CHAPTER 1

INTRODUCTION

In Wireless Sensor Networks (WSNs), geographically distributed independent sensors used to monitor physical world aspects, such as weather, environment, temperature etc. and to pass these informative data to the main location of the network. This fact allows the sensor nodes to be placed in random in remote places or unreachable places especially in flood affected areas or landscape were human intervention is not possible. Above fact also implies that different networking protocols and algorithms of the sensor networks should have self-organizing capacity. Other than in military fields it can also use in farming sector, security services, telecommunication, home automation etc. In all this applications, the WSN may face various obstacles like packet drop/loss, energy efficiency, congestion etc. This affects the performance of the network. WSN is very economical for checking the distinct features of environment and industries. WSNs give assurance as an enabling technology for different types of applications that includes data collection and event detection. Wireless sensor nodes usually have a built-in processor and the nodes perform computation and send the required information to the station. Advanced research in the field of MEMS Technology, Wireless communications and VLSI design of small chips lead to the development of low cost and less energy consumption sensor nodes. The above described features can be applied in a large scale in variety of applications especially in the health sector, military operations and environment monitoring etc. For example, the doctor can monitor the physiological condition of a patient residing in remote area. Sensor networks are used largely in the weather monitoring purpose and they help us to know the concentration of the pollution and location of the polluted affected areas.

Congestion is the important characteristic feature of WSNs that has grabbed interest of research scholars. Congestion in the network leads to less throughput and also battery of the node discharges very quickly, which is an important resource in WSN. So, congestion is very important for QoS of a network. There are multiple cluster heads are available in WSN. Each of the cluster heads assembles the information from their respective cluster's member nodes and transfers the aggregated information to the base station. Cluster heads are receiving the packets from the member of the cluster. A major challenge in WSNs is to

select appropriate cluster heads. In a densely deployed network, the sink nodes are defined to transfer the packets to respective cluster heads. Generally cluster heads are directly connected with the base station. But here 'Traffic Node(s)' are introduced which will manage the data packets coming from cluster heads. Traffic nodes are also defined to receive packets from the nearest cluster heads. Once the cluster heads receive the data from the nodes they will forward it to traffic node. Then traffic node will forward the received packets to base station. Here the intercluster communication is important when the waiting queue is very high as it has a communication with nearby cluster head that can manage to decrease the congestion.

Buffer size of the cluster plays an important role in network communications. Buffer size is essential to ensure communication effectiveness. If buffer size increases the packets are queued unnecessarily causing delay in communication. It may also decrease throughput. To decrease the waiting queue inter-cluster communication is needed. When one of the cluster heads has waiting queue is full, the sink nodes can communicate with nearby cluster head to decrease the congestion. Buffer size helps to manage the nodes in the network. When the buffer size increases the performance time of the algorithm should also increases because each node may not have enough time to manage the packets. If buffer size of the cluster increases the node density also increases. Node density has a very huge effect on the performance of the network. By varying the buffer size we can observe different values for different QoS. The effects of increasing value of nodes can be observed in graphs of throughput, delay, PDR. These values can determine the effectiveness of the algorithm. In consideration to the time constraint, if node density increases the time should also increases. If time increases for the WSN, it may perform better. As it has effect on the each of the QoS parameters. Here, we have to take in consideration of all the parameters as each of them having effect directly or indirectly on the network and its performance. The positions of the nodes are also matters as they are wireless nodes. If the distance between them is more, it may require more battery power as the battery power is limited constraint for the nodes. Wireless sensor nodes are in a routing algorithm where each of them are defined to send the data packets to the base station of the network. The algorithm is having a best suited solution as it has high value of the throughput and PDR which shows the efficiency of the network. As the throughput of the network increases, delay has increased. Delay has increased when buffer size increased, as each node may not have enough time to perform.

The flooding of the packets needs to be controlled as it has impact on the performance of the network. Path between the nodes and cluster heads are defined to have collective information between them.

1.1 Objectives

- To enhance the throughput, algorithm makes certain arrangements of all the sensor nodes to make a WSN.
- To reach high packet delivery ratio, the routing protocol must avoid packet dropping when a possible packet routing path available.
- Through analyzing the traffic the attacker must not find the destination.
- The source location must not be available with the attacker, if attacker is only able to monitor a certain area of the WSN and compromise a few sensor nodes.
- Once the packet is received at the destination side, it should identify the source location through the packet.
- The recovery of the destination position from the received packet should be very efficient.

1.2 Motivation

- Improving congestion is very challenging Task. For the utilization and the security of network is very important as each node contains important information.
- The network policies are defined to describe the value of the information as the algorithm is described to show the QoS.
- As life of sensor nodes are depends on their energy, various protocols are defined to have improvement in network lifetime.
- Along with improvement of the network lifetime, researchers may forget to improve other QoS as they are also important for the network performance.
- This point has caught eyes for improvement of the other QoS.
- To make any changes in existing configuration we have to change in existing path of the packet.
- It’s more cost effective as we have introduced the traffic node and a defined path.

1.3 Proposed System

- In this proposed system, a hierarchical structure is defined for the wireless sensor nodes to create a WSN.
- Each of the sink nodes sends the data packets to respective cluster-heads.
- Cluster-heads sends data to the nearest traffic node(s). Traffic node(s) collects all the data from the respective cluster-heads.
- Then traffic nodes send the data to the base station (BS).
- It has observed the different buffer size of the cluster for effectiveness of the algorithm.

1.4 Scope

The scope of the project is to implement a secure and efficient routing protocol that provides different QoS for WSN. Each node contains some energy to contribute the network as required. The nodes are defined to have a fixed path for the packets to reach base station.

1.5 Report Organization

The chapter 1 and 2 covers the Introduction and Literature Survey. Chapter 3 and 4 cover the System Requirements and System Design and chapter 5 deals with the Implementation. Chapter 6 covers testing and results. Chapter 7 contains comparison with EAMMH. Finally, the project is concluded with details about conclusion and future work in chapter 8.

1.6 Summary

This Chapter deals with the Introduction to WSN and congestion. It also discussed Motivation, Proposed System and Objective of the Project. It also explored all the previous works in the identified domain and shows a path

CHAPTER 2

LITERATURE SURVEY

2.1 Wireless Sensor Network

WSNs contain various sensor nodes which were placed purposely or arbitrarily over a topographical field and arranged over remote connections to design a WSN. The sensor nodes are placed in such an encompassing situation that the sensor nodes can detect the encompassing condition in the encompassing environment that was prepared to uncover the qualities of the marvels happening at the place. The idea about data reconciliation or dispersal can be fulfilled if each of the sensor nodes in WSN is fit for the communication with other sensor nodes and the base station (BS) as well. WSNs are utilized essentially as a part of army system, public affairs and for industrial apps. WSNs applications in army sector incorporate war zone observation, interruption recognition, target field and imaging. In any case, WSNs are presently utilized as part of numerous nonmilitary application regions, including environment and living space observation, health apps, home automation systems, traffic management system. WSNs commonly contain hundreds or thousands of sensor nodes which takes into consideration detecting over larger topographical areas with greater accuracy. Each node has three fundamental components as shown in Figure 2.1.

i. Sensing unit

ii. Processing unit

iii. Transmission unit

Sensor nodes detect the informative data from surroundings, prepare it with required process and send it to the BS. These sensor nodes can setup a path for the data to the BS or to the other nodes with the end goal that the data in long run reaches the base station. For many applications, sensor nodes are having limitation in energy supply and communication bandwidth. These nodes are controlled by indispensible batteries and consequently network lifetime relies upon the battery consumption [8]. Inventive procedures are produced to

proficiently utilize the limited energy and data transfer capacity resource. These systems work by cautious design and administration at all layers of the networking protocol. For instance, at the network layer, it is exceedingly attractive to discover techniques for energy efficient route discovery and relaying of data from the sensor nodes to the base station (BS). So, the lifetime of the network is increased.

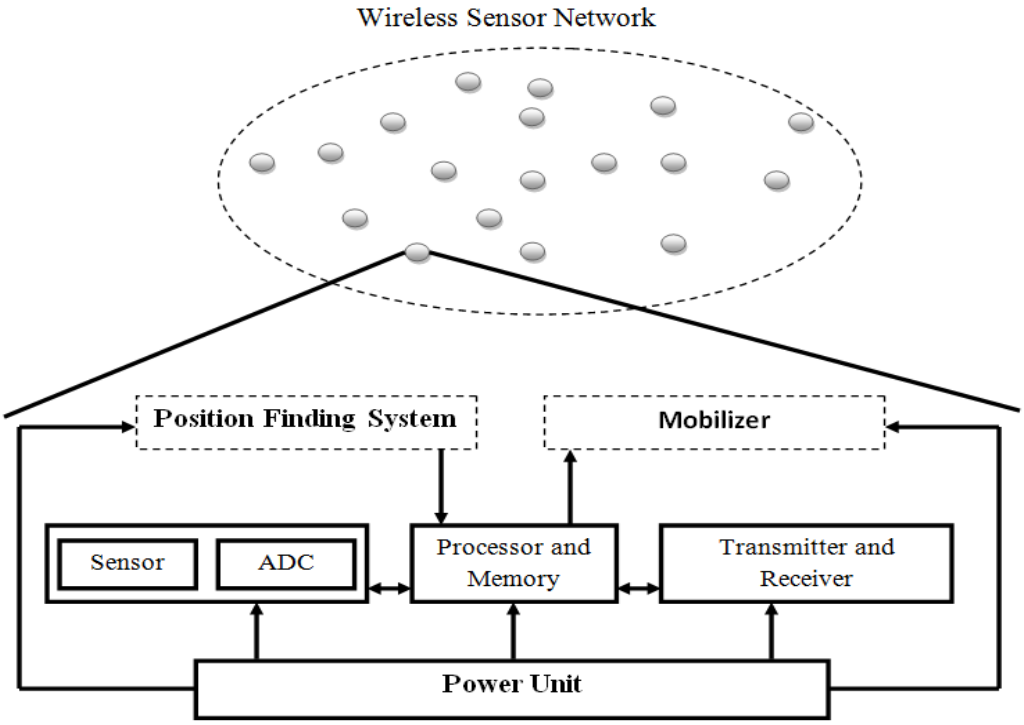

Figure 2.1 WSN and components of sensor node

2.2 Congestion

Congestion is a significant measure of a WSN and it influences the Quality of Service (QOS) parameter of a network. The term Congestion is coined when the node or link in the network receives more data than its processing capacity [2] [3]. Network Congestion happens when traffic load exceed the available capacity at any point in the network. Congestion causes packet loss, excessive energy consumption, delay. Therefore, in order to improve network lifetime and improve fairness and provide better a quality of service, there is a need for congestion estimation and control is important to be considered.

Congestion in the network can be decreased by two methods either by detection or avoidance of congestion in the network.

There are two types of congestion. 1. Node Level Congestion 2. Link Level Congestion. At the node level, congestion must be identified by checking buffer capacity or by identifying busyness of the link in the network. The congestion at node level is the frequent event occurring in typical sensor networks. Node level congestion occurs when buffer overflow occurs in the node and this in turn leads to huge loss of data packets and queuing delay of the network is increased due to this buffer overflow. In Link-Level Congestion, some of the active sensor nodes may face relentless packet collisions because other sensor nodes may also sent the data packets in the same sensitive area at same time. Some of the packets left the buffer may not be able to reach to the consecutive hop as it may cause collision. It may decreases link application and final throughput, but it increases energy consumption and packet delay.

2.3 Routing Algorithms

Routing protocols used in WSNs indicates data dissemination, limited energy efficiency and data transfer capacity in order to facilitate proficient working of the network, in this way expanding the lifetime of the network.

2.3.1 Congestion Control Protocols [2]

Congestion control can be related to control traffic for communications network, in order to keep away from congestive collapse by taking attempt to avoid excessive capacity of refining or link capacities of the mediator nodes or networks to take resource decreasing steps, as decreasing the rate of forwarding packets[2]. It must not be mistaken with flow control that keeps away the sender and receiver from each other. There are two formal ways to control congestion.

1. Resource Management
2. Traffic control

2.3.1.1 Congestion Avoidance and Detection (CODA) [2]

For WSNs, CODA is designed in energy efficient manner for congestion control. By observing sensor nodes' buffer size and the load of the wireless link, CODA can detect the

congestion. A wireless node indicates its nearby sensors to reduce the sending rate if the value of the above two characteristics increases more than predefined fixed value [2]. A wireless sensor node isolates the medium of transmission to fixed periods before sending a packet. It manages to alter a control bit to notify the BS of the congestion if it saw the channel occupied with so much of work than pre-defined times.

2.3.1.2 Congestion Control and Fairness (CCF) [2]

CCF needs to be recognized as congestion lay upon packet service time for MAC layer to manage the congestion based upon simple fairness including hop by hop system [2]. To reduce the possible service rate or identify the congestion in every mediator node, CCF can utilize packet service time. Although the congestion is faced, they illuminate the downstream nodes to decrease their data transmission rate.

2.3.1.3 Adaptive Rate Control (ARC) [2]

ARC oversees infusion of data packets inside the traffic stream and path over traffic [2]. Every node can predict the count of upstream nodes or the transmission capacity is separated equivalently amid path over and locally created traffic, along the choice disposed to the previous one. The final appearance of data transfer capacity assigned to each of the mote is almost fair. Decrease in sending rate of path over traffic has a back stress impact upon upstream nodes, which can diminish their sending rates.

2.3.1.4 SenTCP [2]

SenTCP has two exceptional elements which is an open loop hop by hop congestion control protocol. The elements are (1) They together uses normal local packet service time as well as normal local packet inter-arrival time after estimating current local congestion degree in every mediator sensor node. To figure out congestion degree, packet arrival time and service time can be utilized. It also effectively benefits to make difference between the reasons of packet loss incidents in wireless sensor networks, because arrival time or service time may develop into small (or large). (2) It utilizes hop-by-hop congestion control mechanism. In SenTCP, every mediator sensor node will issue feedback signal backward and hop-by-hop. At the transport layer, sending rate is conformed as the feedback signal which conveys nearby degree of congestion and buffer occupancy ratio is utilized for the

nearby sensor nodes. The utilization of hop by hop feedback control can evacuate congestion rapidly and lessen packet dropping [2]. It can preserves lifetime of the network. SenTCP acknowledges high throughput or great energy efficiency since it clearly diminishes packet dropping; however, SenTCP adapts to just congestion or ensures no unwavering quality.

2.3.1.5 Fairness Aware Congestion Control (FACC) [2]

FACC controls the congestion that can also manages to accomplish with fair transmission capacity assignment for every flow of data. At sink node, FACC can identify the congestion in view of packet drop rate. Nodes are mainly classified in two categories nearby sink node, nearby source node in view of their area in WSNs. A "Warning Message" (WM) has been transmitted by nearby sink node to nearby source node, if a packet has dropped [2]. Subsequent to accepting Warning Message, source node receives a "Control Message" (CM) from nearby source nodes. In the view of the present traffic on the channel and the present sending rate, source nodes confirm their sending rate. Flow rate could balance in view of recently figured sending rate, after getting control message.

2.3.1.6 Fusion [2]

For congestion detection, hop by hop flow control methodology is utilized in Fusion. Congestion can be identified over occupancy by queue and communication media inspecting procedure at every middle node [2]. Whenever the node catches congestion, "Congestion Notification" (CN) bit must be placed in header of every outgoing packet. Once Congestion Notification bit is placed, nearby node can catch it or quit sending packet to congested node.

2.3.1.7 Priority Based Congestion Control Protocol (PCCP) [2]

PCCP reflects significance of every sensor node which has node priority index for congestion control. Nodes are given a priority in view of the tasks they show and its area. The nodes which are nearby the sink node will be given a higher priority [2]. The proportion of packet arrival rate and sending rate shows the congestion detection. The lower sending rate infers that congestion happened. The information related to congestion is included in data packet header alongside the need list. The congestion at nodes relies upon nodes alter

their sending rate. While accomplishing the weighted fairness sending for single or multipath routing, PCCP tries to diminish packet drop in congestion state.

2.3.1.8 Trickle [2]

Spreading or keeping up code upgrades is required for wireless sensor networks as “Trickle” is used for this kind of issues. The essential primitive of Trickle is occasionally, if it has not heard a couple of different nodes send the similar thing, occasionally a node sends code metadata [2]. This permit “Trickle” to scale to thousands of overlay varieties in network density, rapidly propagate overhauls, disperse transmission stack uniformly, be vigorous to transient disengagements, control organize repopulations, and force conservation overhead on the request of a couple of packets per hour per node. Trickle will send all the information to nearby broadcast address. Generally two conceivable answers to a Trickle broadcast are possible: both every node that receives text needs to be updated and a receiver distinguishes the requirement for an overhaul. Recognition can be after effect, either an outdated node receives someone has new code and an updated node receives someone has old code. Here, each node communicates somehow, so they either sends or receives. The requirement for refurbishment will be disclosed. E.g., if node A displays has code φ, but node B has code φ+1, then node B finds that node A requires an update. Likewise, if node B shows that it has φ+1, node A finds that it requires an update. If node B displays updates, without displaying their requirement, all of its nearby nodes can approach them [2]. Few of them might not even have heard node A's transmission. Trickle utilizes "polite gossip" to swap code information with nearest network neighbors. Some Time intervals, and at an irregular point in several interval of time, it can considers to display its information of the code. If Trickle has already been received several other nodes babble the similar information in this interim, it pleasantly remains calm: rehashing what someone else has said is impolite.

2.3.1.9 Siphon [2]

Siphon goes for controlling congestion as well as concern about channeling impact. The packet loss occurred when occasions were conceived under distinct work stack runs fast towards at least a sink node can expands activity at the channeling impact. The movement has been stacked off completely from stacked sensor nodes as the “Virtual Sinks” (VS)

haphazardly appropriated over the sensor network. Initially VS disclosure has ended. VS disclosure was initiated by the physical sink. Node initiated congestion detection rely upon a significant time span, buffer occupancy and channel situation as in CODA [2]. Traffic route is diverted from overloaded physical sink to virtual sinks after congestion detection. By managing redirection bit in network layer header it may ends up.

2.2.1.10 Prioritized Heterogeneous Traffic-oriented Congestion Control Protocol (PHTCCP) [2]

For taking care of different data with various preferences within a single node inspires, PHTCCP is utilized. PHTCCP works on MAC layer to control the congestion [2]. This protocol has concentrated on proficient system with the goal that congestion can be managed by guaranteeing modification sending rates for various kind of data that produced by the sensors have different priorities. It can expect that sink node allocates singular preference for every kind of sensed data and every node has various equivalent estimated sized priority queues for n kinds of sensed data. Miscellaneous applications can reflect the number of queues in a node. In congestion detection mechanism, congestion level at every node performed by packet service proportion.

$r(i) = R_s^i / R_{sch}^i$, R_s^i is the proportion of avg. packet service rate and R_{sch}^i is the packet scheduling rate in each sensor node.

2.3.2 Congestion Avoidance Protocols [2]

Congestion avoidance techniques monitor network traffic loads in an active effort to expect and ignore congestion at regular system bottlenecks. Congestion avoidance is accomplished through packet dropping. Amid the all, the more generally utilized congestion avoidance mechanisms are Random Early Detection (RED), which is ideal for high-speed transmission networks.

2.3.2.1 Random Early Detection [2]

“Random Early Detection” (RED) is utilized on the network hardware's port queue buffer. On network hardware ports with minimum one queue buffer, Weighted RED (WRED) could be utilized if more than one queue buffer is if available on the network hardware ports. RED indirectly indicates to receiver and sender by removing some packets,

e.g. when the avg. queue buffer lengths are higher than e.g. 50% filled and removes continuously more or cubical more packets, up to e.g. 100%. The avg. queue buffer lengths are figured out over 1 second on end.

2.3.2.2 TCP/IP congestion avoidance [2]

The TCP congestion avoidance algorithm is the fundamental basis for the congestion control in the Internet. Problems happen when various simultaneous TCP flows are encountering port queue buffer tail-drops. Then TCP's automated congestion avoidance is insufficient [2]. All flows that face port queue buffer tail-drop will initiate a TCP retrain at a similar minute – this is called TCP global synchronization.

2.3.2.3 TCP Tahoe and Reno [2]

TCP utilizes a multi aspect congestion control procedure to escape from congestion collapse. TCP keeps a congestion window, restricting the aggregate number of unidentified packets that might be in travel end-to-end in each association. This is to some degree for all intents and purposes equivalent to TCP's sliding window utilized for flow control. After connection is established and after an interval, TCP utilizes a system methodology called slow initiation to higher the congestion window. It initiates along window of two circumstances the “Maximum Segment Size” (MSS). In spite of the fact that starting rate is low, the rate of increment is extremely brisk: for each packet recognized, the congestion window increments by 1 MSS so that the congestion window viably copies for each “Round-Trip Time” (RTT) [2]. At a point, when the congestion window surpasses as far as possible, ssthresh limit, algorithm takes entry into a new stage, call congestion avoidance. In few executions, the original ssthresh is huge, thus the primary slow start normally ends after a loss. However, ssthresh is updated at the end of every slow start, and will frequently affect consecutive slow starts.

As far as original ACKs are accepted, the congestion window has supplemental increased by 1 MSS each RTT [2]. The possibility of copy ACKs being received is very high when a packet is lost. The conduct of Tahoe and Reno contrast in how they detect and reply to packet loss:

- Tahoe: Tahoe executions distinguish congestion by taking context of clock for taking a relevant ACK. Tahoe sets ease back begin limit to half of the present congestion window, decreases congestion window to 1 MSS, and restarts to slow-start state.
- Reno: Three duplicate ACKs are accepted; Reno will bisect the congestion window, set up slow beginning limits balanced with the new congestion window, send it quick again, and enter a phase called Fast Recovery. If ACK gets a time out, it will slow beginning with Tahoe.

There are two contradictions between Tahoe and Reno.

1. Tahoe just utilizes a timeout for catching congestion; Reno utilizes timeout and Fast retransmission of packets.
2. Tahoe set up the value of congestion window to 1 after packet drop or loss, Reno set up to half of the recent congestion window.

2.3.2.4 Robust Random Early Detection (RRED) [2]

The “Robust Random Early Detection” (RRED) algorithm was projected to improve the TCP throughput against “Denial-of-Service” (DoS) attacks, especially for “Low-rate Denial-of-Service” (LDoS) attacks [2]. Experiments have affirmed that the present RED-like algorithms are outstandingly unprotected under LDoS attacks as it is a consequence of the wavering TCP queue size carried on by the attacks.

2.3.2.5 Weighted Random Early Detection (WRED) [2]

“Weighted Random Early Detection” (WRED) is a queuing system for a network scheduler suited for congestion avoidance. WRED is the expansion to RED where one of the queues may have few distinct queue thresholds. For each queue, threshold is related to a specific traffic class. In an instance, value of the limit for lower priority packet has been brought down by a queue. A queue development will bring about the lower priority packets to be dropped, thus securing the higher priority packets in the same queue [2]. Along the queue, QoS priority is made feasible for critical packets from a set of packets utilizing the

same buffer. Standard movement will be skipped rather than higher prioritized traffic if it is more probable.

Calculation of average queue size: $avg = o*(1-2^{-n}) + c*(2^{-n})$

Where n is the user defined exponential weight factor, o is the old avg., c is the current queue length [2]. The past avg. will be more critical for huge estimations of n.

2.3.2.6 Adaptive Random Early Detection (ARED) [2]

The Adaptive RED or Active RED (ARED) algorithm induces whether to make RED much forceful in light of the perception of the average queue length. If the average queue length wavers around min threshold then early detection is excessively aggressive. Then again if the average queue length wavers around maximum threshold then early detection is by and large excessively moderate [2]. The algorithm changes the possibility as indicated by forceful sensing and disposing of traffic.

2.3.2.7 Low Energy Adaptive Clustering Hierarchy (LEACH) [2]

"Low Energy Adaptive Clustering Hierarchy" (LEACH) is a "Time Division Multiple Access" (TDMA) based "Media Access Control" (MAC) layer protocol that is organized with a simple routing protocol and clustering in WSNs. LEACH protocol has an objective to bring down the battery power utilization required to make and keep up clusters for sake of improvement in life time of a WSN. As LEACH is a hierarchical protocol, that majority of the nodes send cluster heads and the cluster heads makes all together and shrinks the data and places it to the BS. Every node utilizes an algorithm at each of the round to figure out if it will end up being a cluster head. LEACH expect that each of the node has a radio sufficiently capable to precisely arrive the BS or the closest cluster head, however that utilizing this radio at high power all the time would waste energy.

2.3.2.8 Energy Aware Multipath Multihop Hierarchical(EAMMH)Routing Protocol [1]

EAMMH protocol arranges the motes into clusters and forms a multihop intra-cluster network [1]. It builds up various ways from every sensor node to the cluster head and gives an energy aware heuristic function to choose the optimal path. WSN have unique

attributes, for example denser level of node deployment, higher unreliability of sensor nodes and severe energy, computation and storage constraints which exhibit many challenges in the advancements and applications of WSNs [1]. WSN commonly consists hundreds and thousands of sensor nodes that takes into account for detecting over bigger topographical areas with high precision. Typically the sensor nodes are placed arbitrarily over topographical area and these nodes must communicate with each other to frame a network.

2.3.2.8.1 Pseudo code For EAMMH

1. Initialize number of nodes 'n'
2. Setting up initial node positions arbitrarily //make sure the 2 nodes aren't in exactly the same position
3. Display the position 'p'
4. CREATE neighbor discovery module
5. READ 'n_i' and display
6. START cluster head selection
7. Display the head
8. THEN
9. CREATE the cluster and the routing table
10. THEN
11. Assign time slot using DRAND
12. IF communication to Base station and Data tracing or module display
13. goto 7
14. END IF
15. PRINT result

Where, n - number of nodes
p - position of the node
n_i - neighbours

2.4 QoS Policies

"Quality of Service" (QoS) is the total performance of a system or computer network, especially the performance seen by the users of the network. To quantitatively measure QoS, a few parameters of the network performance are often viewed like error rates, throughput, availability, transmission delay, bit rate, jitter, etc. In computer networking field and other packet switched networks, QoS alludes to traffic prioritization and asset reservation control mechanisms instead of achieved service quality [3]. QoS is the scope to give distinctive need to various users, applications, data flows to ensure a specific level of execution to a data flow.

2.4.1.1Throughput [3]

Throughput is determined as number of packets transmitted per second [3]. Below mentioned equation shows the equation of Throughput.

$$TP = \text{™} \sum PacketSize/ (PacketArrival – PacketStart)$$

2.4.1.2 Delay [3]

Delay is defined as the duration taken by the packets to travel from source to destination path [3]. Average delay is characterized as the ratio of (packet arrival – packet sent) to the number of packets.

Average delay= (Packet Arrival- Packet Start)/n

2.4.1.3 Packet Delivery Ratio [3]

Packet Delivery Ratio is the ratio of the packet sent and packet arrival.

PDR = Packet Sent/ Packet Arrival

2.4.1.4 Jitter [3]

Delay variation is the overall variation caused by the packets travelling in the communication network paths.

2.4.1.5 Energy [3]

Energy = Energy during packet transmission + Energy during receiving the packet.

2.5 Simulators

2.5.1 Network Simulator-2 (NS-2)

- Network Simulator -2 formally known as NS-2 is very useful for the representation of algorithms.
- NS2 provides the simulation of routing protocols of both wired and wireless networks.
- C++ language is used to develop NS2. The interpreter tool used here is Object Tool Command Language. The main function of the interpreter is it converts the input code written into "Tool Command Language" (TCL).
- It supports Tcl which has major elements developed in classes, like object oriented manner.
- It is an open source freely available tool. The results of the tcl files are recorded by trace files.
- From the trace files we can the fetch required results like throughput, delay, PDR etc.

2.5.2 MATLAB

- Matrix Laboratory (MATLAB) is a computing tool for scientific applications.
- A programming language is developed by MathWorks, MATLAB performs functions such as development of UI (User Interface), to plot the data and the functions and performs manipulation of matrix. It is used for interfacing with distinct programming languages such as C#, Python, Java etc.
- MATLAB scripting language is used for developing the matlab application.

2.5.3 NetSim

- NetSim is a simulation and emulation tool and it is used for designing the network and for R & D purpose. Different technology such as Cognitive Radio, WSNs, WLAN, Wi Max, TCP, IP, etc. are covered in NetSim.
- NetSim is mainly used for validating the network design.
- For development of custom protocol.

- Defense related application
- Communication networks in railway transportation.
- Emulating the wireless link or satellite link

2.5.4 OPNET

- OPNET simulator belongs to Riverbed Technologies.
- OPNET Simulator is used to analyze the behavior and performance of the network through simulation.
- The simulator handles all different types of networks and technology. For example Voice over Internet Protocol (VoIP), Transmission Control Protocol (TCP), Open Shortest Path First (OSPF) Version 3, Multiprotocol Label Switching (MPLS), Internet Protocol (IP) version 6 etc. The simulator studies the networks and effect of the designs of the above mentioned technology.
- The simulator tests the different technology designs before it is given for production to increase the research and development productivity. The OPNET simulator helps to develop new protocols and technologies.

2.6 Summary

This chapter contains the Literature Survey of Proposed Work and the existing algorithms associated with the congestion. Further, the introduction about the congestion going to use and the importance of all the techniques are briefly discussed

CHAPTER 3

SYSTEM REQUIREMENTS

It is a Precise Description of Entire System Behavior. It describes complete specification of the behavior of a system under consideration. It provides crucial and sufficient information needed for Project Development. It Contains Functional and Non Functional Requirements. Functional Requirements will list the functions of each individual module. Non Functional Requirements will list the Design Constraints.

3.1 Functional Requirements

Functional requirements are mandatory specification for this project. It will give the clear path to complete the project. Functional Requirements are

- Sensor Nodes configurations
- Traffic Node(s) , Sink Node, Base Station identification
- Sensors associated with specific memory and battery
- Portable memory with base station

3.2 Non Functional Requirements

- Knowledge on Wireless Sensor Networks and Routing Protocols associated with Congestion
- Knowledge on writing and running a Tool Command Language (tcl) program on Network Simulator 2 (NS2) and XGraph.

3.3 Software Requirements

There are the required software required software resources for implementing the system specified. As the solution has to be economical the focus was on the open source tools and platforms. The following are the software requirements for the system

- NS-2.35 Package
- XGraph
- Linux with latest C++ and OTcl compilers

3.4 Hardware Requirements

The hardware requirements for implementing the system are as shown below

- 5GB Minimum Secondary Storage
- Minimum 2GB RAM
- Intel Core Processor Computer System

3.5 Summary

Thus this chapter covers the requirement analysis for the implementation of the project. Its list down the complete Requirements needed during the development and testing phase.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

Data centric network architectures are not appropriate for big scale sensor networks. Covering a huge area without decreasing performance is impractical with data centric architecture. In addition, in data centric architectures, the reporting idleness increases with the extent of the network. The data centric approach additionally causes critical power wasteful aspects as the network develops. The network adaptability issue is tended to in the hierarchical routing. The hierarchical routing has a main goal to effectively keep up network power utilization even in big scale networks. As such, hierarchical routing permits the network to scale in various sensor nodes. Most hierarchical architectures comprise of sensor nodes assembled into cluster heads. Cluster heads assemble intracluster communication with different nodes inside the same cluster, however they likewise construct between intercluster communications with other cluster heads. Cluster heads aggregate all the data obtained from the individual sink nodes and then transfer the same data to the nearest traffic nodes in multihop approach. Then, traffic nodes send the received data to the base station. As this architecture is having multiple paths for sink nodes and three hierarchical levels. Each hierarchical level has the hops where the data has got collected. Figure 3.1 is showing the flow diagram of the proposed algorithm.

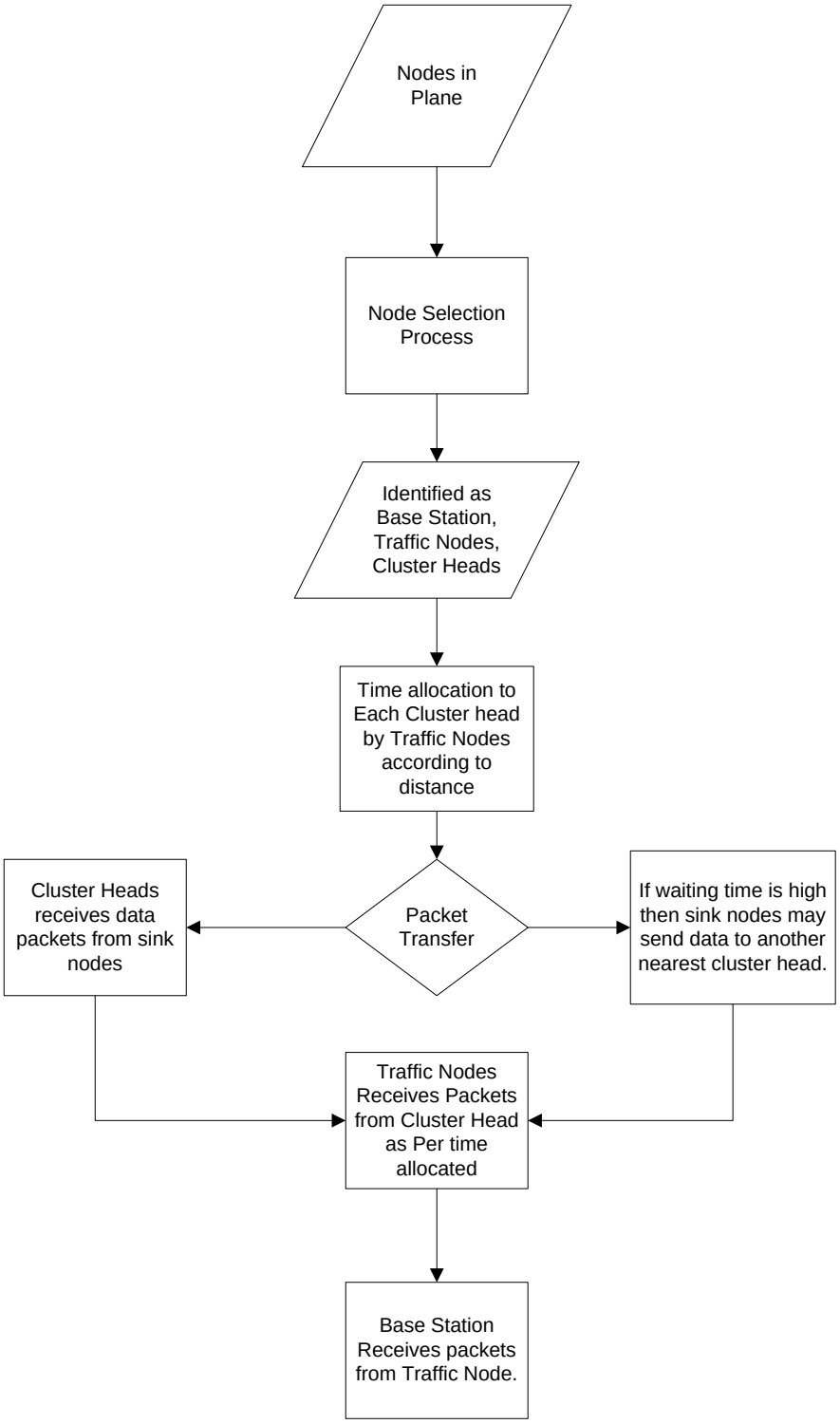

Figure 3.1 Flow Diagram

4.2 Diagrams

4.2.1 Use Case Diagrams

- The UML provides the use case diagram notation to describe the name of the use case actors and relationship between them.
- Actors are Cluster Heads, Sink Nodes, Traffic Nodes, Base Station
- Here, System has the different tasks.
- Links shows actors relation with the task.

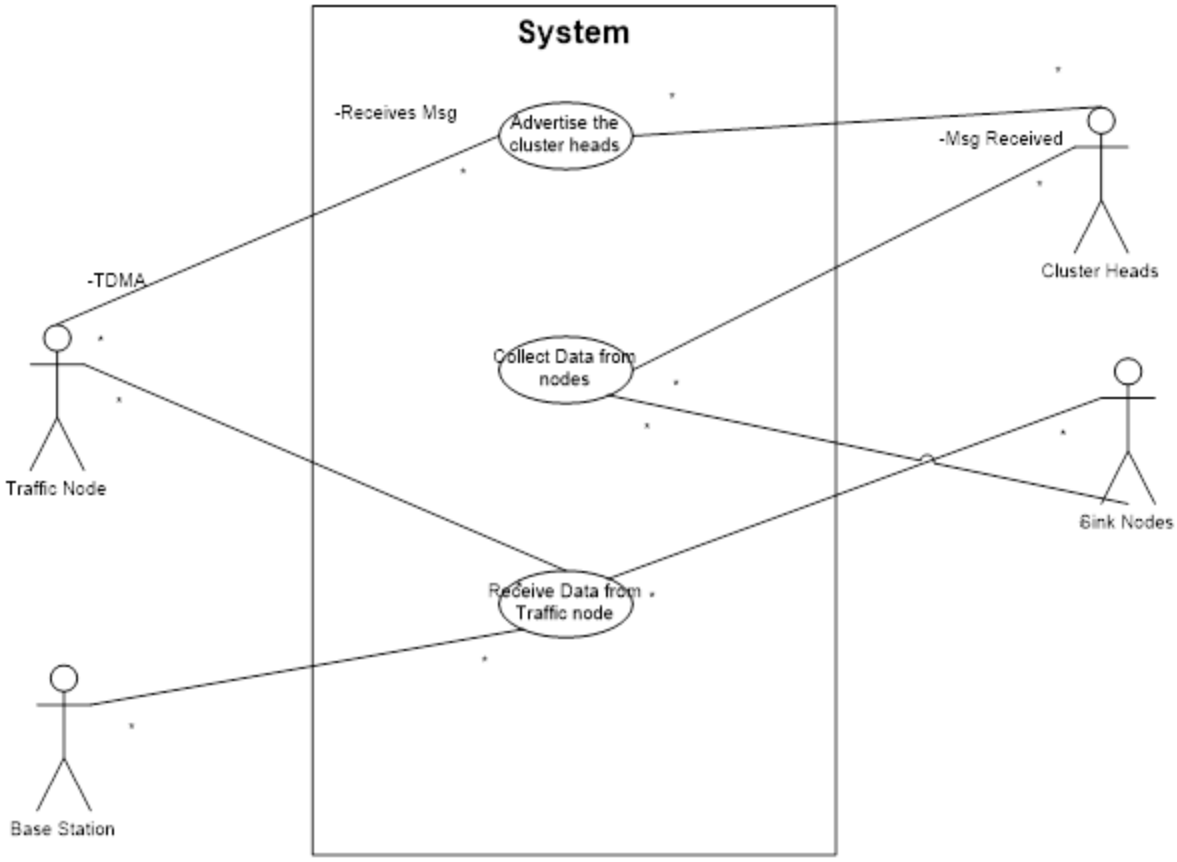

Figure 3.2 Use Case Diagram

4.2.2 Class Diagram

- The UML include the class diagram, to illustrate and their association. They are used for static object modelling.
- Here three different classes are assigned to show the algorithm.
- Each class have different actors, and their functions.
- In each class first row states name, second row states its actors and last row states the functions.
- Links have shown their association with the classes.

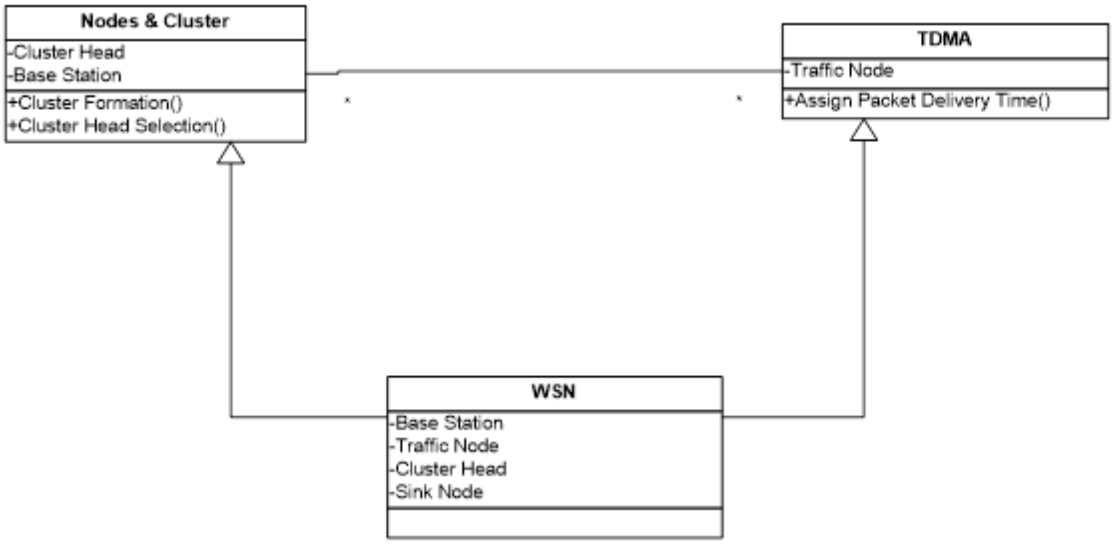

Figure 3.3 Class Diagram

4.2.3 Sequence Diagram

- A Sequence diagram illustrates in a kind of format in which each object interact via messages. It is generalization between two or more specification diagram.
- The Sequence diagram shows the series of events happen between the objects.
- The interaction between them is described in terms of sequence of events.

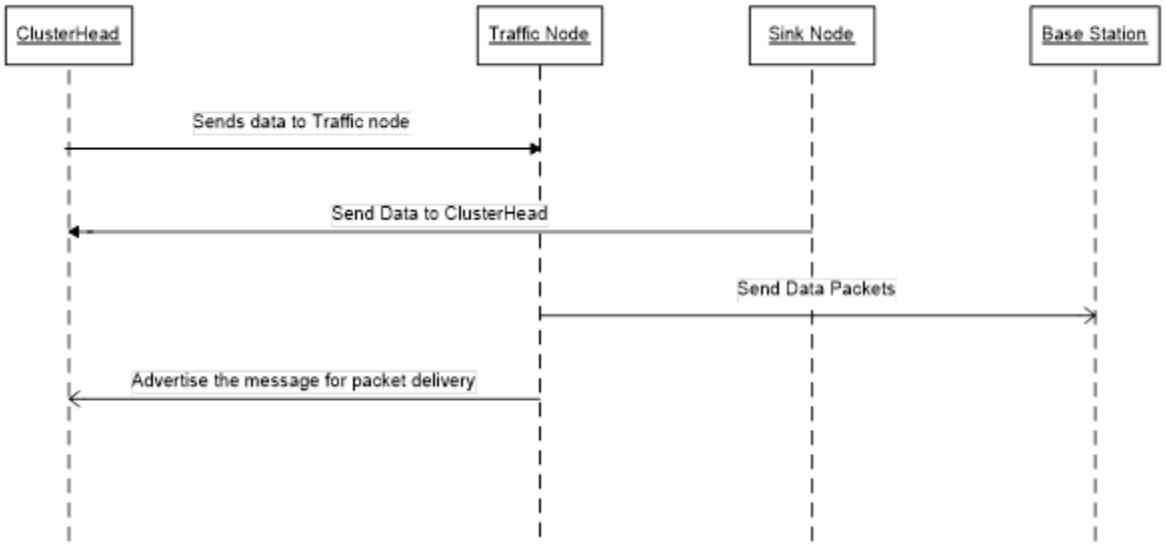

Figure 3.4 Sequence Diagram

4.2.4 Activity Diagram

- Activity diagram shows subsequent activities in a process. They are useful for modeling business, workflows, the data flows and complex algorithm.
- Activity diagram shows the activities happen in algorithm.
- Here each activity is happen in a series.
- It shows the path of the algorithm.

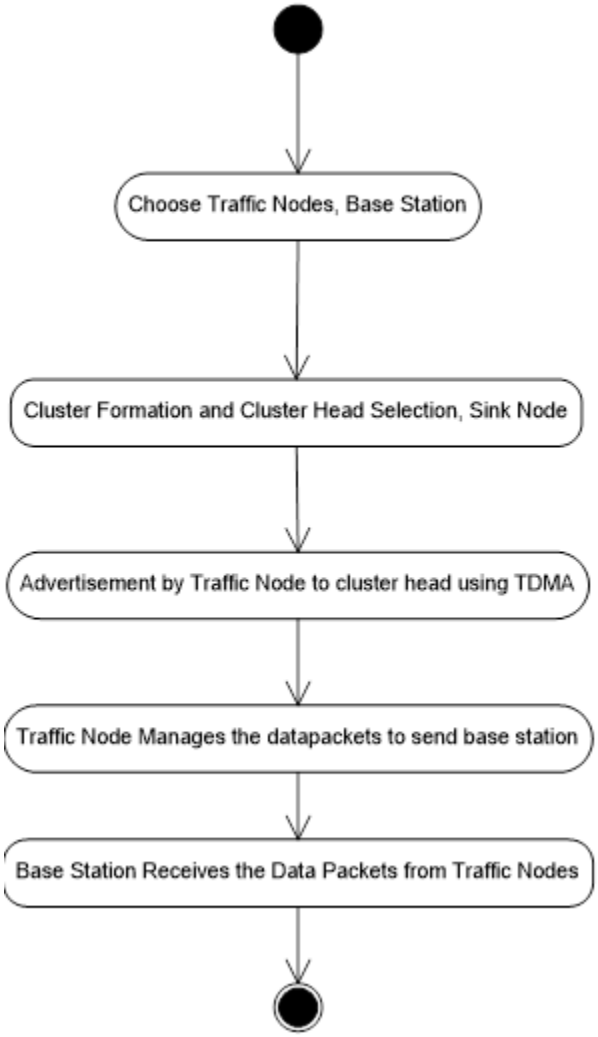

Figure 3.5 Activity Diagram

4.3 Summary

In this chapter proposed system design is discussed in detail. It is containing both Architecture and UML Diagrams. Based on weekly works if any changes are happened its updated properly in the existing Design. This design of each module is acted as a blue print during the development of product.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 Throughput Improved Multipath Multihop Hierarchical (TIMMH) Routing Algorithm

TIMMH routing algorithm describes the route of data packets in a hierarchical structure where the some group of nodes can make a form of cluster. Thus algorithm designed to improve throughput of the WSN. With consideration of Energy Aware Multihop Multipath Hierarchical (EAMMH) Protocol, TIMMH improves throughput. TIMMH has introduced the 'Traffic Node' between base station and cluster head. There are four kind of nodes needs to identified. (1) Sink Nodes, (2) Cluster Heads, (3) Traffic Nodes, (4) Base Station.

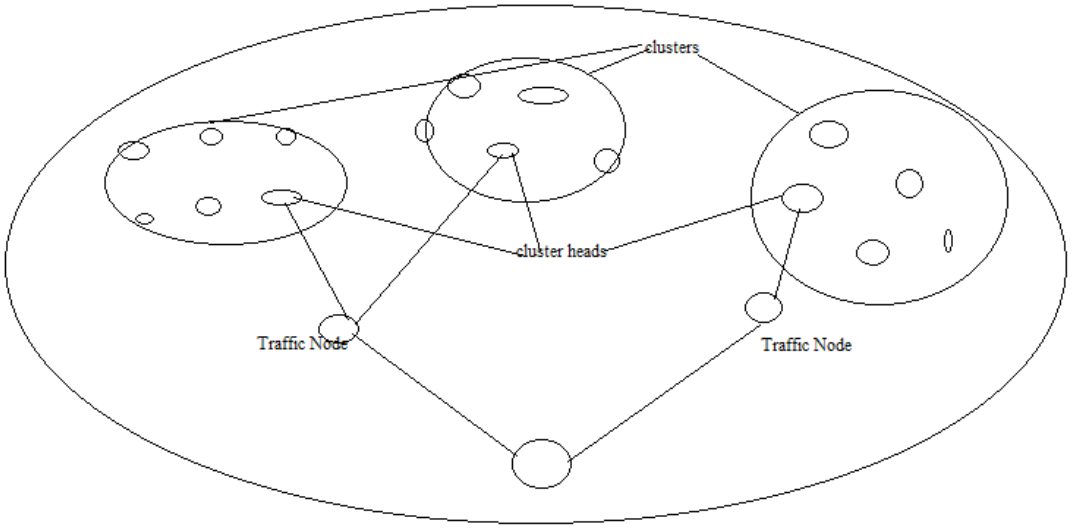

Figure 5.1 System Design

1. Sink Nodes

Sink Nodes are described to have different data packets. They communicate with the cluster head to forward the data packet. In this, three tier hierarchical architecture as shown in figure 4.1 sink nodes are at the lowest part of the system. Sink Nodes having limited battery power and computational capabilities. Sink Node will send the data packets to neighboring cluster head if value of waiting queue is high in its cluster head.

2. Cluster Heads

Cluster Heads are intermediate node between 'Traffic Node' and Sink Nodes. They forward the received data packets from sink nodes to traffic node. Cluster Heads have been identified as an important node as it plays vital role for the network that it also have the computational capabilities with battery power.

3. Traffic Nodes

Traffic Nodes is very important element of the network. Traffic Nodes have the maximum computational capabilities to be organized in such a way that they must intermediate between cluster heads and base station. We can say that the traffic nodes regulate the flow of the network. They receive the flow of the data packets from cluster heads. They forward it to base station. Traffic Node is very important that it can get throughput improved. So, the issue of congestion is also getting reduced. Because the packet loss has also reduced as packet delivery ratio is higher. It is also helps in terms of security. If an attacker attacks the network to access data it may get data from the traffic node but it may not get information from base station.

4. Base Station

Base Station is very important where all data packets collected. Base Station is very important for the user who analyses the data. Base Station has direct access to the network admin. From base station user can analyses the performance of network. In chapter 5, the figure of the network explains the situation of the network and also varying buffer size explains the effectiveness.

5.2 Network Simulator – 2

- Network Simulator -2 formally known as NS-2 is extremely valuable for the representation of algorithms.
- It is used for simulation of routing protocols of both wired or wireless network
- The Simulator is developed using C++. It has an OTcl interpreter which permits input code written in Tool Command Language (Tcl).
- It supports Tcl which has significant components created in classes, similar to object oriented fashion.
- It is openly distributed. The consequences of the tcl documents are recorded by trace files.
- From the trace files we can bring required results like throughput, delay, PDR etc.
- NS2 is a distinct event network simulator. It is aimed at network research, and supports simulation of different aspects of networking, including TCP, multicast, and routing protocols.

- NS2 will be utilized as a backend for the simulated world. It will be utilized to record the wireless topology as agents communicate and gather and calculate metrics. It will likewise be utilized to support different wireless/interference algorithms built into the simulator.

- NS2 is accomplished through a Tcl interface. A user composes the simulation in a Tcl script, and executes the script by running "ns mySim.tcl". NS2 likewise accompanies a different representation program called nam, that can be downloaded and installed with NS2 [26].

- Ordinarily, one will compose a simulation in Tcl, execute it in NS2, and take the subsequent yield record and execute "nammySim.out" to see the simulation run visually.

- NS2 was inherent C++ and gives a simulation interface through OTcl, an object-oriented part of Tcl. The user portrays a network topology by composing OTcl scripts, and after that the primary ns2 program simulates that topology with determined parameters. It keeps running on Linux, FreeBSD, Solaris, Macintosh OS X and on Windows utilizing Cygwin. It is authorized for use under version 2 of the GNU General Public License.

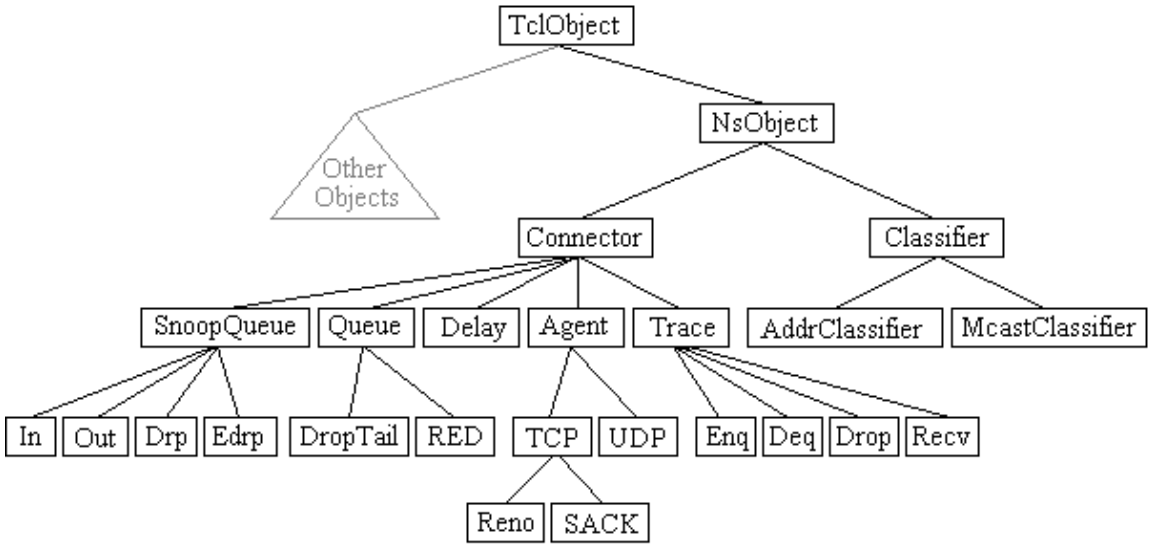

Figure5.2 Classification of Tcl Objects

5.3 Tcl

Tcl is an exceptionally basic scripting language, like Perl or Python [25]. The graphics toolkit frequently joined by the language. OTcl is an object oriented part of Tcl, giving everything of the power and usefulness of any OO language.

For the project, Tcl utilized as essential technique for conjuring NS2 simulations. Tcl is the essential technique of interfacing with ns2. Versions of Tcl/Tk and OTcl can be downloaded alongside ns2 at its website, which is in Appendix A in this report.

NS2 simulator is based on two languages:

- C++ as the programming language for the genuine simulation part since it is intense, effective and quick to run.
- OTcl, a scripting language, to design the simulation environment, set the required parameters and control the simulation.

NS has both facility with two languages, C++ and OTcl. C++ is quick to run however slower to change, making it appropriate for detailed protocol implementation. OTcl

runs slower yet can be altered very fast, making it perfect for simulation setup. NS(via tclcl) provides attachment to make objects and variables appear on both langauges.

This is because the objects used by the simulator are well settled and when the user needs just to put them assemble to make a simulation environment and to modify some parameters, it is quicker to compose and change an OTcl simulation script instead of alter and recompile the entire C++ simulator code. At the point when extra functionalities are required that are not accessible with standard ns2 objects, a user is permitted to compose custom C++ code to actualize new objects to be incorporated in ns2. Detailed simulations of protocols need a systems programming language which can proficiently control bytes, packet headers, and implement algorithms that run over substantial data sets. For these undertakings run time speed is critical and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important.

A large part of network research includes marginally fluctuating parameters or setups, or rapidly investigating various situations. In such cases, iteration time is more vital. Since configuration runs once (at the beginning of the simulation), run-time of this part of the assignment is less imperative.

Utilize OTcl:

• for arrangement, setup, and "one-time" stuff

• In the event that you can do what you need by controlling existing C++ objects

Furthermore, utilize C++:

• If the chance that you are doing *anything* that requires handling each packet of a flow

• If you need to change the conduct of current C++ class in ways that weren't expected.

The class Tcl encapsulates the actual instance of the OTcl interpreter, and gives the methods to get access and communicate with that interpreter. The methods portrayed in this section are applicable to the *ns* programmer who is composing C++ code.

The class gives methodology for the following operations:

• earn a reference to the Tcl case;

• call upon OTcl procedures over the interpreter;

• go back results to the interpreter;

• report error positions and exit in an uniform situation;
• store and lookup “TclObjects”.
• Obtain direct reachability to the interpreter.

5.4 XGraph

- XGraph is a functionality provided with NS-2 package.
- XGraph is very important for the representation of the results of algorithms like throughput, delay, PDR, network lifetime etc.
- It is a graphical representation of the performance of the algorithm.
- XGraph is an application that includes interactive plotting and graphing, animation and deritives, portability and bug fixes.
- A successful installation of NS-2 will install xgraph also.
- To run xgraph from a terminal: #xgraphfilename.xg
- Inside TCL scripts, xgraph should written as: exec xgraphfilename.xg –geometry 500*500
- The values of the graph are taken from the respective trace files.

5.5 Disadvantages

Modeling is an intensely difficult and time consuming task, as NS-2 has been criticized for a long time. Since it has no “Graphical User Interface” (GUI) and everyone needs to learn scripting language, queuing theory various modeling techniques. Of late, there have been objections that results are not constant and that some certain protocols are filled with bugs

Node basics

The essential primitive for creating a node is

“set ns”
“$ns node”

In a case, procedure node constructs a node out of simpler classifier objects. Node is a class in OTcl. Still a large portion of the components of the node are themselves Tcl Objects. The structure of a node is as shown in Figure 5.3. This simple structure contains of

two Tcl Objects: an address classifer(classifer_) and a port classifier (dmux_). Classifiers have to distribute the incoming packets.

Components of the nodes are:

• “an address or id_, monotonically increasing by 1 (from initial value 0) over the simulation namespace as nodes are created”

• “a list of neighbors (neighbor_)"

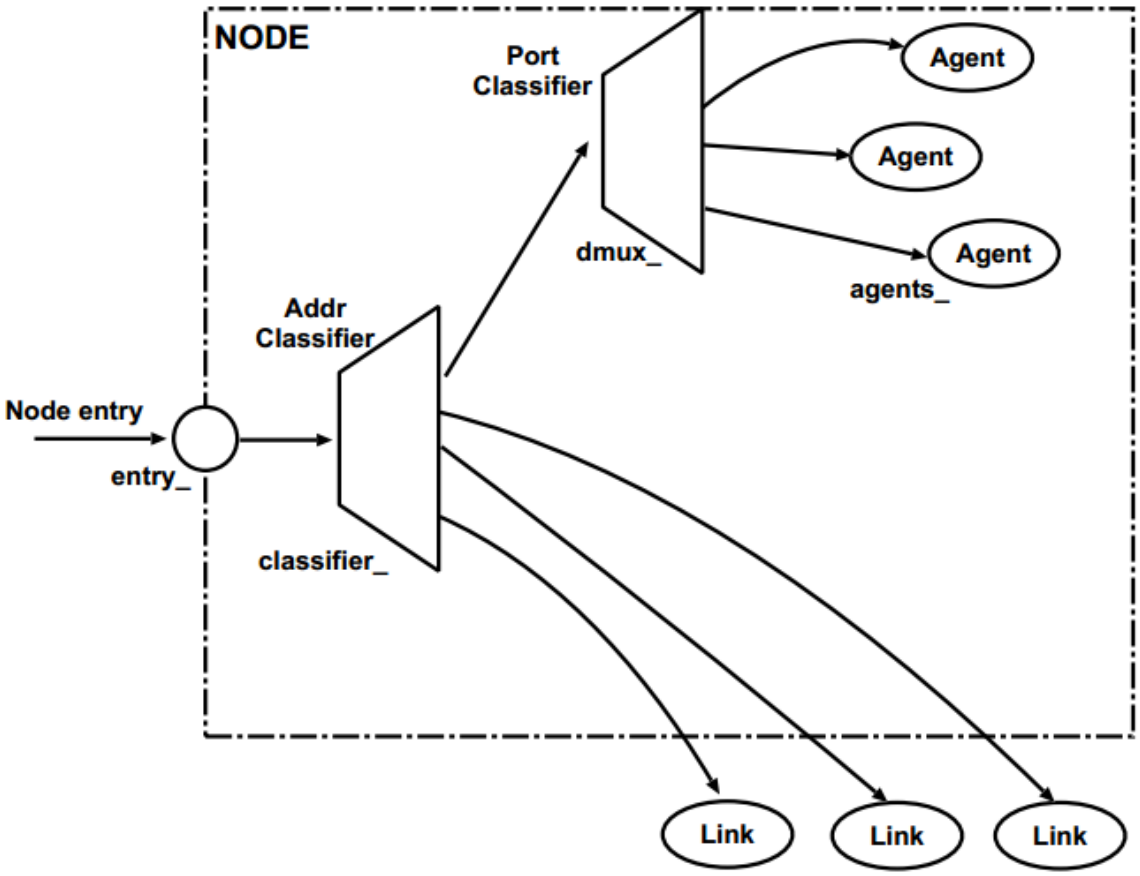

Figure 5.3 Simple structure consists of two TclObjects

option	available values	default
general		
addressType	flat, hierarchical	flat
MPLS	ON, OFF	OFF
both satellite- and wireless-oriented		
wiredRouting	ON, OFF	OFF
llType	LL, LL/Sat	""
macType	Mac/802_11, Mac/Csma/Ca, Mac/Sat, Mac/Sat/UnslottedAloha, Mac/Tdma	""
ifqType	Queue/DropTail, Queue/DropTail/PriQueue	""
phyType	Phy/WirelessPhy, Phy/Sat	""
wireless-oriented		
adhocRouting	DIFFUSION/RATE, DIFFUSION/PROB, DSDV, DSR, FLOODING, OMNIMCAST, AODV, TORA, M-DART PUMA	""
propType	Propagation/TwoRayGround, Propagation/Shadowing	""
propInstance	Propagation/TwoRayGround, Propagation/Shadowing	""
antType	Antenna/OmniAntenna	""
channel	Channel/WirelessChannel, Channel/Sat	""
topoInstance	<topology file>	""
mobileIP	ON, OFF	OFF
energyModel	EnergyModel	""
initialEnergy	<value in Joules>	""
rxPower	<value in W>	""
txPower	<value in W>	""
idlePower	<value in W>	""
agentTrace	ON, OFF	OFF
routerTrace	ON, OFF	OFF
macTrace	ON, OFF	OFF
movementTrace	ON, OFF	OFF
errProc	UniformErrorProc	""
FECProc	?	?
toraDebug	ON, OFF	OFF
satellite-oriented		
satNodeType	polar, geo, terminal, geo-repeater	""
downlinkBW	<bandwidth value, e.g. "2Mb">	""

Table 5.1 Available Options for Node Configuration

5.6 Summary

This chapter listed the softwares chosen and the programming Languages.The implementation of the proposed algorithm is simulated with NS-2. XGraph used for the presentation of the parameters of the algorithm.

CHAPTER 6

SIMULATION TESTING BY VARYING BUFFER SIZE

6.1 Test Case 1 - 30 Nodes

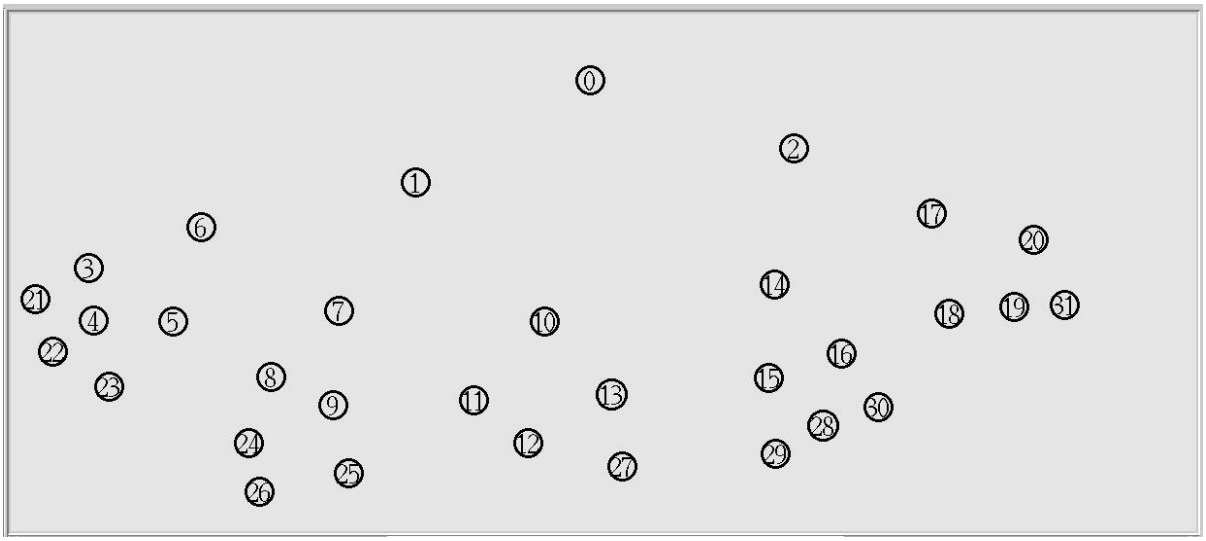

Figure 6.1 WSN with 30 Nodes

- The WSN of 30 nodes simulated in NS-2 for 30 seconds with udp (cbr) link.
- Packet Size is 512 kb.
- Node 0 is base station.
- Node 1 and 2 are Traffic Node.
- Node 6, 7, 10, 14, 17 are cluster heads.
- Remaining nodes are sink nodes.

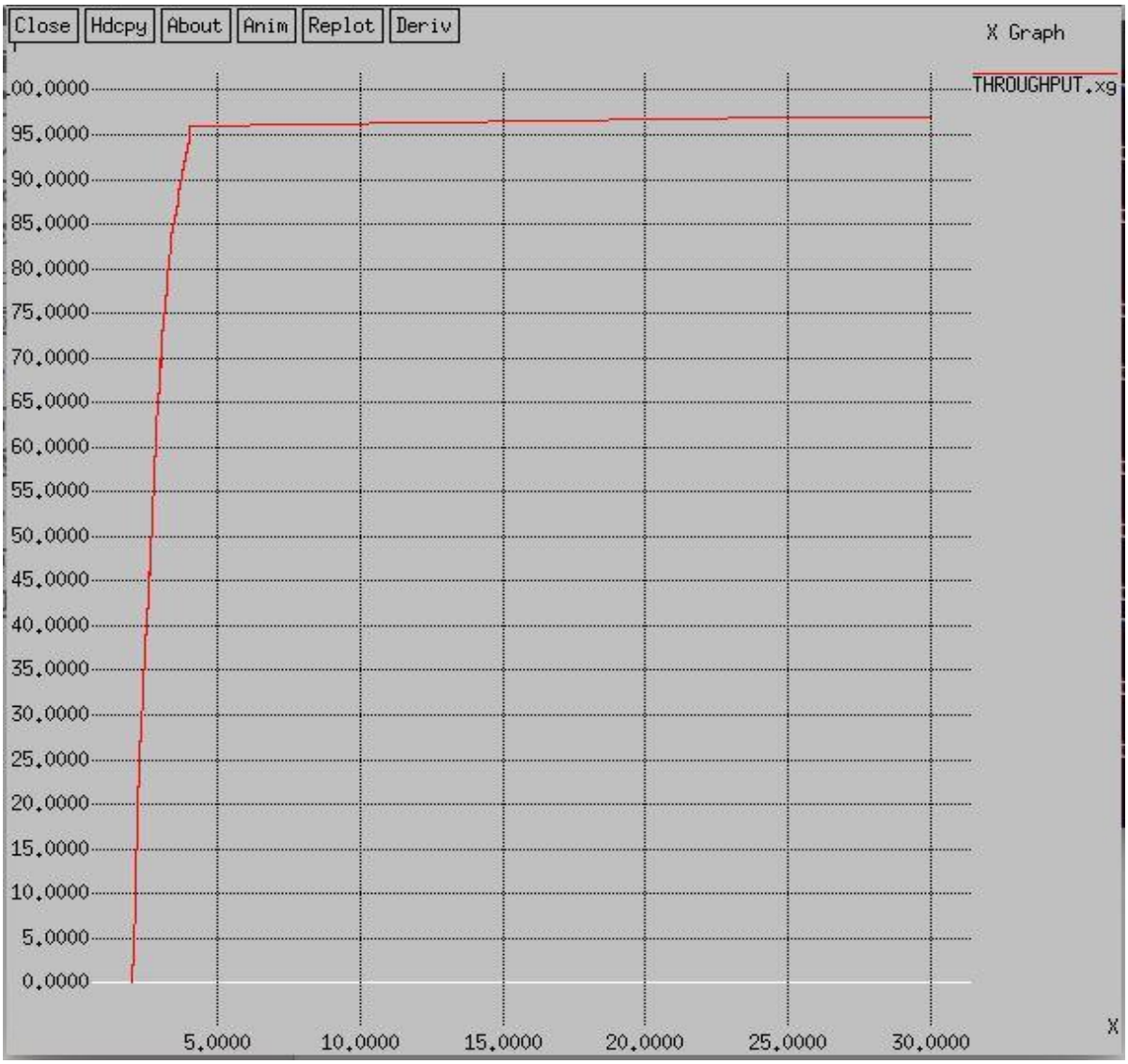

Figure 6.2Throughput for 30 Nodes

6.1.1 Throughput Details

- X- axis ->Simulation Time
- Y- axis-> Throughput (kbps)
- The graph shows throughput has increased as each cluster head has assigned different time to send packets to the nearest Traffic Node.
- The nodes are continuously performing in a timely manner as it has maintained throughput.

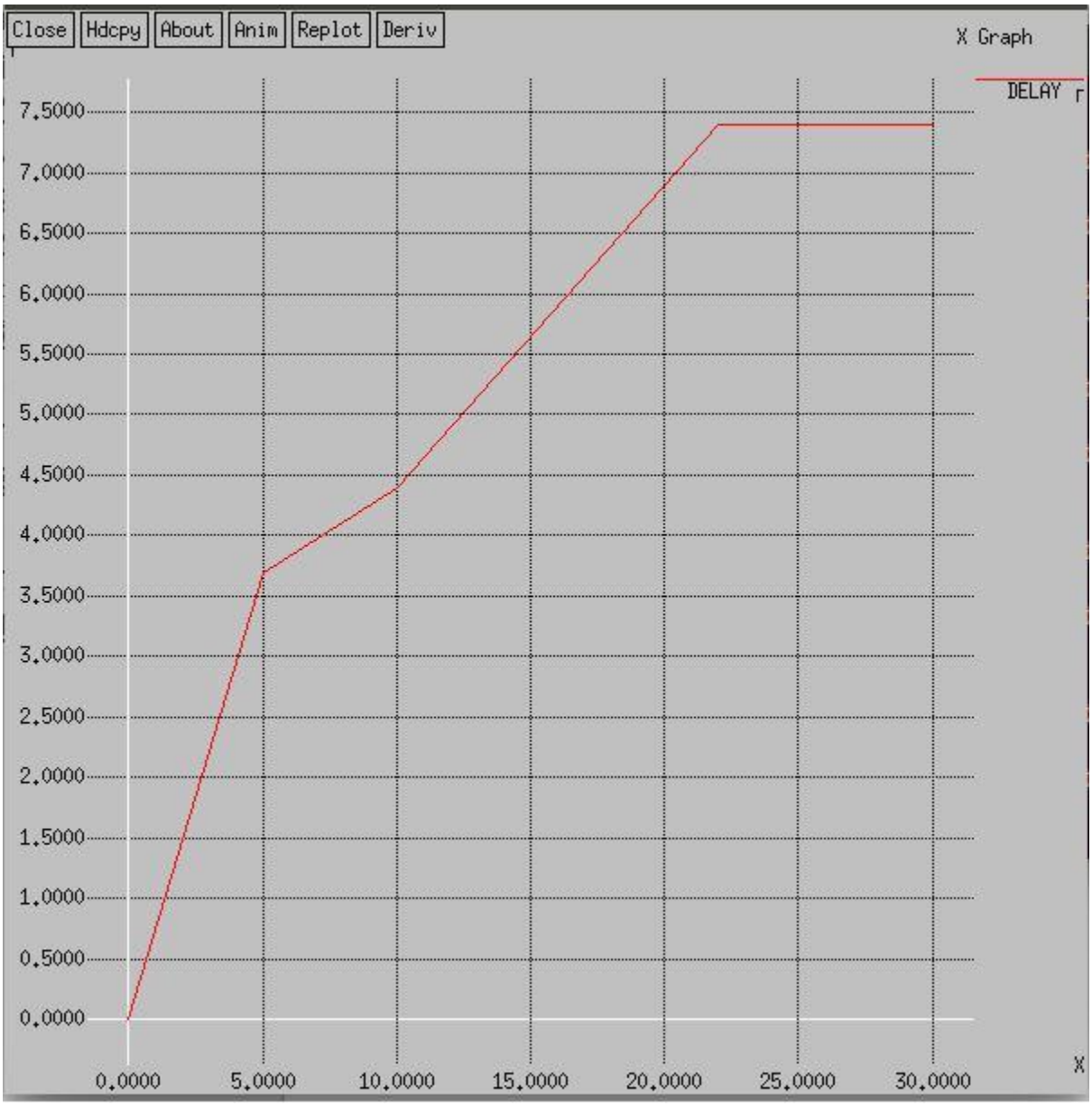

Figure 6.3 Delay 30 Nodes

6.1.2 Delay Details

- X-axis-> Simulation Time
- Y-axis-> Delay(ms)
- The graph shows delay in performing tasks assigned to the nodes as more nodes included.
- The simulation time is not sufficient for all nodes to perform efficiently.

6.1.3 Packet Delivery Ratio- udp (cbr) link s: 1898 r:1896, r/s Ratio: 0.9989

- Packet Delivery Ratio shows the performance of the network.

- The sent and received packets are described to have such values to be used for evaluation of the network performance.

6.2 Test Case 2 - 50 Nodes

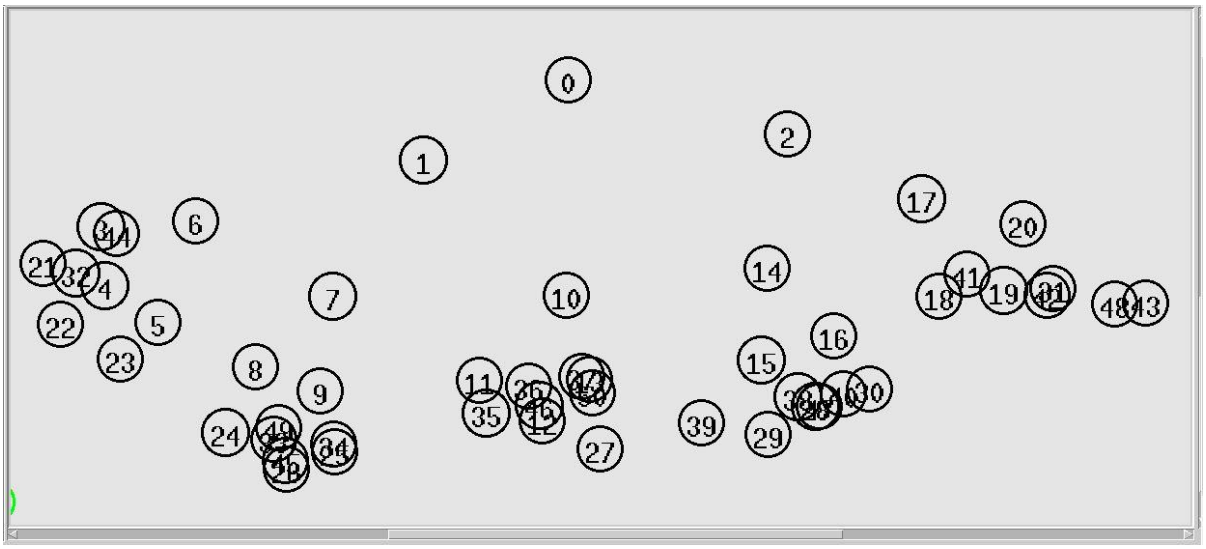

Figure 6.4 WSN with 50 Nodes

- The WSN of 50 nodes simulated in NS-2 for 30 seconds with udp (cbr) link.
- Each cluster has the cluster-heads.
- Node 1 and 2 are traffic nodes.
- Node 0 is base station.
- Node 6, 7, 10, 14, 17 are cluster heads.
- Remaining nodes are sink nodes.

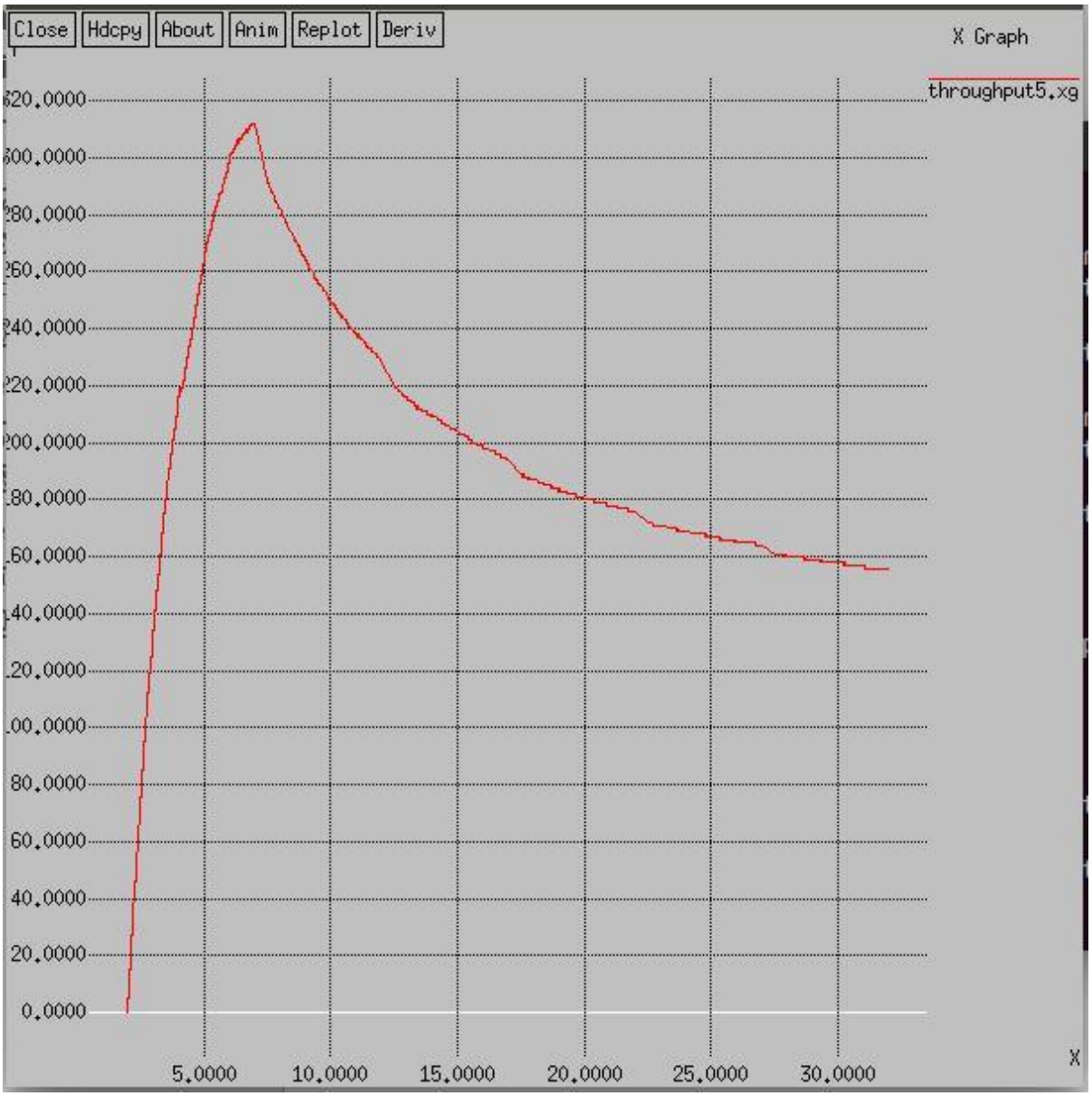

Figure 6.5 Throughput for 50 Nodes

6.2.1 Throughput Details

- X-axis-> Simulation Time
- Y-axis-> Throughput(kbps)
- The graph shows the variation in throughput.
- As the node gets increased the value of the throughput has also increased but it has also gets down as it has decreased with the simulation time.

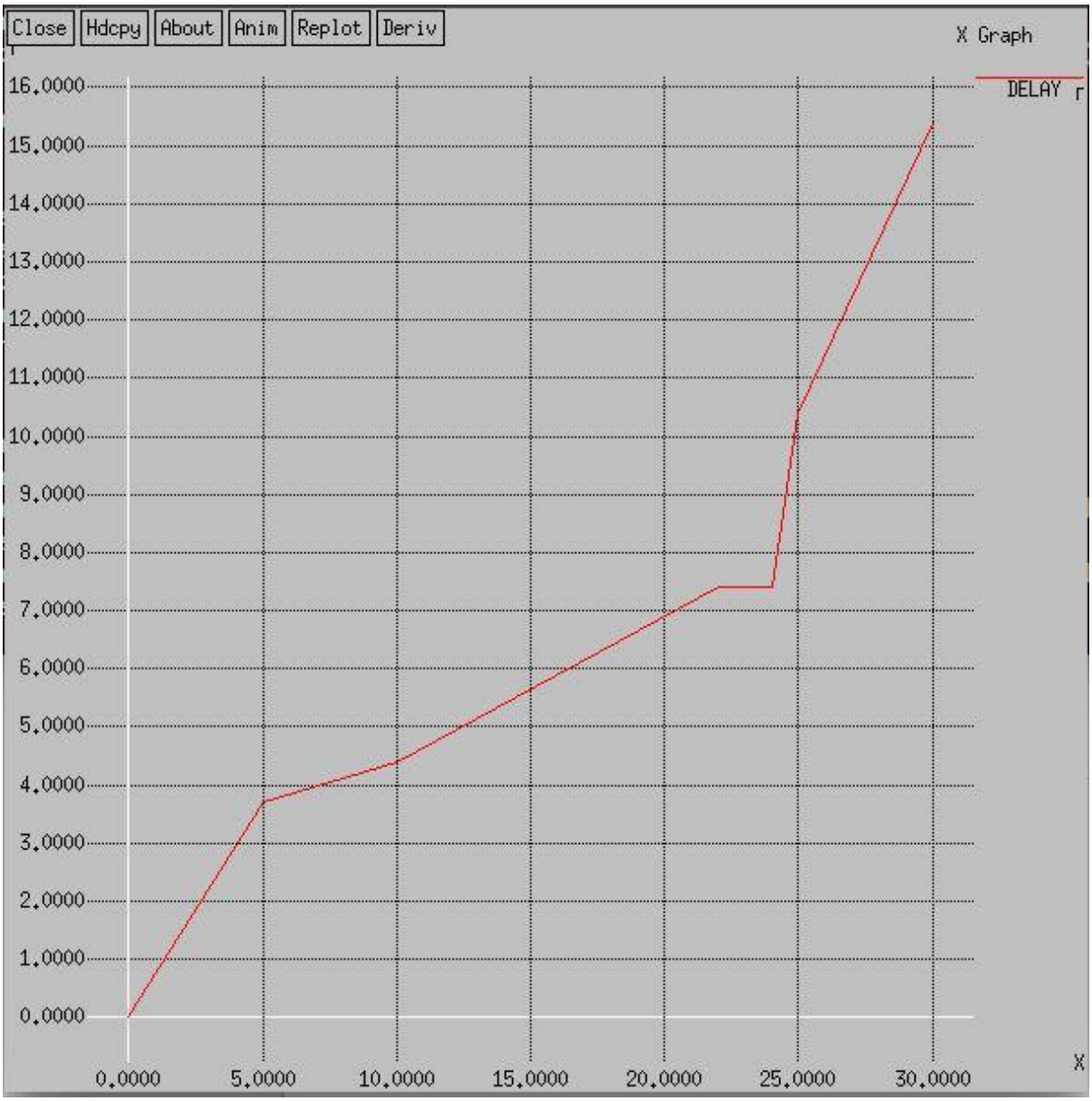

Figure 6.6 Delay for 50 Nodes

6.2.2 Delay Details

- X-axis->Simulation Time
- Y-axis-> Delay(ms)
- Delay graph has shown increase in delay as it has increased in nodes.
- Each node takes time to perform assigned task as 50 nodes takes time to perform.

6.2.3 Packet Delivery Ratio-cbr (udp) s:2504 r:2490, r/s Ratio:0.9944

- Packet Delivery Ratio shows the value as performance has increased the value of the PDR.
- The sent and received packets are shown that link has performed well in the network.

6.3 Test Case 3 - 75 Nodes

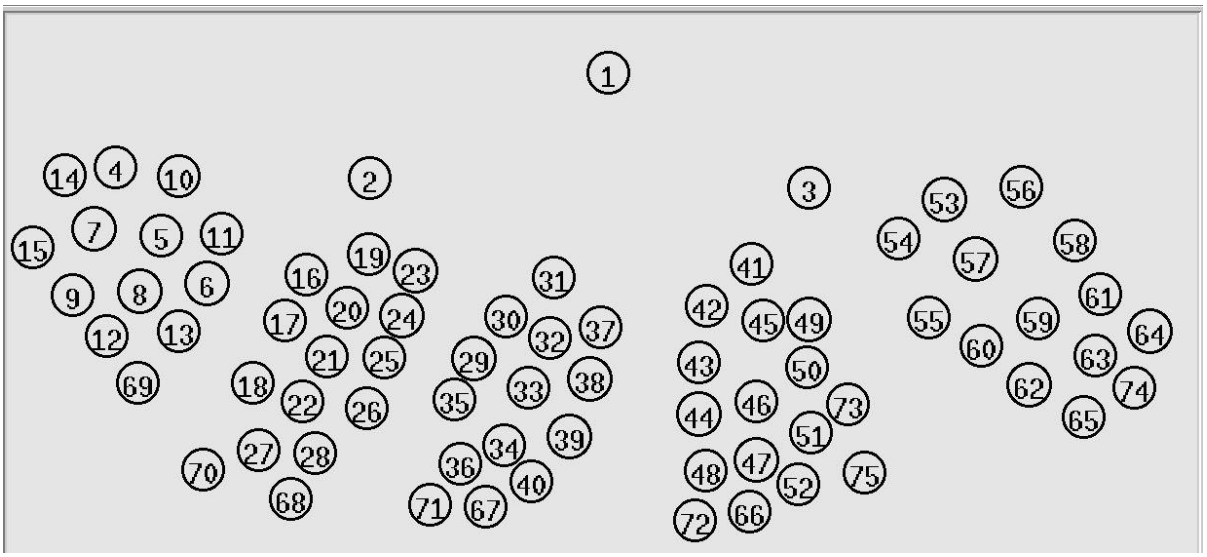

Figure 6.7 WSN with 75 Nodes

- The WSN of 75 nodes simulated in NS-2 for 30 seconds with udp (cbr) link.
- Node 1 is base station.
- Node 2 & 3 are traffic nodes.
- Node 10, 19, 31, 41, 54 are cluster heads.
- Cluster-head 10, 19, 31 communicate to node 2.
- Cluster-head 41, 54 communicate to node 3.

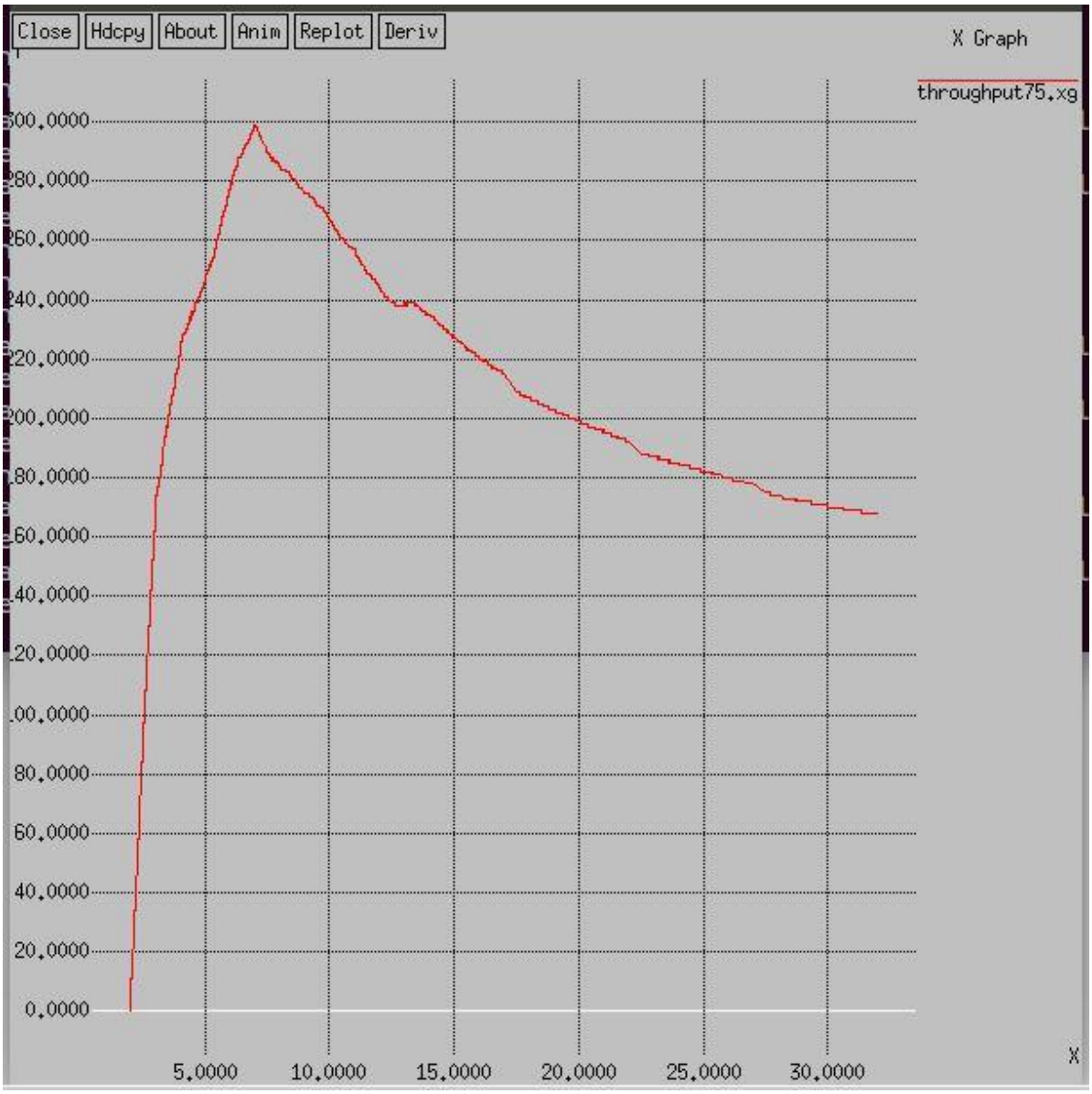

Figure 6.8 Throughput for 75 Nodes

6.3.1 Throughput Details

- X-axis-> Simulation Time
- Y-axis->Throughput(kbps)
- Throughput has increased but as node uses excessive energy to perform tasks throughput decreases.
- As the node density increases the throughput gets decreases but gets stable after certain nodes.

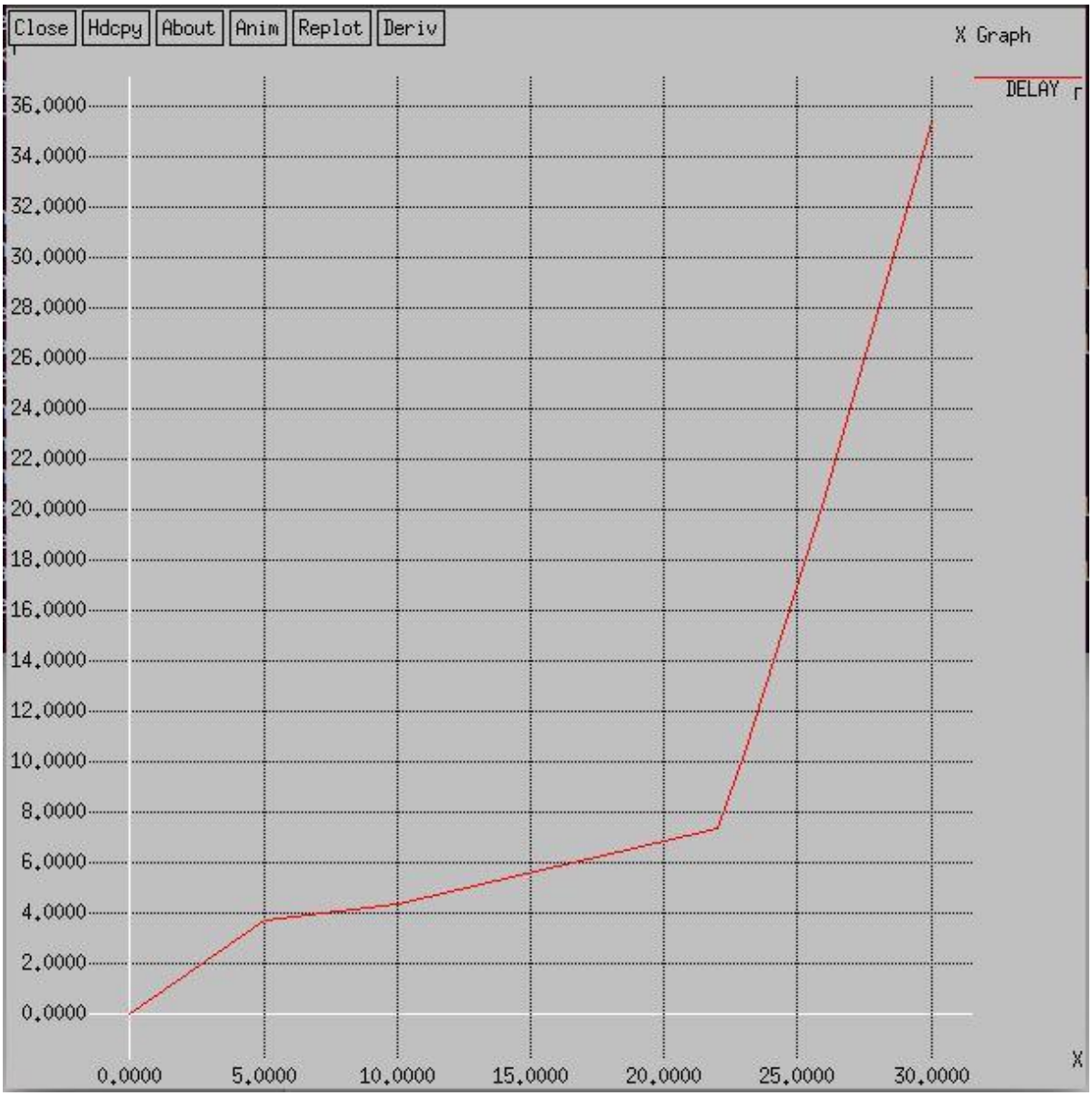

Figure 6.9 Delay for 75 Nodes

6.3.2 Delay Details

- X-axis-> Simulation Time
- Y-axis-> Delay(ms)
- Delay has increased due to increase in number of nodes.
- There are 75 nodes that each node has less time to perform as simulation time is 30 sec.
- So delay has increased as less time to for each node to perform.

6.3.3 Packet Delivery Ratio – cbr (udp) s:3027 r: 2665, r/s Ratio: 0.8804

- The value of PDR states that the less time has increased in packet loss.
- Packet Loss has increased because sort of time in performance by each node.

6.4 Summary

This chapter includes the representation of the algorithm using different buffer sizes. Each buffer size has included with graphs of throughput, delay and value PDR.

CHAPTER 7

COMPARISON WITH EAMMH

7.1 TIMMH vs. EAMMH (30 Nodes)

Parameters	TIMMH	EAMMH
Number of Nodes	75	75
Throughput	High	Low
Delay	High	Low
Packet Delivery Ratio	0.9989	0.8822

Table 7.1 Comparison of 30 Nodes

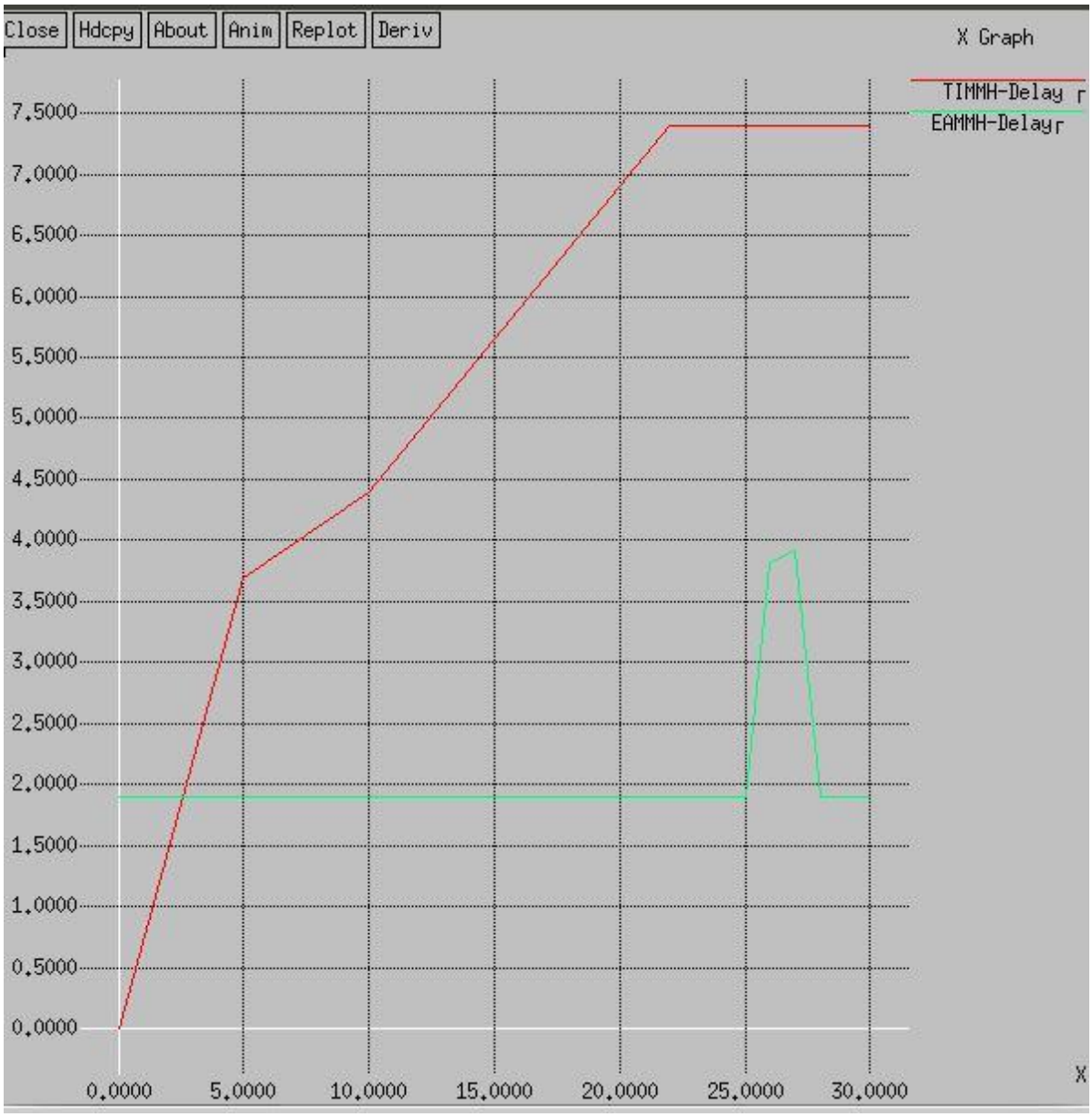

Figure 7.1 Comparison of Delay for 30 Nodes for

7.1.1 Delay Comparison of 30 Nodes

- X-axis-> Simulation Time/ No. of Rounds, Y-axis->Delay(ms)
- Here red line indicates TIMMH which is for Simulation time vs. Delay.
- The green line indicates EAMMH which is for No.of Rounds vs. Delay.
- Both indicate sequential increase in Delay because of sufficient time and node.
- As node increases the delay has also increased because each round of nodes has different energy levels.

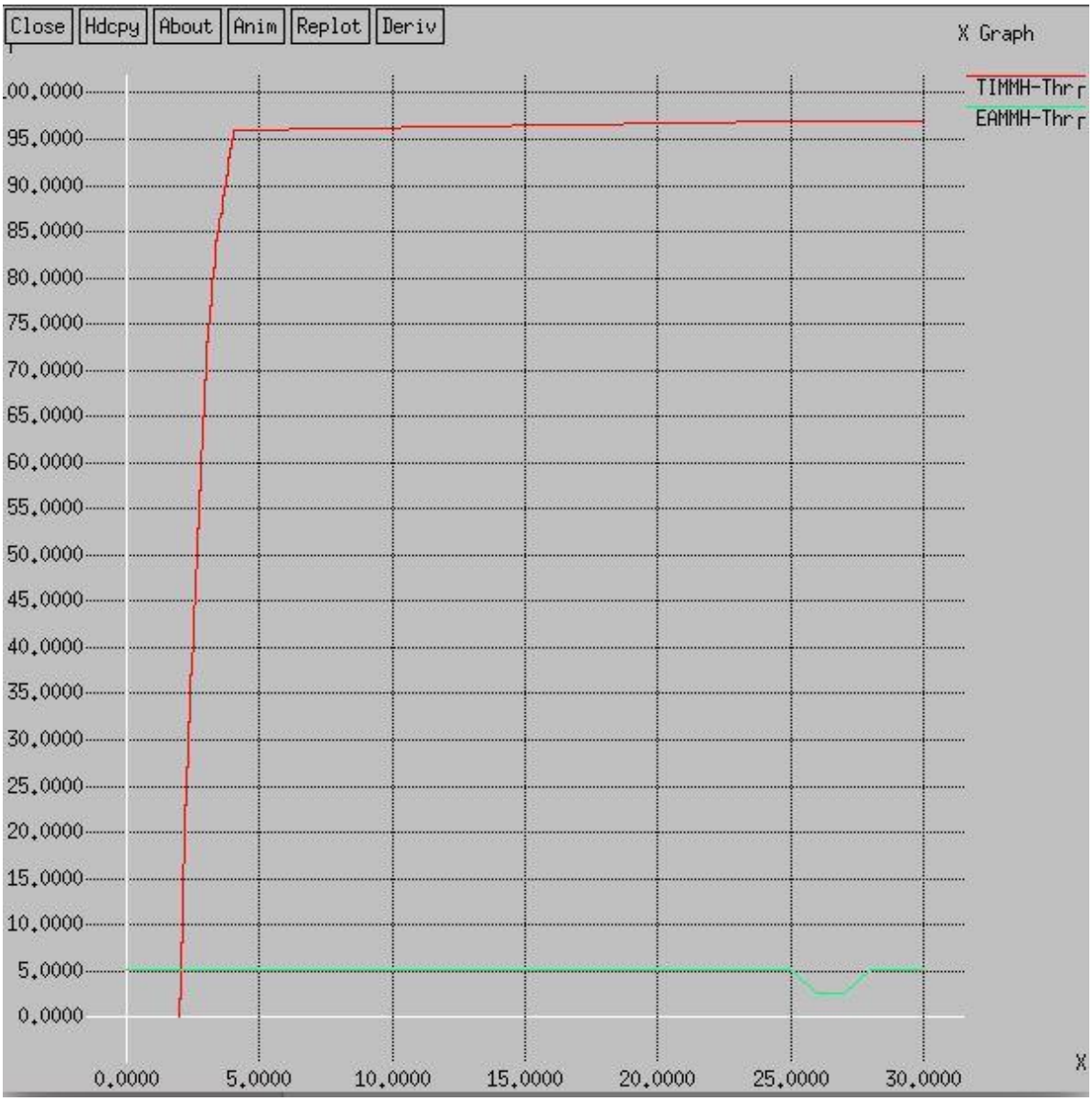

Figure 7.2 Comparison of Throughput for 30 Nodes

7.1.2 Throughput Comparison of 30 Nodes

- X-axis-> Simulation Time/ No. of Rounds, Y-axis-> Throughput(kbps)
- Red Line in graph shows throughput of TIMMH which is for Simulation time vs. Throughput.
- Green Line in graph shows throughput of EAMMH which is for No.of Rounds vs. Throughput.
- In TIMMH algorithm, the graph shows a steady value after more nodes involved.
- In EAMMH algorithm, the graph shows a variation in the value of throughput.

7.2 TIMMH vs. EAMMH (50 Nodes)

Parameters	TIMMH	EAMMH
Number of Nodes	75	75
Throughput	High	Low
Delay	High	Low
Packet Delivery Ratio	0.9944	0.9698

Table 7.2 Comparison 50 Nodes

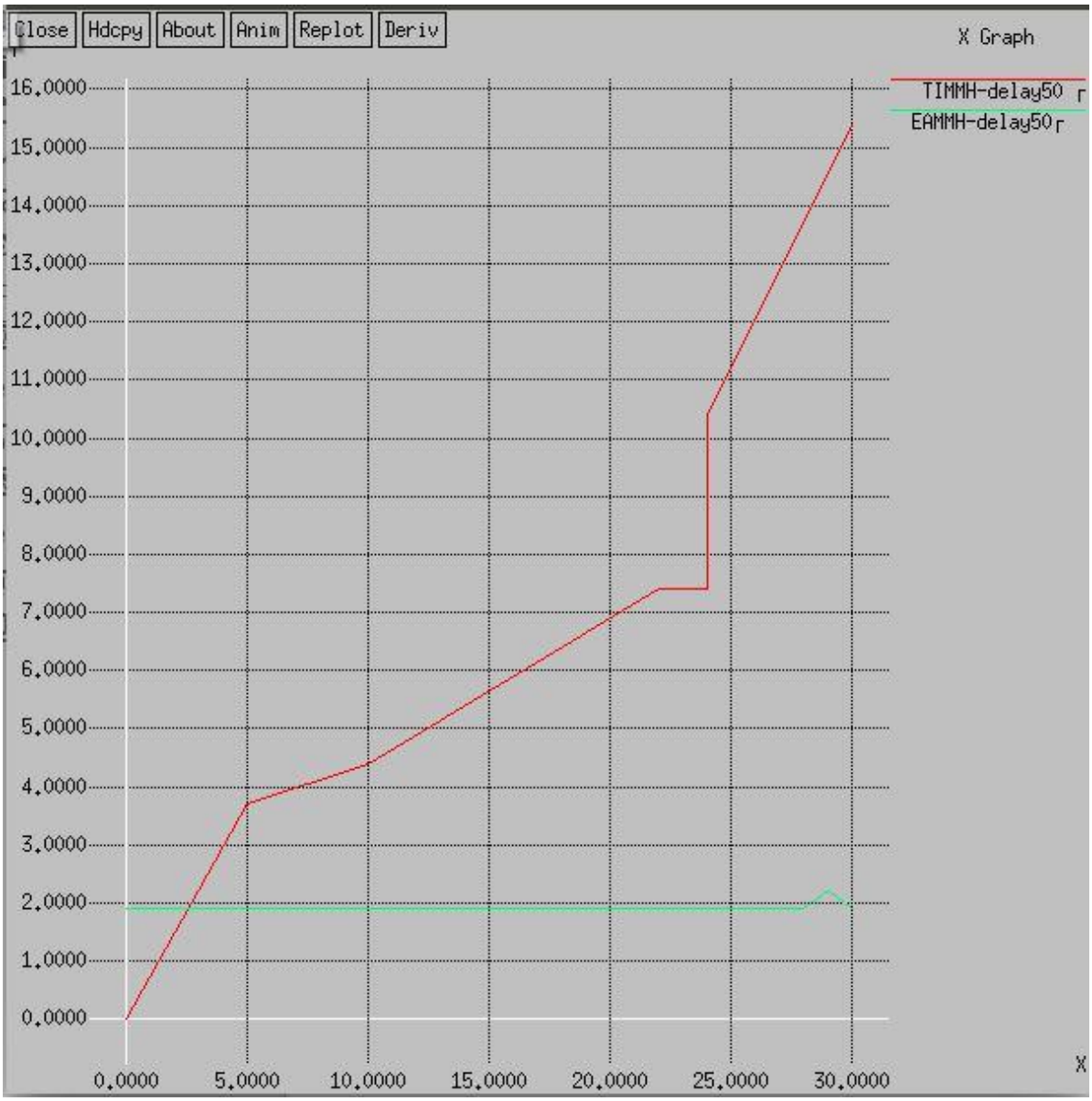

Figure 7.3 Comparison of Delay for 50 Nodes

7.2.1 Delay Comparison of 50 Nodes

- X-axis-> Simulation Time/ No. of Rounds, Y-axis->Delay(ms)
- Here red line indicates TIMMH, Green line indicates EAMMH.
- For TIMMH, Simulation Time vs. Delay and for EAMMH, No. of Rounds vs. Delay
- Both indicate sequential increase in Delay.
- As node increases the delay has also increased.

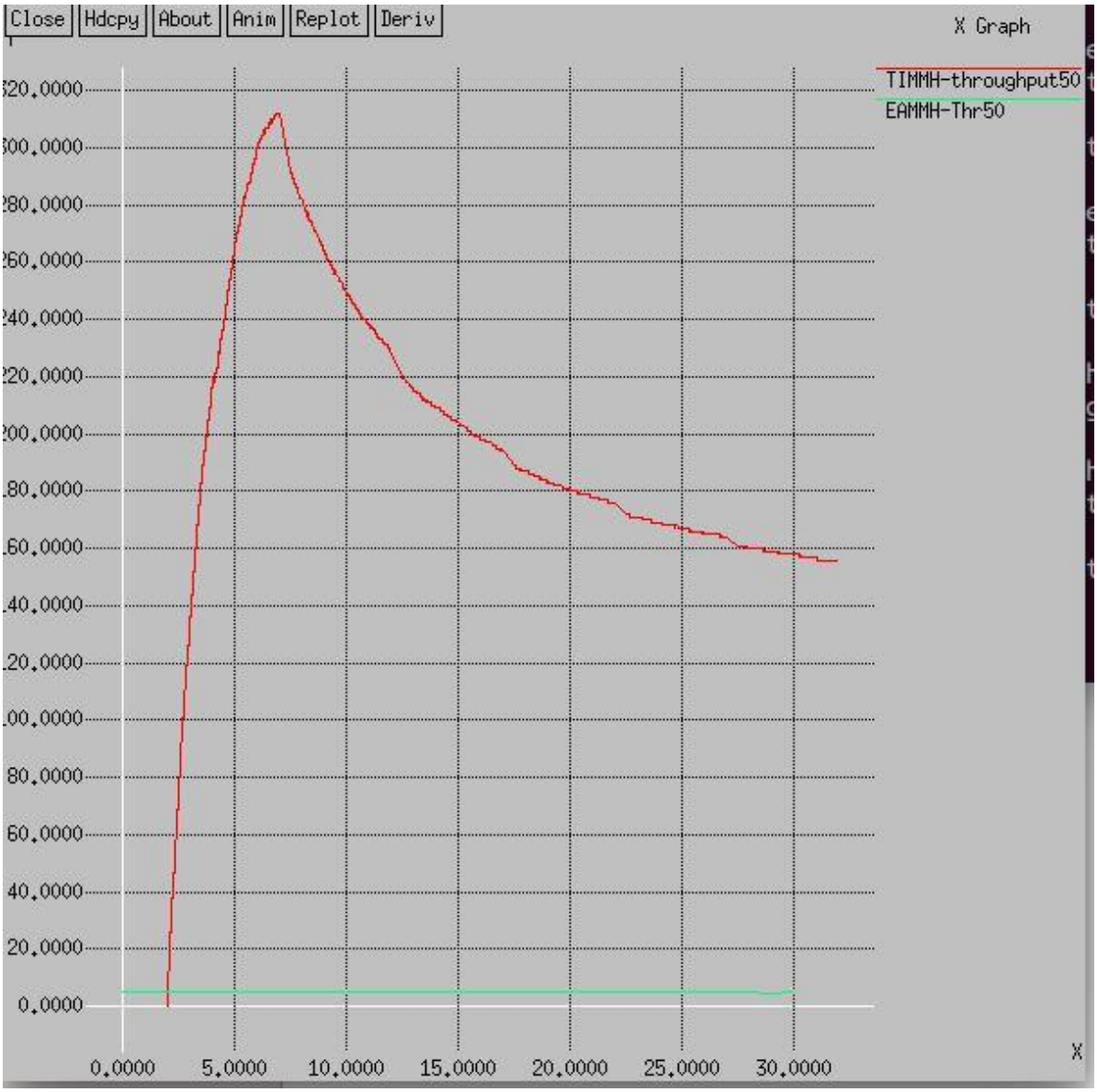

Figure 7.4 Comparison of Throughput for 50 Nodes

7.2.2 Throughput Comparison 50 Nodes

- X-axis-> Simulation Time/No. of Rounds, Y-axis-> Throughput(kbps)
- Red Line indicates TIMMH for Simulation Time vs. Throughput, Green indicates EAMMH which is for No. of Rounds vs. Throughput.
- In TIMMH, the graph shows the throughput is getting low because of the simulation time and also the node density has also increased.
- In EAMMH, the graph shows a variation as increased as each round of the simulation has different energy levels.

7.3TIMMH vs. EAMMH (75 Nodes)

Parameters	TIMMH	EAMMH
Number of Nodes	75	75
Throughput	High	Low
Delay	High	Low
Packet Delivery Ratio	0.9784	0.9640

Table 7.3 Comparisonof 75 Nodes

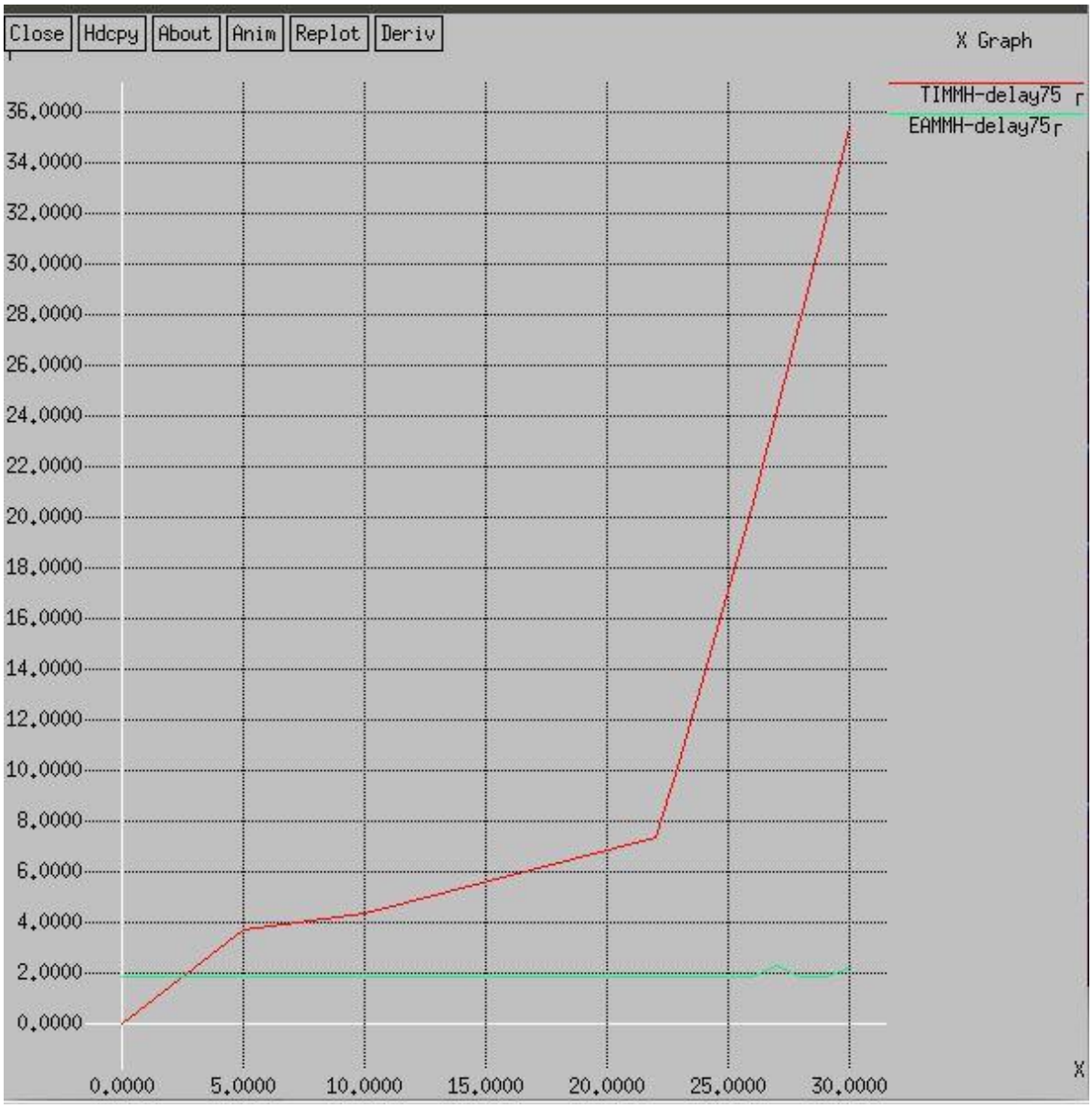

Figure 7.5 Comparison of Delay for 75 Nodes

7.3.1 Delay Comparison of 75 Nodes

- X-axis-> Simulation Time/No .of Nodes, Y-axis->Delay(ms)
- Red line indicates TIMMH which is for Simulation Time vs. Delay and Green line indicates EAMMH which is for No. of Rounds vs. Delay.
- In TIMMH, delay is very high as the 75 nodes have assigned with the different time to send packets. As it have more nodes to perform in less time.
- In, EAMMH the delay is very low because for each round cluster heads have changed and performance have also improved.

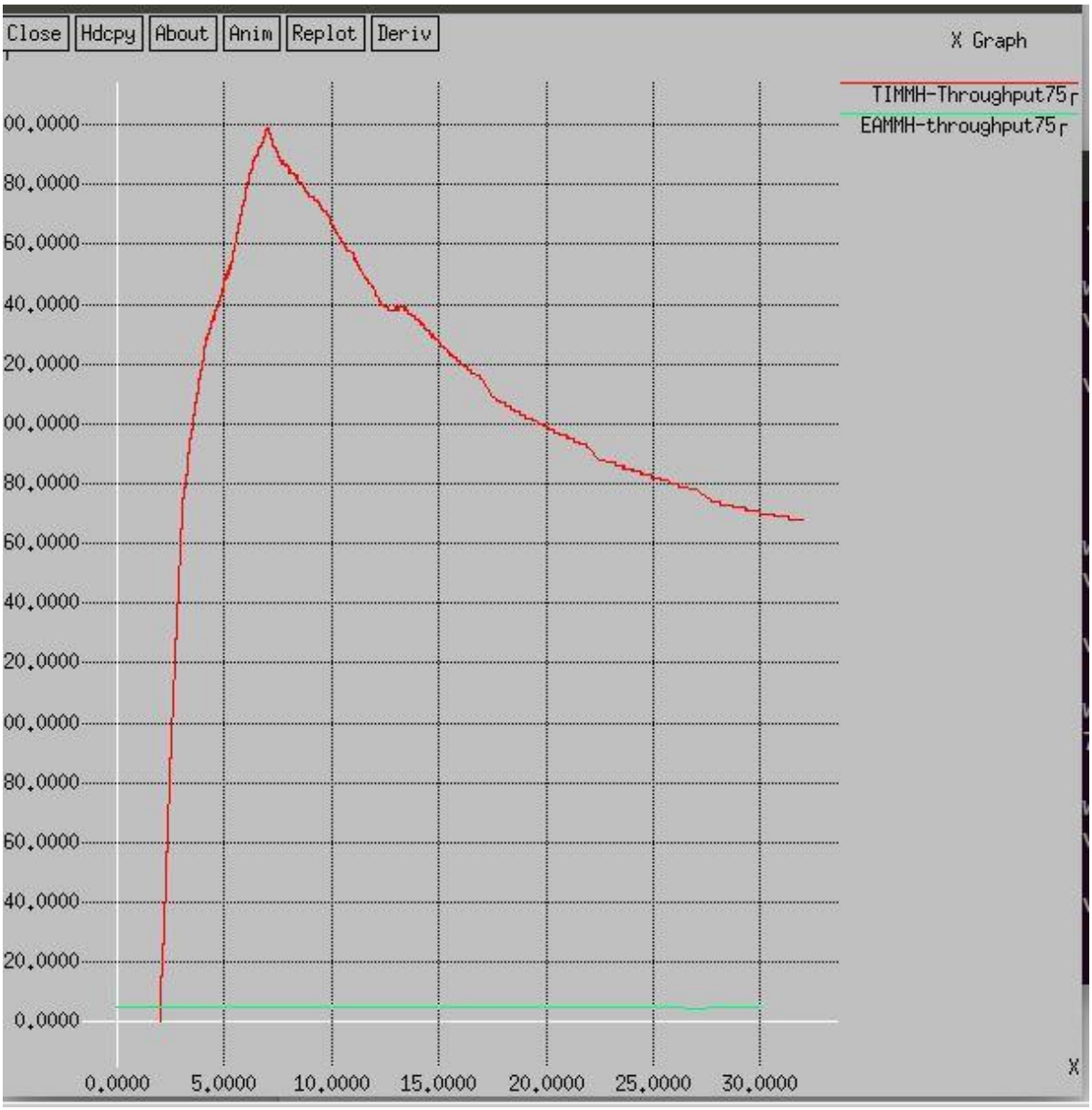

Figure 7.6 Comparison of Throughput for 75 Nodes

7.3.2 Throughput Comparison 75 Nodes

- X-axis-> Simulation Time/No. of Rounds, Y-axis-> Throughput(kbps)
- Red Line shows throughput of TIMMH for Simulation Time vs. Throughput and Green is for EAMMH to show No .of Rounds vs. Throughput.
- In TIMMH algorithm, as simulation time is less throughput is also decreasing.
- As node density is higher and simulation time is also less, nodes cannot perform well.

- In EAMMH algorithm, the throughput is low as various factors like energy, rounds of packet forwarding, death of node etc.

7.4 Summary

This chapter includes comparison of the proposed algorithm TIMMH with EAMMH. The graphs of the Throughput and delay show the comparison between the performances of the algorithm.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this project, we presented Throughput Improved Multihop Multipath Hierarchical Algorithm (TIMMH). It has ability to improve Throughput and PDR. Both theoretical analysis and simulation results show that it has a better routing performance than EAMMH. We have also seen higher delay but it gives higher Throughput and PDR. Delay causes because of the multiple paths, cluster interoperation with Traffic Nodes and security are yet to be resolved successfully for cluster and tree based routing algorithm. Security is also improved compared to EAMMH as the attacker may not know the real location of the base station.

REFERENCES

[1] Monica R. Mundada, V. CyrilRaj, T. Bhuvneshwari, "Energy Aware Multihop Multipath Hierarchical Routing Protocol for Wireless Sensor Network", European Journal of Scientific Research ,ISSN 1450-216X Vol. 88 No 4 October, 2012, pp.520-530

[2] Monica R. Mundada, Pranav B. Desai, Meeradevi, "A Survey of Congestion in Wireless Sensor Networks", International Conference on Advances in Human Machine Interactions (HMI-2016) 3-5 March-2016, DOI: http://dx.doi.org/10.1109/HMI.2016.7449195

[3] Almir Davis, Hwa Chang, "A SURVEY OF WIRELESS SENSOR NETWORK ARCHITECTURES", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.6, December 2012.

[4] Ghanavati S., Abawajy J., Izadi D., "A Fuzzy Technique to Control Congestion in WSN", The International Joint Conference on Neural Networks (IJCNN), 2013.

[5] N. Thrimoorthy and Dr. T .Anuradha, "A Review on Congestion control Mechanisms in Wireless Sensor Networks" Int. Journal of Engineering Research and Applications ISSN : 2248-9622, Vol. 4, Issue 11(Version 2), November 2014, pp.54-59

[6] Raheleh Hashemzehi, Reza Nourmandipour, Farokhkoroupi, "Congestion in Wireless Sensor Networks and Mechanisms for Controlling Congestion", Indian Journal of Computer Science and Engineering (IJCSE),Vol. 4 No.3 Jun-Jul 2013

[7] Attiuttama, Kanojia Sindhuben Babulal, "An Approach for Congestion Control in Wireless Network using Sliding Window" , International Journal of Innovative Research in Computer and Communication Engineering, ISSN(Online): 2320-9801, Vol. 3, Issue 10, October 2015, pp.9526-9532

[8] Prof. Sachin Patel Prof. Rakesh Pandit Mr. Abhijeet Rathod, "Various Techniques Use In Wireless Sensor Network For Congestion Control", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 3, March 2014, pp. 771-774

[9] Raheleh Hashemzehi, Reza Noormandipour, "Congestion of Technical Features of Transport Protocols for Wireless Sensor Networks", Greener Journal of Internet, Information and Communication Systems ISSN: 2354-2373 Vol.1 (1), January 2013, pp. 33-39

[10] (2015, October 15) Network Congestion[online] Available: https://en.wikipedia.org/wiki/Network_congestion_avoidance

[11] Arsh Arora, Lekha Bhambhu, "Performance Analysis of RED & Robust RED", International Journal of Computer Science Trends and Technology (IJCST), Volume 2 Issue 5, Sep-Oct 2014, pp. 51-55

[12] (2014, June 18) Weighted Random Early Detection [online] Available: https://en.wikipedia.org/wiki/Weighted _random_early_detection

[13] (2015, July 16) Random Early Detection [online] Available: https://en.wikipedia.org/wiki/Random_early_detection

[14] Gajendra S. Vyas, Vivek S. Deshpande, "Performance Analysis of Congestion in Wireless Sensor Networks", 3rd IEEE International Advance Computing Conference (IACC),Feb 2013.

[15] Vivek S. Deshpande, Pratibha P. Chavan, Vijay M. Wadhi, Jagdish B. Helonde, "Congestion Control in Wireless Sensor Networks by using Differed Reporting Rate", IEEE Conference 2012 World Congress on Information and Communication Technologies,2012, pp 209-213

[16] Sunitha G P , Dilip Kumar S M and Vijay Kumar B P " Classical and Soft Computing based Congestion Control Protocols in WSNs: A Survey and Comparison." International Journal of Computer Applications (0975-8887) Recent Advances in Information Technology, 2014

[17] Avhad Kalyani B. "Congestion Control in Wireless Sensor Network- A Survey", *IJCOT 2012.*

[18] Chella prabha, B. and S. Chenthur Pandian "A Multipath Energy Efficient Congestion Control Scheme for Wireless Sensor Network", Journal of Computer Science, 2012.

[19] Swastik Brahma, Maninak Chatterjee and Kevin Kwiat, "CCF: Congestion Control and Fairness in Wireless Sensor Networks", 8thIEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM Workshops), 2010, pp. 413-418.

[20] Jenolin Flora, D.F., Kavitha V., Muthuselvi M., "A Survey on Congestion Control Techniques in Wireless Sensor Networks", International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 ,pp.1146-1149

[21] Upasana Bhagat, Rutvij Joshi, Paras Gosai, "A Survey on Congestion for Wireless Sensor Networks", International Journal of Computer Science Trends and Technology (IJCST) – Volume 2 Issue 1, Jan-Feb 2014, pp.75-78

[22] Jizan Zhang, "Congestion Avoidance and Control Mechanism for Multi-paths Routing in WSN", International Conference on Computer Science and Software Engineering 2008, pp.1318-1322

[23] (2015, October 10) TCP congestion avoidance algorithm [online] Available:https://en.wikipedia.org/wiki/TCP_congestion-avoidance_algorithm

[24] Levis P, Patel N, Culler D, Shenker S , 2004, " Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: Proc. First Symposium Networked Sys. Design and Implementation (NSDI).

[25] http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

[26] http://www.tcl.tk/man/tcl/tutorial/tcltutorial.html

[27] K. Sohrabi, et al, (October 2000) "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications, Vol. 7, No. 5, pp. 16-27*.

[28] R. Min, et al., (January 2001), "Low Power Wireless Sensor Networks," *Internation Conference on VLSI Design, Bangalore, India.*

[29] J.M. Rabaey, et al., (July 2000), "PicoRadio supports ad hoc ultra low power wireless networking," *IEEE Computer, Vol. 33, pp. 42-48*.

[30] R. H. Katz, J. M. Kahn and K. S. J. Pister, (August 1999), "Mobile Networking for Smart Dust," *5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, WA*.

[31] I.F.Akyildiz,W.Su, Y.Sankarasubramaniam,E.Cayirci, (Dec 2002), "Wireless Sensor Networks:A survey", *Elsevier science B.V.*

[32] Shio Kumar Singh, M P Singh and D K Singh, (August 2010), "A Survey of Energy Efficient Hierarchical Cluster-Based Routing in wireless Sensor Network," *Int J. of Advanced Networking and Applications Volume:02, Issue:02, Pages:570-580.*

[33] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, (Jan 2000), "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. 33rd Hawaii Int'l. Conf. Sys. Sci*.

[34] Zuniga, M. Z.; Krishnamachari, B. (2002), "Integrating Future Large-Scale Wireless Sensor Networks with the Internet," - *Department of Electrical Engineering, UNiversity of Southern California*.

[35] Dorigo, M., Stützle, T.,(march 2004), "Ant Colony Optimization", *ISBN 0262042193, MIT Press*.

[36] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, (January 2000), "Energy-efficient communication protocol for wireless sensor networks", *Proceeding of the Hawaii International Conference System Sciences,* Hawaii.

[37] Sudarshan Vasudevan, Don Towsley, Dennis Goeckel, Ramin Khalili, (jan 2009)"Neighbor Discovery in Wireless Networks and the Coupon Collector's Problem", Ecole Polytechnique Fédérale de Lausanne (EPFL).

[38] Rajesh Krishnan, David Starobinski, (may 2004), "Efficient clustering algorithms for self-organizing wireless sensor networks", *ECE Department, Boston University, 8 Saint Mary's Street, Boston, MA 02215, USA.*

APPENDIX A

- **Installation of NS-2.35**

- Here, NS-2.35 version installed in Ubuntu 14.04.
- For installation of NS-2.35 version, a zip package of NS-2.35 is required in computer memory.
- There are some steps needs to be followed to install network simulator.
- Some prerequisites are needed for the installation of NS-2.35.
- NS-2.35 zip package folder should be placed in home folder.
- Latest update related with Ubuntu OS required to be installed.
- Command for updating the OS: sudo apt-get update.
- Before installing the NS we have to install some essential packages required by the NS. So run the following commands:

> "sudo apt-get install tcl8.5-dev tk8.5-dev"
>
> "sudo apt-get install build-essential autoconfautomake"
>
> "sudo apt-get install perlxgraphlibxt-dev libx11-dev libxmu-dev"

- Now, the zip folder is required to unzip using below command
  tar -xvzf /home/user_name/Documents/ns-allinone-2.35.tar.gz
- Make the changes as shown in figure below.

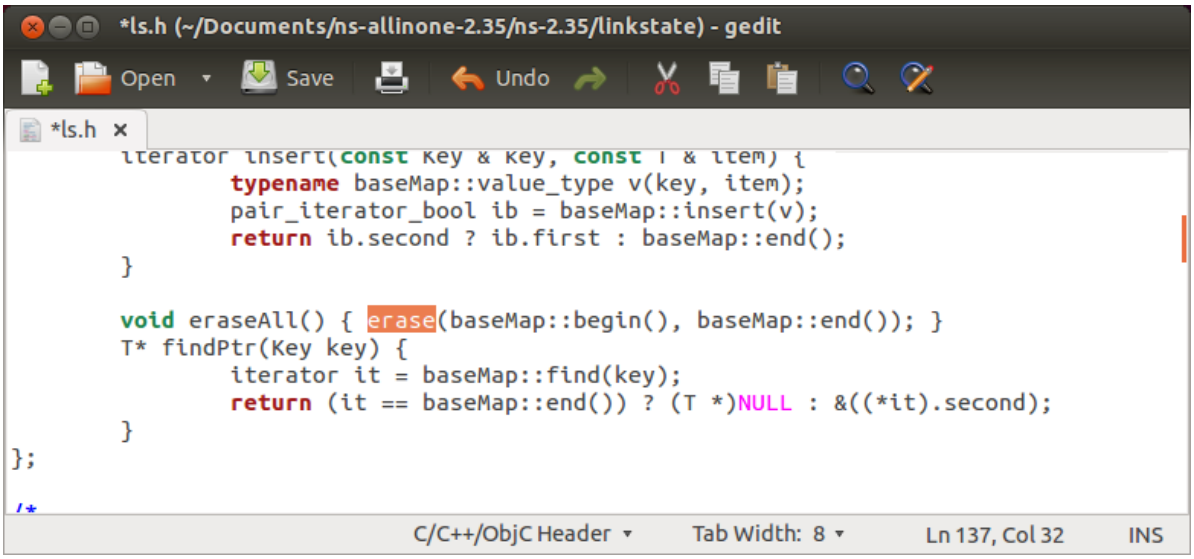

Fig. A Before Change

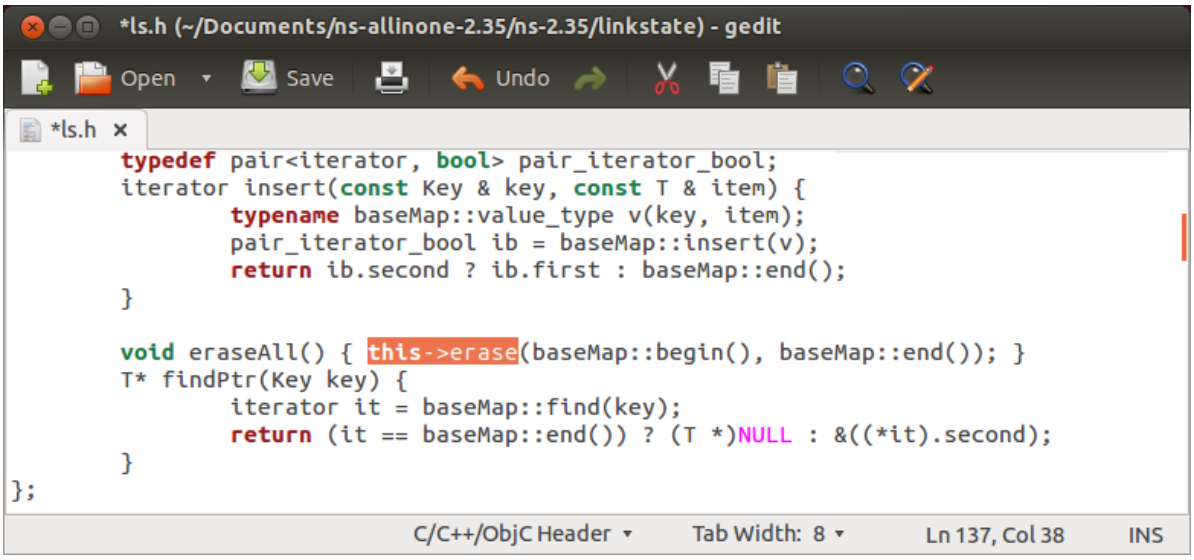

Fig. B Updating with Changes

- Now, Open the terminal and move to the directory where package have extracted.
- In terminal:

  $ cd /home/user_name/Documents/ns-allinone-2.35

  $ ./install
- Now set the environment variables.
- Assuming the previous steps gone well NS is installed. But now some environment variables needs to be added in .bashrc file.
- Open .bashrc file from terminal in home folder.
- Add the following lines at the end of the .bashrc file.

"# LD_LIBRARY_PATH"

"OTCLLIB=/path/to/ns-allinone-2.35/otcl-1.14"

"NS2=/path/to/ns-allinone-2.35/lib"

"USR_LocalLIB=/usr/local/lib"

"export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCLLIB:$NS2:$USR_LocalLIB"

"# TCL_LIBRARY"

"TCLIB=/path/to/ns-allinone-2.35/tk8.5.10/library"

"USRLIB=/usr/lib"

"export TCL_LIBRARY=$TCLIB:$USRLIB"

"# PATH"

"XGRAPH=/path/to/ns-allinone-2.35/xgraph-12.2/:/path/to/ns-allinone-2.35/bin:/path/to/ns-allinone-2.35/tcl8.5.10/unix:/path/to/ns-allinone-2.35/tk8.5.10/unix"

"NS=/path/to/ns-allinone-2.35/ns-2.35"

"NAM=/path/to/ns-allinone-2.35/nam-1.15"

"export PATH=$PATH:$XGRAPH:$NS:$NAM"

- Now, save the file and restart the system.
- To Validate the installation follow the given command below.

```
cd /home/user_name/Documents/ns-allinone-2.35/ns-2.35/
./validate
```

- To check whether NS-2 installed or not type ns in terminal.
- If '%' sign comes the NS-2 is successfully installed.