# Department of Computer Science

**EE353: Computer Networks**

**Class: BESE-13AB**

**Lab 10: Extracting Information from trace files**

**Lab Engineer: Syed Muhammad Ali Musa**

**Name:** Aimen Munawar

**Class**: BESE-13-A

**CMS ID:** 415867

**Lab 10: Extracting Information from trace files**

**Introduction**

Network Simulator 2 (ns2) is a very popular network simulation software for linux operating system. It works really good on the Ubuntu operating system.

**Objectives**

The main target of this lab is to process data from trace files.

**Tools/Software Requirement**

NS2

**Description**

**Awk- An Advanced**

AWK Scripts are very good in processing the data from the log (trace files) which we get from NS2. If you want to process the trace file manually, here is the detail

Here is a sample of trace file from NS2 (However ns2 supports a new type of trace file also), but this post will make you understand the old trace format only.

**r 0.030085562 \_0\_ MAC  --- 0 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]**

**r 0.030110562 \_0\_ RTR  --- 0 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]**

**s 1.119926192 \_0\_ RTR  --- 1 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]**AWK Scripts are very good in processing the data column wise. For example  
the first column in the above trace file represents r, s which indicates receive, sent respectively. If we want to trace the entire r and s alone from this trace file we can represent it as $1  
So  
$1 represents ACTION  
$2 Time  
$3 Node ID  
$4 Layer  
$5 Flags  
$6 seqno  
$7 type  
$8 Size  
$14 Energy (if the network nodes includes EnergyModel)

To run the awk script in Linux,

**gawk –f filename.awk filename.tr**

So, it is necessary for the researchers to know the basics of awk scripts before they are used.  
Here this post will let you know some of the scripts that were used to process the data

**To find the throughput of the Network**

BEGIN {

2: recvdSize = 0

3: startTime = 400

4: stopTime = 0

5: }

6:

7: {

8: event = $1

9: time = $2

10: node\_id = $3

11: pkt\_size = $8

12: level = $4

13:

14: # Store start time

15: if (level == "AGT" &;& event == "s" && pkt\_size >= 512) {

16: if (time <; startTime) {

17: startTime = time

18: }

19: }

20:

21: # Update total received packets' size and store packets arrival time

22: if (level == "AGT" &;& event == "r" && pkt\_size >= 512) {

23: if (time >; stopTime) {

24: stopTime = time

25: }

26: # Rip off the header

27: hdr\_size = pkt\_size % 512

28: pkt\_size -= hdr\_size

29: # Store received packet's size

30: recvdSize += pkt\_size

31: }

32: }

33:

34: END {

35: printf("Average Throughput[kbps] = %.2f\t\t StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/(stopTime-startTime))\*(8/1000),startTime,stopTime)

36: }

**To print the Congestion window size**

1: BEGIN {

2:

3: }

4: {

5: if($6=="cwnd\_") {

6: printf("%f\t%f\n",$1,$7);

7: }

8: }

9: END {

10:

11: }

**To print packet Delivery ratio**

1: BEGIN {

2: sendLine = 0;

3: recvLine = 0;

4: fowardLine = 0;

5: }

6:

7: $0 ~/^s.\* AGT/ {

8: sendLine ++ ;

9: }

10:

11: $0 ~/^r.\* AGT/ {

12: recvLine ++ ;

13: }

14:

15: $0 ~/^f.\* RTR/ {

16: fowardLine ++ ;

17: }

18:

19: END {

20: printf "cbr s:%d r:%d, r/s Ratio:%.4f, f:%d \n", sendLine, recvLine, (recvLine/sendLine),fowardLine;

21: }

22:

**AWK Script for calculating the Send, Received, Dropped Packets, Received Packets, Packet Delivery Ratio and Average end to End Delay**

1: BEGIN {

2: seqno = -1;

3: droppedPackets = 0;

4: receivedPackets = 0;

5: count = 0;

6: }

7: {

8: #packet delivery ratio

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

10: seqno = $6;

11: } else if(($4 == "AGT") && ($1 == "r")) {

12: receivedPackets++;

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

14: droppedPackets++;

15: }

16: #end-to-end delay

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

18: start\_time[$6] = $2;

19: } else if(($7 == "tcp") && ($1 == "r")) {

20: end\_time[$6] = $2;

21: } else if($1 == "D" && $7 == "tcp") {

22: end\_time[$6] = -1;

23: }

24: }

25:

26: END {

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

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

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

30: count++;

31: }

32: else

33: {

34: delay[i] = -1;

35: }

36: }

37: for(i=0; i<count; i++) {

38: if(delay[i] >; 0) {

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

40: }

41: }

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

43: print "\n";

44: print "GeneratedPackets = " seqno+1;

45: print "ReceivedPackets = " receivedPackets;

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

47: "%";

48: print "Total Dropped Packets = " droppedPackets;

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

50: print "\n";

51: }

**Example: The following is a wireless network code, name it as a .tcl file and run it using “ns wireless.tcl” (without quotes), a trace file called wireless\_mitf.tr will be created.**

1: set val(chan) Channel/WirelessChannel ;#Channel Type

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

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

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

5: set val(ifq) Queue/DropTail/PriQueue ;# interface queue type

6: set val(ll) LL ;# link layer type

7: set val(ant) Antenna/OmniAntenna ;# antenna model

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

9: set val(nn) 2 ;# number of mobilenodes

10: set val(rp) DSDV ;# routing protocol

11: #set val(rp) DSR ;# routing protocol

12: set val(x) 500

13: set val(y) 500

14:

15: # Initialize Global Variables

16: set ns\_ [new Simulator]

17: set tracefd [open wireless\_mitf.tr w]

18: $ns\_ trace-all $tracefd

19:

20: set namtrace [open wireless\_mitf.nam w]

21: $ns\_ namtrace-all-wireless $namtrace $val(x) $val(y)

22:

23: # set up topography object

24: set topo [new Topography]

25:

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

27:

28: # Create God

29: create-god $val(nn)

30:

31: # New API to config node:

32: # 1. Create channel (or multiple-channels);

33: # 2. Specify channel in node-config (instead of channelType);

34: # 3. Create nodes for simulations.

35:

36: # Create channel #1 and #2

37: set chan\_1\_ [new $val(chan)]

38: set chan\_2\_ [new $val(chan)]

39:

40: # Create node(0) "attached" to channel #1

41:

42: # configure node, please note the change below.

43: $ns\_ node-config -adhocRouting $val(rp) \

44: -llType $val(ll) \

45: -macType $val(mac) \

46: -ifqType $val(ifq) \

47: -ifqLen $val(ifqlen) \

48: -antType $val(ant) \

49: -propType $val(prop) \

50: -phyType $val(netif) \

51: -topoInstance $topo \

52: -agentTrace ON \

53: -routerTrace ON \

54: -macTrace ON \

55: -movementTrace OFF \

56: -channel $chan\_1\_

57:

58: set node\_(0) [$ns\_ node]

59:

60: # node\_(1) can also be created with the same configuration, or with a different

61: # channel specified.

62: # Uncomment below two lines will create node\_(1) with a different channel.

63: # $ns\_ node-config \

64: # -channel $chan\_2\_

65: set node\_(1) [$ns\_ node]

66:

67: $node\_(0) random-motion 0

68: $node\_(1) random-motion 0

69:

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

71: $ns\_ initial\_node\_pos $node\_($i) 20

72: }

73:

74: #

75: # Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes

76: #

77: $node\_(0) set X\_ 5.0

78: $node\_(0) set Y\_ 2.0

79: $node\_(0) set Z\_ 0.0

80:

81: $node\_(1) set X\_ 8.0

82: $node\_(1) set Y\_ 5.0

83: $node\_(1) set Z\_ 0.0

84:

85: #

86: # Now produce some simple node movements

87: # Node\_(1) starts to move towards node\_(0)

88: #

89: $ns\_ at 3.0 "$node\_(1) setdest 50.0 40.0 25.0"

90: $ns\_ at 3.0 "$node\_(0) setdest 48.0 38.0 5.0"

91:

92: # Node\_(1) then starts to move away from node\_(0)

93: $ns\_ at 20.0 "$node\_(1) setdest 490.0 480.0 30.0"

94:

95: # Setup traffic flow between nodes

96: # TCP connections between node\_(0) and node\_(1)

97:

98: set tcp [new Agent/TCP]

99: $tcp set class\_ 2

100: set sink [new Agent/TCPSink]

101: $ns\_ attach-agent $node\_(0) $tcp

102: $ns\_ attach-agent $node\_(1) $sink

103: $ns\_ connect $tcp $sink

104: set ftp [new Application/FTP]

105: $ftp attach-agent $tcp

106: $ns\_ at 3.0 "$ftp start"

107:

108: #

109: # Tell nodes when the simulation ends

110: #

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

112: $ns\_ at 30.0 "$node\_($i) reset";

113: }

114: $ns\_ at 30.0 "stop"

115: $ns\_ at 30.01 "puts \"NS EXITING...\" ; $ns\_ halt"

116: proc stop {} {

117: global ns\_ tracefd

118: $ns\_ flush-trace

119: close $tracefd

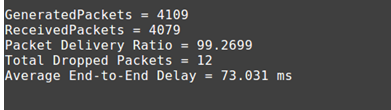
120: }

121:

122: puts "Starting Simulation..."

123: $ns\_ run

**Run the tracefile (wireless\_mitf.tr, which will be created when the above TCL program runs) as given in the screenshot.**



**References**

The above scripts were taken from [http://ns2ultimate.com](http://ns2ultimate.com/) and [http://elmurod.net](http://elmurod.net/) and through search engines.

**Lab Task:**

Do all above tasks

**Deliverables**

Upload TCL file with trace file and snap of NAM file.

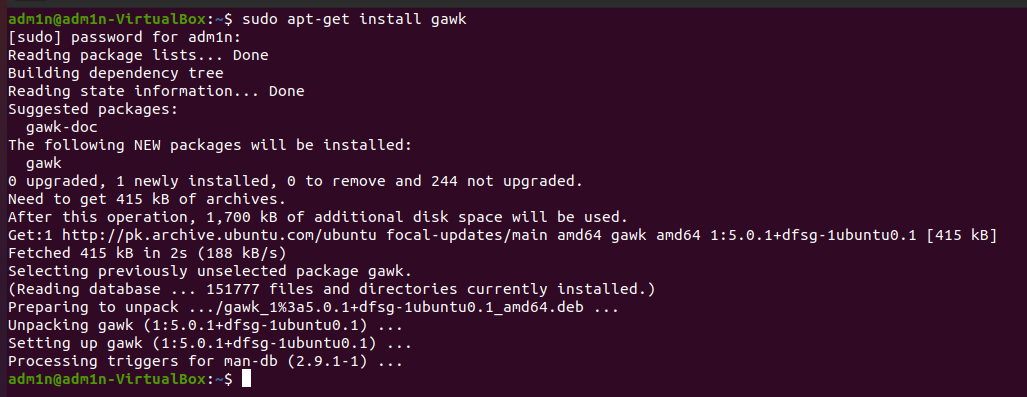
**Grade Criteria**

This lab is graded. Min marks: 0. Max marks: 10.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Minimum** | **Maximum** |
| Documentation with clearly defined understanding of the lab task and approach | Fail | Pass |
| Lab Tasks | 0 | 10 |

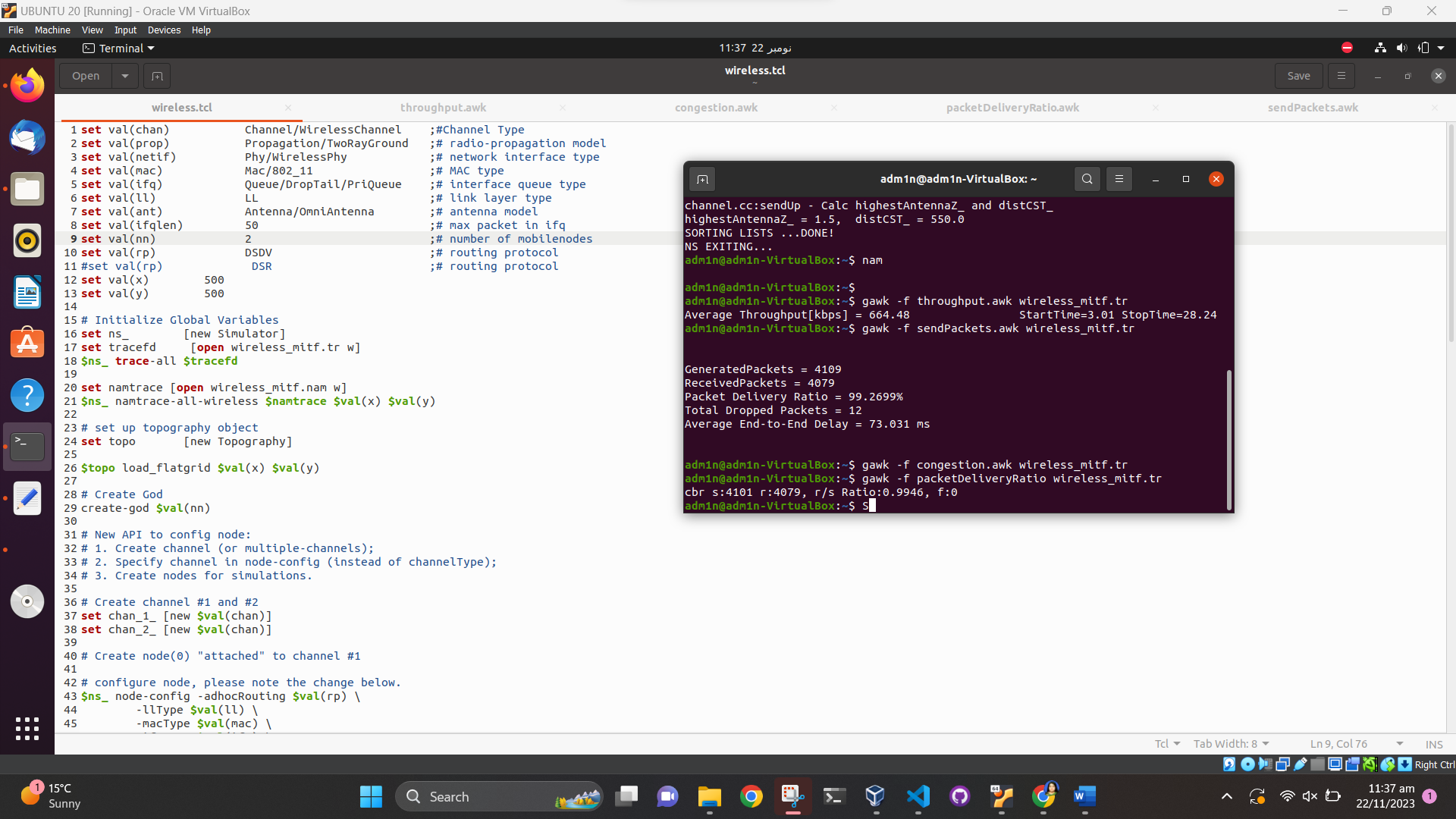
**Step 1: Install gawk using apt-get.**

**Command:** Sudo apt-get install gawk



**Step 2: Creating wireless.tcl file.**

**Command:** gedit wireless.tcl

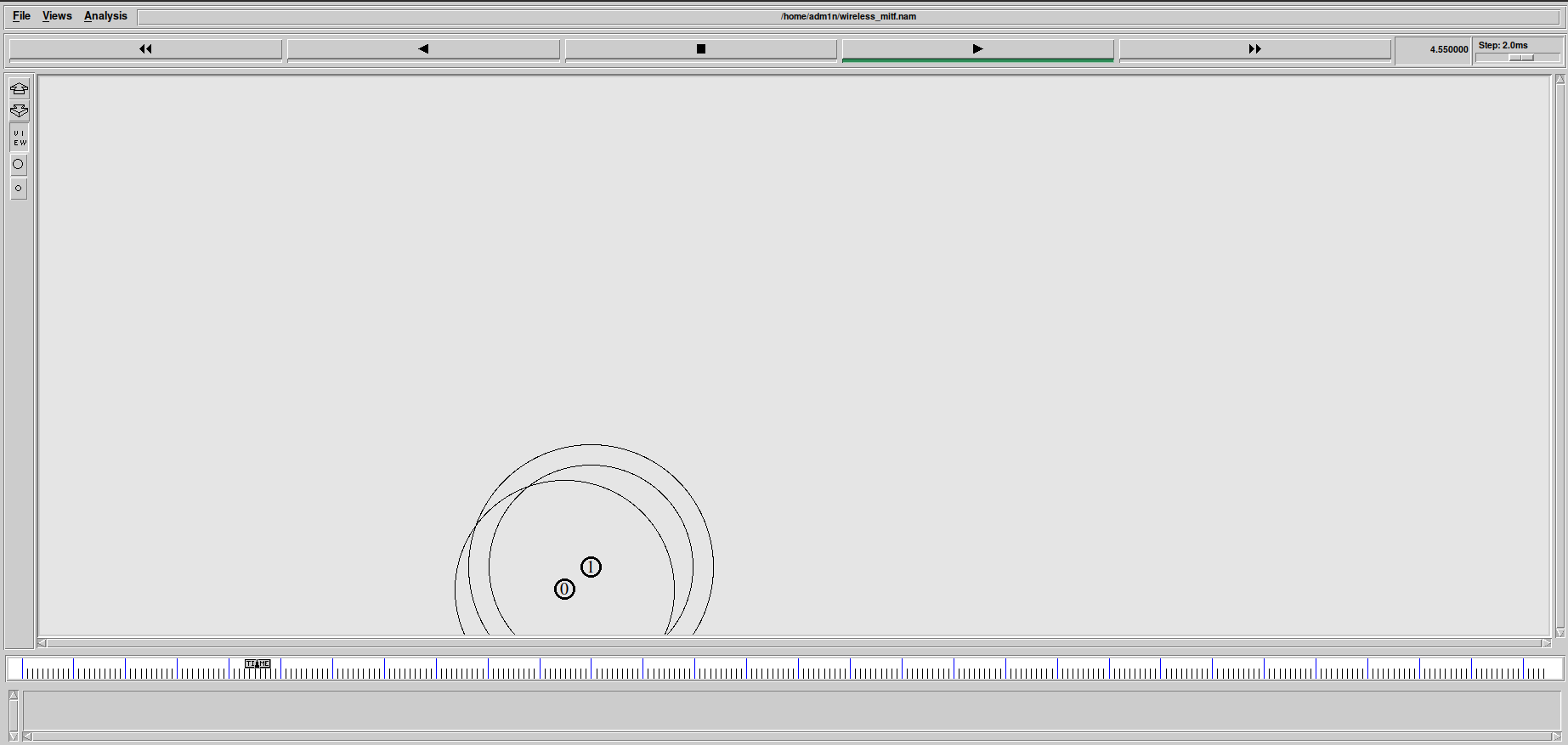
****

**Step 3: Running wirelss.tcl file.**

**A screenshot of a computer

Description automatically generatedCommand:** ns wireless.tcl

**Step 4: Running the NAM file**



A black and white image of a circle

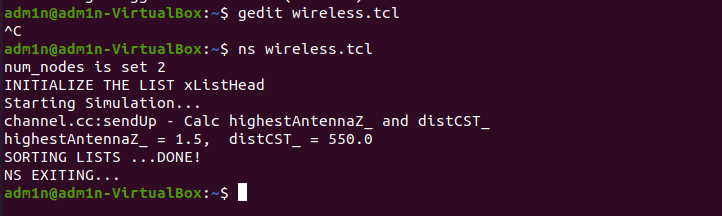
Description automatically generated

**Step 5: Running the wireless.tcl file**

**Commands:**

gedit wireless.tcl

ns wirless.tcl

****

**Step 6: Running AWK Script for throughput, Congestion window size, Packet Delivery Ratio, for calculating the Send, Received, Dropped Packets, Received Packets, Packet Delivery Ratio and Average end to End Delay on given wireless.tcl file.**

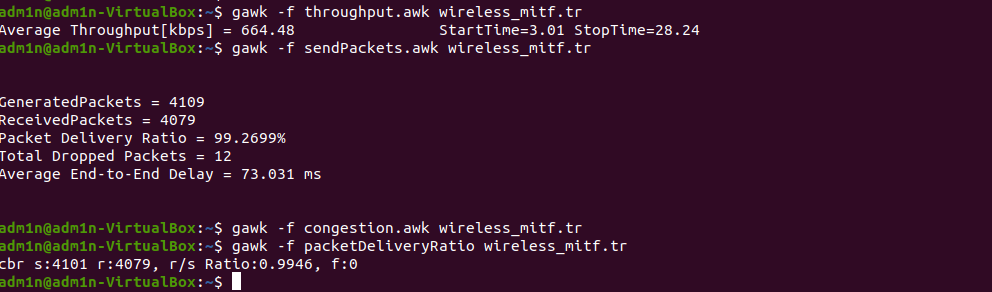
**Commands:**

gawk -f throughput.awk wireless\_mitf.tr

gawk -f sendPackets.awk wireless\_mitf.tr

gawk -f congestion.awk wireless\_mitf.tr

gawk -f packetDeliveryRatio.awk wireless\_mitf.tr



**Overall Screenshot:**

