Dhaka Univ. J. Sci. 58(2): 145-149, 2010 (July)

Implementation and Performance Analysis of a WiMAX Module in ns-2

Tahmina Zebin, Sanwarul Hoque, and Shahida Rafique

Department of Applied Physics, Electronics & Communication Engineering, Dhaka University, Dhaka-1000, Bangladesh

E-mail: tlzebin@yahoo.com

Received on 27.08.2008. Accepted for Publication on 04.02.2010

Abstract

WiMAX is a very promising wireless broadband technology. For all new technologies, performance studies are required and network simulation is considered as a solution to test the performance of any upcoming technology. In ns-2 (a very popular and open source network simulator), may protocol modules (like 802.3, 802.11, 802.15) have been implemented, but the IEEE-802.16 (WiMAX) module have not been incorporated yet (till version ns-2.32). In this paper, a WiMAX module for the ns-2 simulator (v 2.32) has been implemented. This module is based on the analysis presented by Chen et.al,2006. Some simulation scenarios are also been analyzed to test the module and the same simulations are also been done on NCTUns-4.0 that has an implemented 802.16d module.

Keywords: WiMAX, IEEE-802.16, ns-2, NCTUns-4.0, Simulation.

I. Introduction

WiMAX -which stands for-Worldwide Interoperability for Microwave Access, is a very promising wireless broadband technology. It enables efficient data multiplexing and low data latency for broadband data services including streaming video and VoIP with high quality of service (QoS) by providing high data throughput. WiMAX is going to substitute other broadband technologies and has become an excellent solution for the deployment of the well-known last mile infrastructures in places where it is very difficult to get with other technologies, such as cable or DSL, and where the costs of deployment and maintenance of such technologies would not be profitable. WiMAX could connect rural areas in developing countries as well as underserved metropolitan areas. It can even be used to deliver backhaul for carrier structures, enterprise campus, and Wi-Fi hot-spots providing with a cost-effective, rapidly deployable solution [2]. Fig 1. shows some of the deployment areas for WiMAX. WiMAX is being considered as a serious competitor to 3G (third generation)/ 4G cellular systems as high speed mobile data applications can be achieved with 802.16e and other higher amendments [14].

WiMAX technology mitigates the problems [4] resulting from NLOS conditions by using:

- OFDM technology.
- Sub-Channelization.
- Directional antennas and Adaptive antennas.
- Transmit and receive diversity.
- Adaptive modulation and MIMO (multiple input/multiple output) support.
- Error correction techniques and power control

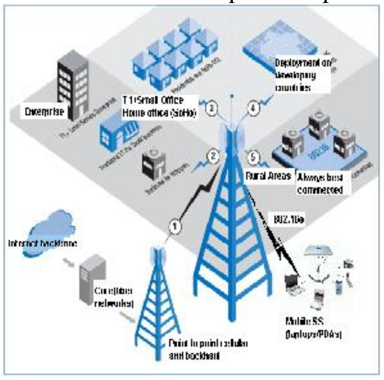

Fig. 1. WiMAX deployment areas.

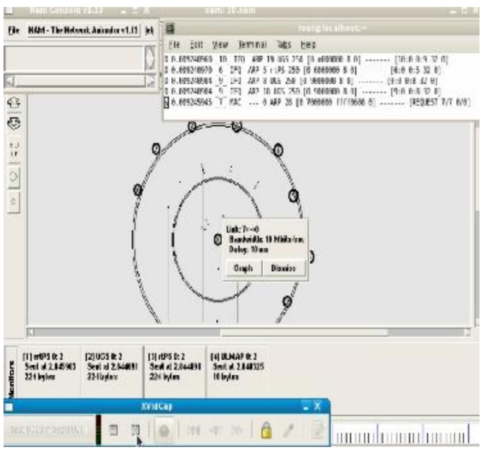

Fig. 2. Working environment in ns-2.32

II. Network Simulation

Network simulators attempt to model real world networks. They will be close enough so as to give a meaningful insight into how the network will work, and how changes will effect its operation. Some of the popular network simulators include OPNET modeler, OMNET++, QUALnet, ns-2, NCTUns etc. Amongst them OPNET (Not available for mass use) and NCTUns (open source-still under development) has built in WiMAX modeler. In this paper, We have used the wireless and mobile infrastructure of ns-2 to implement a workable WiMAX module. The network simulator 2 (ns-2) is a popular and powerful simulation tool for the simulation of packet-switched networks, which provides substantial support for simulation of TCP, routing, and MAC protocol over wired and wireless networks and is widely used in both academic and industrial studies[8].

A. Softwares

- ns-allinone-2.32, nam-1.14(network animator for ns-2) and NCTUns-4.0, xvidcap (to capture simulation instances as shown in Fig 2.) on fedora core 8.
- Tracegraph205Windows along with MATLAB7.1 on windowsXP to analyze tracefile generated by ns-2 simulations.
- Some graphs are also plotted in Microsoft Excel- 2007 with data collected from the tracegraph result on Matlab.

B. Simulator Setting (in Fedora core 8)

ns is an object oriented simulator, written in C++, with an OTcl (Object oriented tool command language) interpreter as a front-end. *ns* uses two languages because simulator has two different kinds of things it needs to do. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. *ns* (via tclcl) provides glue to make objects and variables appear on both languages. ns documentation is available at [9]. The open source ns-allinone packages are available at [10].

ns-2 Installation Steps: One must enter in the graphical mode of fedora to use the simulators. The gcc and gtk library packages must be installed.

1. First the zipped ns-allinone-2.32.tar.gz file is extracted under the directory /bin [it can be any directory other than /bin].

```
$ cp /media/pen/ns-allinone-2.32.tar.gz /bin

$ tar xzvf ns-allinone-2.32.tar.gz
```

2. Then after entering the extracted directory , a script named “install” available there is executed to install tcl, tk, otcl library and then build ns and nam[network animator].

```
$cd /bin/ns-allinone-2.32

$./install
```

3. To run ns without error: The PATH, LD_LIBRARY_PATH and TCL_LIBRARY environment variables should be set in the /root/.bashrc file. If the user is using /bin/bash shell .bashrc contains all the bash shell variable to be started at the beginning of the shell.

```
$gedit .bashrc

PATH=$PATH:/bin/ns-allinone-2.32/bin

export PATH
```

4. Then any network scenario written in tcl script can be run by executing the command:

```
$ns example.tcl
```

5. The trace of the network can be stored in an example.tr file to further analysis. We analyzed the trace file in matlab7.1 with the help of TraceGraph 2.05 available at [12]. The matlab working directory should be changed to the directory containing Tracegraph205Windows which contains the necessary .m files to analyze the trace files.”trgraph dir:/example.tr” is typed on the matlab command window to have tracegraph to get & analyze some built in performance curve based on the input trace file.

6. NAM can view and monitor the packet transfer between different nodes in the playback form. The nam instance can be separately executed once the simulation is done, by

```
$nam example.nam
```

NCTUns-4.0 Installation Steps: NCTUns-4.0 is a high fidelity and extensible network simulator and emulator[11]. This simulator has a built in 802.16d PMP and MESH mode protocol stack [simulation environment obviously different than our ns- module] with rich tools to set the topology. It has a graph-plotting tool at MENU--G_TOOLS--PLOT _GRAPH. The installation steps of a NCTUns simulator are:

1. First the file named "NCTUns-allinone-linux-2.6.23.8-fc8.20071205.tar.gz" is unzipped in /root directory.

```
$tar xzvf NCTUns-allinone-linux-2.6.23.8-fc8.20071205.tar.gz
```

2. Execute the install.sh script available in /root/NCTUns-4.0

```
$ cd /root/NCT Uns-4.0/
$ ./install.sh
```

After completing the installation restart the pc and enter into the new NCTUns-4.0 kernel.

** Must disable the firewall and SELinux before install.

** Must login as root user.

3. After installation. to run the NCTUns-4.0,add thefollowing in .bashrc

```
$ gedit /root/.bashrc
source /usr/local/nctuns/etc/nctuns.bash
```

Open a terminal/console and run the dispatcher

```
$/usr/local/nctuns/bin/dispatcher
```

Run the coordinator on another terminal:

```
$/usr/local/nctuns/bin/coordinator
```

Run nctunsclient on another terminal

```
$/usr/local/nctuns/bin/nctunsclient
```

Nctunsclient will run the simulator on the screen. Then a simulation scenario can be drawn in it in the draw mode[D], different node properties can be set in the edit[E] mode, simulation can be run on the run[R] mode. Once the simulation is done, playback of the simulation can be seen in playback[P] mode which shows visual connection and packet transfer in the network. Xvidcap can be used here to capture the thorough simulation steps. NCTUns is less configurable for a general user; it requires a deep study of the developer manual available at [10] to bring change in any predefined stack.

ns-2 WiMAX Module

The WiMAX module implemented here has been developed in unified modeling language (UML)[1,5]. This module is based on IEEE 802.16-2004 point-tomultipoint (PMP) mode, which means that one BS can serve multiple subscriber stations (SSs) concurrently. We choose the orthogonal frequency-division multiple access (OFDMA) schemes for the physical (PHY) layer. Based on the OFDMA PHY specifications, it has been of major interest for both wireless applications due to its high date rate transmission capability and its robustness to multipath delay spread [13,15].

This 802.16-based WiMAX module named as the Mac802_16 class is in accordance with the specifications of the IEEE 802.16-2004 standard [3] and based on the ns-2 version 2.32 [6]. All modules are designed by using object oriented programming language C++ and modeled as several classes. The relationship between the WiMAX module and legacy ns-2 modules is based on the original network component stack of the ns-2 as shown in Fig 3. It illustrates the type of objects for the traffic generating agent (TGA), the link layer (LL), the interface queue (IFQ), the designed MAC layer (WiMAX module), and the PHY layer (Channel).

The MAC layer in WiMAX basically provides intelligence to the PHY layer. It contains 2 sub layers: service-specific convergence sub-layer (CS) and the MAC common part sub-layer (CPS).

The service-specific convergence sub-layer is responsible for interfacing with upper layers while the MAC common part sub layer carriers out the key MAC functions [7].

A.Service Specific Convergence Sub-Layer

The IEEE 802.16-2004 standard specifies 2 types of service specific convergence sub-layers for mapping service to and from the MAC layer; the ATM sub-layer for mapping ATM services and the packet sub-layer for mapping packet services such as IPv4, IPv6 and Ethernet. The main task of the sub-layer is to map Service Data Units (SDUs) to MAC connections, and to enable QoS and bandwidth allocation based on the parameters received from the upper layers.

B. The MAC Sublayer

The MAC CP(Common Part) sublayer is the main part of the MAC and maintains the MAC operations and management messages of the system. The management messages such as DCD/UCD(Uplink /Downlink Channel descriptor), DL-MAP/UL-MAP(Media Access Protocol), DSA, DSC, DSD, RNG-REQ(range request), RNGRSP(range response), and so forth are generated in this sublayer. The main body of the MAC CPS is constructed by a Mac802 16 class, which contains several independent functions such as Ranging(), Fragmentation(), BandwidthRequest(), and so forth. Some brief descriptions of these functions are provided here:

Ranging: This is the first step for the subscriber station to enter the network. A new SS has to scan for the DL(Downlink) channel and establish synchronization with the BS. After synchronizing with the BS, the SS will obtain transmit parameters from the UCD message, which is periodically generated by the BS, to recognize the channel information for transmission. The ranging function also handles the backoff mechanism, sets the contention window size to avoid collision, provides a CID (connection ID) to the SS after being registered for future communication.

MAC Management: Five kinds of messages, DCD, UCD, DL-MAP, UL-MAP, and bandwidth request (BR) are used in this function.

# 802.16 mac layer is implemented in mac-802_16.{h,cc} and all the packet types needed are defined in packet-802_16.{h,cc} and the defined packets should also be included in ~/ns-2.32/common/packet.h. (.h=header files, .cc=c++ files)

Priority queue: In the BS or SS, the packets that come from the upper layer will be prior delivered to Priority(). According to the TCID and its service type: UGS(5), rtPS(4), ertPS(3), nrtPS(2), BE(1), the Priority() will make a corresponding priority classification.

#The queuing function for different QoS traffic are implemented by adding necessary lines to /bin/nsallinone-2.32/ns-2.32/queue/queue.h

Scheduler: The Scheduler() function is in charge of selecting queued MSDUs according to the admitted bandwidth. The selection policy of the scheduler in the designed module uses the weighted Round-Robin method.

Fragmentation/Packing: This function will handle addition of Generic MAC header to payload depending of QoS. This section will handle transmission of the data also.

Timer Function: The 802.16 Timer class inherits the Handler class with three important functions:

• The start() is used to trigger the timer to start.

• The stop() is used to stop the timer if the event happened before expiration.

• The handle() is used to trigger event while time runs out.

# The Timer class is implemented in timer-802.16.h and timer-802.16.cc within the WiMAX module according to the ns-documentation [6]. To be noted: Timer classes must be derived from an abstract base class TimerHandler defined in *~ns-2.32*/timer-handler.h

Table 1. shows some of the important MAC layer and System time parameters used in the module.

Table. 1. Some of the important parameters used in the module

MAC layer Parameters	
DL/UL Ratio	'3' : '2'
CWmin [contention window size]	32
Cwmax	1024
Max. no. of ranging retry	10
Max. no. of bandwidth req. retry	10
System time parameters	
Spectrum	5.0 GHz
Bandwidth 20 MHz	20 MHz
OFDMA symbol time	100.84 µs
OFDMA frame length	5 ms
DCD/UCD period(downlink/uplink channel descriptor)	10 µs
Ranging interval	1210.08 µs
Bandwidth request interval	1210.08 µs
TTG	200 µs
RTG (Receive transmit gap)	200 µs

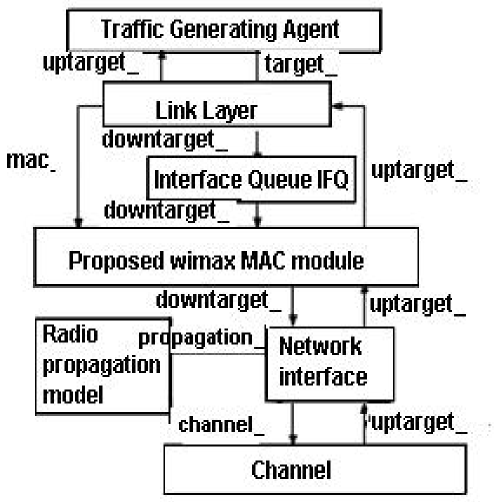

Fig. 3. The relationship between WiMAX module and the legacy ns-2 module[1]

IV. Simulation Scenario & Result Analysis

During this course of work we have simulated PMP WiMAX environments with different number of subscriber stations and studied responses of various parameters.

A. Topology/Scenario writing in tcl scripts in ns:

- Different Environment parameters can be set in a tcl file like "example.tcl" as follows:

```
Set  val(chan)       Channel/WirelessChannel;
                    # channel type
set  val(prop)       Propagation/TwoRayGround
;           #radio-propagation model
set  val(netif)      Phy/WirelessPhy ;
#network interface
set  val(mac)        Mac/802_16 ;
            # MAC type
set  val(ifq)        Queue/DropTail/PriQueue         ;
            # interface queue type
set  val(ant)        Antenna/OmniAntenna ;
            # antenna model
set  val(ifqlen)     50 ;
            # max packet size
set  val(nn)         6 ;
            # no of mobile SS
set  val(rp)         DSDV ;
# routing protocol
set  val(stop)       15.0 ;
            # simulation end time
```

- Connection type between different nodes can be provided like follows:

```
set udp3 [new Agent/UDP];

$ns attach-agent $node_(4) $udp3; # connection type
for node 3 to node 0(UGS-UDP)
set cbr3 [new Application/Traffic/UGS];
$cbr3 attach-agent $udp3
$cbr3 set type_ UGS
$cbr3 set packet_size_ 1000;
$cbr3 set rate_ 512Kb
$cbr3 set random_ false
$ns at 3.0 "$cbr3 start"
    $ns at 10.0 "$cbr3 stop"
```

Further detail on topology creation/tcl script writing can be found in reference [9].

B. Performance Analysis:

We have studied different cases to test the performance of the implemented module. Some of these are presented below:

Observation 1. Fig 4, shows the throughput as a function of simulated time. The simulation time persists for 15 seconds. All the traffic were continuously generated throughout the simulation time. We can see that the curve of the obtained throughput of the system rises with the increase of the simulation time.

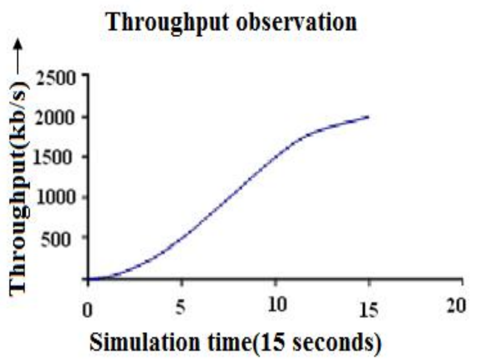

Fig 4. The System throughput versus simulation time in 15 seconds

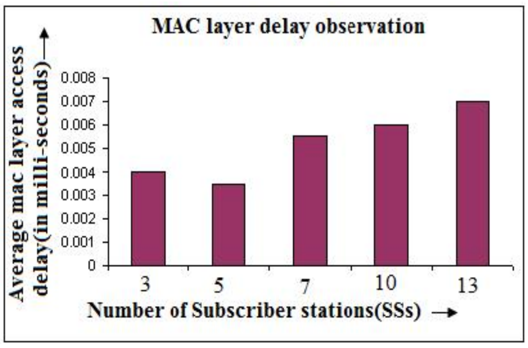

Fig 5. Variation in average MAC layer access delay for different number of SSs(PMP)

The reason why the system throughput is low in the early simulation time is that SSs spend an amount of time in dealing with the initial ranging process and their bandwidth request process. It will then follow the random backoff mechanism with an initial backoff countdown interval and stand on the random backoff approach. Therefore, the system throughput is not revealed well. Finally, we can see that the system throughput will reach around 2 Mbps in the end of the simulation time.

Observation 2. From the bar diagram shown in Fig 5, we can see that the delay time will increase with increasing number of SSs. The MAC delay is a result from the collision of initial ranging and the bandwidth request. After the bandwidth request, the delay will become de balanced since the BS will arrange the transmission time in the simulation. The scheduling of different traffic service types will mainly affect the MAC delay because of limited spectral resource. Using the weighted Round Robin (WRR) to access the priority queues is not a better way to schedule the transmission and it can be replaced to a better scheduler (We are trying on Earliest deadline first / Weighted fair queuing as uplink scheduler).

To be noted here, the tracegraph tool used with MATLAB should be modified to provide graphical results from large trace files.

Observation 3. The WRR algorithm indicates very high average delay for the ertPS class except when the concentration of ertPS SSs is the highest (Fig 6). With a high concentration of ertPS SSs, the weight assigned to them is high resulting in more bandwidth allocated to the SSs. The Queuing Algorithm indicates a high average delay hence the packet loss for the ertPS SSs when their concentration is low.

Observation 4. Though ns module we used and NCTUns WiMAX implemented module uses different way of designing - direct scenario comparison is not possible[13]. In Fig 7, an optimal comparison in the simulation processing time and memory usage of both the simulators is shown. Since NCTUns deals with higher graphics it takes greater execution time and uses more memory than ns.

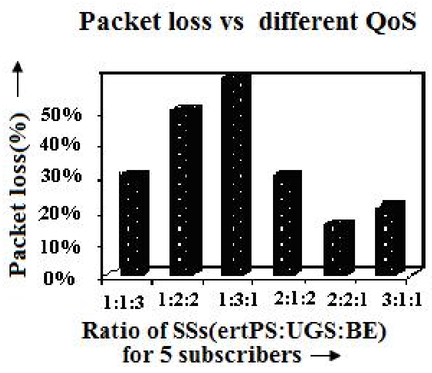

Fig 6. Packet loss variation with QoS based SS ratio

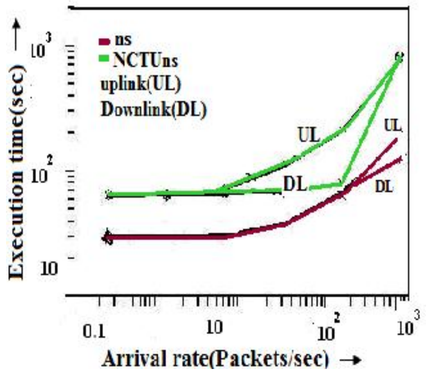

Fig 7. Execution time comparison between ns and NCTUns Simulation

V. Conclusion & Future Work Plan

This paper presents a detailed description and implementation of the WiMAX simulation module for ns-2. The module is based on the IEEE 802.16-2004 standard and the legacy ns-2 version 2.32. This module deals with a basic point-to-multipoint (PMP) IEEE 802.16 function. It comprises fundamental functions of the service-specific convergence sublayer (CS), the MAC common part sublayer (CPS), and the PHY layer. Traffic generators for different QoS classes (UGS, rtPS, nrtPS, BE) are also included in this module**.**

We hope that this preliminary work will help amateur ns workers to get started and use ns in network simulation of any supported form.

In future we hope to work on efficient bandwidth management algorithm and scheduler algorithms to have more resemblance to real networks. Moreover we hope to expand a fully workable mobile WiMAX IEEE802.16e module in ns and NCTUns. Also hope to work on Mobile IP and satellite modules in ns and NCTUns.

----

1. J.Chen, C.Wang, F.Tsai, C.Chang, S.Liu, J.Guo, W.Lien, J.Sum, and C.Hung, (Chen et.al,2006)"The design and implementation of WiMAX module for ns-2 simulator" Proceedings of the 2006 Workshop on Ns-2: the IP Network Simulator.(http://ndsl.csie.cgu.edu.tw)
2. Intel White paper, Wi-Fi and WiMAX Solutions: "Understanding Wi-Fi and WiMAX as Metro-Access Solutions". (http://www.intel.com/netcomms/technologies/wimax/304471.pdf)
3. Documentation of IEEE 802.16-2004, "IEEE Standard for Local and Metropolitan Area Networks – Part 16: Air Interface for Fixed Broadband Wireless Access Systems",2004.
4. Pratik Dhrona, "A Performance Study of Uplink Scheduling Algorithms in Point to Multipoint WiMAX Networks".Report presented at Queens University, Canada,2007.
5. Amalia Roca, "Implementation of a WiMAX simulator in Simulink", 2007, Castellón - Spain.
6. The Network Simulator NS-2 NIST add-on IEEE 802.16 model (MAC+PHY), 2007
7. S. Kim and I. Yeom, "TCP-aware Uplink Scheduling in IEEE 802.16," *Communications Letters*. 2006.
8. WiMAX Forum Documentation," Mobile WiMAX: A Technical Overview and Performance Evaluation",2006.
9. http://www.isi.edu/nsnam/ns/ns_doc.pdf,

   (ns-2 documentation; collected at March, 2008)
10. http://www.sourceforge.net.
11. http://NSL.csie.nctu.edu.tw/nctuns.html.: For NCTUns download and user/developer manual.
12. http://www.tracegraph.com, http://www.geocities.com/tracegraph
13. Shiang-Ming Huang,Ya-Chin Sung, "NCTUns Simulation Tool for WiMAX Modeling",2007.
14. Jeffry G. Andrews, Arunabha Ghosh and Rias Muhamed "Fundamental of WiMAX, Understanding broadband wireless networking",1st Edition,2007, 8-31.
15. K.Lee, J.Hahm and Y.Kim, "QoS Application Method in Portable Internet", Proceedings of Asia-Pacific Conference on Communications, 2005, 237-239.