Assignment No. 2

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

Theory:

The wireless model essentially consists of the Mobile Node at the core, with additional supporting features that allows simulations of multi-hop ad-hoc networks, wireless LANs etc. The Mobile Node object is a split object. The C++ class Mobile Node is derived from parent class Node. A Mobile Node thus is the basic Node object with added functionalities of a wireless and mobile node like ability to move within a given topology, ability to receive and transmit signals to and from a wireless channel etc. A major difference between them, though, is that a Mobile Node is not connected by means of Links to other nodes or mobile nodes. For creating the wireless network, we shall describe the internals of Mobile Node, its routing mechanisms, the routing protocols dsdv, aodv, tora and dsr, creation of network stack allowing channel access in Mobile Node.

Routing Protocols: The four ad-hoc routing protocols that are currently supported are Destination Sequence Distance Vector (DSDV), Dynamic Source Routing (DSR), Temporally ordered Routing Algorithm (TORA) and Adhoc On-demand Distance Vector (AODV). The old APIs for creating a mobile node depended on which routing protocol was used.

Creating Mobile Nodes: Before creating the mobile nodes, their topology has to be manually defined. When the topology of the network to be simulated is drafted, then the first step would be to create the mobile nodes. Then the node movement is set. Then the agents and applications are defined for that particular network topology.

From the topology, a basic idea of the network to be simulated is obtained. For that topology, the node-config parameters are set first. This is shown below.

```
$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           OFF \
                        -macTrace              OFF \
                        -movementTrace          OFF
```

After setting the node configuration parameters, the actual nodes are created.

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

Thus the mobile nodes are created.

Setting Mobile Node Movements: The mobile node is designed to move in a three dimensional topology. However the third dimension (Z) is not used. That is the mobile node is assumed to move always on a flat terrain with Z always equal to 0. Thus the mobile node has X, Y, Z(=0) co-ordinates that is continually adjusted as the node moves. There are two mechanisms to induce movement in mobile nodes. In the first method, starting position of the node and its future destinations may be set explicitly. These directives are normally included in a separate movement scenario file.

The start-position and future destinations for a mobile node may be set by using the following APIs:

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

At $time sec, the node would start moving from its initial position of (x1,y1) towards a destination (x2,y2) at the defined speed. In this method the node-movement-updates are triggered whenever the position of the node at a given time is required to be known. This may be triggered by a query from a neighbouring node seeking to know the distance between them, or the setdest directive described above that changes the direction and speed of the node.

Topology Definition: Irrespective of the methods used to generate node movement, the topography for mobile nodes needs to be defined. It should be defined before creating mobile nodes. Normally flat topology is created by specifying the length and width of the topography using the following primitive:

```
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
```

where val(x) and val(y) are the boundaries used in simulation

Following is a list of commands used in wireless simulations:

```
$ns_ node-config -addressing Type <usually flat or hierarchical used for wireless topologies>
                 -adhocRouting <adhoc rotuing protocol like DSDV, DSR,
                 -llType <LinkLayer>
                 -macType <MAC type like Mac/802_11>
                 -propType <Propagation model like Propagation/TwoRayGround>
                 -ifqType <interface queue type like Queue/DropTail/PriQueue>
                 -ifqLen <interface queue length like 50>
                 -phyType <network inteface type like Phy/WirelessPhy>
                 -antType <antenna type like Antenna/OmniAntenna>
                 -channelType <Channel type like Channel/WirelessChannel>
                 -topoInstance <the topography instance>
                 -wiredRouting <turning wired routing ON or OFF>
                 -mobileIP <setting the flag for mobileIP ON or OFF>
                 -energyModel <EnergyModel type>
                 -initialEnergy <specified in Joules>
                 -rxPower <specified in W>
                 -txPower <specified in W>
                 -agentTrace <tracing at agent level turned ON or OFF>
                 -routerTrace <tracing at router level turned ON or OFF>
                 -macTrace <tracing at mac level turned ON or OFF>
                 -movementTrace <mobilenode movement logging turned ON or OFF>
```

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.

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 node_entry_ point.

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.

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.

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.

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.

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).

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.

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

Description of Tcl Commands used:

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.

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

Sample Code:

```
# Define options #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                         ;# X dimension of topography in metres
set val(y)              400                         ;# Y dimension of topography in metres
#set val(stop)          10.0                        ;# time of simulation end

# Simulator Instance Creation
set ns [new Simulator]
#Nam File Creation nam – network animator
set namfile [open wireless.nam w]
#Tracing all the events and cofiguration
$ns namtrace-all-wireless $namfile $val(x) $val(y)

#Trace File creation
set tracefile [open wireless.tr w]
#Tracing all the events and cofiguration
$ns trace-all $tracefile

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

# general operational descriptor- storing the hop details in the network
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

# Node Creation
set node1 [$ns node]
# Initial color of the node
$node1 color black
set node2 [$ns node]
$node2 color black

#Location fixing for a single node
$node1 set X_ 200
$node1 set Y_ 100
$node1 set Z_ 0
$node2 set X_ 200
$node2 set Y_ 300
$node2 set Z_ 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?
$ns at 6.0 "$node1 setdest 490.0 340.0 25.0"
# Label and coloring
$ns at 1.0 "$node1 label Node1"
```

```
$ns at 2.0 "$node2 label Node2"
$ns at 4.0 "$node1 color blue"
$ns at 5.0 "$node2 color red"
#Size of the node
$ns initial_node_pos $node1 50
$ns initial_node_pos $node2 30

#$ns at 1.0 "start"
$ns at 10.0 "finish"
# ending nam and the simulation
#$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
#$ns at $val(stop) "stop"
#Stopping the scheduler
$ns at 10.01 "puts \"end simulation\" ; $ns halt"
#$ns at 10.01 "$ns halt"
proc finish {} {
        global ns namfile tracefile
        $ns flush-trace
        close $namfile
        close $tracefile
#executing nam file
#exec nam wireless.nam &
}
#Starting scheduler
$ns run
#############################################################
```

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

FAQs:

1. Why propagation model is used in wireless network? What are the different types of it?
2. Explain different types of queue object in wireless network.
3. Draw and explain trace file format in wireless network.
4. What is the role of GOD?
5. How to deal with Very large trace files?