Exploring Congestion Control Mechanism of TCP Variants over Wireless Networks

Report submitted to the SASTRA Deemed to be University

as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

K. SATHYA NAGA PAVAN

124156018, CSE AI&DS

December 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA -613401

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 OF THE UGC ACT, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR-613401

BONAFIDE CERTIFICATE

This is to certify that the report titled "**EXPLORING CONGESTION CONTROL MECHANISMS OF TCP VARIANTS OVER WIRELESS NETWORKS**" submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech**.** is a bonafide record of the work done by **Shri K. SATHYA NAGA PAVAN (Reg. No.124156018, B-Tech CSE Artificial Intelligence and Data Science)** during the academic year 2022-23, in the School of Computing.

Submitted for Project Viva Voce held on ________________

Examiner – I   Examiner - II

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. Umamaheswari**, Dean, School of Computing, and R. Chandramouli, Registrar for their overwhelming support provided during my course span at SASTRA Deemed University.

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

I would specially thank and express my gratitude to **Sujeet Jagtap S**, Associate Professor, School of Computing for providing me with an opportunity to do this project and for his 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 to the completion of my project and for the successful completion of all courses during my academic life at SASTRA Deemed University.

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

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AIMD	Additive Increase Multiplicative Decrease
TCP	Transmission Control Protocol
FTP	File Transfer Protocol
Cnwd	Congestion Window
MSS	Maximum Segment Size
SS	Slow Start

ABSTRACT

Transmission Control Protocol (TCP) is a widely used end-to-end protocol. TCP is a high-reliability protocol for the wired Network. Many TCP variants are modified and developed with respectively with the communication needs. Most TCP current versions include a set of algorithms built to control the congestion in critical links of the network with maintaining the network throughput. Packet losses are not entirely avoidable. These packet losses happen mainly due to congestion. Another key component of TCP is its congestion control mechanisms such as slow start, congestion avoidance, fast retransmit, and fast recovery. But these control mechanisms have reached their limitations in some challenging network environments so it requires further analysis for the development of congestion control mechanisms.TCP works efficiently for the Wireless Network. The Paper also discusses the reasons for congestion control and TCP variants like TCP Tahoe, TCP Vegas, TCP Reno, etc.

KEYWORDS: TCP Tahoe, TCP Vegas, TCP reno, Congestion Avoidance, Slow start, Congestion control

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network.

The TCP/IP model breaks down the data into small bundles and afterward reassembles the bundles into the original message on the opposite end. Sending the information in little bundles of information makes it simpler to maintain efficiency as opposed to sending everything in one go. After a particular message is broken down into bundles, these bundles may travel along multiple routes if one route is jammed but the destination remains the same.

Due to rapid advances in the area of wireless communications and the popularity of the Internet, the provision of packet data services for applications like e-mail, web browsing, mobile computing, etc. over wireless is gaining importance. The TCP/IP protocol suites have several layers, of which the transport layer is used widely for mobility. It uses protocols like TCP and UDP for transferring data.

The wireless environments can be broadly distinguished into three types: Cellular networks, Ad-hoc networks, and Satellite networks. In Cellular Networks, a mobile host is connected to the fixed network with the help of the Base Station. An ad-hoc network is spontaneously formed when devices connect and communicate with each other. These are mostly local area networks (LANs). A Satellite network consists of a ground-based station whose operability is controlled by a transceiver and a network of ground stations that facilitate users to transmit and receive communication traffic over system satellites. But in these Cellular networks is the most common form of Wireless Network currently in use. Mobile devices like cell phones and laptops use this network. Most of the proposed solutions to TCP use this model. All service providers are on the fixed network and hence we have to address the problem of wireless networks only at the endpoint.

CHAPTER 2

RELATED WORKS

Computer networks have experienced explosive growth over the past few years, which has led to some severe congestion problems. Reliable protocols like TCP work well in wired networks where the loss occurs mostly because of congestion. However, in wireless networks, loss occurs because of bit rates and handoffs too. TCP responds to all losses by congestion control and avoidance algorithms, which results in the degradation of TCP's End-To-End performance in wireless networks.

CHAPTER 3

PROPOSED WORK

Transmission Control Protocol (TCP) is one of the important standards in the internet world and is also a very vital element in the internet protocol suite. It provides a connection-oriented service with reliable data transfer over unreliable underlying protocols. It uses sequence numbering and timers to ensure the reliable transfer of packets. TCP's flow control increases the data-sending rate until there are signs of congestion in the network. The basis of TCP congestion control lies in the following algorithms: slow start, congestion, avoidance, fast retransmit, and fast recovery.

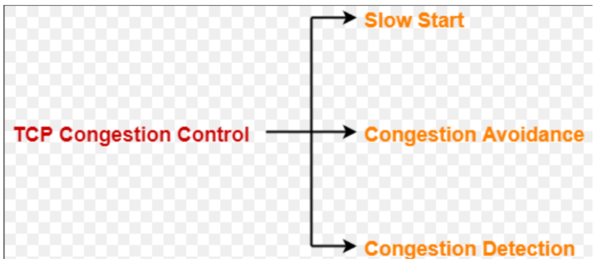
TCP uses the congestion window maintained at the sender side. For each TCP session, a separate congestion window is maintained. This congestion window represents the maximum amount of data that can be sent into the network without being acknowledged.

The size of the sender window is determined by the following two factors-

1. <u>Receiver window size:</u> -
   - Sender should not send data greater than the receiver window size.
   - Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
   - So, the sender should always send data less than or equal to the receiver window size.
   - Receiver dictates its window size to the sender through TCP Header.

2. <u>Congestion window size</u>: -
   - Sender should not send data greater than the congestion window size.
   - Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
   - So, the sender should always send data less than or equal to the congestion window size.
   - Different variants of TCP use different approaches to calculate the size of the congestion window.
   - Congestion window is known only to the sender and is not sent over the links.

The congestion avoidance algorithm is used by the TCP which has mechanisms like Slow-start and congestion window and an AIMD (Additive Increase Multiplicative Decrease), to solve the problem of congestion avoidance. The basis of congestion control in TCP wireless Networks is congestion avoidance. TCP congestion control is working the three phases: Slow-Start Congestion Avoidance Congestion Detection



A. Slow-Start Phase:

In this phase after every RTT the congestion window size increments exponentially.

-starts slowly increment is exponential to the threshold

For Example: Initially cwnd = 1

After 1 RTT, cwnd = 2^(1) = 2

2 RTT, cwnd = 2^(2) = 4

3 RTT, cwnd = 2^(3) = 8

n RTT, cwnd = 2^(n)

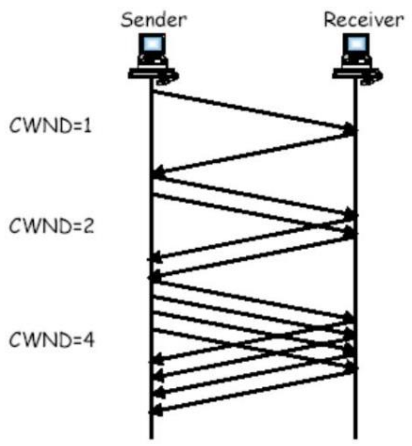


Fig-1. Slow start phase

The above example shows how the congestion window size is going to increase exponentially.

B. Congestion Avoidance Phase:

Additive increment – This phase starts after the threshold value also denoted as *ssthresh*. The size of *cwnd* (congestion window) increases additive. After each RTT cwnd = cwnd + 1.

-After reaching the threshold increment is by 1

For Example: Initially cwnd = i

After 1 RTT, cwnd = i+1

2 RTT, cwnd = i+2

3 RTT, cwnd = i+3

n RTT, cwnd = i+n

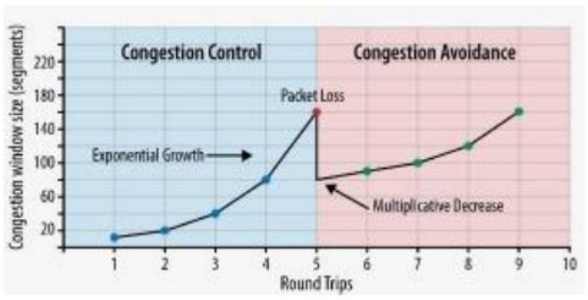


Fig-2. Congestion window size congestion control and avoidance phase

C. Congestion Detection Phase:

If congestion occurs, the congestion window size is decreased. The only way a sender can guess that congestion has occurred is the need to retransmit a segment. Retransmission is needed to recover a missing packet that is assumed to have been dropped by a router due to congestion. Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

- Case 1: Retransmission due to Timeout – In this case, the congestion possibility is high.
  (a) ssthresh is reduced to half of the current window size.
  (b) set cwnd = 1
  (c) start with the slow start phase again.
- Case 2: Retransmission due to 3 Acknowledgement Duplicates – In this case congestion possibility is less. (a) ssthresh value reduces to half of the current windowsize.
  (b)setcwnd=ssthresh
  (c) start with the congestion avoidance phase

Fig-2. Congestion window size congestion control and avoidance phase

CHAPTER 4

SOURCE CODE

```
#===================================
#     Simulation parameters setup
#===================================

set val(chan)   Channel/Wireless Channel   ;# 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)     40                         ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      2282                       ;# X dimension of topography
set val(y)      100                        ;# Y dimension of topography
set val(stop)   25.0                       ;# time of simulation end

#===================================
#       Initialization
#===================================

#Create a ns simulator
set ns [new Simulator]
#Setup topography object
set topo      [new Topography]
```

```
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
#Open the NS trace file
set tracefile [open AODV40.tr w]
$ns trace-all $tracefile
#Open the NAM trace file
set namfile [open AODV40.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];
#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================

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

#===================================
#       Nodes Definition
#===================================


#Create 40 nodes
set n0 [$ns node]
$n0 set X_ 435
$n0 set Y_ 468
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 535
$n1 set Y_ 468
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 635
$n2 set Y_ 468
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 735
$n3 set Y_ 468
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 835
```

```
$n4 set Y_ 468

$n4 set Z_ 0.0

$ns initial_node_pos $n4 20

set n5 [$ns node]

$n5 set X_ 935

$n5 set Y_ 468

$n5 set Z_ 0.0

$ns initial_node_pos $n5 20

set n6 [$ns node]

$n6 set X_ 1035

$n6 set Y_ 468

$n6 set Z_ 0.0

$ns initial_node_pos $n6 20

set n7 [$ns node]

$n7 set X_ 1135

$n7 set Y_ 468

$n7 set Z_ 0.0

$ns initial_node_pos $n7 20

set n8 [$ns node]

$n8 set X_ 435

$n8 set Y_ 368

$n8 set Z_ 0.0

$ns initial_node_pos $n8 20

set n9 [$ns node]

$n9 set X_ 535

$n9 set Y_ 368

$n9 set Z_ 0.0

$ns initial_node_pos $n9 20

set n10 [$ns node]
```

```
$n10 set X_ 635

$n10 set Y_ 368

$n10 set Z_ 0.0

$ns initial_node_pos $n10 20

set n11 [$ns node]

$n11 set X_ 735

$n11 set Y_ 368

$n11 set Z_ 0.0

$ns initial_node_pos $n11 20

set n12 [$ns node]

$n12 set X_ 835

$n12 set Y_ 368

$n12 set Z_ 0.0

$ns initial_node_pos $n12 20

set n13 [$ns node]

$n13 set X_ 935

$n13 set Y_ 368

$n13 set Z_ 0.0

$ns initial_node_pos $n13 20

set n14 [$ns node]

$n14 set X_ 1035

$n14 set Y_ 368

$n14 set Z_ 0.0

$ns initial_node_pos $n14 20

set n15 [$ns node]

$n15 set X_ 1135

$n15 set Y_ 368

$n15 set Z_ 0.0

$ns initial_node_pos $n15 20
```

```
set n16 [$ns node]
$n16 set X_ 435
$n16 set Y_ 268
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 535
$n17 set Y_ 268
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 635
$n18 set Y_ 268
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 735
$n19 set Y_ 268
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 835
$n20 set Y_ 268
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 935
$n21 set Y_ 268
$n21 set Z_ 0.0
```

```
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 1035
$n22 set Y_ 268
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 1135
$n23 set Y_ 268
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 435
$n24 set Y_ 168
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20
set n25 [$ns node]
$n25 set X_ 535
$n25 set Y_ 168
$n25 set Z_ 0.0
$ns initial_node_pos $n25 20
set n26 [$ns node]
$n26 set X_ 635
$n26 set Y_ 168
$n26 set Z_ 0.0
$ns initial_node_pos $n26 20
set n27 [$ns node]
$n27 set X_ 735
$n27 set Y_ 168
```

```
$n27 set Z_ 0.0

$ns initial_node_pos $n27 20

set n28 [$ns node]

$n28 set X_ 835

$n28 set Y_ 168

$n28 set Z_ 0.0

$ns initial_node_pos $n28 20

set n29 [$ns node]

$n29 set X_ 935

$n29 set Y_ 168

$n29 set Z_ 0.0

$ns initial_node_pos $n29 20

set n30 [$ns node]

$n30 set X_ 1035

$n30 set Y_ 168

$n30 set Z_ 0.0

$ns initial_node_pos $n30 20

set n31 [$ns node]

$n31 set X_ 1135

$n31 set Y_ 168

$n31 set Z_ 0.0

$ns initial_node_pos $n31 20

set n32 [$ns node]

$n32 set X_ 435

$n32 set Y_ 68

$n32 set Z_ 0.0

$ns initial_node_pos $n32 20

set n33 [$ns node]

$n33 set X_ 535
```

```
$n33 set Y_ 68
$n33 set Z_ 0.0
$ns initial_node_pos $n33 20
set n34 [$ns node]
$n34 set X_ 635
$n34 set Y_ 68
$n34 set Z_ 0.0
$ns initial_node_pos $n34 20
set n35 [$ns node]
$n35 set X_ 735
$n35 set Y_ 68
$n35 set Z_ 0.0
$ns initial_node_pos $n35 20
set n36 [$ns node]
$n36 set X_ 835
$n36 set Y_ 68
$n36 set Z_ 0.0
$ns initial_node_pos $n36 20
set n37 [$ns node]
$n37 set X_ 935
$n37 set Y_ 68
$n37 set Z_ 0.0
$ns initial_node_pos $n37 20
set n38 [$ns node]
$n38 set X_ 1035
$n38 set Y_ 68
$n38 set Z_ 0.0
$ns initial_node_pos $n38 20
set n39 [$ns node]
```

```
$n39 set X_ 1135

$n39 set Y_ 68

$n39 set Z_ 0.0

$ns initial_node_pos $n39 20


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

#       Agents Definition

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


#Setup a TCP connection

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink2 [new Agent/TCPSink]

$ns attach-agent $n15 $sink2

$ns connect $tcp0 $sink2

$tcp0 set packetSize_ 1500


#Setup a TCP connection

set tcp1 [new Agent/TCP]

$ns attach-agent $n24 $tcp1

set sink3 [new Agent/TCPSink]

$ns attach-agent $n31 $sink3


$ns connect $tcp1 $sink3$tcp1

set packetSize_ 1500


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

  #ApplicationsDefinition

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

```
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 10.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1.0 "$ftp1 start"
$ns at 10.0 "$ftp1 stop"

#===================================
#       Termination
#===================================


#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
   close $namfile
    exec nam AODV40.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

CHAPTER 5

SNAPSHOTS

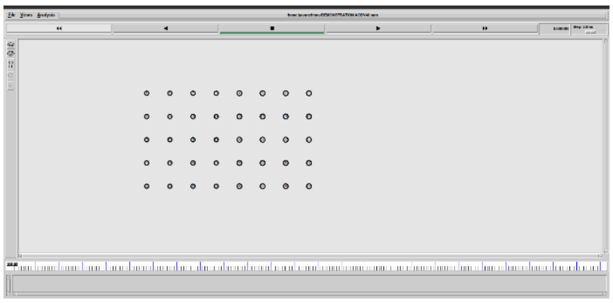
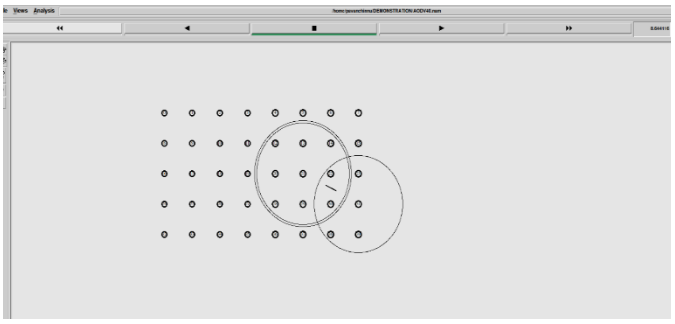
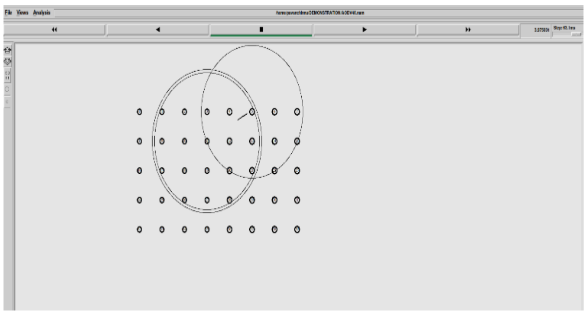


Fig-3. Simulation screen at starting time





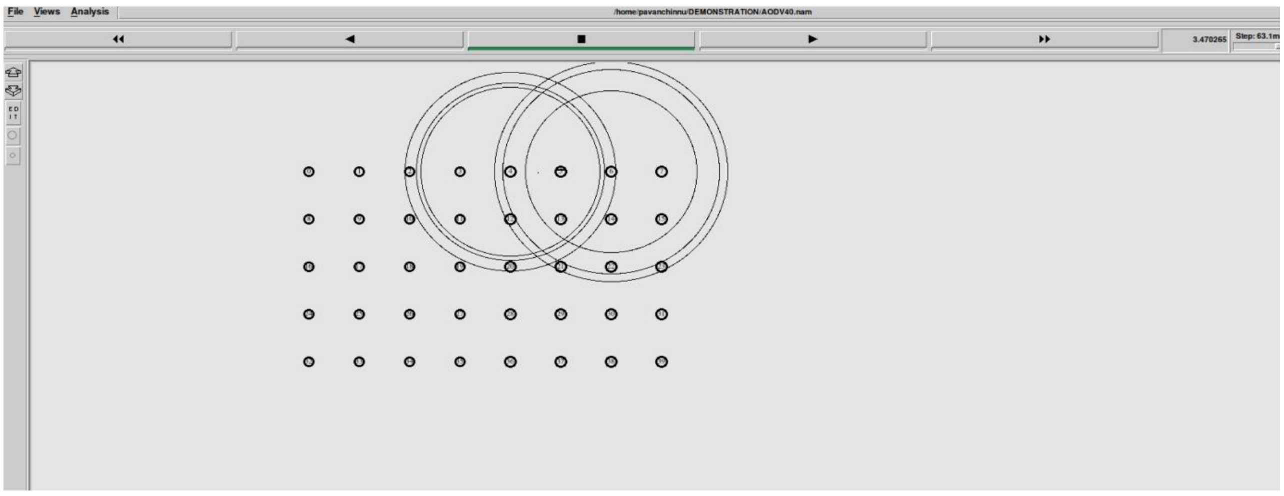


Fig-4,5,6. Simulation screen at intermediate time

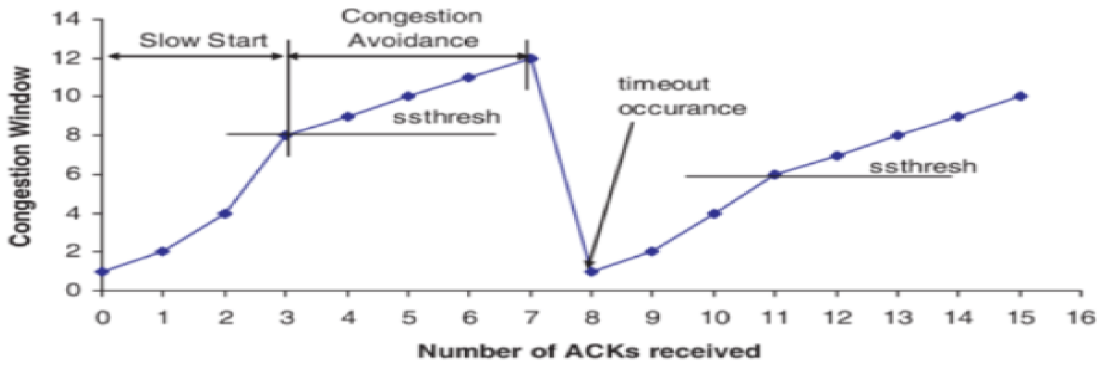


Fig-7: Congestion window graph for TCP

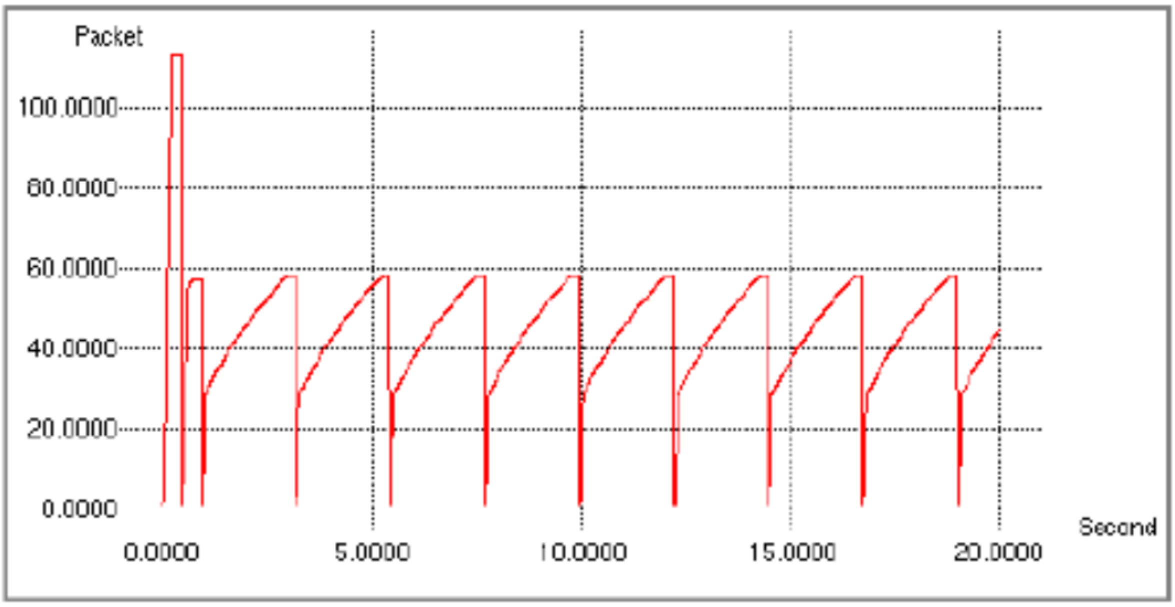


Fig-8: Congestion window graph for Tcp Sack

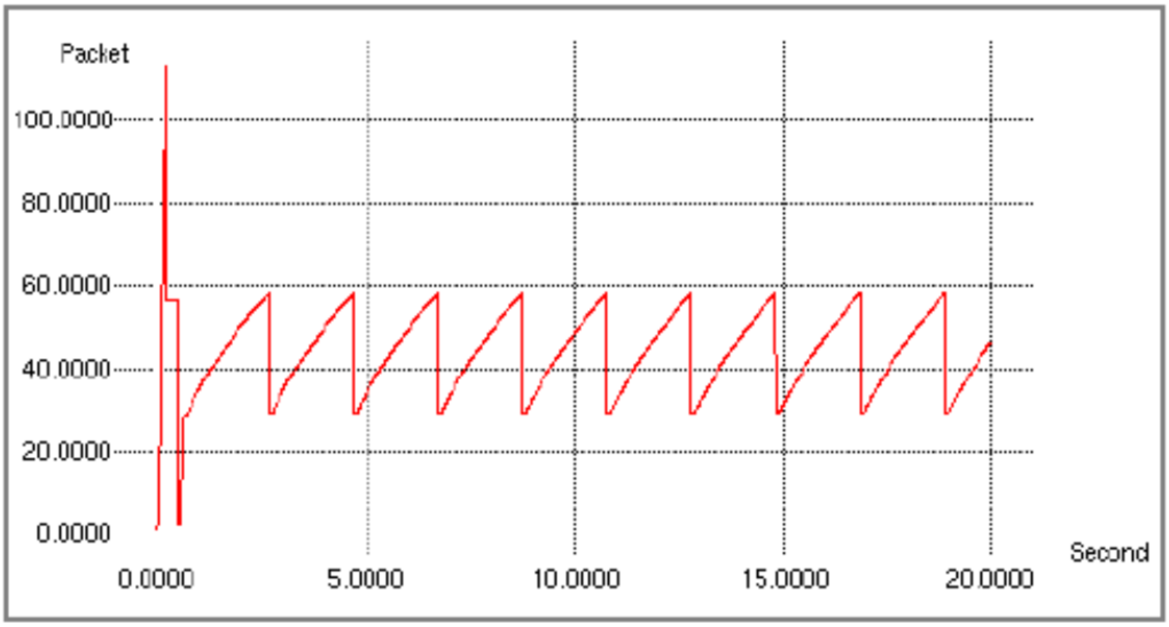


Fig-9. Congestion Window Graph for Tcp Newreno

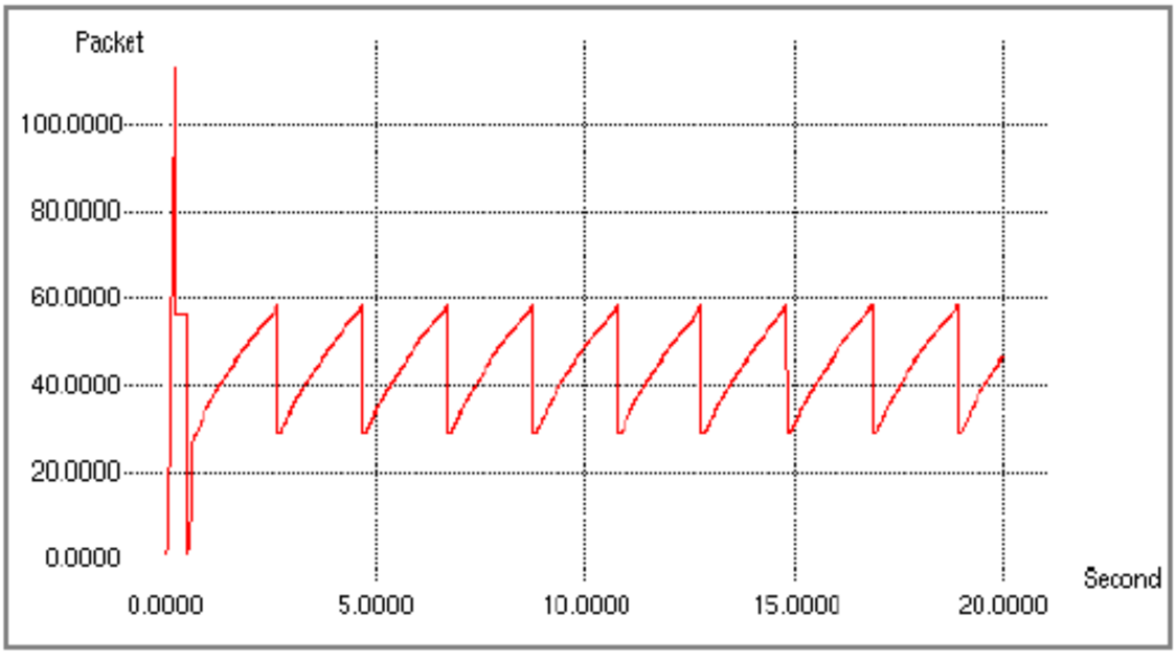


Fig-10. Congestion Window Graph for TCP Reno

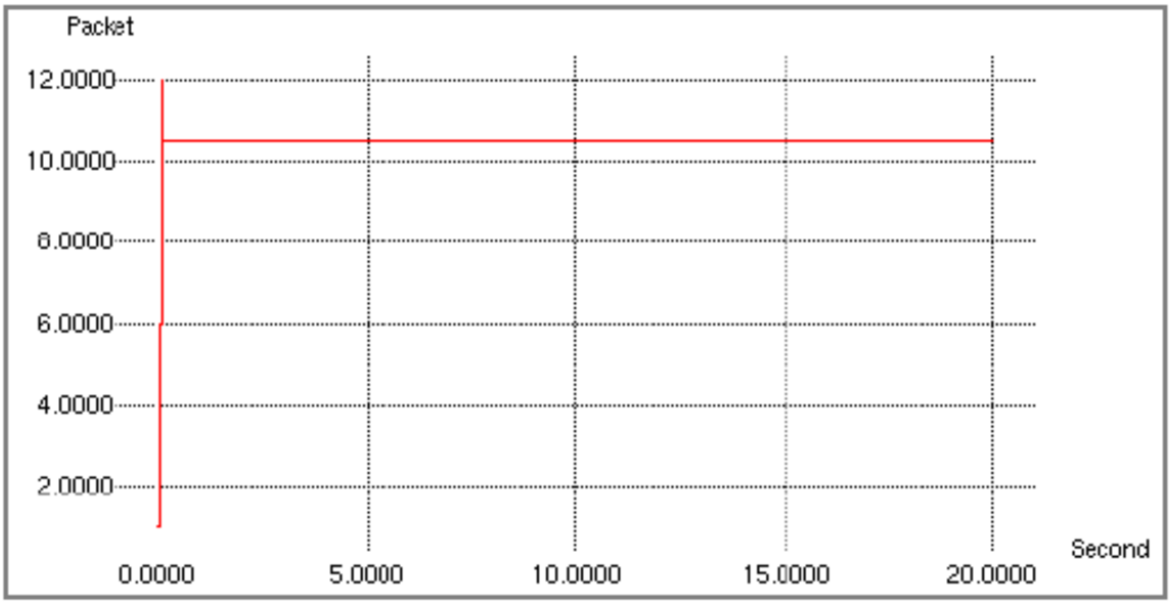


Fig-11. Congestion Window Graph for TCP Vegas

Comparison Graph

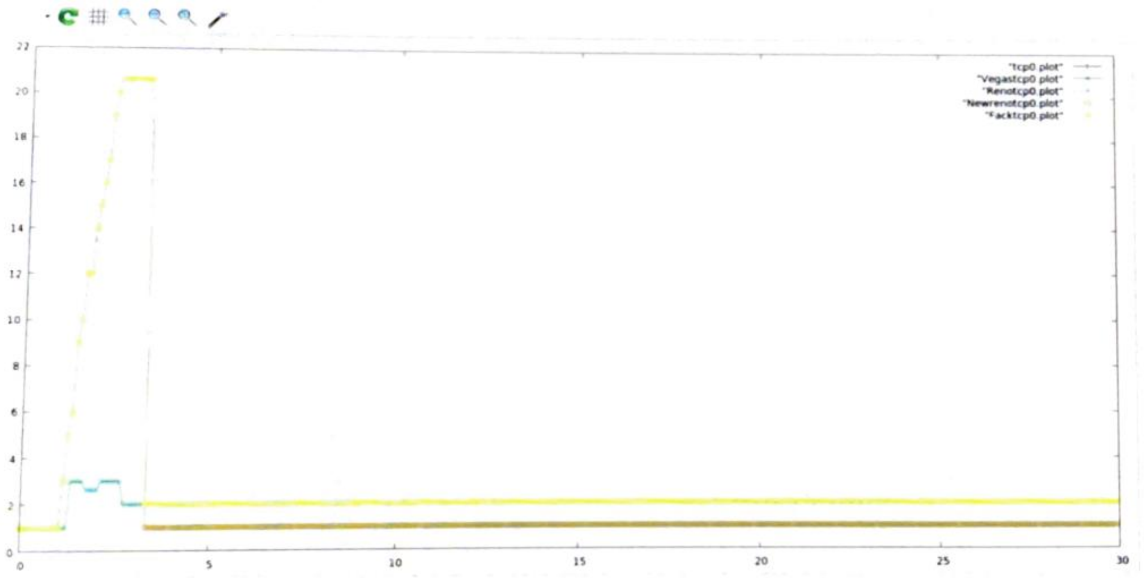


Fig-12. Graph for T Variants

CHAPTER 6

PERFORMANCE EVALUATION

10 Nodes AODV:

Tcp variant	**Packets sent**	**Packets received**	**Packets Delivery percentage**	**Packets loss percentage**	**Throughput**	**Average End-to-End Delay**
Reno	576	560	97.2	2.8	384.12	0.34
New Reno	576	560	97.2	2.8	384.12	0.34
TCP	576	560	97.2	2.8	384.12	0.34
Vegas	516	507	98.2	1.8	351.19	0.01
Sack	576	560	97.2	2.8	384.12	0.34

The throughput and end-to-end delay are least in vegas.

The packet delivery percentage is greater in vegas as there is less amount of packet loss.

20 Nodes AODV:

Tcp variant	**Packets sent**	**Packets received**	**Packets Delivery percentage**	**Packets loss percentage**	**Average Throughput (kbps)**	**Average End-to-End Delay(sec)**
Reno	556	520	93.5	6.5	353.18	0.29
New Reno	568	515	90.06	9.94	350.53	0.32
TCP	570	510	89.4	10.6	382.66	0.35
Vegas	486	472	97.1	2.9	390.62	0.15
Sack	500	472	94.4	5.6	346.86	0.34

The throughput is least in the sack and end-to-end delay is least in vegas.

The overall performance is best in TCP and worst in the sack.

30 Nodes AODV:

Tcp variant	**Packets sent**	**Packets received**	**Packets Delivery percentage**	**Packets loss percentage**	**Average Throughput (kbps)**	**Average End-to-End Delay(sec)**
Reno	185	165	89.19	10.8	90.04	0.21
New Reno	238	213	89.50	10.50	127.66	0.22
TCP	227	201	88.55	11.45	91.47	0.19
Vegas	216	103	95.37	4.63	119.03	0.16
Sack	158	176	86.34	13.66	77.49	0.23

The throughput is least in the sack and end-to-end delay is greater in the sack.

The number of packets delivered is greater in new Reno and the packet loss is the least in vegas.

10 Nodes DSDV:

Tcp variant	Packets sent	Packets received	Packets Delivery percentage	Packet loss percentage	Average Throughput (kbps)	Average End to End Delay(sec)
Reno	2629	2605	99.09	0.91	328.21	0.57
New Reno	2599	2568	98.81	1.19	323.40	0.67
TCP	2615	2582	98.74	1.26	325.28	0.64
Vegas	2101	2052	97.67	2.33	252.06	0.13
Sack	2431	2391	98.35	1.65	301.31	0.81

The throughput is greatest for sack and least in the case of Vegas.

The end-to-end delay is the least in vegas.

The number of packets delivered is greater in TCP.

20 Nodes DSDV:

Tcp Variant	Packets sent	Packets received	Packets Delivery percentage	Packet loss percentage	Average Throughput (kbps)	Average End to End Delay(sec)
Reno	1007	970	96.33	3.67	121.89	0.58
New Reno	1007	970	96.33	3.67	121.89	0.58
TCP	984	943	95.83	4.17	118.49	0.56
Vegas	1021	984	96.38	3.62	120.96	0.15
Sack	865	842	97.34	2.66	107.47	0.53

The minimum throughput is obtained in the sack. The end-to-end delay is the least in vegas.

30 Nodes DSDV:

Tcp Variant	Packets Sent	Packets received	Packets Delivery percentage	Packet loss percentage	Average Throughput (kbps)	Average End to End Delay(sec)
Reno	2031	1980	97.49	2.51	249.12	0.50
New Reno	2096	2043	97.47	2.53	257.17	0.55
TCP	2091	2039	97.51	2.49	256.57	0.54
Vegas	2332	2308	98.97	1.03	283.50	0.09
Sack	2281	2254	98.82	1.18	283.71	0.33

The throughput is greater for sack followed by Vegas. The end-to-end delay is
The least in Vegas. The packets delivered are low in the case of Reno.

CHAPTER 7

CONCLUSION AND FUTURE PLANS

Conclusion:

In the AODV case, Reno shows the best performance followed by New Reno for different cases.

In the DSDV case, New Reno and Sack show good performance and Vegas improves its performance as the number of nodes increases.

We can infer that there is no big difference in Packet Delivery Ratio (PDR) in all variants that perform almost the same.

We can say that Vegas has the lowest delay than other variants.

Future plans:

In the future, the impact of varying speed, pause time, and network size on the energy consumption of the TCP versions can also be analyzed and compared.

CHAPTER 8

REFERENCES

1) Balveer Singh, "*A Comparative Study of Different TCP Variants in Networks*", International Journal of Computer Trends and Technology (IJCTT) vol 4, issue 8, August 2013

2) Range Gowda G J, Sharada U. Shenoy, Sharmila Kumari M, "*Survey on Transmission Control (TCP) Protocol for Wired and Wireless Networks"*, International Journal of Advanced Research in IT and Engineering, vol. 5, issue 9, pp. 5 - 8, May 2016.

3) W. Stevens. "*TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*". RFC 2001, Janua

4) *The Network Simulator - ns-2,* Link: http://www.isi.edu/nsnam/ns