Deanship of graduate studies

AL-QUDS University

End-to-End Protocol for Dynamic Congestion Avoidance in MANETs

Mohammed Jamal Bawatna

M.Sc Thesis

Jerusalem – Palestine

May - 2015

End-to-End Protocol for Dynamic Congestion Avoidance in MANETs

Prepared by:

Mohammed Jamal Bawatna

B.Sc.: Computer Engineering, AL-QUDS University,
Palestine

A thesis submitted to Faculty of Engineering, AL-QDS
University in partial fulfillment of the requirements for
the degree of Master of Electronic and Computer
Engineering

Supervisor: Dr. Rushdi Hamamreh

AL-QUDS University

May, 2015

Electronic and Computer Engineering Master

Program Faculty of Engineering

AL-QUDS University

Thesis Approval

End-to-End Protocol for Dynamic Congestion Avoidance in MANETs

By:

Student name: Mohammed Jamal Bawatna

Reg. No: 21210142

Supervisor: Dr. Rushdi Hamamreh

Master thesis submitted and accepted. Date:________________

The name and signature of the examining committee members are as follows:

1. Head of Committee: Dr. Rushdi Hamamreh Signature:-------
2. Internal Examiner: Dr. Samer Bali Signature:-------
3. External Examiner: Dr. Mohammed Aldasht Signature:-------

Dedication

This thesis is dedicated to my beloved Parents, who have raised me to be the person I am today;

Wife, who has been with me every step of the way through good times and bad;

Sisters and brother, for their encouragement;

Thank you all, I love you!

Declaration

I certify that this thesis submitted for the degree of Master is the result of my own research, except where otherwise acknowledged, and that this thesis has not been submitted for higher degree to any other university or institution.

Signature: ……………..

Yours faithfully,

Mohammed Jamal Bawatna

Date: ………………….

Acknowledgements

I am deeply thankful to Dr. Rushdi Hamamreh who encourages me and taught me how to do research, and at last but not least to all my teachers at AL-QUDS University who do their best to put me in the road of scientific research and provide me with support and encouragement.

Abstract

Network congestion is one of the most challenging issues that degrade the performance of Transmission Control Protocol (TCP) over Mobile Ad hoc Networks (MANETs). In MANET congestion can occur due to limitation in resources and leads to high packet loss, long delay and waste of resource utilization time. The great demand for capacity, place particular emphasis on congestion management approaches. The major objective of congestion control is to best utilize the available network resources by keeping the load below the capacity.

TCP is designed to be reliable and ensure end-to-end delivery in wired network. However, each existing TCP variant over MANET has its weaknesses and strengths when changing MANET factors like: node mobility, traffic loads, network size and wireless channel conditions.

In this thesis a new dynamic end-to-end path congestion estimation mechanism TCP-DCM is proposed based on the measured value of Round Trip Time (RTT) and the results of route request during the route discovery process. Not only has this proposed mechanism the ability to increase the performance of TCP over MANET, but also best utilize the available network resources by keeping the load below the capacity.

Our Dynamic Congestion Model (DCM) is evaluated by five network performance metrics: the overall throughput, the End-to-End Delay, the normalized Overhead, the Packet Delivery and the overall energy consumption.

The simulation results over both AOMDV and DYMO routing protocols is compared with five most used TCPs which are TCP-Newreno, TCP-Vegas, TCP-

Westwood, TCP-WELCOME and ATCP, and show an improvement in TCP performance over MANET in different scenarios in comparison with other congestion control techniques used in TCPs. The best overall network utilization over AOMDV is 0.66 when packet length = 7 Kbytes and node dense = 20 nodes, while the best overall network utilization over DYMO is 0.71 when packet length = 6 Kbytes and node dense = 20 nodes.

البروتوكول الديناميكي لتجنب نقاط الإزدحام بين طرفي الإتصال
End-to-End Protocol for Dynamic Congestion Avoidance in MANETs

إعداد الطالب: محمد جمال محمد بواطنه
إشراف: د. رشدي حمامرة

الملخص

تعتبر مشكلة الإزدحام من أهم المشاكل التي تواجه البروتوكول TCP في الشبكات المتحركة اللاسلكية (MANET), حيث أن كفاءة هذا البروتوكول تنحدر بشكل ملحوظ عند زيادة الإزدحام على نقاط الشبكة التي تشكل الممرات لعبور المعلومات من المرسل للمستقبل على شكل حزم (Packets), بحيث تكون الداخل من الحزم أكثر من قدرة هذه النقاط على المعالجة اللحظية وبشكل يفوق القدرة الإستيعابية للوحدات التخزينية في هذه النقاط المتحركة, الأمر الذي يؤدي إلى حذف الفائض من الحزم وبالتالي انحدار كفاءة وصول المعلومات من النقاط المرسلة إلى النقاط المستقبلة.

لحل هذه المشكلة, قام العديد من الباحثين بإبتكار عدد من الآليات والتقنيات التي تواجه زيادة الإزدحام على بعض النقاط نتيجة استخدام هذه النقاط كممرات لتوصيل الحزم من المرسل للمستقبل وذلك عن طريق التحكم بنافذة الإزدحام (CWND) ومحدد ابتداء تجنب الإزدحام (ssthresh), ولكن يوجد لكل من هذه التقنيات إيجابيات وسلبيات عند تغير الظروف في الشبكات اللاسلكية المتحركة.

في هذه الرسالة تم اقتراح وتصميم بروتوكول جديد لحل مشكلة الإزدحام عن طريق إختيار الممرات الأقل إزدحاما بين المرسل والمستقبل. و يظهر البروتوكول المقترح تحسن قدرة وكفاءة بروتوكول TCP في الشبكات المتحركة اللاسلكية بإستخدام المحاكاة (simulation) في جميع السيناريوهات بالمقارنة مع التقنيات الموجودة حاليا لحل مشكلة الإزدحام من حيث زيادة الإنتاجية (Throughput) وتقليل معدل التأخيير بين طرفي الإتصال (Average End-to-End Delay) وزيادة نسبة تسليم الحزم (Packet Delivery Ratio) وتقليل إجمالي إستهلاك الطاقة (Total Energy Consumption) والنفقات المصاحبة (Normalized Overhead).

حيث يقوم البروتوكول الديناميكي بتجنب الممرات الي يوجد بها نقاط إزدحام تفوق حد معين واختيار ممرات أقل إزدحاما, بل ومعالجة الإزدحام بطريقة تتناسب والطبيعة الديناميكية للشبكات اللاسلكية المتحركة.

Table of Contents

Introduction

Mobile Ad Hoc Networks

Literature Review: Congestion Control Techniques

DCM: Dynamic Congestion Model

Simulation Results and Analysis

Conclusion and Future Works

List of Tables

List of Figures

List of Appendices

Chapter One

Introduction

Contents

Introduction

1.1 Introduction

Wireless ad hoc network is a temporary network connection for a specific purpose of transferring data from one node (source) to another node (destination) in wireless channels. Applications that transfer data between network nodes use one of two major techniques: connectionless based or connection oriented based. The transmission control protocol (TCP), which is the most predominant transport layer protocol in the Internet today, is a connection oriented based protocol. The TCP does not only provides an end-to-end reliable connection and in order delivery of packets, but also responsible for flow control and congestion control. Congestion control, which is the scope of this thesis, handles the overflow traffic in the network that leads to degradation in the performance of the network.

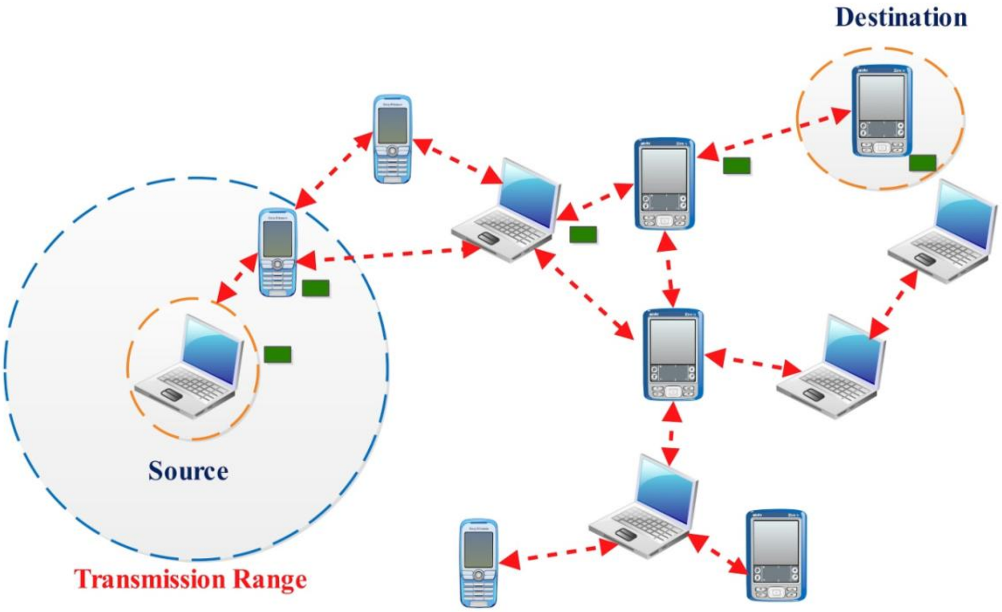


Figure 1-1: Mobile Ad Hoc Networks (MANETs)

In mobile ad hoc networks (MANETs), communication happens via wireless means and it can be between heterogeneous wireless nodes. Besides, there is no centralized control node in MANET since there is no pre-existing Infrastructure; every node has to play the roles of both hosts and routers. In addition to the dynamic topology in MANET which leads to frequent routing updates, resources shared in MANET are mostly the bandwidth of the links and the queues on the routers or switches [1]. These special characteristics of MANET make some critical challenges issues of TCP which is designed to work on a wired network. But in the case of wireless ad hoc network, where the level of noise is not negligible due to the physical medium, all developed mechanisms to overcome the technical deficiencies of TCP have many weaknesses when changing network factors. Packet loss in wired network is identified as network congestion. Although the numerous proposed mechanisms in TCP to handle network congestion, such as slow rate, Active Queue Management (AQM) and Additive Increase Multiplicative Decrease (AIMD) [2], these proposed mechanisms are not designed to handle packet loss due to wireless channel errors and frequent link failure due to dynamic mobility. The problem of link failure increase Retransmission Time Out (RTO) exponentially and that remains in high value even when a new route is discovered. Recently, lots of effective congestion control techniques within TCP variants are proposed to overcome the shortcoming of TCP over MANET.

Each proposed technique has strengths and weaknesses when changing network factors. This thesis focuses on improving the overall of TCP performance over MANET by redesigning the conventional congestion control mechanism.

1.2 Motivation

Transmission Control Protocol (TCP) is a reliable end-to-end transport protocol that is primarily designed for wired networks; in addition it became a very robust and efficient protocol. However, recent researches showed that congestion control techniques that implemented in TCPs perform very poorly over Mobile Ad Hoc Networks (MANETs) and degrade the throughput. This TCP’s problem over MANETs presents the need for design a new efficient and applicable congestion control technique based on the dynamic characteristics of the MANETs.

The important of design a new robust congestion control technique that based on the special characteristics of the Mobile Ad Hoc Networks is became a major challenge.

Congestion control is essential for data communication networks. With congestion control, sender node decides on how fast they can send packets to receiver node over the network. An effective congestion control protocol provides a robust and fair sharing of the underlying network capacity among multiple competing applications.

There are many applications such as streaming video and Internet Telephony, prefer timeliness to reliability in order to be useful for data arriving within a certain deadline. Although these new applications often choose User Datagram Protocol (UDP), the long lasting UDP flows without any congestion control mechanism present a serious threat of network collapse to the Internet. In addition, congestion control techniques are difficult to implement and may behave incorrectly.

The cost for providing reliability and in-order delivery is an arbitrary delay. This presents the need for a dynamic end-to-end model, which provides choices of congestion control algorithm selection or implementation. In this thesis, we study the congestion control problem over mobile ad hoc networks.

1.3 Objectives of this thesis

In this thesis, the main objectives are to improve the performance of TCP over MANET environment through decreasing the technical deficiencies of the previous TCP variants while changing the MANETs factors. The performance improvement is depending on replacing the traditional congestion technique that is developed to work for wired network or infrastructure-based wireless networks with more adaptive dynamic end-to-end technique that is more suitable for MANETs.

1.4 Problem Statements

The problem of TCP over Mobile Ad Hoc Network (MANET) is applying congestion control algorithms to types of packet drop that are not lost due to network congestion. When a packet is detected to be lost, either by timeout or by duplicating Acknowledgments (ACKs), TCP decrease the sending rate by adjusting its congestion window (CWND). Many approaches use Round Trip Time (RTT) and Bandwidth (BW) estimation, but none of them work perfectly in all scenarios without any problems. Recent researches show that TCP-WELCOME performs much better than other TCP variants over MANET, because its ability to differentiate between different types of packet losses. Although TCP-WELCOME success to identify the causes of packet drops over MANET, it implements the traditional congestion algorithm used in TCP-NewRENO which is not designed to operate in dynamic topology such as MANET. This weakness point of TCP-WELCOME makes its performance degrade in congested MANETs, which is the subject of this thesis.

1.5 Thesis Contributions

There are several challenges that decrease the performance of TCP over MANET by misleading the cause of packets losses. The network congestion is one of the most challenges that have to be controlled in more dynamic way to fit the needs of MANETs. In this thesis we have done the following:

- We designed and developed a new Dynamic end-to-end Congestion Avoidance Model for MANETs (TCP-DCM) that avoid the congested nodes in the network and select paths with minimum congestion.
- We designed and implement a new congestion control mechanism to control the CWND and *ssthresh* in more dynamic way.
- We have enhanced the AOMDV routing protocol to be controlled by TCP layer in order to change the congested path.
- We redesigned the DYMO routing protocol to keep at least three minimum congested paths.
- We validate the new DCM using network simulator ns2 with comparison with TCP-WELCOME, ATCP, TCP-Westwood, TCP-Vegas and TCP-NewReno.

1.6 State of Arts

In Mobile Ad hoc Network (MANETs), the principle problem of TCP is its inability to differentiate between packet losses due to network congestion and other types of losses. TCP deals with packet losses as network congestion. Although this assumption is valid over wired networks, it is not valid over MANETs.

There are several types of packet losses in MANETs, including losses by wireless channel errors, losses due to link failure and losses due to network congestion. In order to overcome this problem, several proposals have been made. These proposals are classified into two main categories: single layer solutions and cross-layer solutions. In single layer solutions, the cooperation between sender and receiver is performed to control the Congestion Window (CWND) in order to decrease the packet losses. In cross-layer solutions, the provided information from intermediate nodes is used to avoid network congestion.

In wireless multi-hop networks, the performance of TCP depends on its ability to dynamic estimate the available bandwidth, which depends on length of the path and stability of the network. Therefore most TCPs over Mobile Ad Hoc Networks suffer from performance degradation.

In order to provide network stability, each end-to-end mobile node has to control the transmitting data rate by two sliding windows: Receiver Window and Sender Window. They are used for preventing the transmitted data running over receiver's buffer and network capacity.

The effort paid to improve TCP end-to-end throughput, fairness is a critical issue that definitely deserves more attention. It is shown that in MANETs with multiple flows, the throughput can be significantly different among competing flows. This is particularly evident when comparing flows of short paths to those of long paths. It is crucial for every flow to fairly share the network resource in MANETs, as the network capacity is so limited compared with its counterpart in wired networks.

Most versions of TCPs use facilities to detect network congestion such as Explicit Congestion Notification (ECN) [3] and Random Early Detection (RED) [4]. These facilities are designed to enable the TCP senders to response faster to network congestion in intermediate nodes. In addition, these two facilities provide a feedback to the TCP senders about the congestion status information.

In feedback approaches, the sending node depends on information that obtains from network layer. When a link failure is detected by the intermediate node, then a notification message will be sent to the sender node to enter in a freeze state [5][6]. The main drawback of this approach is the extra overhead produced due to transmission notification packet.

The lack of available information about the intermediate status limits the TCP sender from fast react on the current condition. Although the use of ECN and RED provide a feedback about network congestion status to sender, these two facilities are not able to provide information about the available bandwidth to the TCP senders.

Many recent approaches [7][8][9] have approved that TCPs with a small Congestion Window (CWND) size has better performance over Mobile Ad Hoc networks, since this technique tends to inject more packets to network near the limit that leads to high probability of collision.

[M. Mancuso] propose a sender-side technique of end-to-end Loss Differentiation Algorithm (LDA) and adaptive segmentation in order to improve the TCP performance in heterogeneous networks. This technique enables the sender node to differentiate between packet losses due to network congestion from packet loss due to wireless channel errors [10].

[L. Stephane et. al] propose a new cross layer solution that based on Loss Differentiation Algorithm (LDA) to identify the packet loss causes and response with appropriate recovery algorithm. The proposed solution implemented at the Media Access Control Layer (MAC) in order to enhance the end-to-end TCP performance. The main drawback of this solution is considering only the last node to be wireless connected [11].

[Yao-Nan Lien et al] proposed a new TCP congestion control technique that uses the information provided by intermediate nodes to achieve better congestion control. In order to use this technique, intermediate nodes are required to provide information that enables the sender node to estimate more accurately the remaining capacity over the bottleneck node in the path between the senders to the receivers [12].

The frequent link failures in MANETs environment introduce new challenges to TCP congestion control techniques and leads us to think about the validity of the current congestion control techniques over dynamic environment such as MANETs and the methods used to calculate the retransmission time out (RTO) after discovering link breakage. Thus extra work has to be done to enhance the TCP performance over MANETs.

1.7 Thesis Structure

The rest of this thesis is structured as follows:

- **Chapter Two: Mobile Ad Hoc Networks**

  Chapter Two presents the environment of Mobile Ad Hoc Networks (MANETs) with its main features and constraints.

- **Chapter Three: Literature Review: Congestion Control Techniques**

  In Chapter three, a literature review of the current congestion techniques and research methodologies proposed for solving congestion problems that degrades the performance of TCP over MANETs; the research will include techniques used in TCP-NewReno, TCP-WESTWOOD, TCP-VEGAS, ATCP and TCP-WELCOME with each stringiness and weakness. We will focus on TCP-WELCOME.

- **Chapter Four: DCM: Dynamic Congestion Model**

  Chapter Four: proposes the new dynamic end-to-end congestion avoidance technique in TCP protocol (TCP-DCM) that overcomes the environmental shortcomings and technical deficiencies of TCP over MANET environment.

- **Chapter Five: Simulation Results and Analysis**

  In Chapter five, we will present the results of simulation and analysis to show how the new proposed TCP protocol improves the overall performance while changing MANET factors.

- **Chapter Six: Conclusion and Future Works**

  Chapters Six concludes this thesis with a summary of what has been achieved, and it provides directions for future work.

Chapter Two

Mobile Ad Hoc Networks

Contents

Mobile Ad Hoc Networks

2.1. Introduction

Mobile Ad Hoc Network (MANET) is infrastructure-less based wireless networks. Therefore there is no centralized control node to handle the transaction between the nodes. MANET is a self organized network [13], in which every node plays the roles of both router and host by sharing its resources such as the link's bandwidth and queue. Besides, nodes in MANET can be heterogeneous and can change their locations, which leads to frequent routing updates in order to cover the connectivity between source and destination. MANET is useful when infrastructure is expensive or not available. This dynamic environment is essential in a wide variety of applications such as Search And Rescue (SAR) [14] operations, military environments, meeting rooms, taxi cab network, and Personal Area Networking (PAN) applications.

We will present the main algorithms used in routing and the challenges that face the implementation and evaluation of routing process over MANETs. After that we will discuss the basic operations used in most famous MANETs Routing Protocols: Ad hoc On-demand Distance-Vector Routing (AODV) and Dynamic MANET On-demand Routing (DYMO), Ad hoc On-demand Multipath Distance-Vector Routing (AOMDV), Temporally Ordered Routing Algorithm (TORA), Destination-Sequenced Distance-Vector (DSDV) and Dynamic Source Routing (DSR). Also, we will provide a comparison between MANETs Routing Protocols and finally the conclusion.

2.2. Routing Algorithms

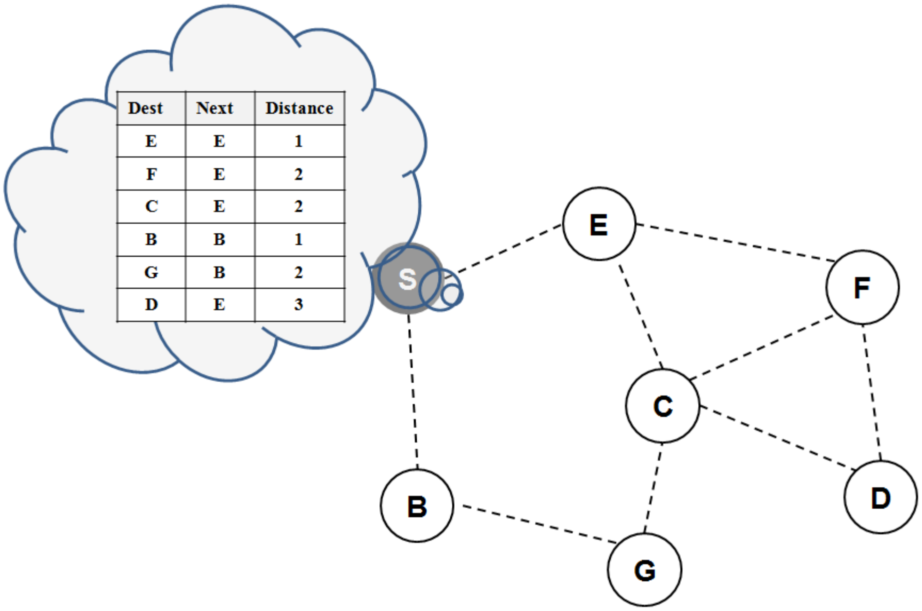
Routing process consists of two basic steps: forwarding packets to next hop that based on network interface and determining how to forward packets that based on algorithm used in routing table. The main objectives of existing routing algorithms are to minimize delay, minimize packet loss and reach the destination with minimum number of hops and cost.

Routing table contains information that used to determine how to forward the packets in two ways: source routing (in which route path is specified in the packets) and hop-by-hop routing (in which routing table contain information about the next hop for each given destination).

The most used distributed algorithms to build routing table are: Distance Vector Algorithms and Link-State Algorithms [15].

1- Distance Vector Algorithms

In this algorithm routing table at each node specify the next hop for each destination as in [16][17] and the distance for that destination.



Dest	Next	Distance
E	E	1
F	E	2
C	E	2
B	B	1
G	B	2
D	E	3

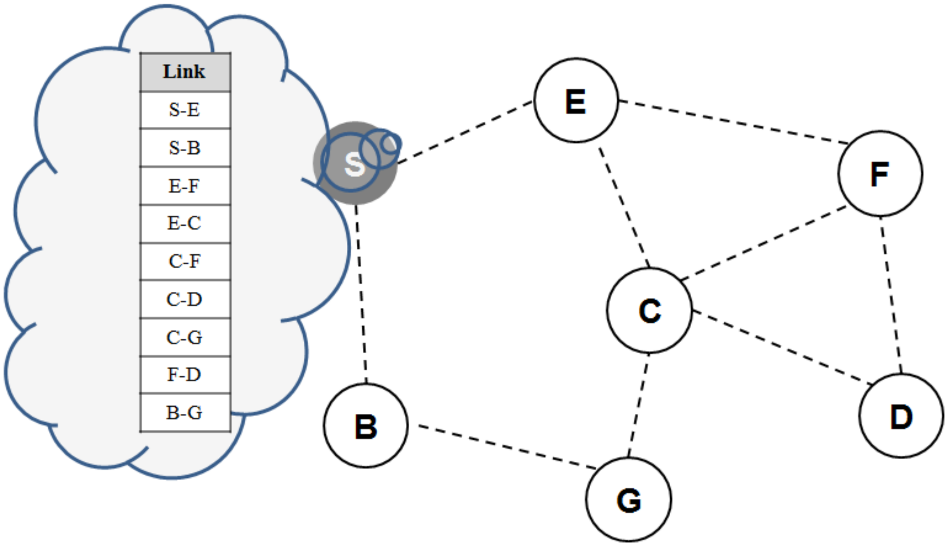
Figure 2-1: Routing table at node S based on distance vector algorithm.

If we want to consider the path from node S to node D as in Figure 2-1, let $d_S(D)$ is the cost of least-cost path from node S to node D and $c(S,E)$ is the cost of direct link from node S to node E, then the distance from node S to node D is expressed as follows:

$$d_S(D) = \min_v (c(S,E) + d_E(D)) \tag{2.1}$$

2- Link State Algorithms

In this algorithm each node shares its link information with its neighbors in order to build a map of the whole network topology as in [18]. The link information is updated when link change as in Figure 2-2. The main algorithm used is the Dijkstra’s Shortest Path Algorithm as in [19].



Link
S-E
S-B
E-F
E-C
C-F
C-D
C-G
F-D
B-G

Figure 2-2: Routing table at node S based on link-state algorithm.

2.3. MANETs Routing Challenges and Classifications

The special characteristics of MANET environment make the implementation and evaluating of routing techniques face several challenges as in [20][21] such as:

1- Hardware constraints: software is implemented in wireless small devices; such devices have obviously limited features, in that power, processing speed, memory and bandwidth.

2- Heterogeneity: nodes should not expect other nodes to have similar abilities because a single network will include different kind of devices.

3- Distributed processing: because MANETs are not centralized, it relies on distributed algorithms. A single node may not be able to handle all the transaction of data and process it.

4- Route discovery and control: nodes are wireless and the network topology is not fixed, so nodes need to discover the identity of others in controllable way.

5- Multi-hop routing: nodes need to rely on other nodes to forward messages to their final destination when the destination is not directly reachable.

Due to these several challenges network routing protocols are classified into three categories [22]: hierarchical routing, flat routing and geographic position assisted routing. In hierarchical routing protocols the idea of clustering is used as in [23] and this type of protocols is suitable for a huge numbers of nodes.

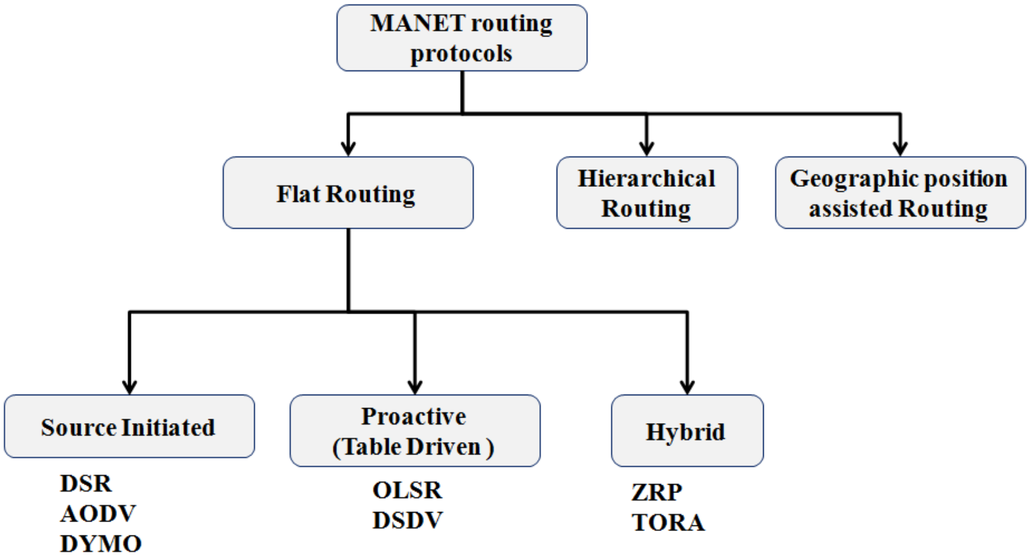


Figure 2-3: Classification of MANET routing protocols

Flat routing is divided into three sub-categories as in Figure 2-3: Source initiated, Table Driven and Hybrid. Source initiated routing protocols is the subject of this thesis for the reasons follows:

1- Table Driven (Proactive): In this type the routes to all destinations are determined at the start up and maintained by using a periodic route update process. The most famous protocols in this sub category are: Optimized Link State Routing (OLSR) and Destination-Sequenced Distance-Vector (DSDV). The Advantage of this type is that routes always available. The Disadvantage of this type is the very high control overhead needed to maintain all routes [38][39].

2- Source Initiated (Reactive): In this type the route is determined only when it is required by the source, and it is maintained as long as it is needed. The most famous protocols in this sub category are: Ad hoc On-demand Distance-Vector Routing (AODV), Dynamic MANET On-demand Routing (DYMO) and Dynamic Source Routing (DSR) The Advantage of this type is the low control overhead needed since it is in demand. The Disadvantage of this type is the high initial delay needed to discover the route to destination [25][26][27].

3- Hybrid: In this type routing protocols try to combine the main features of proactive and reactive protocols to overcome the control overhead which is the disadvantage of proactive protocols and the initial delay which is the disadvantage of reactive protocols. Main feature of Hybrid Routing protocol is that the routing protocol tries to act as proactive for short distances and reactive for long distances. The most famous protocol in this sub category is: Zone Routing Protocol (ZRP). The main drawbacks of hybrid routing protocols is that the nodes have to maintain high level topological information which leads to more memory and power consumption and long delay if route to destination not found immediately[24][33].

Although the table driven (proactive) protocols provide the availability of the routing path between nodes, these protocols consume the resource nodes with routing control messages. Routing control messages will become a serious problem when network size increase and the overall overhead will be a challenging issue. We need to minimize the routing state at each node.

2.4. MANETs Routing Protocols

In this section we will discuss the routing mechanism used in most used flat routing protocols. The discussion will include Optimized Link State Routing (OLSR), Destination-

Sequenced Distance-Vector (DSDV), Dynamic Source Routing (DSR), Ad hoc On-demand Distance-Vector Routing (AODV) and Dynamic MANET On-demand Routing (DYMO).

2.4.1 Ad hoc On-demand Distance-Vector Routing (AODV)

AODV is a reactive routing protocol that uses next hop routing approach. Each node maintains a single path to a destination. AODV consists of two routing operations: Path discovery process and Path maintenance process [25]. Every node maintains two separate counters: Sequence Number and Broadcast ID.

Source node starts path discovery by broadcasting a route request (RREQ) message to its neighbors, which includes source address; source sequence number; broadcast id; destination address; destination sequence number; and hop count. If the receiving node is an intermediate node with a valid route to destination, it will send back a Route Reply (RREP) message using the reverse path only if RREQ's sequence number is smaller than that recorded in the intermediate node, or sequence numbers are equal to smaller hop count in the intermediate node. Otherwise the intermediate node will rebroadcast the RREQ. If the receiving node is the destination node itself, it will send back a RREP using the reverse path.

A RREP contains the following information: source address, destination address, destination sequence number, hop count, and lifetime. In path maintenance process, each intermediate node along the path from source to destination has to monitor the link state. If the link is discovered to be broken, then RERR message is generated from the intermediate node that discovers the link break toward the source node.

As an example, suppose source node S reach destination node D as in Figure 2.1.

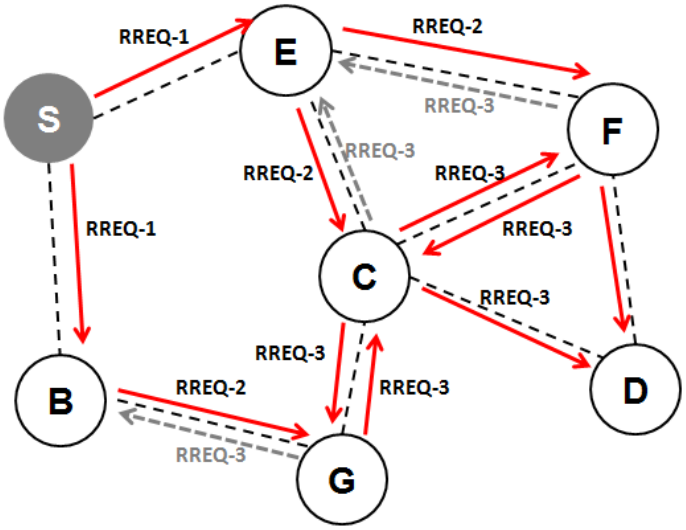


Figure 2-4: Route discovery process. Arrows in dot style are discarded RREQ messages to prevent looping problems.

Initially, source node broadcast RREQ message for one hop nodes as in Figure 2-4, to nodes E and B. This continues until RREQ message reach the destination node. When RREQ message reach the destination node or a node that know a valid path to destination node it generate RREP message through that path to source node as in Figure 2-5.

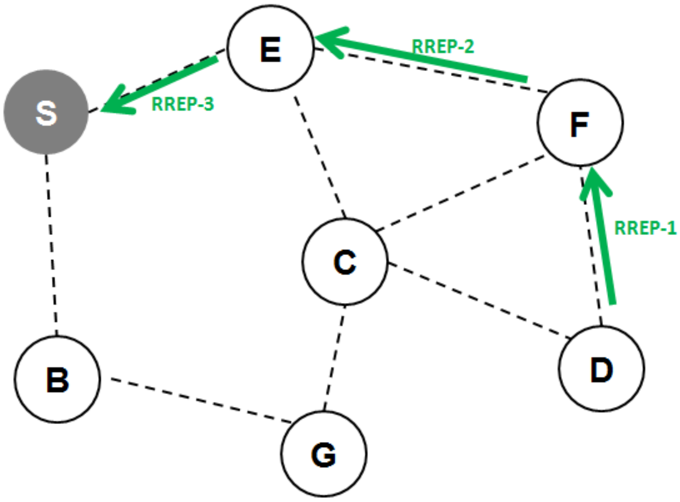


Figure 2-5: Route Reply process.

AODV stores only one route per destination with a certain lifetime. Once a route is established, it must be maintained as long as the route expiration time does not expire. This is done by exchanging “hello” messages periodically.

2.4.2 Dynamic MANET On-demand Routing (DYMO)

Dynamic MANET on Demand routing protocol (DYMO) is a reactive protocol which consists of two operations: route discovery and route maintenance [26]. DYMO is enhanced from AODV or ADOVv2 protocol. The performance metrics such as the overhead and the energy efficiency show that the DYMO protocol is better than AODM protocol for large network size [27]. Besides the DYMO protocol outperforms AODV in performance, it consumes less memory for routing table.

There are two differences between AODV and DYMO during route discovery which are: DYMO allows intermediate nodes to attach or remove additional information to routing messages if they believe that appended information will help [29]. The other difference is related to energy efficiency; if the node’s energy is low, then it has the option of decide whether or not to participate in the route discovery process.

2.4.3 Ad hoc On-demand Multi-Path Distance-Vector Routing (AOMDV)

AOMDV shares several characteristics with AODV [32]. It is based on the distance vector concept and uses multi hop routing approach. In addition, AOMDV also finds routes on demand using a route discovery procedure. The main difference lies in the number of routes found in each route discovery. In AOMDV, RREQ propagation from the source towards the destination establishes multiple reverse paths both at intermediate nodes as well as the destination [33].

Multiple RREPs traverse these reverse paths back to form multiple forward paths to the destination at the source and intermediate nodes. Note that AOMDV also provides intermediate nodes with alternate paths as they are found to be useful in reducing route discovery frequency [34].

AOMDV doesn’t require any special type of control packet to control the overall processing, but use the control mechanism of AODV with an extra field in the header. Multiples Loop-Free paths are achieved using the advertised hop count method at each node. This advertised hop count is required to be maintained at each node in the route table entry. The route entry table at each node also contains a list of next hop along with the corresponding hop counts.

Every node maintains an advertised hop count for the destination. Advertised hop count can be defined as the maximum hop count for all the paths [33]. Route advertisements of the destination are sent using this hop count. An alternate path to the destination is accepted by a node if the hop count is less than the advertised hop count for the destination.

The basic structure of a routing table entry in the AOMDV in comparison with AODV is shown in Table 2-1.

Table 2-1: Routing table entries for AODV and AOMDV [32]

AODV	**AOMDV**
Destination	Destination
Sequence number	Sequence number
Hop count	Advertised_hopcount
Expiration_timeout	Expiration_timeout
_timeout Nexthop	Route_list: {(nexthop1,hopcount1), (nexthop2,hopcount2),…}

There are two main differences:

1- The hop count is replaced by advertised hop count in the AOMDV
2- The next hop is replaced by the route list. The route list is simply the list of next hops and hop counts corresponding to different paths to the destination. The advertised hop count represents the maximum of the hop counts of each of those multiple paths so long as a route update rule is followed.

In order to ensure loop freedom, a node receives a RREQ or RREP packet from a neighbor. As in AODV, routes corresponding to only the highest known sequence number for the destination are maintained. However, AOMDV allows for multiple routes for the same destination sequence number. Multiple routes can form via any neighbor upon receiving a RREQ or RREP from that neighbor.

2.4.4 Temporally Ordered Routing Algorithm (TORA)

The Temporally-Ordered Routing Algorithm (TORA) [35] is a hybrid routing protocol that provide multi-path routing and minimization of the overhead by localization [36] algorithmic. TORA routing protocol TORA provides both of reactive and proactive routing mechanisms. This routing protocol can be separated into three basic operations: creating routes, maintaining routes and erasing Routes.

The reactive mechanism that is provided by TORA is that source node can imitate a route discovery to unknown destination on demand and this route is no longer kept after the end of communication between source and destination. The proactive mechanism that is provided by TORA is the ability to resembling traditional table-driven routing approaches to selected frequent required destinations.

The major disadvantage of TORA routing protocol is that as number of source nodes that transmitting simultaneously is increase, the performance of this routing protocol is decrease due to congestion. Loses of packets makes TORA sends out more UPDATE packets [37] to reconfigure, things that leads to more congestion.

2.4.5 Destination-Sequenced Distance-Vector (DSDV)

Destination-Sequenced Distance-Vector (DSDV) is a proactive routing protocol [38] that implements the distance vector algorithm which uses sequence numbers originated and updated by the destination, to avoid the looping problem caused by stale routing information. Each node knows the state and topology of the entire network by maintaining a routing table which is constantly and periodically updated (not on demand) and advertised to each of the node‘s current neighbors.

There are two main advantages for using DSDV routing protocol: it is suitable for creating Ad Hoc networks with small number of nodes and guarantees for loop free path. The main disadvantages of DSDV routing protocol: DSDV can no longer find a route reliably when there is high mobility [39]. It requires a regular update of its routing tables, which uses up battery power and a small amount of bandwidth even when the network is idle.

2.4.6 Dynamic Source Routing (DSR)

Dynamic Source Routing (DSR): is a source routing protocol that provides loop-free routes and supports unidirectional links. The source node must know the complete route path to the destination and it must be carried in the packet header [40][41].

There are two mechanisms used in DSR: Route discovery and Route maintenance. When a source node try to send a data packet to a destination node for which it does not know its route, a route discovery process is initiated to determine such a route. Route discovery is initiated by flooding the network with route request (RREQ) packets. When the destination node receive RREQ packet, then Route Reply RREP message is routed back to the original source. If a link failure is detected then the source node is notified using a route error (RERR) packet. The source node will delete any broken link from its cache [43].

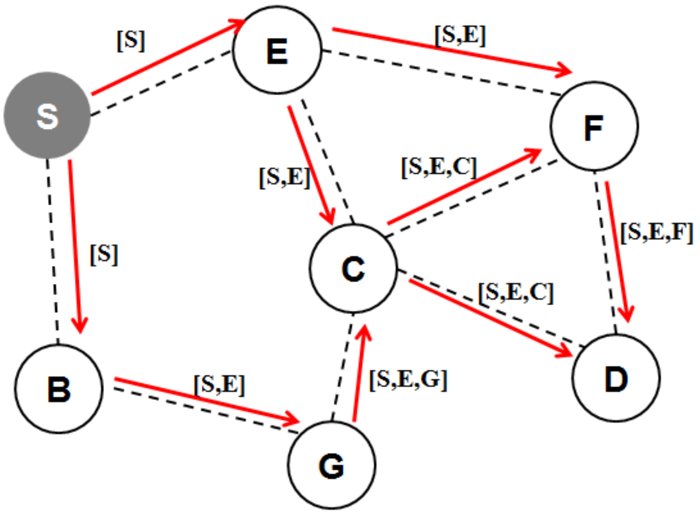


Figure 2-6: Route discovery process in DSR.

There are three main problems arise when using DSR; the first problem is the packet header size that increase as route length increase due to source routing. The second problem is the flooding of RREQ that may reach all nodes in the network.

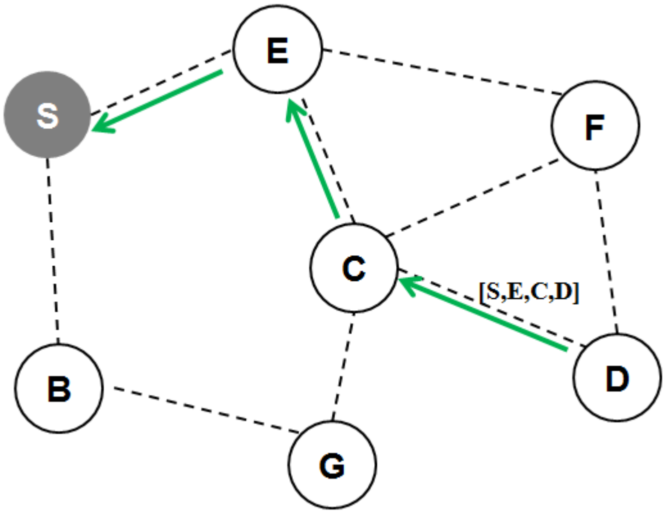


Figure 2-7: Route reply process in DSR.

The third problem is the Route Reply (RREP) storm problem that arises when intermediate nodes produce RREP packet as a response to RREQ packet using their local caches.

2.5. Comparison between MANETs Routing Protocols

According to comparison results in [40][41][42] between DSR, DSDV and AODV routing protocols, the Packet Delivery Ratio (PDR) and throughput of AODV is much better than DSR and DSDV. DSDV is not suitable for MANET when increasing node speed.

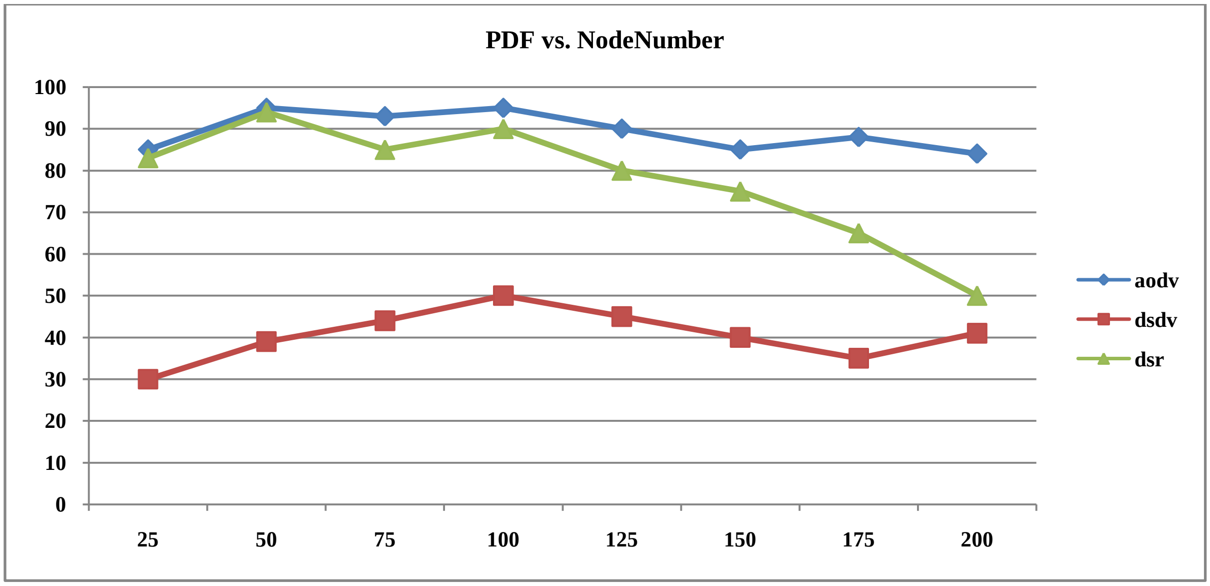


Figure 2-8: Packet Delivery Fraction vs. Number of Nodes (with 10 Connections) [40]

In figure 2-8, the Packet Delivery Fraction versus number of nodes shows that the AODV routing protocol perform much better than DSDV and DSR as number of nodes increase.

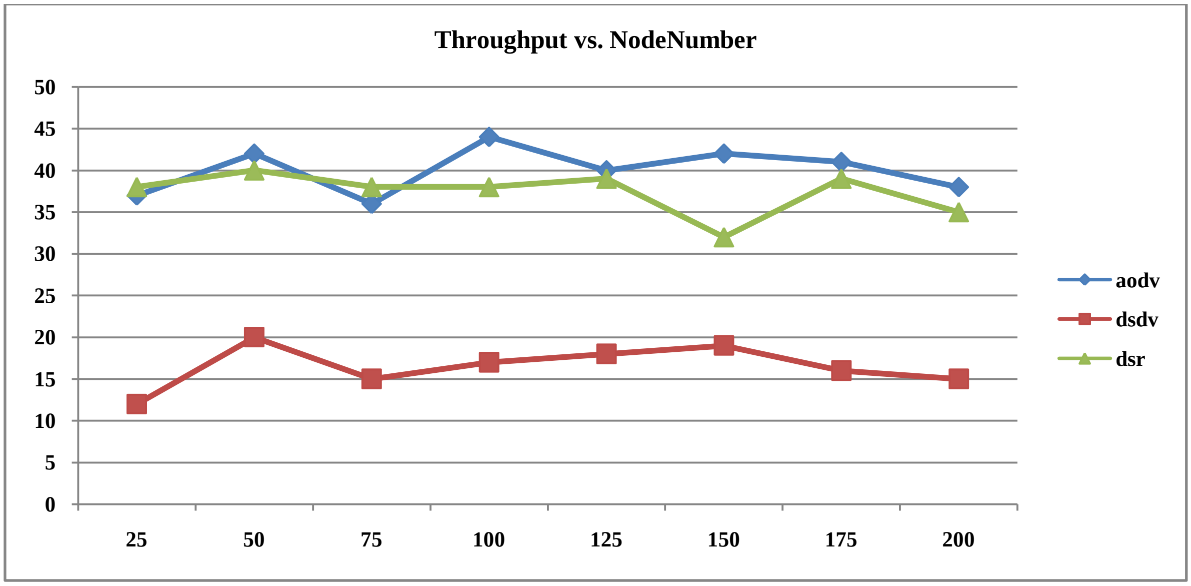


Figure 2-9: Throughput vs. Number of Nodes (with 5 Connections) [40]

According to comparison results in [43][44] between DSR, OLSR, TORA and AODV routing protocols, AODV shows highest throughput followed by OLSR. Thus in most cases AODV routing protocol has the best results according to performance metrics evaluation.

2.6. Summary

Wireless Ad Hoc Networking is an important research area for many applications that need to be developed in a wireless Infrastructure-less based environment. Reactive protocols offer many advantages such as low routing control messages. Besides the AOMDV is an on demand protocol makes use of the advantages from Distance Vector (DV), DYMO is an enhancement version of AODV and is the most standard routing protocol used in MANET today. In this thesis, we will focus on two source initiated (reactive) protocols: AOMDV and DYMO. The problem of initial delay that is the major problem of this category of protocols will be mitigated by the proposed dynamic end-to-end mechanism.

Chapter Three

Literature Review: Congestion Control Techniques

Contents

Literature Review: Congestion Control Techniques

3.1 Introduction

In this chapter, We will present the major techniques used in TCPs to control the end-to-end network congestion; starting with the challenges issues that face the TCP over MANET. The next section is about the three most used congestion techniques: Additive Increase Multiplicative Decrease (AIMD), Active Queue Management (AQM) and slow rate. After that We will discuss the congestion control techniques used in five TCP's: TCP-NewReno, TCP-VEGAS, TCP-WESTWOOD, ATCP and TCP-WELCOME. The last section is the summary.

3.2 TCP Challenges over MANET Environment

There are many TCP challenging issues [45][46][47] that affect its performance over MANET as follows:

1- Frequent link failure due to dynamic topology of the MANET.
2- Bandwidth constraints. The limitation in nodes resources such as buffers of nodes makes a serious challenging problem to TCP performance that leads to network congestion.
3- Power constrains
4- High Bit Error Rate (BER) in wireless channels, hidden terminal and exposed terminal problems.
5- Security issues. MANET environment can be affected to various types of malicious attacks that can degrade its performance, such as Denial of Service (DoS) and SYN-flooding attacks. TCP is not designed to handle these security problems efficiently in a dynamic topology environment.
6- The incorrect differentiation between packet drop due to network congestion and other packet drop causes over MANET such as wireless channel errors and link failure.
7- The inheritance problems of underlying protocols. The type of network layer protocol can affect the overall end-to-end delay, overhead and throughput of TCP. Also the chosen type of Media Access Control (MAC) protocol can affect the fairness throughput of TCP.

3.3 Congestion Control Mechanisms

Congestion can be detected in two major ways: Explicit network signal and implicit network signal. In Explicit network signal, a bit in the packets header is set by the congested router or node in the path between sender and receiver to inform the sender node to decrease the sending rate. This bit is called Explicit Congestion Notification (ECN). In implicit network signal, receiving acknowledgement for new data is determined as no network congestion.

There are several techniques used to handle network congestion such as: Additive Increase Multiplicative Decrease (AIMD), Active Queue Management (AQM) and slow rate.

3.3.1: Additive Increase Multiplicative Decrease (AIMD)

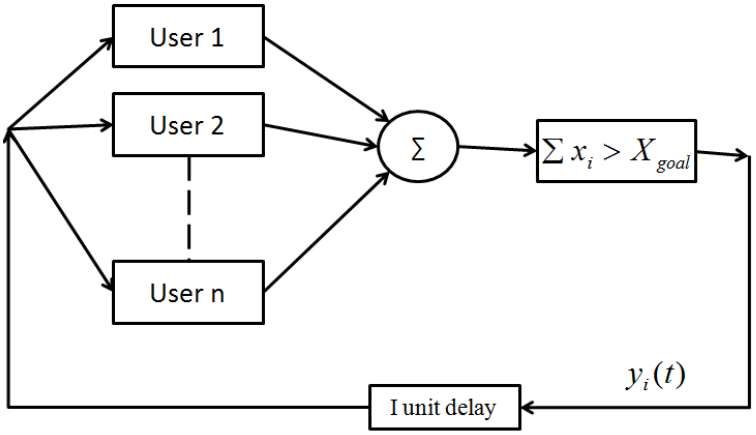


Figure 3-1: Congestion system model [3].

AIMD needs only binary congestion control. In order to explain the technique used in AIMD, suppose we have the feedback control system as in figure 3.1, where X_{goal} is the desired load level at bottleneck resource, and $x_i(t)$ is the load traffic by source i, and $y_i(t)$ binary feedback at time t. The possible control functions are as follows [49].

$$X(t) = \sum x_i(t) \tag{3.1}$$

$$y(t) = \begin{cases} 0 & \text{if } X(\mathrm{t}) \leq X_{goal} \\ 1 & otherwise \end{cases} \tag{3.2}$$

$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{ increase} \\ a_D + b_D x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{ decrease} \end{cases} \tag{3.3}$$

The optimal allocation is: $\sum x_i = X_{goal}$. To operate equation 3.3 in AIMD mode, the coefficients must be as follows: $a_I > 0, \;\; b_i = 1, \;\; a_D = 0, \;\; 0 < b_D < 1$

3.3.2: Active Queue Management (AQM)

TCP-AQM mechanism is implemented at the intermediate nodes to improve the end-to-end congestion control and to provide low delay and low loss in best-effort networks by active congestion notification in advance. TCP-AQM is divided into two types:

1) Random Early Detection (RED)
2) Peripheral Integral Derivative (PID)

1) Random Early Detection (RED).

Not only RED [50][51] controls the queue size to provide congestion avoidance, but also notifies the source before the congestion actually happens.

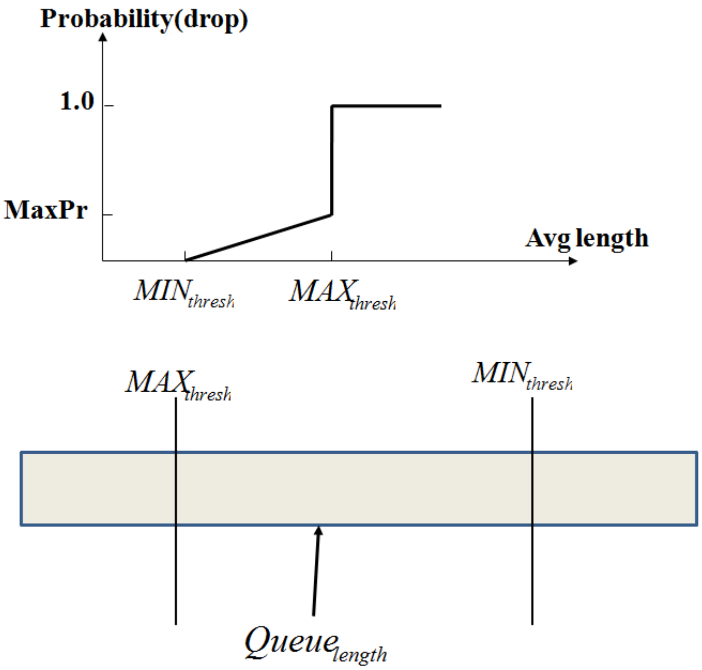


Figure 3-2: Queue management in RED

For Each incoming packet, the Average Queue AVG_Q is calculated as in equation 3.4

$$AVG_Q = (1 - weight) \times AVG_Q + weight \times Queue_{length} \qquad \textbf{(3.4)}$$

The value of AVG_Q is used to calculate the probability of packet drop Pb as in Figure 3.2 by equation 3.5, where MAX_{thresh} is the maximum threshold and MIN_{thresh} is the minimum threshold.

$$Pb = MAX_{pr}(AVG_Q - MIN_{thresh})/(MAX_{thresh} - MIN_{thresh}) \qquad \textbf{(3.5)}$$

]

2) In PID TCP-AQM [52 , the congestion control used for manageing the intermediate nodes is Figure 3.3.

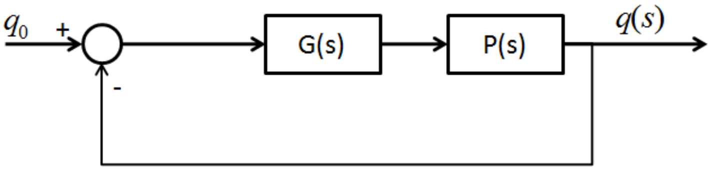


Figure 3-3: The closed-loop system of TCP-AQM linearized model P(s), with the PI controller, G(s). [8]

$$G(s) = K_p + \frac{K_i}{s} + K_d s \tag{3.6}$$

$$P(s) = \frac{\frac{BW^2}{2N} \times e^{-sRTT}}{(s + \frac{2N}{RTT^2 C})(s + \frac{1}{RTT})} \tag{3.7}$$

Where BW is the bandwidth, N is the number of nodes, Kp is the factor for reducing the rise time, Ki is the factor for eliminating the steady-state and Kd is the factor for increasing the stability of the system and improving the transient response.

3.3.3: Slow Rate

In slow rate technique, TCP manages the number of packets sent to the network by increasing and decreasing the congestion window (CWND). The TCP sender starts the session with a congestion window value of one MSS.

The major function of Congestion Window (CWND) is to limit how much data allowed having in transit at a given time. The congestion window is congestion control‘s counterpart to flow control‘s advertised window that is received from the destination. TCP is modified such that the maximum number of bytes of unacknowledged data allowed is now the minimum of the congestion window and the advertised window[54].

$$MaxWindow = \min(CWND, AdvertisedWindow) \tag{3.8}$$

Congestion Window (CWND) has to be calculated by sending side of TCP. Once the ACK is received within the retransmission timeout (RTO) period, the congestion window is doubled and this is called slow start.

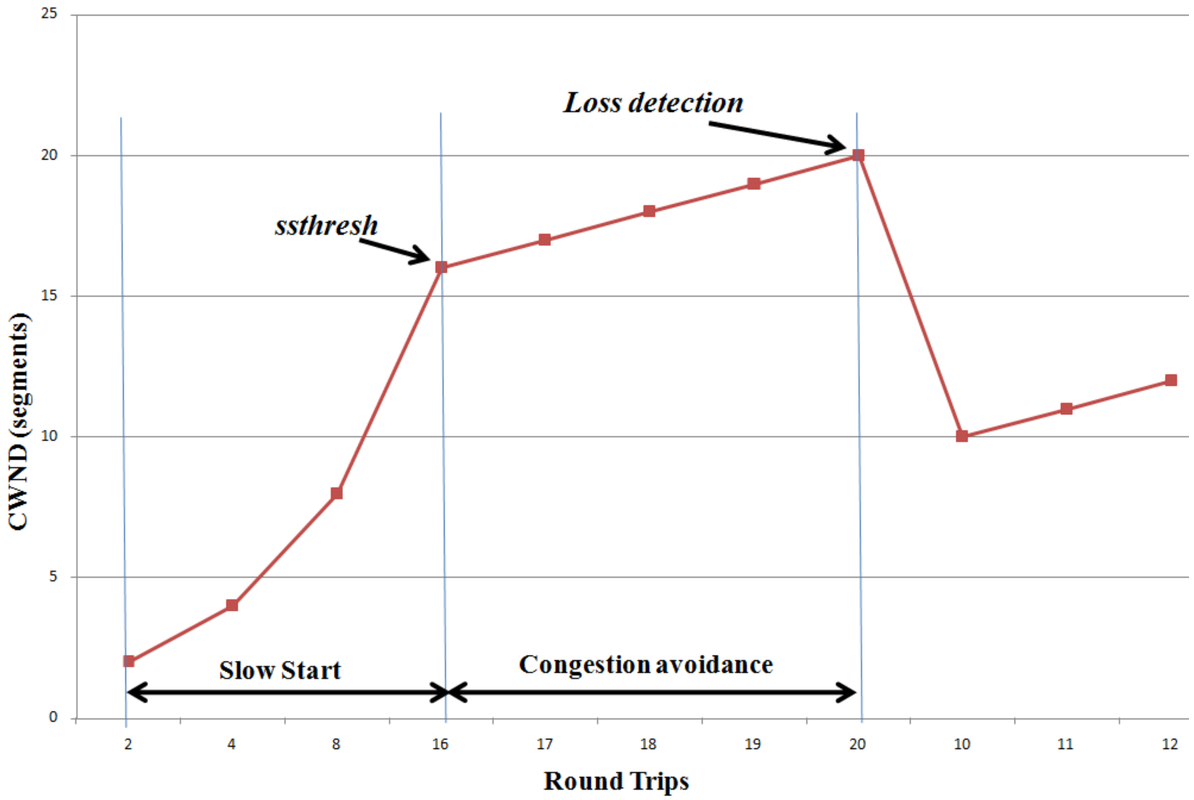


Figure 3-4: Slow start and congestion avoidance.

Once it reaches the slow start threshold it grows linearly by adding one Maximum Segment Size (MSS) [55] to the congestion window every ACK received. This continues until packet loss detected which start congestion avoidance mechanism that reduces the slow start threshold to half the current CWND and reduce the congestion window size to one MSS. Since TCP is widely used, several mechanisms are developed to improve TCP's performance over Mobile Ad Hoc Networks [56].

3.4 Transmission Control Protocols (TCPs)

In this section We will present the mechanisms used for congestion control in most used TCP's: TCP-NewReno, TCP-VEGAS, TCP-WESTWOOD, ATCP and TCP-WELCOME.

3.4.1: TCP-NewReno

TCP New Reno is an effective modification of the original congestion avoidance algorithm in TCP RENO. The modification is an improvement of the Fast Recovery phase. CWND is modified as the following,

$$\mathrm{C}WND_{i+1} = \begin{cases} \mathrm{CWND_i} + 1 & ; \mathrm{CWND_i} < s\mathrm{sthresh_i} \\ \mathrm{CWND_i} + \dfrac{1}{\mathrm{CWND_i}} & ; \mathrm{CWND_i} > s\mathrm{sthresh_i} \end{cases} \tag{3.9}$$

When packet loss is detected, CWND and ssthresh are modified as, [57].

$$ssthresh_{i+1} = \frac{CWND_i}{2} \tag{3.10}$$

$$\mathrm{C}WND_{i+1} = \begin{cases} s\mathrm{sthresh_i} & \text{if three DUPACK received} \\ 1 & \text{if } c\text{oarse timeout expires} \end{cases} \tag{3.11}$$

Disadvantages of TCP-NewReno congestion control algorithm: TCP-NewReno takes one RTT to detect each packet loss. In order to decide which segment lost then Ack of pervious transmitted segment must be received.

3.4.2: TCP-VEGAS

Vegas is an enhancement of RENO. It depends on proactive measures to encounter congestion in a much more efficient than reactive ones. It overcomes the problem of requiring enough duplicate ACKs to detect a packet loss, and it suggests a modified slow start algorithm which prevents it from congesting the network.

The new retransmission mechanism in Vegas extends on the retransmission mechanism of RENO. It keeps track by calculating an estimate of the RTT. TCP Vegas is different from all the other implementation in its behavior during congestion avoidance. It determines congestion by a decrease in sending rate as compared to the expected rate as in Figure 3-5:

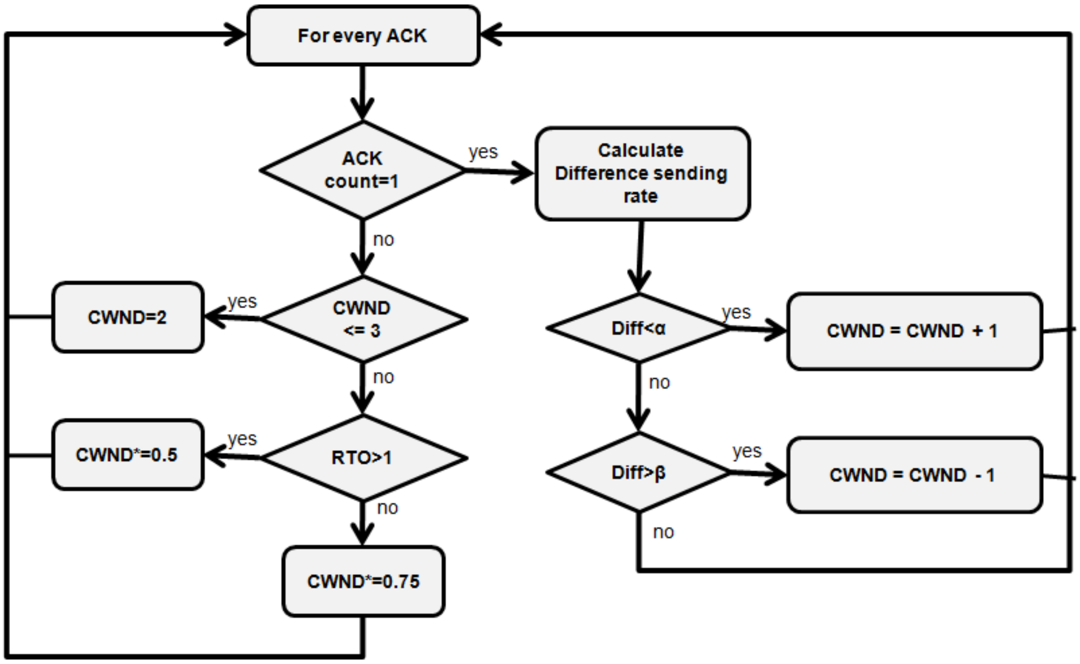


Figure 3-5: The internal operations for congestion control in TCP-VEGAS.

When new Ack is received,

$$\text{Actual sending rate} = \text{CWND}_{\text{last}} \; / \; \text{RTT}_{\text{last CWND}} \qquad \textbf{(3.12)}$$

$$\text{Expected} = \text{CWND}_{\text{current}} / \text{RTT}_{\text{min}} \qquad \textbf{(3.13)}$$

$$\text{Difference} = \text{Expected} - \text{Actual} \tag{3.14}$$

$$\text{cwnd} = \begin{Bmatrix} \text{cwnd} + 1 & \text{diff} < \alpha \\ \text{cwnd} & \alpha \leq \text{diff} < \beta \\ \text{cwnd - 1} & \text{diff} > \beta \end{Bmatrix} \tag{3.15}$$

TCP Vegas increases *cwnd* linearly for the next RTT, if *Diff* <α and decreases *cwnd* linearly, if *Diff* > β. Otherwise, Vegas leaves *cwnd* unchanged [58].

When n duplicate ACKs are received,

$$\text{cwnd} = \begin{Bmatrix} \text{cwnd} \times \frac{3}{4} & ;(c\text{wnd} > 3 \text{ and RTO} < 1) \\ \text{cwnd} \times \frac{1}{2} & ;(c\text{wnd} > 3 \text{ and RTO} > 1) \\ 2 & ;c\text{wnd} <= 3 \\ & \end{Bmatrix} \tag{3.16}$$

Disadvantages of TCP-VEGAS: TCP-VEGAS performance decrease in the case of wrong RTT estimation, since it based on the value of RTT. TCP VEGAS performance also decreases when buffers at routers decrease.

3.4.3: TCP-WESTWOOD

TCP-Westwood uses bandwidth estimation to achieve protocol performance in mixed wired and wireless networks as follows,

When new ACK is received, Congestion Window (CWND) is increased accordingly to the Reno algorithm; the end-to-end bandwidth estimate *BWE* is computed as in [59]

$$\mathrm{BWE} = \frac{\mathrm{Ack\_Size}}{\mathrm{Ack\_Interval}} \tag{3.17}$$

$$ssthresh = \max(2, \frac{\mathrm{BWE} \times \mathrm{RTT}_{\min}}{\mathrm{SegSize}}) \tag{3.18}$$

Value of CWND is updated according to equation (3.11) as in TCP newReno. If packet loss is detected by three duplicated ACK ,then CWND = ssthresh. If packet loss is detected by coarse timeout expires, then CWND =1.

Where: *Ack_Size* =total size of the Ack windows and *Ack_Interval* = Time difference between the last received Ack and current BW estimation.

The main drawback of TCP-WESTWOOD is its inability to differentiate between packet losses cause. This will degrade its performance since link failure is a major reason for packet loss. Also this protocol performs poorly if it estimates incorrect Bandwidth.

3.4.4: Ad Hoc TCP (ATCP)

Ad Hoc TCP (ATCP) is an end-to-end approach that uses network layer feedback to monitor the status of network path [60]. ATCP uses Explicit Congestion Notification (ECN) and implement a thin layer between traditional TCP layer and IP layer in order to minimize the required changes to the TCP layer. Sender node has four states as in Figure 3-6.

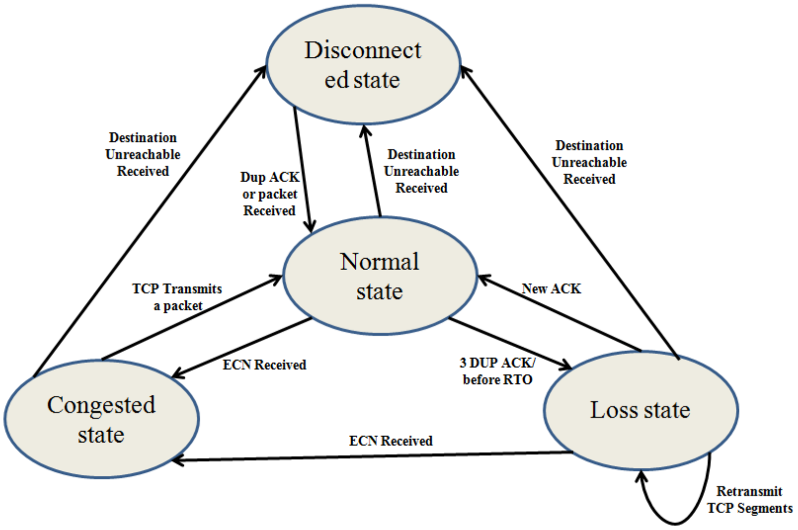


Figure 3-6: State diagram for ATCP [60].

When ATCP enter congested state by ECN, CWND is initially halved and stay in the same value until congested state change.

3.4.5: TCP-WELCOME

TCP Wireless Environment, Link losses, and Congestion packet loss ModEls (TCP-WELCOME) [61] is designed to work over MANET environment to increase the throughput and decrease energy consumption in more efficient way based on measured Round Trip Time (RTT), RTO and three duplicated ACK. Traditional TCP variants are not design to handle dynamic topology with large number of link failure. TCP-WELCOME overcomes this problem by performing two operations: Loss Differentiation Algorithm (LDA) and Loss Recovery Algorithm (LRA). LDA is performed first to detect packet loss reason during data transmission as in Figure 3-8. The causes of packet losses in TCP-WELCOME are divided into three types: network congestion, link failure or wireless channel errors. If the RTT value increases gradually (due to the gradual increase in processing time at congested node’s buffer), then packet loss is identified as network congestion. If RTT remains relatively constant, then the expiration of Retransmission Time Out (RTO) or three duplicate ACK will decide whether the cause of packet loss is due to wireless channel error or due to link failure.

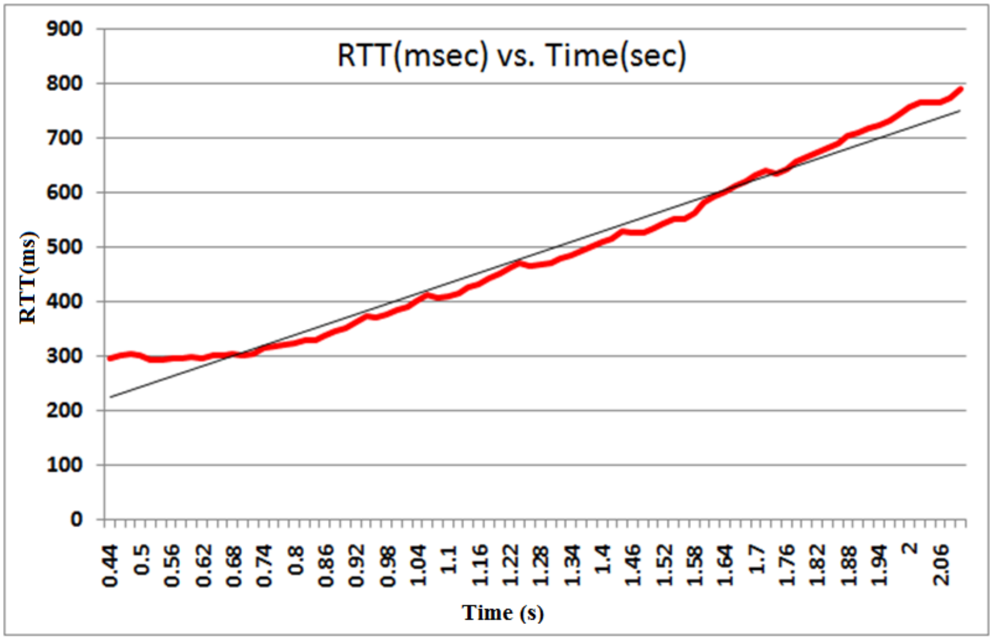


Figure 3-7: RTT evolution due to network congestion in MANET

RTT is measured as the summation of queuing time q(t), processing time p(t) and propagation P(t) time as in (3.19):

$$RTT(t) = 2\sum_{i=1}^{n}\left[q_{d,i}(t) + P_{d,i}(t) + p_{d,l}(t)\right] \qquad \textbf{(3.19)}$$

LRA is performed after LDA to trigger the related recovery algorithm based on packet loss reason. TCP-WELCOME has three loss recovery algorithms which are: network congestion recovery, link failure recovery and wireless channel errors recovery.

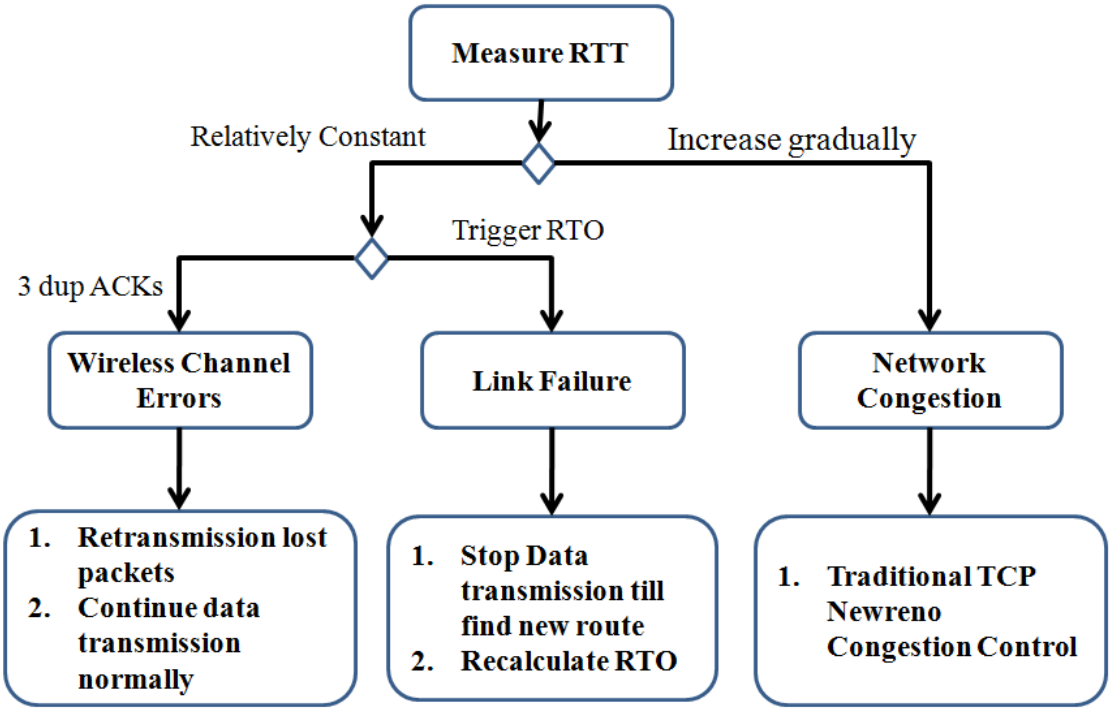


Figure 3-8: TCP-WELCOME LDA and LRA based on RTT, RTO and 3 duplicated ACK

a. Network congestion related packet recovery algorithm. TCP-WELCOME use congestion control algorithm of TCP New Reno for recovery of packet.

b. Link failure related packet loss recovery algorithm. TCP WELCOME adjusts both values of RTO and CWND based on the ratio of RTT new and RTT old values as the following:

$$RTO_{new} = \frac{RTT_{new}}{RTT_{old}} \times RTO_{old} \tag{3.20}$$

$$CWND_{new} = \frac{RTT_{old}}{RTT_{new}} \times CWND_{old} \tag{3.21}$$

c. Wireless related packet loss recovery algorithm. TCP WELCOME does not make any changes, just retransmit the lost packets.

The problem of TCP-WELCOME that degrades its performance is its implementation of TCP New Reno congestion technique for recovery from packet loss that identified as congestion loss.

3.5 Summary

In this chapter, we present the main techniques use for TCP congestion control. After that we will explain the mechanisms used in TCPs. The first three protocols: TCP-NewReno, TCP-VEGAS and TCP-WESTWOOD are not design to work over MANET; however the congestion control mechanisms used in them are robust and strong. The last two protocols: ATCP and TCP-WELCOME are designed to work over MANET, but with congestion control techniques that are no suitable for dynamic environment such as MANET. We have focus on the internal operations used in TCP-WELCOME, because it’s modern and wide spread and more suitable to the Dynamic Congestion Model (DCM) for MANET, which is the subject of this thesis as we will explain in chapter four.

Chapter Four

DCM: Dynamic Congestion Model

Contents

DCM: Dynamic Congestion Model

4.1. Introduction

TCP-WELCOME success to identify causes of packet losses as a key solution to the problems of TCP over MANET, However it has a weakness of applying conventional mechanism used by TCP-NewRENO of congestion control. To solve this problem, we now present a new Dynamic end-to-end Congestion detection protocol for MANET (TCP-DCM), a new solution that uses the results of route request process from routing protocol to early detect the end-to-end congestion, and dynamically select the path with minimum congestion from source to destination.

4.2. Proposed Protocol

In this section, we explore the gains that can be achieved by means of a cross-layer approach, where network layer information is passed to the higher layers.

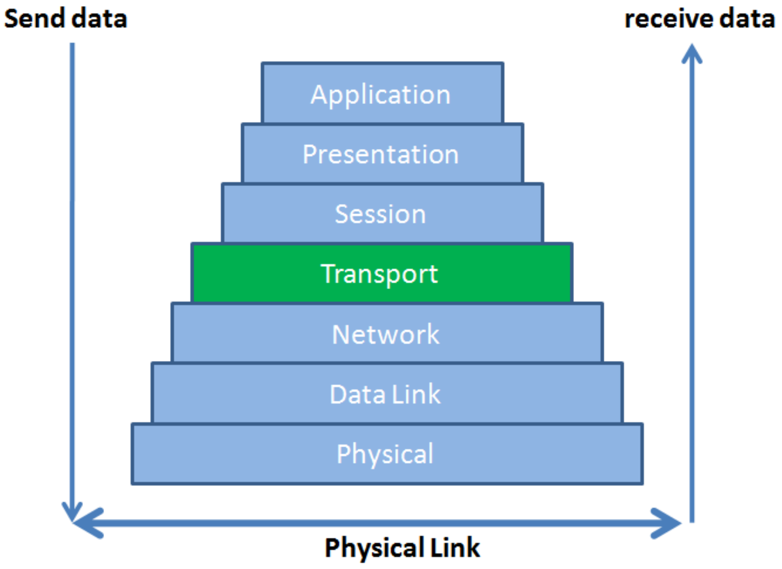


Figure 4-1: Open Systems Interconnection model (OSI Model)

Running TCP efficiently over multipath routing is not as straightforward as with UDP. One problem is that average round trip time (RTT) estimation is not accurate under multipath routing. Namely, the average RTT over several paths may be much shorter than the max RTT (on the longest path). Thus, TCP sender may prematurely timeout packets which happen to take the longest path.

Moreover, packets going through different paths may arrive at the destination out of order and trigger duplicate ACKs, which in turn may trigger unnecessary TCP congestion window reductions. Indeed, from our experiments, we found that using multiple paths simultaneously will actually degrade TCP performance in most simulation scenarions. Thus, it is important to investigate TCP performance over multipath routing to understand if and when there are gains.

Our modification has two phases: one on the destination node before generating Route Reply message (RREP) and the other at sender side during the connection.

4.2.1. DCM: Network layer

As mentioned in section 3 of chapter two, during route request process, sender initiate a Route Request message to one hop nodes surrounding it. The process remains until this message receives the destination. After that destination select the shortest path and generate a Route Reply message to sender. Other valid paths from source to destination will be discarded.

The network layer modifications are as follows:

A) Enhancing AOMDV routing protocol:

The AOMDV routing protocol provides multipath routes from source to destination as in to Table 2-1. The switch between paths can be performed by using route id. Initially the source node chooses the route with the lowest route timeout value.

To switch between route paths, we have modified the function: *handle_link_failure* in /ns-allinone-2.34/ns-2.34/aomdv/aomdv.cc (see appendix C)

After that, source will measure Round Trip time (RTT) under the control of TCP and will select the minimum RTT path as main path from source to destination.

B) Enhancing DYMO routing protocol:

Our enhancement to DYMO is as follows: during the route discovery phase, every intermediate node has to save the path to the request source node in order to send the corresponding reply message to it. When Destination node receives a route request, it sends the reply back through the neighbor node from which it received the packet; the last hop value is the same one contained in the request packet. The first path used by each intermediate node with this last hop value is the valid path, determining its next hop; the node removes the other paths with the same next hop, although with a different last hop.

After the route discovery process, every node will have one or more routes to every possible destination. They must therefore decide how to select them. The maximum number of different routes to be used at a time is 3 routes. For each data packet, the node always chooses the route with the lowest route timeout value and twice the route timeout of the selected one.

We have modified the function: *route_valid_timeout* in dymo_timeout and the c++ files: (dymo_generic.c and main .c) in /ns-allinone-2.34/ns-2.34/dymoum (see appendix c)

4.2.2. DCM: Transport layer

During communication, if TCP detects gradual increase in RTT greater than congestion threshold level (CONGESTION_THREHOLD) as in equation (4.1), then the function **handle_link_failure (***IP_Destination***)** will be invoked. The value of $\alpha = 1.8$ in equation 4.1 based on our measurements from RTT versus time. By this invokation, TCP will notify the network layer to check the validity of other paths available at source. If source node found another path with lower RTT than the current congested path, that path will be selected as major path for data communication.

$$\text{CONGESTION\_THREHOLD} = RTT_{\min} * 1.8 \times \alpha \tag{4.1}$$

If source node succeedes to find another valid path with lower value of RTT, then it has to calculate the new values of CWND, RTO and ssthresh accurately. Selecting a high value of CWND will lead to congestion and cause more packet losses, on the other hand selecting a low value of CWND will decrease the throughput of the network. Accouring to the previous studies, the estimated value of new ssthresh will be calculated depending on equations (3.17) and (3.18) that is used by TCP-Westwood. Estimated value of new RTO is calculated by equation (3.20) that used by TCP-Welcome when selecting a new path. Our equations for estimating the value of CWND when selecting another path is as follows,

$$CWND_{new} = \left(\frac{RTT_{old}}{RTT_{new}} \times ssthresh_{new} \right) \times \beta \tag{4.2}$$

The value of factor β = 0.8 based on our measuremnts to provide the best throughput. Low values of β will decrease the throughput. High values of β will be better if the new estimated value of ssthresh is much higher than value of the previous path.

The following c++ code is an implementation for equation (3.20) and equation (4.2) in /ns-allinone-2.34/ns-2.34/tcp/tcp.cc:

```
if((sendTime !=0.) && (transmits==1)) {
                         // update fine-grained timeout value, and CWND for new paths.
                        double rtt, n;
                        rtt = currentTime - sendTime;
                        RTT_SUM += rtt;
                        ++RTT_COUNT;
                        if(rtt>0) {
         RTO_Based = (rtt / RTT_)*RTO_Based;
         cwnd_ = ((rtt / RTT_)*ssthresh_)*0.8;
                          RTT_ = rtt;
                        }
```

```
                }
```

Otherwise if neither existing paths valid nor have lower value of RTT than current one, TCP will update congestion window (CWND) as:

$$CWND_{i+1} = \left\{ \begin{array}{ll} \mathrm{CWND_i} + \left(\dfrac{\mathrm{RTT_i}}{\mathrm{RTT_{i+1}}} \times 0.9 \right) & ;\mathrm{CWND_i} < ssthresh \\ \mathrm{CWND_i} + \dfrac{\mathrm{RTT_{i+1}}}{\mathrm{CWND_i} \times RTT_i} & ;\mathrm{CWND_i} \geq ssthresh \end{array} \right\} \quad \textbf{(4.3)}$$

The following c++ code is an implementation for equation (4.3) in /ns-allinone-2.34/ns-2.34/tcp/tcp.cc:

```
        if(cwnd_ < ssthresh_) { // slow-start
            CWND_INCREASE_OK = !CWND_INCREASE_OK;
                                        if(!CWND_INCREASE_OK)
                                                Amount_CWND_INCREASE = 0;
                                        else
                                                Amount_CWND_INCREASE = (RTT_/rtt)*0.9;

                                } else { // congestion avoidance
                                                Amount_CWND_INCREASE = (rtt/(RTT_*cwnd_));
                                }
```

In order to avoid fast retransmission requests generated by the receiver node, sender will generate packet carries the sequence number of the segment at the head of the queue buffered at the congested hop and the reply packet have the sequence number of the last successful received segment at receiver node.

TCP receiver will have the ability to understand the packets lost in transition and those buffered at the congested hops.

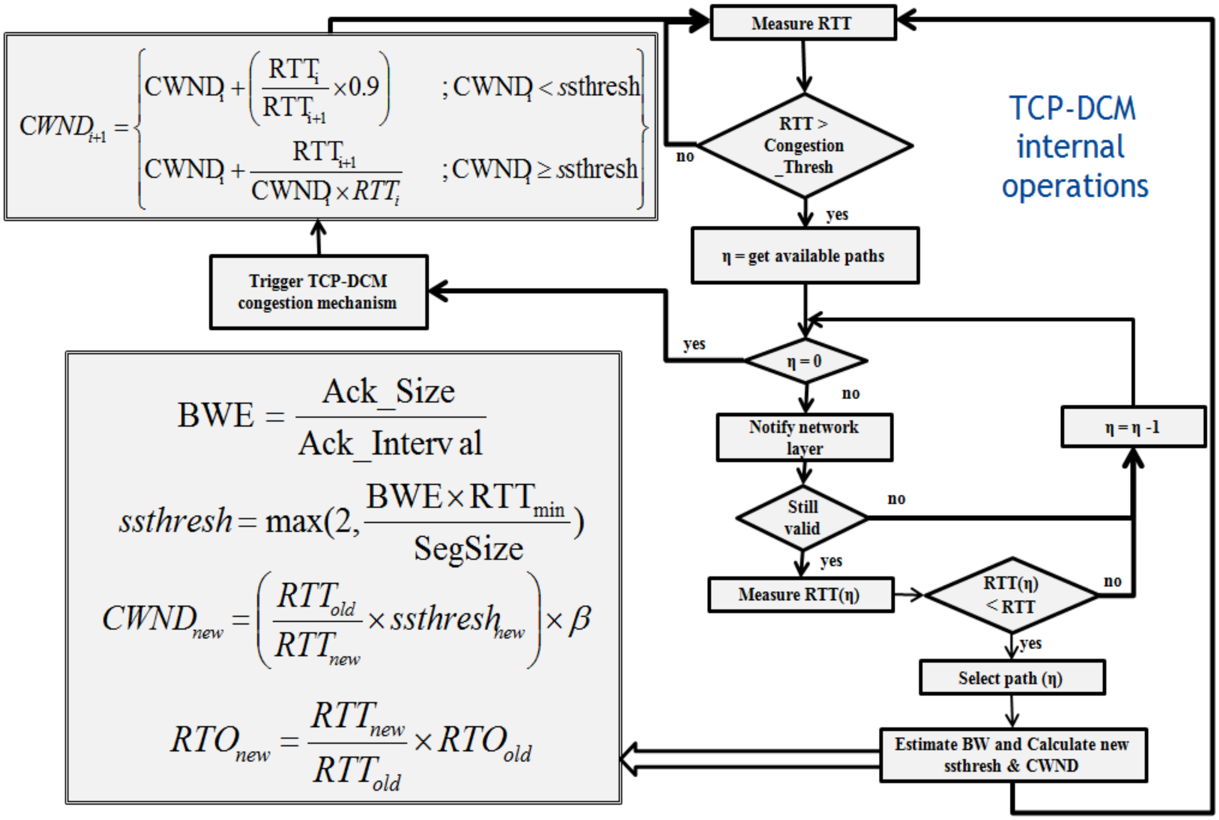


Figure 4-2: TCP-DCM dynamic operations while gradual increase in RTT value.

The entire operations of TCP-PDCM during the gradual increase in RTT value is shown in Figure 4-1.

The following is pseudo code for proposed algorithm at transport layer:

Initialization:

```
//when Route-Reply message receive
```

```
For each path
    // notifying network layer Measure  RTT
End
Select path with minimum RTT
Estimate BW
Calculate new ssthresd and cwnd
Store  second and third paths with each RTT  //if possible
```

Running:

```
          // packet loss detected due to congestion
If currentRTT > RTTthreshold
 For each path
          // notifying network layer
  Measure RTT
    If measuredRTT < currentRTT
        Select path
       Estimate BW
       Calculate new ssthresd and cwnd
        END
    END
END
```

4.3. Proposed model analysis

In order to explain our proposed algorithm, let us first analyze the effect of path congestion on RTT by considering a wireless ad hoc network consist of eight nodes distributed in area of 650x650 m² with node speed = 0 m/s as in Table 4.1.

Table 4-1: Wireless ad hoc networks, node locations in area of 650x650 m²

Node name	Position x (m)	Position y (m)
n1	1	200
n2	200	200
n3	400	200
n4	350	1
n5	50	400
n6	550	200
n7	520	400
n8	100	1

In figure 4-2, a graphical representation for the positions of nodes that are distributed in area of 650x650 m²:

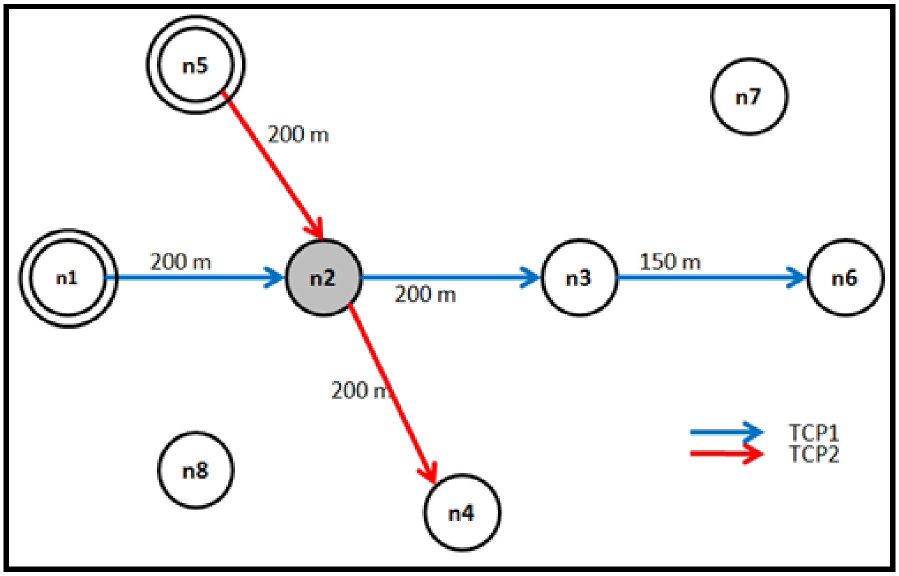


Figure 4-3: Ad hoc networks with two TCP connections: (n1:n6) and (n5:n4).

There are two tcp connections: TCP1:(n1,n2,n3,n6) and TCP2:(n5,n3,n4).

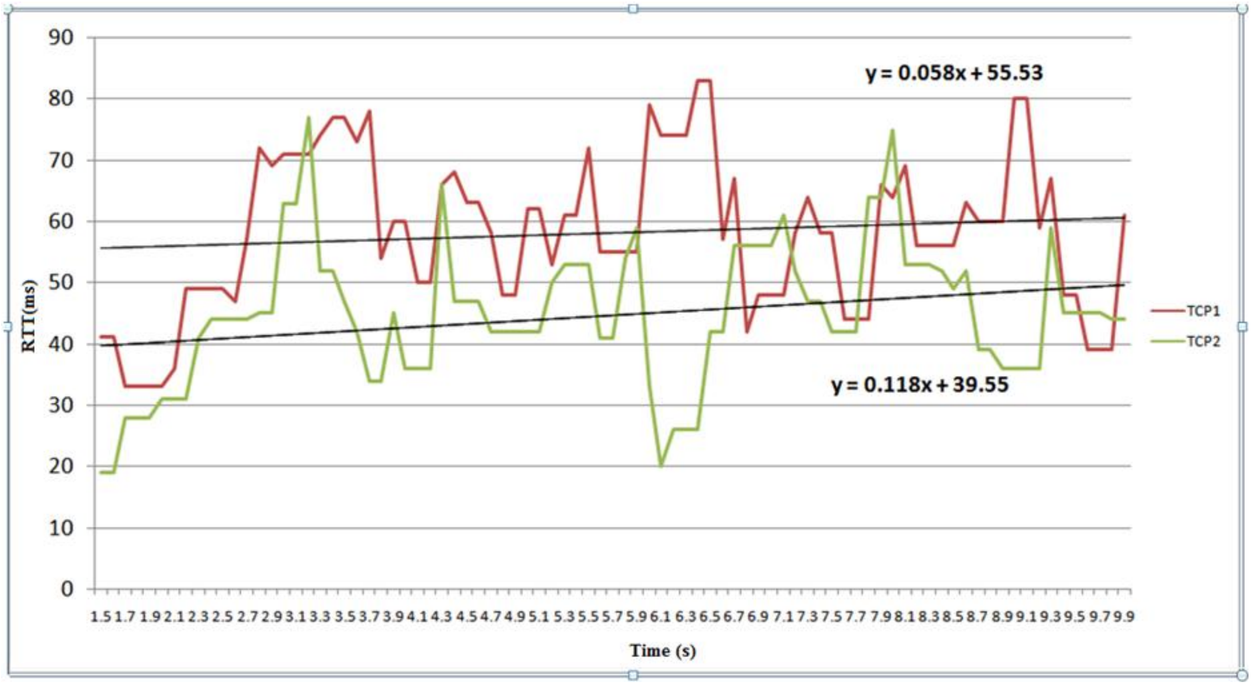


Figure 4-4: RTT vs. Time values for both TCP connections: TCP1 and TCP2 during time period (1.5-9.9) seconds

Before the insertion of the third TCP path, let us consider the value of RTT vs. Time for the first and second TCP paths as in Figure 4.3.

Table 4-2: Values of RTT versus time for TCP1 and TCP2 in a period (1.5-4) seconds

number	Time (s)	TCP1 (ms)	TCP2 (ms)	number	Time (s)	TCP1 (ms)	TCP2 (ms)
1	1.5	41	19	**14**	2.8	72	45
2	1.6	41	19	**15**	2.9	69	45
3	1.7	33	28	**16**	3	71	63
4	1.8	33	28	**17**	3.1	71	63
5	1.9	33	28	**18**	3.2	71	77
6	2	33	31	**19**	3.3	74	52
7	2.1	36	31	**20**	3.4	77	52
8	2.2	49	31	**21**	3.5	77	47
9	2.3	49	41	**22**	3.6	73	42

10	2.4	49	44	**23**	3.7	78	34
11	2.5	49	44	**24**	3.8	54	34
12	2.6	47	44	**25**	3.9	60	45
13	2.7	57	44	**26**	4	60	36
					Average	**56.04**	**41.10**

If we represent the changes in values of RTT vs. time as a function of linear equation in the form of equation 4.4, we can consider that the intercept value β reflect the avarage value of RTT and the slope value α reflect the congestion status of TCP path. The factor α is directly proportional to path congestion.

$$RTT_{TCP} = \alpha \times Time + \beta \qquad \textbf{(4.4)}$$

The intercept of both TCP paths are: 55.53 and 39.55. Note that the first TCP path has two hops n2 and n3 between sender node n1 and receiver node n6, while the second TCP path has only one hop n2 between the sender node n5 and the receiving node n4.

Now let node n7 want to communicate with node n8 as in Figure 4.4.

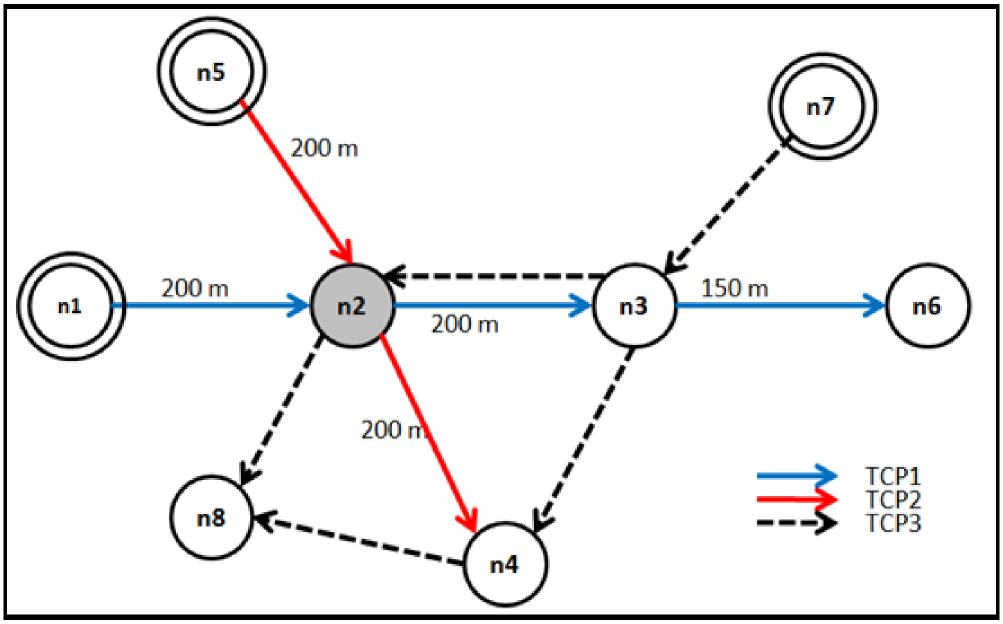


Figure 4-5: Ad hoc network with three TCP connections

The routing protocol will success to find two paths between n7 and n8 as follows: P1:(n7,n3,n4,n8) and P2: (n7,n3,n2,n8). Source node n7 will check both paths before selecting the lowest congested path, by meausring the value of RTT. In this case both paths consist of two hops between source node and destination node. Intiatially path P1will be selected due its lowest value of RTT.

No let us consider the value of RTT versus time for TCP3 in path P1 as in figure 4.5.

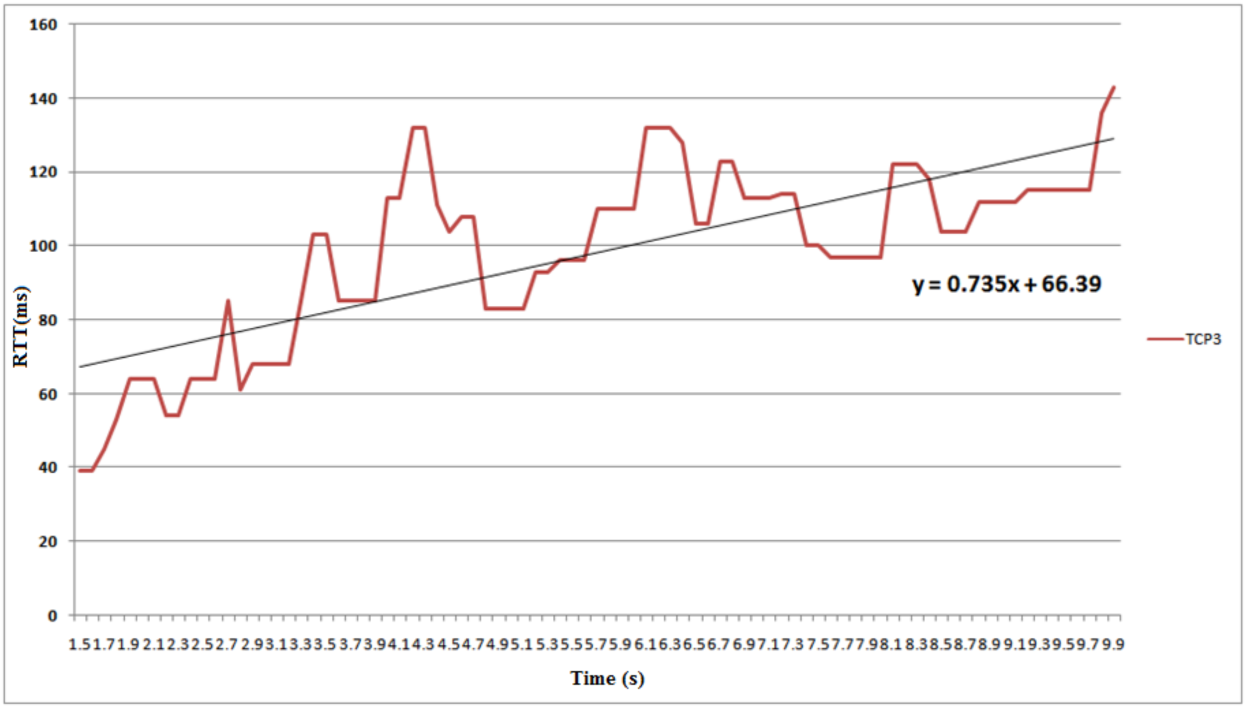


Figure 4-6: RTT vs. Time values for TCP3 connection during time period (1.5-9.9) seconds

Table 4-3: values of RTT versus time for TCP3 in a period (1.5-4) seconds

number	Time (s)	TCP3 (ms)
1	1.5	39
2	1.6	39
3	1.7	45
4	1.8	53
14	2.8	61
15	2.9	68
16	3	68
17	3.1	68

5	1.9	64	**18**	3.2	68
6	2	64	**19**	3.3	85
7	2.1	64	**20**	3.4	103
8	2.2	54	**21**	3.5	103
9	2.3	54	**22**	3.6	85
10	2.4	64	**23**	3.7	85
11	2.5	64	**24**	3.8	85
12	2.6	64	**25**	3.9	85
13	2.7	85	**26**	4	113
				Total Average	**77.21**

The minimum value of RTT in path P1 is 39 ms. Accourding to equation 4.1 with selecting $\alpha = 1.8$, the value of CONGESTION_THREHOLD = 39*1.8 = 70.2 ms. If we consider the avarage value of RTT during time period (1.5-4) in Table 4.3, we can find that it exceeds the value of CONGESTION_THREHOLD, so the algorithm used in DCM will start checking the other paths congestion status. If the other path's RTT value is lower than current then it will be selected, otherwise CWND will be modified accourding to equation 4.3 to control the congestion.

4.4. Summary

In this chapter, we have proposed in section 4.2 the algorithm of Dynamic Congestion Model and how source node can determine the lowest congested path based on value of RTT. In section 4.3, Analysis for the internal operations of DCM on a network consists of eight nodes and three TCP connections. Analysis shows how DCM calculates the average value of RTT and the value of CONGESTION_THREHOLD. In the next chapter, we will show by simulation results how TCP-DCM improves the performance of TCP in congested mobile ad hoc network.

Chapter Five

Simulation Results and Analysis

Contents

Simulation Results and Analysis

5.1 Introduction

Simulation can carry out experiments without the actual hardware and provides a good compromise between complexity and accuracy as in [62]. In this chapter we will present the experimental results of the proposed dynamic congestion model and comparison with other network congestion techniques used in TCP. This chapter is divided as follows: section 2 is about the simulation tool we used to evaluate our DCM. Section 3 is about the environmental assumptions and scenarios. In section 4 we will discuss the major performance evaluation metrics used and what each metric reflects. Section 5 is the simulation results and analysis for the DCM with compare with TCP-WELCOME. In section 6 a comparison with other TCP to evaluate its performance over congested network. Section 7 is the conclusion.

5.2 Simulation tool

The proposed algorithm has been implemented and evaluated over Network Simulator (NS) version 2 which is a discrete event simulator [63]. NS2 is written in C++, which is Object Oriented Language (OOL). NS2 support simulation of different modern TCP types and different routing protocols over wired and wireless networks. Figure 5.1 presents the overall steps of simulation model.

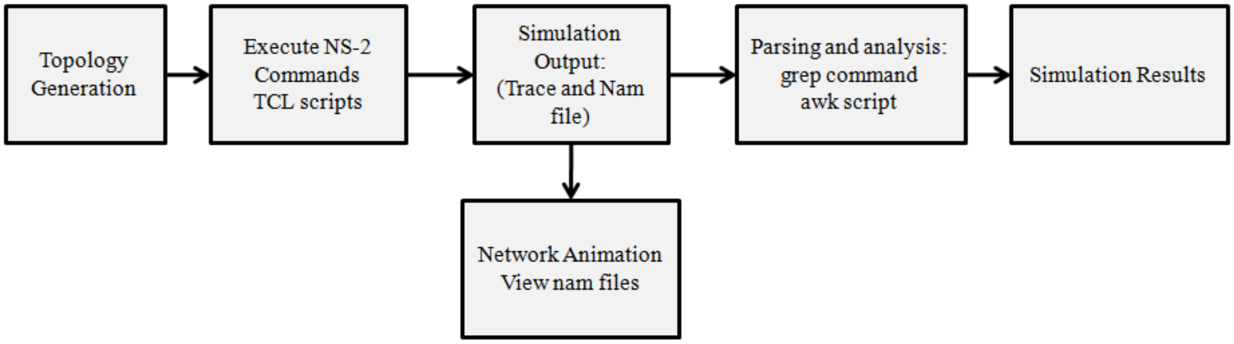


Figure 5-1: Overview of simulation steps.

The results of data processing and analysis are performed by using AWK scripting language as in [64].

5.3 Assumptions and scenario

Environment of implementation in this thesis is done on area of 1000*1000 m², nodes concentration dense: (20, 40, 60, and 80) distributed randomly. We also generate 20 TCP connections between random senders and receivers. In order to ensure the rationality, The simulation results are calculated from the average of three different random scenarios, each has 30 random direction patterns of movement and random velocities. The simulation parameters are listed in Table 5.1.

Node Characteristics:

1) Queue type: Drop-Tail or RED
2) Network Interface type: wireless
3) Channel type: wireless
4) Link Layer Type: Logical Link (LL) type
5) MAC type: 802_11
6) Initial Energy = 100 Watt
    a. Receive Power = $35.28 \times 10^{-3} Watt$
    b. Transmit Power = $31.32 \times 10^{-3} Watt$
    c. Idle Power = $712 \times 10^{-6} Watt$
    d. Sleep Power = $144 \times 10^{-9} Watt$

Table 5-1: Simulation parameters.

MANET Parameter	**Value**
Area	1000x1000 m²
Simulation time	150s

Speeds	(0-3),(3-6),(6-9),(9-12),(12-15) m/s
Routing protocol	AOMDV and DYMO
Mobility	Random
Maximum TCP Connections	20
Number of nodes	20,40,60,80
Packet size (Bytes)	3000,4000,5000,6000,7000
Data Rate	1 Mbps
Traffic Type	File Transfer Protocol (FTP) and Constant Bit Rate (CBR)
MAC Type	Mac/802_11
Maximum packet in queue	30
Antenna	Antenna/Omni Antenna
Link Layer Type	LL
Interface queue Type	D_T and RED
Network interface type	Phy/WirelessPhy

5.4 Performance evaluation metrics

In order to evaluate algorithm effects on TCP performance, the following metrics must be considered:

1) Packet Delivery Ratio (PDR): It is the ratio of the total number of data packets received successfully to the total number of data packets transmitted. This metric is used to measure the efficiency and reliability of the protocol. PDR is expressed as (N is the number of packets) [65]:

$$\mathrm{PDR} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} \text{Data Packets Received}}{\sum_{\mathrm{i}=1}^{\mathrm{N}} \text{Data Packets Transmitted}} \tag{$\mathbf{5.1}$}$$

2) Throughputs: It is the number of packets received successfully with respect to time in unit of Bit/Second or bps. This metric is used to measure the effectiveness of the protocol and the actual speed or data rate in channel [66].

$$\text{Throughput}\left(\frac{\text{bits}}{\text{s}}\right) = \frac{\text{Delivered Packets} \times \text{Packet Size} \times 8}{\text{Total Simulation Period}} \qquad (\mathbf{5.2})$$

3) Average End-to-End Delay ($E2E_D$): The end-to-end-delay is averaged overall surviving data packets from the sources to the destinations which reflect the delay in the interface queues of the intermediate nodes between the sender and the receiver. E2E is expressed as (N is the number of packets):

$$\text{E2E}_\text{D} = \frac{\sum_{i=1}^{N} \text{ReceivedTime}_i - \text{SentTime}_i}{\sum_{i=1}^{N} \text{Data Packets Received}} \qquad (\mathbf{5.3})$$

4) The Normalized Overhead: which reflect how much the cost will be added by the protocol over available bandwidth. It is important to keep the overhead as small as possible. Overhead is expressed as follows:

$$\text{Normalized Overhead} = 1 - \frac{\text{Sent Packets}}{\text{Sent Packets} + \text{Control Packets}} \qquad (\mathbf{5.4})$$

5) Overall Energy consumed (E_C): using LR-WPAN (Low- Rate Wireless Personal Area Network) [67].

a- The Energy send:

$$\text{E(send)} = \text{m}_\text{send} \times \text{PacketSize} + \text{b}_\text{send} \qquad (\mathbf{5.5})$$

b- The Energy received:

$$\text{E(received)} = \text{m}_\text{received} \times \text{PacketSize} + \text{b}_\text{received} \qquad (\mathbf{5.6})$$

Where: $m_{send} = Energy\ of\ sending\ one\ byte$; $m_{received} = Energy\ of\ receiving\ one\ byte$ and $b = Energy\ of\ device\ state\ changes\ and\ channel\ acquisition\ overhead$

5.5 Simulation results

In our simulation we change several network variables to determine its effect on network congestion over the main network environment, which is the average of three different random topology scenarios that reflect the real world MANET environment over 1000x1000 m² area. Each one of the three different scenarios is the average of 50 different movement patterns with random directions and speeds to reflect all the possibilities for mobile nodes. The network variables that are tested: P_L, queue type, node mobility and N_D.

5.5.1 Throughput

In this section we will present the throughput results for our proposed dynamic congestion model DCM by changing network factors Node Dense (N_D), Packet Length (P_L), queue type and Node Speed (N_{SD}) over two routing protocols: AOMDV and DYMO. This performance metric is very important because it reflect the effectiveness of the protocol and the actual speed or data rate in channel. We will also compare our DCM with TCP-WELCOME and ATCP.

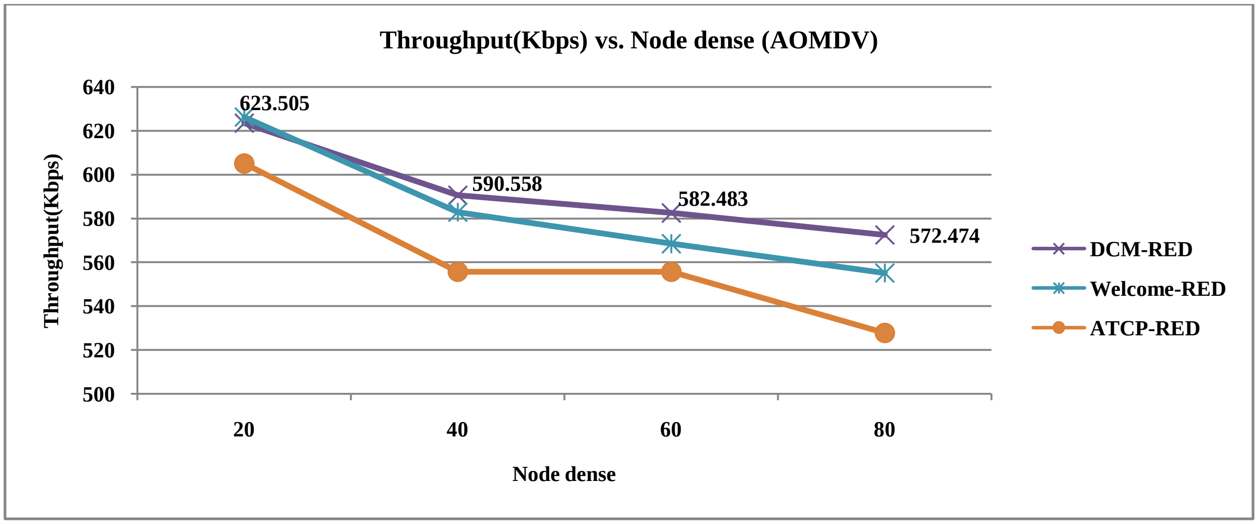


Figure 5-2: Throughput (Kbps) vs. N_D for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol.

In Figure 5-2, the value of throughput decreases as number of node increases over AOMDV routing protocol due to the increase in the overall interference between wireless mobile nodes. ATCP shows the lowest overall throughput, while DCM shows the better throughput than TCP-WELCOME and ATCP. The overall enhancement of DCM over TCP-WELCOME is 1.3% when N_D=40, 2.4% when N_D=60 and 3.1% when N_D=80. Thus DCM overall throughput is much better than both TCP-WELCOME and ATCP as N_D increase

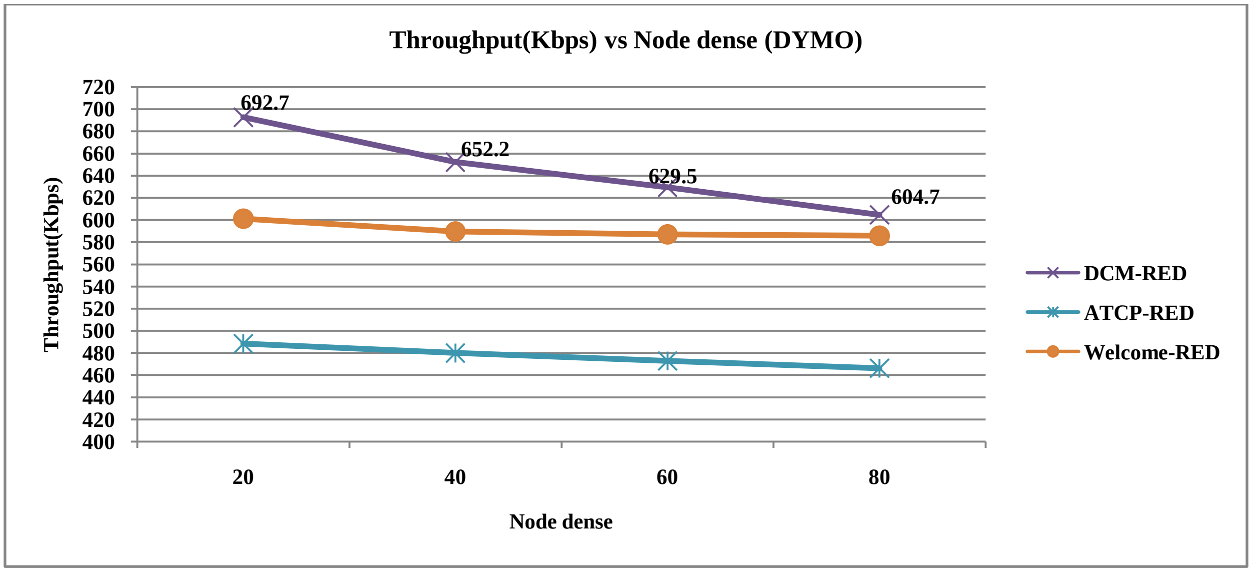


Figure 5-3: Throughput (Kbps) vs. N_D for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over DYMO routing protocol.

In Figure 5-3, As N_D increase, DCM overall throughput is much better than TCP-WELCOME and ATCP over DYMO routing protocol due its dynamic technique that decrease the number of packet drop. The best enhancement in overall throughput of DCM over TCP-WELCOME is 15.2% when N_D = 20, then 10.6% when N_D = 40, then 7.2% when N_D = 60 and 3.7% when N_D = 80. So the best overall throughput enhancement is when N_D = 20 nodes.

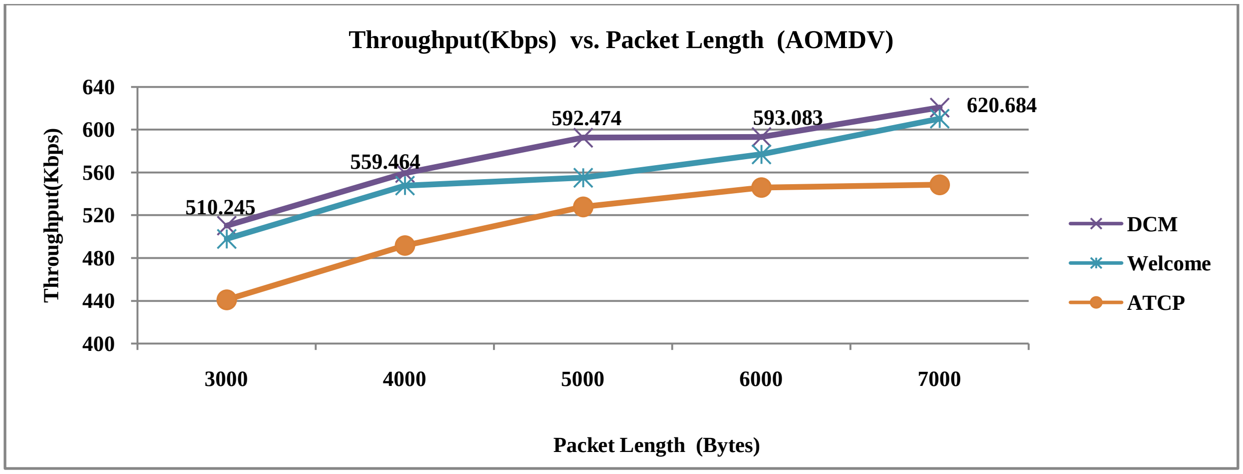


Figure 5-4: Throughput (Kbps) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over AOMDV routing protocol.

In Figure 5-4, as P_L increase the overall throughput must increase, but the problem of congestion increase. DCM overall throughput is better than TCP-WELCOME and ATCP over AOMDV routing protocol. The dynamic technique used in DCM decrease the effect of network congestion that leads to packets drop by dynamically selecting the lowest congested path. The enhancement in overall throughput of DCM over TCP-WELCOME is as follows: 2.5% when P_L = 3000 bytes, 2.1% when P_L = 4000 bytes, 6.7% when P_L = 5000 bytes, 2.8% when P_L = 6000 bytes, 1.7% when P_L = 7000 bytes. So the best throughput enhancement is when P_L = 5000 bytes.

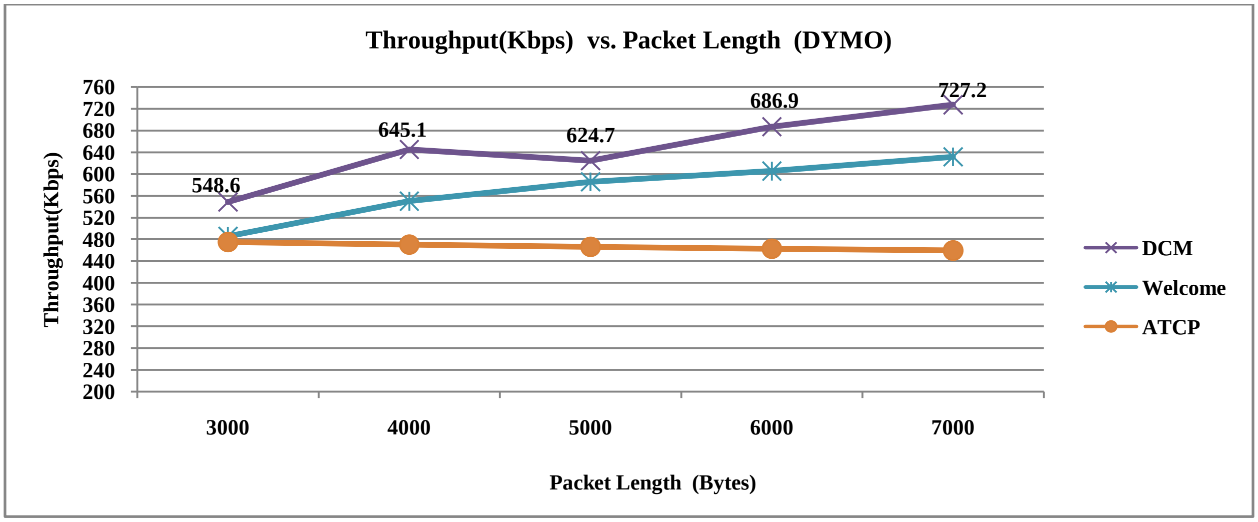


Figure 5-5: Throughput (Kbps) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over DYMO routing protocol.

In Figure 5-5, the overall throughput versus P_L when N_D = 80 nodes and N_S = (0-3) m/s. As P_L increase the overall throughput increase over DYMO routing protocol. ATCP shows the lowest overall throughput that almost remain the same as P_L increase, while DCM shows better overall throughput value than TCP-WELCOME and ATCP.

The throughput enhancement as P_L increase of DCM over TCP-WELCOME is as follows: 12.9% when P_L = 3000 bytes, 17.3% when P_L = 4000 bytes, 6.6% when P_L = 5000 bytes, 13.4% when P_L = 6000 bytes and 15.2% when P_L = 6000 bytes. So the best overall enhancement is when P_L = 4000 bytes.

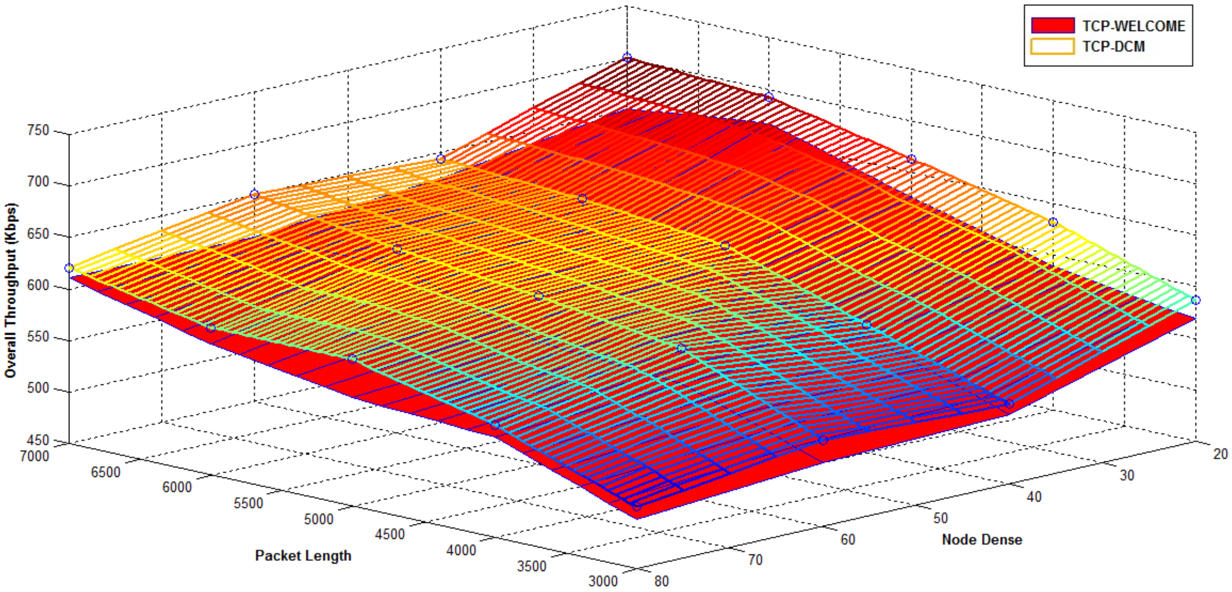


Figure 5-6: Throughput (Kbps) vs. N_D and P_L for TCP-DCM and TCP-WELCOME over AOMDV routing protocol, with queue type RED and N_{SD} = (0-3) m/s.

In figure 5-6, the three dimension graph to represent the effect of varying N_D and P_L on throughput of our proposed DCM and TCP-WELCOME. DCM overall throughput is better than

TCP-WELCOME over AOMDV routing protocol in all scenarios. The lowest enhancement of overall throughput which is 1.9%, is when N_D = 40 nodes and P_L = 3000 bytes.

The best overall enhancement on throughput in all scenarios which is 7.6%, is when P_L = 7000 bytes and N_D = 20 nodes.

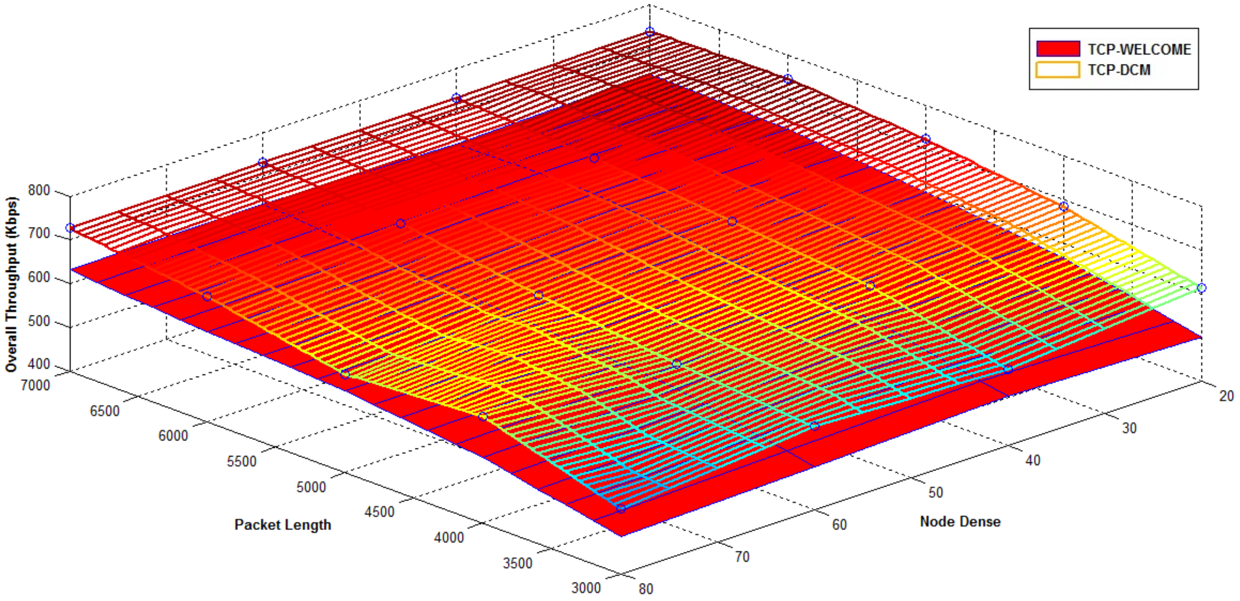


Figure 5-7: Throughput (Kbps) vs. N_D and P_L for TCP-DCM and TCP-WELCOME over DYMO routing protocol, with queue type RED and N_{SD} = (0-3) m/s.

In figure 5-7, the three dimension graph to represent the effect of varying N_D and P_L on overall throughput of our proposed DCM and TCP-WELCOME. DCM enhances the overall throughput over DYMO routing protocol in all the simulation scenarios.

The lowest enhancement of overall throughput which is 6.7%, is when N_D = 80 nodes and P_L = 5000 bytes. The best overall enhancement on throughput in all scenarios which is 20.2%, is when P_L = 5000 bytes and N_D = 20 nodes.

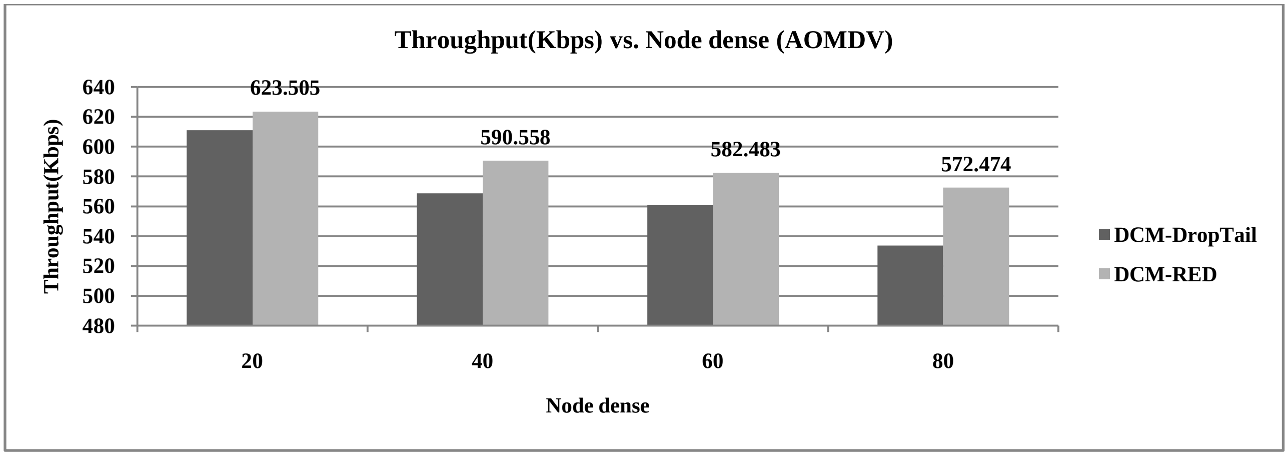


Figure 5-8: Throughput (Kbps) vs. N_D for TCP-DCM over AOMDV routing protocol, queue types: D_T, N_{SD} = (0-3) m/s and RED, P_L = 5000 bytes.

In Figure 5-8, the effect of changing queue management type in overall throughput when N_{SD} = (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol for both DCM and TCP-WELCOME. The Queue management type that is considered in this thesis are RED and Drop-Tail. RED queue management enhances the overall throughput in all N_D scenarios. The overall throughput enhancement of implementing RED over Drop-Tail queue management is when selecting P_L = 5000 bytes is 2.1% when N_D = 20 nodes, 3.8% when N_D = 40 nodes, 3.9% when N_D = 60 nodes and 7.3% when N_D = 80 nodes.

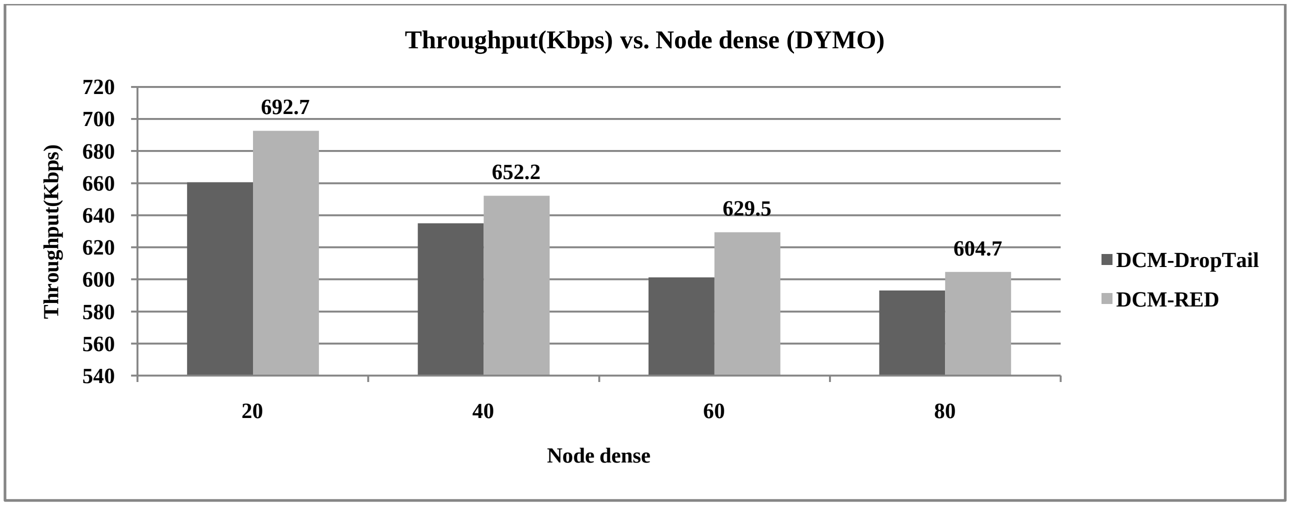


Figure 5-9: Throughput (Kbps) vs. N_D for TCP-DCM over DYMO routing protocol, queue types: D_T, N_{SD} = (0-3) m/s and RED, P_L = 5000 bytes.

In figure 5-9, the effect of implementing RED queue management gives better overall throughput results than D_T type over DYMO routing protocol when selecting P_L = 5000 bytes. The overall throughput enhancement of implementing RED queue management type over D_T is as follows: 4.9% when N_D = 20 nodes, 2.7% when N_D = 40 nodes, 4.7% when N_D = 60 nodes and 1.9% when N_D = 80 nodes.

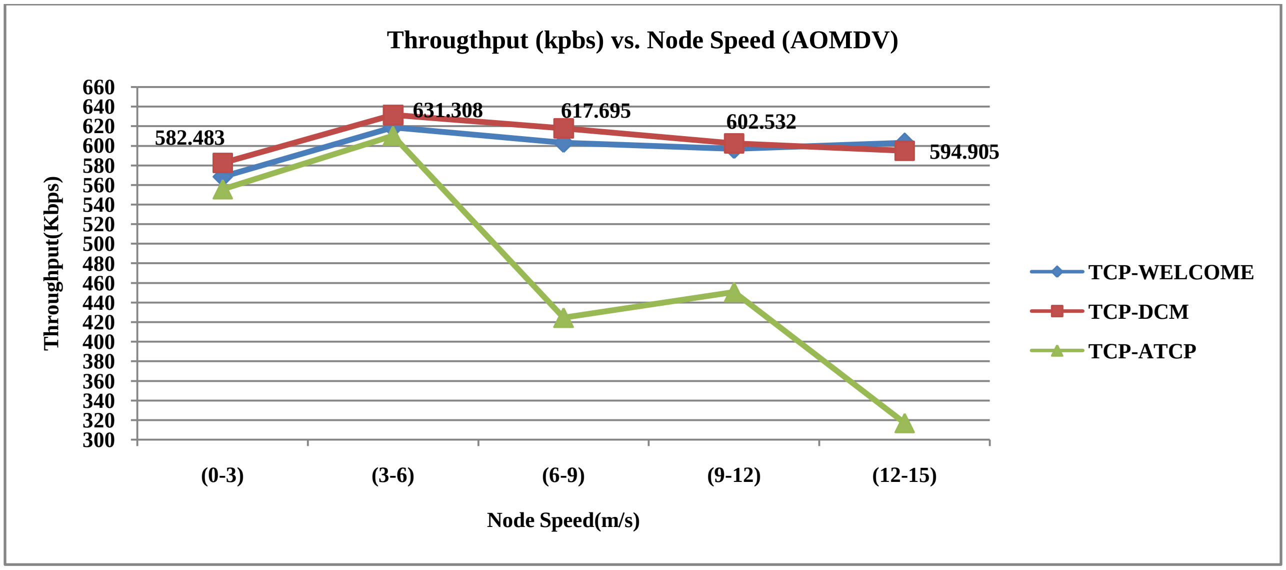


Figure 5-10: Throughput (Kbps) vs. N_{SD} for TCP-DCM over AOMDV routing protocol, queue types: RED, P_L = 5000 bytes and N_D = 60 nodes.

In Figure 5-10, the overall throughput versus N_{SD} when applying P_L = 5000 bytes and N_D = 60 over AOMDV routing protocol. As node speed increase, the overall throughput decreases due to increasing number of link failure.

DCM enhances the overall throughput at low node speeds: (0-10) m/s, and remain stable at higher speeds. ATCP become unstable when node speed increase, thus the throughput of

ATCP decreases dramatically. The best enhancement of overall throughput is 2.4% at N_{SD} = (0-3) m/s, then 2.0% at N_{SD} = (3-6) m/s.

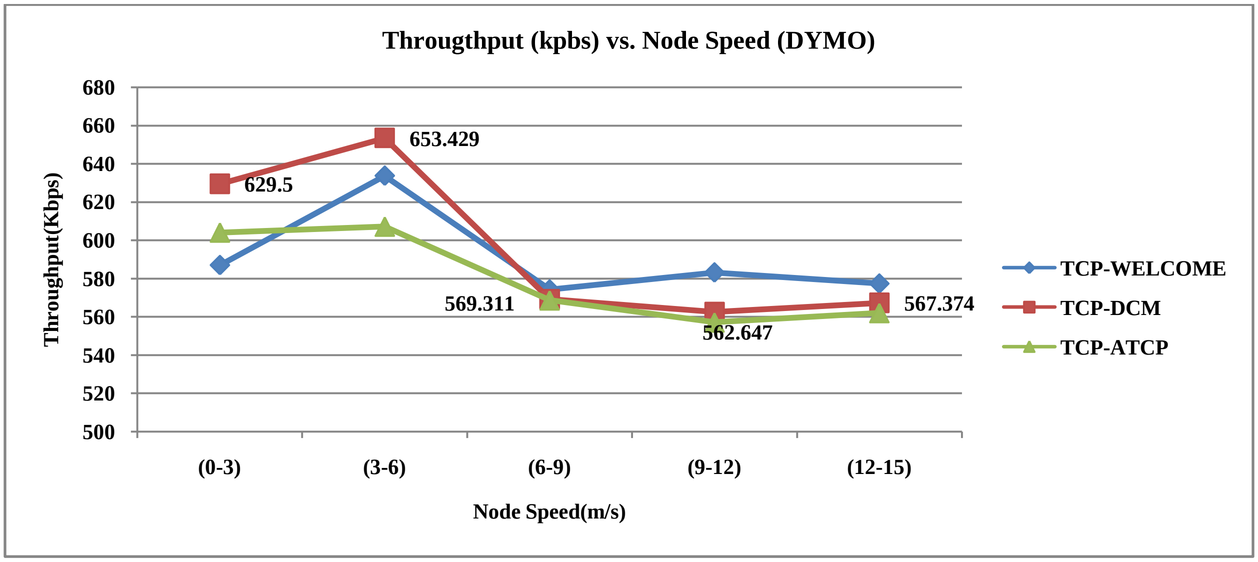


Figure 5-11: Throughput (Kbps) vs. N_S for TCP-DCM over DYMO routing protocol, queue types: RED, P_L = 5000 bytes and N_D = 60 nodes.

In Figure 5-11, the overall throughput is decreasing as N_S increase when P_L = 5000 bytes and N_D = 60. ATCP has the lowest throughput as N_S increase. DCM performs better than TCP-WELCOME in overall throughput at low N_{SD} over DYMO routing protocol. The best enhancement of overall throughput is 7.2% at N_{SD} = (0-3) m/s then the enhancement of overall throughput decrease to 3.1% at N_{SD} = (3-6) m/s. DCM, TCP-WELCOME and ATCP remain stable when increasing node speed over DYMO routing protocol.

5.5.2 Normalized Overhead

In this section we will present the overhead of our proposed dynamic congestion model DCM with changing N_D, P_L, and queue type over two routing protocols: AOMDV and DYMO. This metric is important to the cost will be added by the protocol over available bandwidth.

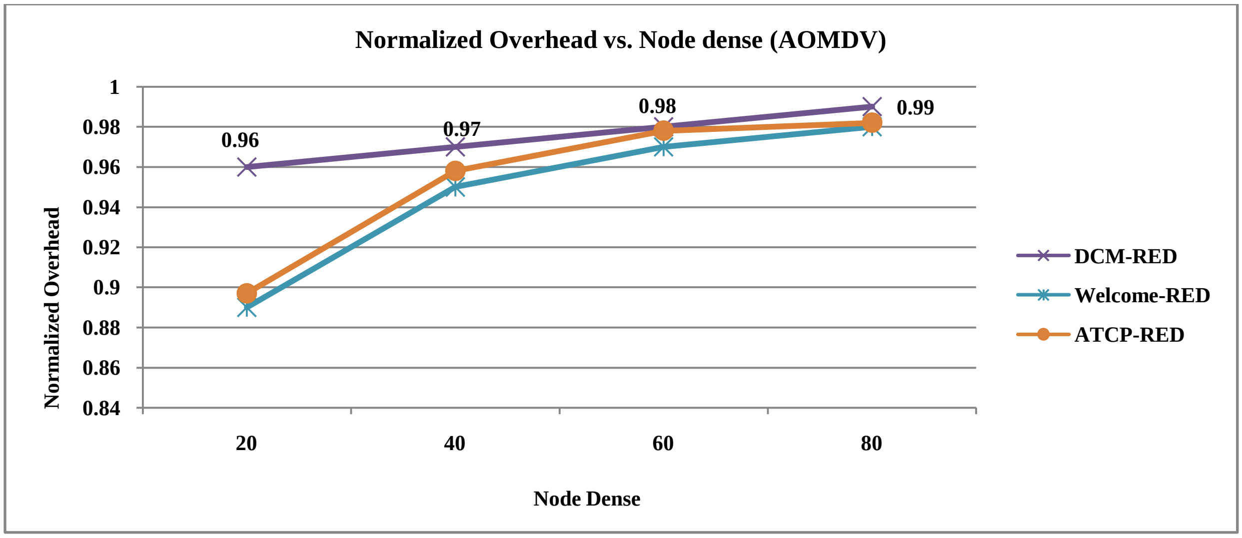


Figure 5-12: Normalized Overhead vs. N_D for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD}= (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol.

In Figure 5-12, the normalized overhead versus node dense over AOMDV routing protocol when N_{SD}= (0-3) m/s and P_L = 5000 bytes. As N_D increase the overhead between nodes increase. DCM has higher overhead then both TCP-WELCOME and ATCP over AOMDV when ND = 20 nodes. The overhead of DCM became almost the same as the overhead ATCP and TCP-WELCOME.

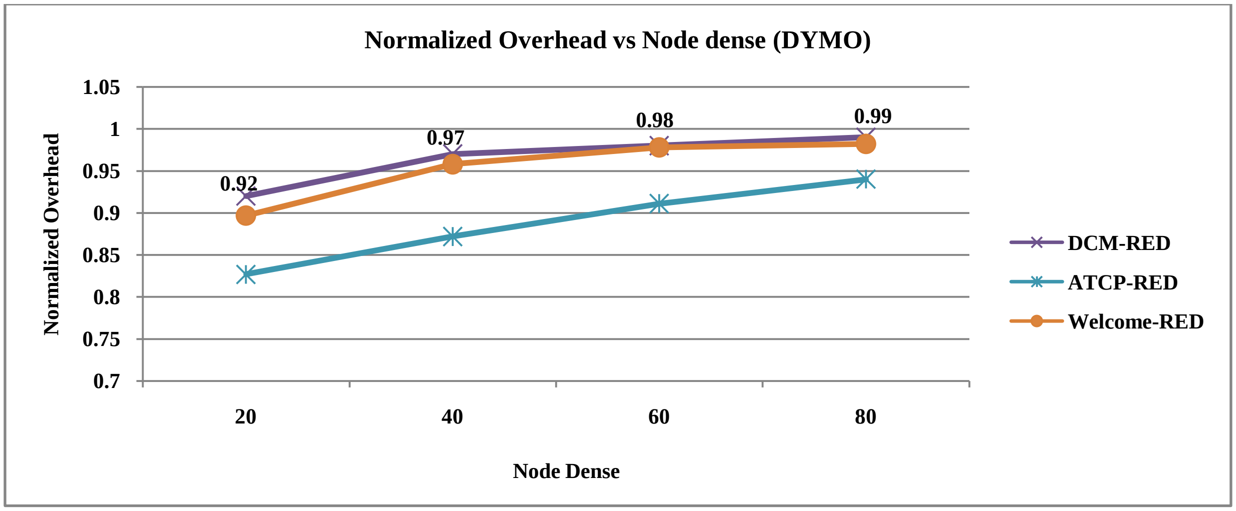


Figure 5-13: Normalized Overhead vs. N_D for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_S= (0-3) m/s and P_L = 5000 bytes over DYMO routing protocol.

In figure 5-13, As N_D increase, the value of overhead is increase over DYMO routing protocols when N_S= (0-3) m/s and P_L = 5000 bytes. Our proposed DCM has slightly higher value of overhead over DYMO routing protocol than ATCP and TCP-WELCOME. The lowest normalized overhead is for ATCP due to its poor management of network congestion.

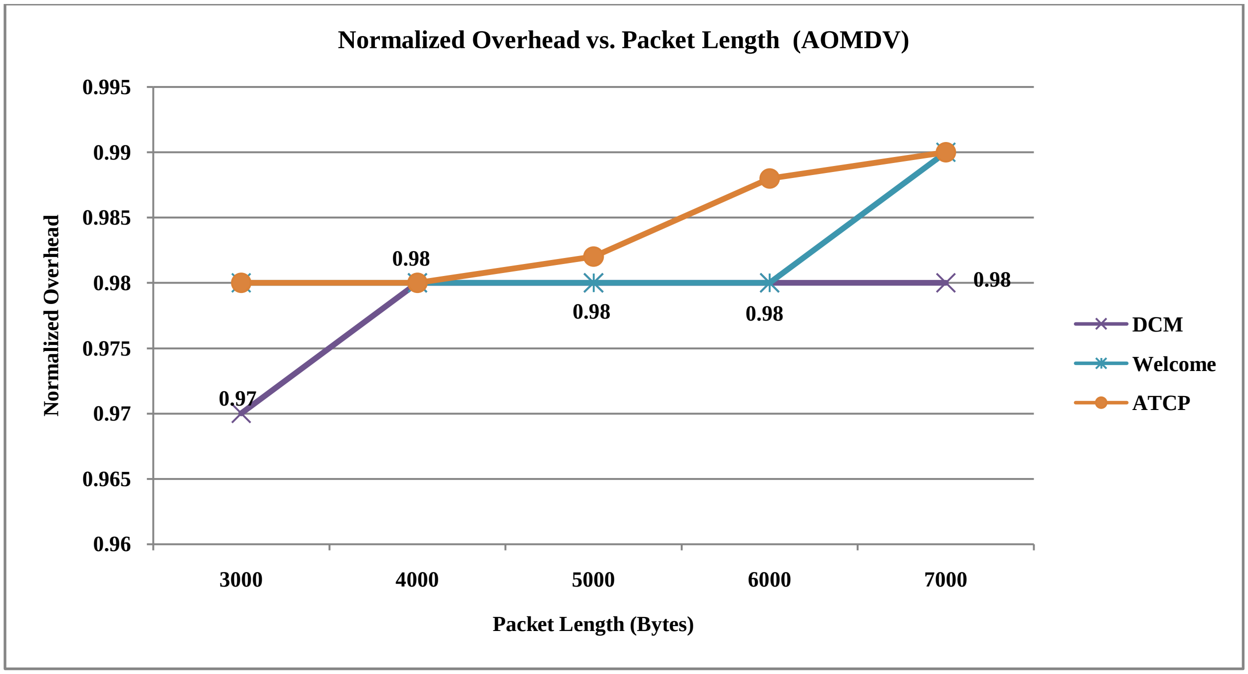


Figure 5-14: Normalized Overhead vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD}= (0-3) m/s and N_D = 80 nodes over AOMDV routing protocol.

In Figure 5-14, the normalized overhead versus packet length when N_{SD}= (0-3) m/s and N_D = 80 bytes over AOMDV routing protocol. DCM has lower overhead than both ATCP and TCP-WELCOME when P_L = 3000 bytes, then the overhead of DCM and TCP-WELCOME become the same within P_L = (4000-6000) bytes. Then the overhead of DCM become lower than both

ATCP and TCP-WELCOME when P_L = 7000 bytes. The overhead enhancement of DCM over TCP-WELCOME is -1% when P_L = 3000 bytes and -1% when P_L = 7000 bytes.

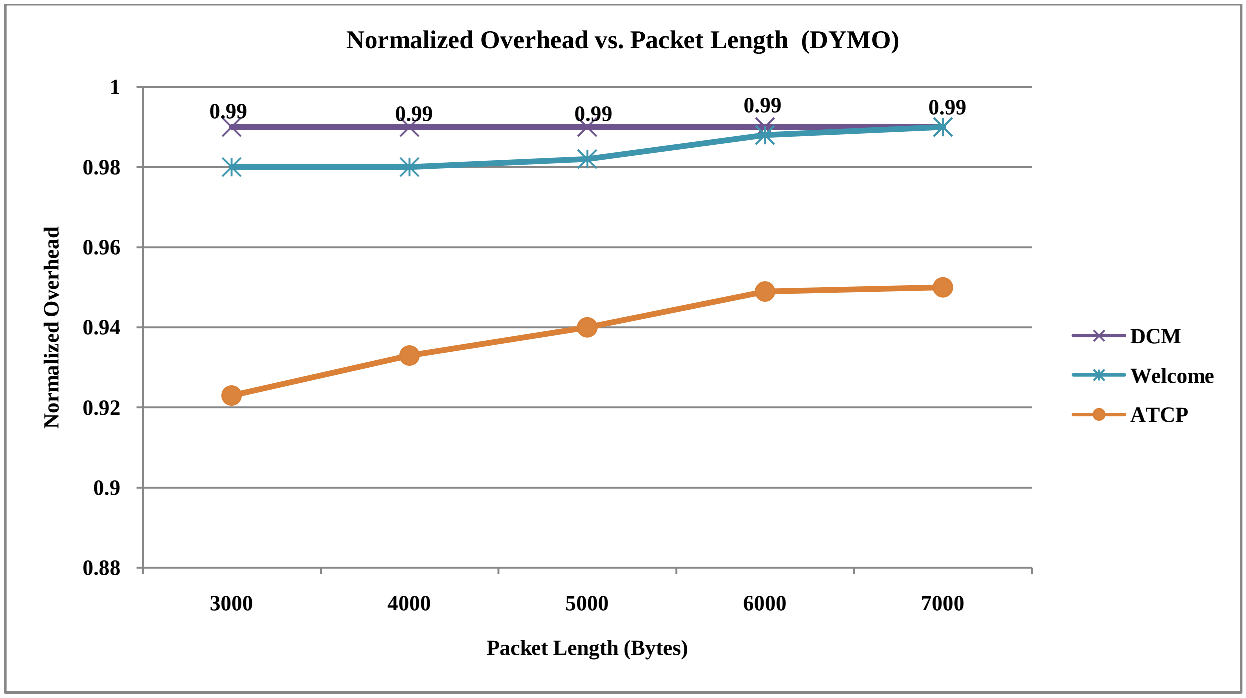


Figure 5-15: Normalized Overhead vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD}= (0-3) m/s and N_D = 80 nodes over DYMO routing protocol.

In Figure 5-15, the overhead versus P_L when N_{SD}= (0-3) m/s and N_D = 80 nodes over DYMO routing protocol. As P_L increase the value of overhead increase. Our proposed DCM model has slightly higher overhead than TCP-WELCOME over DYMO routing protocol within P_L = (3000-6000) bytes, while ATCP has the lowest overhead than DCM and TCP-WELCOME over DYMO routing protocol.

5.5.3 Packet Delivery Ratio

In this section we will present the Packet Delivery Ratio (PDR) of our proposed dynamic congestion model DCM with changing N_D, P_L, queue type and N_{SD} over two routing protocols: AOMDV and DYMO. This metric reflect the efficiency and reliability of the protocol.

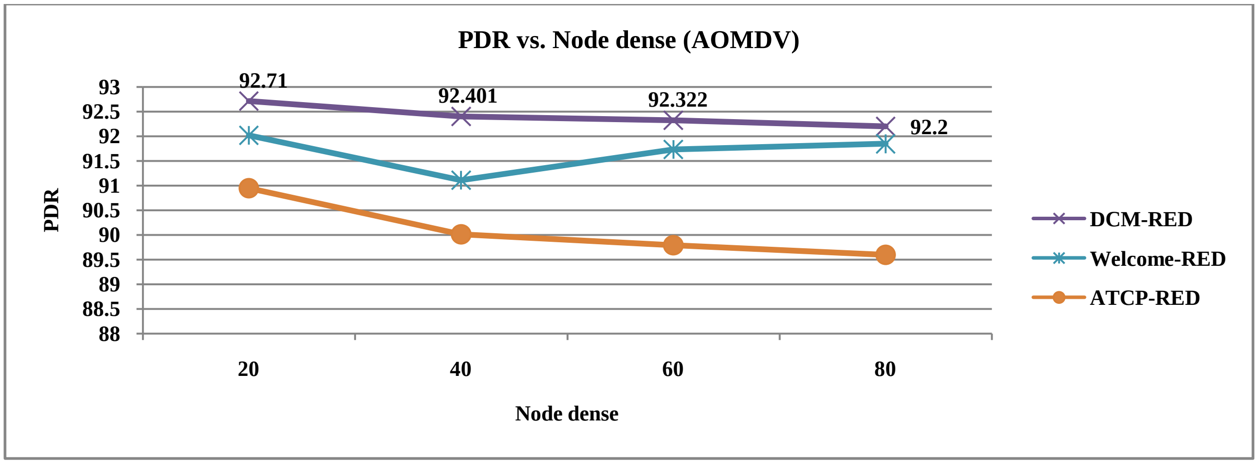


Figure 5-16: Packet Delivery Ratio (PDR) vs. N_D for DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol.

In Figure 5-16, the PDR versus node dense when N_{SD} = (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol. As ND increase, the PDR slightly decrease in DCM, while decrease more in ATCP. TCP-WELCOME has unstable style in decreasing the PDR when N_D increases. DCM has the best PDR over ATCP and TCP-WELCOME. The PDR enhancement of DCM over TCP-WELCOME is as follows: 0.76% when N_D = 20 nodes, 1.4% when N_D = 20 nodes, 0.65% when N_D = 60 nodes, 0.43% when N_D = 80 nodes. The best of overall PDR enhancement of DCM over TCP-WELCOME is when N_D = 20 nodes.

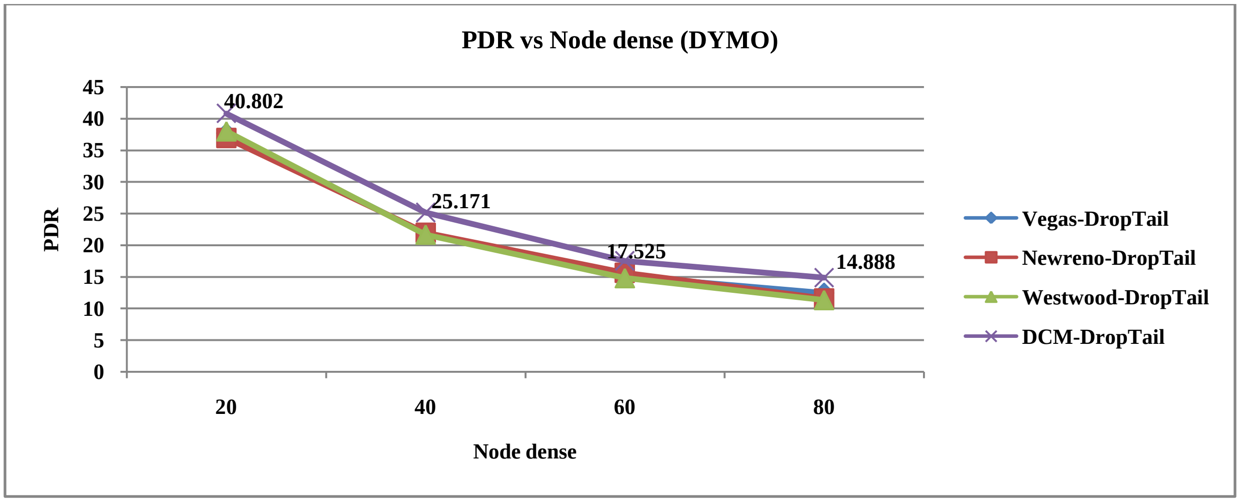


Figure 5-17: Packet Delivery Ratio (PDR) vs. N_D for DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over DYMO routing protocol.

In figure 5-17, The PDR versus node dense when N_{SD} = (0-3) m/s and P_L = 5000 bytes over DYMO routing protocol. PDR decrease under the effect of varying N_D in all TCP's. Our proposed model shows small enhancement of PDR over both DYMO routing protocols. The PDR enhancement of DCM over TCP-WLECOME is as follows: 7.6% when N_D = 20 nodes, 16.2% when N_D = 40 nodes, 19% when N_D = 60 nodes, 30.9% when N_D = 80 nodes. The best overall PDR enhancement from TCP-WELCOME over DYMO is when N_D = 80 nodes.

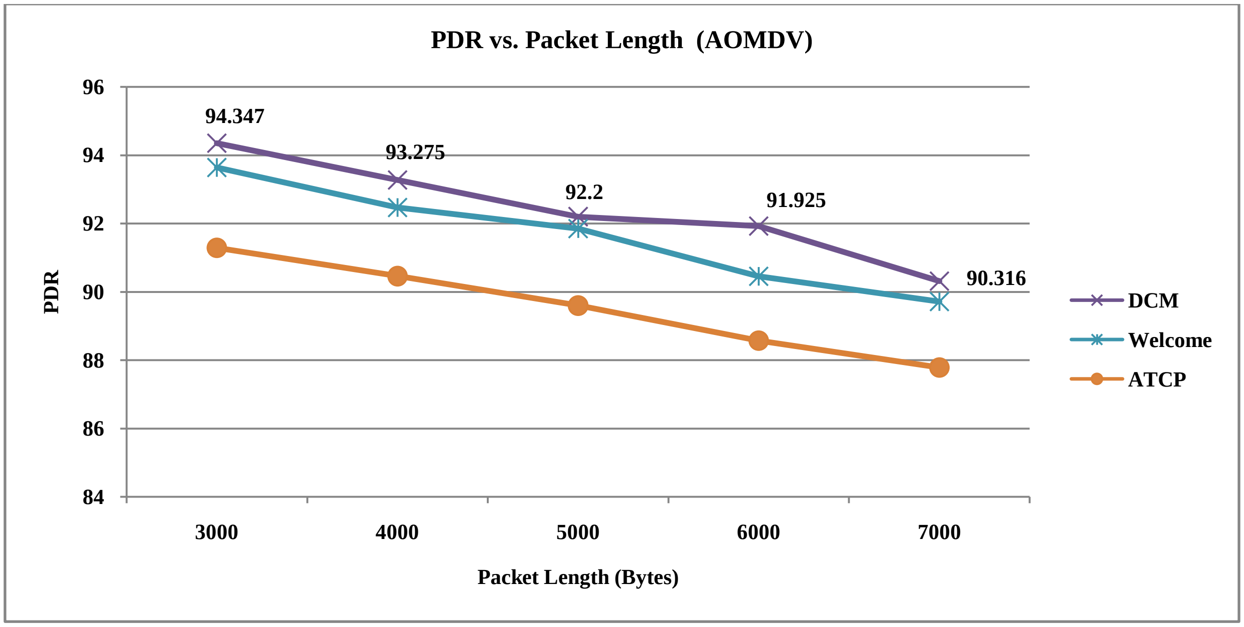


Figure 5-18: Packet Delivery Ratio (PDR) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over AOMDV routing protocol.

In Figure 5-18 the PDR versus packet length when N_{SD} = (0-3) m/s and N_D = 80 bytes over AOMDV routing protocol. As P_L increase, the value of PDR decreases due to the increase in network congestion in all TCPs. ATCP shows the lowest PDR versus N_D, while DCM shows the best PDR versus N_D in all scenarios over AOMDV routing protocol. The PDR enhancement of DCM over TCP-WELCOME is as follows: 0.75% when P_L = 3000 bytes, 0.86% when P_L = 4000 bytes, 0.43% when P_L = 5000 bytes, 1.6% when P_L = 6000 bytes, 0.67% when P_L = 7000 bytes. The best overall PDR enhancement from TCP-WELCOME over AOMDV is 1.6% when P_L = 6000 bytes.

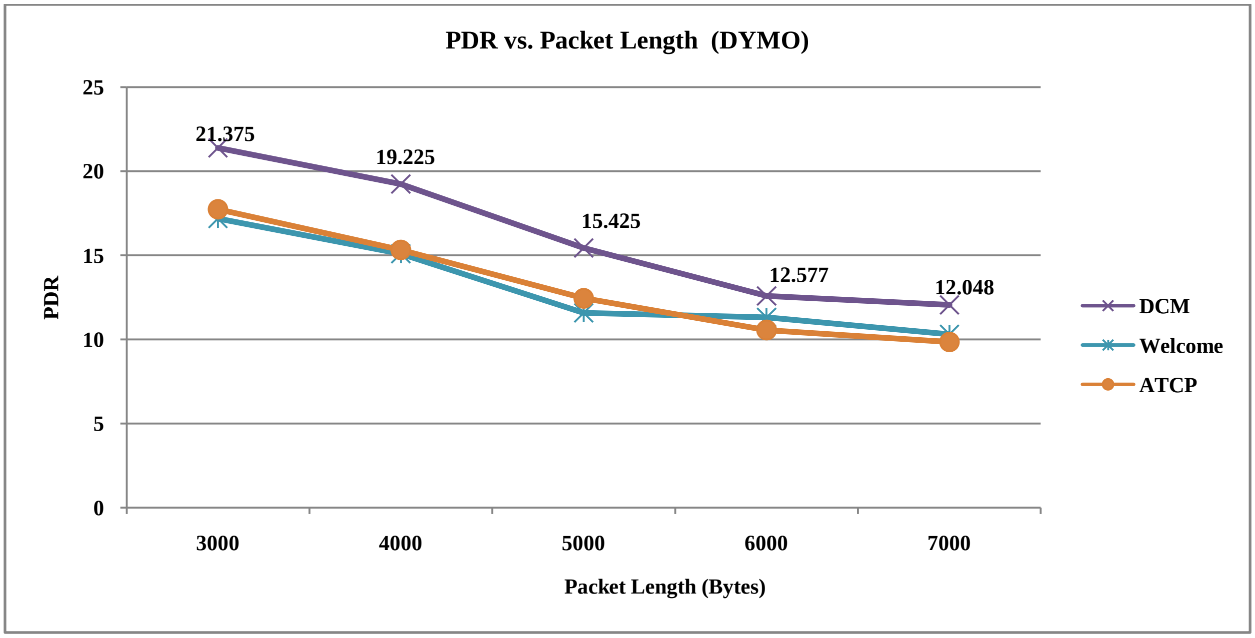


Figure 5-19: Packet Delivery Ratio (PDR) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over DYMO routing protocol.

In Figure 5-19 the PDR versus P_L over DYMO routing protocol when N_{SD} = (0-3) m/s and N_D = 80 nodes. As P_L increase the value of PDR decrease, due to the problem of network congestion. Our DCM dynamic congestion model shows higher PDR than TCP-WELCOME and ATCP.

The overall PDR enhancement from TCP-WELCOME over DYMO routing protocol is as follows: 23.8% when P_L = 3000 bytes, 27.1% when P_L = 4000 bytes, 33.9% when P_L = 5000 bytes, 10.6% when P_L = 6000 bytes, 16.5% when P_L = 7000 bytes. The best overall enhancement of DCM over TCP-WELCOME when N_D = 80 nodes is 33.9% when P_L = 5000 bytes.

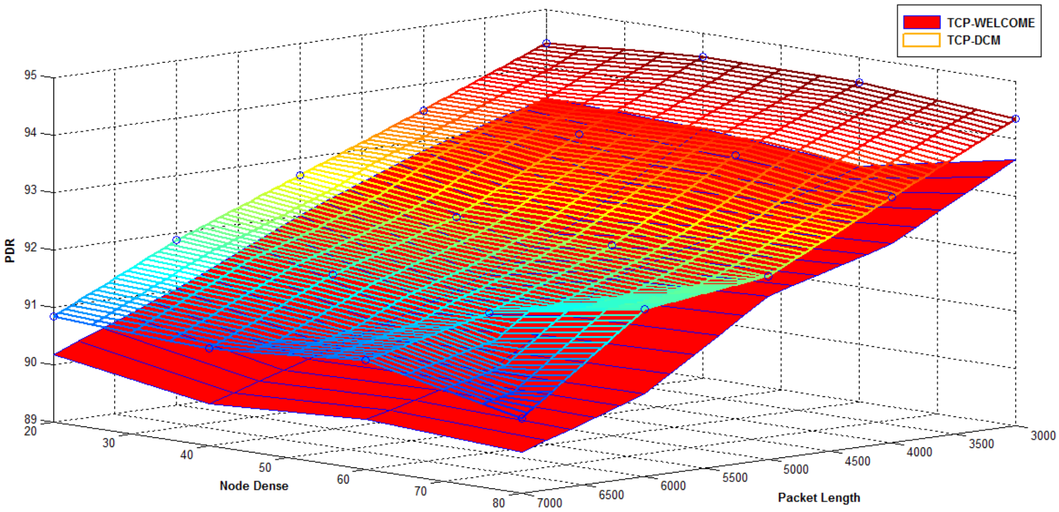


Figure 5-20: Packet Delivery Ratio (PDR) vs. N_D and P_L for TCP-DCM and TCP-WELCOME over AOMDV routing protocol, with queue type RED.

In Figure 5-20, the PDR versus both N_D and P_L over AOMDV routing protocol for DCM and TCP-WELCOME. The PDR of DCM is better than the PDR of TCP-WELCOME in all scenarios. As packet length increase the value of PDR decrease due to the increase in network congestion. The best overall PDR of DCM from TCP-WELCOME over AOMDV is 1.6% when P_L = 6000 bytes and N_D = 80 nodes and the lowest overall enhancement is 0.38% when P_L = 5000 bytes and N_D = 80 nodes.

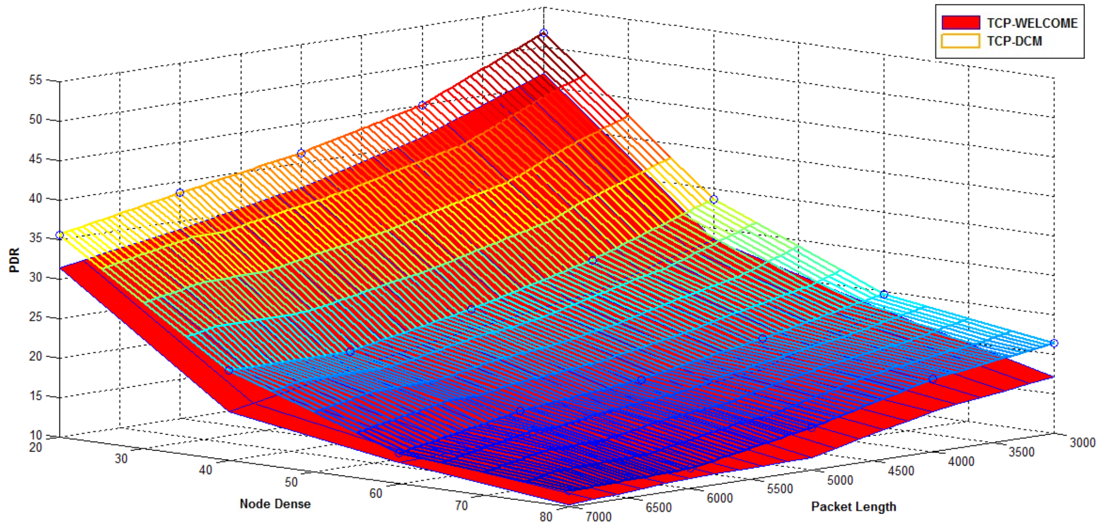


Figure 5-21: Packet Delivery Ratio (PDR) vs. N_D and P_L for TCP-DCM and TCP-WELCOME over DYMO routing protocol, with queue type RED.

In figure 5-21, a three dimension graph to represent the effect of varying N_D and P_L on PDR of our proposed DCM and TCP-WELCOME when N_{SD} = (0-3) m/s. As P_L increase, the number of packets drop increase due to congestion thus the PDR decrease. In addition, the value of PDR decreases as N_D increase. The PDR of is better than PDR of TCP-WELCOME in all scenarios. The best and lowest enhancement of PDR of DCM from TCP-WELCOME over DYMO is as follows: 27.4% when P_L = 4000 bytes and N_D = 80 nodes, 9.1% when P_L = 7000 bytes and N_D = 60 nodes.

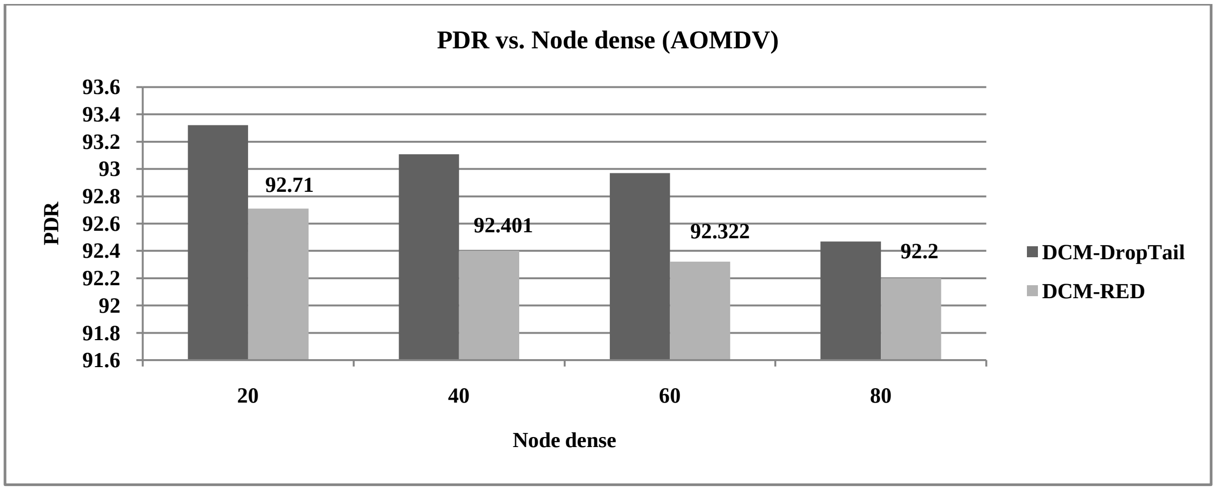


Figure 5-22: Packet Delivery Ratio (PDR) vs. N_D for TCP-DCM over AOMDV routing protocol, queue types: D_T and RED, N_{SD} = (0-3) m/s, P_L = 5000 bytes.

In Figure 5-22, the effect of changing queue management type on the PDR versus node dense when N_{SD} = (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol. The PDR when implementing the Drop-Tail queue management is slightly better than the PDR when implementing RED queue management. In addition, the PDR is decreasing as N_D increase in both queue management mechanisms.

The enhancement of implementing Drop-Tail queue management from RED queue management over AOMDV routing protocol when N_{SD} = (0-3) m/s and P_L = 5000 bytes is as follows: 0.64% when N_D = 20 nodes, 0.76% when N_D = 20 nodes, 0.65% when N_D = 20 nodes, 0.22% when N_D = 20 nodes.

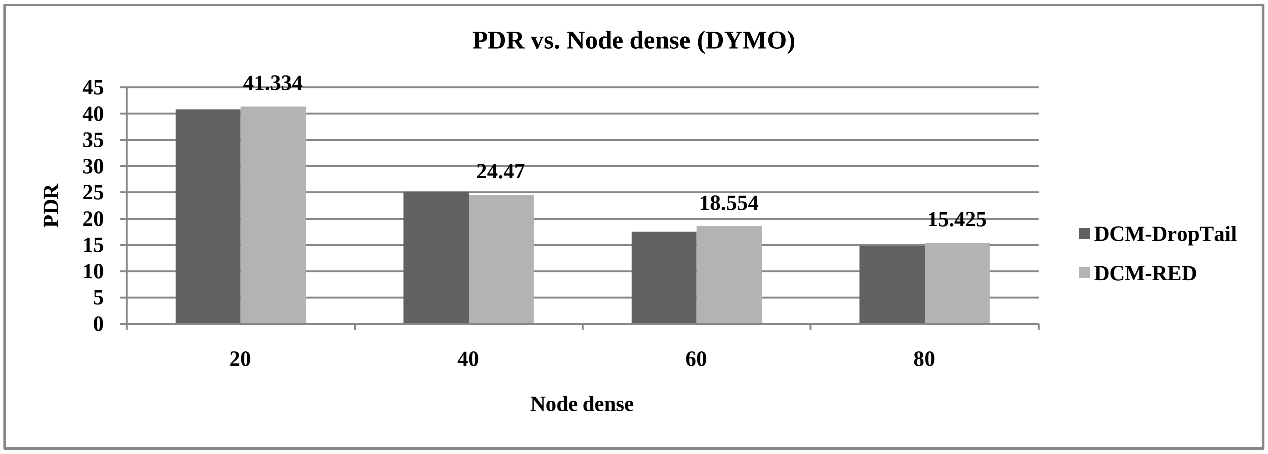


Figure 5-23: Packet Delivery Ratio (PDR) vs. N_D for TCP-DCM over DYMO routing protocol, queue types: D_T and RED, N_{SD} = (0-3) m/s, P_L = 5000 bytes.

In figure 5-23, the effect of changing the queue management type on PDR for both D_T and RED over DYMO when N_{SD} = (0-3) m/s, P_L = 5000 bytes. The PDR versus N_D is of RED queue management is slightly better than the PDR of implementing DT queue management. The PDR is decreasing in both implemented queue management mechanisms as N_D increase. The PDR enhancement of RED queue management over D_T is as follows: 1.2% when N_D = 20 nodes, 2.8% when N_D = 40 nodes, 5.7% when N_D = 60, nodes4.1% when N_D = 80 nodes.

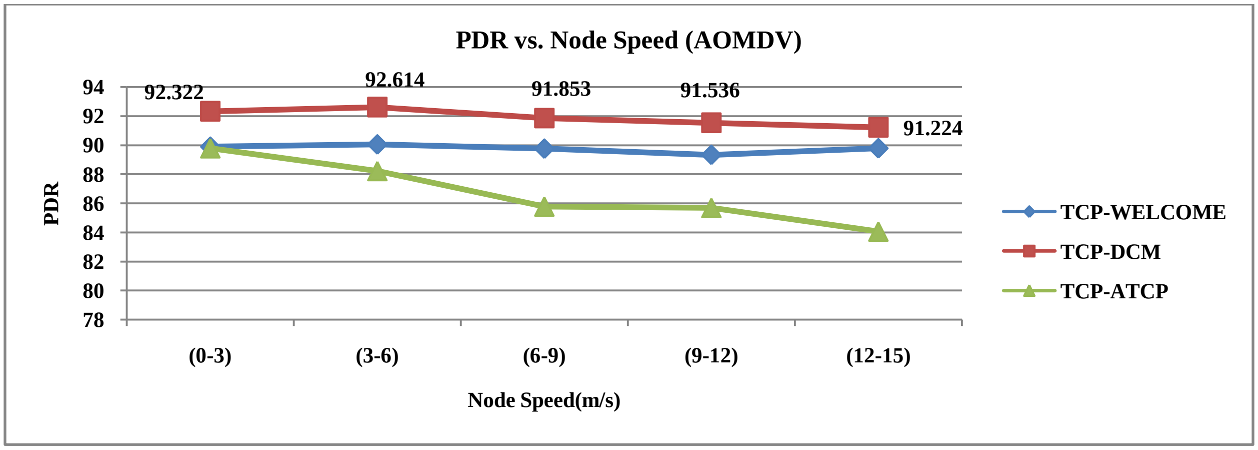


Figure 5-24: Packet Delivery Ratio (PDR) vs. N_{SD} for TCP-DCM over AOMDV routing protocol, queue types: RED, N_D = 60 and RED, P_L = 5000 bytes.

In Figure 5-24, the PDR versus node speed when N_D = 60 and RED, P_L = 5000 bytes over AOMDV routing protocol. As node speed increase, the PDR slightly decrease in all TCPs. DCM has better PDR than TCP-WELCOME and ATCP when changing node speed. The PDR enhancement from TCP-WELCOME over AOMDV is as follows: 2.8% when N_{SD} = (0-3) m/s, 2.9% when N_{SD} = (3-6) m/s, 2.3% when N_{SD} = (6-9) m/s, 2.4% when N_{SD} = (9-12) m/s, 1.7% when N_{SD} = (12-15) m/s. The best PDR enhancement by DCM over TCP-WELCOME is 2.9% when N_{SD} = (3-6) m/s.

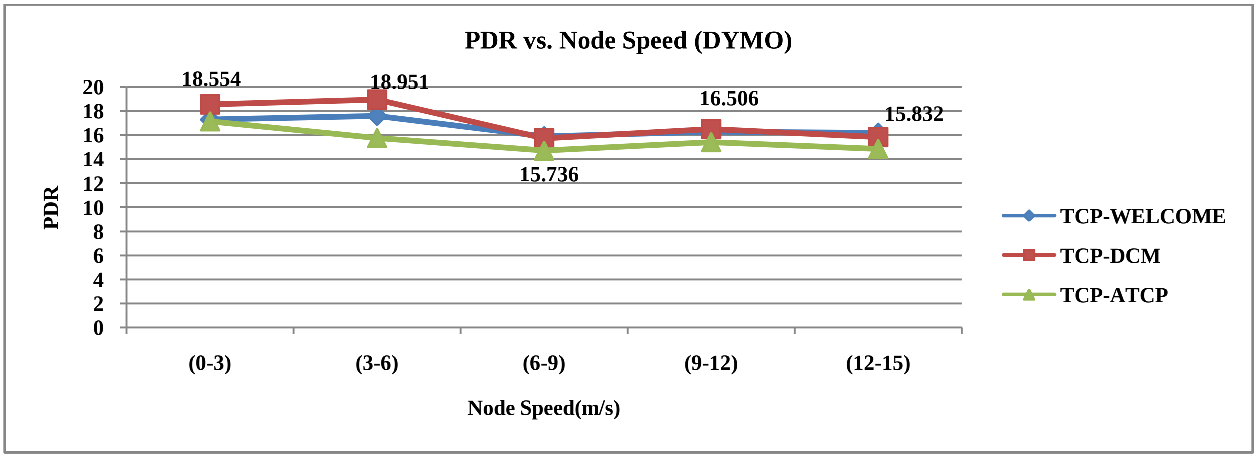


Figure 5-25: Packet Delivery Ratio (PDR) vs. N_S for TCP-DCM over DYMO routing protocol, queue types: D_T, N_D = 60 nodes and RED, P_L = 5000 bytes.

In Figure 5-25, the PDR versus node speed over DYMO routing protocol when N_D = 60 and RED, P_L = 5000 bytes. As N_{SD} increase our proposed DCM has stability in PDR. ATCP has the lowest PDF over DYMO routing protocol. The enhancement in PDR by DCM over TCP-WELCOME is appearing within N_{SD} = (0-6) m/s as follows: 6.9% when N_{SD} = (0-3) m/s and 7.4% when N_{SD} = (3-6) m/s.

5.5.4 Average End-to-End Delay

In this section we will present the $E2E_D$ of our proposed dynamic congestion model DCM with changing N_D, P_L, queue type and node speed over two routing protocols: AOMDV and DYMO. This metric reflect the delay in the interface queues of the intermediate nodes between the sender node and the receiver node.

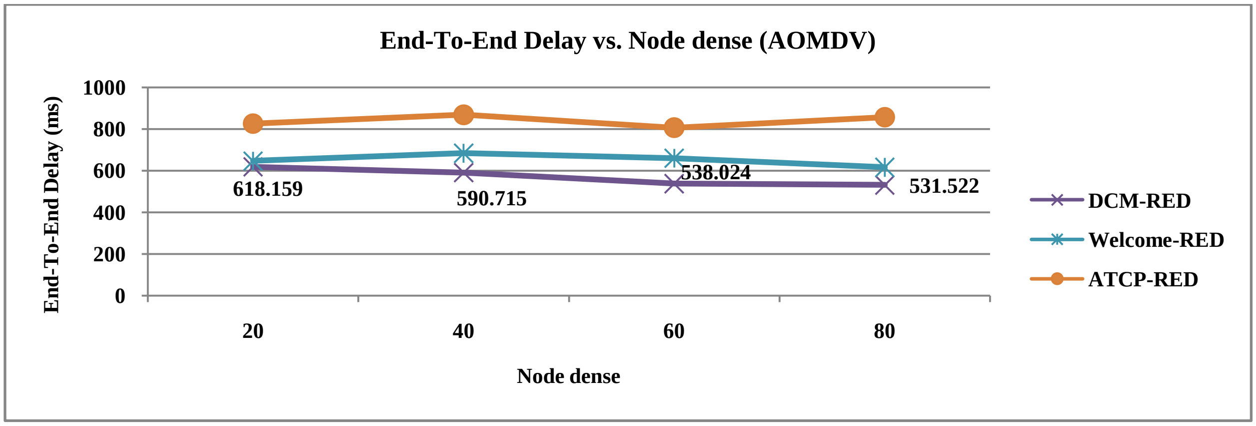


Figure 5-26: $E2E_D$ (ms) vs. N_D for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol.

In Figure 5-26, the end-to-end delay versus node dense over AOMDV routing protocol when N_{SD} = (0-3) m/s and P_L = 5000 bytes. As N_D increase, the $E2E_D$ decrease, because the nodes get closer to each other. The $E2E_D$ of ATCP is lower than the $E2E_D$ of DCM and TCP-WELCOME. DCM has the best $E2E_D$ than ATCP and TCP-WELCOME due its dynamic technique that avoids the congested path. The effect of varying N_D on $E2E_D$ is slightly low over AOMDV routing protocol.

The $E2E_D$ enhancement of DCM from TCP-WELCOME over AOMDV when N_{SD} = (0-3) m/s and P_L = 5000 bytes is as follows: 4.6% when N_D = 20 nodes, 15.8% when N_D = 40 nodes, 22.7% when N_D = 60 nodes, 15.9% when N_D = 80 nodes. The best enhancement of DCM over TCP-WELCOME is 22.7% when N_D = 60 nodes.

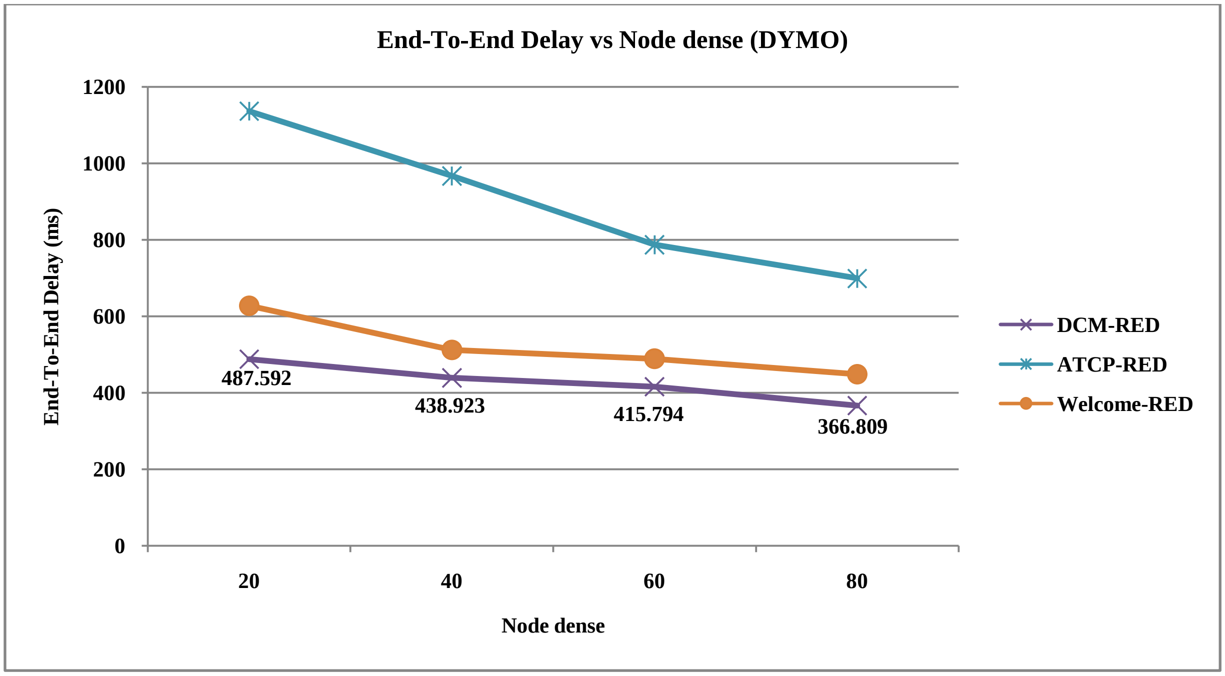


Figure 5-27: $E2E_D$ (ms) vs. N_D for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over DYMO routing protocol.

In figure 5-27, the effect of varying node dense on the value of $E2E_D$ over DYMO routing protocol when N_{SD} = (0-3) m/s and P_L = 5000 bytes. The increase in node dense will decrease slightly the average $E2E_D$ in both TCP-WELCOME and DCM.

The decrease in $E2E_D$ of ATCP as node dense increase is much noticeable than that in DCM and TCP-WELCOME. The $E2E_D$ of DCM is lower than ATCP and TCP-WELCOME. The $E2E_D$ enhancement of DCM from TCP-WELCOME over DYMO routing protocol is as follows: 28.6% when N_D = 20 nodes, 16.7% when N_D = 20 nodes, 17.6% when N_D = 20 nodes, 22.3% when N_D = 20 nodes. The best $E2E_D$ enhancement of DCM over TCP-WELCOME is 28.6% when N_D = 20 nodes and the lowest $E2E_D$ enhancement of DCM over TCP-WELCOME is 16.7% when N_D = 20 nodes.

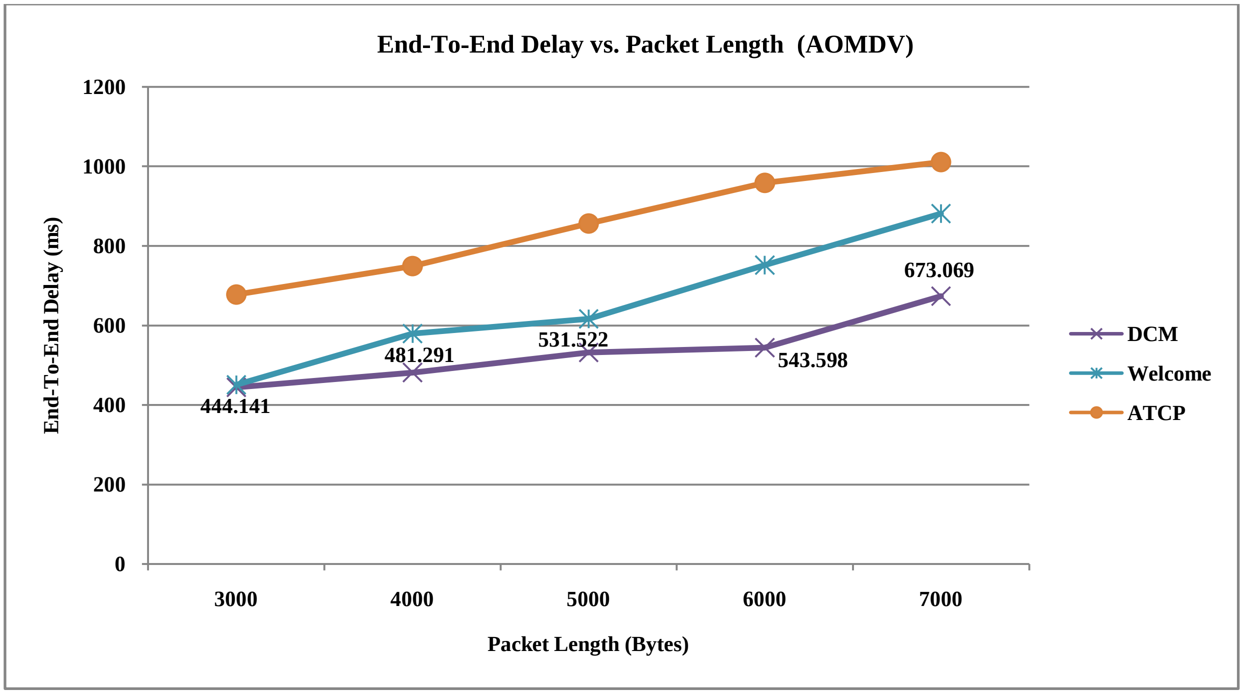


Figure 5-28: $E2E_D$ (ms) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over AOMDV routing protocol.

In Figure 5-28, the value of end-to-end delay versus packet length over AOMDV routing protocol when N_{SD} = (0-3) m/s and N_D = 80 bytes. As P_L increase, the average $E2E_D$ increase due to the increase in the delay of packet processing within congested node's queues. The $E2E_D$ of DCM is lower than the $E2E_D$ of ATCP and TCP-WELCOME, while ATCP has the highest average $E2E_D$.

The average $E2E_D$ enhancement of DCM from TCP-WELCOME over AOMDV when N_{SD} = (0-3) m/s and N_D = 80 bytes is as follows: 1.4% when P_L = 3000 bytes, 20.3% when P_L = 4000 bytes, 15.9% when P_L = 5000 bytes, 38.2% when P_L = 6000 bytes, 30.9% when P_L = 7000 bytes. The highest $E2E_D$ enhancement is 38.2% and is achieved when P_L = 6000 bytes and the lowest $E2E_D$ enhancement is 1.4% when P_L = 3000 bytes.

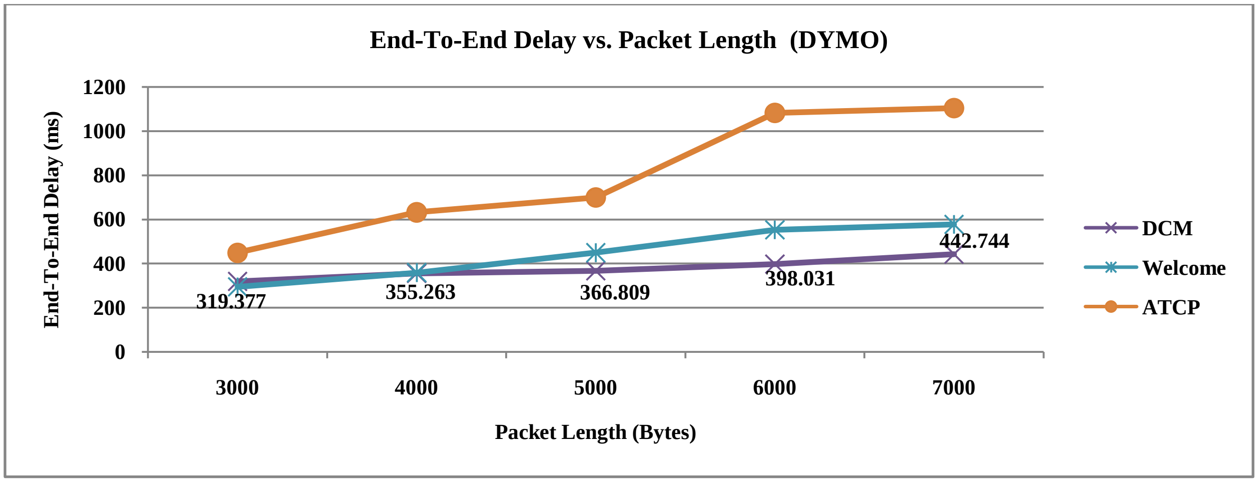


Figure 5-29: $E2E_D$ (ms) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over DYMO routing protocol.

In Figure 5-29, the average $E2E_D$ versus P_L over DYMO routing protocol when N_{SD} = (0-3) m/s and N_D = 80 nodes. As P_L increase the average $E2E_D$ increase, due to the problem of network congestion. Our DCM dynamic congestion model shows lower average $E2E_D$ value than TCP-WELCOME and ATCP. The $E2E_D$ enhancement by DCM over TCP-WELCOME is as follows: -8.5% when P_L = 3000 bytes, 0.92% when P_L = 4000 bytes, 18.3% when P_L = 5000 bytes, 27.9% when P_L = 6000 bytes, 23.2% when P_L = 7000 bytes.

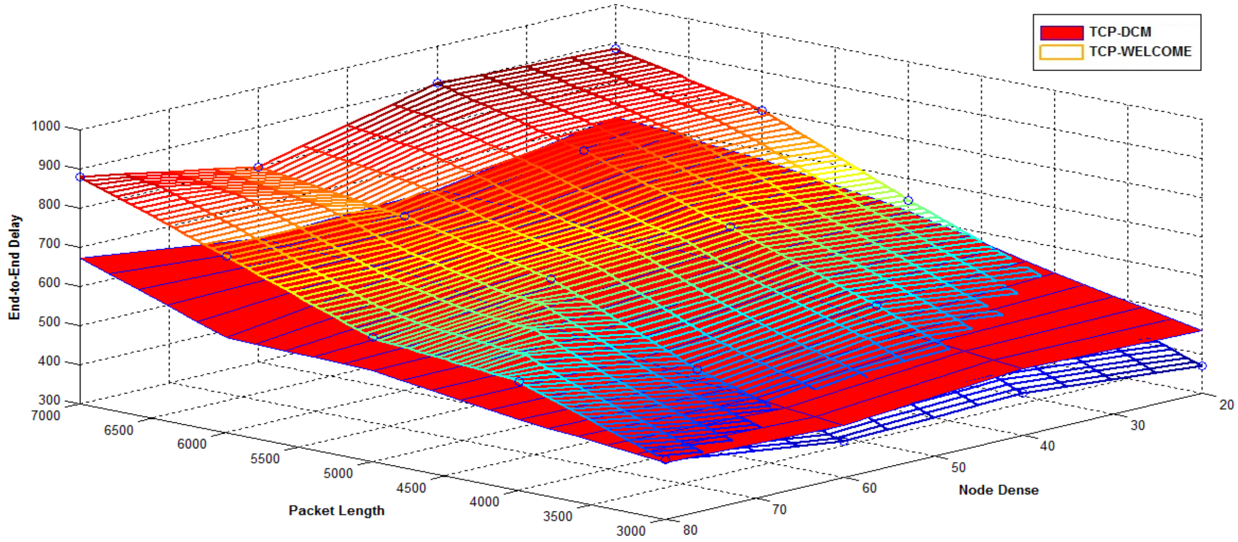


Figure 5-30: $E2E_D$ (ms) vs. N_D and P_L for TCP-DCM and TCP-WELCOME over AOMDV routing protocol, with queue type RED and N_{SD} = (0-3) m/s.

In Figure 5-30, a dimensional graph to represent the average $E2E_D$ versus node dense and packet length over AOMDV when N_{SD} = (0-3) m/s for DCM and TCP-WELCOME. The average $E2E_D$ enhancement by DCM over TCP-WELCOME is achieved as P_L increase above 4000 bytes which means as network become more congested the DCM perform much better. The best $E2E_D$ enhancement of DCM from TCP-WELCOME over AOMDV routing protocol at N_{SD} = (0-3) m/s is 29.9% when P_L = 7000 bytes and N_D = 40 nodes.

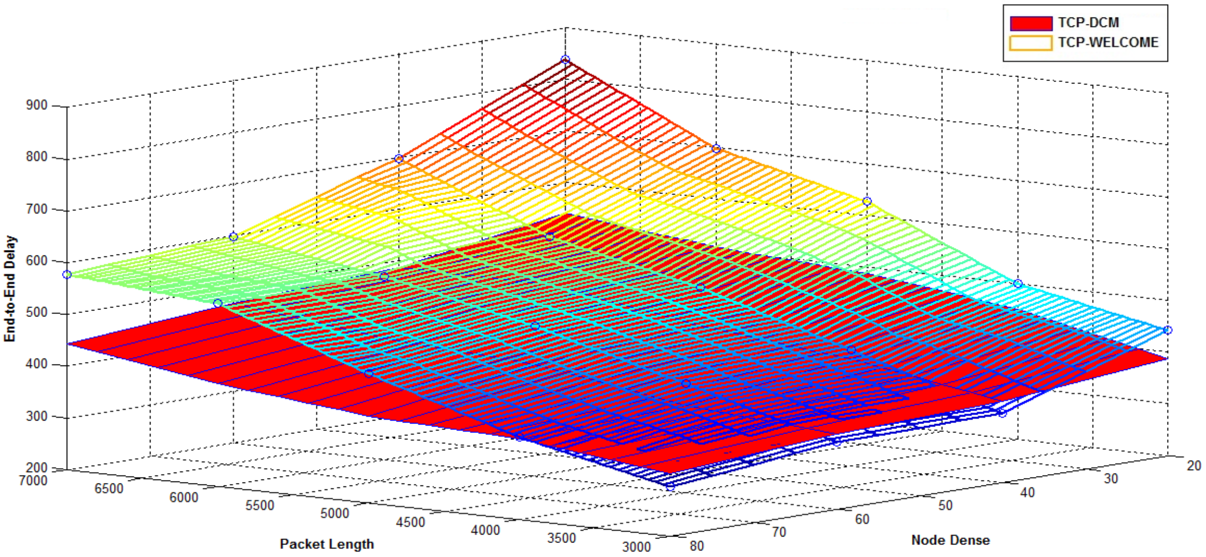


Figure 5-31: $E2E_D$ (ms) vs. N_D and P_L for TCP-DCM and TCP-WELCOME over DYMO routing protocol, with queue type RED and N_{SD} = (0-3) m/s.

In Figure 5-31 a three dimension graphs to represent the effect of varying N_D and P_L on average $E2E_D$ of our proposed DCM and TCP-WELCOME. The average $E2E_D$ of DCM has better enhancement than TCP-WELCOME over DYMO routing protocol in most of the simulation scenarios.

The average $E2E_D$ enhancement of DCM over TCP-WELCOME is achieved as P_L increase. The best overall average $E2E_D$ enhancement of DCM from TCP-WELCOME over DYMO routing protocol when N_{SD} = (0-3) m/s is 35.4% when P_L = 7000 bytes and N_D = 20 nodes.

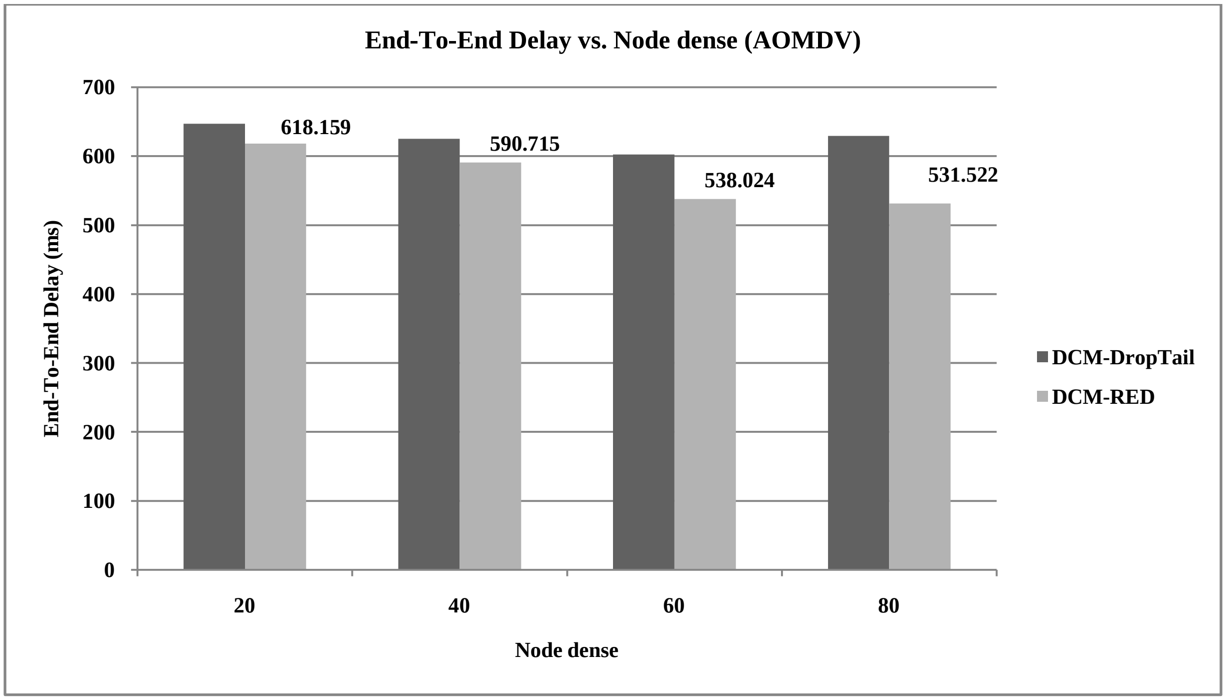


Figure 5-32: $E2E_D$ (ms) vs. N_D for TCP-DCM over AOMDV routing protocol, queue types: D_T, N_{SD} = (0-3) m/s and RED, P_L = 5000 bytes.

In Figure 5-32, the effect of changing the implemented queue management technique on average $E2E_D$ versus node dense over AOMDV routing protocol. The average $E2E_D$ remain stable as node dense increase in both implemented queue management techniques: RED and Drop-Tail. Drop-Tail queue management technique has slightly lower average $E2E_D$ than RED. The average $E2E_D$ enhancement that achieved by implementing Drop-Tail over RED is as follows: 4.7% when N_D = 20 nodes, 5.8% when N_D = 40 nodes, 11.9% when N_D = 60 nodes and 18.4% when N_D = 80 nodes. The best average $E2E_D$ enhancement of DCM over TCP-WELCOME is 18.4% when N_D = 80 nodes.

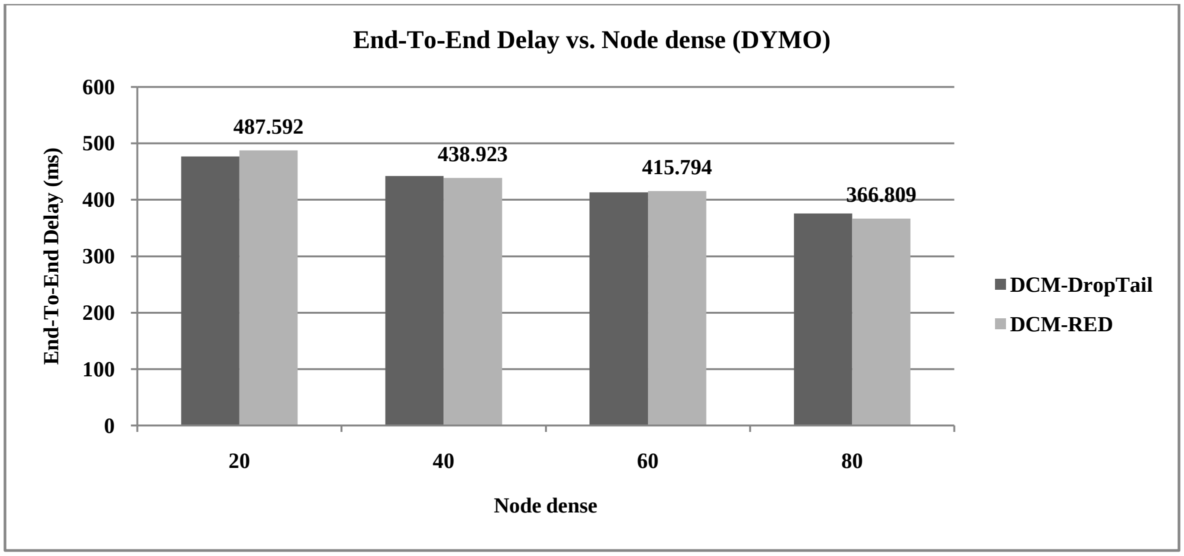


Figure 5-33: $E2E_D$ (ms) vs. N_D for TCP-DCM over DYMO routing protocol, queue types: D_T, N_{SD} = (0-3) m/s and RED, P_L = 5000 bytes.

In figure 5-33 the effect on changing the queue management type on average $E2E_D$ for both Drop-Tail and RED over DYMO routing protocol when N_{SD} = (0-3) m/s. As N_D increase, the average end-to-end delay decreases slightly. The effect of changing the queue management technique over DYMO routing protocol is very low.

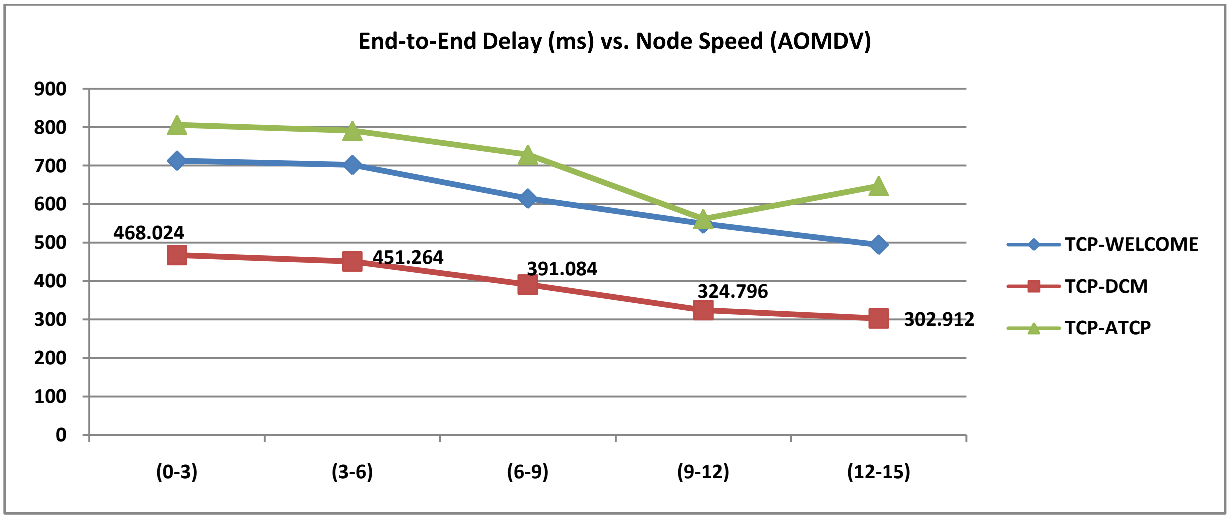


Figure 5-34: End-to-End Delay (ms) vs. N_S for TCP-DCM over AMODV routing protocol, queue types: D_T, N_D = 60 nodes and RED, P_L = 5000 bytes.

In Figure 5-34, the end-to-end delay versus node speed when N_D = 60 and RED, P_L = 5000 bytes over AOMDV routing protocol. As N_{SD} increase, the average $E2E_D$ decrease due to the increase in number of link failure.

The average $E2E_D$ of ATCP is higher than the average $E2E_D$ of DCM and TCP-WELCOME, while the average $E2E_D$ of DCM is lower than average $E2E_D$ of ATCP and TCP-WELCOME. The average $E2E_D$ enhancement of DCM from TCP-WELCOME over AOMDV when N_D = 60 and RED, P_L = 5000 bytes is as follows: 34.3% when N_{SD} = (0-3) m/s, 35.7% when N_{SD} = (3-6) m/s, 36.4% when N_{SD} = (6-9) m/s, 40.9% when N_{SD} = (9-12) m/s, 38.7% when N_{SD} = (12-15) m/s.

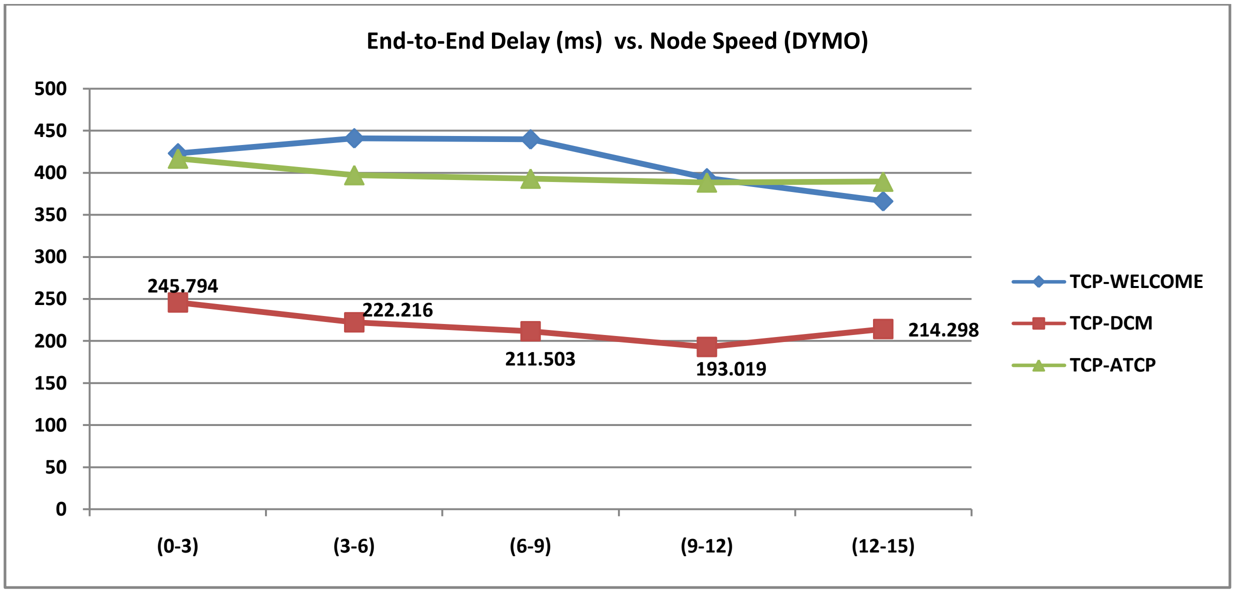


Figure 5-35: End-to-End Delay (ms) vs. N_S for TCP-DCM over DYMO routing protocol, queue types: D_T, N_D = 60 nodes and RED, P_L = 5000 bytes.

In Figure 5-35, the average end-to-end delay versus node speed over DYMO routing protocol when N_D = 60 and RED, P_L = 5000 bytes. The average $E2E_D$ remain stable as N_S increase over DYMO. The average $E2E_D$ of TCP-WELCOME is higher than both ATCP and DCM, while the average $E2E_D$ of DCM is lower than ATCP and TCP-WELCOME. DCM has peter enhancement of the average $E2E_D$ than TCP-WELCOME over DYMO when N_D = 60 and

RED, P_L = 5000 bytes as follows: 41.9% when N_{SD} = (0-3) m/s, 49.6% when N_{SD} = (3-6) m/s, 51.9% when N_{SD} = (6-9) m/s, 50.9% when N_{SD} = (9-12) m/s and 41.5% when N_{SD} = (12-15) m/s. The best average $E2E_D$ of DCM from TCP-WELCOME over DYMO routing protocol is 51.9% and achieved when N_{SD} = (6-9) m/s

5.5.5 Overall Energy Consumption

In this section we will present the Energy Consumption (E_C) of our proposed dynamic congestion model DCM with changing N_D, P_L, queue type and N_S over two routing protocols: AOMDV and DYMO. This metric is important to determine the life time of the network.

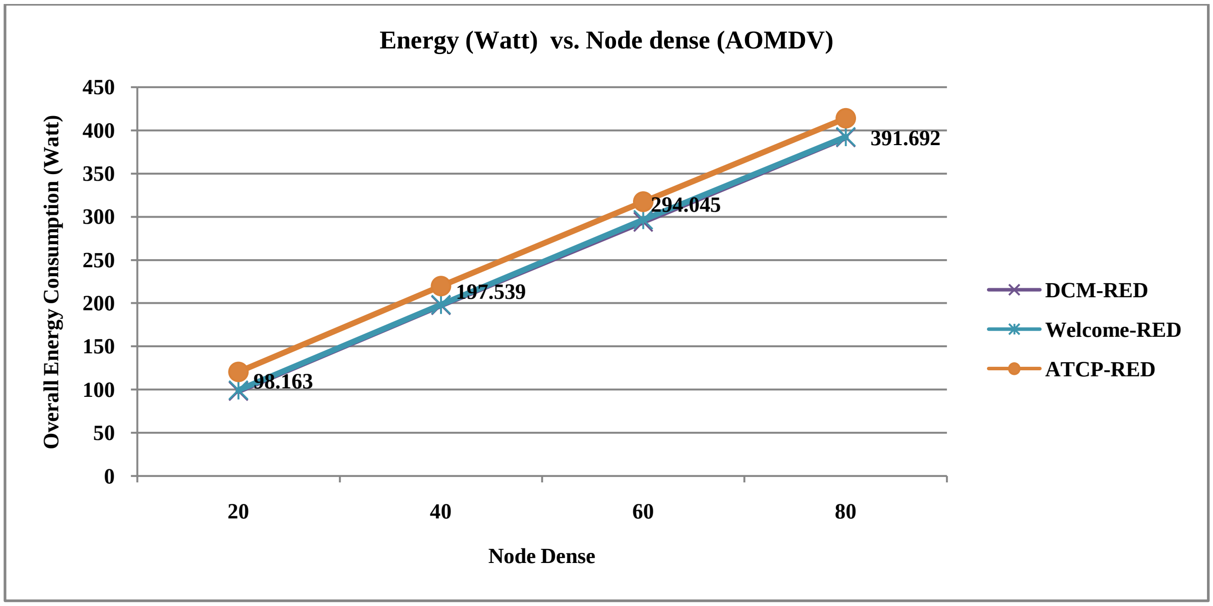


Figure 5-36: Overall E_C (Watt) vs. N_D for DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over AOMDV routing protocol.

In Figure 5-36, the overall energy consumption versus node dense over AOMDV routing protocol when N_{SD} = (0-3) m/s and P_L = 5000 bytes. As N_{SD} increase, the overall E_C is increasing. The overall E_C in ATCP is slightly higher than DCM and TCP-WELCOME, while the E_C in DCM is almost the same as TCP-WELCOME.

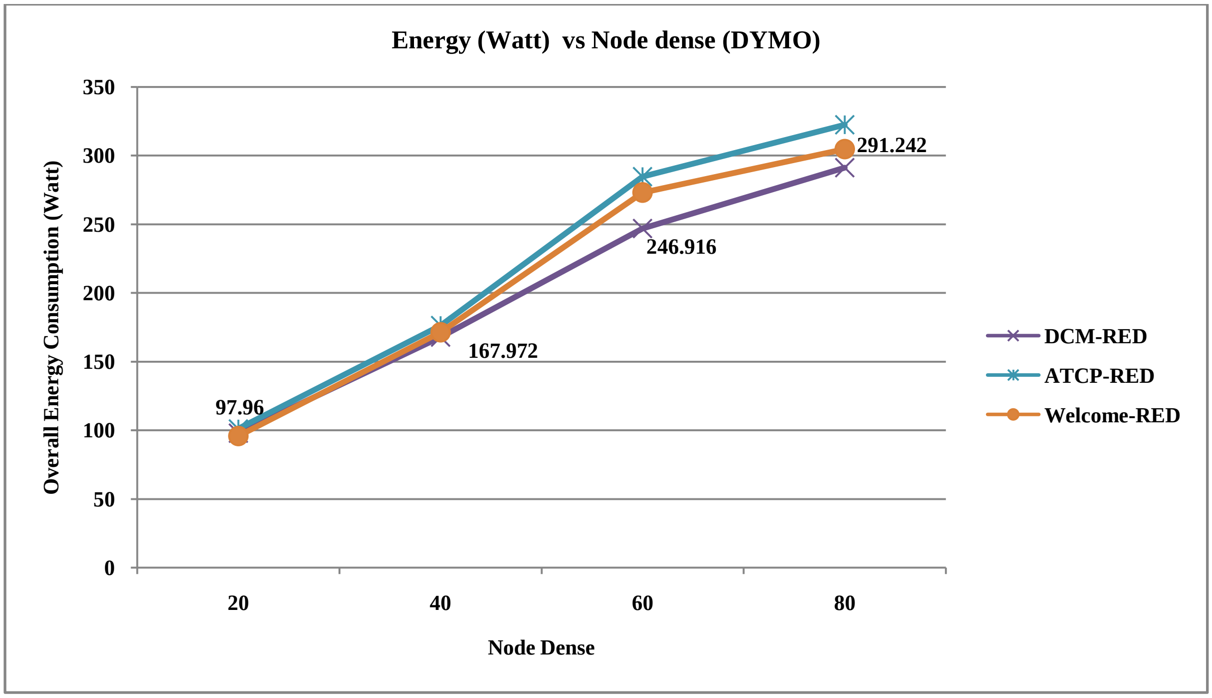


Figure 5-37: Overall E_C (Watt) vs. N_D for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and P_L = 5000 bytes over DYMO routing protocol.

In figure 5-37, The Overall E_C versus node dense over DYMO routing protocol when N_{SD} = (0-3) m/s and P_L = 5000 bytes. As N_D increase, the value of E_C is increase over DYMO routing protocol. Our proposed DCM shows small enhancement of E_C from the overall E_C of TCP-WELCOME over DYMO routing protocol.

The overall EC of DCM from TCP-WELCOME when N_{SD} = (0-3) m/s and P_L = 5000 bytes over DYMO routing protocol is as follows: -2.3% when N_D = 20 nodes, 1.9% when N_D = 40 nodes, 9.5% when N_D = 60 nodes and 4.4% when N_D = 20 nodes. The best overall energy consumption enhancement of DCM over TCP-WELCOME when N_{SD} = (0-3) m/s and P_L = 5000 bytes is 9.5% and achieved when N_D = 60 nodes.

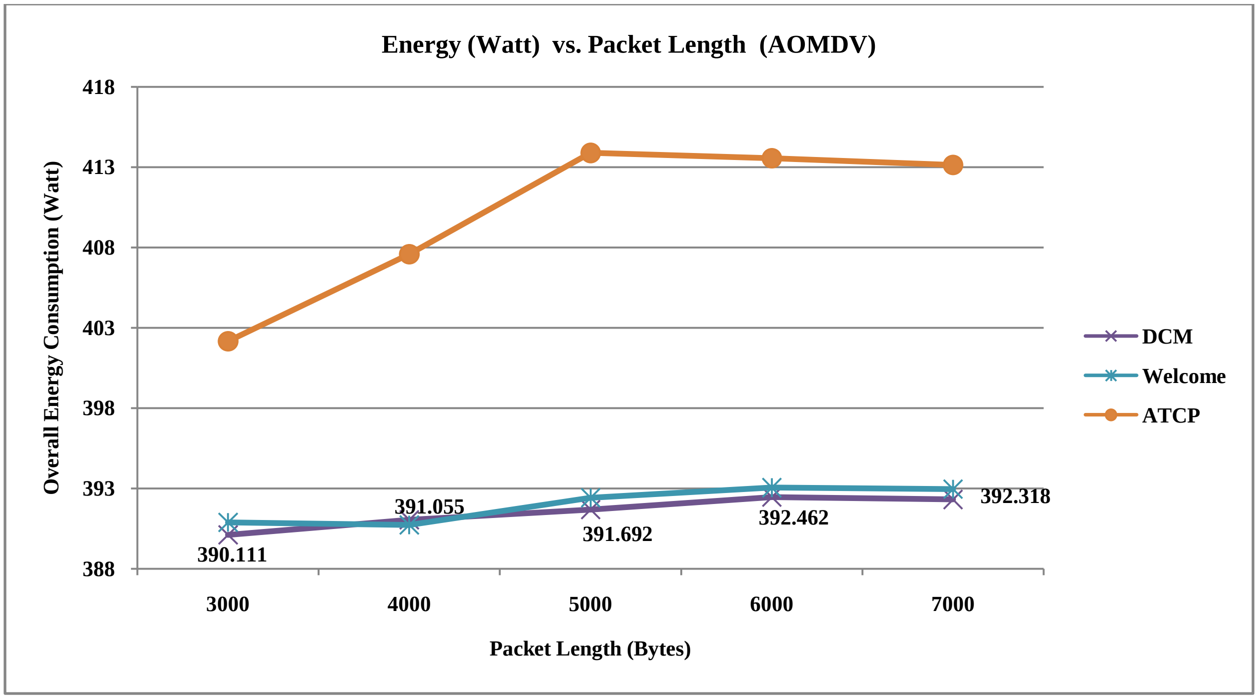


Figure 5-38: Overall Energy consumption (Watt) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over AOMDV routing protocol.

In Figure 5-38, the overall energy consumption versus packet length over AOMDV routing protocol when N_{SD} = (0-3) m/s and N_D = 80 nodes. As P_L increase, the overall E_C in slightly increase due to increase in network congestion which require more control packets that means more energy consumption. The E_C of ATCP is higher than DCM and TCP-WELCOME, while the overall EC of DCM is lower than ATCP and TCP-WELCOME as P_L increase. The overall EC enhancement of DCM from TCP-WELCOME over AOMDV when N_{SD} = (0-3) m/s and N_D = 80 nodes is: 0.18% when P_L = 3000 bytes, -0.08% when P_L = 4000 bytes, 0.2% when P_L = 5000 bytes, 0.15% when P_L = 6000 bytes and 0.15% when P_L = 7000 bytes. The best overall energy consumption of DCM over TCP-WELCOME is 0.2% and can be achieved when P_L = 5000 bytes

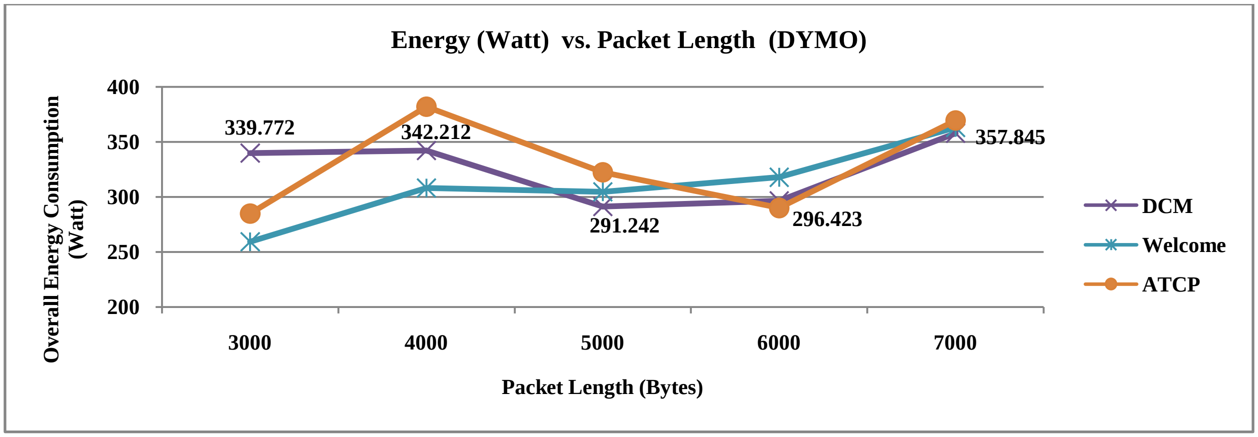


Figure 5-39: Overall Energy consumption (Watt) vs. P_L for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED, N_{SD} = (0-3) m/s and N_D = 80 nodes over DYMO.

In Figure 5-39, the overall energy consumption versus packet length over DYMO when N_{SD} = (0-3) m/s and N_D = 80 nodes. The overall E_C is decreasing when P_L increase from 3000 to 4000 bytes, then EC decrease as P_L increase from 4000 to 6000 bytes, then E_C increase again as P_L increase from 6000 to 7000 bytes. The overall E_C enhancement of DCM from TCP-WELCOME when N_{SD} = (0-3) m/s and N_D = 80 nodes is happen when P_L increase from 3000 to 4000 bytes as follows: 30.9% when P_L = 3000 bytes and 11% when P_L = 4000 bytes.

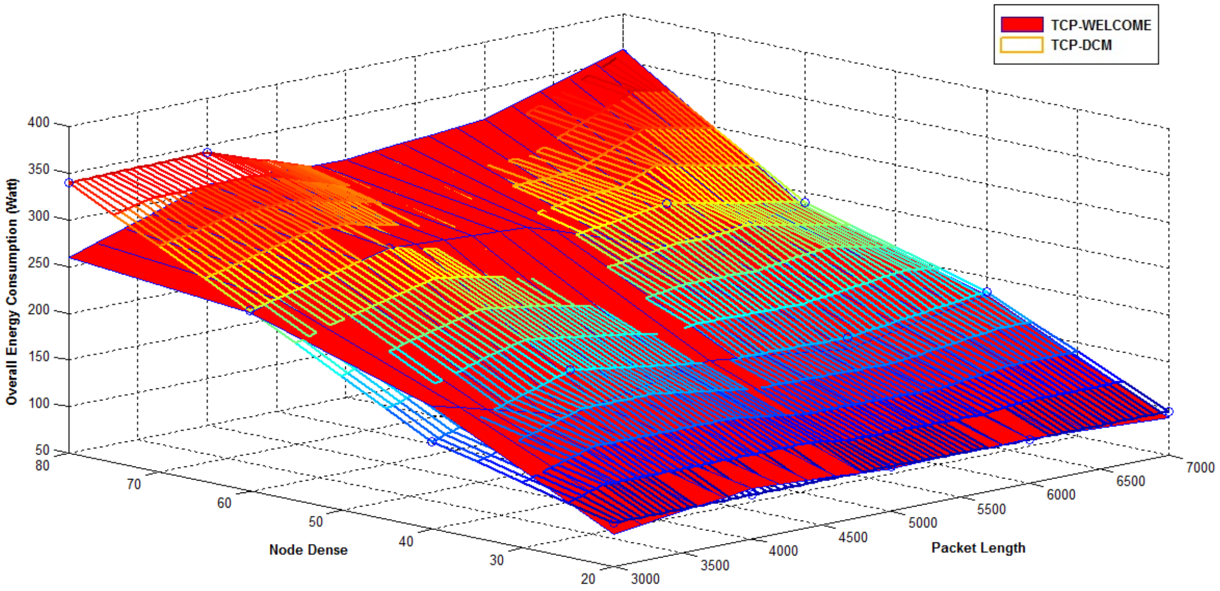


Figure 5-40: Overall Energy consumption (Watt) vs. N_D and P_L for TCP-DCM and TCP-WELCOME over DYMO routing protocol, with queue type RED and N_{SD} = (0-3) m/s.

In figure 5-40, a three dimension graph to represent the effect of varying N_D and P_L on overall E_C when N_{SD} = (0-3) m/s of our proposed DCM and TCP-WELCOME over DYMO routing protocol. As N_D increase, the overall E_C increase. The best overall energy consumption enhancement of DCM from TCP-WELCOME over DYMO is: 20.9% when packet length = 3000 bytes and node dense = 40 nodes. The next energy consumption enhancement is 9.6% when packet length = 5000 bytes and node dense = 60 nodes.

5.6 Comparison with other protocols

In this section I will compare our proposed DCM with other TCPs over two routing protocols: AOMDV and DYMO at low N_S from 0 to 3 m/s, in order to test the congestion technique implemented in each of them. The TCPs we will consider in this comparison are: TCP-Newreno, TCP-WESTWOOD and TCP-VEGAS.

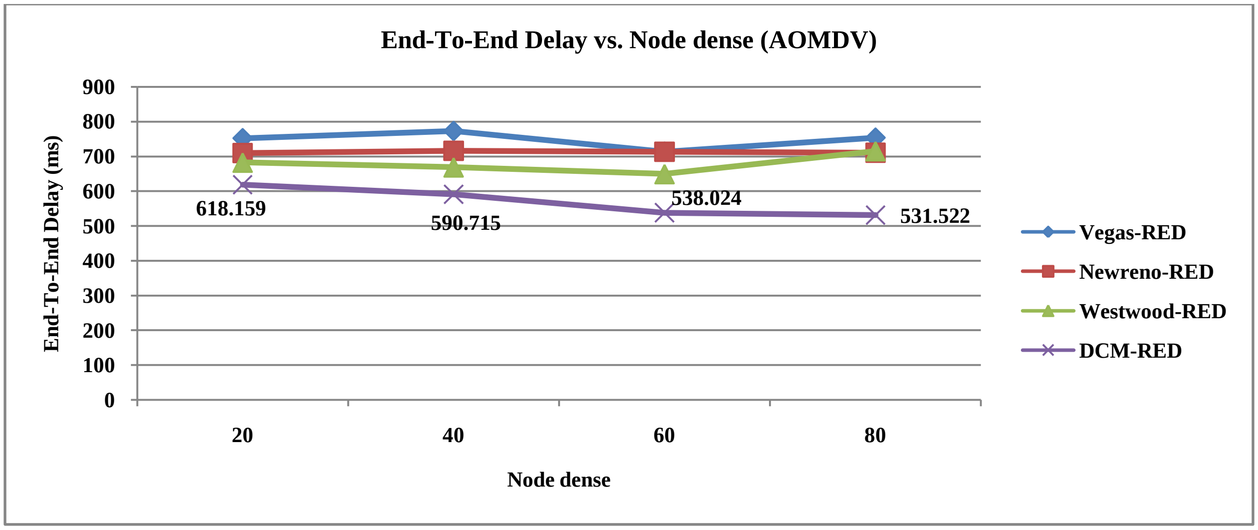


Figure 5-41: End-to-End Delay (ms) vs. N_D for various TCP's, with queue type RED, P_L = 5000 bytes and N_{SD} = (0-3) m/s over AOMDV routing protocol.

In Figure 5-41, the average end-to-end delay versus node dense over AOMDV routing protocol when N_{SD} = (0-3) m/s and P_L = 5000 bytes. As N_D increase, the average $E2E_D$ decreases slightly. The average E2ED of TCP-VEGAS is higher than the average E2ED of other TCPs, while the average E2ED of DCM is the lowest.

The best average $E2E_D$ enhancement of DCM in comparison with TCP-Newreno over AOMDV routing protocol when = 5000 bytes and N_{SD} = (0-3) m/s is 25.1% when N_D = 80 nodes, while the lowest average $E2E_D$ enhancement is 12.8% when N_D = 20 nodes. The best average $E2E_D$ enhancement of DCM over TCP-Vegas is 29.4% when N_D = 80 nodes, while the lowest $E2E_D$ enhancement is 17.8% when N_D = 20 nodes. The best average $E2E_D$ enhancement of DCM over TCP-Westwood is 25.6% when N_D = 80 nodes, while the lowest E2ED enhancement is 9.4% when N_D = 20 nodes.

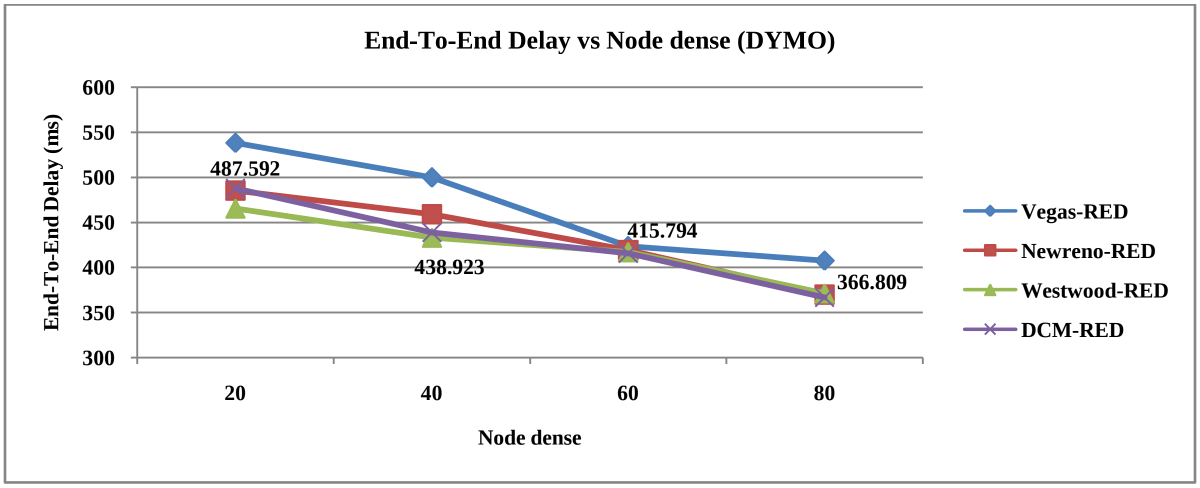


Figure 5-42: End-to-End Delay (ms) vs. N_D for various TCP's, with queue type RED, P_L = 5000 bytes and N_{SD} = (0-3) m/s over DYMO routing protocol.

In Figure 5-42, the $E2E_D$ value of our proposed DCM is decreasing as N_D increase over DYMO routing protocol when P_L = 5000 bytes and N_{SD} = (0-3) m/s. The decrease in the average $E2E_D$ value versus the increase in ND is due to distances between nodes that become shorter.

The best average $E2E_D$ enhancement of DCM over TCP-NewReno is 4.3% when N_D = 40 nodes, while the lowest $E2E_D$ enhancement is 0.82% when N_D = 80 nodes. The best average $E2E_D$ enhancement of DCM over TCP-Vegas is 12.2% when N_D = 40 nodes, while the lowest

$E2E_D$ enhancement is 1.7% when N_D =60 nodes. The best average $E2E_D$ enhancement of DCM over TCP-Westwood is 1.2% when N_D = 80 nodes, while the lowest $E2E_D$ enhancement is 0.3% when N_D = 60 nodes.

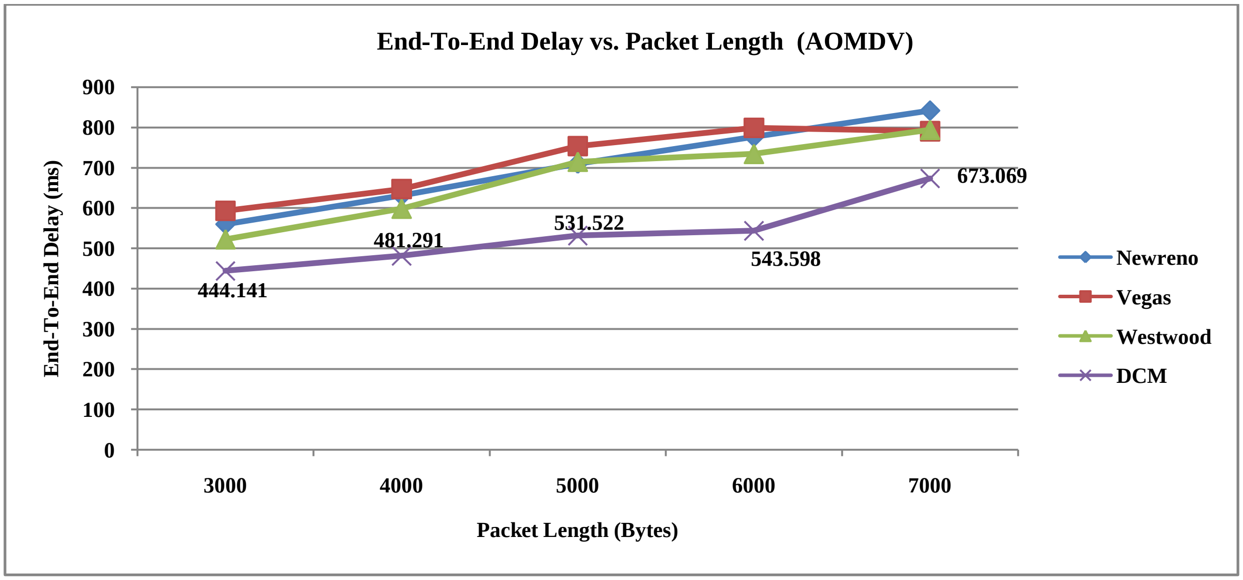


Figure 5-43: End-to-End Delay (ms) vs. P_L for various TCP's, with queue type RED, N_D = 80 nodes and N_{SD} = (0-3) m/s over AOMDV routing protocol.

In Figure 5-43, the average $E2E_D$ versus packet length over AOMDV routing protocol when N_D = 80 nodes and N_{SD} = (0-3) m/s. The $E2E_D$ of TCP-Newreno, TCP-Westwood, TCP-Vegas and DCM increase as P_L increase due to network congestion. TCP-Vegas have the highest average $E2E_D$ with increasing P_L, while our proposed DCM has the lowest.

The best average $E2E_D$ enhancement of DCM over TCP-NewReno is 30% when P_L = 6000 bytes, while the lowest $E2E_D$ enhancement is 20.6% when P_L = 3000 bytes. The best average $E2E_D$ enhancement of DCM over TCP-Vegas is 31.9% when P_L = 6000 bytes, while the lowest $E2E_D$ enhancement is 14.9% when P_L = 7000 bytes. The best average $E2E_D$ enhancement of DCM over TCP-Westwood is 25.9% when P_L = 6000 bytes, while the lowest $E2E_D$ enhancement is 14.9% when P_L = 3000 bytes.

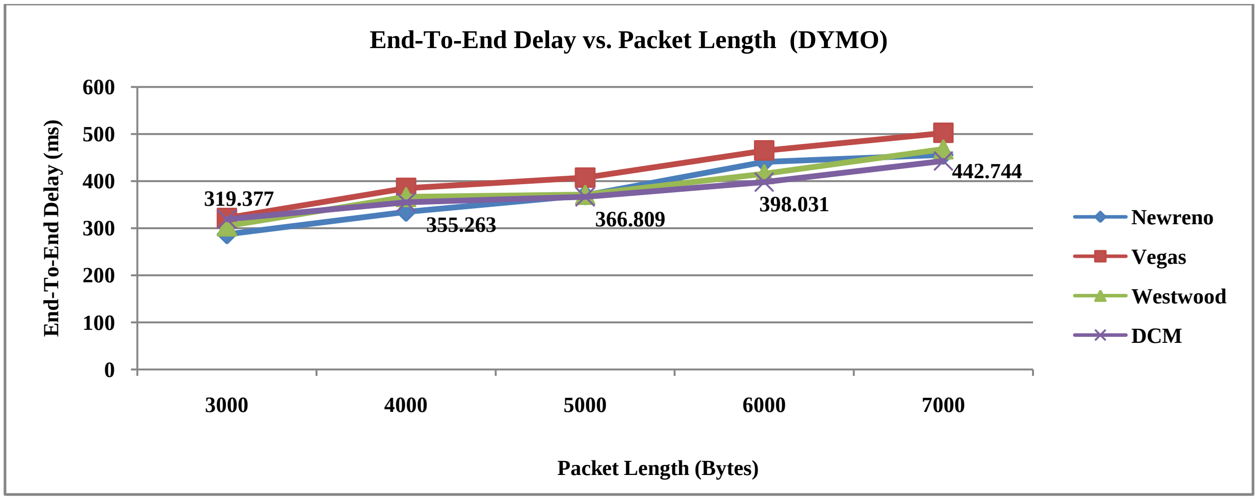


Figure 5-44: End-to-End Delay (ms) vs. P_L for various TCP's, with queue type RED, N_D = 80 nodes and N_{SD} = (0-3) m/s over DYMO routing protocol.

In Figure 5-44, the average end-to-end delay versus packet length over DYMO routing protocol when N_D = 80 nodes and N_{SD} = (0-3) m/s. All presented TCPs $E2E_D$ increase as P_L increase due to the problem of network congestion. TCP-Vegas have the highest average $E2E_D$ in all simulation scenarios; while DCM has the lowest as P_L increase more than 5000 bytes.

The best average $E2E_D$ enhancement of DCM over TCP-NewReno is 9.7% when P_L = 6000 bytes, while the lowest $E2E_D$ enhancement is 0.82% when P_L = 5000 bytes. The best average $E2E_D$ enhancement of DCM over TCP-Vegas is 14.4% when P_L = 6000 bytes, while the lowest $E2E_D$ enhancement is 0.8% when P_L = 4000 bytes. The best average $E2E_D$ enhancement of DCM over TCP-Westwood is 5.3% when P_L = 7000 bytes, while the lowest $E2E_D$ enhancement is 1.2% when P_L = 5000 bytes.

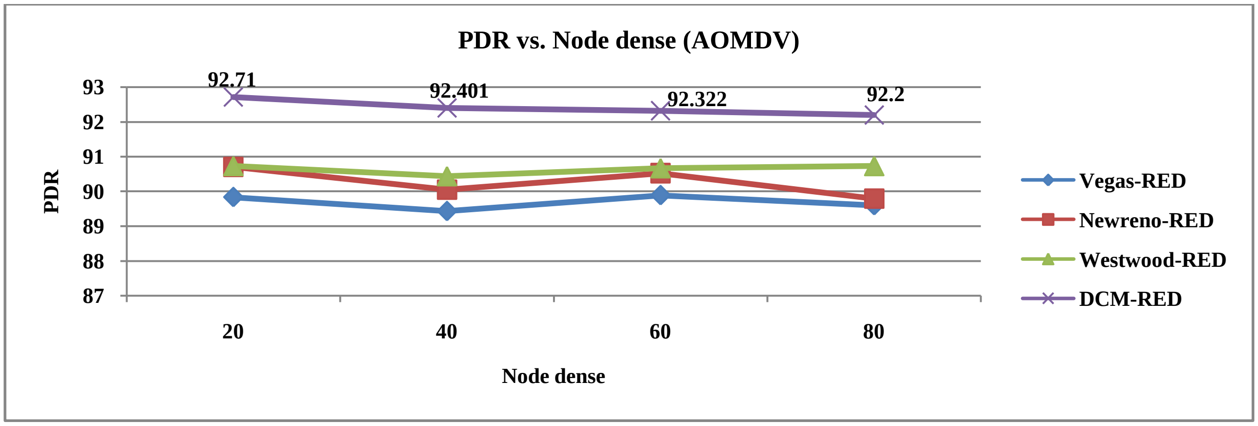


Figure 5-45: PDR vs. N_D for various TCP's, with queue type RED, P_L = 5000 bytes and N_{SD} = (0-3) m/s over AOMDV routing protocol.

In Figure 5-45, The PDR versus node dense over AOMDV routing protocol when P_L = 5000 bytes and N_{SD} = (0-3) m/s. As node dense increase, the PDR slightly decrease due to the increase in node interference. TCP-Vegas have the lowest PDR, while DCM have the highest.

The best PDR enhancement of DCM over TCP-NewReno is 2.6% when N_D = 80 nodes, while the lowest PDR enhancement is 1.9% when N_D = 60 nodes. The best PDR enhancement of DCM over TCP-Vegas is 3.3% when N_D = 40 nodes, while the lowest PDR enhancement is 2.7% when N_D = 60 nodes. The best PDR enhancement of DCM over TCP-Westwood is 2.2% when N_D = 20 nodes, while the lowest PDR enhancement is 1.6% when N_D = 80 nodes.

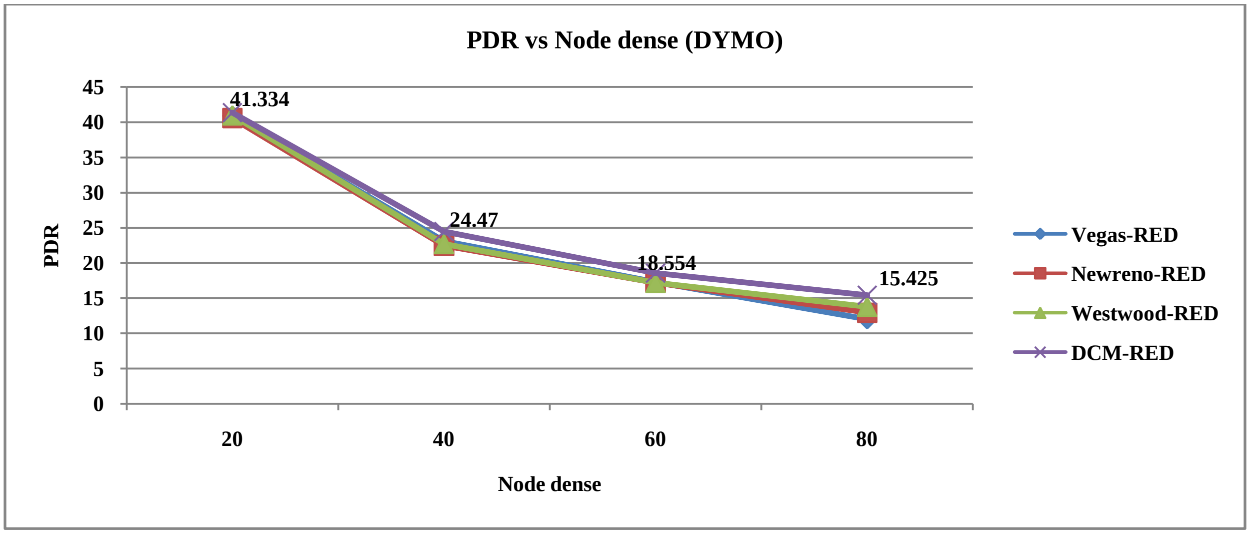


Figure 5-46: PDR vs. N_D for various TCP's, with queue type RED, P_L = 5000 bytes and N_{SD} = (0-3) m/s over DYMO routing protocol

In Figure 5-46, the value of PDR versus N_D over DYMO routing protocol when P_L = 5000 bytes and N_{SD} = (0-3) m/s. PDR decrease as node dense increase over DYMO routing protocol more than the decrease over AOMDV routing protocol due to the nature of DYMO that utilize the queues between nodes. The PDF of DCM is slightly better than other TCPs.

The best PDR enhancement of DCM over TCP-NewReno is 19.2% when N_D = 80 nodes, while the lowest PDR enhancement is 1.8% when N_D = 20 nodes. The best PDR enhancement of DCM over TCP-Vegas is 28.5% when N_D = 80 nodes, while the lowest PDR enhancement is 1% when N_D = 20 nodes. The best PDR enhancement of DCM over TCP-Westwood is 12.1% when N_D = 80 nodes, while the lowest PDR enhancement is 0.8% when N_D = 20 nodes.

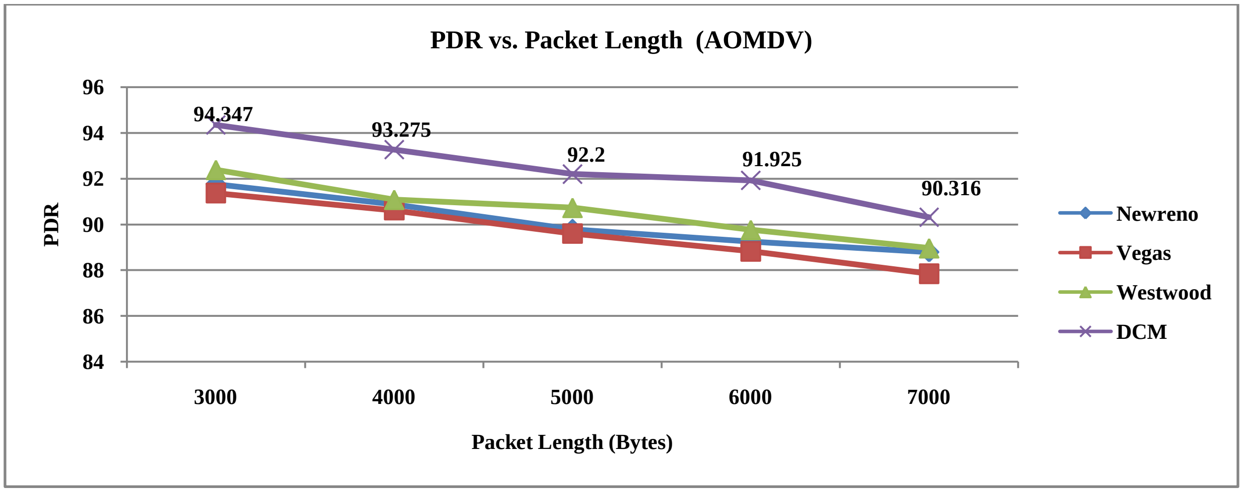


Figure 5-47: PDR vs. P_L for various TCP's, with queue type RED, N_D = 80 nodes and N_{SD} = (0-3) m/s over AOMDV routing protocol.

In Figure 5-47, the PDR versus packet length over AOMDV routing protocol when N_D = 80 and N_{SD} = (0-3) m/s. As P_L increase, the PDR decrease due to the problem of network congestion. TCP-Vegas have the lowest PDR, while DCM has the highest due to its dynamic technique that avoids the congested nodes.

The best PDR enhancement of DCM over TCP-NewReno is 3% when P_L = 6000 bytes, while the lowest PDR enhancement is 1.7% when P_L = 7000 bytes. The best PDR enhancement of DCM over TCP-Vegas is 3.4% when P_L = 6000 bytes, while the lowest PDR enhancement is 2.8% when P_L = 7000 bytes. The best PDR enhancement of DCM over TCP-Westwood is 2.4% when P_L = 4000 bytes, while the lowest PDR enhancement is 1.5% when P_L = 7000 bytes.

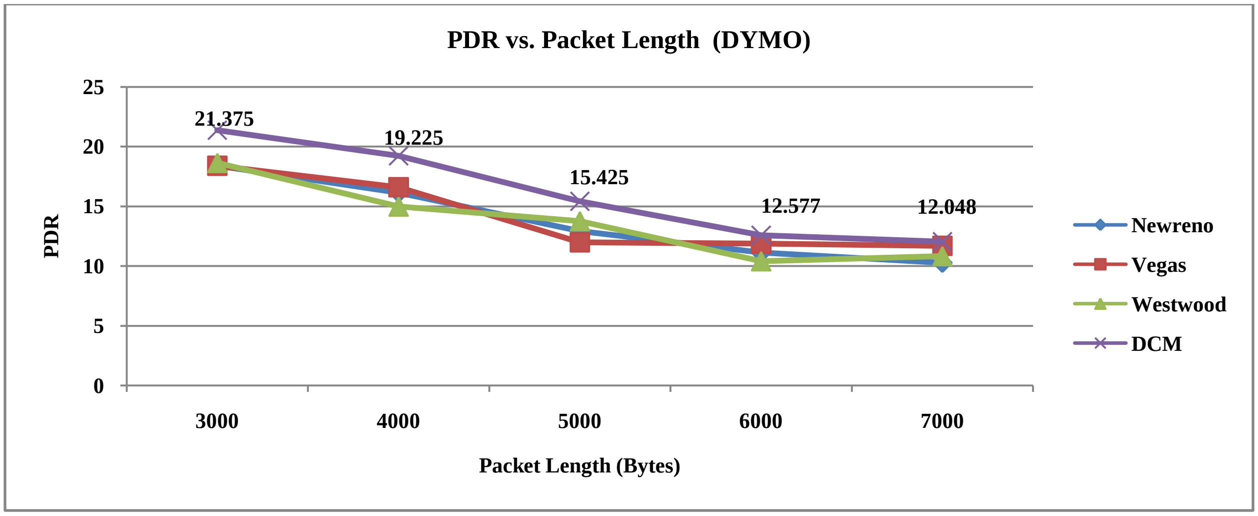


Figure 5-48: PDR vs. P_L for various TCP's, with queue type RED and N_{SD} = (0-3) m/s over DYMO routing protocol.

In Figure 5-48, the value of PDR is decreasing when the P_L increase in all TCPs over DYMO routing protocols. The PDR of proposed DCM is slightly better than other TCPs.

The best PDR enhancement of DCM over TCP-NewReno is 19.2% when P_L = 5000 bytes, while the lowest PDR enhancement is 13% when P_L = 6000 bytes. The best PDR enhancement of DCM over TCP-Vegas is 28.5% when P_L = 5000 bytes, while the lowest PDR enhancement is 2.9% when P_L = 7000 bytes. The best PDR enhancement of DCM over TCP-Westwood is 28.2% when P_L = 4000 bytes, while the lowest PDR enhancement is 11.3% when P_L = 7000 bytes.

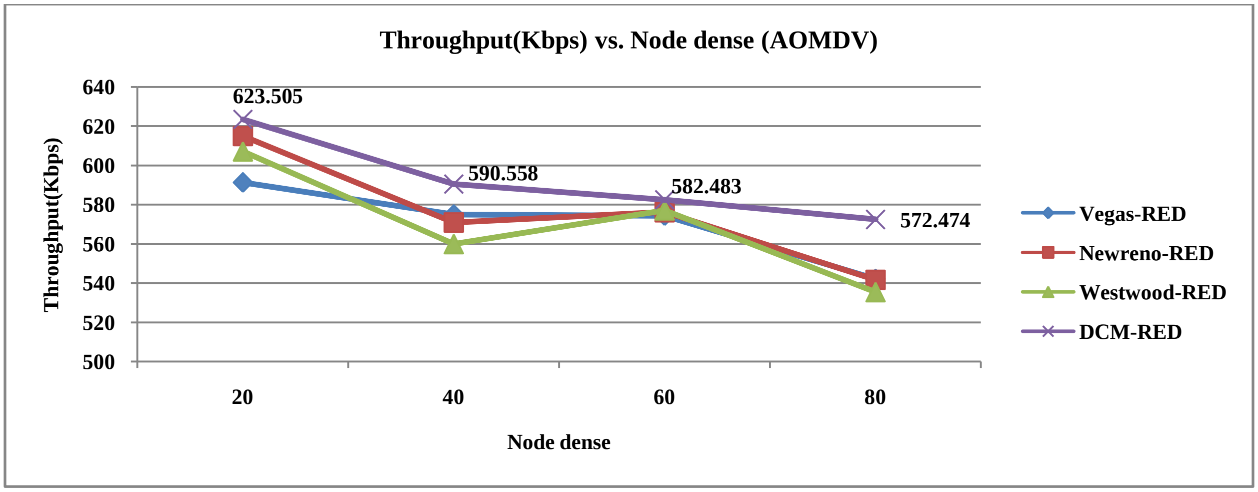


Figure 5-49: Throughput (Kbps) vs. N_D for various TCP's, with queue type RED, P_L = 5000 bytes and N_{SD} = (0-3) m/s over AOMDV routing protocol

In Figure 5-49, the throughput versus node dense over AOMDV routing protocol when P_L = 5000 bytes and N_{SD} = (0-3) m/s. As node dense increase, the throughput decreases due to the increase in the interference between nodes. The throughput of TCP-Westwood is the lowest, while the throughput of DCM is slightly better than other TCPs.

The best throughput enhancement of DCM over TCP-NewReno is 5.6% when N_D = 80 nodes, while the lowest throughput enhancement is 1% when N_D = 60 nodes. The best throughput enhancement of DCM over TCP-Vegas is 5.5% when N_D = 80 nodes, while the lowest throughput enhancement is 1.4% when N_D = 60 nodes. The best throughput enhancement of DCM over TCP-Westwood is 6.9% when N_D = 80 nodes, while the lowest throughput enhancement is 0.93% when N_D = 60 nodes.

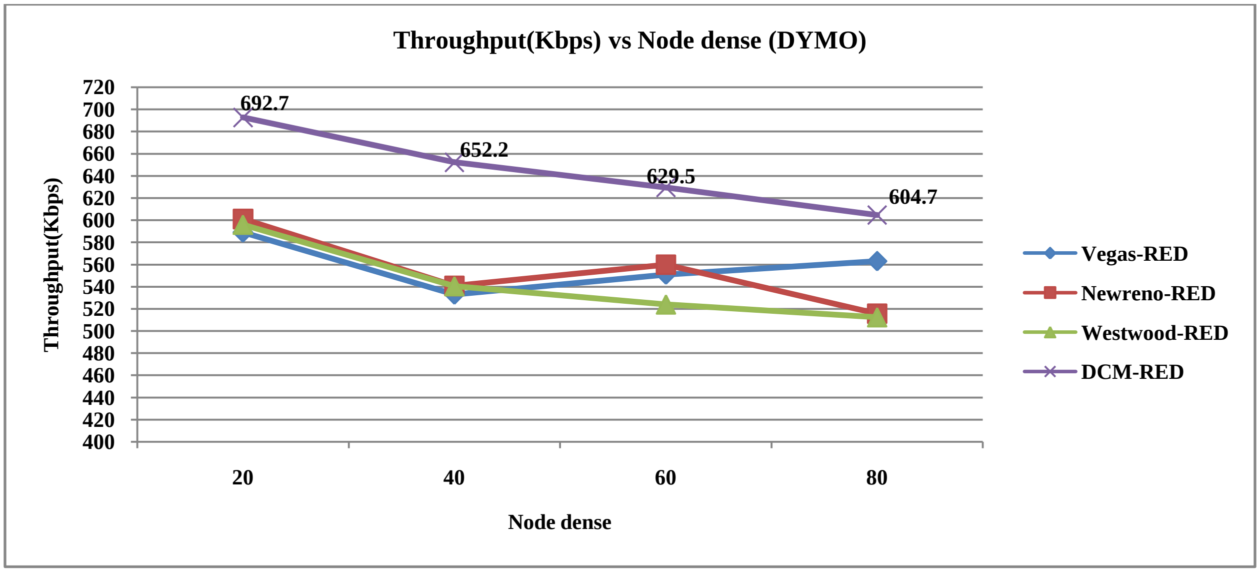


Figure 5-50: Throughput (Kbps) vs. N_D for various TCP's, with queue type RED, P_L = 5000 bytes and N_{SD} = (0-3) m/s over DYMO routing protocol

In Figure 5-49, the value throughput versus N_D over DYMO routing protocol when P_L = 5000 bytes and N_{SD} = (0-3) m/s. The increase in node dense, cause a decrease in throughput in all presented TCPs due to node interference.

The best enhancement of throughput achieved by DCM over TCP-NewReno is 20.6% when N_D = 40 nodes, while the lowest throughput enhancement is 12.4% when N_D = 60 nodes.

The best throughput enhancement of DCM over TCP-Vegas is 22.3% when N_D = 40 nodes, while the lowest throughput enhancement is 7.3% when N_D = 80 nodes. The best throughput enhancement of DCM over TCP-Westwood is 20.6% when N_D = 40 nodes, while the lowest throughput enhancement is 16.2% when N_D = 20 nodes.

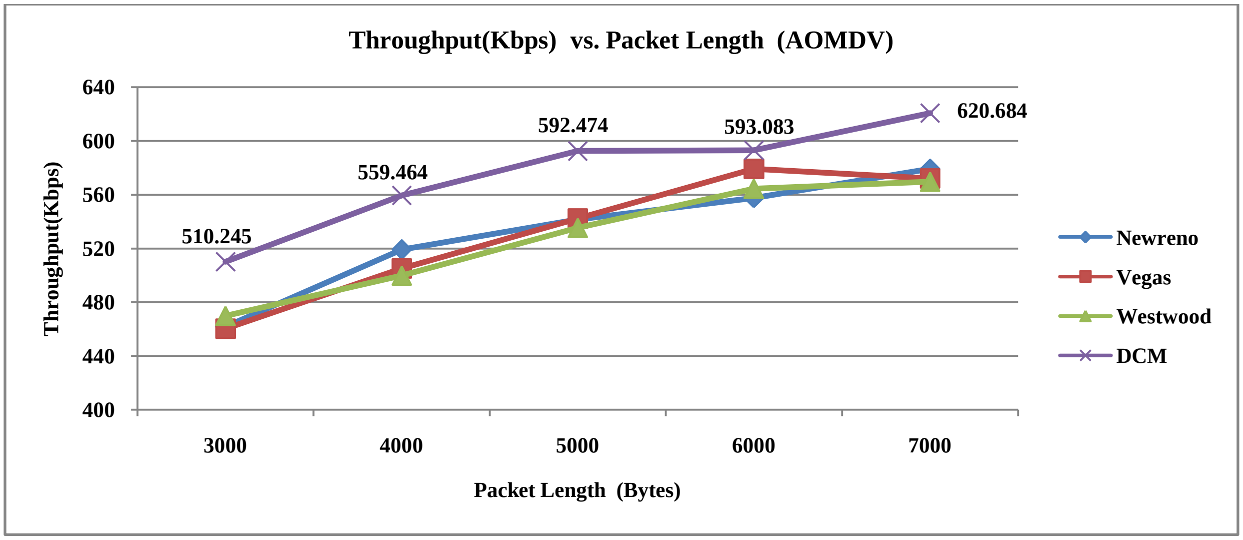


Figure 5-51: Throughput (Kbps) vs. P_L for various TCP's, with queue type RED, N_D = 80 nodes and N_{SD} = (0-3) m/s over AOMDV routing protocol.

In Figure 5-51, the throughput versus packet length AOMDV routing protocol when N_D = 80 nodes and N_{SD} = (0-3) m/s. As packet length increase, the overall throughput increase in all TCPs but the problem of network congestion increase. The throughput of TCP-Westwood is the lowest, while the throughput in our proposed DCM is higher than TCP-Westwood, TCP-NewReno and TCP-Vegas due to its dynamic mechanism that not only avoids the congested node, but also controls the congestion.

The best throughput enhancement of DCM over TCP-NewReno is 6.1% when P_L = 3000 bytes, while the lowest throughput enhancement is 2.8% when P_L = 6000 bytes. The best throughput enhancement of DCM over TCP-Vegas is 6.8% when P_L = 4000 bytes, while the lowest throughput enhancement is 1% when P_L = 6000 bytes. The best throughput enhancement of DCM over TCP-Westwood is 7.9% when P_L = 4000 bytes, while the lowest throughput enhancement is 1.5% when P_L = 6000 bytes.

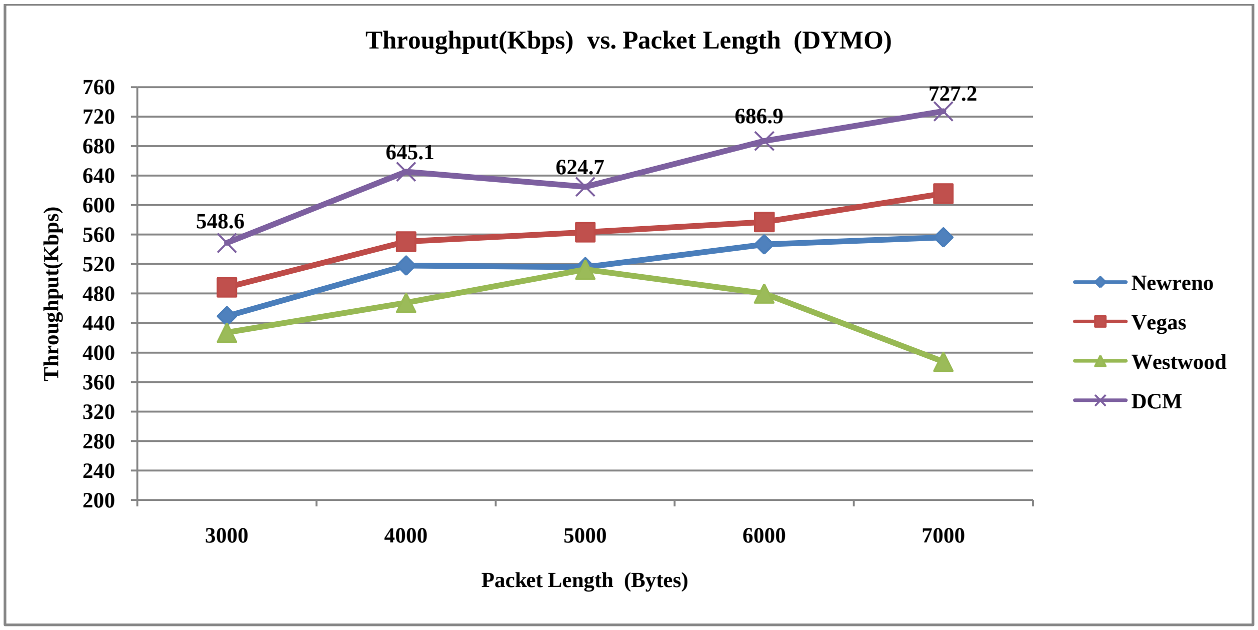


Figure 5-52: Throughput (Kbps) vs. P_L for various TCP's, with queue type RED, N_D = 80 nodes and N_{SD} = (0-3) m/s over DYMO routing protocol

In Figure 5-52, the throughput is increasing as packet length increase over DYMO routing protocol when N_D = 80 nodes and N_{SD} = (0-3) m/s. TCP-Westwood has the lowest throughput over other TCPs, while our proposed DCM has the best throughput over TCP-Westwood, TCP-NewReno and TCP-Vegas.

The best enhancement of throughput achieved by DCM over TCP-NewReno is 27.1% when P_L = 7000 bytes, while the lowest throughput enhancement is 17.2% when P_L = 5000 bytes. The best enhancement of throughput achieved by DCM over TCP-Vegas is 15.6% when

P_L = 6000 bytes, while the lowest throughput enhancement is 7.3% when P_L = 5000 bytes. The best enhancement of throughput achieved by DCM over TCP-Westwood is 82.4% when P_L = 7000 bytes, while the lowest throughput enhancement is 17.9% when P_L = 5000 bytes.

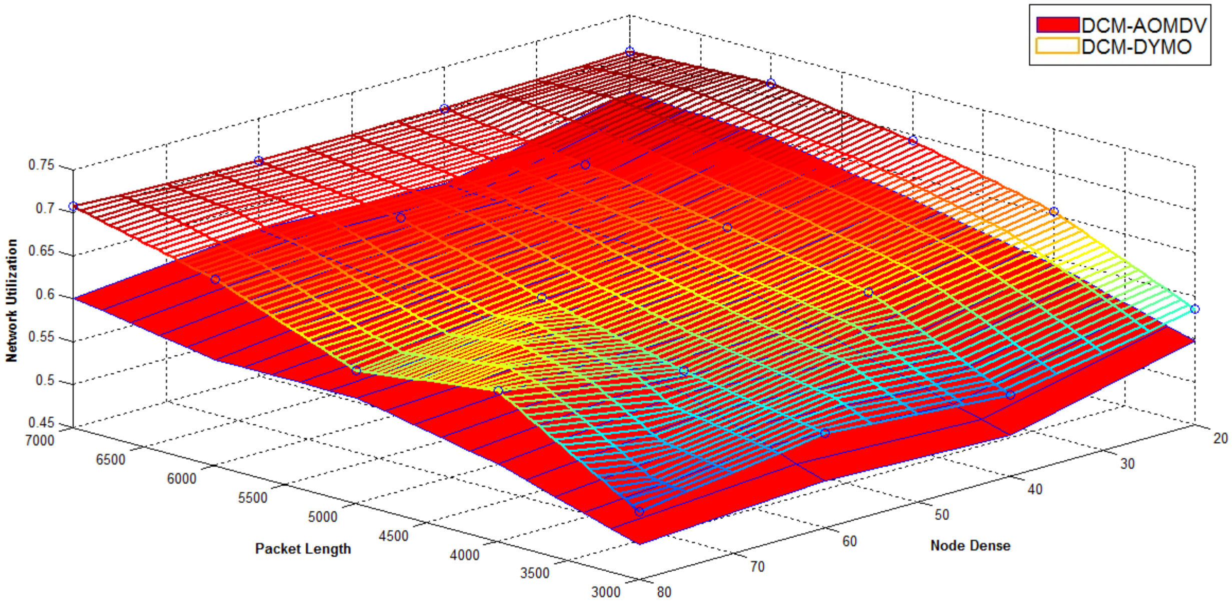


Figure 5-53: Network utilization vs. P_L and N_D for DCM, with queue type RED and N_{SD} = (0-3) m/s over AOMDV and DYMO routing protocol

In Figure 5-53, a three dimensional graph to represent the network utilization versus both node dense and packet length over both AOMDV and DYMO routing protocols when N_{SD} = (0-3) m/s. As PL increase, the network utilization increase. In addition, the network utilization increases also when as ND increase.

The best overall network utilization over AOMDV is 0.66 when P_L = 7000 bytes and N_D = 20 nodes, while the lowest network utilization is 0.49 when P_L = 3000 bytes and N_D = 80 nodes. The best overall utilization over DYMO is 0.71 when P_L = 6000 bytes and N_D = 20 nodes, while the lowest network utilization is 0.54 when P_L = 3000 bytes and N_D = 40 nodes.

5.7 Summary

In this chapter we present the simulation results and analysis based on five performance metrics: Throughput, $E2E_D$, PDR, Normalized Overhead and total consumed energy. Throughput and PDR of our proposed DCM shows the highest value over both AOMDV and DYMO routing protocols when changing P_L and N_D.

The normalized overhead metric shows slightly higher value than other TCPs. This is due to the implementation of 20 TCP connections between several sending nodes and receiving nodes, also the extra control packets to maintain the multi-paths of each TCP connection.

In addition the value of $E2E_D$ of DCM is the lowest. The dynamic congestion technique that is implemented in our model tries to select the minimum congested path over available routes from source to destination, which decrease the value of $E2E_D$. In comparison with other congestion techniques used by TCP-Newreno, TCP-Vegas, TCP-WESTWOOD, TCP-WELCOME and ATCP, the technique used by our proposed DCM shows the best results in congested network.

Chapter Six

Conclusion and Future Works

Contents

Conclusion and Future Works

6.1 Conclusion

This thesis presents the current research on solving TCP congestion problems over MANET by presenting most used TCP variants that preserve end-to-end semantic and there analysis to increase performance of TCP over MANET. As in case of mobile networks, performance of TCP degrades because of its inability to handle efficiently packet losses due to congestion. We have placed special emphasis on TCP-WELCOME, because it is the most successful TCP variant over MANET, due to its ability to differentiate between types of packet losses in MANET.

This thesis proposed the design and implementation of a new dynamic mechanism to replace traditional congestion algorithm of TCP-NEWRENO used in TCP-WELCOME with dynamic minimum congestion path selection based on the measured value of RTT. This new dynamic congestion model (DCM) has two phases: the initialization phase and the running phase. In the initialization phase, the minimum three congested paths between the source and the destination are selected and transmitted to the source node. In the running phase, if the congestion in the selected path exceed the congestion limit, then this path will be replaced with another lower congested path , otherwise the *CWND* will be controlled in more dynamic mechanism.

We validated the developed model by comparing with the most used congestion control techniques in TCP-NEWRENO, TCP-VEGAS, and TCP-WESTOOD at normal speed of walk over two most modern routing protocols: AOMDV and DYMO. We measured five performance metrics: Throughput, Packet Delivery Ratio, End-to-End Delay, Normalized Overhead and the total energy consumed by varying packet length and node dense. In addition to that we validate our proposed model at various node speeds in comparison with TCP-WELCOME and ATCP.

With reference to data analysis and the experimental results, it shows that, our proposed DCM handles packet losses due to network congestion problem in more efficient way than other TCP's does. DCM improves the overall throughput and PDR versus the increase in packet length and node dense.

The lowest enhancement of overall throughput of DCM from TCP-WELCOME over DYMO routing protocol is 6.7%, is when N_D = 80 nodes and P_L = 5000 bytes. The best overall enhancement on throughput of DCM from TCP-WELCOME over DYMO routing protocol is 20.2%, is when P_L = 5000 bytes and N_D = 20 nodes.

The best overall enhancement on throughput in all scenarios of DCM from TCP-WELCOME over AOMDV routing protocol is 7.6%, is when P_L = 7000 bytes and N_D = 20 nodes.

The best overall PDR of DCM from TCP-WELCOME over AOMDV is 1.6% when P_L = 6000 bytes and N_D = 80 nodes.

The best and lowest enhancement of PDR of DCM from TCP-WELCOME over DYMO is as follows: 27.4% when P_L = 4000 bytes and N_D = 80 nodes, 9.1% when P_L = 7000 bytes and N_D = 60 nodes. The best overall network utilization over AOMDV is 0.66 when P_L = 7000 bytes and N_D = 20 nodes, while the best overall utilization over DYMO is 0.71 when P_L = 6000 bytes and N_D = 20 nodes.

The normalized overhead of DCM is slightly higher than other TCPs due to the extra control packets to maintain the multi-path routes between several senders and several receivers. In addition, DCM shows a decrease in the average end-to-end delay and hence it increase TCP performance over MANET.

The best $E2E_D$ enhancement of DCM from TCP-WELCOME over AOMDV routing protocol at N_{SD} = (0-3) m/s is 29.9% when P_L = 7000 bytes and N_D = 40 nodes. The best overall average $E2E_D$ enhancement of DCM from TCP-WELCOME over DYMO routing protocol when N_{SD} = (0-3) m/s is 35.4% when P_L = 7000 bytes and N_D = 20 nodes.

6.2 Future Work

Distributed Denial of Service (DDoS) attack is one of the most challenging security issues over Wireless Ad Hoc Networks that deprive all legitimate flows from a fair share of bandwidth by overwhelming the buffer space of network resources. The attack process is performed by controlling many of hosts called "zombies" to attack a single victim by planting a zombie program on these machines. With lots of zombie hosts cooperation, the size of an attack can be damaging. The great demand for security, place particular emphasis on the detection and prevention approaches.

TCP/IP protocol has numerous weaknesses over Mobile Ad Hoc Networks (MANETs) environment such as incorrectly triggering congestion avoidance technique to handle the DDoS attack. Attacker with malicious objectives exploits these shortcomings to overflow the victim resources with large amount of traffic in order to prevent victim from accessing normal traffic.

My future work will focus on DDoS attack that exploit the weaknesses in Transmission Control Protocol (TCP) over Mobile Ad Hoc networks (MANETs), developing an effective and creative solution to minimize the effect of DDoS attack is an important challenge, because the availability of DDoS attacking tools makes it possible to launch an attack.

References

[1] Kumar, K.D. , Ramya, I. , Masillamani, M.R., “Queue Management in Mobile Adhoc Networks (Manets)”, Green Computing nd Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom), Hangzhou, 18-20 Dec. 2010.

[2] Chunmei Liu,Eytan Modiano, "On the performance of additive increase multiplicative decrease (AIMD) protocols in hybrid space-terrestrial networks", Massachusetts Institute of Technology, Cambridge, MA 02139, United States, 18 September 2004.

[3] Sally Floyd, "TCP and Explicit Congestion Notification," ACM Computer Communication Review, 1994.

[4] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transaction on Networking, 1993, Page(s):397-413.

[5] Kai Chen, Yuan Xue, Nahrstedt K,”On Setting TCP Congestion Window Limit in Mobile Ad Hoc Networks” ICC '03.IEEE International Conference, 2003, Page(s): 1080-1084.

[6] Kim K W, Lorenz P, lee MMO, “A new tuning maximum congestion window for improving TCP performance in MANET”, Systems Communications, Proceedings in IEEE Confrences, 2005.

[7] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, “The impact of multi hop wireless channel on TCP throughput and loss,” IEEE INFOCOM, San Francisco, CA, March 2003.

[8] S. Xu, T. Saadawi, “Does the IEEE 802.11 MAC protocol work well in multi hop wireless ad hoc networks?” IEEE Communications Magazine, June 2001, Page(s): 130–137.

[9] Z. Fu, X. Meng, S. Lu, “How bad TCP can perform in mobile ad-hoc networks” IEEE ICNP ’02, Paris, France, 2002.

[10] Marco Mancuso, "A Novel Scheme of Loss Differentiation and Adaptive Segmentation to Enhance TCP Performance over Wireless Networks, " Politecnico di Milano, Dept. of Electronics and Information , P.zza L. Da Vinci 32, 20133 Milano, ITALY , 2002.

[11] L. Stephane, Y. Ghamri Doudane, G. Pujolle, "CrossLayer Loss Differentiation Algorithms to improve TCP Performances in WLANs," International Conference on Personal Wireless Communications (PWC'06), September 2006.

[12] Yao-Nan Lien, "Hop-by-Hop TCP for Sensor Networks", International Journal of Computer Networks & Communications (IJCNC), Vol.1, No.1, April 2009, Page(s):1-16.

[13] Ismail, D., Johor ; Jaafar, M., “Mobile ad hoc network overview”, Applied Electromagnetics, 2007. APACE 2007. Asia-Pacific Conference, Melaka, 4-6 Dec. 2007.

[14] Joe Hoffert, Douglas C. Schmidt, Aniruddha Gokhale, "Evaluating Transport Protocols for Real-Time Event Stream Processing Middleware and Applications", On the Move to Meaningful Internet Systems: OTM 2009, pp 614-633.

[15] Noreen Imran, Samia Riaz, Asma Shaheen, Muhammad Sharif, Mudassar Raza, "COMPARATIVE ANALYSIS OF LINK STATE AND DISTANCE VECTOR ROUTING PROTOCOLS FOR MOBILE ADHOC NETWORKS", Sci. Int. (Lahore), 2014, Pages:669-674.

[16] Mahesh K. Marina1, Samir R. Das, "Ad hoc on-demand multipath distance vector routing", WIRELESS COMMUNICATIONS AND MOBILE COMPUTING, Published online in Wiley InterScience, 2006.

[17] B. N. Jagdale, Pragati Patil, P. Lahane, D. Javale, "Analysis and Comparison of Distance Vector, DSDV and AODV Protocol of MANET", International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.2, March 2012, Pages:121-131.

[18] Dahai Xu, Jennifer Rexford, "Link-State Routing With Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 19, NO. 6, DECEMBER 2011, Pages: 1717-1730.

[19] Liu Xiao-Yan, Chen Yan-Li, "Application of Dijkstra Algorithm in Logistics Distribution Lines", Proceedings of the Third International Symposium on Computer Science and Computational Technology (ISCSCT ’10) Jiaozuo, P. R. China, 14-15, August 2010, pages:48-50.

[20] Aart, S. S. Tyagi, "Study of MANET: Characteristics, Challenges, Application and Security Attacks", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013, Pages:252-257.

[21] Jyoti Jain, Mehajabeen Fatima, Roopam Gupta, K.Bandhopadhyay, "OVERVIEW AND CHALLENGES OF ROUTING PROTOCOL AND MAC LAYER IN MOBILE AD-HOC NETWORK", Journal of Theoretical and Applied Information Technology, 2009, Pages:6-12.

[22] Kong Ling Pang, Yang Qin, "The performance comparison of flat routing and hierarchical routing in ad hoc wireless networks in the presence of faulty nodes", Mobility '07 Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology, ACM New York, NY, USA, 2007 ,Pages 139-144.

[23] Debajit Sensarma, Koushik Majumder, "A NOVEL HIERARCHICAL ANT BASED QOS AWARE INTELLIGENT ROUTING SCHEME FOR MANETS", International Journal of Computer Networks & Communications (IJCNC) Vol.5, No.6, November 2013, Pages:215-229.

[24] Anuj K. Gupta, Harsh Sadawarti, & Anil k. Verma, (2011) "Review of various Routing Protocols for MANETs," International Journal of Information and Electrical Engineering, Vol. 1 No. 3, Pages:251-259.

[25] Ian D. Chakeres, Elizabeth M. Belding-Royer"AODV Routing Protocol Implementation Design", ICDCSW '04 Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04) - Volume 7, Pages 698-703.

[26] Anuj K. Gupta, Harsh Sadawarti2 and Anil K. Verma, "IMPLEMENTATION OF DYMO ROUTING PROTOCOL ", International Journal of Information Technology, Modeling and Computing (IJITMC) Vol.1, No.2, May 2013, page(s):49-57.

[27] Sukant Kishoro Bisoyi, Sarita Sahu, "Performance analysis of Dynamic MANET Ondemand (DYMO) Routing protocol", Special Issue of IJCCT Vol.1 Issue 2, 3, 4; 2010 for International Conference [ACCTA-2010], 3-5 August 2010.

[28] Anne Marie Hegland, "Towards reliable network services in Ad Hoc Networks: Protecting the routing protocols", PhD Dissertation Department of Informatics,Faculty of Mathematics and Natural Sciences, University of Oslo.

[29] I. Chackeres, and C. Perkins, "Dynamic MANET On-demand (DYMO) Routing," *IETF Internet-Draft,* October, 2006.

[30] P.Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot, "Optimized Link State Routing Protocol for Ad Hoc Networks," Proceedings IEEE INMIC, 2001, Pages:62-68.

[31] Manjeet Gupta, Sonam Kaushik, "Performance Comparison Study of AODV, OLSR and TORA Routing Protocols for MANETS", /International Journal Of Computational Engineering Research (IJCER), Vol. 2, Issue No.3, May-June 2012, pages: 704-711.

[32] Mahesh K. Marina1 and Samir R. Das, Ad hoc on-demand multipath distance vector routing, Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.432, WIRELESS COMMUNICATIONS AND MOBILE COMPUTING,Wirel. Commun. Mob. Comput. 2006.

[33] Mina Vajed Khiavi, Shahram Jamali, Performance Comparison of AODV and AOMDV Routing Protocols in Mobile Ad Hoc Networks, International Research Journal of Applied and Basic Sciences, Vol, 4 (11), 2013, Page(s):3277-3285.

[34] M.Chandrika M.tech student, N.Papanna, Performance comparison of AOMDV and POR Routing Protocols in MANETS, International Journal of Engineering Research and Applications (IJERA), Vol. 3, Issue 4, Jul-Aug 2013, pp.37-41.

[35] V. D. Park, M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", Proceedings of INFOCOMM 97, April 1997.

[36] Rakesh Kumar Jha, Suresh V. Limkar, Upena D. Dalal, "A Performance Comparison of Routing Protocols(DSR and TORA) for Security Issue In MANET(Mobile Ad Hoc

Networks)", IJCA Special Issue on “Mobile Ad-hoc Networks”, MANETs, 2010, Pages: 78-83.

[37] Dinesh Dharmaraju, "Routing and Quality of Service in Mobile AdHoc Networks with TORA/INORA", MASTER'S THESIS, CSHCN MS 2002-1, University of maryland.

[38] C. Perkins and P. Bhagwat. “Highly dynamic Destination-sequenced distance vector routing (DSDV) for mobile computers” ACM SIGCOMM '94, 1994, Pages: 234-244.

[39] Md. Nazmul Islam Khan Rashed Ahmed, “Simulation Based Performance Evaluation of Routing Protocols and TCP Variants in Mobile Ad-hoc Networks”, Master Thesis Electrical Engineering, Thesis no: MEE 10:86, November 2010.

[40] Akshai Aggarwal, Savita Gandhi, Nirbhay Chaubey, "PERFORMANCE ANALYSIS OF AODV,DSDV AND DSR IN MANETS", International Journal of Distributed and Parallel Systems (IJDPS) Vol.2, No.6, November 2011, Pages:167-177.

[41] Ruchi Aggarwal, Amanpreet Kaur, "Performance Analysis of Aodv, Dsdv, Dsr and Lar Routing Protocols In Manet- A Review", IOSR Journal of Computer Engineering (IOSR-JCE), 7Volume 16, Issue 3, Ver. VIII (May-Jun. 2014), pages:36-42.

[42] Sushil Kumar, Dinesh Singh, Mridul Chawla, "Performance Comparison of Routing Protocols in MANET Varying Network Size", International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN) ISSN No. 2248-9738 (Print) Volume-1, Issue-2, 2011, Pages;51-54.

[43] Gagangeet Singh Aujla, Sandeep Singh Kang, "Comprehensive Evaluation of AODV, DSR, GRP, OLSR and TORA Routing Protocols with varying number of nodes and traffic applications over MANETs", IOSR Journal of Computer Engineering (IOSR-JCE), Volume 9, Issue 3 (Mar.-Apr.2013), Pages:54-61.

[44] Davesh Singh Som, Dhananjaya Singh, "Performance Analysis and Simulation of AODV,DSR and TORA Routing Protocols in MANETs", International Journal of Recent Technology and Engineering (IJRTE), Volume-1, Issue-3, August 2012, pages;122-127.

[45] Chiu, D. and R. Jain, Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. J. Comput. Networks ISDN, 1989, pages:1-14.

[46] Jeroen Hoebeke, Ingrid Moerman, Bart Dhoedt, Piet Demeester, "An Overview of Mobile Ad Hoc Networks: Applications and Challenges", Journal of the Communications Network, Vol. 3 (July 2004), Pages:60-66.

[47] JENNIFER J.-N. LIU, IMRICH CHLAMTAC, "MOBILE AD HOC NETWORKING WITH A VIEW OF 4G WIRELESS: IMPERATIVES AND CHALLENGES", Institute of Electrical and Electronics Engineers, Inc., Edited by Basagni, Conti, Giordano, and Stojmenovic., 2004.

[48] Pravin Ghosekar, Girish Katkar, Dr. Pradip Ghorpade, "Mobile Ad Hoc Networking: Imperatives and Challenges", IJCA Special Issue on Mobile Ad-hoc Networks MANETs, 2010, Pages:153-158.

[49] Hayder Natiq Jasem, Zuriati Ahmad Zukarnain, Mohamed Othman, Shamala Subramaniam, "Experimental Evaluation of Bottleneck Link Utilization with New-Additive Increase Multiplicative Decrease Congestion Avoidance and Control Algorithm ", Journal of Computer Science 5 (12): 1058-1062, 2009.

[50] R.J.La,P.Ranjan, E. H. Abed, "Analysis of Adaptive Random Early Detection (ARED)", Department of Electrical and Computer Engineering University of Maryland, College Park, MD, 20742, USA.

[51] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking (TON), Volume 1 Issue 4, Aug. 1993, Pages 397-413.

[52] Deepak Agrawal, Fabrizio Granelli, "Redesigning an active queue management system", Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE Volume 2, 29 Nov.-3 Dec. 2004, Page(s):702-706.

[53] Ahmad T. Al-Hammouri, Vincenzo Liberatore, Michael S. Branicky, and Stephen M. Phillips. Parameterizing PI congestion controllers. In International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID 06), Vancouver, Canada, April 2006.

[54] Swastik Brahma and Mainak Chatterjee, “Congestion control and fairness in wireless sensor networks”, Pervasive Computing and Communications Workshops (PERCOM), 2010 8th IEEE International Conference, March 29 2010-April 2 2010, Pp. 413-418.

[55] Puneet Mehra, Avideh Zakhor, Christophe De Vleeschouwer, “Receiver-Driven Bandwidth Sharing for TCP”, INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications., San Francisco, CA, March 30 2003-April 3 2003.

[56] Ismail, D., Johor ; Jaafar, M., “Mobile ad hoc network overview”, Applied Electromagnetics, 2007. APACE 2007. Asia-Pacific Conference, Melaka, 4-6 Dec. 2007.

[57] Hai Jiang , Shixin Cheng , Xiangning Chen, “TCP Reno and Vegas performance in wireless ad hoc networks”, Communications, 2001. ICC 2001. IEEE International Conference, Helsinki, 11-14 Jun 2001.

[58] Li Huang; Sch. of Comput. Sci. & Technol., Soochow Univ., Suzhou, China ; Jian-de Lu, “Hops adaptive algorithm for TCP vegas in mobile ad hoc network”, Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference, Jilin, 19-22 Aug. 2011.

[59] Grieco, L.A. ; Dipt. di Elettrotecnica ed Elettronica, Politecnico di Bari, Italy ; Mascolo, S., “End-to-end bandwidth estimation algorithms for Westwood TCP congestion control”, IEEE Information Technology Interfaces (ITI) 2003. Proceedings of the 25th International Conference, 16-19 June 2003.

[60] Jian Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks", Selected Areas in Communications, IEEE Journal on (Volume:19 , Issue: 7), Jul 2001, Page(s):1300-1315.

[61] A. Seddik-Ghaleb, Y. Ghamri-Doudane, and S. M. Senouci, "TCP WELCOME TCP Variant for Wireless Environment, Link losses, and Congestion packet loss ModEls," in First International Communication Systems and Networks and Workshops, COMSNETS 2009.

[62] Sujata V. Mallapur, Siddarama. R. Patil, "Survey on Simulation Tools for Mobile Ad-Hoc Networks", IRACST – International Journal of Computer Networks and Wireless Communications (IJCNWC), Vol.2, No.2, April 2012.

[63] [NS-2 network simulator, URL: http://www.nsnam.org/.

[64] G.Jose Moses, D.Sunil Kumar, Prof.P.Suresh Varma, N.Supriya, "A Simulation Based Study of AODV, DSR, DSDV Routing Protocols in MANET Using NS-2", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 3, March 2012.

[65] Pankaj Rohal, Ruchika Dahiya, Prashant Dahiya, "Study and Analysis of Throughput, Delay and Packet Delivery Ratio in MANET for Topology Based Routing Protocols (AODV, DSR and DSDV)", INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY (IJARET), Vol. 1, Issue II, Mar.2013.

[66] A.Boomarani Malany, V.R.Sarma Dhulipala, RM.Chandrasekaran, "Throughput and Delay Comparison of MANET Routing Protocols", Int. J. Open Problems Compt. Math., Vol. 2, No. 3, September 2009.

[67] Moslem Amiri, Wireless sensor networks: Evaluation of power consumption and lifetime bounds, LAP LAMBERT Academic Publishing, November 25, 2011.

Appendix A: Acronyms and Abbreviations

ACKs	Acknowledgment packets
CWND	Congestion Window
FTP	File Transfer Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
MAC	Medium Access Control
DCM	Dynamic Congestion Model
ECN	Explicit Congestion Notification
RED	Random Early Detection
PID	Peripheral Integral Derivative
MANET	Mobile Ad Hoc Network
MSS	Maximum Segment Size
QoS	Quality of Service
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
RED	Random Early Detection
RTO	Retransmission Timeout
RTT	Round Time Trip
TCP-WELCOME	TCP- Wireless Environment, Link losses, and Congestion packet loss ModEls
$\mathbf{D_T}$	Drop-Tail
AODV	Ad hoc On-demand Distance-Vector Routing
DSDV	Destination-Sequenced Distance-Vector
DYMO	Dynamic MANET On-demand Routing
DSR	Dynamic Source Routing
TORA	Temporally Ordered Routing Algorithm
OLSR	Optimized Link State Routing
$\mathbf{P_L}$	Packet Length
$\mathbf{N_{SD}}$	Node Speed
$\mathbf{N_D}$	Node Dense
PDR	Packet Delivery Ratio
$\mathbf{E2E_D}$	End-to-End Delay
DCM	Dynamic Congestion Model
$\mathbf{E_C}$	Energy Consumption

Appendix B: Published Paper “Protocol for Dynamic Avoiding End-to-End Congestion in MANETs”

Appendix C: DCM source codes and scripts used in NS2

AWK scripts to filter the trace files

eToeDelay.awk (awk script to calculate end-to-end Delay)

```
BEGIN {
 sends=0;
 recvs=0;
 routing_packets=0.0;
control_packets=0.0;
 droppedBytes=0;
 droppedPackets=0;
 highest_packet_id =0;
 sum=0;
 recvnum=0;
   }
  {
 time = $3;
 packet_id = $41;
   #============= CALCULATE DELAY     ================================
 if ( start_time[packet_id] == 0 )  start_time[packet_id] = time;
 if (( $1 == "r") &&  ( $35 == "tcp" ) && ( $19=="AGT" ) &&
(start_time[packet_id] != 0 )) {  end_time[packet_id] = time;  }
 else {  end_time[packet_id] = -1;  }

}
  END {
  for ( i in end_time )
 {
 start = start_time[i];
 end = end_time[i];
 packet_duration = end - start;
 if ( packet_duration > 0 )  { sum += packet_duration; recvnum++; }
 }
     delay=sum/recvnum;

printf("%.2f\n",delay*1000)
  }
```

Normalized overhead.awk (awk script to calculate overhead)

```
BEGIN {
 sends=0;
 recvs=0;
 routing_packets=0.0;
```

```
control_packets=0.0;
 droppedBytes=0;
 droppedPackets=0;
 highest_packet_id =0;
 sum=0;
 recvnum=0;
   }
  {
 time = $3;
 packet_id = $41;
#============= TOTAL AODV OVERHEAD  ================================
  if (( $1 == "s")  && ( $19=="AGT" ))   {  sends++; }
  if ( $37 !="1510") control_packets++;
}
  END {

   printf("%.2f\n", 1-(sends/(sends+control_packets)));
}
```

PDR.awk (awk script to calculate packet delivery ratio)

```
BEGIN {
   seqno = -1;
   droppedPackets = 0;
   receivedPackets = 0;
   count = 0;
    }
    {
   #packet delivery ratio
  if($19 == "AGT" && $1 == "s" && seqno < $41) {
   seqno = $41;
   } else if(($19 == "AGT") && ($1 == "r")) {
   receivedPackets++;
   } else if ($1 == "d" && $35 == "tcp" && $37 > 512){
   droppedPackets++;
   }
   }
   END {
printf("%.2f\n",receivedPackets/(seqno+1)*100)
   }
```

throughput.awk (awk script to calculate throughput)

```
BEGIN {
       recvdSize = 0
       startTime = 1000
       stopTime = 0
  }
```

```
  {
             event = $1
             time = $3
             node_id = $5
             pkt_size = $37
             level = $19

  # 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("%.2f\n",(recvdSize/(stopTime-startTime))*(8/1000))
  }
```

handle_link_failure **function implemented in C++** in /ns-allinone-2.34/ns-2.34/aomdv

```
void AOMDV::handle_link_failure(nsaddr_t id) {
bool error=true;
aomdv_rt_entry *rt, *rtn;
Packet *rerr = Packet::alloc();
struct hdr_aomdv_error *re = HDR_AOMDV_ERROR(rerr);
re->DestCount = 0;
for(rt = rtable.head(); rt; rt = rtn) {
 AOMDV_Path* path;
 rtn = rt->rt_link.le_next;
 if ((rt->rt_flags == RTF_UP) && (path=rt->path_lookup(id)) ) {
  assert((rt->rt_seqno%2) == 0);
  rt->path_delete(id);
  if (rt->path_empty()) {
  rt->rt_seqno++;
  rt->rt_seqno = max(rt->rt_seqno, rt->rt_highest_seqno_heard);
   if (rt->rt_error) {
   re->unreachable_dst[re->DestCount] = rt->rt_dst;
   re->unreachable_dst_seqno[re->DestCount] = rt->rt_seqno;
   re->DestCount += 1;
```

```
   rt->rt_error = false;
      }
rt_down(rt);
}}}
}
```

route_valid_timeout function in /ns-allinone-2.34/ns-2.34/dymoum/dymo_timeout.cc as follows:

```
if (reissue_rreq && re->re_blocks[0].res > 5)
        {
                if (entry->tries[seq1] < RREQ_TRIES)
                {
                        rtable_entry_t *rte;

                        entry->tries++;
                        timer_set_timeout(&entry->timer,
                                RREQ_WAIT_TIME << entry->tries[seq1]);
                        timer_add(&entry->timer);

                        rte = rtable_find(entry->dest_addr);
                        if (rte)
                                re_send_rreq(entry->dest_addr, entry->seqnum,
                                        rte->rt_hopcnt);
                        timer_set_timeout(&entry->timer[0],
                                RREQ_WAIT_TIME << entry->tries[seq1]);
                        timer_add(&entry->timer[0]);

                        else
                                re_send_rreq(entry->dest_addr, entry->seqnum,
                                        0);

                        return;
                }
        }
```

***sendReply* function implemented in C++** in /ns-allinone-2.34/ns-2.34/aomdv

```
void AOMDV::sendReply(nsaddr_t ipdst, u_int32_t hop_count, nsaddr_t rpdst,
                         u_int32_t rpseq, double lifetime, double timestamp,
                                nsaddr_t nexthop, u_int32_t bcast_id,
nsaddr_t rp_first_hop) {
Packet *p = Packet::alloc();
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_ip *ih = HDR_IP(p);
struct hdr_aomdv_reply *rp = HDR_AOMDV_REPLY(p);
rp->rp_type = AOMDVTYPE_RREP;
rp->rp_hop_count = hop_count;
rp->rp_dst = rpdst;
rp->rp_dst_seqno = rpseq;
rp->rp_src = index;
rp->rp_lifetime = lifetime;
```

```
rp->rp_timestamp = timestamp;
rp->rp_bcast_id = bcast_id;
rp->rp_first_hop = rp_first_hop;
ch->ptype() = PT_AOMDV;
ch->size() = IP_HDR_LEN + rp->size();
ch->iface() = -2;
ch->error() = 0;
ch->addr_type() = NS_AF_INET;
ch->next_hop_ = nexthop;
ch->xmit_failure_ = aomdv_rt_failed_callback;
ch->xmit_failure_data_ = (void*) this;
ih->saddr() = index;
ih->daddr() = ipdst;
ih->sport() = RT_PORT;
ih->dport() = RT_PORT;
ih->ttl_ = NETWORK_DIAMETER;
Scheduler::instance().schedule(target_, p, 0.);
}
```

TCL scripts used to validate DCM

```
set nnodes [lindex $argv 1]
set nnodes_1  [lindex $argv 2]
# ======================================================================
# Define options
# ======================================================================
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(ifq)            Queue/RED    ;# interface queue type
#set val(ifq)            CMUPriQueue    ;# interface queue type for DSR
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         10                         ;# max packet in ifq
set val(nn)             $nnodes                    ;# number of mobilenodes
set val(rp)             AOMDV                      ;# routing protocol DYMOUM
set val(x)             1000
set val(y)             1000
set val(energymodel)   EnergyModel                ;# Energy Model
set val(initialenergy) 100                        ;# value
set val(stop)            150
# ======================================================================
# Main Program
# ======================================================================
set file1      [lindex $argv 0]
set TCP_Protocol       [lindex $argv 4]
puts "starting $file1"
# Initialize Global Variables
set ns_        [new Simulator]
# control DYMOUM behaviour from this script
Agent/DYMOUM set debug_ false
```

```
Agent/DYMOUM set no_path_acc_ true
Agent/DYMOUM set reissue_rreq_ false
Agent/DYMOUM set s_bit_ true
Agent/DYMOUM set hello_ival_ 1
$ns_ use-newtrace
set packetSize [lindex $argv 3]
set tracefd     [open AOMDV/$file1-$packetSize.tr w]
set namtrace [open scenario.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
# define different colors for nam data flows
$ns_ color 0 Green
$ns_ color 1 Blue
$ns_ color 2 Red
$ns_ color 3 Yellow
$ns_ color 4 Black
# set up topography object
set topo       [new Topography]
set windowX 30
$topo load_flatgrid $val(x) $val(y)
# Create God
create-god $val(nn)
# configure node
        $ns_ node-config -adhocRouting $val(rp) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channelType $val(chan) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                        -movementTrace OFF \
                         -energyModel $val(energymodel) \
             -initialEnergy $val(initialenergy) \
             -rxPower 35.28e-3 \
             -txPower 31.32e-3 \
            -idlePower 712e-6 \
            -sleepPower 144e-9
#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns_ node]
}
for {set i 1} {$i < $val(nn) } { incr i } {
        $node_($i) set X_ [ expr {$val(x) * rand()} ]
```

```
        $node_($i) set Y_ [ expr {$val(y) * rand()} ]
        $node_($i) set Z_ 0
}
$node_(0) label "sender"
$node_($nnodes_1) label "reciever"
for {set i 0} {$i < $val(nn)} { incr i } {
        $ns_ initial_node_pos $node_($i) 10
}
# Loading Data
puts "Loading scenarios file..."
source "DATA/$file1"
# Setup traffic flow between nodes
# -------------------------------------TCP----------------------
set tcp1 [new Agent/TCP/$TCP_Protocol]
$tcp1 set timestamps_ true
$tcp1 set fid_ 0
$tcp1 set class_ 2
$tcp1 set window_ $windowX
$tcp1 set packetSize_ $packetSize
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp1
$ns_ attach-agent $node_($nnodes_1) $sink1
$ns_ connect $tcp1 $sink1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ $packetSize
$cbr1 set interval_ 0.01
$cbr1 attach-agent $tcp1
$ns_ at 0.0 "$cbr1 start"
#---------------------------------------------------------------------------
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ at 0.0 "$ftp1 start"
#---------------------------------------------------------------------------
set c_(1)  9
set c_(2)  10
set c_(3)  11
set c_(4)  12
set c_(5)  13
set c_(6)  14
set c_(7)  15
set c_(8)  16
set c_(9)  17
set c_(10)  18
set c_(11)  1
set c_(12)  2
set c_(13)  3
set c_(14)  4
set c_(15)  5
set c_(16)  6
set c_(17)  7
set c_(18)  8
# -----------------------------------------------------
```

```
for {set i 1} {$i < 19 } { incr i } {
set tcp_($i) [new Agent/TCP/$TCP_Protocol]
$tcp_($i) set timestamps_ true
$tcp_($i) set fid_ 0
$tcp_($i) set class_ 2
$tcp_($i) set window_ $windowX
$tcp_($i) set packetSize_ $packetSize
set sink_($i) [new Agent/TCPSink]
$ns_ attach-agent $node_($i) $tcp_($i)
$ns_ attach-agent $node_($c_($i)) $sink_($i)
$ns_ connect $tcp_($i) $sink_($i)
set cbr2_($i) [new Application/Traffic/CBR]
$cbr2_($i) set packetSize_ $packetSize
$cbr2_($i) set interval_ 0.01
$cbr2_($i) attach-agent $tcp_($i)
$ns_ at 0.0 "$cbr2_($i) start"
#--------------------------------------------------------
set ftp1_($i) [new Application/FTP]
$ftp1_($i) attach-agent $tcp_($i)
$ns_ at 0.0 "$ftp1 start"
}
#Printing the window size
proc plotWindow {tcpSource file} {
global ns_
set time 0.01
set now [$ns_ now]
}
# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0 "stop"
$ns_ at 150.01 "puts \"NS EXITING...\" ; $ns_ halt"

# procedure to plot the congestion window
proc plotWindow {tcpSource outfile} {
   global ns_
   set now [$ns_ now]
   set cwnd [$tcpSource set rtt_]
   puts  $outfile  "$now $cwnd"
   $ns_ at [expr $now+0.1] "plotWindow $tcpSource  $outfile"
}
set outfile [open  "congestion.xg"  w]
$ns_  at  0.0  "plotWindow $tcp1  $outfile"

proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
 exec xgraph congestion.xg -geometry 300x300 &
   exit 0
}
```

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

Bash scripts used to analysis output trace files by the AWK scripts

```
echo "Bash version ${BASH_VERSION}..."
throughput=(0)
overhead=(0)
PDR=(0)
eToeDelay=(0)
for PktLen in {3000..7000..1000}
 do
  for node in {20..80..20}
   do
    for i in {0..49}
     do
     echo "Starting Scenario $i packet length $PktLen nodes = $node"
     throughput[$i]=$(awk -f throughput.awk A100$i-$node-$PktLen.tr)
     echo ${throughput[$i]}
     overhead[$i]=$(awk -f overhead.awk A100$i-$node-$PktLen.tr)
     echo ${overhead[$i]}
     PDR[$i]=$(awk -f PDR.awk A100$i-$node-$PktLen.tr)
     echo ${PDR[$i]}
     eToeDelay[$i]=$(awk -f eToeDelay.awk A100$i-$node-$PktLen.tr)
     echo ${eToeDelay[$i]}
     done
total=0
sum=0
average=0
sum=$( IFS="+"; bc <<< "${throughput[*]}")
average=$(echo $sum / ${#throughput[@]} | bc -l)
echo $PktLen $node $average >> throughput.txt
total=0
sum=0
average=0
sum=$( IFS="+"; bc <<< "${overhead[*]}")
average=$(echo $sum / ${#overhead[@]} | bc -l)
echo $PktLen $node $average >> overhead.txt
total=0
sum=0
average=0
sum=$( IFS="+"; bc <<< "${PDR[*]}")
average=$(echo $sum / ${#PDR[@]} | bc -l)
echo $PktLen $node $average >> PDR.txt
total=0
sum=0
average=0
sum=$( IFS="+"; bc <<< "${eToeDelay[*]}")
average=$(echo $sum / ${#eToeDelay[@]} | bc -l)
echo $PktLen $node $average >> eToeDelay.txt
    done
```

```
 done
```

Bash scripts used to generate output trace files

```
echo "Bash version ${BASH_VERSION}..."
for k in {3000..7000..1000}
do
echo "Starting $k packet length"
for i in {0..9}
  do
     echo "Starting $i Scenario 20"
ns DYMOUM-RED.tcl A200$i-20 20 19 $k DCM
 done
for i in {0..9}
  do
     echo "Starting $i Scenario 40"
ns DYMOUM-RED.tcl A200$i-40 40 39 $k DCM
 done
for i in {0..9}
  do
     echo "Starting $i Scenario 60"
ns DYMOUM-RED.tcl A200$i-60 60 59 $k DCM
 done
for i in {0..9}
  do
     echo "Starting $i Scenario 80"
ns DYMOUM-RED.tcl A200$i-80 80 79 $k DCM
 done
done
```

Appendix D: Simulation results used in chapter five

Table D-1: Throughput (kbps) for TCP-Newreno, TCP-Vegas, TCP-Westwood, TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED over AOMDV routing protocol.

Queue Type		**Random Early Detection (RED)**					
Network Parameter		**TCP**					
P_L	**N_D**	**ATCP**	**WELCOME**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	518.007	569	536.382	517.466	525.954	547.448
3000	**40**	489.11	517.5	506.395	494.951	493.715	497.541
3000	**60**	482.741	511.8	497.271	492.904	487.015	504.083
3000	**80**	441.026	497.9	461.889	460.34	469.946	490.245
4000	**20**	569.242	589.7	579.675	569.333	567.595	592.797
4000	**40**	530.203	563.7	537.934	550.346	534.981	544.038
4000	**60**	514.022	558.7	549.73	550.424	537.558	562.643
4000	**80**	491.786	547.7	519.1	505.116	499.915	539.464
5000	**20**	605.092	626.2	614.981	591.227	607.149	623.505
5000	**40**	555.672	582.9	570.924	575.02	559.918	590.558
5000	**60**	555.67	568.5	576.256	574.381	577.106	582.483

5000	**80**	527.791	555.1	541.675	542.222	535.449	572.474
6000	**20**	623.077	667.3	633.151	628.549	623.451	652.55
6000	**40**	575.748	603.6	608.588	603.81	581.408	605.175
6000	**60**	579.7	611.3	593.915	587.956	597.57	598.21
6000	**80**	546.011	576.9	557.485	579.281	564.418	573.083
7000	**20**	641.661	651.1	656.849	636.039	650.449	660.608
7000	**40**	575.96	625	606.139	620.092	598.732	613.845
7000	**60**	596.845	609.2	622.301	607.146	604.633	620.27
7000	**80**	548.438	610.4	579.218	572.072	569.692	600.684

Table D-2: Throughput (kbps) for TCP-Newreno, TCP-Vegas, TCP-Westwood, TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED over DYMO routing protocol.

Queue Type		**Random Early Detection (RED)**					
Network Parameter		**TCP**					
P_L	**N_D**	**ATCP**	**WELCOME**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	491.773	501.565	504.071	501.891	481.031	584.5
3000	**40**	486.002	512.684	461.344	470.838	449.583	544.4
3000	**60**	480.801	498.6	453.984	459.222	436.292	559.9
3000	**80**	475.007	485.808	449.317	488.297	426.931	528.6
4000	**20**	489.726	567.617	563.343	556	548.484	654.4
4000	**40**	482.348	573.673	501.456	518.691	505.479	620.8
4000	**60**	477.101	560.8	502.614	526.01	492.094	588.7
4000	**80**	470.377	550.121	518.029	550.296	467.472	625.1
5000	**20**	488.465	601.253	601.052	588.961	595.97	692.7
5000	**40**	480.021	589.763	540.668	532.958	540.515	652.2
5000	**60**	472.926	587.042	559.862	551.04	524.144	629.5
5000	**80**	466.391	585.784	515.676	563.059	512.535	604.7
6000	**20**	486.37	628.818	619.479	627.187	619.248	715.3
6000	**40**	477.635	629.682	595.841	568.47	577.614	680.9
6000	**60**	470.213	609.153	557.04	592.909	558.784	678.1
6000	**80**	462.597	605.416	546.597	576.898	480.075	666.9
7000	**20**	485.595	639.782	638.701	650.191	631.906	708.2
7000	**40**	474.591	626.434	624.407	599.009	592.15	702
7000	**60**	467.167	636.561	587.465	579.724	566.406	700.7
7000	**80**	459.524	631.409	556.189	615.429	387.705	707.2

Table D-3: Throughput (kbps) for TCP-Newreno, TCP-Vegas, TCP-Westwood and TCP-DCM, with queue type: D_T over routing protocols: DYMO and AOMDV.

Queue Type		**Drop-Tail**	
Routing Protocol		**AOMDV**	**DYMO**
Network Parameter			
P_L	**N_D**	**TCP**	

		Newreno	Vegas	WESTWOOD	DCM	Newreno	Vegas	WESTWOOD	DCM
3000	**20**	510.707	512.453	512.528	527.714	488.789	494.297	470.172	577.833
3000	**40**	493.707	488.83	509.126	499.221	435.473	467.358	437.142	530.637
3000	**60**	497.958	484.586	489.236	499.203	448.752	462.166	395.089	527.066
3000	**80**	450.295	456.424	447.971	464.642	421.742	470.983	415.784	513.724
4000	**20**	560.605	567.766	542.593	565.956	549.392	560.391	533.056	627.178
4000	**40**	538.723	534.968	544.146	532.274	488.885	529.165	480.181	583.578
4000	**60**	540.826	542.45	530.247	538.781	467.523	514.571	457.103	586.017
4000	**80**	514.201	488.03	494.623	491.222	474.488	513.815	400.366	584.435
5000	**20**	595.058	599.396	582.281	610.948	566.239	601.446	550.892	660.512
5000	**40**	578.517	569.52	563.679	568.665	536.741	573.212	537.62	635.055
5000	**60**	564.915	566.624	564.364	560.808	514.802	545.375	487.314	601.206
5000	**80**	515.388	519.873	525.041	533.645	486.108	523.188	454.744	593.11
6000	**20**	613.463	614.853	609.485	631.048	592.366	619.274	585.94	687.256
6000	**40**	580.89	588.051	586.181	592.417	565.242	602.016	536.735	634.582
6000	**60**	585.094	573.9	573.619	586.462	536.156	564.84	516.968	604.391
6000	**80**	555.678	558.535	543.449	566.196	530.163	587.514	500.226	604.481
7000	**20**	626.754	635.553	619.241	653.666	613.356	635.071	605.384	705.236
7000	**40**	604.254	612.2	597.204	608.36	547.815	598.377	555.108	648.736
7000	**60**	596.677	603.626	599.26	606.201	558.343	590.727	528.48	674.671
7000	**80**	556.137	559.707	558.398	591.793	564.191	557.22	522.775	646.522

Table D-4: E2E$_D$ (ms) for TCP-Newreno, TCP-Vegas, TCP-Westwood, TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED over AOMDV routing protocol.

Queue Type		**Random Early Detection (RED)**					
Network Parameter		**TCP**					
P_L	**N_D**	**ATCP**	**WELCOME**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	594.371	368.956	559.751	578.444	518.687	460.372
3000	**40**	609.401	406.777	563.35	614.057	537.394	470.089
3000	**60**	608.087	390.619	560.856	595.342	525.504	431.398
3000	**80**	677.82	450.49	559.828	592.974	522.204	444.141
4000	**20**	691.955	505.849	657.641	674.632	630.07	533.362
4000	**40**	769.351	559.866	634.071	701.383	618.522	508.715
4000	**60**	737.415	500.735	626.456	662.742	603.771	473.823
4000	**80**	749.136	579.418	631.423	647.113	598.279	481.291
5000	**20**	826.083	646.424	709.447	752.023	682.235	618.159
5000	**40**	868.326	684.25	716.099	772.717	668.599	590.715
5000	**60**	806.305	660.23	713.174	712.984	649.362	538.024
5000	**80**	856.291	616.234	710.288	753.596	714.867	531.522
6000	**20**	908.975	802.74	794.334	816.507	779.675	676.068
6000	**40**	950.719	805.682	775.889	811.402	703.177	616.995
6000	**60**	936.792	745.841	753.601	822.745	726.105	602.644
6000	**80**	958.278	751.38	776.61	799.082	734.396	543.598

7000	**20**	999.02	884.777	867.518	895.478	828.642	711.127
7000	**40**	1089.516	905.047	823.463	867.443	853.073	633.831
7000	**60**	978.94	798.173	791.054	840.411	774.951	613.908
7000	**80**	1010.318	881.056	841.867	790.872	794.136	673.069

Table D-5: $E2E_D$ (ms) for TCP-Newreno, TCP-Vegas, TCP-Westwood, TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED over DYMO routing protocol.

Queue Type		**Random Early Detection (RED)**					
Network Parameter		**TCP**					
P_L	N_D	**ATCP**	**WELCOME**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	703.895	441.868	377.583	420.7	360.156	386.747
3000	**40**	579.866	332.783	366.658	372.3	318.409	356.831
3000	**60**	500.358	329.228	315.696	330.4	296.503	344.863
3000	**80**	449.111	294.227	287.04	322	304.512	319.377
4000	**20**	916.979	499.919	433.634	482.4	401.965	451.631
4000	**40**	655.948	423.155	405.41	401.7	399.637	405.986
4000	**60**	703.195	410.08	390.901	381.8	366.943	368.117
4000	**80**	632.348	358.488	334.781	385.4	366.659	355.263
5000	**20**	1136.091	627.216	485.361	538.2	465.322	487.592
5000	**40**	967.037	512.161	459.009	499.7	433.194	438.923
5000	**60**	787.283	489.16	419.184	423.4	417.038	415.794
5000	**80**	699.093	448.808	369.854	407.5	371.406	366.809
6000	**20**	1064.319	697.968	547.533	545.9	525.715	507.145
6000	**40**	957.793	577.766	471.378	513.9	481.234	486.288
6000	**60**	1020.373	553.966	445.412	458.8	446.145	434.212
6000	**80**	1081.747	552.546	440.707	465.1	415.221	398.031
7000	**20**	1368.059	837.277	576.612	631.2	564.299	540.674
7000	**40**	1245.64	698.224	524.003	510	542.305	487.337
7000	**60**	1372.224	599.049	499.254	494.9	486.621	469.264
7000	**80**	1103.266	576.862	455.857	502.2	467.705	442.744

Table D-6: $E2E_D$ (ms) for TCP-Newreno, TCP-Vegas, TCP-Westwood and TCP-DCM, with queue type: D_T over routing protocols: DYMO and AOMDV.

Queue Type		**Drop-Tail**							
Routing Protocol		**AOMDV**				**DYMO**			
Network Parameter									
P_L	N_D	**TCP**							
		Newreno	**Vegas**	**WESTWOOD**	**DCM**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	589.145	662.69	568.139	498.593	395.816	392.713	380.103	385.336
3000	**40**	650.14	681.056	574.944	482.495	363.285	381.335	358.703	362.262
3000	**60**	578.646	653.335	580.814	457.942	340.135	355.863	331.542	350.745
3000	**80**	625.474	686.83	611.828	446.089	320.541	338.675	299.44	329.693

4000	**20**	735.002	797.855	689.253	585.497	444.892	458.862	425.967	427.199
4000	**40**	749.42	765.786	696.773	574.696	417.905	462.7	404.391	423.387
4000	**60**	686.641	754.319	671.666	507.596	392.423	425.39	397.737	364.896
4000	**80**	668.428	783.289	685.675	566.55	369.424	387.774	348.425	357.287
5000	**20**	813.209	868.183	767.998	647.035	522.068	525.1	473.158	476.689
5000	**40**	802.868	839.207	754.928	625.009	451.185	471.324	459.313	442.342
5000	**60**	759.889	842.048	728.143	602.442	466.62	468.414	435.889	413.128
5000	**80**	815.199	844.946	720.013	629.368	409.915	388.243	417.228	375.99
6000	**20**	916.096	950.918	870.739	738.558	533.579	572.584	502.001	495.602
6000	**40**	855.423	980.094	841.651	699.035	490.697	519.083	468.554	460.35
6000	**60**	810.585	902.524	831.584	668.703	480.733	493.988	476.455	448.53
6000	**80**	842.778	865.454	832.614	644.515	463.385	434.945	415.465	374.234
7000	**20**	964.31	1017.639	959.638	831.587	577.488	594.587	569.865	505.917
7000	**40**	930.34	1062.688	902.996	758.53	549.534	558.426	531.258	492.862
7000	**60**	908.104	999.369	874.319	705.069	506.137	522.326	499.893	461.452
7000	**80**	916.341	968.178	884.975	680.411	475.299	516.285	447.866	422.037

Table D-7: The Normalized Overhead for TCP-Newreno, TCP-Vegas, TCP-Westwood, TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED over AOMDV routing protocol.

Network Parameter		**TCP**					
Routing Protocol		**AOMDV**			**DYMO**		
P_L	**N_D**	**ATCP**	**WELCOME**	**DCM**	**ATCP**	**WELCOME**	**DCM**
3000	**20**	0.879	0.87	0.88	0.821	0.879	0.91
3000	**40**	0.942	0.94	0.94	0.857	0.942	0.96
3000	**60**	0.969	0.96	0.96	0.89	0.969	0.98
3000	**80**	0.98	0.98	0.97	0.923	0.98	0.99
4000	**20**	0.886	0.88	0.89	0.825	0.886	0.92
4000	**40**	0.953	0.95	0.95	0.866	0.953	0.97
4000	**60**	0.972	0.97	0.97	0.904	0.972	0.98
4000	**80**	0.98	0.98	0.98	0.933	0.98	0.99
5000	**20**	0.897	0.89	0.9	0.827	0.897	0.92
5000	**40**	0.958	0.95	0.96	0.872	0.958	0.97
5000	**60**	0.978	0.97	0.97	0.911	0.978	0.98
5000	**80**	0.982	0.98	0.98	0.94	0.982	0.99
6000	**20**	0.905	0.9	0.99	0.831	0.905	0.93
6000	**40**	0.962	0.96	0.96	0.879	0.962	0.97
6000	**60**	0.98	0.97	0.98	0.919	0.98	0.98
6000	**80**	0.988	0.98	0.98	0.949	0.988	0.99
7000	**20**	0.912	0.91	0.91	0.833	0.912	0.94
7000	**40**	0.968	0.96	0.96	0.888	0.968	0.97
7000	**60**	0.981	0.98	0.98	0.925	0.981	0.99
7000	**80**	0.99	0.99	0.98	0.95	0.99	0.99

Table D-8: The Packet Delivery Ratio (PDR) for TCP-Newreno, TCP-Vegas, TCP-Westwood, TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED over AOMDV routing protocol.

Queue Type		**Random Early Detection (RED)**					
Network Parameter		**TCP**					
P_L	**N_D**	**ATCP**	**WELCOME**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	91.911	93.461	91.822	91.867	92.27	94.424
3000	**40**	91.323	93.346	92.192	90.961	92.481	94.603
3000	**60**	91.579	93.105	91.882	91.503	92.515	94.574
3000	**80**	91.29	93.634	91.763	91.37	92.386	94.347
4000	**20**	91.341	92.849	90.938	90.929	91.157	93.549
4000	**40**	90.943	92.042	90.548	90.213	90.755	93.545
4000	**60**	90.395	92.344	91.098	90.516	91.193	93.602
4000	**80**	90.461	92.467	90.886	90.618	91.082	93.275
5000	**20**	90.946	92.016	90.7	89.834	90.732	92.71
5000	**40**	90.011	91.112	90.05	89.431	90.438	92.401
5000	**60**	89.793	91.733	90.521	89.888	90.668	92.322
5000	**80**	89.599	91.846	89.792	89.6	90.731	92.2
6000	**20**	89.618	91.018	89.973	89.514	90.098	91.871
6000	**40**	89.149	90.416	89.81	88.594	89.76	91.693
6000	**60**	89.489	90.422	89.339	88.912	89.844	91.438
6000	**80**	88.571	90.45	89.257	88.826	89.765	91.925
7000	**20**	89.11	90.18	89.062	88.853	89.645	90.838
7000	**40**	88.341	89.732	88.361	87.905	89.354	90.693
7000	**60**	88.563	89.873	89.477	88.623	89.251	90.904
7000	**80**	87.785	89.715	88.795	87.848	88.967	90.316

Table D-9: The Packet Delivery Ratio (PDR) for TCP-Newreno, TCP-Vegas, TCP-Westwood, TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED over DYMO routing protocol.

Queue Type		**Random Early Detection (RED)**					
Network Parameter		**TCP**					
P_L	**N_D**	**ATCP**	**WELCOME**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	47.389	46.679	50.247	49.4	49.458	51.818
3000	**40**	28.122	28.65	30.992	29.5	30.018	33.766
3000	**60**	21.777	22.587	22.882	24.3	22.379	24.711
3000	**80**	17.732	17.18	18.391	18.4	18.644	21.375
4000	**20**	42.985	40.962	44.817	44.7	44.702	45.039
4000	**40**	24.829	24.185	24.129	27.5	26.668	28.367
4000	**60**	18.478	17.478	19.088	19.5	18.935	21.391
4000	**80**	15.328	15.09	16.15	16.6	14.987	19.225
5000	**20**	37.67	36.927	40.585	40.9	40.97	41.334
5000	**40**	21.763	21.935	22.455	23.1	22.67	24.47
5000	**60**	15.134	15.615	17.192	17.3	17.153	18.554

5000	**80**	12.45	11.571	12.938	12	13.758	15.425
6000	**20**	34.038	33.978	36.828	37.8	36.689	38.609
6000	**40**	18.525	19.228	20.113	21.4	21.12	21.497
6000	**60**	13.278	13.334	14.65	16.1	14.515	16.9
6000	**80**	10.567	11.309	11.13	11.9	10.403	12.577
7000	**20**	31.725	31.454	34.582	33.7	34.299	35.642
7000	**40**	17.045	16.119	19.891	18.3	18.726	21.387
7000	**60**	12.118	12.721	13.294	13.9	12.788	13.89
7000	**80**	9.858	10.306	10.286	11.7	10.826	12.048

Table D-10: The Packet Delivery Ratio (PDR) for TCP-Newreno, TCP-Vegas, TCP-Westwood and TCP-DCM, with queue type: D_T over routing protocols: DYMO and AOMDV.

Queue Type		**Drop-Tail**							
Routing Protocol		**AOMDV**				**DYMO**			
Network Parameter									
P_L	**N_D**	**TCP**							
		Newreno	**Vegas**	**WESTWOOD**	**DCM**	**Newreno**	**Vegas**	**WESTWOOD**	**DCM**
3000	**20**	92.415	91.998	92.693	95.025	46.679	47.389	45.658	50.399
3000	**40**	92.571	91.717	93.029	95.203	28.65	28.122	28.533	32.281
3000	**60**	92.751	92.197	92.748	94.899	22.587	21.777	19.785	24.016
3000	**80**	92.491	91.877	92.69	95.187	17.18	17.732	17.76	21.901
4000	**20**	91.826	90.825	91.795	94.273	40.962	42.985	42.141	45.347
4000	**40**	91.734	90.964	92.605	93.882	24.185	24.829	23.673	26.976
4000	**60**	91.884	91.244	92.205	94.168	17.478	18.478	17.572	20.983
4000	**80**	91.707	90.777	92.047	93.685	15.09	15.328	12.431	18.122
5000	**20**	91.118	90.453	91.19	93.322	36.927	37.67	37.978	40.802
5000	**40**	90.958	90.289	91.075	93.108	21.935	21.763	21.659	25.171
5000	**60**	90.978	90.258	91.43	92.97	15.615	15.134	14.787	17.525
5000	**80**	90.444	90.384	91.122	92.47	11.571	12.45	11.322	14.888
6000	**20**	90.829	89.904	90.843	92.096	33.978	34.038	35.241	37.151
6000	**40**	90.088	89.386	90.418	91.909	19.228	18.525	19.152	22.112
6000	**60**	90.739	89.447	90.605	91.794	13.334	13.278	13.336	15.127
6000	**80**	90.172	89.899	90.632	92.093	11.309	10.567	10.979	14.165
7000	**20**	89.646	88.909	90.063	91.417	31.454	31.725	31.58	34.621
7000	**40**	89.747	89.013	89.568	91.472	16.119	17.045	16.987	18.807
7000	**60**	89.857	89.44	89.819	91.201	12.721	12.118	12.262	14.489
7000	**80**	89.339	89.497	89.653	91.059	10.306	9.858	9.463	10.555

Table D-11: The Overall Energy Consumption (Watt) for TCP-DCM, ATCP and TCP-WELCOME, with queue type: RED and D_T over AOMDV routing protocol.

Routing Protocol	**AOMDV**	**DYMO**

Network Parameter		TCP								
Queue Type		RED			D_T	RED				D_T
P_L	N_D	**ATCP**	**WELCOME**	**DCM**	**DCM**	**ATCP**	**WELCOME**	**DCM**	**DCM**	
3000	**20**	110.232	98.734	97.999	97.386	106.56	85.557	97.237	97.237	106.56
3000	**40**	208.006	197.914	196.237	195.569	177.495	181.681	143.674	143.674	177.495
3000	**60**	306.222	294.286	292.193	292.459	263.64	241.38	243.431	243.431	263.64
3000	**80**	402.169	390.895	390.111	388.156	285.008	259.464	339.772	339.772	285.008
4000	**20**	115.334	99.165	98.285	98.251	100.334	104.154	97.343	97.343	100.334
4000	**40**	214.315	196.857	196.515	196.297	184.209	142.514	190.098	190.098	184.209
4000	**60**	312.077	295.699	293.9	293.137	248.68	280.63	280.11	280.11	248.68
4000	**80**	407.579	390.725	391.055	388.076	382.039	308.19	342.212	342.212	382.039
5000	**20**	120.317	99.357	98.163	98.152	101.022	95.734	97.96	97.96	101.022
5000	**40**	219.549	198.293	197.539	197.068	176.24	171.216	167.972	167.972	176.24
5000	**60**	317.287	296.46	294.045	293.952	284.605	273.008	246.916	246.916	284.605
5000	**80**	413.881	392.425	391.692	391.032	322.386	304.603	291.242	291.242	322.386
6000	**20**	120.624	99.797	98.845	98.53	101.303	97.376	96.494	96.494	101.303
6000	**40**	219.362	199.099	197.087	197.156	179.413	152.496	165.409	165.409	179.413
6000	**60**	317.866	295.876	296.493	294.101	281.691	222.679	268.199	268.199	281.691
6000	**80**	413.55	393.049	392.462	391.364	290.036	317.899	296.423	296.423	290.036
7000	**20**	121.049	99.507	98.583	98.549	93.996	90.314	97.048	97.048	93.996
7000	**40**	220.418	198.877	197.618	197.944	196.427	181.087	184.63	184.63	196.427
7000	**60**	319.364	296.658	295.302	294.673	270.042	234.612	239.398	239.398	270.042
7000	**80**	413.137	392.962	392.318	391.55	369.456	363.099	357.845	357.845	369.456