PROJECT REPORT ON

Analyzing the performance of different TCP Variants in Wireless Ad-hoc Networks using different Routing Protocols

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

Submitted by

123003114 – KODAVALLA DURGA AVINASH – CSE

Under the Guidance of

Prof. Sasikala Devi. N

School of Computing
SASTRA DEEMED TO BE UNIVERSITY
(A University established under section 3 of the UGC Act, 1956)
Tirumalaisamudram
Thanjavur - 613401
December (2021)

SHANMUGHA

ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY

(SASTRA DEEMED TO BE UNIVERSITY)

(A University Established under section 3 of the UGC Act, 1956)

TIRUMALAISAMUDRAM, THANJAVUR – 613401

BONAFIDE CERTIFICATE

Certified that this project work entitled **"Analyzing the Performance of different TCP Variants in Wireless Ad-hoc Networks using different Routing Protocols"** submitted to the Shanmugha Arts, Science, Technology & Research Academy (SASTRA Deemed to be University), Tirumalaisamudram - 613401 by Kodavalla Durga Avinash (123003114), CSE in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in their respective programme. This work is an original and independent work carried out under my guidance, during the period August 2021 - December 2021.

Prof. Sasikala Devi. N

ASSOCIATE DEAN
SCHOOL OF COMPUTING

Submitted for Project Viva Voce held on___________________

Examiner – I

Examiner – II

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

First of all, I would like to thank God Almighty for his endless blessings.

I would like to express my sincere gratitude to **Dr. S. Vaidyasubramaniam, Vice – Chancellor** for his encouragement during the span of my academic life at SASTRA Deemed University.

I would forever remain grateful and I would like to thank **Dr. A. Umamakeswari, Dean, School of Computing,** and **R. Chandramouli, Registrar** for their overwhelming support provided during my course span in SASTRA Deemed University.

I am extremely grateful to **Dr. Shankar Sriram, Associate Dean, School of Computing** for his constant support, motivation, and academic help extended for the past three years of my life in the School of Computing.

I would specially thank and express my gratitude to **Dr. N. Sasikala Devi, Associate Professor, School of Computing** for providing me an opportunity to do this project and for her guidance and support to complete the project.

I also thank all the teaching and non – teaching faculty, and all other people who have directly or indirectly helped me through their support, encouragement, and all other assistance extended for completion of my project and successful completion of all courses during my academic life at SASTRA Deemed University.

Finally, I thank my parents and all others who helped me acquire this interest in the project and aided me in completing it within the deadline without much struggle.

ABSTRACT

TCP [Transmission Control Protocol] is a connection-oriented protocol. This indicates a connection is mounted and maintained until the packages at each end have finished replacing messages. The importance of this protocol is its diverse congestion management mechanisms that manipulate TCP sending rate and makes TCP react to congestion alerts. This is a transport layer protocol that works with the internet Protocol [IP – complements tat network layer] which defines how gadgets send packets of data to each other. In this project, performing a comparative analysis of TCP variants like TCP Tahoe, TCP Reno, TCP New Reno, TCP Vegas, and TCP Sack over different routing protocols like AODV, DSDV, and DSR considering factors such as Packet Delivery Ratio [PDR], Delay [in ms], and Throughput [in Mbps] while the mobile nodes are in motion [considering different speeds] using the NS-2 Network Simulator.

KEYWORDS: Transmission Control Protocol [TCP]; TCP Variants; Destination-Sequenced Distance-Vector [DSDV]; Ad-hoc On-demand Distance Vector [AODV]; Dynamic Source Routing [DSR]; NS-2 [Network Simulation Software]

LIST OF FIGURES

ABBREVIATIONS

AODV	Ad – hoc On – demand Distance Vector
DSDV	Destination – Sequenced Distance Vector
DSR	Dynamic Source Routing
ETE	End – to – End
FTP	File Transfer Protocol
GOD	General Operation Director
IFQ	Interface Queue Type
IP	Internet Protocol
NAM	Network Animator
PDR	Packet Delivery Ratio
TCP	Transmission Control Protocol

CHAPTER 1 INTRODUCTION

During the last few decades, improvements in computing have become more and more extensive. Enabling communication between these devices and their power combined then their efficacy is multiplied by many times more. Communication in short consists of transmitting information from one place to another via a medium. Wireless communication has become available for use with personal computing devices. Parallel to this development is the expansion of the Internet, which has seen a huge amount of growth since the introduction of the World Wide Web in the early 1990s. The core of the Internet is the IP packet network, which forwards packets from one place to another. On top of this, we have protocols that provide a data transport service, where the major reliable transport protocol used today is TCP.

The increased use of wireless technologies in our current lives makes TCP over wireless an important topic of research. Knowing the TCP and its variants of approaches used to improve the performance of TCP is a major challenge. Many congestion control algorithms are evolved from basic TCP Tahoe which was introduced by Jacobson and Karels. TCP generally gives a poor performance in high-speed bandwidth links mainly because of slow response of congestion control. Even the high-speed wired and wireless networks are in a growth and TCP must be improved to work better in its limitations by designing algorithms that can work efficiently in those situations. This led many researchers to involve in work to expand the effectiveness of TCP protocols. This paper compares the factors throughput, ETE Delay, and PDR of TCP Tahoe, Reno, New Reno, Vegas, Sack, and Linux on top of routing protocols AODV, DSDV, and DSR with change in mobility speeds of nodes using FTP application in Wireless Ad-hoc Networks.

CHAPTER 2 RELATED WORKS

Bazi, K., & Nassereddine, B. (2020, March). Comparative analysis of TCP congestion control mechanisms. In Proceedings of the 3rd International Conference on Networking, Information Systems & Security (pp. 1-4). This paper discusses, identifies, analyzes, and compares the behavior of congestion control mechanisms.

Kadhim, A. J., & Naser, J. I. (2020). PERFORMANCE OF ROUTING PROTOCOLS WITH CONGESTION ALGORITHMS OF TCP VARIANTS IN MANET. Telecommunications and Radio Engineering, 79(3). This paper studies and analyzes the behavior of DSDV, AOMDV, AODV, and DSR routing techniques with TCP variants and application layer protocol (Telnet).

Patel, S., Shukla, Y., Kumar, N., Sharma, T., & Singh, K. (2020, March). A comparative performance analysis of tcp congestion control algorithms: Newreno, westwood, veno, bic, and cubic. In 2020 6th International Conference on Signal Processing and Communication (ICSC) (pp. 23-28). IEEE. This paper carried out a comparative study on window size for various congestion control algorithms like New Reno, Veno, BIC, CUBIC and Westwood then analyzed performance.

Grazia, C. A. (2019, October). IEEE 802.11 n/AC wireless network efficiency under different TCP congestion controls. In 2019 international conference on wireless and mobile computing, networking and communications (WiMob) (pp. 1-6). IEEE. This paper presents an experimental evaluation of network efficiency using several TCP congestion control algorithms.

Bazi, K., & Nassereddine, B. (2019, October). Analysis of TCP congestion management algorithms for wireless mesh network. In Proceedings of the 4th International Conference on Big Data and Internet of Things (pp. 1-5). This paper tries to evoke the main mechanisms developed to solve the problem of congestion using main algorithms integrated with TCP's implementations for better congestion management and conclude the characteristics of a network.

Das, N., Bisoy, S. K., & Tanty, S. (2019). Performance analysis of tcp variants using routing protocols of MANET in grid topology. In Cognitive Informatics and Soft Computing (pp. 239-245). Springer, Singapore. In this paper, TCP variants such as New Reno and Vegas including Full TCP protocol is analyzed using AODV, DSR, and DSDV routing protocols using grid topology.

Kotian, D., Shetty, S., Begum, S., & Gadagkar, A. V. (2019). Analysis of Various TCP Variants over MANET Routing Protocols. This paper has a performance comparison between TCP variants such as TCP Tahoe, TCP Reno, TCP New Reno, TCP Fack, TCP Sack, TCP Vegas, Westwood, and TCP Lite in DSDV, TORA, AODV, and DSR.

Shenoy, S. U., Kumari, M. S., Shenoy, U. K. K., & Anusha, N. (2017, February). Performance analysis of different TCP variants in wireless ad hoc networks. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) (pp. 891-894). IEEE. This paper performed a comparative analysis of TCP variants in AODV, DSDV, and DSR protocols.

Singh, B. (2013). A Comparative Study of Different TCP Variants in Networks. International Journal of Computer Trends and Technology (IJCTT), 4(8), 2962-2966. This paper includes the study of the performance of TCP variants in different types of networks which include both wired and wireless networks.

Alfredsson, S. (2005). TCP in wireless networks: Challenges, optimizations and evaluations (Doctoral dissertation, Karlstads universitet). In this paper, transport layer behavior in wireless networks was studied. Features of various TCP congestion control algorithms are analyzed and factors like packet loss, signal power, and throughput are considered.

Stoica, I. (2005). A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas. Communication Networks, Student Project. This paper includes the study of various TCP variants like Tahoe, Reno, New Reno, Sack, and Vegas in wireless communication networks, and their performance is analyzed and characteristics are noted.

CHAPTER 3 PROPOSED FRAMEWORK

3.1 UNDERSTANDING TCP VARIANTS

This transport layer protocol facilitates the transmission of data packets from source to destination. It's a connection-oriented protocol that means it establishes the connection before the communication is started between devices in a network. It divides the data into many packets and transfers these packets to a destination. The connection will be active between devices until data is transmitted. This protocol has many congestion avoidance algorithms which can improve PDR, throughput, and decrease ETE delay.

3.1.1 TCP Tahoe

This congestion control algorithm works on the principle of 'conservation of packets' if the connection is active at available bandwidth capacity. A packet is not sent into the network unless a packet is travelled out of network. Tahoe uses 'Additive Increase Multiplicative Decrease'. A packet loss is considered as congestion and saves half of the current window to one and a slow start until it reaches a threshold value.

3.1.2 TCP Reno

This retains a similar principle as Tahoe like slow start and then adding intelligence so that lost packets are detected earlier and pipeline not emptied. This uses an algorithm called 'Fast Re-Transmit'.

3.1.3 TCP New Reno

This is a slightly modified version of Reno, as it can detect multiple packet losses. This is more efficient if this case ever occurs, then it enters into fast re-transmit if duplicate packets received. It reduces the problem faced by Reno and Tahoe reducing the current window multiple times.

3.1.4 TCP Vegas

This algorithm extends on the re-transmission technique of Reno. Vegas is much different from above-stated congestion control algorithms. Congestion is detected if there is a decrease in sending rate than expected rate. In this way, it combats congestion efficiently not wasting the bandwidth and creating congestion in the first place. This is also called 'Modified Slow – Start'.

3.1.5 TCP Sack

TCP with 'Selective Acknowledgements' considering limitations of Reno and New Reno, TCP Sack is designed. This algorithm also uses a few parts of slow-start and fast re-transmit and requires segments not to be acknowledged cumulatively but it should be done selectively.

3.1.6 TCP Linux

This congestion control algorithm modules are imported from the Linux kernel. Generating simulation data that are consistent with the behavior of Linux hosts. These modules are compiled into the NS-2 binary files.

3.2 NETWORK ENVIRONMENT

Analysis of the performance of various TCP variants like TCP Tahoe, TCP Reno, TCP New Reno, TCP Vegas, TCP Sack, and TCP Linux with a combination of different routing protocols like AODV, DSDV, and DSR are performed in the NS-2 simulator.

Considering the MAC protocol be IEEE 802.11, the channel is set to Wireless channel with Two Ray Ground propagation model and Omni Antenna. Ten mobile nodes are considered to simulate the network environment under 450 x 450 – meter topology. The routing protocols are set in the simulation parameters section in the code. The speed of the mobile nodes is changed according to speed considerations. The start and endpoints (coordinates) are set in the code.

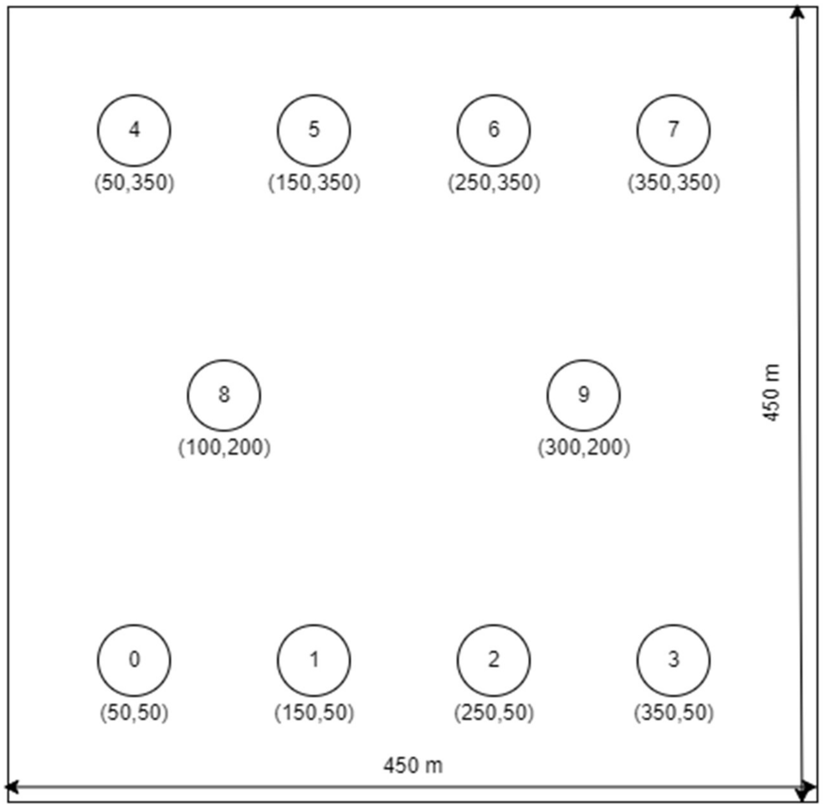


Figure 1. Initial Nodes Position

Initially, the nodes are at the position shown in Figure 1. Node 0 will go to Node 7's initial position and the same for 7 which goes to 0's initial position. Similarly, 1 to 6 and 6 to 1, 2 to 5 and 5 to 2, 3 to 4 and 4 to 3, and then 8 to 9 and 9 to 8. All will move at in same speed in meter per second given according to cases for analysis.

3.3 PARAMETERS FOR ANALYSIS

In this paper, simulation is repeated for different node speeds [5, 10, 15, 20, 25, 30 in meters per second] and three different routing protocols [AODV, DSDV, DSR]. Factors like PDR, Throughput, and ETE Delay are calculated and used for performance analysis.

3.3.1 Packet Delivery Ratio [PDR]

PDR is a ratio of the number of packets received correctly at the destination node to the number of packets dispatched by means of the source node. This value metric is shown in percentage on this paper. The higher the value of PDR then the better the performance of the network, as packets acquired effectively at the destination node will be higher.

3.3.2 Throughput

Throughput is calculated by the division of the size of total packets received at the destination nodes in a network by the total time taken for transmission of those packets. The higher the throughput then the better the transmission speed of packets that is better in performance of the network. This value metric is expressed in Mbps [Megabits per second].

3.3.3 End–to–End [ETE] Delay

ETE Delay is total time taken for a packet to be transmitted in a network from the source node to destination node. The faster a packet reaches the destination, the better the network, and delay will be less. So, less ETE Delay leads to better network. This value is expressed in milliseconds.

3.4 STEPS TO IMPLEMENT THE CODE

We need to use two types of code to simulate the network and analyze the performance of that network. TCL file [.tcl script] is used to make the network we want like setting up the mobile nodes, properties of those nodes, the application we want to transfer between the nodes, mobility of the nodes, and decide the type of routing protocol and other required protocols in the network. AWK file [.awk script] is used to process the data from the trace file created by running the TCL file successfully. That trace file holds the behavior data of that network at every instance there is a change recorded. Analytical data can be processed and achieved using this AWK file.

3.4.1 TCL Script – To Simulate the Network Environment

1. Setting up the simulation parameters which include channel type, propagation model, network interface type, MAC Type, interface queue type, link-layer type, max packet in IFQ, number of mobile nodes, routing protocol, topology range, and simulation end time.
2. Create a simulator object, and set up the topography object.
3. Create GOD node and open NAM, trace, and xgraph files.
4. Define the finish procedure to properly flush and close the code.

5. Defining the record procedure to record and plot the points as a graph for bandwidth vs time analysis. This records data in opened xgraph file.
6. Create a wireless channel and set up the mobile node parameters.
7. Defining the nodes and setting their random movement to zero and the size of nodes.
8. Setting up the initial node positions in the created topology. [refer Figure 1]
9. Setup the mobility of the nodes and the time they start moving and destination positions.
10. Setting up the FTP application over TCP connection and variant. [2 connections created]
11. Schedule the events and then initiate the program termination procedure.
12. Finally, give a command to run the simulator.

3.4.2 AWK Script – To Process Required Data For Analysis From Trace File

1. In the BEGIN section, declare the variables we will use for the entire awk script like dropped packets, received packets, received packet size, start time, and stop time.
2. Using if loops we can trace our requirements using column 4 which holds event type at a particular instance. Then finding out received, dropped packets, and so on while that middle section runs until it covers every row available in trace file.
3. In the END section, we can calculate ETE delay, PDR, and Throughput using the data collected from the middle section of the code which is executed until every row is considered. ETE delay is a sum of delay [end time – start time] for every packet received. PDR is the ratio percentage of received packets by total sent packets. Throughput is the division total received packet size by a difference of end time and start time.
4. All these data after processing can be printed on the terminal which can be considered for further analysis and performance evaluation of that particular network.

3.5 SIMULATION PROCEDURE TO COLLECT DATA POINTS FOR ANALYSIS

Application is set to be FTP which runs over TCP connections.

1. Set the routing protocol to AODV. The mobility speed of the nodes is set to 5 meters per second and the network is simulated. Similarly set the mobility speeds like 10, 15, 20, 25, and 30 and run the simulation. This needs to be done for all TCP variants [six types].
2. Set the routing protocol to DSDV. Run the network simulation at different mobility speeds being 5, 10, 15, 20, 25, and 30 meters per second and also 6 TCP variants.
3. Set the routing protocol to DSR. Run the network simulation at different mobility speeds and TCP variants similar to AODV and DSDV.
4. Process and collect data using AWK file by running it with the trace files created by running the simulation using TCL files with all the variations in routing protocol and node mobility speeds.

Factors for comparison	: PDR, Throughput, and ETE Delay
TCP Variants	: Tahoe, Reno, New Reno, Vegas, Sack, and Linux
Wireless routing protocols	: AODV, DSDV, and DSR
Mobility speeds	: 5, 10, 15, 20, 25, and 30 meters per second
Total data points collected	: 324 [3 x 6 x 3 x 6]

CHAPTER 4 SOURCE CODE

4.1 TCL SCRIPT FOR NETWORK SIMULATION

4.1.1 AODV Routing Protocol

aodv_wireless.tcl

```
#Setting up the simulation parameters
set val(chan)       Channel/WirelessChannel          ;#Channel Type
set val(prop)       Propagation/TwoRayGround         ;#Radio Propagation Model
set val(netif)      Phy/WirelessPhy                  ;#Network Interface Type
set val(mac)        Mac/802_11                       ;#MAC Type
set val(ifq)        Queue/DropTail/PriQueue          ;#Interface Queue Type
set val(ll)         LL                               ;#Link Layer Type
set val(ant)        Antenna/OmniAntenna              ;#Antenna Model
set val(ifqlen)     50                               ;#Max Packet in IFQ
set val(nn)         10                               ;#Number of Mobile Nodes
set val(rp)         AODV                             ;#Routing Protocol
set val(x)          450                              ;#Topography X Dimension
set val(y)          450                              ;#Topography Y Dimension
set val(stop)       50.0                             ;#Simulation End Time

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

#Setup topography object
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)                            ;#General Operation Director Node

#Open NS and NAM trace files
set tracefile [open aodv_tahoe_5.tr w]
$ns trace-all $tracefile

set namfile [open aodv_tahoe_5.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)

#Open a xgraph file for total bandwidth
set file1 [open aodv_tahoe_5.xg w]

#Defining the 'finish' procedure
proc finish {} {
	global ns tracefile namfile file1
	$ns flush-trace
	close $tracefile
```

```
        close $namfile
        exec xgraph aodv_tahoe_5.xg &
        exec nam aodv_tahoe_5.nam &
        exit 0
}

#Defining a 'record' procedure
proc record {} {
        global sink0 sink1 file1
        set ns [Simulator instance]

        set time 0.1
        set nowtime [$ns now]

        set bw0 [$sink0 set bytes_]
        set bw1 [$sink1 set bytes_]
        set bwt [expr $bw0 + $bw1]

        puts $file1 "$nowtime [expr $bwt / $time * 8 / 1000000]"

        $sink0 set bytes_ 0
        $sink1 set bytes_ 0
        $ns at [expr $nowtime + $time] "record"
}

#Create wireless channels
set chan [new $val(chan)]

#Setting up mobile nodes parameters
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#Defining the nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]

#Random movement and size of nodes
$n0 random-motion 0
$n1 random-motion 0
$n2 random-motion 0
$n3 random-motion 0
$n4 random-motion 0
$n5 random-motion 0
$n6 random-motion 0
$n7 random-motion 0
$n8 random-motion 0
$n9 random-motion 0

$ns initial_node_pos $n0 20
$ns initial_node_pos $n1 20
$ns initial_node_pos $n2 20
$ns initial_node_pos $n3 20
$ns initial_node_pos $n4 20
$ns initial_node_pos $n5 20
$ns initial_node_pos $n6 20
$ns initial_node_pos $n7 20
$ns initial_node_pos $n8 20
$ns initial_node_pos $n9 20

#Coordinates of the nodes
$n0 set X_ 50
$n0 set Y_ 50
$n0 set Z_ 0.0

$n1 set X_ 150
$n1 set Y_ 50
$n1 set Z_ 0.0

$n2 set X_ 250
$n2 set Y_ 50
$n2 set Z_ 0.0

$n3 set X_ 350
$n3 set Y_ 50
$n3 set Z_ 0.0

$n4 set X_ 50
$n4 set Y_ 350
$n4 set Z_ 0.0

$n5 set X_ 150
$n5 set Y_ 350
$n5 set Z_ 0.0
```

```
$n6 set X_ 250
$n6 set Y_ 350
$n6 set Z_ 0.0

$n7 set X_ 350
$n7 set Y_ 350
$n7 set Z_ 0.0

$n8 set X_ 100
$n8 set Y_ 200
$n8 set Z_ 0.0

$n9 set X_ 300
$n9 set Y_ 200
$n9 set Z_ 0.0

#Mobility of the nodes
set val(velocity)   5
$ns at 1.0 "$n0 setdest 350.0 350.0 $val(velocity)"
$ns at 1.0 "$n1 setdest 250.0 350.0 $val(velocity)"
$ns at 1s.0 "$n2 setdest 150.0 350.0 $val(velocity)"
$ns at 1.0 "$n3 setdest 50.0 350.0 $val(velocity)"
$ns at 1.0 "$n4 setdest 350.0 50.0 $val(velocity)"
$ns at 1.0 "$n5 setdest 250.0 50.0 $val(velocity)"
$ns at 1.0 "$n6 setdest 150.0 50.0 $val(velocity)"
$ns at 1.0 "$n7 setdest 50.0 50.0 $val(velocity)"
$ns at 1.0 "$n8 setdest 300.0 200.0 $val(velocity)"
$ns at 1.0 "$n9 setdest 100.0 200.0 $val(velocity)"

#Setting up FTP application over TCP connection
set tcp0 [new Agent/TCP]
set tcp1 [new Agent/TCP]

set sink0 [new Agent/TCPSink]
set sink1 [new Agent/TCPSink]

$ns attach-agent $n0 $tcp0
$ns attach-agent $n3 $tcp1

$ns attach-agent $n7 $sink0
$ns attach-agent $n4 $sink1

$ns connect $tcp0 $sink0
$ns connect $tcp1 $sink1

$tcp0 set packetSize_ 1500
$tcp1 set packetSize_ 1500

set ftp0 [new Application/FTP]
set ftp1 [new Application/FTP]

$ftp0 attach-agent $tcp0
```

```
$ftp1 attach-agent $tcp1

#Scheduling the simulation
$ns at 0.0 "record"
$ns at 0.5 "$ftp0 start"
$ns at 0.5 "$ftp1 start"
$ns at 45.0 "$ftp0 stop"
$ns at 45.0 "$ftp1 stop"

#Termination of program
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"

#Run the simulation
$ns run
```

Speed of the mobile nodes can be changed at code line:

```
set val(velocity)   10
```

This is available at the place where the mobility of the nodes is defined. Changing the value once here will change the velocity of all the nodes.

Setting up different TCP source agents:

TCP Tahoe	: `[new Agent/TCP]`
TCP Reno	: `[new Agent/TCP/Reno]`
TCP New Reno	: `[new Agent/TCP/Newreno]`
TCP Vegas	: `[new Agent/TCP/Vegas]`
TCP Sack	: `[new Agent/TCP/Sack1]`
TCP Linux	: `[new Agent/TCP/Linux]`

Setting up different TCP sink agents concerning TCP source agents:

For TCP Sack	: `[new Agent/TCPSink/Sack1]`
For TCP Linux	: `[new Agent/TCPSink/Sack1/DelAck]`
For all other agents	: `[new Agent/TCPSink]`

4.1.2 DSDV Routing Protocol

Using the same code as the AODV routing protocol and changing a few things.

Change the routing protocol as DSDV in setting up simulation parameters.

```
set val(rp)        DSDV                           ;#Routing Protocol
```

Also, change the velocity of the nodes and TCP variants accordingly for network simulation in those particular cases as shown for the AODV protocol.

4.1.3 DSR Routing Protocol

Using the same code as the AODV routing protocol and changing a few things.

Change the routing protocol as DSR in setting up simulation parameters.

```
set val(rp)        DSR                            ;#Routing Protocol
```

We need to add few more lines to the code after setting up simulation parameters that are shown below. This is to choose the IFQ differently for DSR routing to CMUPriQueue.

```
#Choose IFQ
      if {$val(rp) == "DSR"} {
            set val(ifq) CMUPriQueue
      } else {
            set val(ifq) Queue/DropTail/PriQueue
      }
```

Also, change the velocity of the nodes and TCP variants accordingly for network simulation in those particular cases as shown for the AODV protocol.

4.2 AWK SCRIPT FOR TRACE FILE PROCESSING

analysis.awk

```
#AWK Script to Process Trace File Contents

#BEGIN section begins here
BEGIN {
    #Variable to hold sequence number
    sequenceNo = -1;

    #Counter variable
    Counter = 0;

    #Variable to hold dropped packets
    droppedPackets = 0;

    #Variable to hold received packes
    receivedPackets = 0;

    #Variable to hold received packet size
    receivedSize = 0;

    #Variable to hold start time of simulation
    startTime = 500;

    #Variable to hold end time of sumulation
    stopTime = 0;
}

#Middle section begins here
{
    #Getting sequence number
    #Agent level, Sent Packet, and sequence number
    if($4 == "AGT" && seqno < $6 && $1 == "s")
    {
        sequenceNo = $6;
    }
    #Getting no of received packets
    #Agent level and received packet
    else if($4 == "AGT" && $1 == "r")
    {
        receivedPackets++;
    }
    #Getting no of dropped packets
    #Packet dropped, TCP connection, and size > 512 bytes
    else if($1 == "D" && $7 == "tcp" && $8 > 512)
    {
        droppedPackets++;
    }

    #Getting start time of a packet transfer
    #Agent level and sent packet
    if($1 == "s" && $4 == "AGT")
    {
        start_time[$6] = $2;
    }
    #Getting end time of a packet transfer
    #TCP connection and received packet
    else if($7 == "tcp" && $1 == "r")
    {
        end_time[$6] = $2;
    }
    #Getting end time of a packet transfer
    #TCP connection and dropped packet
    else if($1 == "D" && $7 == "tcp")
    {
        end_time[$6] = -1;
    }

    #Variables to hold column data of a row
    event = $1;
    time = $2;
    node_id = $3;
    packet_size = $8;
    level = $4;

    #Getting start time of network simulation
    #Agent level, sent packet, and size > 512 bytes
    if(level == "AGT" && event == "s" && packet_size >= 512)
    {
        if(time < startTime)
        {
            startTime = time;
        }
    }

    #Getting end time of network simulation
    #Getting received packet size
    #Agent level, received packet, and size > 512
    if(level == "AGT" && event == "r" && packet_size >= 512)
    {
        if(time > stopTime)
        {
            stopTime = time;
        }

        hdr_size = packet_size % 512;
        packet_size -= hdr_size;
        receivedSize += packet_size;
    }
```

```
}

#END section begins here
END {
    #Calculating delay for individual packet
    for(i = 0; i <= sequenceNo; i++)
    {
        if(end_time[i] > 0)
        {
            delay[i] = end_time[i] - start_time[i];
            Counter++;
        }
        else
        {
            delay[i] = -1;
        }
    }

    #Calculating ETE delay of network simulation
    for(i = 0; i < sequenceNo; i++)
    {
        if(delay[i] > 0)
        {
            ete_delay += delay[i];
        }
    }
    ete_delay /= Counter;

    #Printing the required results
    print "";
    #print "Start Time               : " startTime "\n";
    #print "Stop Time                : " stopTime "\n";
    #print "Generated Packets        : " sequenceNo + 1 "\n";
    #print "Received Packets         : " receivedPackets "\n";
    #print "Total Dropped Packets    : " droppedPackets "\n";

    #Calculating PDR
    print "Packet Delivery Ratio    : " receivedPackets / (sequenceNo + 1) * 100"%";
    print "End-to-End [ETE] Delay   : " ete_delay * 1000 " ms";
    #Calculating Throughput in Mbps
    print "Throughput [in Mbps]     : " (receivedSize / (stopTime - startTime)) * (8
/ 1000000) "\n";
}
```

CHAPTER 5 RESULTS

5.1 AODV Protocol

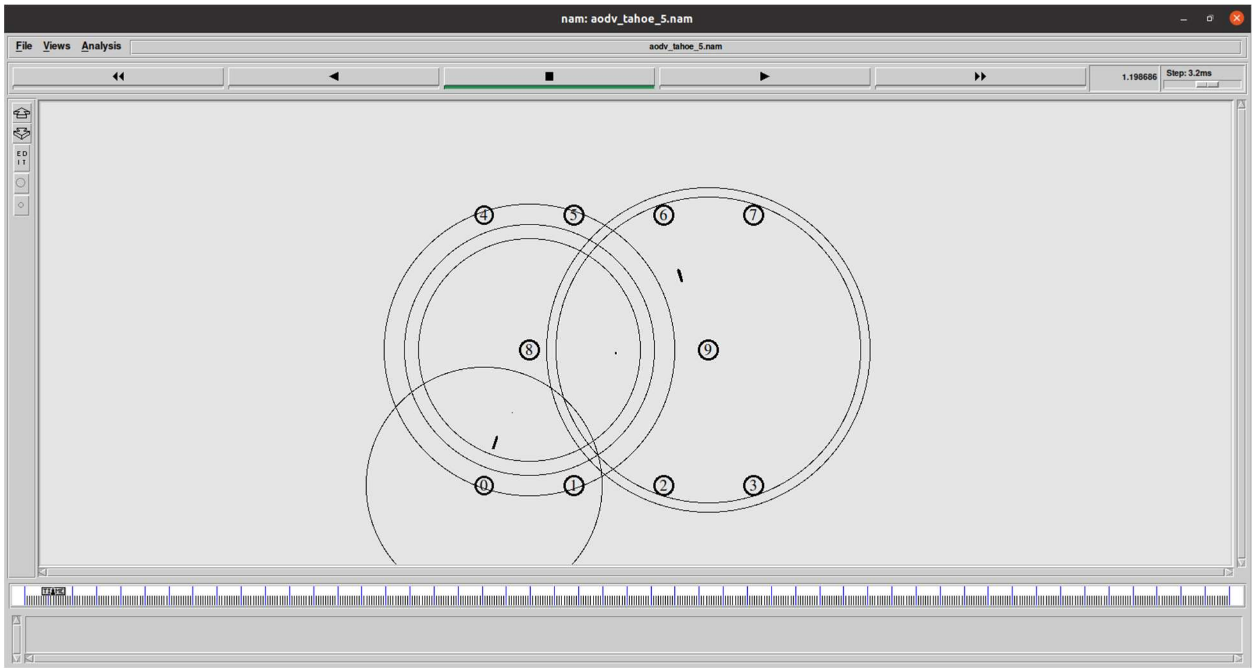


Figure 2. Simulation screen during initial transmission

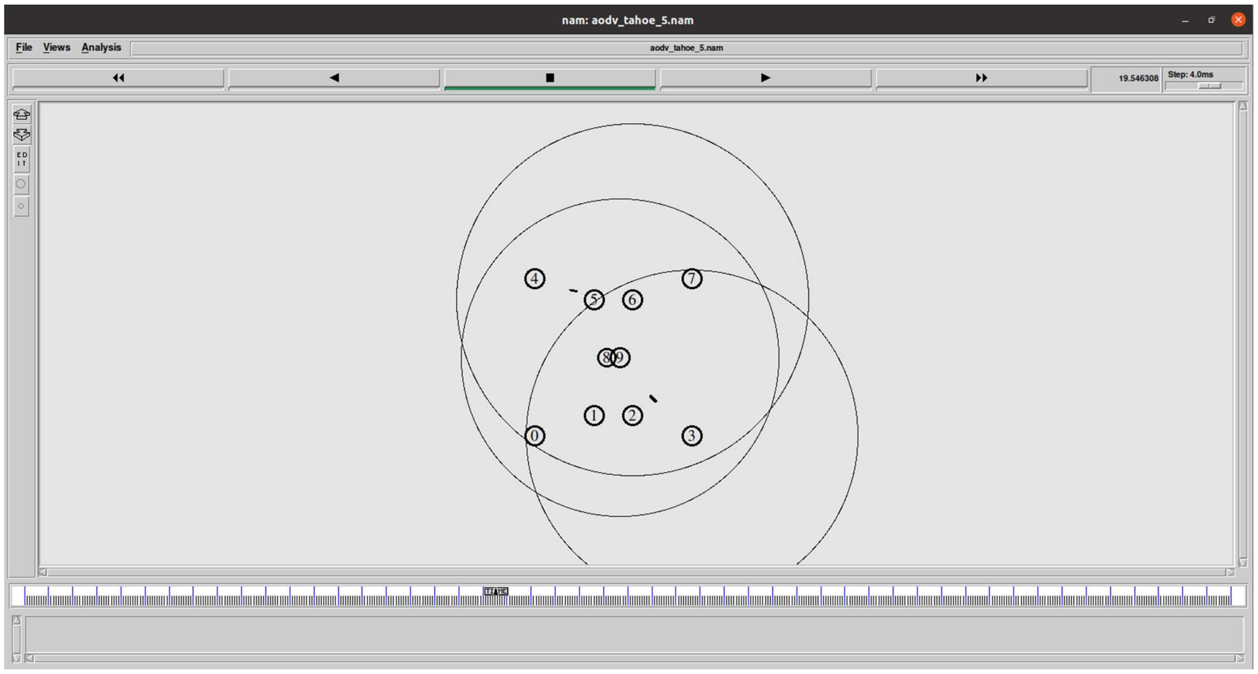


Figure 3. Simulation screen at intermediate time

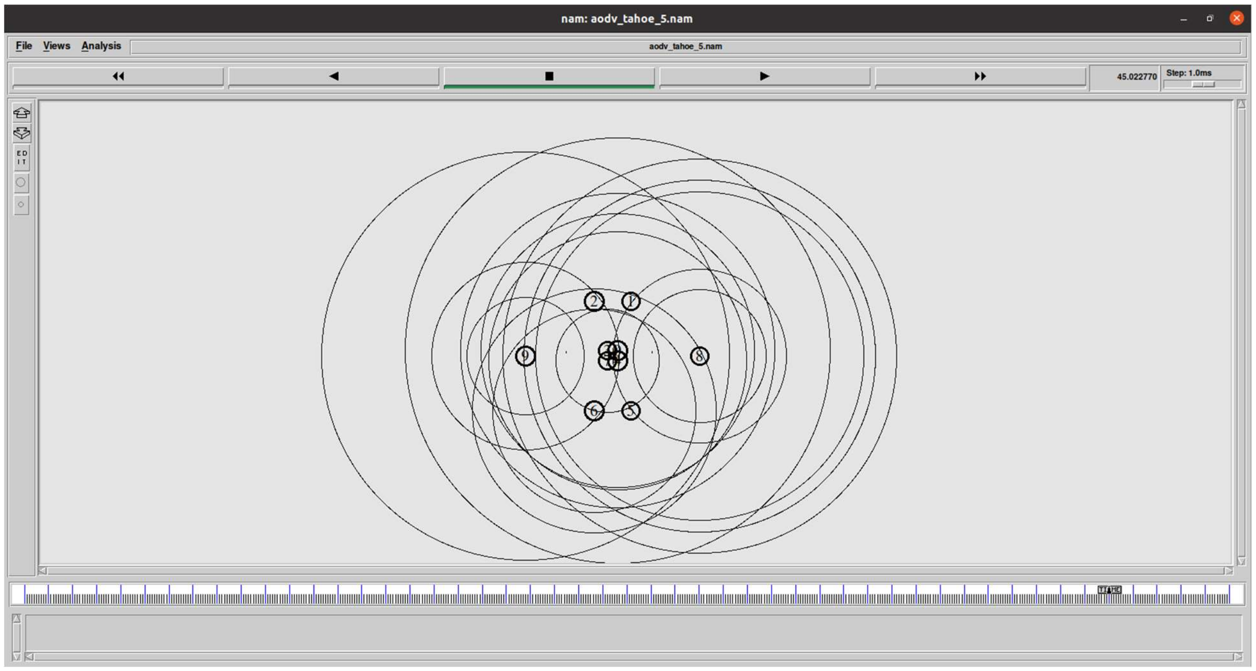


Figure 4. Simulation screen at near finish time

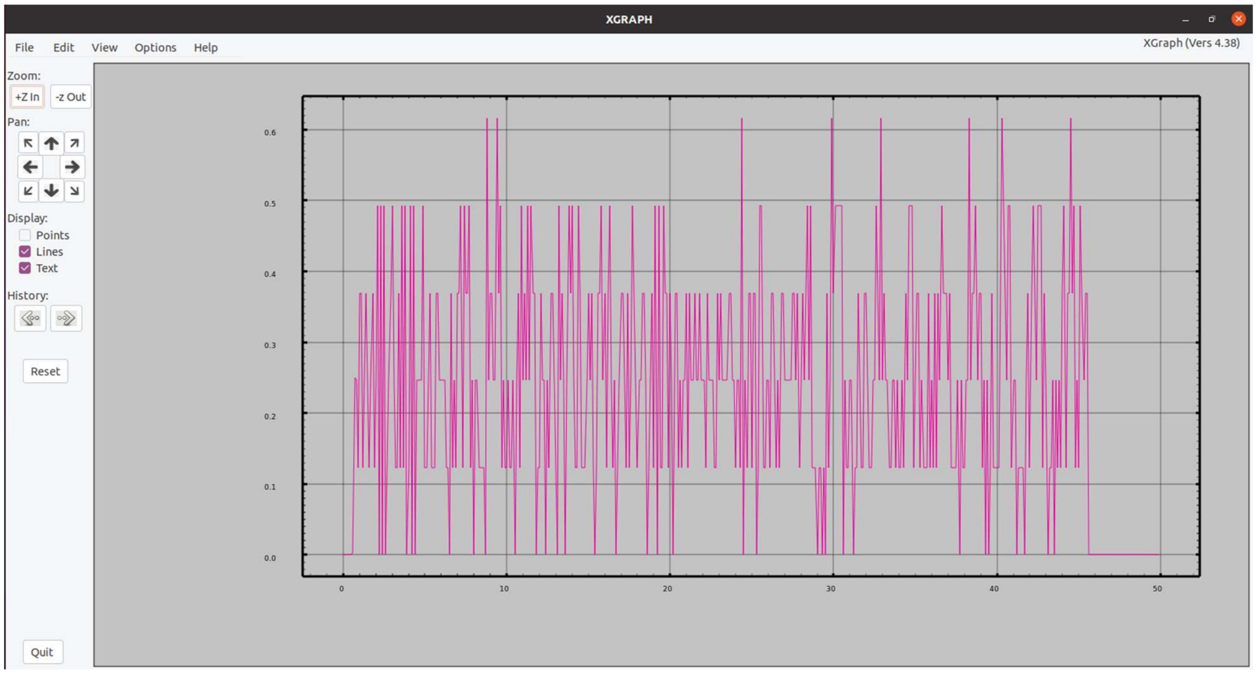
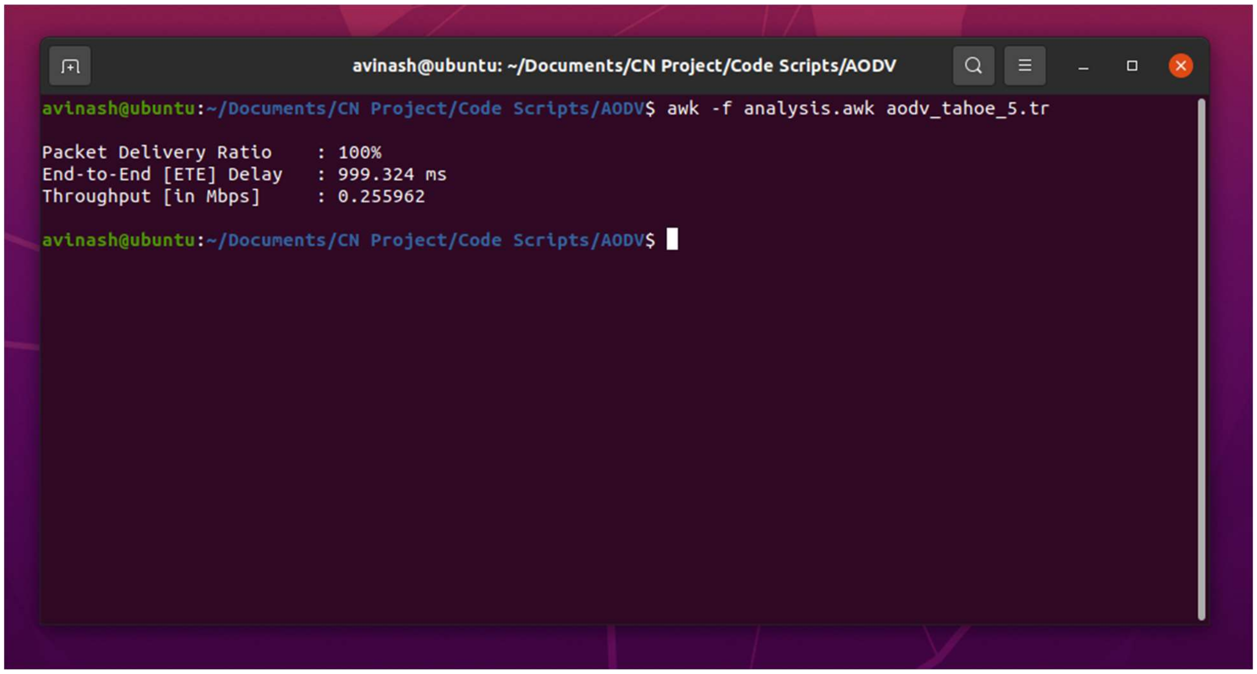


Figure 5. Xgraph file



```
avinash@ubuntu:~/Documents/CN Project/Code Scripts/AODV$ awk -f analysis.awk aodv_tahoe_5.tr

Packet Delivery Ratio     : 100%
End-to-End [ETE] Delay    : 999.324 ms
Throughput [in Mbps]      : 0.255962

avinash@ubuntu:~/Documents/CN Project/Code Scripts/AODV$
```

Figure 6. AWK script processing result

The above screenshots are from a network situation using AODV protocol with TCP Tahoe as TCP variant and mobile nodes motion set to 5 m/s. Similarly, this procedure of running the code and recording data values for each mobile node speed and the TCP variants are performed.

5.2 DSDV and DSR Protocol

As performed for AODV protocol, the same factors like different speeds and TCP variants are considered for both DSDV and DSR routing protocols. All the analytical values like PDR, ETE Delay and Throughput are recorded and will be analyzed in performance evaluation chapter. The above figures are just for representation of how the network is simulated in a particular case, graphs are generated, and analytical values are recorded which are processed by AWK script.

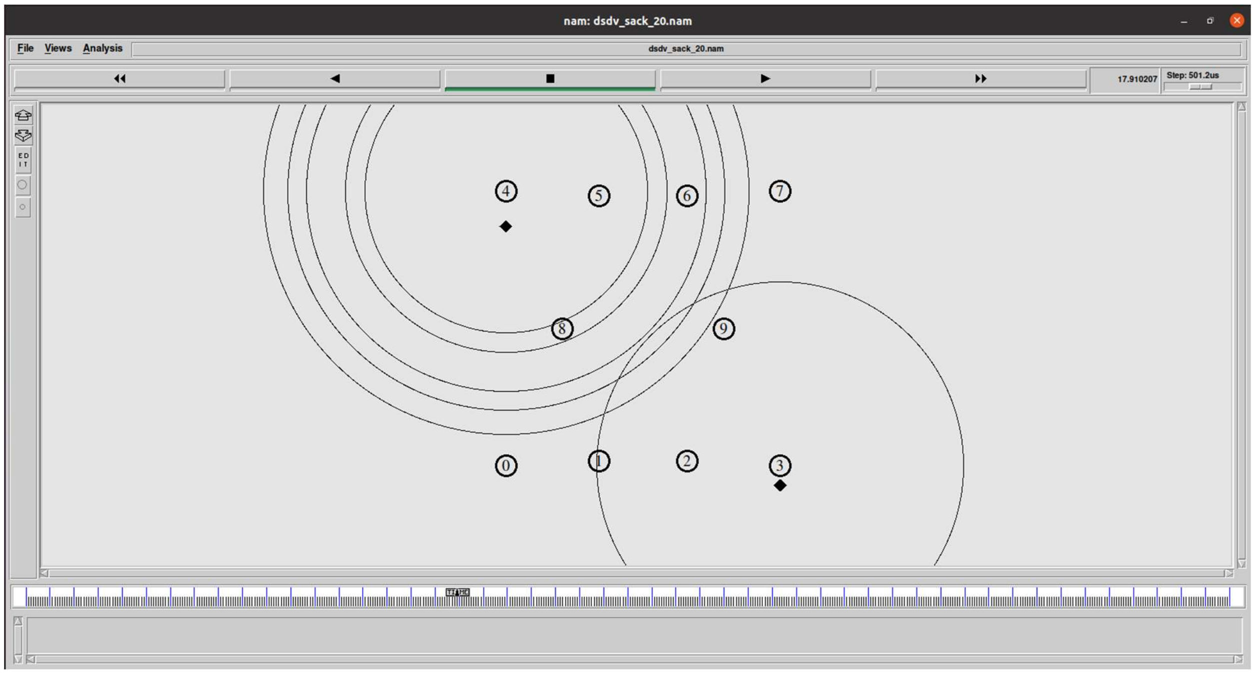


Figure 7. Simulation screen of DSDV routing protocol

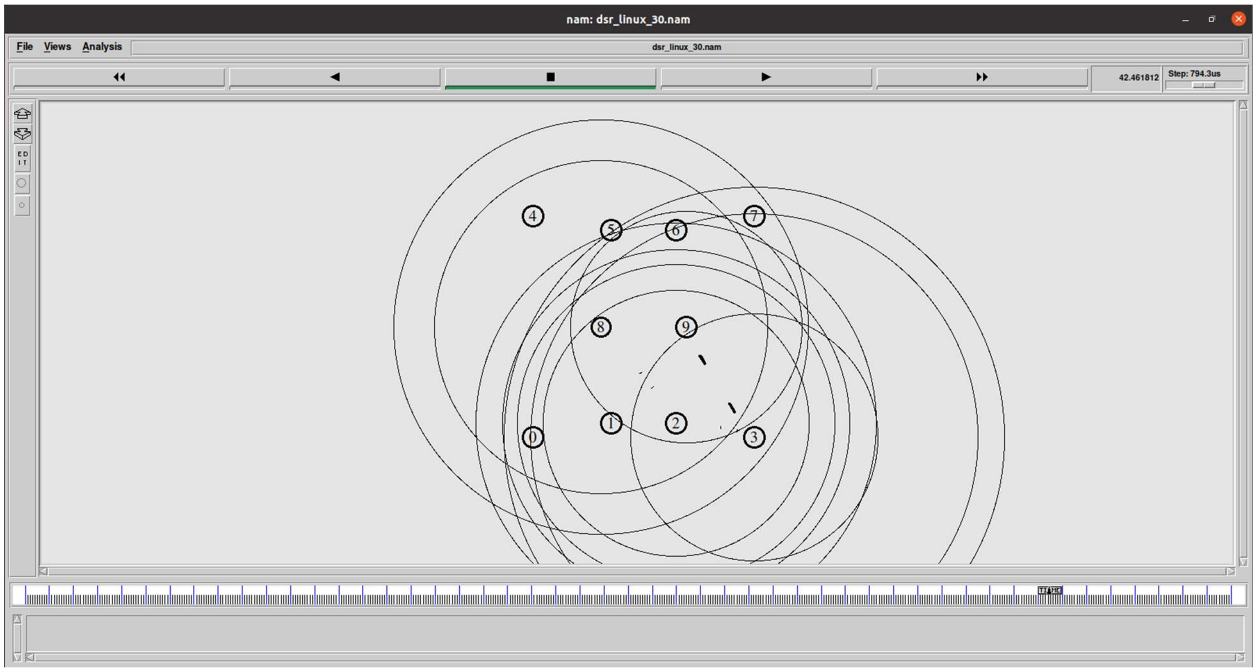


Figure 8. Simulation screen of DSR routing protocol

CHAPTER 6 PERFORMANCE EVALUATION

The proposed wireless ad-hoc network with different routing protocols, TCP variants, and mobile node speeds is simulated in NS-2 simulator.

Here we're going to analyze the performance of different TCP variants in mainly three different routing protocols used. This performance can be affected by also varying the speed of mobile nodes [in m/s]. To analyze the performance, we have considered PDR [in percentage], average throughput [in Mbps], and average ETE delay [in ms].

6.1 TCP VARIANTS PERFORMANCE IN AODV ROUTING PROTOCOL

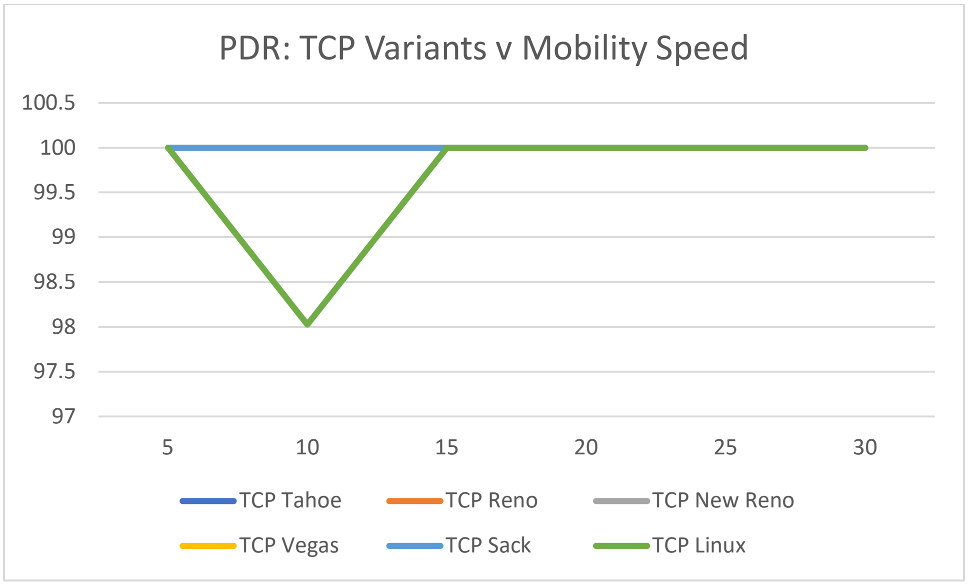


Figure 9. PDR in AODV: TCP Variants v Mobility Speed

In Figure 9, X-axis indicate the speed of mobile nodes in m/s and Y-axis indicate PDR in percentage. It is clearly observed for except TCP Linux, that PDR is 100% for TCP variants irrespective of mobile nodes speed. There's almost no packet loss in transmission.

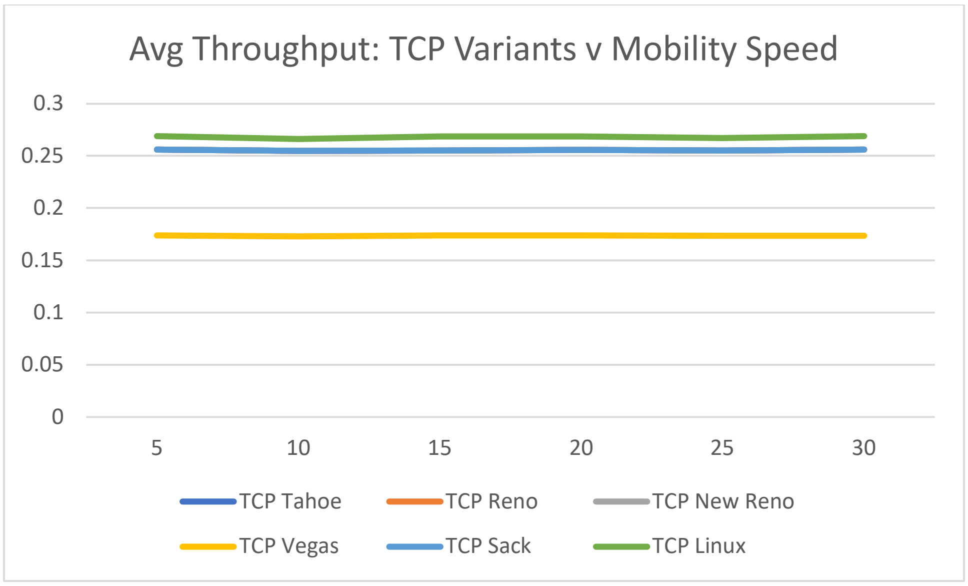


Figure 10. Avg Throughput in AODV: TCP Variants v Mobility Speed

In Figure 10, X-axis indicate the speed of mobile nodes in m/s and Y-axis indicate average throughput in Mbps. Here TCP Tahoe, Reno, New Reno, and Sack have exactly same throughput. TCP Vegas having the lowest throughput and Linux having the highest.

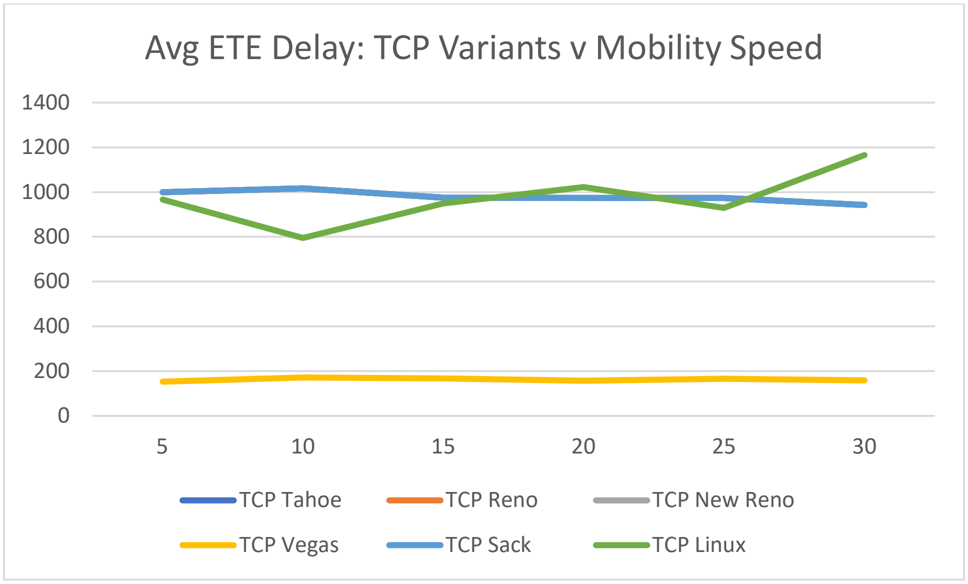


Figure 11. Avg ETE Delay in AODV: TCP Variants v Mobility Speed

In Figure 11, X-axis indicate the speed of mobile nodes in m/s and Y-axis indicate the avg ETE delay in ms. Here TCP Vegas have the lowest avg ETE delay. TCP Tahoe, Reno, New Reno, and Sack have exactly same values and TCP Linux performing almost close to them.

6.2 TCP VARIANTS PERFORMANCE IN DSDV ROUTING PROTOCOL

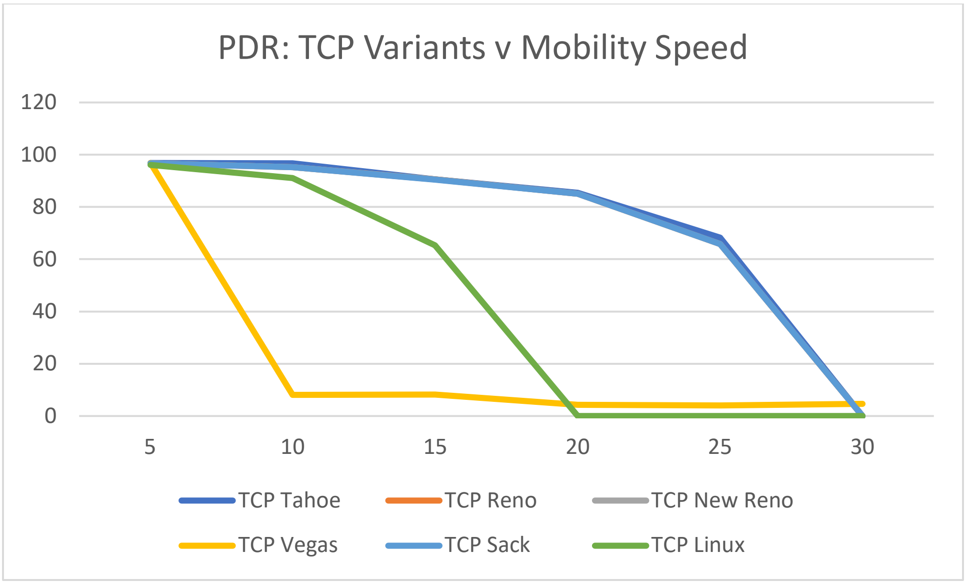


Figure 12. PDR in DSDV: TCP Variants v Mobility Speed

TCP Vegas have the poorest PDR with increase in mobility speed and then Linux fails to hold the PDR up high. TCP Tahoe, Reno, New Reno, and Sack perform almost closely. TCP Vegas can be ruled out to be used in DSDV routing protocol if mobile node speed increases as that huge packet loss at receiver's end will not be useful.

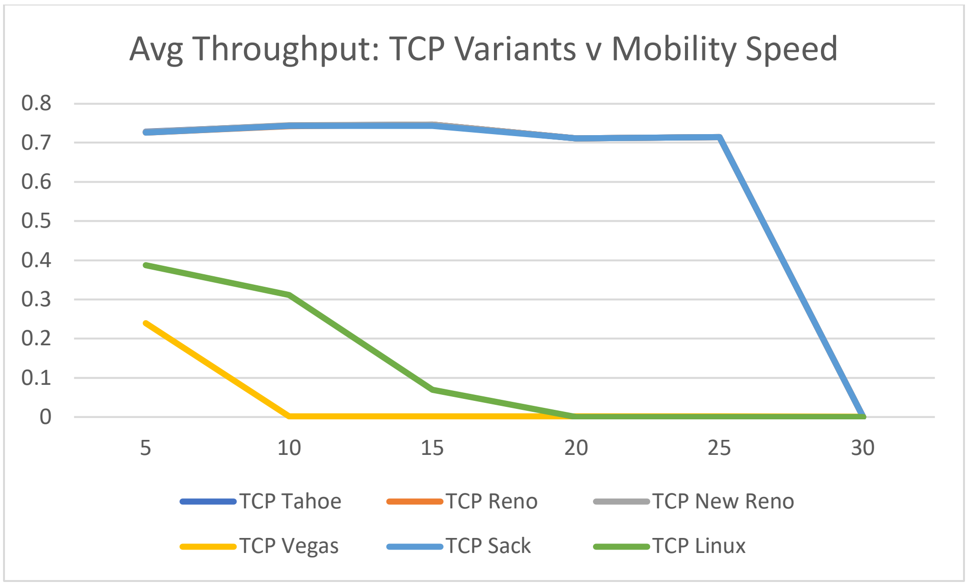


Figure 13. Avg Throughput in DSDV: TCP Variants v Mobility Speed

The same trend as in PDR can be observed even in the case of avg throughput. Both TCP Vegas, and Linux start to fail after the mobile nodes speed increases from 10 m/s. Other TCP variants perform almost the same which completely fails from 25 m/s speed in this network scenario.

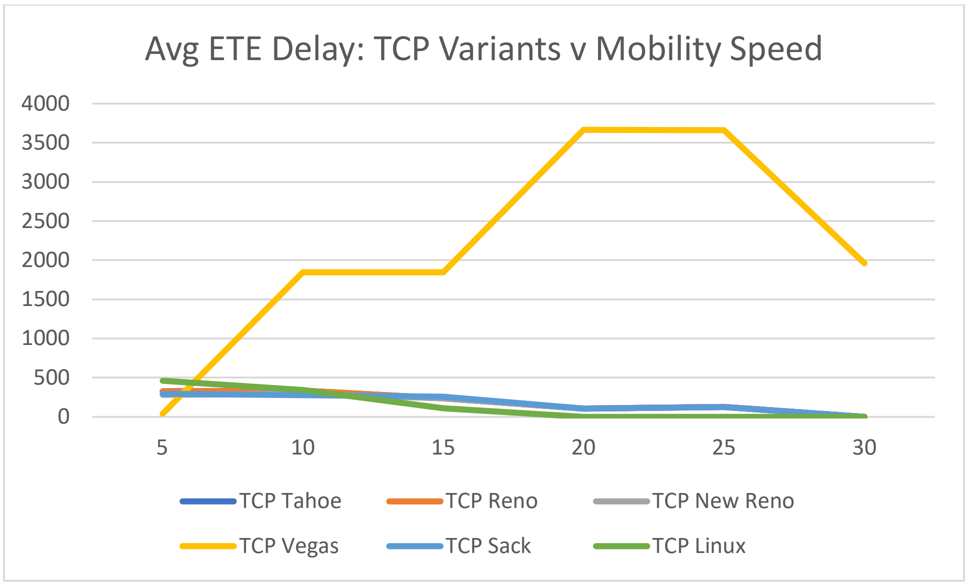


Figure 14. Avg ETE Delay in DSDV: TCP Variants v Mobility Speed

As expected, TCP Vegas performs very badly with high avg ETE delay. All other TCP variants perform similarly in avg ETE delay. So, TCP Tahoe, Reno, New Reno and Sack are comparatively better and performs similarly in this DSDV routing protocol.

6.3 TCP VARIANTS PERFORMANCE IN DSR ROUTING PROTOCOL

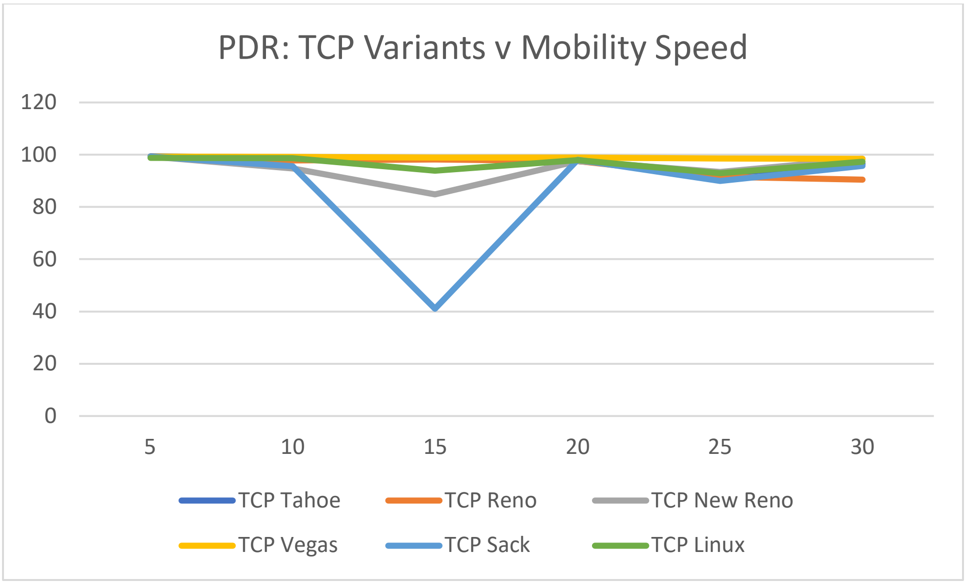


Figure 15. PDR in DSR: TCP Variants v Mobility Speed

TCP Sack comparatively performs weaker in terms of PDR than other TCP variants. TCP Vegas outperforms all other TCP variants having consistent PDR across different mobile node speeds. All other TCP variants are not consistent across the speeds and no proper trend can be observed. As a whole we can consider TCP Vegas and then TCP Linux to have low packet loss at receiver's end which is always preferred

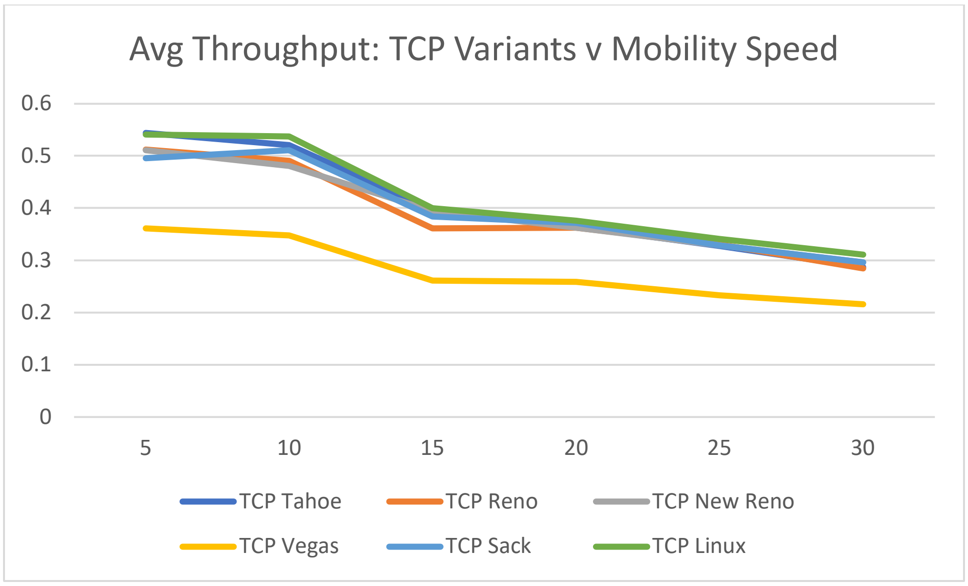


Figure 16. Avg Throughput in DSR: TCP Variants v Mobility Speed

All the TCP variants decreased throughput with the increase in speed of mobile nodes. TCP Vegas having the lowest throughput but keeping the PDR high. Other variants perform almost similarly with higher throughput than Vegas. TCP Linux have the highest avg throughput.

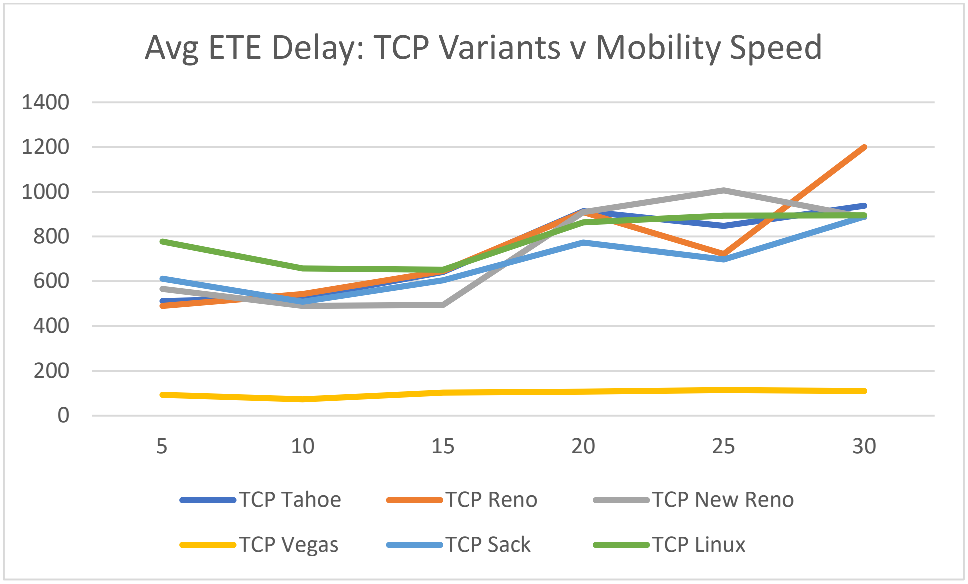


Figure 17. Avg ETE Delay in DSR: TCP Variants v Mobility Speed

TCP Vegas have much lower avg ETE delay than all other TCP variants. The avg ETE delay of Vegas also don’t tend to increase much with increase in speed of the mobile nodes. Whereas all other TCP variants got their avg ETE delay increased with increase in mobility speed. TCP Reno is the worst performing in terms of avg ETE delay. As a whole, TCP Vegas is very preferrable if we need less packet loss and avg ETE delay ignoring avg throughput in DSR routing protocol.

CHAPTER 7 CONCLUSION

Analyzing the Performance of different TCP Variants in Wireless Ad-hoc Networks using different Routing Protocols is done successfully. Considering the factors like PDR, Average Throughput, and Average ETE Delay using different routing protocols like AODV, DSDV, and DSR with different mobile node speeds. By evaluation of analytical values, it is clearly observed that TCP Vegas is best in AODV routing protocol with no packet loss and very low average ETE delay but slightly lower average throughput than other TCP variants. TCP Tahoe, Reno, New Reno, and Sack are top performers in DSDV routing protocol but all protocols tend to fail quickly with increase in speed of mobile nodes. Again, TCP Tahoe is the best performer in DSR routing protocol with consistent PDR, and very low average ETE delay but slightly lower average throughput. By this analysis, we can understand how TCP variants perform with change in routing protocols and varying the speed of mobile nodes.

CHAPTER 8 REFERENCES

S. U. Shenoy, M. S. Kumari, U. K. K. Shenoy and N. Anusha, "Performance analysis of different TCP variants in wireless ad hoc networks," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 891-894, doi: 10.1109/I-SMAC.2017.8058308.

Bazi, K., & Nassereddine, B. (2020, March). Comparative analysis of TCP congestion control mechanisms. In Proceedings of the 3rd International Conference on Networking, Information Systems & Security (pp. 1-4).

S. Patel, Y. Shukla, N. Kumar, T. Sharma and K. Singh, "A Comparative Performance Analysis of TCP Congestion Control Algorithms: Newreno, Westwood, Veno, BIC, and Cubic," 2020 6th International Conference on Signal Processing and Communication (ICSC), 2020, pp. 23-28, doi: 10.1109/ICSC48311.2020.9182733.

Kadhim, A. J., & Naser, J. I. (2020). PERFORMANCE OF ROUTING PROTOCOLS WITH CONGESTION ALGORITHMS OF TCP VARIANTS IN MANET. Telecommunications and Radio Engineering, 79(3).

D. C. Dobhal and S. C. Dimri, "Performance evaluation of proposed-TCP in Mobile Ad Hoc Networks (MANETs)," 2016 International Conference on Inventive Computation Technologies (ICICT), 2016, pp. 1-6, doi: 10.1109/INVENTIVE.2016.7824797.

Istikmal, A. Kurniawan and Hendrawan, "Performance analysis of routing and congestion control cooperation in wireless mobile ad hoc networks," 2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2015, pp. 24-29, doi: 10.1109/ICCEREC.2015.7337048.

P. K. Meher and P. J. Kulkarni, "Analysis and Comparison of Performance of TCP-Vegas in MANET," 2011 International Conference on Communication Systems and Network Technologies, 2011, pp. 67-70, doi: 10.1109/CSNT.2011.21.

S. Waghmare, P. Nikose, A. Parab and S. J. Bhosale, "Comparative analysis of different TCP variants in a wireless environment," 2011 3rd International Conference on Electronics Computer Technology, 2011, pp. 158-162, doi: 10.1109/ICECTECH.2011.5941878.

L. Dong and S. Liu, "Research on TCP Fairness Improvement over Wireless Ad Hoc Networks," 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, 2010, pp. 293-296, doi: 10.1109/DCABES.2010.66.

Abdulatif, Hamidah Hassan Mahmoud. Performance Analysis of Routing Protocols over Ad-Hoc Networks for TCP and HTTP. Diss. University of Khartoum.

L. Cao, J. Tao, H. Shen and G. Bai, "Simulation Study of TCP Performance over Multi-Hop Ad Hoc Networks," 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010, pp. 1-4, doi: 10.1109/WICOM.2010.5601229.

G. Changqing, W. Xiaoyan and B. Xiaoxia, "Improvement of TCP Congestion Control Algorithm in Ad Hoc Networks," 2007 International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, 2007, pp. 197-199, doi: 10.1109/MAPE.2007.4393579.

Alfredsson, Stefan. TCP in wireless networks: Challenges, optimizations and evaluations. Diss. Karlstads universitet, 2005.

A. Jain, A. Pruthi, R. C. Thakur and M. P. S. Bhatia, "TCP analysis over wireless mobile ad hoc networks," 2002 IEEE International Conference on Personal Wireless Communications, 2002, pp. 95-99, doi: 10.1109/ICPWC.2002.1177253.