Experiment-5

For a wireless network consisting of three mobile nodes(n0-n2).

Nodes are configured with the specific parameters of a wireless node.

Initial location of the node is fixed.

Nodes are given mobility with fixed speed and fixed destination location.

TCP agents is attached to the node 0 and TCP sink agent is attached to the node1.

Both the agent are connected and FTP application is attached to TCP agent.

Write a TCL script and make an ad-hoc simulation to analyze the output in the trace file.

Use the routing protocol as adhoc on demand distance vector(AODV).

```
#setting different parameters


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(ifqlen) 50

set val(nn) 3

set val(rp) AODV

set val(x) 500

set val(y) 500

set val(stop) 150
```

```
# scheduler object creation
set ns [new Simulator]

# Enable tracing
$ns use-newtrace

# creating trace file and nam file
set tracefd [open wireless1.tr w]
set windowVsTime2 [open win.tr w]
set namtrace [open wireless1.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

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

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

# Configure the nodes with corrected parameters
$ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propInstance [new $val(prop)] \
            -phyType $val(netif) \
            -channel [new $val(chan)] \
            -topoInstance $topo \
            -wiredRouting OFF \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON

# Create mobile nodes
for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns node]
   $node_($i) random-motion 0  ;# disable random motion
}

# Provide initial location of mobile nodes
$node_(0) set X_ 100.0
$node_(0) set Y_ 100.0
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 300.0
$node_(1) set Y_ 300.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 200.0
$node_(2) set Y_ 200.0
$node_(2) set Z_ 0.0

# Generation of movements - ensure they are within bounds
$ns at 10.0 "$node_(0) setdest 400.0 400.0 20.0"
$ns at 15.0 "$node_(1) setdest 100.0 100.0 15.0"
$ns at 20.0 "$node_(2) setdest 300.0 300.0 10.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$tcp0 set window_ 32
set sink0 [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp0
$ns attach-agent $node_(1) $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 5.0 "$ftp0 start"
```

```
$ns at 100.0 "$ftp0 stop"

# Set another TCP connection between node_(1) and node_(2)
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
$tcp1 set window_ 32
set sink1 [new Agent/TCPSink]
$ns attach-agent $node_(1) $tcp1
$ns attach-agent $node_(2) $sink1
$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 5.0 "$ftp1 start"
$ns at 100.0 "$ftp1 stop"

# Add some UDP traffic for variety
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $node_(0) $udp
$ns attach-agent $node_(2) $null
$ns connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

```
$cbr set packetSize_ 512

$cbr set interval_ 0.1

$ns at 1.0 "$cbr start"

$ns at 120.0 "$cbr stop"


# Printing the window size

proc plotwindow {tcpsource file} {

   global ns

   set time 1.0  ;# increased from 0.01 for better visualization

   set now [$ns now]

   set cwnd [$tcpsource set cwnd_]

   puts $file "$now $cwnd"

   $ns at [expr $now + $time] "plotwindow $tcpsource $file"

}


$ns at 5.1 "plotwindow $tcp0 $windowVsTime2"


# Define node initial position in nam

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

   $ns initial_node_pos $node_($i) 40

}


# Telling nodes when the simulation ends

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

   $ns at $val(stop) "$node_($i) reset"
}


# Ending nam and the simulation

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

$ns at $val(stop) "stop"

$ns at 150.0 "puts \"end simulation\"; $ns halt"


proc stop {} {

   global ns tracefd namtrace windowVsTime2

   $ns flush-trace

   close $tracefd

   close $namtrace

   close $windowVsTime2

   puts "Simulation completed."

   puts "Running nam visualization..."

   exec nam wireless1.nam &

   exit 0

}

puts "Starting simulation with $val(nn) nodes..."

$ns run
```

Execution Steps

```
 gedit 5.tcl
ns 5.tcl
gedit win.tr
gnuplot
 plot “win.tr”
 plot “win.tr” w lines 1
```