Project Report on

Performance Analysis of TCP Variants and Routing Protocols Using NS2 Simulations

Submitted to:

Dr. Melike Erol-Kantarci in partial fulfillment of the requirements for course ELG 5374

Submitted by: -

Amarjit Singh Dhillon
Master of Applied Science
Carleton University *amarjitsingh@cmail.carleton.ca*
Student # 101024549

Ranjit Singh Saini
Masters of Electrical and Computer Engineering
Carleton University *Ranjitsaini@cmail.carleton.ca*
Student # 101024552

1. Table of contents

Introduction:

Routing in an Ad-hoc network is a challenging task because source and destination nodes are mobile and thus routing decisions are to be changed dynamically when link failure or packet delay is encountered. As TCP protocols were initially designed for wired networks, so they are not able to deliver optimized performance, in the case of ad-hoc networks. For ensuring a reliable transfer, various variants of TCP must be used such as TCP-Reno, TCP-Vegas, TCP-Westwood, TCP-New Reno, TCP-Tahoe, TCP-Sack etc. Mobile ad-hoc network is a decentralized network consisting of various mobile nodes. Challenges in MANETS routing includes lack of *apriori* knowledge of underlying topology, which requires using the adaptive protocol to tackle route failures and packet loss scenarios.

Ns2 is a Discrete event driven simulator which uses TCL as scripting language which will generate trace files that are analyzed using TCL and **awt** scripts and **nam** is used to visually show the simulation file using generated **nam** file.

Simulation Parameters: The parameter for all the simulation result is below.

Antenna type	Omni Antenna
Application agent	FTP
Channel type	Channel/Wireless Channel
Interface type	Phy/WirelessPhy
Mac layer protocol	IEEE 802.11
Network area	600 * 600
Number of nodes	50
Propagation type	Propagation/TwoRayGround
Queue length	50 Packets
Queue type	Queue/Droptail/PriorityQueue
Routing protocols	AODV DSR DSDV
Simulation time	50 seconds
Simulator	ns2-allinone 2.35
Speed	10m/s
Transmission range	250 m
Transport agent	1.TCP/Reno 2.TCP/Newreno 3.TCP/Sack 4.TCP/Vegas 5.TCP/Fack 6.TCP/Linux

Steps for Running Simulation:
Now we will explain the steps that has been performed in our simulation.

Some of imperative files in this simulation are as below

1. Start_program.tcl
2. Mobility.tcl
3. Topology.tcl
4. rp_graph_generator_pdr.tcl
5. tcp_graph_generator_pdr.tcl
6. rp_graph_generator_delay.tcl
7. tcp_graph_generator_delay.tcl
8. delay_calculator.awk
9. pdr_caculator.awk
10. pdr_drop.awk

File name **start_program.tcl**

```
# Setting the variables intially
set val(anteena_type_used)                Antenna/OmniAntenna          ;
set val(ifqlen)                           50                           ;
set val(ll)                               LL                           ;
set val(mobile_nodes_count)               50                           ;
set val(propagation_model)                Propagation/TwoRayGround     ;
set val(queuing_model)                    Queue/DropTail/PriQueue      ;
set val(length_of_topography)             600                          ;
set val(simulation_stop_time)              50                          ;
set val(type_of_channel)                  Channel/WirelessChannel      ;
set val(type_of_interface)                Phy/WirelessPhy              ;
set val(type_of_mac)                      Mac/802_11                   ;

set val(type_of_routing_protocol)         AODV                         ;
set val(width_of_topography)              600                          ;
```

--- Providing initial Parameters in simulation ---

Initially, we have described all the parameters that we are using in our simulation. Simulation is using omni Antenna and drop tail queuing model is used having max queue length as 50 nodes. Also, total nodes are 50 which are mobile and are described in 600*600 meter topology. Mac 802.11 protocol is used and we are having wireless interface among nodes

```
puts "\n Please enter the Protocol no. Which you want to Use.\n 1.AODV \n 2.DSR \n 3.DSDV \n Enter Your Value::"
flush stdout
set protocol_name [gets stdin]

if {$protocol_name == 1} {
   set val(type_of_routing_protocol) AODV;
   puts "\nYou have selected AODV routing protocol \n"

} elseif {$protocol_name == 2} {
  set val(type_of_routing_protocol) DSR;
     puts "You have selected DSR routing protocol \n"

  } elseif {$protocol_name == 3} {
    set val(type_of_routing_protocol) DSDV;
    puts "You have selected DSDV routing protocol \n"

    }  else {
error "\n Please renter the correct protocol name. You should enter the numerical value provided in the above description."
   exit
}
```

```
#we ask the user to enter the tcp variant
puts "------------------------------------------------------------------------"
puts "\n Please enter the TCP variant Which you want to Use. \n
1.TCP/Reno
2.TCP/Newreno
3.TCP/Sack1
4.TCP/Vegas
5.TCP/Fack
6.TCP/Linux"
puts "enter Your value for tcp variant::"
```

Then the user is prompted to select the type of routing protocol which he/she want to use. We have provided 3 routing protocols, which can be selected by inputting [1-3]. if a value other than [1-3] is given then program will quit by showing the appropriate error

Similarly, then the user will be prompted to select the type of TCP variant, he or she wants to use. if a value other than 1-6 is given then the program will quit by showing the appropriate error

Then switch cases will select the type of TCP variant that we have used and will set the appropriate variable name accordingly.

```
switch $tcp_name_variant {
 1 {
   set val(tcp_variant_input) Reno;
   }
 2 {
  set val(tcp_variant_input) Newreno;
  }
  3 {
  set val(tcp_variant_input) Sack1;
  }
  4 {
  set val(tcp_variant_input) Vegas;

   }
  5 {
  set val(tcp_variant_input) Fack;
  }

 6 {
  set val(tcp_variant_input) Linux;
  }
 default {

error "Please provide the correct tcp variant name. You should enter the numerical value provided in the above description."
   exit 1
```

```
set ns [new Simulator]

# We open the trace file here
set filenamez $val(type_of_routing_protocol)_$val(tcp_variant_input)
set trace_file_path trace_files/$val(type_of_routing_protocol)_$val(tcp_variant_input).tr
set nam_file_path nam_files/$val(type_of_routing_protocol)_$val(tcp_variant_input).nam

set trace_file [open $trace_file_path  w+]
# now we open the namtrace_file file to write

set namtrace_file [open $nam_file_path w]

# we instruct the simulator object ns to save all simulation data into the trace
$ns trace-all $trace_file

#This command sets up wireless nam tracing. <X> and <Y> are the x-y co-ordinates for the wireless
$ns namtrace-all-wireless $namtrace_file $val(length_of_topography) $val(width_of_topography)
```

1.Now firstly ns is created as network simulator object

2.Then we set the filename variable filenamez that we will use to name a particular type of simulation

3.Trace_file_path is set based upon a particular combination of routing protocol and TCP variant that we have used. The name of this file ends with .tr extension , which means a trace file.

4.nam_file_path is set based upon a particular combination of routing protocol and TCP variant that we have used. The name of this file ends with .nam extension . this file is used by netanim to show the visual trace of the simulation.

5.set trace_file [open $trace_file_path w+] : it open .tr file in write mode , so as to write the trace file into it, which we will use later.

6.set namtrace_file [open $nam_file_path w] : it will open the .nam file in write mode .

7.$ns trace-all $trace_file : it enables the all trace feature in ns-2 . this setting can also be changed, depending upon the level of detail we want in our results.

8.$ns namtrace-all-wireless $namtrace_file $val(length_of_topography) $val(width_of_topography) : This command sets up wireless nam tracing. <X> and <Y> are the x-y co-ordinates for the wireless topology and all wireless nam traces are written into the <namtracefile>.

```
set topo [new Topography]
#This initializes the grid for the topography object. <X> and <Y> are the
$topo load_flatgrid $val(length_of_topography) $val(width_of_topography)
# used to create a God instance. The number of mobilenodes is passed as a
set god_ [create-god $val(mobile_nodes_count)]

set channel_to_use [new $val(type_of_channel)]
```

set topo [new Topography]: his initializes the grid for the topography object. <X> and <Y> are the x-y co-ordinates for the topology and are used for sizing the grid. The grid resolution may be passed as <res>. A default value of 1 is normally used.

$topo load_flatgrid $val(length_of_topography) $val(width_of_topography) : used to create a God instance. The number of mobile nodes is passed as an argument which is used by God to create a matrix to store connectivity information of the topology.

set god_ [create-god $val(mobile_nodes_count)] : GOD(general operation director) will keep that information of each node in because two nodes can transmit the data if they are in the range of one another. Hence when some node is not in range of another node then it will be difficult to perform the transmission. Hence this GOD object will be used to keep the global information so that routing can be done. Also, only one GOD object is created for a particular simulation. This code is creating god object for given number of nodes.

```
#Configuration of nodes using the intitially defined variables
        $ns node-config -adhocRouting $val(type_of_routing_protocol) \
       -llType $val(ll) \
       -macType $val(type_of_mac) \
       -ifqType $val(queuing_model) \
       -ifqLen $val(ifqlen) \
       -antType $val(anteena_type_used) \
       -propType $val(propagation_model) \
       -phyType $val(type_of_interface) \
       -topoInstance $topo \
       -channel $channel_to_use \
       -agentTrace ON \
       -routerTrace ON \
       -macTrace OFF \
       -movementTrace ON
```

All the variables that we have defined at the start of this file are being used here to set the configuration parameters.

```
# here topology contains the value of node position for each node in format of x_pos(node_num) and y_pos(node_num)
source ./topology.tcl

for {set i 0} {$i < $val(mobile_nodes_count) } { incr i } {
# this $x_pos($i) is stored in topology tcl file
$node_($i) set X_ $x_pos($i)
$node_($i) set Y_ $y_pos($i)

}
```

Now the file ***topology.tcl*** is included in this code to provide the initial topology information

Move to this directory in home folder

cd /home/amar/ns-allinone-2.35/ns-2.35/indep-utils/cmu-scen-gen/setdest

General format of executin the setdest is

./setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime] [-x maxx] [-y maxy] > [outdir/movement-file]

example file : this will save data to mobility.tcl file which will be sourced in start_program.tcl file

$ ./setdest -v 1 -n 50 -p 10 –s 20 –t 50 -x 500 -y 500 > mobility.tcl

Here pause time is set to 10 m/s

```
source ./mobility.tcl

set source 5

set destination 37

puts "-----------------------------------------------------------------------"

puts "\n The default source node is $source"

puts "\n The default destination node is $destination \n"
puts "-----------------------------------------------------------------------"

$ns at 1.0 "$node_($source) color #FF1493"

$ns at 1.0 "$node_($destination) color #FF1493"

$ns at 1.0 "$node_($source) label Source"

$ns at 1.0 "$node_($destination) label Destination"

$ns at 1.0 "$ns trace-annotate \" Source node is $source\""

$ns at 1.0 "$ns trace-annotate \" Destination node is $destination\""
```

As we have generated the mobility file in the first case, so here this files is sourced using

source ./mobility.tcl

set source 5 this will set the source node to 5

set destination 37 this will set the destination node to 37

$ns at 1.0 "$ns trace-annotate \" Source node is $source\""

Trace-annotate is used to show the comments at bottom of nam animator.

```
#*******Data Transmission from Source to Destination*****

set tcp [new Agent/TCP/$val(tcp_variant_input)]

$ns attach-agent $node_($source) $tcp; # attaching source node as TCP reno/sack

set ftp [new Application/FTP]

$ftp set interval_ 1.0

$ftp attach-agent $tcp ; # attaching FTP and TCP

set sink [new Agent/TCPSink]

$ns attach-agent $node_($destination) $sink ; # attaching source node as TCPsink

$ns connect $tcp $sink

$ns at 1.0 "$ftp start"
```

```
# Ending nam and the simulation
$ns at $val(simulation_stop_time) "$ns nam-end-wireless $val(simulation_stop_time)"
$ns at $val(simulation_stop_time) "simulation_stop_time"
$ns at 50.01 "puts \"end simulation\" ; $ns halt"
```

set tcp [new Agent/TCP/$val(tcp_variant_input)] : here */$val(tcp_variant_input)]* is a variable used to select the particular TCP variant on source node

$ns attach-agent $node_($source) $tcp : here we are attaching source node as TCP Reno/sack etc.

$ftp attach-agent $tcp ;: here attaching FTP and TCP is done.

$ns attach-agent $node_($destination) $sink; here we are attaching source node as TCPsink

$ns at 1.0 "$ftp start" : FTP transmission is starting at application layer between nodes at 1 second

This command defines the node initial position in nam. <size> denotes the size of node in nam. This function must be called after mobility model has been defined. Here iteration is done on all the nodes.

here simulation is ended at $val(simulation_stop_time) by calling $ns nam-end-wireless procedure As mentioned before the simulation stop time is 50 s, hence at 50.01 simulator is outputting the ends simulation into the command line.

```
# simulation_stop_time Procedure...
proc simulation_stop_time {} {
  global ns trace_file namtrace_file val trace_file_path
  # we set the filenamez here as eg AODV_Reno.nam ..
  set filenamez nam_files/$val(type_of_routing_protocol)_$val(tcp_variant_input).nam

   #This command flushes the trace buffer and is typically called before the simulation run ends.
    $ns flush-trace
    close $trace_file
    close $namtrace_file
    set delay_file_to_save delay_files/$val(type_of_routing_protocol)_$val(tcp_variant_input).tr
    set pdr_to_save pdr_files/$val(type_of_routing_protocol)_$val(tcp_variant_input).tr

    exec awk -f delay_calculator.awk $trace_file_path > $delay_file_to_save &
    exec awk -f pdr_caculator.awk $trace_file_path > $pdr_to_save &

    puts "\n Everything is done Successfully... simulation is ending"
  exec nam $filenamez &
  exit 0
}

$ns run
```

global ns trace_file namtrace_file val trace_file_path : we have to define the trace and nam path as global in this function.

$ns flush-trace: This command flushes the trace buffer and is typically called before the simulation run ends.

close $trace_file and close $namtrace_file will close these respective files.

set delay_file_to_save delay_files/$val(type_of_routing_protocol)_$val(tcp_variant_input).tr : this will generate the delay trace files and put it to delay files for further calculation

set pdr_to_save pdr_files/$val(type_of_routing_protocol)_$val(tcp_variant_input).tr ; this will generate the PDR trace files and put it to delay files for further calculation.

exec awk -f delay_calculator.awk $trace_file_path > $delay_file_to_save : This will use delay_calculator.awk script on $trace_file_path and then save file as $delay_file_to_save .

exec awk -f pdr_caculator.awk $trace_file_path > $pdr_to_save : This will use pdr_calculator.awk script on $trace_file_path and then save file as $pdr_file_to_save .

exec nam $filenamez : this will start the nam file for visual analysis on trace file when the simulation is stopped

Delay_calculator.awk: for calculating end to end delay

```
BEGIN {
}
{
  # finding sending-time from respective trace file
  if($1 == "s" && $4 == "AGT") {
    sent_time = $2;
  }  #if ends

  else if($1 == "r" && $4 == "AGT") {
                  # finding receiving_time from respective trace file
                receive_time= $2;
                end_to_end_delay = receive_time - sent_time;
                print $2 " " end_to_end_delay;
  } #elseif ends
}
END {
  # other Calculation can be performed here
}
```

Here $1 means that it is comparing if column 1 is equal to s and column4 is AGT. If this is true, then it will set its sending time by from Column 4 of the trace file. Similarly, if the column 1 is equal to "r" and column4 is "AGT". If this is true, then it will set its *receiving time* by from Column 2 of the trace file.
Then delay is calculated by subtracting the receiving time and sending time.

```
BEGIN {
  rcvd_pkt = 0;
  gnrt_pkt = 0;
}
{
  # Pattern and action
  if($1 == "s" && $4 == "AGT") {
    gnrt_pkt++;
  }
  else if($1 == "r" && $4 == "AGT") {
    rcvd_pkt++;
        pdr=rcvd_pkt/gnrt_pkt*100;
        print $2 "  "pdr;
  }
}
END {
}
```

pdr_calculator.awk for calculating Packet Delivery Rate

Initially received packets and generated packets are initialized to zero. Then the script will check if ***column1*** is equal to **s** and ***column4*** is **AGT**. If this is true, then it will increment the generated packets by one. Else if the script will check if ***column1*** is equal to **r** and ***column4*** is **AGT**. If this is true, then it will increment the received packets by one
Finally, packet delivery Rate (PDR) is calculated by

$$packet\ delivery\ rate = \frac{received\ packets * 100}{Generated\ packets}$$

```
set list_of_protocol {AODV DSR DSDV}

foreach name_of_protocol $list_of_protocol {

  if { $name_of_protocol == "AODV" } {
    set yrange 0.08
    } elseif { $name_of_protocol == "DSR" } {
      set yrange 0.03

  } elseif {$name_of_protocol == "DSDV"} {
    set yrange 0.06
```

graph_generator_delay.tcl for calculating end-to-end delays

set list_of_protocol {AODV DSR DSDV} : this will generate an initial list for traversing all trace files one by one.
The for each loop will check if the name of protocol that is AODV then it will set *yrange* parameter for *gnuplot* as 0.08 . similarly if the name of protocol that is DSR then it will set *yrange* parameter for *Gnuplot* as 0.03. Finally, if the name of protocol that is AODV then it will set *yrange* parameter for *Gnuplot* as 0.06.

```
set current_file [open $name_of_protocol\_delay.p w+]
puts $current_file "set title \"Comparative analysis of end-to-end delay in $name_of_protocol Routing Protocol and TCP variants combinations\"
set xlabel \"Time(Seconds)\"
set ylabel \"end-to-end delay(s)\"
set xrange \[0:50\]
set yrange \[0:$yrange\]
set grid
set style line 1 lt 2 lw 2 pt 3 ps 0.5
"
close $current_file
```

set current_file [open $name_of_protocol_delay.p w+] : this will open the file named as **$name_of_protocol_delay.p** in write mode , whose value is in current_file variable. The \ is used to prevent _ read as a part of variable $name_of_protocol.

set xlabel \"Time(Seconds)\" : setting the xlabel for graph in gnuplot

set ylabel \"end-to-end delay(s)\" : setting the ylabel for graph in gnuplot

set xrange \[0:50\] : setting the xrange for graph in gnuplot as 0 to 50

set yrange \[0:$yrange\] : setting the xrange for the graph in gnuplot as 0 to range that has been defined previously.

set grid : set the grid for gnuplot

set style line 1 lt 2 lw 2 pt 3 ps 0.5 : change the line parameters.

close $current_file : this will close the file.

```
set current_file_to_append [open $name_of_protocol\_delay.p a+]
# puts  list_of_files_AODV
 set i 0
 foreach file [glob $name_of_protocol\_*.tr] {
  global $i $current_file_to_append
  set title $file

 set title [ string  map {\_ " " } $title ]
 set title [ string  map {\.tr "" } $title ]
```

set current_file [open $name_of_protocol_delay.p a+] : this will open the file named as **$name_of_protocol_delay.p** in append mode.

glob $name_of_protocol_*.tr : here glob is used to collect all the file from the current directory with selected protocol name ending with **.tr**

```
if {$i == 0} {
global $current_file

 puts $current_file_to_append "plot \"$file\" every 70 with lines title \"$title\""

} else  {
   puts $current_file_to_append "replot \"$file\" every 70 with lines title \"$title\""
}
incr i
```

puts $current_file_to_append "plot \"$file\" every 70 with lines title \"$title\"" : this will put the name of current file in the generated **.p** file and will add the plot line.

We have used every 70 lines to skip the 70 lines in the trace file and then generate the graph from that.

puts $current_file_to_append "replot \"$file\" every 70 with lines title \"$title : this is basically replot line that is added to .p file . actually, first line is plot and then all the commands that are after plot file are replot commands.

AODV_pdr.p for generating packet delivered ratio graphs using gnuplot

```
set title "Comparative analysis of Packets delivered ratio in AODV Routing Protocol and TCP variants combinations"
set xlabel "Time(Seconds)"
set ylabel "Packets delivered ratio"
set xrange [0:50]
set yrange [0:100]
set grid
set style line 1 lt 2 lw 2 pt 3 ps 0.5

plot "AODV_Vegas.tr" every 70 with lines title "AODV Vegas"
replot "AODV_Linux.tr" every 70 with lines title "AODV Linux"
replot "AODV_Reno.tr" every 70 with lines title "AODV Reno"
replot "AODV_Newreno.tr" every 70 with lines title "AODV Newreno"
replot "AODV_Sack1.tr" every 70 with lines title "AODV Sack1"
replot "AODV_Fack.tr" every 70 with lines title "AODV Fack"
```

This is an output file that is generated by **graph_generator_pdr.tcl** when **$name_of_protocol** is **AODV.**

DSDV_pdr.p for generating packet delivered ratio graphs using gnuplot

```
set title "Comparative analysis of Packets delivered percentage in DSDV Routing Protocol and TCP variants combinations"
set xlabel "Time(Seconds)"
set ylabel "Packets delivered ratio"
set xrange [0:50]
set yrange [0:100]
set grid
set style line 1 lt 2 lw 2 pt 3 ps 0.5

plot "DSDV_Sack1.tr" every 70 with lines title "DSDV Sack1"
replot "DSDV_Linux.tr" every 70 with lines title "DSDV Linux"
replot "DSDV_Vegas.tr" every 70 with lines title "DSDV Vegas"
replot "DSDV_Reno.tr" every 70 with lines title "DSDV Reno"
replot "DSDV_Newreno.tr" every 70 with lines title "DSDV Newreno"
replot "DSDV_Fack.tr" every 70 with lines title "DSDV Fack"
```

This is an output file that is generated by **graph_generator_pdr.tcl** when **$name_of_protocol** is **DSDV.**

DSR_pdr.p for generating packet delivered ratio graphs using gnuplot

```
set title "Comparative analysis of Packets delivered percentage in DSR Routing Protocol and TCP variants combinations"
set xlabel "Time(Seconds)"
set ylabel "Packets delivered ratio"
set xrange [0:50]
set yrange [0:100]
set grid
set style line 1 lt 2 lw 2 pt 3 ps 0.5

plot "DSR_Newreno.tr" every 70 with lines title "DSR Newreno"
replot "DSR_Reno.tr" every 70 with lines title "DSR Reno"
replot "DSR_Fack.tr" every 70 with lines title "DSR Fack"
replot "DSR_Sack1.tr" every 70 with lines title "DSR Sack1"
```

This is an output file that is generated by **graph_generator_pdr.tcl** when **$name_of_protocol** is **DSR.**

AODV_delay.p for generating end-to-end graphs using gnuplot

```
set title "Comparative analysis of end-to-end delay in AODV Routing Protocol and TCP variants combinations"
set xlabel "Time(Seconds)"
set ylabel "end-to-end delay(s)"
set xrange [0:50]
set yrange [0:0.08]
set grid
set style line 1 lt 2 lw 2 pt 3 ps 0.5

plot "AODV_Vegas.tr" every 70 with lines title "AODV Vegas"
replot "AODV_Linux.tr" every 70 with lines title "AODV Linux"
replot "AODV_Reno.tr" every 70 with lines title "AODV Reno"
replot "AODV_Newreno.tr" every 70 with lines title "AODV Newreno"
replot "AODV_Sack1.tr" every 70 with lines title "AODV Sack1"
replot "AODV_Fack.tr" every 70 with lines title "AODV Fack"
```

This is an output file that is generated by **graph_generator_delay.tcl** when **$name_of_protocol** is **AODV.**

DSDV_delay.p for generating end-to-end graphs using gnuplot

```
set title "Comparative analysis of end-to-end delay in DSDV Routing Protocol and TCP variants combinations"
set xlabel "Time(Seconds)"
set ylabel "end-to-end delay(s)"
set xrange [0:50]
set yrange [0:0.06]
set grid
set style line 1 lt 2 lw 2 pt 3 ps 0.5

plot "DSDV_Sack1.tr" every 70 with lines title "DSDV Sack1"
replot "DSDV_Linux.tr" every 70 with lines title "DSDV Linux"
replot "DSDV_Vegas.tr" every 70 with lines title "DSDV Vegas"
replot "DSDV_Reno.tr" every 70 with lines title "DSDV Reno"
replot "DSDV_Newreno.tr" every 70 with lines title "DSDV Newreno"
replot "DSDV_Fack.tr" every 70 with lines title "DSDV Fack"
```

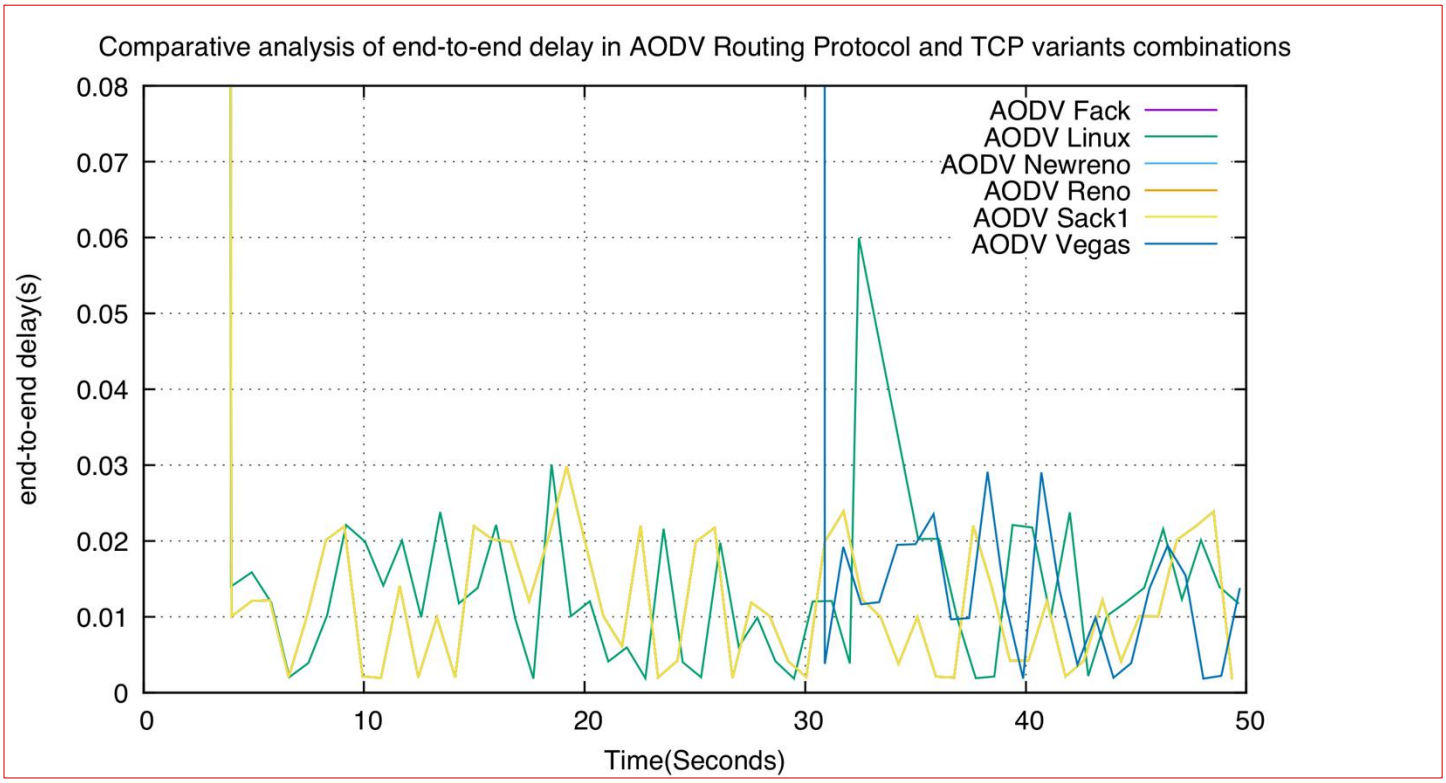
This is an output file that is generated by **graph_generator_delay.tcl** when **$name_of_protocol** is **DSDV.**

DSR_delay.p for generating end-to-end graphs using gnuplot

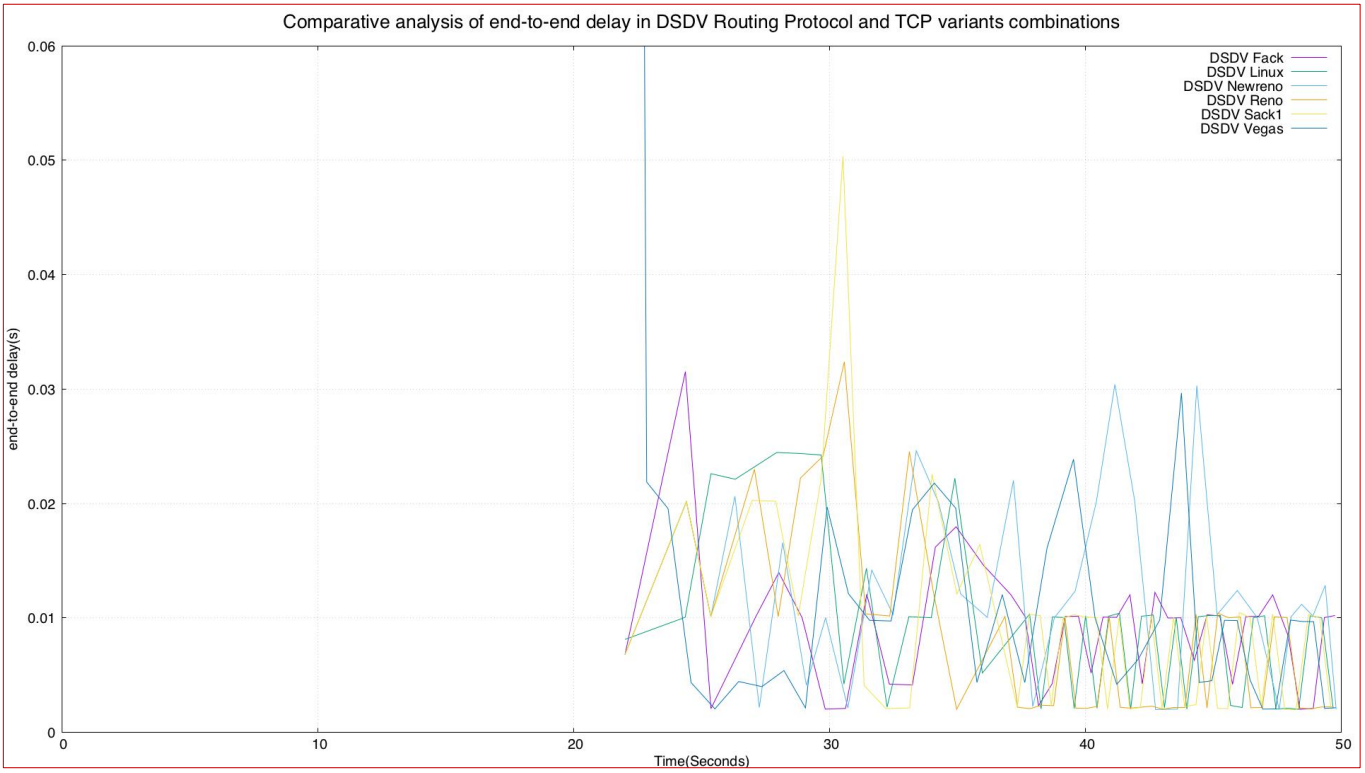
```
set title "Comparative analysis of end-to-end delay in DSR Routing Protocol and TCP variants combinations"
set xlabel "Time(Seconds)"
set ylabel "end-to-end delay(s)"
set xrange [0:50]
set yrange [0:0.03]
set grid
set style line 1 lt 2 lw 2 pt 3 ps 0.5

plot "DSR_Newreno.tr" every 70 with lines title "DSR Newreno"
replot "DSR_Reno.tr" every 70 with lines title "DSR Reno"
replot "DSR_Fack.tr" every 70 with lines title "DSR Fack"
replot "DSR_Sack1.tr" every 70 with lines title "DSR Sack1"
```

This is an output file that is generated by **graph_generator_delay.tcl** when **$name_of_protocol** is **DSR.**

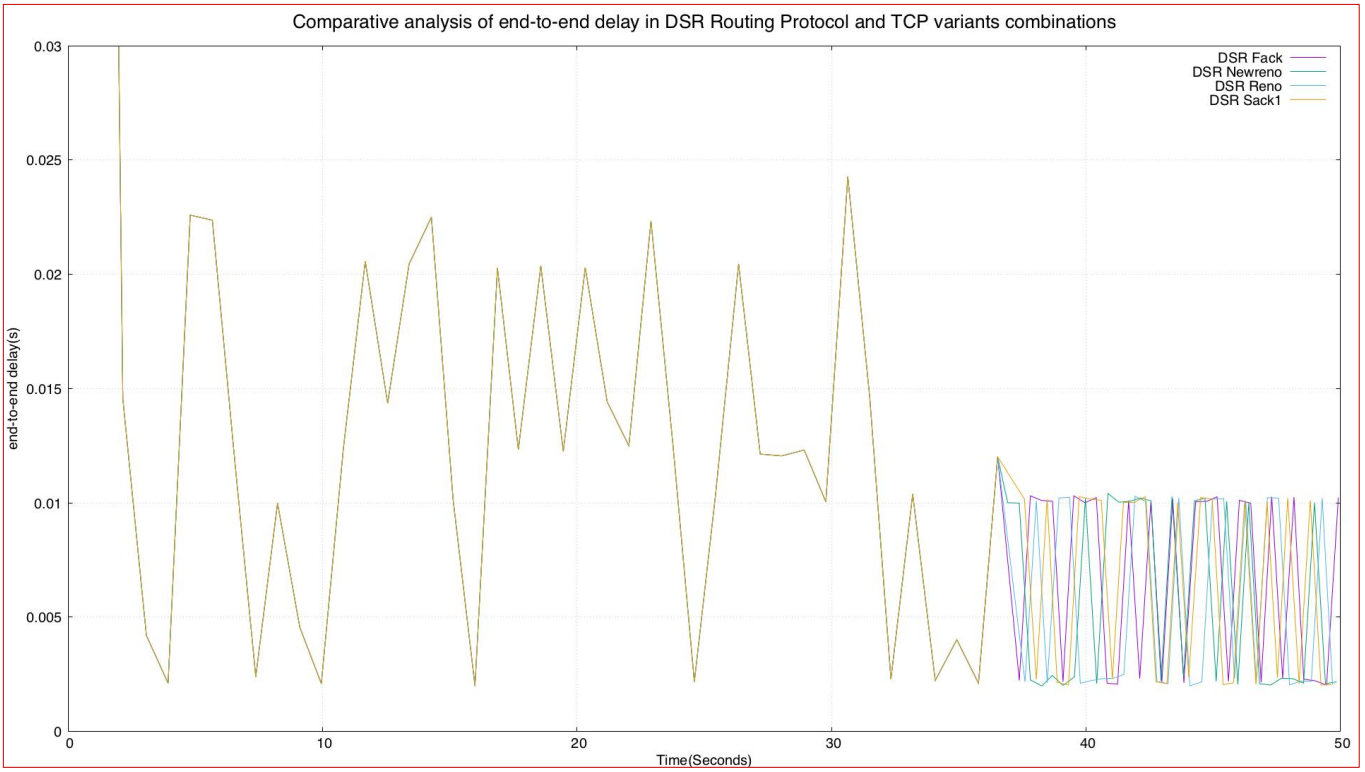
Results



AODV was performing worst in case of **TCP-linux** and best in the case of **TCP-vegas** as end-to-end delay started when time >30 seconds. Hence, we will select TCP-vegas as best combination with AODV routing protocol



When end-to-end delay was calculated in case of DSDV, we have noticed that performance order is Linux > Fack > Sack 1



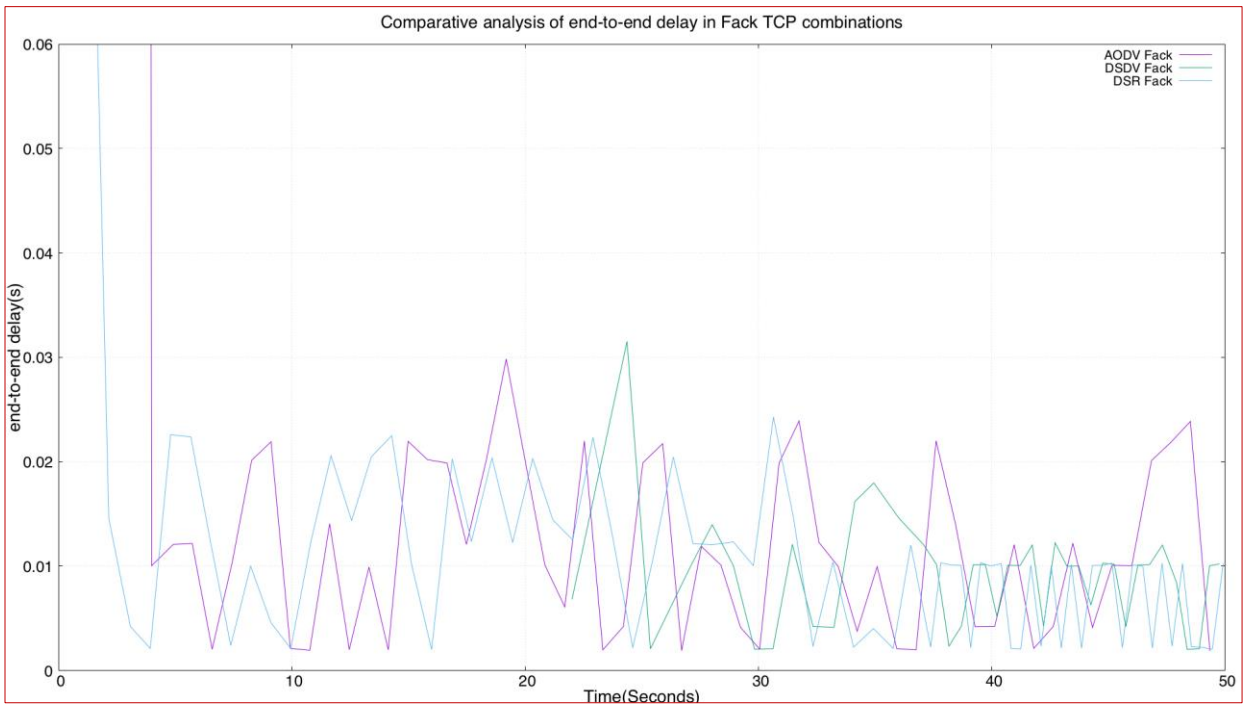
When end-to-end delay was calculated in case of DSDV, we have noticed that Sack-1 resulted in poor performance while others led to almost similar performance.

	Least end –to-end delay	**highest end –to-end delay**
AODV	Vegas	Sack1
DSR	Linux	Sack 1
DSDV	NewReno	Sack1

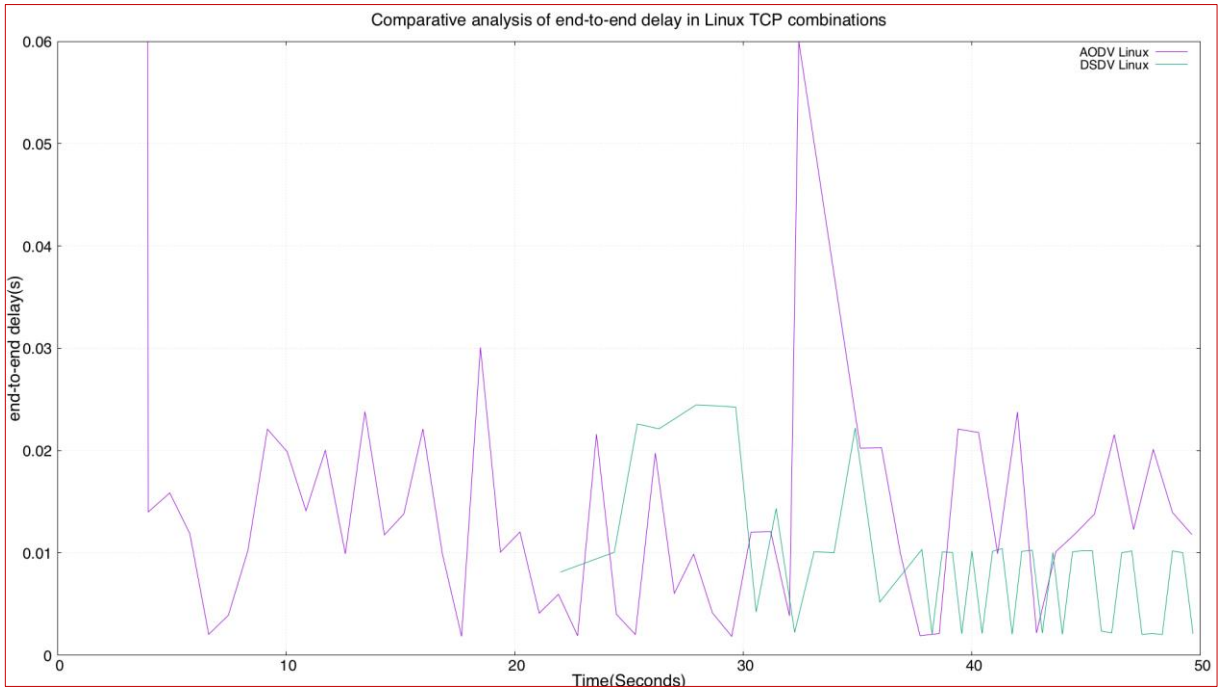
This table shows that worst performance was seen in the case of Sack1, while lease values were seen in different cases

.

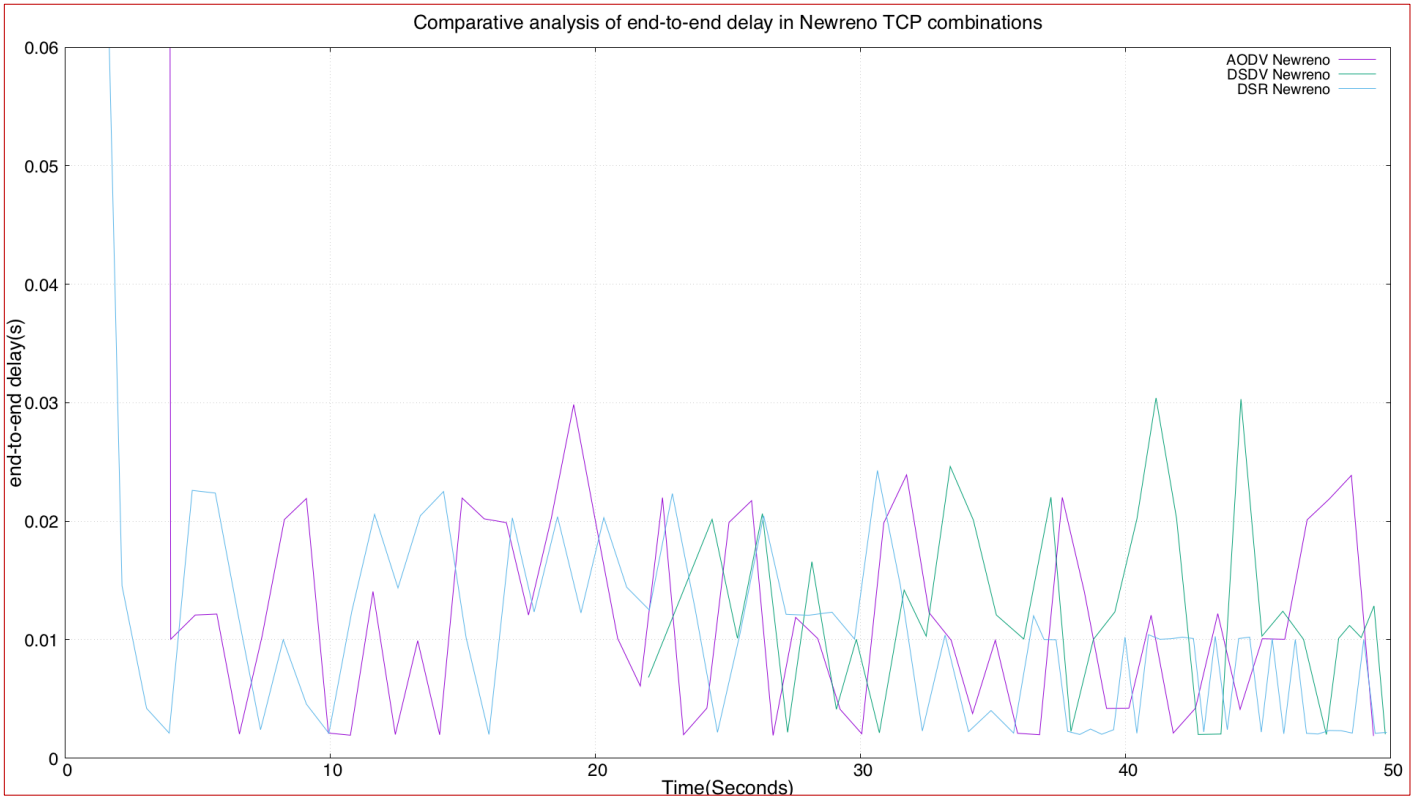
Comparison of least end-to-end delay when different routing protocols were used with same combination of TCP variant



TCP- Fack was performing worst in case of DSR and best in case of DSDV

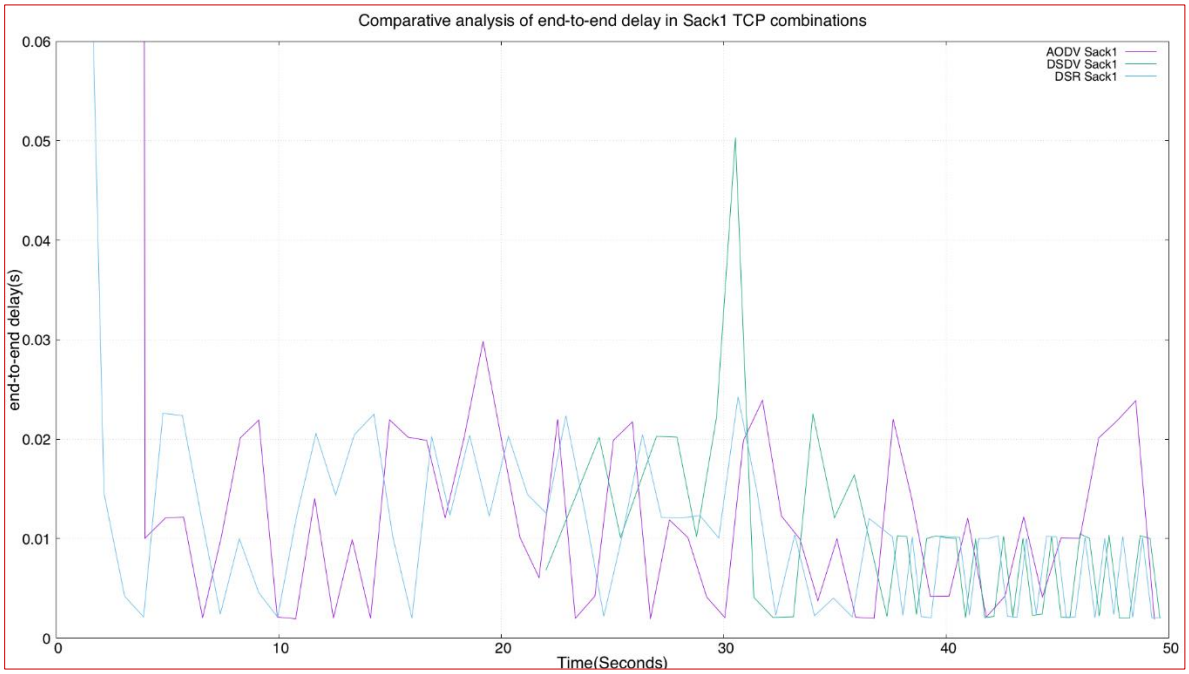


TCP- Linux was performing worst in case of AODV and best in the case of DSDV.

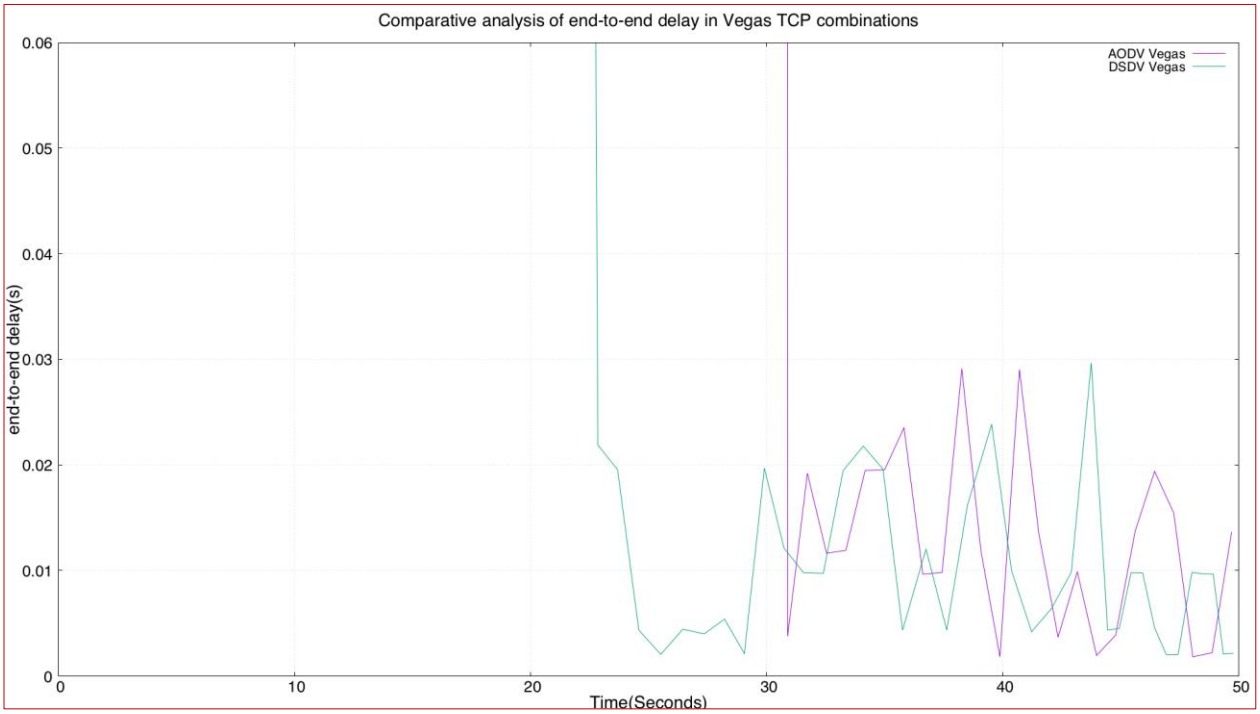


TCP- NewReno was performing worst in case of AODV and best in the case of DSDV.

TCP- Reno was performing worst in case of DSR and best in the case of DSDV.



TCP- Sack1 was performing worst in case of AODV and best in the case of DSDV.

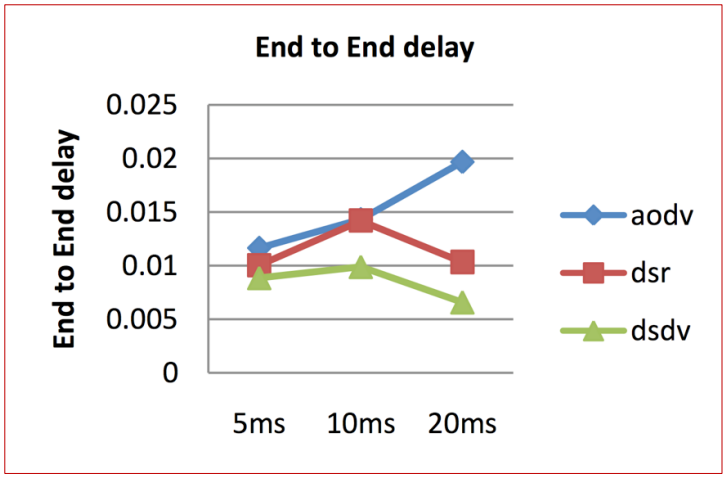


TCP- Vegas was performing worst in case of AODV and best in the case of DSDV.

For end-to-end delay	Worst performance	Best performance
TCP- Fack	DSDV	DSR
TCP- Linux	DSDV	AODV
TCP- NewReno	DSDV	AODV
TCP- Reno	DSDV	DSR
TCP- Sack1	DSDV	AODV
TCP- Vegas	DSDV	AODV

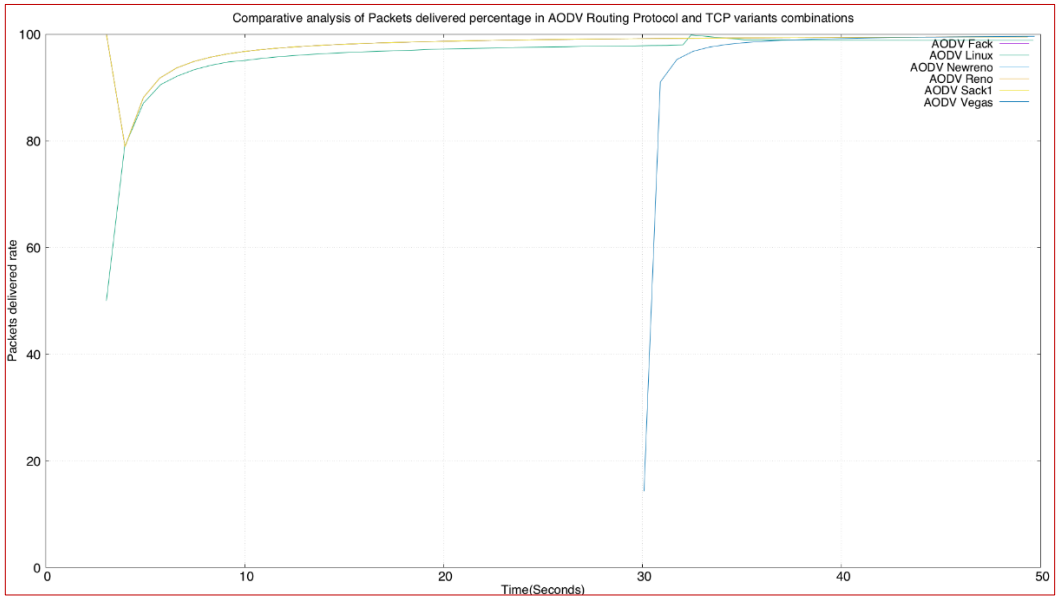
Hence it is clear that least value of end-to-end delay was obtained in DSDV for all cases.

[1] Rajeshkumar, V., & Sivakumar, P. (2013). Comparative study of AODV, DSDV and DSR routing protocols in MANET using network simulator-2. *International Journal of Advanced Research in Computer and Communication Engineering*, *2*(12).



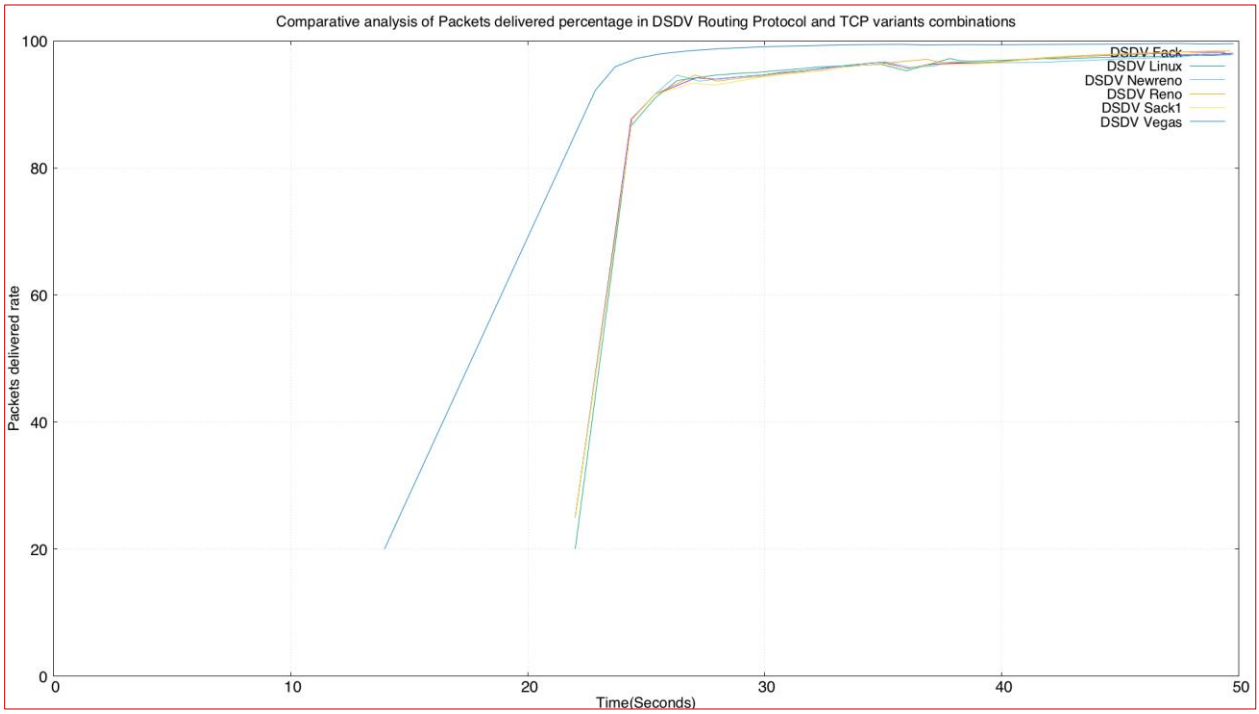
This Figure has been taken from [1], it shows that the value so end-to-end delay is least in DSDV when speed of nodes is varied, hence these results are matching with our results

Graphs for Packet Delivery rate

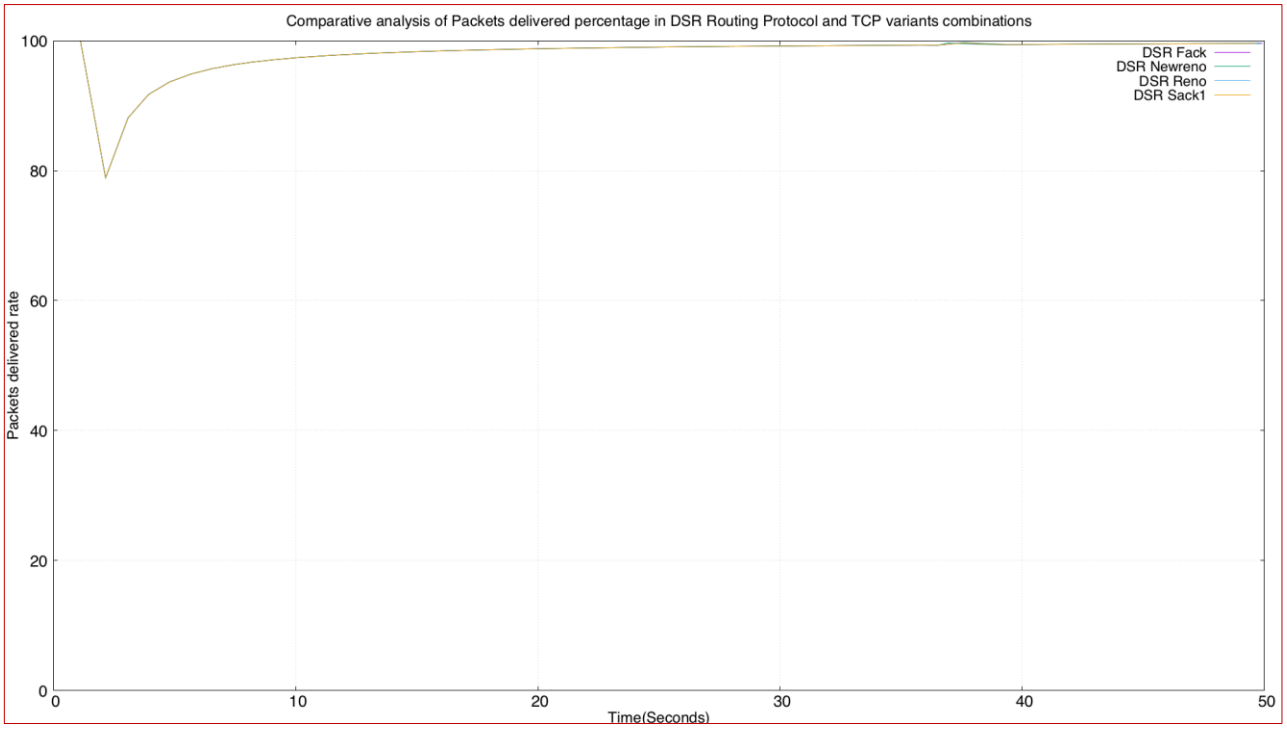


The value of packet delivered rate is $\frac{received\ packets*100}{genrated\ packets}$ hence more value means that more packets are delivered.

AODV was performing best in the case of TCP-Sack1 and worst in case of TCP-Vegas.

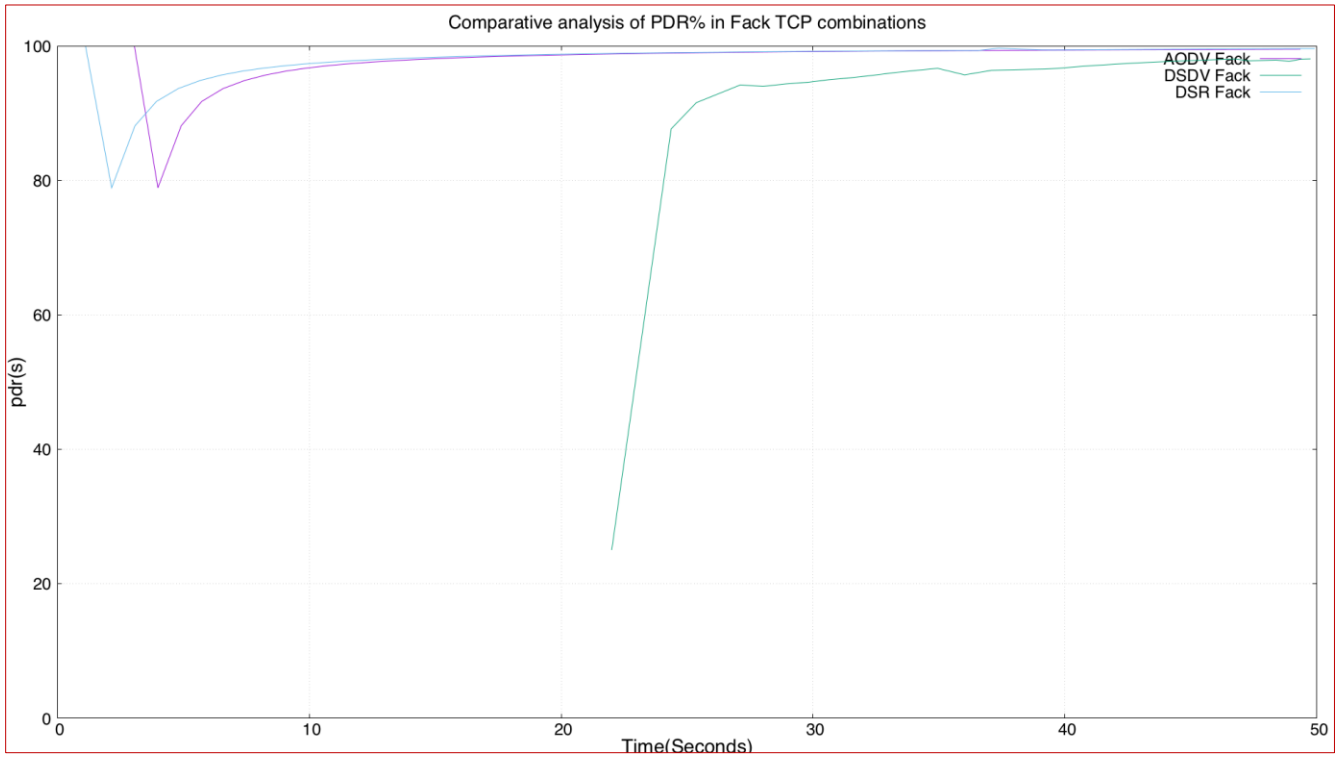


DSDV was performing best in the case of TCP-Vegas and worst in case of TCP-Linux.

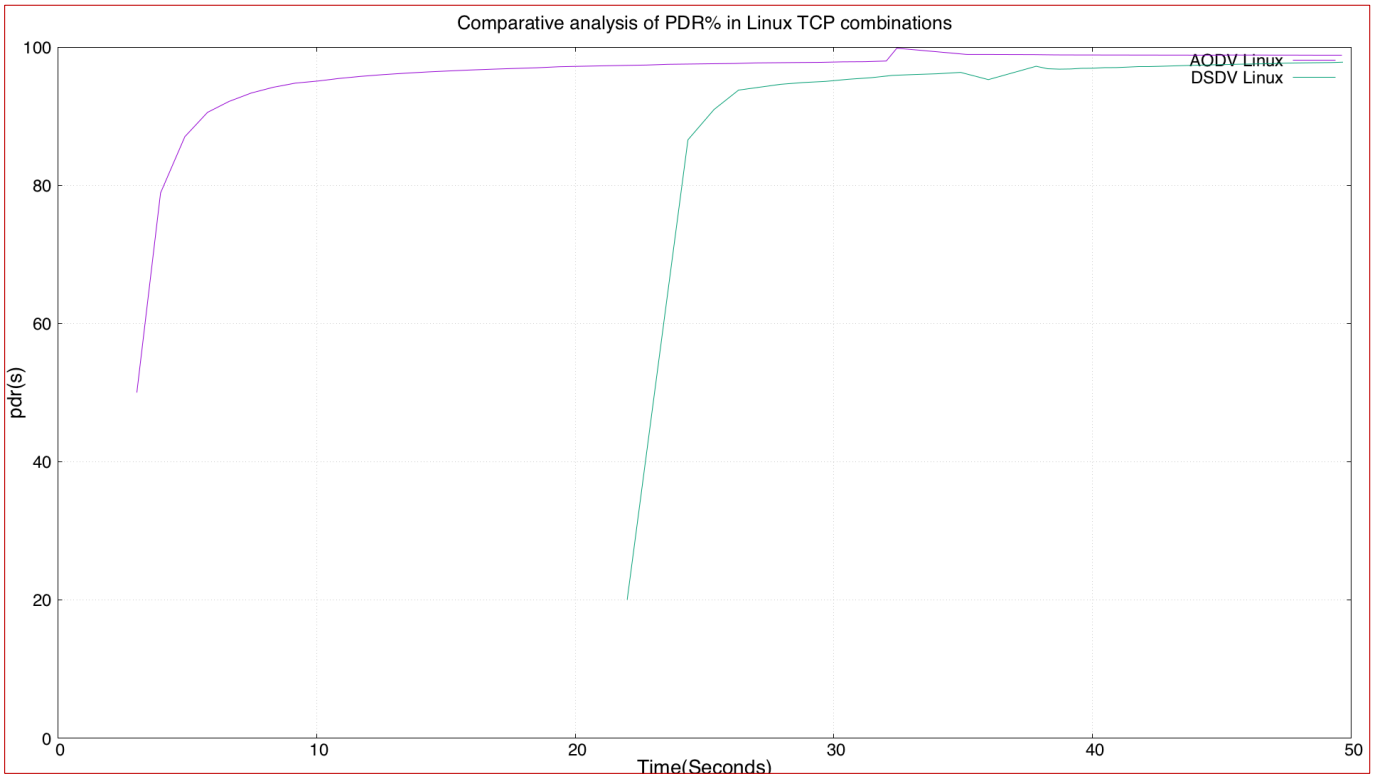


DSR was working only in case of Sack1

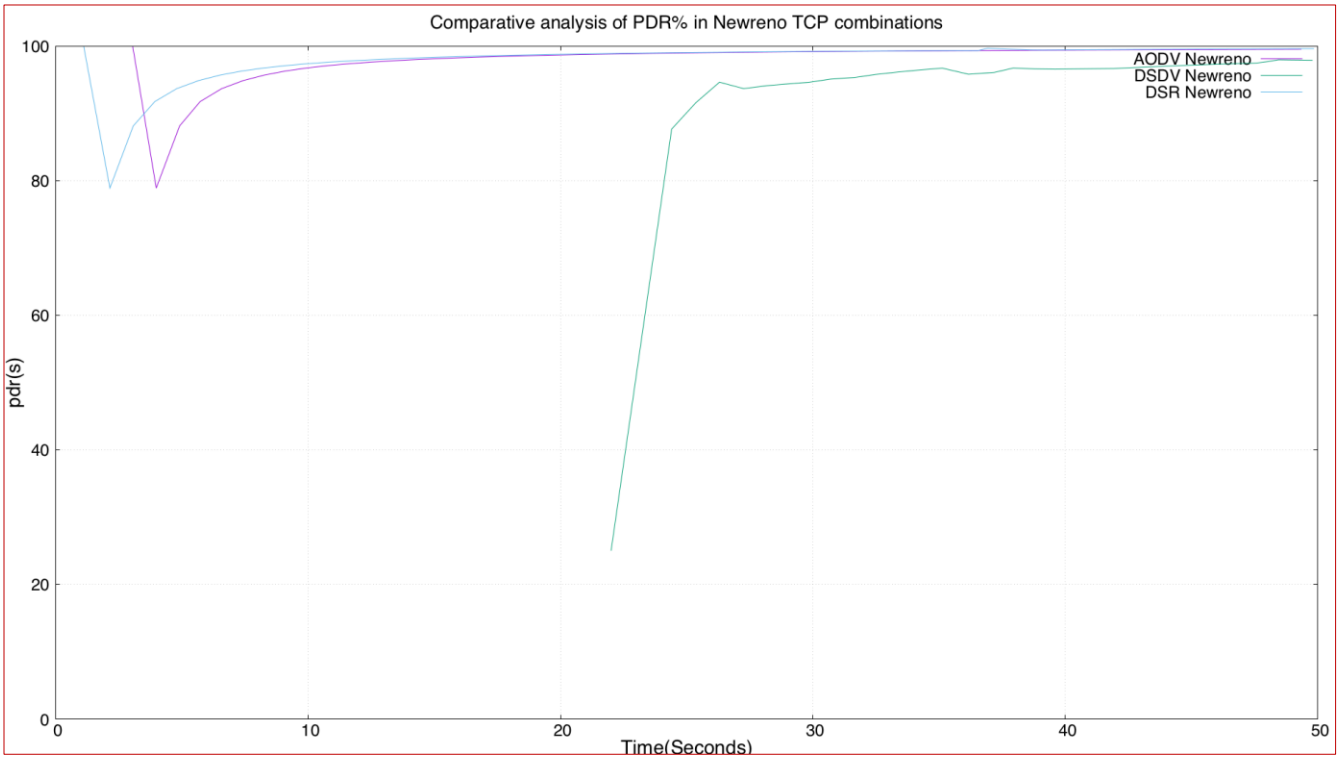
	Best performance	**Worst performance**
AODV	Sack1	Vegas
DSR	Sack1	NA
DSDV	Vegas	Linux



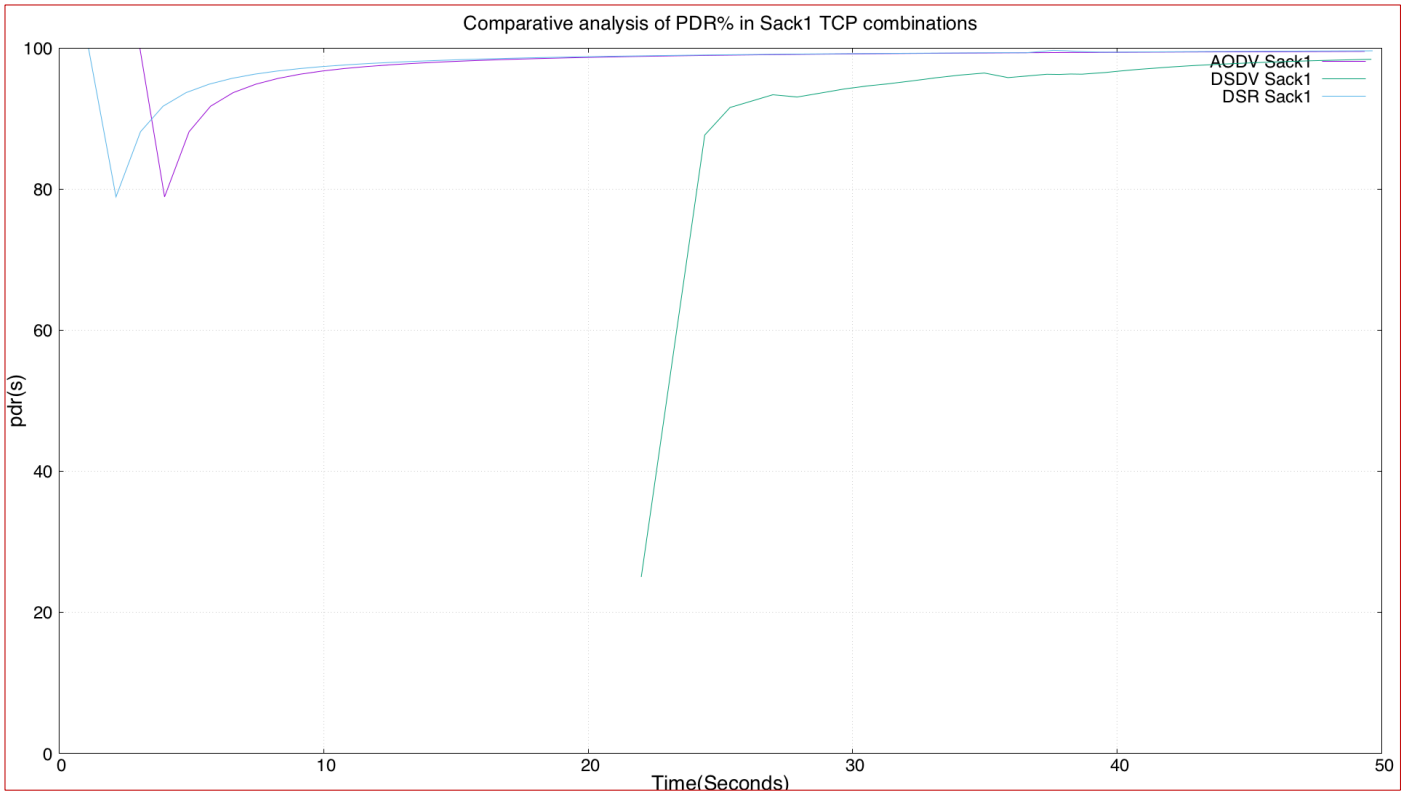
TCP- Fack was performing worst in case of DSDV and best in case of DSR



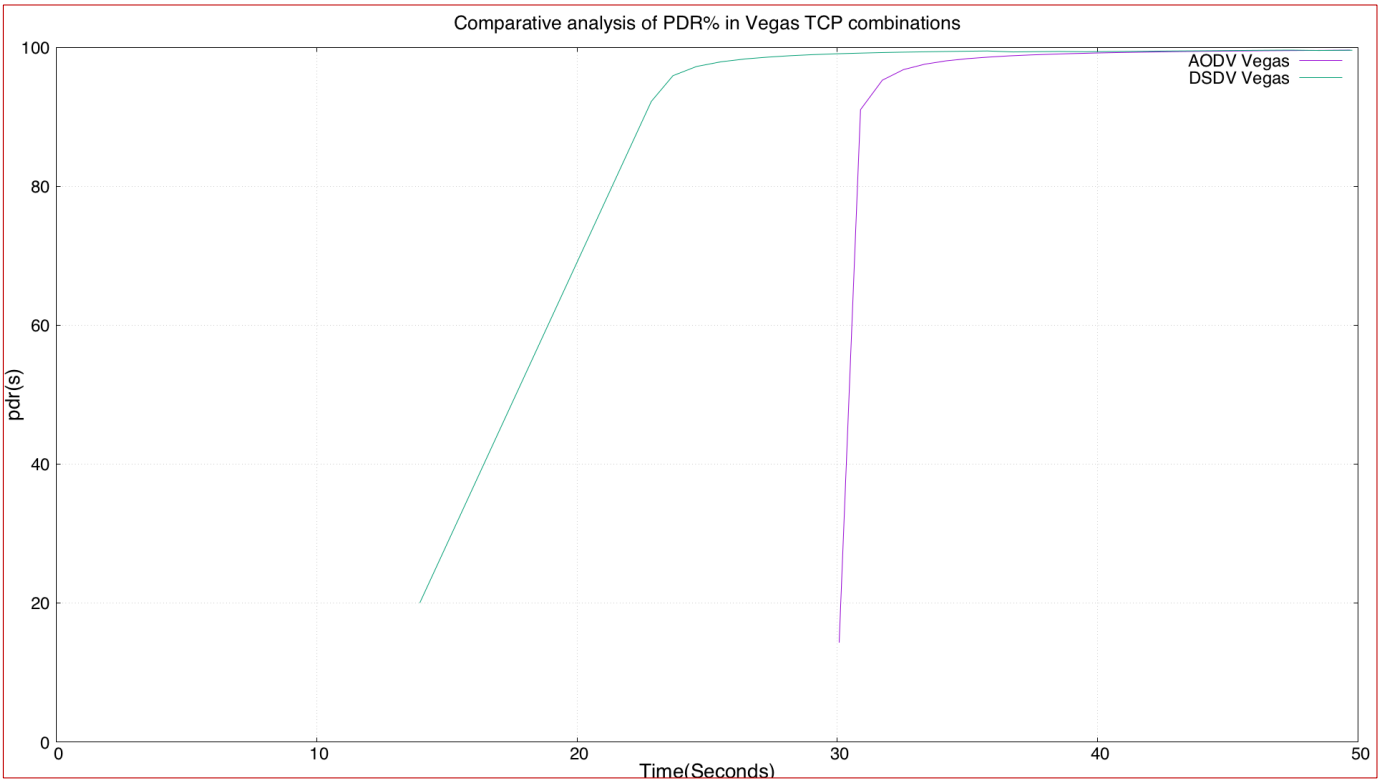
TCP- Linux was performing worst in case of DSDV and best in case of AODV



TCP- NewReno was performing worst in case of DSDV and best in the case of DSR.



TCP- NewSack1 was performing worst in case of DSDV and best in the case of DSR.



TCP- Vegas was performing worst in case of DSDV and best in the case of AODV.

For PDR	worst performance	best performance
TCP- Fack	DSDV	DSR
TCP- Linux	DSDV	AODV
TCP- NewReno	DSDV	DSR
TCP-Reno	DSDV	RENO
TCP- Sack1	DSDV	DSR
TCP- Vegas	DSDV	AODV

Analysis of our results

For end-to-end delay	Worst performance	Best performance
TCP- Fack	DSDV	DSR
TCP- Linux	DSDV	AODV
TCP- NewReno	DSDV	AODV
TCP- Reno	DSDV	DSR
TCP- Sack1	DSDV	AODV
TCP- Vegas	DSDV	AODV

For PDR	worst performance	best performance
TCP- Fack	DSDV	DSR
TCP- Linux	DSDV	AODV
TCP- NewReno	DSDV	DSR
TCP-Reno	DSDV	RENO
TCP- Sack1	DSDV	DSR
TCP- Vegas	DSDV	AODV

Problem faced during ns2 installation along with their Solutions:

1. GnuPlot

1.a) Plotting multiple Values: Most researchers use *Xgraph* to plot the graphs due to existing code availability. We installed *Xgraph* and found it to be outdated as compared to *GnuPlot* which has more mathematical functions and output format support like pngcairo,wxt enhanced

The major problem faced with *GnuPlot* while generating the graphs was that, it plotted only single line. Please note that, we have given multiple input but they are not displayed in the graph. The reason for this error may be the incorrect path for the other input files. Here is the error that it outputs only one line.

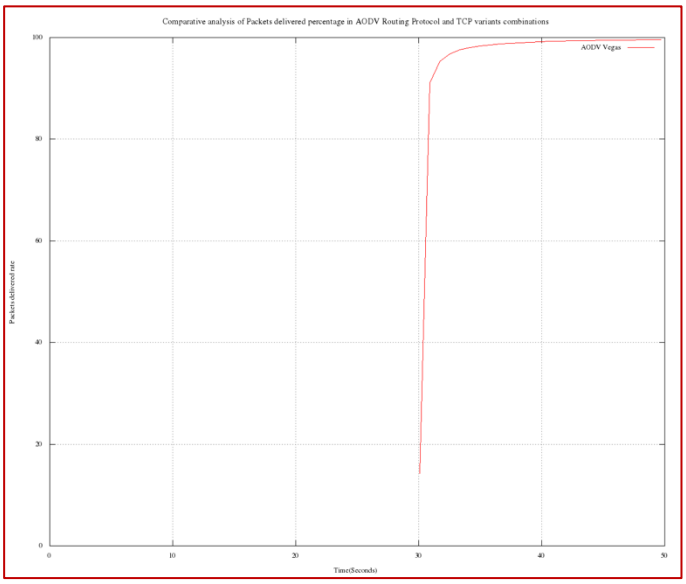


Figure 1 Problem as only single line is plotted

Solution # 1.a :

```
plot "AODV_Newreno.tr" every 70 with lines title "AODV Newreno"
replot "DSDV_Newreno.tr" every 70 with lines title "DSDV Newreno"
replot "DSR_Newreno.tr" every 70 with lines title "DSR Newreno"
set term pngcairo size 1920,1080
set output "tcp/graphs/Newreno_delay.png"
replot
```

Here, first we check the file paths and then we set *term* which is short form of terminal, to **pngcairo** (it is the output format more clear than typical png). We have given the plot commands *prior* to saving the output file **"Newreno_delay.png".** Some people found solution while setting the terminal and output values prior to plotting.

1.b) Plotting after time interval in GnuPlot: The simplified data from trace file has data format like below; it is having two columns, The problem is that we have multiple values of x axis and if we generate it's graph , then it will be not consistent , as the values of y axis are changing rapidly. Hence as X-axis values are order of ~3000 per file, so if we generate it's graph it will look very uneven, so that it is difficult to analyze it.

```
40.002953345  0.0101857
40.013099617  0.0101463
40.015125319  0.0020257
40.017291588  0.00216627
40.019377858  0.00208627
```

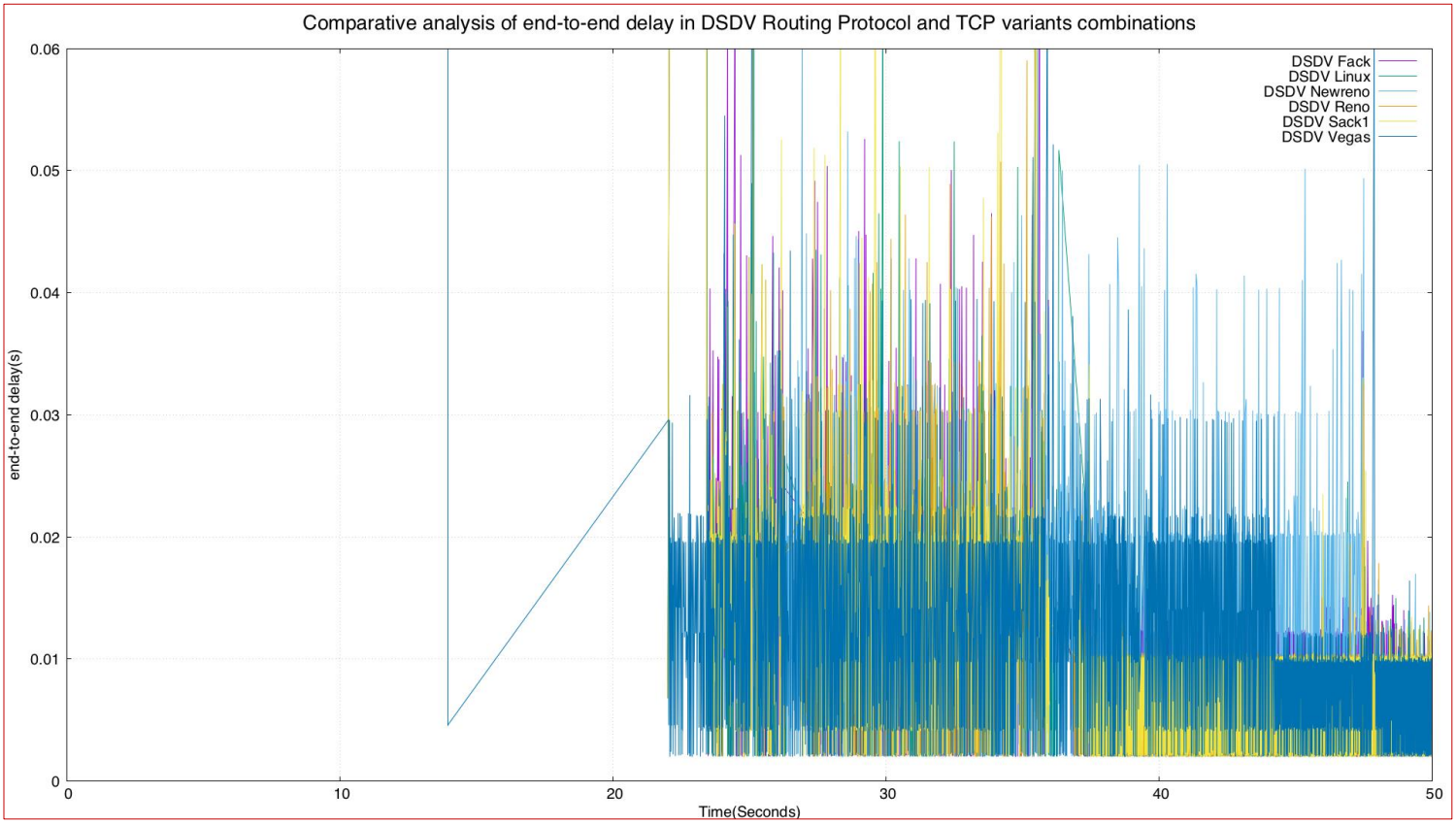


Figure 2 Result showing end to end delay , when every 70 was not used

Most of the research papers have plotted value at 10, 20, 30, 40, 50 seconds. They have rounded left hand side for the first value of 40th seconds. For example, the value of 40th second is 0.0101857

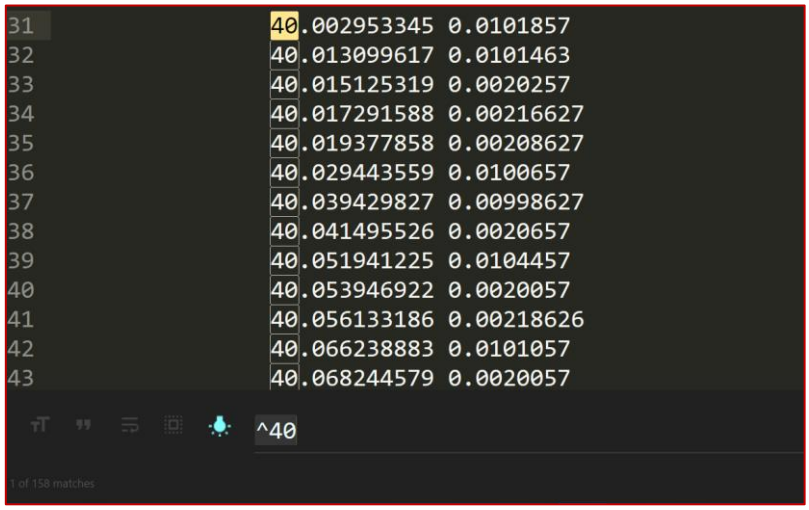
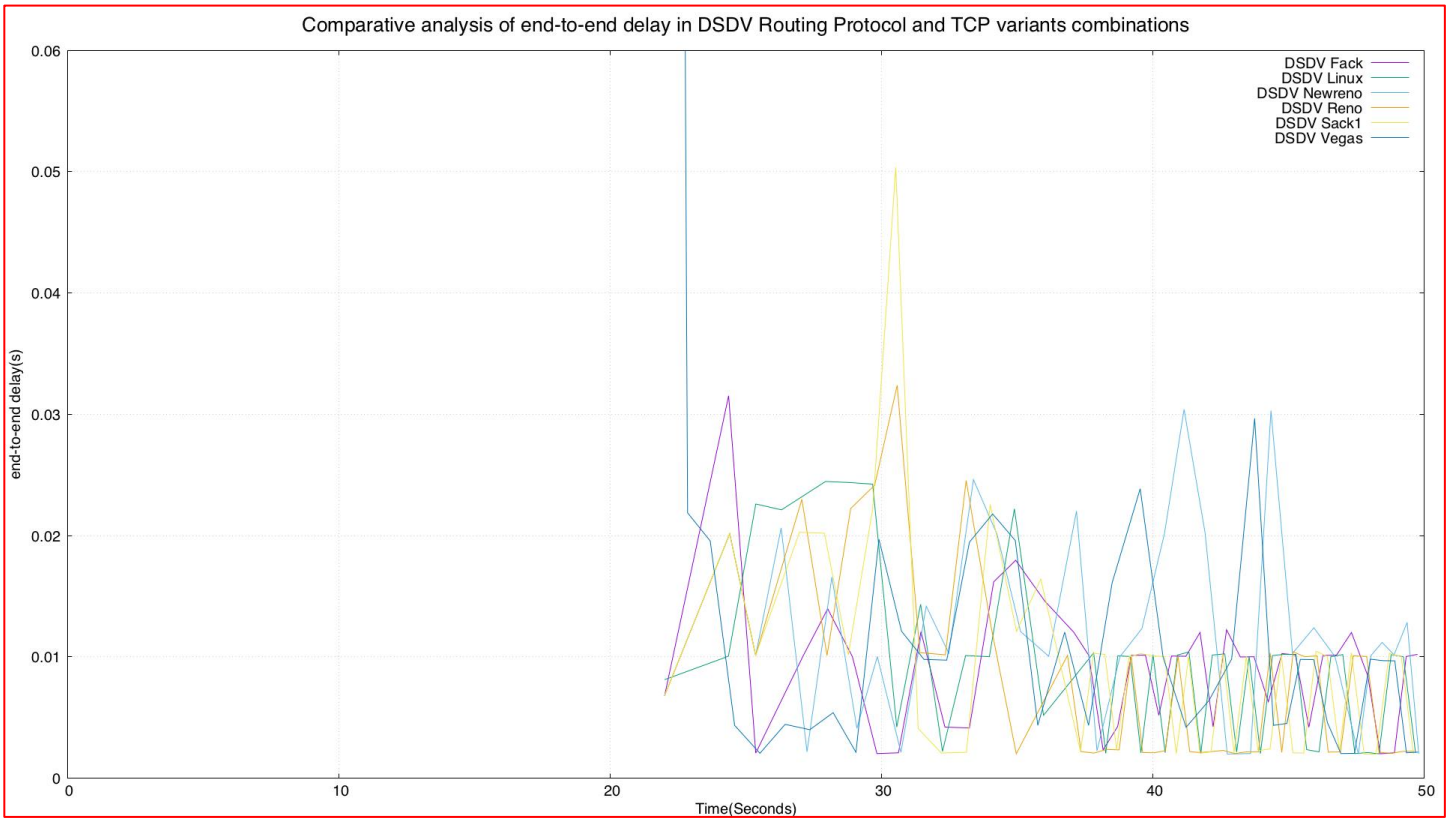


Figure 3 Example of Regex. in Sublime text-3 for getting estimate of number of values

Solution: Here, we need to take interval automatically. We calculated number of value for a single second. For example, 46 second around 150 values. We plot value after 70 intervals. So now we will show the graph when ***every 70*** was used



2.a) Segmentation Fault in NS2: We faced segmentation fault when we tried to Dsr with TCP/vegas and TCP/linux. There are patches updated for these problems. After installing the patches, user needs to recompile the ns2 directory a lot of time. Due to time constraints, we haven't patched NS 2.35

```
macTrace: OFF

movementTrace: ON

**********************************************

INITIALIZE THE LIST xListHead
--------------------------------------------------------------------------------

 The default source node is 5

 The default destination node is 37

--------------------------------------------------------------------------------
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
Segmentation fault (core dumped)
bigmike@bigmikev:~/Dropbox/elg$
```

2. Problems related with installation of Network Simulator 2

We followed the guide at which is very detailed.

https://www.howtoforge.com/tutorial/ns2-network-simulator-on-ubuntu-14.04/

This can also be accessed at
https://archive.fo/IycBu

The main is step # 5

It is important to have correct environment variable in **bash.rc** as it took a while to get them right.

```
# LD_LIBRARY_PATH

OTCL_LIB=/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/otcl-1.14/
NS2_LIB=/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/lib/
USR_Local_LIB=/usr/local/lib/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$USR_Local_LIB

# TCL_LIBRARY

TCL_LIB=/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/tcl8.5.10/library/
USR_LIB=/usr/lib/
export TCL_LIBRARY=$TCL_LIBRARY:$TCL_LIB:$USR_LIB

# PATH

XGRAPH=/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/xgraph-12.2/:/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/bin/:/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/tcl8.5.10/unix/:/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/tk8.5.10/unix/
NS=/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/ns-2.35/
NAM=/home/YOUR_USERNAME/Downloads/ns-allinone-2.35/nam-1.15/
export PATH=$PATH:$XGRAPH:$NS:$NAM
```

3) Problem with Vmware VirtualBox

3.a) Shared folder Problem

When we tried to access the shared folder in Ubuntu, it was not shared between Ubuntu and Vmware.

Firstly, install *VBoxGuestAdditions* in Ubuntu.

Then, the solution was to add current user to the usergroup **vboxsf** group which has the correct permission for accessing the files.

sudo usermod -aG vboxsf **your_username**

Then restart the virtualbox.

4. Mac

4.a) NS2 installation on MAC:

This installation guide is very detailed here**.** Other error depended on missing dependencies which can different on different machines.

4.b) Xcode Outdated error:

The install Xcode version was 7.2 and compiling needed latest version.

```
Mac:~ Amar$ brew install gnuplot --with-xll
Error: Your Xcode (7.3.1) is too outdated.
Please update to Xcode 8.2 (or delete it).
Xcode can be updated from the App Store
```

but, when we checked if Xcode needs to be updated, we get following

```
Mac:~ Amar$ softwareupdate --install xcode
Software Update Tool
Copyright 2002-2015 Apple Inc.

xcode: No such update
No updates are available.
```

Solution : The solution is to uninstall old version and install the Latest version of Xcode from the developers account.

5. Problem in graphing results

5.a) Problem of same value of TCP/Fack and TCP/Sack1

TCP/Fack agent is the implementation of "forward ACK" TCP, a modification of *Sack* TCP so both gave almost same result for us and thus only one of the line is visible for the graphs.