**CS 261 - COMPUTER NETWORKS**

**B.Tech CSE (V Semester)**

**Project Report**

**Group No.: 3**

**Section: S6-S7**

**Submitted By:**

Kumar Aniket (U101115FCS197)

Manik Garg (U101115FCS199)

Lavkush Singh (U101115FCS113)

Katragadda Akash (U101115FCS108)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NIIT UNIVERSITY, NEEMRANA (RJ)**

**November 2017**

TABLE OF CONTENTS

Contents Page no.

1. Chapter 1 1

Introduction

1. Chapter 2 4

Implementation and study of Telnet and Comparison of throughput with other network protocols (Part A)

2.1 Introduction to Protocol

Protocol features with diagram

2.2 Evaluation Features / Parameters

Parameters with formulae

2.3 Graphical Analysis of Protocol

Graphs based on parameters.

1. Chapter 3 20

Implementation of AODV routing protocol and CBR Traffic (Part B)

3.1 Introduction to Protocol

Protocol features with diagram

3.2 Evaluation Features / Parameters

Parameters with formulae

3.3 Graphical Analysis of Protocol

Graphs based on parameters.

1. Chapter 4 36

Study and Implementation of stop and wait Protocol (Part C)

4.1 Introduction to Protocol

Protocol features with diagram

4.2 Evaluation Features / Parameters

Parameters with formulae

4.3 Graphical Analysis of Protocol

Graphs based on parameters.

5. References 52

6. AWK codes for Part A, B and C 53

Chapter 1: Introduction

This Report consists of study and implementation of:

* TELNET
* AODV Routing Protocol
* Stop and Wait ARQ

Graphs have been prepared using the NS2 Simulation of the respective TCL Scripts and various factors have been analyzed. This report also contains images, AWK file scripts, TCL scripts, graphs so generated with data, and references apart the content.

Chapter 2: TELNET PROTOCOL

2.1 Introduction

TELNET: TELNET (TELEcommunication NETwork) is a network protocol used on Internet or Local Area Network (LAN) connections. It is a network protocol used to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection. It is developed in the year 1969

Typically, telnet provides access to a command-line interface on a remote host via a virtual terminal connection which consists of an 8-bit byte oriented data connection over the Transmission Control Protocol (TCP).

Telnet can be used to connect to virtually any machine that listens on ports. In other words, you can connect to any machine that has certain ports open. Once connected to a machine, you need to issue UNIX based commands to interact with the remote service. To use Telnet, you need to know the address of the host whose resources you want to use.

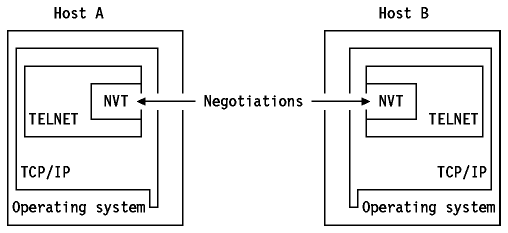
Your Telnet client contacts the host using its internet address. When you contact the host, the distant computer and your computer negotiate how they will communicate with each other. They decide which terminal emulation will be used. Telnet emulation determines how your keyboard will transmit information to the distant computer and how information will be displayed on your screen.

A Telnet command request looks like this (the computer name is made-up):

telnet the.libraryat.whatis.edu

The result of this request would be an invitation to log on with a userid and a prompt for a password. If accepted, you would be logged on like any user who used this computer every day.

Telnet is most likely to be used by program developers and anyone who has a need to use specific applications or data located at a particular host computer.



**Advantages:**

Accessing Remote computers: You can view the contents of another machine from your home device.

Saves Time: One does not have to physically be present on the system to use it

Universal: Runs on all Systems new/old

Router configuration: Since it uses plain text it is very easy to fix problems. This allows for more access and less problems with data transmission.

Most of its implementations do not have any authentication which lead to easy interceptions. This lead to the rise of SSH which replaced telnet everywhere

**Disadvantages:**

Not Secure: Telnet doesn’t encrypt any data sent over the connection, so it is often easy to listen to this communication by anyone who has access to router, switch, hub or gateway located on the transmission network.

Difficult to Use: It is purely command driven hence it can get tricky to use at times and isn’t easy for beginners.

Old: Can only send plain text. No color or Graphics involved

2.2 Evaluation Parameters:

Packet Delivery Rate:

It is the number of packets sent on that particular connection over the simulation time of that particular connection.

PDR:

No of Packets Rec / (Protocol Finish Time - Protocol Start Time)

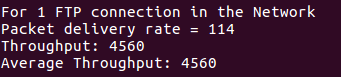
Throughput:

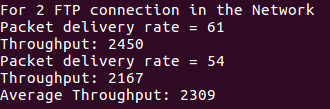
It is the data rate of that particular network. Given by the formula:

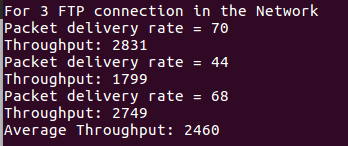
Throughput = PDR\*Size

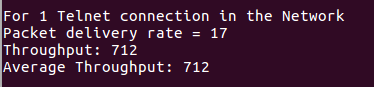
2.3 Graphical Analysis of Data Simulated:

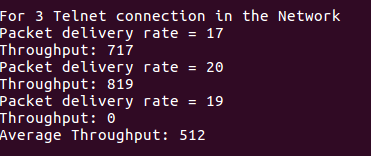
Data:

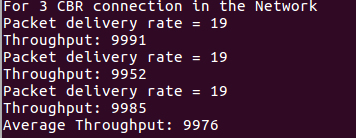


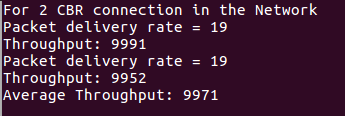


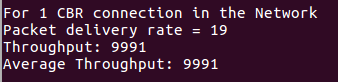


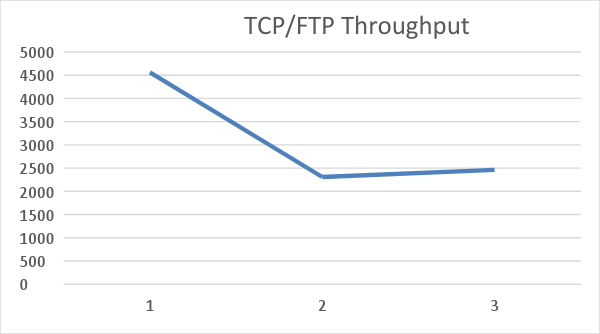


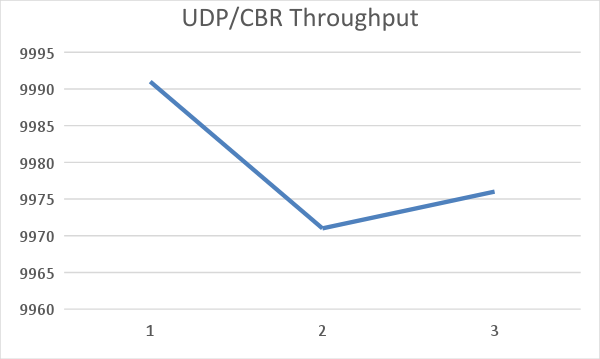


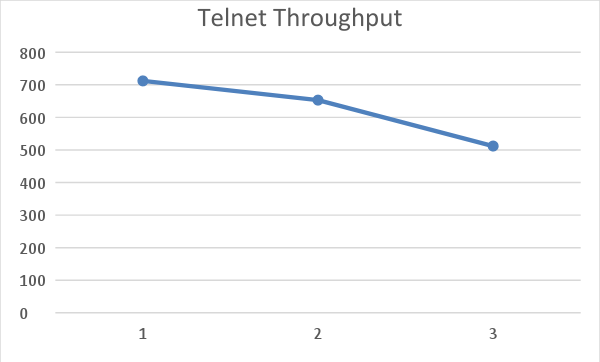






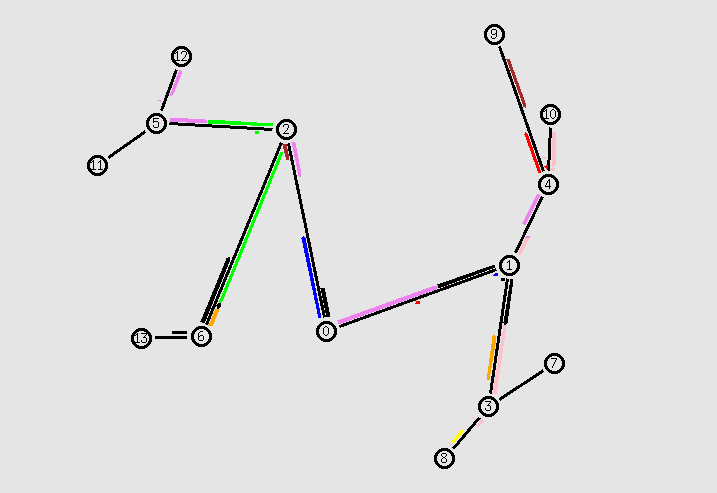






TCL Script:

14 Node Tree Topology Created and Simulated with 9 Connections :



set ns [new Simulator]

set f [open out.tr w]

$ns trace-all $f

set nf [open out.nam w]

$ns namtrace-all $nf

$ns color 1 Red

$ns color 2 Blue

$ns color 3 Green

$ns color 4 Orange

$ns color 5 Yellow

$ns color 6 Brown

$ns color 7 Pink

$ns color 8 Black

$ns color 9 Violet

proc finish {} {

global ns nf f

$ns flush-trace

close $nf

close $f

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

set n9 [$ns node]

set n10 [$ns node]

set n11 [$ns node]

set n12 [$ns node]

set n13 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link $n1 $n3 1Mb 10ms DropTail

$ns duplex-link $n1 $n4 1Mb 10ms DropTail

$ns duplex-link $n2 $n5 1Mb 10ms DropTail

$ns duplex-link $n2 $n6 1Mb 10ms DropTail

$ns duplex-link $n3 $n7 1Mb 10ms DropTail

$ns duplex-link $n3 $n8 1Mb 10ms DropTail

$ns duplex-link $n4 $n9 1Mb 10ms DropTail

$ns duplex-link $n4 $n10 1Mb 10ms DropTail

$ns duplex-link $n5 $n11 1Mb 10ms DropTail

$ns duplex-link $n5 $n12 1Mb 10ms DropTail

$ns duplex-link $n6 $n13 1Mb 10ms DropTail

#0-Source 9-Destination

set telnet\_TCP\_agent\_0 [new Agent/TCP]

$ns attach-agent $n0 $telnet\_TCP\_agent\_0

set telnet\_TCP\_sink\_9 [new Agent/TCPSink]

$ns attach-agent $n9 $telnet\_TCP\_sink\_9

$ns connect $telnet\_TCP\_agent\_0 $telnet\_TCP\_sink\_9

$telnet\_TCP\_agent\_0 set fid\_ 1

set telnet\_TELNET\_source\_0 [new Application/Telnet]

$telnet\_TELNET\_source\_0 attach-agent $telnet\_TCP\_agent\_0

$telnet\_TELNET\_source\_0 set interval\_ 0.05

#7-Source 5-Destination Telnet

set telnet\_TCP\_agent\_7 [new Agent/TCP]

$ns attach-agent $n7 $telnet\_TCP\_agent\_7

set telnet\_TCP\_sink\_5 [new Agent/TCPSink]

$ns attach-agent $n5 $telnet\_TCP\_sink\_5

$ns connect $telnet\_TCP\_agent\_7 $telnet\_TCP\_sink\_5

$telnet\_TCP\_agent\_7 set fid\_ 2

set telnet\_TELNET\_source\_7 [new Application/Telnet]

$telnet\_TELNET\_source\_7 attach-agent $telnet\_TCP\_agent\_7

$telnet\_TELNET\_source\_7 set interval\_ 0.05

#11-Source 13-Destination Telnet

set telnet\_TCP\_agent\_11 [new Agent/TCP]

$ns attach-agent $n11 $telnet\_TCP\_agent\_11

set telnet\_TCP\_sink\_13 [new Agent/TCPSink]

$ns attach-agent $n13 $telnet\_TCP\_sink\_13

$ns connect $telnet\_TCP\_agent\_11 $telnet\_TCP\_sink\_13

$telnet\_TCP\_agent\_11 set fid\_ 3

set telnet\_TELNET\_source\_11 [new Application/Telnet]

$telnet\_TELNET\_source\_11 attach-agent $telnet\_TCP\_agent\_11

$telnet\_TELNET\_source\_11 set interval\_ 0.05

#3-source 6-dest UDP

set udp3 [ new Agent/UDP ]

$ns attach-agent $n3 $udp3

set null6 [ new Agent/Null ]

$ns attach-agent $n6 $null6

$ns connect $udp3 $null6

$udp3 set fid\_ 4

set cbr3 [ new Application/Traffic/CBR ]

$cbr3 set packetSize\_ 500

$cbr3 set interval\_ 0.05

$cbr3 attach-agent $udp3

#8-source 12-dest UDP

set udp8 [ new Agent/UDP ]

$ns attach-agent $n8 $udp8

set null12 [ new Agent/Null ]

$ns attach-agent $n12 $null12

$ns connect $udp8 $null12

$udp8 set fid\_ 5

set cbr8 [ new Application/Traffic/CBR ]

$cbr8 set packetSize\_ 500

$cbr8 set interval\_ 0.05

$cbr8 attach-agent $udp8

#9-source 2-destination UDP

set udp9 [ new Agent/UDP ]

$ns attach-agent $n9 $udp9

set null2 [ new Agent/Null ]

$ns attach-agent $n2 $null2

$ns connect $udp9 $null2

$udp9 set fid\_ 6

set cbr9 [ new Application/Traffic/CBR ]

$cbr9 set packetSize\_ 500

$cbr9 set interval\_ 0.05

$cbr9 attach-agent $udp9

#10-source 8-destination ftp

set tcp10 [ new Agent/TCP ]

$ns attach-agent $n10 $tcp10

set sink8 [new Agent/TCPSink]

$ns attach-agent $n8 $sink8

$ns connect $tcp10 $sink8

$tcp10 set fid\_ 7

set ftp10 [ new Application/FTP ]

$ftp10 attach-agent $tcp10

#13-source 3-destination ftp

set tcp13 [ new Agent/TCP ]

$ns attach-agent $n13 $tcp13

set sink3 [new Agent/TCPSink]

$ns attach-agent $n3 $sink3

$ns connect $tcp13 $sink3

$tcp13 set fid\_ 8

set ftp13 [ new Application/FTP ]

$ftp13 attach-agent $tcp13

#12-source 4-destination ftp

set tcp12 [ new Agent/TCP ]

$ns attach-agent $n12 $tcp12

set sink4 [new Agent/TCPSink]

$ns attach-agent $n4 $sink4

$ns connect $tcp12 $sink4

$tcp12 set fid\_ 9

set ftp12 [ new Application/FTP ]

$ftp12 attach-agent $tcp12

$ns at 1.0 "$telnet\_TELNET\_source\_0 start"

$ns at 1.0 "$telnet\_TELNET\_source\_7 start"

$ns at 1.0 "$telnet\_TELNET\_source\_11 start"

$ns at 1.0 "$cbr3 start"

$ns at 1.0 "$cbr8 start"

$ns at 1.0 "$cbr9 start"

$ns at 1.0 "$ftp10 start"

$ns at 1.0 "$ftp13 start"

$ns at 1.0 "$ftp12 start"

$ns at 8.0 "$telnet\_TELNET\_source\_0 stop"

$ns at 8.0 "$telnet\_TELNET\_source\_7 stop"

$ns at 8.0 "$telnet\_TELNET\_source\_11 stop"

$ns at 8.0 "$cbr3 stop"

$ns at 8.0 "$cbr8 stop"

$ns at 8.0 "$cbr9 stop"

$ns at 8.0 "$ftp10 stop"

$ns at 8.0 "$ftp13 stop"

$ns at 8.0 "$ftp12 stop"

$ns at 10.0 "finish"

$ns run

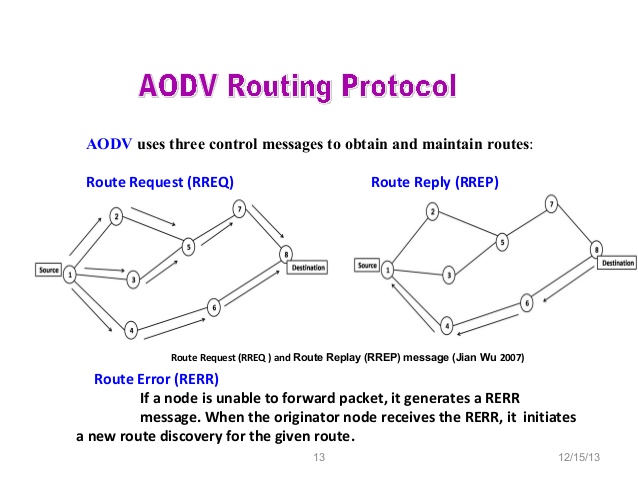
Chapter 3: AODV Routing Protocol

3.1 Introduction

AODV Routing Protocol, which stands for Ad-hoc On-demand Distance Vector Routing Protocol, belongs to the mobile nodes in an ad-hoc network and is referred to as reactive protocol. An Ad-hoc network does not have any access point. The communication is achieved by the data transfer among the nodes which are present in their respective transmission range. The AODV algorithm is designed such that the route is established only if requested by the nodes present in the corresponding ad-hoc network, which facilitates the nodes to enter and exit into the network as per the need. The path or the route so established for data transmission remains active for the session until data transfer takes place. As soon as the data transmission stops, the route becomes invalid or expired after some defined ‘timeout’ period.

When a node wishes to transmit traffic to a host to which it has no route, it will generate a route request (RREQ) message that will be flooded in a limited way to other nodes. This causes control traffic overhead to be dynamic and it will result in an initial delay when initiating such communication. A route is considered found when the RREQ message reaches either the destination itself, or an intermediate node with a valid route entry for the destination. For as long as a route exists between two endpoints, AODV remains passive. When the route becomes invalid or lost, AODV will again issue a request.

AODV avoids the “counting to infinity” problem from the classical distance vector algorithm by using sequence numbers for every route. The counting to infinity problem is the situation where nodes update each other in a loop.



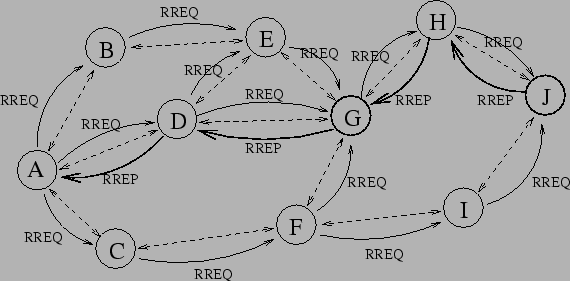
**Table 1: Diagram describing AODV Routing**

AODV defines three types of control messages for route maintenance:

**RREQ** - A *route request* message is transmitted by a node requiring a route to a node.

**RREP** - A *route reply* message is unicasted back to the originator of a RREQ if the receiver is either the node using the requested address, or it has a valid route to the requested address.

**RERR** - Nodes monitor the link status of next hops in active routes. When a link breakage in an active route is detected, a RERR message is used to notify other nodes of the loss of the link.



**Table 2: RREQ and RREP in AODV**

3.2 Parameters with formulae

For this analysis, we have tried to study and analyze the following parameters in AODV with CBR implementation:

* Average end-to-end delay
* Throughput of the system
* Packet delivery ratio
* Route Request
* Route Reply

The number of nodes have been varied and the effect of the above stated parameters was observed via graphical analysis of the obtained primary data via simulation of the scenario in NS-2 Software using TCL scripts and running AWK files in Ubuntu 17.04 Operating System.

The logic of each parameter is extracted via the ‘tr’ file generated. The corresponding formulas can be referred from the AWK codes given in section 6.

TCL Script:

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

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

set val(ant) Antenna/OmniAntenna ;# Antenna type

set val(ll) LL ;# Link layer type

set val(ifq) Queue/DropTail/PriQueue ;# Interface queue type

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

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

set val(rp) AODV ;# ad-hoc routing protocol

set val(nn) 10;# number of mobilenodes

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

set val(x) 500;

set val(y) 500;

set val(stop) 150; # time of simulation end

set ns [new Simulator]

set tracefd [open aodv.tr w]

set windowVsTime2 [open win.tr w]

set namtrace [open aodv.nam w]

$ns trace-all $tracefd

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

# set up topography object

set topo [new Topography]

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

create-god $val(nn)

# configure the nodes

$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) \

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF \

-movementTrace ON

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

set node\_($i) [$ns node]

$node\_($i) set X\_ [ expr 10+round(rand()\*480) ]

$node\_($i) set Y\_ [ expr 10+round(rand()\*380) ]

$node\_($i) set Z\_ 0.0

}

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

$ns at [ expr 15+round(rand()\*60) ] "$node\_($i) setdest [ expr 10+round(rand()\*480) ] [ expr 10+round(rand()\*380) ] [ expr 2+round(rand()\*15) ]"

}

# Provide initial location of mobilenodes

#$node\_(0) set X\_ 5.0

#$node\_(0) set Y\_ 5.0

#$node\_(0) set Z\_ 0.0

#$node\_(1) set X\_ 490.0

#$node\_(1) set Y\_ 285.0

#$node\_(1) set Z\_ 0.0

#$node\_(2) set X\_ 150.0

#$node\_(2) set Y\_ 240.0

#$node\_(2) set Z\_ 0.0

#$node\_(3) set X\_ 350.0

#$node\_(3) set Y\_ 310.0

#$node\_(3) set Z\_ 0.0

#$node\_(4) set X\_ 550.0

#$node\_(4) set Y\_ 520.0

#$node\_(4) set Z\_ 0.0

#$node\_(5) set X\_ 740.0

#$node\_(5) set Y\_ 730.0

#$node\_(5) set Z\_ 0.0

# Generation of movements

#$ns at 10.0 "$node\_(0) setdest 250.0 250.0 3.0"

#$ns at 15.0 "$node\_(1) setdest 45.0 285.0 5.0"

#$ns at 110.0 "$node\_(0) setdest 480.0 300.0 5.0"

# Set a UDP connection between node\_(0) and node\_(1)

set udp [new Agent/UDP]

$udp set class\_ 2

set null [new Agent/Null]

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

$ns attach-agent $node\_(1) $null

$ns connect $udp $null

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$ns at 10.0 "$cbr start"

# Printing the window size

#proc plotWindow {tcpSource file} {

#global ns

#set time 0.01

#set now [$ns now]

#set cwnd [$tcpSource set cwnd\_]

#puts $file "$now $cwnd"

#$ns at [expr $now+$time] "plotWindow $tcpSource $file" }

#$ns at 10.1 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam

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

# 30 defines the node size for nam

$ns initial\_node\_pos $node\_($i) 30

}

# Telling nodes when the simulation ends

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

$ns at $val(stop) "$node\_($i) reset";

}

# ending nam and the simulation

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

$ns at $val(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

global ns tracefd namtrace

$ns flush-trace

close $tracefd

close $namtrace

exec nam aodv.nam &

exit 0

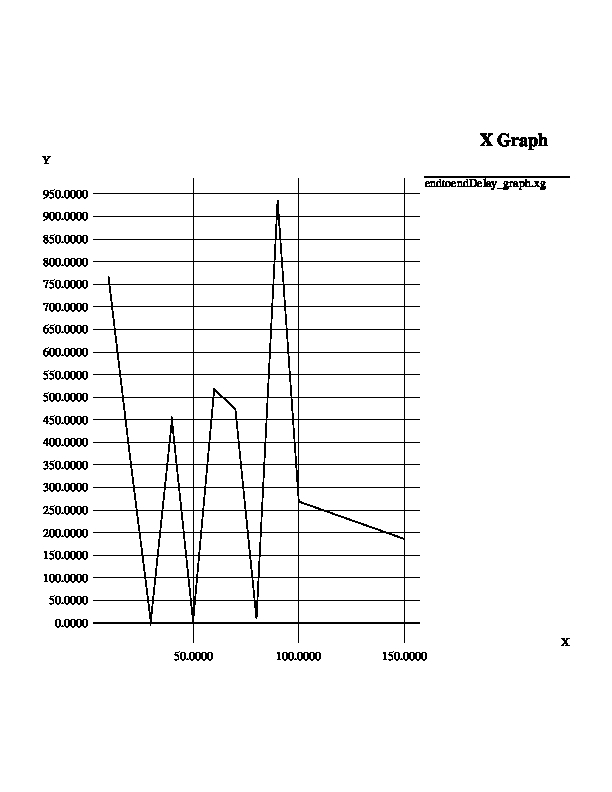
}

$ns run

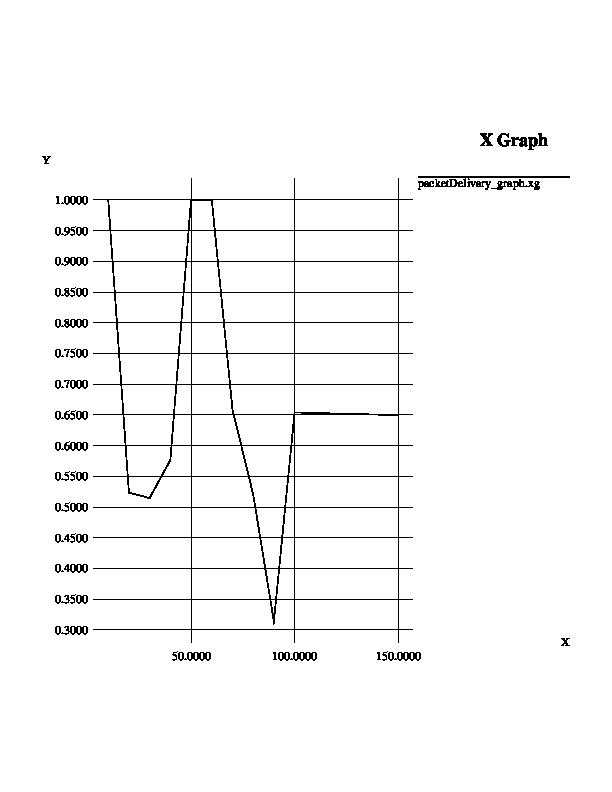
3.3 Graphical analysis of the Protocol.

Number of nodes were varied in the difference of 10, starting from 10 nodes to the maximum of 100 nodes in TCL file using AODV routing with CBR traffic. The following graphs were obtained:

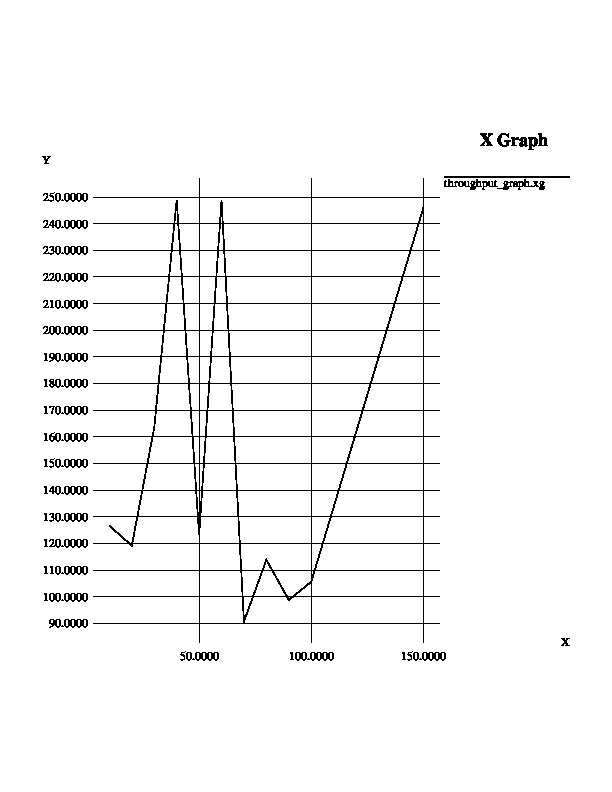
I. End to End Delay



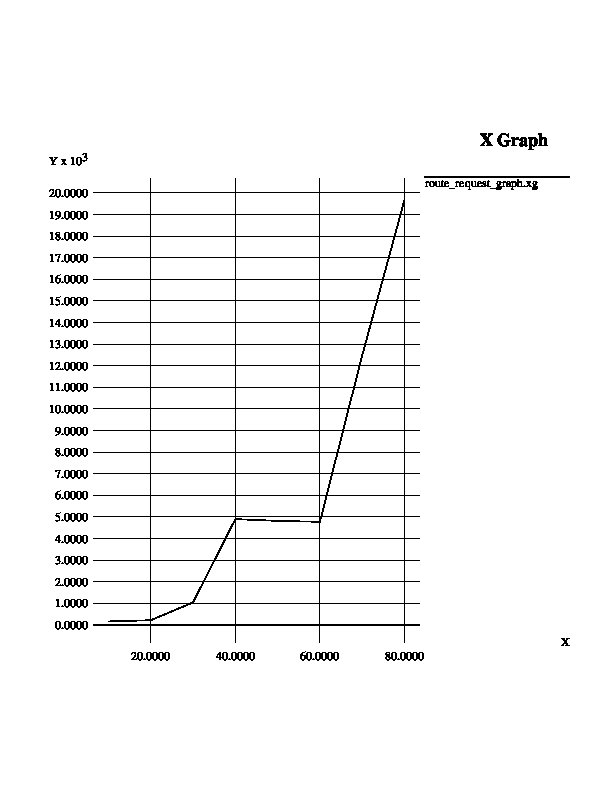
II. Packet Delivery Ratio:



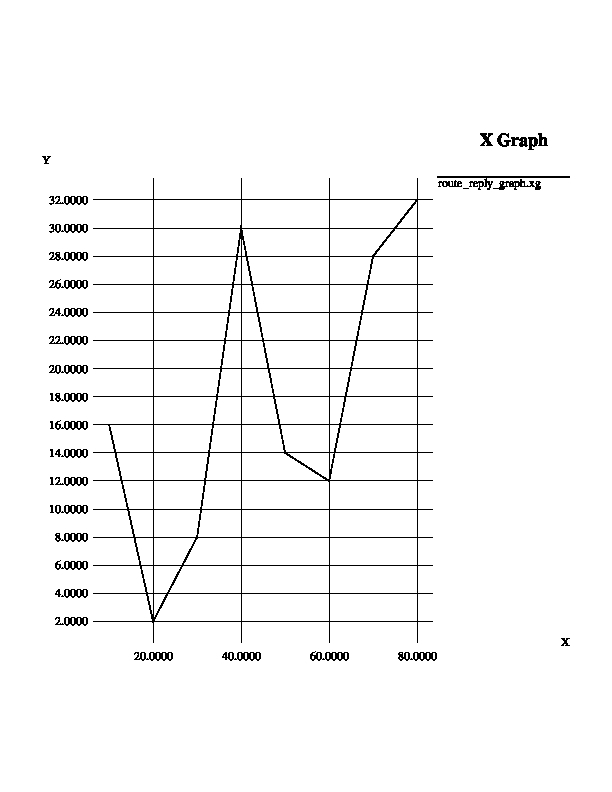
III. Throughput of the system



IV. Route Request generated:



V. Route Reply Received:



Chapter 3: Stop and Wait Protocol

3.1 Introduction

**Stop-and-wait ARQ**, also referred to as alternating bit Protocol, is a method in telecommunications to send information between two connected devices. It ensures that information is not lost due to dropped packets and that packets are received in the correct order. It is the simplest automatic repeat-request (ARQ) mechanism. A stop-and wait ARQ sender sends one frame at a time; it is a special case of the general sliding window protocol with transmit and receive window sizes equal to one and greater than one respectively. After sending each frame, the sender doesn't send any further frames until it receives an acknowledgement (ACK) signal.

Characteristics

* Used in Connection-oriented communication.
* It offers error and flow control
* It is used in Data Link and Transport Layers
* Stop and Wait ARQ mainly implements Sliding

Window Protocol concept with Window Size 1 (one)

Throughput

U=Tf/Tt

Problems

* Lost data
* Lost acknowledgement
* Delayed acknowledgement

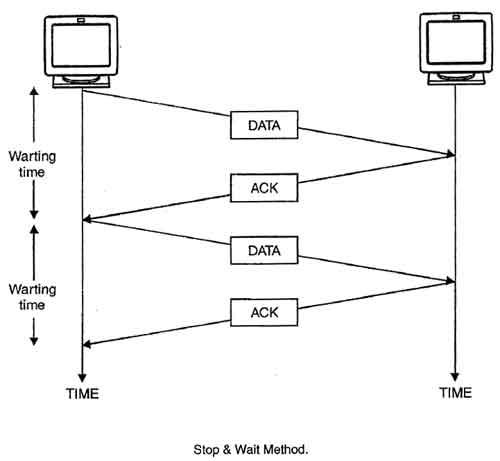
Solution

The above problems are solved respectively by these 3 methods:

* Time out
* Sequence number (data)
* Sequence number (ack)

The main advantage of stop & wait protocols is its accuracy. Next frame is transmitted only when the first frame is acknowledged. So there is no chance of frame being lost.

The main disadvantage of this method is that it is inefficient. It makes the transmission process slow. In this method single frame travels from source to destination and single acknowledgment travels from destination to source. As a result each frame sent and received uses the entire time needed to traverse the link. Moreover, if two devices are distance apart, a lot of time is wasted waiting for ACKs that leads to increase in total transmission time.



Stop and Wait TCL script with normal operation:

# stop and wait protocol in normal situation

# features : labeling, annotation, nam-graph, and window size monitoring

set ns [new Simulator]

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$ns at 0.0 "$n0 label Sender"

$ns at 0.0 "$n5 label Receiver"

set nf [open A1-stop-n-wait.nam w]

$ns namtrace-all $nf

set f [open A1-stop-n-wait.tr w]

$ns trace-all $f

$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail

$ns duplex-link-op $n0 $n1 orient right

$ns queue-limit $n0 $n1 10

$ns duplex-link $n1 $n2 0.2Mb 200ms DropTail

$ns duplex-link-op $n1 $n2 orient right

$ns queue-limit $n1 $n2 10

$ns duplex-link $n2 $n3 0.2Mb 200ms DropTail

$ns duplex-link-op $n2 $n3 orient right

$ns queue-limit $n2 $n3 10

$ns duplex-link $n3 $n4 0.2Mb 200ms DropTail

$ns duplex-link-op $n3 $n4 orient right

$ns queue-limit $n3 $n4 10

$ns duplex-link $n4 $n5 0.2Mb 200ms DropTail

$ns duplex-link-op $n4 $n5 orient right

$ns queue-limit $n4 $n5 10

Agent/TCP set nam\_tracevar\_ true

set tcp [new Agent/TCP]

$tcp set window\_ 1

$tcp set maxcwnd\_ 1

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

set tcp1 [new Agent/TCP]

$tcp set window\_ 1

$tcp set maxcwnd\_ 1

$ns attach-agent $n3 $tcp1

set sink1 [new Agent/TCPSink]

$ns attach-agent $n4 $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ns add-agent-trace $tcp tcp

$ns monitor-agent-trace $tcp

$tcp tracevar cwnd\_

$ns add-agent-trace $tcp1 tcp1

$ns monitor-agent-trace $tcp1

$tcp1 tracevar cwnd\_

$ns at 0.1 "$ftp start"

$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"

$ns at 3.5 "finish"

$ns at 0.1 "$ftp1 start"

$ns at 3.0 "$ns detach-agent $n3 $tcp1 ; $ns detach-agent $n1 $sink"

$ns at 3.5 "finish"

$ns at 0.0 "$ns trace-annotate \"Stop and Wait with normal operation\""

$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""

$ns at 0.11 "$ns trace-annotate \"Send Packet\_0\""

$ns at 0.35 "$ns trace-annotate \"Receive Ack\_0\""

$ns at 0.56 "$ns trace-annotate \"Send Packet\_1\""

$ns at 0.79 "$ns trace-annotate \"Receive Ack\_1\""

$ns at 0.99 "$ns trace-annotate \"Send Packet\_2\""

$ns at 1.23 "$ns trace-annotate \"Receive Ack\_2 \""

$ns at 1.43 "$ns trace-annotate \"Send Packet\_3\""

$ns at 1.67 "$ns trace-annotate \"Receive Ack\_3\""

$ns at 1.88 "$ns trace-annotate \"Send Packet\_4\""

$ns at 2.11 "$ns trace-annotate \"Receive Ack\_4\""

$ns at 2.32 "$ns trace-annotate \"Send Packet\_5\""

$ns at 2.55 "$ns trace-annotate \"Receive Ack\_5 \""

$ns at 2.75 "$ns trace-annotate \"Send Packet\_6\""

$ns at 2.99 "$ns trace-annotate \"Receive Ack\_6\""

$ns at 3.1 "$ns trace-annotate \"FTP stops\""

proc finish {} {

global ns nf f

$ns flush-trace

close $nf

close $f

#puts "filtering..."

#exec tclsh ../bin/namfilter.tcl A1-stop-n-wait.nam

#puts "running nam..."

exec nam A1-stop-n-wait.nam &

exit 0

}

$ns run

TCL Script: Stop and Wait mechanism with packet loss

# stop and wait mechanism with packet loss

# features : labeling, annotation, nam-graph, and window size monitoring

set ns [new Simulator]

set n0 [$ns node]

set n1 [$ns node]

$ns at 0.0 "$n0 label Sender"

$ns at 0.0 "$n1 label Receiver"

set nf [open A2-stop-n-wait-loss.nam w]

$ns namtrace-all $nf

set f [open A2-stop-n-wait-loss.tr w]

$ns trace-all $f

$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail

$ns duplex-link-op $n0 $n1 orient right

$ns duplex-link-op $n0 $n1 queuePos 0.5

$ns queue-limit $n0 $n1 10

Agent/TCP set nam\_tracevar\_ true

set tcp [new Agent/TCP]

$tcp set window\_ 1

$tcp set maxcwnd\_ 1

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns add-agent-trace $tcp tcp

$ns monitor-agent-trace $tcp

$tcp tracevar cwnd\_

$ns at 0.1 "$ftp start"

$ns at 1.3 "$ns queue-limit $n0 $n1 0"

$ns at 1.5 "$ns queue-limit $n0 $n1 10"

$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"

$ns at 3.5 "finish"

$ns at 0.0 "$ns trace-annotate \"Stop and Wait with Packet Loss\""

$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""

$ns at 0.11 "$ns trace-annotate \"Send Packet\_0\""

$ns at 0.35 "$ns trace-annotate \"Receive Ack\_0\""

$ns at 0.56 "$ns trace-annotate \"Send Packet\_1\""

$ns at 0.79 "$ns trace-annotate \"Receive Ack\_1\""

$ns at 0.99 "$ns trace-annotate \"Send Packet\_2\""

$ns at 1.23 "$ns trace-annotate \"Receive Ack\_2 \""

$ns at 1.43 "$ns trace-annotate \"Lost Packet\_3\""

$ns at 1.5 "$ns trace-annotate \"Waiting for Ack\_3\""

$ns at 2.43 "$ns trace-annotate \"Send Packet\_3 again (cause of timeout)\""

$ns at 2.67 "$ns trace-annotate \"Receive Ack\_3\""

$ns at 2.88 "$ns trace-annotate \"Send Packet\_4\""

$ns at 3.1 "$ns trace-annotate \"FTP stops\""

proc finish {} {

global ns nf f

$ns flush-trace

close $nf

close $f

#puts "filtering..."

#exec tclsh ../bin/namfilter.tcl A2-stop-n-wait-loss.nam

#puts "running nam..."

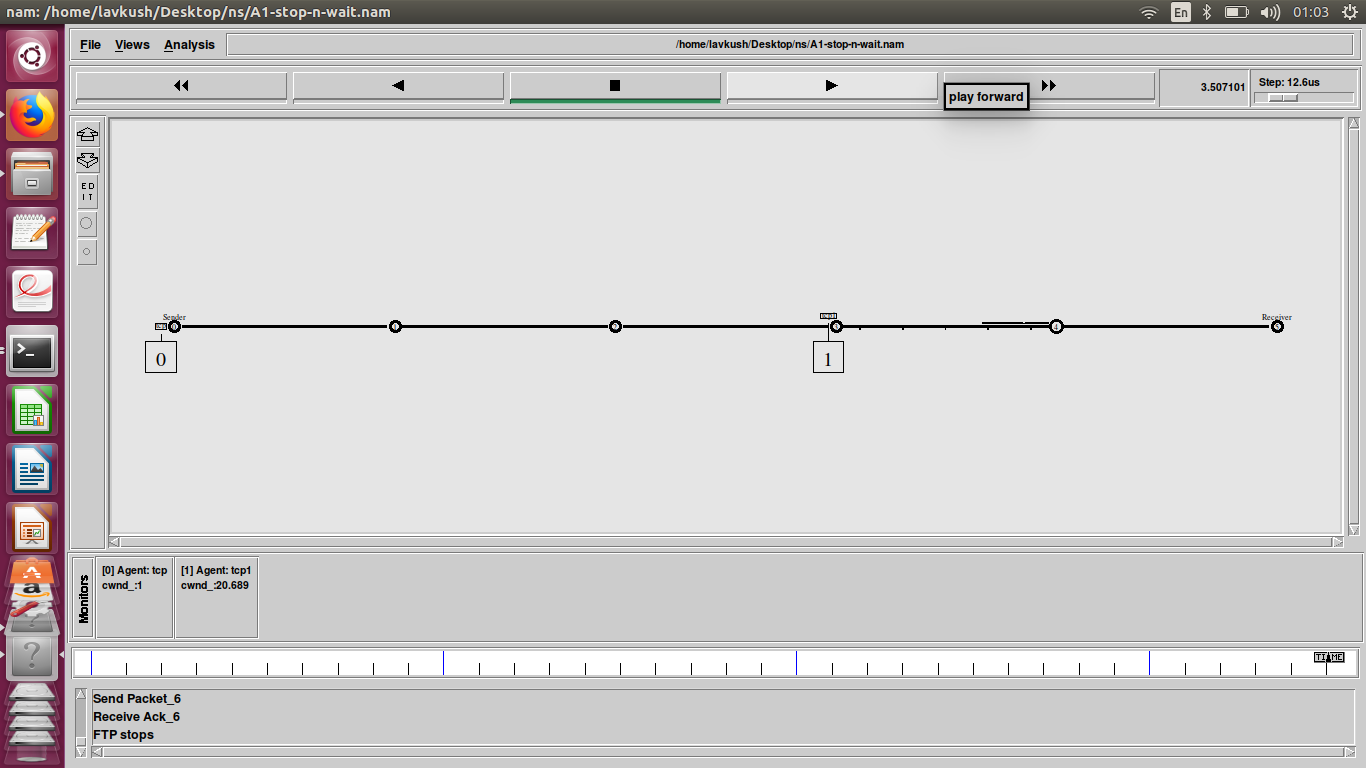
exec nam A2-stop-n-wait-loss.nam &

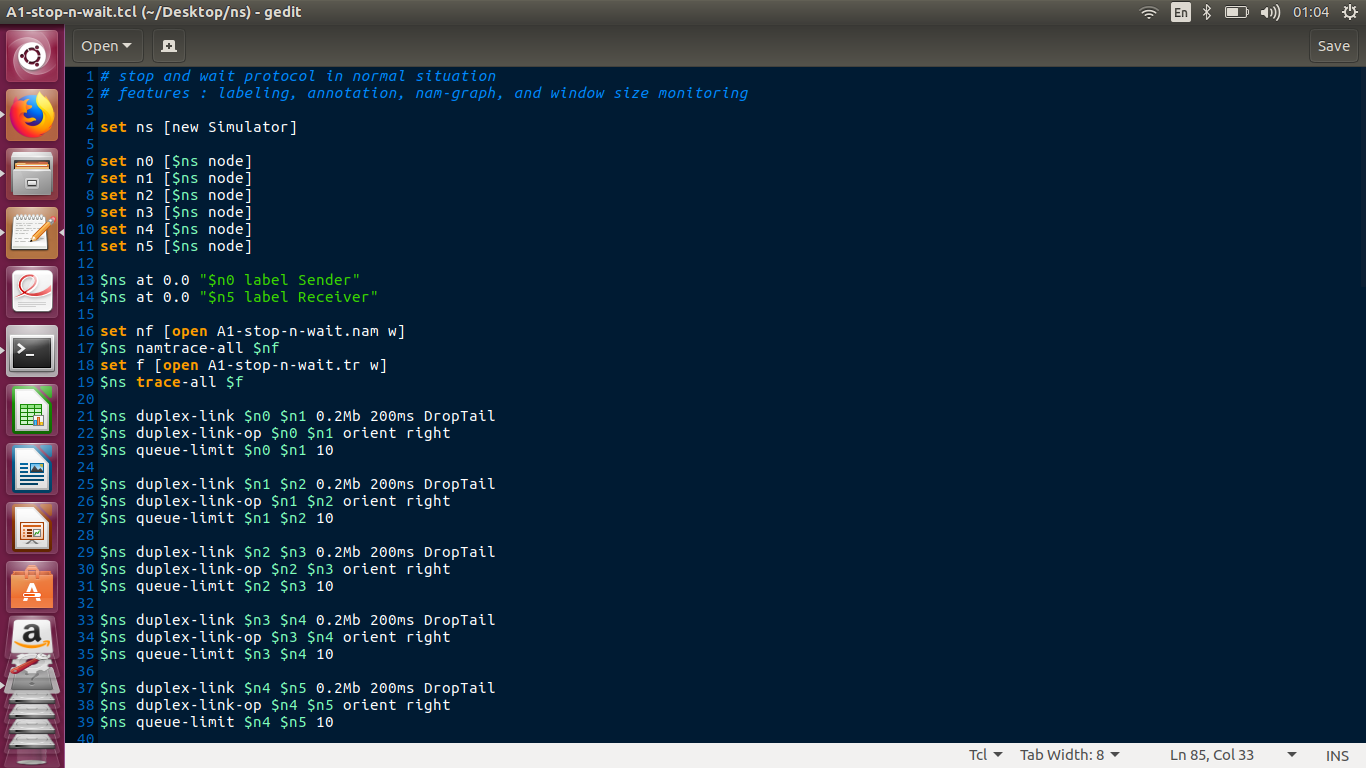
exit 0

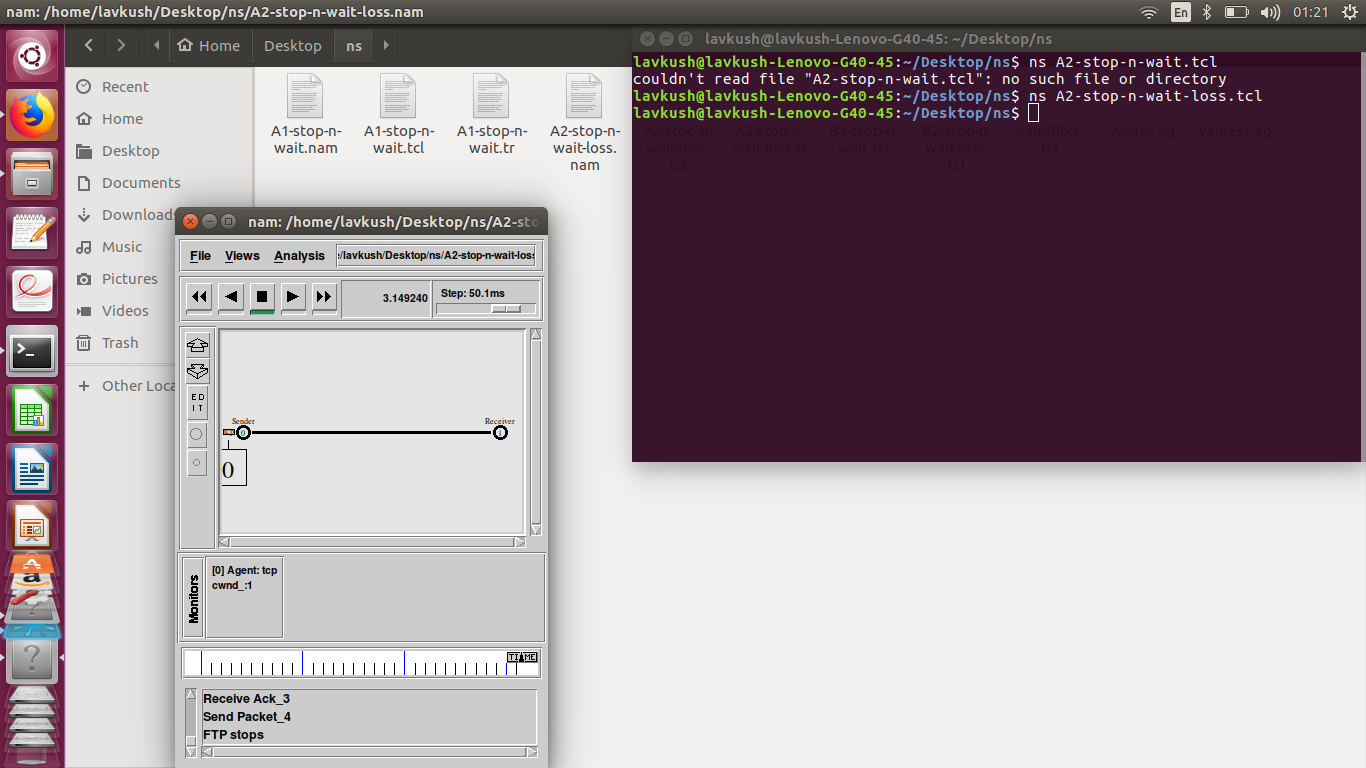
}

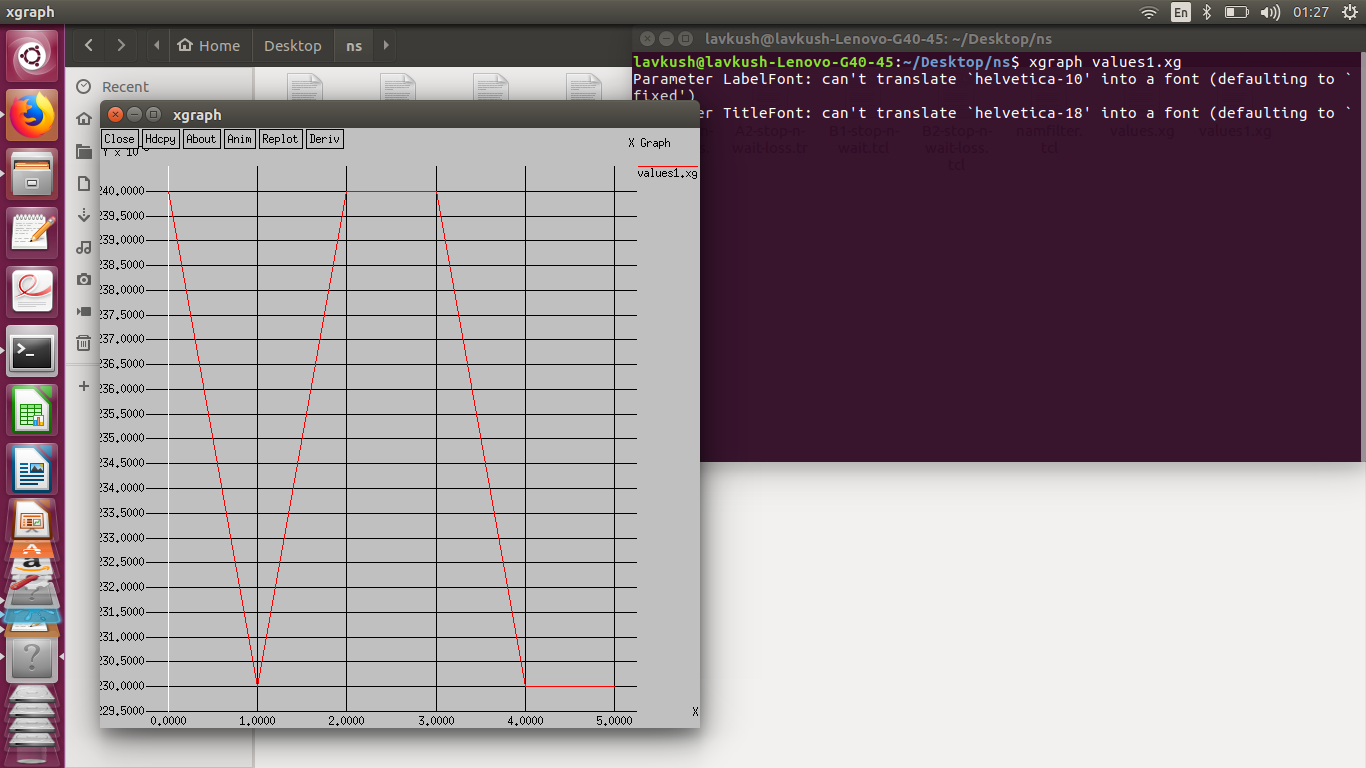
$ns run

Images:









5. References

5.1 TELNET Reference

<https://en.wikipedia.org/wiki/Telnet>

<http://searchnetworking.techtarget.com/definition/Telnet>

5.2 AODV References

<https://www.techopedia.com/definition/2922/ad-hoc-on-demand-distance-vector-aodv>

<http://www.olsr.org/docs/report_html/node16.html>

<http://www.olsr.org/docs/report_html/img6.png>

<https://tools.ietf.org/html/rfc3561>

<https://www.ee.iitb.ac.in/~comlab/seminar/nishant1_Ref1.pdf>

<https://www.webopedia.com/TERM/A/AODV.html>

<https://image.slidesharecdn.com/ismailziko-2003-131215112552-phpapp01/95/comparison-of-routing-protocols-for-ad-hoc-wireless-network-with-medical-data-13-638.jpg?cb=1387106833>

5.3 Stop and wait References

<https://en.wikipedia.org/wiki/Stop-and-wait_ARQ>

<http://www.geeksforgeeks.org/stop-and-wait-arq/>

<https://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_HYUNAH/D-Research/stop-n-wait.html>

6. AWK files:

6.1 TELNET (Part A):

a) Throughput AWK :

BEGIN {

rcvpkt1 = 0;

rcvpkt2 = 0;

rcvpkt3 = 0;

rcvpkt5 = 0;

rcvpkt6 = 0;

rcvpkt7 = 0;

rcvpkt4 = 0;

rcvpkt8 = 0;

rcvpkt9 = 0;

starttime1 = 0;

starttime2 = 0;

starttime3 = 0;

starttime4 = 0;

starttime5 = 0;

starttime6 = 0;

starttime7 = 0;

starttime8 = 0;

starttime9 = 0;

endtime1 = 0;

endtime2 = 0;

endtime3 = 0;

endtime4 = 0;

endtime5 = 0;

endtime6 = 0;

endtime7 = 0;

endtime8 = 0;

endtime9 = 0;

flag1 = 0;

flag2 = 0;

flag3 = 0;

flag4 = 0;

flag5 = 0;

flag6 = 0;

flag7 = 0;

flag8 = 0;

flag9 = 0;

}

{

event = $1;

time = $2;

from = $3;

to = $4;

type = $5;

size = $6;

flowid = $8;

src = $9;

dest = $10;

seq = $11;

pid = $12;

if(flowid == 1 && event == "r" && to == 9)

{

rcvpkt1++;

endtime1 = time;

}

if(flowid == 1 && flag1 == 0 && from == 0)

{

starttime1 = time;

flag1 = 1;

}

if(flowid == 2 && event == "r" && to == 5)

{

rcvpkt2++;

endtime2 = time;

}

if(flowid == 2 && flag2 == 0 && from == 7)

{

starttime2 = time;

flag2 = 1;

}

if(flowid == 3 && event == "r" && to == 13)

{

rcvpkt3++;

endtime3 = time;

}

if(flowid == 3 && flag3 == 0 && from == 11)

{

starttime3 = time;

flag3 = 1;

}

if(flowid == 4 && event == "r" && to == 6)

{

rcvpkt4++;

endtime4 = time;

}

if(flowid == 4 && flag4 == 0 && from == 3)

{

starttime4 = time;

flag4 = 1;

}

if(flowid == 5 && event == "r" && to == 12)

{

rcvpkt5++;

endtime5 = time;

}

if(flowid == 5 && flag5 == 0 && from == 8)

{

starttime5 = time;

flag5 = 1;

}

if(flowid == 6 && event == "r" && to == 2)

{

rcvpkt6++;

endtime6 = time;

}

if(flowid == 6 && flag6 == 0 && from == 9)

{

starttime6 = time;

flag6 = 1;

}

if(flowid == 7 && event == "r" && to == 8)

{

rcvpkt7++;

endtime7 = time;

}

if(flowid == 7 && flag7 == 0 && from == 10)

{

starttime7 = time;

flag7 = 1;

}

if(flowid == 8 && event == "r" && to == 3)

{

rcvpkt8++;

endtime8 = time;

}

if(flowid == 8 && flag8 == 0 && from == 13)

{

starttime8 = time;

flag8 = 1;

}

if(flowid == 9 && event == "r" && to == 4)

{

rcvpkt9++;

endtime9 = time;

}

if(flowid == 9 && flag9 == 0 && from == 12)

{

starttime9 = time;

flag9 = 1;

}

}

END {

#telnet1 = (rcvpkt1)/(endtime1-starttime1);

#thrtelnet1 = (rcvpkt1\*size)/(endtime1-starttime1);

#telnet2 = (rcvpkt2)/(endtime2-starttime2);

#thrtelnet2 = (rcvpkt2\*size)/(endtime2-starttime2);

#telnet3 = (rcvpkt3)/(endtime3-starttime3);

#thrtelnet3 = (rcvpkt3\*size)/(endtime3-starttime3);

#cbr1 = (rcvpkt4)/(endtime4-starttime4);

#thrcbr1 = (rcvpkt4\*size)/(endtime4-starttime4);

#cbr2 = (rcvpkt5)/(endtime5-starttime5);

#thrcbr2 = (rcvpkt5\*size)/(endtime5-starttime5);

#cbr3 = (rcvpkt6)/(endtime6-starttime6);

#thrcbr3 = (rcvpkt6\*size)/(endtime6-starttime6);

ftp1 = (rcvpkt7)/(endtime7-starttime7);

thrftp1 = (rcvpkt7\*size)/(endtime7-starttime7);

#ftp2 = (rcvpkt8)/(endtime8-starttime8);

#thrftp2 = (rcvpkt8\*size)/(endtime8-starttime8);

#ftp3 = (rcvpkt9)/(endtime9-starttime9);

#thrftp3 = (rcvpkt9\*size)/(endtime9-starttime9);

#thravg = (thrtelnet1 + thrtelnet2 + thrtlnet3)/3;

#thravg = (thrcbr1 + thrcbr2 +thrcbr3)/3;

#thravg = (thrftp1 + thrftp2 + thrftp3)/3;

#thravg = (thrtelnet1 + thrtelnet2)/2;

#thravg = (thrcbr1 + thrcbr2)/2;

#thravg = (thrftp1 + thrftp2)/2;

#thravg = (thrtelnet1)/1;

#thravg = (thrcbr1)/1;

thravg = (thrftp1)/1;

printf("\nFor 1 FTP connection in the Network");

#printf("\nPacket delivery rate = %d",telnet1);

#printf("\nThroughput: %d ",thrtelnet1);

#printf("\nPacket delivery rate = %d",telnet2);

#printf("\nThroughput: %d ",thrtelnet2);

#printf("\nPacket delivery rate = %d",telnet3);

#printf("\nThroughput: %d ",thrtelet3);

#printf("\nPacket delivery rate = %d",cbr1);

#printf("\nThroughput: %d ",thrcbr1);

#printf("\nPacket delivery rate = %d",cbr2);

#printf("\nThroughput: %d ",thrcbr2);

#printf("\nPacket delivery rate = %d",cbr3);

#printf("\nThroughput: %d ",thrcbr3);

printf("\nPacket delivery rate = %d",ftp1);

printf("\nThroughput: %d ",thrftp1);

#printf("\nPacket delivery rate = %d",ftp2);

#printf("\nThroughput: %d ",thrftp2);

#printf("\nPacket delivery rate = %d",ftp3);

#printf("\nThroughput: %d ",thrftp3);

printf("\nAverage Throughput: %d \n\n",thravg);

}

6.2 AODV (Part B)

a) End to End Delay:

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

# AWK Script for calculating:

# => Average End-to-End Delay.

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

BEGIN {

seqno = -1;

# droppedPackets = 0;

# receivedPackets = 0;

count = 0;

}

{

if($4 == "AGT" && $1 == "s" && seqno < $6) {

seqno = $6;

}

# else if(($4 == "AGT") && ($1 == "r")) {

# receivedPackets++;

# } else if ($1 == "D" && $7 == "tcp" && $8 > 512){

# droppedPackets++;

# }

#end-to-end delay

if($4 == "AGT" && $1 == "s") {

start\_time[$6] = $2;

} else if(($7 == "cbr") && ($1 == "r")) {

end\_time[$6] = $2;

} else if($1 == "D" && $7 == "cbr") {

end\_time[$6] = -1;

}

}

END {

for(i=0; i<=seqno; i++) {

if(end\_time[i] > 0) {

delay[i] = end\_time[i] - start\_time[i];

count++;

}

else

{

delay[i] = -1;

}

}

for(i=0; i<=seqno; i++) {

if(delay[i] > 0) {

n\_to\_n\_delay = n\_to\_n\_delay + delay[i];

}

}

n\_to\_n\_delay = n\_to\_n\_delay/count;

print "\n";

# print "GeneratedPackets = " seqno+1;

# print "ReceivedPackets = " receivedPackets;

# print "Packet Delivery Ratio = " receivedPackets/(seqno+1)\*100

#"%";

# print "Total Dropped Packets = " droppedPackets;

print "Average End-to-End Delay = " n\_to\_n\_delay \* 1000 " ms";

print "\n";

}

b) Throughput.awk

BEGIN {

recvdSize = 0

txsize=0

drpSize=0

startTime = 0

stopTime = 0

thru=0

}

{

event = $1

time = $2

node\_id = $3

pkt\_size = $8

level = $4

# Store start time

if (level == "AGT" && event == "s" ) {

if (time < startTime) {

startTime = time

}

# hdr\_size = pkt\_size % 400

# pkt\_size -= hdr\_size

# Store transmitted packet’s size

txsize++;

}

# Update total received packets’ size and store packets arrival time

if (level == "AGT" && event == "r" ) {

if (time > stopTime) {

stopTime = time

}

# Rip off the header

# hdr\_size = pkt\_size % 400

# pkt\_size -= hdr\_size

# Store received packet’s size

recvdSize++

# thru=(recvdSize/txsize)

# printf(” %.2f %.2f \n” ,time,thru)>”tru2.tr”

}

if (level == "AGT" && event == "D" ) {

# hdr\_size = pkt\_size % 400

# pkt\_size -= hdr\_size

# Store received packet’s size

drpSize++

}

}

END {

printf("Average Throughput[kbps] = %.2f\ns=%.2f\nd=%.2f\nr=%.2f\nStartTime=%.2f\nStopTime=%.2f\n",(recvdSize/(stopTime-startTime)),txsize,drpSize,recvdSize,startTime,stopTime)

}

c) Route Request and Route Reply

BEGIN {

route\_requested = 0;

route\_replied = 0;

outRequest = "Request.log";

outReply = "Reply.log";

}

{

req = $25;

reply = $23;

time = $2;

if ( req == "(REQUEST)" ) {

route\_requested++;

}

if ( reply == "(REPLY)" ) {

route\_replied++;

}

}

END {

printf("\n\nNumber of Route Request : %d\n", route\_requested);

printf("\nNumber of Route Replies recieved : %d\n\n" , route\_replied);

}

d) PDR ad Sent/Received Packets

# Initialization settings

BEGIN {

droppacket=0;

sendLine = 0;

recvLine = 0;

fowardLine = 0;

if(mseq==0)

mseq=10000;

for(i=0;i<mseq;i++){

rseq[i]=-1;

sseq[i]=-1;

}

}

# Applications received packet

$0 ~/^s.\* AGT/ {

# if(sseq[$6]==-1){

sendLine ++ ;

# sseq[$6]=$6;

# }

}

# Applications to send packets

$0 ~/^r.\* AGT/{

# if(rreq[$6]==-1){

recvLine ++ ;

# sseq[$6]=$6;

# }

}

# Routing procedures to forward the packet

$0 ~/^f.\* RTR/ {

fowardLine ++ ;

}

#150

#170

#300

#400

#500

# Final output

END {

droppacket=sendLine-recvLine

printf "cbr Send Packets:%d\nrecieve Packets:%d\nPacket Delivery Ratio :%.4f\nNumber of packet forwarded:%d\nNumber of Packet dropped=%d\n", sendLine, recvLine, (recvLine/sendLine), fowardLine ,droppacket;

}

6.3