**PROJECT REPORT ON**

**Exploring Congestion Control Mechanisms of TCP Variants over Wired, Wireless, and mixed Wired and Wireless Networks.**

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

**BACHELOR OF TECHNOLOGY**

**Submitted by**

12003301 – YEGYANATHAN V – 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

November(2021)

**SHANMUGHA**

**ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY (SASTRA DEEMED TO BE UNIVERSITY)**

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

**TIRUMALAISAMUDRAM, THANJAVUR – 613401**



**BONAFIDE CERTIFICATE**

Certified that this project work entitled **“Exploring Congestion Control Mechanisms of TCP Variants over Wired, Wireless and mixed wired and wireless Networks.”** submitted to the Shanmugha Arts, Science, Technology & Research Academy (SASTRA Deemed to be University), Tirumalaisamudram - 613401 by Yegyanathan V(123003301), 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**

| S. NO | TOPIC | Page No. |
| --- | --- | --- |
| 1 | ACKNOWLEDGEMENTS | 4 |
| 2 | ABSTRACT | 5 |
| 3 | CHAPTER 1 INTRODUCTION | 12 |
| 4 | CHAPTER 2 RELATED WORKS | 16 |
| 5 | CHAPTER 3 PROPOSED WORK | 20 |
| 6 | CHAPTER 4 SOURCE CODE | 27 |
| 7 | CHAPTER 5 RESULTS | 52 |
| 8 | CHAPTER 6 PERFORMANCE EVALUATION | 57 |
| 9 | CHAPTER 7 CONCLUSION AND FUTURE WORKS | 64 |
| 10 | CHAPTER 8 REFERENCES | 65 |

# **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 support and academic help.

I would like to express my sincere 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 completion the project.

I am grateful to 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.

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**

Network congestion is the reduced quality of service that occurs when a computer node is exposed to an overwhelming quantity of data. Queueing delay, Overflowing of router buffers, packet loss and blocking of new connections are all indications of network congestion. Although disconnected from the intricacies of the network, the end-user often faces the gravity of the issue. He may not be able to use the services provided by the internet as a result of a “network crash”.

TCP is a connection-oriented protocol that resides in the transport layer of the OSI model. It offers reliable and secure communication services to processes that run on different hosts. TCP can be employed in both wired, wireless and hybrid networks. It is equipped with a robust congestion control mechanism which is generally designed using the Additive Increase Multiplicative Decrease (AIMD) approach. It has been shown to optimize congested flow rates in a distributed fashion and is stable. In this approach, the senders can increase the data sending rate until congestion is detected, then decrease the sending rate. Formally, the Congestion Window Size (cwnd) is dynamically adjusted in response to inferred network congestion. Different variants of the TCP protocol are being introduced based on how their congestion control mechanism is implemented. Each variant is either a better version of the previous variant or addresses a unique problem.

In this paper, we will analyze three variants of TCP, TCP Tahoe, TCP Reno, and TCP Vegas by implementing them in wired, wireless, and mixed wired and wireless (hybrid) network topologies. We will numerically compute the throughput, bandwidth and congestion window size in each of the cases. Network simulations for all the cases will be implemented using NS2. Python libraries such as Matplotlib and Seaborn will be used to generate the required graphs. In addition to this, we modify the opencwnd() function by adding a new case and add a new version of the limited\_slow\_start() function named my\_limited\_slow\_start(). These functions reside in the tcp.cc file. The new routine will be combined with TCP Tahoe and will be compared with the other TCP variants.

**KEYWORDS:** Transmission Control Protocol (TCP); Additive Increase Multiplicative Decrease (AIMD); Throughput; Congestion Window Size (cwnd); Acknowledgement; Round Trip Time (RTT); Slow Start; Congestion Avoidance.

**ABBREVIATIONS**

| TCP | Transmission Control Protocol |
| --- | --- |
| FTP | File Transfer Protocol |
| ACK | Acknowledgement |
| Cwnd | Congestion Window Size |
| MSS | Maximum Segment Size |
| RTT | Round Trip Time |
| tt | Total Throughput |
| tb | Total Bandwidth |
| AIMD | Additive Increase Multiplicative Decrease |

**INTRODUCTION**

Mobile data traffic is increasing at a drastic rate. A study from Cisco in the year 2016 predicted that by 2020, videos will account for a staggering 75% of the network traffic, and traffic from mobile phones alone will be 30 exabytes per day. Solving problems related to network congestion is critical since we increasingly rely on our communication networks for services that are considered fundamental such as health, finance, and education.

The Transmission Control Protocol (TCP) ensures secure and reliable transmission of data between a server and a client. It runs only in the end systems and is an abstraction for the intermediate network elements such as the routers and the switches. In TCP protocol, both the server process and the client process are involved in a connection establishment procedure before data packets could be sent. Initially, the client sends an SYN segment to the server, asking for synchronization. An SYN-ACK segment is sent by the server to the client in response to the SYN segment. Finally, the client sends an ACK segment after which the connection is established. The amount of time taken for the signal sent by the server plus the time it takes for the acknowledgment of the signal having been received by the sender is called the Round-Trip Time (RTT). Once the connection is established data packets can be transferred between the processes irrespective of which process initiated the connection.

The server process transfers a stream of data via the socket to the TCP running in the server process. The TCP transfers the data to the send buffer. The send buffer segments the data into pieces and sends them to the receive buffer on the client-side. The TCP running in the client is responsible for reassembling the data as per their port numbers. The maximum data that can be accommodated in a segment is determined by the Maximum Segment Size (MSS). In addition to sending buffer, receive buffer, and other variables (LastByteAcked, LastByteSent, and so on), the TCP connection maintains an additional variable called the Congestion Window Size (cwnd). The cwnd determines the upper bound on the traffic that can be sent by the sender into the network. Based on the network congestion, the congestion control algorithm of TCP dynamically alters the size of the congestion window.

The congestion control algorithm undergoes three phases and they are Slow-Start, Congestion Avoidance, And Congestion Detection. When connection begins, cwnd is set to 1 MSS. In the Slow-Start phase, the cwnd is doubled for every RTT and the sending rate increases exponentially until a loss event occurs. The Congestion Avoidance phase begins when cwnd is greater than the Slow Start Threshold (SSThresh). Thereafter, cwnd is increased linearly. The lost packets are also dealt with. In the Congestion Detection phase, the congestion is inferred and the value of cwnd is dynamically reduced to control the traffic.

In addition to the Slow-Start Phase, Limited Slow-Start Phase was proposed for TCP with Large Congestion Windows. Limited Slow-Start, limits the number of data segments by which the congestion window is increased for one window of data during slow-start, in order to improve performance for TCP connections with large congestion windows.

The congestion control algorithms are classified into loss-based, delay-based and hybrid. Loss-based algorithms respond to packet loss whereas delay-based algorithms respond to an increase in the value of RTT.

TCP has many variants. The variants discussed in this paper are TCP Tahoe, TCP Reno and TCP Vegas. TCP Tahoe and TCP Reno are examples of Loss-based algorithms whereas TCP Vegas is a delay-based algorithm.

**RELATED WORKS**

Vivek Kumar Majithiya et.al (Vivek Kumar Majithiya, 2014) has analyzed five variants of TCP, TCP Tahoe, TCP Reno, TCP NewReno, TCP Vegas, and TCP Westwood by calculating its throughput and delay in wired as well as wireless scenario.

S Islam et.al (S Islam, 2018) proposed TCP Vegas and TCP SACK make some performance improvements to TCP Reno in low bandwidth condition. TCP Vegas achieves higher throughput than Reno and SACK for a large loss rate. The paper also proposes TCP SACK which performs better when more than one packet dropped in one window. TCP Vegas causes much fewer packet retransmissions than TCP Reno and SACK. Finally, the paper concludes with a detailed comparison table and some suggestions in improving the drawback of TCP variants and try to comment which variation is better for which network situation.

Rani K. V. et.al (Rani K. V., 2017) proposes TCP-RQB Agent Algorithm is better than the customary TCP Reno Protocol. The paper also explores New Reno, which identifies various packet losses that are accessible, so it is valuable when there are more packet losses of information. At the point when hearty is basic to various packet losses in one window of information, and loss of information is costly, Sack is reasonable from different Protocols. The paper prescribes Vegas in situations where substantial transmission capacity is required because now is the ideal time based data transmission estimation calculation.

Herlein et.al (Herlein, 2018) has explored the effects of interaction between two or more TCP variants competing for shared resources in a network. The paper explores in order to establish how such coexistence could be achieved, determining in what way two data flows of TCP variants might achieve a balanced state in a channel.

AP Pande et.al (AP Pande, 2018) explores recent TCP variants which are described in categories, such as Loss-Based, Delay-Based and Hybrid using the algorithms and network media. It has been observed that these variants are not developed taking in to consideration of the end applications which has a particular need and expectation from the TCP and hence need to be analyzed for same

HK Molia et.al (HK Molia, 2018) focuses on discussion of traditional TCP variants and various losses in MANETs. TCP variants for MANETs are explained which are classified into cross-layer approaches and layered approaches. The paper proposes a set of TCP variants based on loss handling approaches according to loss differentiation, loss prediction and loss avoidance approaches.

CN González et.al (CN González, 2019) studies the interaction between two or more flows competing for shared resources in a network, and has helped in the understanding of the congestion control mechanisms of various TCP variants.

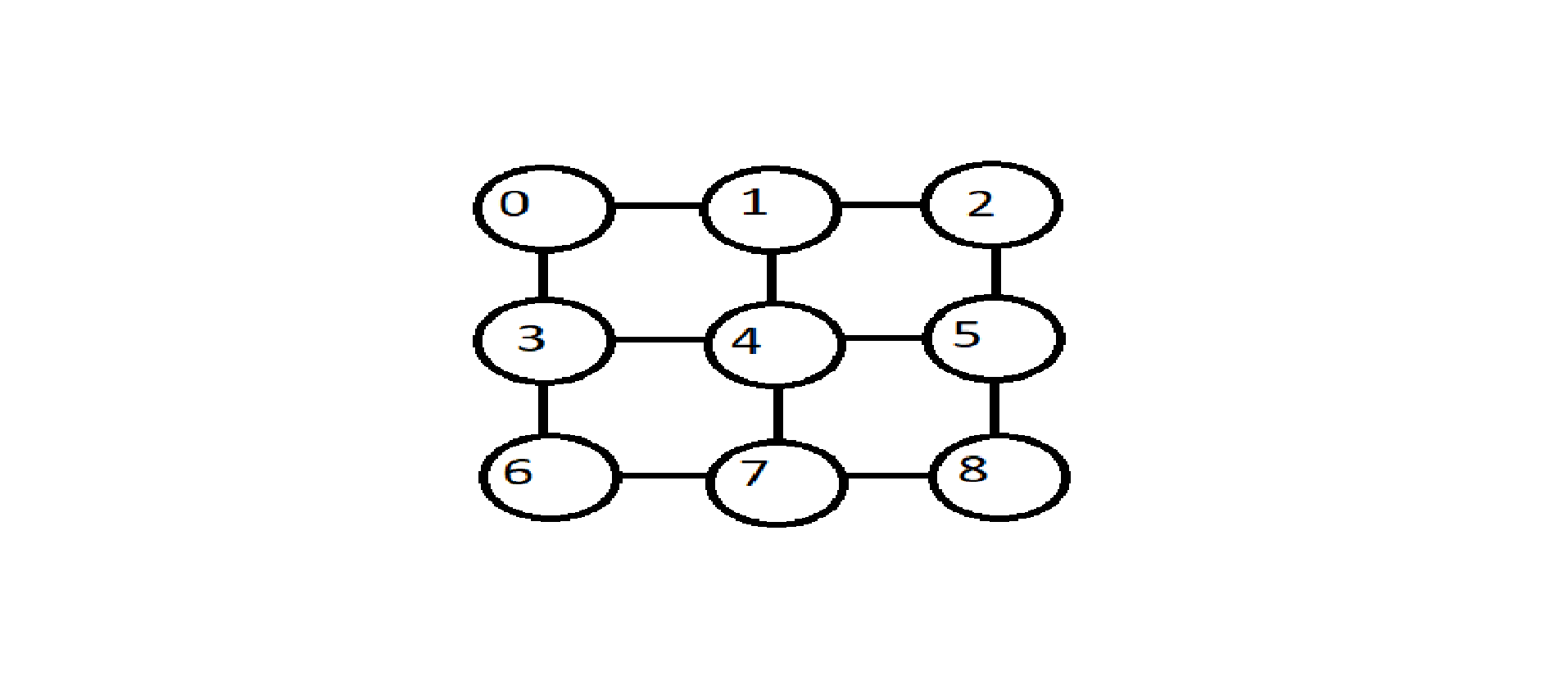
J VijiPriya et.al (J VijiPriya, 2016) aims at designing a congestion control algorithm that performs well across wired-cum-wireless networks and Mobile Adhoc Network. The paper proposes Newton Raphson Congestion Control algorithm and tests the same over wired-cum-wireless network. The paper also shows that the experimental results were significantly improved as compared to other high speed TCP variants.

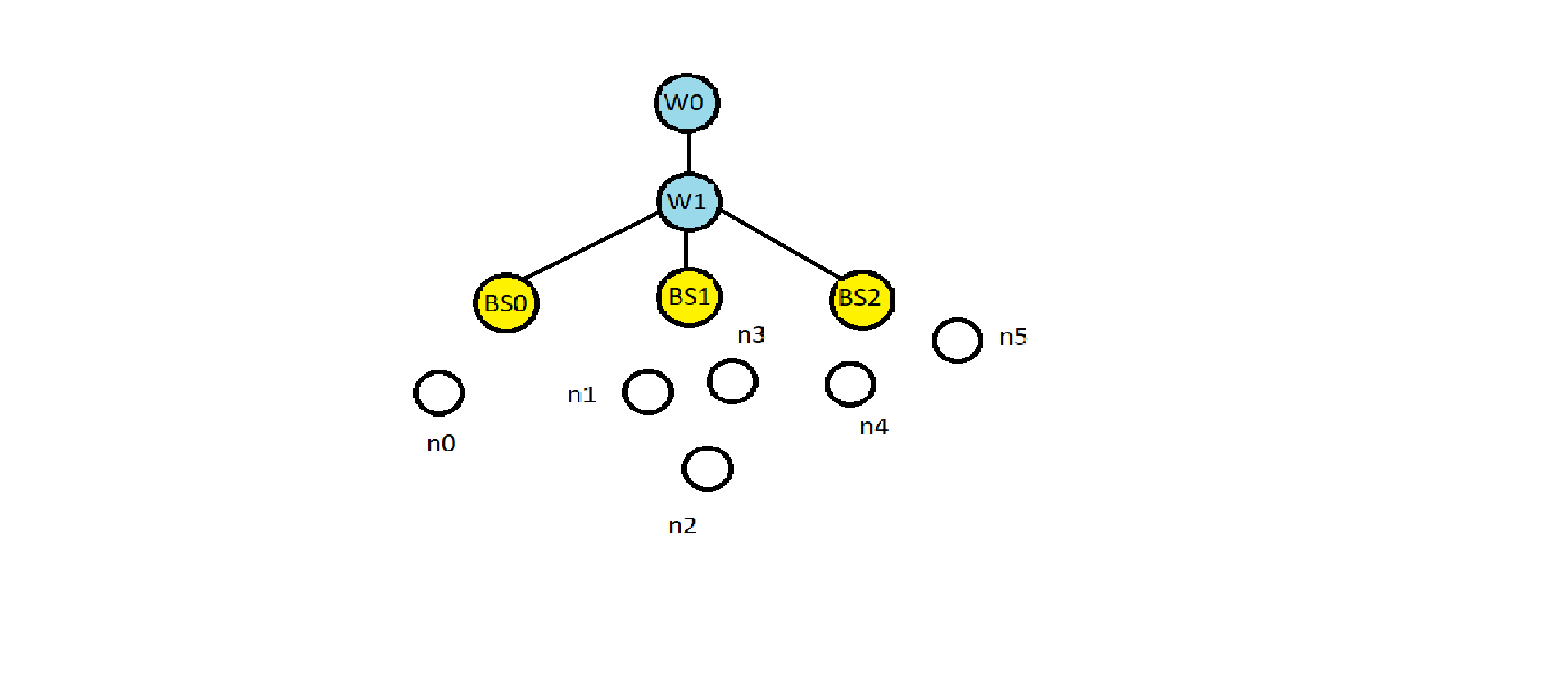
Z Wang et.al (Z Wang, 2016) identifies the available bandwidth, [round trip time](https://www.sciencedirect.com/topics/computer-science/round-trip-time) (RTT) and [packet loss](https://www.sciencedirect.com/topics/engineering/packet-loss) rate can vary over many orders of magnitude, which characterizes the heterogeneity of the Internet.The paper proposes a new TCP protocol known as INVS to cope with the heterogeneity in Internet congestion control.

N Bhargava et.al (N Bhargava, 2017) . Proposes an effective cluster selection protocol for the clustering approach in wireless networking, based on geographical location of the node and its residual energy to reduce the amounts of energy lost during transmission to the base station.

**PROPOSED FRAMEWORK**

Grid Topology:

****

Wired and Wireless Topology (Hybrid):****

**Loss-based Congestion Control Algorithms**

Algorithms that detect packet loss and as a reactive mechanism decrement their Congestion Window Size (cwnd) to control the traffic. Eg. TCP Tahoe, TCP Reno.

Formulas

1. ***Cwnd*** *>=* ***LastByteSent*** *-* ***LastByteAcked***

where,

**Cwnd** - Congestion Window Size

**LastByteSent** - Serial number denoting the Byte of data sent to the stream recently.

**LastByteAcked** - Serial number denoting the Byte of data for which an acknowledgement has been received recently.

1. ***Average Sending Rate*** *=* ***cwnd / RTT***

where,

**RTT -** Round-Trip Time

Algorithm

Slow-Start Phase

When the connection begins, initialize cwnd to 1 Maximum Segment Size (MSS).

cwnd = 1

Thereafter double the cwnd for every ACK received ( Sending rate increases exponentially).

cwnd = cwnd\*2

Congestion Avoidance Phase

When cwnd == SSThresh (cwnd / 2),

increment cwnd linearly i.e,

cwnd = cwnd +1.

Also lost packets are handled.

Congestion Detection Phase

When a loss event (congestion) is detected, cwnd is reduced to control the traffic sent into the network.

For TCP Tahoe,

Loss event is when the server is taking too long to reply to a data request made from another device (timeout).

Cwnd is decremented to the base value of 1 MSS.

cwnd = 1

For TCP Reno,

Loss event is when the server detects a triple duplicate ACK.

Cwnd is cut half

cwnd = cwnd / 2 (sstresh)

**Delay-Based Congestion Control Algorithms**

Algorithms that recognize TCP congestion by detecting the increase in RTT associated with the filling of the queue and handles the cwnd accordingly. Eg. TCP Vegas.

Formulas

**RTT**min = RTT measured considering variable delayed to be minimum (Time taken to traverse uncongested path).

**Uncongested Throughput** = **cwnd** / **RTT**min

**Measured Throughput** = **Bytes sent in the previous RTT** / **RTT**measured

**queue\_use = cwnd − BWE×RTTmin = cwnd × (1 − RTTmin/RTTmeasured)**

Algorithm

If measured throughput very close in value to uncongested throughput:

cwnd = cwnd + 1

If measured throughput below uncongested throughput:

cwnd = cwnd - 1

**SOURCE CODE**

Tahoe\_grid.tcl:

set ns [new Simulator]

set nf [open Tahoe\_grid.nam w]

$ns namtrace-all $nf

set nftr [open Tahoe\_grid.tr w]

$ns trace-all $nftr

set fa [open "tt\_Tahoe\_grid.xg" "w"]

set f1 [open "tb\_Tahoe\_grid.xg" "w"]

proc finish {} {

global ns nf nftr fa f1

$ns flush-trace

close $nf

close $fa

close $f1

exec xgraph tt\_Tahoe\_grid.xg tb\_Tahoe\_grid.xg Tahoe\_grid1.xg Tahoe\_grid2.xg Tahoe\_grid3.xg -geometry 300x300 &

exec nam Tahoe\_grid.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 ftp3 fa f1

global sink1 sink2 sink3

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set bw2 [$sink3 set bytes\_]

set totbw [expr $bw0 + $bw1 + $bw2]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$sink3 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set rows 3

set cols 3

set start [lindex $argv 2]

set stop [lindex $argv 3]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 50ms DropTail

$ns duplex-link $n1 $n4 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 50ms DropTail

$ns duplex-link $n2 $n5 5Mb 10ms DropTail

$ns duplex-link $n3 $n4 2Mb 10ms DropTail

$ns duplex-link $n4 $n5 5Mb 40ms DropTail

$ns duplex-link $n3 $n6 2Mb 10ms DropTail

$ns duplex-link $n4 $n7 3Mb 100ms DropTail

$ns duplex-link $n5 $n8 0.3Mb 200ms DropTail

$ns duplex-link $n6 $n7 0.3Mb 200ms DropTail

$ns duplex-link $n7 $n8 3Mb 200ms DropTail

# Sending node is 0 with agent as Tahoe Agent

set tcp1 [new Agent/TCP]

#$tcp1 set windowOption\_ 25

set tcp2 [new Agent/TCP]

#$tcp2 set windowOption\_ 25

set tcp3 [new Agent/TCP]

#$tcp3 set windowOption\_ 25

$ns attach-agent $n0 $tcp1

$ns attach-agent $n1 $tcp2

$ns attach-agent $n1 $tcp3

# receiving (sink) node is n4

set sink1 [new Agent/TCPSink]

set sink2 [new Agent/TCPSink]

set sink3 [new Agent/TCPSink]

$ns attach-agent $n6 $sink1

$ns attach-agent $n2 $sink2

$ns attach-agent $n7 $sink3

# establish the traffic between the source and sink

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns connect $tcp3 $sink3

# Setup a FTP traffic generator on "tcp1"

set ftp1 [new Application/FTP]

set ftp2 [new Application/FTP]

set ftp3 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ftp2 attach-agent $tcp2

$ftp3 attach-agent $tcp3

$ftp1 set type\_ FTP

$ftp2 set type\_ FTP

$ftp3 set type\_ FTP

# start/stop the traffic

$ns at 0.1 "$ftp1 start"

$ns at 0.55 "record"

$ns at 5 "$ftp2 start"

$ns at 5 "$ftp3 start"

$ns at 40.0 "$ftp1 finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Tahoe\_grid1.xg w]

set outfile2 [open Tahoe\_grid2.xg w]

set outfile3 [open Tahoe\_grid3.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns at 0.0 "plotWindow $tcp3 $outfile3"

$ns run

Reno\_grid.tcl:

#declaring simulation variable

set ns [new Simulator]

#declaring trace variable

set nf [open Reno\_grid.nam w]

$ns namtrace-all $nf

set nftr [open Reno\_grid.tr w]

$ns trace-all $nftr

set fa [open "tt\_Reno\_grid.xg" "w"]

set f1 [open "tb\_Reno\_grid.xg" "w"]

#defining finish function

proc finish {} {

global ns nf nftr fa f1

$ns flush-trace

close $nf

close $fa

close $f1

exec xgraph tt\_Reno\_grid.xg tb\_Reno\_grid.xg Reno\_grid1.xg Reno\_grid2.xg Reno\_grid3.xg -geometry 300x300 &

exec nam Reno\_grid.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 ftp3 fa f1

global sink1 sink2 sink3

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set bw2 [$sink3 set bytes\_]

set totbw [expr $bw0 + $bw1 + $bw2]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$sink3 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set rows 3

set cols 3

set start [lindex $argv 2]

set stop [lindex $argv 3]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 50ms DropTail

$ns duplex-link $n1 $n4 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 50ms DropTail

$ns duplex-link $n2 $n5 5Mb 10ms DropTail

$ns duplex-link $n3 $n4 2Mb 10ms DropTail

$ns duplex-link $n4 $n5 5Mb 40ms DropTail

$ns duplex-link $n3 $n6 2Mb 10ms DropTail

$ns duplex-link $n4 $n7 3Mb 100ms DropTail

$ns duplex-link $n5 $n8 0.3Mb 200ms DropTail

$ns duplex-link $n6 $n7 0.3Mb 200ms DropTail

$ns duplex-link $n7 $n8 3Mb 200ms DropTail

# Sending node is 0 with agent as Tahoe Agent

set tcp1 [new Agent/TCP/Reno]

set tcp2 [new Agent/TCP/Reno]

set tcp3 [new Agent/TCP/Reno]

$ns attach-agent $n0 $tcp1

$ns attach-agent $n1 $tcp2

$ns attach-agent $n1 $tcp3

# receiving (sink) node is n4

set sink1 [new Agent/TCPSink]

set sink2 [new Agent/TCPSink]

set sink3 [new Agent/TCPSink]

$ns attach-agent $n6 $sink1

$ns attach-agent $n2 $sink2

$ns attach-agent $n7 $sink3

# establish the traffic between the source and sink

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns connect $tcp3 $sink3

# Setup a FTP traffic generator on "tcp1"

set ftp1 [new Application/FTP]

set ftp2 [new Application/FTP]

set ftp3 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ftp2 attach-agent $tcp2

$ftp3 attach-agent $tcp3

$ftp1 set type\_ FTP

$ftp2 set type\_ FTP

$ftp3 set type\_ FTP

# start/stop the traffic

$ns at 0.1 "$ftp1 start"

$ns at 0.55 "record"

$ns at 5 "$ftp2 start"

$ns at 5 "$ftp3 start"

$ns at 40.0 "$ftp1 finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Reno\_grid1.xg w]

set outfile2 [open Reno\_grid2.xg w]

set outfile3 [open Reno\_grid3.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns at 0.0 "plotWindow $tcp3 $outfile3"

$ns run

Vegas\_grid.tcl:

set ns [new Simulator]

set nf [open Vegas\_grid.nam w]

$ns namtrace-all $nf

set nftr [open Vegas\_grid.tr w]

$ns trace-all $nftr

set fa [open "tt\_Vegas\_grid.xg" "w"]

set f1 [open "tb\_Vegas\_grid.xg" "w"]

proc finish {} {

global ns nf nftr fa f1

$ns flush-trace

close $nf

close $fa

close $f1

exec xgraph tt\_Vegas\_grid.xg tb\_Vegas\_grid.xg Vegas\_grid1.xg Vegas\_grid2.xg Vegas\_grid3.xg -geometry 300x300 &

exec nam Vegas\_grid.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 ftp3 fa f1

global sink1 sink2 sink3

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set bw2 [$sink3 set bytes\_]

set totbw [expr $bw0 + $bw1 + $bw2]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$sink3 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set rows 3

set cols 3

#set start [lindex $argv 2]

#set stop [lindex $argv 3]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 50ms DropTail

$ns duplex-link $n1 $n4 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 50ms DropTail

$ns duplex-link $n2 $n5 5Mb 10ms DropTail

$ns duplex-link $n3 $n4 2Mb 10ms DropTail

$ns duplex-link $n4 $n5 5Mb 40ms DropTail

$ns duplex-link $n3 $n6 2Mb 10ms DropTail

$ns duplex-link $n4 $n7 3Mb 100ms DropTail

$ns duplex-link $n5 $n8 0.3Mb 200ms DropTail

$ns duplex-link $n6 $n7 0.3Mb 200ms DropTail

$ns duplex-link $n7 $n8 3Mb 200ms DropTail

# Sending node is 0 with agent as Tahoe Agent

set tcp1 [new Agent/TCP/Vegas]

set tcp2 [new Agent/TCP/Vegas]

set tcp3 [new Agent/TCP/Vegas]

$ns attach-agent $n0 $tcp1

$ns attach-agent $n1 $tcp2

$ns attach-agent $n1 $tcp3

# receiving (sink) node is n4

set sink1 [new Agent/TCPSink]

set sink2 [new Agent/TCPSink]

set sink3 [new Agent/TCPSink]

$ns attach-agent $n6 $sink1

$ns attach-agent $n2 $sink2

$ns attach-agent $n7 $sink3

# establish the traffic between the source and sink

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns connect $tcp3 $sink3

# Setup a FTP traffic generator on "tcp1"

set ftp1 [new Application/FTP]

set ftp2 [new Application/FTP]

set ftp3 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ftp2 attach-agent $tcp2

$ftp3 attach-agent $tcp3

$ftp1 set type\_ FTP

$ftp2 set type\_ FTP

$ftp3 set type\_ FTP

# start/stop the traffic

$ns at 0.1 "$ftp1 start"

$ns at 0.55 "record"

$ns at 5 "$ftp2 start"

$ns at 5 "$ftp3 start"

$ns at 40.0 "$ftp1 finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Vegas\_grid1.xg w]

set outfile2 [open Vegas\_grid2.xg w]

set outfile3 [open Vegas\_grid3.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns at 0.0 "plotWindow $tcp3 $outfile3"

$ns run

Tahoe\_wireless.tcl:

set val(chan) Channel/WirelessChannel

set val(prop) Propagation/TwoRayGround

set val(netif) Phy/WirelessPhy

set val(mac) Mac/802\_11

set val(ifq) Queue/DropTail/PriQueue

set val(ll) LL

set val(ant) Antenna/OmniAntenna

set val(ifqlen) 50

set val(nn) 10

set val(rp) AODV

set val(x) 500

set val(y) 500

#declaring simulation and trace variable.

set ns [new Simulator]

set tracefile [open Tahoe\_wireless.tr w]

$ns trace-all $tracefile

set namfile [open Tahoe\_wireless.nam w]

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

set fa [open "tt\_Tahoe\_wireless.xg" "w"]

set f1 [open "tb\_Tahoe\_wireless.xg" "w"]

proc finish {} {

global ns tracefile namfile fa f1

$ns flush-trace

close $tracefile

close $namfile

close $fa

close $f1

exec xgraph tt\_Tahoe\_wireless.xg tb\_Tahoe\_wireless.xg Tahoe\_wireless1.xg Tahoe\_wireless2.xg -geometry 300x300 &

exec nam Tahoe\_wireless.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

#creating god variable

create-god $val(nn)

$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) \

-topoInstance $topo \

-agentTrace ON \

-macTrace ON \

-routerTrace ON \

-movementTrace ON \

-channel [new $val(chan)]

#defining the number of nodes.

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

set n($i) [$ns node]

$n($i) random-motion 0

$ns initial\_node\_pos $n($i) 50

}

$n(0) set X\_ -110.0

$n(0) set Y\_ 350.0

$n(0) set Z\_ 0.0

$n(1) set X\_ -10.0

$n(1) set Y\_ 350.0

$n(1) set Z\_ 0.0

$n(2) set X\_ 50.0

$n(2) set Y\_ 450.0

$n(2) set Z\_ 0.0

$n(3) set X\_ -120.0

$n(3) set Y\_ 100.0

$n(3) set Z\_ 0.0

$n(4) set X\_ 90.0

$n(4) set Y\_ 100.0

$n(4) set Z\_ 0.0

$n(5) set X\_ -200.0

$n(5) set Y\_ 200.0

$n(5) set Z\_ 0.0

$n(6) set X\_ 90.0

$n(6) set Y\_ 200.0

$n(6) set Z\_ 0.0

$n(7) set X\_ 400.0

$n(7) set Y\_ 100.0

$n(7) set Z\_ 0.0

$n(8) set X\_ -410.0

$n(8) set Y\_ 320.0

$n(8) set Z\_ 0.0

$n(9) set X\_ 280.0

$n(9) set Y\_ -140.0

$n(9) set Z\_ 0.0

$ns at 1.0 "$n(0) setdest 70.0 50.0 0.0"

$ns at 1.0 "$n(1) setdest 20.0 30.0 0.0"

$ns at 1.0 "$n(2) setdest 200.0 450.0 85.0"

$ns at 1.0 "$n(3) setdest 40.0 100.0 0.0"

$ns at 1.0 "$n(4) setdest 30.0 45.0 0.0"

$ns at 1.0 "$n(5) setdest 80.0 200.0 0.0"

$ns at 1.0 "$n(6) setdest 50.0 350.0 0.0"

$ns at 1.0 "$n(7) setdest 300.0 145.0 0.0"

$ns at 1.0 "$n(8) setdest 110.0 230.0 0.0"

$ns at 1.0 "$n(9) setdest 250.0 50.0 0.0"

$n(0) color "red"

$ns at 1.0 "$n(0) color red"

$n(5) color "red"

$ns at 1.0 "$n(5) color red"

$n(2) color "darkgreen"

$ns at 1.0 "$n(2) color darkgreen"

# setting tcp and ftp agents

set tcp1 [new Agent/TCP]

set sink1 [new Agent/TCPSink]

$ns attach-agent $n(0) $tcp1

$ns attach-agent $n(5) $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

set tcp2 [new Agent/TCP]

set sink2 [new Agent/TCPSink]

$ns attach-agent $n(2) $tcp2

$ns attach-agent $n(6) $sink2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns at 0.55 "record"

$ns at 1.0 "$ftp1 start"

$ns at 1.0 "$ftp2 start"

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns at 14.0 "$ftp1 stop"

$ns at 10.0 "$ftp2 stop"

$ns at 15.0 "finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Tahoe\_wireless1.xg w]

set outfile2 [open Tahoe\_wireless2.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns run

Reno\_wireless.tcl:

set val(chan) Channel/WirelessChannel

set val(prop) Propagation/TwoRayGround

set val(netif) Phy/WirelessPhy

set val(mac) Mac/802\_11

set val(ifq) Queue/DropTail/PriQueue

set val(ll) LL

set val(ant) Antenna/OmniAntenna

set val(ifqlen) 50

set val(nn) 10

set val(rp) AODV

set val(x) 500

set val(y) 500

set ns [new Simulator]

set tracefile [open Reno\_wireless.tr w]

$ns trace-all $tracefile

set namfile [open Reno\_wireless.nam w]

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

set fa [open "tt\_Reno\_wireless.xg" "w"]

set f1 [open "tb\_Reno\_wireless.xg" "w"]

proc finish {} {

global ns tracefile namfile fa f1

$ns flush-trace

close $tracefile

close $namfile

close $fa

close $f1

exec xgraph tt\_Reno\_wireless.xg tb\_Reno\_wireless.xg Reno\_wireless1.xg Reno\_wireless2.xg -geometry 300x300 &

exec nam Reno\_wireless.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

#creating god variable

create-god $val(nn)

$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) \

-topoInstance $topo \

-agentTrace ON \

-macTrace ON \

-routerTrace ON \

-movementTrace ON \

-channel [new $val(chan)]

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

set n($i) [$ns node]

$n($i) random-motion 0

$ns initial\_node\_pos $n($i) 50

}

$n(0) set X\_ -110.0

$n(0) set Y\_ 350.0

$n(0) set Z\_ 0.0

$n(1) set X\_ -10.0

$n(1) set Y\_ 350.0

$n(1) set Z\_ 0.0

$n(2) set X\_ 50.0

$n(2) set Y\_ 450.0

$n(2) set Z\_ 0.0

$n(3) set X\_ -120.0

$n(3) set Y\_ 100.0

$n(3) set Z\_ 0.0

$n(4) set X\_ 90.0

$n(4) set Y\_ 100.0

$n(4) set Z\_ 0.0

$n(5) set X\_ -200.0

$n(5) set Y\_ 200.0

$n(5) set Z\_ 0.0

$n(6) set X\_ 90.0

$n(6) set Y\_ 200.0

$n(6) set Z\_ 0.0

$n(7) set X\_ 400.0

$n(7) set Y\_ 100.0

$n(7) set Z\_ 0.0

$n(8) set X\_ -410.0

$n(8) set Y\_ 320.0

$n(8) set Z\_ 0.0

$n(9) set X\_ 280.0

$n(9) set Y\_ -140.0

$n(9) set Z\_ 0.0

$ns at 1.0 "$n(0) setdest 70.0 50.0 0.0"

$ns at 1.0 "$n(1) setdest 20.0 30.0 0.0"

$ns at 1.0 "$n(2) setdest 200.0 450.0 85.0"

$ns at 1.0 "$n(3) setdest 40.0 100.0 0.0"

$ns at 1.0 "$n(4) setdest 30.0 45.0 0.0"

$ns at 1.0 "$n(5) setdest 80.0 200.0 0.0"

$ns at 1.0 "$n(6) setdest 50.0 350.0 0.0"

$ns at 1.0 "$n(7) setdest 300.0 145.0 0.0"

$ns at 1.0 "$n(8) setdest 110.0 230.0 0.0"

$ns at 1.0 "$n(9) setdest 250.0 50.0 0.0"

$n(0) color "red"

$ns at 1.0 "$n(0) color red"

$n(5) color "red"

$ns at 1.0 "$n(5) color red"

$n(2) color "darkgreen"

$ns at 1.0 "$n(2) color darkgreen"

#setting tcp and ftp agents

set tcp1 [new Agent/TCP/Reno]

set sink1 [new Agent/TCPSink]

$ns attach-agent $n(0) $tcp1

$ns attach-agent $n(5) $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

set tcp2 [new Agent/TCP/Reno]

set sink2 [new Agent/TCPSink]

$ns attach-agent $n(2) $tcp2

$ns attach-agent $n(6) $sink2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns at 0.55 "record"

$ns at 1.0 "$ftp1 start"

$ns at 1.0 "$ftp2 start"

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns at 14.0 "$ftp1 stop"

$ns at 10.0 "$ftp2 stop"

$ns at 15.0 "finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Reno\_wireless1.xg w]

set outfile2 [open Reno\_wireless2.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns run

Vegas\_wireless.tcl:

set val(chan) Channel/WirelessChannel

set val(prop) Propagation/TwoRayGround

set val(netif) Phy/WirelessPhy

set val(mac) Mac/802\_11

set val(ifq) Queue/DropTail/PriQueue

set val(ll) LL

set val(ant) Antenna/OmniAntenna

set val(ifqlen) 50

set val(nn) 10

set val(rp) AODV

set val(x) 500

set val(y) 500

set ns [new Simulator]

set tracefile [open Vegas\_wireless.tr w]

$ns trace-all $tracefile

set namfile [open Vegas\_wireless.nam w]

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

set fa [open "tt\_Vegas\_wireless.xg" "w"]

set f1 [open "tb\_Vegas\_wireless.xg" "w"]

proc finish {} {

global ns tracefile namfile fa f1

$ns flush-trace

close $tracefile

close $namfile

close $fa

close $f1

exec xgraph tt\_Vegas\_wireless.xg tb\_Vegas\_wireless.xg Vegas\_wireless1.xg Vegas\_wireless2.xg -geometry 300x300 &

exec nam Vegas\_wireless.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

create-god $val(nn)

$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) \

-topoInstance $topo \

-agentTrace ON \

-macTrace ON \

-routerTrace ON \

-movementTrace ON \

-channel [new $val(chan)]

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

set n($i) [$ns node]

$n($i) random-motion 0

$ns initial\_node\_pos $n($i) 50

}

$n(0) set X\_ -110.0

$n(0) set Y\_ 350.0

$n(0) set Z\_ 0.0

$n(1) set X\_ -10.0

$n(1) set Y\_ 350.0

$n(1) set Z\_ 0.0

$n(2) set X\_ 50.0

$n(2) set Y\_ 450.0

$n(2) set Z\_ 0.0

$n(3) set X\_ -120.0

$n(3) set Y\_ 100.0

$n(3) set Z\_ 0.0

$n(4) set X\_ 90.0

$n(4) set Y\_ 100.0

$n(4) set Z\_ 0.0

$n(5) set X\_ -200.0

$n(5) set Y\_ 200.0

$n(5) set Z\_ 0.0

$n(6) set X\_ 90.0

$n(6) set Y\_ 200.0

$n(6) set Z\_ 0.0

$n(7) set X\_ 400.0

$n(7) set Y\_ 100.0

$n(7) set Z\_ 0.0

$n(8) set X\_ -410.0

$n(8) set Y\_ 320.0

$n(8) set Z\_ 0.0

$n(9) set X\_ 280.0

$n(9) set Y\_ -140.0

$n(9) set Z\_ 0.0

$ns at 1.0 "$n(0) setdest 70.0 50.0 0.0"

$ns at 1.0 "$n(1) setdest 20.0 30.0 0.0"

$ns at 1.0 "$n(2) setdest 200.0 450.0 85.0"

$ns at 1.0 "$n(3) setdest 40.0 100.0 0.0"

$ns at 1.0 "$n(4) setdest 30.0 45.0 0.0"

$ns at 1.0 "$n(5) setdest 80.0 200.0 0.0"

$ns at 1.0 "$n(6) setdest 50.0 350.0 0.0"

$ns at 1.0 "$n(7) setdest 300.0 145.0 0.0"

$ns at 1.0 "$n(8) setdest 110.0 230.0 0.0"

$ns at 1.0 "$n(9) setdest 250.0 50.0 0.0"

$n(0) color "red"

$ns at 1.0 "$n(0) color red"

$n(5) color "red"

$ns at 1.0 "$n(5) color red"

$n(2) color "darkgreen"

$ns at 1.0 "$n(2) color darkgreen"

set tcp1 [new Agent/TCP/Vegas]

set sink1 [new Agent/TCPSink]

$ns attach-agent $n(0) $tcp1

$ns attach-agent $n(5) $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

set tcp2 [new Agent/TCP/Vegas]

set sink2 [new Agent/TCPSink]

$ns attach-agent $n(2) $tcp2

$ns attach-agent $n(6) $sink2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns at 0.55 "record"

$ns at 1.0 "$ftp1 start"

$ns at 1.0 "$ftp2 start"

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns at 14.0 "$ftp1 stop"

$ns at 10.0 "$ftp2 stop"

$ns at 15.0 "finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Vegas\_wireless1.xg w]

set outfile2 [open Vegas\_wireless2.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns run

Tahoe\_wandw.tcl:

set opt(chan) Channel/WirelessChannel ;# channel type

set opt(prop) Propagation/TwoRayGround ;# radio-propagation model

set opt(netif) Phy/WirelessPhy ;# network interface type

set opt(mac) Mac/802\_11 ;# MAC type

set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type

set opt(ll) LL ;# link layer type

set opt(ant) Antenna/OmniAntenna ;# antenna model

set opt(ifqlen) 50 ;# max packet in ifq

set opt(nn) 6 ;# number of mobilenodes

set opt(adhocRouting) DSDV ;# routing protocol

#set opt(cp) "tcpgen.tcl" ;# connection pattern file

#set opt(sc) "scene-7" ;# node movement file.

set opt(x) 670 ;# x coordinate of topology

set opt(y) 670 ;# y coordinate of topology

set opt(seed) 0.0 ;# seed for random number gen.

set opt(stop) 250 ;# time to stop simulation

set opt(ftp1-start) 160.0

set opt(ftp2-start) 170.0

set num\_wired\_nodes 2

set num\_bs\_nodes 3

#11 nodes.

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

# check for boundary parameters and random seed

if { $opt(x) == 0 || $opt(y) == 0 } {

puts "No X-Y boundary values given for wireless topology\n"

}

if {$opt(seed) > 0} {

puts "Seeding Random number generator with $opt(seed)\n"

ns-random $opt(seed)

}

# create simulator instance

set ns\_ [new Simulator]

# set up for hierarchical routing

$ns\_ node-config -addressType hierarchical

AddrParams set domain\_num\_ 2 ;# number of domains

lappend cluster\_num 2 3 ;# number of clusters in each domain

#cluster 0, cluster 1, cluster 2 and cluster 3

AddrParams set cluster\_num\_ $cluster\_num

lappend eilastlevel 1 1 4 2 3 ;# number of nodes in each cluster

AddrParams set nodes\_num\_ $eilastlevel ;# of each domain

set tracefd [open Tahoe\_wandw.tr w]

set namtrace [open Tahoe\_wandw.nam w]

$ns\_ trace-all $tracefd

$ns\_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

set fa [open "tt\_Tahoe\_wandw.xg" "w"]

set f1 [open "tb\_Tahoe\_wandw.xg" "w"]

proc finish {} {

global ns\_ tracefd namtrace

$ns\_ flush-trace

close $tracefd

close $namtrace

exec xgraph tt\_Tahoe\_wandw.xg tb\_Tahoe\_wandw.xg Tahoe\_wandw1.xg Tahoe\_wandw2.xg -geometry 300x300 &

exec nam Tahoe\_wandw.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

# Create topography object

set topo [new Topography]

# define topology

$topo load\_flatgrid $opt(x) $opt(y)

# create God

create-god [expr $opt(nn) + $num\_bs\_nodes]

# 9 wireless nodes, GOD object will take care.

#create wired nodes

set temp {0.0.0 0.1.0} ;# hierarchical addresses for wired domain

for {set i 0} {$i < $num\_wired\_nodes} {incr i} {

set W($i) [$ns\_ node [lindex $temp $i]]

}

# configure for base-station node

$ns\_ node-config -adhocRouting $opt(adhocRouting) \

-llType $opt(ll) \

-macType $opt(mac) \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-channelType $opt(chan) \

-topoInstance $topo \

-wiredRouting ON \

-agentTrace ON \

-routerTrace OFF \

-macTrace OFF

#create base-station node

set temp {1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 1.1.0 1.1.1 1.1.2 1.1.3 } ;# hier address to be used for wireless

;# domain

set BS(0) [$ns\_ node 1.0.0]

$BS(0) random-motion 0 ;# disable random motion

set BS(1) [$ns\_ node 1.1.0]

$BS(1) random-motion 0

set BS(2) [$ns\_ node 1.2.0]

$BS(2) random-motion 0

#provide some co-ord (fixed) to base station node

$BS(0) set X\_ 200.0

$BS(0) set Y\_ 200.0

$BS(0) set Z\_ 0.0

$BS(1) set X\_ 250.0

$BS(1) set Y\_ 250.0

$BS(1) set Z\_ 0.0

$BS(2) set X\_ 100.0

$BS(2) set Y\_ 180.0

$BS(2) set Z\_ 0.0

# create mobilenodes in the same domain as BS(0)

# note the position and movement of mobilenodes is as defined

# in $opt(sc)

#configure for mobilenodes

$ns\_ node-config -wiredRouting OFF

# attaching the mobile nodes to Base Station 0

set node\_(0) [$ns\_ node 1.0.1]

$node\_(0) base-station [AddrParams addr2id [$BS(0) node-addr]]

#attaching mobile nodes to Base station 1

set node\_(1) [$ns\_ node 1.1.1]

$node\_(1) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(2) [$ns\_ node 1.1.2]

$node\_(2) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(3) [$ns\_ node 1.1.3]

$node\_(3) base-station [AddrParams addr2id [$BS(1) node-addr]]

#attaching mobile nodes to Base station 2

set node\_(4) [$ns\_ node 1.2.1]

$node\_(4) base-station [AddrParams addr2id [$BS(2) node-addr]]

set node\_(5) [$ns\_ node 1.2.2]

$node\_(5) base-station [AddrParams addr2id [$BS(2) node-addr]]

for {set j 0} {$j < 6} {incr j} {

$ns\_ initial\_node\_pos $node\_($j) 40

}

$node\_(0) set X\_ 433.194578912685

$node\_(0) set Y\_ 86.805779539422

$node\_(0) set Z\_ 0.000000000000

$node\_(1) set X\_ 182.146343432004

$node\_(1) set Y\_ 589.457950263110

$node\_(1) set Z\_ 0.000000000000

$node\_(2) set X\_ 636.712900082647

$node\_(2) set Y\_ 227.793717407876

$node\_(2) set Z\_ 0.000000000000

$node\_(3) set X\_ 355.730559281529

$node\_(3) set Y\_ 636.858769794140

$node\_(3) set Z\_ 0.000000000000

$node\_(4) set X\_ 412.570805501828

$node\_(4) set Y\_ 140.390797313602

$node\_(4) set Z\_ 0.000000000000

$node\_(5) set X\_ 192.498199648602

$node\_(5) set Y\_ 389.172056773162

$node\_(5) set Z\_ 0.000000000000

$ns\_ at 0.000000000000 "$node\_(0) setdest 533.982439606531 316.260671641729 10.420081258606"

$ns\_ at 0.000000000000 "$node\_(1) setdest 161.810128580897 443.165191912176 10.922787425094"

$ns\_ at 0.000000000000 "$node\_(2) setdest 639.905294214911 171.358564559506 15.318998910103"

$ns\_ at 0.000000000000 "$node\_(3) setdest 542.673393030666 413.384013118659 12.552371653936"

$ns\_ at 0.000000000000 "$node\_(4) setdest 498.263533357255 244.647811683068 14.896733048027"

$ns\_ at 0.000000000000 "$node\_(5) setdest 597.554489327905 123.928273538379 16.983474910390"

#create links between wired and BS nodes

$ns\_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail

$ns\_ duplex-link $W(1) $BS(0) 5Mb 0.2ms DropTail

$ns\_ duplex-link $W(1) $BS(1) 15Mb 3ms DropTail

$ns\_ duplex-link $W(1) $BS(2) 5Mb 3ms DropTail

$ns\_ duplex-link-op $W(0) $W(1) orient down

$ns\_ duplex-link-op $W(1) $BS(0) orient left-down

$ns\_ duplex-link-op $W(1) $BS(1) orient down

$ns\_ duplex-link-op $W(1) $BS(2) orient right-down

# setup TCP connections

set tcp1 [new Agent/TCP]

$tcp1 set class\_ 2

set sink1 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(0) $tcp1

$ns\_ attach-agent $W(0) $sink1

$ns\_ connect $tcp1 $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ns\_ at $opt(ftp1-start) "$ftp1 start"

set tcp2 [new Agent/TCP]

$tcp2 set class\_ 2

set sink2 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(4) $tcp2

$ns\_ attach-agent $node\_(2) $sink2

$ns\_ connect $tcp2 $sink2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns\_ at $opt(ftp2-start) "$ftp2 start"

for {set j 0} {$j <$opt(nn)} {incr j} {

$ns\_ at $opt(stop).0 "$node\_($j) reset";

}

# Tell all nodes when the simulation ends

$ns\_ at $opt(stop).0 "$BS(0) reset";

$ns\_ at $opt(stop).0 "$BS(1) reset";

$ns\_ at $opt(stop).0 "$BS(2) reset";

$ns\_ at 0.55 "record"

$ns\_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns\_ halt"

$ns\_ at $opt(stop).0001 "finish"

proc plotWindow {tcpSource outfile} {

global ns\_

set now [$ns\_ now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns\_ at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Tahoe\_wandw1.xg w]

set outfile2 [open Tahoe\_wandw2.xg w]

$ns\_ at 0.0 "plotWindow $tcp1 $outfile1"

$ns\_ at 0.0 "plotWindow $tcp2 $outfile2"

$ns\_ run

Reno\_wandw.tcl:

set opt(chan) Channel/WirelessChannel ;# channel type

set opt(prop) Propagation/TwoRayGround ;# radio-propagation model

set opt(netif) Phy/WirelessPhy ;# network interface type

set opt(mac) Mac/802\_11 ;# MAC type

set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type

set opt(ll) LL ;# link layer type

set opt(ant) Antenna/OmniAntenna ;# antenna model

set opt(ifqlen) 50 ;# max packet in ifq

set opt(nn) 6 ;# number of mobilenodes

set opt(adhocRouting) DSDV ;# routing protocol

#set opt(cp) "tcpgen.tcl" ;# connection pattern file

#set opt(sc) "scene-7" ;# node movement file.

set opt(x) 670 ;# x coordinate of topology

set opt(y) 670 ;# y coordinate of topology

set opt(seed) 0.0 ;# seed for random number gen.

set opt(stop) 250 ;# time to stop simulation

set opt(ftp1-start) 160.0

set opt(ftp2-start) 170.0

set num\_wired\_nodes 2

set num\_bs\_nodes 3

#11 nodes.

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

# check for boundary parameters and random seed

if { $opt(x) == 0 || $opt(y) == 0 } {

puts "No X-Y boundary values given for wireless topology\n"

}

if {$opt(seed) > 0} {

puts "Seeding Random number generator with $opt(seed)\n"

ns-random $opt(seed)

}

# create simulator instance

set ns\_ [new Simulator]

# set up for hierarchical routing

$ns\_ node-config -addressType hierarchical

AddrParams set domain\_num\_ 2 ;# number of domains

lappend cluster\_num 2 3 ;# number of clusters in each domain

#cluster 0, cluster 1, cluster 2 and cluster 3

AddrParams set cluster\_num\_ $cluster\_num

lappend eilastlevel 1 1 4 2 3 ;# number of nodes in each cluster

AddrParams set nodes\_num\_ $eilastlevel ;# of each domain

set tracefd [open Reno\_wandw.tr w]

set namtrace [open Reno\_wandw.nam w]

$ns\_ trace-all $tracefd

$ns\_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

set fa [open "tt\_Reno\_wandw.xg" "w"]

set f1 [open "tb\_Reno\_wandw.xg" "w"]

proc finish {} {

global ns\_ tracefd namtrace

$ns\_ flush-trace

close $tracefd

close $namtrace

exec xgraph tt\_Reno\_wandw.xg tb\_Reno\_wandw.xg Reno\_wandw1.xg Reno\_wandw2.xg -geometry 300x300 &

exec nam Reno\_wandw.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

# Create topography object

set topo [new Topography]

# define topology

$topo load\_flatgrid $opt(x) $opt(y)

# create God

create-god [expr $opt(nn) + $num\_bs\_nodes]

# 9 wireless nodes, GOD object will take care.

#create wired nodes

set temp {0.0.0 0.1.0} ;# hierarchical addresses for wired domain

for {set i 0} {$i < $num\_wired\_nodes} {incr i} {

set W($i) [$ns\_ node [lindex $temp $i]]

}

# configure for base-station node

$ns\_ node-config -adhocRouting $opt(adhocRouting) \

-llType $opt(ll) \

-macType $opt(mac) \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-channelType $opt(chan) \

-topoInstance $topo \

-wiredRouting ON \

-agentTrace ON \

-routerTrace OFF \

-macTrace OFF

#create base-station node

set temp {1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 1.1.0 1.1.1 1.1.2 1.1.3 } ;# hier address to be used for wireless

;# domain

set BS(0) [$ns\_ node 1.0.0]

$BS(0) random-motion 0 ;# disable random motion

set BS(1) [$ns\_ node 1.1.0]

$BS(1) random-motion 0

set BS(2) [$ns\_ node 1.2.0]

$BS(2) random-motion 0

#provide some co-ord (fixed) to base station node

$BS(0) set X\_ 200.0

$BS(0) set Y\_ 200.0

$BS(0) set Z\_ 0.0

$BS(1) set X\_ 250.0

$BS(1) set Y\_ 250.0

$BS(1) set Z\_ 0.0

$BS(2) set X\_ 100.0

$BS(2) set Y\_ 180.0

$BS(2) set Z\_ 0.0

# create mobilenodes in the same domain as BS(0)

# note the position and movement of mobilenodes is as defined

# in $opt(sc)

#configure for mobilenodes

$ns\_ node-config -wiredRouting OFF

# attaching the mobile nodes to Base Station 0

set node\_(0) [$ns\_ node 1.0.1]

$node\_(0) base-station [AddrParams addr2id [$BS(0) node-addr]]

#attaching mobile nodes to Base station 1

set node\_(1) [$ns\_ node 1.1.1]

$node\_(1) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(2) [$ns\_ node 1.1.2]

$node\_(2) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(3) [$ns\_ node 1.1.3]

$node\_(3) base-station [AddrParams addr2id [$BS(1) node-addr]]

#attaching mobile nodes to Base station 2

set node\_(4) [$ns\_ node 1.2.1]

$node\_(4) base-station [AddrParams addr2id [$BS(2) node-addr]]

set node\_(5) [$ns\_ node 1.2.2]

$node\_(5) base-station [AddrParams addr2id [$BS(2) node-addr]]

for {set j 0} {$j < 6} {incr j} {

$ns\_ initial\_node\_pos $node\_($j) 40

}

$node\_(0) set X\_ 433.194578912685

$node\_(0) set Y\_ 86.805779539422

$node\_(0) set Z\_ 0.000000000000

$node\_(1) set X\_ 182.146343432004

$node\_(1) set Y\_ 589.457950263110

$node\_(1) set Z\_ 0.000000000000

$node\_(2) set X\_ 636.712900082647

$node\_(2) set Y\_ 227.793717407876

$node\_(2) set Z\_ 0.000000000000

$node\_(3) set X\_ 355.730559281529

$node\_(3) set Y\_ 636.858769794140

$node\_(3) set Z\_ 0.000000000000

$node\_(4) set X\_ 412.570805501828

$node\_(4) set Y\_ 140.390797313602

$node\_(4) set Z\_ 0.000000000000

$node\_(5) set X\_ 192.498199648602

$node\_(5) set Y\_ 389.172056773162

$node\_(5) set Z\_ 0.000000000000

$ns\_ at 0.000000000000 "$node\_(0) setdest 533.982439606531 316.260671641729 10.420081258606"

$ns\_ at 0.000000000000 "$node\_(1) setdest 161.810128580897 443.165191912176 10.922787425094"

$ns\_ at 0.000000000000 "$node\_(2) setdest 639.905294214911 171.358564559506 15.318998910103"

$ns\_ at 0.000000000000 "$node\_(3) setdest 542.673393030666 413.384013118659 12.552371653936"

$ns\_ at 0.000000000000 "$node\_(4) setdest 498.263533357255 244.647811683068 14.896733048027"

$ns\_ at 0.000000000000 "$node\_(5) setdest 597.554489327905 123.928273538379 16.983474910390"

#create links between wired and BS nodes

$ns\_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail

$ns\_ duplex-link $W(1) $BS(0) 5Mb 0.2ms DropTail

$ns\_ duplex-link $W(1) $BS(1) 15Mb 3ms DropTail

$ns\_ duplex-link $W(1) $BS(2) 5Mb 3ms DropTail

$ns\_ duplex-link-op $W(0) $W(1) orient down

$ns\_ duplex-link-op $W(1) $BS(0) orient left-down

$ns\_ duplex-link-op $W(1) $BS(1) orient down

$ns\_ duplex-link-op $W(1) $BS(2) orient right-down

# setup TCP connections

set tcp1 [new Agent/TCP/Reno]

$tcp1 set class\_ 2

set sink1 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(0) $tcp1

$ns\_ attach-agent $W(0) $sink1

$ns\_ connect $tcp1 $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ns\_ at $opt(ftp1-start) "$ftp1 start"

set tcp2 [new Agent/TCP/Reno]

$tcp2 set class\_ 2

set sink2 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(4) $tcp2

$ns\_ attach-agent $node\_(2) $sink2

$ns\_ connect $tcp2 $sink2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns\_ at $opt(ftp2-start) "$ftp2 start"

for {set j 0} {$j <$opt(nn)} {incr j} {

$ns\_ at $opt(stop).0 "$node\_($j) reset";

}

# Tell all nodes when the simulation ends

$ns\_ at $opt(stop).0 "$BS(0) reset";

$ns\_ at $opt(stop).0 "$BS(1) reset";

$ns\_ at $opt(stop).0 "$BS(2) reset";

$ns\_ at 0.55 "record"

$ns\_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns\_ halt"

$ns\_ at $opt(stop).0001 "finish"

proc plotWindow {tcpSource outfile} {

global ns\_

set now [$ns\_ now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns\_ at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Reno\_wandw1.xg w]

set outfile2 [open Reno\_wand2.xg w]

$ns\_ at 0.0 "plotWindow $tcp1 $outfile1"

$ns\_ at 0.0 "plotWindow $tcp2 $outfile2"

$ns\_ run

Vegas\_wandw.tcl:

set opt(chan) Channel/WirelessChannel ;# channel type

set opt(prop) Propagation/TwoRayGround ;# radio-propagation model

set opt(netif) Phy/WirelessPhy ;# network interface type

set opt(mac) Mac/802\_11 ;# MAC type

set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type

set opt(ll) LL ;# link layer type

set opt(ant) Antenna/OmniAntenna ;# antenna model

set opt(ifqlen) 50 ;# max packet in ifq

set opt(nn) 6 ;# number of mobilenodes

set opt(adhocRouting) DSDV ;# routing protocol

#set opt(cp) "tcpgen.tcl" ;# connection pattern file

#set opt(sc) "scene-7" ;# node movement file.

set opt(x) 670 ;# x coordinate of topology

set opt(y) 670 ;# y coordinate of topology

set opt(seed) 0.0 ;# seed for random number gen.

set opt(stop) 250 ;# time to stop simulation

set opt(ftp1-start) 160.0

set opt(ftp2-start) 170.0

set num\_wired\_nodes 2

set num\_bs\_nodes 3

#11 nodes.

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

# check for boundary parameters and random seed

if { $opt(x) == 0 || $opt(y) == 0 } {

puts "No X-Y boundary values given for wireless topology\n"

}

if {$opt(seed) > 0} {

puts "Seeding Random number generator with $opt(seed)\n"

ns-random $opt(seed)

}

# create simulator instance

set ns\_ [new Simulator]

# set up for hierarchical routing

$ns\_ node-config -addressType hierarchical

AddrParams set domain\_num\_ 2 ;# number of domains

lappend cluster\_num 2 3 ;# number of clusters in each domain

#cluster 0, cluster 1, cluster 2 and cluster 3

AddrParams set cluster\_num\_ $cluster\_num

lappend eilastlevel 1 1 4 2 3 ;# number of nodes in each cluster

AddrParams set nodes\_num\_ $eilastlevel ;# of each domain

set tracefd [open Vegas\_wandw.tr w]

set namtrace [open Vegas\_wandw.nam w]

$ns\_ trace-all $tracefd

$ns\_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

set fa [open "tt\_Vegas\_wandw.xg" "w"]

set f1 [open "tb\_Vegas\_wandw.xg" "w"]

proc finish {} {

global ns\_ tracefd namtrace

$ns\_ flush-trace

close $tracefd

close $namtrace

exec xgraph tt\_Vegas\_wandw.xg tb\_Vegas\_wandw.xg Vegas\_wandw1.xg Vegas\_wandw2.xg -geometry 300x300 &

exec nam Tahoe\_wandw.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

# Create topography object

set topo [new Topography]

# define topology

$topo load\_flatgrid $opt(x) $opt(y)

# create God

create-god [expr $opt(nn) + $num\_bs\_nodes]

# 9 wireless nodes, GOD object will take care.

#create wired nodes

set temp {0.0.0 0.1.0} ;# hierarchical addresses for wired domain

for {set i 0} {$i < $num\_wired\_nodes} {incr i} {

set W($i) [$ns\_ node [lindex $temp $i]]

}

# configure for base-station node

$ns\_ node-config -adhocRouting $opt(adhocRouting) \

-llType $opt(ll) \

-macType $opt(mac) \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-channelType $opt(chan) \

-topoInstance $topo \

-wiredRouting ON \

-agentTrace ON \

-routerTrace OFF \

-macTrace OFF

#create base-station node

set temp {1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 1.1.0 1.1.1 1.1.2 1.1.3 } ;# hier address to be used for wireless

;# domain

set BS(0) [$ns\_ node 1.0.0]

$BS(0) random-motion 0 ;# disable random motion

set BS(1) [$ns\_ node 1.1.0]

$BS(1) random-motion 0

set BS(2) [$ns\_ node 1.2.0]

$BS(2) random-motion 0

#provide some co-ord (fixed) to base station node

$BS(0) set X\_ 200.0

$BS(0) set Y\_ 200.0

$BS(0) set Z\_ 0.0

$BS(1) set X\_ 250.0

$BS(1) set Y\_ 250.0

$BS(1) set Z\_ 0.0

$BS(2) set X\_ 100.0

$BS(2) set Y\_ 180.0

$BS(2) set Z\_ 0.0

# create mobilenodes in the same domain as BS(0)

# note the position and movement of mobilenodes is as defined

# in $opt(sc)

#configure for mobilenodes

$ns\_ node-config -wiredRouting OFF

# attaching the mobile nodes to Base Station 0

set node\_(0) [$ns\_ node 1.0.1]

$node\_(0) base-station [AddrParams addr2id [$BS(0) node-addr]]

#attaching mobile nodes to Base station 1

set node\_(1) [$ns\_ node 1.1.1]

$node\_(1) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(2) [$ns\_ node 1.1.2]

$node\_(2) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(3) [$ns\_ node 1.1.3]

$node\_(3) base-station [AddrParams addr2id [$BS(1) node-addr]]

#attaching mobile nodes to Base station 2

set node\_(4) [$ns\_ node 1.2.1]

$node\_(4) base-station [AddrParams addr2id [$BS(2) node-addr]]

set node\_(5) [$ns\_ node 1.2.2]

$node\_(5) base-station [AddrParams addr2id [$BS(2) node-addr]]

for {set j 0} {$j < 6} {incr j} {

$ns\_ initial\_node\_pos $node\_($j) 40

}

$node\_(0) set X\_ 433.194578912685

$node\_(0) set Y\_ 86.805779539422

$node\_(0) set Z\_ 0.000000000000

$node\_(1) set X\_ 182.146343432004

$node\_(1) set Y\_ 589.457950263110

$node\_(1) set Z\_ 0.000000000000

$node\_(2) set X\_ 636.712900082647

$node\_(2) set Y\_ 227.793717407876

$node\_(2) set Z\_ 0.000000000000

$node\_(3) set X\_ 355.730559281529

$node\_(3) set Y\_ 636.858769794140

$node\_(3) set Z\_ 0.000000000000

$node\_(4) set X\_ 412.570805501828

$node\_(4) set Y\_ 140.390797313602

$node\_(4) set Z\_ 0.000000000000

$node\_(5) set X\_ 192.498199648602

$node\_(5) set Y\_ 389.172056773162

$node\_(5) set Z\_ 0.000000000000

$ns\_ at 0.000000000000 "$node\_(0) setdest 533.982439606531 316.260671641729 10.420081258606"

$ns\_ at 0.000000000000 "$node\_(1) setdest 161.810128580897 443.165191912176 10.922787425094"

$ns\_ at 0.000000000000 "$node\_(2) setdest 639.905294214911 171.358564559506 15.318998910103"

$ns\_ at 0.000000000000 "$node\_(3) setdest 542.673393030666 413.384013118659 12.552371653936"

$ns\_ at 0.000000000000 "$node\_(4) setdest 498.263533357255 244.647811683068 14.896733048027"

$ns\_ at 0.000000000000 "$node\_(5) setdest 597.554489327905 123.928273538379 16.983474910390"

#create links between wired and BS nodes

$ns\_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail

$ns\_ duplex-link $W(1) $BS(0) 5Mb 0.2ms DropTail

$ns\_ duplex-link $W(1) $BS(1) 15Mb 3ms DropTail

$ns\_ duplex-link $W(1) $BS(2) 5Mb 3ms DropTail

$ns\_ duplex-link-op $W(0) $W(1) orient down

$ns\_ duplex-link-op $W(1) $BS(0) orient left-down

$ns\_ duplex-link-op $W(1) $BS(1) orient down

$ns\_ duplex-link-op $W(1) $BS(2) orient right-down

# setup TCP connections

set tcp1 [new Agent/TCP/Vegas]

$tcp1 set class\_ 2

set sink1 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(0) $tcp1

$ns\_ attach-agent $W(0) $sink1

$ns\_ connect $tcp1 $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ns\_ at $opt(ftp1-start) "$ftp1 start"

set tcp2 [new Agent/TCP/Vegas]

$tcp2 set class\_ 2

set sink2 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(4) $tcp2

$ns\_ attach-agent $node\_(2) $sink2

$ns\_ connect $tcp2 $sink2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns\_ at $opt(ftp2-start) "$ftp2 start"

for {set j 0} {$j <$opt(nn)} {incr j} {

$ns\_ at $opt(stop).0 "$node\_($j) reset";

}

# Tell all nodes when the simulation ends

$ns\_ at $opt(stop).0 "$BS(0) reset";

$ns\_ at $opt(stop).0 "$BS(1) reset";

$ns\_ at $opt(stop).0 "$BS(2) reset";

$ns\_ at 0.55 "record"

$ns\_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns\_ halt"

$ns\_ at $opt(stop).0001 "finish"

proc plotWindow {tcpSource outfile} {

global ns\_

set now [$ns\_ now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns\_ at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open Vegas\_wandw1.xg w]

set outfile2 [open Vegas\_wand2.xg w]

$ns\_ at 0.0 "plotWindow $tcp1 $outfile1"

$ns\_ at 0.0 "plotWindow $tcp2 $outfile2"

$ns\_ run

Tcp.cc:

double TcpAgent::**my\_limited\_slow\_start**(double cwnd, int max\_ssthresh, double increment)

{

if (max\_ssthresh <= 0) {

return increment;

} else {

double increment1 = 0.0;

int round = int(cwnd / (double(max\_ssthresh)/6.0));

if (round > 0) {

increment1 = 1.0/double(round);

}

if (increment < increment1) {

return increment1;

} else {

return increment;

}

}

}

void TcpAgent::opencwnd()

{

double increment;

if (cwnd\_ < ssthresh\_) {

/\* slow-start (exponential) \*/

cwnd\_ += 1;

} else {

/\* linear \*/

double f;

switch (wnd\_option\_) {

case 0:

if (++count\_ >= cwnd\_) {

count\_ = 0;

++cwnd\_;

}

break;

case 1:

/\* This is the standard algorithm. \*/

increment = increase\_num\_ / cwnd\_;

if ((last\_cwnd\_action\_ == 0 ||

last\_cwnd\_action\_ == CWND\_ACTION\_TIMEOUT)

&& max\_ssthresh\_ > 0) {

increment = limited\_slow\_start(cwnd\_,

max\_ssthresh\_, increment);

}

cwnd\_ += increment;

break;

case 2:

/\* These are window increase algorithms

\* for experimental purposes only. \*/

/\* This is the Constant-Rate increase algorithm

\* from the 1991 paper by S. Floyd on "Connections

\* with Multiple Congested Gateways".

\* The window is increased by roughly

\* wnd\_const\_\*RTT^2 packets per round-trip time. \*/

f = (t\_srtt\_ >> T\_SRTT\_BITS) \* tcp\_tick\_;

f \*= f;

f \*= wnd\_const\_;

/\* f = wnd\_const\_ \* RTT^2 \*/

f += fcnt\_;

if (f > cwnd\_) {

fcnt\_ = 0;

++cwnd\_;

} else

fcnt\_ = f;

break;

case 3:

/\* The window is increased by roughly

\* awnd\_^2 \* wnd\_const\_ packets per RTT,

\* for awnd\_ the average congestion window. \*/

f = awnd\_;

f \*= f;

f \*= wnd\_const\_;

f += fcnt\_;

if (f > cwnd\_) {

fcnt\_ = 0;

++cwnd\_;

} else

fcnt\_ = f;

break;

case 4:

/\* The window is increased by roughly

\* awnd\_ \* wnd\_const\_ packets per RTT,

\* for awnd\_ the average congestion window. \*/

f = awnd\_;

f \*= wnd\_const\_;

f += fcnt\_;

if (f > cwnd\_) {

fcnt\_ = 0;

++cwnd\_;

} else

fcnt\_ = f;

break;

case 5:

/\* The window is increased by roughly wnd\_const\_\*RTT

\* packets per round-trip time, as discussed in

\* the 1992 paper by S. Floyd on "On Traffic

\* Phase Effects in Packet-Switched Gateways". \*/

f = (t\_srtt\_ >> T\_SRTT\_BITS) \* tcp\_tick\_;

f \*= wnd\_const\_;

f += fcnt\_;

if (f > cwnd\_) {

fcnt\_ = 0;

++cwnd\_;

} else

fcnt\_ = f;

break;

case 6:

/\* binomial controls \*/

cwnd\_ += increase\_num\_ / (cwnd\_\*pow(cwnd\_,k\_parameter\_));

break;

case 8:

/\* high-speed TCP, RFC 3649 \*/

increment = increase\_param();

if ((last\_cwnd\_action\_ == 0 ||

last\_cwnd\_action\_ == CWND\_ACTION\_TIMEOUT)

&& max\_ssthresh\_ > 0) {

increment = limited\_slow\_start(cwnd\_,

max\_ssthresh\_, increment);

}

cwnd\_ += increment;

break;

**case 25:**

increment = t\_srtt\_ / cwnd\_;

if ((last\_cwnd\_action\_ == 0 ||

last\_cwnd\_action\_ == CWND\_ACTION\_TIMEOUT)

&& max\_ssthresh\_ > 0) {

increment = limited\_slow\_start(cwnd\_,

max\_ssthresh\_, increment);

}

cwnd\_ += increment;

break;

default:

#ifdef notdef

/XXX/

error("illegal window option %d", wnd\_option\_);

#endif

abort();

}

}

// if maxcwnd\_ is set (nonzero), make it the cwnd limit

if (maxcwnd\_ && (int(cwnd\_) > maxcwnd\_))

cwnd\_ = maxcwnd\_;

return;

}

my\_algo\_grid.tcl:

set ns [new Simulator]

set nf [open my\_algo\_grid.nam w]

$ns namtrace-all $nf

set nftr [open my\_algo\_grid.tr w]

$ns trace-all $nftr

set fa [open "tt\_my\_algo\_grid.xg" "w"]

set f1 [open "tb\_my\_algo\_grid.xg" "w"]

proc finish {} {

global ns nf nftr fa f1

$ns flush-trace

close $nf

close $fa

close $f1

exec xgraph tt\_my\_algo\_grid.xg tb\_my\_algo\_grid.xg my\_algo\_grid1.xg my\_algo\_grid2.xg my\_algo\_grid3.xg -geometry 300x300 &

exec nam Tahoe\_grid.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 ftp3 fa f1

global sink1 sink2 sink3

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set bw2 [$sink3 set bytes\_]

set totbw [expr $bw0 + $bw1 + $bw2]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$sink3 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set rows 3

set cols 3

set start [lindex $argv 2]

set stop [lindex $argv 3]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 50ms DropTail

$ns duplex-link $n1 $n4 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 50ms DropTail

$ns duplex-link $n2 $n5 5Mb 10ms DropTail

$ns duplex-link $n3 $n4 2Mb 10ms DropTail

$ns duplex-link $n4 $n5 5Mb 40ms DropTail

$ns duplex-link $n3 $n6 2Mb 10ms DropTail

$ns duplex-link $n4 $n7 3Mb 100ms DropTail

$ns duplex-link $n5 $n8 0.3Mb 200ms DropTail

$ns duplex-link $n6 $n7 0.3Mb 200ms DropTail

$ns duplex-link $n7 $n8 3Mb 200ms DropTail

# Sending node is 0 with agent as Tahoe Agent

set tcp1 [new Agent/TCP]

set tcp2 [new Agent/TCP]

set tcp3 [new Agent/TCP]

$ns attach-agent $n0 $tcp1

$ns attach-agent $n1 $tcp2

$ns attach-agent $n1 $tcp3

# receiving (sink) node is n4

set sink1 [new Agent/TCPSink]

set sink2 [new Agent/TCPSink]

set sink3 [new Agent/TCPSink]

$ns attach-agent $n6 $sink1

$ns attach-agent $n2 $sink2

$ns attach-agent $n7 $sink3

# establish the traffic between the source and sink

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns connect $tcp3 $sink3

#setting window option to my new case 25.

$tcp1 set windowOption\_ 25

$tcp2 set windowOption\_ 25

$tcp3 set windowOption\_ 25

# Setup a FTP traffic generator on "tcp1"

set ftp1 [new Application/FTP]

set ftp2 [new Application/FTP]

set ftp3 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ftp2 attach-agent $tcp2

$ftp3 attach-agent $tcp3

$ftp1 set type\_ FTP

$ftp2 set type\_ FTP

$ftp3 set type\_ FTP

# start/stop the traffic

$ns at 0.1 "$ftp1 start"

$ns at 0.55 "record"

$ns at 5 "$ftp2 start"

$ns at 5 "$ftp3 start"

$ns at 40.0 "$ftp1 finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg (this can be plotted # using xgraph or gnuplot. this example uses xgraph to plot the cwnd\_

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open my\_algo\_grid1.xg w]

set outfile2 [open my\_algo\_grid2.xg w]

set outfile3 [open my\_algo\_grid3.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns at 0.0 "plotWindow $tcp3 $outfile3"

$ns run

My\_algo\_wireless.tcl:

set val(chan) Channel/WirelessChannel

set val(prop) Propagation/TwoRayGround

set val(netif) Phy/WirelessPhy

set val(mac) Mac/802\_11

set val(ifq) Queue/DropTail/PriQueue

set val(ll) LL

set val(ant) Antenna/OmniAntenna

set val(ifqlen) 50

set val(nn) 10

set val(rp) AODV

set val(x) 500

set val(y) 500

set ns [new Simulator]

set tracefile [open my\_algo\_wireless.tr w]

$ns trace-all $tracefile

set namfile [open my\_algo\_wireless.nam w]

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

set fa [open "tt\_my\_algo\_wireless.xg" "w"]

set f1 [open "tb\_my\_algo\_wireless.xg" "w"]

proc finish {} {

global ns tracefile namfile fa f1

$ns flush-trace

close $tracefile

close $namfile

close $fa

close $f1

exec xgraph tt\_my\_algo\_wireless.xg tb\_my\_algo\_wireless.xg my\_algo\_wireless1.xg my\_algo\_wireless2.xg -geometry 300x300 &

exec nam Tahoe\_wireless.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

create-god $val(nn)

$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) \

-topoInstance $topo \

-agentTrace ON \

-macTrace ON \

-routerTrace ON \

-movementTrace ON \

-channel [new $val(chan)]

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

set n($i) [$ns node]

$n($i) random-motion 0

$ns initial\_node\_pos $n($i) 50

}

$n(0) set X\_ -110.0

$n(0) set Y\_ 350.0

$n(0) set Z\_ 0.0

$n(1) set X\_ -10.0

$n(1) set Y\_ 350.0

$n(1) set Z\_ 0.0

$n(2) set X\_ 50.0

$n(2) set Y\_ 450.0

$n(2) set Z\_ 0.0

$n(3) set X\_ -120.0

$n(3) set Y\_ 100.0

$n(3) set Z\_ 0.0

$n(4) set X\_ 90.0

$n(4) set Y\_ 100.0

$n(4) set Z\_ 0.0

$n(5) set X\_ -200.0

$n(5) set Y\_ 200.0

$n(5) set Z\_ 0.0

$n(6) set X\_ 90.0

$n(6) set Y\_ 200.0

$n(6) set Z\_ 0.0

$n(7) set X\_ 400.0

$n(7) set Y\_ 100.0

$n(7) set Z\_ 0.0

$n(8) set X\_ -410.0

$n(8) set Y\_ 320.0

$n(8) set Z\_ 0.0

$n(9) set X\_ 280.0

$n(9) set Y\_ -140.0

$n(9) set Z\_ 0.0

$ns at 1.0 "$n(0) setdest 70.0 50.0 0.0"

$ns at 1.0 "$n(1) setdest 20.0 30.0 0.0"

$ns at 1.0 "$n(2) setdest 200.0 450.0 85.0"

$ns at 1.0 "$n(3) setdest 40.0 100.0 0.0"

$ns at 1.0 "$n(4) setdest 30.0 45.0 0.0"

$ns at 1.0 "$n(5) setdest 80.0 200.0 0.0"

$ns at 1.0 "$n(6) setdest 50.0 350.0 0.0"

$ns at 1.0 "$n(7) setdest 300.0 145.0 0.0"

$ns at 1.0 "$n(8) setdest 110.0 230.0 0.0"

$ns at 1.0 "$n(9) setdest 250.0 50.0 0.0"

$n(0) color "red"

$ns at 1.0 "$n(0) color red"

$n(5) color "red"

$ns at 1.0 "$n(5) color red"

$n(2) color "darkgreen"

$ns at 1.0 "$n(2) color darkgreen"

set tcp1 [new Agent/TCP]

set sink1 [new Agent/TCPSink]

$ns attach-agent $n(0) $tcp1

$ns attach-agent $n(5) $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

set tcp2 [new Agent/TCP]

set sink2 [new Agent/TCPSink]

$ns attach-agent $n(2) $tcp2

$ns attach-agent $n(6) $sink2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns at 0.55 "record"

$ns at 1.0 "$ftp1 start"

$ns at 1.0 "$ftp2 start"

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$tcp1 set windowOption\_ 25

$tcp2 set windowOption\_ 25

$ns at 14.0 "$ftp1 stop"

$ns at 10.0 "$ftp2 stop"

$ns at 15.0 "finish"

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

puts $outfile "$now $cwnd"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open my\_algo\_wireless1.xg w]

set outfile2 [open my\_algo\_wireless2.xg w]

$ns at 0.0 "plotWindow $tcp1 $outfile1"

$ns at 0.0 "plotWindow $tcp2 $outfile2"

$ns run

My\_algo\_wandw.tcl:

set opt(chan) Channel/WirelessChannel ;# channel type

set opt(prop) Propagation/TwoRayGround ;# radio-propagation model

set opt(netif) Phy/WirelessPhy ;# network interface type

set opt(mac) Mac/802\_11 ;# MAC type

set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type

set opt(ll) LL ;# link layer type

set opt(ant) Antenna/OmniAntenna ;# antenna model

set opt(ifqlen) 50 ;# max packet in ifq

set opt(nn) 6 ;# number of mobilenodes

set opt(adhocRouting) DSDV ;# routing protocol

#set opt(cp) "tcpgen.tcl" ;# connection pattern file

#set opt(sc) "scene-7" ;# node movement file.

set opt(x) 670 ;# x coordinate of topology

set opt(y) 670 ;# y coordinate of topology

set opt(seed) 0.0 ;# seed for random number gen.

set opt(stop) 250 ;# time to stop simulation

set opt(ftp1-start) 160.0

set opt(ftp2-start) 170.0

set num\_wired\_nodes 2

set num\_bs\_nodes 3

#11 nodes.

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

# check for boundary parameters and random seed

if { $opt(x) == 0 || $opt(y) == 0 } {

puts "No X-Y boundary values given for wireless topology\n"

}

if {$opt(seed) > 0} {

puts "Seeding Random number generator with $opt(seed)\n"

ns-random $opt(seed)

}

# create simulator instance

set ns\_ [new Simulator]

# set up for hierarchical routing

$ns\_ node-config -addressType hierarchical

AddrParams set domain\_num\_ 2 ;# number of domains

lappend cluster\_num 2 3 ;# number of clusters in each domain

#cluster 0, cluster 1, cluster 2 and cluster 3

AddrParams set cluster\_num\_ $cluster\_num

lappend eilastlevel 1 1 4 2 3 ;# number of nodes in each cluster

AddrParams set nodes\_num\_ $eilastlevel ;# of each domain

set tracefd [open my\_algo\_wandw.tr w]

set namtrace [open my\_algo\_wandw.nam w]

$ns\_ trace-all $tracefd

$ns\_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

set fa [open "tt\_my\_algo\_wandw.xg" "w"]

set f1 [open "tb\_my\_algo\_wandw.xg" "w"]

proc finish {} {

global ns\_ tracefd namtrace

$ns\_ flush-trace

close $tracefd

close $namtrace

exec xgraph tt\_my\_algo\_wandw.xg tb\_my\_algo\_wandw.xg my\_algo\_wandw1.xg my\_algo\_wandw2.xg -geometry 300x300 &

exec nam Tahoe\_wandw.nam &

exit 0

}

proc record {} {

global ftp1 ftp2 fa f1

global sink1 sink2

set ns [Simulator instance]

set time 0.1

set now [$ns now]

set bw0 [$sink1 set bytes\_]

set bw1 [$sink2 set bytes\_]

set totbw [expr $bw0 + $bw1]

puts $fa "$now [expr $totbw/$time\*8/1000000]"

puts $f1 "$now [expr $totbw]"

$sink1 set bytes\_ 0

$sink2 set bytes\_ 0

$ns at [expr $now+$time] "record"

}

# Create topography object

set topo [new Topography]

# define topology

$topo load\_flatgrid $opt(x) $opt(y)

# create God

create-god [expr $opt(nn) + $num\_bs\_nodes]

# 9 wireless nodes, GOD object will take care.

#create wired nodes

set temp {0.0.0 0.1.0} ;# hierarchical addresses for wired domain

for {set i 0} {$i < $num\_wired\_nodes} {incr i} {

set W($i) [$ns\_ node [lindex $temp $i]]

}

# configure for base-station node

$ns\_ node-config -adhocRouting $opt(adhocRouting) \

-llType $opt(ll) \

-macType $opt(mac) \

-ifqType $opt(ifq) \

-ifqLen $opt(ifqlen) \

-antType $opt(ant) \

-propType $opt(prop) \

-phyType $opt(netif) \

-channelType $opt(chan) \

-topoInstance $topo \

-wiredRouting ON \

-agentTrace ON \

-routerTrace OFF \

-macTrace OFF

#create base-station node

set temp {1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 1.1.0 1.1.1 1.1.2 1.1.3 } ;# hier address to be used for wireless

;# domain

set BS(0) [$ns\_ node 1.0.0]

$BS(0) random-motion 0 ;# disable random motion

set BS(1) [$ns\_ node 1.1.0]

$BS(1) random-motion 0

set BS(2) [$ns\_ node 1.2.0]

$BS(2) random-motion 0

#provide some co-ord (fixed) to base station node

$BS(0) set X\_ 200.0

$BS(0) set Y\_ 200.0

$BS(0) set Z\_ 0.0

$BS(1) set X\_ 250.0

$BS(1) set Y\_ 250.0

$BS(1) set Z\_ 0.0

$BS(2) set X\_ 100.0

$BS(2) set Y\_ 180.0

$BS(2) set Z\_ 0.0

# create mobilenodes in the same domain as BS(0)

# note the position and movement of mobilenodes is as defined

# in $opt(sc)

#configure for mobilenodes

$ns\_ node-config -wiredRouting OFF

# attaching the mobile nodes to Base Station 0

set node\_(0) [$ns\_ node 1.0.1]

$node\_(0) base-station [AddrParams addr2id [$BS(0) node-addr]]

#attaching mobile nodes to Base station 1

set node\_(1) [$ns\_ node 1.1.1]

$node\_(1) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(2) [$ns\_ node 1.1.2]

$node\_(2) base-station [AddrParams addr2id [$BS(1) node-addr]]

set node\_(3) [$ns\_ node 1.1.3]

$node\_(3) base-station [AddrParams addr2id [$BS(1) node-addr]]

#attaching mobile nodes to Base station 2

set node\_(4) [$ns\_ node 1.2.1]

$node\_(4) base-station [AddrParams addr2id [$BS(2) node-addr]]

set node\_(5) [$ns\_ node 1.2.2]

$node\_(5) base-station [AddrParams addr2id [$BS(2) node-addr]]

for {set j 0} {$j < 6} {incr j} {

$ns\_ initial\_node\_pos $node\_($j) 40

}

$node\_(0) set X\_ 433.194578912685

$node\_(0) set Y\_ 86.805779539422

$node\_(0) set Z\_ 0.000000000000

$node\_(1) set X\_ 182.146343432004

$node\_(1) set Y\_ 589.457950263110

$node\_(1) set Z\_ 0.000000000000

$node\_(2) set X\_ 636.712900082647

$node\_(2) set Y\_ 227.793717407876

$node\_(2) set Z\_ 0.000000000000

$node\_(3) set X\_ 355.730559281529

$node\_(3) set Y\_ 636.858769794140

$node\_(3) set Z\_ 0.000000000000

$node\_(4) set X\_ 412.570805501828

$node\_(4) set Y\_ 140.390797313602

$node\_(4) set Z\_ 0.000000000000

$node\_(5) set X\_ 192.498199648602

$node\_(5) set Y\_ 389.172056773162

$node\_(5) set Z\_ 0.000000000000

$ns\_ at 0.000000000000 "$node\_(0) setdest 533.982439606531 316.260671641729 10.420081258606"

$ns\_ at 0.000000000000 "$node\_(1) setdest 161.810128580897 443.165191912176 10.922787425094"

$ns\_ at 0.000000000000 "$node\_(2) setdest 639.905294214911 171.358564559506 15.318998910103"

$ns\_ at 0.000000000000 "$node\_(3) setdest 542.673393030666 413.384013118659 12.552371653936"

$ns\_ at 0.000000000000 "$node\_(4) setdest 498.263533357255 244.647811683068 14.896733048027"

$ns\_ at 0.000000000000 "$node\_(5) setdest 597.554489327905 123.928273538379 16.983474910390"

#create links between wired and BS nodes

$ns\_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail

$ns\_ duplex-link $W(1) $BS(0) 5Mb 0.2ms DropTail

$ns\_ duplex-link $W(1) $BS(1) 15Mb 3ms DropTail

$ns\_ duplex-link $W(1) $BS(2) 5Mb 3ms DropTail

$ns\_ duplex-link-op $W(0) $W(1) orient down

$ns\_ duplex-link-op $W(1) $BS(0) orient left-down

$ns\_ duplex-link-op $W(1) $BS(1) orient down

$ns\_ duplex-link-op $W(1) $BS(2) orient right-down

# setup TCP connections

set tcp1 [new Agent/TCP]

$tcp1 set class\_ 2

set sink1 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(0) $tcp1

$ns\_ attach-agent $W(0) $sink1

$ns\_ connect $tcp1 $sink1

$tcp1 set windowOption\_ 25

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ns\_ at $opt(ftp1-start) "$ftp1 start"

set tcp2 [new Agent/TCP]

$tcp2 set class\_ 2

set sink2 [new Agent/TCPSink]

$ns\_ attach-agent $node\_(4) $tcp2

$ns\_ attach-agent $node\_(2) $sink2

$ns\_ connect $tcp2 $sink2

$tcp2 set windowOption\_ 25

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ns\_ at $opt(ftp2-start) "$ftp2 start"

for {set j 0} {$j <$opt(nn)} {incr j} {

$ns\_ at $opt(stop).0 "$node\_($j) reset";

}

# Tell all nodes when the simulation ends

$ns\_ at $opt(stop).0 "$BS(0) reset";

$ns\_ at $opt(stop).0 "$BS(1) reset";

$ns\_ at $opt(stop).0 "$BS(2) reset";

$ns\_ at 0.55 "record"

$ns\_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns\_ halt"

$ns\_ at $opt(stop).0001 "finish"

proc plotWindow {tcpSource outfile} {

global ns\_

set now [$ns\_ now]

set cwnd [$tcpSource set cwnd\_]

puts $outfile "$now $cwnd"

$ns\_ at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile1 [open my\_algo\_wandw1.xg w]

set outfile2 [open my\_algo\_wandw2.xg w]

$ns\_ at 0.0 "plotWindow $tcp1 $outfile1"

$ns\_ at 0.0 "plotWindow $tcp2 $outfile2"

$ns\_ run

**Python Code:**

def strip\_data(xgfile):

lines = xgfile.readlines()

timeStamps = [ ]

measures = [ ]

for line in lines:

time,measure = line.split(' ')

timeStamps.append(float(time))

measures.append(float(measure))

timeStamps = np.array(timeStamps)

measures = np.array(measures)

return timeStamps,measures

def cwndgraph(files,ax,i):

flag = False

main\_df = pd.DataFrame()

counter = 1

for file **in** files:

timeStamps,measures = strip\_data(file)

string = 'cwnd of agent'

if(flag == False):

main\_df['Time'] = pd.Series(timeStamps)

main\_df["**{}{}**".format(string,counter)] = pd.Series(measures)

filename = os.path.basename(file.name)

variant,last = filename.split('\_')

topo,residue = last.split('.')

topo = topo[:-1]

flag = True

main\_df["**{}{}**".format(string,counter)] = pd.Series(measures)

counter = counter + 1

plt.figure(figsize=(15,8))

palette = sns.color\_palette("bright")

res = random.sample(palette, len(files))

ax = sns.lineplot(x='Time',

y='value',

hue='variable',

ax=ax[i],

data=pd.melt(main\_df, ['Time']),

palette=res)

plt.style.use(["dark\_background"])

ax.set\_xlabel("Time")

ax.set\_ylabel("Congestion Window Size")

ax.set\_title(variant+' variant '+topo+' topology')

return ax

def bw\_tp\_graph(files,ax,i):

flag = False

main\_df = pd.DataFrame()

counter = 1

for file in files:

timeStamps,measures = strip\_data(file)

if(flag == False):

filename = os.path.basename(file.name)

measure,variant,last = filename.split('\_')

topo,residue = last.split('.')

main\_df['Time'] = pd.Series(timeStamps)

main\_df[variant] = pd.Series(measures)

flag = True

filename = os.path.basename(file.name)

measure,variant,last = filename.split('\_')

topo,residue = last.split('.')

main\_df[variant] = pd.Series(measures)

counter = counter + 1

plt.figure(figsize=(15,8))

palette = sns.color\_palette("bright")

res = random.sample(palette, len(files))

ax = sns.lineplot(x='Time',

y='value',

hue='variable',

ax=ax[i],

data=pd.melt(main\_df, ['Time']),

palette=res)

plt.style.use(["dark\_background"])

ax.set\_xlabel("Time")

if measure == "tb":

ax.set\_ylabel("total bandwidth")

elif measure == "tt":

ax.set\_ylabel("total throughput")

ax.set\_title(topo+' topology')

return ax

**EXPERIMENTAL ANALYSIS**

The experimental setup of the project is as follows:

**Wired - Grid Topology**

Network Simulation - Network Simulator version 2.35

Number of nodes - 3 rows x 3 columns = 9 nodes

Simulation Time - 40ms

**Wireless**

Network Simulation - Network Simulator version 2.35

Number of nodes - 10

Topology Size - 500x500

Routing Protocol - AODV

Traffic Type - FTP

Data Type - TCP

MAC - 802.11

Channel Type - Wireless Channel

Radio Propagation Type - Two Ray Ground

Network Interface - WirelessPhy

Interface Queue Length - 50

Simulation Time - 15ms

**Mixed Wired and Wireless**

Network Simulation - Network Simulator version 2.35

Number of nodes - 11 ( 2 wired + 3 Base-Stations + 6 Mobile nodes )

Topology Size - 670x670

Routing Protocol - DSDV

Traffic Type - FTP

Data Type - TCP

MAC - 802.11

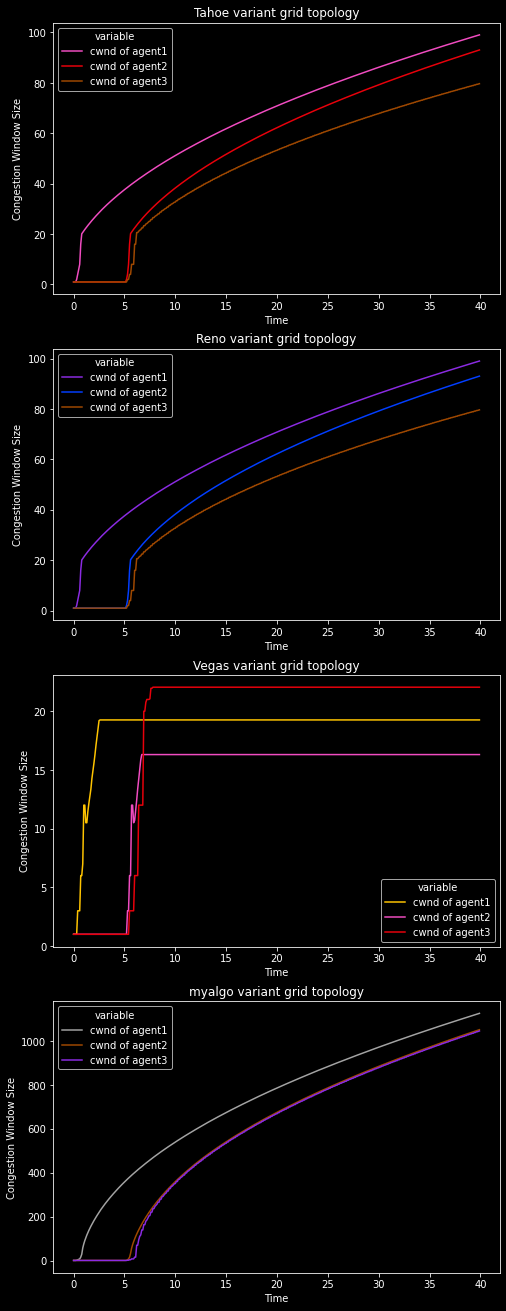
Channel Type - Wireless Channel

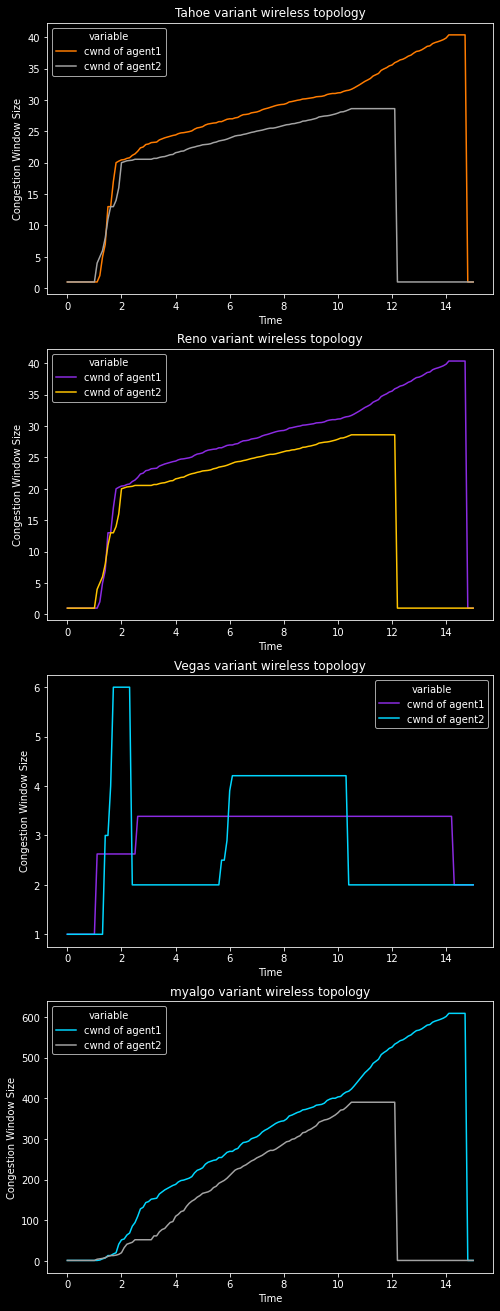
Radio Propagation Type - Two Ray Ground

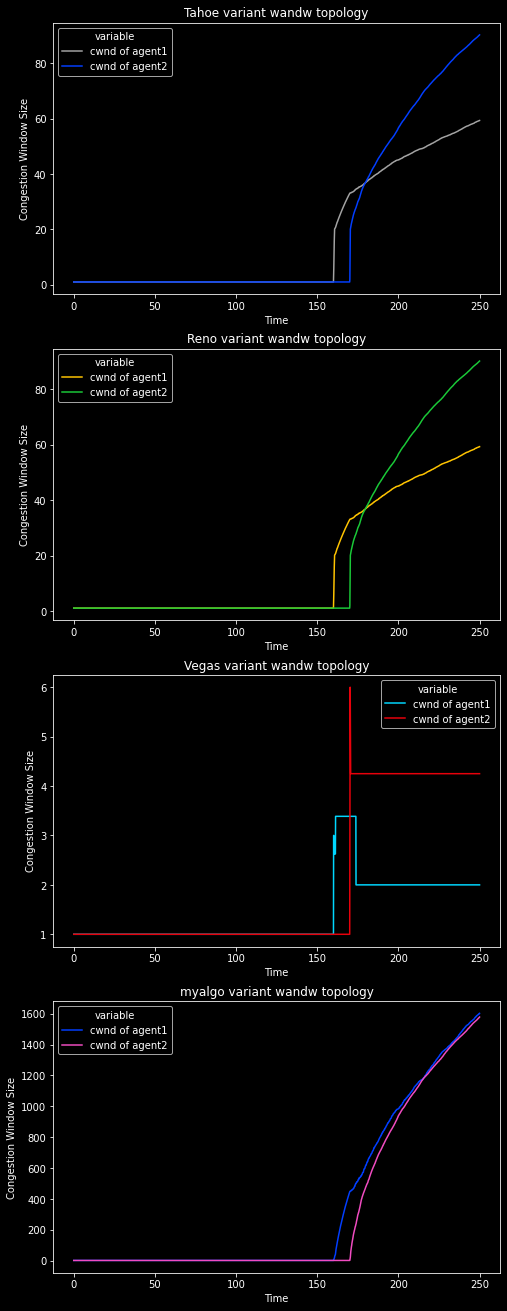
Network Interface - WirelessPhy

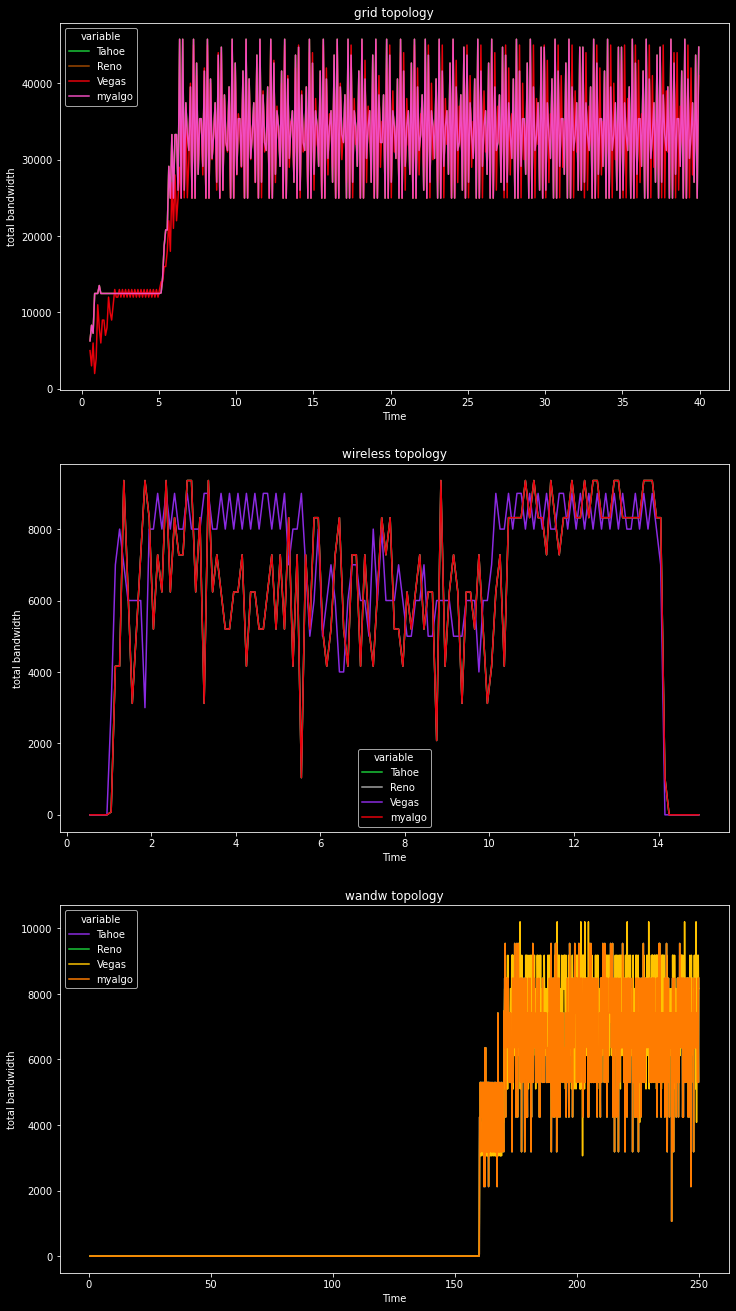
Interface Queue Length - 50

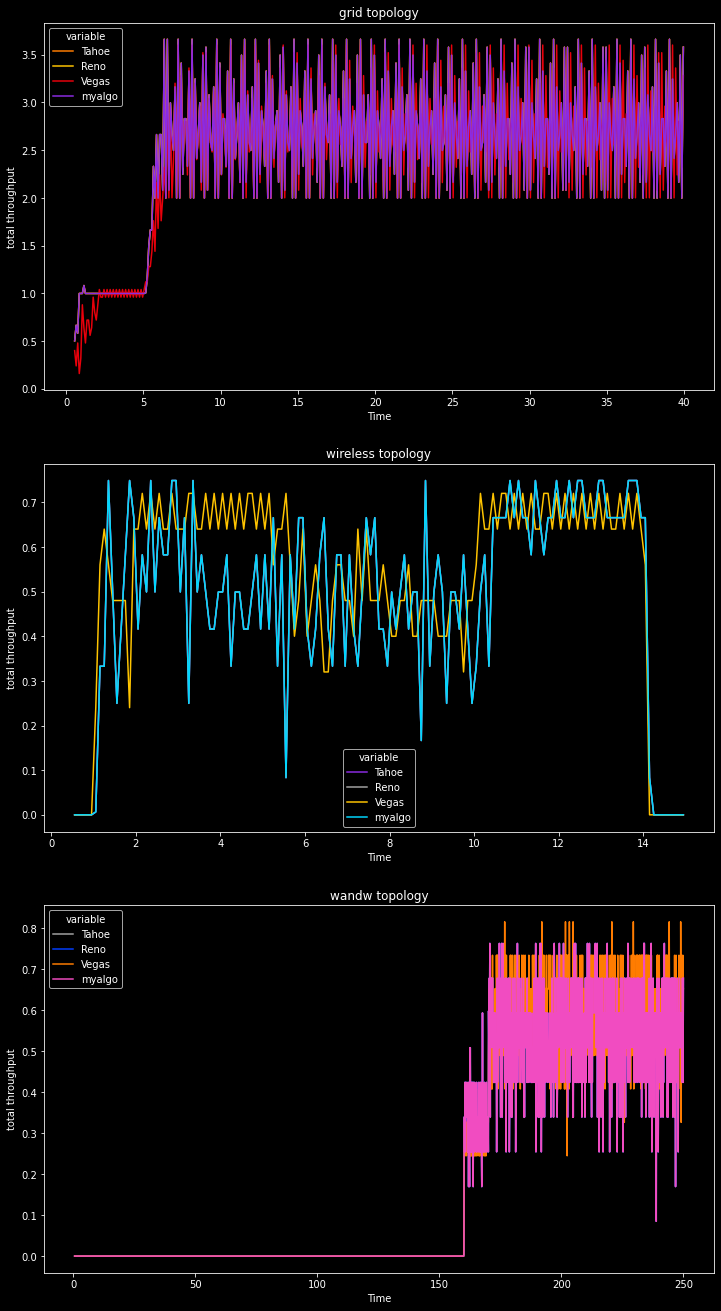
Simulation Time - 250ms











**CONCLUSION**

From the graphs we can observe that TCP Tahoe and TCP Reno perform almost the same irrespective of the topology. TCP Vegas on the other hand performs efficiently compared to other variants which is visible from the throughput graphs. TCP Vegas decreases its cwnd as soon as it detects network congestion, contrary to TCP Reno which waits until a timeout is observed. This fact is visible in the cwnd graphs. The rapid descent in cwnd value of TCP Vegas in wireless and hybrid topologies attributed to the fact that along with network congestion, mobility of nodes also leads to packet loss. Further research and analysis is required to determine the comparative performances of the TCP variants. Cwnd size and throughput as a function of number of nodes can also be computed for each TCP variant. My tweaked loss-based congestion control algorithm is observed to be successfully implemented and cwnd graph provides insight into how the algorithm works and progresses through the simulation time.

**REFERENCES**

(Vivek Kumar Majithiya, 2014) Bhatt, Priyang. (2013). *Performance Analysis of TCP variants in Wired and Wireless scenarios*. 2. 2321-613.

(S Islam, 2018) Islam, S., Kuri, R., Kabir, M. H., & Hossain, M. J. (2018). Exploring Congestion Control Mechanism of TCP Vari-ants over Wired & Wireless Networks. *International Journal of Scientific & Engineering Research*, *9*(9).

(Rani K. V., 2017) Rani, K. V., & Reddy, P. C. TCP Congestion Control Mechanisms evaluation over Wired/Wireless Environments: A Survey.

Herlein et.al (Herlein, 2018) Herlein, D. R. R., Talay, C. A., González, C. N., Trinidad, F. A., Almada, M. L., & Marrone, L. A. (2018, October). Contention Analysis of Congestion Control Mechanisms in a Wireless Access Scenario. In *Argentine Congress of Computer Science* (pp. 249-263). Springer, Cham.

(AP Pande, 2018) Pande, A. P., & Devane, S. R. (2018, August). Study and Analysis of Different TCP Variants. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (pp. 1-8). IEEE.

(HK Molia, 2018) Molia, H. K., & Kothari, A. D. (2018). TCP variants for mobile adhoc networks: Challenges and solutions. *Wireless Personal Communications*, *100*(4), 1791-1836.

(CN González, 2019) González, C. N., Trinidad, F. A., Almada, M. L., & Marrone, L. A. (2019, May). Contention Analysis of Congestion Control Mechanisms in a Wireless Access Scenario. In *Computer Science–CACIC 2018: 24th Argentine Congress, Tandil, Argentina, October 8–12, 2018, Revised Selected Papers* (Vol. 995, p. 249). Springer.

(J VijiPriya, 2016) VijiPriya, J., & Suppiah, S. (2016). An Extended Study On Newton Raphson Congestion Control. *International Journal of Advanced Research*, *4*(2), 574-58.

(Z Wang, 2016) Wang, Z., Zeng, X., Liu, X., Xu, M., Wen, Y., & Chen, L. (2016). TCP congestion control algorithm for heterogeneous Internet. *Journal of network and computer applications*, *68*, 56-64.

(N Bhargava, 2017) Bhargava, N., Bhargava, R., Rathore, P. S., & Watts, G. B. INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY AN ADAPTIVE APPROACH FOR TCP VARIANTS APPLYING WIRELESS MULTIHOP NETWORKS.