Kwame Nkrumah University of Science and Technology, Kumasi

COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ADAPTIVE CONGESTION CONTROL PROTOCOL IN WIRELESS SENSOR NETWORKS

BY

DELALI KWASI DAKE

MAY, 2013

ADAPTIVE CONGESTION CONTROL PROTOCOL IN WIRELESS SENSOR NETWORKS

By

Delali Kwasi Dake

(B.Sc. Computer Engineering)

A Thesis submitted to the Department of Computer Engineering

Kwame Nkrumah University of Science and Technology

In partial fulfillment of the requirements for the degree

of

MASTER OF PHILOSOPHY IN

COMPUTER ENGINEERING

College of Engineering

May, 2013

DECLARATION

I hereby declare that this thesis is my own work towards the MPhil. and that, to the best of my knowledge, it contains no material previously published by another person or material which has been accepted for the award of any other degree by the university or any other university, except where due acknowledgement has been made in the context.

Delali Kwasi Dake (PG5904811)	[illegible]	30/05/2013
(Student's Name & ID)	Signature	Date

Certified by:

Dr. James Dzisi Gadze	[illegible]	30/05/2013
(Supervisor)	Signature	Date
Dr. Kwame Osei Boateng	[illegible]	30/05/2013
(Head of Department)	Signature	Date

Acknowledgements

Firstly I wish to thank Computer Engineering Department of K.N.U.S.T for giving me the opportunity to pursue my Master's Programme. For during my stay, I had the opportunity to meet with some remarkable and ambitious people.

I would like to express my deepest gratitude to my supervisor Dr. James Dzisi Gadze, for his support, encouragement and taking time to review the thesis work. The 'rebuttal experience is really deep'. I would also like to thank my Head of Department, Dr. Kwame Osei Boateng for his academic advice. My appreciation also goes to my former Head of Department at University of Ghana, Dr. G.A Mills for your motivation and encouraging words. To my wireless sensor network group on face book, wow! Your research group is the best in the world. Thank you.

My friends Selorm Dovlo (UG), Selorm (KNUST), Danna (KNUST) and Comfort (UG), I say ayeekoo.

Mere words are insufficient to express my gratitude to my parents and siblings for their love and incredible support without which this would not have been possible.

Above all, I thank you Almighty God for your mercy and love, forever grateful.

Abstract

In Wireless Sensor Networks (WSN) when an event is detected there is an increase in data traffic that might lead to network instability and sensor nodes close to the burst traffic region enters crisis state and become congested. The WSN experience a decrease in network performance due to packet loss, long delays, and reduction in throughput. In the project, we developed an Adaptive Congestion Control Protocol (ACCP) that monitors network utilization and adjust traffic levels and/or increases network resources to improve throughput and conserve energy. The traffic congestion control protocol DelStatic is developed by introducing backpressure mechanism into NOAH. We analyzed various routing protocols and established that DSR has a higher resource congestion control capability. The proposed protocol, ACCP uses a sink switching algorithm to trigger DelStatic or DSR feedback to a congested node based on its Node Rank. From the simulation results, ACCP does not only improve throughput but also conserves energy which is critical to sensor application survivability on the field. Our Adaptive Congestion Control Protocol achieved reliability, high throughput and energy efficiency.

Table of Contents

List of Figures

List of Tables

List of Acronyms

WSN	Wireless Sensor Network
OPEC	Organization of Petroleum Exporting Countries
VOCs	Volatile Organic Compounds
ACCP	Adaptive Congestion Control Protocol
DSR	Dynamic Source Routing
TADD	TORA AODV DSR DSDV
NOAH	NO Ad-Hoc Routing
CODA	Congestion Detection and Avoidance
AIMD	Additive Increase / Multiple Decrease
ACK	Acknowledgement
ESRT	Event to Sink Reliable Transport
TCP	Transmission Control Protocol
DPCC	Decentralized Predictive Congestion Control
DPC	Distributed Power Control
HCCP	Hybrid Congestion Control Protocol
CSMA	Carrier Sensing Multiple Access
RTS	Request-to-Send
CTS	Clear-to-Send
DSP	Digital Signal Processing
NR	Node Rank
CB	Congestion Bit
RAF	Rate Adjustment Feedback

SIFS	Short Inter Frame Space
Bsize	Buffer Size
HC	Hop Count
NS-2	Network Simulator 2
TCL	Tool Command Language
NAM	Network Animator
AODV	Ad hoc On-demand Vector Protocol
TORA	Temporarily Ordered Routing Algorithm
DSDV	Destination-Sequenced Distance Vector
ARP	Address Resolution Protocol
CBR	Constant Bandwidth Rate
IMEP	Internet MANET Encapsulation
MAC	Media Access Control

1 Introduction

1.1 Introduction

The emerging field of wireless sensor technologies and remote application development has made wireless communication a ubiquitous means for transporting information across many different domains. Within the framework of Wireless Sensor Networks (WSNs), there are many application areas where sensor networks are deployed for environmental monitoring, battlefield surveillance, health and industrial monitoring control.

Ghana recently joined the Organization of Petroleum Exporting Countries (OPEC) and as at the first quarter of 2012 produced 63,100 barrels per day [1]. Though this is a boost to the Country's economic fortunes, the industry is the largest industrial source emissions of Volatile Organic Compounds (VOCs); VOCs are the group of chemicals that contribute to the formation of ground level ozone (smog) [2]. Exposure to ozone is linked to a wide range of health effects including aggravated asthma, increased emergency room visits, hospital admission and premature death. The future challenge for Ghana is to deploy a suitable network for measuring, forecasting and checking the quality of air. This essential_VOCs monitoring can be done with wireless sensor technology due to its remote deployment capabilities.

Wireless Sensor Networks consist of spatially distributed autonomous sensors called motes to monitor physical or environmental conditions.

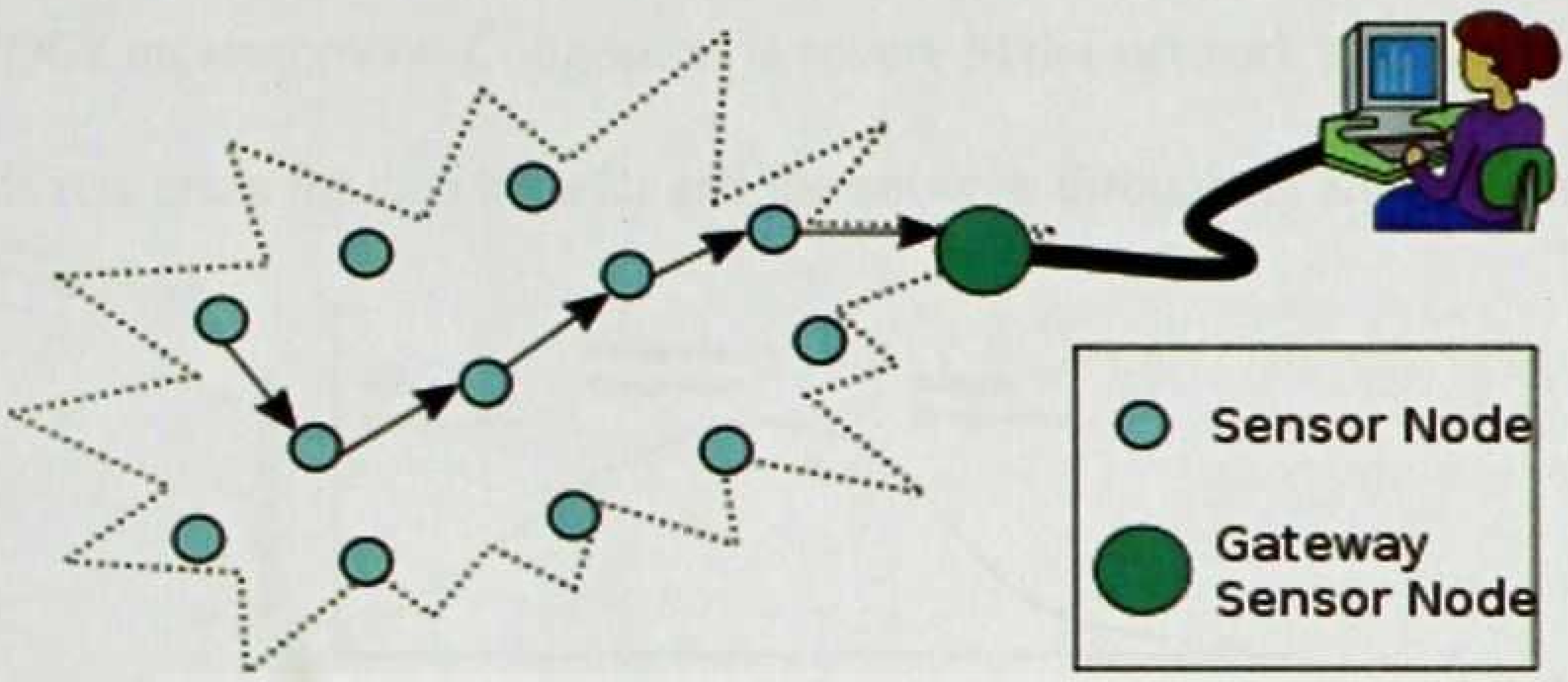


Figure 1.1 a remote monitoring of sensor application.

A typical scenario as shown in Figure 1.1 is a Wireless Applications Engineer using distributed sensor network to monitor pollution from the oil field to a gateway computer.

The graph [3] in Figure1.2 shows the relationship between the barrels of oil produced (load) and the measured VOCs (throughput). When the network load was small, the graph showed a linear relationship between network throughput and load. This implies that before region A, VOCs measurement from the oil field was ideal and the amount of emitted gas measured by the gateway computers amounted to pollution from the oil field. This is a no congestion scenario. As production increases, VOCs emission also increases. To record the extra VOCs emission, the sensor networks needs a higher sending rate. During this period, region A-B, there is moderate congestion due to limited bandwidth resources available for extra data transfer.

When the network load goes over region B, even a higher data sending rate is required for ideal VOCs measurement. Congestion is severe in the network at this region due to limited node resources for data transfer and the network throughput sharply decline.

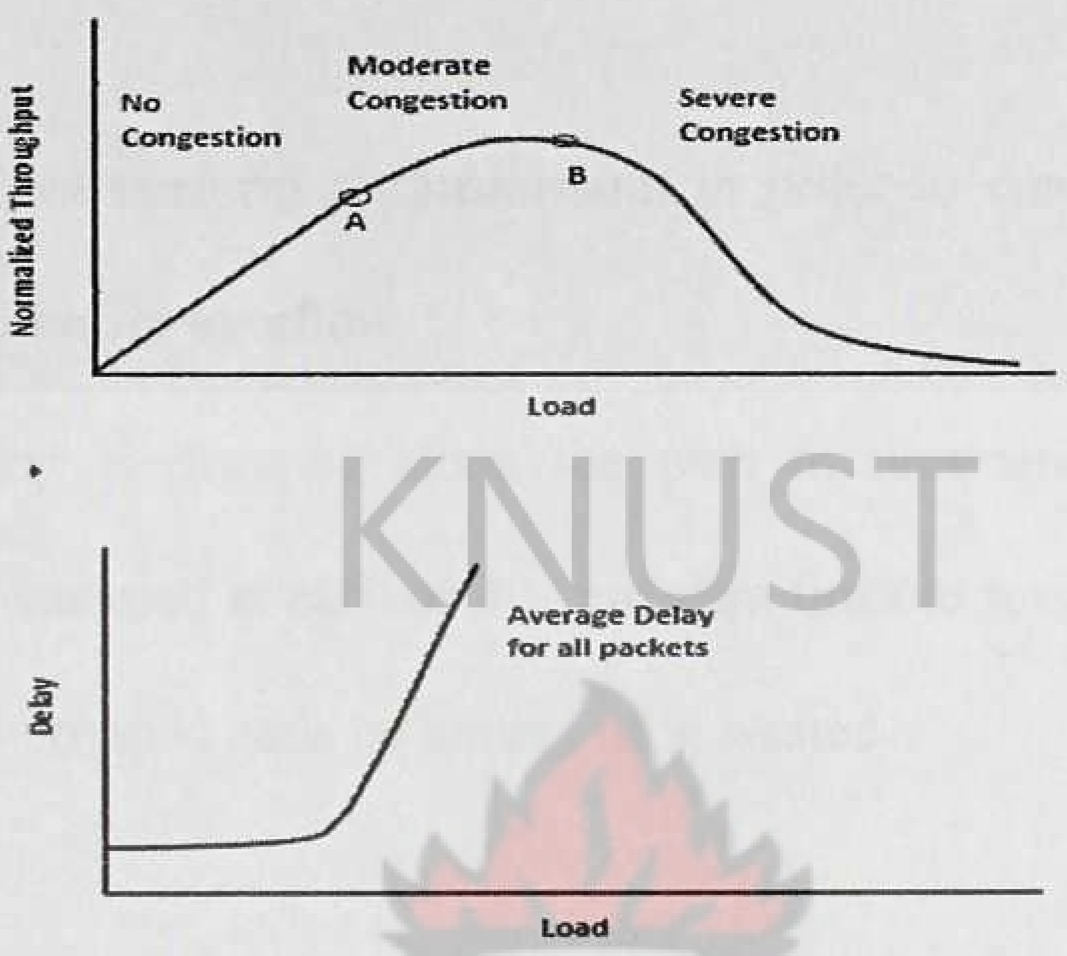


Figure 1.2 load in network

Congestion is a state in a network when the total sum of demands on network resources is more than its available capacity. Mathematically:

$\sum$ Demand > Available Resources **(1.1)**

In WSNs, congestion happens due to contention caused by concurrent transmission, buffer overflows and dynamic time varying wireless channel condition [4].

1.1.1 Effects of Congestion

~~As~~ WSN is a multi-hop network, congestion taking place at a single node may diffuse to the whole network and degrade performance drastically [5]. Congestion causes many folds of drawbacks:

- Increases energy dissipation rates of sensor nodes.

- Causes a lot of packet loss, which in turn diminish the network throughput.
- Hinders fair event detections and reliable data transmission.
- Large queuing delays are experienced as the packet arrival rate nears the link capacity.
- The sender must perform retransmission in order to compensate for dropped packets due to buffer overflow.
- When a packet is dropped along the path to destination, the transmission capacity that was used at each of the upstream links to forward the packet to the point that it is dropped ends up having been wasted.

1.1.2 Congestion Control

Congestion control is the technique of monitoring network utilization and adjusting transmission rates or providing additional resources to keep the number of packets below a level for optimum node performance.

This is significant in achieving reliable event detection for practical realization of WSN based applications. It aims to make the network operate around region A (as shown in Figure 1.2). We found that one of the key reasons for congestion in sensor networks is allowing motes to transfer as many packets as they can. The high amount of data transferred by sensing nodes can overwhelm the capacity of downstream nodes, particularly the nodes near to sink [6].

This thesis proposes an Adaptive Congestion Control Protocol (ACCP) which is more efficient in resource distribution among the sensing nodes. With adaptive control, there is minimal energy consumption which is achieved through traffic control and maximum capacity data transfer to the gateway computer is also met. This is achieved using resource control. The protocol is based on a sink switching mechanism between traffic and resource control.

1.2 Background and Motivation

For the past decade, most researchers [7][8][9] have focused on power management, security of sensors, mobility, node localization and clock synchronization with little or no attention to congestion control as previously seen to be a minor threat to sensor technology survivability. Recently, as more sensitive and modern application in the medical and environmental monitoring field is deployed, there is a need to have accurate data from the sink for efficient result analysis. This is when Engineers and Researchers [10][11] identified congestion as a bigger threat to network instability and packet loss.

WSN has three basic characteristics: centralized data collection, multi-hop data transmission, and many-to-~~one traffic~~ patterns. It means that the nodes closer to the base stations due to the multiple data traffic from upstream nodes need to increase their data sending rate to avoid overflow of buffer. This crisis state may lead to severe packet collisions, network congestion, packet loss and in most severe cases it even results in collapse congestion [12], referred to as the funneling effect [13].

Throttling data traffic was the initial idea of relevant congestion control as seen in CODA [14] ESRT [15] and SenTCP [16] to match data traffic with the bandwidth available to the downstream nodes. This initially solved part of the congestion problem and saved battery power which posses a constant threat to the duration of sensor operation on the field. However, most of the data packet using the traffic approach is lost through throttling and kept data integrity still questionable for sensitive applications. Many researchers came out with an idea to increase the number of intermediate nodes involved in congestion control and use multipath algorithm [17[18[19] to transport data from the source to the sink node. The involvement of additional nodes in resource congestion control is a detriment to energy which is the scarcest resource available to a sensor network.

A typical sensor network is shown in Figure 1.3

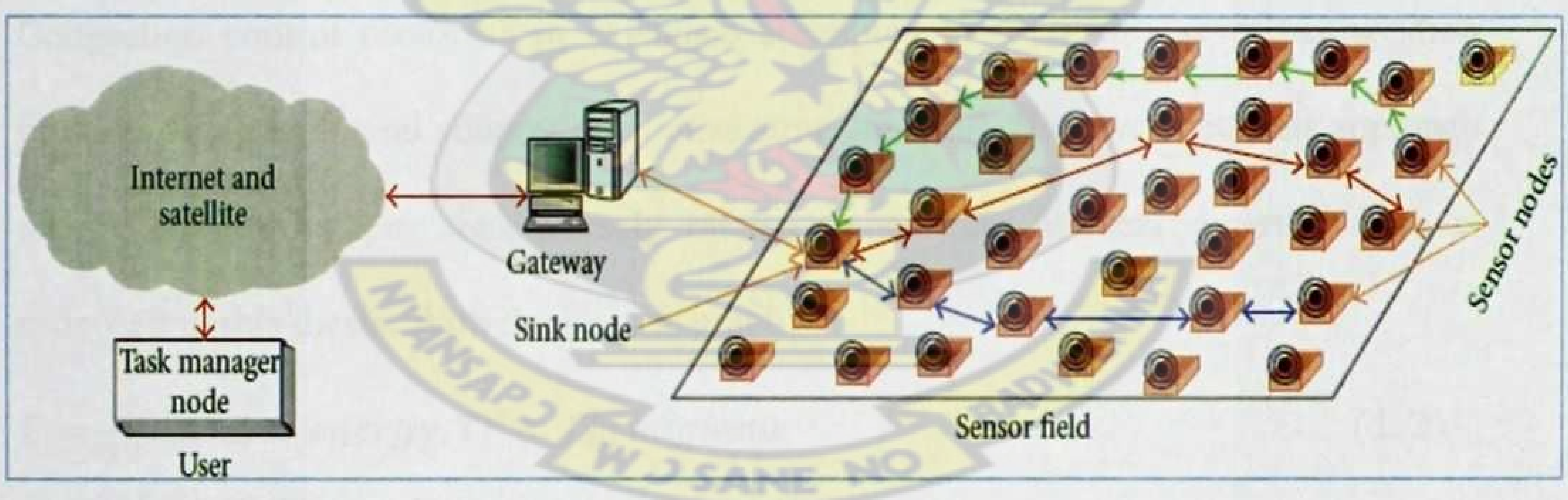


Figure 1.3 sensor nodes scattered in a sensor field

In Figure 1.3, source nodes send data to a sink node through a sensor field. The sink node is the gateway where information is received and made available to a remote user. Each node that is involved in the data transport consumes battery power during the process. The idle nodes are either dead or left for immense traffic scenario. During the idle state, little or no energy is consumed.

1.2.1 Problem Statement

From the congestion graph in Figure 1.2, the sharp decline of throughput from region B as load increases is a state where the intended application of sensor network is rendered ineffective and waste of critical network resource such as energy. This is a state where bandwidth and buffer overflow have reached a threshold value and any further packet arrival leads to severe packet collision. In terms of the VOCs monitoring from the oil field, the gateway computer will detect and report data that is misleading due to the severe congestion at this state of the graph.

Congestion control protocols in existence attempted to solve this severe congestion problem using unilateral congestion control strategies [20]. However with this approach energy and throughput are inversely related rather than a linear growth which is proposed in this thesis. With the inverse graph;

$$E = \frac{1}{TP} \qquad E = energy, TP = Throughput \tag{1.2}$$

Energy parameter is favored at the expense of throughput and vice versa.

1.3 Overview of ACCP Concept

The development of ACCP scenario introduces a switching mechanism in collision detection and control between DelStatic (traffic) and Dynamic Source Routing (DSR) (resource) protocol. DelStatic protocol makes use of backpressure algorithm by dynamic routing of downstream traffic over a multi-hop network using congestion gradients. This algorithm is based on Lypunov networking drift by Neely [21].

This algorithm is implemented to make DelStatic protocol efficient for Traffic control analysis. DSR protocol resource control efficiency is drawn from the on-demand routing algorithm based on the concept of source routing. It allows the network to be completely self-configuring and does not need any existing network infrastructure or administration. DSR is composed of the two mechanisms of Route Discovery and Route Maintenance, which work together to allow nodes to discover and maintain source routes to arbitrary destinations in the network.

In ACCP, there is sink node diffusion request based on the metrics energy remaining to the network and priority of sensor application to switch between DelStatic and DSR protocol.

```
$switch\ (congestion\ control)\{$
$case\ 0:$
$if\ (Energy > threshold)\{resource\ control\}$
$case\ 1:$
$if(Energy < threshold)\{traffic\ control\}$
$case\ 3:$
$if(Hight\ Prior.Application)\{resource\ control\}$
$case\ 4:$
$if(low\ Prior.Application)\{(traffic\ control)\}$
```

The basic concept of adaptive control is to optimize WSN congestion control capability with minimum power utilization by using DelStatic control and maximum throughput by DSR control.

1.4 Research Hypothesis

The development of DelStatic and ACCP will eliminate congestion in WSN with optimum battery performance and data accuracy. Hence critical future application development using sensor networks will be void of huge error margins and real time data analysis will be of essence to researchers again.

1.5 General Objective

The overall objective of this thesis is to improve congestion control in Wireless Sensor Networks by conserving energy and increasing throughput.

1.6 Specific Objective

The first objective of this thesis is to analyze TADD (TORA AODV DSDV DSR) protocols and to establish which one has the highest resource congestion control capability when subjected to network metrics such as throughput, end to end delay, traffic in bytes generated and fairness.

The second objective is to develop a new protocol, DelStatic from NOAH (NO Ad-Hoc Routing Agent n.d.) protocol by using a backpressure algorithm [22][23][24]. DelStatic protocol instead of throttling incoming data in traffic congestion control uses a backpressure algorithm to handle data traffic. A framework module is developed in NS-2 for DelStatic protocol.

The third and major objective is to sink switch DSR or DelStatic protocol (ACCP) as a result of objective one and two outcome.

1.7 Significance and Contribution

Adaptive Congestion control is significant in preventing:

- Increasing energy dissipation rates of sensor nodes.
- Packet loss, which in turn diminish the network throughput.
- Unfair event detections and reliable data transmission.
- Data retransmission in order to compensate for dropped packets due to buffer overflow.

This thesis contributes to sensor technology advancement by:

- The development of new traffic control protocol, DelStatic.
- Analyzing TADD protocols and to establish which one has the highest resource congestion control capability.
- The analysis and simulation of ACCP with switching algorithm.

1.8 Methodology

The methodology adopted in this thesis includes a switching algorithm, ns2 simulation, awk scripts, backpressure algorithm, excel and jTrana script graph. A backpressure algorithm is integrated into NOAH protocol in this project. A switching algorithm for

ACCP analysis is also used. A detailed simulated analysis on TADD using jTrana script graph and excel is of essence in this dissertation.

1.9 Organisation of Dissertation

The rest of the thesis is organized as follows: Chapter two reviews literature on congestion control, Chapter three deals with design and analysis with Chapter four, a vital part of the thesis giving simulation results with generated graphs. The final chapter, Chapter five contains a conclusion and recommendation for future projects and studies.

2
Literature Review

2.1 Literature Review

A number of previous works have addressed the issue of congestion control in Wireless Sensor Networks [25], however most of the works have adopted the unilateral approach to alleviate congestion which is the implementation of congestion control using either traffic or resource approach. Chen and Yan [26] used a hybrid of channel collision and buffer congestion strategy to control congestion.

2.2 Traffic Congestion Control

Congestion at a node happens when the incoming traffic volume exceeds the resource amount available to the node. To alleviate congestion, the incoming data is throttled (referred to as traffic control) mostly using hop-by-hop backpressure.

2.2.1 Congestion Detection and Avoidance (CODA)

Congestion Detection and Avoidance (CODA) falls under Traffic Congestion Control and belongs to upstream congestion control. It contains three components: congestion detection, open-loop hop-by-hop backpressure, and closed-loop end-to-end multi-source regulation. CODA attempts to detect congestion by monitoring current buffer occupancy and wireless channel load. If buffer occupancy or wireless channel load exceeds a threshold-based value, it implies that congestion occurs. Then node detecting congestion will notify its upstream neighbor nodes to decrease rate, with the manner of open-loop hop-by-hop backpressure. The upstream neighbor nodes will trigger to decrease output rate like AIMD (Additive Increase/Multiple Decrease) and to replay backpressure continuously, after they receive backpressure signal. Finally CODA can regulate multi-source rate through closed-loop end-to-end approach, which works as follows:

- When a sensor rate overruns theoretical throughput, it will set "regulation" bit in even packet.
- If the event packet received by sink has "regulation" bit, sink should send ACK control message to sensors and to inform them to decrease their rate.
- If congestion is cleared, sink will actively send ACK control message to sensors and to inform them to increase their rate

CODA deals with traffic-controlling aspect of congestion control (Eisenman and Campbell, 2003). It basically tries to reduce the incoming traffic into the network during congestion. But when congestion is transient, reducing traffic hurts the accuracy level significantly. Increasing resources such as hop-by-hop bandwidth; additional data path may be a better option to cope with congestion. In addition CODA used Closed Loop end-to-end upstream approach to control congestion. This approach generally has a longer response time when congestion occurs, and in-turn will result in lots of segment dropping (Postel, 1981). The segment dropping means that the limited energy resources available to a sensor node for effective data transmission to the sink get wasted. Also the long response time will make it hard to fully fill wireless channel after congestion due to the connection-oriented protocol used by end-to-end CODA for data transmission. The pros of CODA include:

- The use Buffer size and Channel condition to detect congestion and this is an accurate method in terms of congestion detection.
- To some extent an appreciable amount of energy conservation.

However CODA faced some major challenges such as:

- Unidirectional control from sensors to sink.
- The delay or response time of closed-loop multi-source regulation will be increased under heavy congestion since the ACK issued from sink would loss high probability at this time.

2.2.2 Event-to-Sink Reliable Transport (ESRT) [27]

ESRT is another protocol that implements traffic congestion scheme and aims to providing reliability from sensors to sink with congestion control simultaneously. It belongs to upstream reliability guarantee. Firstly it needs to periodically compute the factual reliability r according to successfully received packets in a time interval. Also and the most importantly, ~~ESRT deduces~~ the required sensor report frequency *f* from *r.f = G(r)* and finally, ESRT informs *f* to all sensors through an assumed channel with higher power and sensors can report event and transmit packets with frequency *f*.

ESRT is an end-to-end approach to guarantee a desired reliability through regulating sensor report frequency. It provides reliability for applications not for each single packet. ESRT is not efficient in terms of congestion control. Since it is an End-to-End approach, connection must be established, similar to traditional Transmission Control Protocol (TCP) before data can be transmitted from sensor nodes to sinks and this is not profitable to sensor networks because of accuracy and time dependent nature of sensor applications

- Energy-conservation is the main advantage since it can control sensor report frequency.

The challenges faced include:

- ESRT regulates report frequency of all sensors using the same value. But it may be more reasonable if using different value since it sensor may have different contributions to congestion.
- ESRT mainly considers reliability not congestion control.
- ESRT assumes and uses a channel (one-hop) with high power that will influence the on-going data transmission.

2.2.3 SenTCP [28]

SenTCP is an open-loop hop-by-bop traffic congestion control protocol with two special features.

- 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 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.

- It uses hop-by-hop congestion control. In SenTCP, each intermediate sensor node will issue 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

SenTCP uses hop-by-hop feedback control which gives it higher output in terms of network connectivity and response time. It is one of the efficient transport layer protocol for sensor network.

Advantages of SenTCP include:

- The use of hop-by-hop feedback control removes congestion quickly.
- Packet dropping is minimized under SenTCP.
- Good energy-efficiency.

However SenTCP focuses only on congestion control and guarantees no data reliability from sensor node to the sink.

2.2.4 Predictive Congestion Control

Available congestion control schemes, for example transport control protocol (TCP), when applied to wireless networks, result in a large number of packet drops, unfair scenarios and low throughput with significant amount of packet drops, unfair scenarios due to retransmission. To fully utilize the hop by hop feedback information, a novel, Decentralized Predictive Congestion Control (DPCC) for wireless sensor networks method was adopted. The DPCC consist of an adaptive flow and adaptive back-off interval selection schemes that work in consent with energy, efficient, Distributed Power Control (DPC). The DPCC detects the onset of congestion using queue utilization and the embedded channel estimator algorithm in DPC that predicts the channel quality. Then, an adaptive flow control scheme selects suitable rate which is enforced by the newly proposed adaptive back off interval selection scheme. An optional adaptive scheduling scheme updates weights associated with each packet to guarantee the weighted fairness during congestion. Closed-loop stability of the proposed hop-by-hop congestion control is demonstrated by using the Lyapunov-based approach. Simulation results show that the DPCC reduces congestion and improved performance over CODA because it prevents retransmission from occurring in the sensor network. The only disadvantage is that, it cannot prevent congestion when the data congestion traffic is very high. This means that, the adaptive resource approach coupled with predictive congestion detection will be better [29].

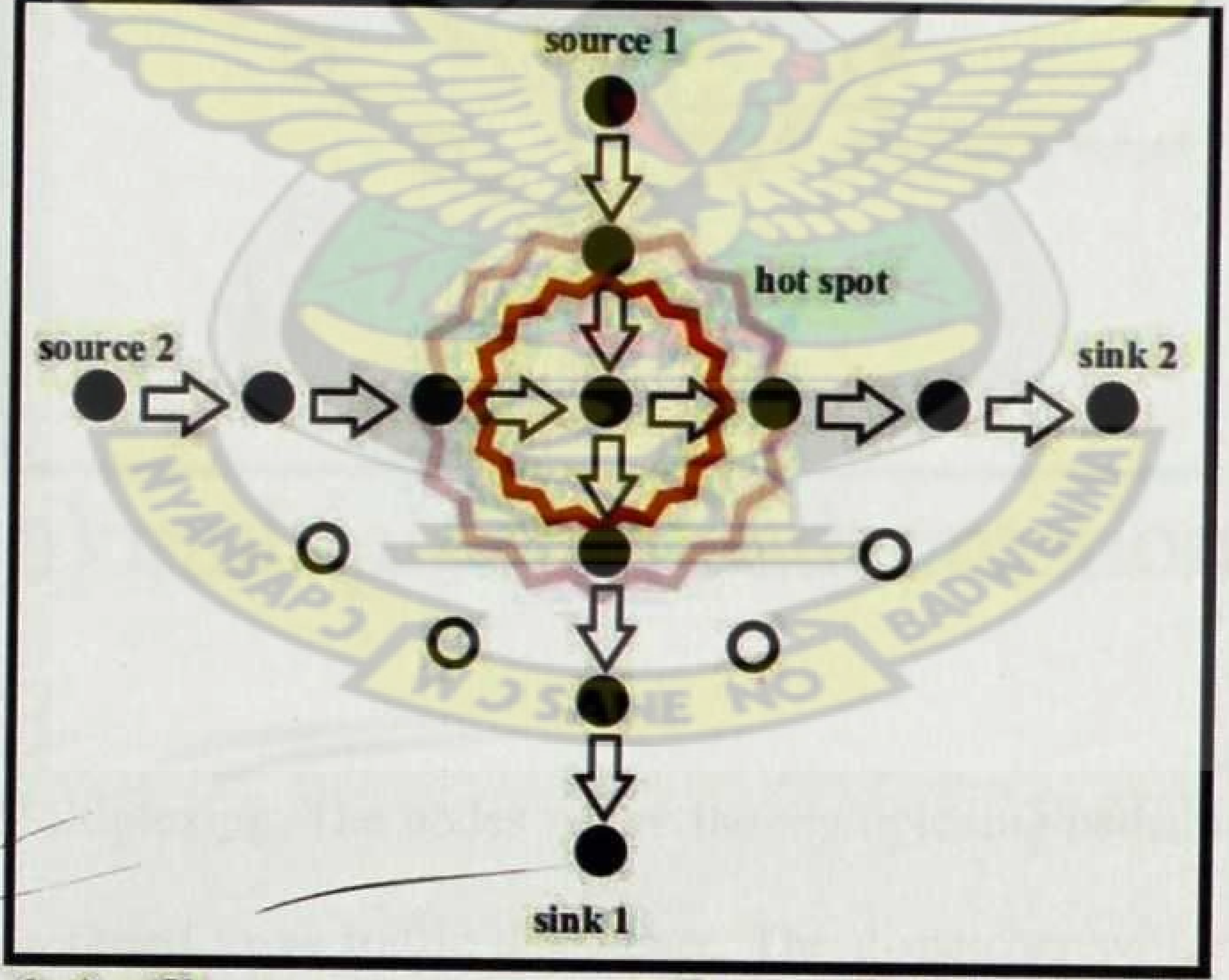
2.3 Resource Congestion Control

When data traffic at a node increases beyond the resources available at that node, the available resources to the node should be increased in order for the node to cope with congestion. This is referred to as resource congestion control.

2.3.1 Adaptive Resource Control Scheme

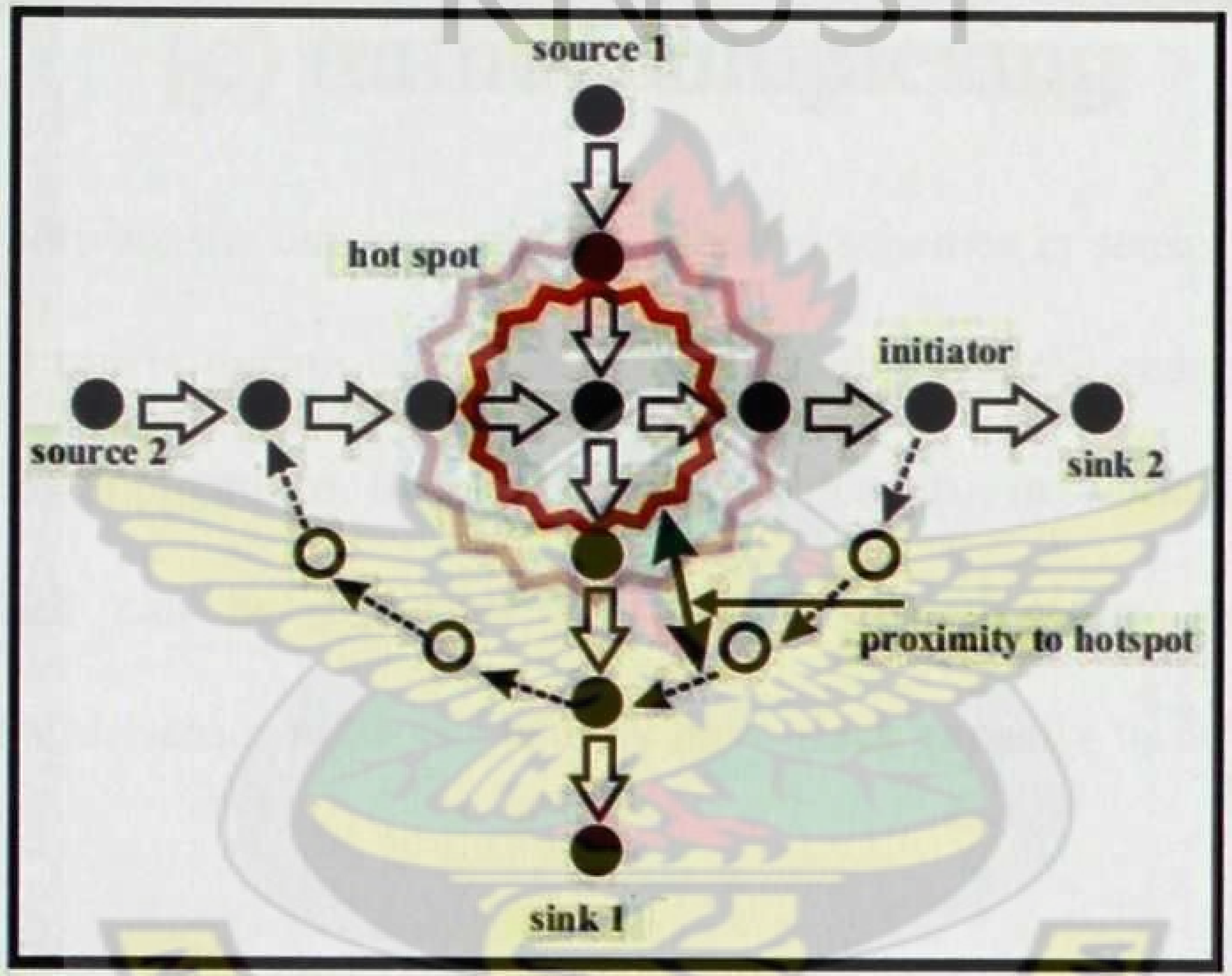
During a dormant state, the sensor network usually turns off a large number of nodes to extend the network lifetime [30]. As soon as flow-level congestion is detected, the nodes are incorporated into routing by forming one or more additional paths called multiplexing paths, and distributing the incoming traffic over the original path and the multiplexing paths. The algorithm involves three steps:

a. Congestion Detection. The upstream nodes of flow periodically notify their congestion level to downstream nodes by embedding their perceived congestion level into the header of data packets. After detecting the flow-level congestion exceeds a predefined congestion threshold, the first node whose congestion level is below the hotspot proximity threshold will become the initiator and start creating multiplexing paths.



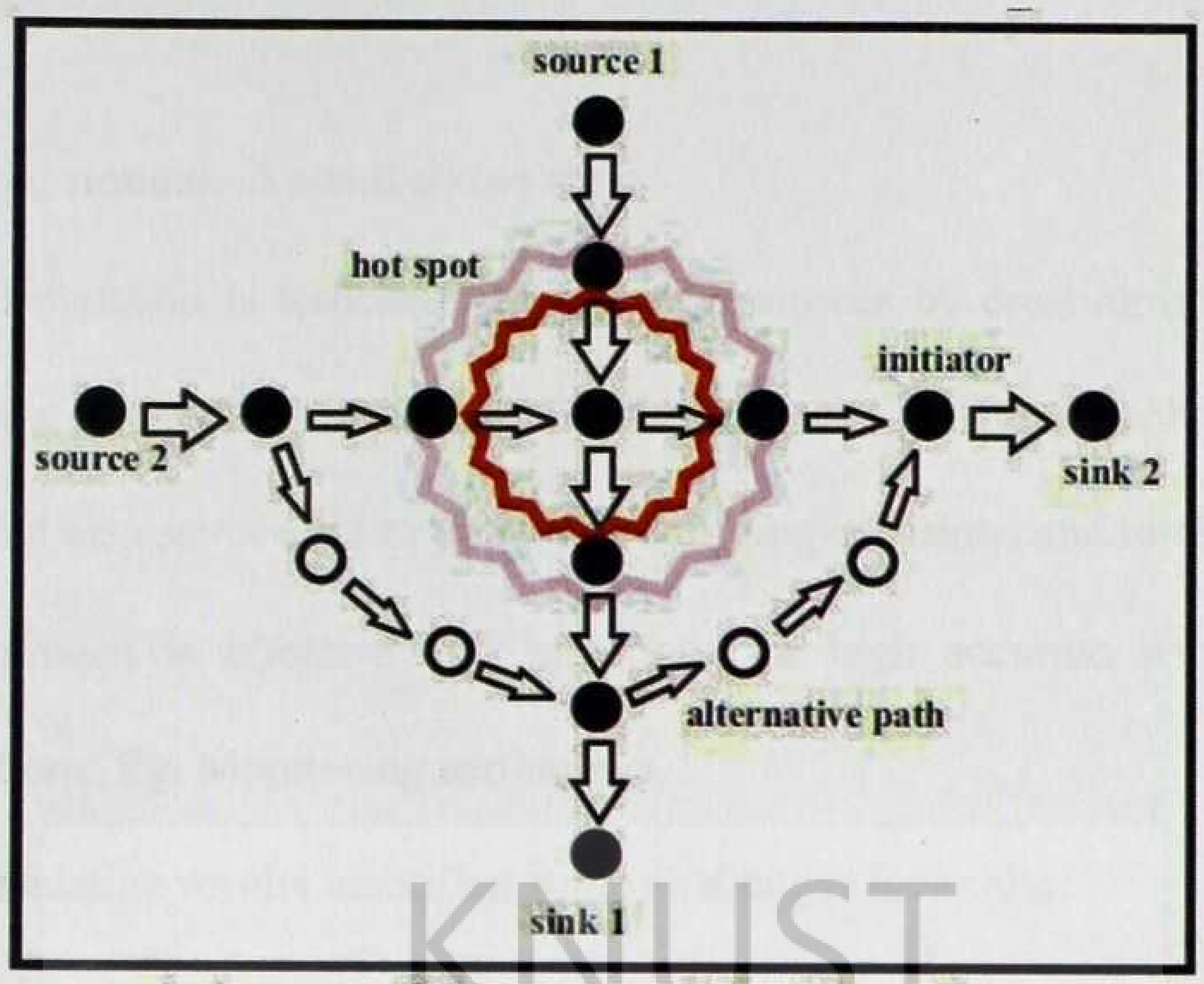
(a) flow congestion detection

b. Alternative Path Creation [31][32]Building multiplexing paths is a challenging task. If the path is too close to the hotspot, it does not help, but it will further interfere with the already bad congestion. On the other hand, if the path is too far from the original one, then the packet delivery latency and the energy consumption will increase. In addition, building multiplexing path can be made even more complicated by other factors. For example, one may not be able to find a path from the initiator to the source; or there may exist another hotspot within the network.



(b) multiplexing path creation

c. Traffic Multiplexing. The nodes where the multiplexing paths meet the original path are referred to as traffic dispatcher. The dispatcher will evenly distribute the traffic between multiples paths in a round robin fashion so that congestion on the original path can be alleviated



(c) traffic multiplexing

Three factors motivate the usage of resource control schemes in sensor networks. First, reducing source traffic during crisis state is undesirable since it reduces the accuracy level observed by applications. Second, congestion in sensor networks is often transient by nature which can be alleviated by quickly adjusting the network resource provisioning. Third, sensor networks usually have slack capacity to be turned on when needed.

This resource control scheme has two phases. To increase resource provisioning as soon as congestion occurs; and to reduce the resource budget as soon as congestion subsides. The scheme has managed to balance the need of reliably shipping data packets to the sink during a crisis state and the need of conserving energy consumption. Delay turnoff of nodes after congestion has die consequence on the energy consumption of the network, and this is what adaptive resource control scheme tries to avoid.

Adaptive resource simulation result shows that:

- When congestion is transient, increasing resources by creating multiple paths around the hotspot effectively increase not only the delivered packets (accuracy level), but also saves a lot of energy by avoiding collisions and retransmissions.
- The approach is effective and important for high accurate wireless sensor applications. Eg. Monitoring earthquake.

However, the simulation results identified some challenges including:

- The complex nature of multiplexing protocol design.
- In the case where additional sensor nodes are not turned off immediately after partaking in congestion control, will lead to high energy consumption in the network.

2.4 Hybrid Congestion Control

Hybrid Congestion Control is the protocol approach where two or more congestion control protocols are combined with the aim of alleviating congestion from the wireless network system.

2.4.1 Hybrid Congestion Control Protocol in Wireless Sensor Networks [33]

In this paper, a Hybrid Congestion Control Protocol, considering both the packets delivery rate and remaining buffer size of each node is proposed. The scheme does not need to maintain the global flow information and each node makes use of its current remaining buffer size and net flow size to calculate its congestion degree information.

The congestion degree is defined to reflect the current congestion level at each node. Then, the congestion degree is exchanged periodically between neighbors. Therefore, each node can use its congestion degree and neighbor's congestion degrees to prevent the emergence of congestion. The simulation result show that the protocol can reduce packets drop rate and increase packets ration effectively.

In the analysis of HCCP, a hybrid congestion control protocol which considers both the packets delivery rate and remaining buffer size of each node. In congestion detection phase, HCCP detects the congestion in advance with a time period T and takes the preventive measures. In data rate adjustment phase, the upstream neighbors that tend to congest will be allocated more data rate.
HCCP achieved a higher output in terms of total source rate congestion control but had a higher control overhead.

3
Design and Analysis

3.1 Design and Analysis

In this chapter, we provide the mathematical and programming background for the design. The procedures, strategies and algorithm used to optimize Adaptive Congestion Control Protocol (ACCP) is further discussed. We will justify the impact ACCP has on resolving congestion in a sensor network with improved energy-efficiency and throughput.

3.2 Adaptive Congestion Detection and Control Mechanism

As shown in Figure 3.1, an intermediate node receives data traffic from multiple source nodes. This data flow mechanism poses threat to the resources available to the downstream nodes and if bandwidth and buffer occupancy usage of the node exceeds a threshold, congestion emanates.

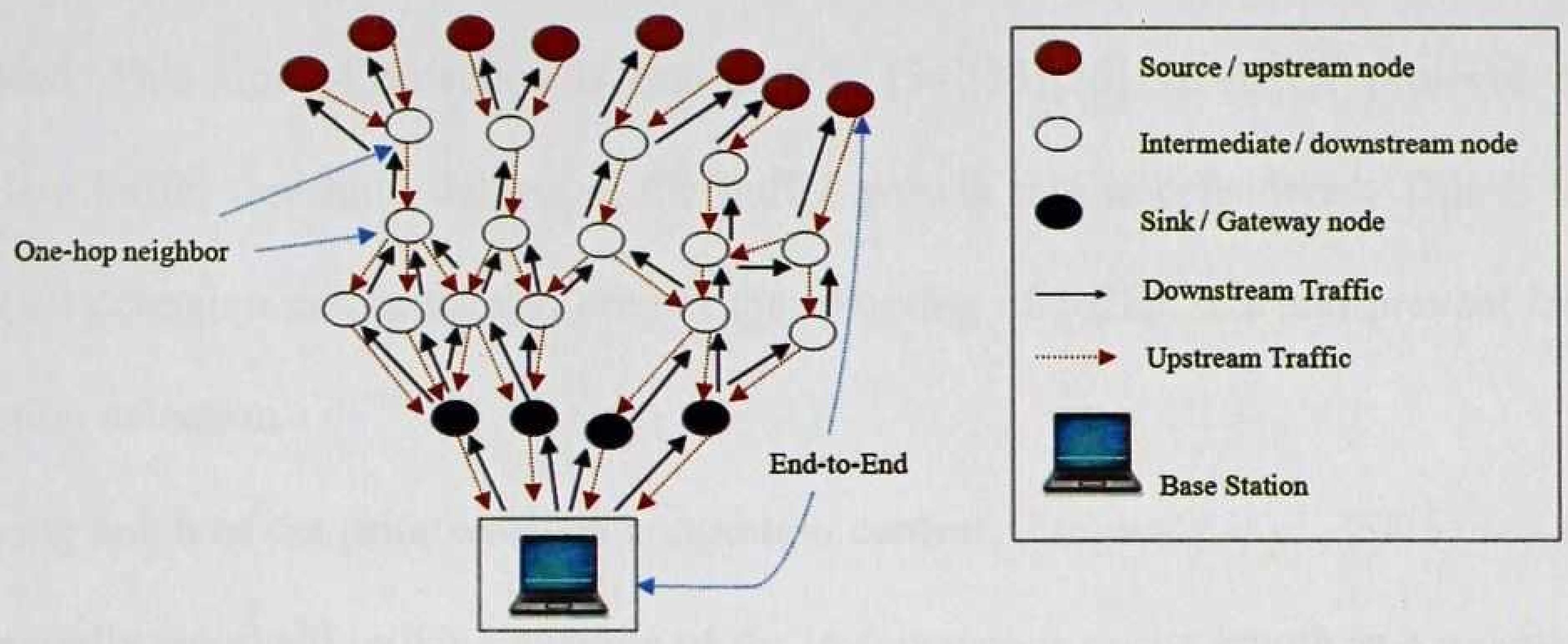


Figure 3.1 Wireless Sensor Network and data transmission

It is important to control congestion at the intermediate node to guarantee a successful transmission of the data carried by the source nodes meant to be disseminated to the base station in an efficient manner. To effectively detect congestion in a sensor application, we implement a double congestion detection mechanism which is efficient for ACCP. To do this, the channel utilization strategy is complimented with buffer occupancy in ACCP congestion detection. Adaptive congestion control is based on two critical techniques:

- Detect congestion at a node using Buffer Occupancy & Channel Utilization Strategy.
- Control the detected congestion by initializing ACCP.

3.2.1 Buffer Occupancy Detection

In buffer occupancy, we compare instantaneous buffer occupancy of intermediate nodes against a set of threshold value. If the threshold value is exceeded, then congestion is diagnosed. This kind of detection is also seen in [34][35][36]. In ACCP protocol, to avoid late buffer threshold detection, the buffer growth rate is considered. This is an intelligent detection mechanism to predict the sampling of buffer size and prevent late congestion detection.

Following much of the prior work on congestion control, Rangwala *et al.*, 2006 uses an exponentially weighted moving average of the instantaneous queue length as a measure of congestion:

$$AVG_q = (1 - w_q) * AVG_q + w_q * inst_q \qquad \textbf{(3.1)}$$

where $AVG_q = average\ queue\ length\ of\ node$

w_q = weighted queue length of node

$inst_q = queue\ lenght\ of\ node\ at\ the\ current\ instant$

The average queue length is updated whenever a packet is inserted into the queue. Thus, if AVG_q exceeds a certain upper threshold U, the node is said to be congested. The node remains in a congestion state until AVG_q falls below a lower threshold L. In practice, a single threshold is too coarse-grained to effectively react to congestion.

Buffer occupancy alone is not a reliable congestion indicator because packets can be lost in the channel due to collision or hidden terminal situations and have no chance to reach a buffer [37]. ACCP implements both Channel Utilization Strategy and buffer occupancy for effective congestion detection.

3.2.2 Channel Utilization and Detection Strategy

Channel Utilization technique uses Carrier Sensing Multiple Access (CSMA) algorithm to listen to the communication medium before data is transmitted. This is accomplished with the help of the Media Access Control (MAC) Protocol. The MAC protocol ensures communication in the wireless medium such that the communication links between nodes are established and connectivity is provided throughout the network. To maximize channel utilization detection strategy, the channel detection delay of wireless medium is of essence (Bertsekas and Gallagher, 1991).

$$S_{max} \approx \frac{1}{(1+2\sqrt{\beta})}\left(for\ \beta = \frac{\tau C}{L} \ll 1\right), \qquad (3.2)$$

S_{max} = CSMA maximum theoretical throughput approximation

β = the measure of radio propagation delay and channel detection delay

τ = the sum of both radio propagation delay and channel idle detection delay in seconds

C = the raw channel bit rate

L = the expected number of bits in a data packet

In achieving a high S_{max} value for good performance, we lowered the β value, which determines how quickly a node can detect idle periods. CSMA efficiency is highly dependent on the β value.

For data packet collision avoidance in the transmission medium, CSMA algorithm requires a user to be sure the medium is free before data transmission. This is the carrier sensing part. If medium is busy, the user has to delay for a period and then re-sense. The random period is to avoid congestion since collision can occur if another node is sending data at the same time. As soon as the channel is idle, the user can start transmission by RTS/CTS.

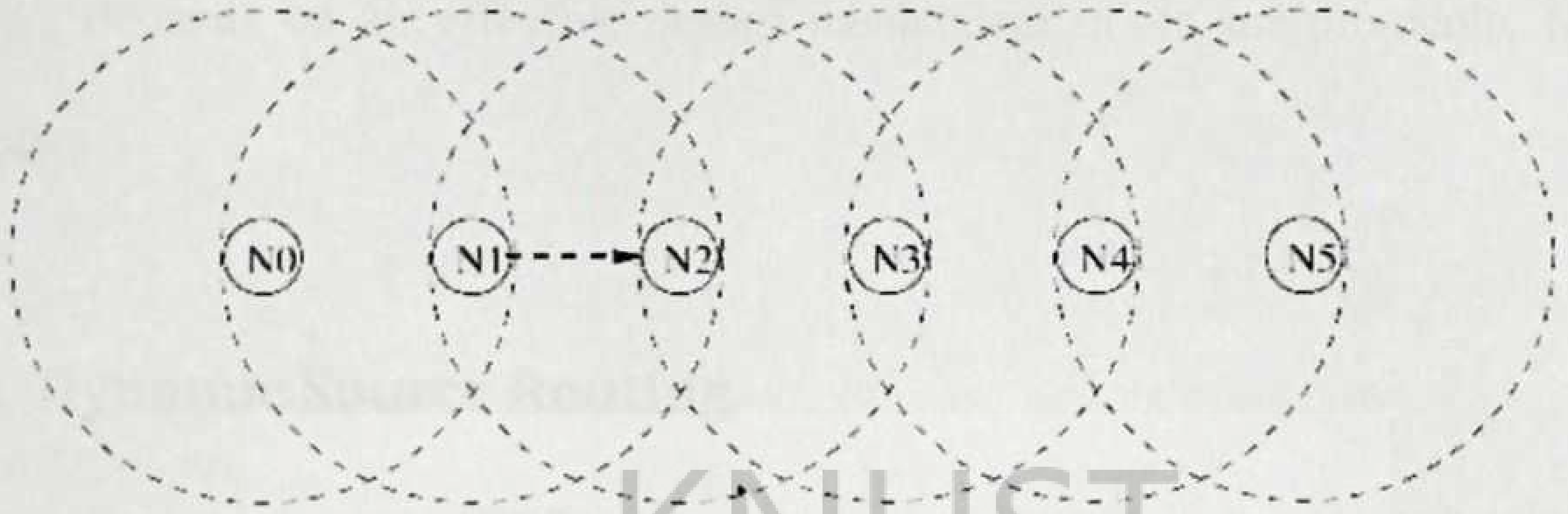


Figure 3.2 interference in wireless sensor networks

Figure 3.2 illustrates the mechanism of RTS/CTS; Node N1 sends a RTS frame to node N2 before the real transmission. Node N0 also receives the RTS and is blocked by it. Upon receiving the RTS, node N2 broadcasts the CTS frame to its neighbors. Thus, node N3 is also blocked. Node N1 starts transmitting data once receiving the CTS frame from node N2. The RTS/CTS mechanism is to deal with hidden terminal problems.

N5 is a hidden node of the transmission from N1 to N2, since N5 is beyond the interference range of N2 (two hops). Node N5 cannot sense the data flow from N1 to N2 and will think the medium is idle. If there is no RTS/CTS, node N5 will directly start sending data packets to N4. In this case, the ACK frames from node N4 will be very likely to collide with the data received by N2. With the use of RTS/CTS, node N5 ~~won't~~ get the CTS from N4 and cause interference to N1 and N2 since N4 can detect the flow between N1 and N2.

As mentioned in Section 3.1, after detecting congestion, ACCP is initialized to alleviate it from the system.

3.3 ACCP Implementation Details

In this section, we analyze the design of DSR and DelStatic protocols which forms NS-2 modular implementation of ACCP. The efficiency of ACCP in congestion control therefore depends on an effective design mechanism of its sub-protocols, DSR and DelStatic.

3.3.1 Dynamic Source Routing

Dynamic Source Routing (DSR) is a member of TADD protocols and designed to handle the resource controlling aspect of ACCP. Once a sink node triggers a DSR control, the congested node becomes intelligent to re-route upstream data traffic through the redundant path created by the DSR protocol. In Figure 3.3, a DSR protocol creates a redundant path from congested node n4 by waking up sleeping node n6.

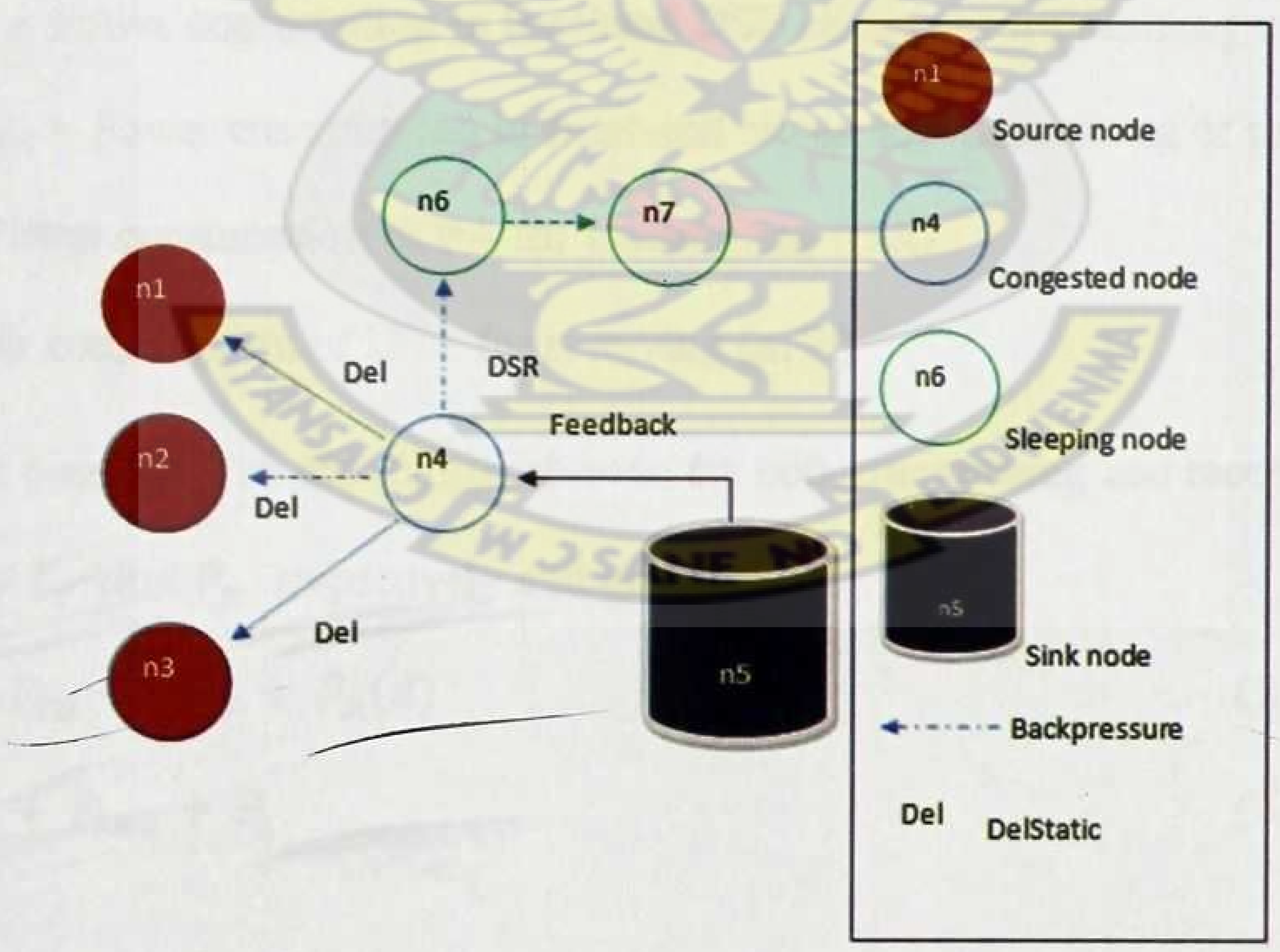


Figure 3.3 Dynamic Source Routing making use of additional node.

With this redundant path, node n4 will switch most of the upstream traffic to node n6. The new node consumes extreme energy resources through data transmission, receiving and communication to other nodes.

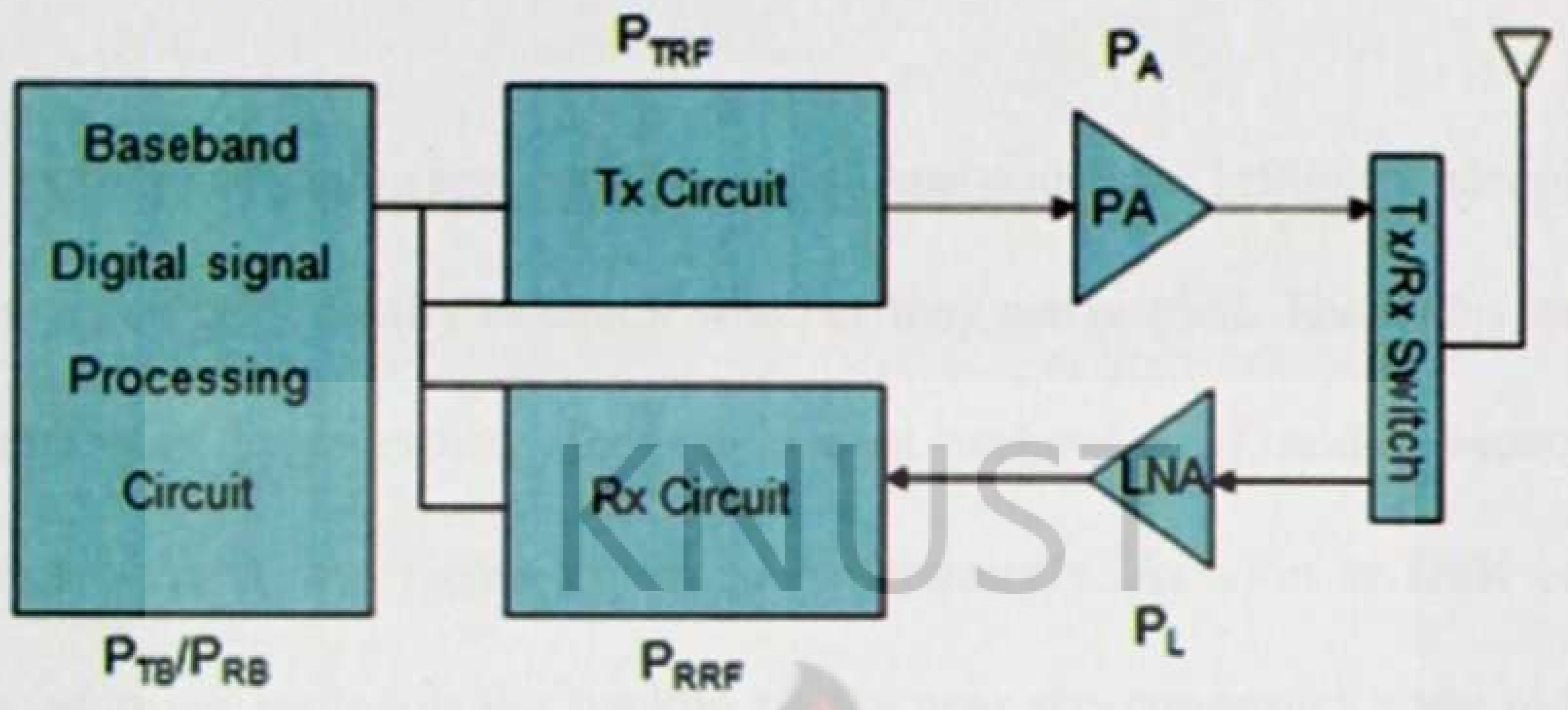


Figure 3.4 WSN node Communication Module Structure

$P_T\ (d)$ = Power consumption for transmission which is a function of distance.

P_R = Power consumption for receiving

$P_{TB}\ /\ P_{RB}$ = Power consumption in baseband DSP circuit for transmitting or receiving (mW)

$P_{TRF}\ /\ P_{RRF}$ = Power consumption in front-end circuit for transmitting or receiving (mW)

$P_A(d)$ = Power consumption of PA for transmitting (mA)

P_L = Power consumption of LNA for receiving (mW)

The power consumed by the additional node for both transmitting and receiving is denoted by P_T *and* P_R respectively and given as:

$$P_T\ (d) = P_{TB} + P_{TRF} + P_A(d) \tag{3.3}$$

$$P_R = P_{RB} + P_{RRF} + P_L \tag{3.4}$$

As seen in Equations (3.3) and (3.4), the total power consumed by a sensor node is given by $P_T(d) + P_R$ and any dormant node triggered into active state utilizes this amount of energy in one data transmission [38].

Due to this extreme energy usage, most of the backup nodes are left in the sleeping state and only wakes up periodically to check whether they are needed. The sleep interval of nodes is important to determine the subsequent wake-ups of nodes because it is undesirable to wake up too frequently or too infrequently. As soon as DSR control is triggered in adaptive protocol, the backup nodes near the congested node reduces its sleeping interval and becomes active. Once they become active, they communicate with each other using DSR Route Discovery and Route Maintenance algorithm to trace the sink node. Similarly, a large number of back up nodes that are not too far away from the hotspot can thus be woken up and be ready to be picked to form the Route Discovery path.

Our DSR control has two phases:

- To increase resource provisioning as soon as congestion occurs;
- To reduce the resource budget as soon as congestion subsides

3.3.1.1 DSR Control Strategies to Alleviate Congestion

The Dynamic Source Routing (DSR) [39] is an on-demand routing protocol that is based on the concept of source routing where the sender of a packet determines the complete sequence of nodes through which, the packets are forwarded. This operating

mechanism allows the sensor network to be completely self-organizing and self-configuring and does not need any existing network infrastructure or administration.
DSR uses no periodic messages in data transmission thereby reduce network bandwidth overhead, conserve battery power and avoid large routing updates. However DSR needs support from the MAC layer to identify link failure. DSR is composed of the two mechanisms of Route Discovery and Route Maintenance, which work together to allow nodes to discover and maintain source routes to arbitrary destinations in the network.
DSR has another unique advantage by virtue of source routing. As the route is part of the packet itself, routing loops, either short – lived or long – lived, cannot be formed as they can be immediately detected and eliminated. This property opens up the protocol to a variety of useful optimizations.

In DSR, each node maintains a route – cache of all known self – to – destination pairs. If a node has a packet to send, it attempts to use this cache to deliver the packet. If the destination does not exist in the cache, then route discovery phase is initiated to discover a route to destination by sending a route request. This request includes the destination address, source address and a unique identification number. If a route is available from the route – cache but is not valid any more, a route maintenance procedure may be initiated.

A node processes the route request packet only if it has not previously processed the packet and its address is not present in the route cache. A route reply is generated by the destination or by any of the intermediate nodes when it knows about how to reach the destinations.

3.3.2 DelStatic Congestion Control

The traffic controlling aspect of Adaptive Congestion Control is handled by DelStatic Protocol. In DelStatic congestion control, a congested node operates on a backpressure algorithm [40] by broadcasting a suppression message to the upstream nodes to reduce data transmission. This scenario is shown in Figure 3.5

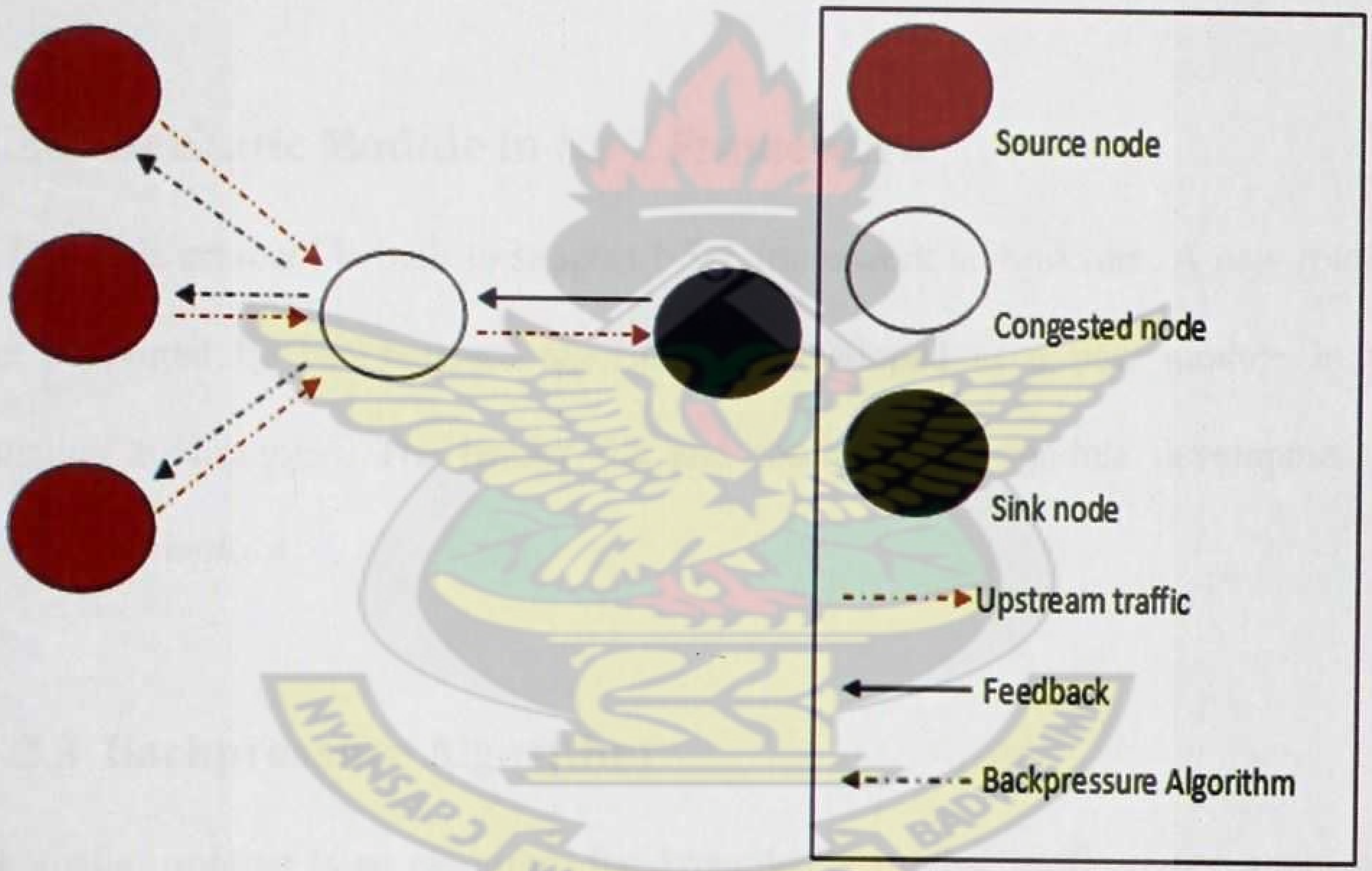


Figure 3.5 DelStatic Traffic Reduction Scenario

3.3.2.1 DelStatic Direct Communication Technique

DelStatic is a wireless routing agent that in contrast to DSR only supports direct communication between wireless nodes. This allows simulating scenarios where multi-hop wireless routing is undesired and energy expenditure is minimal. DelStatic does not send any routing related packets, meaning that it is a static protocol which works on only pre-selected node paths.

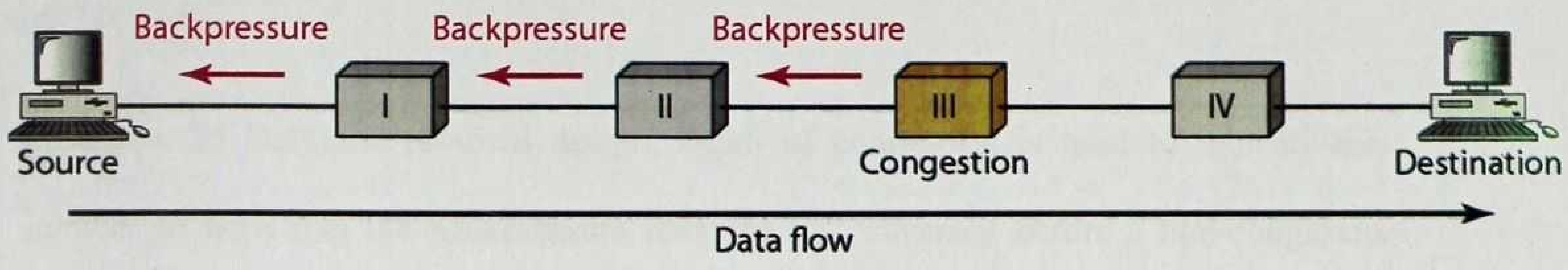
In DelStatic protocol, further update is done to allow static multi-hop routes. The routes are set up using the routing command which takes as parameters the number of destinations and then as many tuples of destination and next hop address. This sets up static routing for a line of nodes.

3.3.2.2 DelStatic Module in NS-2 Framework

The DelStatic protocol is built to support NS-2 framework architecture. A new routing agent supported by backpressure algorithm is developed as a new module in the simulation environment. The header file and routing agent module development is shown in *Appendix A*.

3.3.2.3 Backpressure Algorithm

Backpressure routing is an algorithm for dynamically routing traffic over a multi-hop network by using congestion gradients. The policy, which we call the adaptive back-pressure congestion control policy which states that the upstream nodes can only drop data packets but forbidden to select additional path.



In DelStatic, when a sensor channel load exceeds a threshold or the buffer occupancy reaches a certain high watermark level, congestion is implicit. The said node then broadcasts a suppression message to upstream nodes by using open-loop hop-by-hop data flow strategy until the message reaches the source nodes. Each node then handles this suppression message by throttling its sending rate or drop packets using packet drop and Additive Increase Multiple Decrease (AIMD) policy.

Although there is no guarantee that all immediate upstream nodes will get the suppression message, at least some neighboring nodes will get it probabilistically. Depending on the hop- count of congested node from the source nodes, a congested node will continue to broadcast this message up to the number of times the source node receives the suppression signal.

A node *B* receiving a backpressure message from another node *A* can drop packets or reduce its rate based on its congestion level. Node *B* can also decide to forward the backpressure message further toward the data source. If node *B* is itself congested, it can increase a counter in the backpressure message before forwarding it. As soon as the message reaches a non-congested node, the size of the congested area can be roughly inferred from the counter value and no backpressure message is further propagated upstream. In DelStatic protocol design, depth of congestion is used to indicate the number of hops that the backpressure message has traversed before a non-congested node is encountered. The backpressure message provides the basis for the open loop backpressure mechanism and can also serve as on-demand "Clear To Send" (CTS) signal, so that all other neighbors except a single sender (which is picked randomly) can be silenced at least for a single packet transmission time. This method is open-loop because the node issuing the backpressure message receives no direct feedback.

The core of this project is to adapt DelStatic and DSR congestion control protocol to work in one framework by implementing a switching mechanism which is controlled by the sink node.

3.3.3 ACCP Strategies to Alleviate Congestion

The proposed protocol investigates the possibility of adapting both DSR and DelStatic congestion control in a sensor network. From the protocol design, the adaptive control protocol introduces a switching mechanism between DelStatic (Traffic Control) and Dynamic Source Routing (Resource Control) protocols.

With adaptive control, each node calculates its Node Rank (NR) based on its Buffer level and Channel Utilization ~~Ratio~~. Here the channel busy ratio represents the interference level and is defined as the ratio of time intervals when the channel is busy due to successful transmission or collision to the total time. After estimating its rank, a node forwards this to its downstream node. When NR crosses a threshold T, then the node will set the congestion bit (CB) in every packet it forwards.

On receiving this NR, the downstream node will first check for the congestion bit. If it is not set, it will simply compute its node rank and adds it to the rank obtained from its previous node and passes on to the next node. On the other hand, if the congestion bit is set, the node sends a feed to the sink node to confirm whether to trigger DelStatic or DSR control. Depending on the energy remaining to the network and the sensor application priority, the sink will send a feedback control to the sending node. This node will intern based on the information received from the sink set Rate Adjustment Feedback (RAF) either triggers an alternate path (DSR Control) or transmit it towards the source as a feedback. On receiving the feedback, the source nodes will adjust their transmission rate (DelStatic).

This algorithm for adaptive congestion control is a major build up on Kamal *et al.* (2010) but introduces the sink algorithm and DSR protocol for efficient congestion mitigation in sensor networks.

A. MAC Overhead

In adaptive control protocol, congestion detection using IEEE 802.11 MAC with Distributed Coordination Function (DCF) is adopted. It has the packet sequence as request-to-send (RTS), clear-to-send (CTS), and data, acknowledge (ACK). The time difference between the receipt of one packet and the transmission of the next is called a short inter frame space (SIFS). Then the channel occupation due to MAC contention will be

$$C_{occ} = t_{RTS} + t_{CTS} + 3t_{SIFS} \quad \textbf{(3.5)}$$

Where t_{RTS} and t_{CTS} are the time consumed on *RTS* and *CTS*, respectively and t_{SIFS} is the SIFS period.

Then the *MAC* overhead OH_{MAC} can be represented as

$$OH_{MAC} = C_{occ} + t_{acc} \tag{3.6}$$

Where t_{acc} is the time taken due to access contention. The amount of *MAC* overhead is mainly dependent upon the medium access contention, and the number of packet collision. That is,

OH_{MAC} is strongly related to the congestion around a given node.

B. Channel Busy Ratio

Channel Busy Ratio represents the interference level and is given by

$$CHBR = \frac{Blnt}{Ttot} \tag{3.7}$$

where $Blnt$ represents the time interval when the channel is busy due to successful transmission or collision and $Ttot$ represents the total time.

3.3.3.1 Algorithm to Support Adaptive Framework

Each node calculates its NR based on the following parameters *BSize, HC, CBHR* and OH_{MAC}, where *BSize* is the buffer size of the node and *HC* is the hop count value.

$$NR = \alpha 1.BSize_{n1} + \alpha 2.H + \alpha 3.CHBR + \alpha 4\, OH_{MAC} \tag{3.8}$$

Here $\alpha 1, \alpha 2, \alpha 3$ *and* $\alpha 4$ are constant weight factors whose values between 0 and 1

Each data packet has a congestion bit (CB) in its header. Every sensor node maintains a threshold T. When NR crosses the value of T the node will set its CB in every packet if forwards.

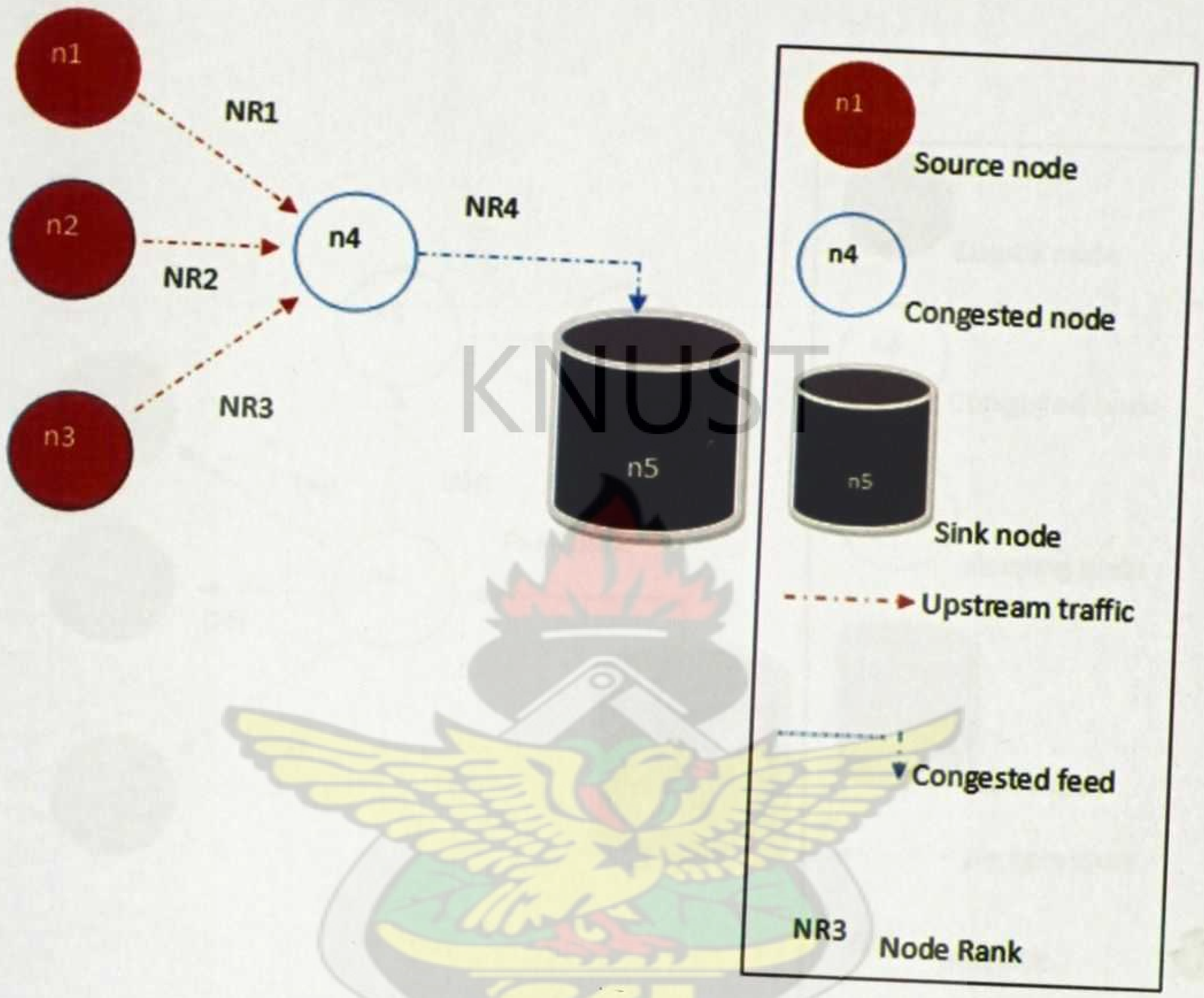


Figure 3.6 Node Rank Propagation

In Figure 3.6, Nodes n1, n2, n3 estimates their rank NR1, NR2, NR3 and forwards this to its downstream node n4 along with data packets. On receiving NR1, NR2 and NR3, n4 will first check their CB value. If it is not set, it will simply compute its node rank NR4 and adds it to the rank obtained from nodes n1, n2, and n3 and passes it to the next node or sink. Figure 3.7 illustrates this process.

On the other hand, if the CB is set at n1, n2 and n3, the node n4 will send a feed to the sink about the congestion scenario. The sink node uses energy remaining (*ER*) to the network and priority of sensor application (*SP*) to determine which control to use for feedback.

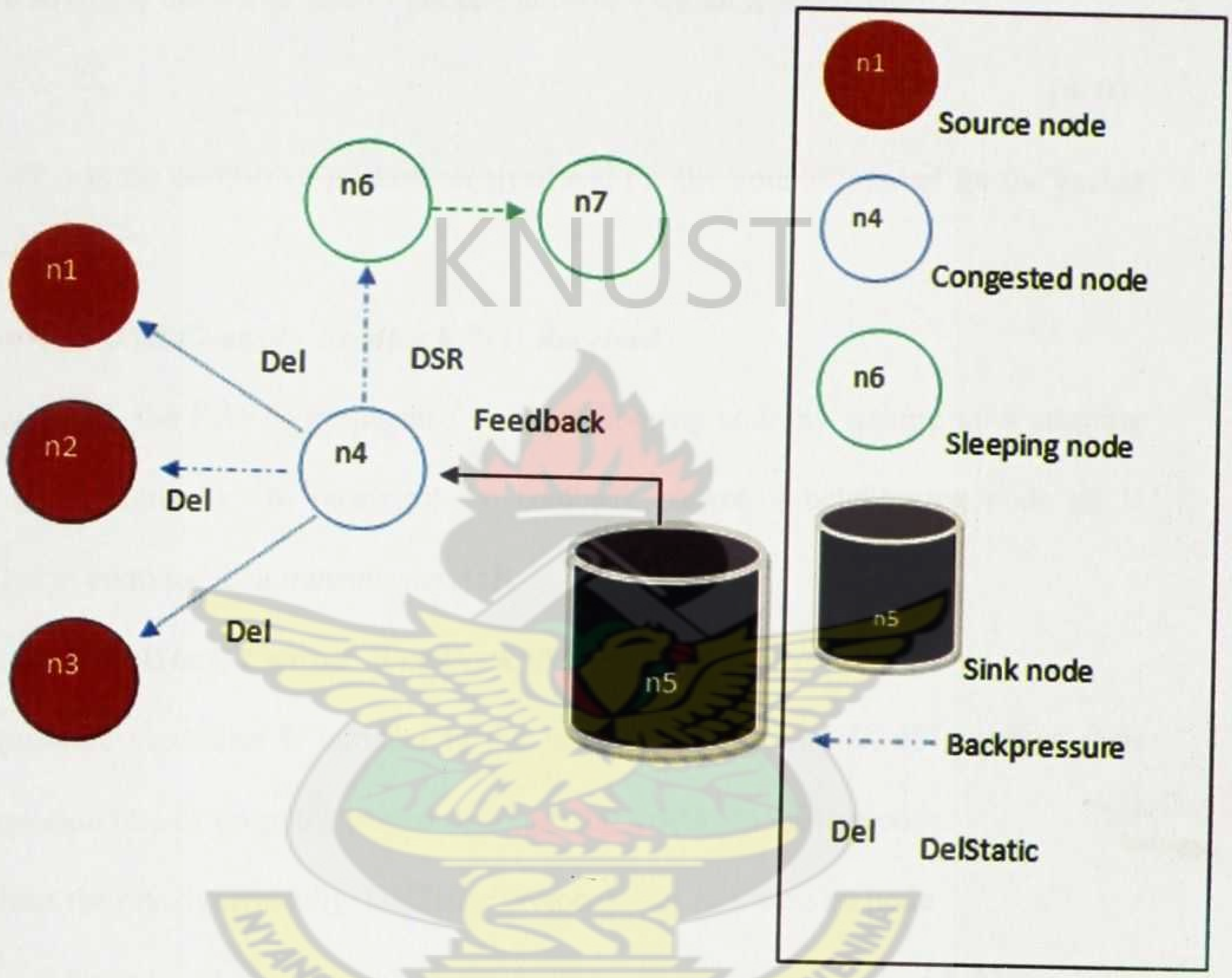


Figure 3.7 Feedback Propagation

The sending node receives either *N1* (DSR Control) or *N0* (DelStatic Control) from the sink as a feedback signal. The sink operates the feedback with request from directed diffusion protocol to disseminate *N1* and *N0* information to the congested nodes.

The sending node based on the *N1* or *N0* feedback information from sink calculates its Rate Adjustment Feedback (*RAF*) based on the rank as

$$RAF = \left(\frac{Arate}{HC}\right) - \Sigma^{OH}/_{MACi} - \Sigma CBHRi \qquad \textbf{(3.9)}$$

Where $Arate$ is the arrival rate of packets at node *n* which is given as:

$$Arate = \frac{NP}{t} \qquad \textbf{(4.0)}$$

Here NP – is the number of packets received and t is the time of interval for the packet transmission.

Scenario 1: DSR Control - Feedback (N1) Received

In Figure 3.7, the RAF is propagated to a neighboring node by waking up a sleeping node (DSR Control). On receiving the feedback packet, a neighboring node n6 is signaled to continue data transmission **OR**

Scenario 2: DelStatic Control - Feedback (N0) Received

Backpressure algorithm is initiated to the upstream nodes n1, n2, n3 to adjust their transmission rate by dropping packet or triggering a delay in milliseconds.

To adjust the rate dynamically, DelStatic protocol uses formulae on node

$$Nrate = Nrate - RAF \qquad \textbf{(4.1)}$$

Thus the traffic rate is adaptively adjusted according to the MAC contention and buffer size. The procedures are repeated for all the hops towards the sinks which are congested.

3.3.3.2 Framework to Support Adaptive Switching Control

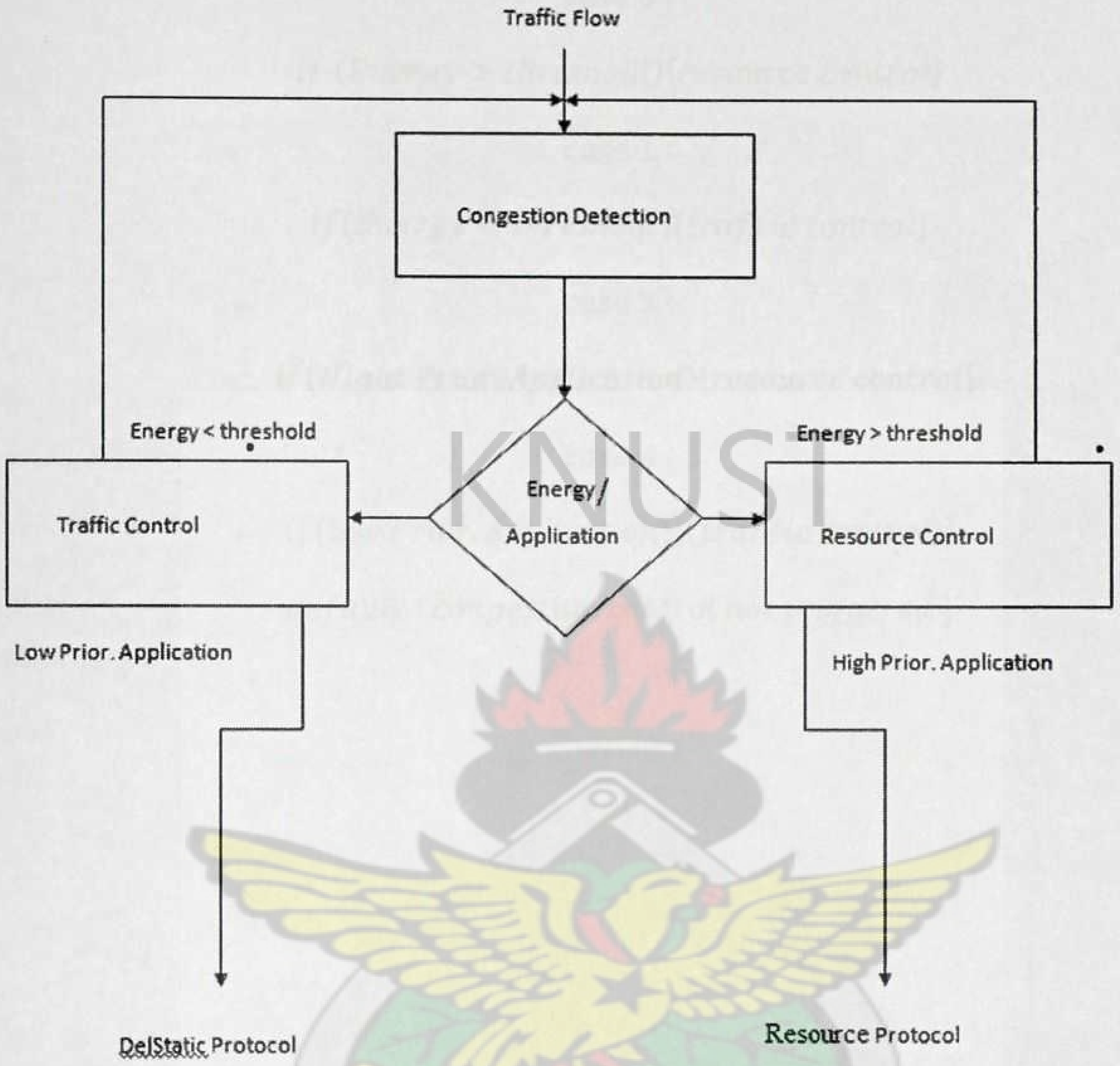


Figure 3.8 framework for Adaptive Protocol

From the framework diagram in Figure 3.8, congestion detection is done at the node level with a feedback sent to the sink to report congestion in the network. The sink uses two important parameters, Energy Remaining (*ER*) and Sensor Application Priority (*SP*) to determine the control to the sending node. Either Traffic Control (DelStatic) or Resource Control (DSR) is triggered by the sink depending on the algorithm below;

```
$switch\ (congestion\ control)\{$
$case\ 0:$
$if\ (Energy > threshold)\{resource\ control\}$
$case\ 1:$
$if(Energy < threshold)\{traffic\ control\}$
$case\ 3:$
$if(Hight\ Prior.Application)\{resource\ control\}$
$case\ 4:$
$if(low\ Prior.Application)\{(traffic\ control)\}$
$default:\ congestion\ control\ not\ triggered\ \}$
```

4 Simulation and Results

4.1 Simulation and Results

This chapter presents a detailed discussion on the simulation results. The discussion is based purely on the results generated by trace file and analysed using awk scripts, jTrana, and excel. The chapter is divided into three parts; DSR congestion control analysis using TADD protocols, traffic congestion control analysis using DelStatic protocol and ACCP simulation analysis.

It is important to mention that NS-2 is an open source framework, provides some extension for wireless sensor nodes and protocol. For the purpose of achieving our objective, needed node configuration and protocols are used from the default version of NS-2 with class and header file addition and modification of specific objects.

4.2 Simulation Environment

The simulator used to simulate the various protocols is NS-2 from Berkeley. NS-2 is the result of an on-going effort of research and development that is administrated by researchers at Berkeley. It is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols.

The simulator is written in C++ and a script language called OTcl. NS-2 uses OTcl interpreter towards the user. This means that the user writes an OTcl script that defines the network, the traffic in the network and which protocols it will use. This script is then used by NS-2 during simulations. The result of the simulations is an output trace file that can be used to do data processing and to visualize the simulation with a program called Network Animator (NAM). For this project, NS version 2.34 is used.

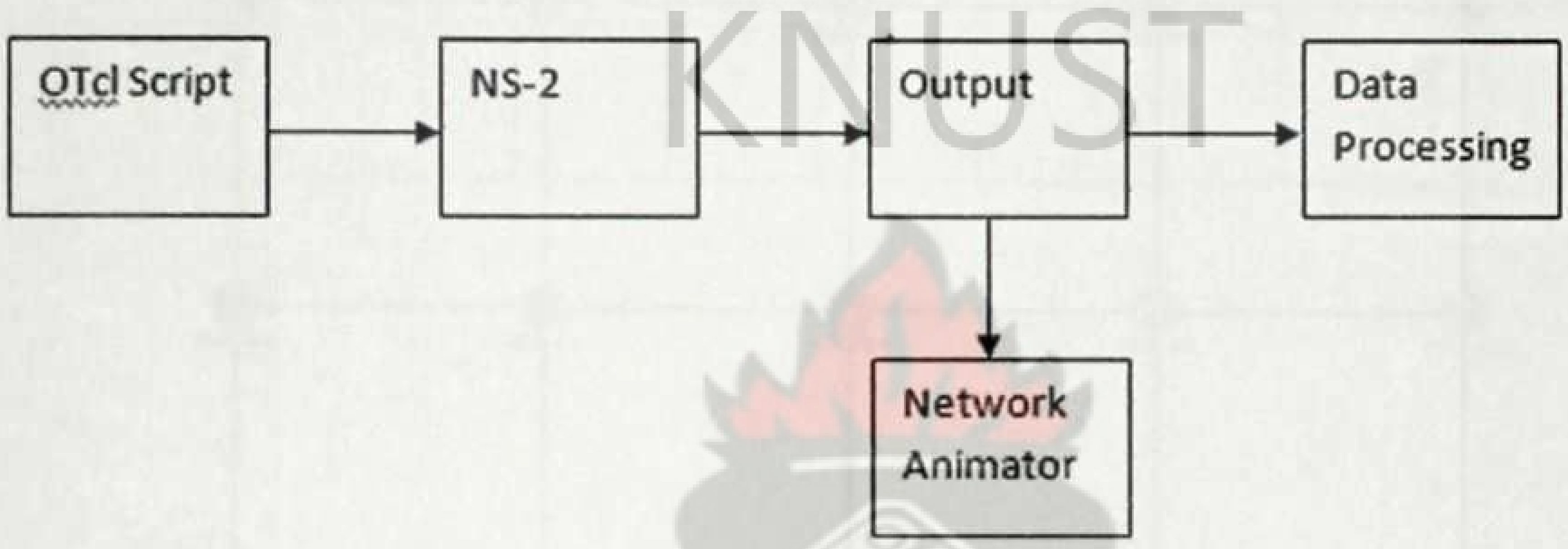


Figure 4.1 top level overview of input/output in Network Simulator – 2

4.3 PART 1 : TADD Protocol Simulation

In this section, simulation is conducted on Ad hoc On-demand Vector Protocol (AODV), Destination-Sequenced Distance Vector (DSDV), Dynamic Source Routing (DSR) and Temporarily Ordered Routing Algorithm (TORA) which forms the TADD protocols. The aim is to compare these routing protocols and their performance in terms of traffic bytes generated, end to end delay and throughput. The protocol with the highest metric value is used for the resource controlling part of ACCP.

4.3.1 Grid Topology [2source/1sink] – 25 nodes

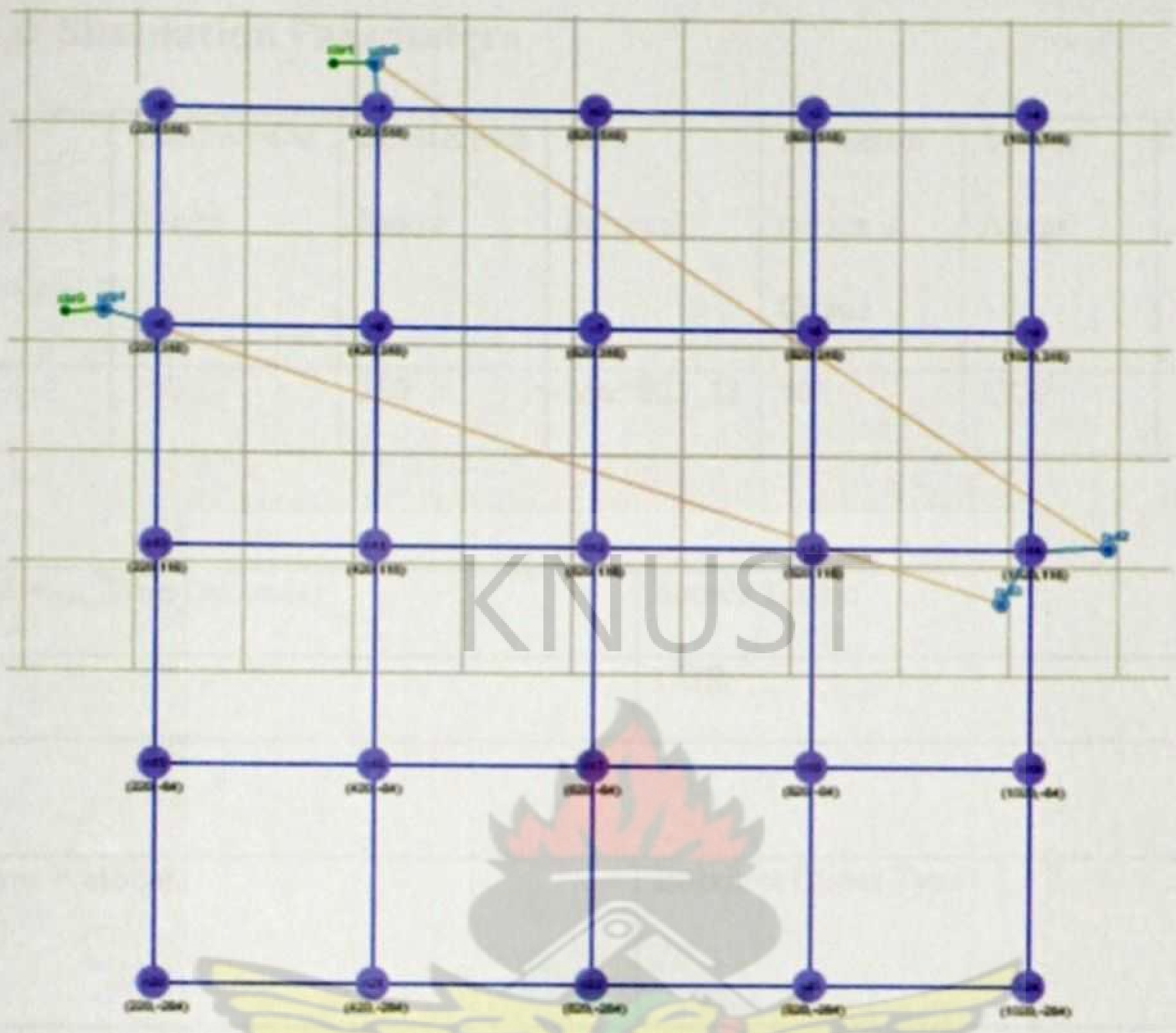


Figure 4.2 Grid Topology [2source/1sink]

In Figure 4.2, a grid topology of 25 nodes is generated with each node having the capacity to send data traffic. Nodes n1and n5 are the source nodes whiles node n14 is the sink node.

4.3.1.0 Simulation Parameters

No. of nodes	Transmission Range	Interference Range	Mac Protocol	Maximum packet in Queue	Source Agent	Sink Agent
25	250	550	Mac/802_11	50	UDP	null

Simulation Time (seconds)	Source Traffic
10	CBR

Ad-hoc Protocol	Interface Queue Type
AODV	Queue/DropTail/PriQueue
DSDV	Queue/DropTail/PriQueue
DSR	CMUPriQueue
TORA	Queue/DropTail/PriQueue

Table 4.1 Simulation Parameters

4.3.1.1 Traffic Bytes Generated

The traffic in bytes generated describes the various data source traffic, algorithm and protocols that helped in successful source to sink node data transfer.

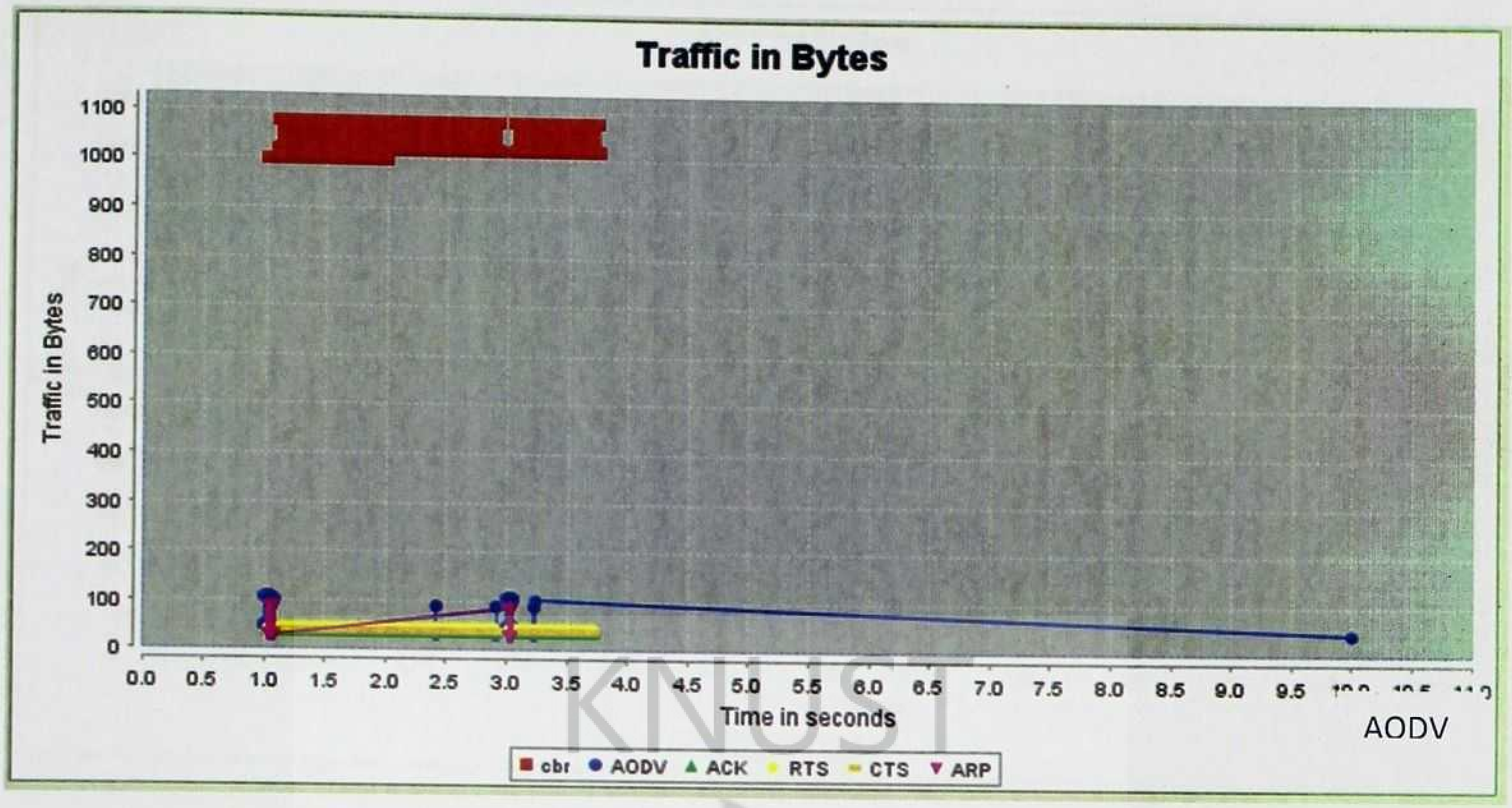


Figure 4.4 traffic in bytes generated by AODV protocol

In Figure 4.4: a high value of Constant Bandwidth Rate (CBR) traffic is generated at the initial period of simulation and lasted for a short interval. RTS/CTS of MAC layer is of importance during this high traffic period and effectively utilized in AODV. The Address Resolution Protocol (ARP) is also implemented in AODV traffic. ARP translates IP-addresses to hardware MAC addresses. This takes place before the packets are sent down to the MAC layer. The simulation lasted for 10 seconds.

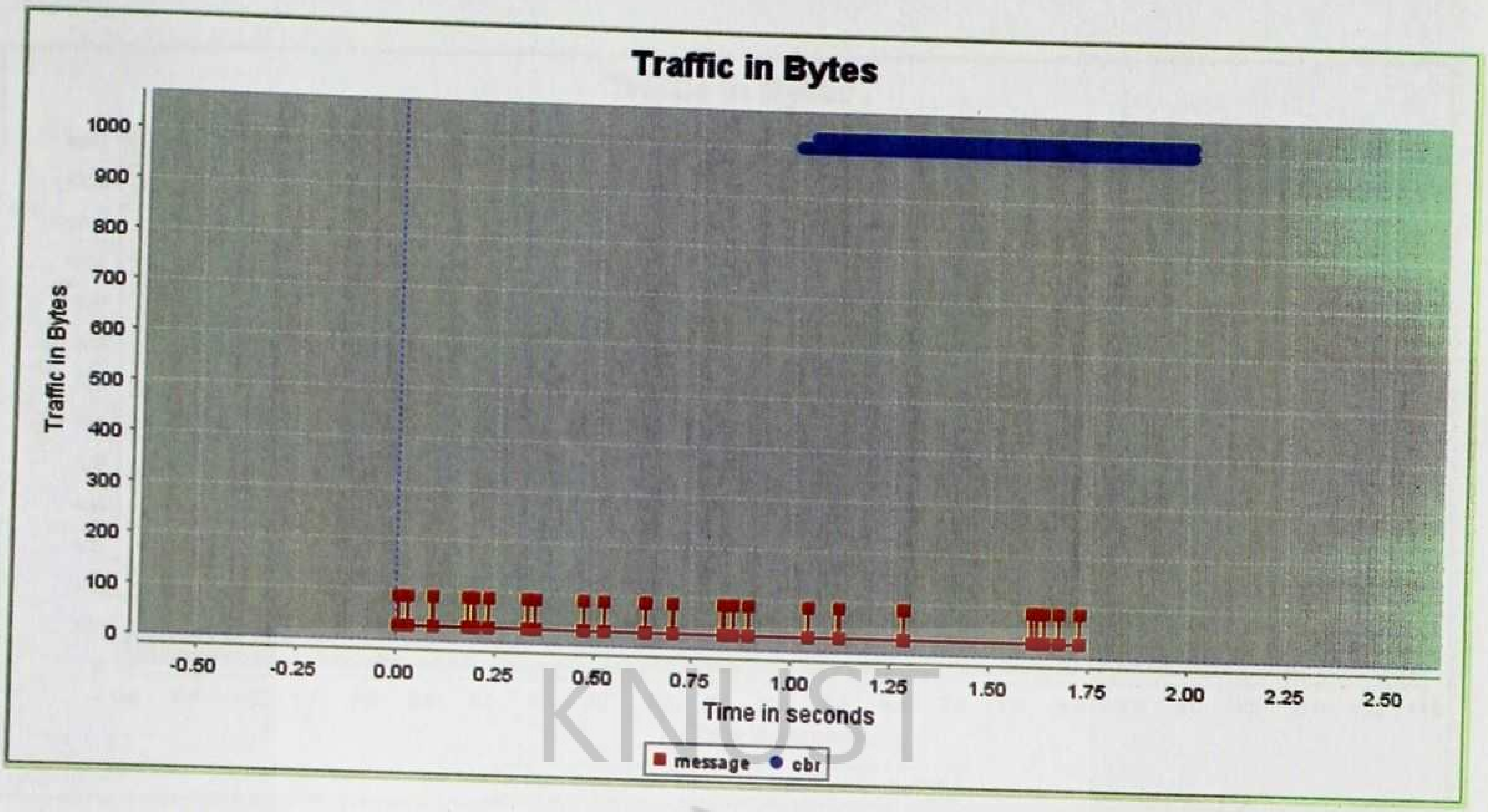


Figure 4.5 traffic in bytes generated by DSDV protocol

In Figure 4.5 DSDV protocol produced a low CBR source rates with a messaging algorithm .This shows that DSDV does not use MAC algorithm to effectively deliver the CBR traffic generated. It rather makes use of a messaging algorithm which is less efficient in congestion control. The simulation lasted for only 1.75 seconds which is an indication of severity of congestion leading to abrupt termination of simulation.

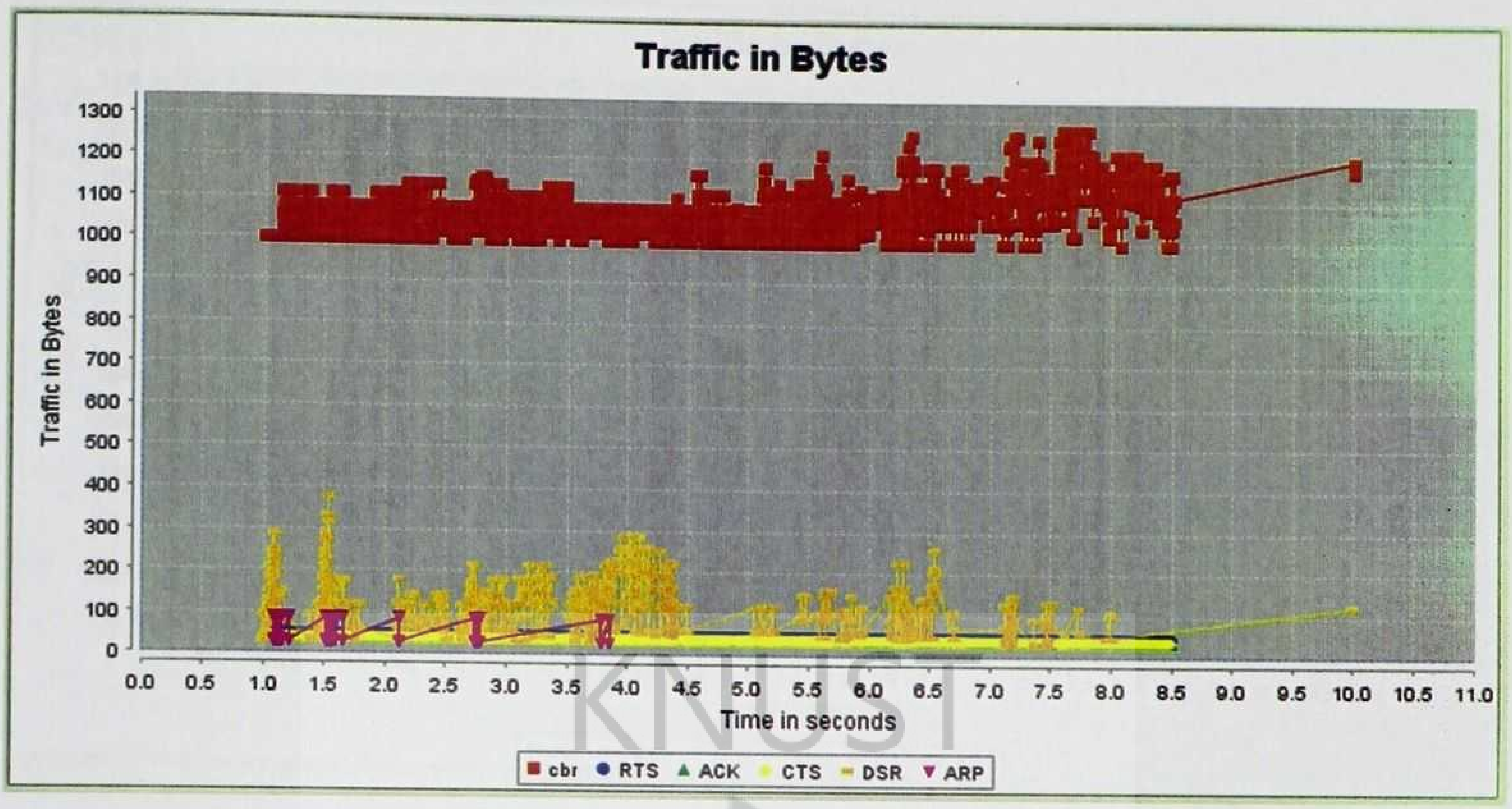


Figure 4.6 traffic in bytes generated by DSR protocol

In Figure 4.6, DSR protocol generated a high value of CBR traffic and lasted throughout the simulation time. The high traffic indicates the ability of the source nodes to send CBR data without much congestion interference. RTS/CTS algorithm, ARP and ACK were efficiently utilized during CBR source transfer. As shown in the graph, DSR protocol is efficient in generating and transmitting CBR data compared to the other TADD protocols. The simulation lasted for 9 seconds of simulation time.

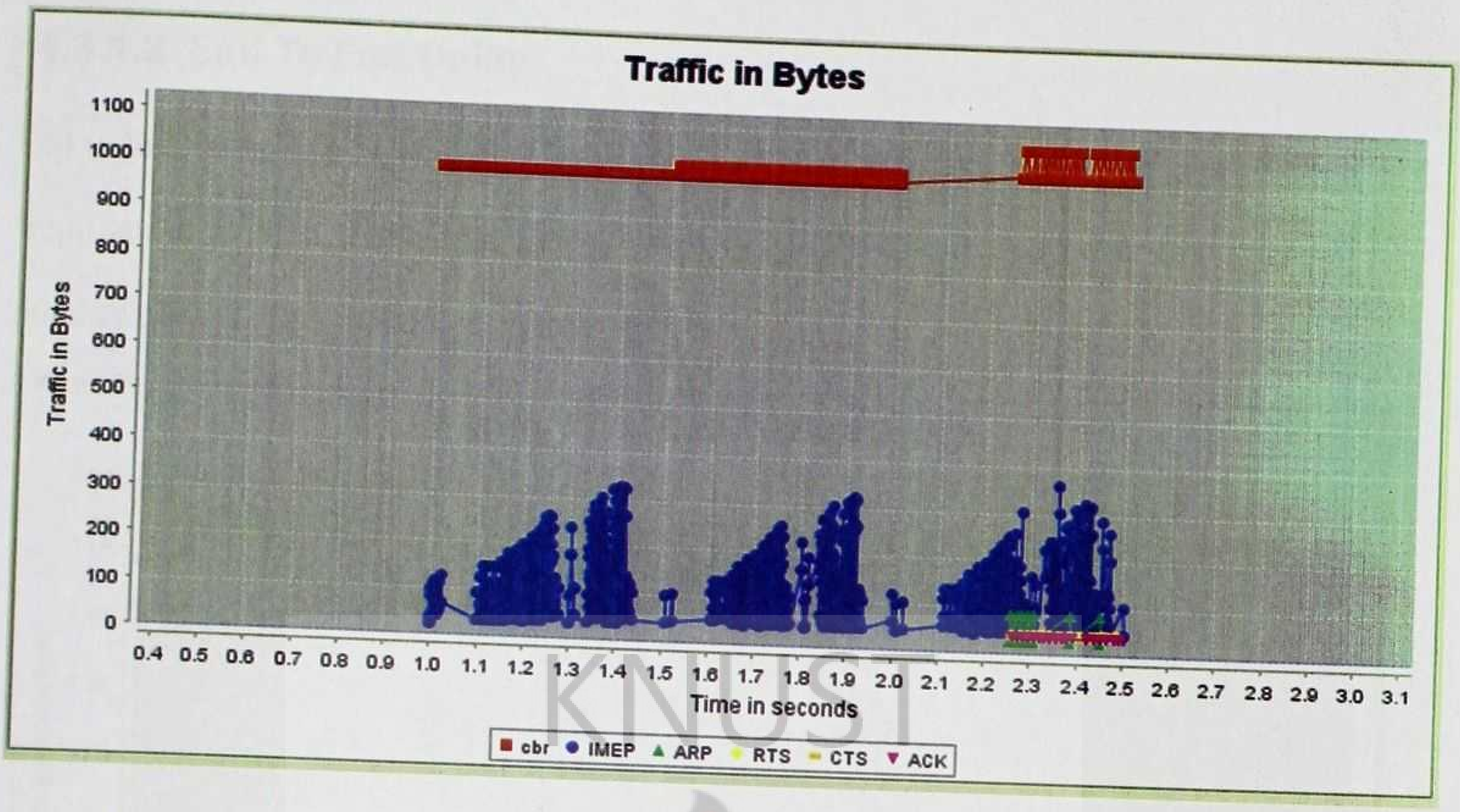


Figure 4.7 traffic in bytes generated by TORA protocol

In Figure 4.7, TORA generated CBR traffic which lasted for few seconds. TORA made use of IMEP (Internet MANET Encapsulation) to provide reliable delivery of route-messages and to inform the routing protocol of any changes of links to its neighbors. Other relevant algorithm and protocol for traffic efficiency were utilized. The simulation lasted for 1.5 seconds, which is an indication of data transmission failure.

4.3.1.2 End To End Delay

End to End delay refers to the time taken for a packet to be transmitted across a network from source to sink. It is significant to note that, end to end delay depends on the number of CBR data packets generated.

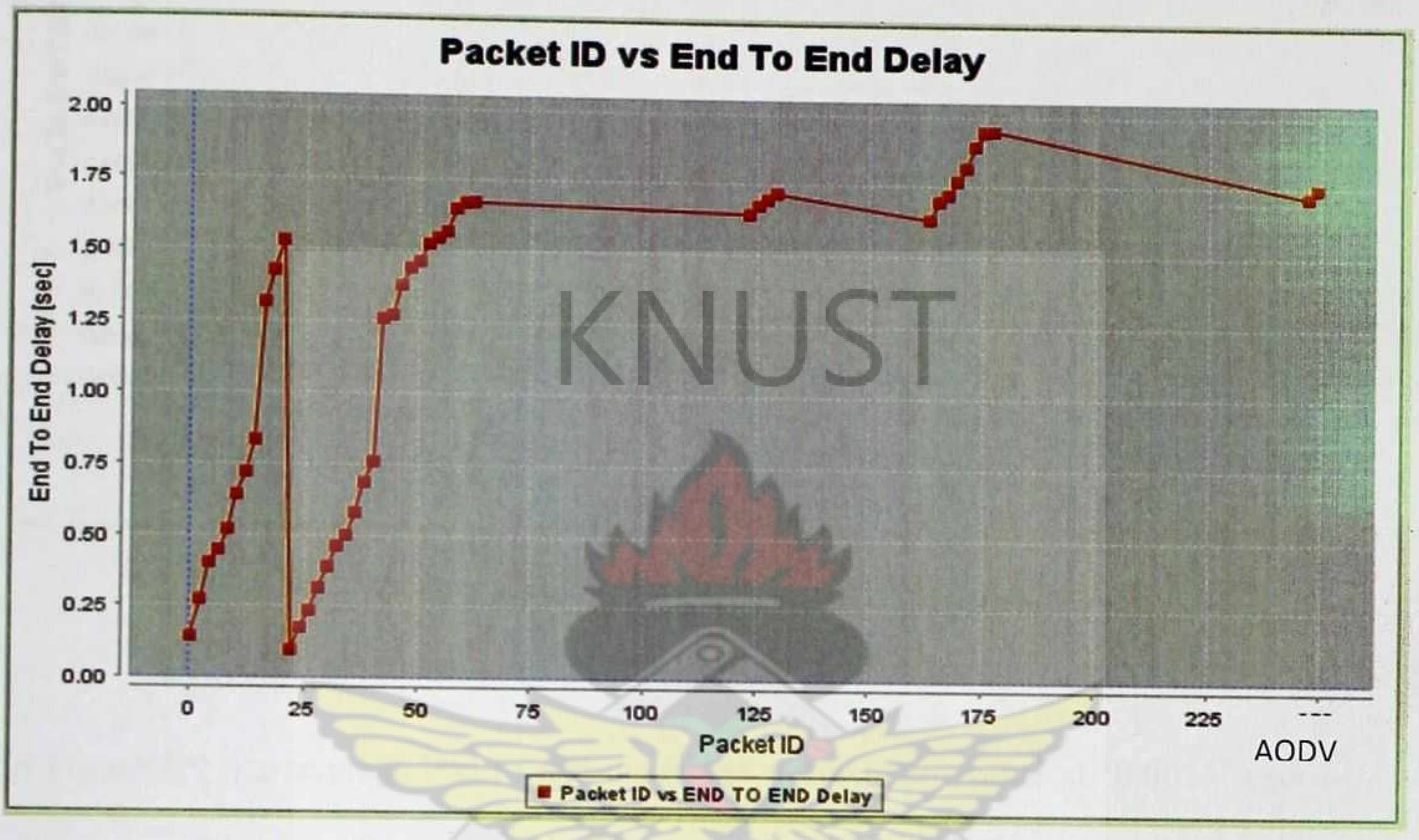


Figure 4.8 End to End Delay of AODV Protocol

In Figure 4.8, the average end to end delay has a maximum value at '1.98' and a minimum at '0.125'. Most of the CBR data packet got transmitted during this range. End to End delay is proportional to the CBR source data traffic. Having a low delay value is as a result of severe congestion faced by AODV protocol.

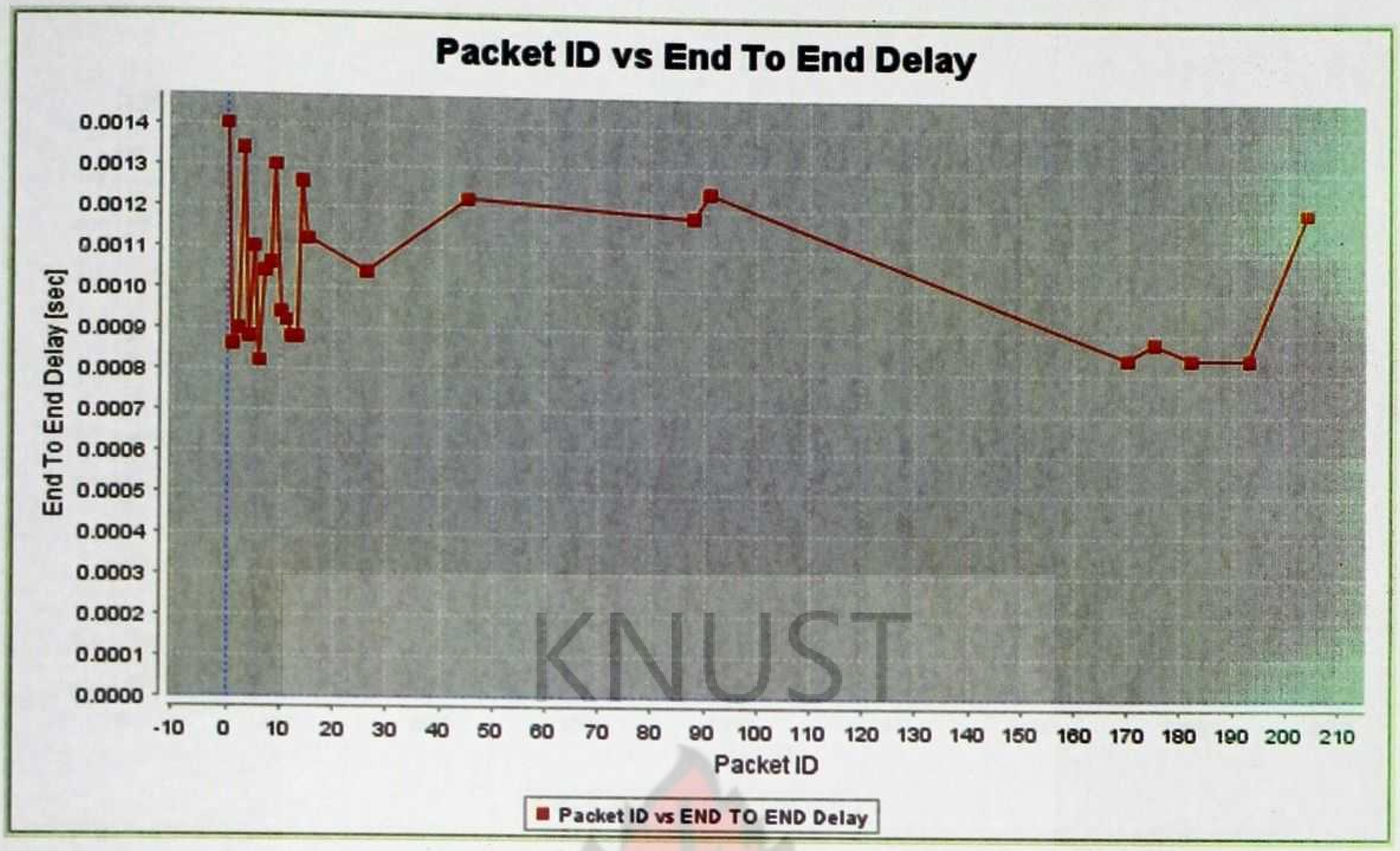


Figure 4.9 End to End Delay of DSDV Protocol

In Figure 4.9, the average end to end delay has a maximum value at '0.0014' and a minimum at '0.0008'. Though this value compared to that in Figure 4.8 is significantly low, it is also an indication of the lost in CBR data packet during congestion.

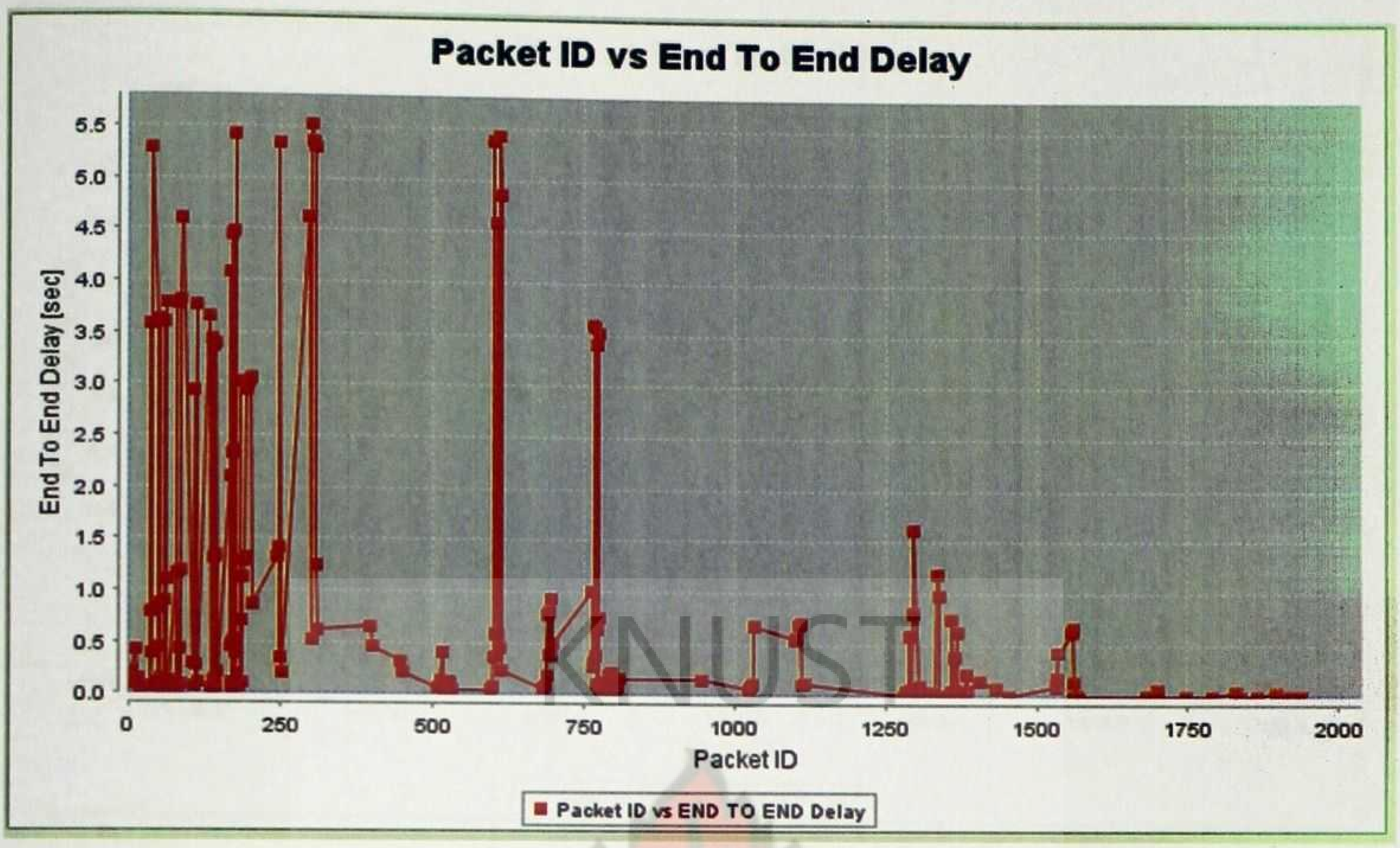


Figure 4.10 End to End Delay of DSR Protocol

In Figure 4.10, the average end to end delay has a maximum value at '5.5' and a minimum at '0.0'. This is a positive result from DSR protocol. Though 5.5 seconds is on the high side, it is an indication that CBR source data is transmitted over a significant period. Congestion is effectively dealt with during the transmission process.

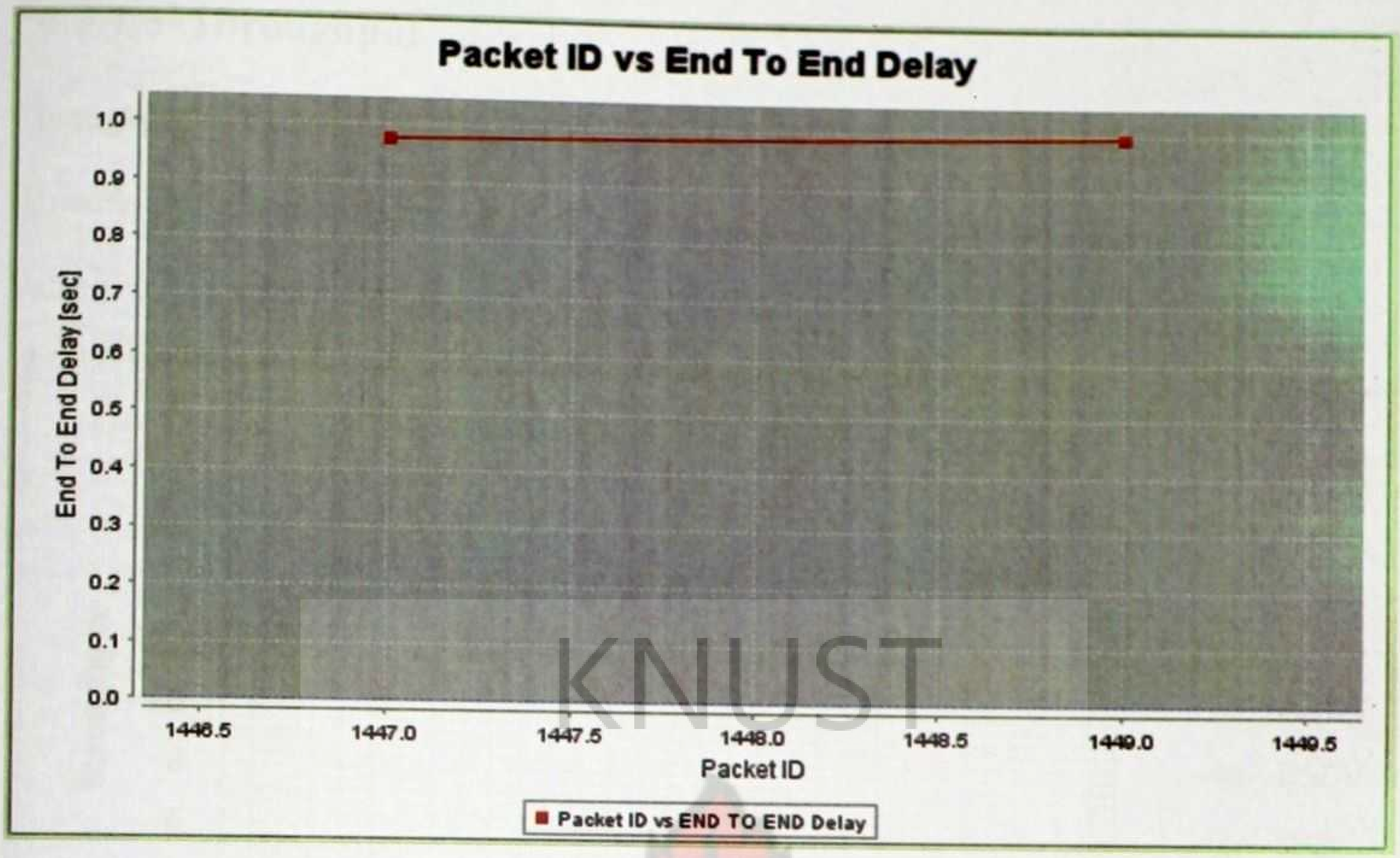


Figure 4.11 End to End Delay of TORA Protocol

In Figure 4.11, TORA protocol behaved poorly in congested networks with persistent shortest path to sink not available due to low CBR traffic. This confirms the questioning of the link reversal algorithm implemented by TORA. A linear graph is generated with maximum end to end delay value at "1.0" and minimum at "0.98".

4.3.1.3 Throughput

Throughput is the average rate of successful packet delivered to the sink in a sensor network. A high throughput value is a good measure of congestion control but this value is dependent on the CBR data packets generated by the source nodes.

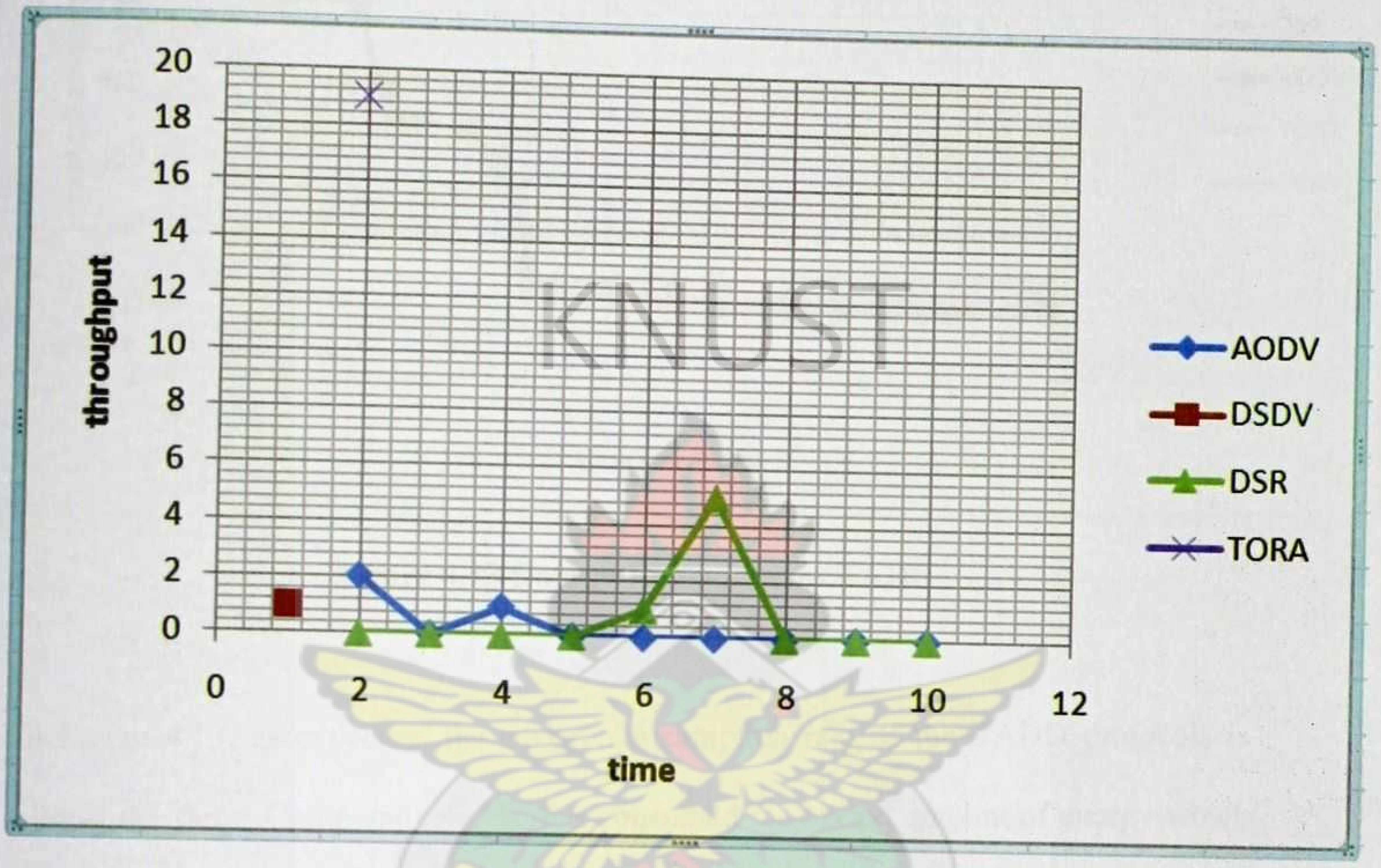


Figure 4.12 Throughput of TADD

In Figure 4.12, TORA has the highest throughput value at 2 seconds, followed by DSR at 7 seconds and AODV at 2 seconds.

4.3.1.4 Energy Parameters

```
Set opt(initialenergy)  900          ;# Initial energy in Joules
-rxPower 0.3 \                       ;# power consumption in state (watt)
-txPower 0.6 \                       ;# power consumption in sleep state (watt)
```

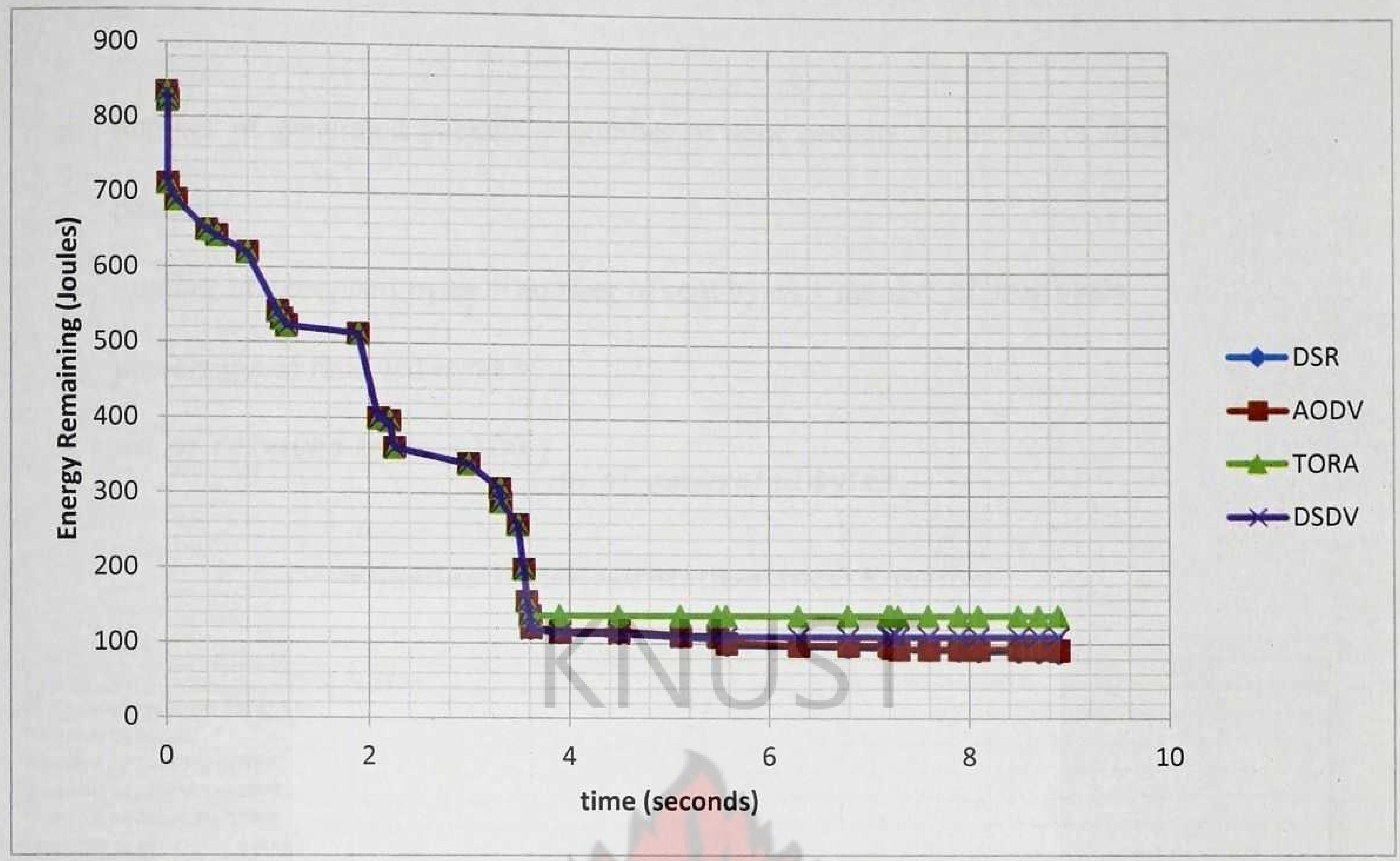


Figure 4.13 Energy Remaining of TADD

In Figure 4.13, as expected, the energy consumption rate of the TADD protocols is almost the same. Comparatively, TORA consumed the lowest amount of energy which is slightly above the 100J mark. The increase in throughput performance adversely affects the energy consumption of TADD protocols.

4.3.1.5 Detailed Summary

The detailed summary generated by JTrana simulation software gave a total summary of the results. The summary is done for only AODV and DSR protocol due to their significant end to end and throughput values.

The following equation emerged from the summary:

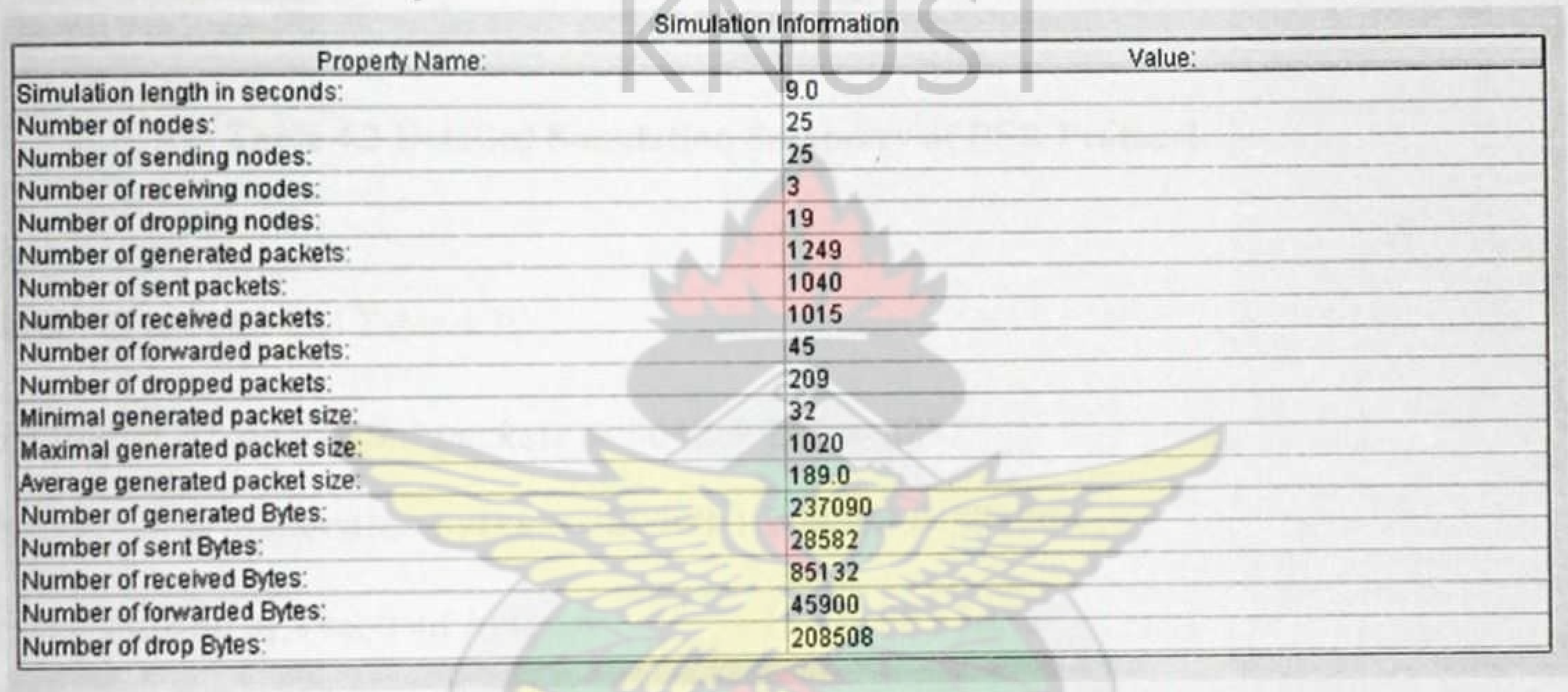
a. number of generated packets = number of sent packets + number of dropped packets

b. number of generated bytes = number of sent bytes + number of drop bytes

c. percentage of received bytes =

$$no.of\ received\ bytes * 100 \Big/_{no.of\ generated\ bytes}$$

Equation 1: Derived Mathematical Equation



Simulation Information

Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	25
Number of sending nodes:	25
Number of receiving nodes:	3
Number of dropping nodes:	19
Number of generated packets:	1249
Number of sent packets:	1040
Number of received packets:	1015
Number of forwarded packets:	45
Number of dropped packets:	209
Minimal generated packet size:	32
Maximal generated packet size:	1020
Average generated packet size:	189.0
Number of generated Bytes:	237090
Number of sent Bytes:	28582
Number of received Bytes:	85132
Number of forwarded Bytes:	45900
Number of drop Bytes:	208508

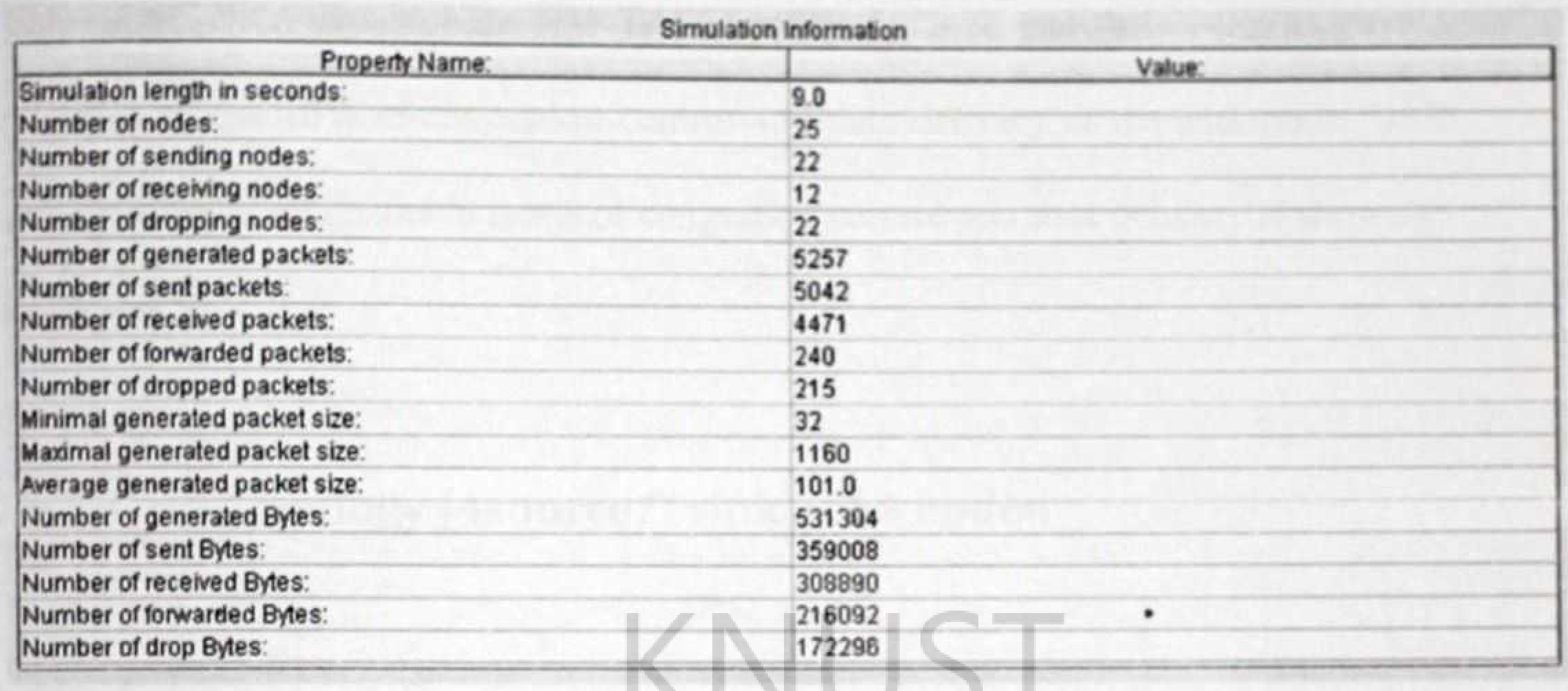
Table 4.2 Detailed Simulation Summary of AODV Protocol

From Equation 1 and Table 4.2:

$$a.number\ of\ generated\ packets = 1040 + 209 = 1249$$

$$b.number\ of\ generated\ bytes = 28582 + 208508 = 237090$$

$$c.percentage\ of\ received\ bytes = \frac{85132}{237090} * 100 = 35.907\ \%$$



Simulation Information

Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	25
Number of sending nodes:	22
Number of receiving nodes:	12
Number of dropping nodes:	22
Number of generated packets:	5257
Number of sent packets:	5042
Number of received packets:	4471
Number of forwarded packets:	240
Number of dropped packets:	215
Minimal generated packet size:	32
Maximal generated packet size:	1160
Average generated packet size:	101.0
Number of generated Bytes:	531304
Number of sent Bytes:	359008
Number of received Bytes:	308890
Number of forwarded Bytes:	216092
Number of drop Bytes:	172298

Table 4.3 Detailed Simulation Summary of DSR Protocol

From Equation 1 and Table 4.3:

$$a. number\ of\ generated\ packets = 5042 + 215 = 5257$$

$$b. number\ of\ generated\ bytes = 359008 + 208508 = 531304$$

$$c. percentage\ of\ received\ bytes = \frac{308890}{531304} * 100 = 58.138\ \%$$

4.3.1.6 GRID Topology (25 nodes) Results Summary

From the simulation results, AODV and DSR protocols gave significant figures in terms of percentage of received bytes. Though a sent byte can be retransmitted making it a negative value, a received byte is the measure of successful data packets delivered to the sink node. AODV from the simulation result has a percentage data delivery of 35.907% . Meaning that, 64.093% of generated packet by the source is lost through congestion and packet retransmission. On the other hand, DSR protocol from the simulation results has data delivery percentage of 58.138% .

A significant 41.862% data is loss through congestion and data retransmission. Though DSDV and TORA protocol are also TADD protocols, their simulation results gave a low value below 10% in congestion control and data delivery at the sink node. DSR from the simulation results in terms of congestion control and data delivery is shown to be efficient.

4.3.2 Grid Topology [4source/1sink] – 64 nodes

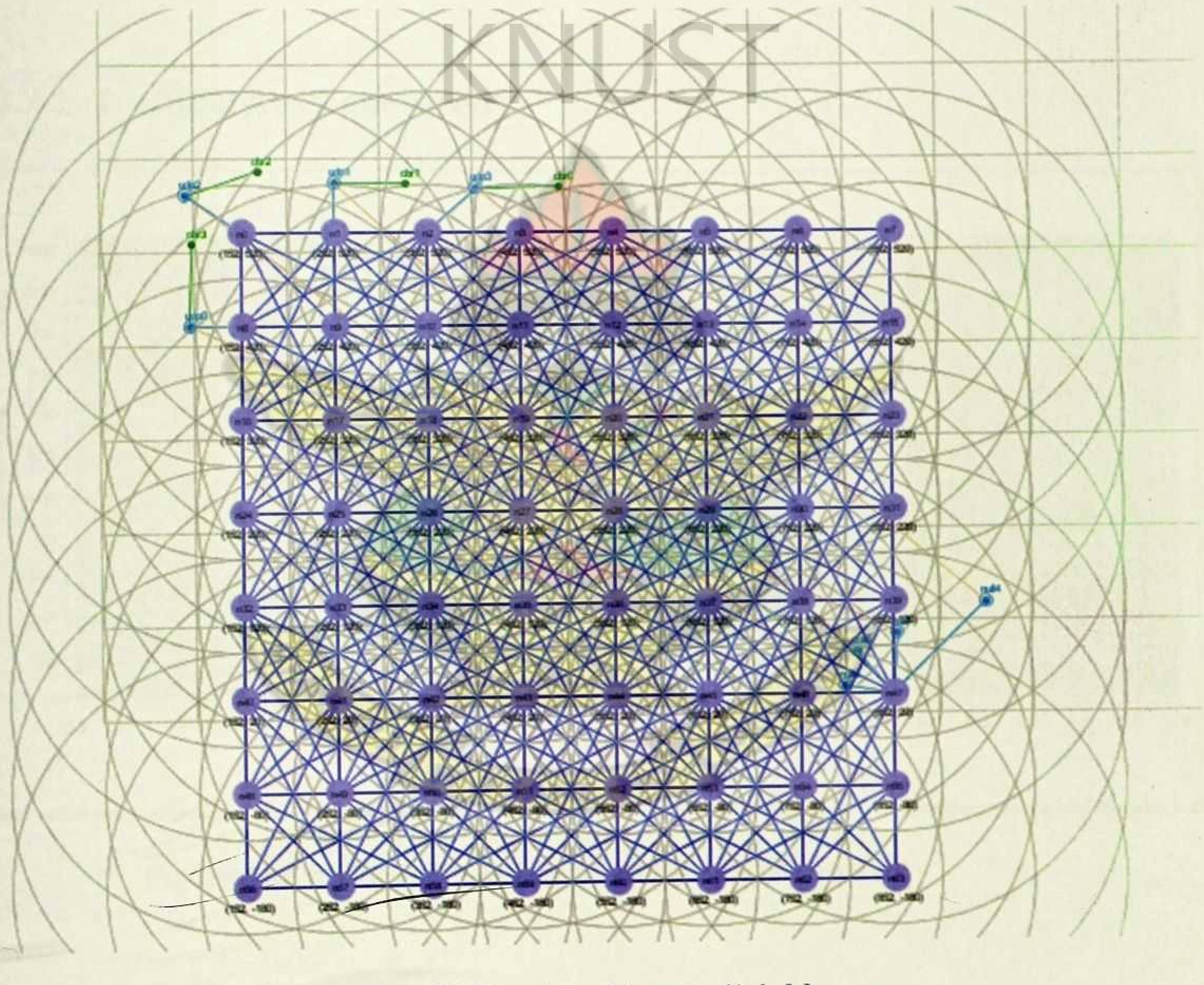


Figure 4.14 Grid Topology [4source/1sink]

In Figure 4.14, a grid topology of 64 nodes is generated with each node having the capacity to send data traffic. The importance is to test TADD protocols congestion control capability with increased intermediate nodes. Node n0, n1, n2 and n8 are the source nodes whiles node n47 is the sink node.

4.3.2.0 Simulation Parameters

Same simulation parameter in Table 4.1.

4.3.2.1 Traffic Bytes Generated

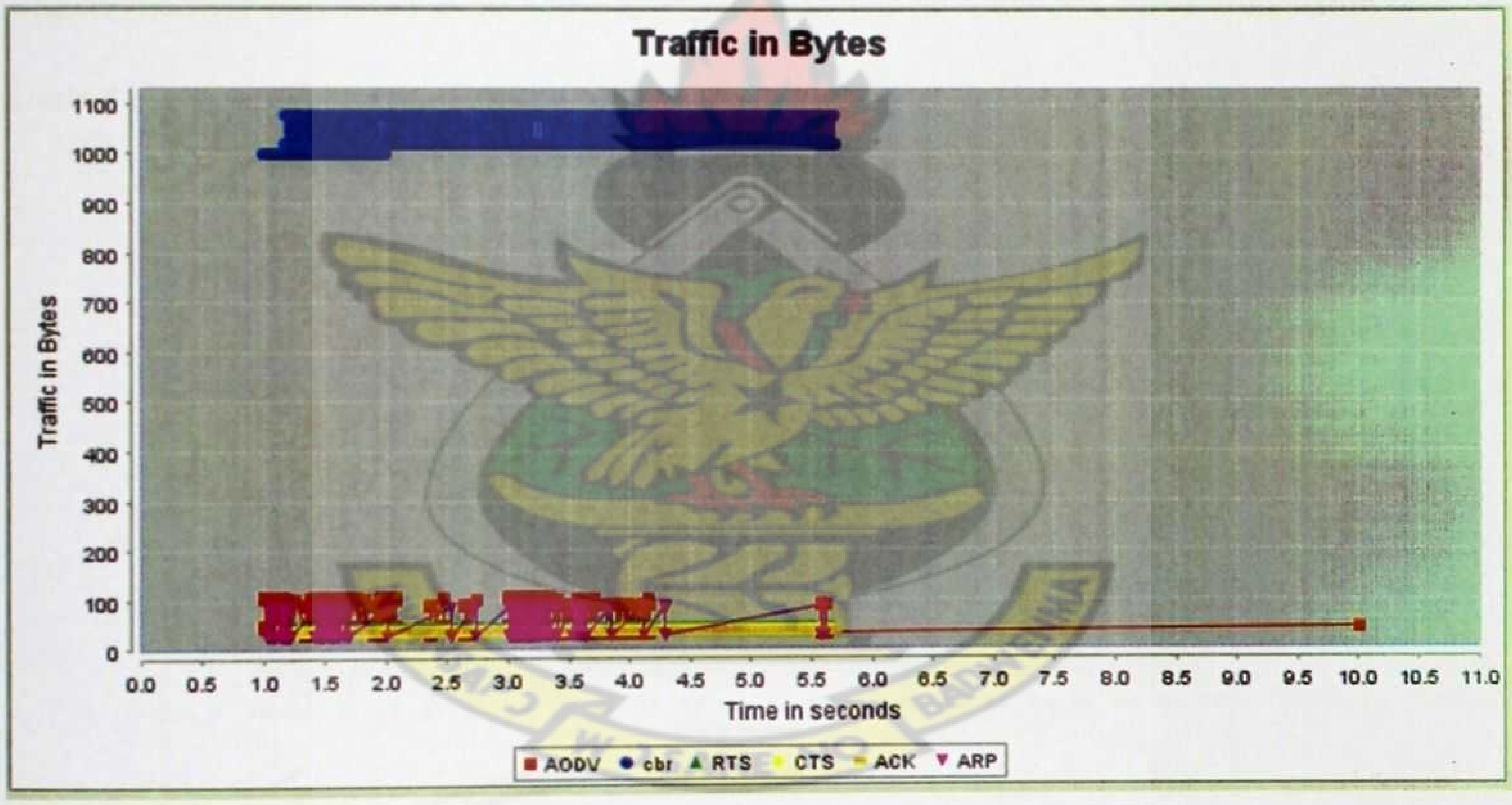


Figure 4.15 traffic in bytes generated by AODV protocol

The traffic bytes generated is similar to that of Figure 4.4 above but the CBR traffic generated by source nodes lasted for 4.5 seconds of simulation time. The increase is as a result of the additional source nodes involved in packet generation.

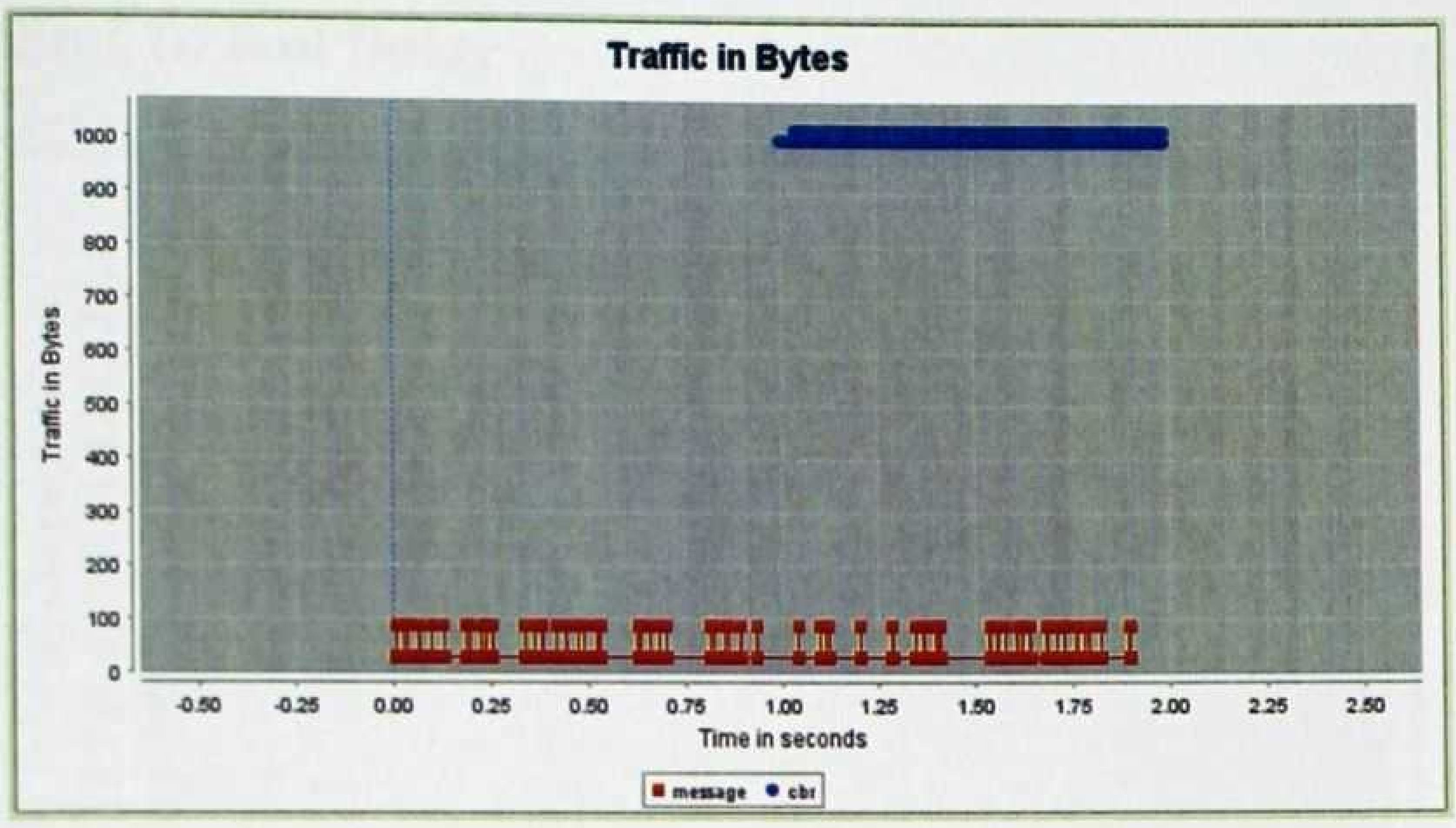


Figure 4.16 traffic in bytes generated by DSDV protocol

The traffic byte generated is the same as Figure 4.5. The absence of CSMA congestion control alleviation algorithm makes congestion dominant in DSDV protocol.

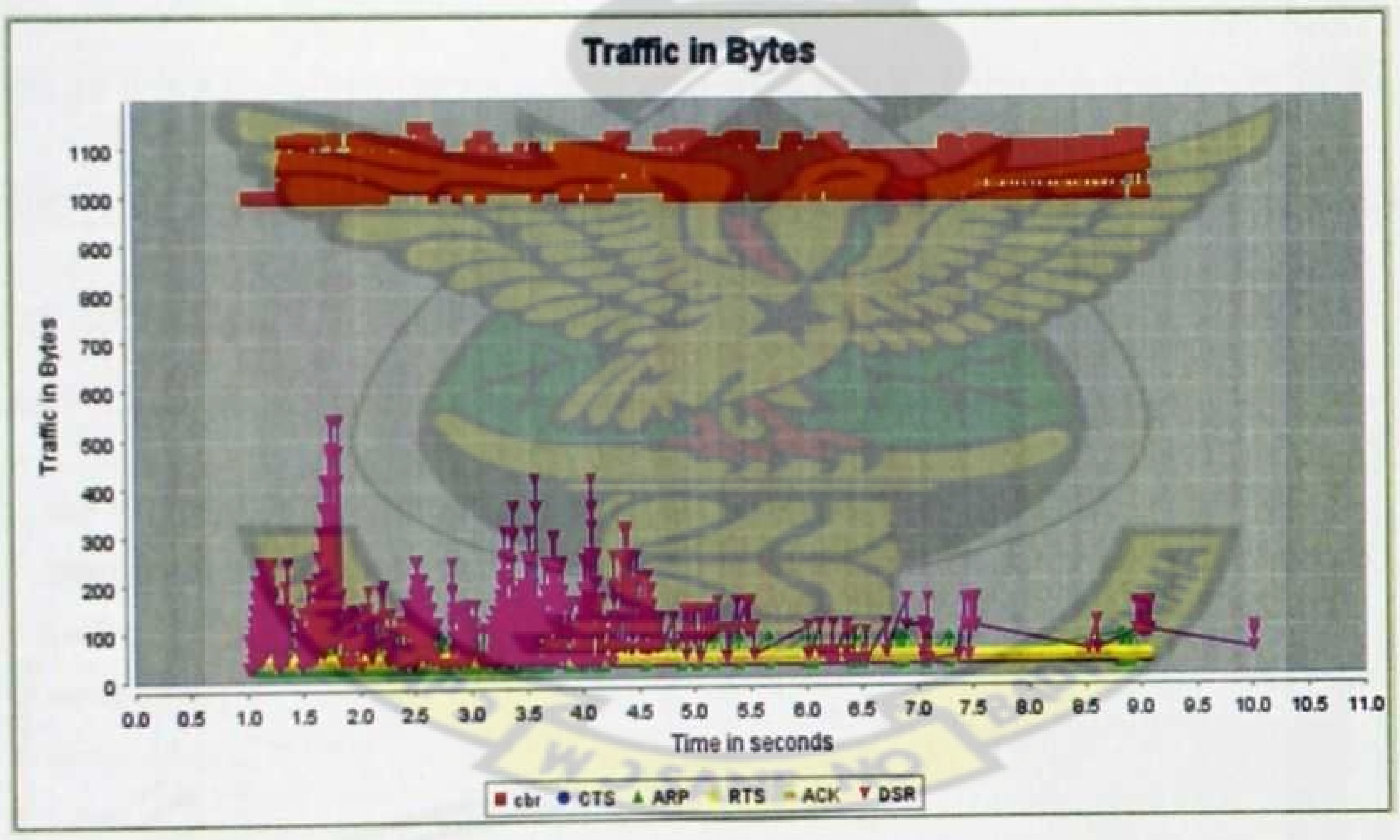


Figure 4.17 traffic in bytes generated by DSR protocol

As expected, CBR traffic for DSR lasted for 8 seconds of the simulation time. The traffic pattern is similar to that of Figure 4.6 but with a higher CBR data traffic due to additional source nodes.

4.3.2.2 End To End Delay

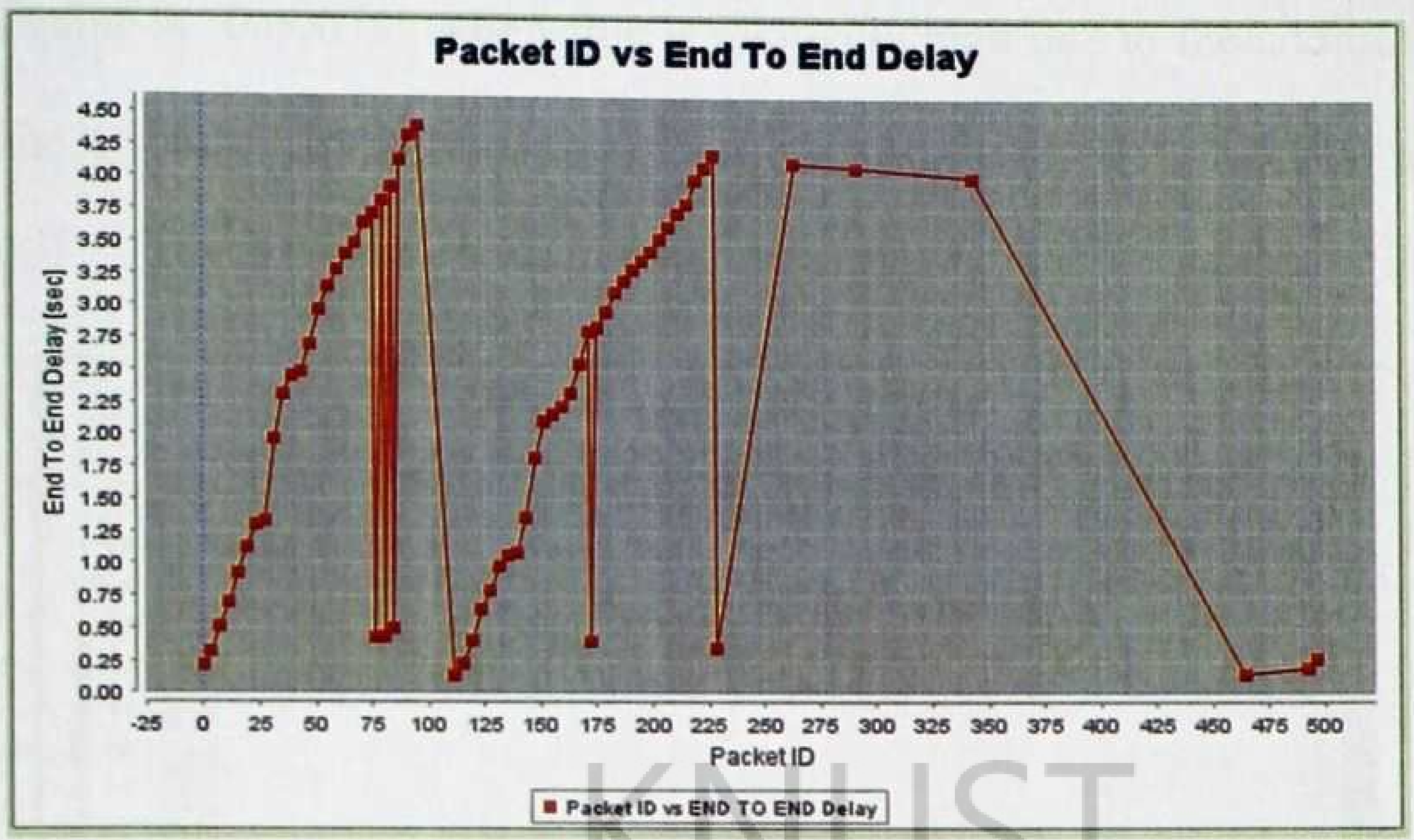


Figure 4.18 End to End Delay of AODV Protocol

In Figure 4.18, the average end to end delay is maximal at '4.50' and minimal at '0.125'. This is a 300% increase as compared to AODV 25 node topology in Figure 4.8. This increase is as a result of the additional CBR traffic generated by the source nodes and the delay for the packet to reach the sink.

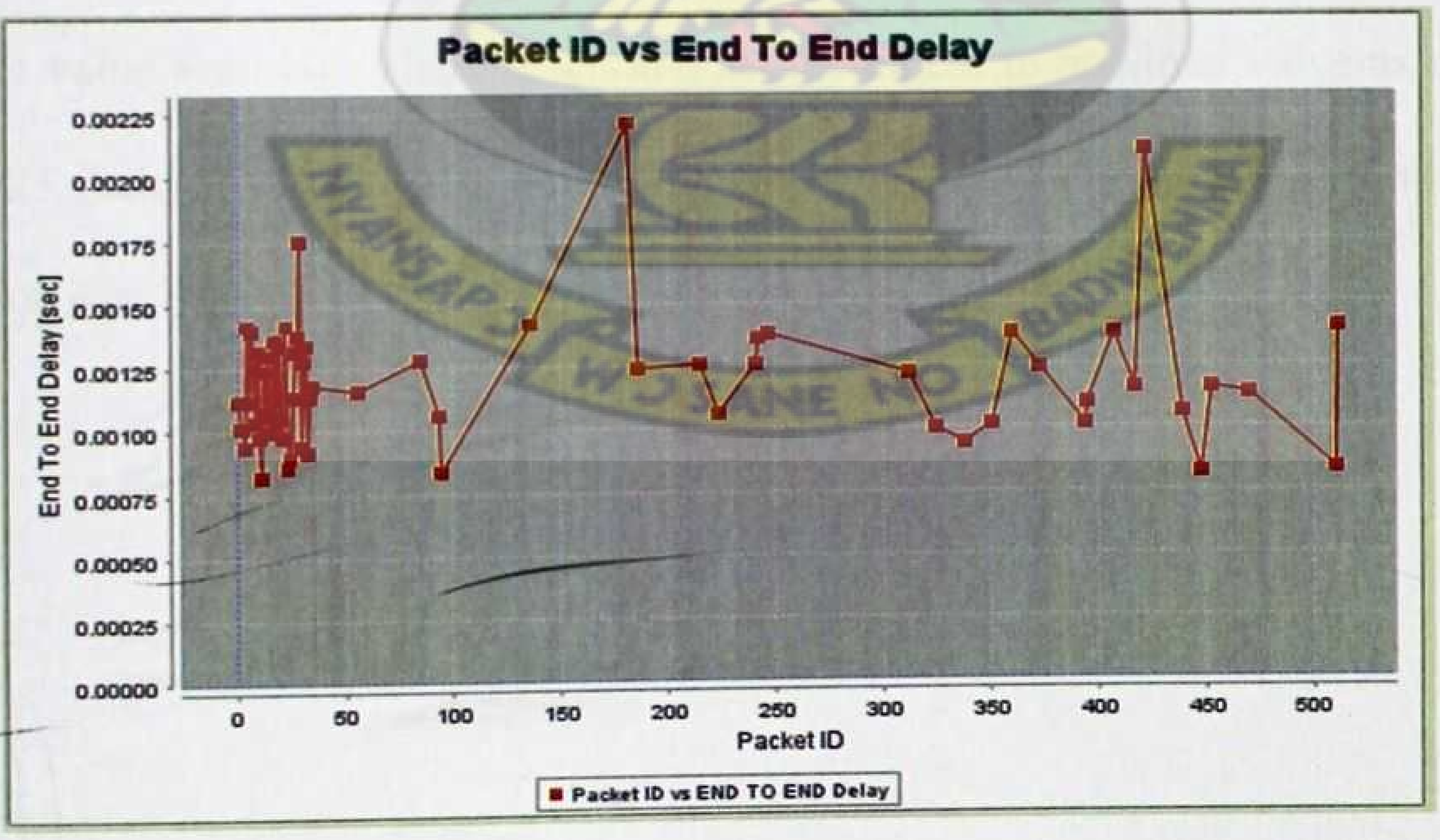


Figure 4.19 End to End Delay of DSDV Protocol

In Figure 4.19, the average end to end delay has a maximum value at '0.00225' and a minimum value at '0.00075'. The value is relatively high due to the addition of source nodes in the 64 node topology compared to the previous DSDV 25 node topology in Figure 4.9

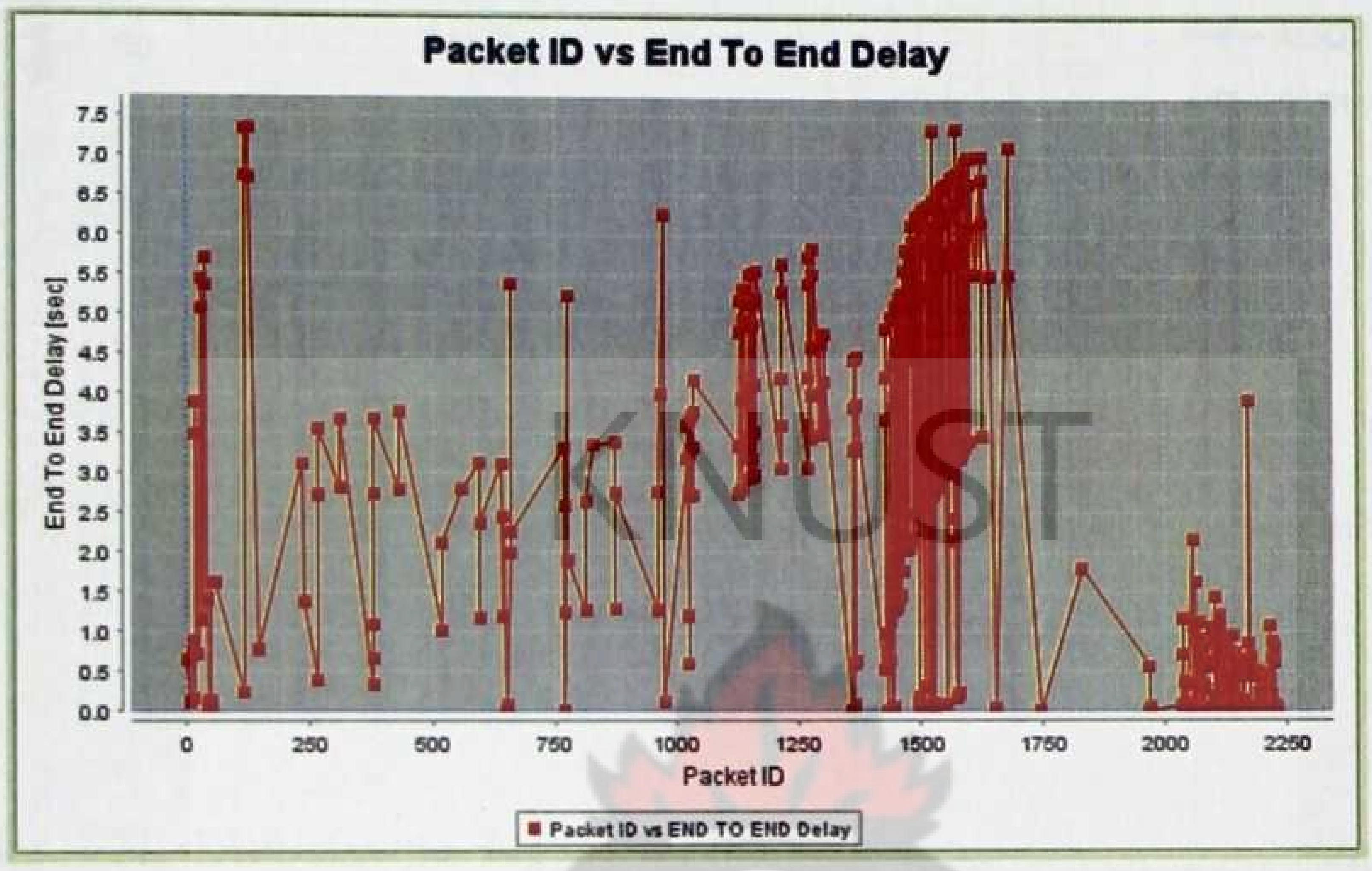


Figure 4.20 End to End Delay of DSR Protocol

In Figure 4.20, the average end to end delay has a maximum value at '7.5' and a minimum value at '0.00'. There is an obvious increment in maximal value as compared to DSR 25 node of Figure 4.10. Additional source nodes have the capacity to generate more CBR data traffic.

4.3.2.3 Throughput

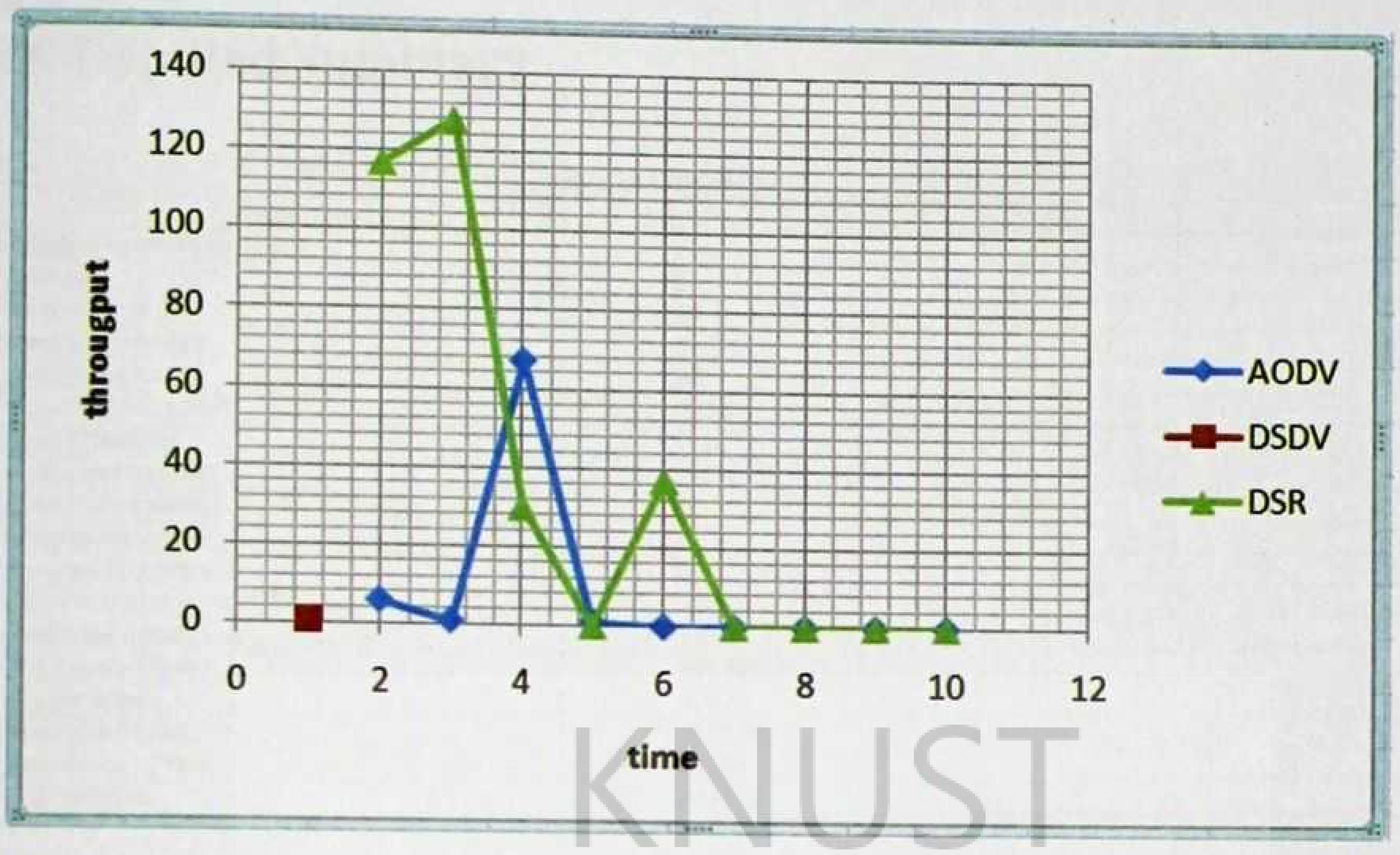
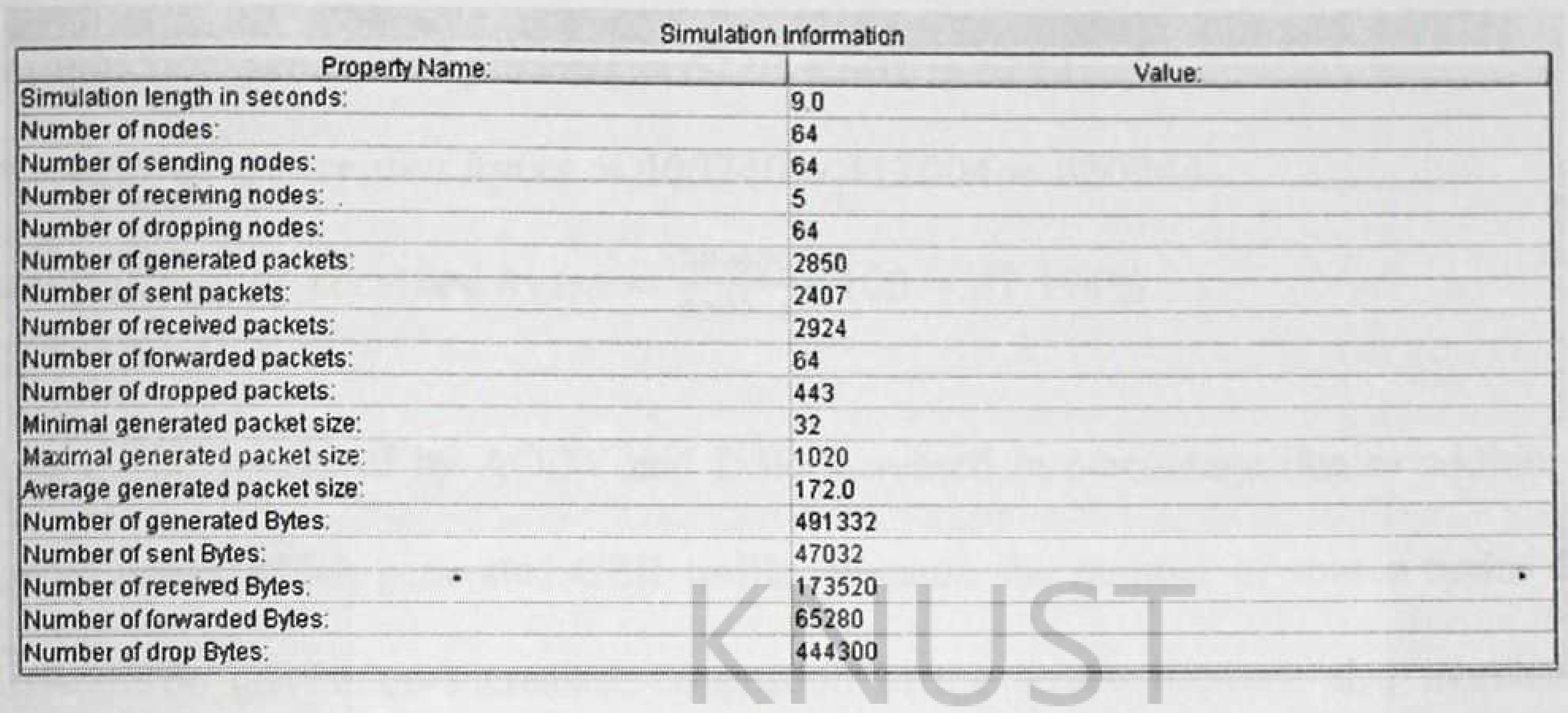


Figure 4.21 Throughput of TADD

From Figure 4.21, DSR has the highest throughput value which decreased at time 5 seconds and increase again before decreasing to zero at time 7 seconds. This is followed by AODV protocol with the second highest throughput value and DSDV. TORA is not mentioned here because of its low throughput value which is insignificant to congestion control.

4.3.2.4 Detailed Summary



Simulation Information

Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	64
Number of sending nodes:	64
Number of receiving nodes:	5
Number of dropping nodes:	64
Number of generated packets:	2850
Number of sent packets:	2407
Number of received packets:	2924
Number of forwarded packets:	64
Number of dropped packets:	443
Minimal generated packet size:	32
Maximal generated packet size:	1020
Average generated packet size:	172.0
Number of generated Bytes:	491332
Number of sent Bytes:	47032
Number of received Bytes:	173520
Number of forwarded Bytes:	65280
Number of drop Bytes:	444300

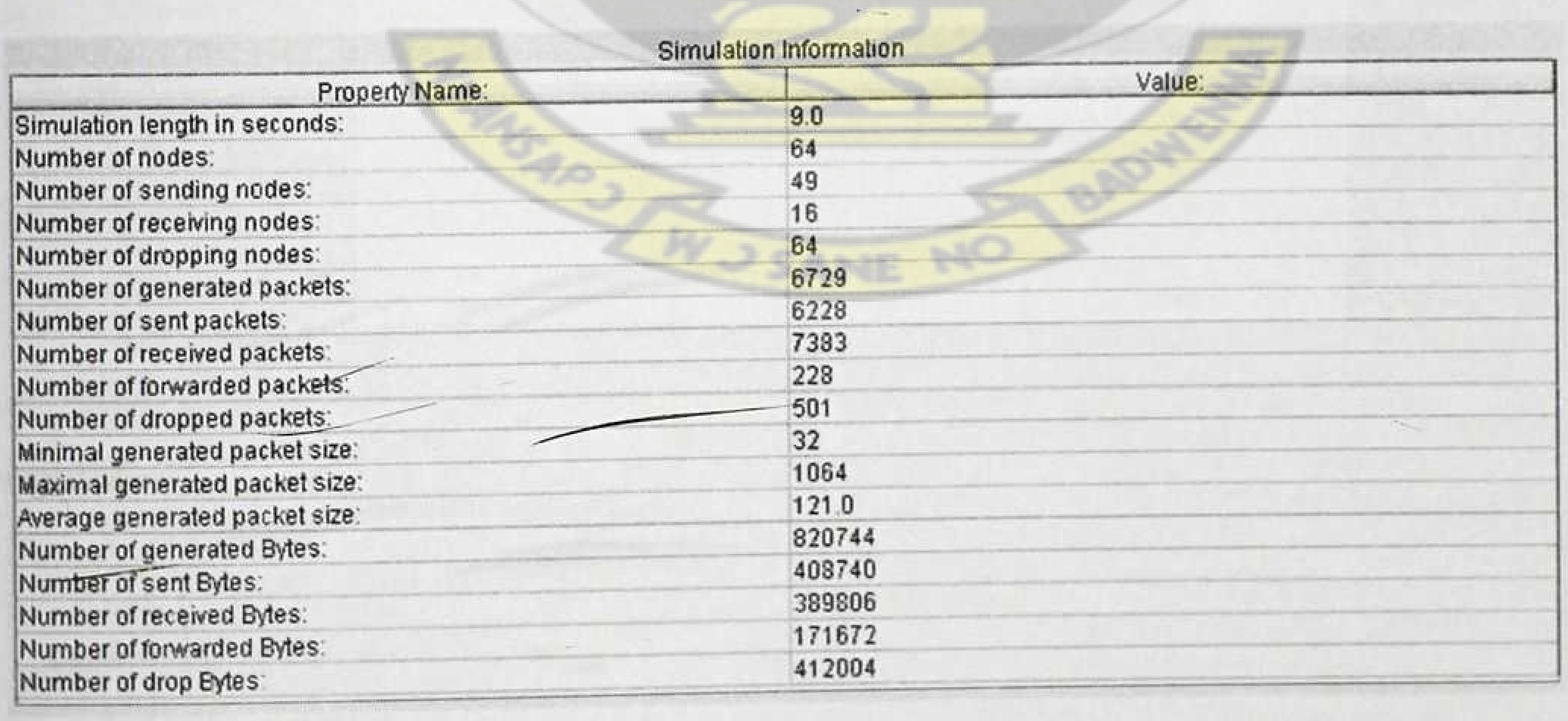
Table 4.4 Detailed Simulation Summary of AODV Protocol

From Equation 1 and Table 4.4:

$$a. number\ of\ generated\ packets = 2407 + 443 = 2850$$

$$b. number\ of\ generated\ bytes = 47032 + 444300 = 491332$$

$$c. percentage\ of\ received\ bytes = \frac{173520}{491332} * 100 = 35.316\ \%$$



Simulation Information

Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	64
Number of sending nodes:	49
Number of receiving nodes:	16
Number of dropping nodes:	64
Number of generated packets:	6729
Number of sent packets:	6228
Number of received packets:	7383
Number of forwarded packets:	228
Number of dropped packets:	501
Minimal generated packet size:	32
Maximal generated packet size:	1064
Average generated packet size:	121.0
Number of generated Bytes:	820744
Number of sent Bytes:	408740
Number of received Bytes:	389806
Number of forwarded Bytes:	171672
Number of drop Bytes:	412004

Table 4.5 Detailed Simulation Summary of DSR Protocol

From Equation 1 and Table 4.5:

$$a. number\ of\ generated\ packets = 6228 + 501 = 6729$$

$$b. number\ of\ generated\ bytes = 408740 + 412004 = 820744$$

$$c. percentage\ of\ received\ bytes = \frac{389806}{820744} * 100 = 47.494\%$$

Data traffic generated by AODV and DSR increased in percentage due to additional source nodes which generated CBR traffic. Because the number of source nodes for CBR traffic generation increased, congestion in the sensor network also increased. TORA routing protocol gave no results and was not used in this part of simulation. DSDV protocol also gave unusual results. DSR in terms of data delivery is 47.494% compared to AODV data delivery of 35.316%. Data lost due to congestion is 52.506% and 64.684% respectively.

4.3.3 Random Topology [3source/1sink] – 25 nodes

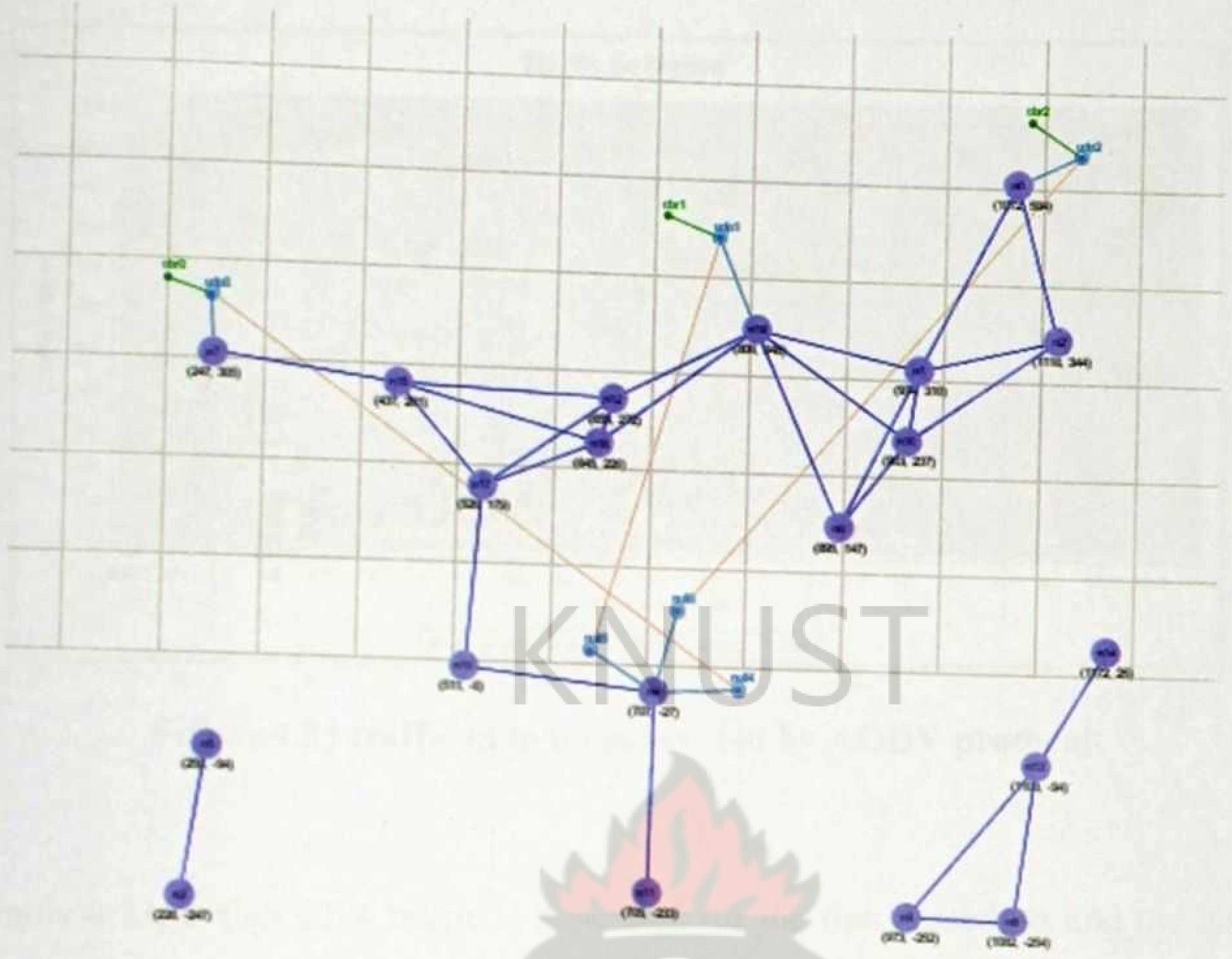


Figure 4.22 Random Topology [3source/1sink]

4.3.3.0 Simulation Parameters

Same simulation parameters in Table **4.1**

4.3.3.1 Traffic Bytes Generated

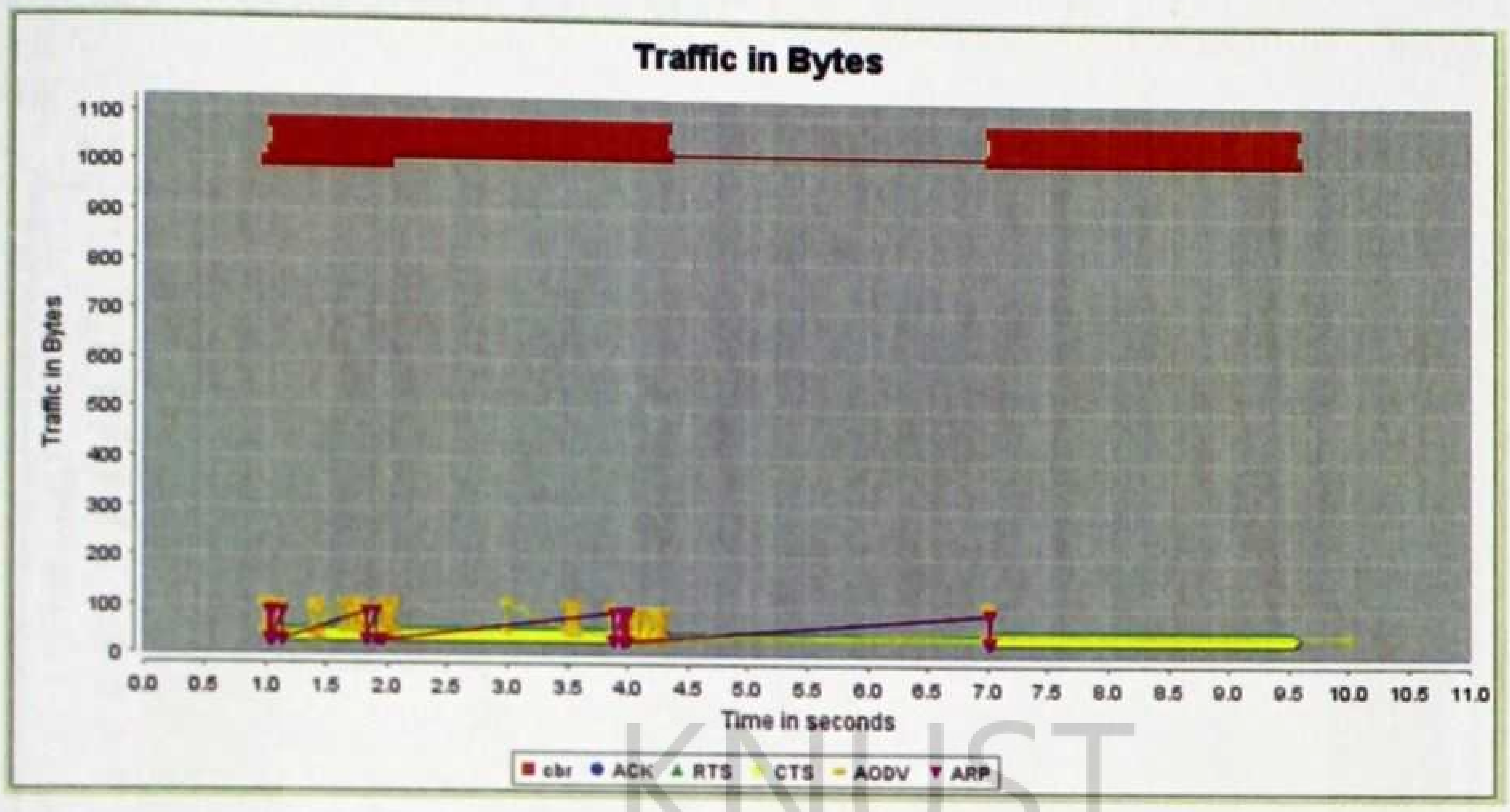


Figure 4.23 traffic in bytes generated by AODV protocol

In Figure 4.23, a high CBR traffic is generated for the first 3 seconds and the last 2.5 seconds. Though MAC layer protocol is eminent throughout the 9 seconds simulation period, ARP lasted for only the first 6 seconds of simulation time.

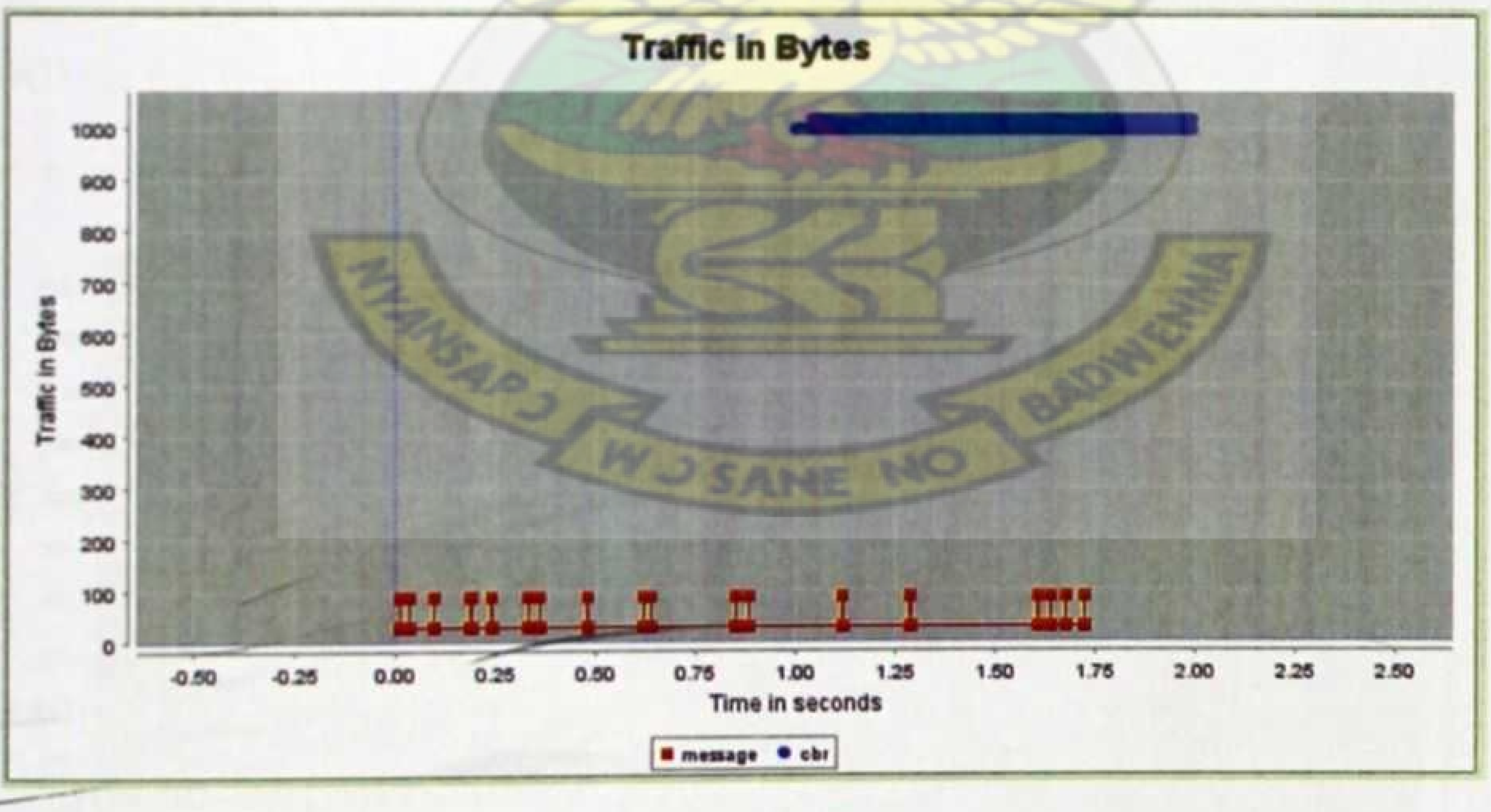


Figure 4.24 traffic in bytes generated by DSDV protocol

In Figure 4.24, DSDV protocol gave similar traffic pattern and not suitable for the purpose of congestion control since MAC layer protocol for congestion control is not present.

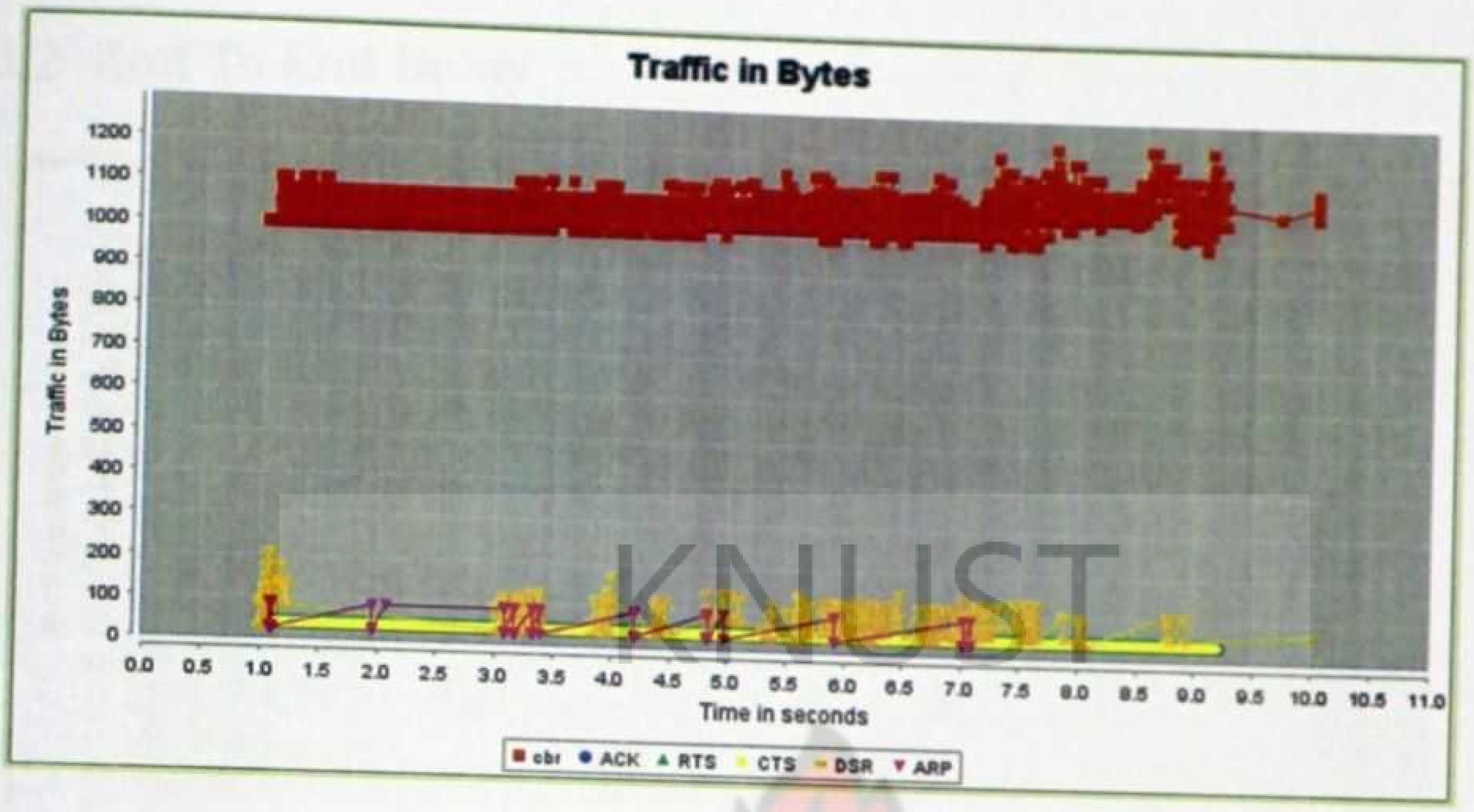


Figure 4.25 traffic in bytes generated by DSR protocol

CBR traffic in Figure 4.25 is between 1000 and a little above 1100 bytes and lasted for total simulation time. The MAC layer protocol is present for the entire simulation time with ARP for the first 6 seconds of simulation.

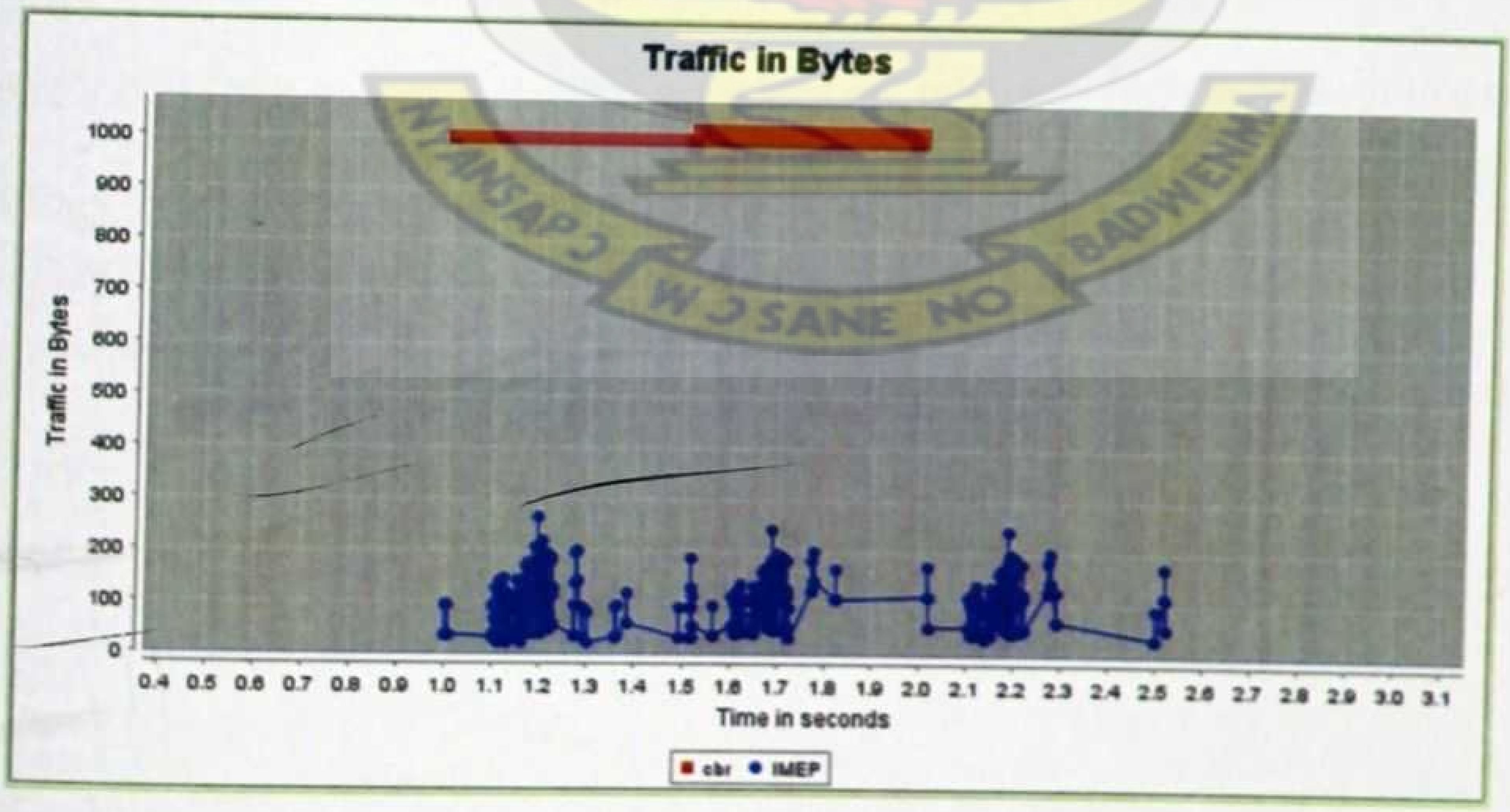


Figure 4.26 traffic in bytes generated by TORA protocol

The presence of IMEP lasted for total simulation time with CBR Traffic for a second. RTS/CTS of MAC protocol is not used for congestion control.

4.3.3.2 End To End Delay

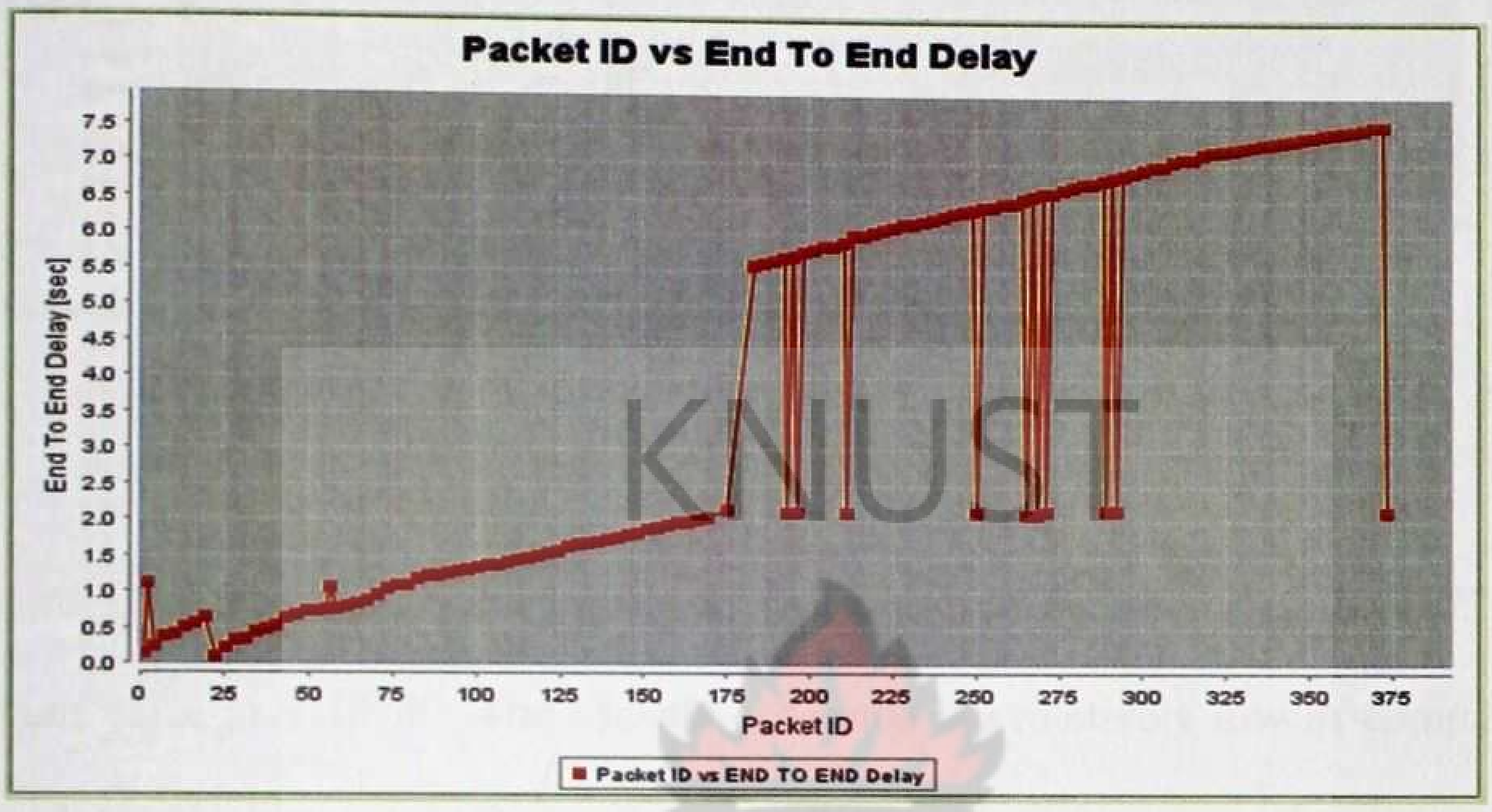


Figure 4.27 End to End Delay of AODV Protocol

In Figure 4.27, the 25 node random topology has maximal end to end delay at '7.5' and minimal at '0.00'. This gave a high value of packet generated by AODV compared to Figure 7.0 of DSR protocol. Packets generated are relatively high because it is a random topology.

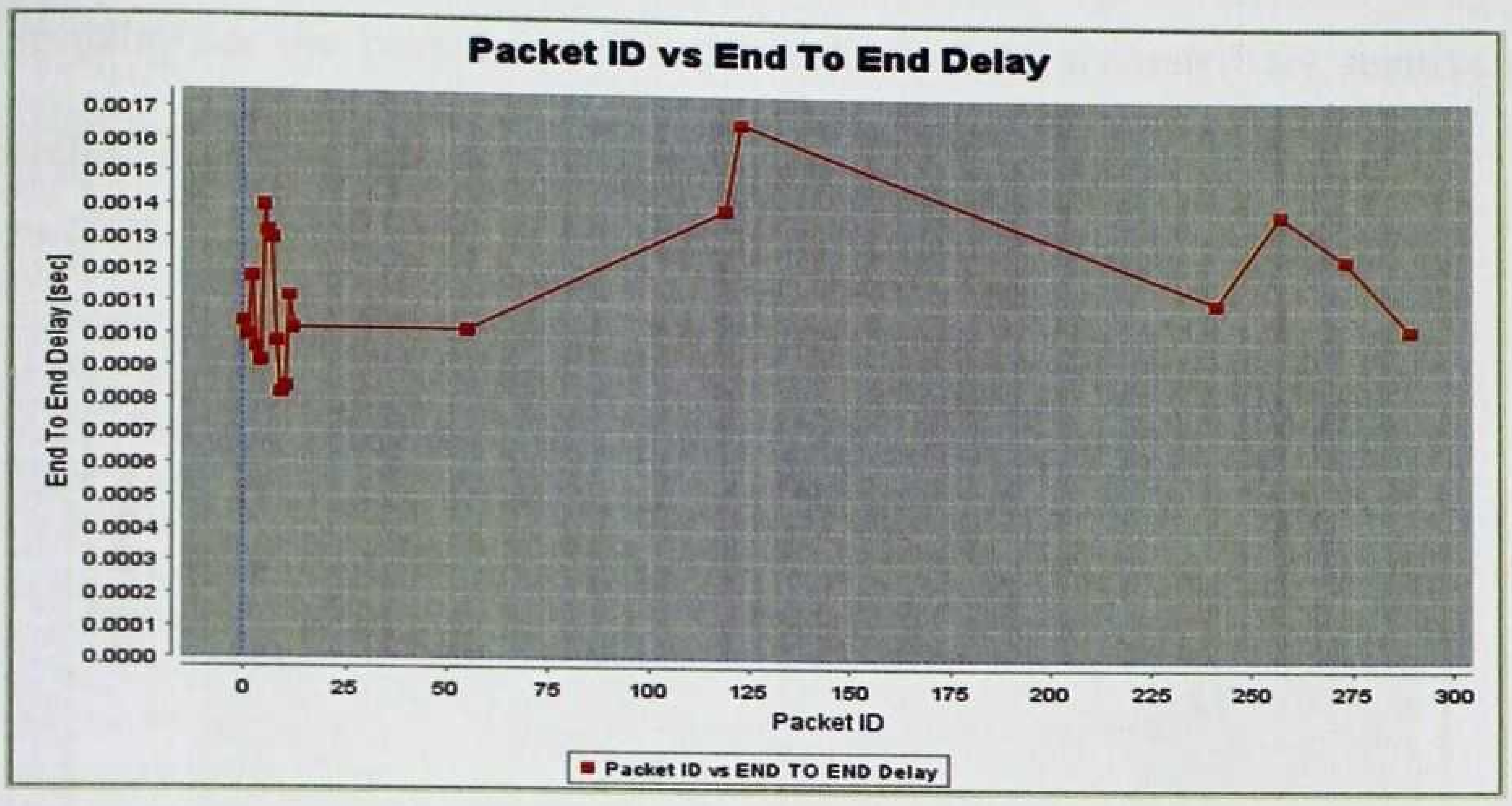


Figure 4.28 End to End Delay of DSDV Protocol

In Figure 4.28, the average end to end delay has a maximum value at '0.0017' and a minimum value at '0.0008'. The packets generated are relatively low as compared to DSR and AODV protocols in Figure 7.0 and 6.8 respectively.

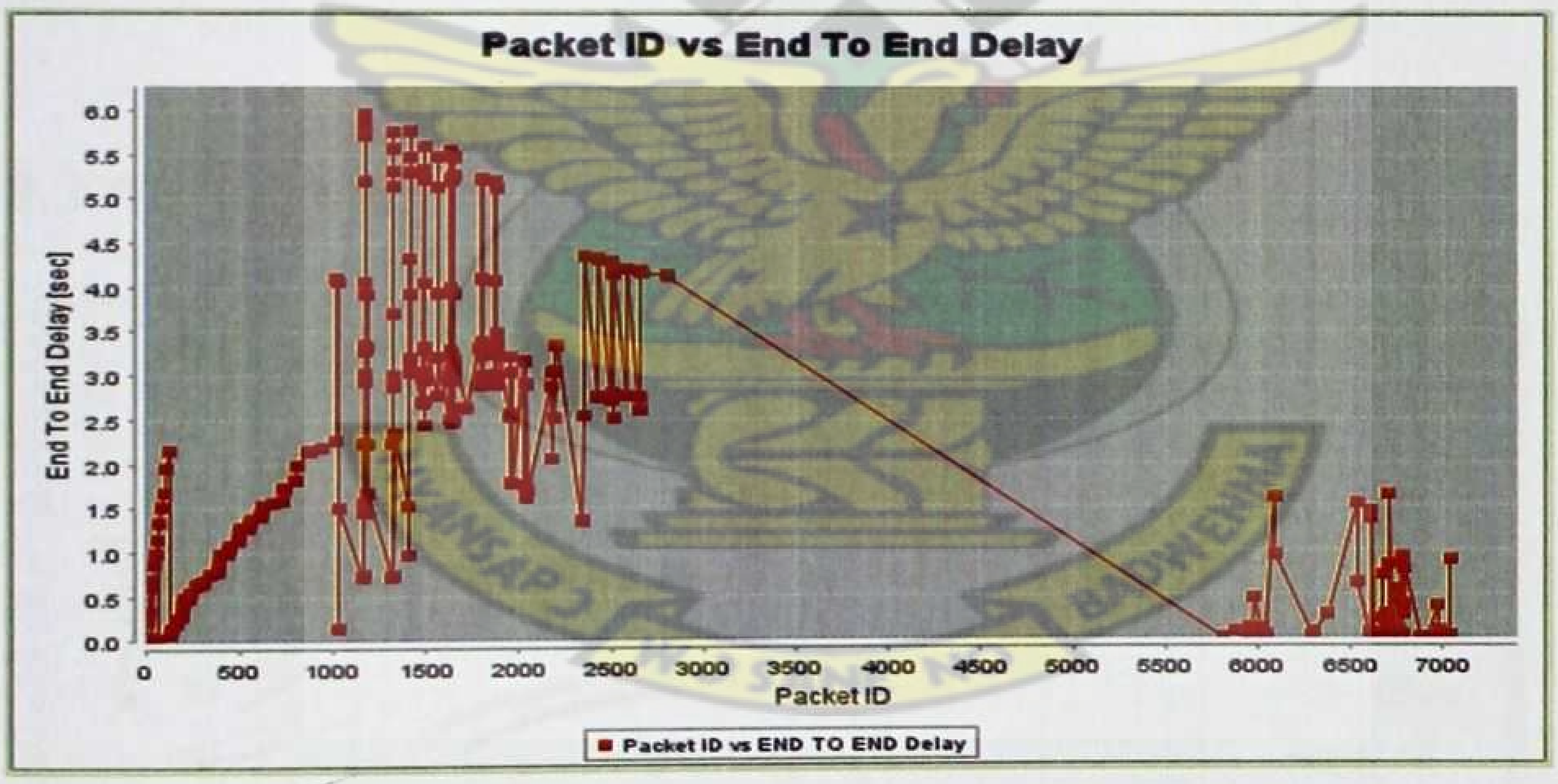


Figure 4.29 End to End Delay of DSR Protocol

In Figure 4.29, the average end to end delay has a maximum value at '6.0' and a minimum value at '0.00'. The value is dependent on the number of packet generated and the delay for the packet to reach the sink. Packets generated are relatively low compared to Figure 4.27 of AODV protocol.

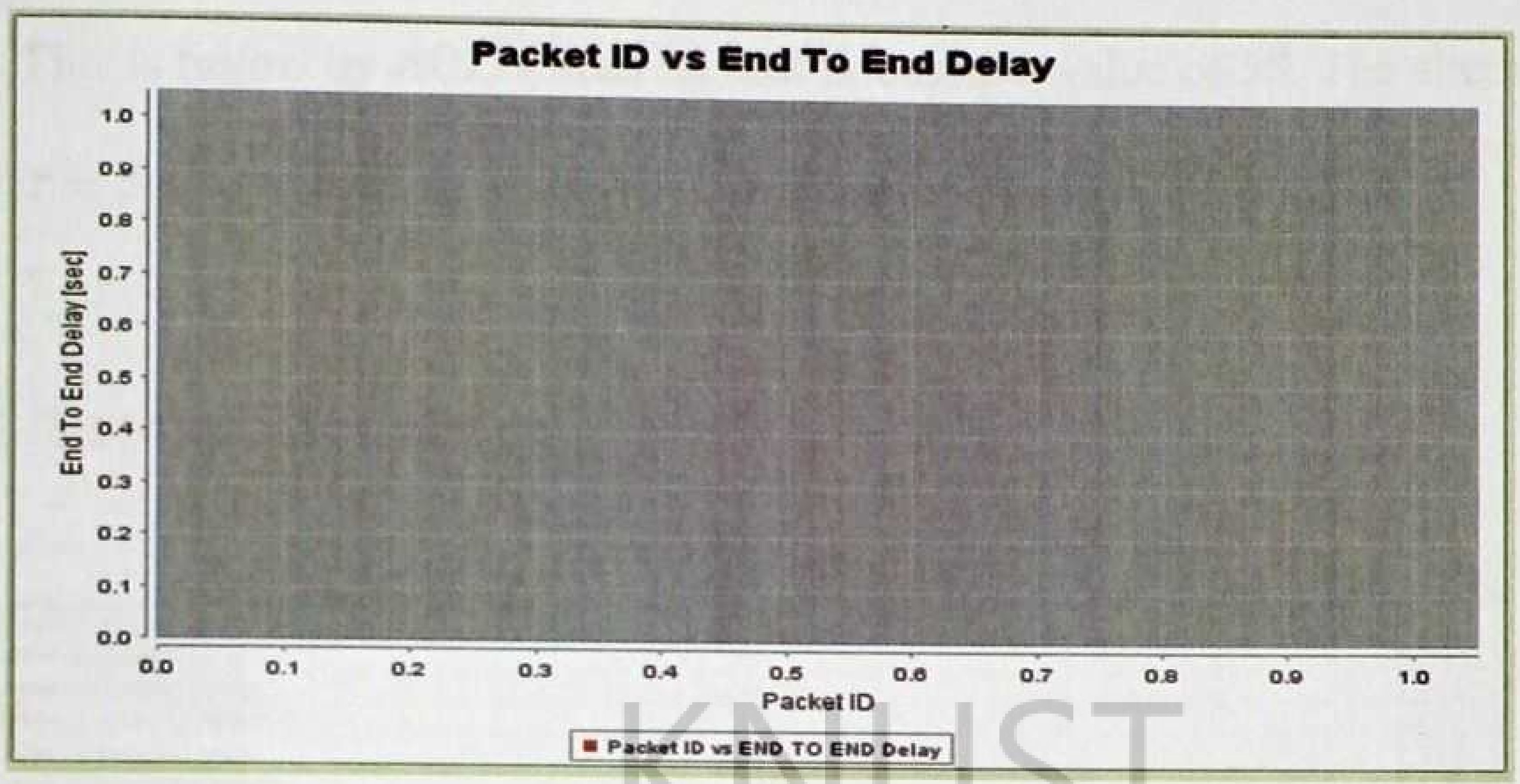


Figure 4.30 End to End Delay of TORA Protocol

As seen in Figure 4.30, TORA protocol generated no end to end delay graph. The algorithm is weak for congestion control.

4.3.3.3 Throughput

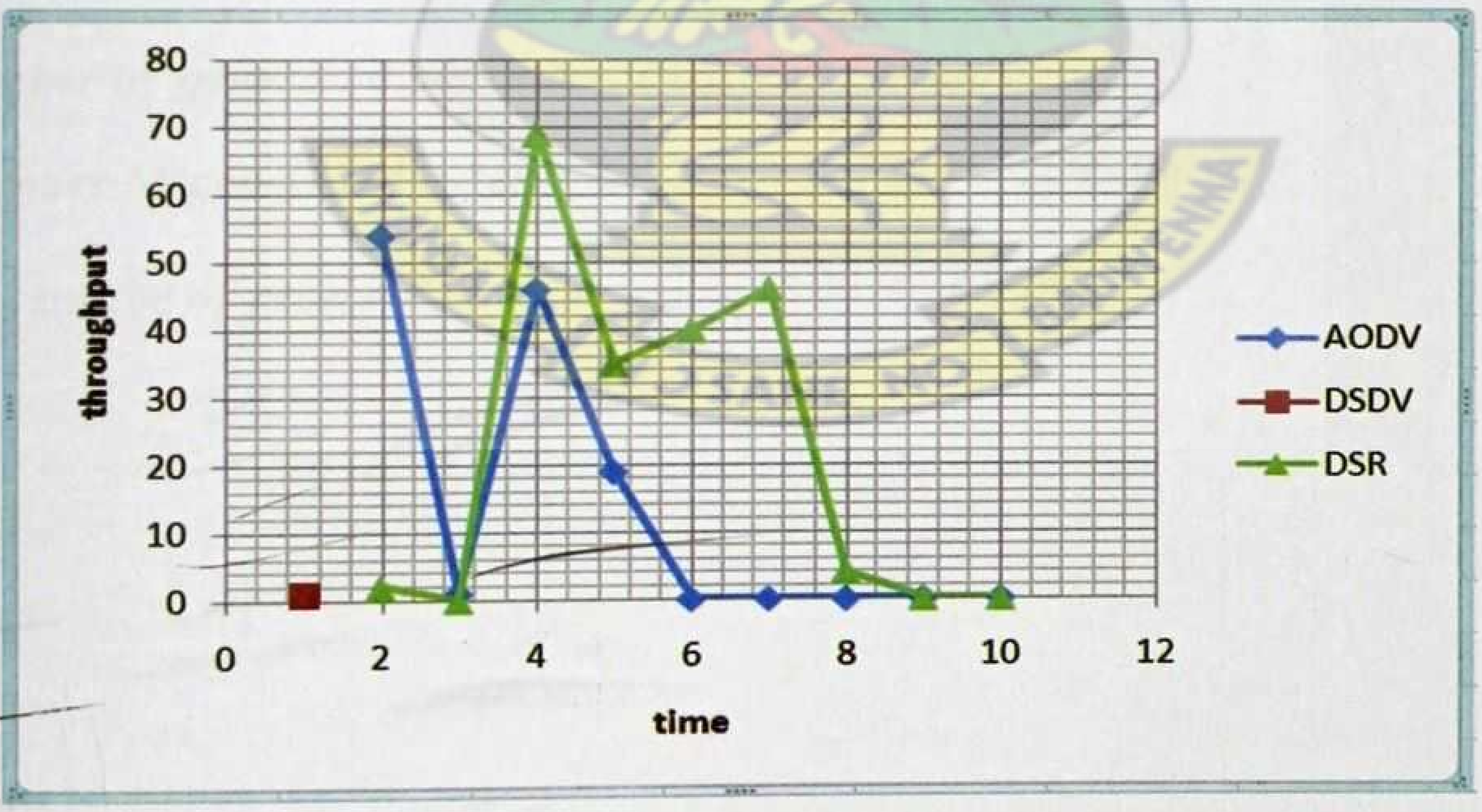
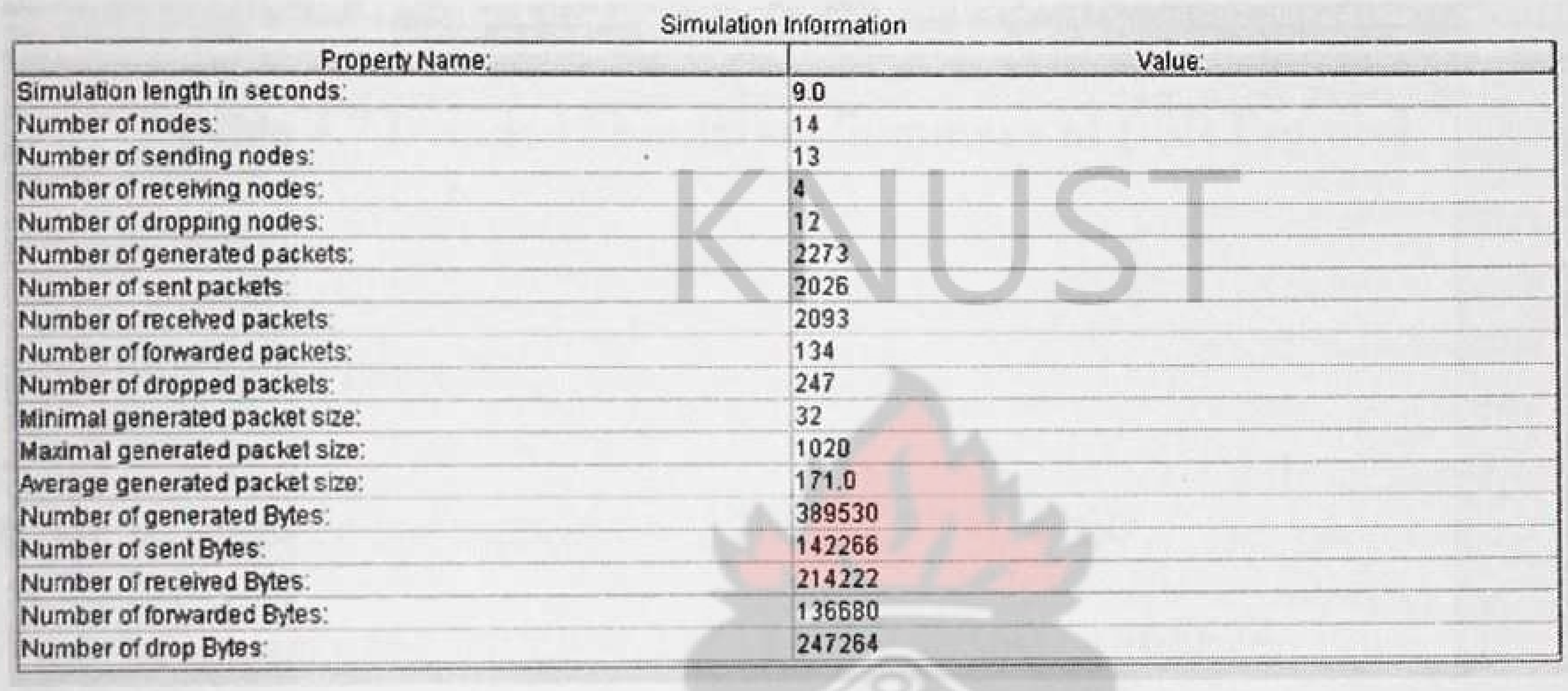


Figure 4.31 Throughput of TADD

In Figure 4.31, throughput of DSR started slowly but increased gradually to 70 at time 4 seconds. This is follow by AODV with highest throughput value of 55. The alternating throughput is due to re-routing which is a characteristic of ad hoc protocols.

4.3.3.4 Detailed Summary



Simulation Information

Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	14
Number of sending nodes:	13
Number of receiving nodes:	4
Number of dropping nodes:	12
Number of generated packets:	2273
Number of sent packets:	2026
Number of received packets:	2093
Number of forwarded packets:	134
Number of dropped packets:	247
Minimal generated packet size:	32
Maximal generated packet size:	1020
Average generated packet size:	171.0
Number of generated Bytes:	389530
Number of sent Bytes:	142266
Number of received Bytes:	214222
Number of forwarded Bytes:	136680
Number of drop Bytes:	247264

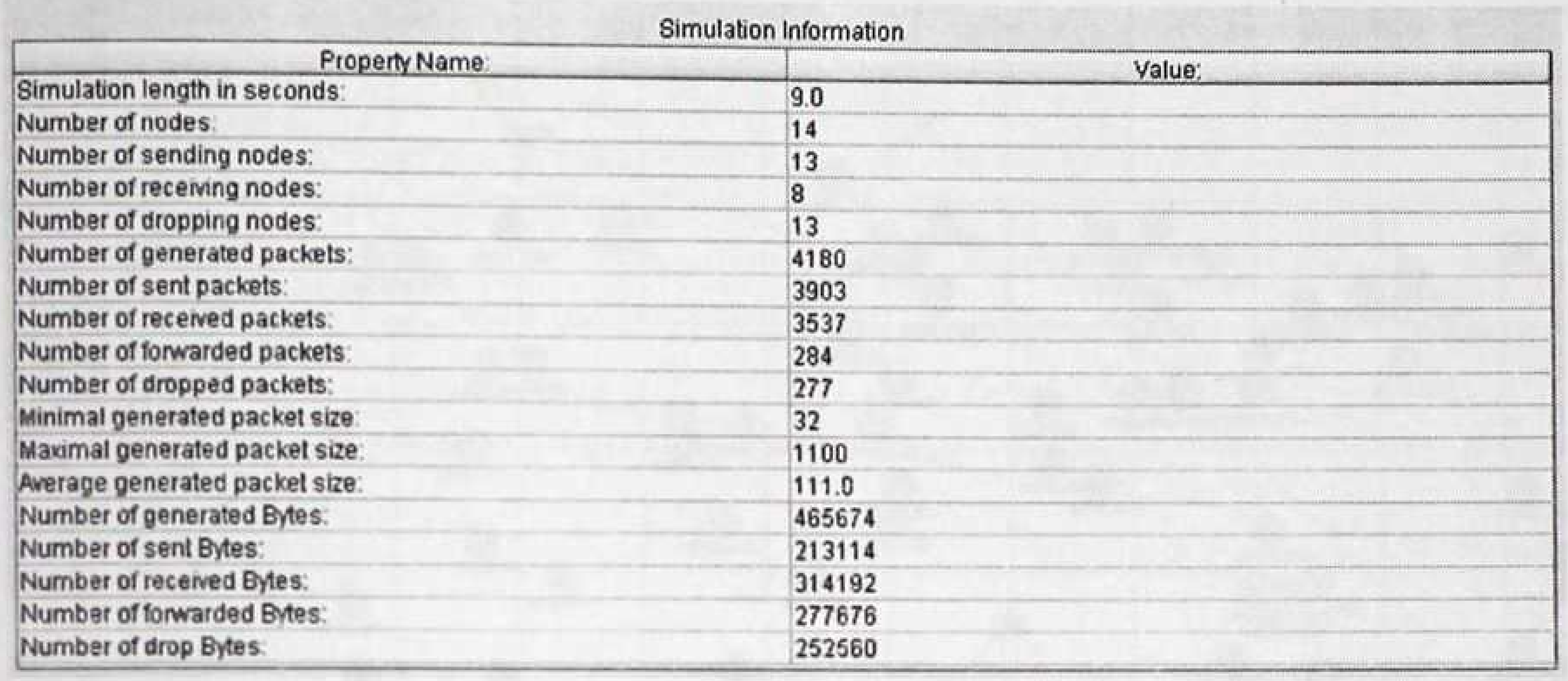
Table 4.6 Detailed Simulation Summary of AODV Protocol

From Equation 1 and Table 4.6:

$a. number\ of\ generated\ packets = 2026 + 247 = 2273$

$b. number\ of\ generated\ bytes = 142266 + 247264 = 389530$

$c. percentage\ of\ received\ bytes = \frac{214222}{389530} * 100 = 54.995\%$


Simulation Information

Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	14
Number of sending nodes:	13
Number of receiving nodes:	8
Number of dropping nodes:	13
Number of generated packets:	4180
Number of sent packets:	3903
Number of received packets:	3537
Number of forwarded packets:	284
Number of dropped packets:	277
Minimal generated packet size:	32
Maximal generated packet size:	1100
Average generated packet size:	111.0
Number of generated Bytes:	465674
Number of sent Bytes:	213114
Number of received Bytes:	314192
Number of forwarded Bytes:	277676
Number of drop Bytes:	252560

Table 4.7 Detailed Simulation Summary of DSR Protocol

From Equation 1 and Table 4.7:

$$a. number\ of\ generated\ packets = 3903 + 277 = 4180$$

$$b. number\ of\ generated\ bytes = 213114 + 252560 = 465674$$

$$d. percentage\ of\ received\ bytes = \frac{314192}{465674} * 100 = 67.470\%$$

4.3.3.5 RANDOM Topology (25 nodes) Results Summary

The random topology results show an increase in data delivery by AODV protocol from 35.907% in Table 4.2 to 54.995% in Table 4.6. The number of received bytes also increased in DSR protocol from 58.138% in Table 4.3 to 67.470% in Table 4.7. Though a significant data loss is recorded due to congestion, DSR from the simulation result still has a higher data delivery ratio than all other protocols which implies a higher throughput value.

4.3.4 Random Topology [3 source/ 2 sink] - 150 nodes

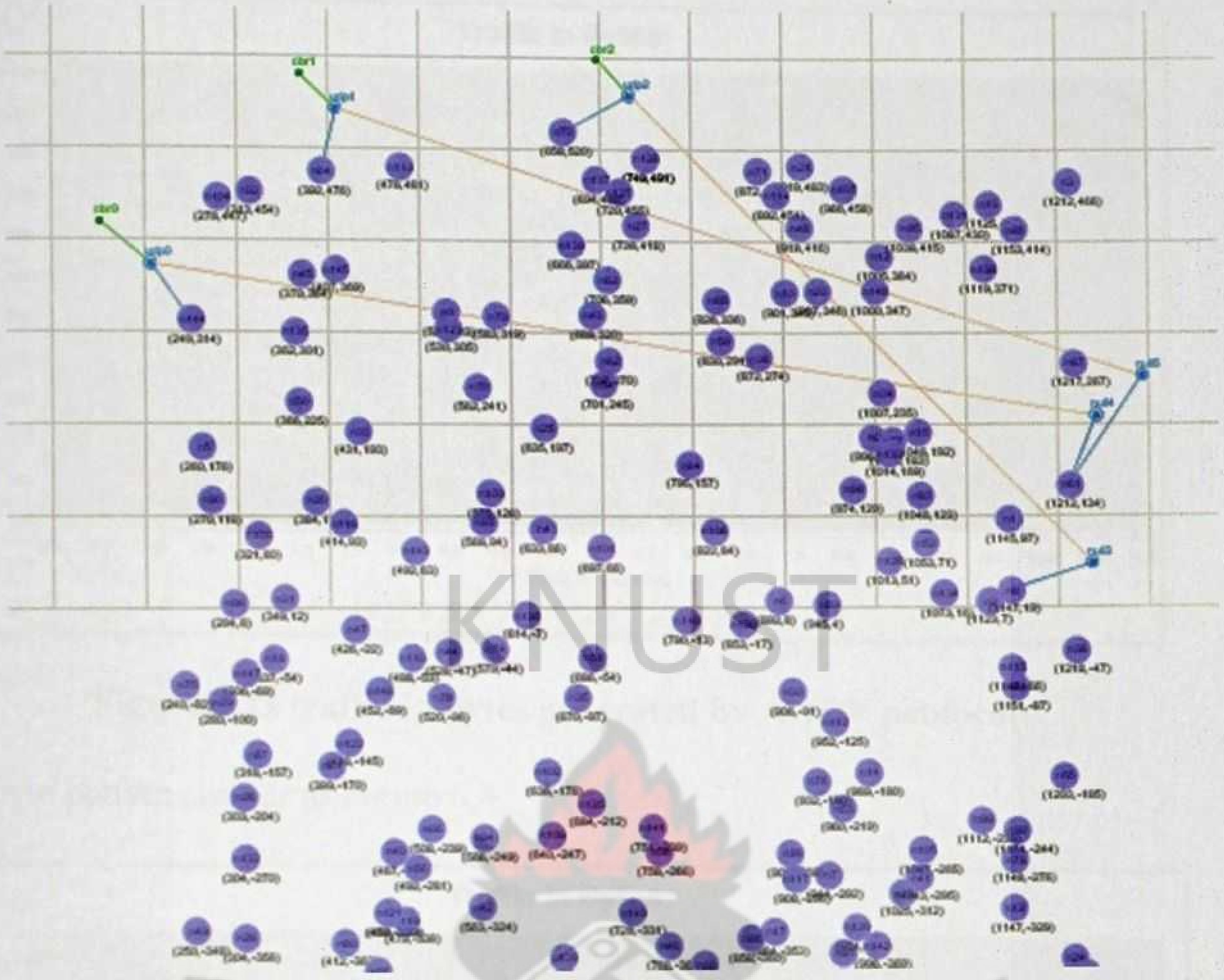


Figure 4.32 Random Topology [3source/2sink]

In Figure 4.32, a random topology of 150 nodes is deployed and simulated. Three source nodes n144, n64, n12 and sink nodes n61, n9.

4.3.4.0 Simulation Parameters

Same simulation parameters in Table **4.1**

4.3.4.1 Traffic Bytes Generated

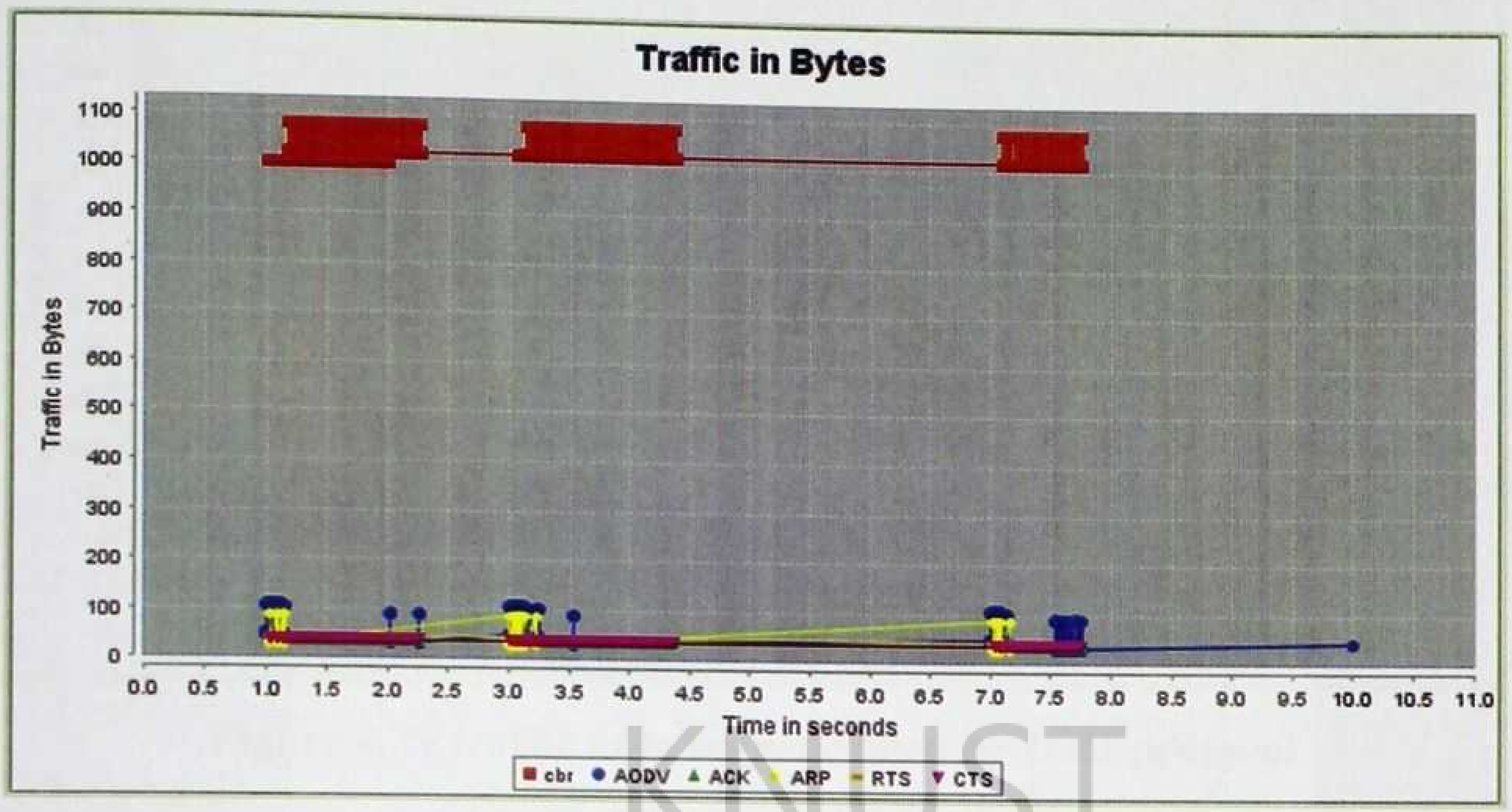


Figure 4.33 traffic in bytes generated by AODV protocol

Traffic byte pattern similar to Figure 6.4

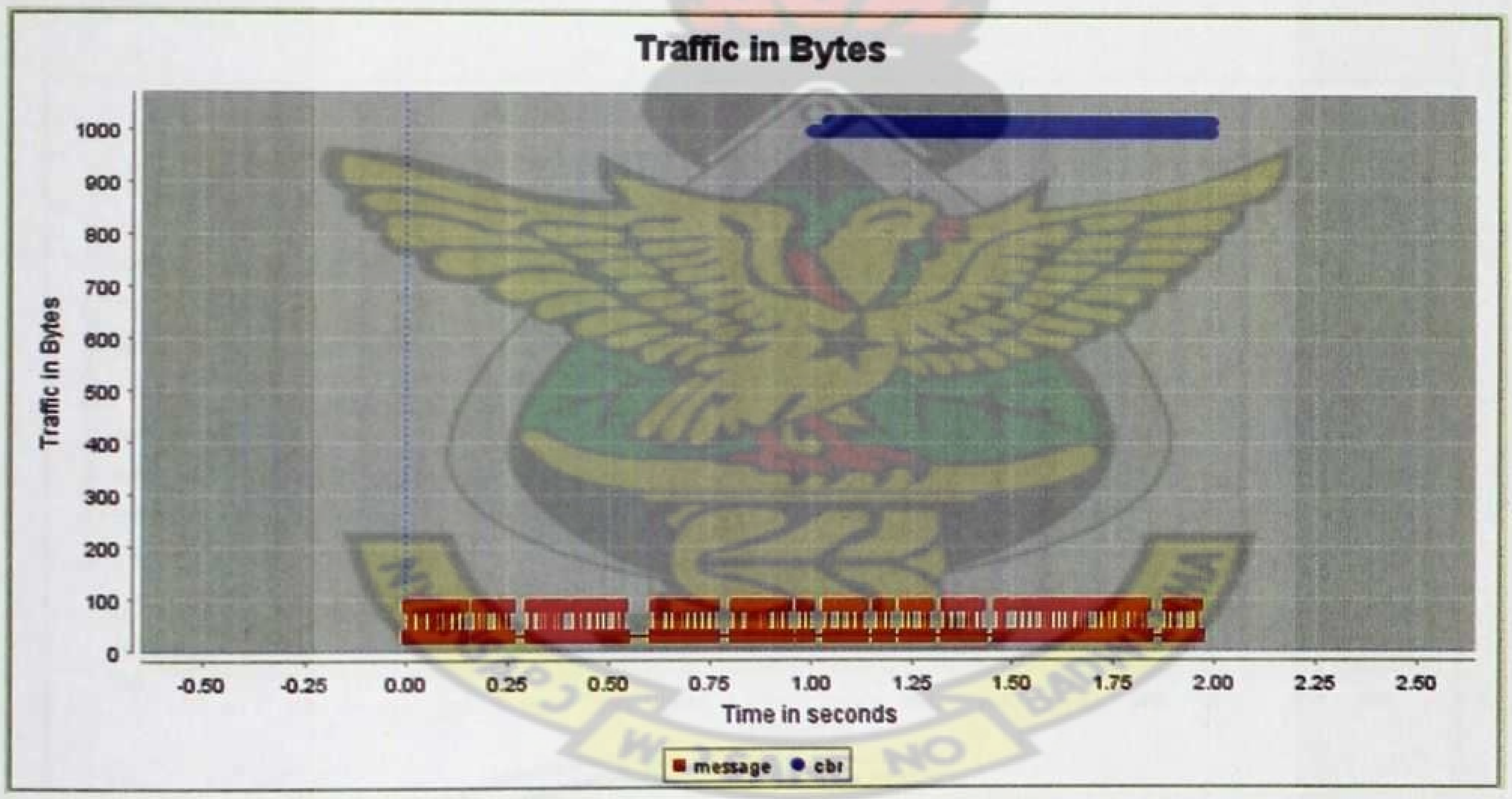


Figure 4.34 traffic in bytes generated by DSDV protocol

Traffic byte pattern similar to Figure 6.5

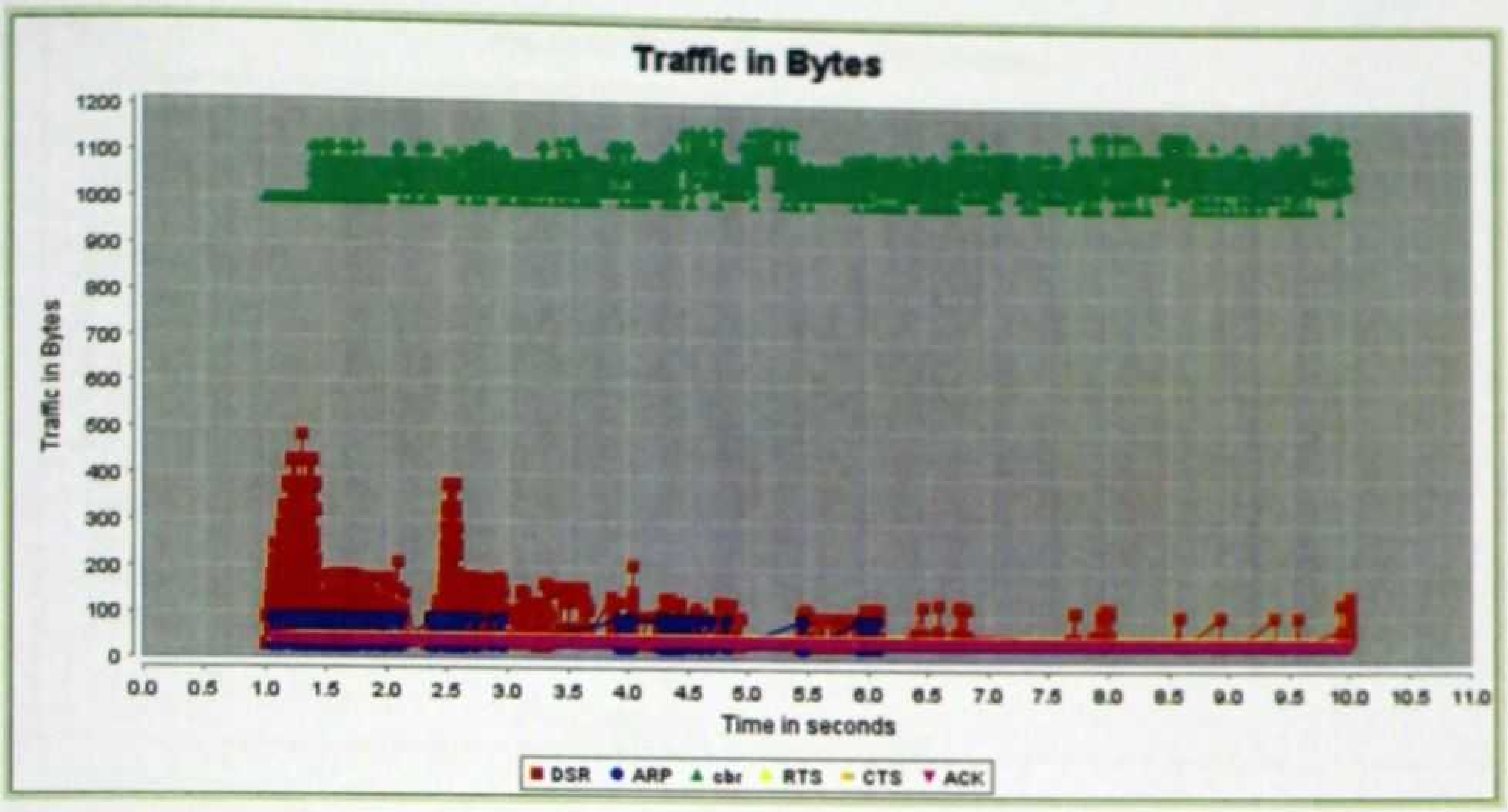


Figure 4.35 traffic in bytes generated by DSR protocol

Traffic byte pattern similar to Figure 4.25.

4.3.4.2 End to End Delay

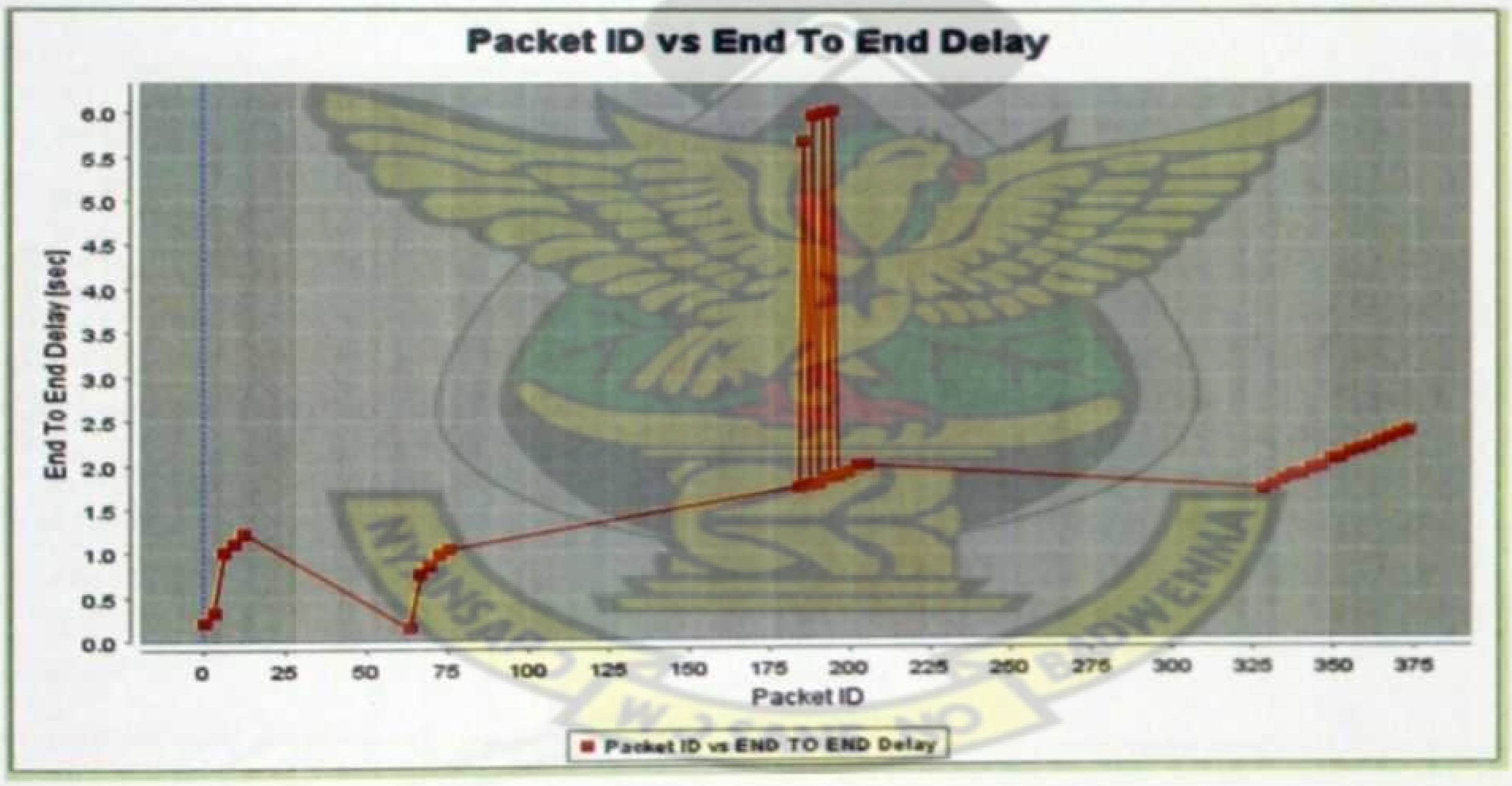


Figure 4.36 End to End Delay of AODV Protocol

In Figure 4.36, the 150 node random topology has maximum end to end delay value at '6.0' and a minimum value at '0.1'. This gave a low value of packet generated by AODV compared to Figure 4.38 of DSR protocol.

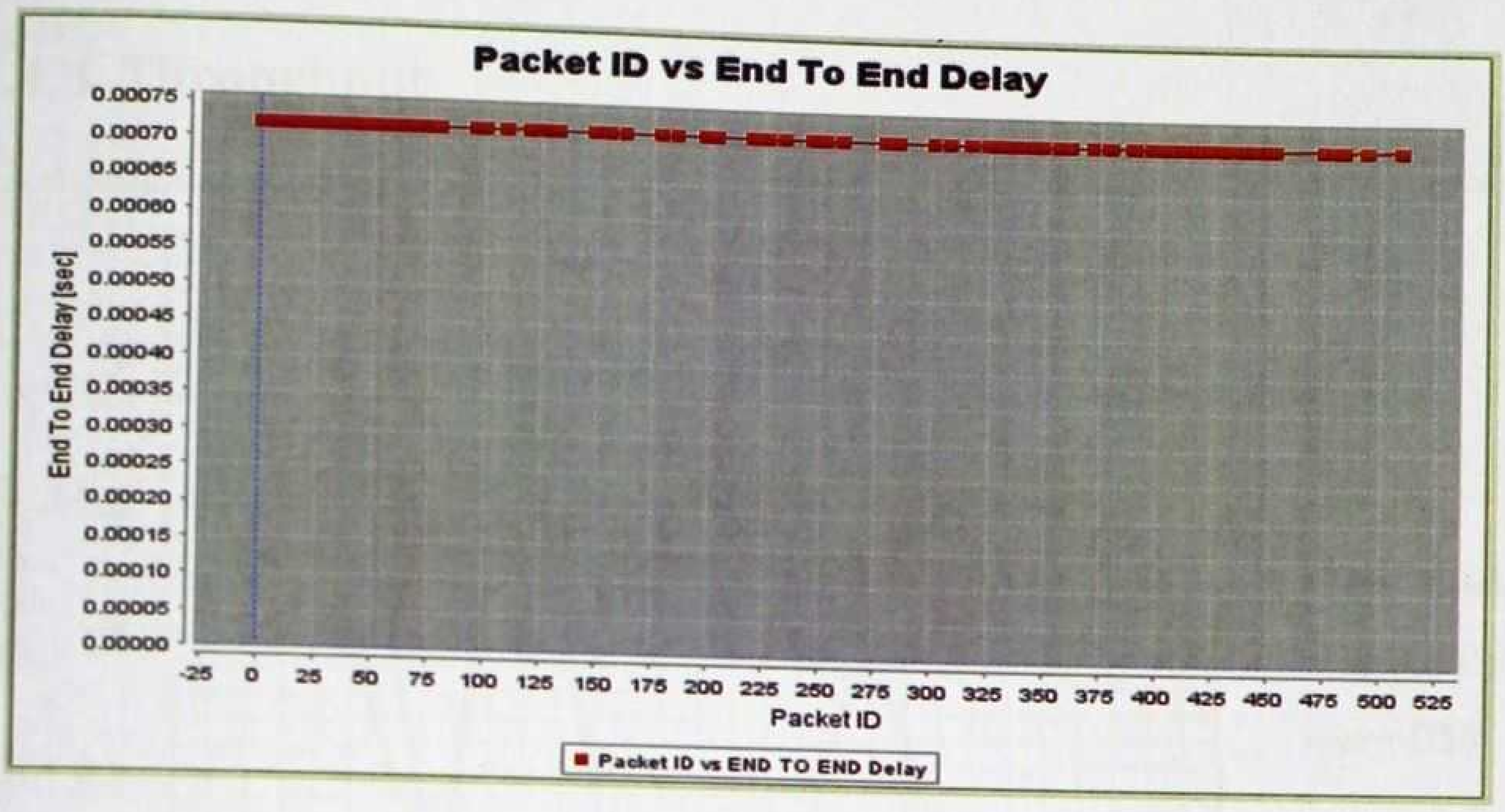


Figure 4.37 End to End Delay of DSDV Protocol

In Figure 4.37, the 150 node random topology has a maximum end to end delay value at '0.00072' and a minimum at '0.00072'.

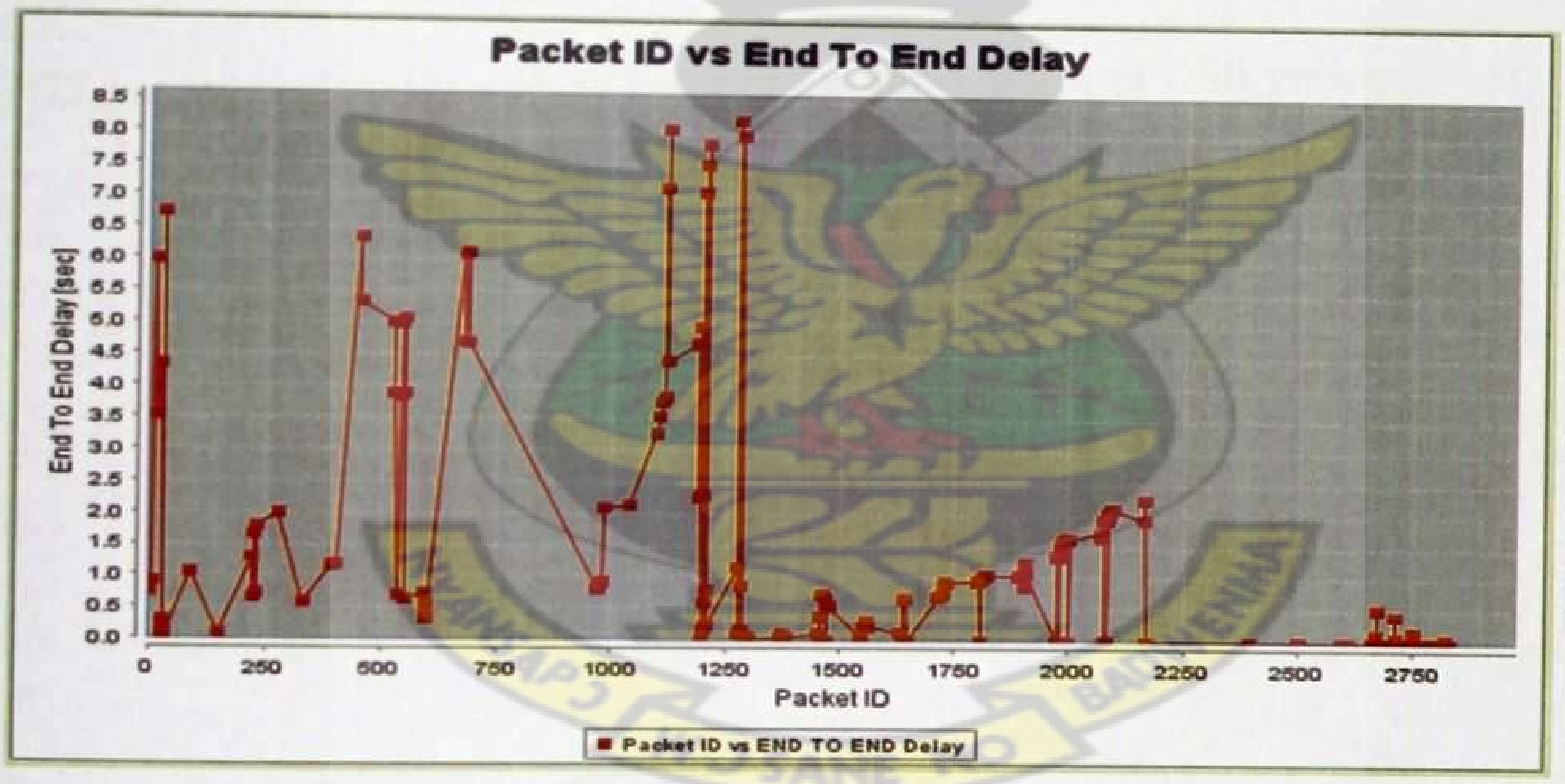


Figure 4.38 End to End Delay of DSR Protocol

In Figure 4.38, the 150 node random topology has a maximum end to end delay value at '8.2' and a minimum value at '0.0'. This is a high value of packets generated by DSR compared to Figure 4.36 of AODV protocol.

4.3.4.3 Throughput

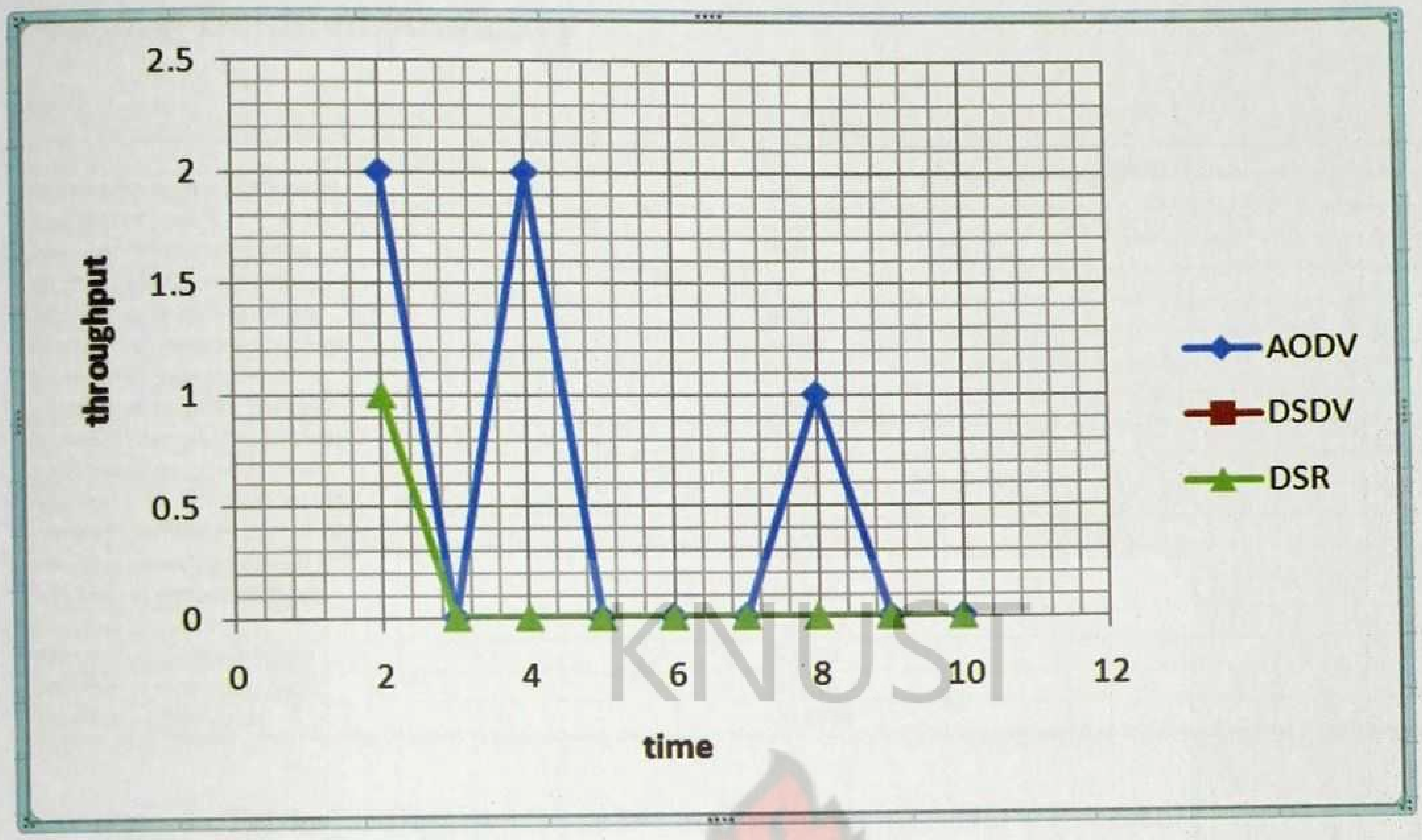
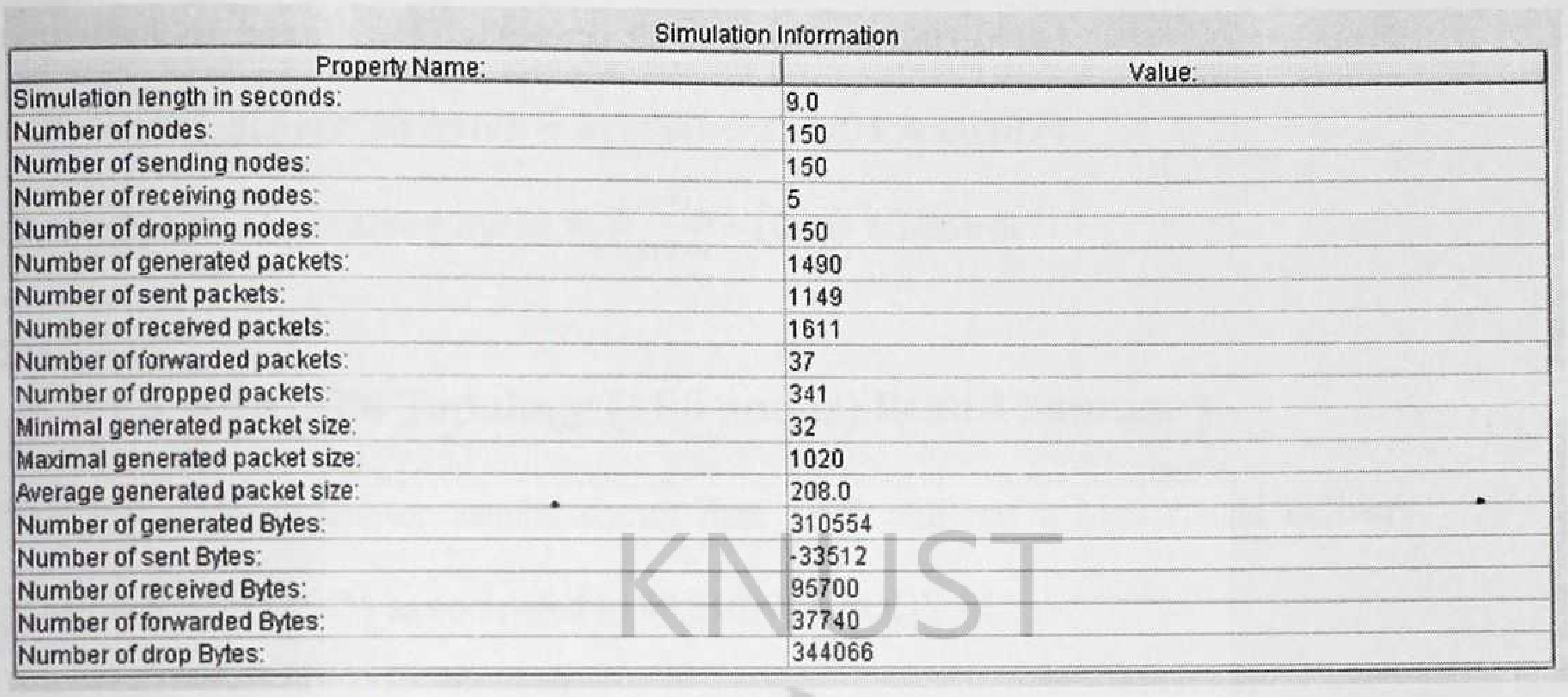


Figure 4.39 Throughput of TADD

AODV has the highest throughput value of 2 compared to 1 of DSR protocol.

4.3.4.4 Detailed Summary



Simulation Information

Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	150
Number of sending nodes:	150
Number of receiving nodes:	5
Number of dropping nodes:	150
Number of generated packets:	1490
Number of sent packets:	1149
Number of received packets:	1611
Number of forwarded packets:	37
Number of dropped packets:	341
Minimal generated packet size:	32
Maximal generated packet size:	1020
Average generated packet size:	208.0
Number of generated Bytes:	310554
Number of sent Bytes:	-33512
Number of received Bytes:	95700
Number of forwarded Bytes:	37740
Number of drop Bytes:	344066

Table 4.8 Detailed Simulation Summary of AODV Protocol

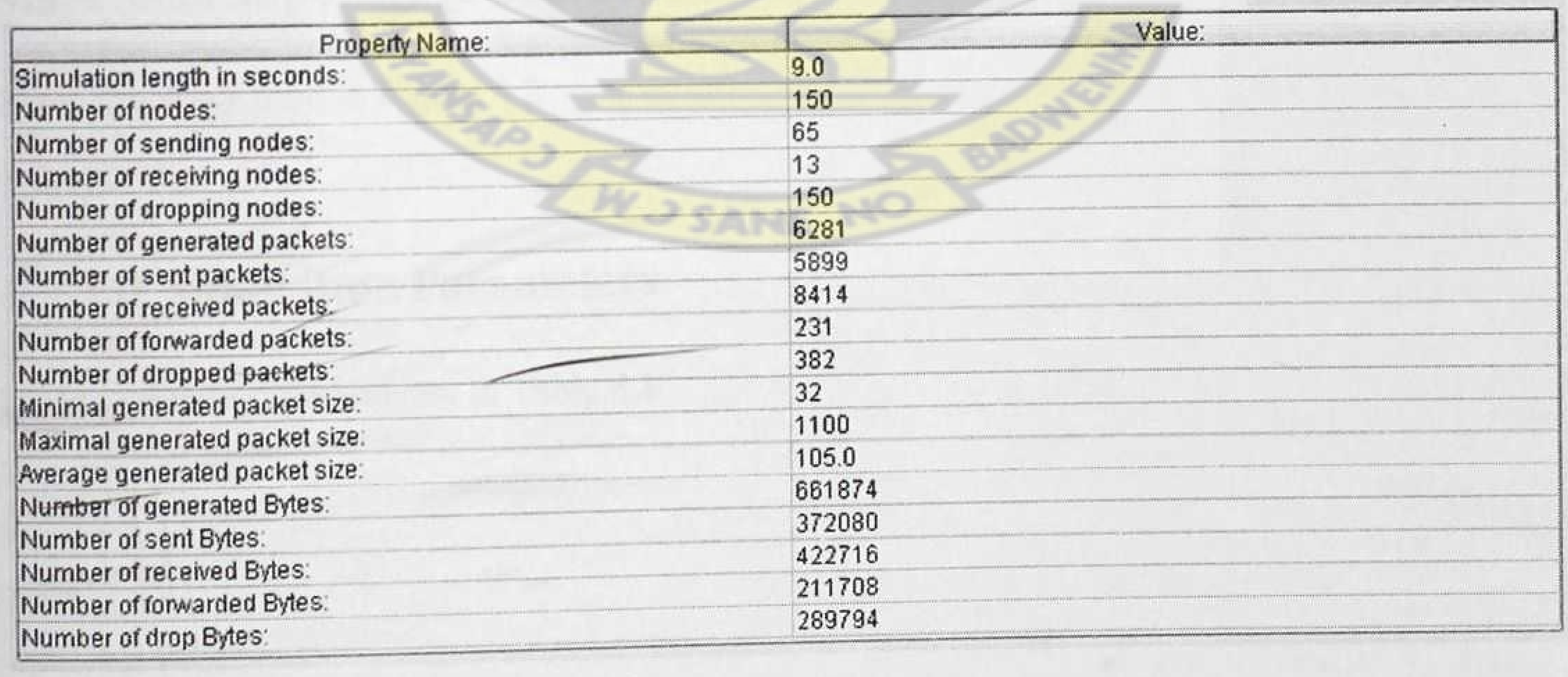
From Equation 1 and Table 4.8:

$$a. number\ of\ generated\ packets = 5899 + 382 = 6281$$

$$a. number\ of\ generated\ packets = 1149 + 341 = 1490$$

$$b. number\ of\ generated\ bytes = -33512 + 344066 = 310554$$

$$c. percentage\ of\ received\ bytes = \frac{95700}{310554} * 100 = 30.816\%$$



Property Name:	Value:
Simulation length in seconds:	9.0
Number of nodes:	150
Number of sending nodes:	65
Number of receiving nodes:	13
Number of dropping nodes:	150
Number of generated packets:	6281
Number of sent packets:	5899
Number of received packets:	8414
Number of forwarded packets:	231
Number of dropped packets:	382
Minimal generated packet size:	32
Maximal generated packet size:	1100
Average generated packet size:	105.0
Number of generated Bytes:	661874
Number of sent Bytes:	372080
Number of received Bytes:	422716
Number of forwarded Bytes:	211708
Number of drop Bytes:	289794

Table 4.9 Detailed Simulation Summary of DSR Protocol

From Equation 1 and Table 4.9:

$$a.\, number\ of\ generated\ packets = 5899 + 382 = 6281$$

$$b.\, number\ of\ generated\ bytes = 372080 + 289794 = 661874$$

$$c.\, percentage\ of\ received\ bytes = \frac{422716}{661874} * 100 = 63.866\%$$

4.3.4.5 RANDOM Topology (150 nodes) Result Summary

The 150 node simulation results shows that, DSR still has a higher data delivery percentage of 65.866% as compared to 30.816% of AODV.

4.4 PART 2 : Traffic Protocol NOAH and DelStatic

In this section, an improved DelStatic protocol is compared with NOAH protocol using metrics end to end delay and throughput. The generated graph shows a significant throughput increase in DelStatic protocol over a NOAH protocol.

4.4.1 Grid Topology [2 source/1 sink] – 25 nodes

Same topology used in Figure 4.2

4.4.1.0 Simulation Parameters

Same simulation parameters in Table **4.1**

4.4.1.1 End To End Delay

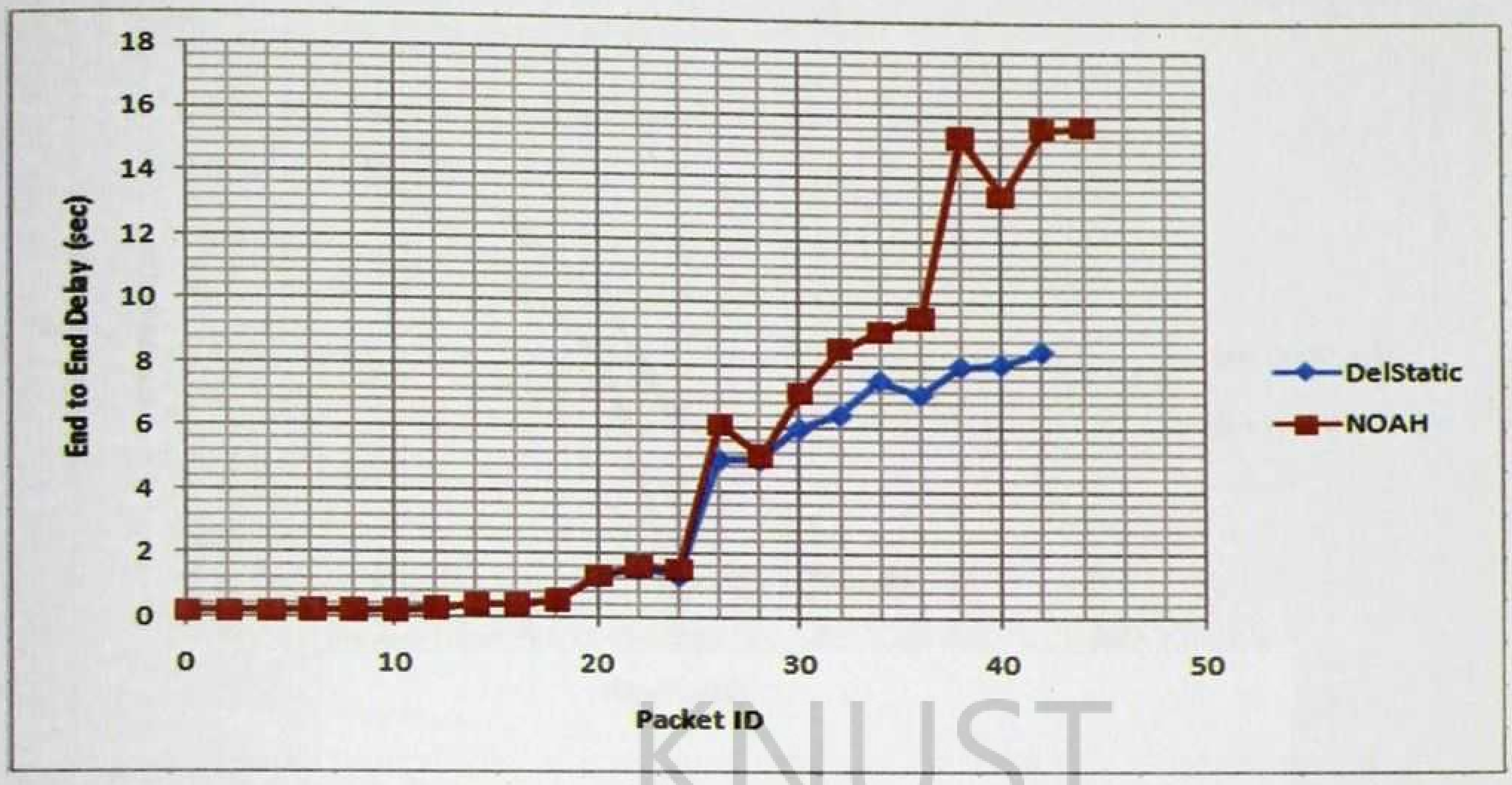


Figure 4.40 End to End Delay of DelStatic and NOAH Protocol.

In Figure 4.40, the delay graph in DelStatic shows a shorter time interval for data packet arrival to NOAH graph. The short time range is possible due to the introduction of backpressure algorithm which is significant in controlling congestion during data packet transmission from source to sink node.

4.4.1.2 Throughput

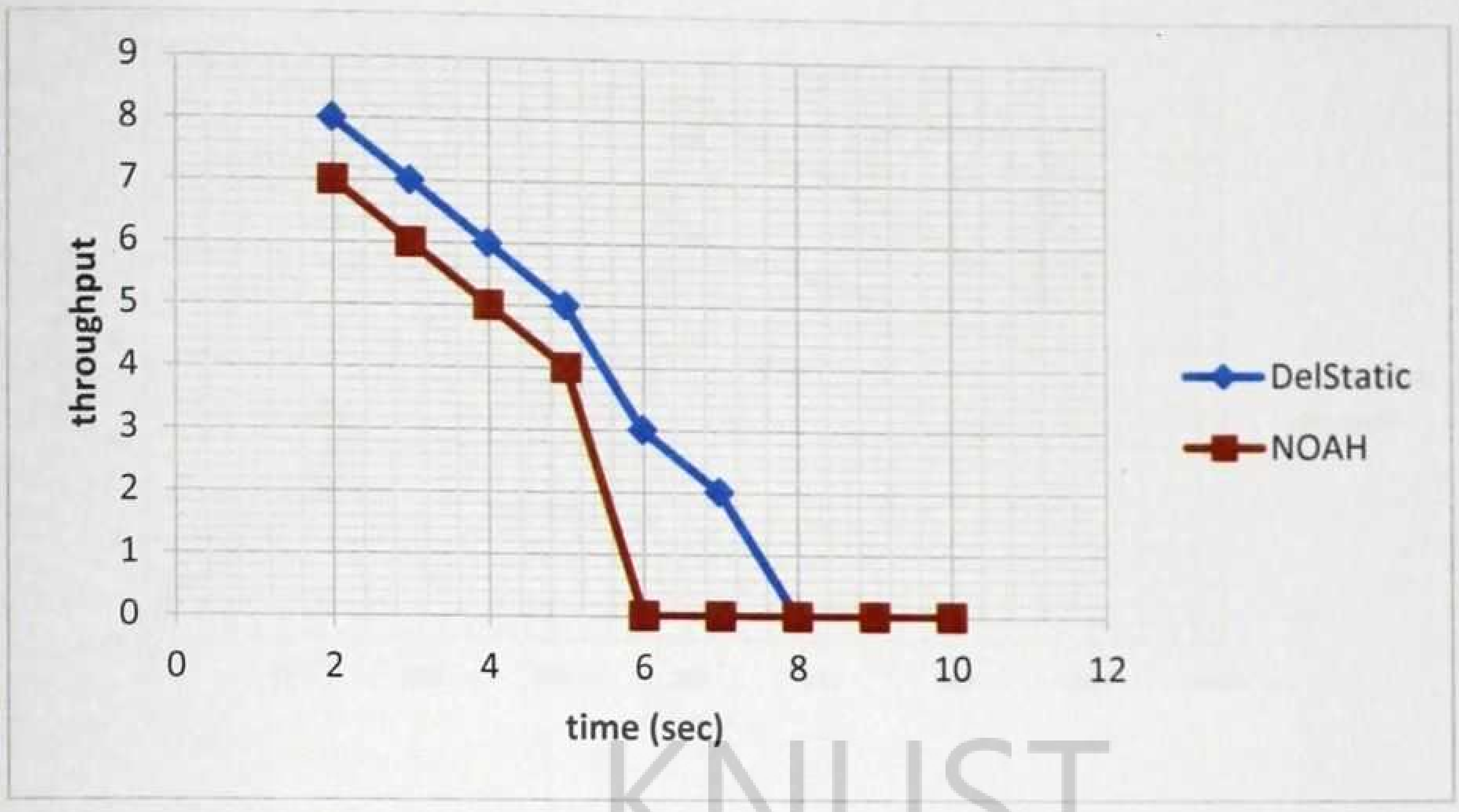


Figure 4.41 Throughput of DelStatic and NOAH Protocol

Throughput is the average rate of successful packet delivered to the sink over a sensor network. From Figure 4.41, DelStatic protocol still performed better in congested network with higher throughput data delivery ratio compared to NOAH protocol. An average throughput value of 8 is recorded during the initial stages of simulation as compared to 7 of NOAH protocol.

4.4.2 Random Topology [3 source/2 sink] – 150 nodes

Simulation is conducted again using the topology in Figure 7.3

4.4.2.0 Simulation Parameters

Same as Table **4.1** but the number of nodes = 150

4.4.2.1 End To End Delay

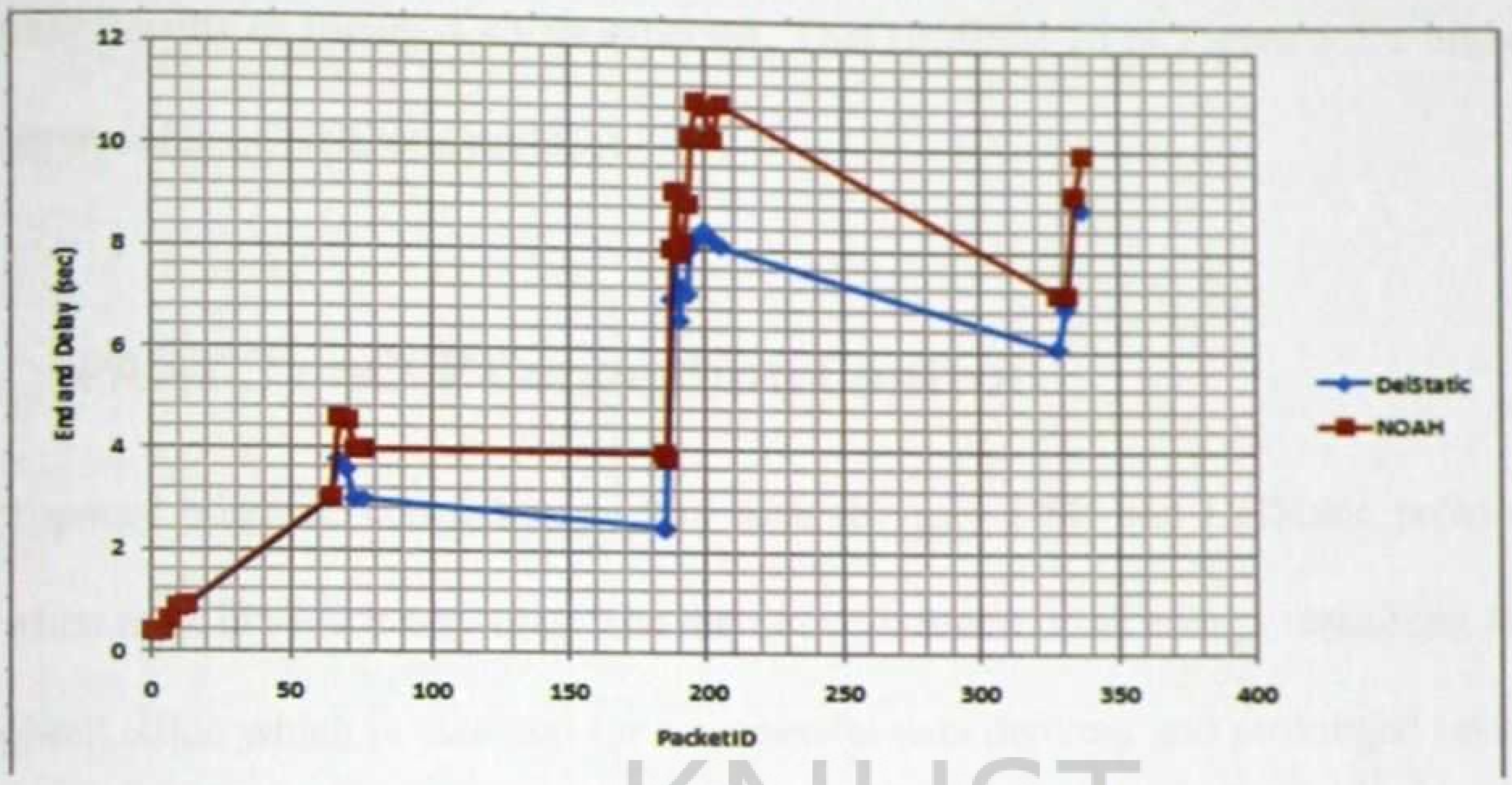


Figure 4.42 End to End Delay of DelStatic and NOAH Protocol

From Figure 4.42, NOAH protocol results are as expected. As can be seen from the graph DelStatic still had a higher end to end delay values. The additional node points obtained is as a result of the 150 node structure.

4.4.2.2 Throughput

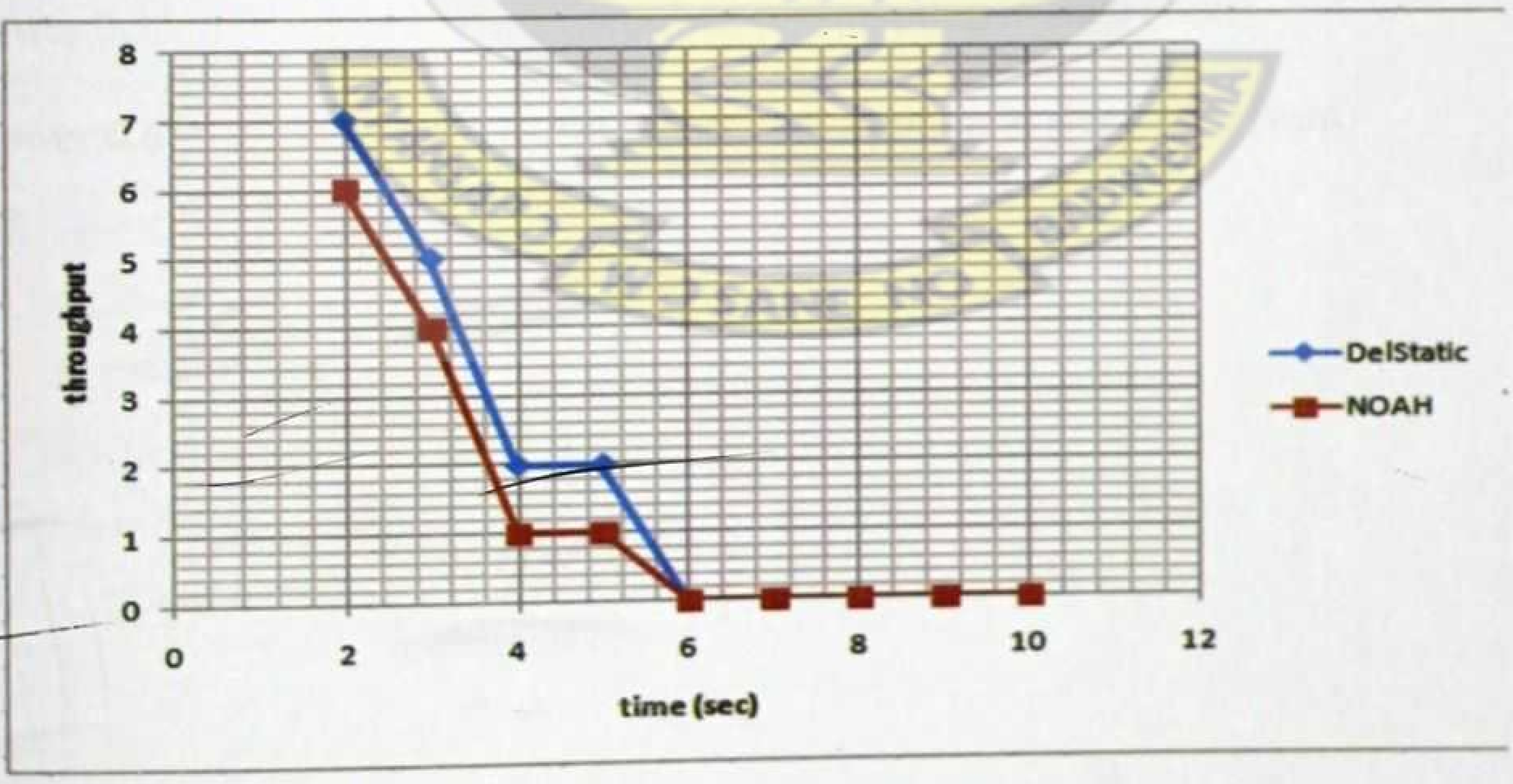


Figure 4.43 Throughput of DelStatic and NOAH Protocol

The graph results in Figure 4.43, is expected. This confirms from Figure 8.2 a higher throughput value of DelStatic protocol.

4.5 PART 3 : ACCP Congestion Control

The proposed adaptive congestion control seek the gain DSR and DelStatic protocol have when used in sensor networks. The metrics of essence is the energy remaining and throughput value which is essential for a successful data delivery and prolonged sensor life deployment.

4.5.1 Grid Topology [2 source / 1 sink] – 25 nodes

Grid topology in Figure 4.0 is simulated with the introduction of energy parameters.

4.5.1.0 Energy Parameters

```
set opt(initialenergy)  900        ;# Initial energy in Joules
-rxPower 0.3 \                     ;# power consumption in state (watt)
 -txPower 0.6 \                    ;# power consumption in sleep state (watt)
```

4.5.1.1 Energy Remaining, ACCP

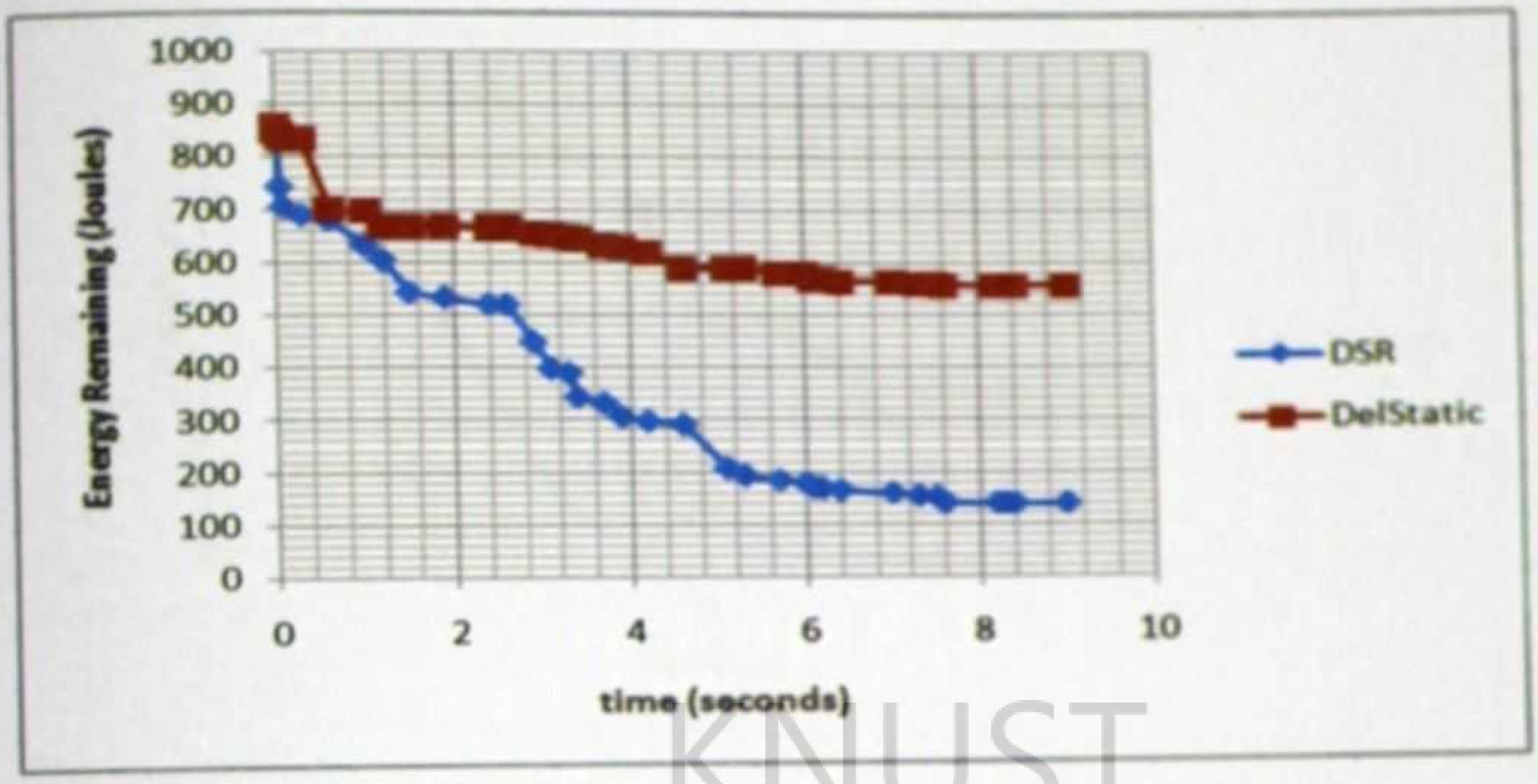


Figure 4.44 Energy Remaining, ACCP

In Figure 4.44, a sink sets a threshold energy value of 500 Joules in simulation. The simulation is tested with ACCP protocol which consists of DSR and DelStatic protocol, for a time interval of 10 seconds. After simulation, it was noted that, DelStatic protocol in controlling congestion depleted energy to a value of 600J which is above the set threshold value whiles DSR congestion control sup parsed the threshold value to 100J. ACCP protocol is set to save a sensor application which is **reliant on energy** 500 J (600-100J) if implemented to control congestion. The sink node of such an application will have ACCP switching algorithm which triggers DelStatic protocol anytime energy consumption is below a constant threshold.

4.5.1.2 Throughput, ACCP

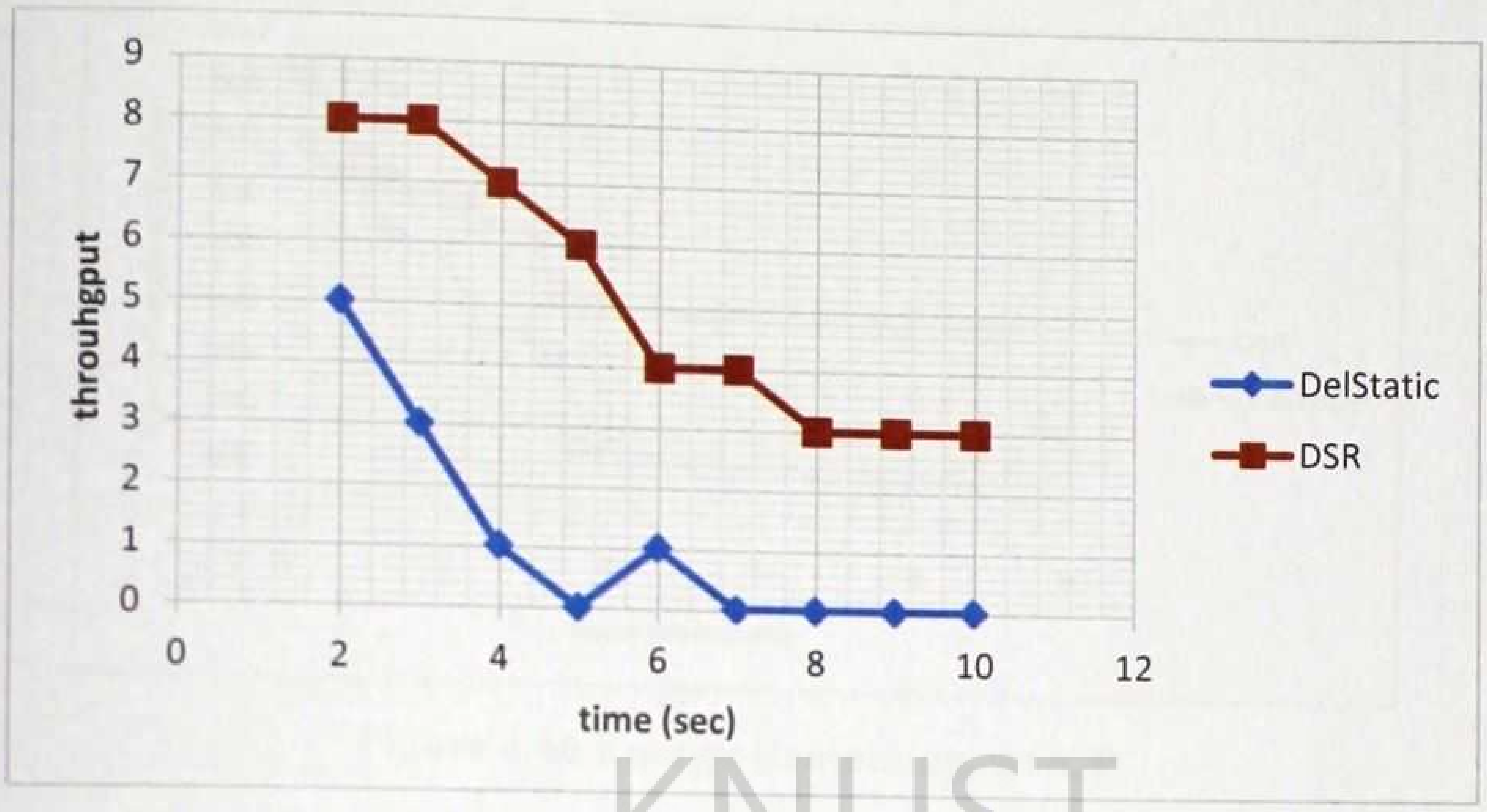


Figure 4.45 Throughput, ACCP

In Figure 4.45, ACCP protocol is tested for throughput values. As expected, the graph result shows a higher throughput value of DSR over DelStatic control. This throughput value of DSR is significant from the simulation graph obtained. A sensor application which aims to **achieve a high data accuracy level** with little preference to energy will implement ACCP protocol to switch DSR control.

4.5.2 Random Topology [3 source / 2 sink] – 150 nodes

Same topology in Figure 7.3.

4.5.2.0 Energy Remaining

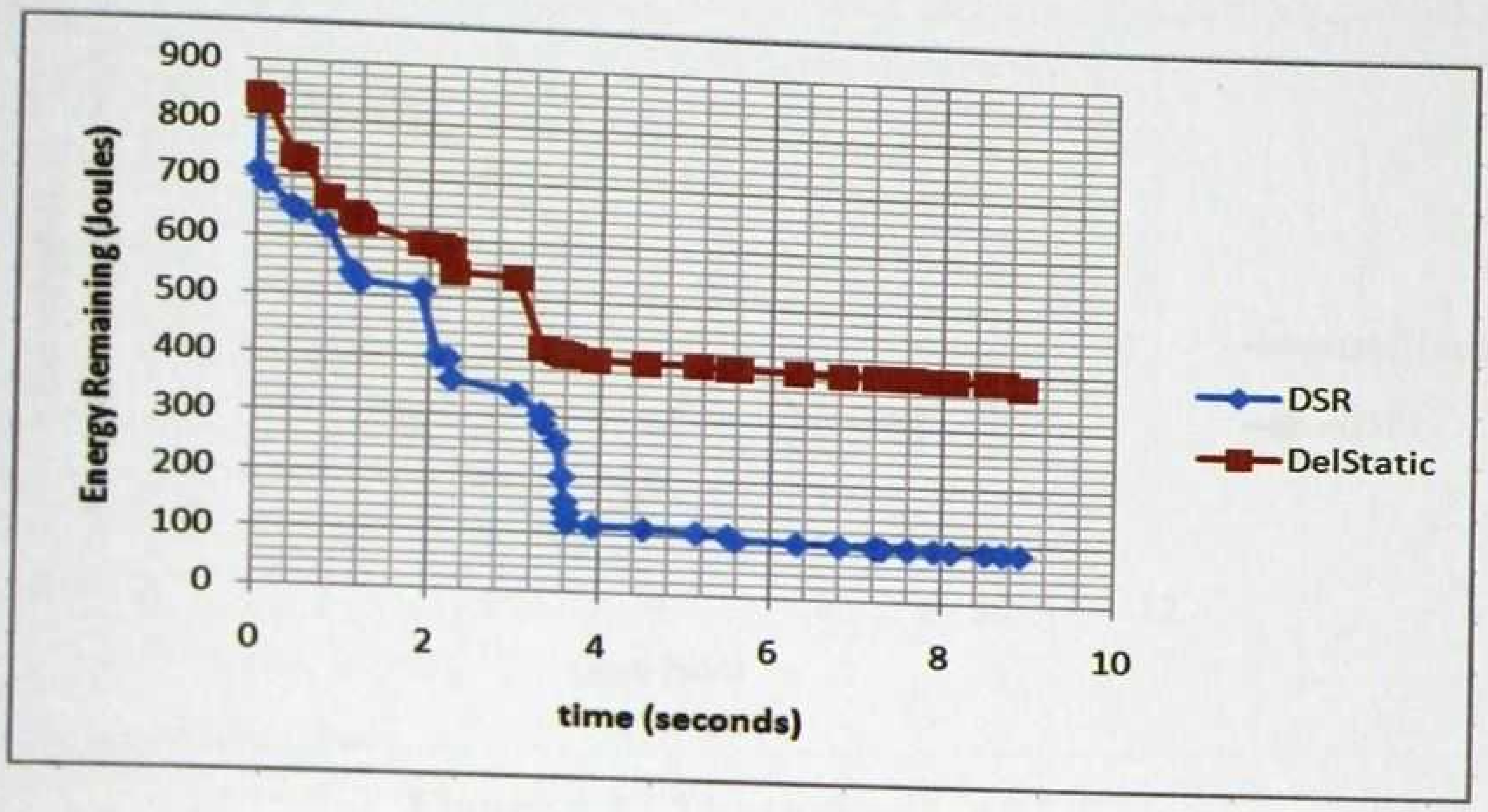


Figure 4.46 Energy Remaining, ACCP

In Figure 4.46, we increased the number of nodes involved in data transmission to 150. It is observed that, the energy consumption compared to Figure 4.44 depreciated further. This obviously is expected because of the additional nodes in the sensor experiment which consumes more energy. The simulation is carried out to confirm the results arrived at in Figure 4.44.

4.5.2.1 Throughput

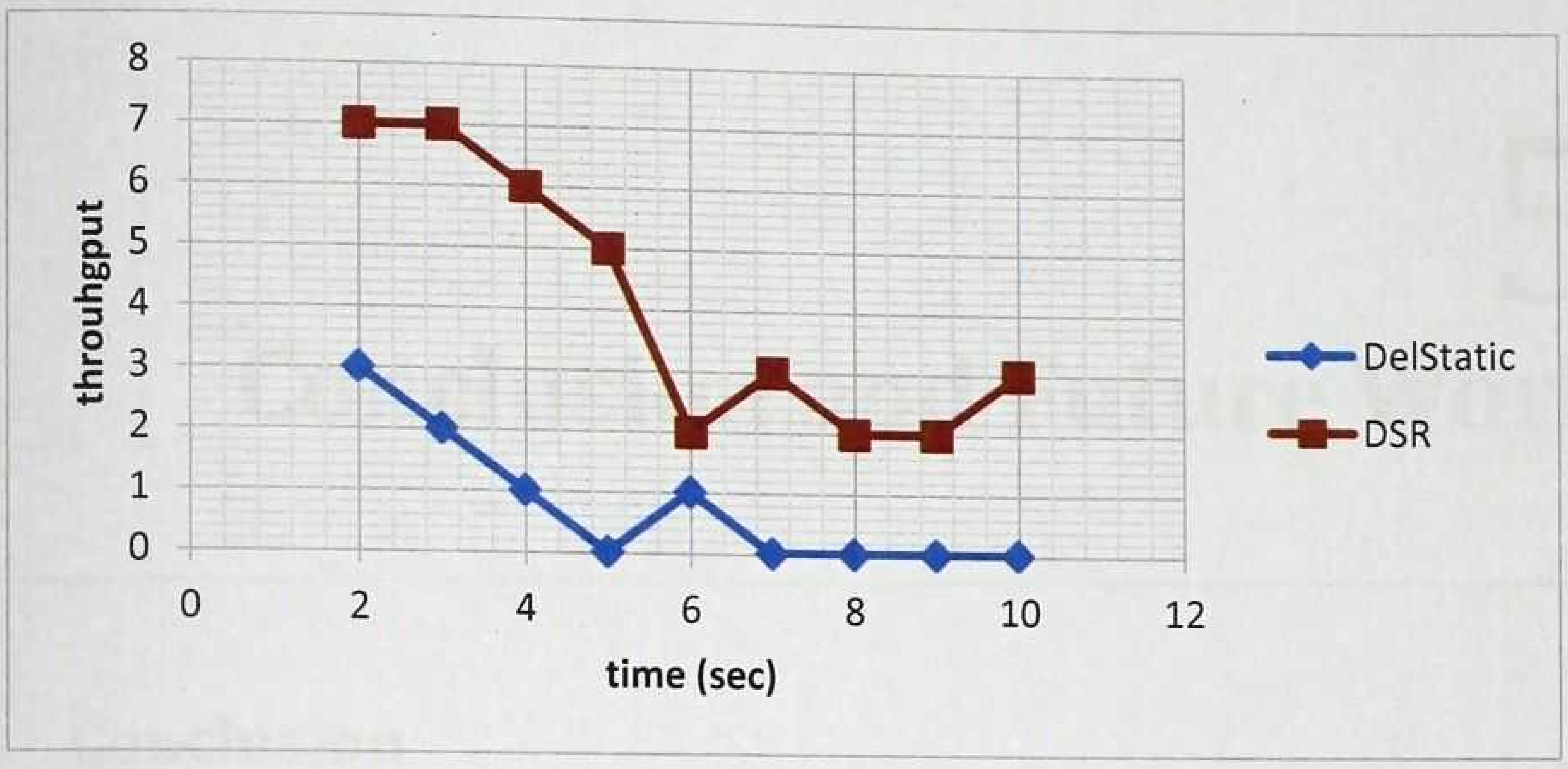


Figure 4.47 Throughput, ACCP

Throughput confirmation for ACCP protocol is carried out in Figure 4.47. It is significant to note that throughput value of ACCP protocol decreased a bit due to the additional sensor nodes. But as expected, the same conclusion in Figure 4.45 is arrived at after the simulation process.

5 Conclusion and Future Work

5.1 Conclusion

In this thesis we addressed the issue of congestion in sensor network with Adaptive Congestion Control protocol (ACCP). We proposed a protocol which tries to prevent congestion by adapting two protocols, DSR and DelStatic. This is achieved by switching a congested node to trigger DSR or DelStatic depending on the feedback information available from the controlling sink node. In contrast to most congestion control protocol, ACCP is effective in two ways:

- Energy management.
- Increase in throughput.

ACCP is based on two efficient path finding algorithm; on-demand routing and direct communication algorithms which helps to find the most efficient path to the next node. To alter or to reduce traffic is a logical sink-switched control deployed for ACCP module architecture in NS-2 framework.

All the objectives proposed in the thesis were achieved from the discussed simulation results presented in the previous chapter. We determined that; ACCP is energy efficient and has a high throughput values.

The efficiency of ACCP protocol is tested and implemented with NS-2 simulator. We incorporated changes to NS-2 module and added new protocol module ACCP and DelStatic to implement our protocol. We created different topologies with distinct sink and source nodes and made simulation runs for each of these scenarios to see how our protocol performs for varying topology and path formation.

From the result obtained, graph is generated using jTrana, awk scripts and trace files. A significant improvement on energy usage and rise in throughput values as expected is shown in the generated graph.

5.2 Future Work

Results obtained from the simulation using NS-2 is very promising and the adaptive framework is the future of congestion control in sensor network if only data accuracy and energy efficiency are to be considered a necessity to future sensor network deployment. There is much to be done on hybrid and adaptive control in sensor network. A possibility is to use a sink with directed diffusion algorithm. With this, all the nodes in the sensor network can receive the feed messages simultaneously. Future work can also include adding a fairness algorithm to the ACCP congestion control protocol.

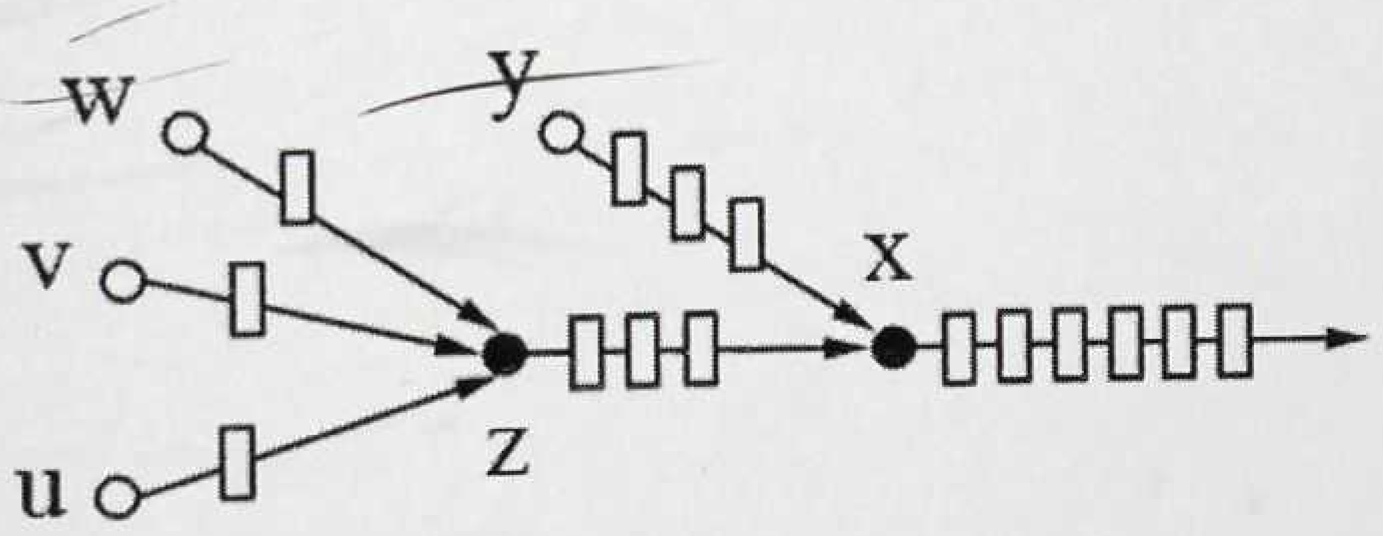


Figure 5.0 local fairness among upstream neighbors causes end-to-end unfairness

For example, in Figure 8.9, x should allocate 75% of its bandwidth to z and 25% to y. In resource congestion control, fairness becomes considerably harder due to multiple routing paths as compared to traffic congestion control. Almost 90% efficiency can be achieved if AFA algorithm is implemented with adaptive congestion control [44].

References

[1] Ghana Business News 2012, 'Jubilee oil production averages 63,100 barrels per day in 2012 first quarter', retrieved 20th April 2012, http://www.ghanabusinessnews.com/jubilee-oil-production-averages-63100-barrels-per-day-in-2012-first-quarter/.

[2] United States Environmental Protection Agency n.d., 'Oil and Gas air pollution standard', retrieved 20th April 2012, http://www.epa.gov/airquality/oilandgas/basic.html.

[3] Perrilo M, Zhao C, Heinzelman W. "On the problem of unbalanced load distribution in wireless sensor networks,", In: Regency H, ed. *Proc. of the IEEE GLOBECOM Workshops on Wireless Ad Hoc and Sensor Networks*. Dallas: IEEE Press, 2004. pp. 74-79.

[4] Wan, C.Y., Eisenman, S.B., Campbel, A.T.: CODA: Congestion Detection and Avoidance in Sensor Networks. In: the proceedings of ACM SenSys, Los Angeles, pp. 266279. ACM Press, New York (2003), pp. 146-148

[5] Sankarasubramaniam, Y., Ozgur, A., Akyildiz, I.: ESRT Event-to-Sink Reliable Transport in wireless sensor networks. In: the proceedings of ACM Mobihoc, pp. 177-189. ACM Press, New York (2003)

[6] Md. Mamun-or-Rashid, Muhammad Mahbub Alam, Md. Abdur Razzaque, and Choong Seon Hong, Reliable Event Detection and Congestion Avoidance in Wireless Sensor Networks, (2002)

[7] Xu, Y., Heidemann, J., Estrin, D.: Adaptive energy-conserving routing for multihop ad hoc networks. Technical Report Tech. Rep. 527, USC/ISI (2000)

[8] Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: An energy efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In: Proceedings of Mobicom 2001, pp. 85-89

[9] Xu, Y., Heidemann J., Estrin, D.: Geographically-informed energy conservation for ad hoc routing. Im: Proc. Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Rome (2001), pp. 15-17

[10] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad hoc Networks," *Mobile Computing*, ed. T. Imielinski and H. Korth, Kluwer Academic Publishers, 1996, pp. 153-181

[11] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient Mac Protocol for Wireless Sensor Networks. In *Proceedings of IEEE INFOCOM'02*, June 2002, pp. 121-124

[12] Shah RC, Rabaey JM. "Energy aware routing for low energy ad hoc sensor networks", In: *Proc. of the 3rd IEEE Wireless Communication and Networking Conf. (WCNC)*. Orlando: IEEE Press, 2002. 151-165

[13] Gupta G, Younis M. "Performance evaluation of load-balancing clustering of wireless sensor networks", In: *Proc. of the 10th Int'l Conf. on Telecommunications (ICT)*. IEEE Press, 2003. 1557-1583

[14] Wan, C.Y., Eisenman, S.B., Campbel, A.T.: CODA: Congestion Detection and Avoidance in Sensor Networks. In: the proceedings of ACM SenSys, Los Angeles, pp. 266279. ACM Press, New York (2003), pp. 146-148

[15] Y. Sankarasubramaniam, O. B. Akan, and I.F. Akyidiz, "ESRT: Proceedings of ACM

[16] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocols for Wireless Sensor Networks. In Proceedings of the ACM SenSys, 2003

[17] Y. Tseng, C. Hsu, and T. Hsieh. Power-Saving Protocols for IEEE 802.11 Based Multi-hop Ad Hoc Networks. In Proceedings of IEEE INFORM'02, June 2012

[18] R. Zheng, J.C. Hou, and L. Sha. Asynchronous Wakeup for Ad Hoc Networks: Theory and Protocol Design. In proceedings of the ACM MobiHoc Conference, 2003.

[19] R. Zheng and R. Kravets. On-demand Power Management of Ad Hoc Networks. In Proceedings of IEEE INFOCOM'03, March 2003

[20] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad hoc Networks," *Mobile Computing*, ed. T. Imielinski and H. Korth, Kluwer Academic Publishers, 1996, pp. 153-181

[21] M. Neely and R. Urgaonkar. Opportunism, backpressure, and stochastic optimization with the wireless broadcast advantage. IEEE SSC, Jan 2008.

[22] L. Georgiadis, M. Neely~~, M. Neely~~, and L. Tassiulas. Resource allocation and cross layer control in wireless networks. 2006.

[23] M. J. Neely, Dynamic power allocation and routing for satellite and wireless networks with time varying channels. PhD Thesis, Massachusetts Institute of Technology, 2003.

[24] M. J. Neely, E. Modiano, and C-P. Li. Fairness and optimal stochastic control for heterogeneous networks. IEEE INFOCOM, Sep 2005

[25] Wang. C., Li, B., Sohraby, K., Daneshmand, M., Hu, Y. A Survey of transport protocols for wireless sensor networks. IEEE Network, vol. 20, no. 3, May-June 2006, 34-40

[26] S. Chen and N. Yan, "Congestion avoidance based on lightweight buffer management in sensor networks," in *Proceedings of* ICPADS, pp. 934 – 946, Sep. 2006

[27] Sankarasubramaniam, Y., Ozgur, A., Akyildiz, I.: ESRT Event-to-Sink Reliable Transport in wireless sensor networks. In: the proceedings of ACM Mobihoc, pp. 177-189. ACM Press, New York (2003)

[28] Wang, C., Sohraby, K., Li, B., Daneshmand, M., Hu, Y.: A survery of transport protocols for wireless sensor networks. IEEE Network Magazine 20(3), 3440 (2005)

[29] M. Zawodniok, and S. Jagannathan, "Predictive Congestion Control Protocol for Wireless Sensor Networks". In Preceedings of IEEE Transaction on Wireless Communications, Vol.6 ~~NO.11,~~ November 2007

[30] R. Zheng, J.C. Hou, and L.Sha. Asynchronous Wakeup for Ad Hoc Networks: Theory and Protocol Design, 2008.

[31] S. Chen and Z. Zhang. "Localized algorithm for aggregate fairness in wireless sensor networks," in Proceedings of MobiCOM, pp. 274-285, Sep. 2006.

[32] Xu, Y., Heidemann J., Estrin, D.: Geographically-informed energy conservation for ad hoc routing. Im: Proc. Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Rome (2001), pp. 15-17

[33] S. Chen, N. Yang Congestion Avoidance based on Light-Weight Buffer Management in Sensor Networks, University of Florida, Gainesville, FL 32611, USA, 2005

[34] Wan, C.Y., Eisenman, S.B., Campbel, A.T.: CODA: Congestion Detection and Avoidance in Sensor Networks. In: the proceedings of ACM SenSys, Los Angeles, pp. 266279. ACM Press, New York (2003), pp. 146-148

[35] Hull, B., Jamieson, K., and Balakrishnan, H, Bandwidth management in wireless sensor networks, Tech. Rep. 909, MIT Laboratory for Computer Science, July 2003.

[36] Glauche, Ingmar, Krause, Wolfram, Sollacher, Rudolf, Martin, Distributive routing and congestion control in wireless multihop ad hoc communication networks, Elsevier, 2004.

[37] Wan, C.Y., Eisenman, S.B., Campbel, A.T.: CODA: Congestion Detection and Avoidance in Sensor Networks. In: the proceedings of ACM SenSys, Los Angeles, pp. 266279. ACM Press, New York (2003), pp. 146-148

[38] Y. Tseng, C. Hsu, and T. Hsieh. Power-Saving Protocols for IEEE 802.11 Based Multi-hop Ad Hoc Networks. In Proceedings of IEEE INFORM'02, June 2012

[39] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad hoc Networks," *Mobile Computing*, ed. T. Imielinski and H. Korth, Kluwer Academic Publishers, 1996, pp. 153-181

[40] M. Neely and R. Urgaonkar. Opportunism, backpressure, and stochastic optimization with the wireless broadcast advantage. IEEE SSC, Jan 2008.

[41] Kamal , K.S., Harbhajan, S and, R.B Patel 2010, 'A Hop by Hop Congestion Control Protocol to Mitigate Traffic Contention in Wireless Sensor Networks', International Journal of Computer Theory and Engineering, vol. 3, No. 6, pp. 1793-8201.

[44] S. Tilak, M. B. Abu-Ghazeleh, and W. Heinzelman, "Infrastructure Tradeoffs for Sensor Networks," Proc. of 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA'02), September 2002, pp. 6-9

- http://www.isi.edu/nsnam/ns/doc/node180.html
- http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/
- http://www.wisegeek.com/what-is-an-algorithm.htm

Appendix A

A1. AODV Scenario Code Written in TCL language for Grid Topology 25 nodes

```
# aodv2s1sink

# 25 grid


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

#    Simulation parameters setup

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

set val(chan)   Channel/WirelessChannel    ;# channel type

set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model

set val(netif)  Phy/WirelessPhy            ;# network interface type

set val(mac)    Mac/802_11                 ;# MAC type

set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type

set val(ll)     LL                         ;# link layer type

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

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

set val(nn)     25                         ;# number of mobilenodes

set val(rp)     AODV                       ;# routing protocol

set val(x)      1233                       ;# X dimension of topography

set val(y)      100                        ;# Y dimension of topography

set val(stop)   10.0                       ;# time of simulation end


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

#     Initialization

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

#Create a ns simulator

set ns [new Simulator]
```

```
#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open AODV2S1S.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open AODV2S1S.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#    Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType       $val(ll) \
                -macType      $val(mac) \
                -ifqType      $val(ifq) \
                -ifqLen       $val(ifqlen) \
                -antType      $val(ant) \
                -propType     $val(prop) \
                -phyType      $val(netif) \
                -channel      $chan \
            -topoInstance  $topo \
            -agentTrace    ON \
            -routerTrace   ON \
            -macTrace      ON \
            -movementTrace ON

#===================================
#       Nodes Definition
#===================================
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 220
$n0 set Y_ 516
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 420
$n1 set Y_ 516
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 620
$n2 set Y_ 516
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 820
```

```
$n3 set Y_ 516
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 1020
$n4 set Y_ 516
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 220
$n5 set Y_ 316
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 420
$n6 set Y_ 316
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 620
$n7 set Y_ 316
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 820
$n8 set Y_ 316
$n8 set Z_ 0.0
```

```
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 1020
$n9 set Y_ 316
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 220
$n10 set Y_ 116
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 420
$n11 set Y_ 116
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20
set n12 [$ns node]
$n12 set X_ 620
$n12 set Y_ 116
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 820
$n13 set Y_ 116
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
```

```
$n14 set X_ 1020
$n14 set Y_ 116
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 220
$n15 set Y_ -84
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 420
$n16 set Y_ -84
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 620
$n17 set Y_ -84
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 820
$n18 set Y_ -84
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 1020
$n19 set Y_ -84
```

```
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 220
$n20 set Y_ -284
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 420
$n21 set Y_ -284
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 620
$n22 set Y_ -284
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 820
$n23 set Y_ -284
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 1020
$n24 set Y_ -284
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20
```

```
#==========================================
#       Agents Definition
#==========================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set null2 [new Agent/Null]
$ns attach-agent $n14 $null2
$ns connect $udp0 $null2
$udp0 set packetSize_ 1500

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n5 $udp1
set null3 [new Agent/Null]
$ns attach-agent $n14 $null3
$ns connect $udp1 $null3
$udp1 set packetSize_ 1500


#==========================================
#       Applications Definition
#==========================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp1
```

```
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp0
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 2.0 "$cbr1 stop"


#===================================
#     Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam AODV2S1S.nam &
   exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

A2. NS-2 Modification TCL Script for DelStatic

```
# DelStatic2s1sink
# 25 grid


#=====================================
#     Simulation parameters setup
#=====================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     25                         ;# number of mobilenodes
set val(rp)     DelStatic                  ;# routing protocol
set val(x)      1233                       ;# X dimension of topography
set val(y)      100                    ;# Y dimension of topography
set val(stop)   10.0                     ;# time of simulation end


#========================================
#       Initialization
#========================================
#Create a ns simulator
set ns [new Simulator]


#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
```

A3. C++ Code for Implementation of New Routing Agent (DelStatic)

```
//Class file modification
class DelStaticTable;   // routing table
class DelStatic_ent;  // routing table entry
```

//Routing Agent Class

Start-up function, called in the command "start-DelStatic". In this function, we call makeRoutingtable function. routing information will be "fscanf" from a file and adding to the tables. Here, we are going to use "fscanf" to read three parameters. "int32" "int32 ""uint16", to represent <dst, ~~hop, metric~~>. The filename must be specified. Different nodes read different sections of the routing table file. The special format of this file is:

```
<src><dst> <next-hop><number of hops>
0 3 1 3
```

....

So, only those entries where 'src' addr is equal to its addresss will be read by a node.

The auto-generation of routing table could be easily implemented by an algorithm from the topology file.

// Packet Forwarding Functions

according to the routing table, the agent needs to handle both "broadcast" and "unicast" packets. "Forwarding" means determining the next-hop of a unicast packet. Once this field of the header is set, it is done. In DelStatic, broadcast occurres only locally.

The agent's interface to MAC componet is a little complex:

1) Message interface: packets going to mac with next-hop set, ready for layer 2 transmission.

2) Function interface: once link of loss is detected, the MAC will "callback" (inform) routing agent to do correspondingly. In this Fix Routing agent, this would be useless. The agent will do nothing for "mac_callback" function in this static routing design.

// Interface (linkage) to otcl

```
switch -exact $routingAgent_ {

        NOAH {

                set ragent [$self create-noah-agent $node]

        }

        DelStatic {

                set ragent [$self create-delstatic-agent $node]

        }

  ...

 }
```

the "create-delstatic-agent" procedure is also need to be defined. In this procedure, a "$self at 0.0 "$ragent start-delstatic" " will be given to the routing agent. Therefore, we have to initialize routing table in the start-delstatic command.

```
Simulator instproc create-delstatic-agent { node } {
        # Create a fix-path routing agent for this node
        set ragent [new Agent/DelStatic]
        # Setup address (supports hier-addr) for noah agent
        # and mobilenode
        set addr [$node node-addr]
        $ragent addr $addr
        $ragent node $node
        if [Simulator set mobile_ip_] {
                $ragent port-dmux [$node demux]
        }
        $node addr $addr
        $node set ragent_ $ragent
        $self at 0.0 "$ragent start-delstatic"    ;# start updates
        return $ragent
}
```

A4. Developing NOAH Routing Agent

Further update to allow static multi-hop routes. The routes can be set up using the routing command which takes as parameters the number of destinations and then as many tuples of destination and next hop address. The following example sets up static routing for a line of nodes:

```
# setup static routing for line of nodes
for {set i 0} ~~{$i < $val(nn) } {incr i} {~~
    set cmd "[$node_($i) set ragent_] routing $val(nn)"
    for {set to 0} {$to < $val(nn) } {incr to} {
        if {$to < $i} {
            set hop [expr $i - 1]
        } elseif {$to > $i} {
            set hop [expr $i + 1]
        } else {
            set hop $i
        }
        set cmd "$cmd $to $hop"
    }
    eval $cmd
}
```

//set up noah in ns-2.34

Makefile.in add noah/noah.o \ to OBJ_CC and tcl/mobility/noah.tcl \ to NS_TCL_LIB

noah/noah.{h,cc}	add noah.h and noah.cc to a new subdirectory noah/
tcl/mobility/noah.tcl	add noah.tcl to tcl/mobility/
tcl/lib/ns-lib.tcl	line 191 (for v2.34 line 197): add source ../mobility/noah.tcl

line 603ff (for v2.34 line 649ff): add

```
        NOAH {
              set ragent [$self create-noah-agent $node]
        }
```

line 768ff (for v2.29 line 839ff): add

```
Simulator instproc create-noah-agent { node } {
    # Create a noah routing agent for this node
    set ragent [new Agent/NOAH]
```

```
  ## setup address (supports hier-addr) for noah agent
  ## and mobilenode
  set addr [$node node-addr]

  $ragent addr $addr
  $ragent node $node

  if [Simulator set mobile_ip_] {
      $ragent port-dmux [$node demux]
  }
  $node addr $addr
  $node set ragent_ $ragent
  return $ragent
}
```

A5. Awk code for throughput

```
BEGIN {
     recvdSize = 0
     startTime = 400
     stopTime = 0
  }

  {
          event = $1
          time = $2
          node_id = $3
          pkt_size = $8
```

```
        level = $4


# Store start time
if (level == "AGT" && event == "s" && pkt_size >= 512) {
 if (time < startTime) {
        startTime = time
        }
  }


# Update total received packets' size and store packets arrival time
if (level == "AGT" && event == "r" && pkt_size >= 512) {
    if (time > stopTime) {
        stopTime = time
        }
    # Rip off the header
    hdr_size = pkt_size % 512
    pkt_size -= hdr_size
    # Store received packet's size
    recvdSize += pkt_size
    }
}


 END {
     printf("Average Throughput[kbps] = %.2f\t\t
StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/(stopTime-
startTime))*(8/1000),startTime,stopTime)

 }
```

A6. Awk code for end to end delay

```
#
==============================================================================
=====

# AWK Script for calculating:

#    => Average End-to-End Delay.

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


BEGIN {

    seqno = -1;

#    droppedPackets = 0;

#    receivedPackets = 0;

    count = 0;

}

{
  if($4 == "AGT" && $1 == "s" && seqno < $6) {

      seqno = $6;

  }
#        else if(($4 == "AGT") && ($1 == "r")) {

#          receivedPackets++;

#   } else if ($1 == "D" && $7 == "tcp" && $8 > 512){

#          droppedPackets++;

# }

   #end-to-end delay

   if($4 == "AGT" && $1 == "s") {

        start_time[$6] = $2;

   } else if(($7 == "tcp") && ($1 == "r")) {

       end_time[$6] = $2;

   } else if($1 == "D" && $7 == "tcp") {
        end_time[$6] = -1;

    }

}

END {

    for(i=0; i<=seqno; i++) {

        if(end_time[i] > 0) {

            delay[i] = end_time[i] - start_time[i];

                count++;

        }

        else

        {

            delay[i] = -1;

        }
```

```
}

for(i=0; i<=seqno; i++) {

    if(delay[i] > 0) {

        n_to_n_delay = n_to_n_delay + delay[i];

    }

}

n_to_n_delay = n_to_n_delay/count;


print "\n";

#   print "GeneratedPackets            = " seqno+1;

#   print "ReceivedPackets             = " receivedPackets;

#   print "Packet Delivery Ratio      = " receivedPackets/(seqno+1)*100
#"%";

#   print "Total Dropped Packets = " droppedPackets;
```

```
  print "Average End-to-End Delay   = " n_to_n_delay * 1000 " ms";

  print "\n";

}
```

Energy Sample for DSR

```
# Modified for new node structure

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

# Define options
#
==========================================================================================
========

set opt(chan)         Channel/WirelessChannel
set opt(prop)         Propagation/TwoRayGround
set opt(netif)        Phy/WirelessPhy
set opt(mac)          Mac/802_11
set opt(ifq)          Queue/DropTail/PriQueue
set opt(ll)           LL
set opt(ant)       Antenna/OmniAntenna

set opt(x)            670     ;# X dimension of the topography
set opt(y)            670              ;# Y dimension of the topography
set opt(cp)           "../mobility/scene/cbr-50-10-4-512"
set opt(sc)           "../mobility/scene/scen-670x670-50-600-20-0"
set opt(ifqlen)         50              ;# max packet in ifq
set opt(nn)             50              ;# number of nodes
set opt(seed)           0.0
set opt(stop)           10.0            ;# simulation time
set opt(tr)             aodvenergy.tr  ;# trace file
set opt(rp)         dsr         ;# routing protocol script
set opt(lm)         "off"       ;# log movement
set opt(agent)      Agent/DSR
set opt(energymodel)    EnergyModel     ;
set opt(initialenergy)  900         ;# Initial energy in Joules
#set opt(logenergy)     "on"        ;# log energy every 150 seconds

# ========================================================================
========
# needs to be fixed later
set AgentTrace                          ON
set RouterTrace                         ON
set MacTrace                    OFF

LL set mindelay_                50us
LL set delay_                   25us
LL set bandwidth_               0       ;# not used

Agent/Null set sport_           0
Agent/Null set dport_           0
```

```
Agent/CBR set sport_          0
Agent/CBR set dport_          0

Agent/TCPSink set sport_      0
Agent/TCPSink set dport_      0

Agent/TCP set sport_          0
Agent/TCP set dport_          0
Agent/TCP set packetSize_     1460

Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.2818
```

```
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0

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

proc usage { argv0 }  {
        puts "Usage: $argv0"
        puts "\tmandatory arguments:"
        puts "\t\t\[-x MAXX\] \[-y MAXY\]"
        puts "\toptional arguments:"
        puts "\t\t\[-cp conn pattern\] \[-sc scenario\] \[-nn nodes\]"
        puts "\t\t\[-seed seed\] \[-stop sec\] \[-tr tracefile\]\n"
}

proc getopt {argc argv} {
        global opt
        lappend optlist cp nn seed sc stop tr x y

        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue

                set name [string range $arg 1 end]
                set opt($name) [lindex $argv [expr $i+1]]

        }
}

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

# Main Program
#
==============================================================================
=========

getopt $argc $argv

#source ../lib/ns-bsnode.tcl
#source ../mobility/com.tcl

# do the get opt again incase the routing protocol file added some more
# options to look for
getopt $argc $argv

if { $opt(x) == 0 || $opt(y) == 0 } {
        usage $argv0
        exit 1
}

if {$opt(seed) > 0} {
        puts "Seeding Random number generator with $opt(seed)\n"
        ns-random $opt(seed)
}

#
```

```
# Initialize Global Variables
#
set ns_          [new Simulator]
set topo         [new Topography]

set tracefd      [open $opt(tr) w]

$topo load_flatgrid $opt(x) $opt(y)

$ns_ trace-all $tracefd
#Open the NAM trace file
set namfile [open dsdvenergy.nam w]
$ns_ namtrace-all $namfile
$ns_  namtrace-all-wireless $namfile $opt(x) $opt(y)
set chan [new $opt(chan)];#Create wireless channel


#
# Create God
#
create-god $opt(nn)


#
# Create the specified number of nodes $opt(nn) and "attach" them
# the channel.
# Each routing protocol script is expected to have defined a proc
# create-mobile-node that builds a mobile node and inserts it into the
# array global $node_($i)
#

        #global node setting

      $ns_ node-config -adhocRouting AODV \
                       -llType $opt(ll) \
                       -macType $opt(mac) \
                       -ifqType $opt(ifq) \
                       -ifqLen $opt(ifqlen) \
                       -antType $opt(ant) \
                       -propType $opt(prop) \
                       -phyType $opt(netif) \
                       -channelType $opt(chan) \
                       -topoInstance $topo \
                       -energyModel $opt(energymodel) \
                       -rxPower 0.3 \
                       -txPower 0.6 \
                       -initialEnergy $opt(initialenergy)

        for {set i 0} {$i < $opt(nn) } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 0          ;# disable random motion
        }

#
```

```
# Source the Connection and Movement scripts
#
if { $opt(cp) == "" } {
        puts "*** NOTE: no connection pattern specified."
      set opt(cp) "none"
} else {
        puts "Loading connection pattern..."
        source $opt(cp)
}


#
# Tell all the nodes when the simulation ends
#
for {set i 0} {$i < $opt(nn) } {incr i} {
   $ns_ at $opt(stop).000000001 "$node_($i) reset";
}
$ns_ at $opt(stop).00000001 "puts \"NS EXITING...\" ; $ns_ halt"


if { $opt(sc) == "" } {
        puts "*** NOTE: no scenario file specified."
      set opt(sc) "none"
} else {
        puts "Loading scenario file..."
        source $opt(sc)
        puts "Load complete..."
}

puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp $opt(rp)"
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

puts "Starting Simulation..."
$ns_ run
```

A8. Full Thesis Material with completed code on CD.

For detailed codes on TCL files, awk fiels, tracefiles, simulation results are included in "CD" attached and submitted to my Supervisor.