MIT World Peace University

Wireless Devices and Mobile Security
Third Year B. Tech, Semester 5

Simulation of Two Network Nodes in NS2

Lab Assignment 2

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

September 24, 2023

Contents

1 Aim

Write a program to simulate two node wireless network. You may use NetSimor NS2 or QualNet for this experiment.

2 Theory

2.1 The Wireless Model

2.2 Routing Protocols for Wireless Devices

2.3 Creating Mobile Nodes

```
for { set j 0 } { $j < $val(nn)} {incr j} {
    set node_($j) [ $ns_ node ]
    $node_($i) random-motion 0 ;# disable random motion
    }
```

2.4 Setting Mobile Node Movements

```
$node set X_ <x1>
$node set Y_ <y1>
$node set Z_ <z1>
$ns at $time $node setdest <x2> <y2> <speed>
```

2.5 Topology Definition

```
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
where val(x) and val(y) are the boundaries used in simulation .
```

2.6 Network Components in Mobile node:

The network stack for a mobilenode consists of a link layer(LL), an ARP module connected to LL, an interface priority queue(IFq), a mac layer(MAC), a network interface(netIF), all connected to the channel. These network components are created and plumbed together in OTcl. Each component is briefly described here.

2.7 Link Layer :

The LL used by mobilenode is same as described in Chapter 14. The only difference being the link layer for mobilenode, has an ARP module connected to it which resolves all IP to hardware (Mac) address conversions. Normally for all outgoing (into the channel) packets, the packets are handed down to the LL by the Routing Agent. The LL hands down packets to the interface queue. For all incoming packets (out of the channel), the mac layer hands up packets to the LL which is then handed off at the nodeentry point.

2.8 ARP:

The Address Resolution Protocol (implemented in BSD style) module receives queries from Link layer. If ARP has the hardware address for destination, it writes it into the mac header of the packet. Otherwise it broadcasts an ARP query, and caches the packet temporarily. For each unknown destination hardware address, there is a buffer for a single packet. Incase additional packets to the same destination is sent to ARP, the earlier buffered packet is dropped. Once the hardware address of a packet's next hop is known, the packet is inserted into the interface queue. The class ARPTable is implemented in ns/arp.cc,h and ns/tcl/lib/ns-mobilenode.tcl.

2.9 Interface Queue:

The class PriQueue is implemented as a priority queuewhich gives priority to routing rotocol packets, inserting them at the head of the queue. It supports running a filter over all packets in the queue and removes those with a specified destination address. See ns/priqueue.cc,h for interface queue implementation.

2.10 Mac Layer:

Historically, ns-2 (prior to release ns-2.33) has used the implementation of IEEE 802.11 distributed coordination function (DCF) from CMU. Starting with ns-2.33, several 802.11 implementations are available.

2.11 Tap Agents:

Agents that subclass themselves as class Tap defined in mac.h can register themselves with the mac object using method installTap(). If the particular Mac protocol permits it, the tap will promiscuously be given all packets received by the mac layer, before address filtering is done.

2.12 Network Interfaces:

The Network Interphase layer serves as a hardware interface which is used by mobilenode to access thechannel. The wireless shared media interface is implemented as class Phy/WirelessPhy. This interface subject to collisions and the radio propagation model receives packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength etc. This meta- data in pkt header is used by the propagation model in receiving network interface to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. The model approximates the DSSS radio interface (LucentWaveLan direct-sequence spread-spectrum).

2.13 Radio Propagation Model:

It uses Friss-space attenuation (1/r2) at near distances and an approximation to Two ray Ground (1/r4) at far distances. The approximation assumes specular reflection off a flat ground plane.

2.14 Antenna:

An omni-directional antenna having unity gain is used by mobilenodes.

2.15 Description of some TCL Commands

1. The 'set' and 'val( )' keywords are used to initialize the configuration parameters, as shown below.

```
"set val(chan) Channel/WirelessChannel"
```

2. The 'new' keyword is used to create a new object reference to a particular class, as shown below.

```
"set ns [new Simulator]"
```

3. The 'open' keyword is used to open a file in the given r/w/x mode. If that particular file does not exist, it is created and opened, as shown below.

```
"set tf [open wireless.tr w]"
```

4. The 'trace-all' function is used to trace the events in the opened trace file (*.tr).
5. The 'namtrace-all-wireless' function is to trace the events in the nam file created (*.nam).
6. The 'load-flatgrid' function is used to load the topography value of the simulation, like 1000 x 1000, as shown below.

```
$topo load_flatgrid 500 500"
```

7. The 'create-god' function is used to create the General Operations Director.
8. The 'node-config' function is used to configure the node by setting in it the configuration parameters.
9. The 'attach-agent' function is used to link one agent/application to another node/agent respectively.
10. The 'setdest' function is used to set the position of the node at a particular time.
11. The 'start' and 'stop' keywords are used to start and stop the application respectively.
12. The 'proc' keyword is used to indicate a procedure or a function.
13. The 'flush-trace' function is used to flush the traced events into the trace files.
14. The 'run' keyword is used to run the file.

3 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers or Interpreters: Python 3.10.1

4 Screenshots

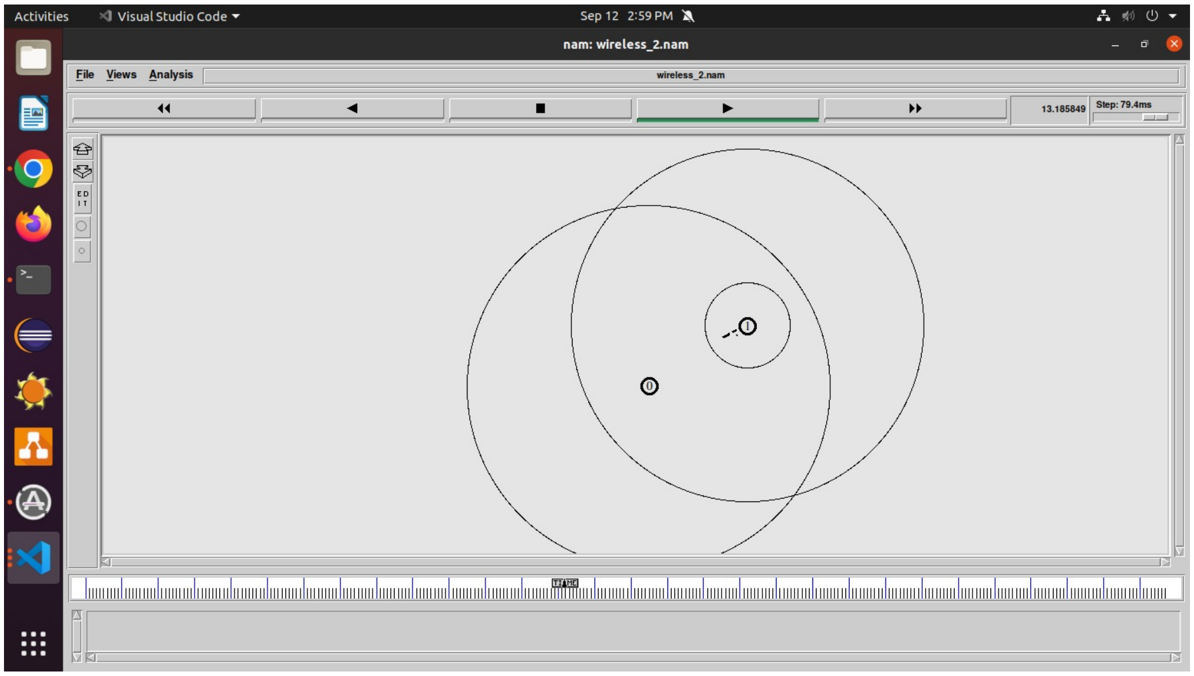


Figure 1: Watching the Animation of the output in NAM

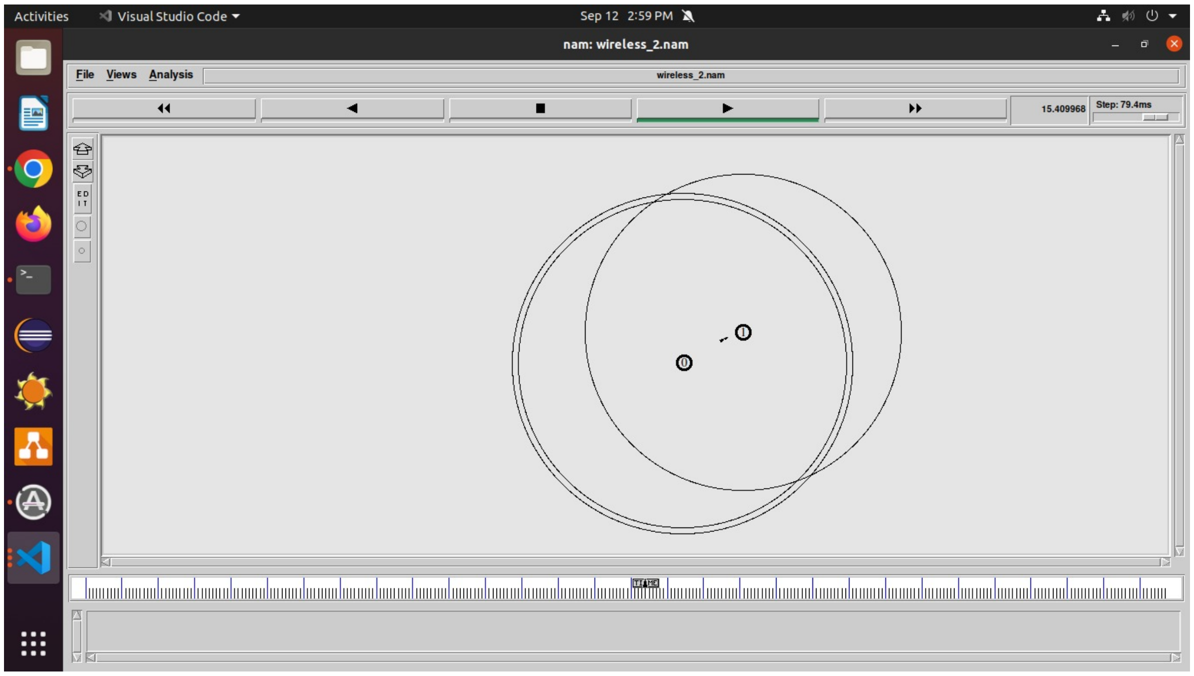


Figure 2: Watching the Animation of the output in NAM

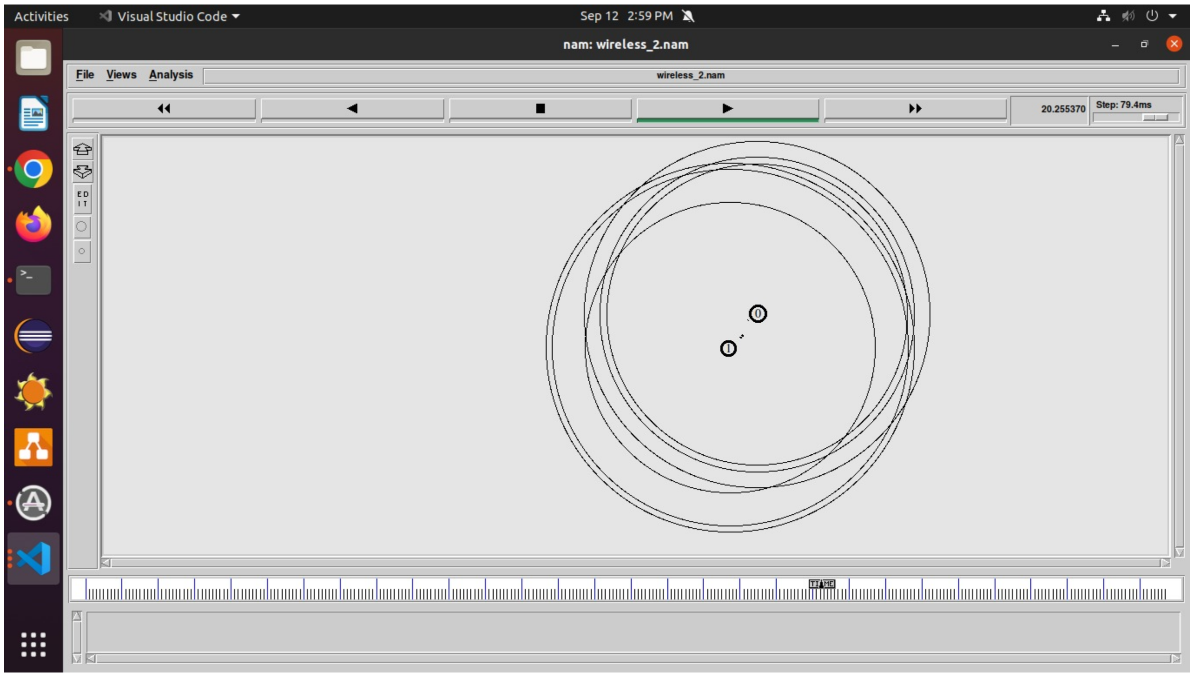
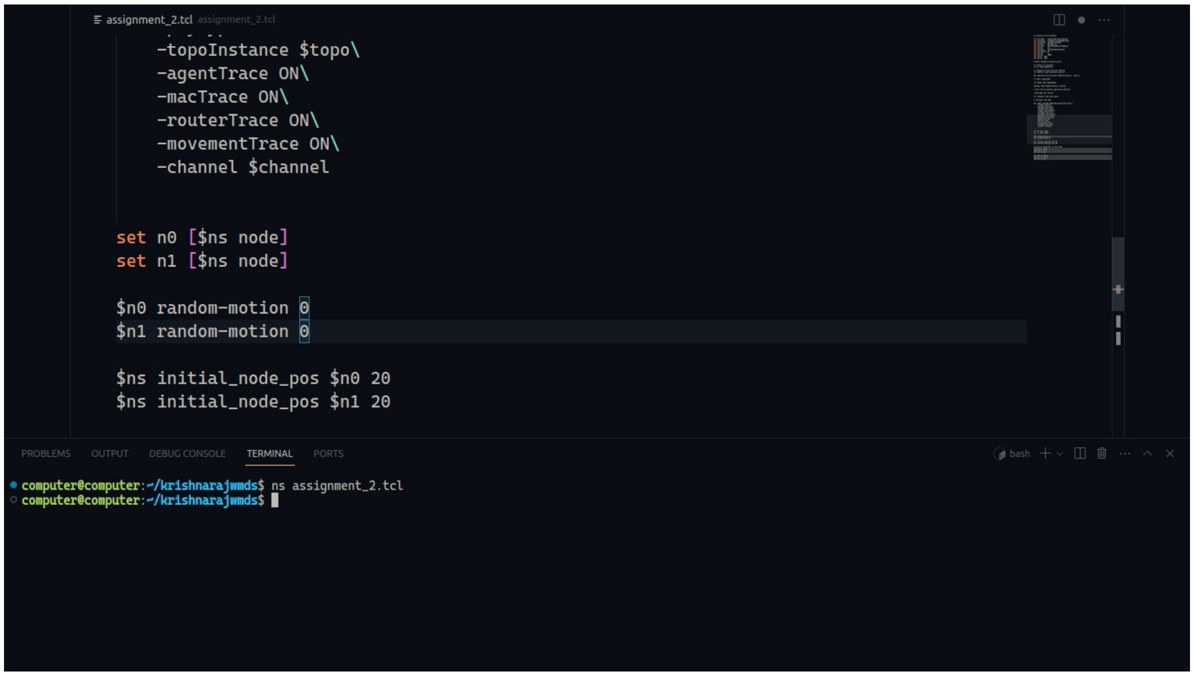


Figure 3: Watching the Animation of the output in NAM



```
    -topoInstance $topo\
    -agentTrace ON\
    -macTrace ON\
    -routerTrace ON\
    -movementTrace ON\
    -channel $channel


set n0 [$ns node]
set n1 [$ns node]

$n0 random-motion 0
$n1 random-motion 0

$ns initial_node_pos $n0 20
$ns initial_node_pos $n1 20
```

```
computer@computer:~/krishnarajwmds$ ns assignment_2.tcl
computer@computer:~/krishnarajwmds$
```

Figure 4: Running the TCL Script.

5 Code and Algorithm

5.1 Algorithm

1. Initialize variables

2. Create a Simulator object
3. Create Tracing and animation file
4. Create Topography
5. Create GOD - General Operations Director
6. Create nodes
7. Create Channel (Communication PATH)
8. Position of the nodes (Wireless nodes needs a location)
9. Any mobility codes (if the nodes are moving)
10. Run the simulation

5.2 Code

```
#initialize the variables
set val(chan)           Channel/WirelessChannel    ;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy      ;# network interface type WAVELAN
    DSSS 2.4GHz
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)             2                          ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol
set val(x)  500   ;# in metres
set val(y)  500   ;# in metres
#Adhoc OnDemand Distance Vector

#creation of Simulator
set ns [new Simulator]

#creation of Trace and namfile
set tracefile [open wireless_2.tr w]
$ns trace-all $tracefile

#Creation of Network Animation file
set namfile [open wireless_2.nam w]
$ns namtrace-all-wireless $namfile $val(x) $val(y)

#create topography
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#GOD Creation - General Operations Director
create-god $val(nn)

set channel1 [new $val(chan)]

#configure the node
$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 $channel1

set n0 [$ns node]
set n1 [$ns node]


$n0 random-motion 0
$n1 random-motion 0



$ns initial_node_pos $n0 20
$ns initial_node_pos $n1 20


#initial coordinates of the nodes
$n0 set X_ 10.0
$n0 set Y_ 20.0
$n0 set Z_ 0.0



$n1 set X_ 430.0
$n1 set Y_ 320.0
$n1 set Z_ 0.0


#Dont mention any values above than 500 because in this example, we use X and Y as
     500,500

#mobility of the nodes
#At what Time? Which node? Where to? at What Speed?
$ns at 1.0 "$n0 setdest 490.0 340.0 25.0"
$ns at 1.0 "$n1 setdest 300.0 130.0 5.0"

#the nodes can move any number of times at any location during the simulation (
    runtime)
$ns at 20.0 "$n1 setdest 100.0 200.0 30.0"

#creation of agents
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp

$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
```

```tcl
$ftp attach-agent $tcp
$ns at 1.0 "$ftp start"


#set udp [new Agent/UDP]
#set null [new Agent/Null]
#$ns attach-agent $n2 $udp
#$ns attach-agent $n3 $null
#$ns connect $udp $null
#set cbr [new Application/Traffic/CBR]
#$cbr attach-agent $udp
#$ns at 1.0 "$cbr start"


$ns at 30.0 "finish"

proc finish {} {
 global ns tracefile namfile
 $ns flush-trace
 close $tracefile
 close $namfile
 exit 0
}

puts "Starting Simulation"
$ns run
```

Listing 1: Assignment 2.tcl

6 Conclusion

Thus, we have studied and simulated wireless nodes with mobility.

7 FAQ

1. *Why propagation model is used in wireless network? What are the different types of it?*

   Propagation models are essential in wireless networks for predicting how radio signals propagate in the environment. They serve the following purposes:

   - Predict Signal Coverage: Propagation models help estimate the coverage area of wireless transmitters, enabling network planners to determine where signals can be received.
   - Network Design: By understanding signal behavior, network designers can optimize antenna placement, power levels, and frequency allocation.
   - Signal Analysis: During simulations or real-world deployments, propagation models assist in analyzing signal quality, interference, and path loss.

   Different types of propagation models include:

   - Free-Space Path Loss Model (FSPL)
   - Two-Ray Ground Reflection Model
   - Log-Distance Path Loss Model
   - Shadowing and Fading Models
   - ITU-R P.1411 Urban Propagation Model
   - Okumura-Hata Model

2. *Explain different types of queue object in wireless network.*

   In wireless networks, queue objects manage the flow of data packets. Several types of queue objects are used:

   - DropTail Queue: This simple queue drops packets when it reaches its capacity, which can lead to unfairness.
   - Random Early Detection (RED) Queue: RED prevents congestion by randomly dropping packets before the queue becomes full, providing better fairness.
   - Class-Based Queueing (CBQ): CBQ divides traffic into classes with different priorities, allowing for QoS management.

3. *Draw and explain trace file format in wireless network.*

   Trace files record events and parameters in wireless network simulations. A typical trace file includes lines with the following format:

   ```
   Event Type   Time   Node ID   Parameters...
   ----------------------------------------
   Send         0.1    Node 1    Packet ID: 1, Destination: Node 2
   Receive      0.2    Node 2    Packet ID: 1
   Drop         0.3    Node 3    Packet ID: 2, Reason: Queue Full

   Event Type indicates the type of event, Time is the simulation time,
   Node ID identifies the involved node, and Parameters
   provide event-specific details.
   ```

4. *What is the role of GOD?*

   In the context of wireless network simulations using NS-2, GOD (Graphical Object Debugger) serves as a crucial tool for debugging and visualization. Its roles include real-time visualization of network simulations, packet tracing, node debugging, and event monitoring, enhancing the debugging and analysis process.

   (a) **Real-Time Visualization** GOD provides real-time visualization of the network simulation. It allows users to observe the movement of nodes, packet transmissions, and interactions within the simulated environment.

   (b) **Packet Tracing** GOD enables users to trace the paths of packets as they traverse the network. This is essential for debugging and understanding how packets move through the wireless network.

   (c) **Node Debugging** Users can interactively debug nodes within the simulation. This includes examining node states, variable values, and the execution of event handlers.

   (d) **Visualization of Node Mobility** For scenarios involving mobile nodes, GOD can display node trajectories, making it easier to assess node movements and behaviors.

   (e) **Event Monitoring** GOD allows users to monitor and inspect events as they occur during the simulation. This is useful for diagnosing issues and understanding the sequence of events.

5. *How to deal with Very large trace files?*

   Managing very large trace files in wireless network simulations can be challenging. Strategies include subsampling, data compression, parallel processing, filtering, summary statistics, database storage, visualization tools, incremental analysis, resource scaling, data sampling, and archiving.

   (a) **Subsampling** Instead of analyzing the entire trace, consider subsampling by extracting a representative subset of data for analysis. This reduces the file size while retaining essential information.

   (b) **Data Compression** Use data compression techniques to reduce the size of trace files. Tools like gzip or tar can compress trace files before storage.

   (c) **Parallel Processing** If possible, leverage parallel processing or distributed computing to analyze large trace files more efficiently. Divide the analysis workload among multiple processors or nodes.