Simulation of a WiMAX network to evaluate the performance of MAC IEEE 802.16 during the IR phase of Network Entry and Initialization

Namratha M
Mtech in Software Engineering,
Department of Information Science,
PESIT Bangalore, India

Pradeep
Mtech in Software Engineering,
Department of Information Science,
PESIT Bangalore, India

Manu G V
BE, MBA, Mtech in Computer Science, SJCIT, Chikkabalapur
Technical Lead-Testing, Calsoft Labs, Bangalore, India

***ABSTRACT*—Pervasive Computing is also called as Ubiquitous Computing, which means "being present everywhere at once" or "constantly encountered". The main idea behind making these pervasive computing systems is that these systems improve living by performing computations on their own, without having to be monitored by anyone. These systems are targeted to become invisible to the user i.e., they perform their tasks without the user's knowledge.**

To achieve this environment, the underlying requirement is a Network. One of the biggest constraints in achieving this environment is the "Last Mile" problem. It refers to the last leg of delivering connectivity from a communications provider to a customer. In recent years there has been increasing interest shown in wireless technologies for subscriber access, as an alternative to traditional twisted-pair local loop.

WiMAX, Worldwide Interoperability for Microwave Access, is a Telecommunications technology that provides wireless transmission of data and is based on the IEEE 802.16 standard (also called Broadband Wireless Access). 802.16 uses paired radio channels Up Link Channel (UL) and Down Link Channel (DL) for establishing a communication between Base Station (BS) and Subscriber Station (SS). When a SS wants to establish connectivity with a BS it goes through the Network Entry and Initialization procedure of which Initial Ranging (IR) is a very important part. IR is the process of acquiring the correct timing offset and power adjustments such that the SS's transmissions are aligned to maintain the UL connection with the BS. All the SS's of a BS will compete for the contention slots for their network entry. Whenever the SS has to transmit the request packets it performs the Truncated Binary Exponential Backoff procedure. This method is the contention resolution procedure used in IEEE 802.16 networks.

Our focus here was to simulate a WiMAX network so as to evaluate the performance of MAC IEEE 802.16 during the IR phase of Network Entry and Initialization. We have used Network Simulator-2 (NS-2) for our simulation purposes. We are using WiMAX "patch" which simulates the PHY and the MAC features of a WiMAX network. We have evaluated the performance of MAC IEEE 802.16 for various topologies.

Keywords—Backoff delay; Circular topology; Linear topology; Markov Model; Pervasive computing; Ubiquitous computing; WiMAX

I. INTRODUCTION

Ubiquitous computing is roughly the opposite of virtual reality. Where virtual reality puts people inside a computer-generated world, ubiquitous computing forces the computer to live out here in the world with people.

Pervasive computing has many potential applications, from health and home care to environmental monitoring and intelligent transport systems. Pervasive computing systems[1] and services may lead to a greater degree of user knowledge of, or control over, the surrounding environment, whether at home, or in an office or car. They may also show a form of 'intelligence' that will make life easier. A few examples are worth mentioning here. A refrigerator can automatically signal the grocery shop once the milk or juice comes below a certain level, thus alleviating the owner from the task of notifying the grocery shop or going and buying. Various home appliances can become pervasive computing systems with similar 'intelligence', like a television that automatically switches on and tunes itself to a particular channel at a pre-defined time or an air conditioner that switches on automatically and brings the room to a specified temperature. A phone call can be made or answered only by gestures.

The main idea behind making these pervasive computing systems are, that these systems improve the living of the humans by performing the computation on their own, without having to be monitored by anyone. These systems are targeted to become invisible to the user i.e., they perform their tasks without the user's knowledge.

II. MATHEMATICAL ANALYSIS OF THE DELAY FOR THE IR SCHEME

In this section we analyze the IR scheme and evaluate mathematically the delay involved in the procedure. First we structure the IR mechanism as a set of distinct states with transitions among these states. Then this information is used to model IR as a Markov process [1]. In a Markov process, the probability of the system making a transition to a particular state depends only on the state the system is currently in.

Also, in this Markov process, we calculate the delays associated with the transitions between the states. Finally, by making use of the delays and probabilities associated with each of the transitions, we derive a mathematical equation describing the total IR delay. The Markov process is derived for IR in case of OFDMA PHY, since it covers all the steps in the OFDM based procedure as well. In case of the OFDMA PHY, Code Division Multiple Access (CDMA) codes are used instead of the RNG-REQ messages in the first part of the IR procedure.

A. *Modeling IR as a Markov Process*

Markov processes[2] provide very flexible, powerful, and efficient means for the description and analysis of dynamic (computer) system properties. Performance and dependability measures can be easily derived. Moreover, Markov processes constitute the fundamental theory underlying the concept of queuing systems. In fact, the notation of queuing systems has been viewed sometimes as a high-level specification technique for (a sub-class of) Markov processes. A stochastic process is defined as a family of random T, which is usually called the time parameter if T is a subset of the set of positive real numbers. The set of all possible values of X_t (for each $t \in T$) is known as the state space S of the stochastic process. A large number of stochastic processes belong to the important class of Markov processes. A stochastic process is a Markov process when the future state of the process depends only the current state and not on the history of the system. A Markov process is a memory-less stochastic process.

After analyzing the Initial Ranging procedure [3], we enumerate the following states as well as transitions needed for modeling the procedure.

State 1: Waiting for UL-MAP. This is also the start state.

State 2: SS is performing Backoff procedure.

State 3: Waiting for an RNG-RSP message from BS.

State 4: Continue

State 5: Success State – Wait for CDMA Allocation IE.

State 6: Abort – Start network entry procedure at a different DL channel

State 7: Waits for RNG-RSP again.

State 8: Proceed to next phase of network entry

State 9: Commence Periodic Ranging

The transitions among the states are as follows. In State 1, the SS waits for a UL-MAP. After receiving this message it makes a transition to State 2. Transmission of CDMA code occurs at end of State 2. Also a timer is set for waiting for RNG-RSP message. This transition leaves the system in State 3. When in State 3, if the timer for RNG-RSP expires then SS increments the power level and goes back to State 1. When in State 3, if RNG – RSP is obtained with Ranging code as well as the Ranging slot, then it makes a transition to State 4. Here the necessary adjustments specified in RNG-RSP are made and system moves to State 1.

When in State 3, if RNG-RSP is obtained with success status, then the system transits to State 5. Here it waits for CDMA Allocation IE. After reception it sends RNG-REQ message on the allocated bandwidth and moves to State 7. When in State 7, on reception of RNG-RSP with success status it moves to State 8. On reception of RNG-RSP with continue status it moves to State 9. Else on reception of RNG-RSP with abort status, it goes to State 6 and SS starts the network entry procedure again. When in State 3, if RNG-RSP is obtained with abort status then the system goes to State 6 and SS starts the network entry procedure again. The following matrix diagram shows the transition probability matrix for IR. Using these probabilities we design the Markov process representation of IR. The states 6, 8 and 9 lead out of IR and are the absorbing states. For these states, transition occurs back to the same state with a probability one. In states 3 and 7, the outgoing probabilities are marked with algebraic symbols a1 to a4 and b1 to b3. This is because the probabilities of the transitions originating from these states are non-deterministic in nature. The sum of probabilities of all transitions originating from states 3 and 6 are still equal to 1. Next, the transition matrix is used to obtain the overall delay formula.

The details of the delays [4] involved along with the associated probabilities are given in the table 2.

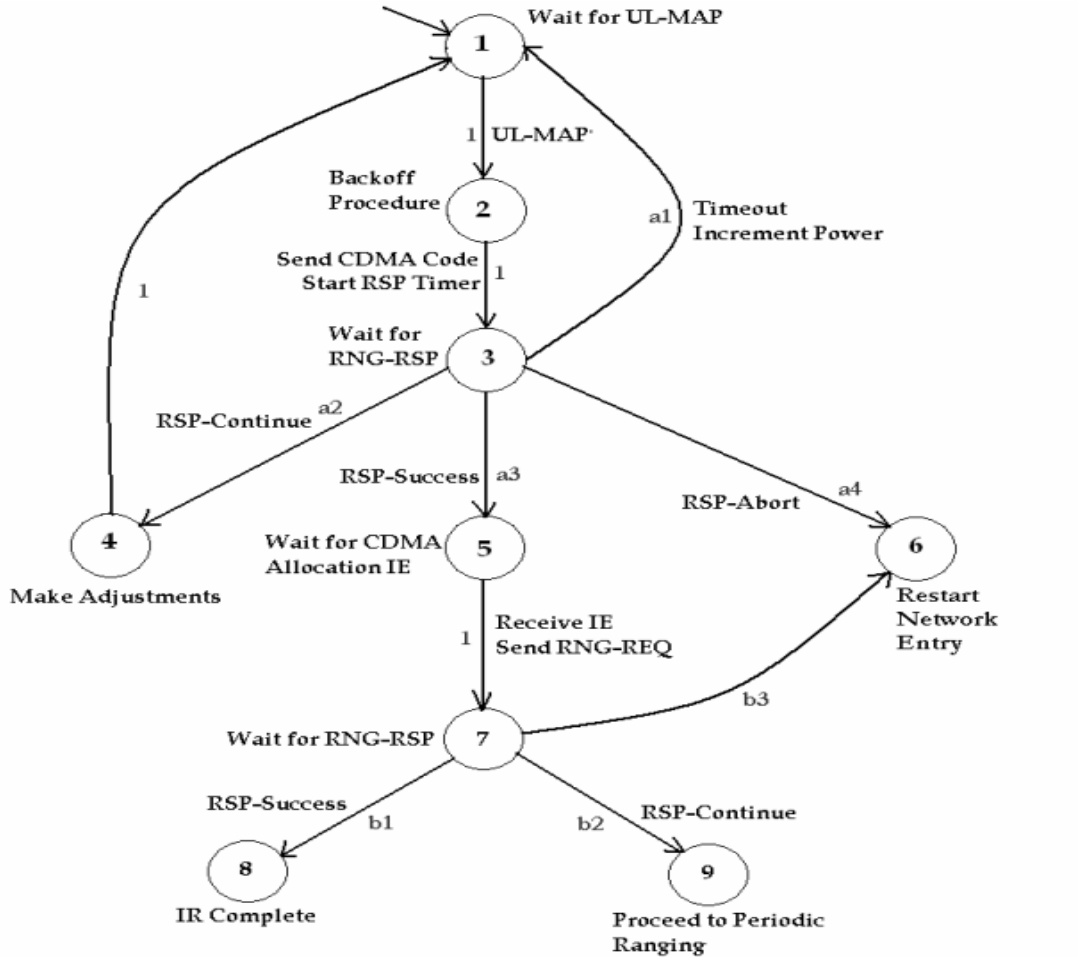

Fig.1. IR Procedure as a Markov Process

Table 2-Delay components in IR

Delay Involved	Probabilities
UL-MAP Reception (1 to 2)	1
Backoff Delay + Sending CDMA (2 to 3)	1
RNG-RSP Timeout (3 to 1)	a1
RNG-RSP Reception + Processing (3 to 4, 5 or 6)	a2,a3,a4
IE Allocation Delay + Sending RNG-REQ (5 to 7)	1
RNG-RSP Reception + Processing (7 to 8, 9 or 6)	b1,b2,b3

B. Mathematical Derivation of the Backoff Delay

Consider the first time an SS enters Backoff procedure [5]. Let the Initial Contention window be w. The random number will be picked in the range [0, w-1]. Let this random number be called k. The SS has to defer a total of k contention slots (CSs). Let the number of CSs in a frame be n. The number of frames that have to be deferred is k / n.00cscs. The delay involved here will be (k / ncs) * frame length. After k / ncs frames have passed the SS defers a further k modulo ncs CSs. The delay involved here is equal to (k % n cs) * Tcs, where Tcs is the length of one CS and % denotes the modulo operation. Therefore the total delay incurred so far is (k / n_{CS}) * frame length + (k % n_{CS}) * T_{CS}. Here the value of k can vary from 0 to w_0-1. Thus, we take an average of the delay over the random number.

AD_0 = (1/w_0) *Sum of [(k/n_{CS})*frame length + (k % n_{CS})*T_{CS}] as k varies from 0 to w_0-1.

Next we make an assumption that the probability of a successful transmission in a CS is ‘p’. Thus, probability of failure will be ‘1-p’. In case of a failure the contention window is doubled in size. Let the new window be equal to [0, w_1-1]. Similar to previous derivation the delay involved will be

AD_1 = (1/w_1) *Sum of [(k/n_{CS})*frame length + (k% n_{CS})*T_{CS}] as k varies from 0 to w_1-1.

Here $w_1 = 2 * w_0$.

Again there could be success or failure. So, it will enter the third Backoff window phase [0, w_2-1]. Continuing in this fashion, we get the following delays for the next three phases.

AD_2 = (1/w_2) *Sum of [(k/ n_{CS})*frame length + (k% n_{CS})* T_{CS}] as k varies from 0 to w_2-1.

AD_3 = (1/w_3) *Sum of [(k/ n_{CS})*frame length + (k% n_{CS})* T_{CS}] as k varies from 0 to w_3-1.

AD_4 = (1/w_4) *Sum of [(k/ n_{CS})*frame length + (k% n_{CS})* T_{CS}] as k varies from 0 to w_4-1.

Here $w_2 = 2 * w_1$, $w_3 = 2 * w_2$ and $w_4 = 2 * w_3$.

We make another assumption at this point. The SS is assumed to complete successful transmission of its CDMA code, in a maximum of 5 Backoff phases. Thus, the worst case of transmission will be four failures followed be a success. The final formula for the delay will be as follows.

$$\text{Backoff Delay (BD)} = p*\{AD_0 + t/2\}$$
$$+ ((1-p))*p*\{[AD_0 + t] + [AD_1 + t/2]\}$$
$$+ ((1-p)^2)*p*\{[AD0 + AD_1 + 2t] + [AD_2 + t/2]\}$$
$$+ ((1-p)^3)*p*\{[AD0 + AD_1 + AD_2 + 3t] + [AD_3 + t/2]\}$$
$$+ ((1-p)^4)*p*\{[AD0 + AD_1 + AD_2 + AD_3 + 4t]$$
$$+ [AD_4 + t/2]\}$$

Here t is the time-out after which failure is assumed. So, we take half that value for success i.e. t/2.

C. Mathematical Derivation of the Overall IR delay

By traversing the transition diagram[6] and multiplying the probabilities with the corresponding delays, the total delay can be calculated. The first part of the delay is in the loops 1-2-3-1 and 1-2-3-4-1. We call this D_{loop}. Then either success or abort occurs which is added to this part to get the final formula.

D_{loop} = 1*UL-MAP + 1*(BD + CDMA sending) + a_1 * (Timeout T3 + D-loop) + a_2 * (RSP + D-loop) Simplifying we get,

Now, the total delay involved can be represented

$$D_{loop} = \frac{1- (a_1+a_2)UL + BD + CDMA\ sending + a_1*T3 + a_2*RSP}{1- (a_1+a_2) + a_3* (RSP + CDMA\_IE + RNG\text{-}REQ + RSP) + a_4 * RSP}$$

Now, the total delay involved can be represented using the formula given below.

$D_{total} = D_{loop} + a_3 * (RSP + CDMA\_IE + RNG\text{-}REQ + (b_1+ b_2 + b_3)*RSP) + a_4 * RSP$

(here $b_1 + b_2 + b_3 = 1$)

Substituting the expression for the delay in the loop into the formula for overall delay in IR, we get the following final formula.

$$D_{loop} = \frac{UL + BD + CDMA\ sending + a_1*T3 + a_2*RSP}{1- (a_1+a_2) + a_3* (RSP + CDMA\_IE + RNG\text{-}REQ + RSP) + a_4 * RSP}$$

III. Working Of The Code

A. TCL script

The Tcl file [8],[9] contains the codes to run the simulation for 1 base station and variable number of mobile station. The file accepts the following arguments

i. The number of mobile nodes/stations
ii. The seed value (for random number generation)

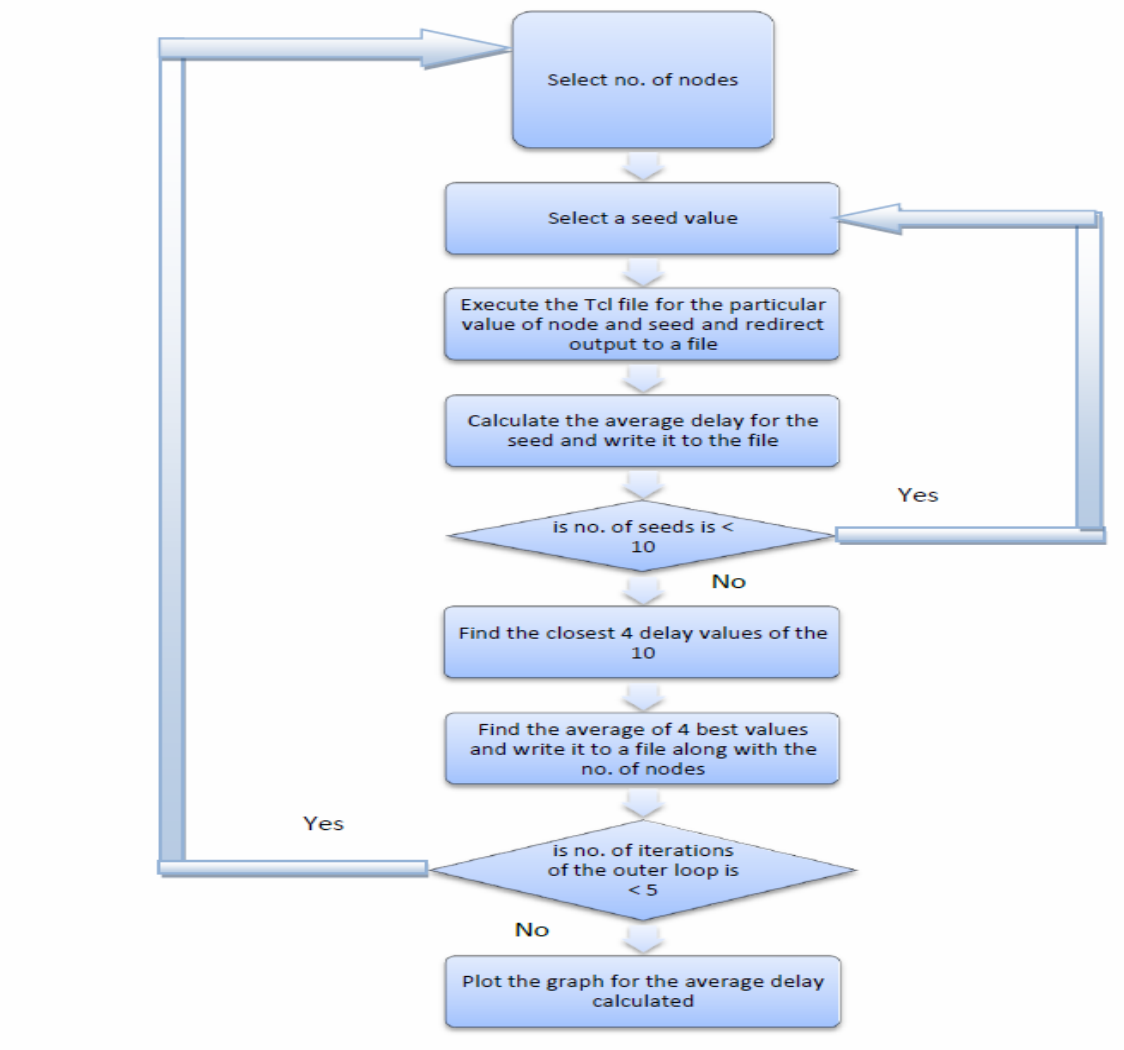

Fig.2. Process Flow

```
set nb_mn [lindex $argv 0]
set enteredSeed [lindex $argv 1]                    ;# max number of mobile node
set packet_size          1500                       ;# packet size in bytes at CBR
Applications
set output_dir .
set gap_size 1 ;#compute gap size between packets
puts "gap size=$gap_size"
set traffic_start 5 set traffic_stop    20          set
simulation_stop 30
set diuc 7 ;#modulation for MNs
Mac/802_16 set debug_ 0
Mac/802_16 set rtg_ 20
Mac/802_16 set ttg_ 20
Mac/802_16 set client_timeout_ 50                   ;#to avoid BS disconnecting the
SS
Phy/WirelessPhy/OFDM set g_ 0.25
#define debug values
Mac/802_16 set debug_ 1
Mac/802_16 set print_stats_ 1
Mac/802_16 set t21_timeout_ 20
```

```
#Fix: instead of using the send-scan manually, use a link going down trigger
Mac/802_16 set lgd_factor_ 1.8                    ;#note: the higher the value the earlier the trigger
Mac/802_16 set client_timeout_ 60 ;#to avoid BS disconnecting the SS
Agent/WimaxCtrl set adv_interval_ 1.0
Agent/WimaxCtrl set default_association_level_ 0
Agent/WimaxCtrl set synch_frame_delay_ 0.5
Agent/WimaxCtrl set debug_ 1

#define coverage area for base station: 20m coverage
Phy/WirelessPhy set Pt_ 0.025
Phy/WirelessPhy set RXThresh_ 2.025e-12 ;#500m radius
Phy/WirelessPhy set CSThresh_ [expr 0.9*[Phy/WirelessPhy set RXThresh_]]
# Parameter for wireless nodes
set opt(chan)                   Channel/WirelessChannel          ;# channel type
set opt(prop)                   Propagation/TwoRayGround         ;# radio-propagation
model
set opt(netif)                  Phy/WirelessPhy/OFDM             ;# network interface type
set   opt(mac)                  Mac/802_16/BS                    ;#   MAC type
set   opt(ifq)                  Queue/DropTail/PriQueue          ;#   interface queue type
set   opt(ll)                   LL                               ;#   link layer type
set   opt(ant)                  Antenna/OmniAntenna              ;#   antenna model
set   opt(ifqlen)               50                    ;#   max packet in      ifq
set   opt(adhocRouting)         DSDV                             ;# routing      protocol
set opt(namtr)                  newrng-irmsa.nam                 ;# for the nam trace file
set   opt(tr)           newrng-irmsa.tr                          ;# for the trace file
set   opt(x)            1100                          ;#   X  dimension of the topography
set   opt(y)            1100                          ;#   Y  dimens
```

The above are used to setup the WiMAX environment [10],[11] by assigning appropriate values to the various parameters.

```
set ns [new Simulator]
$ns use-newtrace
#create the topography
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
#puts "Topology created"
#nam
set nf [open newrng-irmsa.nam w]
$ns namtrace-all-wireless $nf $opt(x) $opt(y)
#open file for trace
set tf [open newrng-irmsa.tr w]
$ns trace-all $tf
puts "Output file configured"

global defaultRNG
$defaultRNG seed $enteredSeed
# set up for hierarchical routing (needed for routing over a basestation)
#puts "start hierarchical addressing"
$ns node-config -addressType hierarchical
AddrParams set domain_num_ 2                              ;# domain number lappend cluster_num 1 1
                                                  ;# cluster number for each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 [expr ($nb_mn+1)]           ;# number of nodes for
each cluster (1 for sink and one for mobile nodes + base station
AddrParams set nodes_num_ $eilastlevel
puts "Configuration of hierarchical addressing done"
```

```
# Create God
Create-god [expr ($nb_mn + 2)]                    ;# nb_mn + 2 (base station and sink node)
#puts "God node created"
```

The above lines are used to setup the simulation environment [12], which includes-

i. Creating a new instance of the ns2 simulator
ii. Creating a new topology
iii. Configuring the nam and trace files
iv. Specifying that the entered value of the seed should be used by the default Random Number Generator
v. Creating an address hierarchy, and configuring it *
vi. Creating a 'God object'. The number of mobile nodes is passed as argument which is used by God to create a matrix to store connectivity information of the topology

```
set sinkNode [$ns node 0.0.0]
#provide some co-ord (fixed) to base station node
$sinkNode set X_ 600.0
$sinkNode set Y_ 500.0
$sinkNode set Z_ 0.0
#puts "sink node created"
#creates the Access Point (Base station)
$ns node-config -adhocRouting $opt(adhocRouting) \
                -llType $opt(ll) \
                -macType Mac/802_16/BS \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop)          \
                -phyType $opt(netif) \
                -channel [new $opt(chan)] \
                -topoInstance $topo \
                -wiredRouting ON \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON        \
                -movementTrace OFF
puts "Configuration of base station"

set bstation [$ns node 1.0.0]
$bstation random-motion 0
#puts "Base-Station node created"
#provide some co-ord (fixed) to base station node
$bstation set X_ 550.0
$bstation set Y_ 550.0
$bstation set Z_ 0.0
[$bstation set mac_(0)] set-channel 0
```

The above lines are used to-

i. Create a sink node. The sink node acts as a dropped target (any packets dropped at the base station are sent to the sink node.)
ii. Change the node configuration and create a base station

```
$ns node-config -macType Mac/802_16/SS \
                -wiredRouting ON \
                -macTrace ON                    ;# Mobile nodes cannot do
                                                Routing.
```

The following lines are used for configuring the mobile node-

```
set wl_node [$ns node 1.0.[expr $i + 1]] ;# create the node with particular        #address
$wl_node random-motion 0
$wl_node base-station [AddrParams addr2id [$bstation node-addr]]
$wl_node set X_ [expr 340.0+$i*10]
```

```
$wl_node set Y_ [expr 550.0-$i*10]
$wl_node set Z_ 0.0
$ns at 0 "$wl_node setdest 1060.0 550.0 1.0" puts "wireless node $i created ..."
[$wl_node set mac_(0)] set-channel 0 [$wl_node set mac_(0)] set-diuc
$diuc
```

The above lines are used for the following

i. Creating the mobile node and setting the location of the node in the topology created
ii. Disabling the random motion of the nodE
iii. Establishing the connection with Base Station(BS)
iv. Setting the motion for the mobile node
v. Setting the channel and Downlink Interval Usage Code(DIUC)

```
set udp [new Agent/UDP]
$udp set packetSize_ 100
$ns attach-agent $wl_node $udp
```

These lines set up the UDP agent for each of the mobile node

```
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 100
$cbr set rate_ 0.2Mb
$cbr attach-agent $udp
```

The above lines help in setting up a Constant Bit Rate (CBR) traffic source ,which is then attached to a UDP agent ('udp' in this case). The mobile nodes cannot exchange data on their own. They must have agents attached to them for this purpose. In simple words, traffic sources create traffic (packets), and agents exchange them.

```
# Create the Null agent to sink traffic set null [new Agent/Null]
$ns attach-agent $sinkNode $null
# Attach the 2 agents
$ns connect $udp $null
```

The above lines set up a Null agent[13] (whose default behavior is to receive packets). Then the UDP agent and the Null agent are connected to establish a link between two nodes.

The above 4 code snippets are placed inside a for loop, which iterates 'n' number of times , where n is the number of mobile stations i.e. the for loop creates n mobile stations.

```
$ns at $traffic_start "$cbr start"
$ns at $traffic_stop "$cbr stop"
```

The above lines initiates and terminates the traffic source which will be in a for loop whose termination condition will be number of nodes at the time specified in the variables traffic_start and traffic_stop.

```
proc finish {} {
        global ns tf out.txt nb_mn nf
        $ns flush-trace
        close $tf close $nf
        exec nam newrng-irmsa.nam &
}
```

This procedure is used to flush the trace of NAM output and close all the open file descriptor which was used for trace file and nam trace file. It also executes the nam file.

The above indicates to the simulator to stop the simulation at the time specified in variable simulation_stop.

```
$ns at $simulation_stop "finish"
```

```
$ns run
```

The above line marks the starting point for the simulation.

B. Shell Script[14],[15]

1. for nodes in 4 8 16 32 40

This line initializes an array which contains the number of nodes for which we are going to be running the simulation(s).

2. for seed in 1973272912 1822174485 1998078925 678622600 999157082 934100682 558746720 2081634991 144443207 513791680

This line initializes an array which contains the seed values for which we are going to be running the simulation(s).

3. ns newrng-irmsa.tcl $nodes $seed > out.txt

This line runs the ns command for the file newrng-irmsa.tcl, passing $node and $seed as parameters. The output is redirected into the file out.txt.

4. grep "found ranging opportunity" out.txt | cut -d " " -f 2 > startTime.txt

This line searches the file out.txt for lines containing "found ranging opportunity", which indicates that a ranging opportunity has been found at the specific times. The lines thus selected are passed to the cut function, where we select the 2nd column (field). The values in this column are stored in startTimes.txt, which thus contains the times at which the nodes start the initial ranging procedure.

5. grep "Ranging response" out.txt | cut -d " " -f 2 > endTime.txt

This line searches the file out.txt for lines containing "Ranging Response", which indicates that a ranging response has been found at the specific times. The lines thus selected are passed to the cut function, where we select the 2nd column (field). The values in this column are stored in endTimes.txt, which thus contains the times at which the nodes finish the initial ranging procedure.

6. nl -n ln endTime.txt | tr -d '\t' | tr -s ' '> ourNewEndTimes.txt nl -n ln startTime.txt | tr -d '\t' | tr -s ' '>
ourNewStartTimes.txt

These two lines add lines numbers, deleted tabs and compress spaces in the files startTimes.txt and endTimes.txt, and store the results in ourNewStartTimes.txt and ourNewEndTimes.txt respectively. The adding of lines, deletion of tabs and compression of spaces is needed for further steps.

```
8. lineVar=0
   total_delay=0
   while [ $lineVar -lt $nodes ]
   do
         #echo "$total_delay + ${delay[$li
         total_delay=`echo "$total_delay +
         lineVar=`expr $lineVar + 1`
   done
```

These lines calculate the total delay, for a particular value of nodes and seed. The total delay is simply the sum of delays for individual nodes.

9. avg=`echo "scale=10; $total_delay / $nodes" | bc`

This line calculates the average delay for a particular value of nodes and seed. The average delay is the total delay, divided by the number of need.

7.newCounter=1

```
while [ $newCounter -le $nodes ]
do
endt=`grep ^"$newCounter " ourNewEndTimes.txt | cut -d " " -f 2` startt=`grep ^"$newCounter " ourNewStartTimes.txt | cut -d " " -f 2` delay[`expr $newCounter - 1`]=`echo $endt - $startt | bc` newCounter=`expr $newCounter + 1`
done
```

These lines calculate the delays for all nodes. The start times are obtained from ourNewStartTimes.txt and the end times are obtained from ourNewEndTimes.txt. The delay for a node is the difference between the end time and start time for that particular node. We find the delay for all nodes by using a while loop, which iterates n number of times, where n is the number of nodes for this particular simulation. The delays are stored in an array called delay.

10. Echo "$nodes $seed $avg" >> finalAvg$nodes.txt

This line send the number of nodes, seed value and average delay for a particular simulation into a file called finalAvg$nodes.txt. The average delay for 4 nodes, and different seeds, will be stored in a file called finalAvg4.txt, and the average delay for 8 nodes, and different seeds, will be stored in a file called finalAvg8.txt, and so on.

11. Steps 3 to 10 are performed for all seed values in the array called seed (10 values), using the variable seed.

12. Steps 3 to 11 are performed for all node values in the array called node (5 values), using the variable nodes.

IV. SIMULATION RESULTS

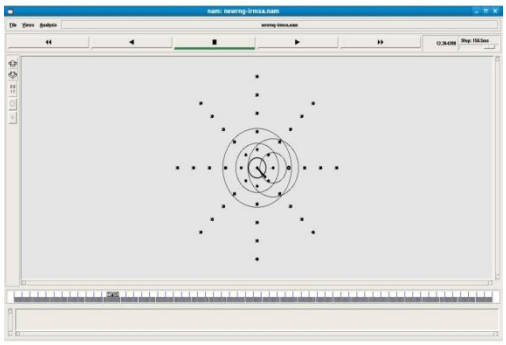

Fig.3. Circular Topology with No Motion

No. of nodes	Average Delay
4	0.107043
8	0.2336790937
16	0.3319343906
32	0.5845947968

Table 3- Average Delay for Circular topology with no motion

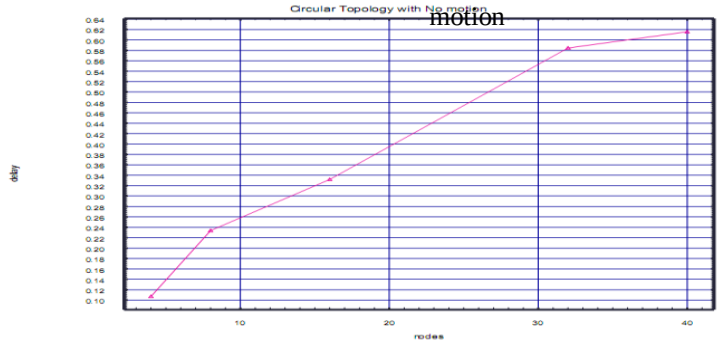

Fig.4. Delay Graph for Circular Topology with No Motion

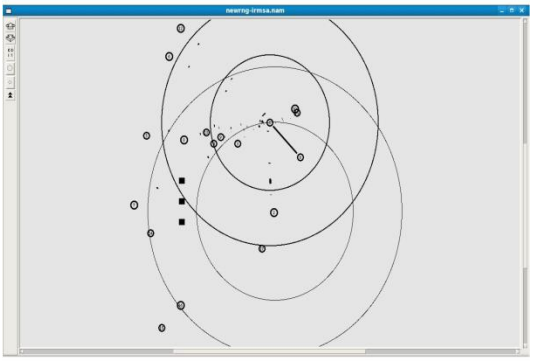

Fig.5. Circular Topology with Random Motion

No of nodes	Average Delay
4	0.1060478750
8	0.1580645937
16	0.4885501400
32	0.6211589296
40	0.63626995621

Table 4-Average Delay for Circular topology with random motion

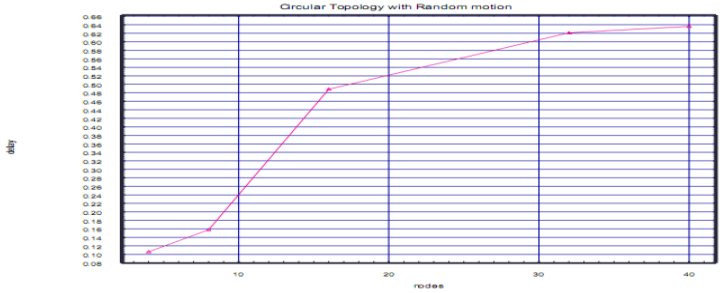

Fig.6. Delay Graph for Circular Topology with Random Motion

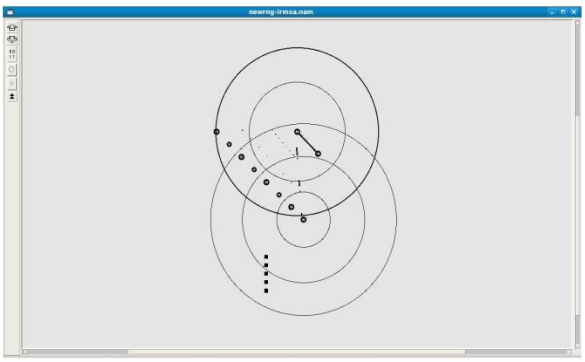

Fig.7. Linear Topology with Linear Motion

No of nodes	Average Delay
4	0.1070430000
8	0.2033068437
16	0.4859907031
32	0.6827474921
40	0.6965447375

Table 6-Average Delay for Linear topology with linear motion

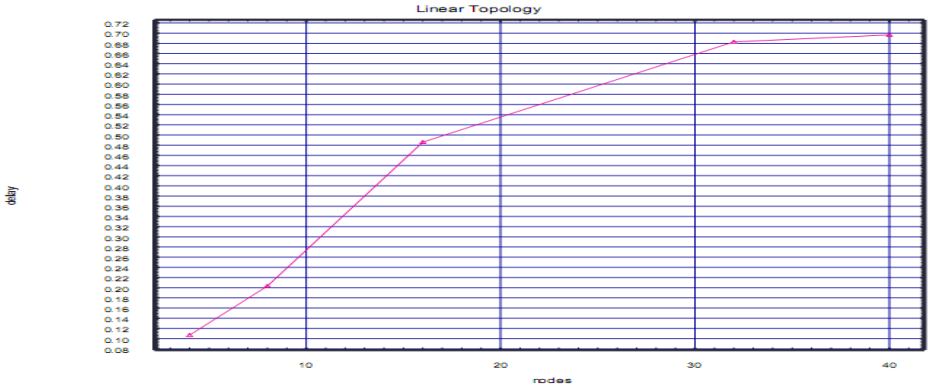

Fig.8. Delay Graph for Linear Topology with Linear Motion

REFERENCES

[1] Fundamentals of WiMAX by Jeffrey G. Andrews, Arunabha Ghosh, Rias Muhamed
[2] IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems by LAN/MAN Standards Committee of the IEEE Computer Society and the IEEE Microwave Theory and Techniques Society
[3] The Design and Implementation of WiMAX Module for ns-2 Simulator by Jenhui Chen,Chih-Chieh Wang,Frank Chee-Da Tsai,Chiang-Wei Chang,Syao-Syuan Liu,Jhenjhong Guo, Wei-Jen Lien, Jui-Hsiang Sum, and Chih-Hsin Hung
[4] The ns Manual by the VINT project
[5] Evaluation and Enhancement of the Initial Ranging Mechanism for MAC 802.16 inWiMAX Networks by R. Bhakthavathsalam and Aniruddha Niranjan
[6] The Network Simulator NS-2 NIST add-on IEEE 802.16 model (MAC+PHY)
[7] Tutorial for the Network Simulator "ns" by Marc Greiss
[8] cl/Tk: A Developer's Guide by Clif Flynt
[9] On Shortcomings of the ns-2 Random Number Generator by Bernhard Hechenleitner and Karl Entacher
[10] Experiences with the ns-2 Network Simulator - Explicitly Setting Seeds Considered Harmful by Martina Umlauft and Peter Reichl
[11] Deploying Mobile WiMAX By Max Riegel, Aik Chindapol, Dirk Kroeselberg
[12] Introduction to Network Simulator NS2 by Teerawat Issariyakul and Ekram Hossain
[13] NS Simulator for beginners by Eitan Altman and Tania Jimenez
[14] WiMAX Networks: Techno-Economic Vision and Challenges by Ramjee Prasad, Fernando J. Velez
[15] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4446432&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4446432