PROJECT REPORT ON

Performance Analysis of TCP Variants Using AODV, DSDV, DSR Routing Protocols in MANETs

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

BACHELOR OF TECHNOLOGY

Submitted by

123003269- VISHNU NANDAKUMAR - 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

<u>BONAFIDE CERTIFICATE</u>

Certified that this project work entitled **"Performance Analysis of TCP Variants Using AODV, DSDV, DSR Routing Protocols in MANETs"** submitted to the Shanmugha Arts, Science, Technology & Research Academy (SASTRA Deemed to be University), Tirumalaisamudram - 613401 by Sai Mahathi P V N(122003214), 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.Umamakeswri, 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 School of Computing.

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

Transmission Control Protocol(TCP) is one of the most reliable protocols in the Internet protocol suite. It provides a reliable end-to-end packet delivery from source to destination. On the other hand, MANET is a decentralized type of wireless network where every mobile node acts as a router and host in an Adhoc wireless network without having a fixed infrastructure. It was found that using an unmodified TCP on MANETs, failed to give the same performance as in wired networks. So different variants of TCP were introduced. Therefore, comparative analysis on the performance of various TCP variants on different routing protocols in MANET could help to choose the right TCP variant in different situations. In this analysis, we will compare the different TCP variants such as TCP, TCP Tahoe, TCP Reno, TCP newReno, TCP Vegas, on various routing protocols in MANETs such as Table Driven proactive and On-Demand Driven reactive routing protocols. In proactive and Reactive routing protocols, the comparison on DSDV , DSR, AODV respectively using TCP variants will be analysed. The simulation will be carried out in the NS2 Simulator and we will compare performance such as throughput, packet delivery ratio, delay, packet drop, number of acknowledgments of various TCP variants using routing protocols in MANETs.

KEY WORDS: Mobile Ad Hoc Networks (MANETs); Ad-hoc On-Demand Distance Vector (AODV); Opportunistic routing; Position based routing; Hybrid routing protocols; Greedy forwarding scheme; Link quality; Node Density; Packet Delivery ratio (PDR); Throughput; End-to-end delay

Source Code:- https://github.com/Vishnu2001RV/CN_Project/tree/main/Project

List of Tables

List of Figures

NOTATIONS

NOTATION	DESCRIPTION
totalTIme	Total Time of Execution
AvgThroughput	Average ThroughPut
Delay	End to end Delay
Sent Packets	Number of Packets sent
ReceivedPackets	Number of Packets received

ABBREVIATION

MANET	Mobile Ad Hoc Networks
AODV	Ad-hoc On-Demand Distance Vector
DSR	Packet Delivery ratio
DSDV	Expected Transmission Count
RREQ	Route Request Packet
RREP	Route Reply
RERR	Route error
AIMD	additive-increase/multiplicative-decrease
CWND	Congestion window
RTO	Recovery Time Objective
RTT	Round Trip Time
MSS	Maximum Segment Size

CHAPTER 1

INTRODUCTION

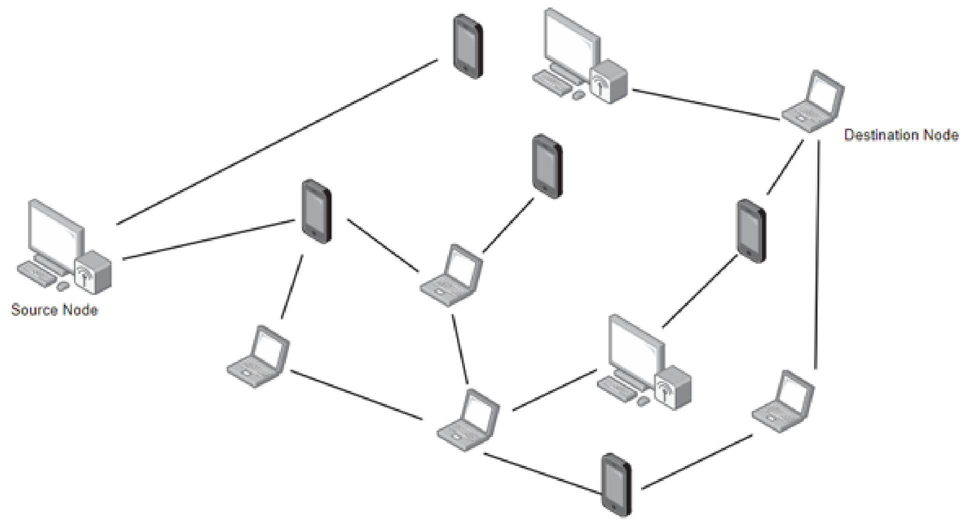


Figure 1.1 Mobile Ad Hoc Network Architecture

A Mobile Ad Hoc Network (MANET) is a system of interconnected wireless nodes which communicate over bandwidth-constrained wireless links. The characteristics of MANET include Dynamic Topologies, Bandwidth constrained, variable capacity links, Autonomous Behavior, Energy Constrained Operation etc. MANET consists of reactive and proactive routing protocols.

Destination Sequenced Distance Vector Routing Protocol (DSDV) is an example of proactive routing protocol which works on the Bellman-ford routing algorithm. The advantage of DSDV is that it requires less delay in the path setup process.

Ad-hoc On-demand Distance Vector (AODV) is a reactive routing protocol where it only keeps routes that are on demand.

Dynamic source routing protocol (DSR) is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach.

The specific contributions of the proposed routing protocol are as follows:

- Studying the performance impact of various tcp variants using MANETs routing protocol on different scenarios.
- Studying the impact of varying the number of nodes and node mobility.

CHAPTER 2

2.1 MANET ROUTING PROTOCOLS

MANET is a decentralized type of wireless network where every mobile node acts as a router and host in an Adhoc wireless network without having a fixed infrastructure. It was found that using an unmodified TCP on MANETs, failed to give the same performance as in wired networks. So different variants of TCP were introduced.

2.1.1 Proactive routing protocol.

Proactive routing protocol sends updates frequently by sending broadcast messages.

The Proactive routing protocols discussed in this report is the DSDV routing protocol.

2.1.2 Reactive routing protocol

Reactive routing protocol updates the routing table on-demand. Some Reactive routing protocols discussed in this report are AODV and DSR routing protocols.

2.1.1 AODV

It is a reactive routing protocol which uses an on-demand approach to find the ideal routes. For establishing the node AODV uses four types of messages namely RREQ,RREP,RERR and hello message.

2.1.1.1 Advantages of AODV

In AODV the route discovery process is on demand. Which is more efficient in the dynamic nature of mobile ad-hoc networks.

2.1.1.2 Disadvantages of AODV

Periodic check of routes is not done due to its demand nature so, there is a delay after discovery of the new route.

2.2.2 DSDV

DSDV is a table driven proactive routing protocol. Its algorithm is based on the number of hops to reach the destination. The protocol has 3 main features : to avoid loops, count to infinity problems and to reduce high routing overhead.

2.2.2.1 Advantages of DSDV

DSDV is best for adhoc networks with a very small number of nodes.

2.2.2.2 Disadvantages of DSDV

It requires regular updating of its routing tables, which uses up the battery and small amount of bandwidth.

2.2.3 DSR

DSR is similar to AODV routing protocol, except that it uses source routing instead of relying on the routing table at each intermediate device.

2.2.3.1 Advantages of DSR

It is able to reduce the overhead and collision by utilizing the route cache information efficiently and therefore creating routes only when necessary.

2.2.3.2 Disadvantages of DSR

The delay is higher than the table driven protocol and its route maintenance mechanism does not locally repair the broken links.

2.3 TCP

The TCP is one of the most reliable transport layer protocols. The reliability depends on the acknowledged data packets that are sent from destination to source node. It implements a 3-way handshake between the source and the destination node. Due to heavy traffic from the acknowledgements, the TCP uses a window based congestion mechanism where the window size is adjusted to adapt to congestion. It uses timeouts or three duplicate acknowledgements for finding out losses. It also uses additive increase/multiplicative decrease (AIMD) and slow start to achieve congestion avoidance.

2.3.1 Additive Increase/Multiplicative decrease(AIMD)

It is the feedback control algorithm for TCP congestion control. When there is no congestion detected it uses linear growth and when congestion is detected it uses exponential reduction .

$$w(t+1) = \begin{cases} w(t) + a & \text{if congestion is not detected} \\ w(t) \times b & \text{if congestion is detected} \end{cases}$$

figure 2.1 Mathematical formula for congestion window size where (a>0) additive increase and (0<b<1)multiplicative decrease factor.

2.3.2 Slow Start Mechanism

It is a strategy used by TCP to avoid forwarding more data into the network which causes congestion. Slow Start begins with initial CWND, which later on increases by a value of 1 MSS for each received ACK. Until the detection of packet loss transmission rate is increased by Slow Start Mechanism. Slow start assumes that unacknowledged segments are due to network congestion. On the contrary other factors such as poor data link layer quality could also result in unacknowledged segments, slow start performs poorly in such scenarios.

Ssthresh is used to find out if slow start or congestion control algorithm is to be used.
If the CWND reaches ssthresh, TCP changes to a congestion avoidance algorithm. For each RTT it is increased by 1MSS

2.4 TCP VARIANTS

It was found that using an unmodified TCP on MANETs, failed to give the same performance as in wired networks. So different variants of TCP were introduced.

TCP Variants include, TCP Tahoe, TCP Reno, TCP NewReno, TCP Vegas.

2.4.1 TCP TAHOE

Tcp Tahoe considers RTO and 3 duplicate ACKs as packet loss events. TCP Tahoe on receiving 3 duplicate TCP starts to perform fast retransmission and sets the Ssthresh to half the current CWND and reduces the CWND to 1 MSS and resets the slow start state.

In TCP Tahoe when packet loss occurs, the retransmit is sent and half of the current CWND is saved as ssthresh and slow start begins again from its initial CWND.

2.4.2 TCP Reno

TCP Reno is moreover the same as TCP Tahoe. The difference here is on how TCP Reno deals with duplicate ACKs.

Here when receiving 3 Duplicate packets TCP Reno starts to perform fast retransmission and skips the slow start by reducing the congestion window by half rather than setting it to 1MSS.

The congestion window size for TCP Reno is as figure 2.2.

$$S_{new} = \frac{S_{cw}}{2} + 3 * MSS$$

Figure 2.2 Congestion window size.

Thus when there is a heavy packet loss, the performance of TCP Reno decreases.

2.4.3 TCP NewReno

TCP NewReno is an improved version from TCP Reno. During multiple packet loss, the window may reduce too much which has a negative impact on performance, so TCP New Reno does not exit fast-recovery and wont reset ssthresh to half immediately. During the fast-recovery phase, to keep the transmission window full, a new unset packet is sent from the end of the congestion window for every duplicate ACK that is returned.

The Problem with NewReno is that it sometimes enters fast recovery even though there is no packet loss and only reordering of more than 3 packet sequence numbers has taken place. Thus there is an increase of needless transmission. The next problem with TCP NewReno is that it can retransmit a maximum of one loss packet per RTT due to which available bandwidth is unutilized.

2.4.4 TCP Vegas

In the Past years TCP Vegas has been able to attain high throughput, along with lesser retransmission than TCP Reno. Based on increasing RTT value of data segments, Tcp Vegas is able to predict the congestion at early stages i.e. it detects expected traffic rather than actual traffic and detects actual packet drop. Due to its slow start and modified congestion avoidance algorithm with decreased retransmit, the congestion window has not decreased to a great extent. Its performance mainly decreased during rerouting.

CHAPTER 3

PROPOSED FRAMEWORK

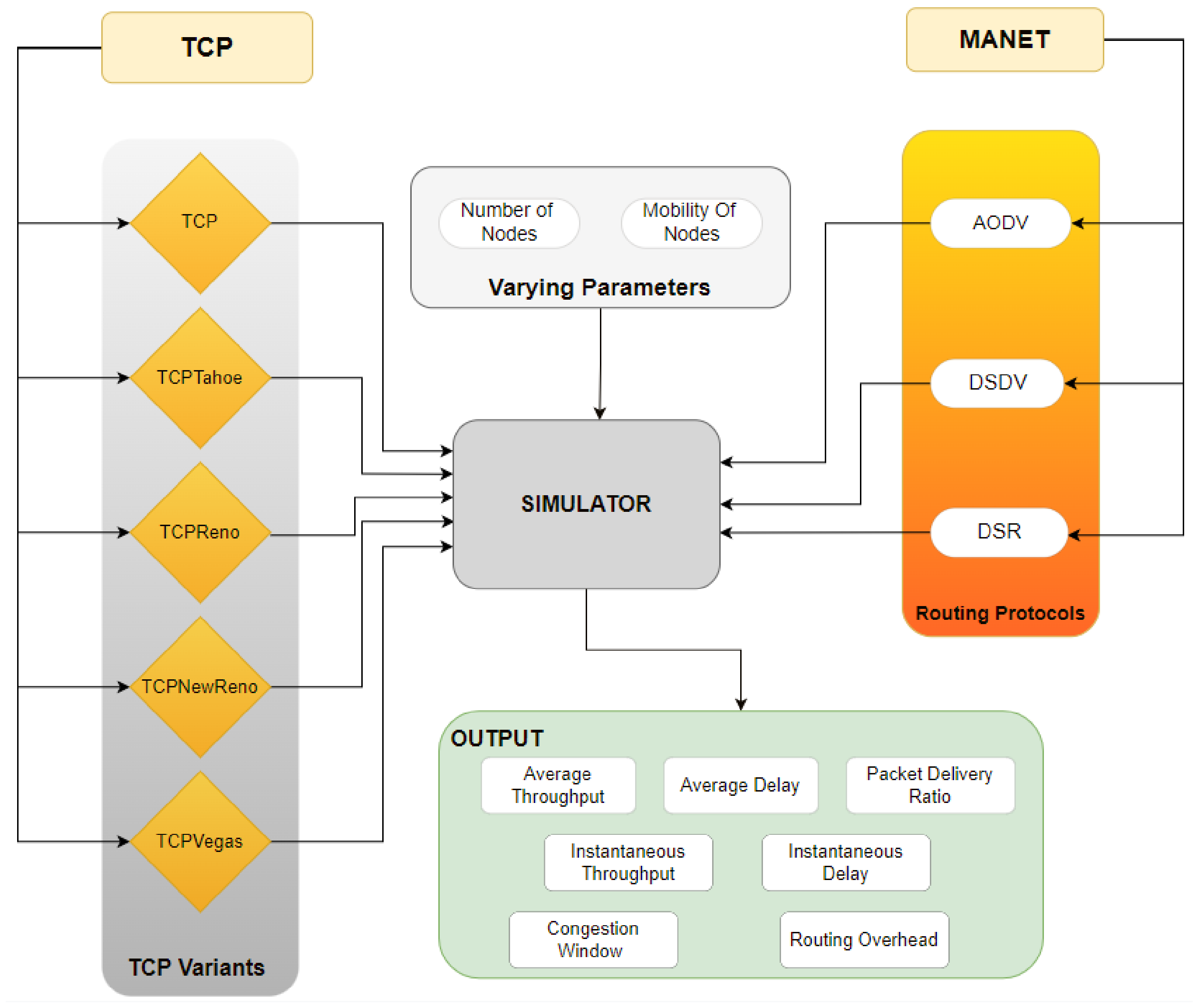


Figure 3.1 Flow diagram for proposed framework.

From the above proposed diagram we could compare 5 different TCP Variants over three MANETs routing protocol. Here we also change the Number of Nodes and Mobility of the Nodes to understand more on how these TCP Variants work with these routing protocols

The mathematical formulas used are:-

$$\text{Average End to End Delay} = \frac{\text{Sum of all End to End Delays}}{\text{Total number of Packets}}$$

$$\text{Instantaneous End to End Delay} = \frac{\text{Sum of all End to End Delay(in 25 sec)}}{\text{Total number of Packets Received in 25 sec}}$$

$$\text{Average ThroughPut} = \frac{\text{Total Number of packets * Packet Size(in bits)}}{\text{(End Time - Start Time)}}$$

$$\text{Average ThroughPut} = \frac{\text{Total Number of packets(in 25 sec) * Packet Size(in bits)}}{\text{25sec}}$$

$$\text{Packet Delivery Ratio} = \frac{\text{Number of Packets delivered Successfully}}{\text{Total Number of Packets}}$$

$$\text{Routing Overhead} = \frac{\text{Number of Forwarded Packets}}{\text{Total Number of Packets}}$$

CHAPTER 4
SOURCE CODE

generator.tcl

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
#set val(nn)     10                        ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      2453                       ;# X dimension of topography
set val(y)      1939                       ;# Y dimension of topography
set val(stop)   200.0                      ;# time of simulation end

puts "CHOOSE From the given CASE"
puts "1. 10Nodes"
puts "2. 30Nodes"
puts "3. 60Nodes"
set case [gets stdin];
if {$case == 1} {
   set root "10Nodes" ;
   file mkdir $root  ;
} elseif {$case == 2} {
   set root 30Nodes ;
```

```
    file mkdir $root  ;
} elseif {$case == 3} {
    set root 60Nodes ;
    file mkdir $root  ;
} else {
    puts "Wrong Choice"
    exit 2
}
puts "Enter the TCP Agent in mobile networking";
puts "1. TCP"
puts "2. TCPTahoe"
puts "3. TCPReno"
puts "4. TCPNewReno"
puts "5. TCPVegas"
set tcpAgent [gets stdin];
#Setup a TCP connection
if {$tcpAgent == 1} {
    set filename "TCP_"
} elseif {$tcpAgent == 2} {
    set filename "TCPTahoe_"
} elseif {$tcpAgent == 3} {
    set filename "TCPReno_"
} elseif {$tcpAgent == 4} {
    set filename "TCPNewReno_"
} elseif {$tcpAgent ==5} {
    set filename "TCPVegas_"
} else {
    puts "Wrong Choice"
    exit 2
}
puts "Enter the Routing Agents in mobile networking"
puts "1. AODV"
puts "2. DSDV"
puts "3. DSR"
set routingProtocol [gets stdin];
if {$routingProtocol == 1} {
    set path ${root}/temp/AODV ;
    file mkdir $path ;
    set val(ifq)    Queue/DropTail/PriQueue;
    set val(rp)     AODV                  ;# routing protocol
    set filename "${filename}AODV";
    set tracefile [open "${path}/${filename}.tr" w];
    set namfile [open "${path}/${filename}.nam" w];
} elseif {$routingProtocol == 2} {
    set path ${root}/temp/DSDV ;
    file mkdir $path ;
    set val(ifq)    Queue/DropTail/PriQueue;
    set val(rp)     DSDV                  ;# routing protocol
    set filename "${filename}DSDV";
    set tracefile [open "${path}/${filename}.tr" w];
    set namfile [open "${path}/${filename}.nam" w];
} elseif {$routingProtocol == 3} {
    set path ${root}/temp/DSR ;
    file mkdir $path ;
    set val(ifq)    CMUPriQueue;
    set val(rp)     DSR                   ;# routing protocol
    set filename "${filename}DSR";
    set tracefile [open "${path}/${filename}.tr" w];
    set namfile [open "${path}/${filename}.nam" w];
}  else {
    puts "Wrong choice"
    exit 2
}
if {$case == 1} {
    set starting "_0_"
    set ending "_9_"
    source source1_10Nodes.tcl
} elseif {$case == 2} {
    set starting "_0_"
    set ending "_29_"
    source source1_30Nodes.tcl
} elseif {$case == 3} {
    set starting "_0_"
    set ending "_59_"
    source source1_60Nodes.tcl
} else {
    puts "Wrong Choice"
    exit 2
}
#===================================
#       Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile filename path starting ending
    $ns flush-trace
    close $tracefile
    close $namfile
    file mkdir $path/cwnd
    file mkdir $path/inst/TP
    file mkdir $path/inst/E2ED
    exec nam "${path}/${filename}.nam" &
    exec gawk -v startNode=${starting} -v endNode=${ending} -f inst_ThroughPut.awk
"${path}/${filename}.tr" > "${path}/inst/TP/${filename}_inst_TP.tr"
    exec gawk -v startNode=${starting} -v endNode=${ending} -f inst_E_2_E_Delay.awk
"${path}/${filename}.tr"  > "${path}/inst/E2ED/${filename}_inst_E2ED.tr"
```

```
    exec gawk -v startNode=${starting} -v endNode=${ending} -f avg_ThroughPut.awk
"${path}/${filename}.tr" &

    exec gawk -v startNode=${starting} -v endNode=${ending} -f PDR_Avg_E_2_E.awk "${path}/${filename}.tr"
&

    exec gawk -f congestion_window.awk "${path}/${filename}.tr" > "${path}/cwnd/${filename}_cwnd.tr"

    exit 0

}

for {set i 0} {$i < $val(nn) } { incr i } {

    $ns at $val(stop) "\$n$i reset"

}

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

$ns at $val(stop) "finish"

$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run
```

source1_10Nodes.tcl

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

#       Initialization

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

set val(nn)     10                      ;# number of mobilenodes

set val(x)      2453                    ;# X dimension of topography

set val(y)      1939                    ;# Y dimension of topography

#Create a ns simulator

set ns [new Simulator]

#Setup topography object

set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#Open the NS trace file

$ns trace-all $tracefile

#Open the NAM trace file

$ns namtrace-all $namfile

$ns namtrace-all-wireless $namfile $val(x) $val(y)
```

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

#===================================
#        Nodes Definition
#===================================
#Create 10 nodes
set n0 [$ns node]
$n0 set X_ 1202
$n0 set Y_ 1698
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 1372
$n1 set Y_ 1538
```

```
$n1 set Z_ 0.0

$ns initial_node_pos $n1 20

set n2 [$ns node]

$n2 set X_ 1361

$n2 set Y_ 1839

$n2 set Z_ 0.0

$ns initial_node_pos $n2 20

set n3 [$ns node]

$n3 set X_ 1549

$n3 set Y_ 1911

$n3 set Z_ 0.0

$ns initial_node_pos $n3 20

set n4 [$ns node]

$n4 set X_ 1636

$n4 set Y_ 1644

$n4 set Z_ 0.0

$ns initial_node_pos $n4 20

set n5 [$ns node]

$n5 set X_ 1733

$n5 set Y_ 1808

$n5 set Z_ 0.0

$ns initial_node_pos $n5 20

set n6 [$ns node]

$n6 set X_ 1522

$n6 set Y_ 1551

$n6 set Z_ 0.0

$ns initial_node_pos $n6 20

set n7 [$ns node]

$n7 set X_ 1736

$n7 set Y_ 1598

$n7 set Z_ 0.0
```

```
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 1829
$n8 set Y_ 1761
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 1836
$n9 set Y_ 1892
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
#===================================
#       Generate movement
#===================================
$ns at 1.0 " $n2 setdest 1424 1740 10 "
$ns at 14 " $n2 setdest 1361 1839 10 "
$ns at 28 " $n2 setdest 1424 1740 10 "
$ns at 42 " $n2 setdest 1361 1839 10 "
$ns at 56 " $n2 setdest 1424 1740 10 "
$ns at 60 " $n2 setdest 1361 1839 10 "
$ns at 74 " $n2 setdest 1424 1740 10 "
$ns at 88 " $n2 setdest 1361 1839 10 "
$ns at 74 " $n4 setdest 1522 1698 10 "
$ns at 1.0 " $n5 setdest 1625 1789 10 "
$ns at 14 " $n5 setdest 1733 1808 10 "
$ns at 28 " $n5 setdest 1625 1789 10 "
$ns at 42 " $n5 setdest 1733 1808 10 "
#===================================
#       Agents Definition
#===================================
#Setup a TCP connection
```

```
if {$tcpAgent == 1} {
   set tcp0 [new Agent/TCP]
} elseif {$tcpAgent == 2} {
   set tcp0 [new Agent/TCP/FullTcp/Tahoe]
} elseif {$tcpAgent == 3} {
   set tcp0 [new Agent/TCP/Reno]
} elseif {$tcpAgent == 4} {
   set tcp0 [new Agent/TCP/Newreno]
} elseif {$tcpAgent ==5} {
   set tcp0 [new Agent/TCP/Vegas]
}
$tcp0 attach $tracefile
$tcp0 tracevar cwnd_
$tcp0 tracevar ssthresh_
$tcp0 tracevar ack_
$tcp0 tracevar maxseq_
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n9 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 2500
#===================================
#        Applications Definition
#===================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 200.0 "$ftp0 stop"
```

source1_30Nodes.tcl

https://github.com/Vishnu2001RV/CN_Project/tree/main/Project

source1_60Nodes.tcl

https://github.com/Vishnu2001RV/CN_Project/tree/main/Project

mobility60Nodes.attach

https://github.com/Vishnu2001RV/CN_Project/tree/main/Project

avg_ThroughPut.awk

```
BEGIN {
        recvdSize_tcp = 0
        startTime_tcp = 1e6
        stopTime_tcp =0
        recvdNum_tcp = 0
}
{
        # Trace line format: normal
        if ($2 != "-t") {
                event = $1
                time = $2
                node_id = $3
                flow_id = $8
                pkt_id = $12
                pkt_size = $8
                #previous $5
                flow_t = $7
                #agt or other ifq for drop etc
                level = $4
        }
        # Trace line format: new
        if ($2 == "-t") {
                event = $1
                time = $3
                node_id = $5
                flow_id = $39
                pkt_id = $41
                pkt_size = $37
                flow_t = $45
                level = $19
        }
        if ((level == "AGT" || level == "IFQ") && (event == "+" || event == "s")&&(node_id==startNode)) {
            if(flow_t == "tcp")
              if (time < startTime_tcp)
                 startTime_tcp = time
        }
        if ((level == "AGT" || level == "IFQ") && event == "r"&&(node_id==endNode)) {
            if(flow_t == "tcp"||flow_t == "ack")
              {
                    recvdSize_tcp += pkt_size
                    recvdNum_tcp += 1
                if (time > stopTime_tcp)
                   stopTime_tcp = time
              }
        }
}
END {
        if (recvdNum_tcp == 0) {
                printf("Warning: no packets were received, simulation may be too short for tcp \n")
        }
  printf("\nAverage ThroughPut")
        printf("\nTCP \n")
        printf(" %15s:  %f\n", "startTime", startTime_tcp)
        printf(" %15s:  %f\n", "stopTime", stopTime_tcp)
        printf(" %15s:  %g\n", "receivedPkts", recvdNum_tcp)
     printf("     Average ThroughPut %15s:  %g\n", "avgTput[kbps]",
(recvdSize_tcp/(stopTime_tcp-startTime_tcp))*(8/1000))
```

```
}
```

inst_E_2_E_Delay.awk

```
BEGIN {
seqno_tcp = 0;
mini_n_to_n_delay_tcp = 0
tic = 25 #0.1
}
{
        # Trace line format: normal
        if ($2 != "-t") {
                event = $1
                time = $2
                node_id = $3
                flow_id = $8
                #pkt_id = $12
                pkt_size = $8
                #previous $5
                flow_t = $7
                #agt or other ifq for drop etc
                level = $4
        }
        # Trace line format: new
        if ($2 == "-t") {
                event = $1
                time = $3
                node_id = $5
                flow_id = $39
                #pkt_id = $41
                pkt_size = $37
                flow_t = $45
                level = $19
        }
```

```
if(prevTime == 0)
          prevTime = time
          #end-to-end delay
if((level == "AGT" || level == "IFQ") && (event == "+" || event == "s")&&(node_id==startNode)) {
         if(flow_t == "tcp"||flow_t == "ack")
         {
           seqno_tcp++;
           start_time_tcp[seqno_tcp] = $2;
         }
      }
 if((level == "AGT" || level == "IFQ") && event == "r"&&(node_id==endNode)) {
     if(flow_t == "tcp"||flow_t == "ack")
      {
         end_time_tcp[seqno_tcp] = $2;
      }
 }
 else if(event == "D") {
    if(flow_t == "tcp"||flow_t == "ack")
      {
        end_time_tcp[seqno_tcp] = 0;
      }
 }
if(flow_t == "tcp"||flow_t == "ack") {
      delay_tcp[seqno_tcp] = end_time_tcp[seqno_tcp] - start_time_tcp[seqno_tcp];
      if(delay_tcp[seqno_tcp] > 0)
          mini_n_to_n_delay_tcp = mini_n_to_n_delay_tcp + delay_tcp[seqno_tcp];
 }
currTime += (time - prevTime)
if (currTime>= tic) {
   tcp_delay = 0.0
   if(seqno_tcp!=0)
       tcp_delay = mini_n_to_n_delay_tcp /(seqno_tcp) * 1000
```

```
        printf("%f %f\n",time,tcp_delay)
        currTime = 0
        mini_n_to_n_delay_tcp = 0
        seqno_tcp = 0
    }
    prevTime = time
}
```

inst_ThroughPut.awk

```
BEGIN {
    recv_tcp = 0
            currTime = prevTime = 0
            tic = 25 #0.1
}
{
            # Trace line format: normal
            if ($2 != "-t") {
                        event = $1
                        time = $2
                        node_id = $3
                        flow_id = $8
                        #pkt_id = $12
                        pkt_size = $8
                        #previous $5
                        flow_t = $7
                        #agt or other ifq for drop etc
                        level = $4
            }
            # Trace line format: new
            if ($2 == "-t") {
                        event = $1
                        time = $3
                        node_id = $5
```

```
                flow_id = $39
                #pkt_id = $41
                pkt_size = $37
                flow_t = $45
                level = $19
        }
        # Init prevTime to the first packet recv time
        if(prevTime == 0)
                prevTime = time
        if ((level == "AGT" || level == "IFQ") && event == "r"&&(node_id==endNode)) {
          if ((flow_t == "tcp")||(flow_t == "ack"))
          {
          #tcp
          recv_tcp += pkt_size
          }
          currTime += (time - prevTime)
          if (currTime >= tic) {
      printf("%f %f\n",time,(recv_tcp/currTime)*(8/1000))
            recv_tcp=0
            currTime = 0
                }
                prevTime = time
        }
}
END {
        printf("\n\n")
}
```

PDR_Avg_E_2_E.awk

```
BEGIN {
  seqno_tcp = 0;
  droppedPackets_tcp = 0;
  receivedPackets_tcp = 0;
```

```
   n_to_n_delay_tcp = 0
   count = 0;
}
{
          # Trace line format: normal
          if ($2 != "-t") {
                    event = $1
                    time = $2
                    node_id = $3
                    flow_id = $8
                    #pkt_id = $12
                    pkt_size = $8
                    #previous $5
                    flow_t = $7
                    #agt or other ifq for drop etc
                    level = $4
          }
          # Trace line format: new
          if ($2 == "-t") {
                    event = $1
                    time = $3
                    node_id = $5
                    flow_id = $39
                    #pkt_id = $41
                    pkt_size = $37
                    flow_t = $45
                    level = $19
          }
  #packet delivery ratio
  if((level == "AGT" || level == "IFQ") && (event == "+" || event == "s")&&(node_id==startNode))
  {
    if(flow_t == "tcp")
  {
    seqno_tcp++;
  }
}
if((level == "AGT" || level == "IFQ") && event == "r"&&(node_id==endNode)) {
  if(flow_t == "tcp"||flow_t == "ack")
  {
    receivedPackets_tcp++;
  }
}
else if (event == "D"){
  if(flow_t == "tcp"||flow_t == "ack")
  {
    droppedPackets_tcp++;
  }
}
#end-to-end delay
if((level == "AGT" || level == "IFQ") && (event == "+" || event == "s")&&(node_id==startNode)) {
  if(flow_t == "tcp"||flow_t == "ack")
  {
    start_time_tcp[seqno_tcp] = $2;
  }
}
else if((level == "AGT" || level == "IFQ") && event == "r"&&(node_id==endNode)) {
   if(flow_t == "tcp"||flow_t == "ack")
    {
       end_time_tcp[seqno_tcp] = $2;
    }
}
else if(event == "D") {
  if(flow_t == "tcp"||flow_t == "ack")
    {
          end_time_tcp[seqno_tcp] = 0;
        }
  }
}
END {
  for(i=0; i <= seqno_tcp; i++) {
     if(end_time_tcp[i] > 0) {
         delay_tcp[i] = end_time_tcp[i] - start_time_tcp[i];
         if(delay_tcp[i] > 0) {
             n_to_n_delay_tcp = n_to_n_delay_tcp + delay_tcp[i];
         }
     }
 }
  packet_ratio_tcp = 0
  if(seqno_tcp>0){
      packet_ratio_tcp = receivedPackets_tcp/(seqno_tcp)*100
      n_to_n_delay_tcp = n_to_n_delay_tcp /(seqno_tcp);
  }
  print "\n";
  print "PacketDelivaryRatio And End to End Delay"
  print "TCP "
  print "     GeneratedPackets = " seqno_tcp;
  print "     ReceivedPackets = " receivedPackets_tcp;
  print "     Packet Delivery Ratio = " packet_ratio_tcp "%";
  print "     Total Dropped Packets = " seqno_tcp-receivedPackets_tcp;#droppedPackets_tcp;
  print "     Average End-to-End Delay = " n_to_n_delay_tcp * 1000 " ms";
  print "\n";
}
```

congestion_window.awk

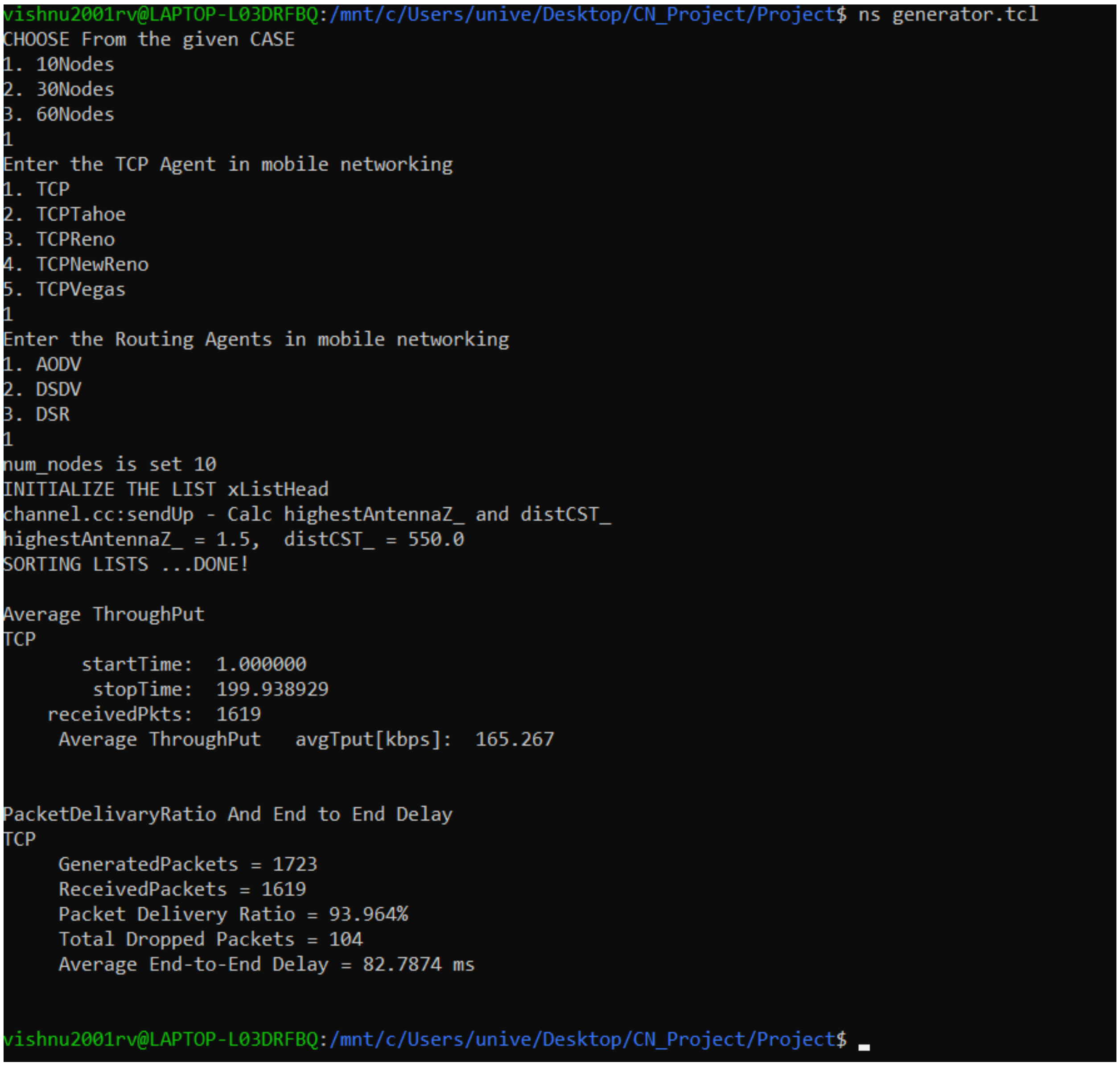
```
BEGIN {
}
{
if($6=="cwnd_") {
   printf("%f\t%f\n",$1,$7);
}
}
{
}
```

normalizedRoutingOverhead.awk

```
BEGIN{
recvs = 0;
routing_packets = 0;
}
{
if (( $1 == "r") &&  ( $7 == "tcp" ) && ( $4=="AGT" ))  recvs++;
if (($1 == "s" || $1 == "f") && $4 == "RTR" && ( $7 =="AODV" || $7 =="message" || $7 =="DSR") )
routing_packets++;
}
END{
printf("NRL = %.3f", routing_packets/recvs);
}
```

CHAPTER 5

RESULTS



```
vishnu2001rv@LAPTOP-L03DRFBQ:/mnt/c/Users/unive/Desktop/CN_Project/Project$ ns generator.tcl
CHOOSE From the given CASE
1. 10Nodes
2. 30Nodes
3. 60Nodes
1
Enter the TCP Agent in mobile networking
1. TCP
2. TCPTahoe
3. TCPReno
4. TCPNewReno
5. TCPVegas
1
Enter the Routing Agents in mobile networking
1. AODV
2. DSDV
3. DSR
1
num_nodes is set 10
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!

Average ThroughPut
TCP
       startTime:  1.000000
        stopTime:  199.938929
    receivedPkts:  1619
     Average ThroughPut   avgTput[kbps]:  165.267


PacketDelivaryRatio And End to End Delay
TCP
     GeneratedPackets = 1723
     ReceivedPackets = 1619
     Packet Delivery Ratio = 93.964%
     Total Dropped Packets = 104
     Average End-to-End Delay = 82.7874 ms


vishnu2001rv@LAPTOP-L03DRFBQ:/mnt/c/Users/unive/Desktop/CN_Project/Project$
```

Figure 5.1 The console output got from generator file

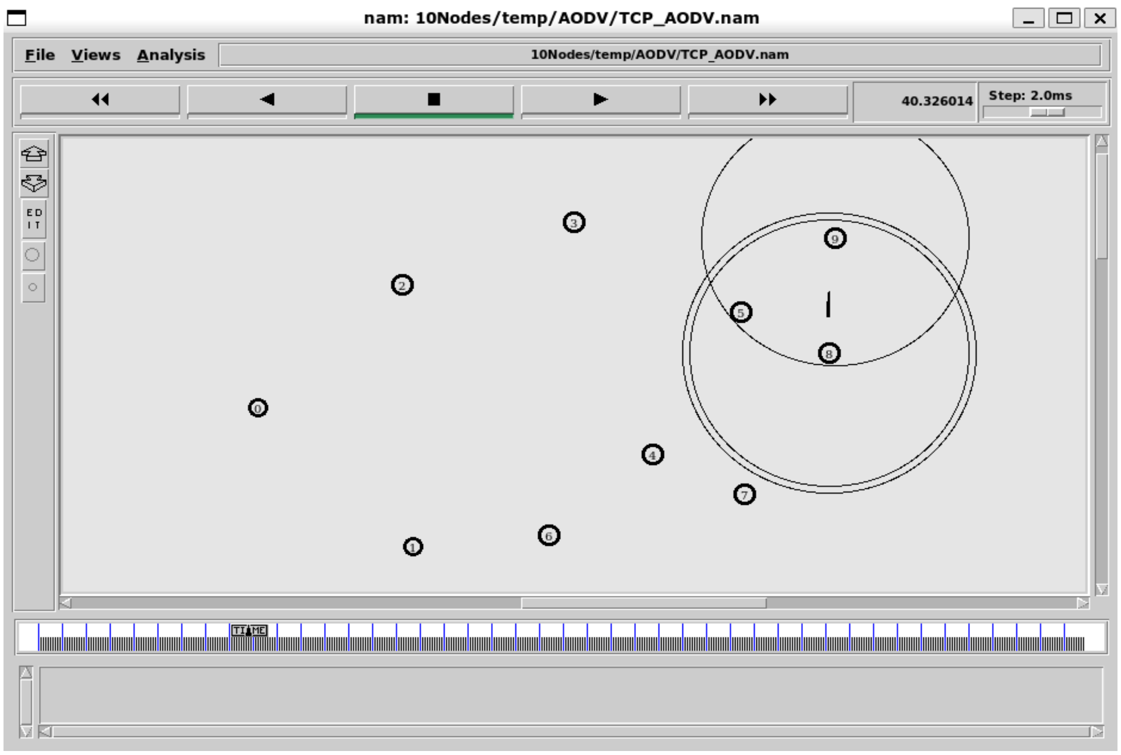

nam: 10Nodes/temp/AODV/TCP_AODV.nam
File
Views
Analysis
10Nodes/temp/AODV/TCP_AODV.nam
40.326014
Step: 2.0ms
TIME
0
1
2
3
4
5
6
7
8
9

Figure 5.2 Simulation at Number of Nodes 10

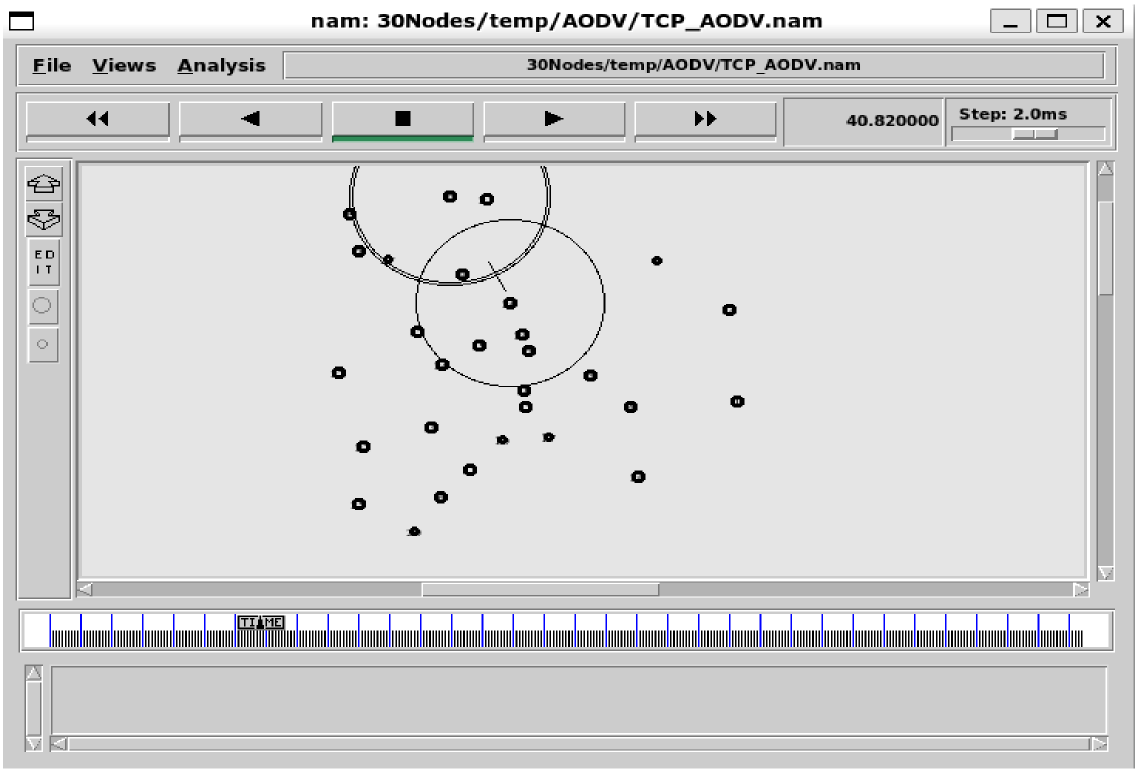

nam: 30Nodes/temp/AODV/TCP_AODV.nam
File
Views
Analysis
30Nodes/temp/AODV/TCP_AODV.nam
40.820000
Step: 2.0ms
TIME

Figure 5.3 Simulation at Number of Nodes 30

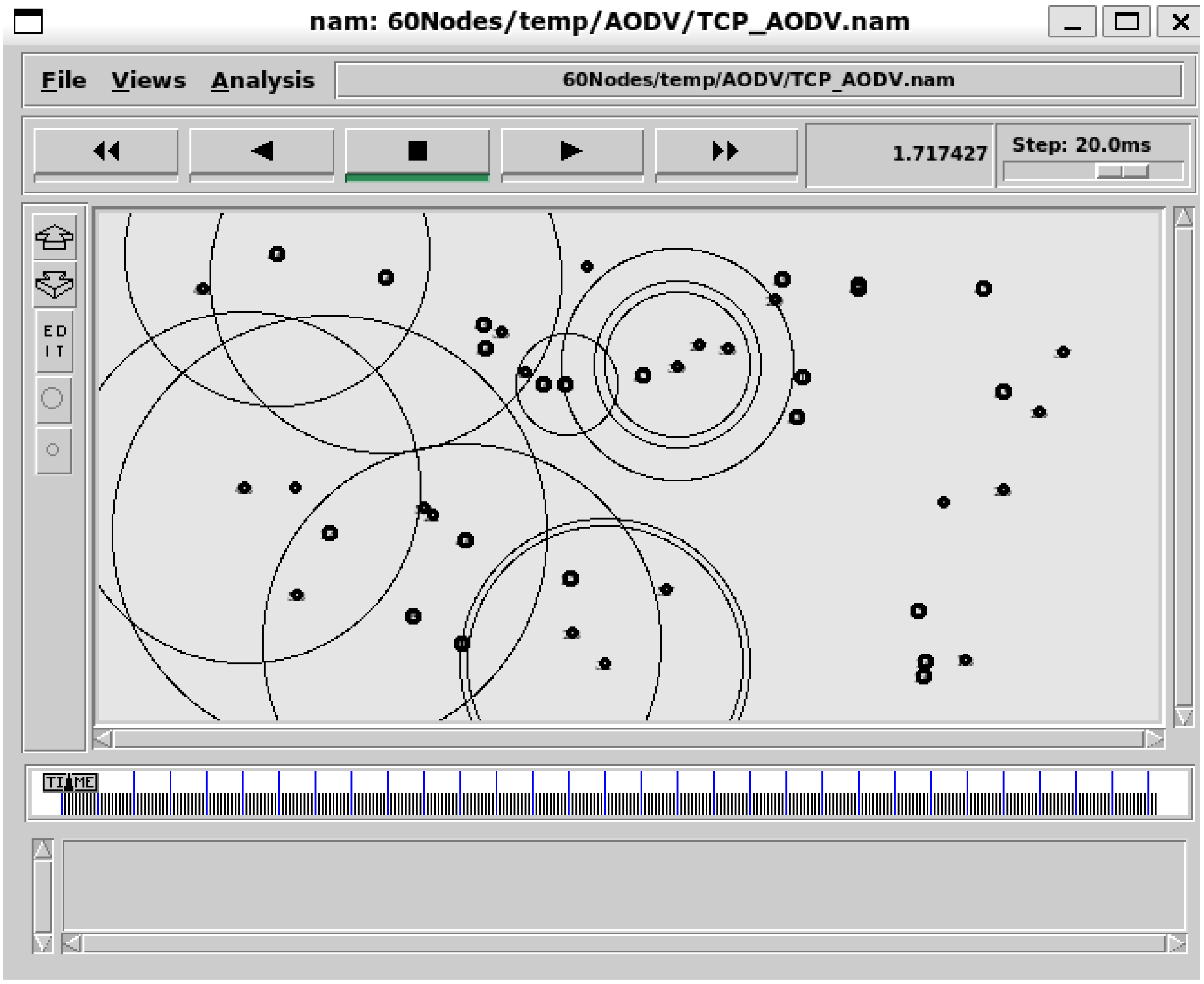

nam: 60Nodes/temp/AODV/TCP_AODV.nam
File
Views
Analysis
60Nodes/temp/AODV/TCP_AODV.nam
1.717427
Step: 20.0ms
EDIT
TIME

Figure 5.4 Simulation at Number of Nodes 60

CHAPTER 6
PERFORMANCE ANALYSIS

The analysis of TCP Variants using AODV, DSDV, DSR Routing Protocols in MANETs is simulated using NS2 simulator.

In this we have analyzed three scenarios, in each scenario we have analyzed seven parameters such as Average throughput, Average delay and Packet Delivery Ratio(PDR), Instantaneous throughput, Instantaneous end-to-end delay, congestion window, routing overhead.

SCENARIO-1:

The simulation parameters for the first scenario are as follows:

PARAMETER	VALUE
Simulation Environment	2453 x 1939 sq.mt
Transmission range	250 m
Inference range	550 m
Number of nodes	10
Simulation time	200 sec
MAC Protocol	IEEE 802.11
Speed of each Node	10 m/s
Topography	flatgrid
Node Mobility	random

Table 6.1 Simulation Parameters for Scenario 1

Average end-to-end delay of TCP Variants over AODV Routing Protocol:

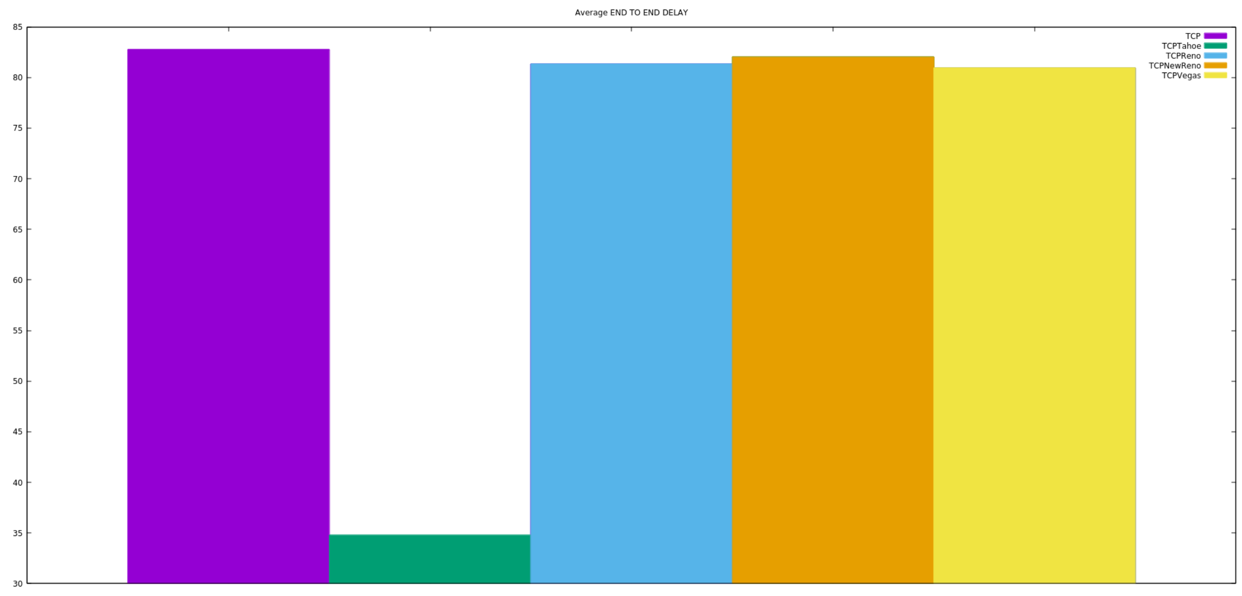


Figure 6.1 Scenario1 Average end-to-end delay of TCP Variants over AODV Routing Protocol

Average ThroughPut of TCP Variants over AODV Routing Protocol:

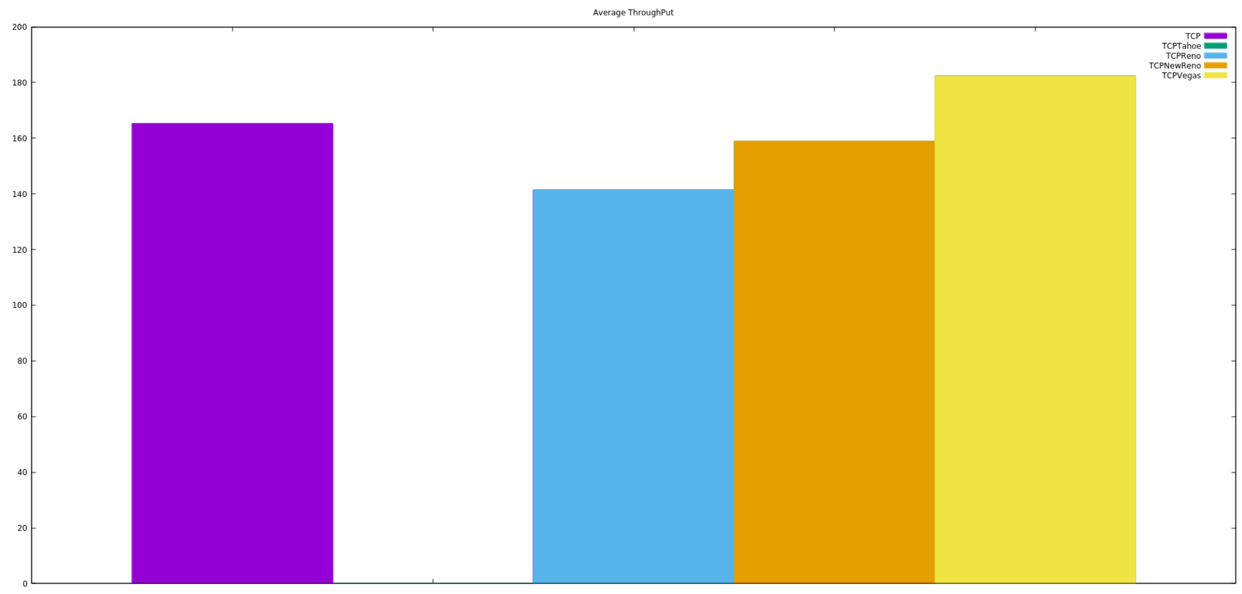


Figure 6.2 Scenario1 Average ThroughPut of TCP Variants over AODV Routing Protocol

Packet Delivery Ratio of TCP Variants over AODV Routing Protocol:

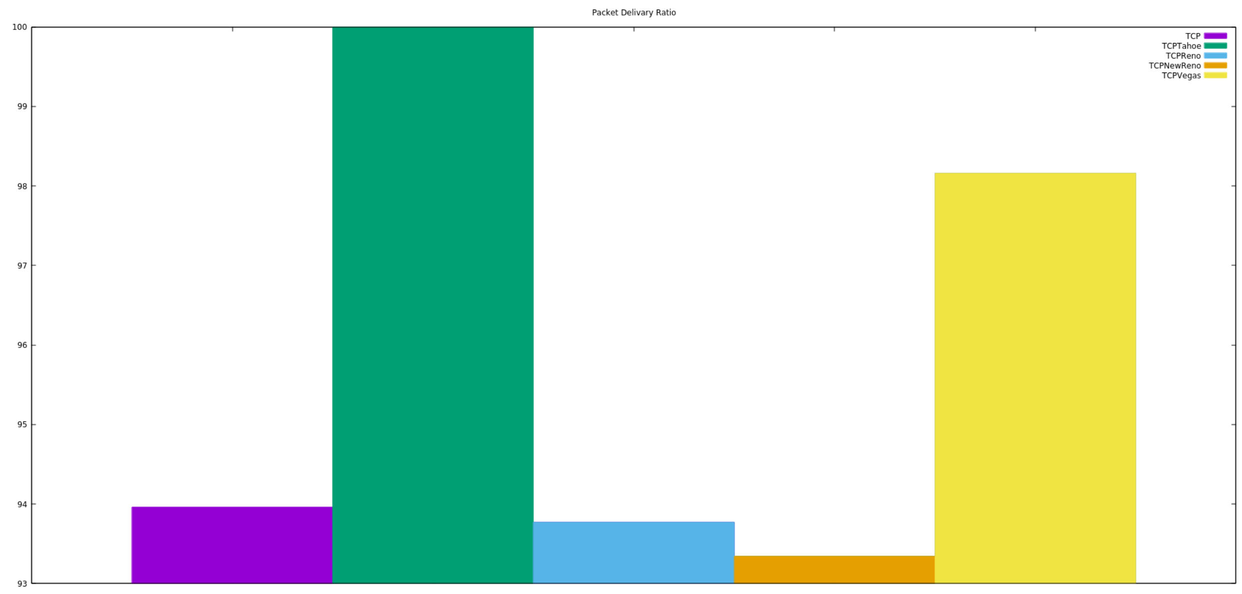


Figure 6.3 Scenario1 Packet Delivery Ratio of TCP Variants over AODV Routing Protocol

Instantaneous end-to-end delay of TCP Variants over AODV Routing Protocol:

Note:- Calculated for every 25sec

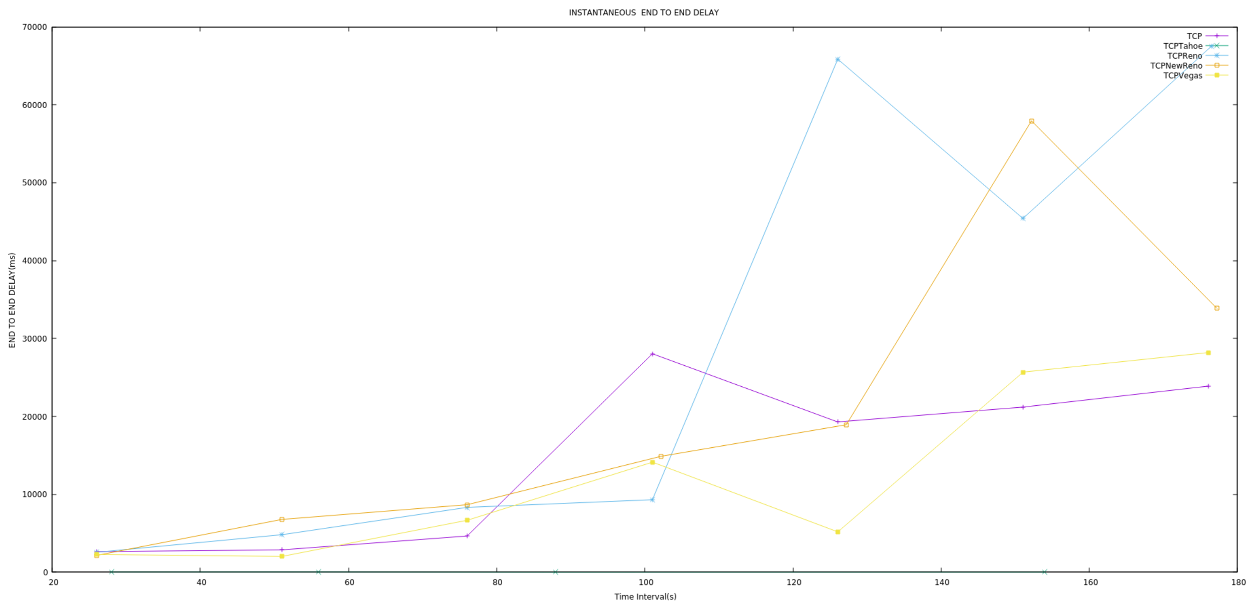


Figure 6.4 Scenario1 Instantaneous end-to-end delay of TCP Variants over AODV Routing Protocol

Instantaneous ThroughPut of TCP Variants over AODV Routing Protocol:

Note:- Calculated for every 25sec

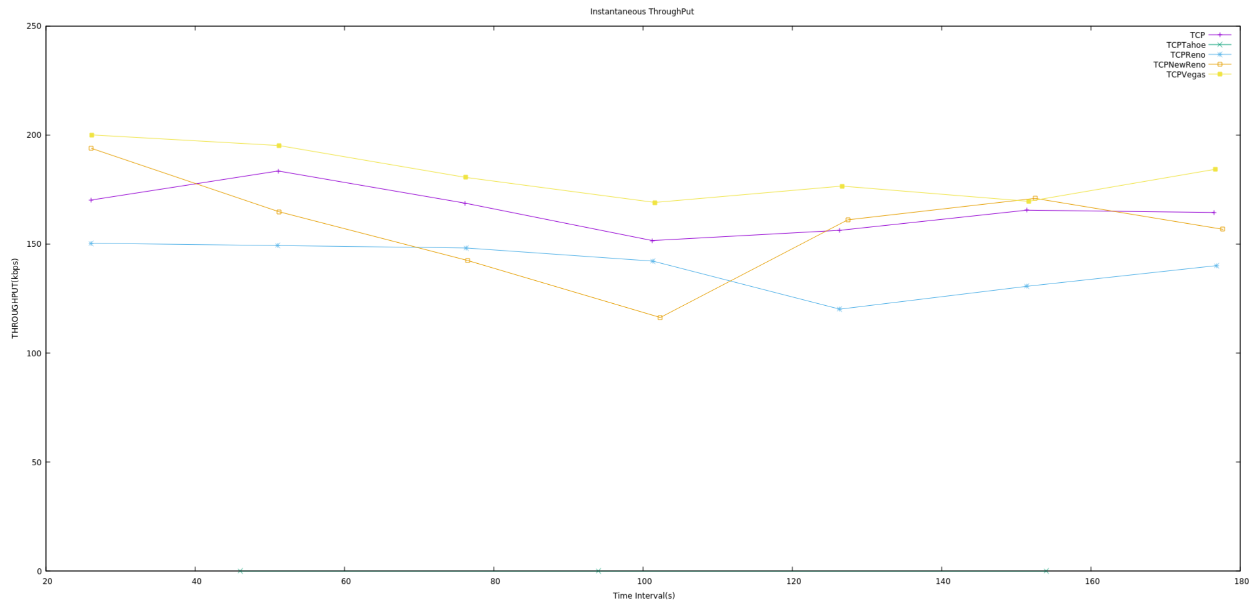


Figure 6.5. Scenario1 Instantaneous ThroughPut of TCP Variants over AODV Routing Protocol

Congestion Window of Various TCP Variants over AODV Routing Protocol:

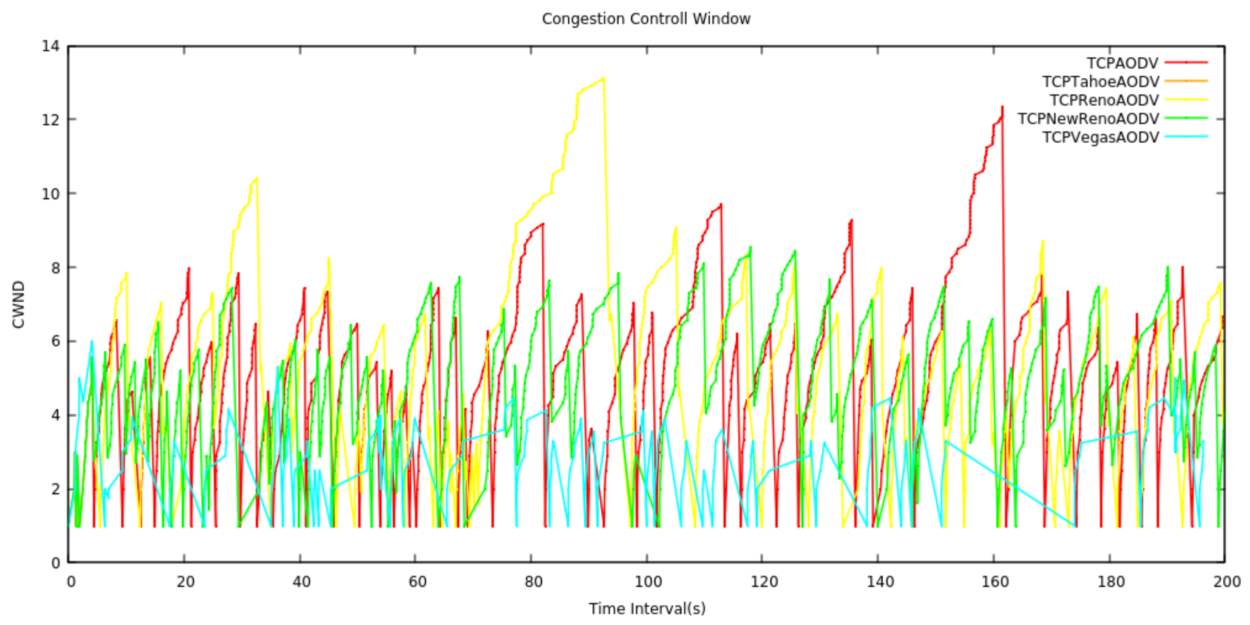


Figure 6.6 Scenario 1 Congestion Window of Various TCP Variants over AODV Routing Protocol

Average end-to-end delay of TCP Variants over DSDV Routing Protocol:

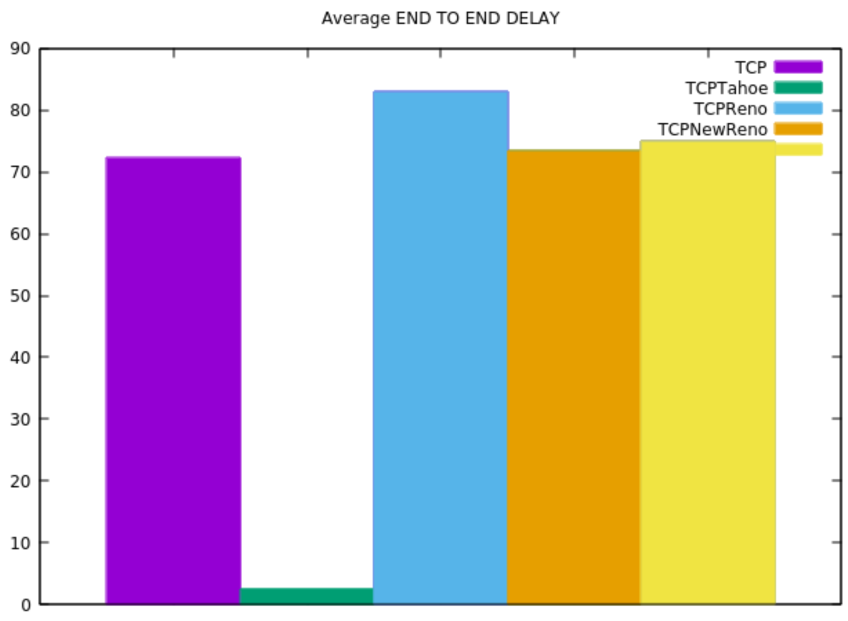


Figure 6.7 Scenario1 Average end-to-end delay of TCP Variants over DSDV Routing Protocol

Average ThroughPut of TCP Variants over DSDV Routing Protocol:

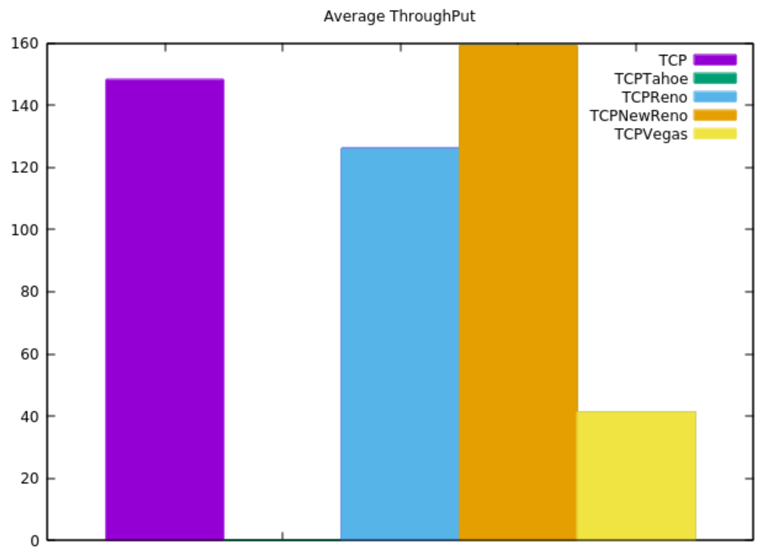


Figure 6.8 Scenario1 Average ThroughPut of TCP Variants over DSDV Routing Protocol

Packet Delivery Ratio of TCP Variants over DSDV Routing Protocol:

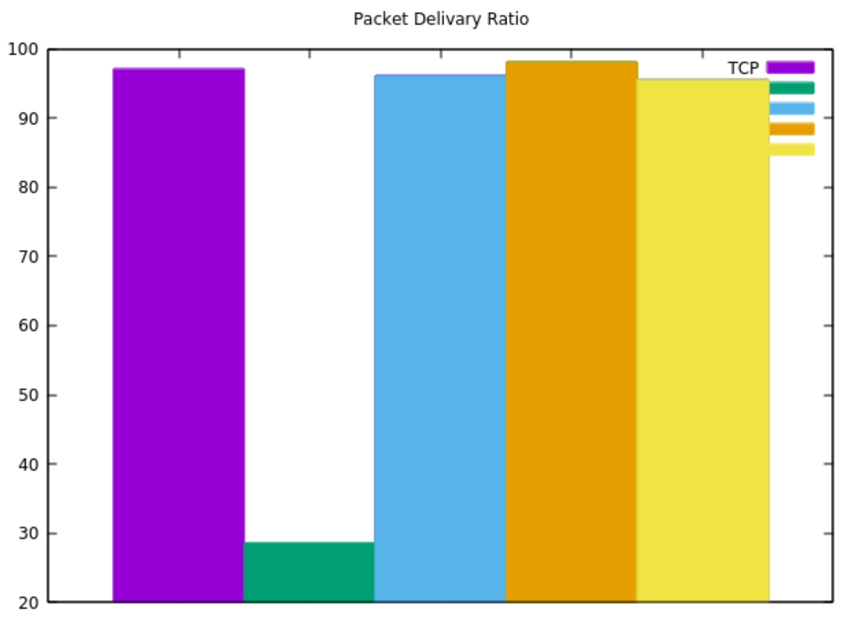


Figure 6.9 Scenario1 Packet Delivery Ratio of TCP Variants over DSDV Routing Protocol

Instantaneous end-to-end delay of TCP Variants over DSDV Routing Protocol:

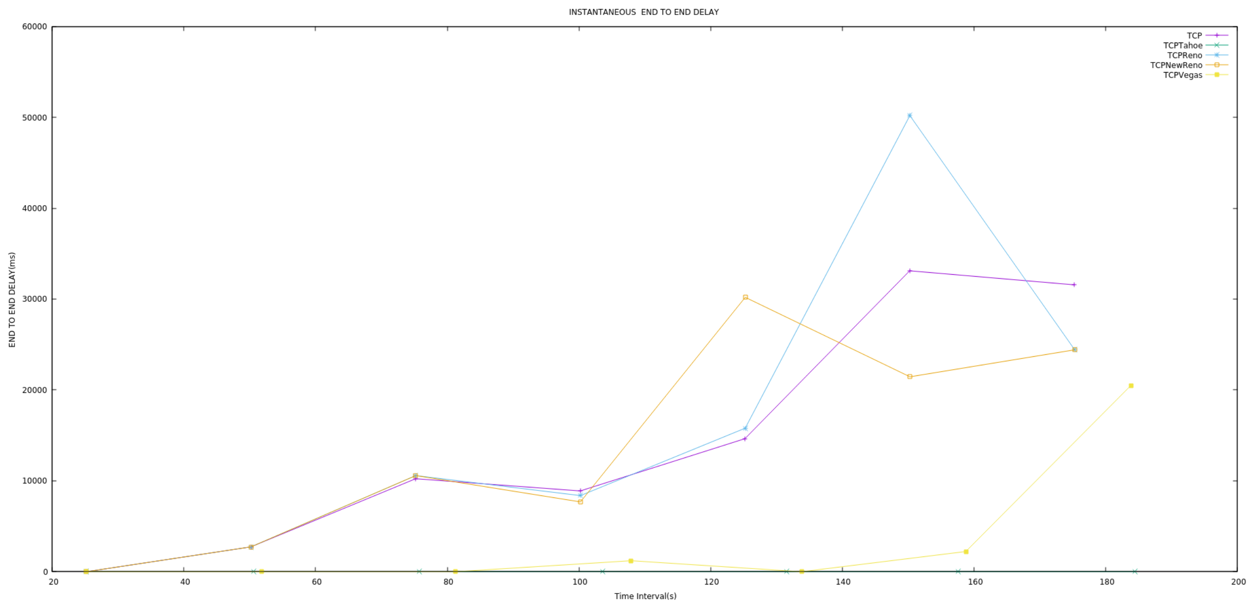


Figure 6.10 Scenario1 Instantaneous end-to-end delay of TCP Variants over DSDV Routing Protocol

nstantaneous ThroughPut of TCP Variants over DSDV Routing Protocol:

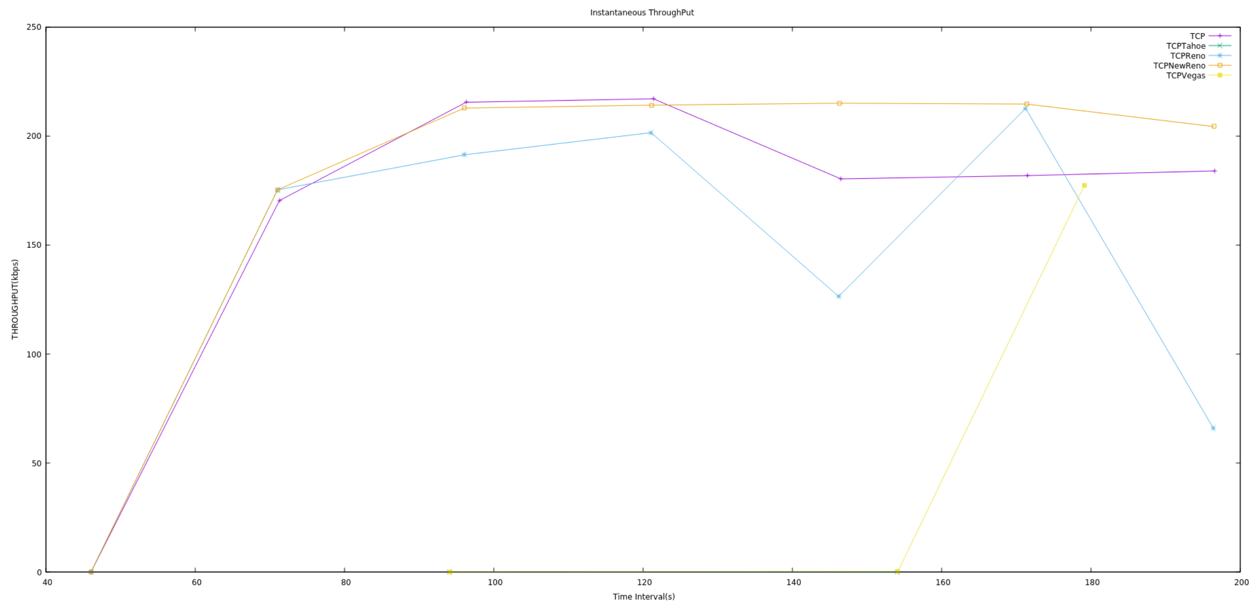


Figure 6.11 Scenario1 Instantaneous ThroughPut of TCP Variants over DSDV Routing Protocol

Congestion Window of various TCP Variants over DSDV Routing Protocol:

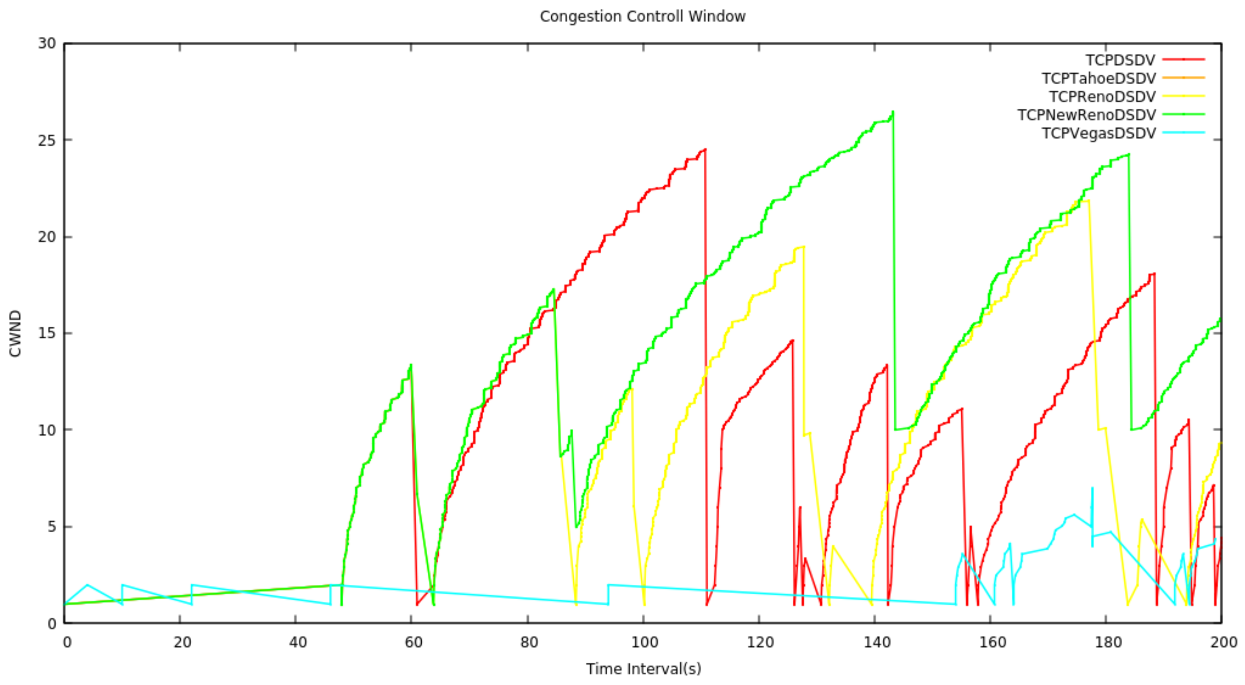


Figure 6.12 Scenario 1 Congestion Window of various TCP Variants over DSDV Routing Protocol

Average end-to-end delay of TCP Variants over DSR Routing Protocol:

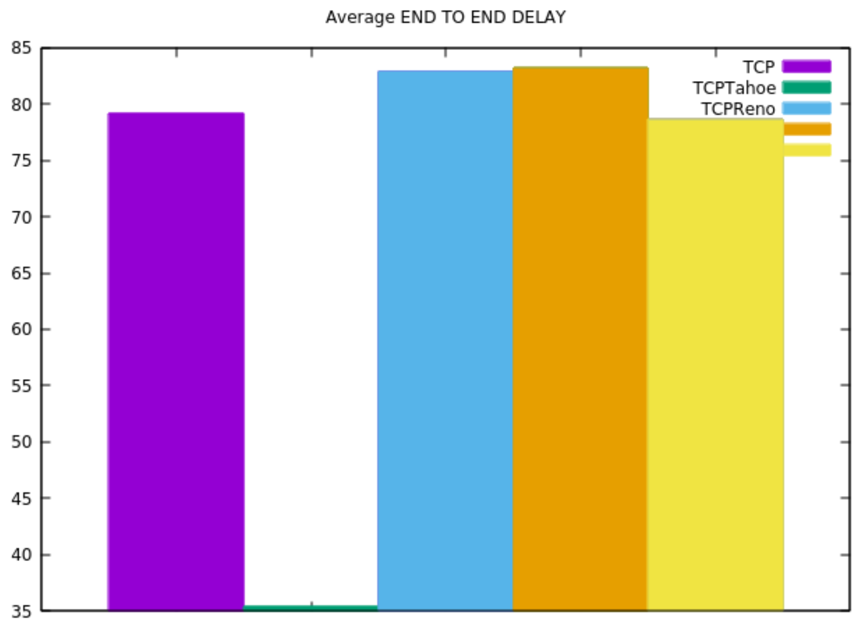


Figure 6.13 Scenario1 Average end-to-end delay of TCP Variants over DSR Routing Protocol

Average ThroughPut of TCP Variants over DSR Routing Protocol:

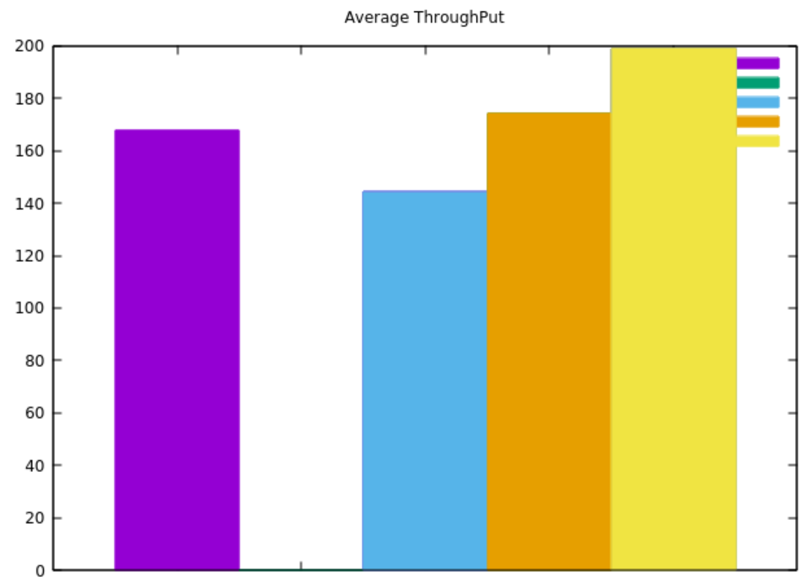


Figure 6.14 Scenario1 Average ThroughPut of TCP Variants over DSR Routing Protocol

Packet Delivery Ratio of TCP Variants over DSR Routing Protocol:

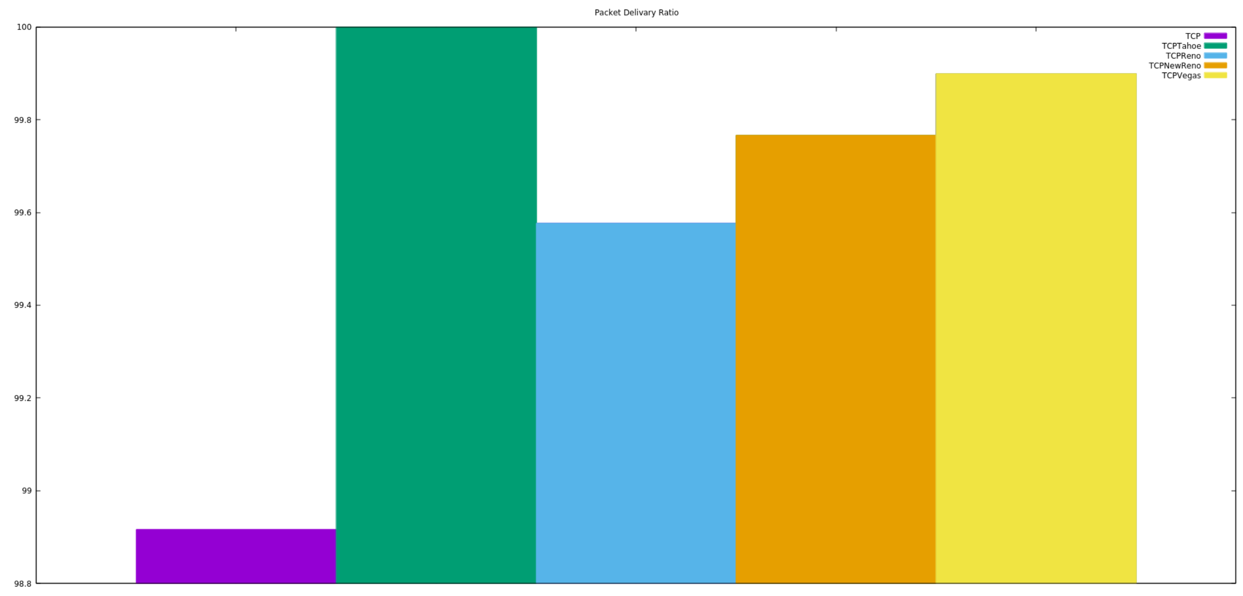


Figure 6.15 Scenario1 Packet Delivery Ratio of TCP Variants over DSR Routing Protocol

Instantaneous end-to-end delay of TCP Variants over DSR Routing Protocol:

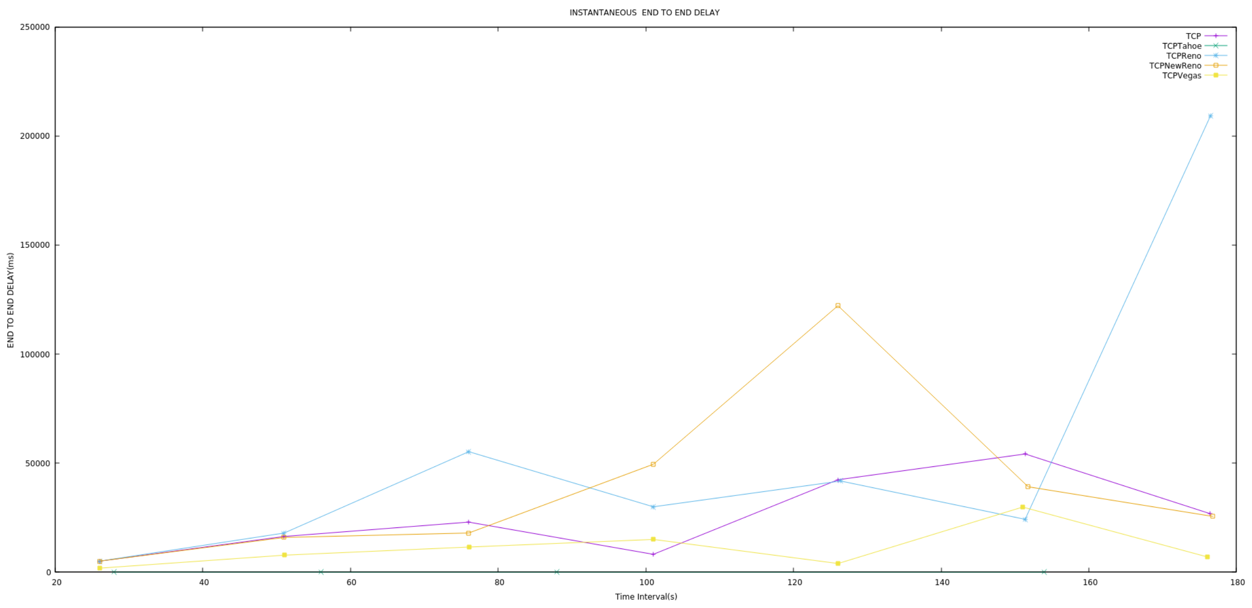


Figure 6.16 Scenario1 Instantaneous end-to-end delay of TCP Variants over DSR Routing Protocol

Instantaneous ThroughPut of TCP Variants over DSR Routing Protocol:

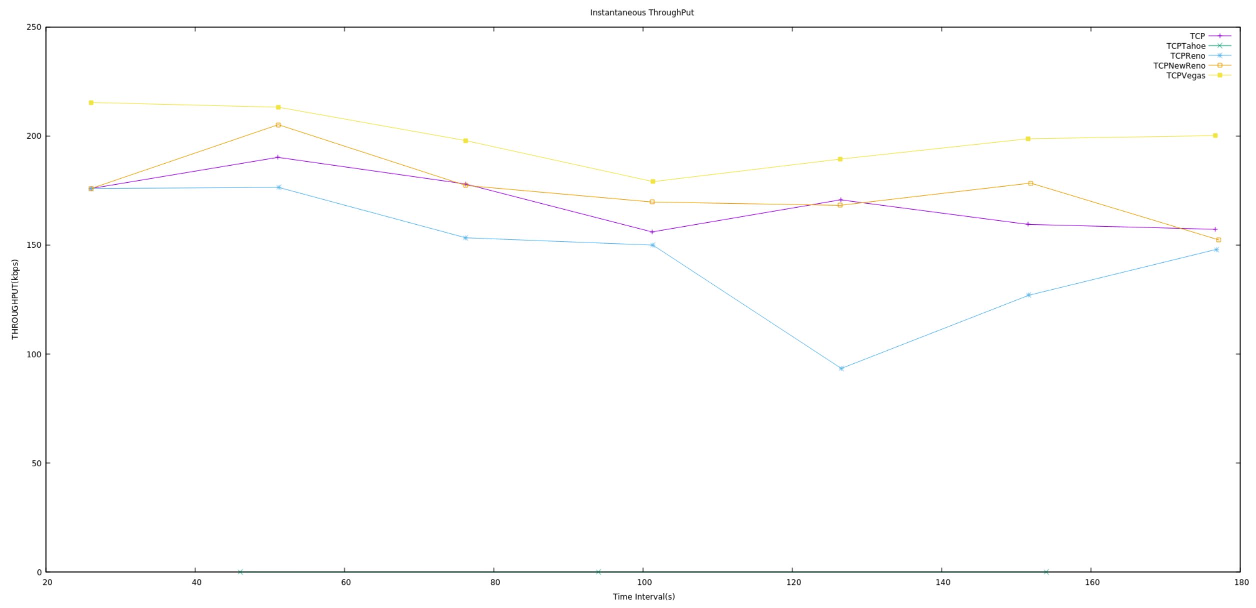


Figure 6.17 Scenario1 Instantaneous ThroughPut of TCP Variants over DSR Routing Protocol

Congestion window of various TCP Variants over DSR Routing Protocol:

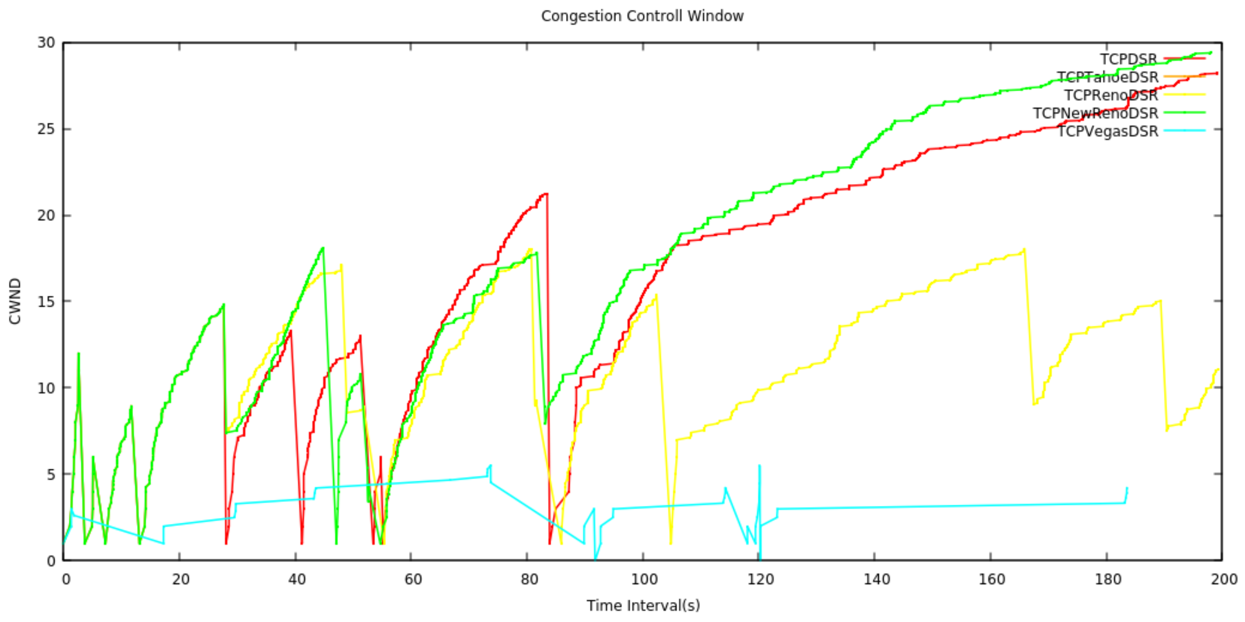


Figure 6.18 Scenario1 Congestion window of various TCP Variants over DSR Routing Protocol

Routing OverHead of various TCP Variants over DSR Routing Protocol:

	TCP	TCP Tahoe	TCP Reno	TCP NewReno	TCP Vegas
AODV	3.032	9.571	2.969	3.062	2.298
DSDV	0.147	108.500	0.177	0.138	0.516
DSR	0.701	3.143	0.740	0.617	0.205

Table 6.2 Scenario1 Routing OverHead of various TCP Variants over DSR Routing Protocol

Observation:-

AODV

For number of nodes 10, The Normal TCP has slightly higher End-to-End Delay for Scenario 1 when compared to other TCP Variants except TCP Tahoe(Figure 6.1). From Figure 6.6 we could see that TCP NewReno can retransmit a maximum of one loss packet per RTT due to which remaining bandwidth is unutilized thus there is decrease in throughput.

When it comes to average throughput and Packet Delivery Ratio TCP Vegas is high when compared to other TCP Variants which could be seen through Figure 6.2 and Figure 6.3. Therefore TCP Vegas is performing better for a small number of nodes in AODV while having an optimum congestion window for predicting packet loss.

From table 6.2 in AODV TCP Vegas has less routing overhead which also indicates that it works well in AODV.

DSDV

For Scenario1 TCP Reno has highest end to end Delay when compared to other TCP variants in DSDV as a result throughput of TCP Reno is decreased compared to AODV.

TCP New Reno Performs extremely well and provides high throughput in DSDV with a small number of nodes, but the performance throughput of TCP Vegas has decreased Drastically. Thus predicting congestion at early stages has not proven to be effective for TCP Vegas in DSDV.Therefore TCP Vegas is not suited for DSDV with a small number of nodes.

TCP Reno throughput has decreased due to low congestion window and high end to end delay which could be seen from figure 6.7 and 6.12.

From the Figure 6.12 it is evident that TCP NewReno, not exiting the fast-recovery has proven to be effective in DSDV.

From table 6.2 TCP New Reno has less routing overhead as it works well with DSDV.

DSR

TCP Reno performs low when compared to other TCP Variants with respect to throughput and packet delivery ratio.

From figure 6.16 TCP NewReno end-to-end delay increased when mobility was started after 60 sec which impacted its throughput.

From Figure 6.13 and Figure 6.14 TCP Vegas was found performing extremely well in the DSR Routing protocol for a small number of nodes.

From figure 6.18 we could understand that even though the TCP Vegas Congestion window is small, the high packet delivery ratio from dynamic setting of windows size has proven to increase the Throughput in scenario1 and from the instantaneous throughput we could say that TCP Vegas works well for DSR routing protocol for scenario1 even when there is mobility after 60 secs.

From table 6.2 TCP Vegas has less routing overhead as it works well with DSR.

SCENARIO-2:

The simulation parameters for the second scenario are as follows:

PARAMETER	VALUE
Simulation Environment	8254 x 3941 sq.mt
Transmission range	250 m
Inference range	550 m
Number of nodes	30
Simulation time	200 sec
MAC Protocol	IEEE 802.11
Speed of each Node	10 m/s or 20m/s
Topography	flatgrid
Node Mobility	Random

Table 6.3 Simulation Parameters for Scenario 2

Average end-to-end delay of TCP Variants over AODV Routing Protocol:

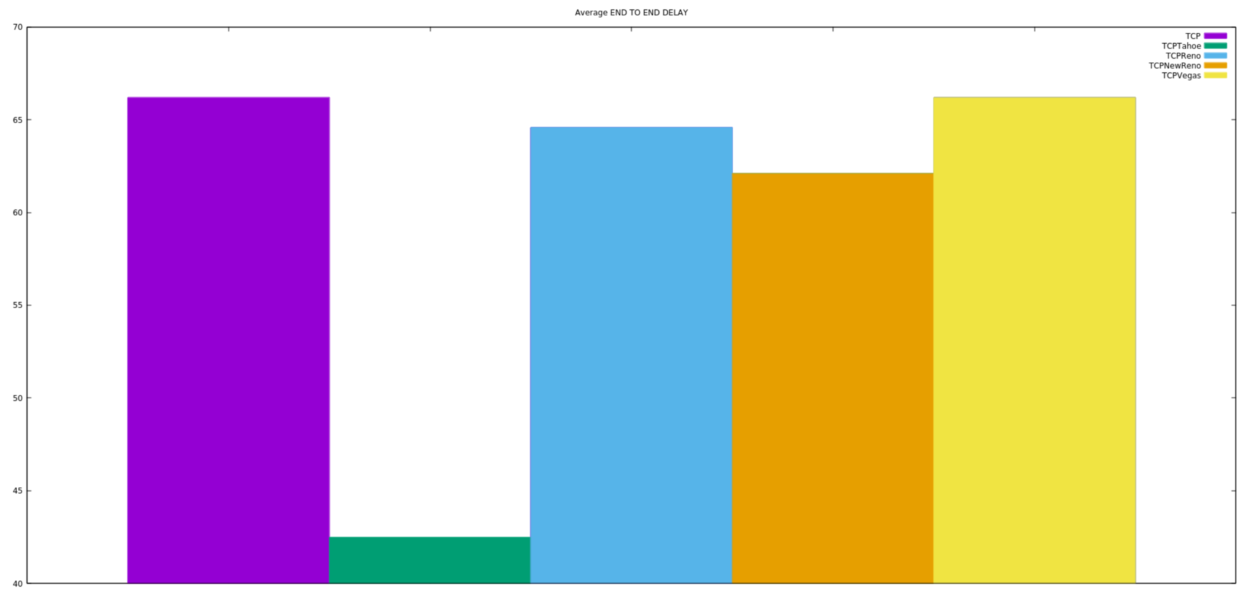


Figure 6.19 Scenario2 Average end-to-end delay of TCP Variants over AODV Routing Protocol

Average ThroughPut of TCP Variants over AODV Routing Protocol:

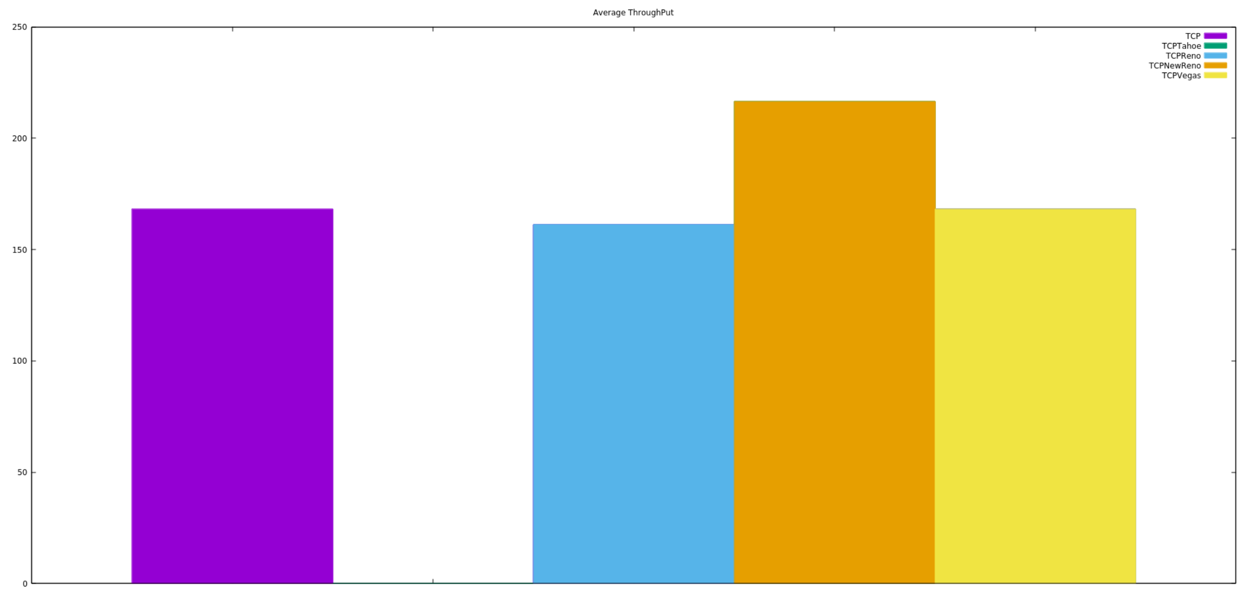


Figure 6.20 Scenario2 Average ThroughPut of TCP Variants over AODV Routing Protocol

Packet Delivery Ratio of TCP Variants over AODV Routing Protocol:

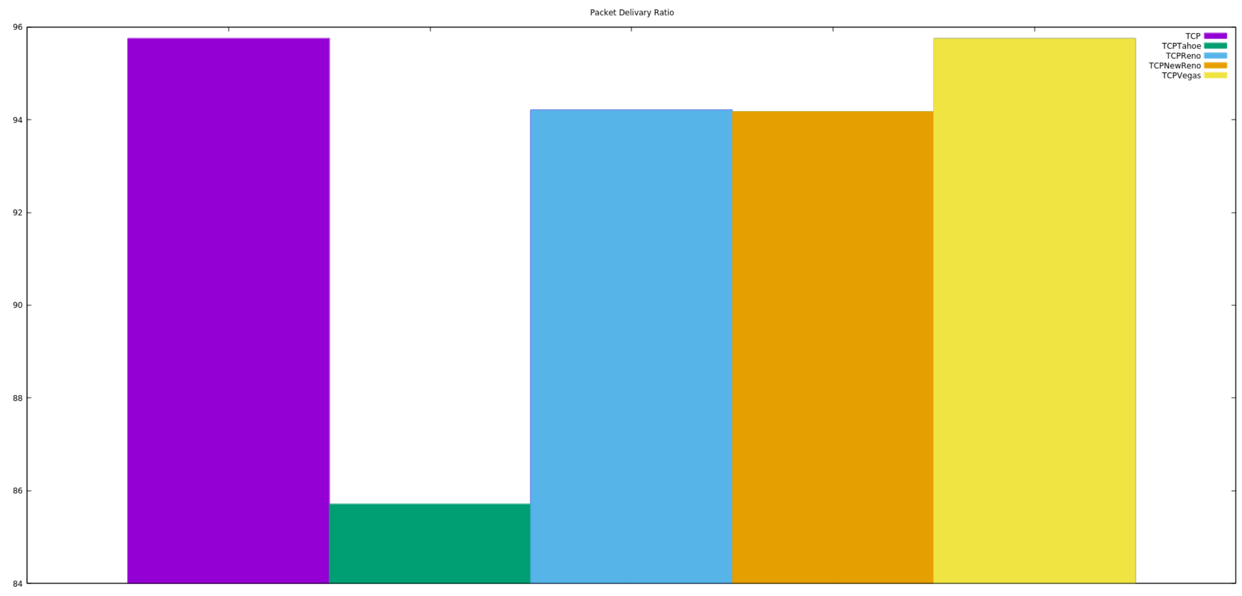


Figure 6.21 Scenario2 Packet Delivery Ratio of TCP Variants over AODV Routing Protocol

Instantaneous end-to-end delay of TCP Variants over AODV Routing Protocol:

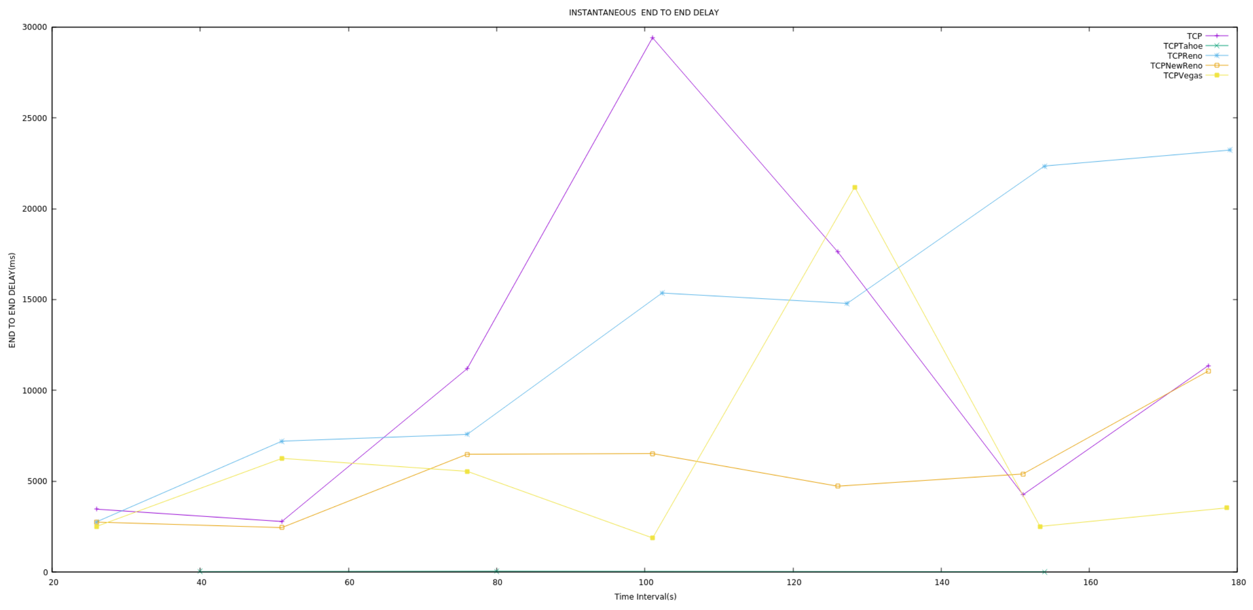


Figure 6.22 Scenario2 Instantaneous end-to-end delay of TCP Variants over AODV Routing Protocol

Instantaneous ThroughPut of TCP Variants over AODV Routing Protocol:

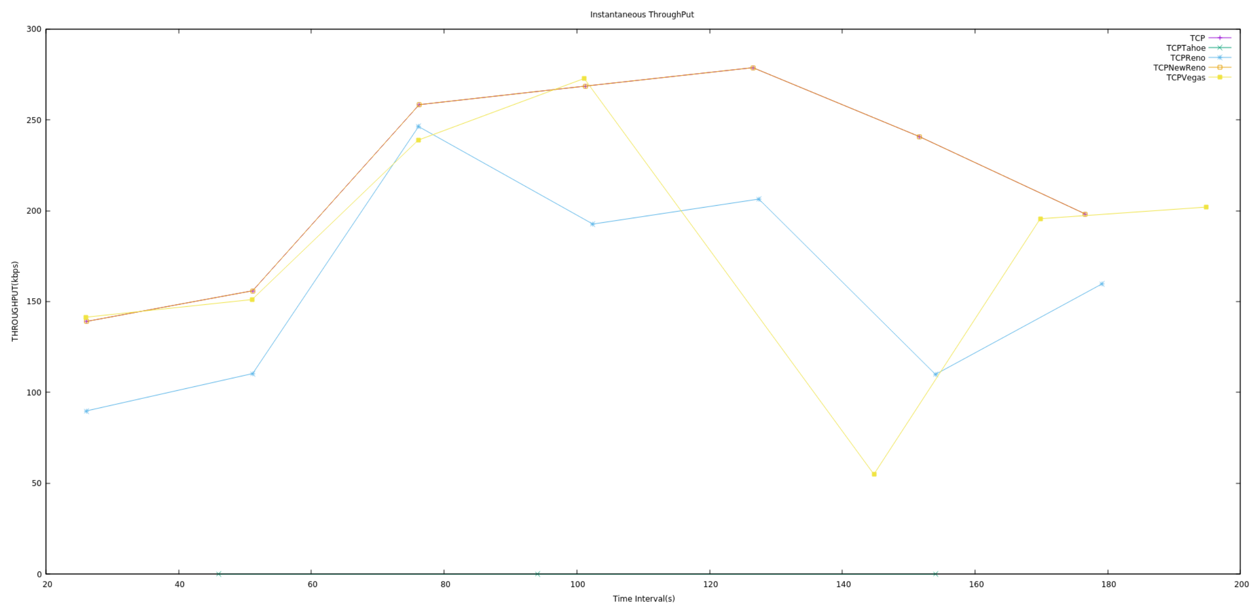


Figure 6.23 Scenario2 Instantaneous ThroughPut of TCP Variants over AODV Routing Protocol

Congestion window of various TCP Variants over AODV Routing Protocol:

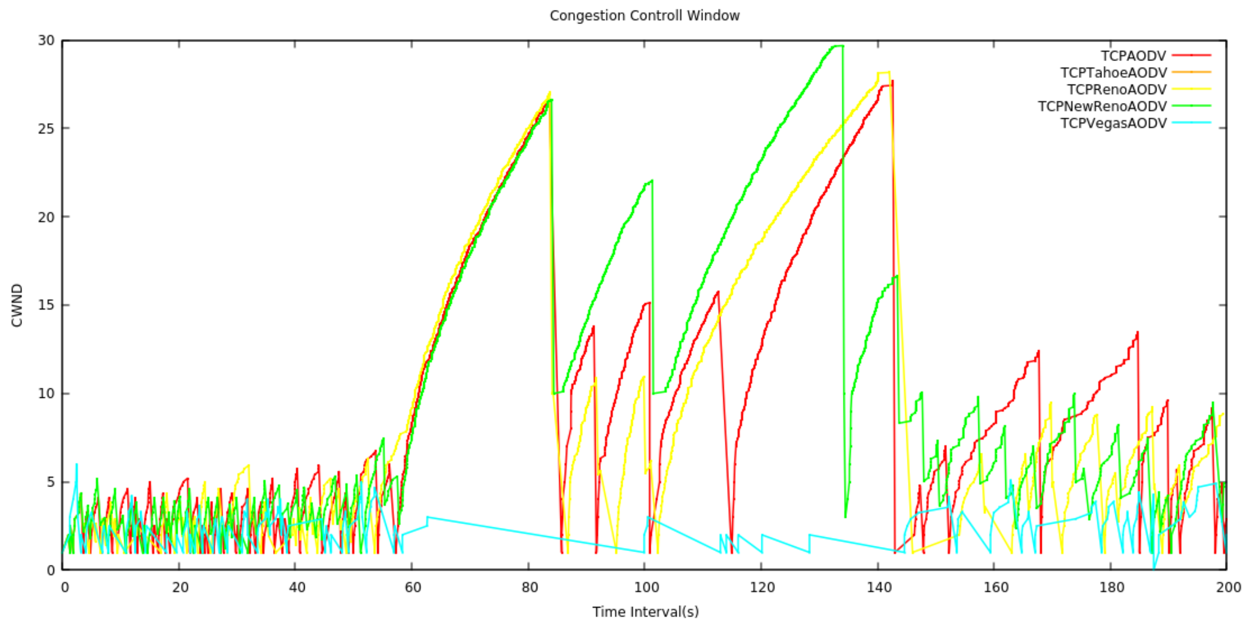


Figure 6.24 Scenario2 Congestion window of various TCP Variants over AODV Routing Protocol

Average end-to-end delay of TCP Variants over DSDV Routing Protocol:

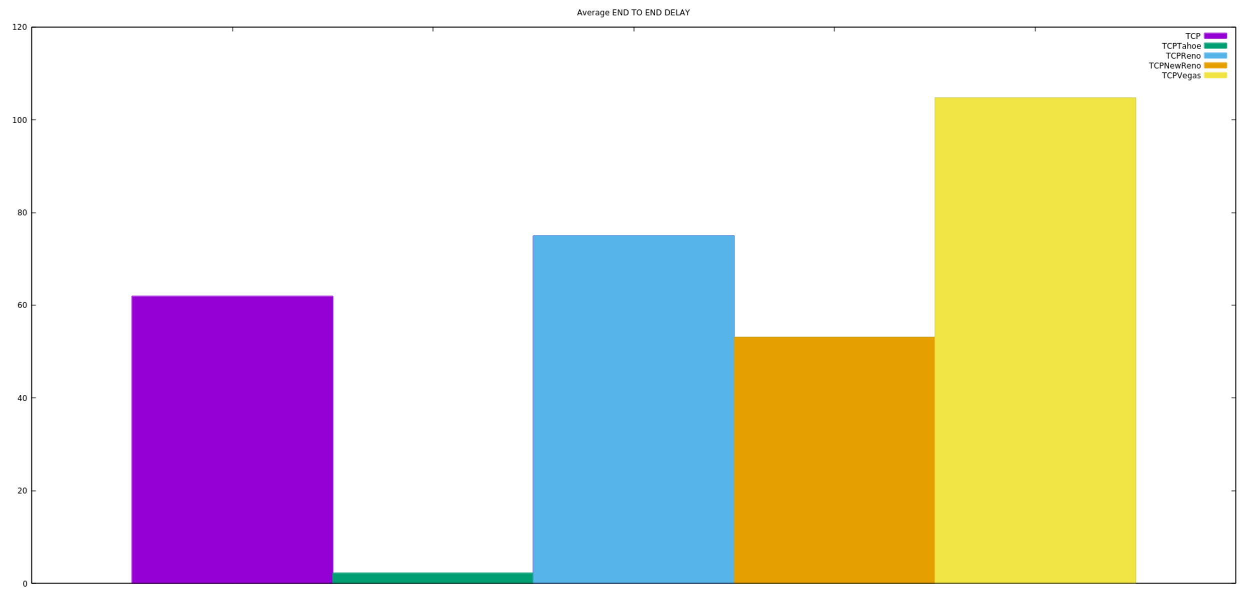


Figure 6.25 Scenario2 Average end-to-end delay of TCP Variants over DSDV Routing Protocol

Average ThroughPut of TCP Variants over DSDV Routing Protocol:

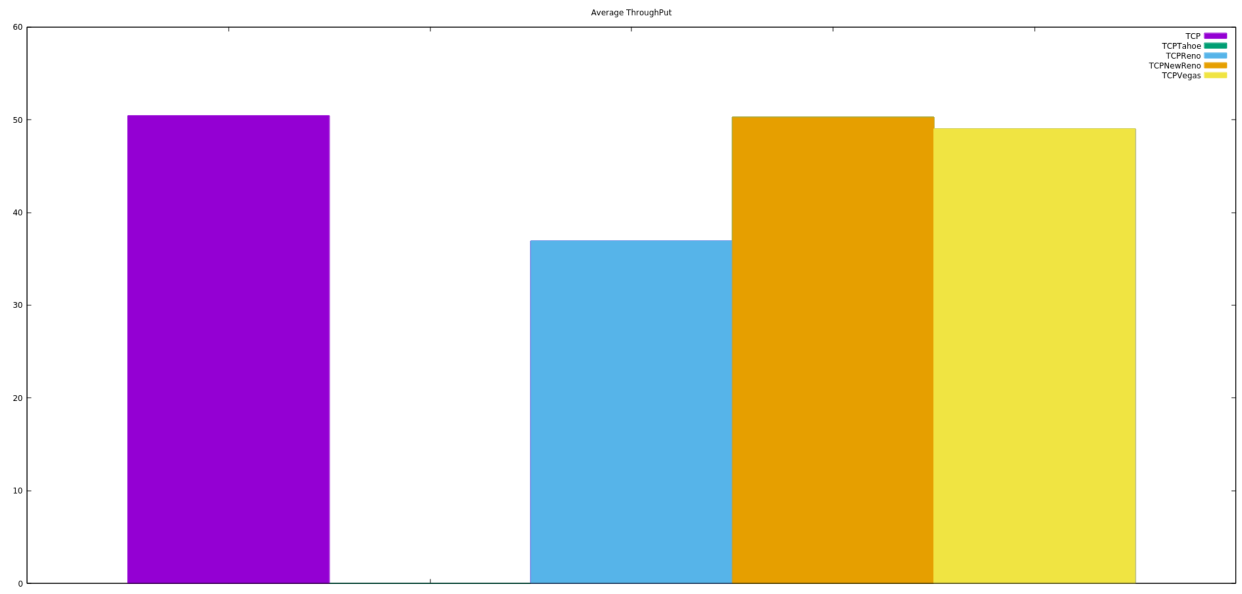


Figure 6.26 Scenario2 Average ThroughPut of TCP Variants over DSDV Routing Protocol

Packet Delivery Ratio of TCP Variants over DSDV Routing Protocol:

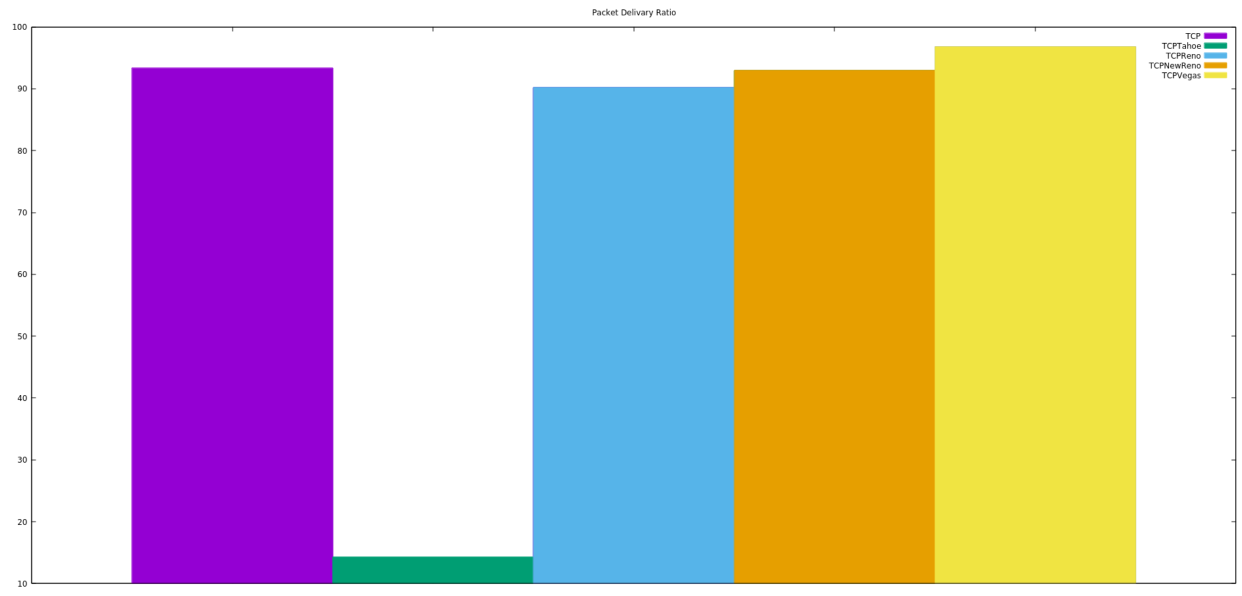


Figure 6.27 Scenario2 Packet Delivery Ratio of TCP Variants over DSDV Routing Protocol

Instantaneous end-to-end delay of TCP Variants over DSDV Routing Protocol:

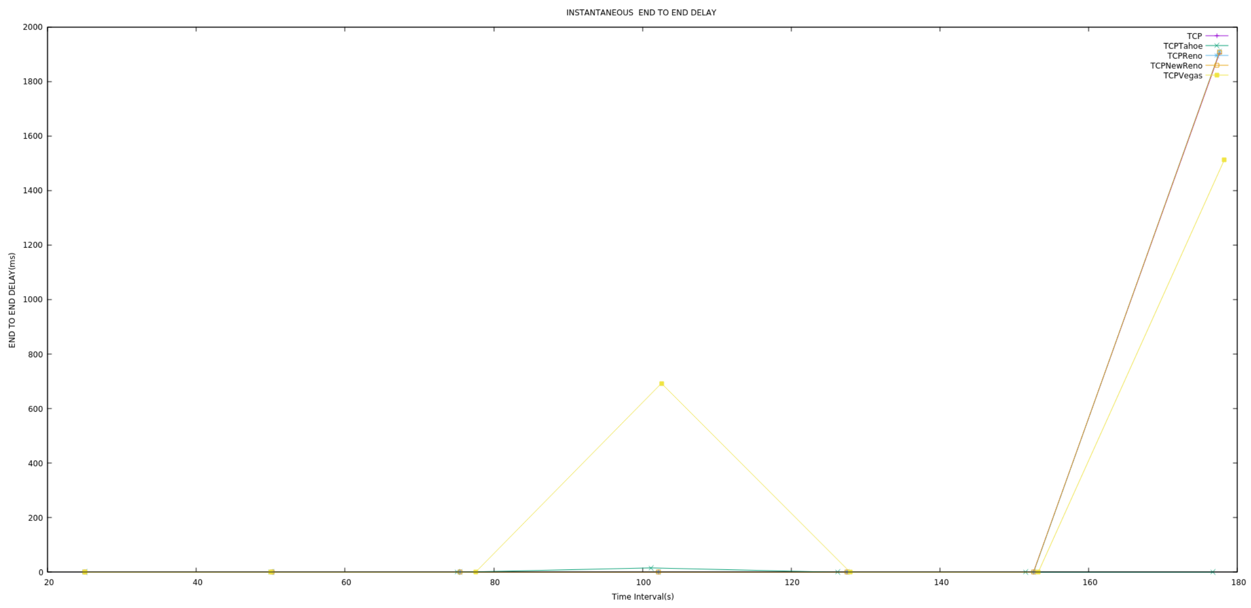


Figure 6.28 Scenario2 Instantaneous end-to-end delay of TCP Variants over DSDV Routing Protocol

Instantaneous ThroughPut of TCP Variants over DSDV Routing Protocol:

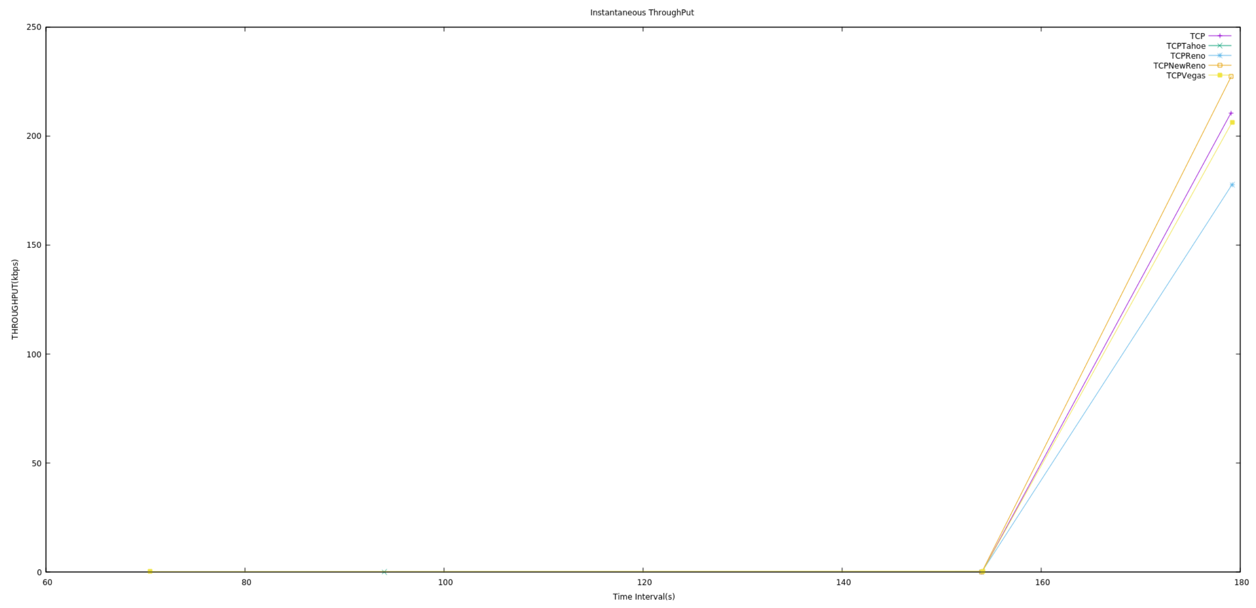


Figure 6.29 Scenario2 Instantaneous ThroughPut of TCP Variants over DSDV Routing Protocol

Congestion window of various TCP Variants over DSDV Routing Protocol:

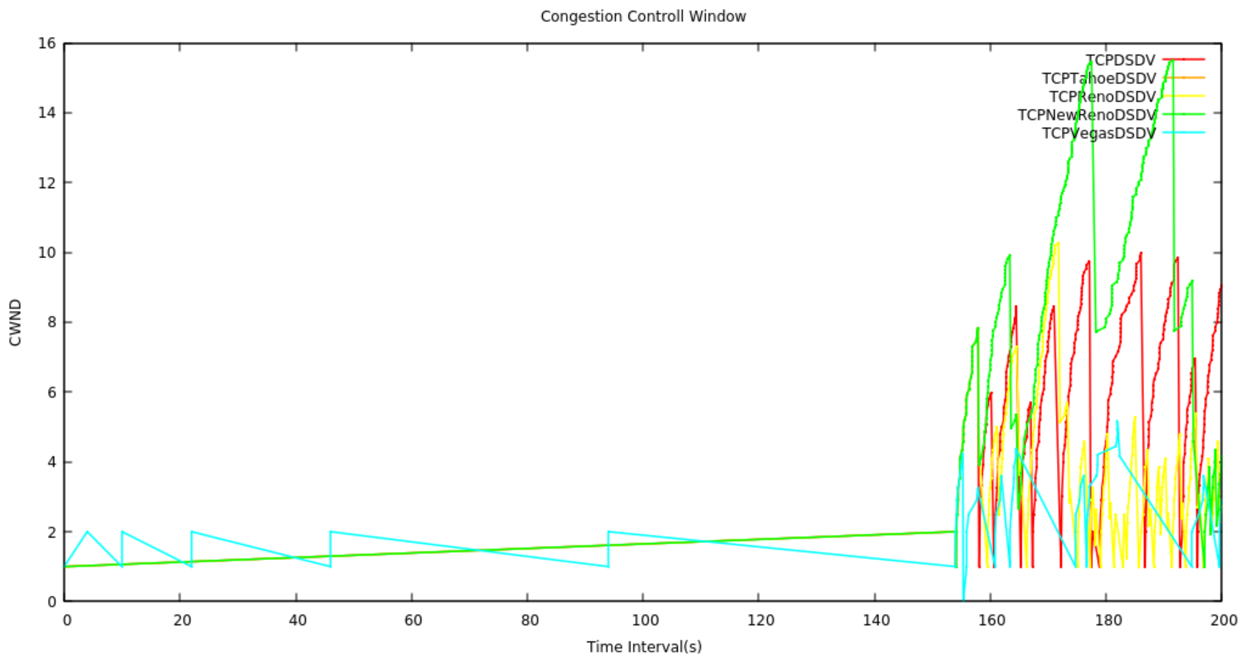


Figure 6.30 Scenario2 Congestion window of various TCP Variants over DSDV Routing Protocol

Average end-to-end delay of TCP Variants over DSR Routing Protocol:

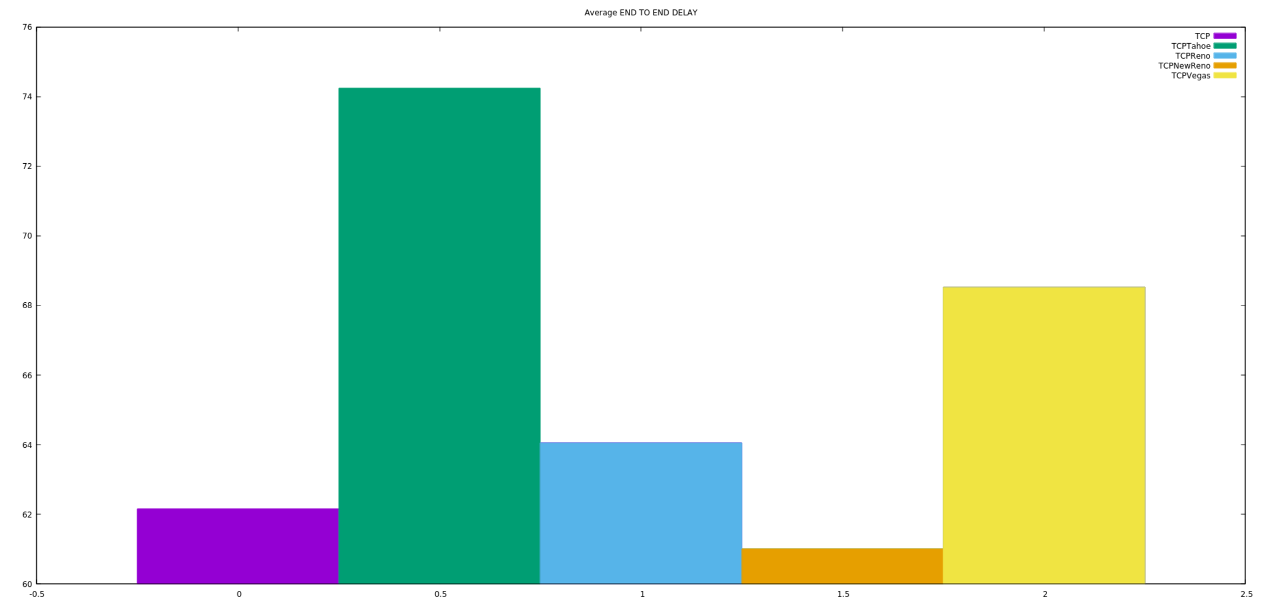


Figure 6.31 Scenario2 Average end-to-end delay of TCP Variants over DSR Routing Protocol

Average ThroughPut of TCP Variants over DSR Routing Protocol:

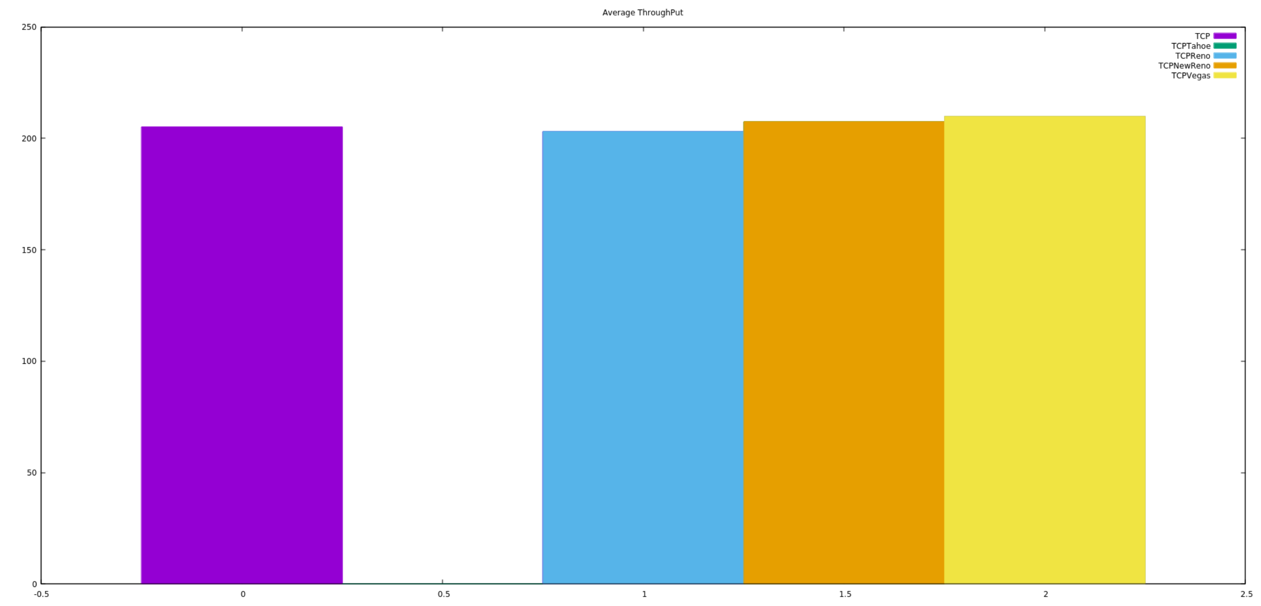


Figure 6.32 Scenario2 Average ThroughPut of TCP Variants over DSR Routing Protocol

Packet Delivery Ratio of TCP Variants over DSR Routing Protocol:

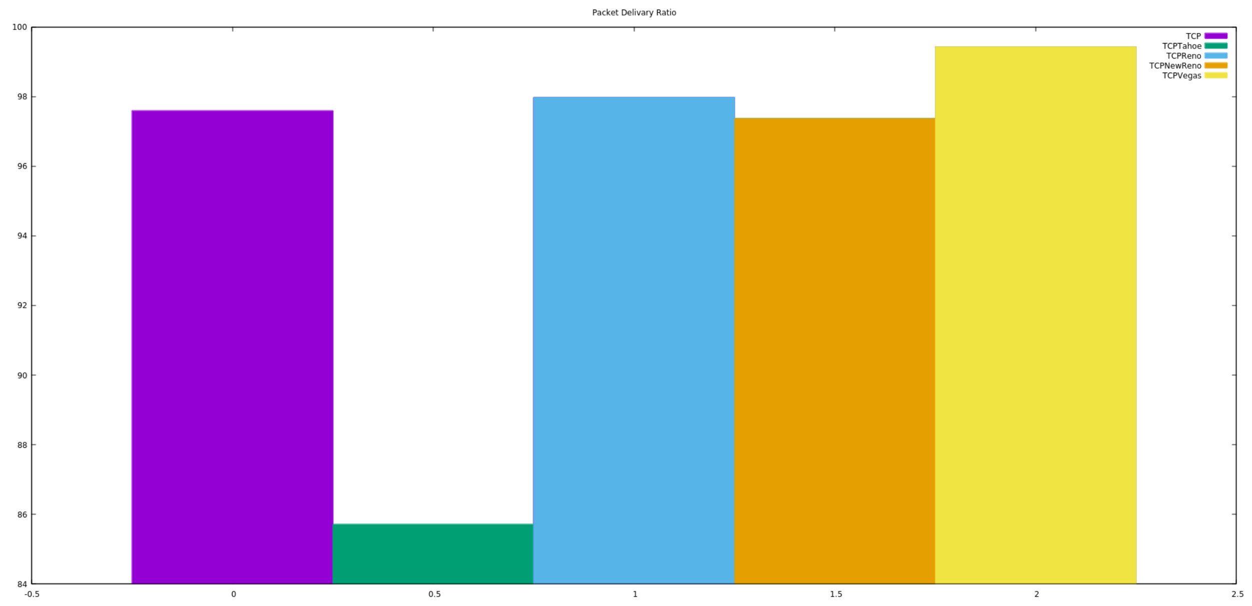


Figure 6.33 Scenario2 Packet Delivery Ratio of TCP Variants over DSR Routing Protocol

Instantaneous end-to-end delay of TCP Variants over DSR Routing Protocol:

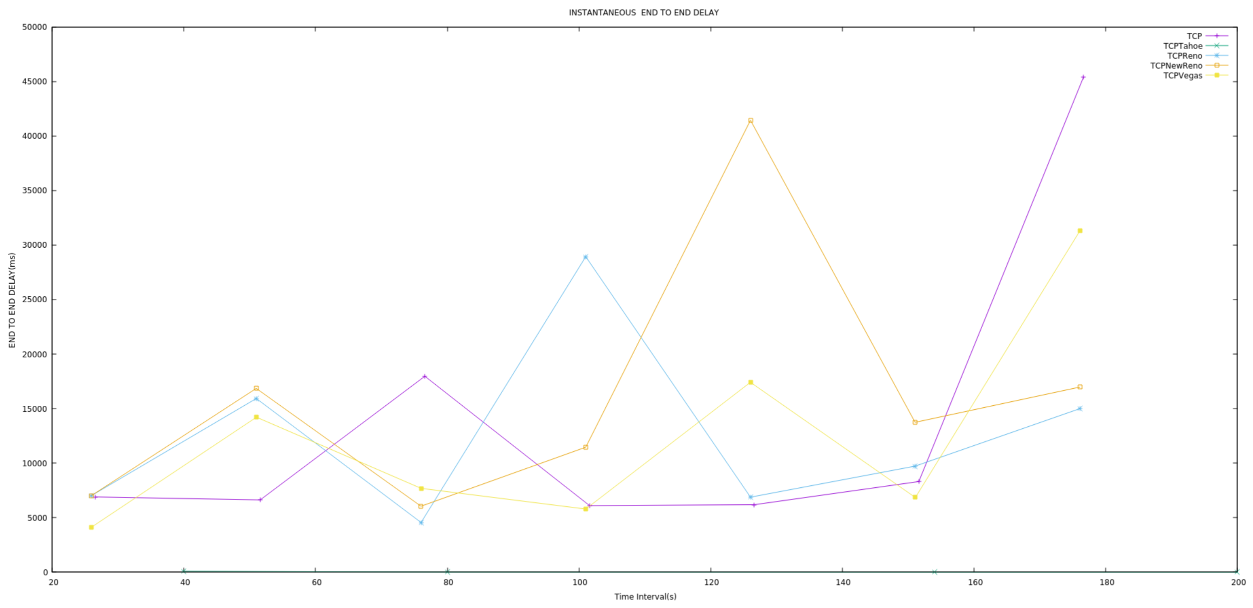


Figure 6.34 Scenario2 Instantaneous end-to-end delay of TCP Variants over DSR Routing Protocol

Instantaneous ThroughPut of TCP Variants over DSR Routing Protocol:

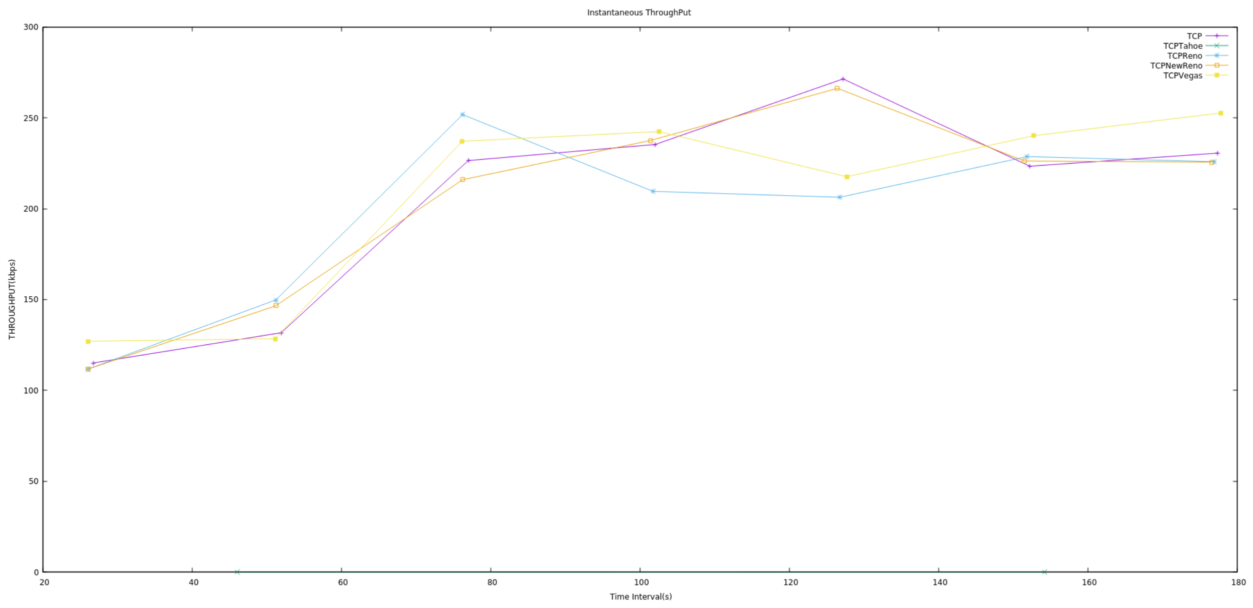


Figure 6.35 Scenario2 Instantaneous ThroughPut of TCP Variants over DSR Routing Protocol

Congestion window of various TCP Variants over DSR Routing Protocol:

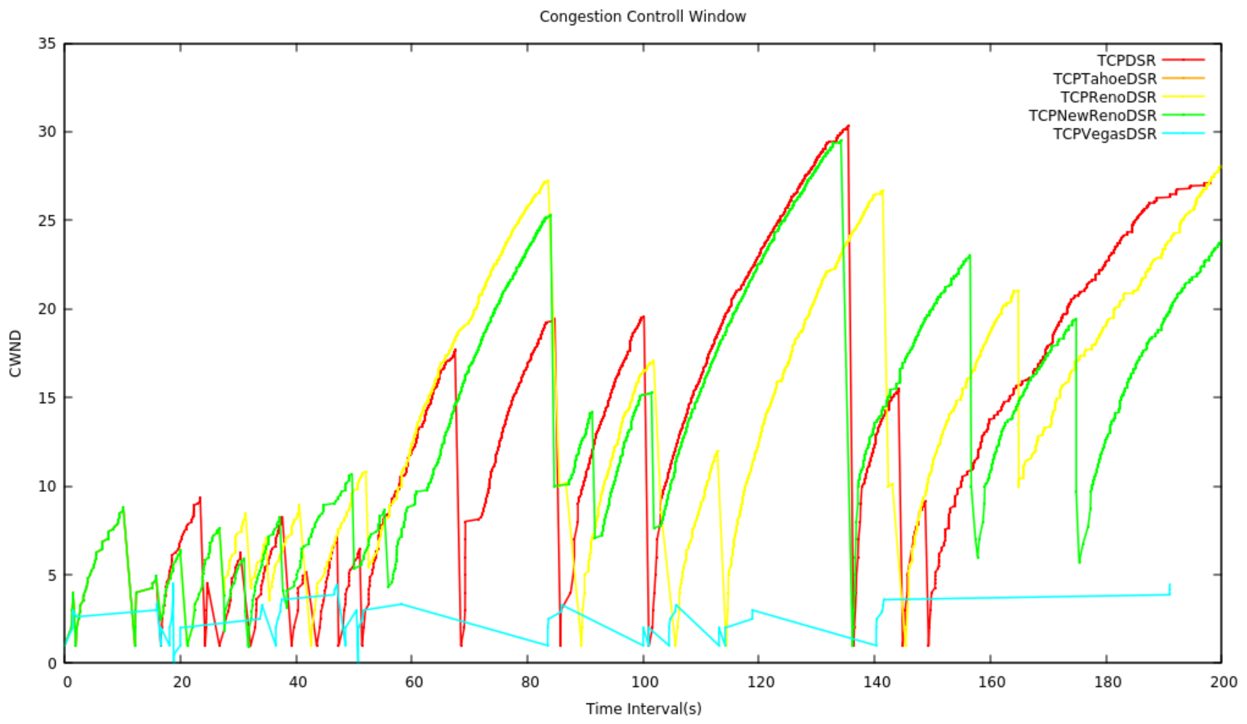


Figure 6.36 Scenario2 Congestion window of various TCP Variants over DSR Routing Protocol

Routing OverHead of various TCP Variants over DSR Routing Protocol:

	TCP	TCP Tahoe	TCP Reno	TCP NewReno	TCP Vegas
AODV	2.913	25.833	2.610	2.872	2.717
DSDV	1.947	923.000	2.647	1.955	2.048
DSR	0.922	19.833	0.999	0.779	0.779

Table 6.4 Scenario2 Routing OverHead of various TCP Variants over DSR Routing Protocol

Observation:

AODV

From figure 6.19,6.20 we could see that TCP Vegas performance was comparatively low when the number of Nodes increased and from figure 6.22 and 6.33 the during mobility the instantaneous throughput of TCP Reno and TCP Vegas has significantly decreased which impacted the average their average throughput.

TCP New Reno on the other hand did not experience degradation of throughput from factors such as increased number of nodes and mobility. From figure 6.24 it is found that TCP NewReno has utilized its fast-recovery mechanism to achieve high throughput. From figure 6.21 TCP New Reno packet delivery ratio is less compared to TCP Vegas and from increased routing overhead (table 6.4) we could say that there is needless transmission of packets from TCP New Reno in AODV.

DSDV

In DSDV when the number of nodes were increased, the throughput of nearly all the TCP Variants were decreased. From table 6.29 on analysing the instantaneous throughput it is found that TCP Variants began late as DSDV on larger number of nodes took time to setup the routing table due to mobility of nodes.From table 6.4 the routing overhead TCP and TCP NewReno has less routing overhead compared to other variants.Hence the TCP vegas performed slightly better when compared to other variants.

DSR

In DSR, when the number of nodes were increased, the throughput of nearly all the TCP Variants remained the same. From figure 6.32 and figure 6.34 TCP Vegas has performed well when compared to other TCP Variants and from figure 6.36 TCPReno on avoiding slow start state still did not perform well with other TCP Variants. The routing overhead (Table 6.4) shows that TCP variants on DSR have less routing overhead due to its less route discovery process.

SCENARIO-3:

In this scenario nearly all the nodes are mobile and have a velocity of 10m/s to 20m/s. The main aim of this scenario is to find out which TCP Variant in each routing protocol does better under large numbers of nodes and with increased mobility.

The simulation parameters for the first scenario are as follows:

PARAMETER	VALUE	
Simulation Environment	1000 x 500 sq.mt	
Transmission range	250 m	
Inference range	550 m	
Number of nodes	60	
Simulation time	200 sec	
MAC Protocol	IEEE 802.11	
Speed of each Node	min	10 m/s
	max	20 m/s
Topography	flatgrid	
Node Mobility	cmu-scen-gen	

Table-6.5 Simulation Parameters for Scenario 3

Average end-to-end delay of TCP Variants over AODV Routing Protocol:

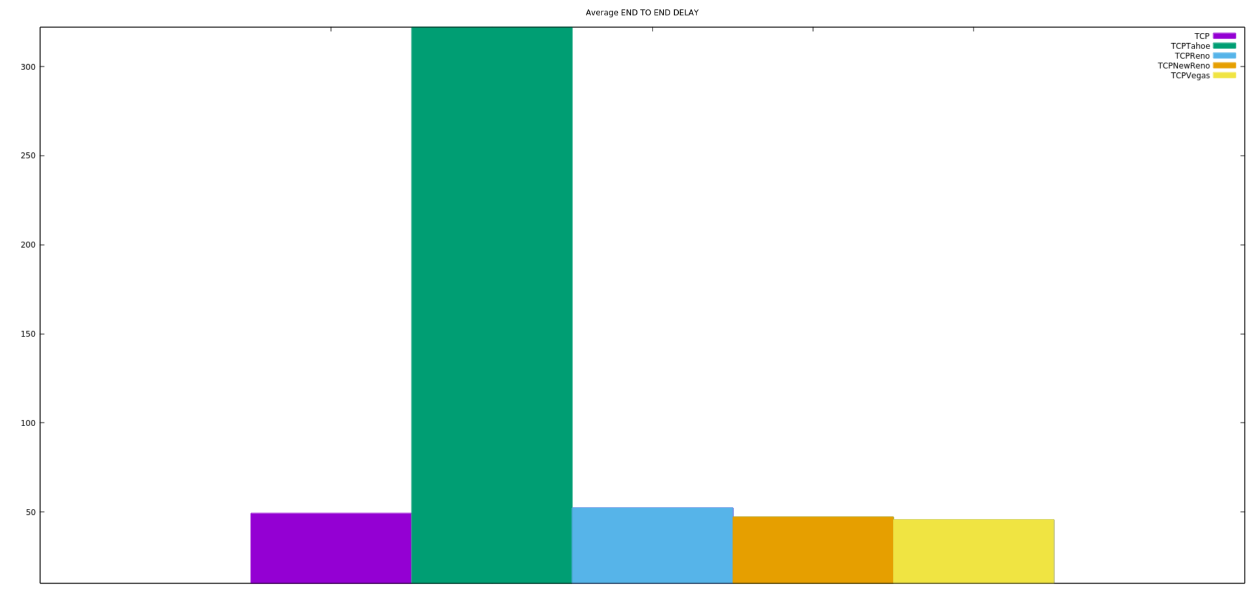


Figure 6.37 Scenario3 Average end-to-end delay of TCP Variants over AODV Routing Protocol

Average ThroughPut of TCP Variants over AODV Routing Protocol:

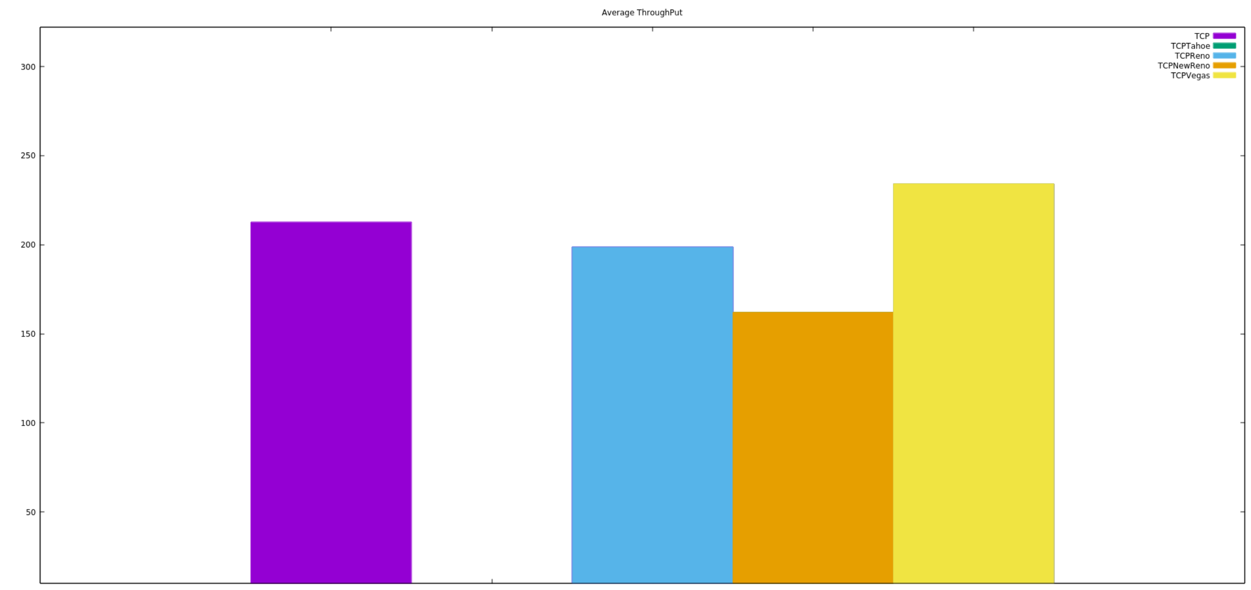


Figure 6.38 Scenario3 Average ThroughPut of TCP Variants over AODV Routing Protocol

Packet Delivery Ratio of TCP Variants over AODV Routing Protocol:

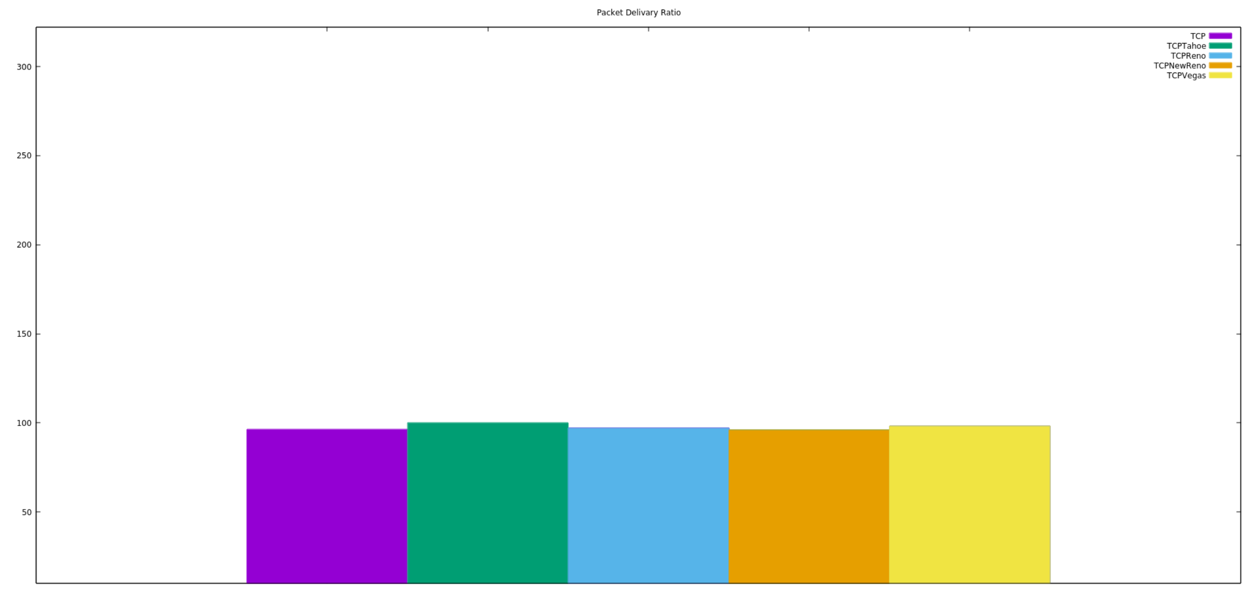


Figure 6.39 Scenario3 Packet Delivery Ratio of TCP Variants over AODV Routing Protocol

Instantaneous end-to-end delay of TCP Variants over AODV Routing Protocol:

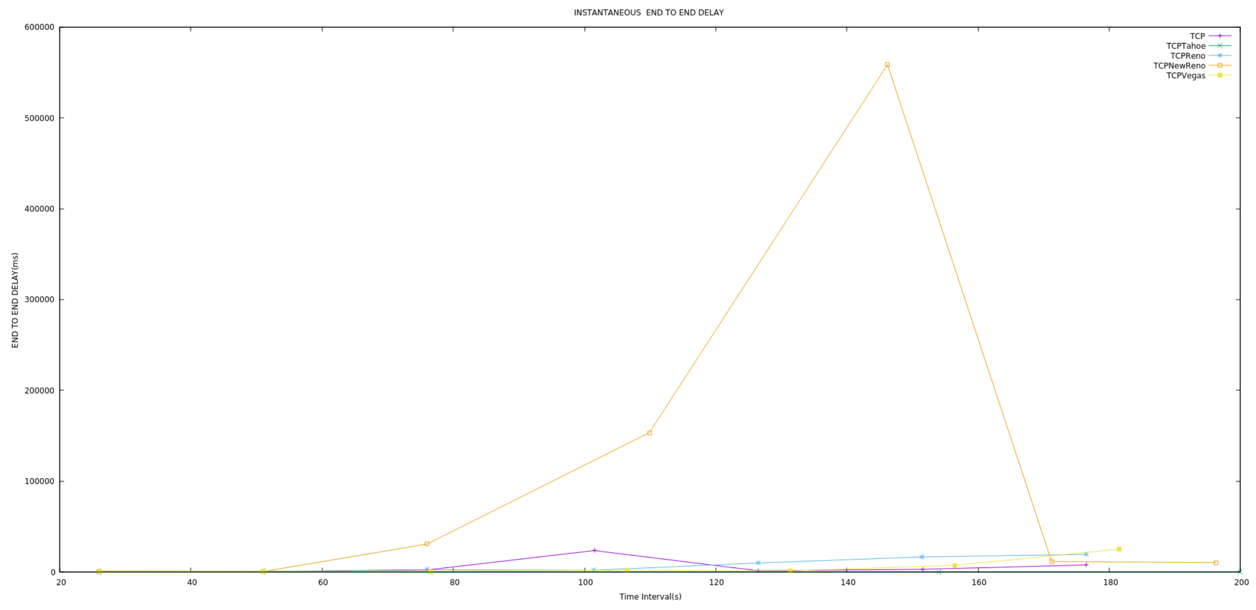


Figure 6.40 Scenario3 Instantaneous end-to-end delay of TCP Variants over AODV Routing Protocol

Instantaneous ThroughPut of TCP Variants over AODV Routing Protocol:

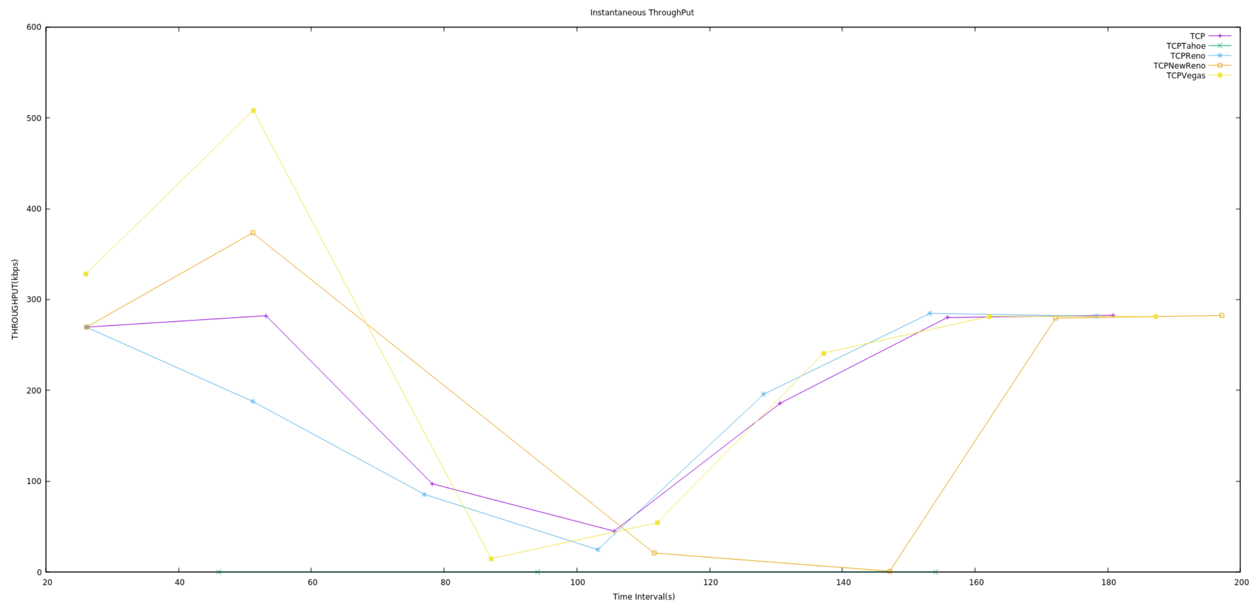


Figure 6.41 Scenario3 Instantaneous ThroughPut of TCP Variants over AODV Routing Protocol

Congestion window of various TCP Variants over AODV Routing Protocol:

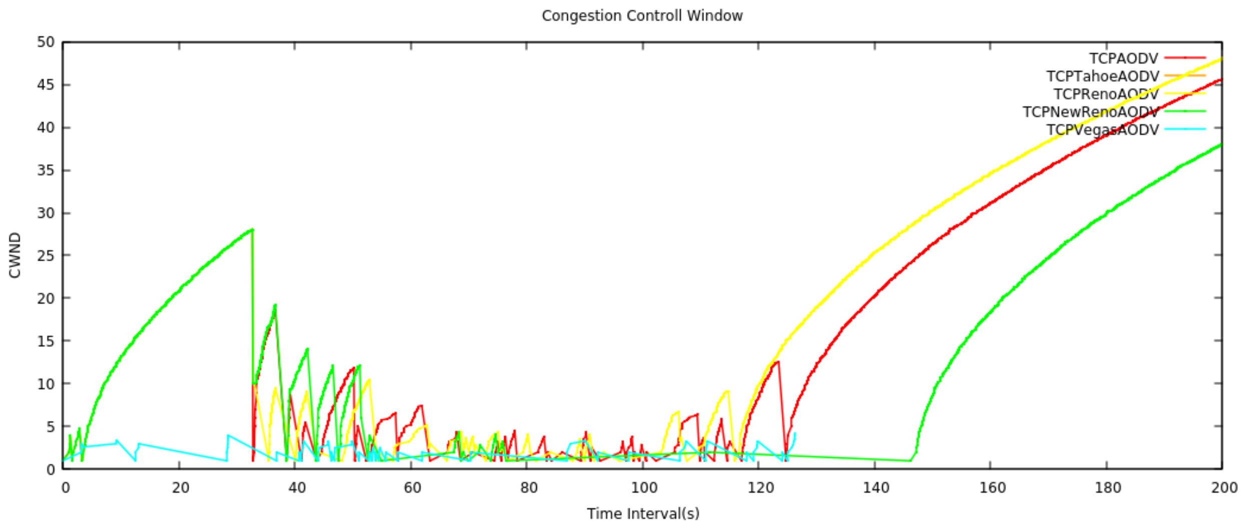


Figure 6.42 Scenario3 Congestion window of various TCP Variants over AODV Routing Protocol

Average end-to-end delay of TCP Variants over DSDV Routing Protocol:

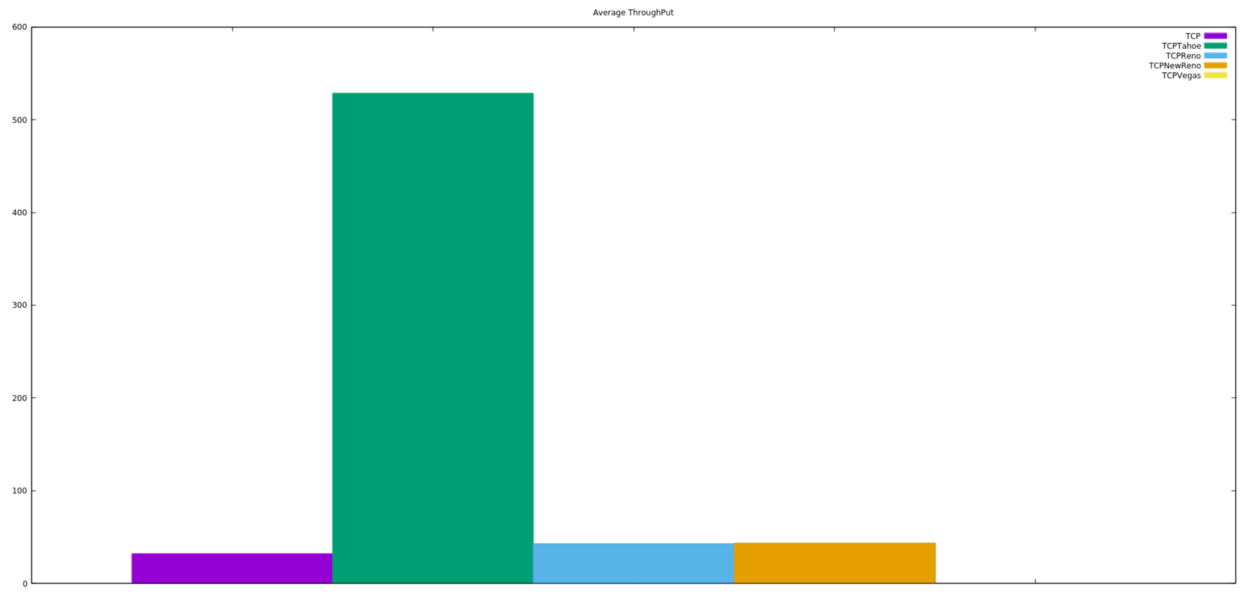


Figure 6.43 Scenario3 Average end-to-end delay of TCP Variants over DSDV Routing Protocol

Average ThroughPut of TCP Variants over DSDV Routing Protocol:

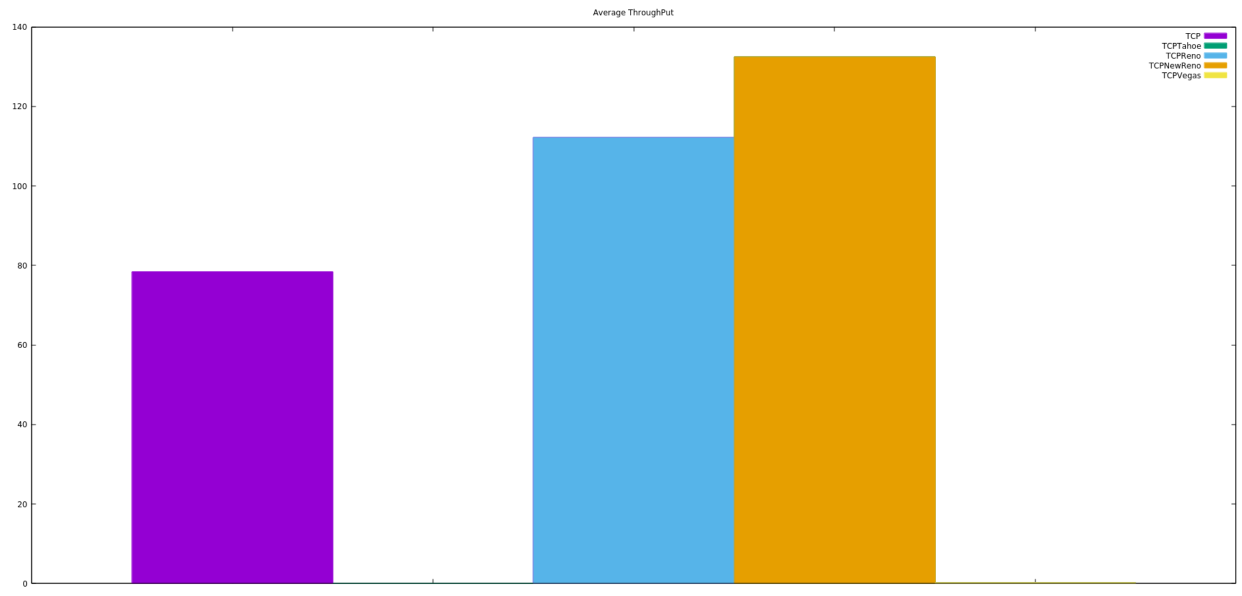


Figure 6.44 Scenario3 Average ThroughPut of TCP Variants over DSDV Routing Protocol

Packet Delivery Ratio of TCP Variants over DSDV Routing Protocol:

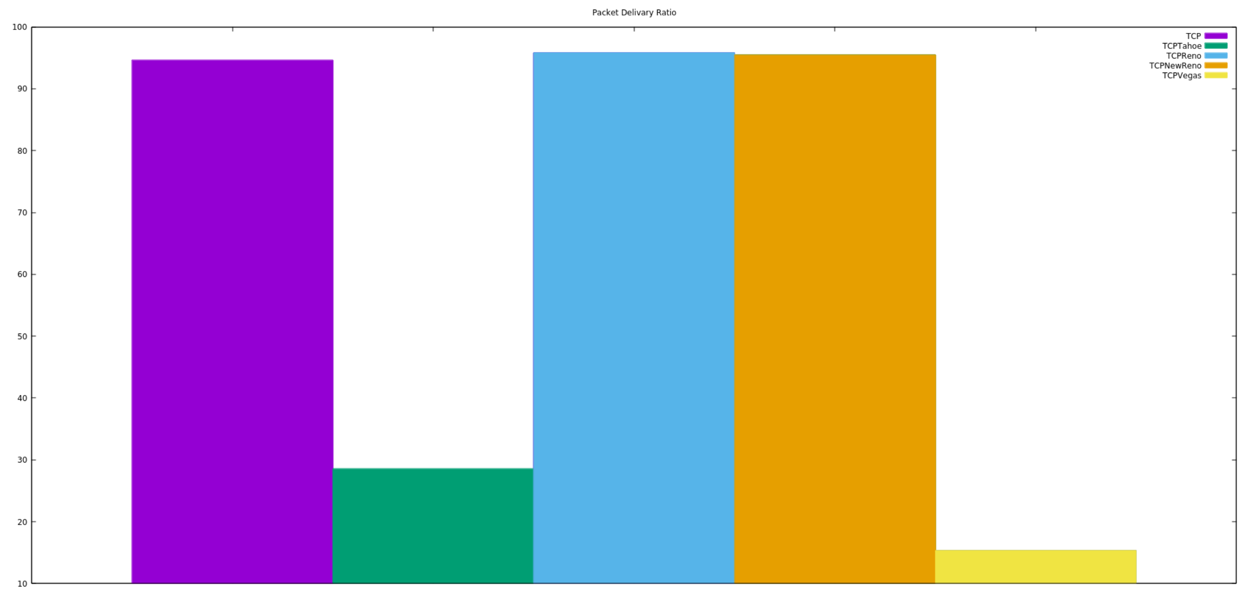


Figure 6.45 Scenario3 Packet Delivery Ratio of TCP Variants over DSDV Routing Protocol

Instantaneous end-to-end delay of TCP Variants over DSDV Routing Protocol:

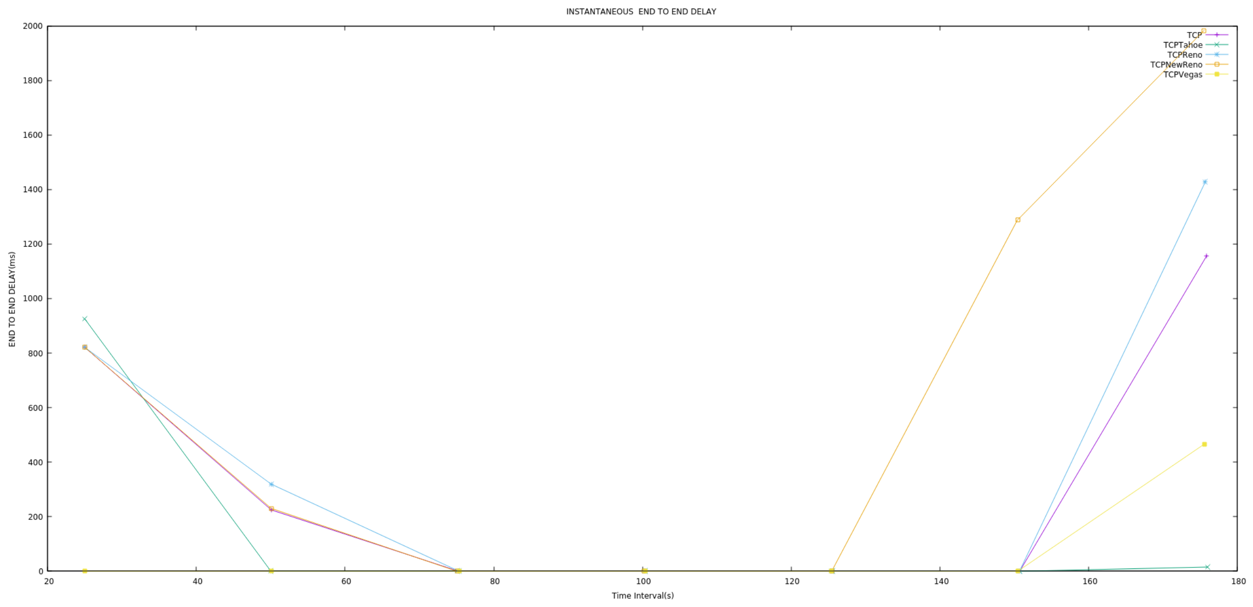


Figure 6.46 Scenario3 Instantaneous end-to-end delay of TCP Variants over DSDV Routing Protocol

Instantaneous ThroughPut of TCP Variants over DSDV Routing Protocol:

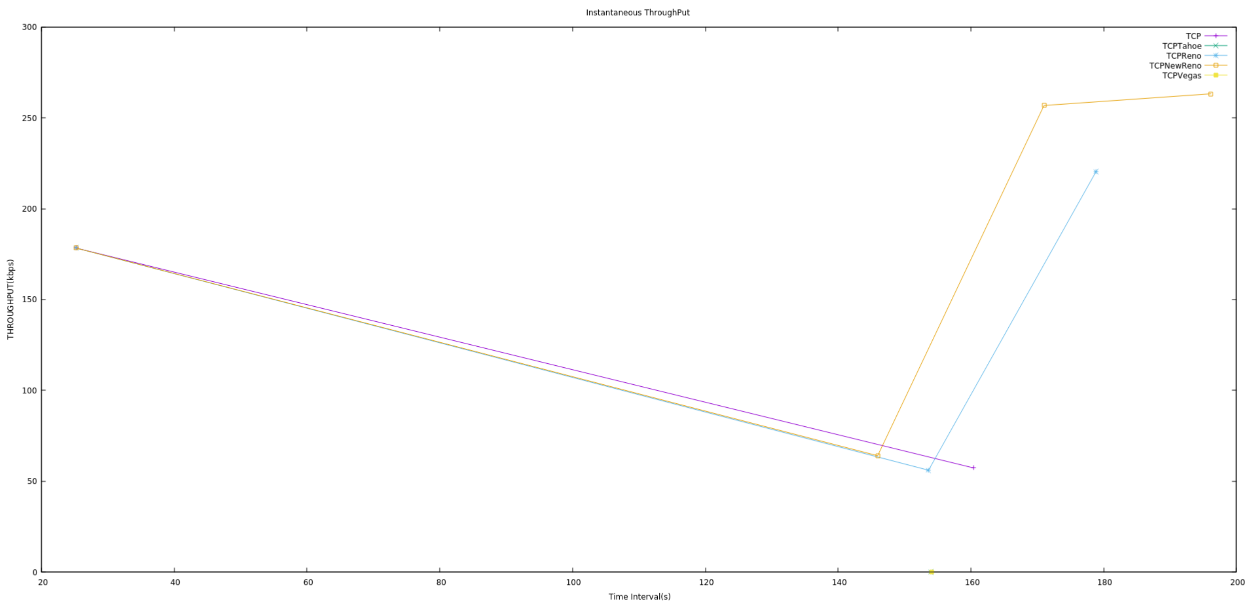


Figure 6.47 Scenario3 Instantaneous ThroughPut of TCP Variants over DSDV Routing Protocol

Congestion window of various TCP Variants over DSDV Routing Protocol:

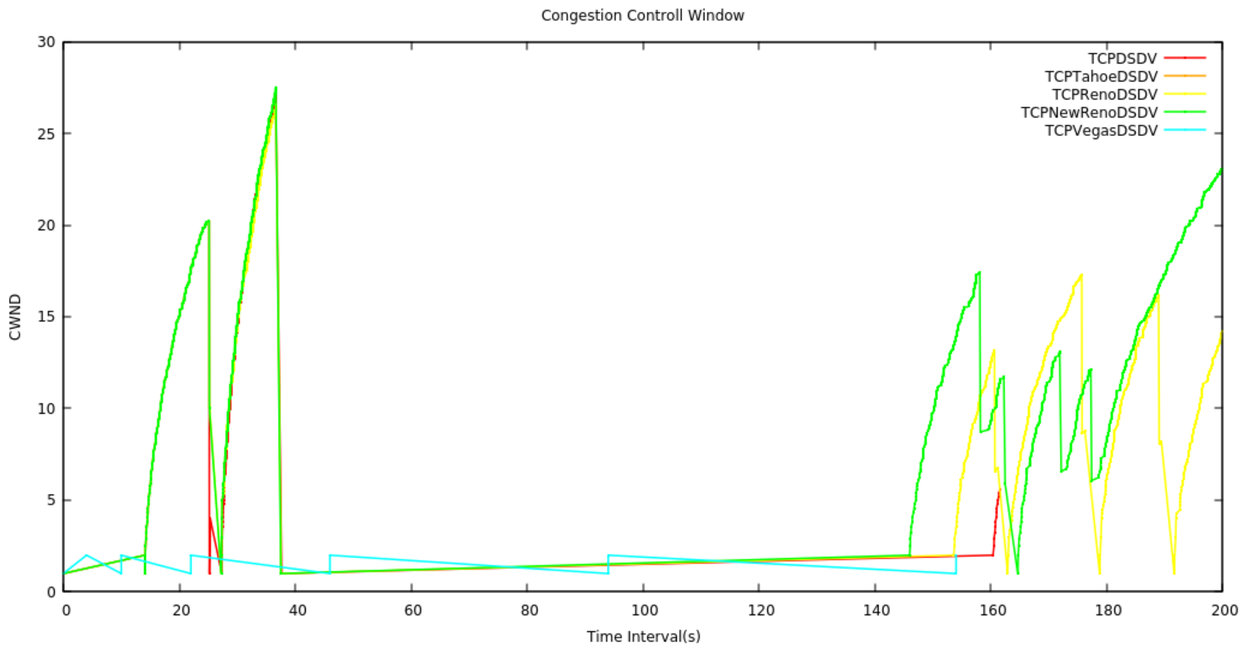


Figure 6.48 Scenario3 Congestion window of various TCP Variants over DSDV Routing Protocol

Average end-to-end delay of TCP Variants over DSR Routing Protocol:

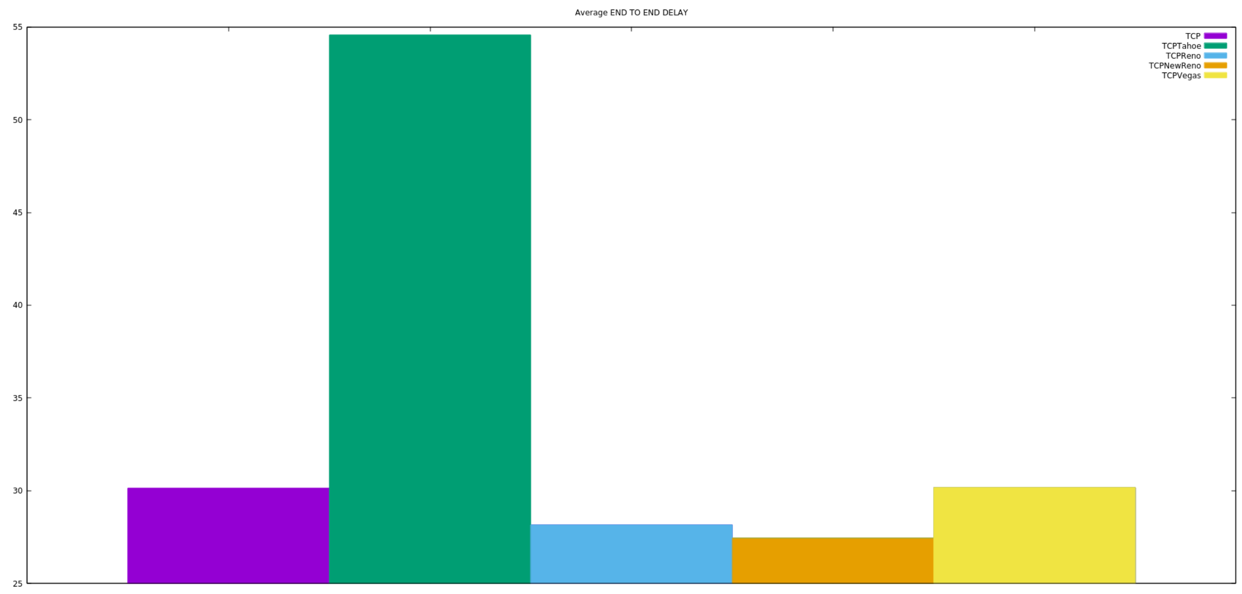


Figure 6.49 Scenario3 Average end-to-end delay of TCP Variants over DSR Routing Protocol

Average ThroughPut of TCP Variants over DSR Routing Protocol:

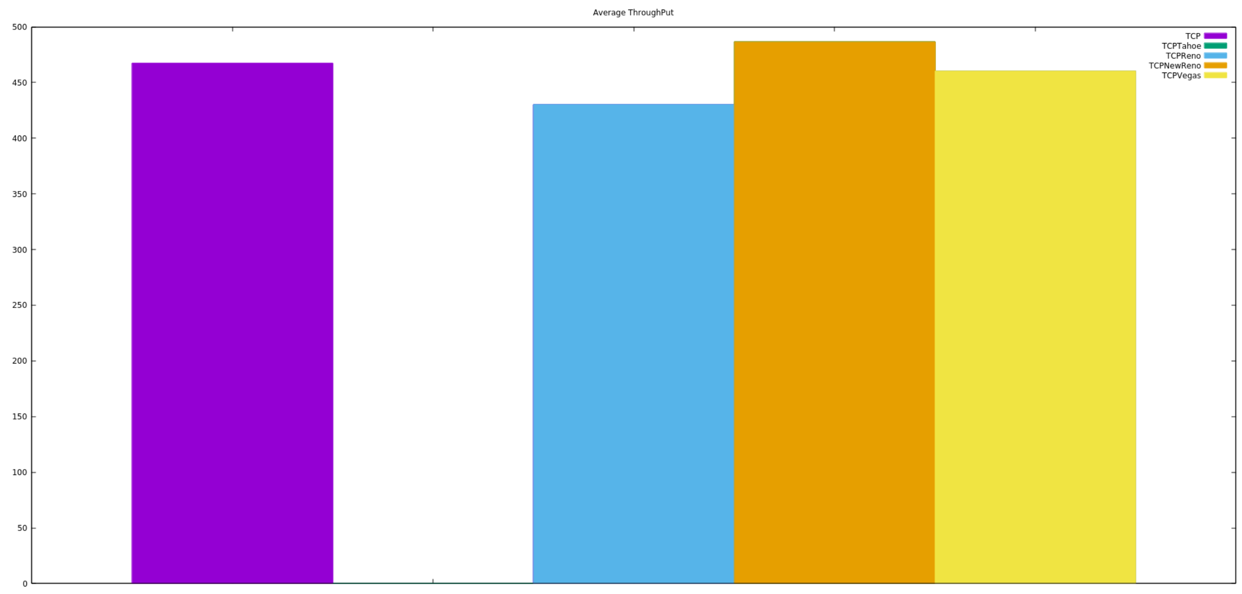


Figure 6.50 Scenario3 Average ThroughPut of TCP Variants over DSR Routing Protocol

Packet Delivery Ratio of TCP Variants over DSR Routing Protocol:

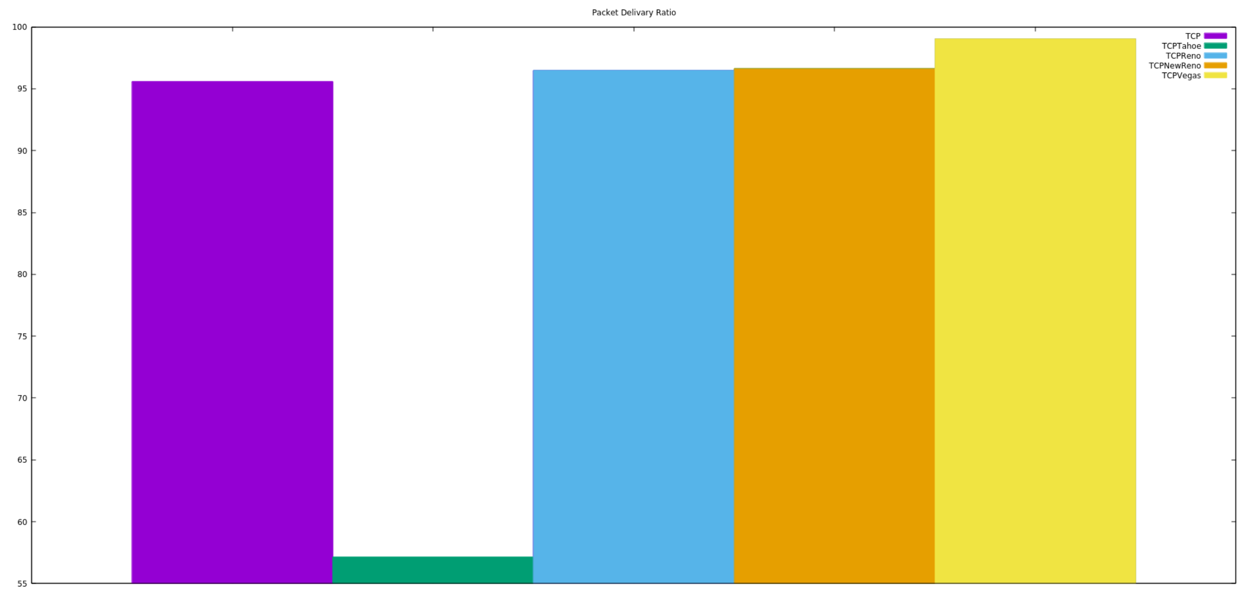


Figure 6.51 Scenario3 Packet Delivery Ratio of TCP Variants over DSR Routing Protocol

Instantaneous end-to-end delay of TCP Variants over DSR Routing Protocol:

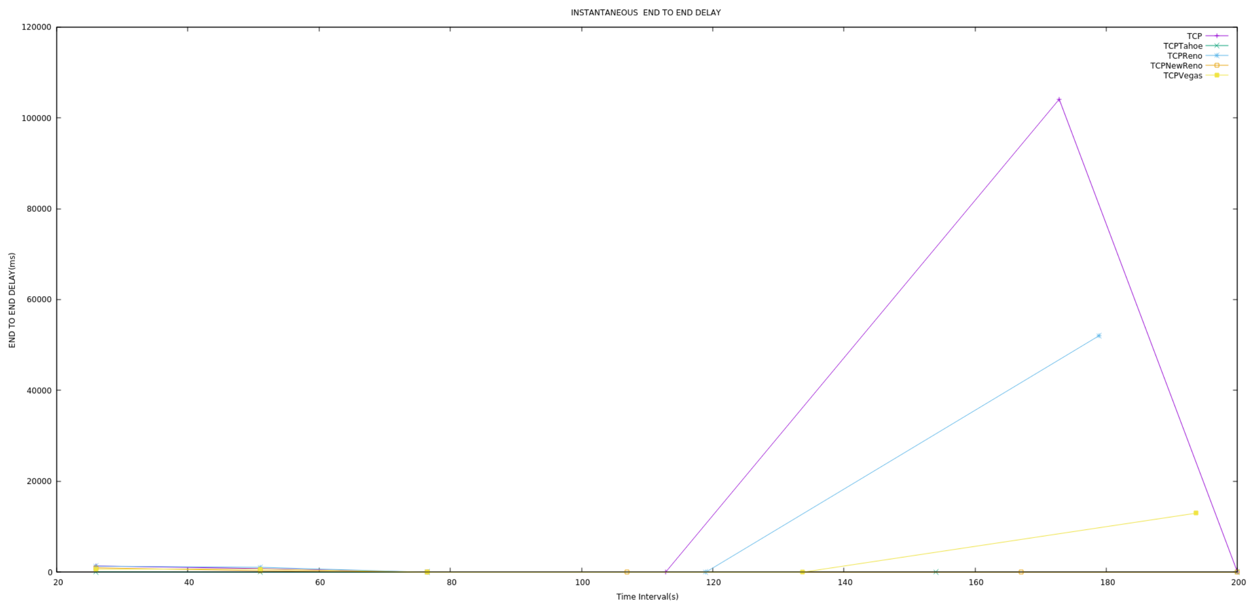


Figure 6.52 Scenario3 Instantaneous end-to-end delay of TCP Variants over DSR Routing Protocol

Instantaneous ThroughPut of TCP Variants over DSR Routing Protocol:

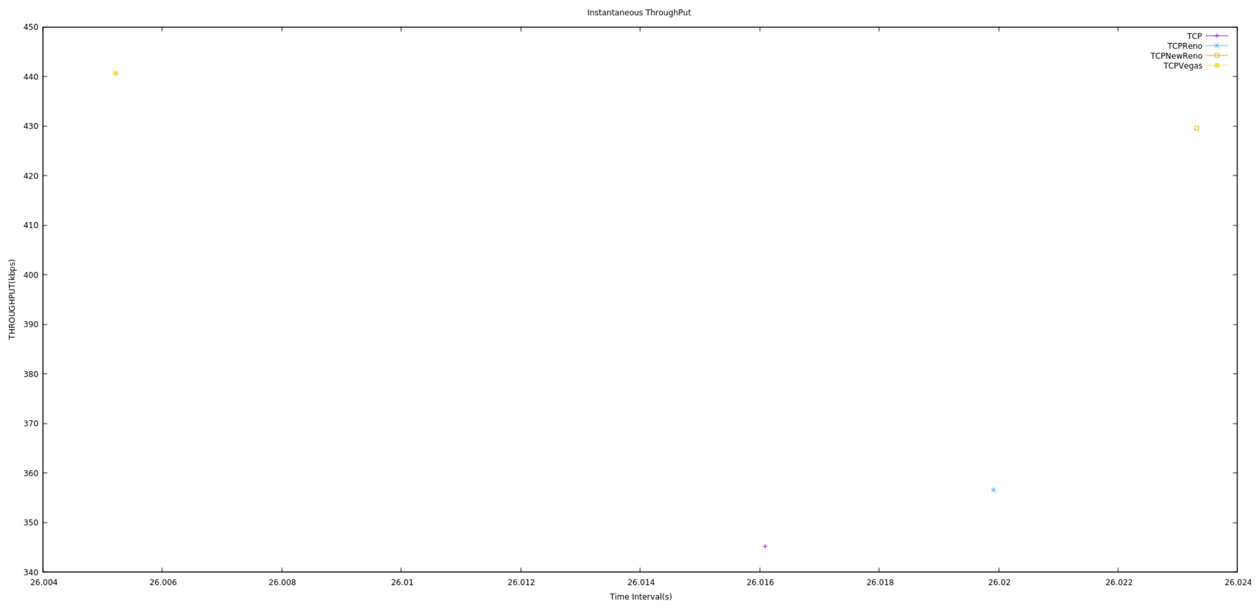


Figure 6.53 Scenario3 Instantaneous ThroughPut of TCP Variants over DSR Routing Protocol

Congestion window of various TCP Variants over DSR Routing Protocol:

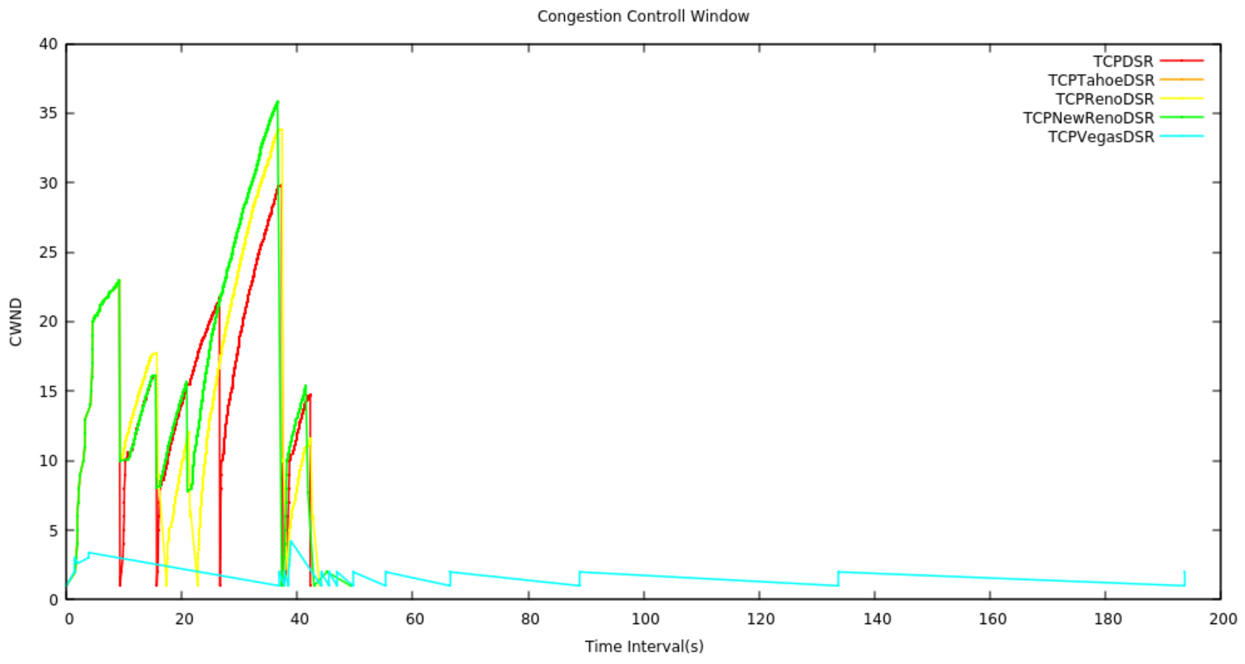


Figure 6.54 Congestion window of various TCP Variants over DSR Routing Protocol

Routing OverHead of various TCP Variants over DSR Routing Protocol:

	TCP	TCP Tahoe	TCP Reno	TCP NewReno	TCP Vegas
AODV	2.434	49.857	2.348	1.366	1.133
DSDV	4.110	1255.500	2.260	1.996	1226.000
DSR	0.372	37.750	0.399	0.341	0.236

Table 6.6 Scenario3 Routing OverHead of various TCP Variants over DSR Routing Protocol

Observation:

AODV

From figure 6.41 nearly all TCP variants throughput were drastically decreased due to mobility of nodes and from Figure 6.42 we could see that due to heavy packet loss the CWND has not increased properly for all the TCP Variants

DSDV

In DSDV the TCP Vegas performed worse when compared to other TCP Variants (Figure 6.44 and Figure 6.45).This could also be verified using Table 6.6. On the other hand TCP NewReno performed comparatively better with respect to throughput and packet delivery ratio.

DSR

TCP Vegas performed best under DSR Routing protocol based on throughput and packet delivery ratio and its routing overhead was also less.

CHAPTER 7
CONCLUSION

From analysing the conclusion from three different scenarios of five different TCP variants over three Manet Routing Protocol we could say that different TCP Variants have pros and cons in different scenarios.

From scenario 1 we could understand that for a small number of nodes and less mobility TCP Vegas performed best among AODV and DSR routing protocols and TCP New Reno Performed well for DSDV routing protocol.

From scenario 2 we could understand that without high mobility of different nodes TCP Vegas performed best in DSDV and DSR and TCP New Reno performed well with AODV.

From scenario 3 we could understand that high mobility of nodes affected TCP variants' performance which could be seen in their instantaneous throughput and congestion window. Still TCP New Reno performed best among the three MANETs routing protocols.

FUTURE PLANS:

- Comparison of Performance analysis of TCP Variants on larger number nodes.
- Extending the current scenarios to find the performance analysis from hybrid routing protocols.
- Comparison based on varying packet size and delay.

CHAPTER 8
REFERENCES

Rajnesh Singh, Neeta Singh, Aarti G. Dinker, “Performance Analysis of TCP Variants Using AODV and DSDV Routing Protocols in MANETs”, Recent Advances in Computer Science and Communications, DOI: 10.2174/2666255813666190911114130

https://en.wikipedia.org/wiki/Additive_increase/multiplicative_decrease

https://en.wikipedia.org/wiki/TCP_congestion_control

https://en.wikipedia.org/wiki/TCP_congestion_control#Slow_start