**COMPUTER NETWORKS (CS F303)**
**SECOND SEMESTER 2018-19**
**LAB-SHEET – 10**
**TOPIC: Wireless Network Simulation using ns-2**

## Learning Objectives:

- **To learn wireless network topology design in ns-2**
- **To learn node mobility, traffic pattern and traffic routing for wireless networks**

Till now we have been doing the experiments on the network that involve the nodes with a wired interface only. The most common name for this wired interface is Ethernet. In this lab, we will learn how to perform the simulation of wireless networks, i.e., the network with hosts (in wireless terminology we also call these host machines as nodes) that communicate with other nodes using some wireless technology. Unlike wired communication, some choices such as WiFi (802.11), ZigBee (802.15.4), GSM 3G, 4G, WiMax, etc., are available for wireless communication. At this point we will not go into the details of these technologies and just using the term "Wireless Node" to represent a host in a wireless network. Another, property that differentiates the wireless network from wired networks is that the wireless nodes may be mobile. As a consequence, the topology of the network (the connections between the nodes) becomes dynamic, and hence IP based hierarchical routing which is based on the fixed topology is not applicable. Moreover, the Mobile IP based solutions are restricted to the scenario where nodes change their point of contact to the Internet and cannot be applied in the wireless network scenario where all the nodes in the network may be changing their position continuously.

In a nutshell, the wireless networks or the wireless devices if seen from the perspective of OSI network model differ at their bottom three layers i.e., Physical layer, Data Link Layer and Networking Layer. The upper layers, i.e., transport layer (TCP) and application layer (HTTP) are completely unaware of the fact that they are running over the wireless network or wired network.

In this lab, you are going to learn to use the wireless simulation using in NS2. The lab is divided into two parts. In part one, you will learn to create and run a simple 2-node wireless network simulation, and in the next part, we will extend the scenario of part 1 to create a relatively more complex wireless scenario.

*In NS2 also, while writing a script for a wireless simulation, most of the code that is different when compared to wired simulation are related to these three bottom layers only. That means the transport layer agents and the application layer are attached to the nodes exactly in the same manner as in wired simulation.*

## Exercise #1: Creating a simple wireless simulation scenario with only two-nodes

The topology consists of two mobile wireless nodes. The nodes move about within an area whose boundary is defined in this example as 500mX500m. The nodes start initially at two opposite ends of the boundary. Then they move towards each other in the first half of the simulation and again move away for the second half. A TCP connection is setup between the two wireless nodes. Packets are exchanged between the nodes as they come within hearing range of one another. As they move away, packets start getting dropped.

***The tcl script ( simple-wireless.tcl ) for this exercise can be downloaded from the course website***

In the following, we will understand the concepts of performing wireless simulation by having a walkthrough of sample tcl script, but only the code which is different from the wired simulation would be discussed here.

**Defining Options**

At the beginning of a wireless simulation, we need to define Physical layer, Link Layer and the MAC layer and the properties of the wireless node as well as the routing protocol used in the network. However, defining the routing protocol for this exercise does not make much sense as there are only two nodes. Note that this step of defining options was missing in case of wired node. See comments in the code below for a brief description of each variable defined.

```
# ======================================================================
# Define options
# ======================================================================
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(mac)            Mac/802_11                 ;# MAC type
set val(rp)             DSDV                       ;# ad-hoc routing protocol
set val(nn)             2                          ;# number of wireless nodes
# ======================================================================
```

You can consider val() as an array that contains the values of various parameters required to configure a wireless node. We will use this array while actually configuring the node properties with the help of an API.

### Creating a topology of 500mX500m

```
set topo       [new Topography]
$topo load_flatgrid 500 500
```

Here the topology is broken up into grids, and the default value of grid resolution is 1. A different value can be passed as a third parameter to load_flatgrid {} above.

## Creating "God" object

The "God (General Operations Director) is the object that is used to store global information about the state of the environment, network or nodes that an omniscient observer would have, but that should not be made known to any participant in the simulation." Currently, God object stores the total number of wireless nodes and a table of the shortest number of hops required to reach from one node to another.

## Creating and configuring wireless nodes

First, we to configure nodes before we can create them. Node configuration API consists of defining the type of ad-hoc routing protocol, Link Layer, MAC layer, IfQ, etc. The configuration API can be defined as follows:

```
# Configure 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) \
                         -topoInstance $topo \
                         -channelType $val(chan) \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF \
                         -movementTrace OFF
```

You can experiment with the traces by turning all of them on. AgentTraces are marked with AGT, RouterTrace with RTR and MacTrace with MAC in their 5th fields. MovementTrace, when turned on, shows the movement of the wireless nodes and the trace is marked with M in their 2nd field. *In this experiment, we have turned on only AgentTrace and RouterTrace*

### Creating the two wireless nodes

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

The random-motion for nodes is disabled here, as we are going to provide node position and movement (speed & direction) manually.

Now that we have created wireless nodes, we need to give them a position to start with,

```
# Provide initial (X,Y, for now Z=0) co-ordinates for node_(0) and node_(1)

$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 390.0
$node_(1) set Y_ 385.0
$node_(1) set Z_ 0.0
```

Node0 has a starting position of (5,2) while Node1 starts off at location (390,385).

Next produce some node movements,

```
# Node_(1) starts to move towards node_(0)
$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0"
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"

# Node_(1) then starts to move away from node_(0)
$ns_ at 100.0 "$node_(1) setdest 490.0 480.0 15.0"
```

$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0" means at time 50.0s, node1 starts to move towards the destination (x=25,y=20) at a speed of 15m/s. This API is used to change direction and speed of movement of the wireless nodes.

**Run the simulation and make the following observations.**

- At the end of the simulation run, trace-output file simple.tr is created.
- Locate the message exchanged between the routers for routing purpose
- When the TCP flow is starting from node0 and when the node0 actually stated receiving it.
- When node0 and node1 actually stated exchanging route message.
- Locate the process of TCP handshake
- Is node0 able to send node0 is not able to send the data after a certain point in time even when the TCP connection is not broken.

**Visit the following link to know the details on wireless trace file format.**

**http://nsnam.sourceforge.net/wiki/index.php/NS-2_Trace_Formats#Wireless_Trace_Formats**

## Exercise #2: Using node-movement/traffic-pattern files and other features in wireless simulations

As an extension to the previous exercise, we are going to simulate a simple multihop wireless scenario consisting of 3 wireless nodes here. As before, the wireless nodes move within the boundaries of a defined topology. However, the node movements for this example shall be read from a node-movement file called scen-3-test. scen-3-test defines random node movements for the 3 wireless nodes within a topology of 670mX670m. This file is available as a part of the ns distribution and can be found, along with other node-movement files, under directory ~ns/tcl/mobility/scene.

In addition to node-movements, traffic flows that are setup between the wireless nodes, are also read from a traffic-pattern file called cbr-3-test. cbr-3-test is also available under ~ns/tcl/mobility/scene. Random CBR and TCP flows are setup between the 3 wireless nodes and data packets are sent, forwarded or received by nodes within hearing range of one another. See cbr-3-test to find out more about the traffic flows that are setup.

We make changes to the script, **simple-wireless.tcl**, we had created in Exercise1 and call the resulting file **wireless-simulation-routing.tcl**. For a copy of **wireless-simulation-routing.tcl** download from the course web page. In addition to the variables (LL, MAC, antenna etc.) that were declared at the beginning of the script, we now define some more parameters like the connection-pattern and node-movement file, x and y values for the topology boundary, a seed value for the random-number generator, time for the simulation to stop.

```
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(x)              670   ;# X dimension of the topography
set val(y)              670   ;# Y dimension of the topography
set val(ifqlen)         50            ;# max packet in ifq
set val(seed)           0.0
set val(adhocRouting)   DSR
set val(nn)             3             ;# how many nodes are simulated
set val(cp)             "cbr-3-test"
set val(sc)             "scen-3-test"
set val(stop)           2000.0          ;# simulation time
```

Number of wireless nodes is changed to 3; Also we use DSR (dynamic source routing) as the adhoc routing protocol inplace of DSDV (Destination sequence distance vector);

Next (after creation of Wireless nodes) source node-movement and connection pattern files that were defined earlier as val(sc) and val(cp) respectively.

```
#
# Define node movement model
#
puts "Loading connection pattern..."
source $val(cp)

#
# Define traffic model
#
puts "Loading scenario file..."
source $val(sc)
```

We also see other lines like

```
$god_ set-dist 1 2 2
```

These are command lines used to load the god object with the shortest hop information. It means the shortest path between node 1 and 2 is 2 hops. By providing this information, the calculation of the shortest distance between nodes by the god object during simulation runs, which can be quite time-consuming, is prevented.

***Note: All the files required for this exercise can be downloaded from the course website***

On completion of the run, output file "wireless1-out.tr" and nam output file "wireless1-out.nam" are created. Running wireless1-out.nam we see the three wireless nodes moving in nam window. However, as mentioned earlier no traffic flow can be seen (not supported as yet). For a variety of coarse and fine-grained trace outputs turn on/off AgentTrace, RouteTrace, MacTrace and movement trace as shown earlier in Exercise 1. From the output, we find nodes 0 and 2 are out of range and so cannot hear one another. Node1 is in range with nodes 0 and 2 and can communicate with both of them. Thus all pkts destined for nodes 0 and 2 are routed through node 1.

*****