help of flowchart in the figure 3.4.

EESOA

AODV

Network Scenario

SO\_AODV Algorithm

Protocol not Implemented in Ns2

Is Implemented in Ns2

Nam File

Run Simulation in Ns2

Trace File

Packet Delivery Ratio

Convergence Time

Average End to end delay

Network Lifetime

Inform User

Is Trace File in Correct Format

Compare SO\_AODV with Traditional AODV and EESOA

Figure 3.4 Functional Behaviour of the System

The inputs to the system are the routing algorithm and the network scenario. A valid routing algorithm is said to be any algorithm that has been implemented in Ns2. If an algorithm that has not been implemented in Ns2 is given, the simulation exits with an error. The network scenario, however, can take on any value.

After accepting the network scenario parameters, the system generates the scenario using generators available with Ns2. The generated scenarios and the routing algorithms are then given as inputs to the network simulator Ns2, which runs in the simulation. The result of this simulation is a trace file, which logs all the selected events in chronological order. The events to be logged are specified by the system by default. It is this file that enables the calculation of convergence time, end to end delay, network lifetime and packet delivery ratio. The simulation gives the Nam file as output, which can be given to the Network Animator, a simulation visualization tool that comes with Ns2.

The trace file is read to calculate the convergence time, end to end delay, network lifetime and packet delivery ratio. The trace file needs to be in a particular format. Hence a check is done to see if the file is in an appropriate format. If it is not, the user is informed of this. Also, the method to generate the trace file in the required format is suggested, so that the user can make this change and continue with measuring the metrics.

After analyzing the trace file, the tool outputs the packet delivery ratio, convergence time, average end to end delay and network lifetime of the specified algorithm for a specified scenario. This process can be repeated for different scenarios and the result can be tabulated or graphed and analyzed, to find the strengths and weaknesses of the developed algorithm.

**3.3.2 Software Requirements**

Linux operating system – fedora, red hat, Ubuntu 7.10 or higher

Network simulator Ns2 – Ns2.34

Perl Interpreter

Graph Plotting Tool - Microsoft Office Excel

**3.3.3 Hardware Requirements**

Pentium 4 or higher processor

1GB RAM

512 MB memory (minimum)

**3.4 Interface Requirements**

**3.4.1 User Interface**

The user interface in this project is a command line interpreter. This interface is responsible for:

* Taking inputs from the user
* Checking inputs for validity
* Passing inputs for modules for computation
* Displaying error messages in readable and easy understandable format
* Suggesting methods for correcting the errors
* Displaying output in a concise manner

The user interface needs to be simply and easy to use.

**3.5 Performance Requirements**

In this project, the static requirements are: one terminal, two scenario files as input to Ns2- one connection pattern and one mobility scenario, one trace file in new trace format as input to Perl file calculating packet delivery ratio, network lifetime, end to end delay and convergence time, and the same Perl file for the user interface.

The dynamic requirements of this project are: an upper limit of 30 seconds was placed on the response time of the tool. Here the simulation time depends on the simulation tool selected.

**3.6 General Requirements**

The system should be able to measure convergence time, packet delivery ratio, network lifetime and end to end delay.

**Chapter 4**

**SYSTEM DESIGN**

**4.1 Introduction and Design Overview**

The aim is to alter EESOA to take into account the fact that a cluster is rendered useless if a leader fails in the case of EESOA, until a new leader is elected. SO\_AODV in this case uses the basic route discovery proposed by AODV.

The design of SO\_AODV involves using the degree of the node and the remaining energy. The product of these two attributes is used as a weight to select the leader. After designing SO\_AODV, its performance evaluation can be done using the metrics: network lifetime, average end to end delay, delivery ratio and convergence time. These metrics are used to compare the algorithm developed with traditional AODV and EESOA.

The rest of the document is organized as follows. In section 4.2, describes the system architecture chosen, and the system interface description. 4.3 gives detailed explanation of the various components involved in the system. 4.4 discusses about the user interface. Section 4.5 provides the test plan. The test plan describes the features to be tested, the testing tool and environment used.

**4.2 System Architecture Design**

The whole system is viewed as a layered architecture, and is divided into multiple sub-systems which are related to and interact with each other to perform the necessary functional requirements.

Layered architecture has two main advantages:

1. Maintainability: Designed using fine grained self-contained component that may readily be changed. Each layer depends on the immediately above layer and does not affect the higher layers.

2. Security: Components which require most amount of protection are embedded in the lower most layers and each of the higher layers protect the lower layers, hence layered architecture provides maximum security to the components that require it.

**4.2.1 Chosen System Architecture**

The layered architecture reflects the stages of processing in the system namely implementation of SO\_AODV protocol in the lowest layer, ns2 simulation and measurement of the performance metrics namely average end to end delay, network lifetime, convergence time and packet delivery ratio in the second layer and interface in the first layer as shown in the figure 4.1.

**<< Subsystem >>**

**Node**

**<< Subsystem >>**

**Processing Layer**

**Ns2 Simulation**

**Performance metrics measure:** Average end to end delay, network lifetime, convergence time and packet delivery ratio

**<<Subsystem>>**

**Self-Organization Protocol + Routing Protocol**

**SO\_AODV**

Figure 4.1 Layered Architecture

A layered architecture is appropriate in this case because each stage relies only on the processing of the previous stage for its operation. SO\_AODV can be viewed as the data collection layer. Measurement of metrics: average end to end delay, network lifetime, convergence time and packet delivery ratio and ns2 simulation is present in processing layer which takes them from the trace file generated from simulation of the routing protocol. Interface layer is the user interface layer which takes the input from the command line and displays the output in the form of graphs, trace file and the network animator NAM.

**4.2.2 System Interface Description**

The whole system is divided into multiple modules (sub-systems). The system interface gives the top level design of the system which describes the modules within the system and the interactions between them. The modules together satisfy the functional requirements of the system.

In this case, the modules to be included are as follows:

1. Module 1 to run the user interface and also calculate the metrics – Perl script
2. Module 2 to run the simulation – ns2
3. Module 3 for SO\_AODV algorithm

These modules are again an agglomeration of modules, as described below:

1. A module to run user interface and to calculate average end to end delay, network lifetime, convergence time and packet delivery ratio – perl script.
   1. A module to accept user input through command line interface.
   2. A module to display the output in the form of a graph – Microsoft Office Excel.
   3. A module to visualize the working of the network – Nam.
   4. A module to accept input file and check for compatibility.
   5. A module to parse the input file and measure average end to end delay.
   6. A module to parse the input file and measure convergence time.
   7. A module to parse the input file and measure average packet delivery ratio.
   8. A module to parse the input file and measure the network lifetime.
   9. A module to record details of parsing input and output file and give a summary output to user.
2. A module to run the simulation – ns2.
   1. A module to generate network scenario.
      1. A module to generate the connection pattern between the nodes.
      2. A module to generate mobility scenario of the nodes.
   2. A module to specify which events are to be traced in the trace file, which Input files to use, and to generate the file to be used for visualizing the simulation–tcl file.
3. A module for SO\_AODV algorithm.
   1. A module to perform efficient route discovery.
   2. A module to appoint roles to each node depending on the weight calculated (leader, member and bridge).

The interaction of the modules is as described in the figure 4.3. The command line interface to the user prompts the user to enter the network characteristics. These inputs are then passed on to the module responsible for generating the network scenario. This module uses two files - one describing the connection pattern within the network, and another describing the movement of nodes. These files are given as inputs to the tcl file, which specifies what events are to be recorded in the trace file, and also generates the file to be used as input to Nam. The Nam file is passed to the network animator, which allows the user to graphically visualize the simulation which has just taken place.

**<<Subsystem>>**

**Measure average end to end delay, packet delivery ratio, convergence time and network lifetime**

**Accept Input and check for compatibility**

**Measure Network lifetime**

**Measure Average packet delivery ratio**

**Measure Average end to end delay**

**User**

**Interface**

**Graph Plotting Tool**

**Command Line Input**

**Measure Convergence Time**

**<<Subsystem>>**

**Ns2 Simulation**

**Self-Organizing\_AODV**

**Generate network scenario**

**Tcl File – Specify what to trace in the trace file**

**Network Animator (nam) – Generate nam file**

Figure 4.2 Subsystem Design- Modules and their Interaction

The trace file is used to measure the average end to end delay, network lifetime, convergence time and packet delivery ratio of the SO\_AODV protocol under specified network scenarios. The average end to end delay, network lifetime, convergence time and packet delivery ratio are given as the summary output to the user. Interaction between the modules can be better understood through the activity diagram as shown in figure 4.4.

**Ns2 simulation Measure of Performance Metrics Interface**

**Command line input**

**Check user Input**

**Check trace file for compatibility**

**Run Simulation**

**Pass inputs to tcl file**

**Generate Scenario**

**Implement SO\_AODV**

**Notify user of error**

**Measure delivery ratio average end to end delay, convergence time and network lifetime**

**Graph Display**

**Pass nam file to nam**

**Not Compatible**

**Compatible**

Figure 4.3 Activity Diagram of the System

**Development and evaluation of SO\_AODV**

<<**subsystem>>**

**Interface**

<<**subsystem>>**

**NS2 Simulation and Measure of average end to end delay, network lifetime, convergence time and packet delivery ratio**

<<**subsystem>>**

**SO\_AODV**

**Manages all external communication**

**Simulate the protocol, check trace file for compatibility and measure average end to end delay, network lifetime, convergence time and packet delivery ratio**

**SO\_AODV is present in NS2 simulator which is a combination of AODV and EESOA**

Figure 4.4 Architecture of the System

**4.3 Detailed Description of Components**

The system consists of three main components, the user interface, the ns2 simulator and the SO\_AODV algorithm.

**4.3.1 SO\_AODV**

AODV protocol is one of the most energy efficient protocols among all other routing protocols for MANETs but it still does not take into account the network lifetime in case of applications where node energy is critical.

The design of SO\_AODV involves using the degree of the node and the remaining energy.

Table 4.1 Notations

|  |  |
| --- | --- |
| Notations | Meaning |
| Wi | Weight of node vi |
| Di | Degree of node vi |
| Ei(t) | Remaining energy of node vi at time t |

Explanation of notations for the following formulae is given in table 4.1. The weight of the node is calculated as follows:

Wi = Di \* Ei (t)

To incorporate these two metrics AODV algorithm is altered such that the Hello packets are enabled to ensure neighbour management. The reply header that is appended to each Hello packet contains fields that provide a measure of the remaining energy, whether the source node is inhibited or not, role of the source node and degree of the source node. Route request packet is not altered. The AODV reply header is shown in table 4.3.

Table 4.2 Extended Reply Header

|  |  |  |
| --- | --- | --- |
| Type | Reserved | Hop Count |
| RREQ ID | | |
| Destination IP Address | | |
| Destination Sequence Number | | |
| Originator IP Address | | |
| Lifetime | | |
| Timestamp | | |
| **Remaining Energy** | | |
| **Whether the source node is inhibited or not** | | |
| **Role of Node** | | |
| **Degree of Node** | | |

The neighbour tables in each of the nodes are altered. The following entries have been added to the existing neighbor table entries:

* Whether the neighbor is inhibited or not
* Neighbour’s role
* Neighbour’s degree

**4.3.2 Simulator Chosen**

A discrete event simulator with wireless and the mobility capabilities built into it is required. The network simulators ns2, ns3 and Qualnet are the ones that are normally used.

NS3 does not have any energy framework or the ability to incorporate energy into network simulations. It does not provide any support for modeling energy consumption or energy sources. On the other hand NS2 has an energy framework and allows us to incorporate energy into network simulations. Energy is the area of focus and hence Ns2 is chosen.

Qualnet is commercial software. As it is not freely available there is lesser support (code samples etc.) available on the web. NS2 is an open source software, which supports mobility and wireless transmission, and has implemented several routing protocols for use. Since it has in built protocols it is easier to manipulate them and get the algorithm accepted in the research community. For the above reasons NS2 is chosen to run these simulations.

**4.3.3 Network Scenarios Chosen**

**4.3.3.1 Mobility Model**

The Random Waypoint Model is chosen as the mobility model for the simulation. According to this model, a node waits in its current position for duration of time specified by pause time. At the expiration of this pause time, it chooses a destination randomly, and moves to it with a speed chosen from the uniform distribution [0, max\_speed]. This process is repeated until the end of the simulation.

**4.3.3.2 Scenario Generators**

It is decided that the network scenarios should be generated using the generators provided with ns2. Ns2 provides two generators created by CUM Monarch. They are: setdest and cbrgen.

**4.3.3.2.1 Setdest**

To create Mobile node Movement Scenario files, the command line that needs to be run under directory: ns-allinone-2.34/ns-2.34/indep-utils/cmu-scen-gen/setdest:

./setdest [-n num\_of\_nodes] [-p pausetime] [-M maxspeed] [-t simtime] [-x maxx] [-y maxy] > [output-file]

Setdest is a generator used to create the mobility scenario. That is, it specifies the movement of each node for the duration of the simulation. The parameters to be given to the generator are:

For example,

$./setdest –n 50 –p 10 –M 10.0 –t 250 –x 500 –y 500 >filename

Creates the mobility scenario for 50 nodes with a pause time of 10s and a maximum speed of 10m/s. The simulation time is given to be 250s, and the topology is a 500m X 500m flatgrid. The output is saved in a file called filename.

**4.3.3.2.2 Cbrgen.tcl**

To create TCP traffic scenario files, under directory: ns-allinone-2.35/bin, run:

ns ../ns-2.35/indep-utils/cmu-scen-gen/cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate packet/second for one connection] > [output-file]

Cbrgen.tcl is a generator used to create the connection pattern for the simulation. It specifies the pairs of nodes between which traffic is sent. It is capable of generating two types of traffic: tcp and cbr (constant bit rate). For this project’s simulation, TCP traffic is chosen.

For example,

$ns cbrgen.tcl –type tcp –nn 50 –mc 25 –seed 0.0 > filename

Creates the connection pattern for 50 nodes. TCP traffic is created between a maximum of 25 pairs of nodes, and output is stored in filename.

**4.3.3.3 Specific scenarios**

The parameters that define the MANET scenario are node density and node mobility. In this project’s simulations, the node density can be varied by varying the number of nodes, while the mobility can be varied by varying the pause time. That is, if a node pauses for a longer time at each waypoint, its overall mobility is less, while pausing for a very short time, say 1s, means its mobility is very high.

Number of nodes = 10 and pause time = 3 would represent one end of the spectrum with low node density and high mobility, while number of nodes = 50 and pause time = 10 would be the other end of the spectrum with high node density and low mobility.

**4.3.4 Metrics chosen for Comparison**

**Average packet delivery ratio:** The number of packets received by the destination node divided by the number of packets transmitted by the source node.

**Network Lifetime:** Is defined as the time from the start of the simulation to when the first node is dead.

**Convergence Time:** Has been defined as the time between detection of an interface being down, and the time when the new routing information is available.

**Average end to end delay:** Time taken for packet to be transmitted from source to destination.

**4.4 User Interface Design**

**4.4.1 Description of the User Interface**

The user’s perspective:

**Perl UI – perl script takes input**

**Simulation – Otcl Script**

**OTcl Interpreter**

**C++ Libraries**

**Input**

**NS - 2**

**Output**

**Nam visualization**

**Trace File**

**Perl UI – prints performance metrics on screen**

**SO\_AODV Routing Protocol**

Figure 4.5 User Interface of the System

Block diagram of the user interface is shown in figure 4.11. The input to the simulator is given through the perl script, these parameters are passed to the tcl file which processes it and generates the Nam and trace file as output. The values in the trace file are used to calculate the performance metrics. The Nam file is sent to the Nam visualizer which displays the working of the scenario. Microsoft Office Excel is a tool used to display the performance of SO\_AODV against Traditional AODV and EESOA.

**4.4.2 Perl**

Perl is a high-level, general-purpose, interpreted, dynamic programming language. Perl was originally developed by Larry Wall, a linguist working as a systems administrator for NASA, in 1987, as a general purpose Unix scripting language to make report processing easier. Since then, it has undergone many changes and revisions and become widely popular among programmers. Larry Wall continues to oversee development of the core language, and its upcoming version, Perl 6.

Perl borrows features from other programming languages including C, shell scripting (sh), AWK, and sed. The language provides powerful text processing facilities without the arbitrary data length limits of many contemporary Unix tools, facilitating easy manipulation of text files. It is also used for graphics programming, system administration, network programming, applications that require database access and CGI programming on the Web. Perl is nicknamed “the Swiss Army chainsaw of programming languages” due to its flexibility and adaptability.

The Perl language includes a specialized syntax for writing regular expressions (RE, or regexes), and the interpreter contains an engine for matching strings to regular expressions. The regular-expression engine uses a backtracking algorithm, extending its capabilities from simple pattern matching to string capture and substitution. The regular-expression engine is derived from regex written by Henry Spencer.

Regular-expression syntax is extremely compact, owing to history. The first regular-expression dialects were only slightly more expressive than globs, and the syntax was designed so that an expression would resemble the text that it matches. This meant using no more than a single punctuation character or a pair of delimiting characters to express the few supported assertions. Over time, the expressiveness of regular expressions grew tremendously.

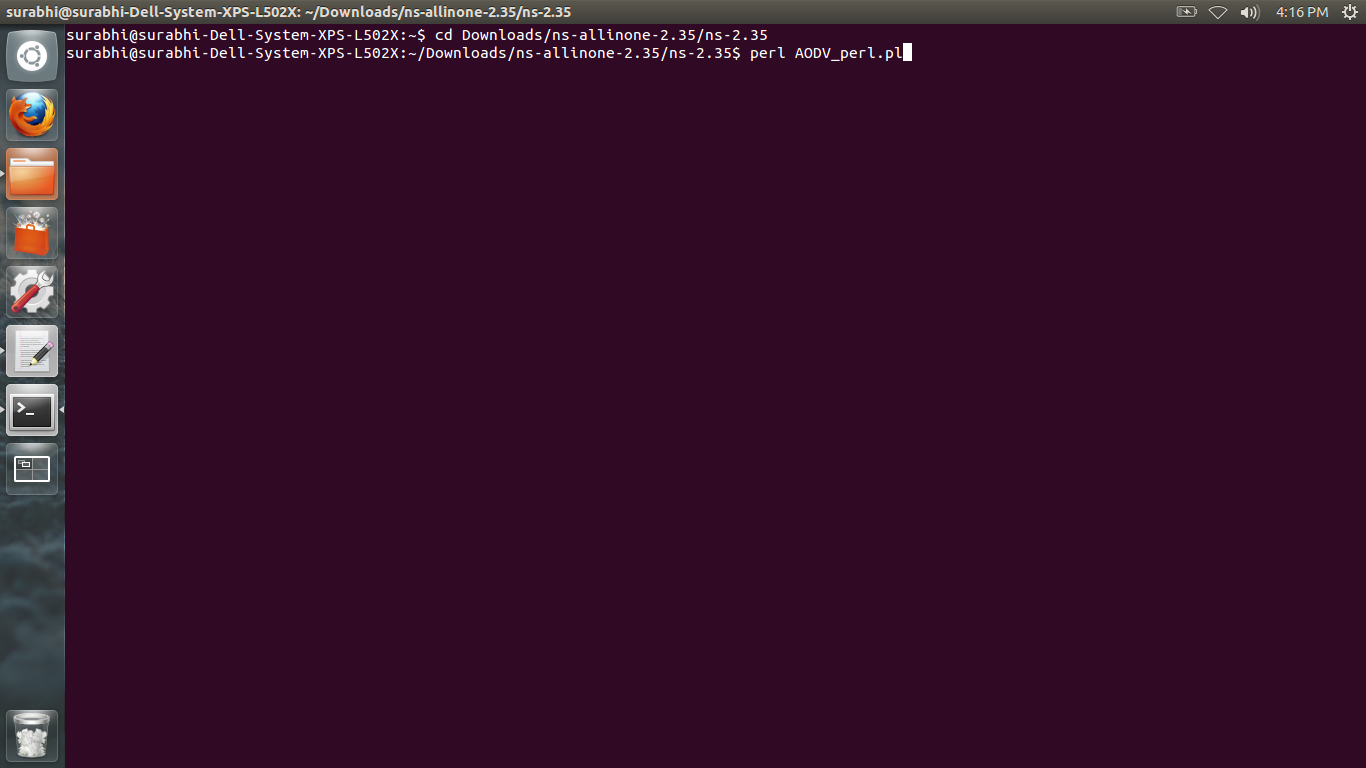
****

Figure 4.6 Perl UI Screenshot-1

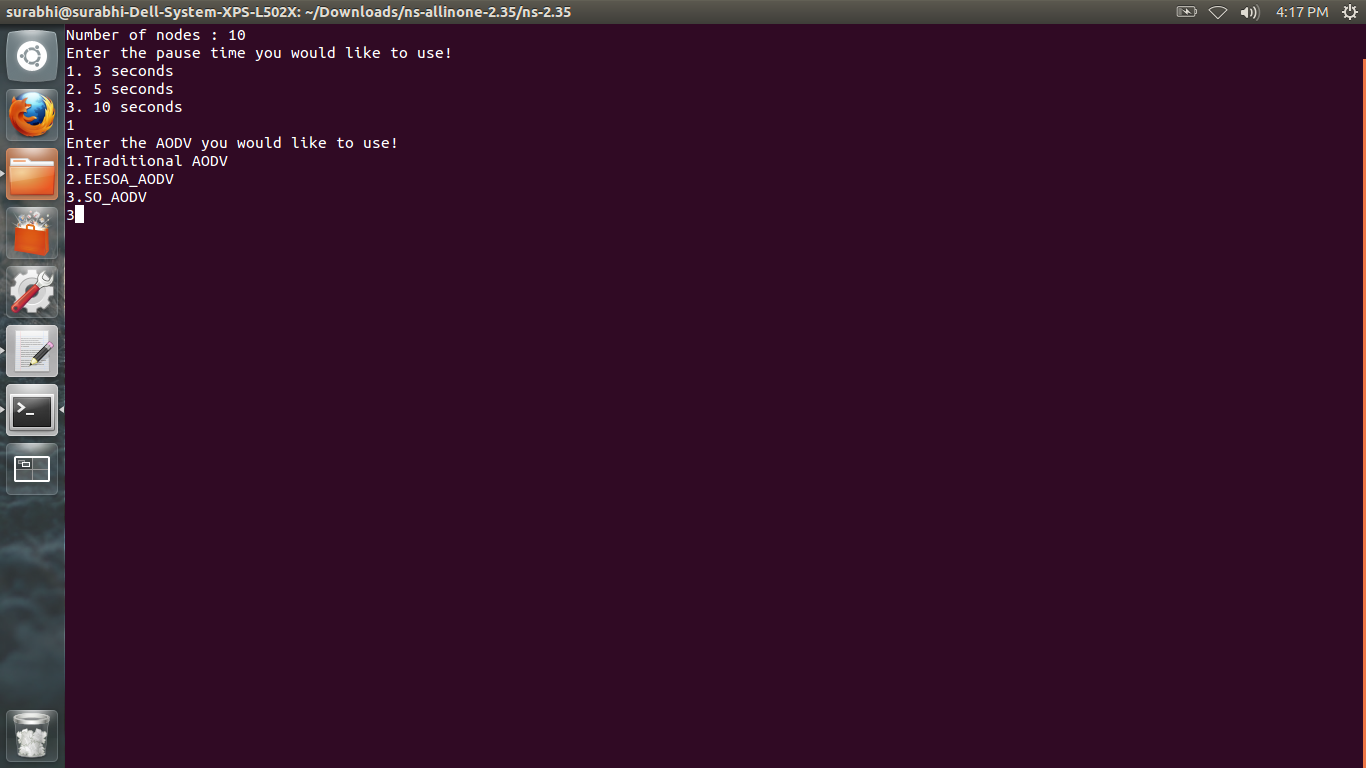


Figure 4.7 Perl UI Screenshot-2

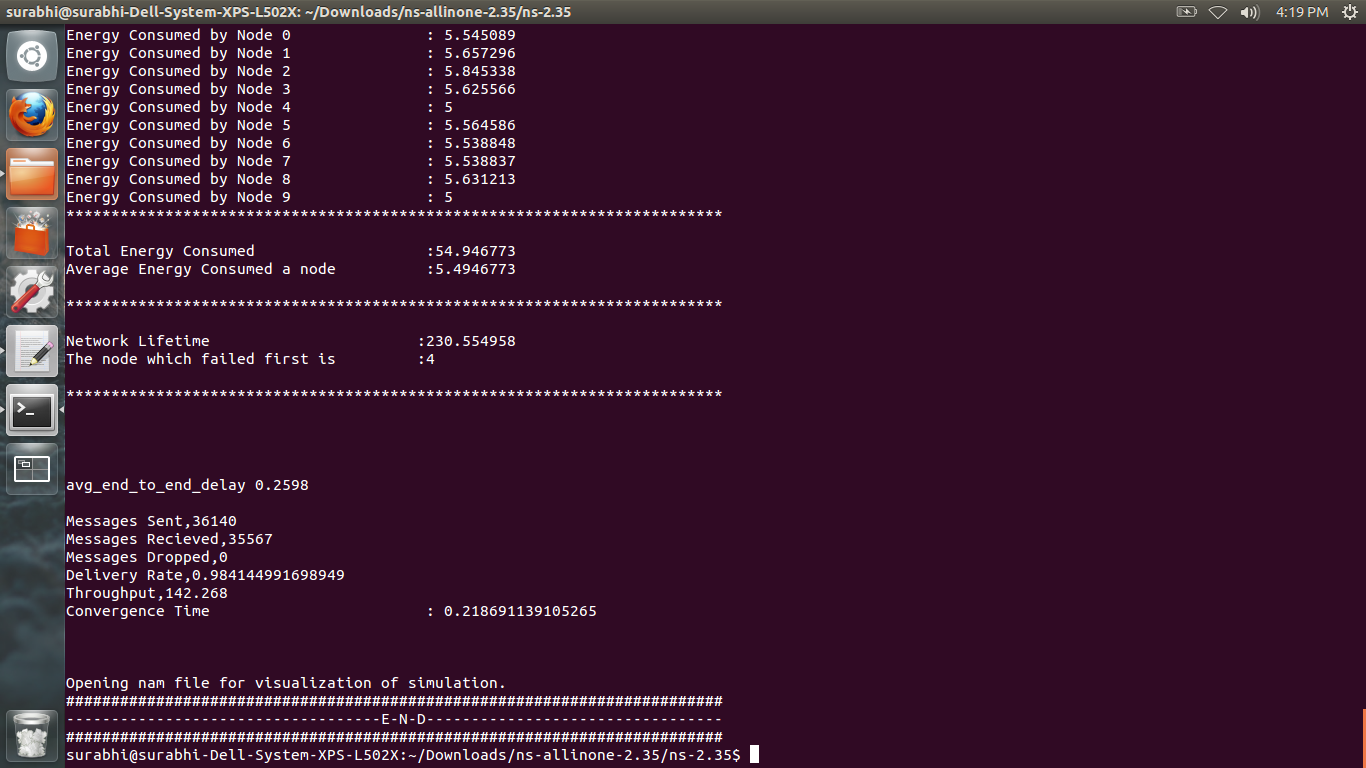


Figure 4.8 Perl UI Screenshot-3

**4.4.3 NAM (Network Animator)**

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. NAM began at LBL. It has evolved substantially over the past few years. The NAM development effort was on ongoing collaboration with the VINT project.

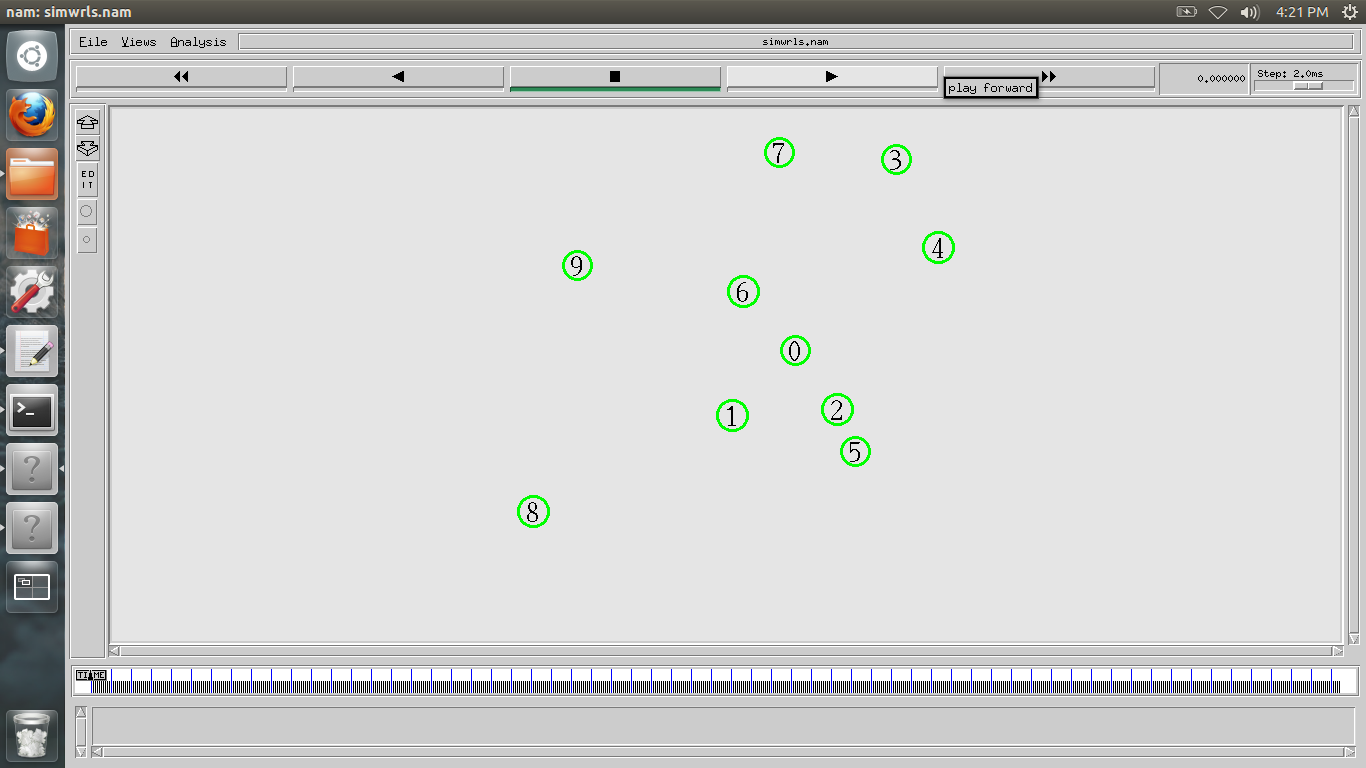


Figure 4.9 Network Animator

**4.5 Test Plan**

**4.5.1 Features to be tested**

Testing includes comparison of SO\_AODV with traditional AODV and EESOA. Comparison is done using the parameters: average end to end delay, network lifetime, convergence time and packet delivery ratio.

**4.5.2 Testing Tools and Environment**

Simulations will be done using ns2 simulator. Multiple simulations with certain boundary conditions will be done to narrow down on the correct results. Testing Environment:

* Channel type: WirelessChannel

In mobile ad hoc network, nodes communicate through wireless channels hence the channel type isinitialized to WirelessChannel.

* Propagation model: TwoRayGround

Radio Propagation is chosen. This model is used to predict the received signal power of each packet. If the node receives a packet with signal power less than the threshold value it can be discarded and the node need not spend time processing the packet.

Radio Propagation model are of three types:

Free Space Model

Two-ray Ground Reflection Model

Shadowing Model

TwoRayGround Model is chosen. The two-ray ground reflection model considers both the direct path and a path reflected of the ground. A single line-of-sight path between two mobile nodes is seldom the only means of propagation. It is shown that this model gives more accurate prediction for long distance than the free space model hence this model is used.

* Interface type: WirelessPhy

The interface type should be wireless because the node communicate through wireless channels.

* MAC layer protocol: Mac 802.11

Mac 802.11 protocol is the de facto standard chosen for wireless network research.

* Interface Queue type : DropTail/PriQueue

DropTail implements FIFO scheduling and drop-on-overflow buffer management typical of most present-day internet router.

* Interface Queue Length: 50

Research studies use 50 as standard Interface Queue Length.

* Antenna type: OmniAntenna

Omni Antennas are used as they transmit packets with the same signal power in all direction.

* Transport Layer Protocol: TCP

TCP is the transport layer protocol. It is reliable and hence it is a suitable for ad hoc network routing.

* X-Dimension Topography: 500m and Y-Dimension Topography: 500m

Radio propagation model configures each node with a radius of 250m for propagation which equates to a diameter of 500m.

* Simulation Time: 250s

The cbrgen.tcl generator assumes a simulation time of at least 250s. Hence the simulation time in this project has been fixed for all scenarios to 250 seconds.

* Speed of Node Movement: 10m/s

Node speed is set to 10m/s to obtain reliable results.

* Mobility Model: Random Way Point Model

It allows to randomize the direction and speed of node movement hence it simulates a network similar to the real time environment.

* Traffic Pattern: FTP

FTP works on TCP which is the transport layer protocol.

**Chapter 5**

**IMPLEMENTATION**

**5.1 Module to Create User Interface**

User interface is created using perl script, this module accepts user inputs such as MANET routing algorithms, parameters to create mobility and traffic files. Some of the routing algorithms are inbuilt in Ns2 such as AODV, DSR, DSDV and TORA. This project is to build a self-organizing and energy efficient routing algorithm based on AODV.

After designing the algorithm the user interface is run by configuring the nodes to accept the routing algorithm and use the mobility scenarios and traffic files. Trace files obtained from simulation of this algorithm are used to calculate average energy consumed, network lifetime, convergence time and packet delivery ratio. After simulation graphs for these parameters are plotted using Microsoft Excel.

**5.2 Module to Create and Run Simulation**

**Mobile Node Parameters**: Once the project scope is decided, a suitable network scenario for wireless network to imitate MANET is designed. Each mobile node needs some options to be configured like routing protocol, MAC layer protocol, antenna type, channel type etc. Node configuration for each node in the network is designed. In this project, wireless channel, TwoRayGround as radio-propagation model, WirelessPhy for network interface tpe, MAC 802.11, DropTail and Priority Queue as interface queue type, OmniAntenna for antenna model, maximum number of packets is set to 50, and link layer is chosen.

The programming of the options is done in tcl script as shown.

Channel type    (Channel/WirelessChannel)

                              Propagation model    (Propagation/TwoRayGround)

                                    Interface type    (Phy/WirelessPhy)

                                    MAC layer protocol    (Mac/802\_11)

                                    Routing protocol    (AODV/ZRP/MPOLSR)

                                    Interface Queue type    (DropTail/PriQueue)

                                    Interface Queue Length    (50)

                                    Antenna type    (Antenna/OmniAntenna)

                    LL type    (LL)

**Topology Parameters**: The topology for the simulation is specified. The topology parameters are the configuration parameters for the topology structure, like the dimensions of the grid, number of nodes present etc. A list of all of them is given below for reference.

                                    x-dimension of the topography

                                    y-dimension of the topography

                                    Number of nodes

The dimensions of the topography are fixed to 500 m x 500 m, and the number of nodes is varied in steps of tens, from the interval [10, 50].

Other parameters are, Total simulation time, and Trace file name

The cbrgen.tcl generator assumes a simulation time of at least 250s. Hence the simulation time in this project has been fixed for all scenarios to 250 seconds.

**Scenario file:** The nodes positions have to be defined for the simulation. Creating individual node movements is possible but cumbersome for a simulation running at the order of 100 seconds and more. For ease of use, a script has been provided which generates these movements automatically in a separate file, called the scenario file. This is generated using the **setdest**script present in **ns/indep-utils/cmu-scen-gen/**directory**.** The command to generate the scenario is:

**./setdest –n <num\_of\_nodes> -p <pausetime> -s <maxspeed> -t <simtime> -x <maxx> -y <maxy>><outdir>/<scenario-file>**

For this project, the pause time has been varied with values 3, 5 and 10.

**Traffic pattern file:** Further, traffic generators at nodes and sinks at the destinations are required. The creation commands for these traffic agents are entered in a separate file, the traffic pattern file. Command to create connection pattern (traffic)

**./ns indep-utils/cmu-scen-gen/cbrgen.tcl -type tcp -nn 20 -mc 10 -seed  0.0 > pattern-20**

**5.3 Module of SO\_AODV**

SO\_AODV incorporates degree of the node and remaining energy.

Table 5.1 Notations

|  |  |
| --- | --- |
| Notations | Meaning |
| Wi | Weight of node vi |
| Di | Degree of node vi |
| Ei(t) | Remaining energy of node vi at time t |

Explanation of notations for the following formulae is given in table 4.1.

The weight of the node is calculated as follows:

Wi = Di \* Ei(t)

To incorporate these two metrics AODV algorithm is altered such that the reply header appended to Hello packets contains fields that provide a measure of the transmitted energy and node lifetime.

Table 5.2 Extended Reply Header

|  |  |  |
| --- | --- | --- |
| Type | Reserved | Hop Count |
| RREQ ID | | |
| Destination IP Address | | |
| Destination Sequence Number | | |
| Originator IP Address | | |
| Lifetime | | |
| Timestamp | | |
| **Remaining Energy** | | |
| **Whether the source node is inhibited or not** | | |
| **Role of Node** | | |
| **Degree of Node** | | |

Code: aodv\_packet.h is a header file which defines the packet formats for AODV.

structhdr\_aodv\_reply

{

u\_int8\_t rp\_type; // Packet Type

u\_int8\_t reserved[2];

u\_int8\_t rp\_hop\_count; // Hop Count

nsaddr\_trp\_dst; // Destination IP Address

u\_int32\_t rp\_dst\_seqno; // Destination Sequence Number

nsaddr\_trp\_src; // Source IP Address

double rp\_lifetime; // Lifetime

doublerp\_timestamp; // when corresponding REQ sent;

**doubleremenergy;**

**intpackinhibit;**

**intpackrole;**

**intpackdegree;**

inlineint size() {

intsz = 0;

sz = 6\*sizeof(u\_int32\_t);

assert (sz>= 0);

returnsz; }

};

The neighbour tables in each of the nodes are altered. The following entries have been added to the existing neighbor table entries:

* Whether the neighbor is inhibited or not
* Neighbour’s role
* Neighbour’s degree

Code: aodv\_rtable.h defines the route table for each node using AODV.

classAODV\_Neighbor

{

friend class AODV;

friend class aodv\_rt\_entry;

public:

AODV\_Neighbor(u\_int32\_t a) { nb\_addr = a; }

protected:

LIST\_ENTRY(AODV\_Neighbor) nb\_link;

nsaddr\_tnb\_addr;

**doubleremenergy; //remaining energy calculation**

**intnb\_inhibited; //see if packet is inhibited**

**intnb\_role; //leader, member or bridge**

**intnb\_degree; //number of neighbours**

doublenb\_expire; // ALLOWED\_HELLO\_LOSS \* HELLO\_INTERVAL

};

**//SO\_AODV**

**inhibited=0;**

**role=0;**

Hello packets are used by nodes to maintain information about their neighbours. When any node receives a Hello packet, it compares its weight against the weights of all of its neighbours. If it has the largest weight, it appoints itself a leader. Each node also cycles through its list of neighbours to count the number of leaders in its vicinity. If two or more are found, then the node is made a bridge. Otherwise, the node becomes a member node. Further, a node marks itself as inhibited if it cannot be the leader in its neighbourhood.

Code: aodv.cc contains the code used to select a role for each node

//leader selection process

AODV\_Neighbor \*nb1 = nbhead.lh\_first;

int lead=1;

intcheckinhibit=0;

float r;

for(; nb1; nb1 = nb1->nb\_link.le\_next) {

if(nb1->nb\_role==0)

checkinhibit++;

r=nb1->remenergy\*nb1->nb\_degree-degree\*remenergy;

if((r>0)&&(nb1->nb\_inhibited==0))

{

inhibited=1;

lead=0;

}

}

if(lead==1)

{

printf("Node %d=leader",index);

role=0;

inhibited=0;

}

else if(checkinhibit==1)

{

role=2;

printf("Node %d=member",index);

}

else if(checkinhibit>1)

{

role=1;

printf("Node %d=bridge",index);

}

Packet::free(p);

During route discovery, a member node will always send the route request packet (RREQ) to its nearest leader. Leaders and bridges broadcast any RREQ packets that they receive. If a leader has failed or moved away, and a member cannot find its leader, then the member node switches back to traditional AODV, i.e., it broadcasts the RREQ packet.

//aodv.cc:ensuring adaptability in the network formation

int yahoo=0;

if(role==2)

{

for(; nb; nb = nb->nb\_link.le\_next) {

if(nb->nb\_role == 0)

{

ih->daddr()=nb->nb\_addr;

yahoo=1;

break;

}

}

if(yahoo==0)

{ih->daddr() = IP\_BROADCAST;}

**5.4 Module to Calculate Average End to End Delay, Network Lifetime, Convergence Time and Packet Delivery Ratio**

**5.4.1 Format of Trace File**

The output after running the simulation is entered in a trace file (.tr). The trace format can be divided into the following fields:

**Event type**

The first field describes the type of event taking place at the node and can be one of the four types:

**-s** Send

**-r** Receive

**-d** Drop

**-f** Forward

**General tag**

The second field starting with "-t" may stand for time or global setting

**-t** Time

**-t** \* (global setting)

**Node property tags**

This field denotes the node properties like node-id, the level at which tracing is being done like agent, router or MAC. The tags start with a leading "-N" and are listed as below:

**-Ni** Node id

**-Nx** Node's x-coordinate

**-Ny** Node's y-coordinate

**-Nz** Node's z-coordinate

**-Ne** Node energy level

**-Nl** Trace level, such as AGT, RTR, MAC

**-Nw** Reason for the event.

The different reasons for dropping a packet are given below:

**"END"** DROP\_END\_OF\_SIMULATION

**"COL"** DROP\_MAC\_COLLISION

**"DUP"** DROP\_MAC\_DUPLICATE

**"ERR"** DROP\_MAC\_PACKET\_ERROR

**"RET"** DROP\_MAC\_RETRY\_COUNT\_EXCEEDED

**"STA"** DROP\_MAC\_INVALID\_STATE

**"BSY"** DROP\_MAC\_BUSY

**"NRTE"** DROP\_RTR\_NO\_ROUTE i.e. no route is available.

**"LOOP"** DROP\_RTR\_ROUTE\_LOOP i.e. there is a routing loop

**"TTL"** DROP\_RTR\_TTL i.e. TTL has reached zero.

**"TOUT"** DROP\_RTR\_QTIMEOUT i.e. packet has expired.

**"CBK"** DROP\_RTR\_MAC\_CALLBACK

**"IFQ"** DROP\_IFQ\_QFULL i.e. no buffer space in IFQ.

**"ARP"** DROP\_IFQ\_ARP\_FULL i.e. dropped by ARP

**"OUT"** DROP\_OUTSIDE\_SUBNET i.e. dropped by base stations on receiving routing updates from nodes outside its domain.

**Packet Information at IP level**

The tags for this field start with a leading "-I" and are listed along with their explanations as following:

**-Is** Source address.source port number

**-Id** Destination address.destination port number

**-It** Packet type

**-Il** Packet size

**-If** Flow id

**-Ii** Unique id

**-Iv** TTL value

**Next hop Information**

This field provides next hop info and the tag starts with a leading "-H".

**-Hs** Id for this node

**-Hd** Id for next hop towards the destination.

**Packet Information at MAC level**

This field gives MAC layer information and starts with a leading "-M" as shown below:

**-Ma** Duration

**-Md** Destination ethernet address

**-Ms** Source ethernet address

**-Mt** Ethernet type

**Packet Information at "Application level"**

The packet information at application level consists of the type of application like ARP, TCP, the type of adhoc routing protocol like DSDV, DSR, AODV etc. being traced. This field consists of a leading "-P" and list of tags for different application is listed as below:

**-P arp** Address Resolution Protocol. Details for ARP are given by the following tags:

**-Po** ARP Request/Reply

**-Pm** Source mac address

**-Ps** Source address

**-Pa** Destination mac address

**-Pd** Destination address

**-P dsr** This denotes the adhoc routing protocol called Dynamic source routing.

Information on DSR is represented by the following tags:

**-Pn** How many nodes are traversed

**-Pq** Routing request flag

**-Pi** Route request sequence number

**-Pp** Routing reply flag

**-Pl** Reply length

**-Pe** Source of source routing->destination of the source routing

**-Pw** Error report flag?

**-Pm** Number of errors

**-Pc** Report to whom

**-Pb L**ink error from linka->linkb

**-P cbr** Constant bit rate.

The new trace format available with ns2 is used. This can be enabled using the command.

$ns use-newtrace

The trace file is as shown in figure 5.1:

s -t 0.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 314.92 -Ny 412.49 -Nz 0.00 -Ne 6.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.255 -Id -1.255 -It AODV -Il 60 -If 0 -Ii 0 -Iv 1 -P aodv -Pt 0x1 -Ph 1 -Pd 0 -Pds 2 -Pl 4.000000 -Pc HELLO

s -t 0.000000000 -Hs 1 -Hd -2 -Ni 1 -Nx 407.89 -Ny 479.13 -Nz 0.00 -Ne 9.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.255 -Id -1.255 -It AODV -Il 60 -If 0 -Ii 0 -Iv 1 -P aodv -Pt 0x1 -Ph 1 -Pd 1 -Pds 2 -Pl 4.000000 -Pc HELLO

s -t 0.000000000 -Hs 2 -Hd -2 -Ni 2 -Nx 387.13 -Ny 467.23 -Nz 0.00 -Ne 7.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 2.255 -Id -1.255 -It AODV -Il 60 -If 0 -Ii 0 -Iv 1 -P aodv -Pt 0x1 -Ph 1 -Pd 2 -Pds 2 -Pl 4.000000 -Pc HELLO

s -t 0.000000000 -Hs 3 -Hd -2 -Ni 3 -Nx 184.38 -Ny 459.28 -Nz 0.00 -Ne 6.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is

Figure 5.1 Example of Trace File

**5.4.2 Calculate Performance Metrics**

This module consists of five phases. They are:

* Accept inputs and check for correctness and compatibility.
* Parse the trace file and measure the average end to end delay.
* Parse the trace file and measure the network lifetime.
* Parse the trace file and measure the packet delivery ratio.
* Parse the trace file and convergence time.

**5.4.2.1 Accepting inputs**

The input to this module is the trace file which is generated by ns2. The system needs to check whether the trace file exists and ensure that it is non-empty. This is done using the flag –e in perl.

Verification of the trace file to be in the new trace format is required. This is done by checking for the flag –Nl in the trace file, as this is present only in the new trace format.

**5.4.2.2 Parse the Trace File and Measure Average End to End Delay**

The trace file is passed to the perl script to analyze and calculate the average end to end delay. The average end to end delay is the time taken for packet to be transmitted from source to destination.

**5.4.2.3 Parse the Trace File and Measure Network Lifetime**

The trace file is passed to the perl script to analyze and calculate the network lifetime.

The network lifetime is defined as the duration for which all nodes in the network are alive.

The trace file can be analyzed to determine the time using the –t option at which any node’s energy becomes zero and determine the node id using –Ni option. The value that the –t option provides is the network lifetime.

**5.4.2.4 Parse the Trace File and Measure Packet Delivery Ratio**

The trace file is passed to the perl script to analyze and calculate the packet delivery ratio.

Packet delivery ratio is the ratio of the number of data packets delivered to the destination to the number of data packets expected to be received. Perl script accept trace file and calculates packet delivery ratio as shown below

Packet deliver ratio = (data packet delivered / data packet sent) \* 100

**5.4.2.5 Parse the Trace File and Measure Convergence Time**

The trace file is passed to the perl script to analyze and calculate the convergence time.

The convergence time is the time between a fault detection, and restoration of new, valid, path information. To calculate this the time the first packet is dropped due to invalid path, and the time when the first packet is successfully received when the new path is found.

There can be many reasons for a packet drop, such as interface queue being full (IFQ), end of simulation (END), mac packet error (ERR), etc. When the packet is not delivered to the destination because the current path is no longer valid, the mac entity calls back the router entity. This is recorded by ns2 by setting the value of the flag –Nw as CBK. This is the time required to measure to obtain the time when the first packet is dropped due to invalid path. When such a packet is observed, the source and destination pair for which the packet is dropped by examining the fields –Is (source id) and –Id (destination id) is recorded.

Next, subsequent packets are monitored to check when the first packet originating from this source is received by the destination. That is, the flags –Is (source id) and –Id (destination id) must be the same as in the CBK packet recorded earlier, and the packet must be received by the application entity (-Nl = agent). This indicates the successful discovery of a new valid path by the algorithm. This time is recorded, and the interval between this time and the time when the packet was dropped is taken as convergence time for this flow. To obtain convergence time of the algorithm for the overall scenario, the average of all such convergence times is taken.

**Chapter 6**

**TESTING**

**6.1 System Testing**

In this phase of testing, requirement specifications are validated against the software system developed. This test provides the final assurance that the software meets all the functionality, behavioural and performance requirements. Here four modules average energy consumed, network lifetime, throughput and packet delivery ratio are tested for all three protocols. Protocols are tested with four conditions

* low density and low mobility
* low density and high mobility
* high density and low mobility
* high density and high mobility

In order to be able to cover most if not all the types of scenarios the algorithms might face, both the node density (number of nodes) and the node mobility (pause time) is varied. The node density (number of nodes) is varied in the range [10, 50] in steps of 10 (5 different node densities). Thus 10 nodes represent the low node density case, while 50 nodes represent the high node density case. Also, three different pause times were used: 3, 5 and 10 seconds. The pause time of 10 implies that the nodes pause in their initial positions for 10 seconds. It represents nodes which have low mobility. Similarly, pause time 3 represents very high mobility where the nodes are in constant motion. Thus each algorithm is tested over 5 node densities x 3 pause times = 15 scenarios, or a total of 45 simulations.

Also, each scenario is generated by varying initial energies in a range of 5-10 with the same parameters. Each of these cases is run for SO\_AODV, EESOA (integrated with AODV) and traditional AODV. Connection Patterns and Mobility Scenarios are kept the same for all three protocols to achieve consistent behaviour. The values are tabulated. At the conclusion of the project, a total of 15 simulations have been run.

Graphs have been depicted for pause times 3, 5 and 10 and number of nodes 10 to 50 varying in steps of 10.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No** | **Scenario** | **Expected Output** | **Actual Output** | **Remarks** |
| 1 | No of Nodes=10  Pause time =10 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |
| 2 | No of Nodes=20  Pause Time = 3 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |
| 3 | No of Nodes=30  Pause time = 5 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |
| 4 | No of Nodes=40  Pause Time = 3 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |
| 5 | No of Nodes=50  Pause time = 5 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |
| 6 | No of Nodes=10  Pause Time = 5 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |
| 7 | No of Nodes=30  Pause time = 3 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |
| 8 | No of Nodes=50  Pause Time = 10 | SO\_AODV should have a much lesser delay than traditional AODV | SO\_AODV exhibits a much lesser delay than Traditional AODV, and it is nearly the same as EESOA | Pass |

**6.2 Test Cases for End-to-End Delay**

Table 6.1 Result of simulations run for End-to-End delay

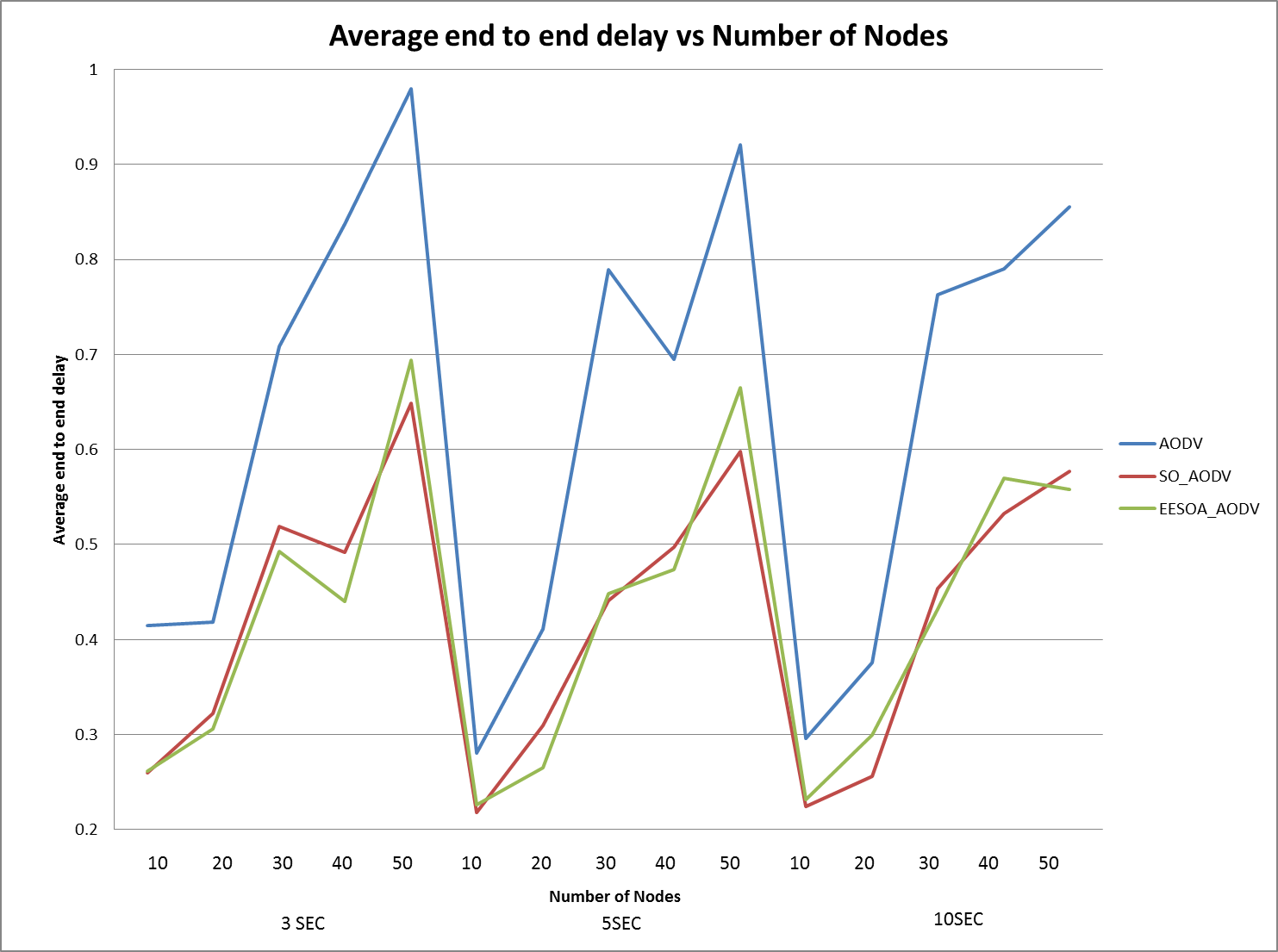


Figure 6.1 Average End to End Delay Vs Number of Nodes

From the graphs it can be seen that when the mobility of the nodes is high (pause time 3 seconds) the end to end delay of traditional AODV is quite high. When the density of the nodes reaches maximum (50), the delay nearly reaches one second. As the density of nodes increases, the end to end delay of all three algorithms increases. However the end to end delays of SO\_AODV and EESOA are both quite low compared to AODV (about 40 percent lesser), and there is no significant difference between the two. The reduction in end to end delay can be attributed to the virtual backbone created.

A similar situation is seen for the moderate mobility case (pause time 5 seconds). AODV has a much higher end to end delay than either of the two. However the end to end delays of all three protocols has reduced slightly. This is due to the reduced mobility of the nodes.

For the low mobility situation (pause time 10 seconds), again AODV has a very high end to end delay while the other two protocols show comparable values. End to end delay continues to increase with node density.

The downward spikes in the graph as the graph moves from one pause time to another are not indicative of a change in performance. The graph that has been plotted is merely a combination of three plots, one for each pause time, varying node density for every pause time.

**6.3 Test Cases for Network Lifetime**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No** | **Scenario** | **Expected Output** | **Actual Output** | **Remarks** |
| 1 | No of Nodes=10  Pause time =3 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |
| 2 | No of Nodes=20  Pause Time = 10 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |
| 3 | No of Nodes=30  Pause time = 5 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |
| 4 | No of Nodes=40  Pause Time = 5 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |
| 5 | No of Nodes=50  Pause time = 3 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |
| 6 | No of Nodes=30  Pause Time = 3 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |
| 7 | No of Nodes=40  Pause time = 10 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |
| 8 | No of Nodes=50  Pause Time = 10 | SO\_AODV is expected to have more Network Lifetime than traditional AODV | The Network Lifetime increases both in EESOA and SO\_AODV compared to Traditional AODV | Pass |

Table 6.2 Result of simulations run for Network Lifetime

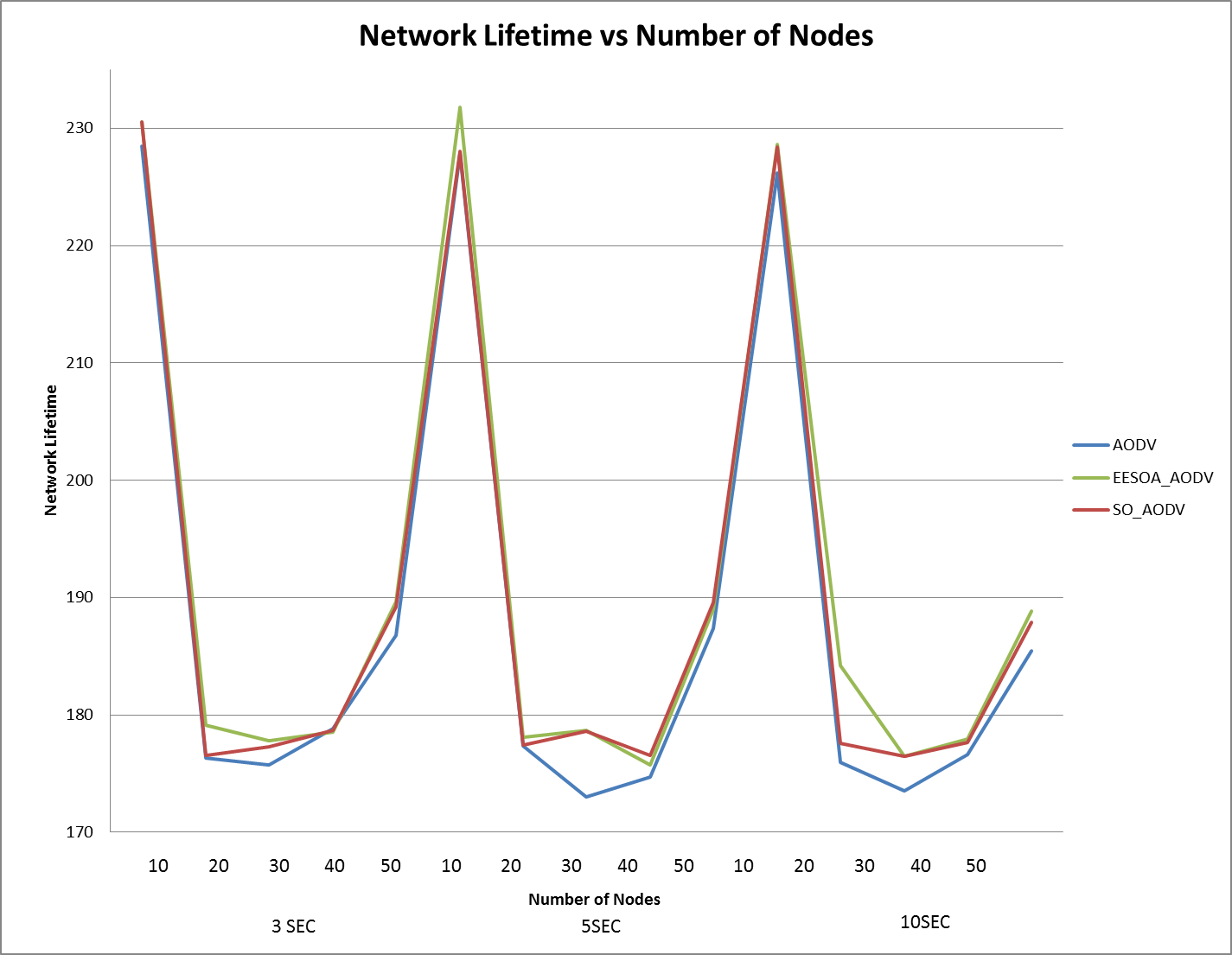


Figure 6.2 Network Lifetime Vs Number of Nodes

From the graph, for the high mobility case (pause time 3 seconds), it can be seen that SO\_AODV and EESOA show consistently better network lifetimes than AODV. For the low density situation (10 nodes), the network lifetime for all 3 protocols is quite high. This can be attributed to the fact that there are far fewer connections and much lesser traffic through each node. On close observation for this case, it can be seen that SO\_AODV still has a better lifetime than AODV (about one percent). For the high mobility case, the difference in network lifetimes is not drastic.

For cases with slightly lesser mobility (5 seconds pause time), the performance of SO\_AODV and EESOA can be truly appreciated. The protocols show a large increase innetwork lifetime for different node densities (about 3 percent). Thus the selection of leaders to perform the bulk of the routing has ensured that nodes with lower energies last longer.

The situations with pause time 10 seconds tell a similar story. Once again the network lifetimes of SO\_AODV and EESOA are significantly better than AODV.

The upward spikes in the graph as the graph moves from one pause time to another are not indicative of a change in performance. The graph that has been plotted is merely a combination of three plots, one for each pause time, varying node density for every pause time.

**6.4 Test Cases for Convergence time**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No** | **Scenario** | **Expected Output** | **Actual Output** | **Remarks** |
| 1 | No of Nodes=20  Pause time =3 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV clearly has a reduced convergence time compared to EESOA | Pass |
| 2 | No of Nodes=30  Pause Time = 5 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV clearly has a reduced convergence time compared to EESOA | Pass |
| 3 | No of Nodes=40  Pause time = 10 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV clearly has a reduced convergence time compared to EESOA | Pass |
| 4 | No of Nodes=50  Pause Time = 5 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV clearly has a reduced convergence time compared to EESOA | Pass |
| 5 | No of Nodes=20  Pause time = 10 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV clearly has a reduced convergence time compared to EESOA | Pass |
| 6 | No of Nodes=10  Pause Time = 10 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV has a convergence time slightly worse than EESOA | Fail |
| 7 | No of Nodes=50  Pause time = 10 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV clearly has a reduced convergence time compared to EESOA | Pass |
| 8 | No of Nodes=30  Pause Time = 3 | SO\_AODV should have a much better convergence time than EESOA | SO\_AODV clearly has a reduced convergence time compared to EESOA | Pass |

Table 6.3 Result of simulations run for Convergence time

SO\_AODV does not pass the test case when number of nodes = 10 and pause time =10. Since its convergence time is better than EESOA for nearly all the test cases, this anomaly can be attributed to this particular instance of the topology and the connection pattern.

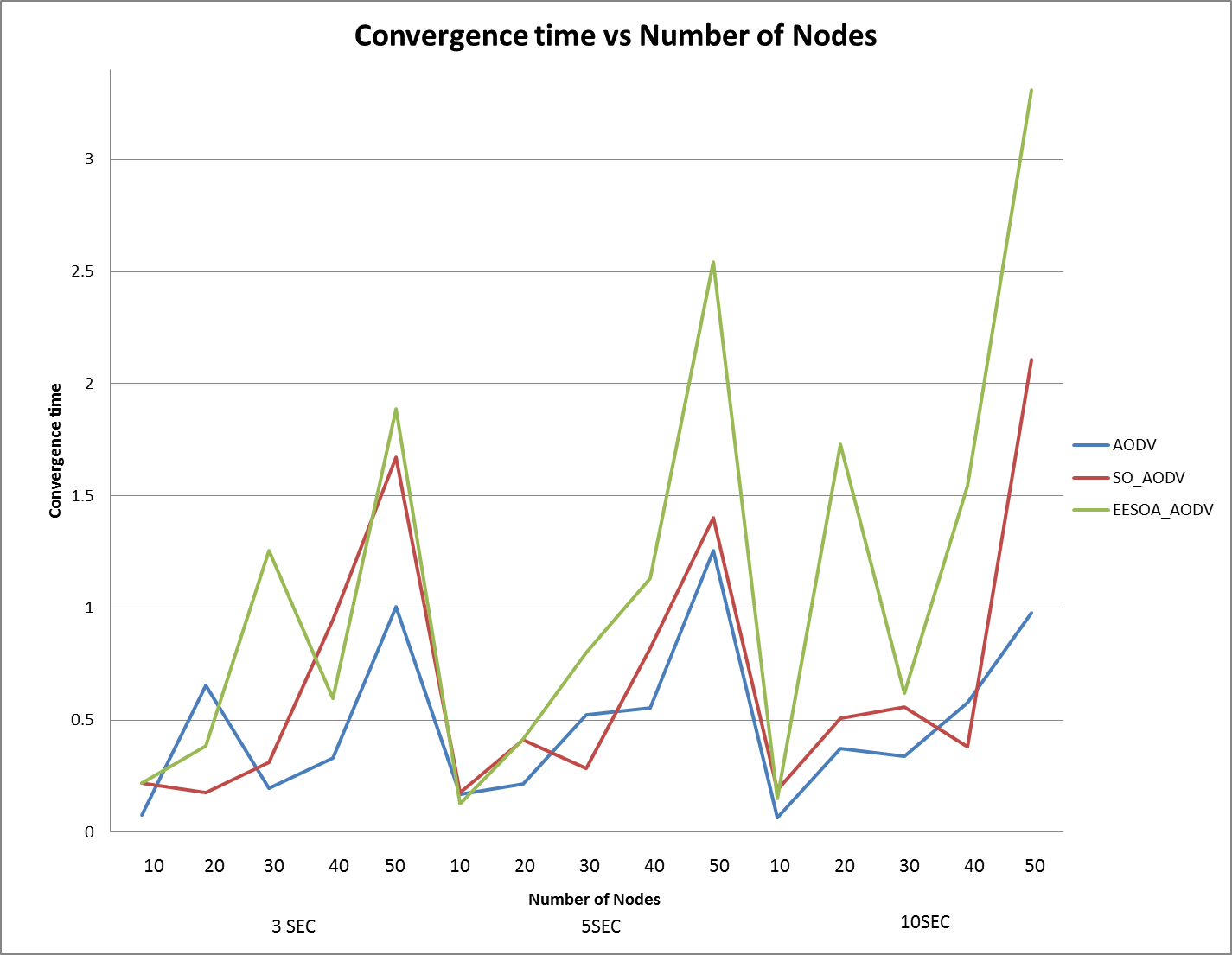


Figure 6.3 Convergence Time Vs Number of Nodes

From the graphs, for the high mobility case (pause time 3 seconds), it can be seen that AODV has the lowest convergence time. As the density of nodes goes up, link failures increase as well, and thus convergence times increase. But in nearly all of the test cases for the high mobility situation, SO\_AODV has lower convergence times than EESOA (about 50 percent lesser). Thus it is able to deal with changes to the topology better (i.e., changing of a cluster’s leader).

For situations with lower mobility (pause time 5 seconds), SO\_AODV shows a remarkable decrease in convergence time in comparison to EESOA. In all the test cases, convergence times for SO\_AODV are even comparable to AODV, and in some cases lesser. For the high density case (50 nodes), EESOA takes more than 2.5 seconds to recover from a failure, while SO\_AODV takes less than 1.5 seconds.

For the test cases with least mobility (pause time 10 seconds), a similar situation can be seen. SO\_AODV has much lesser convergence times than EESOA, and is comparable to traditional AODV. EESOA takes much longer to re-establish a link, since it must wait till the election of a new leader for communication to proceed. But SO\_AODV can continue communicating in this situation using broadcast.

The downward spikes in the graph as the graph moves from one pause time to another are not indicative of a change in performance. The graph that has been plotted is merely a combination of three plots, one for each pause time, varying node density for every pause time.

**6.5 Test Cases for Packet Delivery Ratio**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No** | **Scenario** | **Expected Output** | **Actual Output** | **Remarks** |
| 1 | No of Nodes=10  Pause time =5 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.98172  Traditional AODV=0.9786 | Pass |
| 2 | No of Nodes=20  Pause Time = 3 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.9768  Traditional AODV=0.97942 | Pass |
| 3 | No of Nodes=30  Pause time = 5 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.9686  Traditional AODV=0.9775 | Pass |
| 4 | No of Nodes=40  Pause Time = 10 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.96567  Traditional AODV=0.97456 | Pass |
| 5 | No of Nodes=10  Pause time = 3 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.98414  Traditional AODV=0.9909 | Pass |
| 6 | No of Nodes=50  Pause Time = 3 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.948711  Traditional AODV=0.9544 | Fail |
| 7 | No of Nodes=20  Pause time = 10 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.98086  Traditional AODV=0.9786 | Pass |
| 8 | No of Nodes=40  Pause Time = 3 | SO\_AODV should have a packet delivery ratio more than 0.95, and should not be much lesser than traditional AODV | SO\_AODV=0.96225  Traditional AODV=0.96625 | Pass |

Table 6.4 Result of simulations run for Packet Delivery Ratio

SO\_AODV has a packet delivery ratio less than 0.95 when the number of nodes is 50 and the pause time is 3 seconds. This can be attributed to the fact that there is a lot more traffic passing through each leader node.

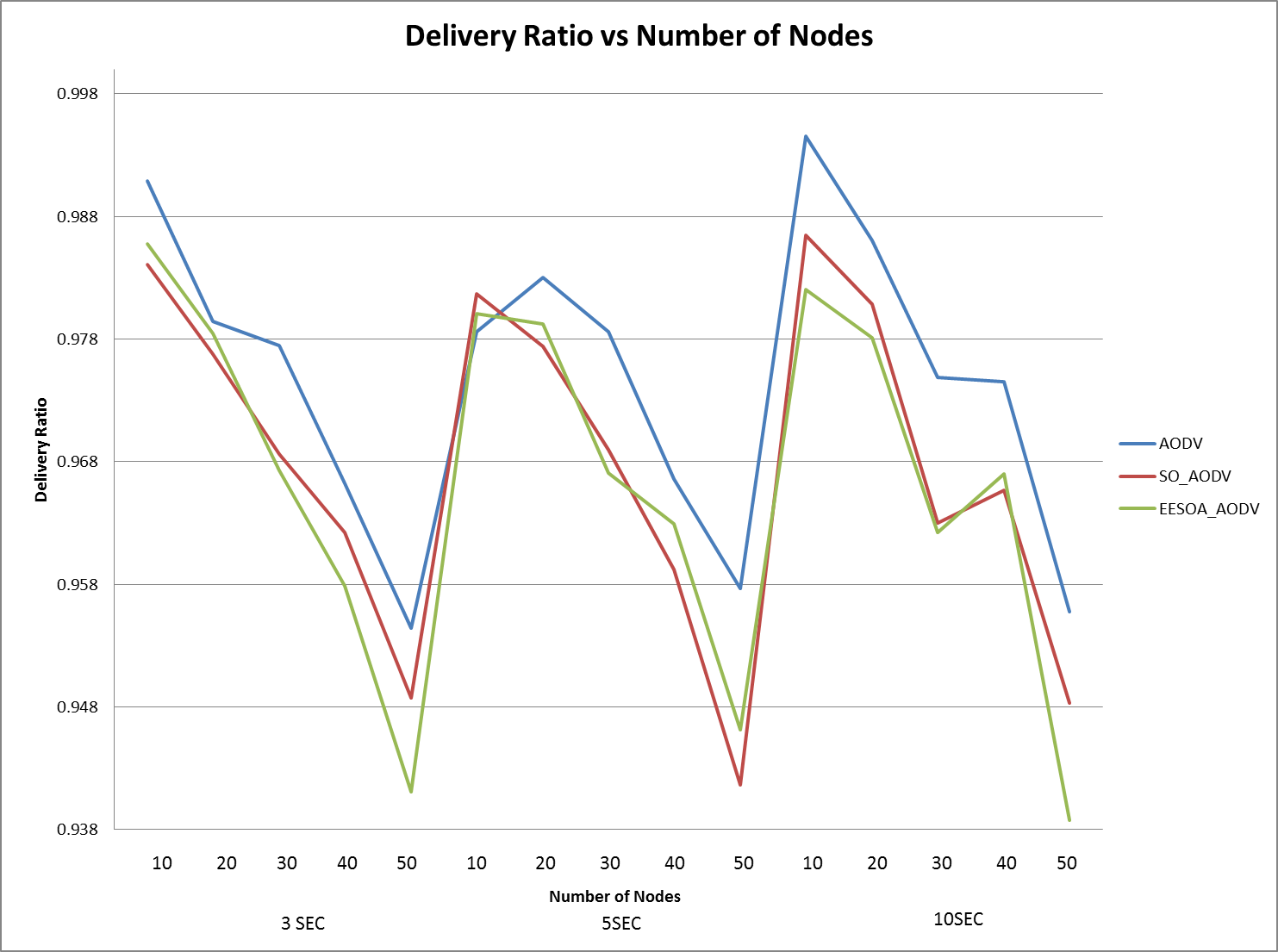


Figure 6.4 Packet Delivery Ratio Vs. Number of Nodes

From the graphs, it can be seen that in the high mobility situation (pause time 3 seconds), AODV has consistently better values for packet delivery ratio (less than one percent better). This could be attributed to the fact that in SO\_AODV and EESOA, the leaders have a lot of traffic passing through them, and may thus drop more packets than AODV does. The packet delivery ratio for all three protocols drops as node density (and number of connections increases), and for most node densities, the delivery ratios of SO\_AODV and EESOA are comparable.

For the cases with lower mobility (pause time 5 seconds), a similar situation can be seen. AODV has the highest packet delivery ratio, and the ratios of the other two protocols are nearly the same. Packet delivery ratio again drops as node density increases.

For cases with least mobility (pause time 10 seconds), the situation is similar. AODV has the best packet delivery ratio.

However in nearly all the test cases, the delivery ratio of SO\_AODV is quite acceptable.

The upward spikes in the graph as the graph moves from one pause time to another are not indicative of a change in performance. The graph that has been plotted is merely a combination of three plots, one for each pause time, varying node density for every pause time.

**Chapter 7**

# CONCLUSION AND FUTURE ENHANCEMENTS

**7.1 Conclusion**

The aim of this project was to evaluate and compare the performance of SO\_AODV routing protocol with Traditional AODV and basic EESOA routing protocol and uncover their strengths and weaknesses. The comparisons were made based on the performance metrics – network lifetime, average end to end delay, delivery ratio and convergence time. NS2 was chosen to simulate these algorithms, UML was used for the design, oTCL script and C++ script was used by NS2 and a perl script was written to measure the metrics.

When the results of simulation were analyzed it was found that SO\_AODV had better values for network lifetime and end to end delay than Traditional AODV whereas Traditional AODV had better values for delivery ratio than SO\_AODV. SO\_AODV had better values of convergence time than EESOA and both SO\_AODV and EESOA had the same end to end delay and network lifetime and packet delivery ratio. The application of self-organizing techniques indeed improved the performance of AODV with respect to the metrics mentioned.

The objectives of the project were reached. An understanding of MANETs and the algorithms that facilitate routing in MANETs was obtained. From the above mentioned comparison of metrics it was concluded that the objective of making the SO\_AODV algorithm better than the EESOA algorithm was achieved. The aim of making SO\_AODV better than AODV also was achieved with respect to the network lifetime and end to end delay. It could not be achieved with respect to packet delivery ratio.

**7.2 Future Work**

In this project the topology of the network and the connection pattern were designed. Certain options such as traffic pattern, initial energy and packet queue length were configured for each mobile node. The same options were chosen for all the nodes. An algorithm can be developed to handle various communication capacities and link qualities in MANETs. This would make the simulation closer to a realistic situation.

During the tests it was found that Traditional AODV had better values for throughput, convergence time and delivery ratio than SO\_AODV. Methods to improve the QoS parameters of SO\_AODV such as packet delivery ratio (which was reduced due to congestion at the leader node) can be taken up as future work.

**Chapter 8**

**REFERENCES**

1. Lu Han, “Wireless Ad-hoc Networks”, October 8, 2004.
2. Falko Dressler, “Self-Organization in Ad Hoc Networks: Overview and Classification”, 2006.
3. [JeroenHoebeke](http://www.citeulike.org/user/mcnerney/author/Hoebeke:J), [Ingrid Moerman](http://www.citeulike.org/user/mcnerney/author/Moerman:I), [Bart Dhoedt](http://www.citeulike.org/user/mcnerney/author/Dhoedt:B), [Piet Demeester](http://www.citeulike.org/user/mcnerney/author/Demeester:P), “An Overview of Mobile Ad Hoc Networks: Applications and Challenges”, Journal of the Communications Network, Vol. 3, pp. 60-66, July 2004.
4. KavitaTaneja, R.B.Patel, “An Overview of Mobile Ad hoc Networks: Challenges and Future”.
5. BartoszBiskupski, Jim Dowling, Jan Sacha, “Properties and mechanisms of self-organizing MANET and P2P systems,” in *ACM Transactions on Autonomous and Adaptive Systems (TAAS),* vol. 2, Issue 1, no. 1, March 2007.
6. Marco Fotino and Floriano De Rango, "Energy Issues and Energy Aware Routing in Wireless Ad Hoc Networks," in *Mobile Ad-Hoc Networks: Protocol Design,* Prof. [Xin Wang](http://www.intechopen.com/books/editor/mobile-ad-hoc-networks-protocol-design) (Ed.), Italy: InTech, 2011, ch. 15, pp. 281-296.
7. *The network simulator - ns-2*. [Online]. Available: http://www.isi.edu/nsnam/ns 2009.
8. The ns Manual (formerly ns Notes and Documentation) The VINT Project a Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. Kevin Fall, Editor KannanVaradhan 2006.
9. B. Tuch, “Development of wavelan, an ism band wireless lan,” *AT&T Technical Journal*, vol. 72, no. 4, pp. 27–37, July/August 1993.
10. *NS2 versus NS3.* [Online]. Available: http://wrc-ejust.org/crn/images/Tutorials/ns2vsns3.pdf
11. *Appendix B. Comparison BetweenQualnet and NS2*. [Online]. Available: http://my.safaribooksonline.com/book/programming/mobile/9788131731666/comparison-between-qualnet-and-ns2/app02
12. J. Guadalupe Olascuaga-Cabrera, Andres Mendez-Vazquez, Ernesto López-Mellado, “A Novel Distributed Energy-Efficient Self-Organized Algorithm for Wireless Ad-hoc Networks”, in *Eighth International Conference on Intelligent Environments*, 2012.
13. F. Chiti, A. De Cristofaro, R. Fantacci, D. Tarchi, G. Collodo, G. Giorgetti, and A. Manes, “Energy efficient routing algorithms for application to agro-food wireless sensor networks,” in *IEEE International Conference on Communications (ICC)*, vol. 5, may 2005, pp. 3063–3067.
14. B. O’Flyrm, R. Martinez, J. Cleary, C. Slater, F. Regan, D. Diamond, and H. Murphy, “Smartcoast: A wireless sensor network for water quality monitoring,” in *32nd IEEE Conference on Local Computer Networks (LCN)*, Oct. 2007, pp. 815 –816.
15. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, pp. 393–422, 2002.
16. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen and J.-P. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *MobiCom: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1999, pp. 151–162.
17. O. K. Tonguz, N. Wisitpongphan, J. S. Parikht, and F. Bait, “On the broadcast storm problem in ad hoc wireless networks,” in *3rd International Conference on Broadband Communications, Networks and Systems*, Oct 2006, pp. 1–11.
18. S. Funke, A. Kesselman, U. Meyer, and M. Segal, “A simple improved distributed algorithm for minimum cds in unit disk graphs,” in *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, vol. 2, Aug 2005, pp. 220–223.
19. M. Rai, S. Verma, and S. Tapaswi, “A power aware minimum connected dominating set for wireless sensor networks,” *Journal of Networks*, vol. 4, no. 6, pp. 511–519, Aug 2009.
20. J.-L. Lu, F. Valois, and D. Barthel, “Low-energy selforganization scheme for wireless ad hoc sensor networks,” in *Fourth Annual Conference on Wireless on Demand Network Systems and Services, 2007.*, Jan. 2007, pp. 138 –145.
21. H. Guo, Y. Qian, K. Lu, and N. Moayeri, “Backbone construction for heterogeneous wireless ad hoc networks.” in *ICC*. IEEE, 2009, pp. 1–5.
22. J. Olascuaga-Cabrera, E. L´opez-Mellado, A. Mendez- Vazquez, and F. Ramos-Corchado, “A self-organization algorithm for robust networking of wireless devices,” *Sensors Journal, IEEE*, vol. 11, no. 3, pp. 771–780, march 2011.
23. N. Li, J. C. Hou, and L. Sha, “Design and analysis of an mst-based topology control algorithm,” *IEEE Transactions on Wireless Communications*, vol. 4, no. 3, pp. 1195–1206, 2005.
24. T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*, 1st ed. Springer Publishing Company, Incorporated, 2008.
25. L. H. Correia, D. F. Macedo, A. L. dos Santos, A. A. Loureiro, and J. M. S. Nogueira, “Transmission power control techniques for wireless sensor networks,” *Computer Networks*, vol. 51, no. 17, pp. 4765–4779, July 2007.
26. T. C. Collier and C. E. Taylor, “Self-organization in sensor networks,” *Journal of Parallel and Distributed Computing*, vol. 64, no. 7, pp. 866–873, July 2004.
27. I. Amadou and F. Valois, “Performance Evaluation of Distributed Self-Organization Protocols in Wireless Sensor Networks,” in *7th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, Bodrum, Turquie, Oct. 2010.
28. Y. Iima, A. Kanzaki, T. Hara, and S. Nishio, “Overhearing based data transmission reduction for periodical data gathering in wireless sensor networks,” in *International Conference on Complex, Intelligent and Software Intensive Systems, (CISIS),* March 2009, pp. 1048 –1053.
29. M. Huang and Y. H. Hu, “Collaborative sampling in wireless sensor networks,” in *Global Telecommunications Conference (IEEE GLOBECOM)*, Dec. 2010, pp. 1 –5.

**Chapter 9**

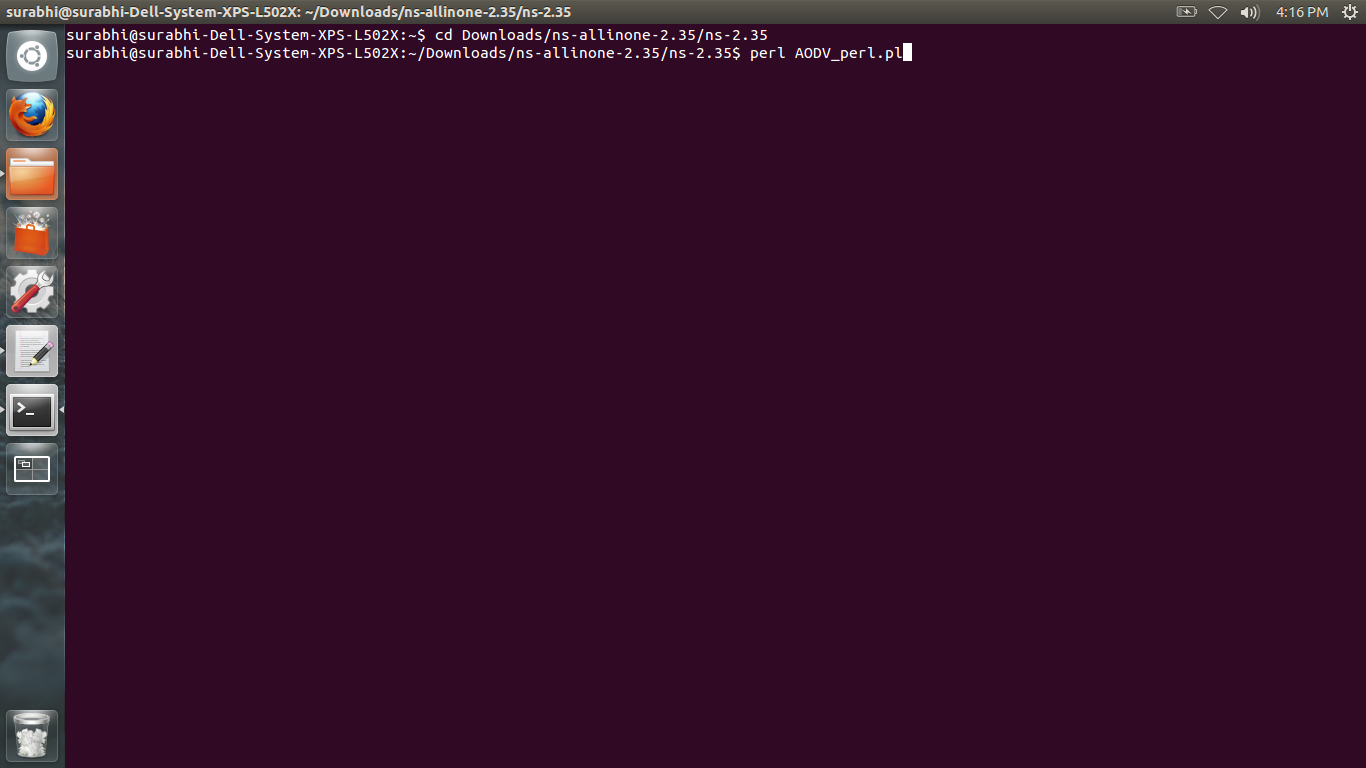
**APPENDIX**

**9.1 User Manual**

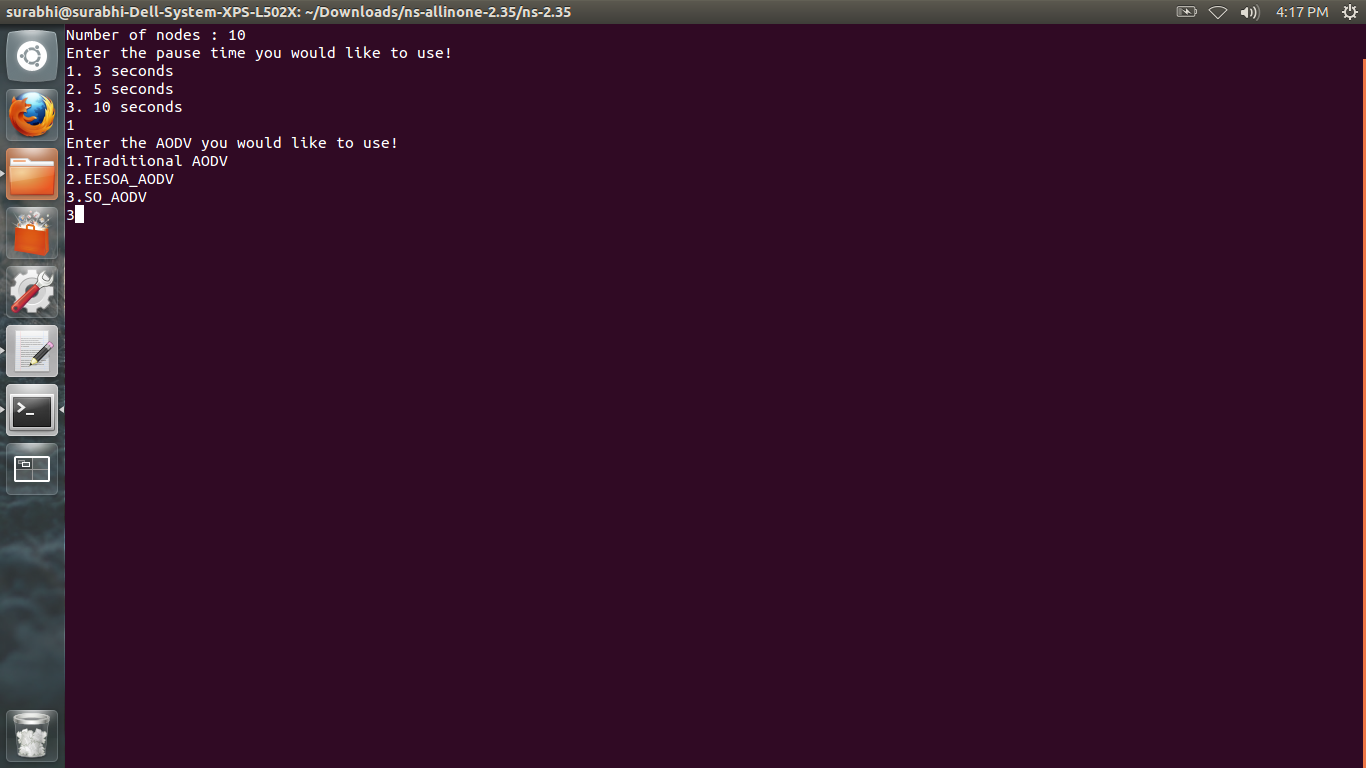
The user manual describes how to run SO\_AODV and to use the simulation to calculate end to end delay, network lifetime, convergence time and packet delivery ratio. It describes the options available, and explains the behaviour of the system for valid and invalid inputs.

The user interface is a command line interface, which prompts the user for inputs and performs input checking. If the input is invalid, the user interface informs the user of the error.

Tcl scripts for simulation of routing protocols, calculation of average end to end delay, network lifetime, convergence time and packet delivery ratio scripts are invoked through user interface; perl script. User interface is invoked by typing “perl AODV\_ perl.pl” at the shell; the snapshot is shown in the figure.

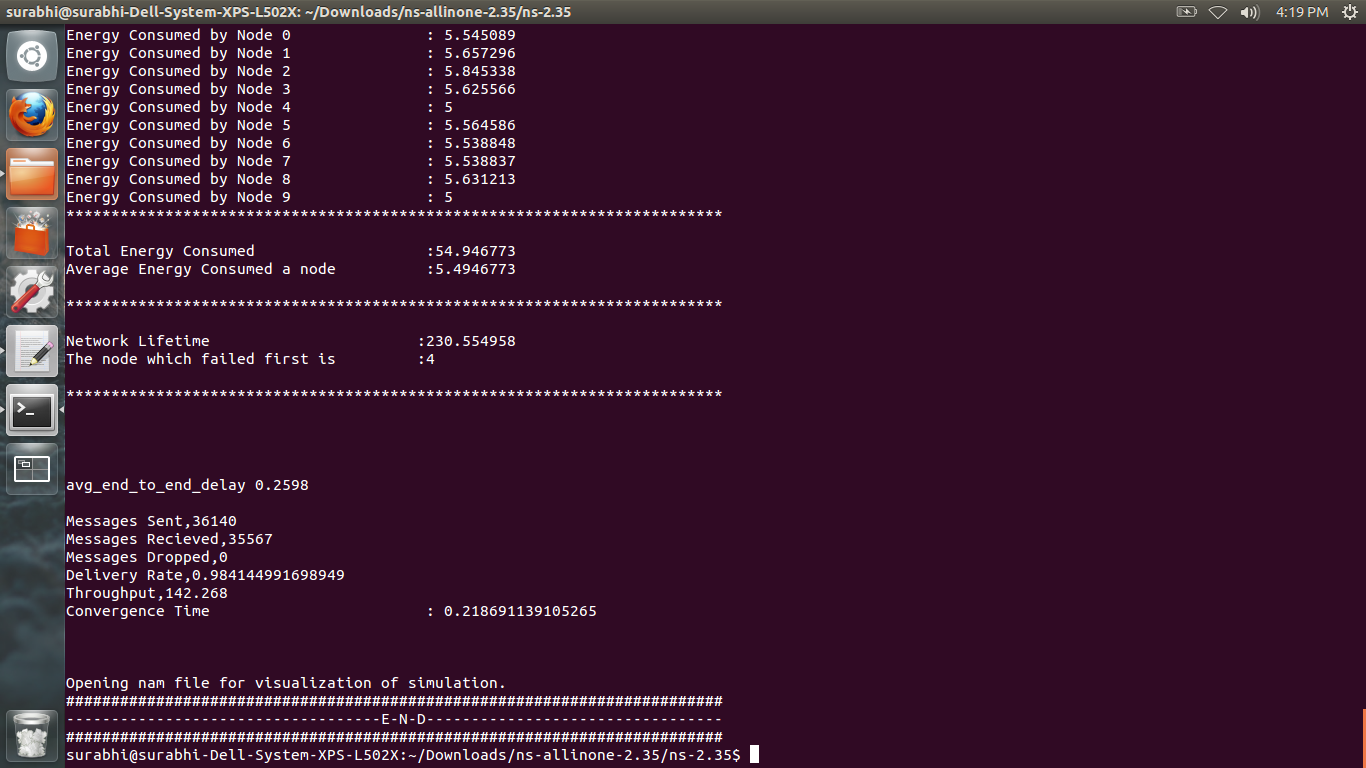
****Figure 9.1 Snapshot of User Interface

User interface accepts number of nodes, pause time and the type of protocol to be tested as the input to run the scenario. Snapshot of it is shown in the figure 9.2.

Figure 9.2 Snapshot to Generate Connection Pattern and Select Protocol

Connection pattern and mobility files along with routing protocol needed to be simulated are sent to tcl script for the simulation. If the tcl script uses old trace format, error is displayed at user interface.

User interface internally calls average end to end delay, convergence time, network lifetime and packet delivery ratio measuring modules. Trace file generated from simulation are sent to these modules as the arguments. Average end to end delay, network lifetime, convergence time and packet delivery ratio are displayed at the user interface, snapshot of this is given in the figure 9.3

Figure 9.3Snapshot of Calculated Average End to End Delay, Convergence Time, Network Lifetime, and Packet Delivery Ratio

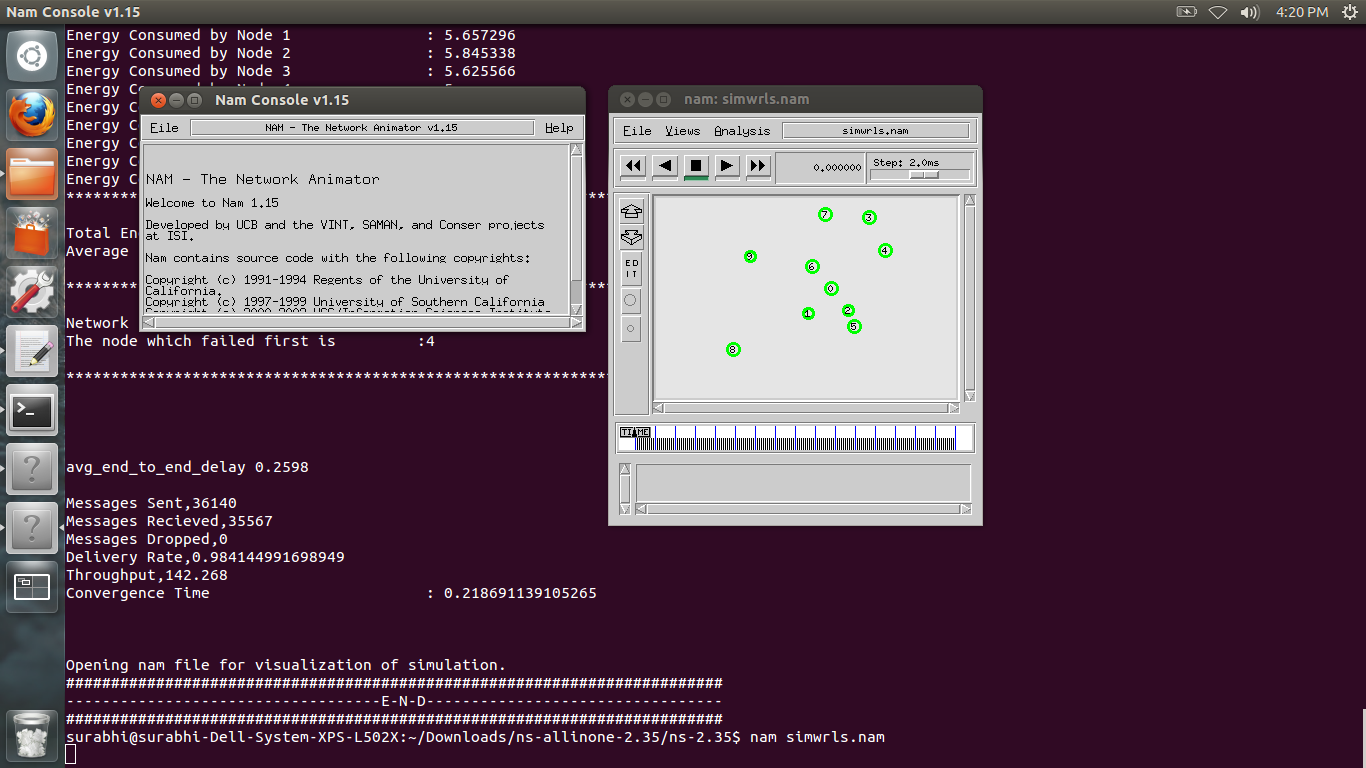
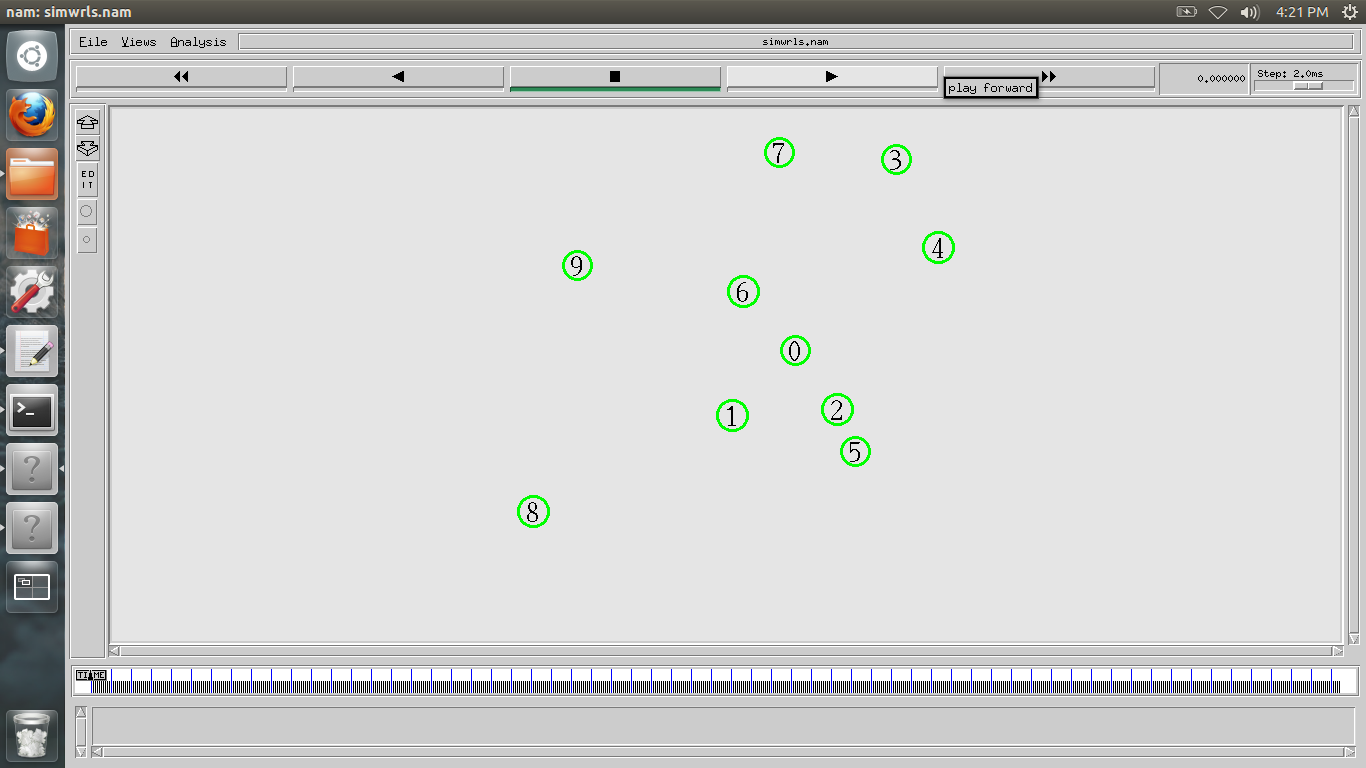
Figure 9.4 Snapshot that calls NAM internally

Figure 9.4 shows the snapshot that is called through user interface. ./nam takes nam file generated by tcl script as an argument to open nam file for the visualization of simulation through NAM.

Figure 9.5 shows NAM of SO\_AODV routing protocol simulated with 10 nodes.

Figure 9.5 NAM of SO\_AODV routing protocol simulated with 10 nodes

In order to evaluate SO\_AODV, EESOA and AODV using average end to end delay, network lifetime, convergence time and packet delivery ratio as performance metrics, a graph is plotted to compare SO\_AODV, EESOA and AODV by varying nodes, and varying pause time. These graphs are shown in chapter 6.

**9.2 NS2**

NS2 [8] is used as a network simulator tool to simulate the networks and see the result to measure the performance of the hybrid routing protocols. NS or Network Simulator is a discrete event simulator targeted at networking research. NS provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks.

NS began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB and USI/ISI. Currently ns development is supported through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researches including ACIRI. NS has always included substantial contribution from other researches, including wireless code from the UCB Daedelus and CMC Monarch project and Sun Microsystems.

The usage of NS-2 and other simulators: a network simulator is a software program that imitates the working of a computer network. In simulation, the computer network is typically modeled with devices, traffic etc. and the performance is analyzed. Typically, user can then customize the simulator to fulfill their specific analyzed needs. Simulators typically come with support for the most popular protocols in use today, such as IPv4, IPv6, UDP and TCP.

The components that come with the NS-2 packages are:

* NS – Simulator
* NAM – Network

Visual demonstration of NS output

* Preprocessing
  + Handwritten TCL or
  + Topology generator
* Post Analysis
  + Trace analysis using Perl/TCL/AWK/MATLAB

The User’s Perspective:

From the user’s perspective, NS-2 is an OTcl interpreter that takes an OTcl script as input and produces a trace file as output as in figure 2.1.

**Simulation – Otcl Script**

**OTcl Interpreter**

**C++**

**Libraries**

**Simulation Results**

**NS 2**

Figure 9.6 NS2 from user’s perspective

NS2 is a discrete event driven simulation where the physical activities are translated to events. As shown in figure 2.2, the events are queued and processed in the order of their scheduled occurrences. The time progresses as the events are processed.

**Time: 1.5 sec**

**Time: 1.7 sec**

**Time: 2.0 sec**

**Time: 1.8 sec**

**1**

**2**

Figure 9.7 Discrete Event Simulator

**C++ / OTcl:**

NS-2 code normally contains two sets of languages, namely C++ and OTcl. C++ is used for the creation of objects because of speed and efficiency. While OTcl is used as a front-end to setup the simulator, configure objects and schedule events because of its ease of use.

***Why two languages?*** NS2 uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols require a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important.

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

NS meets both of these needs with two languages, C++ and OTcl. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. *NS* (via tcl) provides glue to make objects and variables appear on both languages.

***Which language for what?*** Having two languages raises the question of which language should be used for what purpose.

It is advised is to use OTcl:

* For configuration, setup, and “one-time” stuff.
* If you can do what you want by manipulating existing C++ objects and use C++.
* If you are doing *anything* that requires processing each packet of a flow.
* If you have to change the behaviour of an existing C++ class in ways that weren’t anticipated.

**Simple example of Tcl script:**

#Create a simulator object

set ns [new Simulator]

#Open the nam trace file

setnf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the trace file

close $nf

#Executenam on the trace file

execnam –a out.nam&

exit 0

}

#Create two nodes

set n0 [$ns node]

set n1 [$ns node]

#Create a duplex link between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

This script creates two nodes. And the trace file is created called out.nam this file is can be executed to visualize using NAM editor. The agent can be attached to the node to create traffic in the nodes.

**NAM (Network Animator)**

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. NAM began at LBL. It has evolved substantially over the past few years. The NAM development effort was on ongoing collaboration with the VINT project. Figure 2.3 shows the visualization of the simulation of the above example.

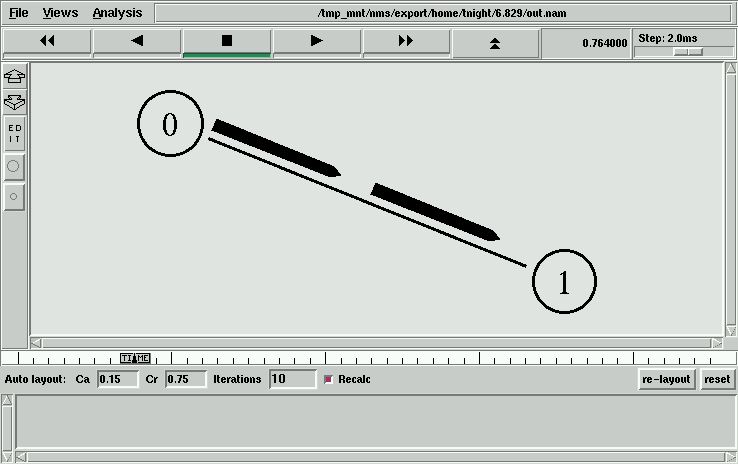


Figure 9.8 NAM for above Example

**9.2.1 NS2 versus NS3**

NS-3 is a new software development effort focused on improving upon the core architecture, software integration, models, and educational components of NS-2. The project commenced in July 2006 and the first release was made on June 30, 2008.

**9.2.1.1 Comparison between NS2 and NS3**

Table 9.1 Comparison between NS2 and NS3 [10]

|  |  |  |
| --- | --- | --- |
|  | **NS2** | **NS3** |
| **Programming languages** | * NS2 is implemented using a combination of oTCL (for scripts describing the network topology) and C++ (The core of the simulator). * This system was chosen in the early 1990s to avoid the recompilation of C++ as it was very time consuming using the hardware available at that time; oTCL recompilation takes less time than C++. * oTCL disadvantage: there is overhead introduced with large simulations. * oTCL is the only available scripting language. | * NS3 is implemented using C++. * With modern hardware capabilities, compilation time was not an issue like for NS2, NS3 can be developed with C++ entirely. * A simulation script can be written as a C++ program, which is not possible in NS2. * There is a limited support for Python in scripting and visualization. |
| **Memory management** | * NS2 requires basic manual C++ memory management functions. | * Because NS3 is implemented in C++, all normal C++ memory management functions such as new, delete, malloc, and free are still available. * Automatic de-allocation of objects is supported using reference counting (track number of pointers to an object); this is useful when dealing with Packet objects. |
| **Packets** | * A packet consists of 2 distinct regions; one for headers, and the second stores payload data. * NS2 never frees memory used to store packets until the simulation terminates, it just reuses the allocated packets repeatedly, as a result, the header region of any packet includes all headers defined as part of the used protocol even if that particular packet won't use that particular header, but just to be available when this packet allocation is reused. | * A packet consists of a single buffer of bytes, and optionally a collection of small tags containing meta-data. * The buffer corresponds exactly to the stream of bits that would be sent over a real network. * Information is added to the packet by using subclasses; Header, which adds information to the beginning of the buffer, Trailer, which adds to the end. * Unlike NS2, there is generally easy way to determine if a specific header is attached. |
| **Performance** | * The total computation time required to run a simulation scales better in NS3 than NS2. * This is due to the removal of the overhead associated with interfacing oTcl with C++, and the overhead associated with the oTcl interpreter. | * NS3 performs better than NS2 in terms of memory management. * The aggregation system prevents unneeded parameters from being stored, and packets don't contain unused reserved header space. |
| **Simulation output** | * NS2 comes with a package called NAM (Network Animator); it's a Tcl based animation system that produces a visual representation of the network described. | * NS3 employs a package known as PyViz, which is a python based realtime visualization package. |

**9.2.1.2 NS2 and NS3 existing models**

There is very limited number of models and contributed codes in NS3 in comparison with NS2 as shown in Figure 2.4.



Figure 2.4NS2 and NS3 existing core capabilities [10]

**9.2.2 Qualnet vs. NS2**

Table 9.2 Comparison betweenQualnet and NS2 [11]

| **Comparison of Qualnet and NS2 network simulators** | | | |
| --- | --- | --- | --- |
| **Qualnet** | | **NS2** | |
| 1. | Commercial simulator, based upon GloMoSim simulator. | 1. | Freely available for research and educational purposes. |
| 2. | Uses the parallel simulation environment for complex systems (PARSEC) for basic operations, hence can run on distributed machines. | 2. | Runs on a single machine, no parallel execution support in NS2. |
| 3. | Mainly developed for wireless scenario simulations, but wired networks also supported. | 3. | Mainly developed for wired networks, but its CMU extension facilitates the wireless network simulation. |
| 4. | Qualnet includes a graphical user interface for creating the model and its specification. So, it is very easy to specify small to medium networks by using the GUI. | 4. | To create and simulate a model, we have to specify all connections in a special model file manually. Uses OTcl for model file specifications. |
| 5. | Since it uses primarily Java for the GUI, it is available for Linux as well as for Windows. The simulator itself is the specified target system optimized C program. | 5. | It is designed for Unix systems but runs under Windows CygWin as well. |
| 6. | Faster simulation speeds and greater scalability are achievable through smart architecture and optimized memory management of Qualnet. | 6. | Not as fast and scalable. |
| 7. | Not used much in research as it is not freely available, hence lesser support (code samples etc.) available on Web. | 7. | Widely used for research, hence large number of resources available on Web. |
| 8. | Simulation of wireless sensor networks is supported in Qualnet 4.5 (using ZigBee library). | 8. | No such support available as of now. |
| 9. | Simulation of GSM mobile networks also supported. | 9. | GSM not supported in NS2. |
| 10. | Includes a variety of advanced libraries such as mesh networking, battery models, network security toolkit, and a large number of protocols at different layers. | 10. | These advanced libraries for wireless support are not available in NS2. |
| 11. | Includes a 3D visualizer for better visualization of a scenario. | 11. | 2D animator (NAM) is used with NS2. |
| 12. | Much easier for beginners who want to evaluate and test different existing routing protocols, as it can be done with GUI, without writing even a single line of code. | 12. | Requires knowledge of Tcl scripting before you can begin with NS2. |
| 13. | For implementing new protocols, Qualnet uses C/C++ and follows a procedural paradigm. | 13. | NS2 also uses C++ for new protocol/model development, but it uses object-oriented paradigm (usage of different classes) for programming. |

**9.3 Perl**

Perl is a high-level, general-purpose, interpreted, dynamic programming language. Perl was originally developed by Larry Wall, a linguist working as a systems administrator for NASA, in 1987, as a general purpose Unix scripting language to make report processing easier. Since then, it has undergone many changes and revisions and become widely popular among programmers. Larry Wall continues to oversee development of the core language, and its upcoming version, Perl 6.

Perl borrows features from other programming languages including C, shell scripting (sh), AWK, and sed. The language provides powerful text processing facilities without the arbitrary data length limits of many contemporary Unix tools, facilitating easy manipulation of text files. It is also used for graphics programming, system administration, network programming, applications that require database access and CGI programming on the Web. Perl is nicknamed “the Swiss Army chainsaw of programming languages” due to its flexibility and adaptability.

The Perl language includes a specialized syntax for writing regular expressions (RE, or regexes), and the interpreter contains an engine for matching strings to regular expressions. The regular-expression engine uses a backtracking algorithm, extending its capabilities from simple pattern matching to string capture and substitution. The regular-expression engine is derived from regex written by Henry Spencer.

Regular-expression syntax is extremely compact, owing to history. The first regular-expression dialects were only slightly more expressive than globs, and the syntax was designed so that an expression would resemble the text that it matches. This meant using no more than a single punctuation character or a pair of delimiting characters to express the few supported assertions. Over time, the expressiveness of regular expressions grew tremendously.