**AN EFFICIENT APPROACH FOR THE DETECTION AND**

**PREVENTION OF GRAY-HOLE ATTACKS IN VANETS**

**ABSTRACT:**

This research proposes an innovative and efficient approach for the detection and prevention of Gray-Hole attacks in Vehicular Ad Hoc Networks (VANETs). Gray-Hole attacks pose a significant threat to the reliability and security of VANETs by selectively dropping or modifying network packets. Our approach employs a combination of anomaly detection techniques and dynamic trust management to identify suspicious nodes exhibiting abnormal behavior indicative of Gray-Hole attacks. Upon detection, a proactive prevention mechanism is triggered to isolate and mitigate the impact of compromised nodes. The proposed solution is implemented in Network Simulator 2 (NS-2) using Tcl programming language, demonstrating its feasibility and effectiveness in enhancing the resilience of VANETs against Gray-Hole attacks. The results showcase improved network performance and security, making our approach a valuable contribution to the field of VANET security.

**CHAPTER 1**

**INTRODUCTION**

Wireless sensor networks (WSN), composed of numerous sensor nodes with small, low-power, inexpensive radios, have attracted a large amount of research that has led to interesting and innovative applications. However, challenging problems still exist. One of the most challenging problems in WSN is maintaining network connectivity to reliably deliver data to a specified point, or sink, in an energy-efficient manner. Disrupted connectivity, known as a “cut”, can lead to skewed data, ill-informed decisions and even entire network outages. It can also lead to memory and power exhaustion in disconnected nodes and network congestion in disconnected segments.

Such data loss and wasted resources can be avoided if a node can independently determine if a cut exists in the network. Cut detection algorithms attempt to recognize and locate cuts. Using a state based convergence mechanism, the current state-of-the-art cut detection algorithm, Distributed Source Separation Detection (DSSD), reliably detects arbitrarily-shaped cuts and allows individual nodes to perform cut detection autonomously. However, the algorithm suffers from a number of problems. First, DSSD fails to address security, a critical component of sensor deployments in unattended environments. Second, the algorithm requires a lengthy, iterative convergence process. Finally, all nodes participate in the frequent broadcasts required to achieve convergence.

This is cost-prohibitive with regards to power, especially in denser networks. In light of these problems, we propose an algorithm with two principal components. Outlier detection, a statistical data analysis technique, resolves the security threat where a malicious node injects erroneous data into the cut detection process. Using data analysis, outlier detection identifies malicious source data and provides a light-weight, energy-efficient mechanism to validate neighbour data. Additionally, we propose an improved cut detection algorithm called robust cluster-based cut detection. This algorithm divides the network into a set of location-based clusters. Cluster leaders form a virtual grid network and the cut detection algorithm runs on this high-level network. As the algorithm executes, leaders converge to some state. A leader finding inconsistency in its expected state informs its neighbors and the sink that a cut has happened.

**INTRODUCTION TO WIRELESS SENSOR NETWORKS**

Wireless Sensor Networks consists of a large number of low cost, low-power, low maintenance sensor nodes. Sensor nodes usually consist of sensing, communicating, computing, storing and power components. The sensor nodes are capable of storing data sensed from the environment according to their specific application, collaborating and computing the data to perform some specific task and communicating those data to the higher level or the base station (BS) for decision making. BSs have enhanced capabilities over simple sensor nodes and can do complex data processing. Sensor nodes are often randomly deployed in huge numbers over a large area to monitor physical conditions like temperature, humidity, intensity, vibration, pressure, motion, pollutants etc.

They are also deployed in military battlefield to track enemy movements. Based on the characteristics,

Wireless Sensor Networks has following main features:

a) Because of the limited energy supply for the wireless nodes, the wireless links between nodes in the Sensor Network are quite inconsistent for the communication participants.

b) Due to high memory and energy constraint, sensor nodes have restricted computational capacity and communication bandwidth.

c) They may remain unattended in hostile environment. Due to this, the major feature concerning WSN is energy efficient routing.

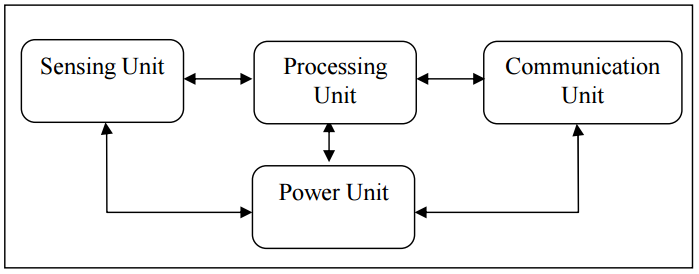
To offer energy efficient routing, one of the challenges is to ensure that the network is connected. The connectivity of the network can easily be disrupted due to unpredictable wireless channels, early depletion of node’s energy, and physical tampering by hostile users. Network disconnection, typically referred as a network cut, may cause a number of problems. For example, ill-informed decisions to route data to a node located in a disconnected segment of the network might lead to data loss, wasted power consumption, and congestion around the network cut.

Wireless sensor networks (WSN), sometimes called wireless sensor and actuator networks (WSAN), are spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. Today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

Wireless sensor nodes are equipped with sensing unit, a processing unit, communication unit and power unit. Each and every node is capable to perform data gathering, sensing, processing and communicating with other nodes. The sensing unit senses the environment, the processing unit computes the confined permutations of the sensed data, and the communication unit performs exchange of processed information among neighbouring sensor nodes.

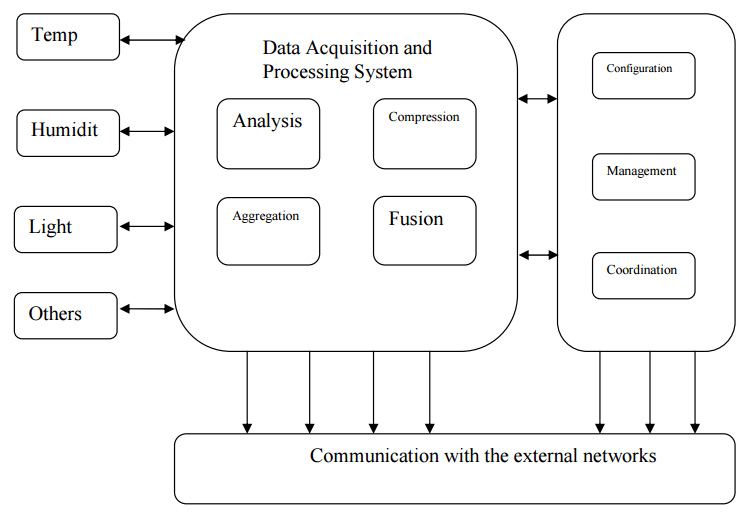
The basic building block of a sensor node is shown



**Basic Building Blocks of Sensor Node**

The sensing unit of sensor nodes integrates different types of sensors like thermal sensors, magnetic sensors, vibration sensors, chemical sensors, bio sensors, and light sensors. The measured parameters from the external environment by sensing unit of sensor node are fed into the processing unit. The analog signal generated by the sensors are digitized by using Analog to Digital converter (ADC) and sent to controller for further processing. The processing unit is the important core unit of the sensor node. The processor executes different tasks and controls the functionality of other components.

The required services for the processing unit are pre-programmed and loaded into the processor of sensor nodes. The energy utilization rate of the processor varies depending upon the functionality of the nodes. The variation in the performance of the processor is identified by the evaluating factors like processing speed, data rate, memory and peripherals supported by the processors. Mostly ATMEGA 16, ATMEGA 128L, MSP 430 controllers are used in commercial motes. The computations are performed in the processing unit and the acquired result is transmitted to the base station through the communication unit. In communication unit, a common transceiver act as a communication unit and it is mainly used to transmit and receive the information among the nodes and base station and vice versa. There are four states in the communication unit: transmit, receive, idle and sleep.



**Functionality of a Sensor Node**

The major characteristics of the sensor node used to evaluate the performance of WSN

**1. Fault tolerance**

Each node in the network is prone to unanticipated failure. Fault tolerance is the capability to maintain sensor network functionalities without any break due to sensor node failures.

**2. Mobility of nodes**

In order to increase the communication efficiency, the nodes can move anywhere within the sensor field based on the type of applications.

**3. Dynamic network topology**

Connection between sensor nodes follows some standard topology. The WSN should have the capability to work in the dynamic topology.

**4. Communication failures**

If any node in the WSN fails to exchange data with other nodes, it should be informed without delay to the base station or gateway node.

**5. Heterogeneity of nodes**

The sensor nodes deployed in the WSN may be of various types and need to work in a cooperative fashion.

**6. Scalability**

The number of sensor nodes in a sensor network can be in the order of hundreds or even thousands. Hence, WSN designed for sensor networks is supposed to be highly scalable.

**7. Independency**

The WSN should have the capability to work without any central control point.

**8. Programmability**

The option for reprogramming or reconfiguring should be available for the WSN to become adaptive for any dynamic changes in the network.

**9. Utilization of sensors**

The sensors should be utilized in a way that produces the maximum performance with less energy.

**10. Impracticality of public key cryptosystems**

The limited computation and power resources of sensor nodes often make it undesirable to use public key algorithms.

**11. Lack of aprior knowledge of post-deployment configuration**

If a sensor network is deployed via random distribution, the protocols will not be aware of the communication status between each nodes after deployment.

The following metrics are used to evaluate the performance of a WSN:

* Network coverage,
* Node coverage,
* Efficiency in terms of system lifetime,
* Effortless deployment,
* Data accuracy,
* System response time,
* Fault tolerance,
* Scalability,
* Network throughput,
* Sample rate,
* Security,
* The cost of the network and
* Network architecture used.

The individual sensor node in the WSN is evaluated using flexibility, robustness, computation, communication, security, synchronization, node size and cost.

The components of WSN system are sensor node, rely node, actor node, cluster head, gateway and base station which are explained below.

**Sensor node**

Capable of executing data processing, data gathering and communicating with additional associated nodes in the network. A distinctive sensor node capability is about 4-8 MHz, having 4 KB of RAM, 128 KB flash and preferably 916 MHz of radio frequency.

**Relay node**

It is a midway node used to communicate with the adjacent node. It is used to enhance the network reliability. A rely node is a special type of field device that does not have process sensor or control equipment and as such does not interface with the 7 process itself. A distinctive rely node processor speed is about 8 MHz, having 8 KB of RAM, 128 KB flash and preferably 916 MHz of radio frequency. Actor node: It is a high end node used to perform and construct a decision depending upon the application requirements. Typically these nodes are resource rich devices which are outfitted with high quality processing capabilities, greater transmission powers and greater battery life.

A distinctive actor node processor capability is about 8 MHz, having 16 KB of RAM, 128 KB flash and preferably 916 MHz of radio frequency. Cluster head: It is a high bandwidth sensing node used to perform data fusion and data aggregation functions in WSN. Based on the system requirements and applications, there will be more than one cluster head inside the cluster. A distinctive cluster head processor is about 4-8 MHz, having 512 KB of RAM, 4 MB flash and preferably 2.4 GHz of radio frequency. This node assumed to be highly reliable, secure and is trusted by all the nodes in the sensor network.

**Gateway**

Gateway is an interface between sensor networks and outside networks. Compared with the sensor node and cluster head the gateway node is most powerful in terms of program and data memory, the processor used, transceiver range and the possibility of expansion through external memory. A distinctive gateway processor speed is about 16 MHz, having 512 KB of RAM, 32 MB flash and preferably 2.4 GHz of radio frequency.

**Base station**

It is an extraordinary type of nodes having high computational energy and processing capability. 8 Attractive functionality of sensor nodes in a WSN includes effortlessness installation, fault indication, energy level diagnosis, highly reliablity, easy coordination with other nodes in the network, control protocols and simple network interfaces with other smart devices. In WSN, based on the sensing range and environment, the sensor nodes are classified into four groups, namely specialized sensing node, generic sensing node, high bandwidth sensing node and gateway node. The radio bandwidth for the sensor nodes are <50 Kbps, <100 Kbps, ≈500 Kbps and >500 Kbps respectively.

On board processing, computational rate and communication ranges differ from node to node in WSN. Particularly for some dedicated application sensor nodes with different capabilities are used. For example, smart specialized sensing nodes are preferred for special purpose devices, intelligent generic sensing node preferred for generic functions. For interconnectivity functions high end smart bandwidth sensing node and gateway nodes are preferred.

**CHAPTER2**

**LITERATURE REVIEW**

**TITLE:** Scalable and distributed detection analysis on wormhole links in wireless sensor networks for networked systems

**AUTHOR:** J. Padmanabhan and V. Manickavasagam

**PUBLICATION:** IEEE Access, vol. 6, pp. 17531763

**YEAR:** 2018

**DESCRIPTION:**

In this paper, a scalable and distributed scheme which uses sequential probability ratio test is proposed, to avoid single point failures and to handle high mobility, with no additional resource requirements. It is observed that wormholes are detected using few packets and detection is faster with increasing mobility. The system is highly customizable as system parameters can be chosen to balance the speed and accuracy of detection. System overheads in terms of communication, computation, and storage aspects are analyzed and presented.

**TITLE:** Wireless body area networks MAC protocol for energy efciency and extending lifetime

**AUTHOR:** X. Yang, L. Wang, and Z. Zhang

**PUBLICATION:** IEEE Sensors Lett., vol. 2, no. 1,

**YEAR:** 2018

**DESCRIPTION:**

 We design a novel medium access control (MAC) layer protocol to enhance the energy efficiency and extend the lifetime of body sensor nodes in WBANs. The proposed protocol utilizes a hybrid scheme by taking advantage of carrier-sense multiple accesses with the collision avoidance and time division multiple access schemes. In addition, we allocate the main transmission overhead at the personal station side and design a novel awaiting order state for sensor nodes to improve energy efficiency.

**TITLE:** LEACH: An energy efficient routing protocol using OmnetCC for Wireless Sensor Network

**AUTHOR:** D. M. Birajdar and S. S. Solapure

**Publication:** Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT), Coimbatore, India

**YEAR:** 2017

**DESCRIPTION:**

We propose a novel modified routing protocol in this paper. The newly proposed improved energy-efficient LEACH (IEE-LEACH) protocol considers the residual node energy and the average energy of the networks. To achieve satisfactory performance in terms of reducing the sensor energy consumption, the proposed IEE-LEACH accounts for the numbers of the optimal CHs and prohibits the nodes that are closer to the base station (BS) to join in the cluster formation. Furthermore, the proposed IEE-LEACH uses a new threshold for electing CHs among the sensor nodes, and employs single hop, multi-hop, and hybrid communications to further improve the energy efficiency of the networks.

**TITLE:** Improved LEACH protocol with cache nodes to increase lifetime of wireless sensor networks

**AUTHOR:** K. Roshan and K. R. Sharma

**Publication:** Proc. 2nd Int. Conf. Trends Electron. Informat. (ICOEI), Tirunelveli, India,

**YEAR:** 2018

In this research paper, the Improved LEACH protocol is further improved by deploying cache nodes in the network. In the Improved LEACH protocol with cache nodes, the cluster head are selected on the basis of distance and energy. The cluster heads transmit data to cache node which is nearest and has minimum access time. The simulation of proposed modal is done in MATLAB. To validate the results of Improved LEACH protocol is compared with Improved LEACH protocol with cache node.

**TITLE:** Routing Techniques in Wireless Sensor Networks: A Survey

**AUTHOR:** J. N. Al-Karaki and A. E. Kamal

**YEAR:** 2004

**DESCRIPTION:** Extensive usage of wireless sensor network (WSN) is the reason of development of many routing protocols. Recent advances in WSN now witness the increased interest in the potential use in applications like Military, Environmental, Health (Scanning), Space Exploration, Vehicular Movement, Mechanical stress levels on attached objects, disaster management, combat field reconnaissance etc. Sensors are expected to be remotely deployed in unattended environments. Routing as one key technologies of wireless sensor network has now become a hot research because the applications of WSN is everywhere, it is impossible that there is a routing protocol suitable for all applications. In this paper, the various routing protocol are classified and described. The growing interest in WSN and the continual emergence of new architectural techniques inspired surveying the characteristics, applications and communication protocols for such a technical area.

**TITLE:** Evolving a Hybrid K-Means Clusterin Algorithm for Wireless Sensor Network Using PSO and GAs,"

**AUTHOR:** A. Sheta and B. Solaiman

**YEAR:** 2015

**DESCRIPTION:** Wireless Sensor Networks (WSN) became an essential component in many real-life applications such as military, smart energy, commercial, health and many others. However, WSN still suffer many problems related to energy consumption. Clustering found to be an effective technique to solve the energy consumption problem for WSN by avoiding long distance communication. In this paper, we explore our initial idea on developing a hybrid clustering algorithm which has two folds 1) Use the K-Means unsupervised learning algorithms to select the sensors belonging to each cluster using an arbitrary number of clusters 2) Use Particle Swarm Optimization (PSO) and Genetic Algorithms (GAs) separately to select the best CHs. We name these two algorithms as KPSO and KGAs. The developed hybrid algorithms are tested over number of experiments with various layouts. KPSO provided better results compared to the KGAs.

**TITLE:** Minimization of Average Energy Consumption to Prolong Lifetime of Wireless Sensor Network

**AUTHOR:** A. Chunawale and S. Sirsikar

**YEAR:** 2015

**DESCRIPTION:** Wireless Sensor Network (WSN) is a network of sensor nodes that can sense the environment and send the information through wireless links to a sink. Wireless sensor nodes possess limited processing capability, storage and energy resources. The existence of sensor network depends on the life of sensor nodes i.e. ultimately on the energy consumption during its operation. Thus, in WSN, the efficient use of energy resources is very much necessary. Clustering is one of the approaches for energy saving in WSN. A cluster is a group of sensor nodes with one central entity named Cluster Head (CH). In this article, a new way of clustering for CH selection and cluster formation in WSNs is proposed; in which sensor network is divided into zones as per geographic locations of nodes. Clusters are formed within the zones by taking into account the residual energy of nodes and node distance. All cluster members send the sensed data to their respective CH. Hence, CH consumes more energy as it processes the collected data before forwarding it to Base Station (BS). Dropping of cluster head's residual energy below threshold value initiates cluster reformation. Performance analysis and simulation results are given with variations in number of nodes and transmission range. The obtained results show good performance of our algorithm in terms of reduced energy consumption, increased network lifetime and scalability of the network.

**CHAPTER3**

**CENTRALIZED ROUTING PROTOCOL FOR DETECTING**

**WORMHOLE ATTACKS IN WIRELESS SENSOR NETWORKS**

**EXISTING SYSTEM:**

In the current landscape of Vehicular Ad Hoc Networks (VANETs), security remains a critical concern due to the susceptibility to various attacks, including the Gray-Hole threat. Existing systems often rely on conventional security mechanisms, such as encryption and authentication, to safeguard communication channels. These methods may fall short in detecting and preventing sophisticated attacks like Gray-Hole, which involve nodes selectively manipulating or dropping packets without overtly violating encryption or authentication protocols. Traditional approaches may lack the adaptability required to address the dynamic nature of VANETs, where nodes frequently join or leave the network. There is a pressing need for an advanced system that goes beyond conventional security measures and provides a comprehensive solution to identify and mitigate Gray-Hole attacks effectively in VANETs.

**EXISTING TECHNIQUE:**

* Low-Energy Adaptive Clustering Hierarchy (LEACH)

**DISADVANTAGE:**

* Less Security
* Low Throughput
* More Losses

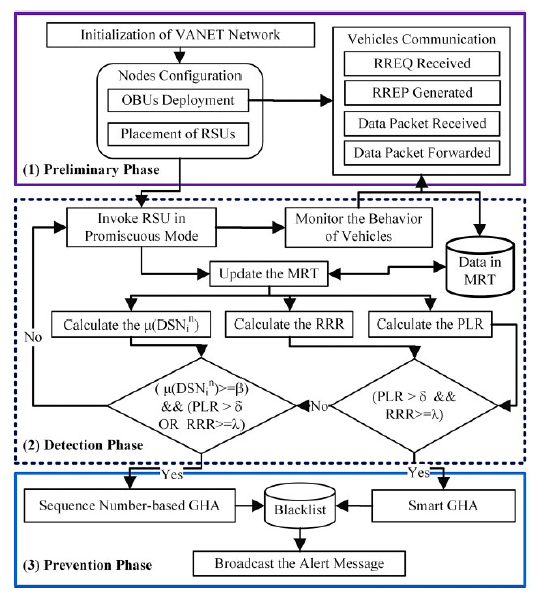
**OBJECTIVE:**

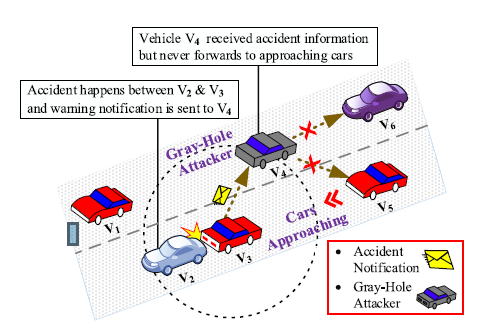
The primary objective of this research is to develop a robust solution for the detection and prevention of Gray-Hole attacks in Vehicular Ad Hoc Networks (VANETs). Gray-Hole attacks pose a severe threat to the integrity of communication in VANETs by selectively manipulating or dropping data packets. This research aims to enhance the security and reliability of VANETs by employing innovative anomaly detection techniques and dynamic trust management mechanisms. The ultimate goal is to provide an efficient and effective means of identifying and mitigating Gray-Hole attacks in real-time, thereby safeguarding the communication infrastructure of vehicular networks.

**PROPOSED SYSTEM:**

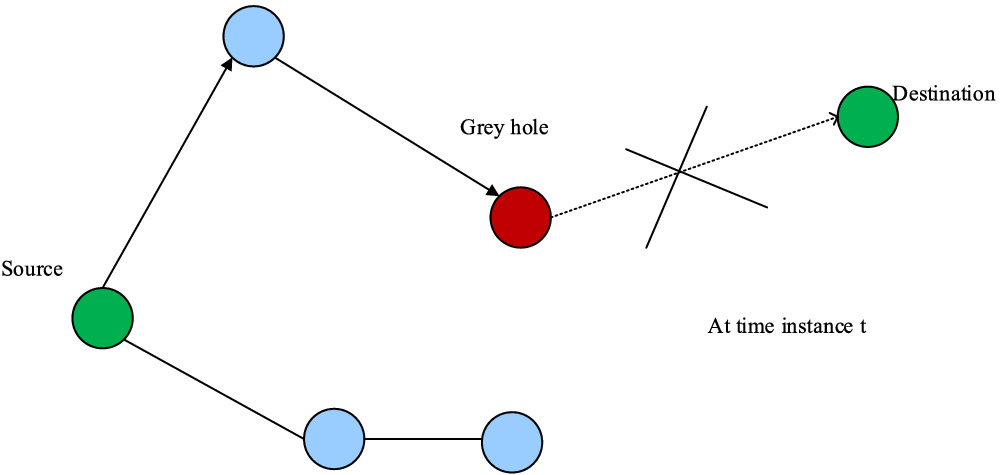
Our proposed algorithm uses an end-to-end time delay between sensor nodes and the BS to pin point links that are potentially under wormhole attacks and in turn be avoided in the shortest path. We will specify some assumptions in our sensor network. Then, we propose the concept of Time Ratio Threshold to detect wormhole links in the network. The foundation of MCRP is based on a single BS, which is a high-energy node with an unlimited amount of energy supply. Thus, MCRP employs the BS to facilitate network management to improve the performance of establishing a routing path from the source nodes to the sink.

**PROPOSED BLOCK DIAGRAM:**



**PROCESS DIAGRAM:**

**PROGRESS DIAGRAM:**



**PROPOSED TECHNIQUE:**

* Detection and Prevention of GHA’’ (DPGHA)

**PROPOSED SYSTEM ADVANTAGES:**

Improves the network scalability in terms of

* Energy consumption
* End-to-end delay
* Throughput
* Frame Delivery Ratio

**APLLICATIONS:**

WSN has wide application possibilities, such as

* temperature, pressure, humidity, and habitat monitoring
* disaster management
* Military reconnaissance
* forest re-tracking
* Building automation
* Security surveillance

**CHAPTER 4**

**METHODOLOGY**

**MODULES EXPLANATIONS:**

**Cluster:**

**Cluster Formation:**

Cluster formation is the process of grouping the nodes in to a single is known as cluster formation. This cluster has n number of nodes those are having same characteristics and also they are in the single area. This group of single nodes forms the network. There are lot of clusters comes under a base station.

**Cluster Head Selection:**

Initially, when clusters are being created, each node decides whether or not to become a cluster head for each round as specified by the original LEACH protocol. Each self-selected cluster head, broadcasts an advertisement (ADV) message using non-persistent carrier sense multiple access (CSMA) protocol. The message contains the header identifier (ID).

**Cluster Setup:**

Each non-cluster head node chooses one of the strongest received signal strength (RSS) of the advertisement as its cluster head, and transmits a join-request (Join-REQ) message back to the chosen cluster head. The information about the node’s capability of being a cooperative node, that is, its current energy status is added into the message. If a cluster head receives the advertisement message from another cluster head y, and if the received RSS exceeds a threshold, it will mark cluster head y as the neighboring cluster head and it record ID. If the sink receives the advertisement message, it will find the cluster head with the maximum RSS, and sends the sink-position message to that cluster head marking it as the target cluster head (TCH).

**Schedule Creation:**

After all the cluster heads has received the join-REQ message, each cluster head creates a time division multiple access (TDMA) schedule and broadcasts the schedule to its cluster members as in original LEACH protocol. This prevents collision among data messages and allows the radio of each non-cluster head node to be turned off until its allocated transmission time to save energy.

**Node Failure:**

**Route Discovery:**

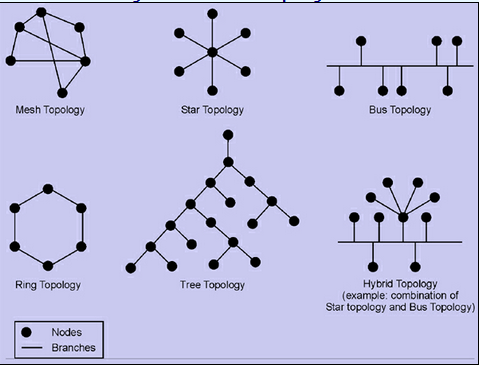
Route discovery is the process of selecting the route for the destination. It used RREQ and RREP for the route discovery process. In this process the RREQ is the broadcast message that is called as a route request. When the destination is identified, the destination sends the unicast RREP through the same path. This is used to find the best path between the source and destination.

**Energy Drain:**

After finding route, source can make communication with destination. For every data transmission and reception energy level will be decreased at initial energy level of node. Most Probably intermediate node energy can firstly dry due to trans-receiver characteristics. If its energy level is below 1 tends to node failure. It will not able to make communication with others.

**1. Topology Formation:**

Topology formation is an important issue in a wireless sensor network. Performance parameters such as energy consumption, network lifetime, data delivery delay, sensor field coverage depend on the network topology. Distances between nodes, physical interconnections, transmission rates, or signal types may differ between two networks, yet their topologies may be identical. Wireless sensor network mainly used for monitoring the events such as disaster tactical in military surveillance. It can be placed in two different manners 1) Regular manner and 2) Irregular manner. Mostly in irregular manner we are deploying sensors in irregularly is the chance for create a fault in sensor networks. Battery depletion is the attack, it drained the energy of sensors. Fault are created by these kind of attack also. It is considered that the sensors are distributed randomly over a large target region A, and designed to detect specified events. Each every one of sensors can sense specified events in its sensing range, and communicate with others in its transmission range.



**2. Isolated Node and Repairing Node**

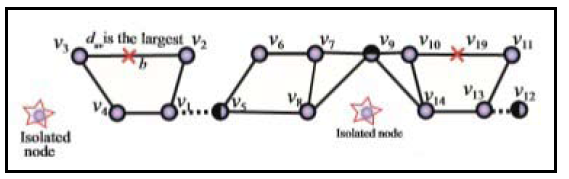
When a sensor does not have a path or communication to another sensor in any cluster, or may receive broken messages, the sensor is an isolated sensor. Some sensors often have irregular communication or unstable connectivity with others, due to reasons such as the location not being suitable for wireless sensor node communication, the environmental interference being very high, etc. Missing RREQ messages can be used to detect the failure of actors**.** After that it’s just check whether failed node is critical node or not.

**3. Worm whole detection**

In this step we finding smallest or large disjoint block**.** Search and Place algorithm is distributed and light weight. The boundary nodes are detected by that are struck nodes, which will launch the Warm whole discovery and the healing process even if these nodes are actually not stuck nodes. In order to differentiate the network border and hole border we are comparing coordinate value of nodes. In real wireless sensor network the node deployment is not uniform. So these networks contain some regions that are not covered by any sensor nodes, called holes. Topological hole and border detection methods are simple distributed approach to locate nodes near the boundary of the sensor field and the hole boundaries. This method purely relay on the topology of the communication graph. Here the only information available is, whether the nodes can communicate with each other or not. The topological methods never use the any location informations about the sensor nodes. The communication graph has node and edges if corresponding wireless station can communicate with each other. If two nodes can communicate with each other then they come under a common communication radius.

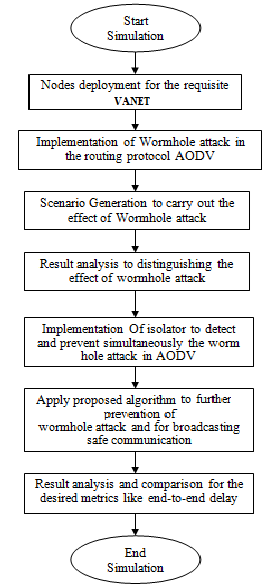
**4. Backup Sensor Selection**

A middle point between two sensors and of a cluster is an Repair points (RP) which is with the longest and irregular transmission distance and the link between r1 and r2 is vulnerable.Find Critical midpoint, Search an place algorithm implement to place backup sensor in an network. The RP’s identified are provided with the backup sensors of similar configuration as primary sensors. It improves the stability of the network by making system fault tolerant. The search for repair points runs continuously through all the clusters till all the RP’s are provided with backup sensors.



**Repair points**

**FLOW CHART:**

****

**CHAPTER 6**

**SOFTWARE REQUIREMENTS**

**IMPLEMENTATION TOOLS**

**4.1 NS-2 OVERVIEW**

**Operating system**: windows 7

**Simulator**: ns2

Ns (from network simulator) is a name for series of discrete event network simulators, specifically ns-1, ns-2 and ns-3. All of them are discrete-event network simulator, primarily used in research and teaching. ns-3 is free software, publicly available under the GNU GPLv2 license for research, development, and use. The goal of the ns-3 project is to create an open simulation environment for networking research that will be preferred inside the research communityIt should be aligned with the simulation needs of modern networking research. It should encourage community contribution, peer review, and validation of the software. Since the process of creation of a network simulator that contains a sufficient number of high-quality validated, tested and actively maintained models requires a lot of work, ns-3 project spreads this workload over a large community of users and developers.

In 1996-97, ns version 2 (ns-2) was initiated based on a refactoring by Steve McCanne. Use of Tcl was replaced by MIT's Object Tcl (OTcl), an object-oriented dialect Tcl. The core of ns-2 is also written in C++, but the C++ simulation objects are linked to shadow objects in OTcl and variables can be linked between both language realms. Simulation scripts are written in the OTcl language, an extension of the Tcl scripting language. Presently, ns-2 consists of over 300,000 lines of source code, and there is probably a comparable amount of contributed code that is not integrated directly into the main distribution (many forks of ns-2 exist, both maintained and unmaintained). It runs on GNU/Linux, FreeBSD, Solaris, Mac OS X and Windows versions that support Cygwin. It is licensed for use under version 2 of the GNU General Public License.

The general process of creating a simulation can be divided into several steps: Topology definition: to ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process. Model development: models are added to simulation (for example, UDP, IPv4, point-to-point devices and links, applications); most of the time this is done using helpers.

Node and link configuration: models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system. Execution: simulation facilities generate events, data requested by the user is logged. Performance analysis: after the simulation is finished and data is available as a time-stamped event trace. This data can then be statistically analysed with tools like R to draw conclusions. Graphical Visualization: raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

Ns-2 is often criticize because modelling is a very complex and time-consuming task, since it has no GUI and one needs to learn scripting language, queuing theory and modelling techniques. Also, of late, there have been complaints that results are not consistent (probably because of continuous changes in the code base) and that certain protocols have unacceptable bugs.

Ns-3 is often criticized[by whom?] for its lack of support for protocols (like WSN, MANET etc.) which were supported in ns-2, as well as for the lack of backward compatibility with ns-2. As with ns-2, ns-3 is also time consuming to learn and use compared to GUI-based simulators.

**DISCRETE EVENT SIMULATOR**

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. Although NS is fairly easy to use once you get to know the simulator, it is quite difficult for a first time user, because there are few user-friendly manuals. Even though there is a lot of documentation written by the developers which has in depth explanation of the simulator, it is written with the depth of a skilled NS user. The purpose of this project is to give a new user some basic idea of how the simulator works, how to setup simulation networks, where to look for further information about network components in simulator codes, how to create new network components, etc., mainly by giving simple examples and brief explanations based on our experiences. Although all the usage of the simulator or possible network simulation setups may not be covered in this project, the project should help a new user to get started quickly.

Ns 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 (local and satellite) networks. Ns began as a variant of the [REAL network simulator](http://www.cs.cornell.edu/home/skeshav/real/overview.html) in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the [VINT project](http://www.isi.edu/nsnam/vint/index.html) at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is support through DARPA with [SAMAN](http://www.isi.edu/saman/index.html) and through NSF with [CONSER](http://www.isi.edu/conser/index.html), both in collaboration with other researchers including [ACIRI](http://www.aciri.org/). Ns has always included substantal contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems.

**4.2 NS2 FEATURES**

NS is an object oriented discrete event simulator

* + - Single thread of control: no locking or race conditions
    - Simulator maintains list of events and executes in sequence order i.e.) one event after another.

Back end is C++ event scheduler

* + - Protocols mostly
    - Fast to run, more control

Front end is OTCL

* Creating scenarios, extensions to C++ protocols
* Fast to write and change

**PLATFORM TO RUN NS2**

* Unix and Unix like systems
* Linux (Use Fedora or Ubuntu versions)
* Free BSD
* SunOS/Solaris
* Windows 95/98/NT/2000/XP

**4.3 NS PROGRAMMING STRUCTURE**

* Create the event scheduler
* Turn on tracing
* Create network topology
* Create transport connections
* Generate traffic
* Insert errors

**EVENT SCHEDULAR**

In this Event scheduler while we processing many data at a time it will process one by one (i.e.) FIFO concept, so there is no congestion while transferring the packets.

**TRACE ANALYSIS**

Trace packets on individual link Trace file format.This section shows a trace analysis a trace analysis a tree analysis .Running the TCL script generates a NAM trace file that is going to be used as an input to NAM and a trace file called “ont.tr” that will be used for our simulation analysis.

Each trace line starts with an event (+,-, d, r)descriptor followed by the simulation time (in seconds)of that event, and from and to node , which identify the like on which the event occurred information in the line before flagsis packet type and size (in byte).

The next field is flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script. Each though fid field may not be used in a simulator, users can use this field for analysis purpose.The fid field is also used when specifying stream color for the NAM display. The next two fields are source and destination address in forms of “node. Port”.

The next field shows the network layer protocols’s packet sequence number. The last field shows the unique id of the packet.

**Create network topology**

Create network topology in Physical layer. ThePhysical Layer is the first and lowest layer in the seven-layer OSI model of computer networking. The implementation of this layer is often termed PHY. The Physical Layer which consists of the basic hardware and bit transmission over transmission medium in network. It is a fundamental layer underlying the logical data structures of the higher level functions in a network.

The Physical Layer defines the means of transmitting stream of bits rather than logical data packets over a physical link connecting networking nodes. The bit stream may be grouped into code words or symbols and converted to a physical that is transmitted over hardware.

**TRANSPORT CONNECTION**

Transport connection which performs in Transport layer. Transport layers are contained in both the TCP/IP which is the foundation of the INTERNET and the OSI model of general networking. The definitions of the Transport Layer are slightly different in these two models.

This article primarily refers to the TCP/IP model, in which TCP is largely for a convenient application programming interface to internet hosts, as opposed to the OSI model of definition interface. The most well-known transport protocol is the (TCP). It lent its name to the title of the entire internet protocol suite TCP/IP.

It is used for connection-oriented transmissions, whereas the connectionless user datagram suite (UDP) is used for simpler messaging transmissions. TCP is the more complex protocol, due to its statefull design incorporating reliable transmission and data stream services.

**GENERATE TRAFFIC (APPLICATION LAYER)**

In TCP/IP, the Application Layer contains all protocols and methods that fall into the realm of process-to-process communications via an Internet Protocol (IP) network using the Transport layer protocols to establish underlying host-to-host connections. In the OSI model, the definition of its Application Layer is narrower in scope, explicitly distinguishing additional functionality above the Transport Layer at two additional levels: session layer and presentation layer OSI specifies strict modular separation of functionality at these layers and provides protocol for each layer.

**INSERT ERRORS:** Start debugging of errors.

**4.4 NETWORK ANIMATOR (NAM)**

Network animator (NAM) is an animator tool for viewing network simulation and real world packet teaches. It supports topology layout, packet level animation and various DATA inspection tool. Before to use NAM, trace file need to be created. This trace file is usually generated by NS.

It contains topology information, e.g nodes and links, as well as packet traces. During a simulation, the user can produce topology configuration, layout information and packet trace using tracing events in NS. Once the trace file is generated, NAM can be used to animate it.

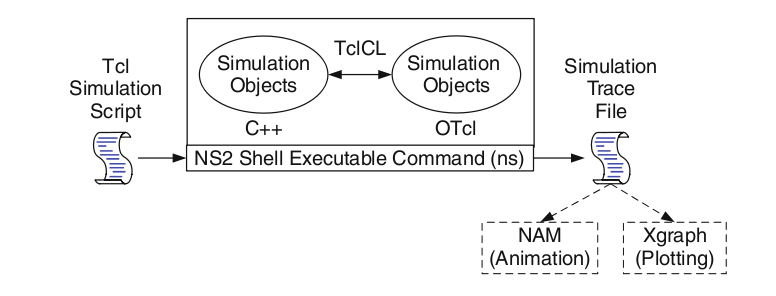
**ADVANTAGES OF NS2**

* NS is an ongoing effort of research and development.
* It does not require costly equipment.
* Simulators help in easy verification of protocols in less time, money
* Complex scenarios can be easily tested.
* NS offers support for simulating a variety of protocol suites and scenarios

**SIMULATION&LANGUAGE SPECIFICATION**

**INTRODUCTION:**

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors.

****Fig 5.1 Basic NS2 architecture

**MOTIVATION FOR SIMULATION:**

1. Cheap does not require costly equipment
2. Complex scenarios can be easily tested
3. Results can be quickly obtained – more ideas can be tested in a smaller timeframe
4. The real thing isn't yet available
5. Controlled experimental conditions – Repeatability helps aid debugging
6. Disadvantages: Real systems too complex to model.

**NS FEATURES:**

1. NS is an object oriented discrete event simulator – Simulator maintains list of events and executes one event after another
2. Single thread of control: no locking or race conditions
3. Back end is C++ event scheduler
   1. Protocols mostly
   2. Fast to run, more control
4. Front end is OTCL
   1. Creating scenarios, extensions to C++ protocols
   2. Fast to write and change

**NS PROGRAMMING STRUCTURES:**

1. Create the event scheduler
2. Turn on tracing
3. Create network topology
4. Create transport connections
5. Generate traffic
6. Insert errors

**EVENT SCHEDULER:**

In this Event scheduler while we processing many data’s at a time it will process one by one (FIFOconcept, so there is no congestion while transferring the packets.

**PACKETS**

It is the collection of data, whether header is called or not, all header files where present in the stack registers.

|  |
| --- |
| Cmn header |
| Ip header |
| Tcp header |
| Rtp header |
| Trace header |

Table 5.1 Stack Register of Packet

**TURN ON TRACING**

Trace packets on individual link Trace file format

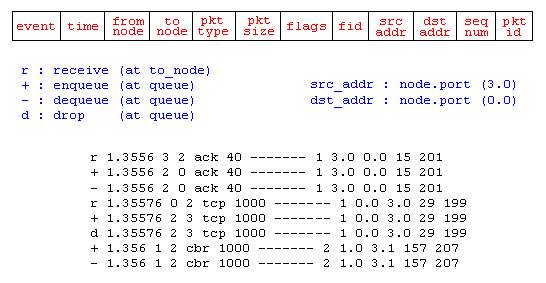


Fig 5.2 Trace of a packet

**CREATING NETWORK TOPOLOGY (PHYSICAL LAYER)**

The Physical Layer is the first and lowest layer in the seven-layer OSI model of computer networking. The implementation of this layer is often termed PHY.

The Physical Layer consists of the basic hardware transmission technologies of a network. It is a fundamental layer underlying the logical data structures of the higher-level functions in a network.

Due to the plethora of available hardware technologies with widely varying characteristics, this is perhaps the most complex layer in the OSI architecture.

The Physical Layer defines the means of transmitting raw bits rather than logical data packets over a physical link connecting networking nodes.

The bit stream may be grouped into code words or symbols and converted to a physical that is transmitted over hardware.

**TRANSPORT CONNECTION (TRANSPORT LAYER)**

Transport layers are contained in both the TCP/IP which is the foundation of the INTERNET and the OSI model of general networking. The definitions of the Transport Layer are slightly different in these two models.

This article primarily refers to the TCP/IP model, in which TCP is largely for a convenient application-programming interface to Internet hosts, as opposed to the OIS model of definition interface.

The most well-known transport protocol is the (TCP). It lent its name to the title of the entire Internet protocol suite TCP/IP.

It is used for connection-oriented transmissions, whereas the connectionless user datagram suite (UDP) is used for simpler messaging transmissions. TCP is the more complex protocol, due to its design incorporating reliable transmission and data stream services.

**GENERATE TRAFFIC (APPLICATION LAYER)**

In TCP/IP, the Application Layer contains all protocols and methods that fall into the realm of process-to-process communications via an Internet Protocol (IP) network using the Transport layer protocols to establish underlying host-to-host connections.

In the OSI model, the definition of its Application Layer is narrower in scope, explicitly distinguishing additional functionality above the Transport Layer at two additional levels: session layer and presentation layer OSI specifies strict modular separation of functionality at these layers and provides protocol for each layer.

**4.1.1 MOTIVATION FOR SIMULATIONS**

1. Cheap does not require costly equipment

2. Complex scenarios can be easily tested

3. Results can be quickly obtained – more ideas can be tested in a smaller timeframe

4. The real thing isn't yet available

5. Controlled experimental conditions

– Repeatability helps aid debugging

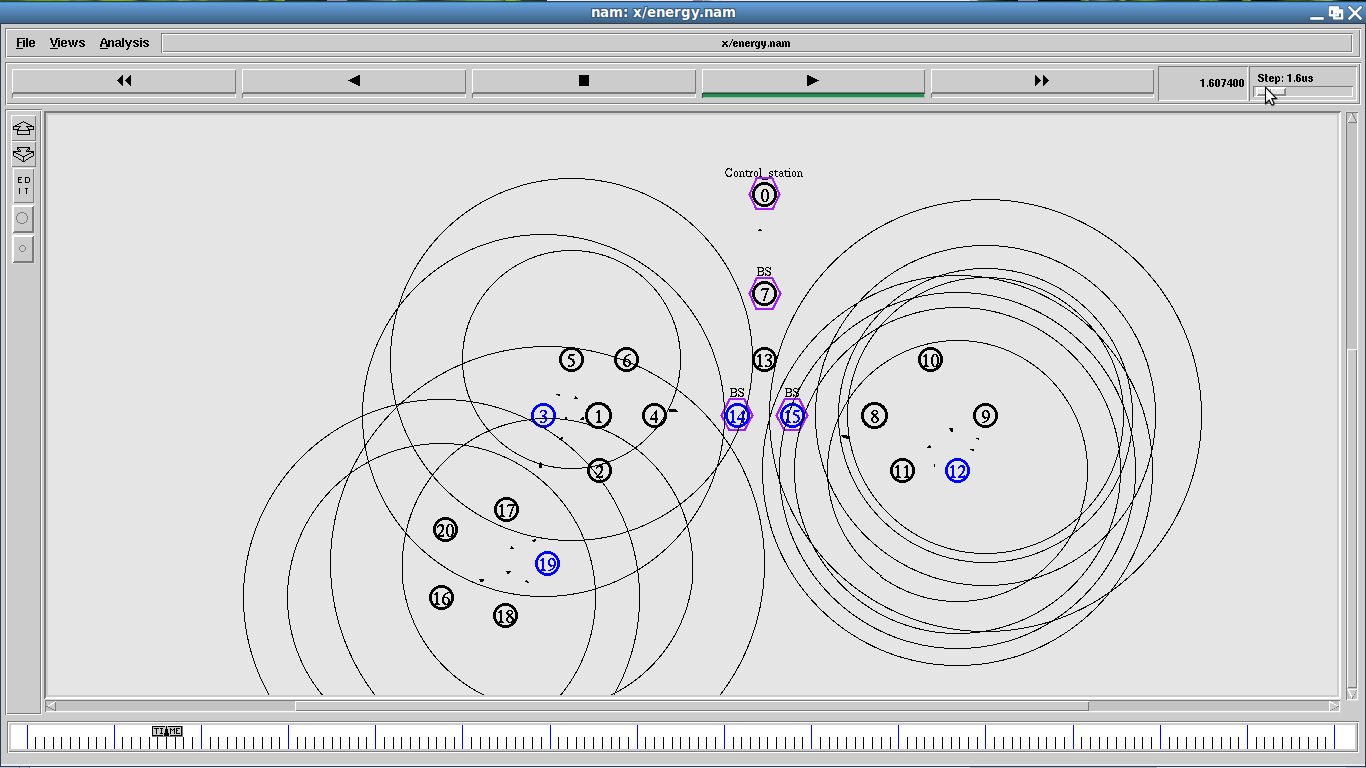
6. Disadvantages: Real systems too complex to model

**MODEL OUTPUT:**

* There are three types of outputs
  + Nam window
  + Tracing
  + Xgraph

**NETWORK ANIMATOR**

Network Animator (NAM) provides the packet level simulation output in a graphical manner. Network Animator is an animation tool for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation and various data inspection tools. Before starting to use NAM, a trace file needs to be created. This trace file is usually generated by NS. It contains topology information, e.g. nodes and links, as well as packet traces. During a simulation, the user can produce topology configurations, layout information and packet traces using tracing events in NS. Once the trace file is generated, NAM can be used to animate it as shown in figure 5.1. Upon startup, NAM will read the trace file, create topology, pop up a window, do layout if necessary and then pause at time 0. Through its user interface, NAM provides control over many aspects of animation.



* Fig 5.1: Simulated output of NAM

**Tracing**

Trace packets on individual link Trace file format

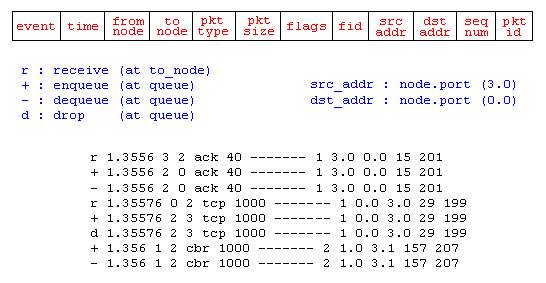


Figure 4.3: Model

**Xgraph**

Provides the throughput comparison based on a graph which will be generated automatically based on the TCL coding as shown in figure 5.2.

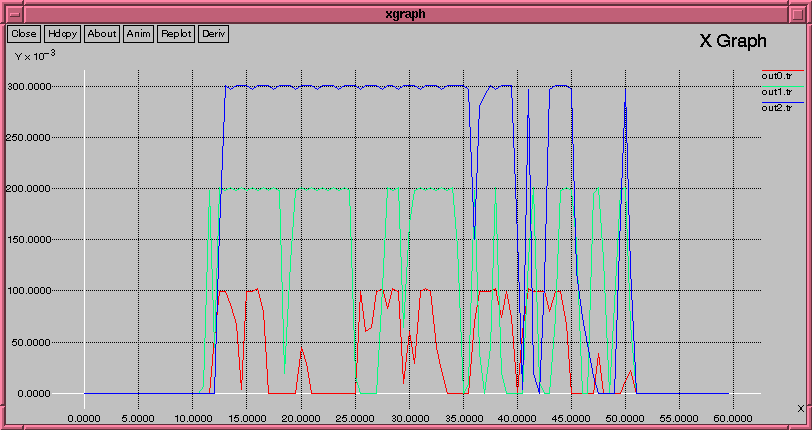


Fig 5.2: Xgraph for simulated output

**NS FEATURES**

● NS is an object oriented discrete event simulator

– Simulator maintains list of events and executes one event after another

– Single thread of control: no locking or race conditions

● Back end is C++ event scheduler

– Protocols mostly

– Fast to run, more control

● Front end is OTCL

– Creating scenarios, extensions to C++ protocols

– Fast to write and change

**SOFTWARE INSTALTION**

**INSTALLATIONN OF NS2**

After completing the installation of ubuntu 10.04, update all the things in that. How to update means Goto the top menu**, system-Administration-update manager**

**STEP 1**

* Open the terminal. For this Goto **Application-Accessories-Terminal** (or press Ctrl+alt+T)
* Then paste the following command

**sudo apt-get install build-essential autoconf automake libxmu-dev gcc-4.3**

**STEP 2**

Then u need to some files. So For this, open the Ns-allinone2.34 folder on the desktop, in that open folder OTCL1.13. In that folder, you need to modify the following files.

* Modify **Makefile.in** file
  + - In this file, u need to replace **CC= @CC@**
    - With **CC= gcc-4.3**
    - To replace this press **ctrl+H.**
    - Then save the document.
* Modify **configure.in** and **configure** files.

In these two files u need to modify. Means u need to replace **ld -shared** with **gcc –shared.** For replacing **ctrl+H** then one dialog box will come. Then save the document.

**STEP 3**

* Open the **ns-allinone2.34** folder which is present on the desktop,

Then press **ctrl+L.** Then copy the path on the address bar.

**STEP 4**

* Then open **Terminal** and do the following.
* Type **$cd “paste the path here” without quotes and press enter**.
* Then Type **$./install**
* Then wait for some time, finally if successfully installed, it will display a message as **“for Related Posts”**. Otherwise some problems might have occurred.

**STEP 5**

* Then execute the following command in the **Terminal**

**$ gedit ~/.bashrc**

* Then one document will be opened automatically.
* Then delete all the content in that document and paste the following code in it.

**# LD\_LIBRARY\_PATH  
OTCL\_LIB=/home/programmer/ns-allinone-2.33/otcl-1.13  
NS2\_LIB=/home/programmer/ns-allinone-2.33/lib  
X11\_LIB=/usr/X11R6/lib  
USR\_LOCAL\_LIB=/usr/local/lib  
export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$OTCL\_LIB:$NS2\_LIB  
:$X11\_LIB:$USR\_LOCAL\_LIB  
  
# TCL\_LIBRARY  
TCL\_LIB=/home/programmer/ns-allinone-2.33/tcl8.4.18/library  
USR\_LIB=/usr/lib  
export TCL\_LIBRARY=$TCL\_LIB:$USR\_LIB  
# PATH  
XGRAPH=/home/programmer/ns-allinone-2.33/bin:/home/programmer/ns-allinone-2.33/tcl8.4.18/unix:/home/programmer/ns-allinone-2.33/tk8.4.18/unix:/home/programmer/ns-allinone-2.33/xgraph-12.1/  
NS=/home/programmer/ns-allinone-2.33/ns-2.33/  
NAM=/home/programmer/ns-allinone-2.33/nam-1.13/  
export PATH=$PATH:$XGRAPH:$NS:$NAM**

Then after doing this, press Ctrl+H and replace the following.

**/home/programmer/ns-allinone-2.33**

Replace this with

**/home/your system name/Desktop/ns-allinone-2.34**

* And in that document, at the 2nd line from the last, put **1.14** instead of **1.13**
* And at the 3rd line from the last, make the change as **ns-2.34** instead of **ns-2.33**
* **Then save the document and close it.**

**STEP 6**

* Then finally copy the following command into the **Terminal**

**“sudo ln -s /home/programmer/ns-allinone-2.33/ns-2.33/ns /usr/bin/ns”**

* After pasting it into the **Terminal,** just make the following modifications and then press Enter.
* **Make 2.33 as 2.34**
* And in the place of **“programmer”** in that command, just type your system name. Then the command should be like this.
  + **“sudo ln -s /home/ur system name/Desktop/ns-allinone-2.34/ns- 2.34/ns /usr/bin/ns”**
* Then press Enter.
* It will ask for password, enter the password**.**

**STEP 7**

* Then finally type **“ns”** in the **Terminal** and press Enter**.**
* Then “%” symbol will be displayed.
* Then it was successfully installed. Otherwise some problems might have occurred in the installation process**.**

**STEP 8**

* Then finally execute the following command in the Terminal
* Type **“sudo apt-get install xgraph”** and press enter.
* Finished installing Network Simulator.

**SYSTEM MODELLING:**

System modeling refers to an act of representing an actual system in a simply way. System modeling is extremely important in system design and development, since it gives an idea of how the system would perform if actually implemented.

Traditionally, there are two modeling approaches: analytical approach and simulation approach.

**ANALYTICAL APPROACH:**

The general concept of analytical modeling approach is to first come up with a way to describe a system mathematically with the help of applied mathematical tools such as queuing and probability theories, and then apply numerical methods to gain insight from the developed mathematical model. When the system is simple and relatively small, analytical modeling would be preferable (over simulation). In this case, the model tends to be mathematically tractable. The numerical solutions to this model in effect require lightweight computational efforts.

If properly employed, analytical modeling can be cost-effective and can provide an abstract view of the components interacting with one another in the system. However, if many simplifying assumptions on the system are made during the modeling process, analytical models may not give an accurate representation of the real system.

**SIMULATION APPROACH:**

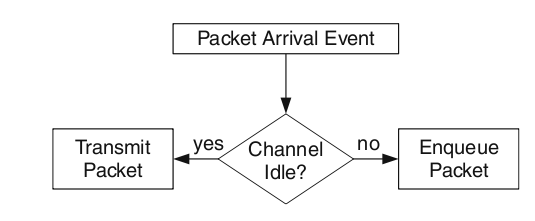
Simulation is a process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and/or evaluating various strategies for the operation of the system.

Simulation is widely-used in system modeling for applications ranging from engineering research, business analysis, manufacturing planning, and biological science experimentation, just to name a few.

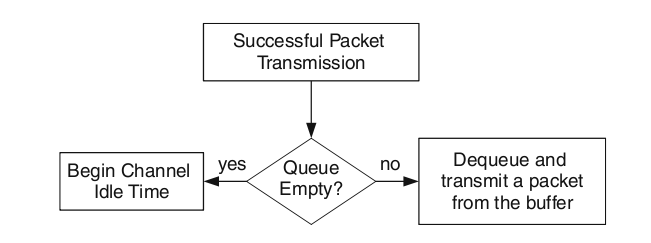
Compared to analytical modeling, simulation usually requires less abstraction in the model (i.e., fewer simplifying assumptions) since almost every possible detail of the specifications of the system can be put into the simulation model to best describe the actual system.

When the system is rather large and complex, a straightforward mathematical formulation may not be feasible. In this case, the simulation approach is usually preferred to the analytical approach.

In common with analytical modeling, simulation modeling may leave out some details, since too many details may result in an unmanageable simulation and substantial computation effort. It is important to carefully consider a measure under consideration and not to include irrelevant detail into the simulation.



Packet Arrival event

****

Successful Packet Transmission

**CHAPTER 5**

**PROGRAMMING AND OUTPUT RESULTS**

**MAIL TCL PROGRAM:**

puts "Enter option Leach(0) or Lightweight(1) or comp(2)"

set opt\_ [gets stdin]

set bopt\_ 1

if {$opt\_==0} {

set bopt\_ 0

}

if {$opt\_==2} {

exec xgraph en\_grp\_0 en\_grp\_1 -x "time" -y "energy" -lw 2 &

exec xgraph PDF\_0 PDF\_1 -x "time s" -y "pkts" -lw 2 &

#exec xgraph TPDF\_0 TPDF\_1 TPDF\_2 -x "time s" -y "pkts" -nl -bar -brw 0.5 &

exec xgraph OH\_0 OH\_1 -x "time s" -y "pkts" -lw 2 &

exec xgraph delay\_0 delay\_1 -x "tech" -y "Delay rate" -bg "black" -fg "white" -bar -brw 1 -nl &

exec xgraph th\_0 th\_1 -x "time" -y "Packet size" -t "Throughput" -bg "black" -fg "white" -lw 3 &

exit

}

# ======================================================================

# Define options

# ======================================================================

set val(chan) Channel/WirelessChannel ;# Channel Type

set val(prop) Propagation/TwoRayGround ;#Nakagami

;#TwoRayGround ;# radio-propagation model

set val(netif) Phy/WirelessPhy

set val(ant) Antenna/OmniAntenna ;# antenna model

set val(mac) Mac/802\_11

set val(ll) LL ;# link layer type

set val(rp) DSDV ;# AODV;# routing protocol

set val(ifq) Queue/DropTail/PriQueue ;# ; #; CMUPriQueue ;# # Queue/DropTail/PriQueue for AODV & DSDV CMUPriQueue for DSR

set val(ifqlen) 50 ;# max packet in ifq

set val(nn) 51 ;# number of mobilenodes

set val(x) 500

set val(y) 500

#set val(nam) out.nam

#set val(traffic) ftp ;# cbr/poisson/ftp

#============================================================================

set ns\_ [new Simulator]

set ns $ns\_

set ran [new RNG] ;#random number generator

set tracefd [open out.tr w]

set namtrace [open out.nam w]

set En\_fl [open en\_grp\_$opt\_ w]

$ns\_ namtrace-all-wireless $namtrace $val(x) $val(y)

$ns\_ trace-all $tracefd

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

set god\_ [create-god 100]

#=============================================================================

if {1} {

set txp 0.001

$ns\_ node-config -energyModel EnergyModel \

-rxPower 0.0005 \

-txPower $txp \

-initialEnergy 100 \

-idlePower 0.5

}

#========================================================================

$ns\_ node-config -adhocRouting $val(rp) \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channelType $val(chan)\

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace ON \

-movementTrace ON

#============================================

$ns\_ color 1 red

$ns\_ color 2 green

$ns\_ color 3 purple

#==========================================

source trace

source Lightweight.tcl

source Leach.tcl

set fg 300

set PS [new Pan\_Sim]

for {set i 0} {$i<$val(nn)} {incr i} {

set node\_($i) [$ns\_ node]

set nd\_($i) [$PS Node]

set CH\_lst [list]

$node\_($i) color black

}

$node\_(0) set X\_ 83

$node\_(0) set Y\_ 168

$node\_(1) set X\_ 59

$node\_(1) set Y\_ 110

$node\_(2) set X\_ 100

$node\_(2) set Y\_ 130

$node\_(3) set X\_ 120

$node\_(3) set Y\_ 100

$node\_(4) set X\_ 140

$node\_(4) set Y\_ 160

$node\_(5) set X\_ 155

$node\_(5) set Y\_ 270

$node\_(5) set X\_ 1

$node\_(5) set Y\_ 270

set y 200

set z 100

set n 6

#$ns\_ at 7.1 "$nd\_(1) make-failure"

for {set i 200} {$i<280} {incr i +70} {

for {set j 100} {$j<180} {incr j +70} {

$node\_($n) set X\_ $i

$node\_($n) set Y\_ $j

incr n

if {$n >=$val(nn)} {break}

}

if {$n >=$val(nn)} {break}

}

$node\_(10) set X\_ 237

$node\_(10) set Y\_ 134

$node\_(11) set X\_ 165

$node\_(11) set Y\_ 280

$node\_(11) set X\_ -25.4646

$node\_(11) set Y\_ 473.608

set n 12

for {set i 310} {$i<390} {incr i +70} {

for {set j 250} {$j<330} {incr j +70} {

$node\_($n) set X\_ $i

$node\_($n) set Y\_ $j

incr n

if {$n >=$val(nn)} {break}

}

if {$n >=$val(nn)} {break}

}

$node\_(16) set X\_ 344

$node\_(16) set Y\_ 290

$node\_(17) set X\_ 100

$node\_(17) set Y\_ 399

$node\_(18) set X\_ 170

$node\_(18) set Y\_ 402

$node\_(19) set X\_ 100

$node\_(19) set Y\_ 481

$node\_(20) set X\_ 170

$node\_(20) set Y\_ 481

$node\_(21) set X\_ 135

$node\_(21) set Y\_ 446

$node\_(22) set X\_ -87.9252

$node\_(22) set Y\_ 425.696

$node\_(23) set X\_ -58.3266

$node\_(23) set Y\_ 444.847

$node\_(24) set X\_ -106.207

$node\_(24) set Y\_ 442.236

$node\_(25) set X\_ -55.715

$node\_(25) set Y\_ 420.472

$node\_(26) set X\_ 194.132

$node\_(26) set Y\_ 237.658

$node\_(27) set X\_ 242.883

$node\_(27) set Y\_ 245.493

$node\_(28) set X\_ 221.119

$node\_(28) set Y\_ 268.128

$node\_(29) set X\_ 279.446

$node\_(29) set Y\_ 237.659

$node\_(30) set X\_ 266.387

$node\_(30) set Y\_ 270.74

$node\_(31) set X\_ 227.213

$node\_(31) set Y\_ 329.066

$node\_(32) set X\_ 208.061

$node\_(32) set Y\_ 310.784

$node\_(33) set X\_ 181.944

$node\_(33) set Y\_ 321.23

$node\_(34) set X\_ 151.475

$node\_(34) set Y\_ 346.476

$node\_(35) set X\_ 152.455

$node\_(35) set Y\_ 305.017

$node\_(36) set X\_ -52.8857

$node\_(36) set Y\_ 137.329

$node\_(37) set X\_ -29.2721

$node\_(37) set Y\_ 190.323

$node\_(38) set X\_ -75.5199

$node\_(38) set Y\_ 161.921

$node\_(39) set X\_ -24.7018

$node\_(39) set Y\_ 163.336

$node\_(40) set X\_ -63.6587

$node\_(40) set Y\_ 189.561

$node\_(41) set X\_ -110.124

$node\_(41) set Y\_ 248.432

$node\_(42) set X\_ -161.813

$node\_(42) set Y\_ 260.619

$node\_(43) set X\_ -122.856

$node\_(43) set Y\_ 200.225

$node\_(44) set X\_ -178.244

$node\_(44) set Y\_ 229.824

$node\_(45) set X\_ -148.102

$node\_(45) set Y\_ 228.736

$node\_(46) set X\_ -102.289

$node\_(46) set Y\_ 299.794

$node\_(47) set X\_ -69.6437

$node\_(47) set Y\_ 350.504

$node\_(48) set X\_ -113.171

$node\_(48) set Y\_ 356.924

$node\_(49) set X\_ -160.724

$node\_(49) set Y\_ 326.019

$node\_(50) set X\_ -117.85

$node\_(50) set Y\_ 324.082

$nd\_(0) set-en 70

$nd\_(7) set-en 70

$nd\_(0) set drt\_ 0.015

$nd\_(1) set drt\_ 0.018

$nd\_(1) set drt\_ 0.021

$nd\_(6) set drt\_ 0.014

$nd\_(7) set drt\_ 0.018

$nd\_(8) set drt\_ 0.022

set BS 5

#$ns\_ at 1.0 "$nd\_(5) label BaseStation"

$ns\_ at 0 "$node\_(5) label BS"

$ns\_ at 0 "$node\_(5) color red"

#$ns\_ initial\_node\_pos $node\_(5) 30

#set Mc 11

$ns\_ at 1.1 "$nd\_(0) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(1) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(2) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(3) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(4) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(6) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(7) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(8) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(9) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(10) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(16) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(12) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(13) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(14) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(15) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(21) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(17) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(18) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(19) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(20) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(22) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(23) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(24) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(25) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(11) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(26) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(27) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(28) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(29) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(30) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(31) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(32) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(33) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(34) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(35) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(36) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(37) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(38) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(39) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(40) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(41) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(42) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(43) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(44) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(45) send\_Data $BS"

$ns\_ at 1.1 "$nd\_(46) send\_Data $BS"

$ns\_ at 1.11 "$nd\_(47) send\_Data $BS"

$ns\_ at 1.12 "$nd\_(48) send\_Data $BS"

$ns\_ at 1.13 "$nd\_(49) send\_Data $BS"

$ns\_ at 1.14 "$nd\_(50) send\_Data $BS"

if {$opt\_>0} {

$ns\_ at 1.3 "$nd\_(6) make-failure"

} else {

$ns\_ at 1.3 "$nd\_(8) make-failure"

$ns\_ at 1.3 "$nd\_(50) make-failure"

$ns\_ at 1.3 "$nd\_(33) make-failure"

}

proc Grph {} {

global En\_fl nd\_ ns\_

set en 100

set nd x

foreach n [array names nd\_] {

set ene [$nd\_($n) set energy]

if {$en>$ene && ($n!=5 && $n!=11) && $ene>2} {

set en $ene

set nd $n

}

}

puts "$nd -- $en"

puts $En\_fl "[$ns\_ now] $en"

$ns\_ at [expr [$ns\_ now]+1] "Grph"

}

$ns\_ at 0 "Grph"

for {set i 0} {$i<$val(nn)} {incr i} {

$ns\_ initial\_node\_pos $node\_($i) 10

}

proc finish {} {

global ns\_ namtrace nd\_ opt\_ En\_fl

$ns\_ flush-trace

close $namtrace

close $En\_fl

exec awk -f pdf.awk out.tr >PDF\_$opt\_

exec awk -f OH.awk out.tr >OH\_$opt\_

exec awk -f th.awk out.tr >th\_$opt\_

#exec awk -f LT.awk out.tr >LifeTime\_$opt\_

#exec awk -f energy.awk out.tr >en\_grp\_$opt\_

exec nam out.nam &

#exec xgraph en\_grp\_$opt\_ -x "time" -y "energy" -lw 2 &

exec xgraph th\_$opt\_ -x "time s" -y "pkts" &

exec xgraph PDF\_$opt\_ -x "time s" -y "pkts" &

exec xgraph OH\_$opt\_ -x "time s" -y "pkts" &

#exec xgraph LifeTime\_$opt\_ -x "No of Rounds" -y "No of Active Nodes" &

exit

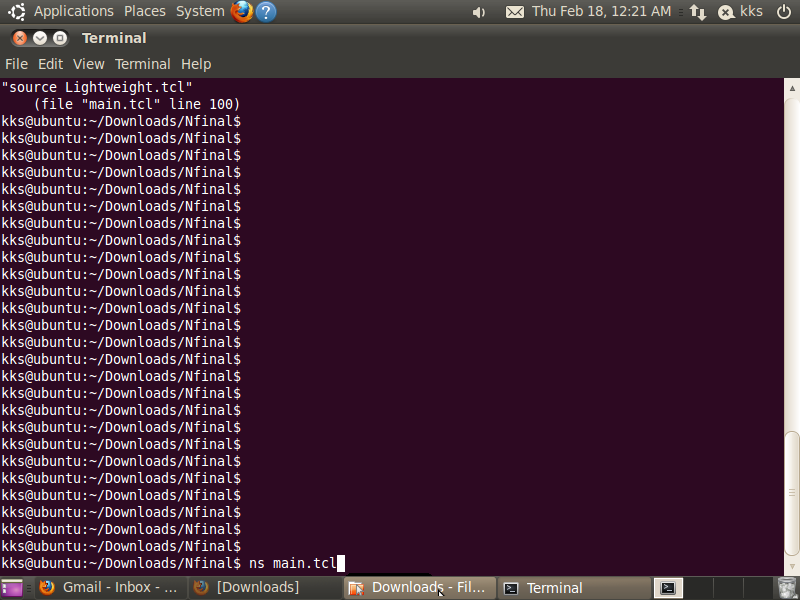
}

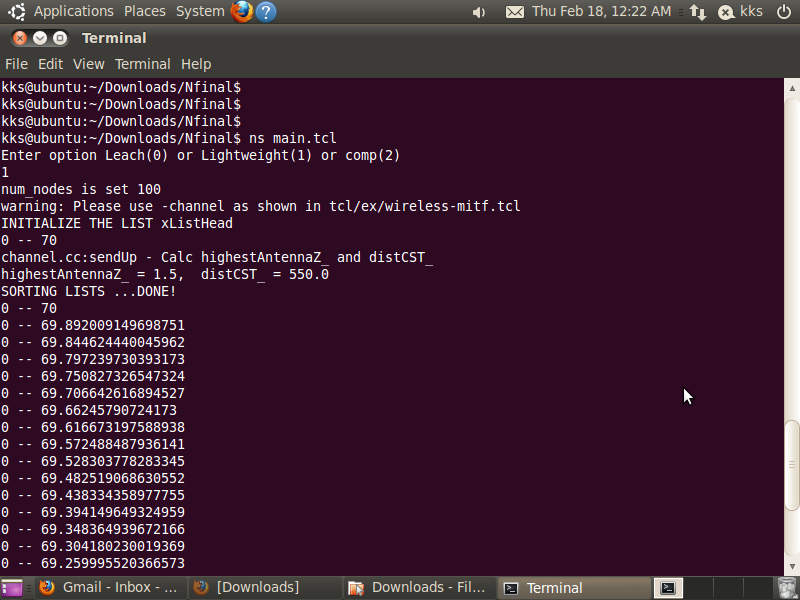
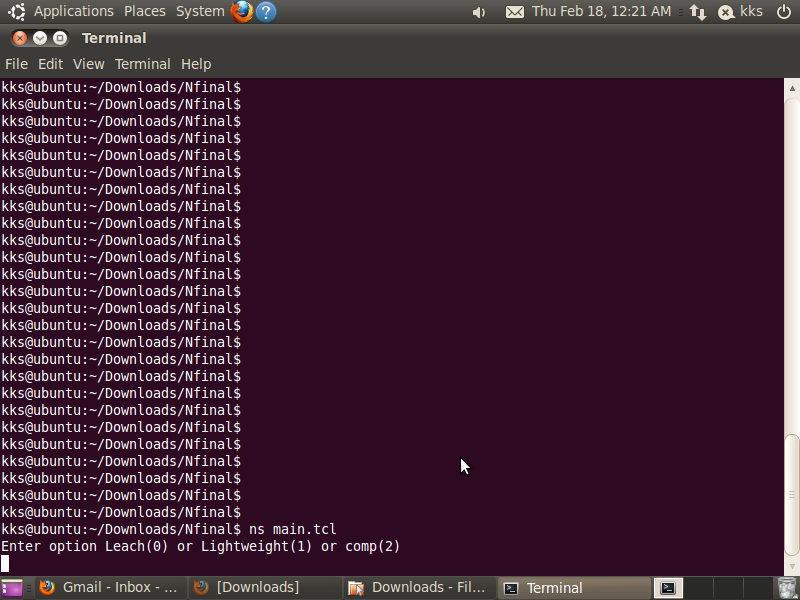
$ns\_ at 40 "finish"

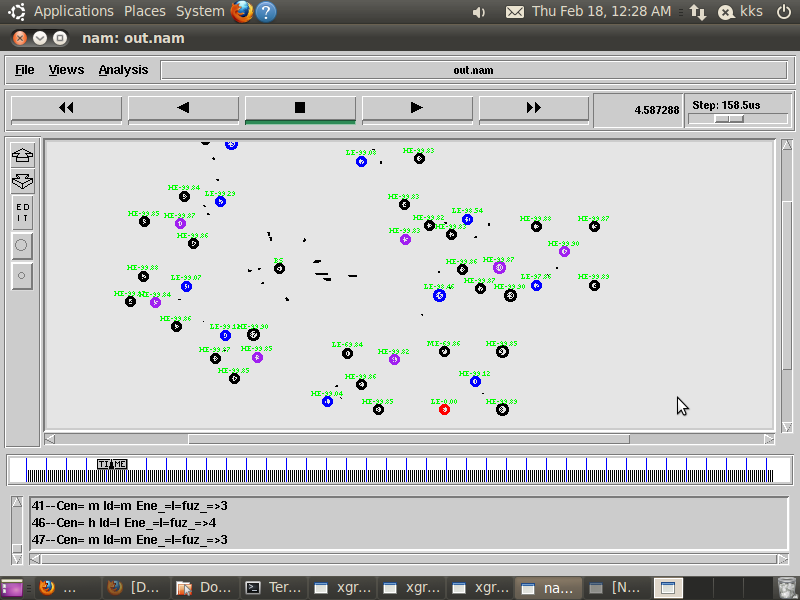
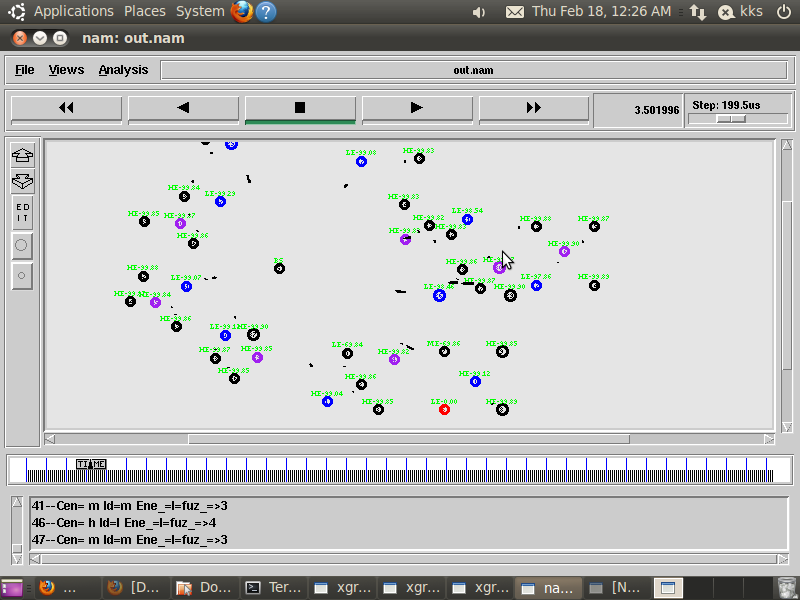
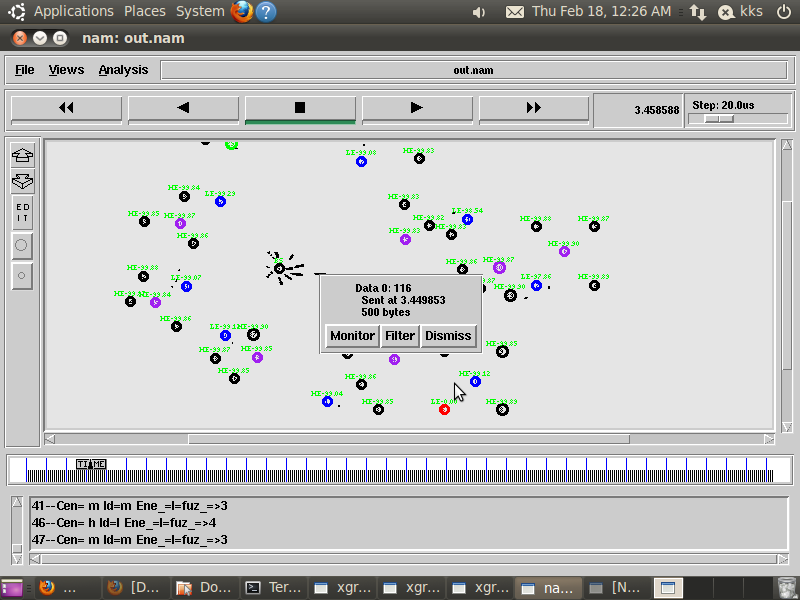
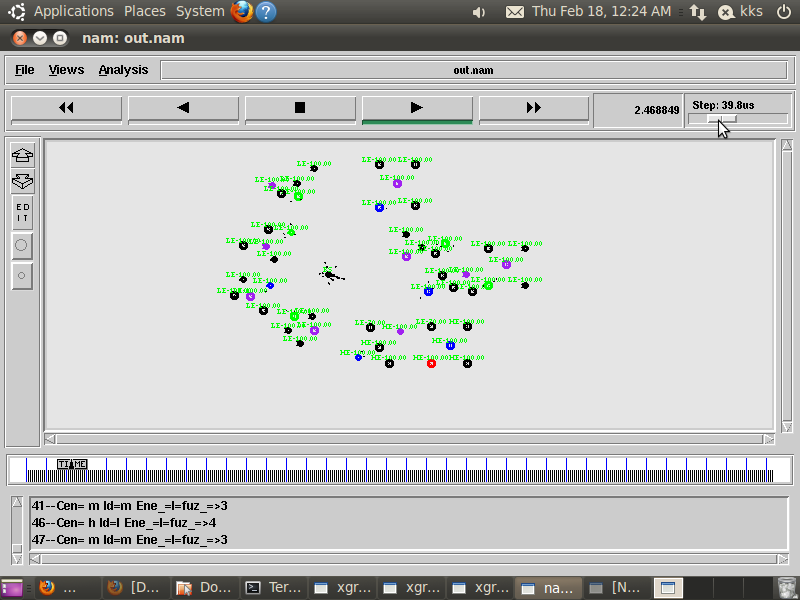
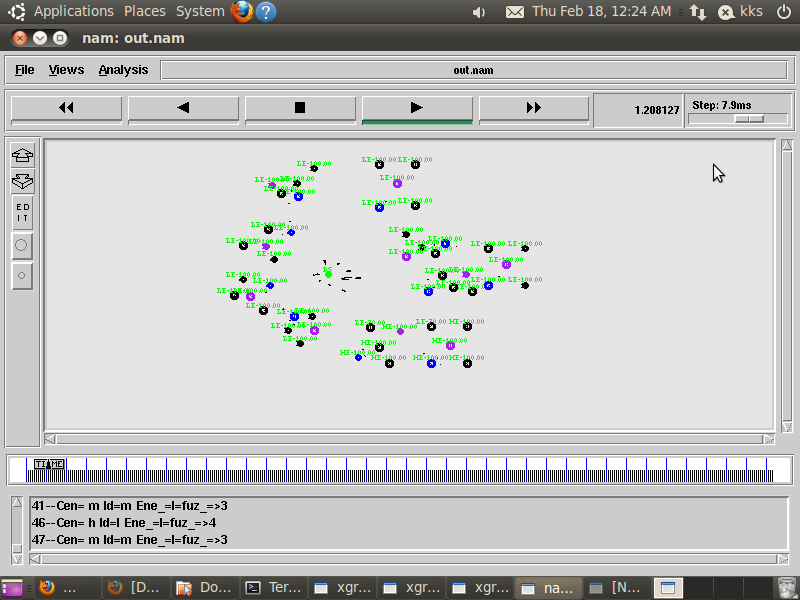
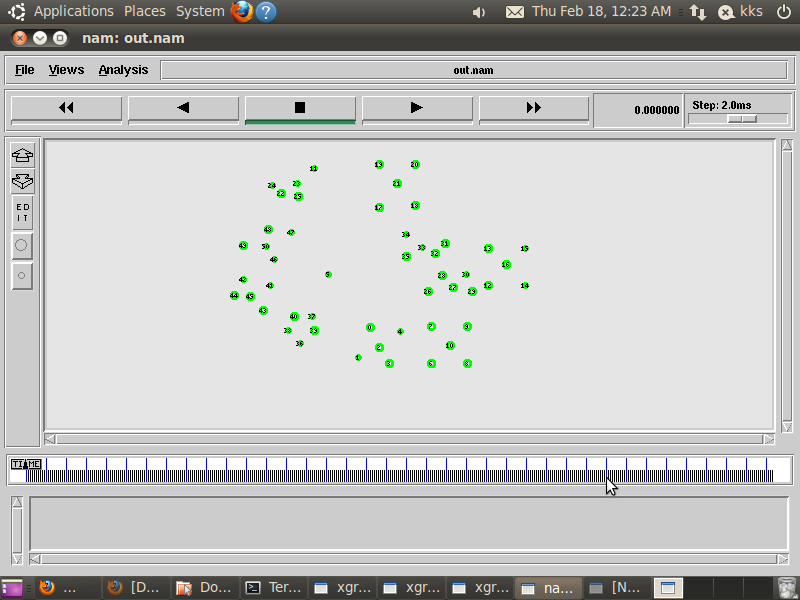
$ns\_ run

**RESULTS:**

**Simulation Result:**

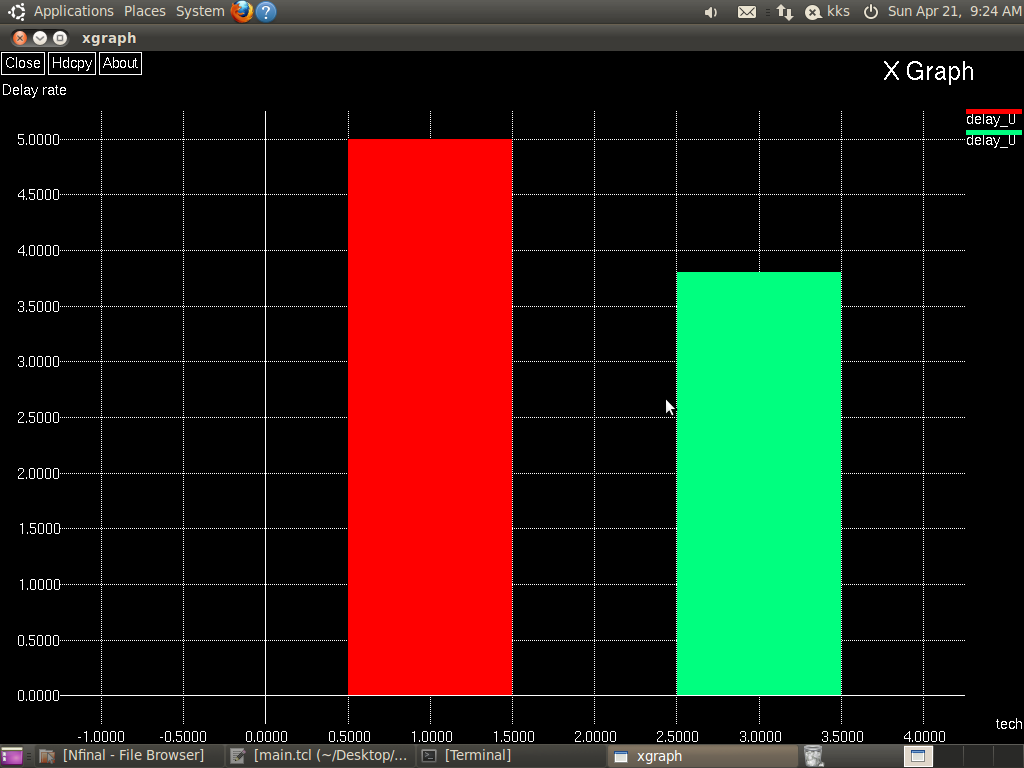
****

****

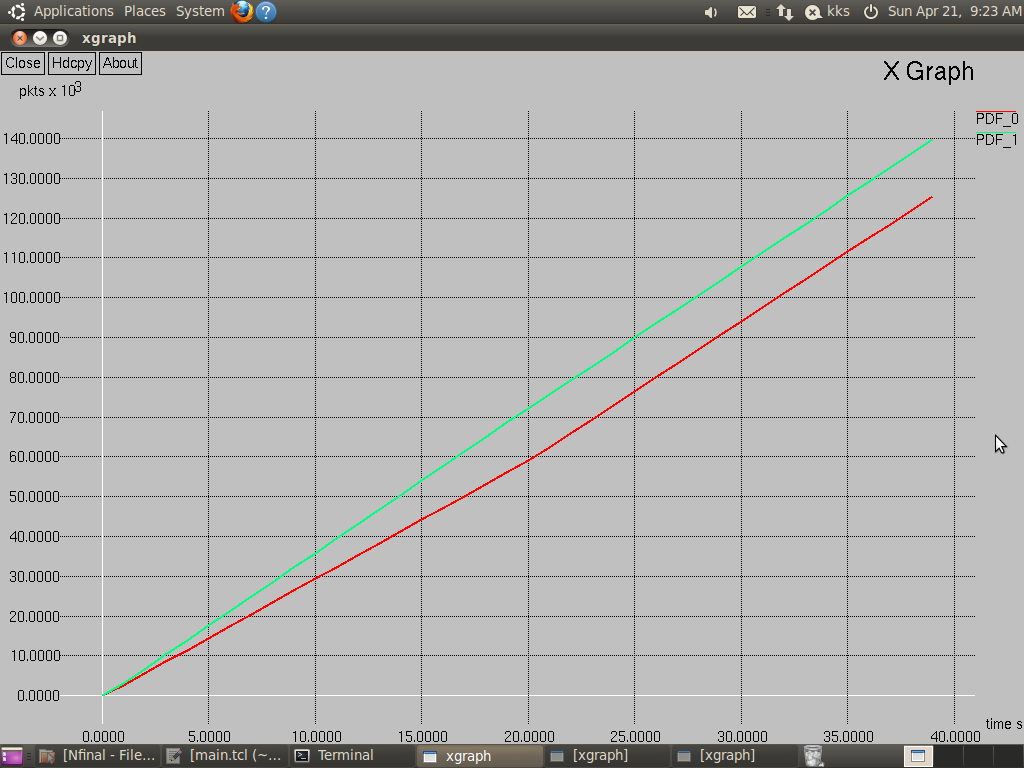
****

**Throughput:**

****

**DELAY:**

**POCKET DELIVERY FACTOR:**

****

**REFERENCES OR BIBLIOGRAPHY:**

[1] V. Tabus, D. Moltchanov, Y. Koucheryavy, I. Tabus, and J. Astola, ``Energy efcient wireless sensor networks using linear-programming optimiza- tion of the communication schedule,'' J. Commun. Netw., vol. 17, no. 2, pp. 184197, Apr. 2015.

[2] M. Sefuba and T. Walingo, ``Energy-efcient medium access control and routing protocol for multihop wireless sensor networks,'' IET Wireless Sensor Syst., vol. 8, no. 3, pp. 99108, Jun. 2018.

[3] S. K. Singh, P. Kumar, and J. P. Singh, ``A survey on successors of LEACH protocol,'' IEEE Access, vol. 5, pp. 42984328, 2017.

[4] H. Kim and S.-W. Han, ``An efficient sensor deployment scheme for large- scale wireless sensor networks,'' IEEE Commun. Lett., vol. 19, no. 1, pp. 98101, Jan. 2015.

[5] K. Eritmen and M. Keskinoz, ``Improving the performance of wireless sensor networks through optimized complex eld network coding,'' IEEE Sensors J., vol. 15, no. 5, pp. 29342946, May 2015.

[6] D. Dong, M. Li, Y. Liu, X.-Y. Li, and X. Liao, ``Topological detection on wormholes in wireless ad hoc and sensor networks,'' IEEE/ACM Trans. Netw., vol. 19, no. 6, pp. 17871796, Dec. 2011.

[7] J. Padmanabhan and V. Manickavasagam, ``Scalable and distributed detection analysis on wormhole links in wireless sensor networks for networked systems,'' IEEE Access, vol. 6, pp. 17531763, 2018.

[8] M. Ditzel and K. Langendoen, ``D3: Data-centric data dissemination in wireless sensor networks,'' in Proc. Eur. Conf. Wireless Technol., Paris, France, 2005, pp. 185188.

[9] P. R and M. Pushpalatha, ``SDN enabled SPIN routing protocol for wireless sensor networks,'' in Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET), Chennai, India, Mar. 2016, pp. 639643.

[10] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, ``Directed diffusion for wireless sensor networking,'' IEEE/ACM Trans. Netw., vol. 11, no. 1, pp. 216, Feb. 2003.

[11] P. Mishra and R. Tripathi, ``Optimum energy path (OEP) protocol for wireless sensor network,'' in Proc. Int. Conf. Circuits, Syst., Commun. Inf. Technol. Appl. (CSCITA), Mumbai, India, Apr. 2014, pp. 116120.

[12] X. Yang, L. Wang, and Z. Zhang, ``Wireless body area networks MAC protocol for energy efciency and extending lifetime,'' IEEE Sensors Lett., vol. 2, no. 1, Mar. 2018, Art. no. 7500404.

[13] M. Elsersy, T. M. Elfouly, and M. H. Ahmed, ``Joint optimal placement, routing, and ow assignment in wireless sensor networks for structural health monitoring,'' IEEE Sensors J., vol. 16, no. 12, pp. 50955106, Jun. 2016.

[14] S. Puvaneshwari and S. Vijayashaarathi, ``Efficient monitoring system for cardiac patients using wireless sensor networks (WSN),'' in Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET), Chennai, India, 2016, pp. 15581561.

[15] H. Lin, L. Wang, and R. Kong, ``Energy efcient clustering protocol for large-scale sensor networks,'' IEEE Sensors J., vol. 15, no. 12, pp. 71507160, Dec. 2015.

[16] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, ``An application-specific protocol architecture for wireless microsensor networks,'' IEEE Trans. Wireless Commun., vol. 1, no. 4, pp. 660670, Oct. 2002.

[17] D. M. Birajdar and S. S. Solapure, ``LEACH: An energy efcient routing protocol using OmnetCC for Wireless Sensor Network,'' in Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT), Coimbatore, India, 2017, pp. 465470.

[18] T. Giannetsos and T. Dimitriou, ``LDAC: A localized and decentralized algorithm for efficiently countering wormholes in mobile wireless networks,'' J. Comput. Syst. Sci., vol. 80, no. 3, pp. 618643, May 2014.