# Department of Computer Science

**EE353: Computer Networks**

**Class: BESE-13AB**

**Lab 11: Generate TCL Script for NS2 using Java based Tool (NSG)**

**Lab Engineer: Syed Muhammad Ali Musa**

**Name:** Aimen Munawar

**Class:** BESE-13-A

**CMS ID:** 415867

**Lab 11: Generate TCL Script for NS2 using Java based Tool (NSG)**

**Introduction**

The Network Simulation Generator (NSG) tool is a powerful utility for enhancing network simulations performed using NS2. This lab focuses on utilizing NSG to streamline the process of creating network scenarios and generating trace files in NS2.

**Objectives**

The primary objectives of this lab include:

Understanding the role of NSG in NS2 simulations.

Generating network scenarios using NSG.

Creating and customizing trace files with NSG.

Analyzing and processing trace files using AWK scripts**.**

**Tools/Software Requirement**

NS2

NSG (Network Simulation Generator)

**Description**

NSG simplifies the process of creating complex network scenarios by providing a graphical interface for designing topologies and configuring parameters. It seamlessly integrates with NS2 to enhance the simulation experience.

For NSG Installation

<https://sites.google.com/site/pengjungwu/nsg2>

**Detailed Lab Demonstration**

**Using NSG for Scenario Generation**

Open NSG and create a new project.

Design a network topology by adding nodes, links, and defining node properties.

Configure network parameters such as channel models, propagation models, and mobility models.

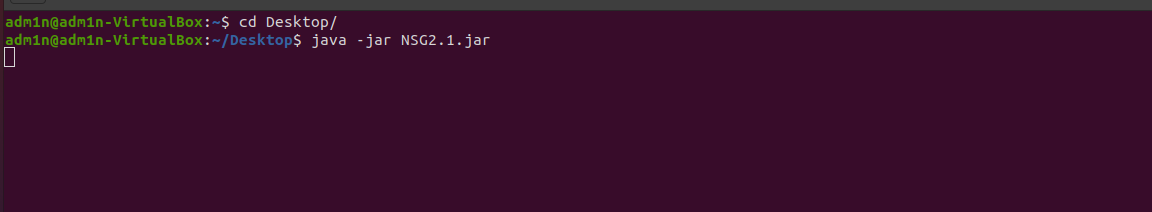
Save the generated scenario.

<https://youtu.be/YSb63Z29rs0?si=MUJdeW4IXwFIqH0J>

**Lab Task**

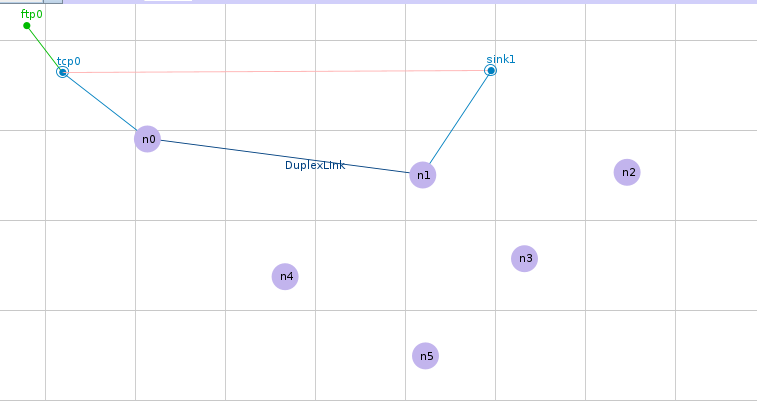
**Perform the following tasks:**

* Install NSG



A computer screen shot of a blue screen

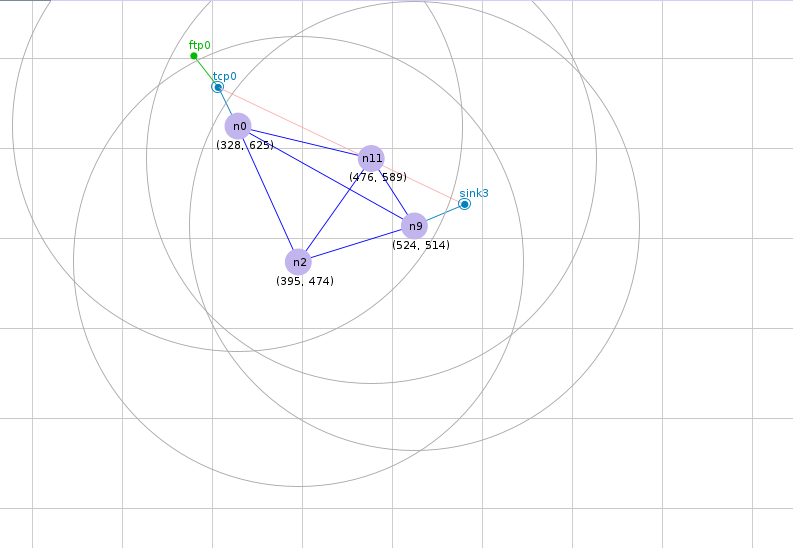
Description automatically generated



A graph with circles and numbers

Description automatically generated

* Create a network scenario using NSG.
* Generate trace files by running a simulation using NS2



A screenshot of a computer

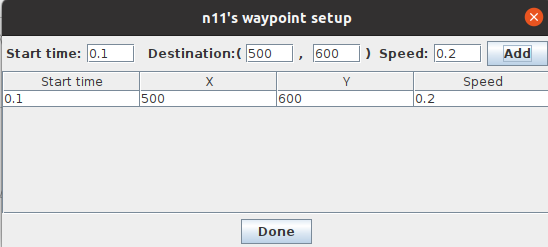
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



**Wirelesssimulation.tcl file Code:**

# This script is created by NSG2 beta1

# <http://wushoupong.googlepages.com/nsg>

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

# Simulation parameters setup

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

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

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

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

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

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

set val(ll) LL ;# link layer type

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

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

set val(nn) 4 ;# number of mobilenodes

set val(rp) DSDV ;# routing protocol

set val(x) 682 ;# X dimension of topography

set val(y) 800 ;# Y dimension of topography

set val(stop) 10.0 ;# time of simulation end

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

# Initialization

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

#Create a ns simulator

set ns [new Simulator]

#Setup topography object

set topo [new Topography]

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

create-god $val(nn)

#Open the NS trace file

set tracefile [open testing2.tr w]

$ns trace-all $tracefile

#Open the NAM trace file

set namfile [open testing2.nam w]

$ns namtrace-all $namfile

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

set chan [new $val(chan)];#Create wireless channel

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

# Mobile node parameter setup

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

$ns node-config -adhocRouting $val(rp) \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channel $chan \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace ON \

-movementTrace ON

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

# Nodes Definition

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

#Create 4 nodes

set n0 [$ns node]

$n0 set X\_ 328

$n0 set Y\_ 625

$n0 set Z\_ 0.0

$ns initial\_node\_pos $n0 20

set n2 [$ns node]

$n2 set X\_ 395

$n2 set Y\_ 474

$n2 set Z\_ 0.0

$ns initial\_node\_pos $n2 20

set n9 [$ns node]

$n9 set X\_ 524

$n9 set Y\_ 514

$n9 set Z\_ 0.0

$ns initial\_node\_pos $n9 20

set n11 [$ns node]

$n11 set X\_ 476

$n11 set Y\_ 589

$n11 set Z\_ 0.0

$ns initial\_node\_pos $n11 20

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

# Generate movement

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

$ns at 0.1 " $n0 setdest 400 700 0.2 "

$ns at 0.1 " $n2 setdest 430 500 0.2 "

$ns at 0.1 " $n9 setdest 540 530 0.2 "

$ns at 0.1 " $n11 setdest 500 600 0.2 "

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

# Agents Definition

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

#Setup a TCP connection

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink3 [new Agent/TCPSink]

$ns attach-agent $n9 $sink3

$ns connect $tcp0 $sink3

$tcp0 set packetSize\_ 1500

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

# Applications Definition

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

#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 2.0 "$ftp0 stop"

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

# Termination

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

# Define a 'finish' procedure

proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile

close $namfile

exec nam testing2.nam &

# Reset all nodes

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

set nodeVar "\$n$i"

$ns at $val(stop) "$nodeVar reset"

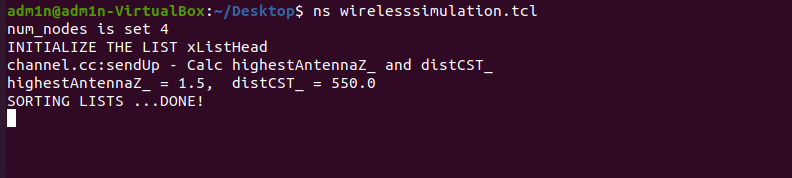
}

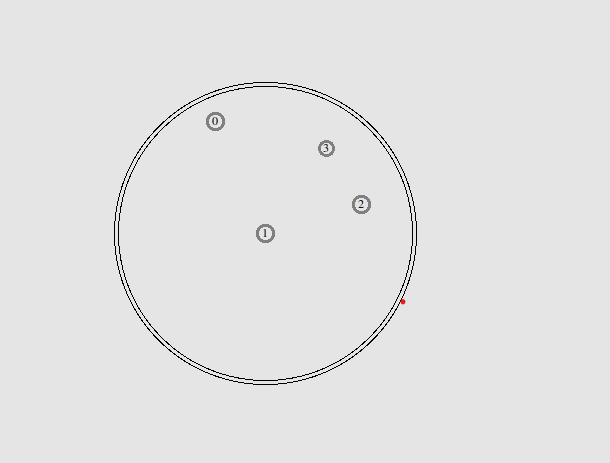
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "puts \"done\" ; $ns halt"

}

$ns run





* Apply the provided AWK scripts in your previous Lab 10 to analyze trace files.

**AWK Scripts:**

**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

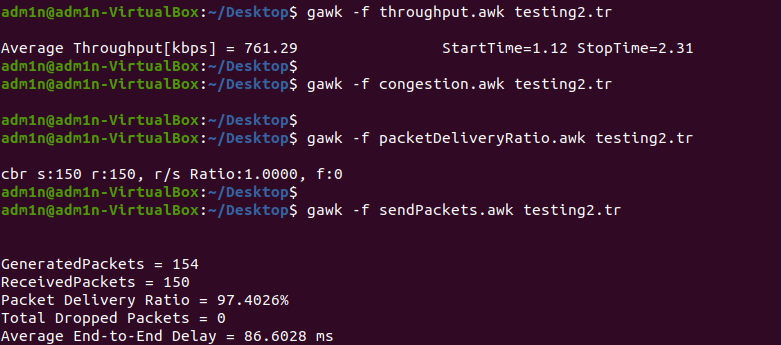
47: "%";

48: print "Total Dropped Packets = " droppedPackets;

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

50: print "\n";

51: }



**Deliverables**

Upload the NSG project file (.nsg) containing the network scenario along with your generated TCL Script and Results SS by using AWK scripts. Also Include a screenshot of the NAM file for visualization.