ABSTRACT

In Wireless Sensor Networks (WSN), Quality of Service (QoS) plays a significant role as networks performance is dependent on QoS only. WSN is very popular as it has wide application range. WSN is more cost effective for monitoring the different aspects of environment and industries. WSN has inadequate resources such as computational power and energy constraint. Congestion is one critical subject which has drawn attention of many researchers. Congestion results into reduced network performance and also drains the battery of the node, which is a limited resource in WSN. So congestion must be reduced to improve QoS and lifespan of a network. In this project, we present an effective approach for improving congestion in wireless sensor networks. This proposed algorithm may reduce the congestion and gives an effective solution. It establishes multiple paths from each sensor node to the cluster head and passes it to a ‘traffic node’ that manages the congestion and then sends it to base station. Traffic Node is intermediate node between cluster head and base station.

CHAPTER 1

INTRODUCTION

Wireless Sensor Networks (WSNs) are generally composed of one or more sinks and tens or thousands of sensor nodes scattered in a physical space. The purpose of congestion control is to improve the network throughput, reduce the time of data transmitted delay. In WSNs, Quality of Service (QoS) plays a significant role as networks performance is dependent on QoS only. WSN is very popular as it has wide application range. WSN is very cost effective for monitoring the different aspects of environment and industries. WSN has inadequate resources such as computational power and energy constraint. Congestion is one important subject which has drawn attention of many researchers. Congestion results into reduced less throughput and also drains the battery of the node, which is a limited resource in WSN. So, congestion is very important for QoS for a network. There are multiple cluster heads are available in WSN. Each of the cluster heads collects the data from respective cluster's nodes and forwards the aggregated data to base station. A major challenge in WSNs is to select appropriate cluster heads. We have used an alternative way to select the cluster heads in WSN as defined in Energy Aware Multipath Multihop Hierarchical (EAMMH) Protocol [1]. In a densely deployed network, the sink nodes are defined to forward the packets to respective cluster heads. In addition to that we will use 'Traffic Node(s)' 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.

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. 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.

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 routing path exists.
- The attacker must not be able to get the destination information by analyzing the traffic.
- The attacker must not be able to get the source location information if attacker is only able to monitor a certain area of the WSN and compromise a few sensor nodes.
- Only the destination node is able to identify the source position through the packet received.
- 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. Traffic node(s) collects all the data from the cluster-heads.
- Then traffic nodes forward the data to the base station.
- 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 implementation and chapter 5 deals with the testing and results. Finally, the project is concluded with details about conclusion and future work in chapter 6.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Congestion is one of the major issues which adversely affect all the QoS parameters. Congestion is considered to be occurred when any of the node or link gets the data more than its processing capacity [2] [3]. Congestion occurs due to high reporting rate, node density as large number of packets thumbed into the network. Congestion not only causes packet loss, but also leads to excessive energy consumption as well as delay. Therefore, in order to prolong network lifetime and improve fairness and provide better quality of service, developing a novel solution for congestion estimation and control is important to be considered. Congestion can be reduced by two means either by congestion detection or congestion avoidance. Congestion can be detected at node level by measuring buffer occupancy or at link level by measuring links busyness.

There are two types of congestion. 1. Node-Level Congestion 2. Link-Level Congestion. Congestion can be detected at node level by measuring buffer occupancy or at link level by measuring links busyness. In node-level congestion, the node-level congestion that is common in conventional networks. It is caused by buffer overflow in the node and can result in packet loss, and increased queuing delay. In Link-Level Congestion, a particular area has severe collisions could occur when multiple active sensor nodes within range of one another attempt to transmit at the same time. Packets that leave the buffer might fail to reach the next hop as a result of collision. This type of congestion decreases both link utilization and overall throughput, while increasing both packet delay and energy waste.

2.2 Routing Algorithms

Routing protocols in WSNs emphasize on data dissemination, limited battery power and bandwidth constraints in order to facilitate efficient working of the network, thereby increasing the lifetime of the network.

2.2.1 Congestion Control Protocols [2]

Congestion control concerns controlling traffic entry into a telecommunications network, so as to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. [2] It should not be confused with flow control, which prevents the sender from overwhelming the receiver. Two general approaches to control congestion are

1. Resource Management
2. Traffic control

2.2.1.1 Congestion Avoidance and Detection (CODA) [2]

CODA is energy efficient congestion control mechanism designed for WSNs. CODA, detects the congestion by observing the buffer size of sensor nodes and the load of the wireless channel. If these two characteristic exceed from a pre-defined threshold, a sensor node informs its neighbour to decrease the transmission rate [2]. Before transmitting a packet, a sensor node divides the channel to fixed periods. If it found the channel busier than pre-defined times, it adjusts a control bit to inform the base station of the congestion.

2.2.1.2 Congestion Control and Fairness (CCF) [2]

CCF detects congestion based on packet service time at MAC layer and control congestion based on hop-by hop manner with simple fairness [2]. CCF uses packets service time to deduce the available service rate and detect the congestion in each intermediate node. When the congestion is experienced, it informs the downstream nodes to reduce their data transmission rate and vice versa.

2.2.1.3 Adaptive Rate Control (ARC) [2]

ARC monitors the injection of packets into the traffic stream as well as route-through traffic [2]. Each node estimates the number of upstream nodes and the bandwidth is split proportionally between route-through and locally generated traffic, with preference given to the former. The resulting bandwidth allocated to each node is thus approximately fair.

Also, reduction in transmission rate of route-through traffic has a backpressure effect on upstream nodes, which in turn can reduce their transmission rates.

2.2.1.4 SenTCP [2]

SenTCP is an open-loop hop-by-hop congestion control protocol with two special features: 1) It jointly uses average local packet service time and average local packet inter-arrival time in order to estimate current local congestion degree in each intermediate sensor node. The use of packet arrival time and service time not only precisely calculates congestion degree, but effectively helps to differentiate the reason of packet loss occurrence in wireless environments, since arrival time (or service time) may become small (or large) if congestion occurs. 2) It uses hop-by-hop congestion control. In SenTCP, each intermediate sensor node will issues feedback signal backward and hop-by-hop. The feedback signal, which carries local congestion degree and the buffer occupancy ratio, is used for the neighboring sensor nodes to adjust their sending rate in the transport layer. The use of hop- by-hop feedback control can remove congestion quickly and reduce packet dropping, which in turn conserves energy [2]. SenTCP realizes higher throughput and good energy-efficiency since it obviously reduces packet dropping; however, SenTCP copes with only congestion and guarantees no reliability.

2.2.1.5 Fairness Aware Congestion Control (FACC) [2]

FACC is a congestion control mechanism, which controls the congestion and achieves fair bandwidth allocation for each flow of data. FACC detects the congestion based on packet drop rate at the sink node. In FACC nodes are divided in to two categories near sink node and near source node based on their location in WSNs. When a packet is lost, then the near sink nodes send a Warning Message (WM) to the near source node [2]. After receiving WM the near source nodes send a Control Message (CM) to the source node. The source nodes adjust their sending rate based on the current traffic on the channel and the current sending rate. After receiving CM, flow rate would be adjusted based on newly calculated sending rate.

2.2.1.6 Fusion [2]

In Fusion hop by hop flow control mechanism is used for congestion detection as well as congestion mitigation. Congestion is detected through queue occupancy and channel

sampling technique at each intermediate node [2]. Congestion notification (CN) bit will set in the header of every outgoing packet when the node detects congestion. Once the CN bit is set, neighboring node can overhear it and stop forwarding packet to the congested node.

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

PCCP is a congestion control mechanism based on node priority index that is introduced to reflect the importance of each sensor node. Nodes are assigned a priority based on the function they perform and its location. Nodes near the sink have a higher priority [2]. The congestion is detected based on the ratio of sending rate to the packet arrival rate. If the sending rate is lower, it implies that congestion has occurred. The congestion information is piggybacked in data packet header along with the priority index. Nodes adjust their sending rate depending on the congestion at the node itself. PCCP tries to reduce packet loss in congestion state while achieving the weighted fairness transmission for single-path and multipath routing.

2.2.1.8 Trickle [2]

Trickle, an algorithm for propagating and maintaining code updates in wireless sensor networks. Trickle's basic primitive is simple: every so often, a mote transmits code metadata if it has not heard a few other motes transmit the same thing. This allows Trickle to scale to thousand-fold variations in network density, quickly propagate updates, distribute transmission load evenly, be robust to transient disconnections, handle network repopulations, and impose maintenance overhead on the order of a few packets per hour per mote [2]. Trickle sends all messages to the local broadcast address. There are two possible results to a Trickle broadcast: either every mote that hears the message is up to date or a recipient detects the need for an update. Detection can be the result of either an out-of-date mote hearing someone has new code, or an updated mote hearing someone has old code. As long as every mote communicates somehow - either receives or transmits - the need for an update will be detected. For example, if mote A broadcasts that it has code φ, but B has code $\varphi+1$, then B knows that A needs an update. Similarly, if B broadcasts that it has $\varphi+1$, A knows that it needs an update. If B broadcasts updates, then all of its neighbors can receive them without having to advertise their need [2]. Some of these recipients might not even have heard A's transmission. Trickle uses "polite gossip" to exchange code metadata with nearby network neighbors. It breaks time into intervals, and at a random point in each

interval, it considers broadcasting its code metadata. If Trickle has already heard several other motes gossip the same metadata in this interval, it politely stays quiet: repeating what someone else has said is rude.

2.2.1.9 Siphon [2]

Siphon aims at controlling congestion as well as handling funneling effect. Funneling effect is where events generated under various work load moves quickly towards one or more sink nodes, which increases traffic at sink which leads to packet loss. Virtual sinks are randomly distributed across the sensor network which takes the traffic load off the already loaded sensor node. In siphon initially VS discovery is done. Virtual sink discovery is initiated by the physical sink by as explained in . Node initiated congestion detection is based on past and present channel condition and buffer occupancy as in CODA [2]. After congestion detection traffic is redirected from overloaded physical sink to virtual sinks. It is done by setting redirection bit in network layer header.

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

PHTCCP is an efficient congestion control protocol for handling diverse data with different priorities within a single node motivates. PHTCCP module works interacting with the MAC layer to perform congestion control function [2]. In this protocol, we focus on efficient mechanism so that congestion could be controlled by ensuring adjustment transmission rates for different type of data that generated by the sensors have various priorities. We assume that the sink node assigns individual priority for each type of sensed data and each node has n number of equal sized priority queues for n types of sensed data Heterogeneous applications can reflect the number of queues in a node. In congestion detection method , congestion level at each sensor node presented by packet service ratio.

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

2.2.2 Congestion Avoidance Protocols [2]

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottlenecks. Congestion avoidance is achieved through packet dropping. Among the more commonly used congestion avoidance

mechanisms is Random Early Detection (RED), which is optimum for high-speed transit networks.

2.2.2.1 Random Early Detection [2]

One solution is to use random early detection (RED) on the network equipment's port queue buffer. On network equipment ports with more than one queue buffer, weighted random early detection (WRED) could be used if available. RED indirectly signals to sender and receiver by deleting some packets, e.g. when the average queue buffer lengths are more than e.g. 50% (lower threshold) filled and deletes linearly more or (better according to paper) cubical more packets, up to e.g. 100% (higher threshold). The average queue buffer lengths are computed over 1 second at a time.

2.2.2.2 TCP/IP congestion avoidance [2]

The TCP congestion avoidance algorithm is the primary basis for congestion control in the Internet. Problems occur when many concurrent TCP flows are experiencing port queue buffer tail-drops. Then TCP's automatic congestion avoidance is not enough [2]. All flows that experience port queue buffer tail-drop will begin a TCP retrain at the same moment – this is called TCP global synchronization.

2.2.2.3 TCP Tahoe and Reno [2]

To avoid congestion collapse, TCP uses a multi-faceted congestion-control strategy. For each connection, TCP maintains a congestion window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control. TCP uses a mechanism called slow start to increase the congestion window after a connection is initialized and after a timeout. It starts with a window of two times the maximum segment size (MSS). Although the initial rate is low, the rate of increase is very rapid: for every packet acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every round-trip time (RTT) [2]. When the congestion window exceeds the ssthresh threshold, the algorithm enters a new state, called congestion avoidance. In some implementations (e.g., Linux), the initial ssthresh is large, and so the first slow start usually ends after a loss. However, ssthresh is updated at the end of each slow start, and will often affect subsequent slow starts.

As long as non-duplicate ACKs are received, the congestion window is additively increased by one MSS every round trip time [2]. When a packet is lost, the likelihood of duplicate ACKs being received is very high (it's possible though unlikely that the stream just underwent extreme packet reordering, which would also prompt duplicate ACKs). The behavior of Tahoe and Reno differ in how they detect and react to packet loss:

- Tahoe: Common Tahoe implementations detect congestion only by setting a timer for receiving a related ACK. Tahoe sets the slow start threshold to half of the current congestion window, reduces the congestion window to 1 MSS, and resets to slow-start state.
- Reno: If three duplicate ACKs are received (i.e., four ACKs acknowledging the same packet, which are not piggybacked on data, and do not change the receiver's advertised window), Reno will halve the congestion window (instead of setting it to 1 MSS like Tahoe), set the slow start threshold equal to the new congestion window, perform a fast retransmit, and enter a phase called Fast Recovery. If an ACK times out, slow start is used as it is with Tahoe.

The two main differences between Tahoe and Reno are:

1. Tahoe only uses a timeout for detecting congestion, while Reno uses timeout and Fast-Retransmit
2. Tahoe sets the congestion window to 1 after packet loss, while Reno sets it to half of the latest congestion window.

2.2.2.4 Robust random early detection (RRED) [2]

The Robust Random Early Detection (RRED) algorithm was proposed to improve the TCP throughput against denial-of-service (DoS) attacks, particularly low-rate denial-of-service (LDoS) attacks [2]. Experiments have confirmed that the existing RED-like algorithms are notably vulnerable under Low-rate Denial-of-Service (LDoS) attacks due to the oscillating TCP queue size caused by the attacks.

2.2.2.5 Weighted Random Early Detection (WRED) [2]

WRED is a queuing discipline for a network scheduler suited for congestion avoidance. It is an extension to random early detection (RED) where a single queue may have several different queue thresholds. Each queue threshold is associated to a particular traffic class. For example, a queue may have lower thresholds for lower priority packet. A queue buildup will cause the lower priority packets to be dropped, hence protecting the higher priority packets in the same queue [2]. In this way quality of service prioritization is made possible for important packets from a pool of packets using the same buffer. It is more likely that standard traffic will be dropped instead of higher prioritized traffic.

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

where n is the user-configurable exponential weight factor, o is the old average and c is the current queue length [2]. The previous average will be more important for high values of n. Peaks and Lows in queue length will be smoothed by a high value. Low values of n allow the average queue size to stay close to the current queue size.

2.2.2.6 Adaptive Random Early Detection (ARED) [2]

The adaptive RED or active RED (ARED) algorithm infers whether to make RED more or less aggressive based on the observation of the average queue length. If the average queue length oscillates around min threshold then early detection is too aggressive. On the other hand if the average queue length oscillates around max threshold then early detection is being too conservative [2]. The algorithm changes the probability according to how aggressively it senses it has been discarding traffic.

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

Low Energy Adaptive Clustering Hierarchy (LEACH) is a TDMA-based MAC protocol which is integrated with clustering and a simple routing protocol in wireless sensor networks (WSNs). The goal of LEACH is to lower the energy consumption required to create and maintain clusters in order to improve the life time of a wireless sensor network. LEACH is a hierarchical protocol in which most nodes transmit to cluster heads, and the cluster heads aggregate and compress the data and forward it to the base station (sink). Each node uses a stochastic algorithm at each round to determine whether it will become a cluster head in this round. LEACH assumes that each node has a radio powerful

enough to directly reach the base station or the nearest cluster head, but that using this radio at full power all the time would waste energy.

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

EAMMH protocol organizes the sensor nodes into clusters and forms a multihop intra-cluster network [1]. It establishes multiple paths from each sensor node to the cluster head and provides an energy aware heuristic function to choose the optimal path. WSN have unique characteristics such as denser level of node deployment, higher unreliability of sensor nodes and severe energy, computation and storage constraints which present many challenges in the development and application of WSN [1]. WSN typically contains hundreds or thousands of sensor nodes which allows for sensing over larger geographical regions with greater accuracy. Usually the sensor nodes are deployed randomly over geographical location and these nodes communicate with each other to form a network.

2.3 QoS Policies

2.3.1 Quality of Service (QoS)

Quality of Service (QoS) is the overall performance of a telephony or computer network, particularly the performance seen by the users of the network. To quantitatively measure quality of service, several related aspects of the network service are often considered, such as error rates, bit rate, throughput, transmission delay, availability, jitter, etc. In the field of computer networking and other packet-switched telecommunication networks, quality of service refers to traffic prioritization and resource reservation control mechanisms rather than the achieved service quality [3]. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.

2.3.1.1Throughput [3]

Throughput is a measure of the date rate (bits per second) generated by the application [3]. Equation shows the calculation for throughput TP, where Packet Size is the packet size of the ith packet reaching the destination, Packet Start is the time when the first packet left the source and Packet Arrival is the time when the last packet arrived.

TP = ™ ∑ Packet Size/ (Packet Arrival – Packet Start)

2.3.1.2 Delay [3]

Delay or latency would be time taken by the packets to transverse from the source to the destination [3]. The main sources of delay can be further categorized into: source-processing delay, propagation delay, network delay and destination processing delay.

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

2.3.1.3 Packet Delivery Ratio [3]

Packet Delivery Ratio means the number of successfully received packets to the total number of packets sent by sender (including re-transmissions).

PDR = Packet Sent/ Packet Arrival

2.3.1.4 Jitter [3]

Delay variation is the variation in the delay introduced by the components along the communication path [3]. It is the variation in the time between packets arriving.

Jitter= ((Packet Arrival+1) - (Packet Start+1))- ((Packet Arrival)-(Packet Start))/n-1

2.3.1.5 Energy [3]

The total number of energy consumed for packets transmitted and packet receiving during the simulation.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATIONS

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

The Non Functional requirements are as follows.

- 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 System Architecture

Data-centric network topologies are not suitable for large-scale sensor networks. Covering a large area without performance degradation is not possible with data-centric architecture. Moreover, in data-centric architectures, the reporting latency increases with the size of the network. The data-centric approach also causes significant power inefficiencies as the network grows. The network scalability issue is addressed in hierarchical routing. The hierarchical routing's main goal is to efficiently maintain network power consumption even in large-scale networks. In other words, hierarchical routing allows the network to scale in a number of sensor nodes. Most hierarchical architectures consist of sensor nodes grouped into cluster heads. Cluster heads build intra-cluster communication with other nodes within the same cluster, but they also build inter-cluster communicate with other cluster heads. Cluster heads aggregate data obtained individual sensors and then transfer the same information mostly in a multi-hop approach to the base. Figure 3.1 is showing the flow diagram of the proposed algorithm.

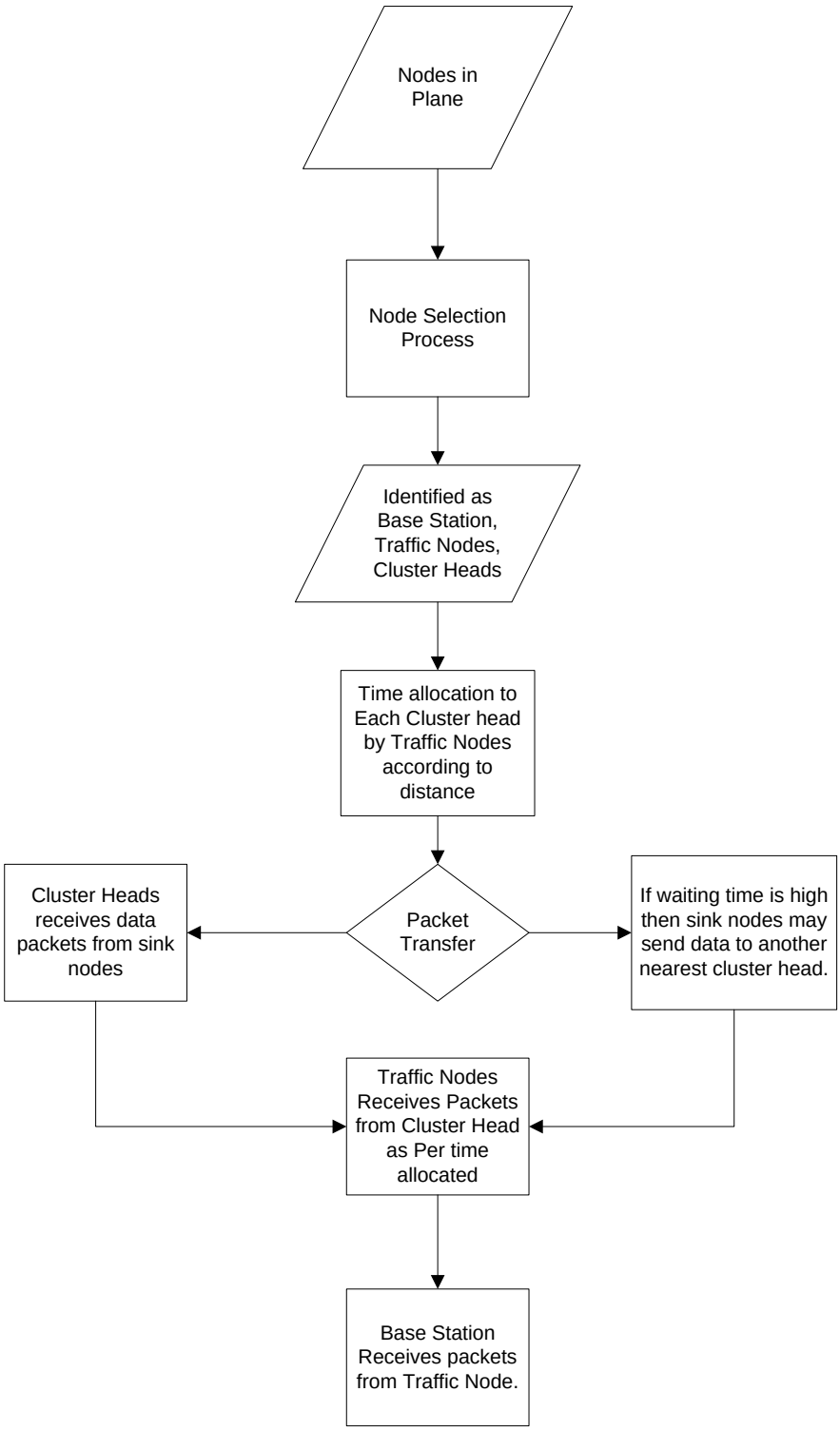

Figure 3.1 Flow Diagram

3.4 Diagrams

3.4.1 Use Case Diagrams

- The UML provides the use case diagram notation to illustrate 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.

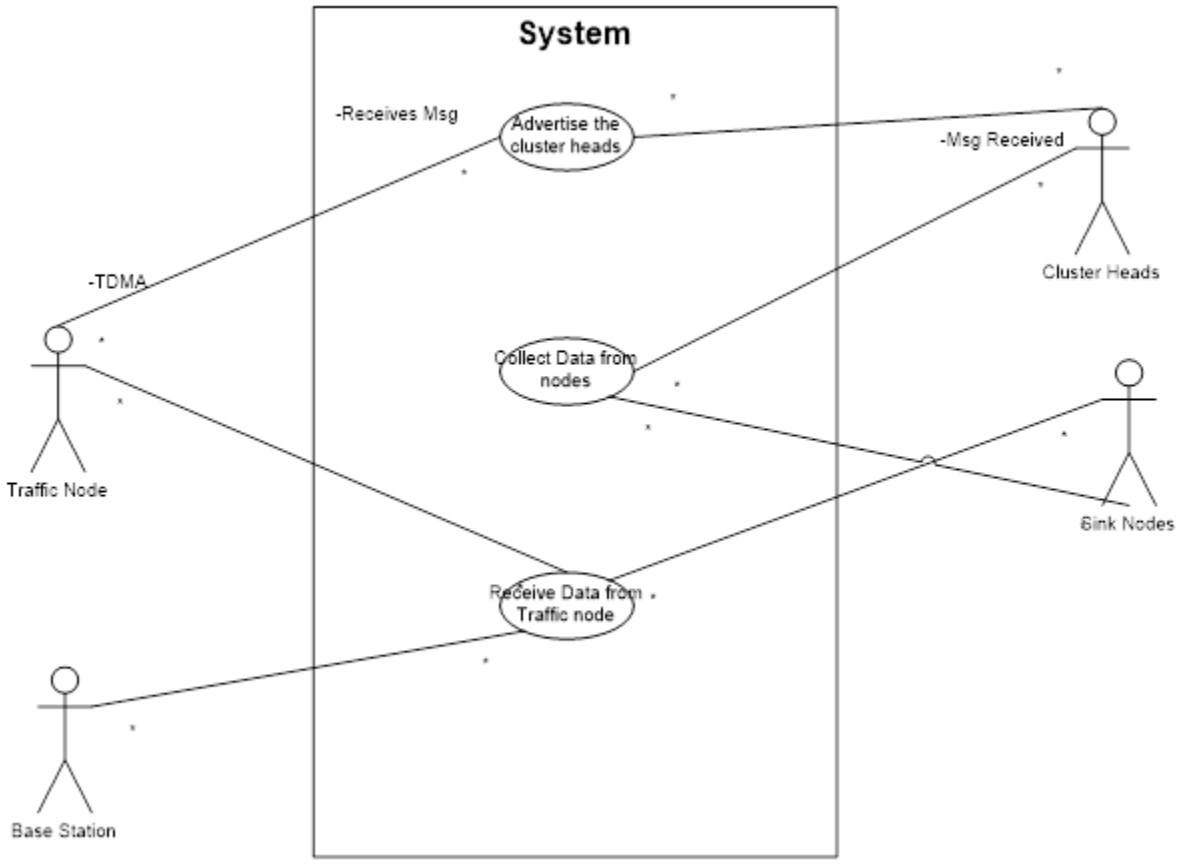

Figure 3.2 Use Case Diagram

3.4.2 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.

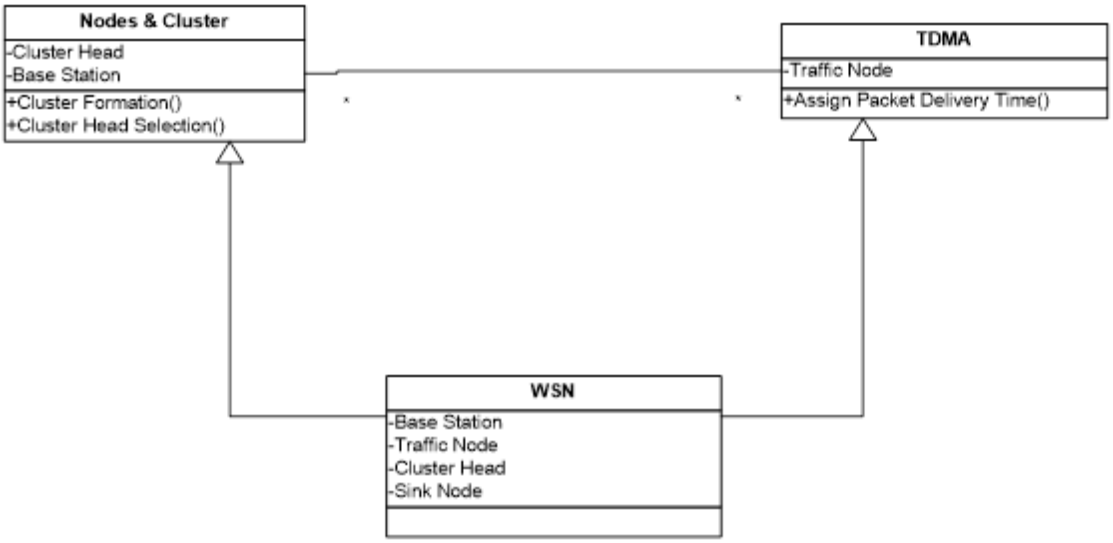

Figure 3.3 Class Diagram

3.4.3 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.

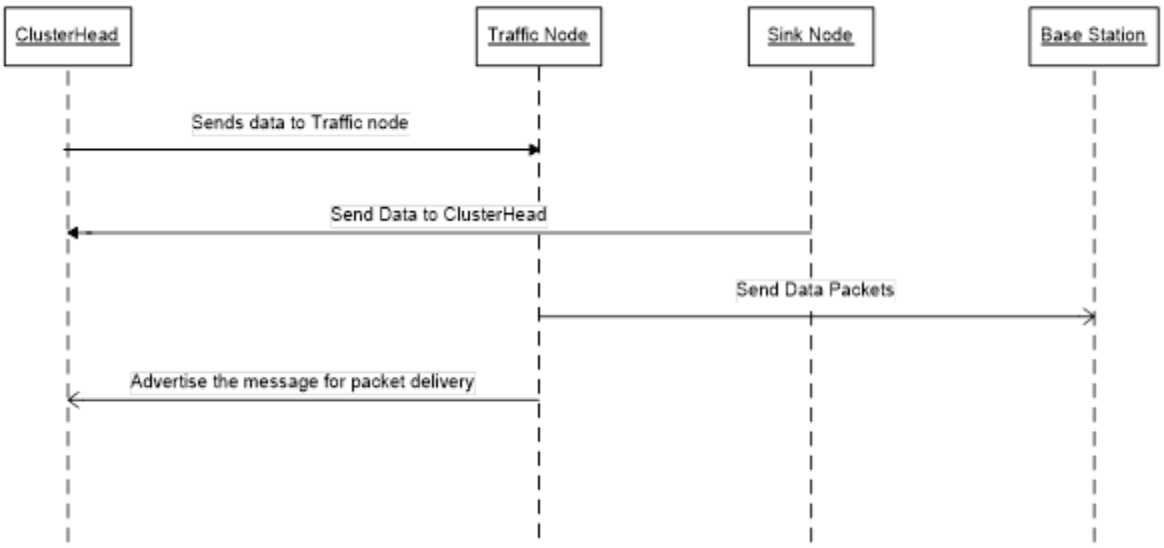

Figure 3.4 Sequence Diagram

3.4.4 Activity Diagram

- Activity diagram shows sequential 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.

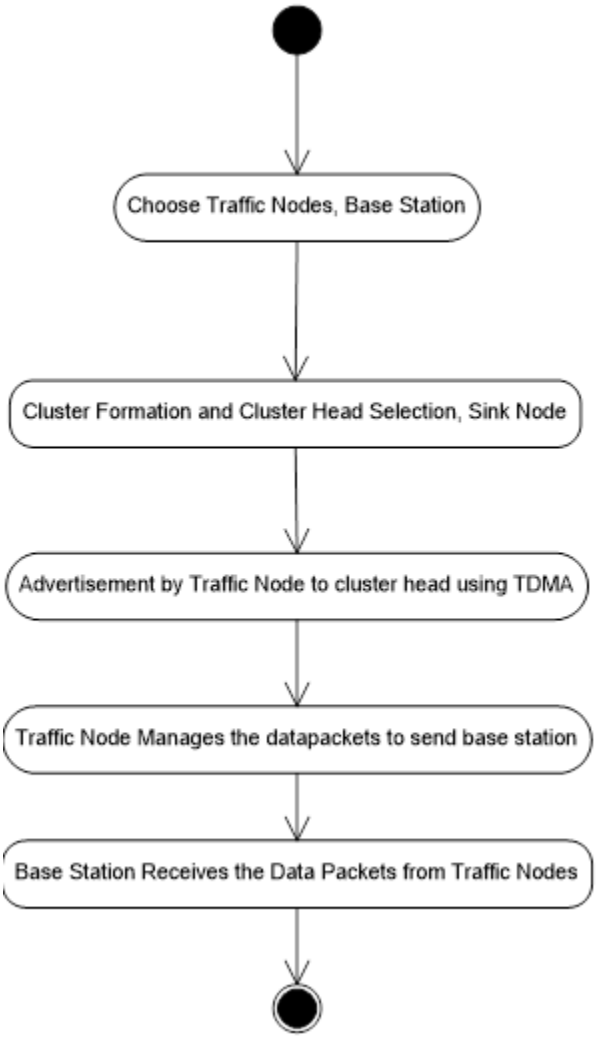

Figure 3.5 Activity Diagram

CHAPTER 4

ALGORITHM & IMPLEMENTATION

4.1 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 need to identified. (1) Sink Nodes, (2) Cluster Heads, (3) Traffic Nodes, (4) Base Station.

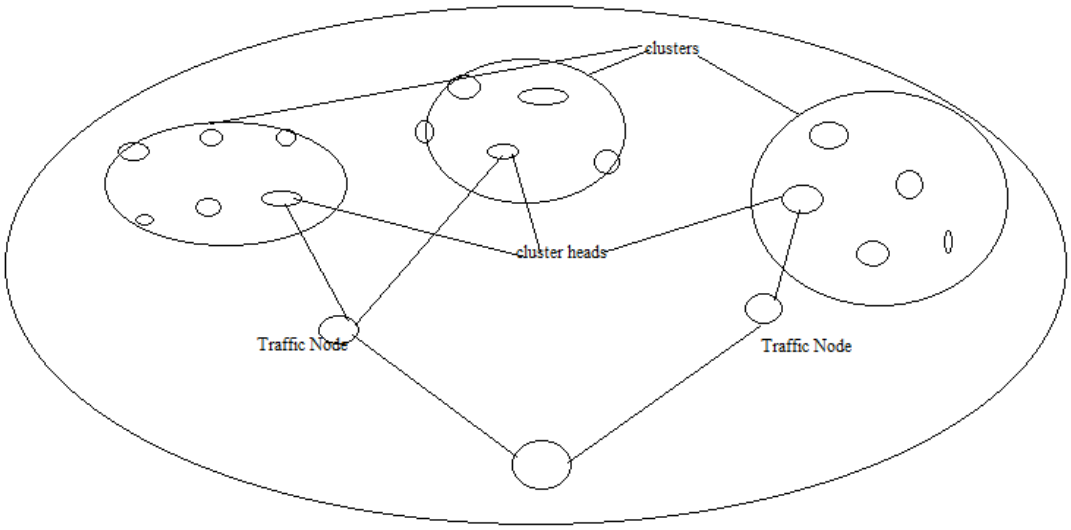

Figure 4.1 System Design

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.

4.2 Network Simulator – 2

- Network Simulator -2 formally known as NS-2 is very useful for the representation of algorithms. It provides support for simulation of such routing of MAC protocols for wired and wireless network.
- The Simulator core written in C++. It has an OTcl (Object Tool Command Language) interpreter which allows input code written in Tool Command Language (Tcl).
- It supports Tcl which has major elements developed in classes, like object oriented fashion.
- It is freely distributed. 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.

4.3 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.

CHAPTER 5

TESTING BY VARYING BUFFER SIZE

5.1 Test Case 1 - 30 Nodes

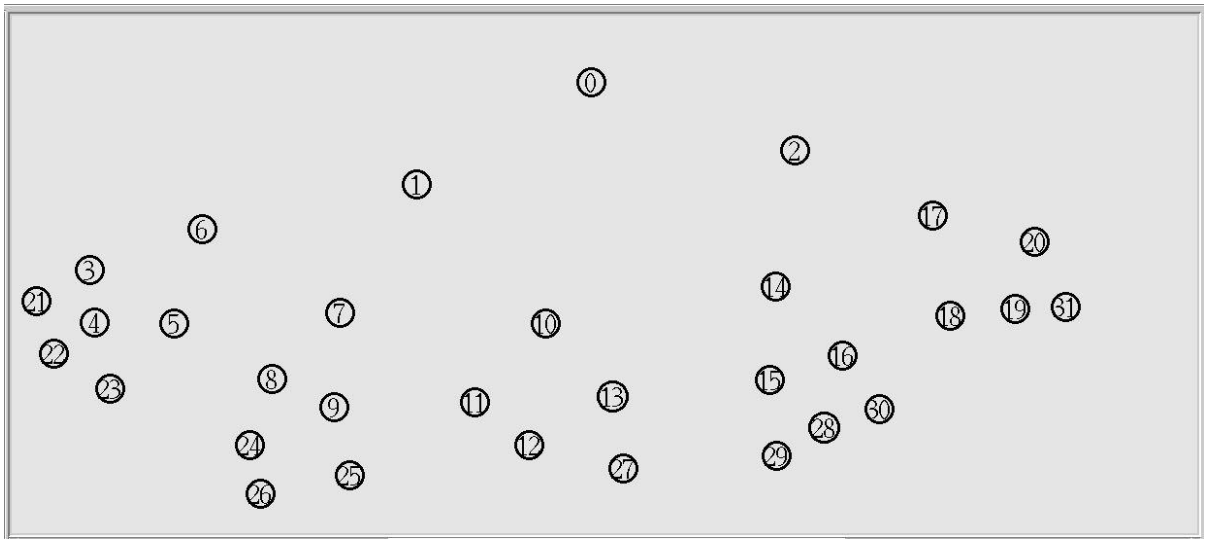

Figure 5.1 WSN with 30 Nodes

- 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.

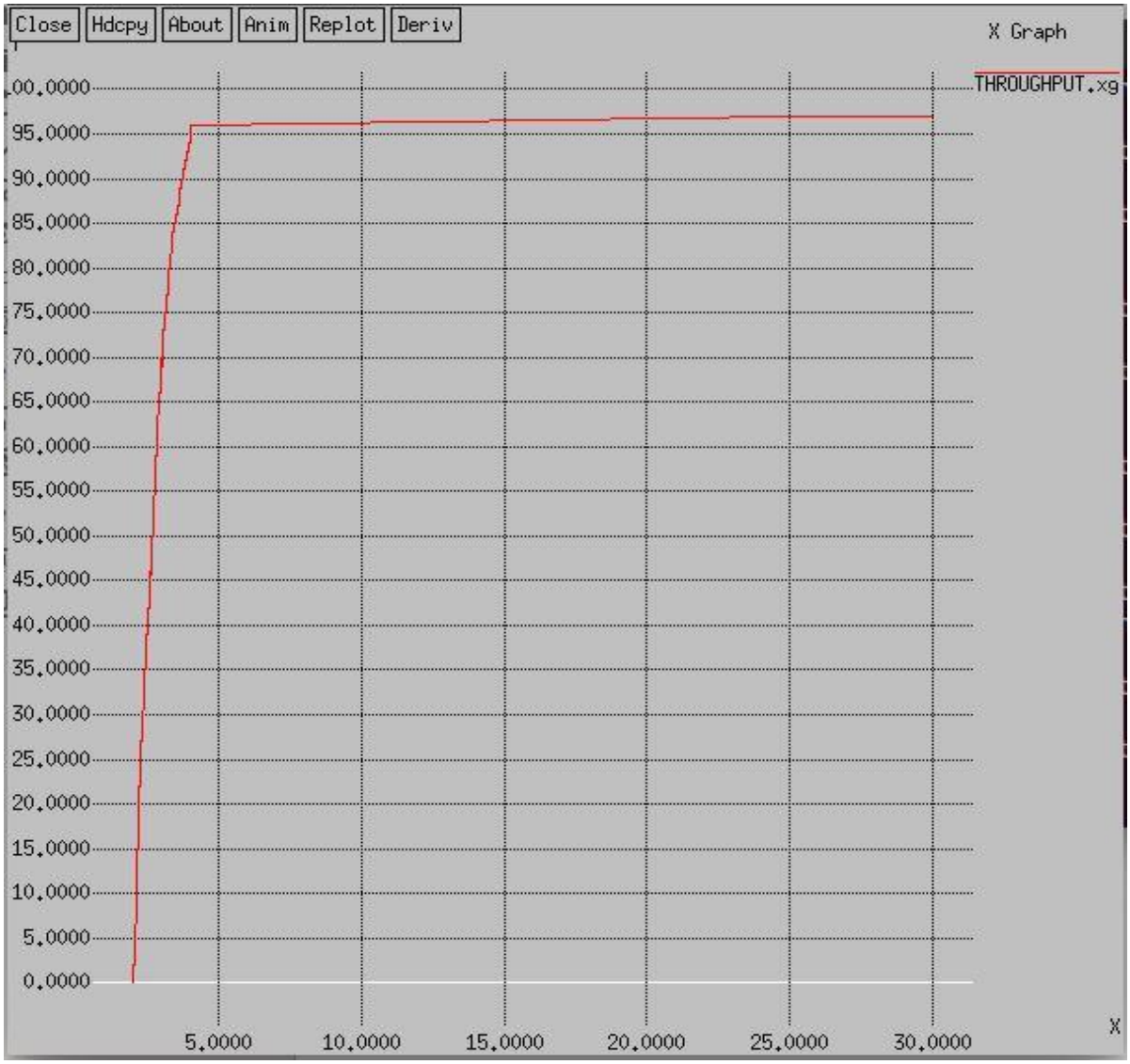

Figure 5.2 Throughput 30 Nodes

5.1.1 Throughput Details

- X- axis -> Nodes
- 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.

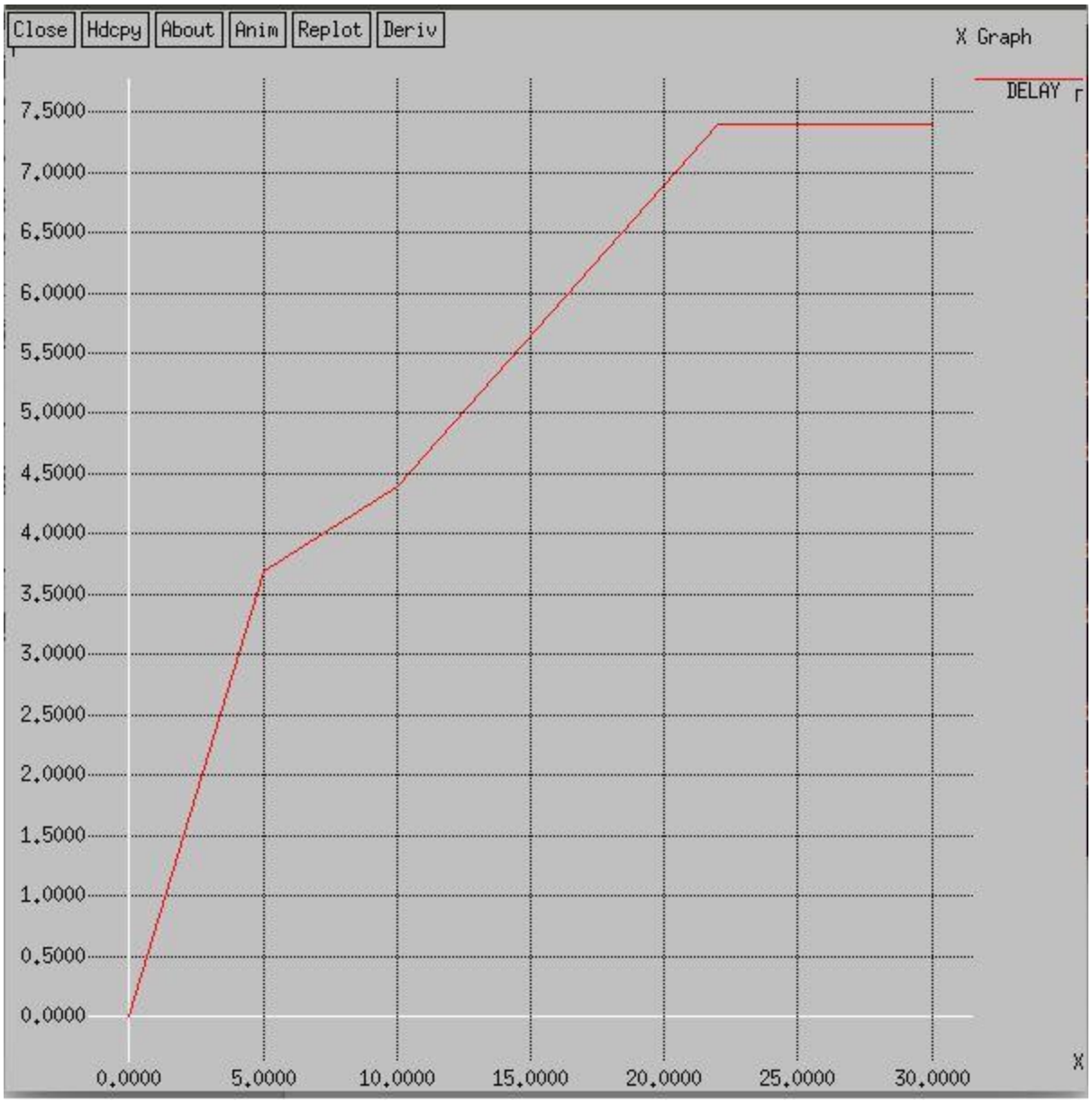

Figure 5.3 Delay 30 Nodes

5.1.2 Delay Details

- X-axis-> Nodes
- 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.

5.1.3 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.

5.2 Test Case 2 - 50 Nodes

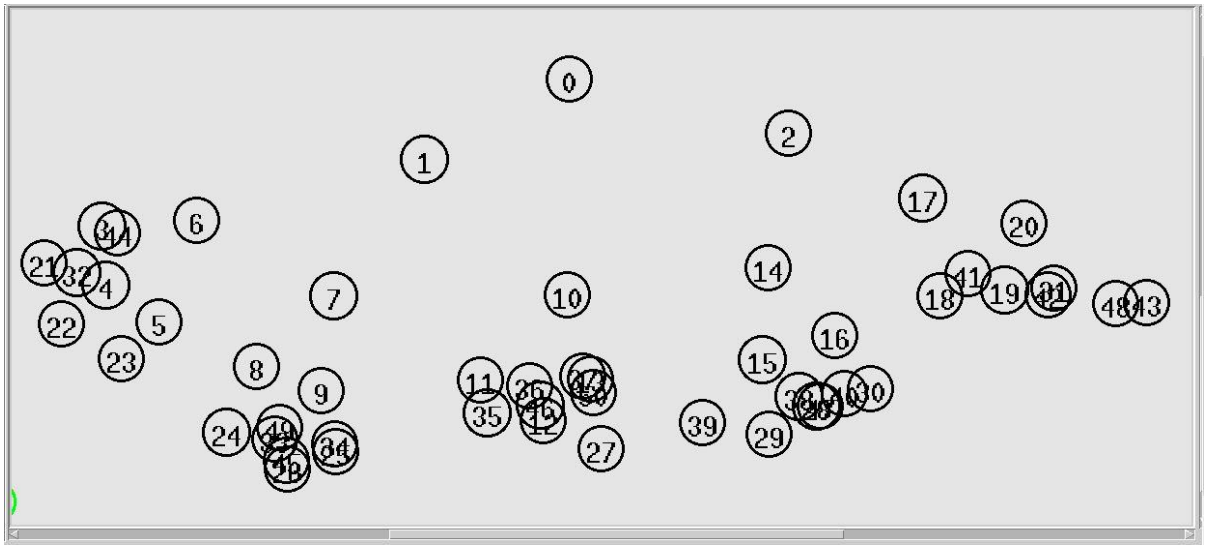

Figure 5.4 WSN with 50 Nodes

- 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.

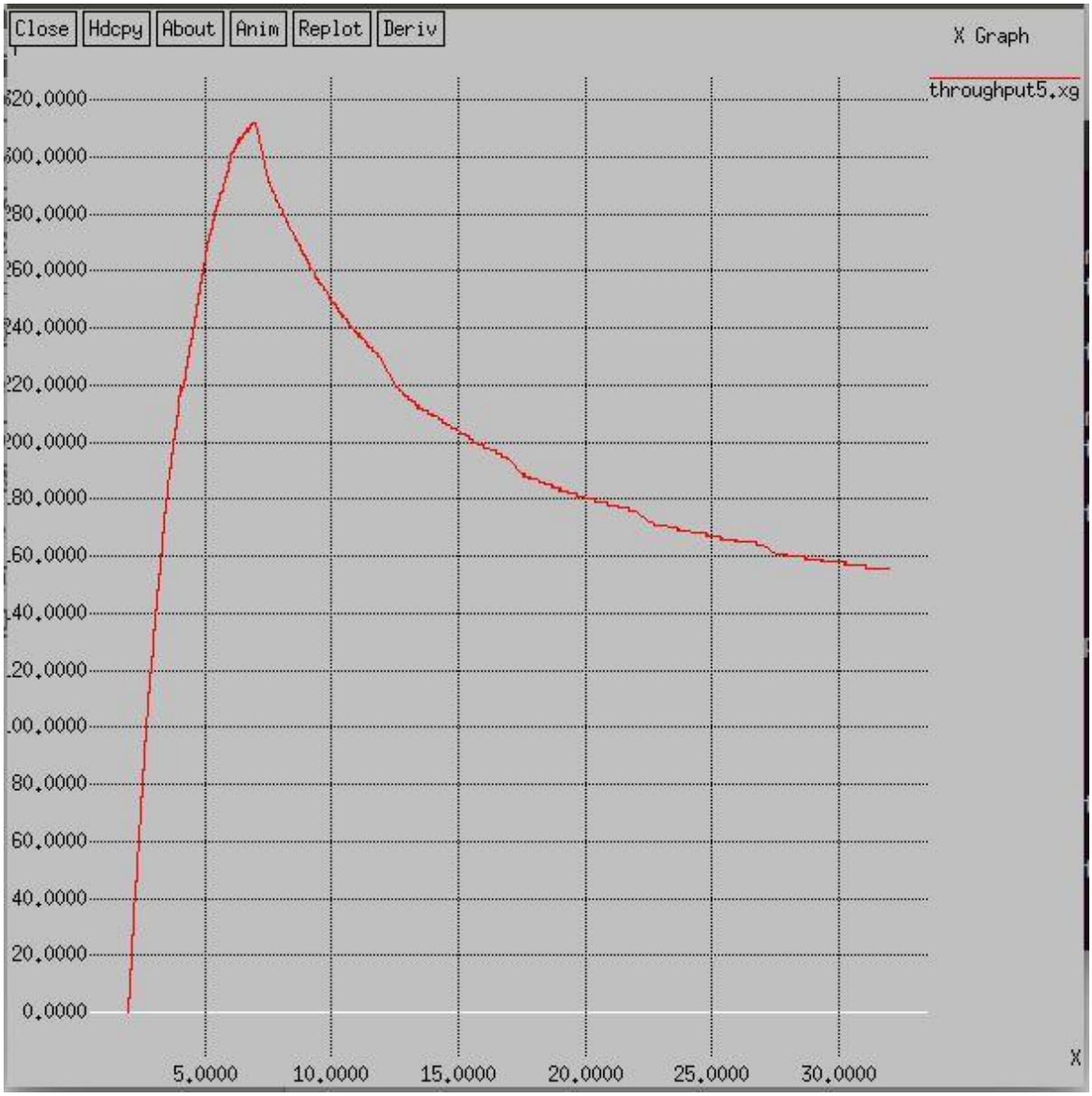

Figure 5.5 Throughput 50 Nodes

5.2.1 Throughput Details

- X-axis-> Nodes
- 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.

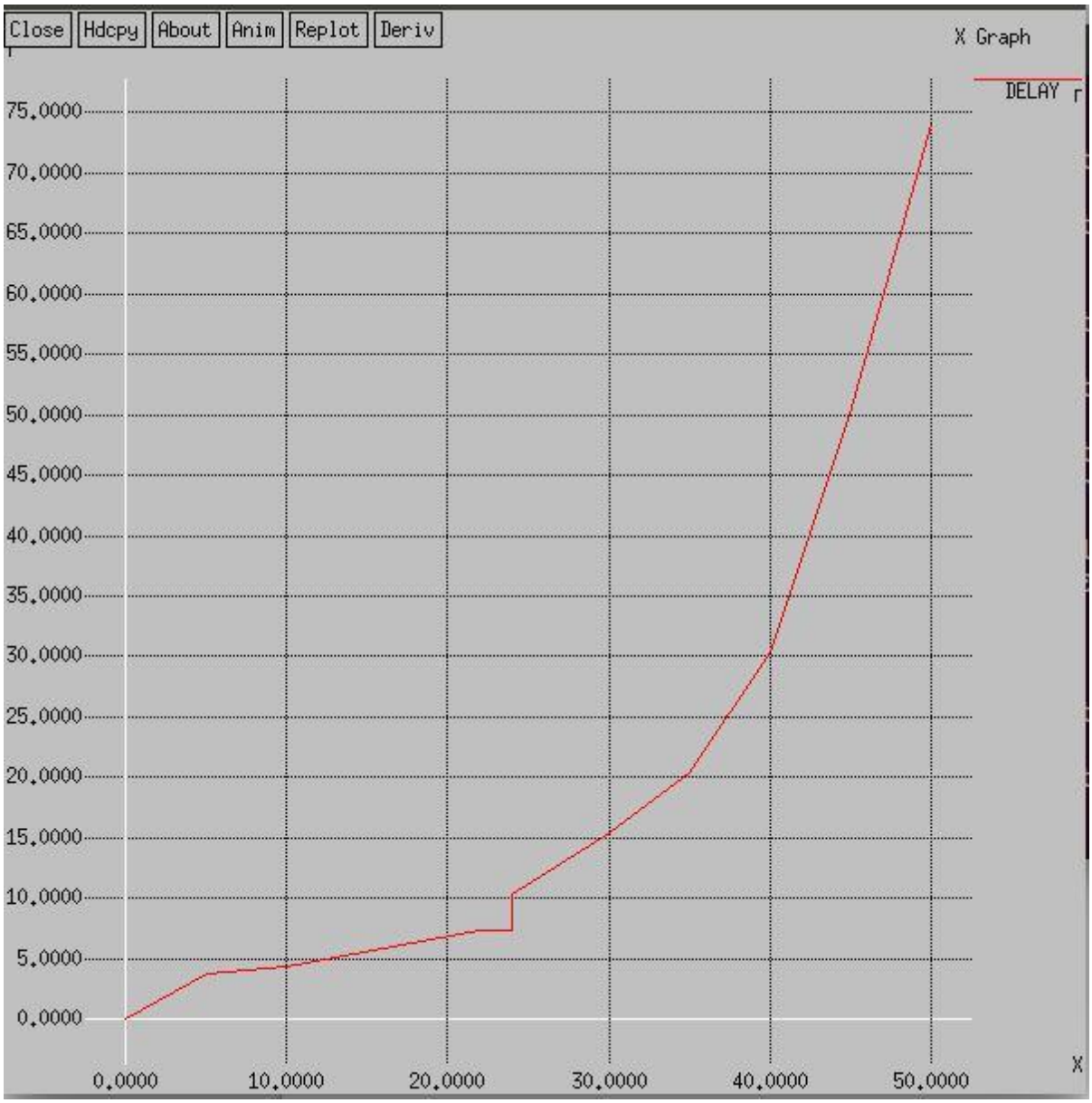

Figure 5.6 Delay 50 Nodes

5.2.2 Delay Details

- X-axis-> Nodes
- 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.

5.2.3 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.

5.3 Test Case 3 - 75 Nodes

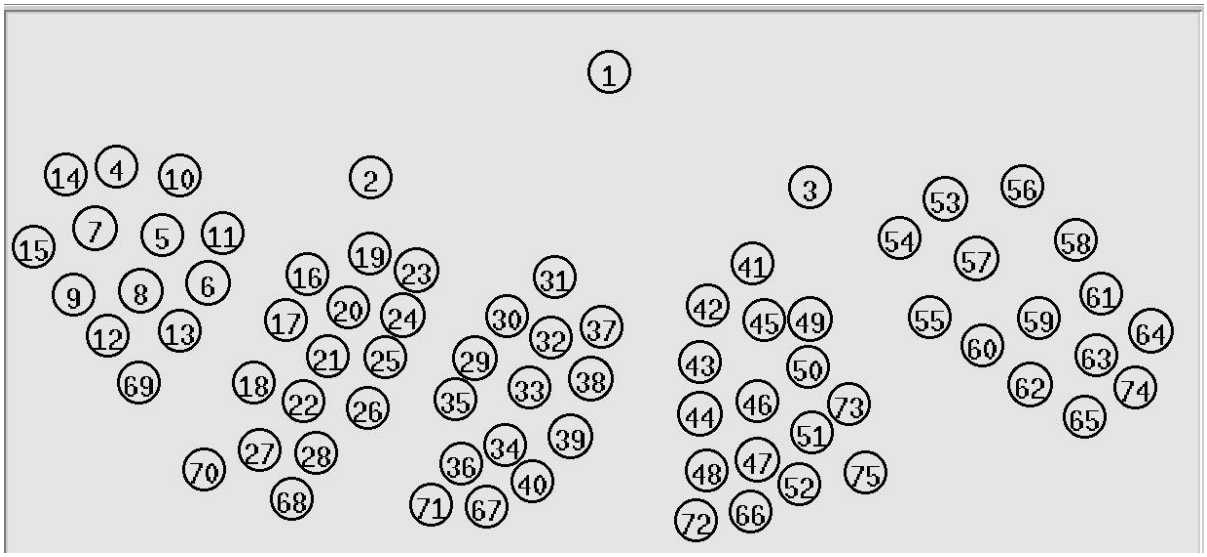

Figure 5.7 WSN with 75 Nodes

- 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.

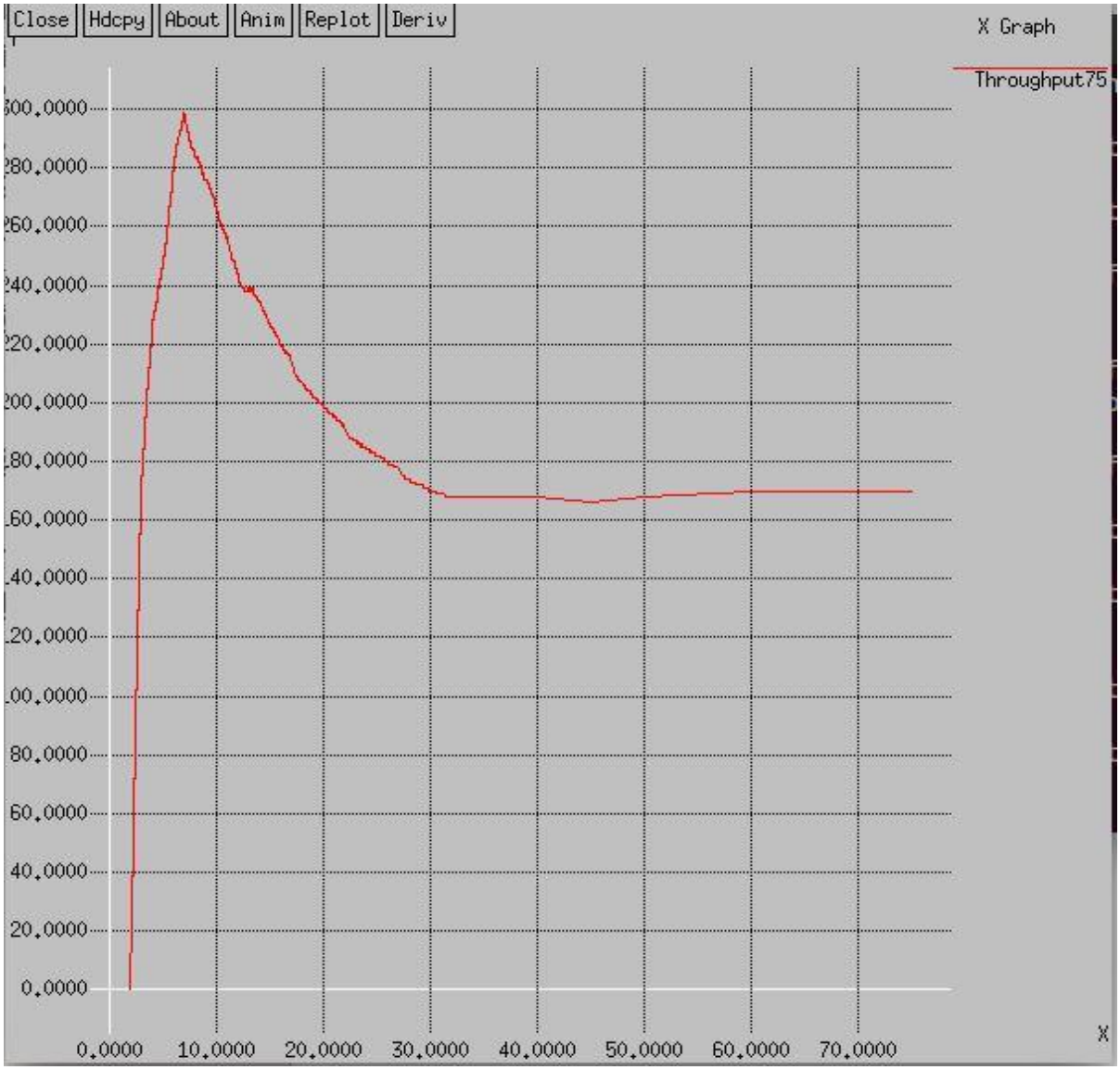

Figure 5.8 Throughput 75 Nodes

5.3.1 Throughput Details

- X-axis-> Nodes
- 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.

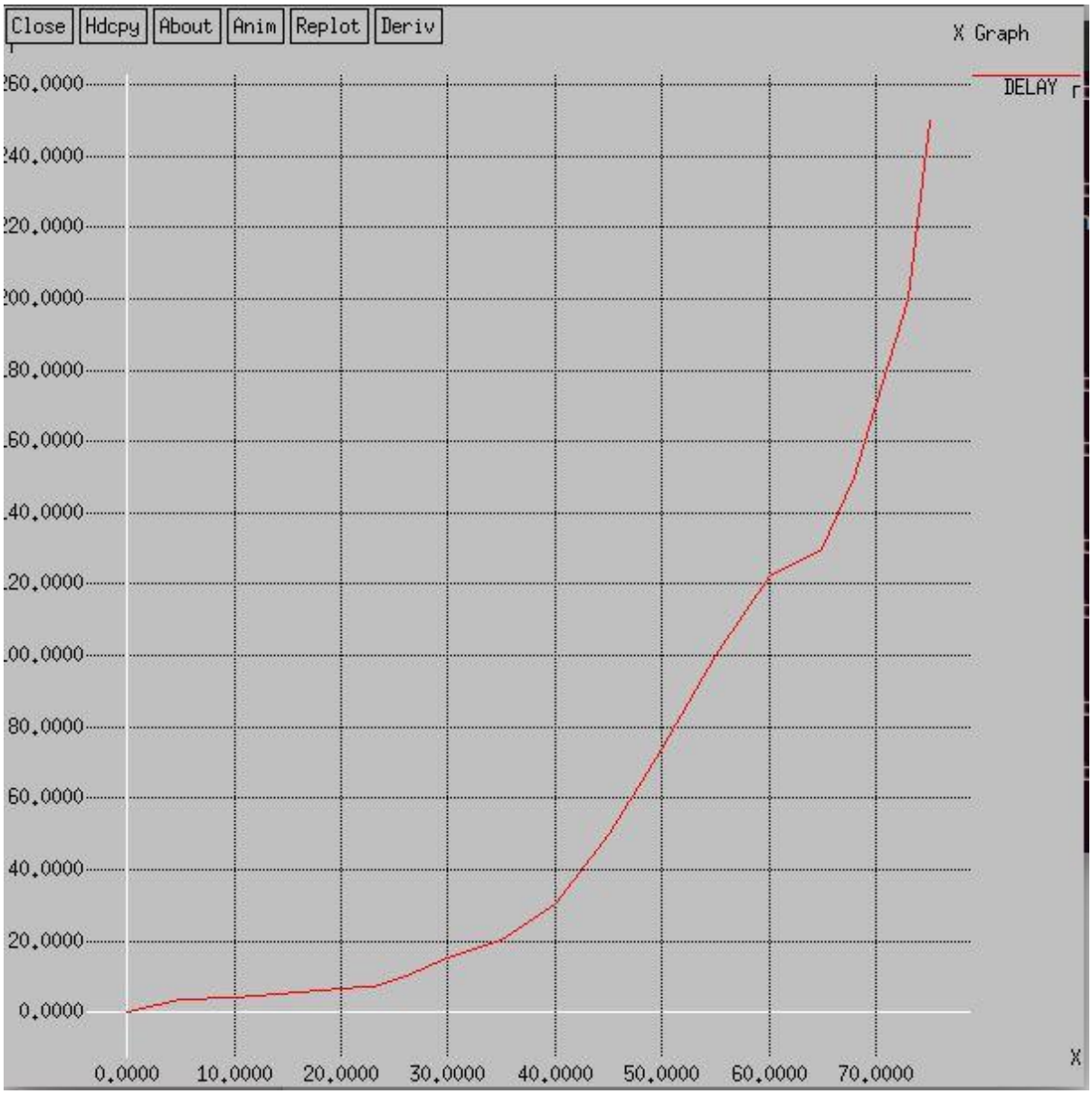

Figure 5.9 Delay 75 Nodes

5.3.2 Delay Details

- X-axis-> Nodes
- 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.

5.3.3 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.

5.4 Test Case Comparison with EAMMH

5.4.1 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 5.4 Comparison 30 Nodes

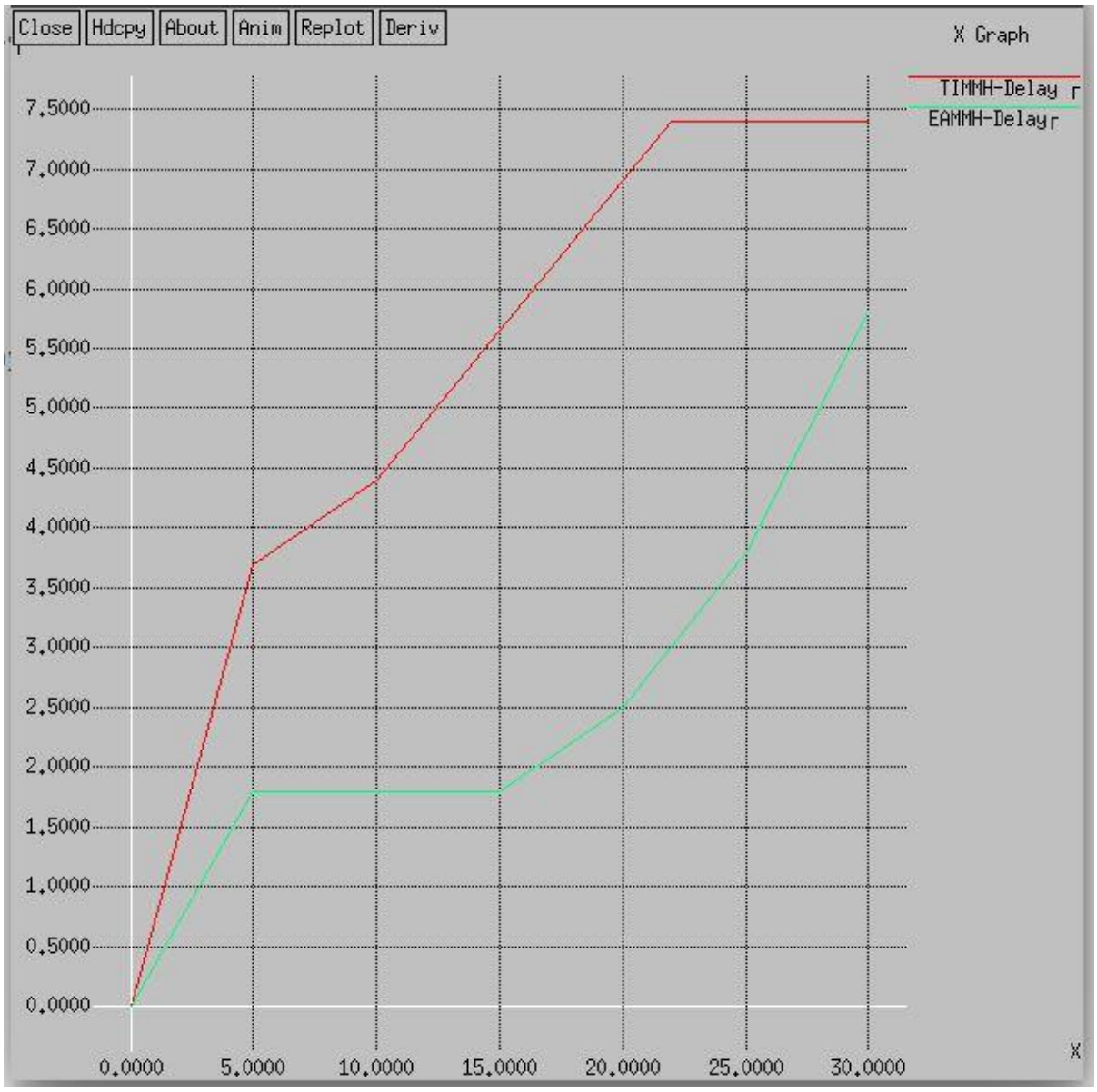

Figure 5.10 Comparison 30 Nodes Delay

5.4.1.1 Delay Comparison of 30 Nodes

- X-axis-> Nodes, Y-axis->Delay(ms)
- Here red line indicates TIMMH algorithm.
- The green line indicates EAMMH algorithm.
- 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.

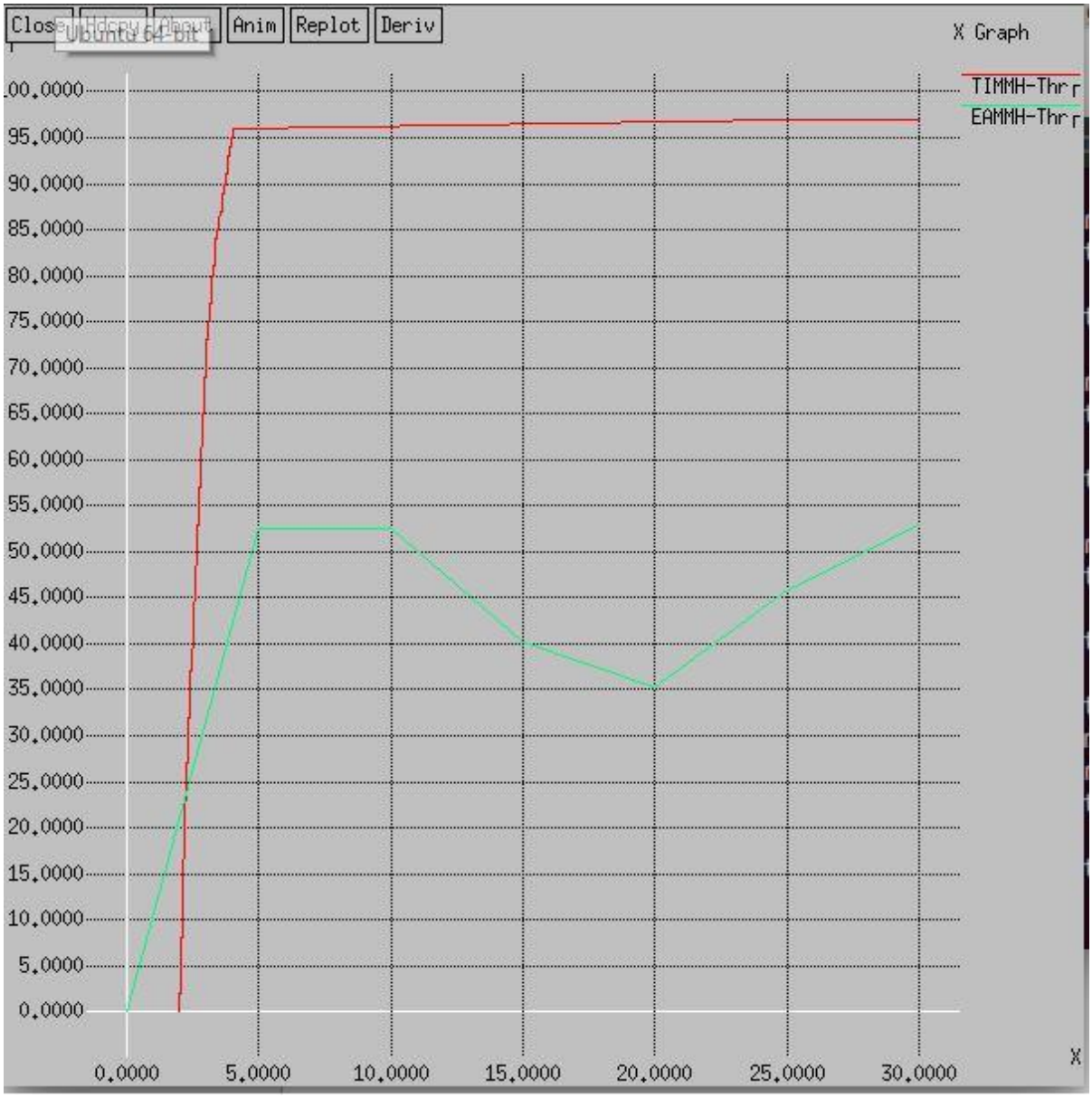

Figure 5.11 Comparison Throughput 30 Nodes

5.4.1.2 Throughput Comparison 30 Nodes

- X-axis-> Nodes, Y-axis-> Throughput(kbps)
- Red Line in graph shows throughput of TIMMH.
- Green Line in graph shows throughput of EAMMH.
- 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.

5.4.2 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 5.5 Comparison 50 Nodes

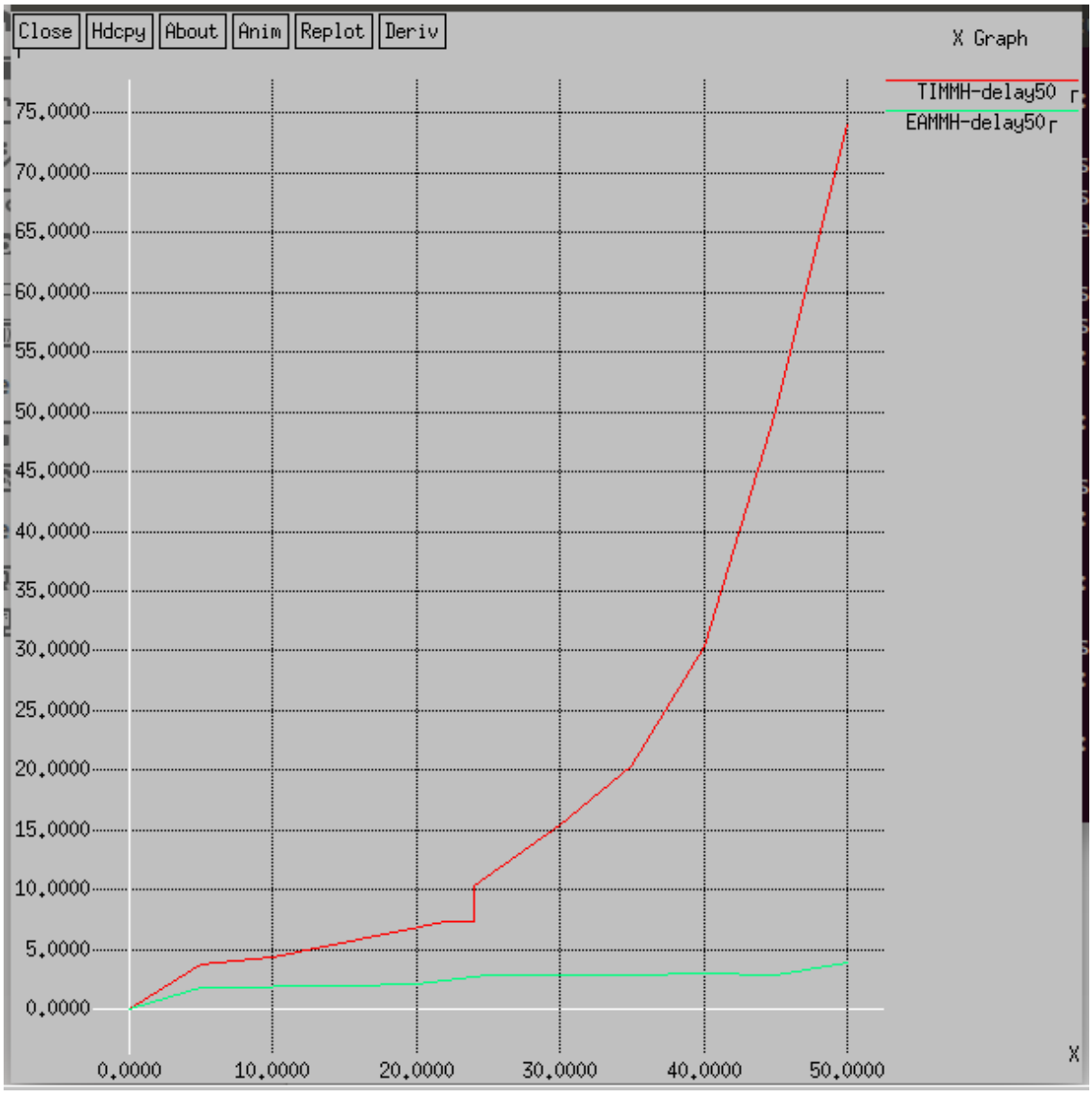

Figure 5.12 Comparison Delay 50 Nodes

5.4.2.1 Delay Comparison of 50 Nodes

- X-axis-> Nodes, Y-axis->Delay(ms)
- Here red line indicates TIMMH, Green line indicates EAMMH.
- Both indicate sequential increase in Delay.
- As node increases the delay has also increased.

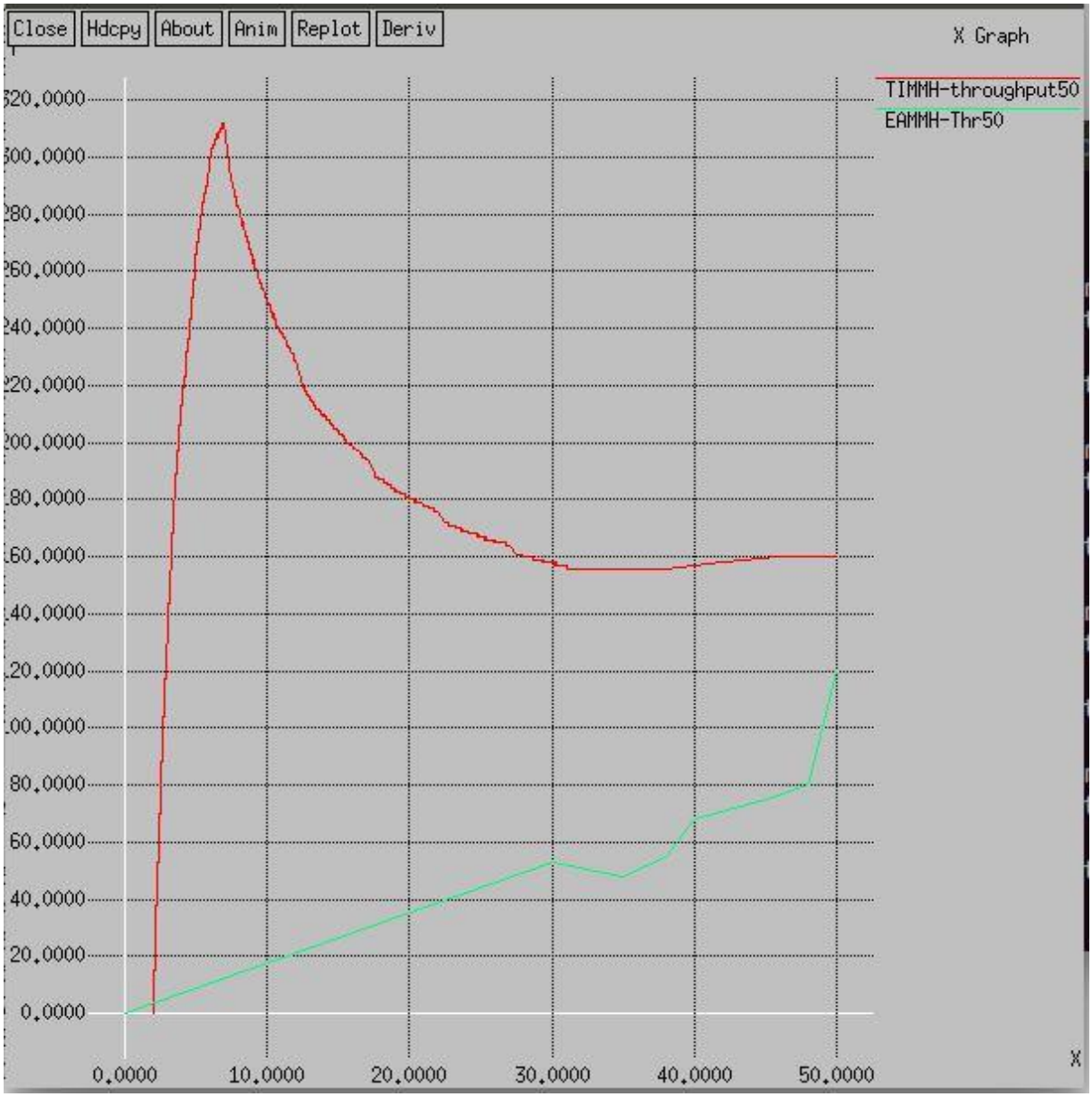

Figure 5.13 Comparison Throughput 50 Nodes

5.4.2.2 Throughput Comparison 50 Nodes

- X-axis-> Nodes, Y-axis-> Throughput(kbps)
- Red Line indicates TIMMH, Green indicates EAMMH.
- In TIMMH, the graph shows the throughput is getting low because of the simulation time as 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.

5.4.3 TIMMH 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 5.6 Comparison 75 Nodes

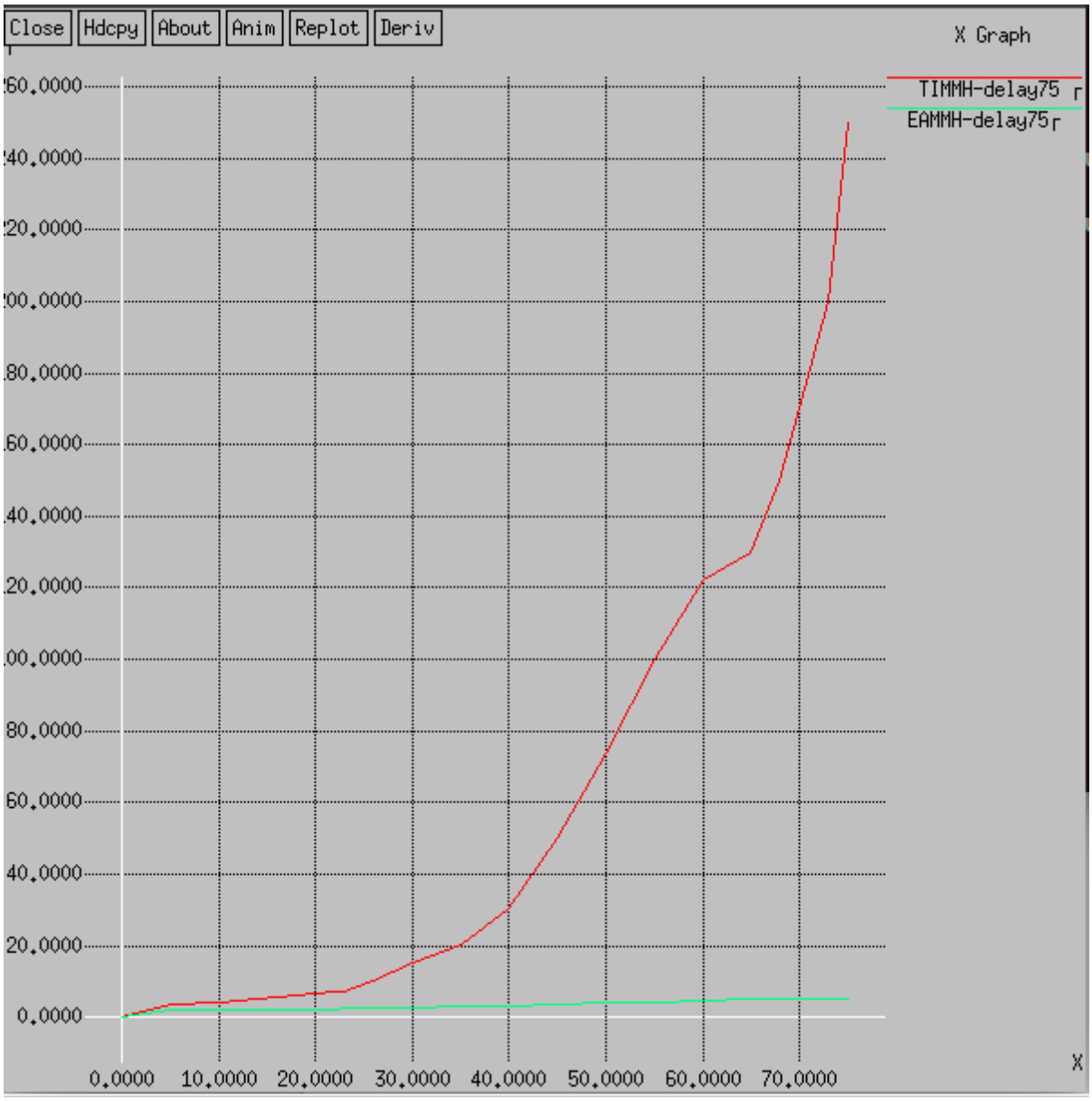

Figure 5.14 Comparison Delay 75 Nodes

5.4.2.1 Delay Comparison of 75 Nodes

- X-axis-> Nodes, Y-axis->Delay(ms)
- Red line indicates TIMMH and Green line indicates EAMMH.
- 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.

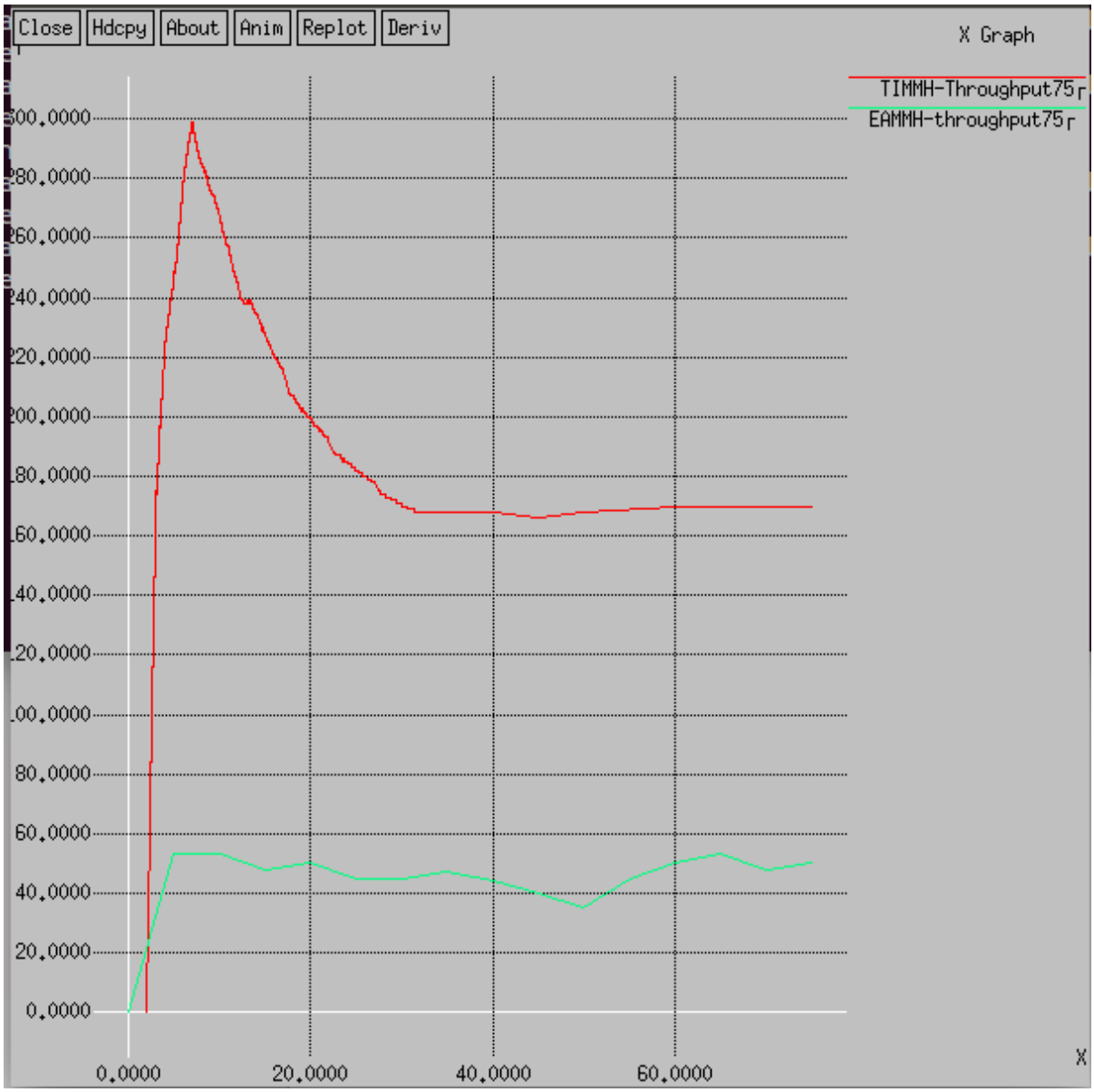

Figure 5.15 Comparison Throughput 75 Nodes

5.4.3.2 Throughput Comparison 75 Nodes

- X-axis-> Nodes, Y-axis-> Throughput(kbps)
- Red Line shows throughput of TIMMH and Green is for EAMMH.
- 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.

CHAPTER 6

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).