Ns-3 Based WiMAX Emulation System

Muhammad Imran Tariq
School of Information Science,
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, Japan
Email: imran@jaist.ac.jp

Razvan Beuran
Hokuriku StarBED Technology Center,
National Institute of Information and Communications Technology
2-12 Asahidai, Nomi, Ishikawa, Japan
Email: razvan@nict.go.jp

Yoichi Shinoda
Research Center for Advanced Computing Infrastructure,
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, Japan
Email: shinoda@jaist.ac.jp

***Abstract*—Evaluating a system in real-time to test its performance has always been a challenging task. Although real-world wireless testbeds offer great value, on the other hand they are expensive in terms of hardware cost and very hard to maintain. Network simulations are synthetic environments for running representations of mathematical models. However, network emulation studies are preferred for system modeling and its evaluation in a real-time environment.**

In this paper, we present the design and implementation of a WiMAX emulation system. By making use of the open-source ns-3 simulation framework, we developed an emulation system of WiMAX entities, protocols and procedures. Our approach provides the means of transferring real application data on real network between ns-3 simulation entities, and allows physical hosts to connect with the simulated WiMAX network. Thus, the proposed system offers realism and accuracy similar to that of hardware with the cost, configurability and scalability of simulation. To show the feasibility of our design, we include several validation experiments, including using real application data in the emulated environment. Through this system we ease the prototyping of technology that will bring us closer to the realization of the present and future mobile Internet.

Keywords—WiMAX, network emulation, real-time measurement, ns-3.

I. INTRODUCTION

The advances in broadband wireless access (BWA) standards like IEEE 802.16 WiMAX [1] have achieved a great attestation in recent years. Many carriers all over the world have deployed these next-generation networks. In respect to this development, more and more opportunities for applications and service models are being provided.

The key concern of these wireless systems is the development as well as realistic performance evaluation of the applications and protocols running over these systems. The realistic evaluation has always been a challenging task. In this regard, there are two key concerns. First, it is fundamental to reiterate the experimental process multiple times in a deterministic manner. Second, a major constraint is the capability/facility of investigating these kinds of systems in a real-world environment [2]. Typically, testing of the respective system and the performance evaluation under real conditions are performed within a wireless testbed of prototype implementation.

In order to model novel and existing networking protocols and algorithms, researchers mostly rely on wireless testbeds or simulation to execute experiments. *Network simulation* is an invaluable approach for researchers; it allows us to model the network, equipment and applications of a whole wireless system. Primarily, renowned simulators, such as ns-2 [3] or ns-3 [4], QualNet [5], OPNET [6] and OMNeT++ [7] are designed to control and reproduce the condition of various wireless networks in a scalable fashion. Particularly, discrete event simulation technique due to its trait of formulating state transition as sequences, allows to model telecommunication networks. Although simulation is a widely adopted technique of evaluation in academia or research communities, but its accuracy depends on the accuracy of the used models. Abstract simulation models frequently make simplified assumptions of the real world. Often, they neglect the system framework of networking protocols and run-time environments, such as that of an underlying operating system, regarding timing synchronization, concurrent processes, resources limitation [8]. These issues are critical in understanding the system behavior seen in real-world situation. As simulation is lacking the fundamental concept of realism, also its abstraction limits its applicability in measuring network performance.

Testbeds are ideal solutions to evaluate the wireless system under real conditions and has become the source of both reproducible and realistic results. But, constructing such testbeds is expensive in term of hardware and labor cost. Moreover, the interaction between higher layers of protocol stack and Medium Access Control (MAC) wireless layer made us realize how difficult is it to extract meaningful information from protocols implemented in hardware or within operating system kernel. In addition, insufficient scalability hinders the study of legacy wireless networks.

Network emulation [9] [10], is a technique to scale up the experiment size cheaply, maintain a high degree of realism, increase controllability and reproducibility. A network emulator may be considered a hybrid investigation environment where a testbed or simulation are used together and synchronized to a real-world clock. For this purpose, real-world machine passes traffic through simulated network that reproduces the behavior of a communicating network. This way, by imitating packet propagation characteristic within the simulator, it enables simulated protocol modules to interact with the standard implementations.

To satisfy such needs, in this paper we design and implement an ns-3 based emulation system for the evaluation of WiMAX network. More specifically, the contributions of this paper are threefold that can be summarized as follows:

978-1-4799-3711-0/14/$31.00 ©2014 IEEE

- We provide emulation support of the existing physical layer in the WiMAX module. By doing this, physical layer receives from and transmits to the real network. To achieve this goal we implement the following functionalities: first, modify the burst's class to recreate the new burst out of the provided buffer and copy the burst's whole data into new buffer. We also implement a structure to carry the necessary parameters of physical layer to emulate exactly what is happening in the simulation or real-world, the second functionality is to integrate the WiMAX network with the real-world, we developed the real network communication interface to send packet over the real network, besides its real network communication interface to transform it into original burst with parameters.

- Our implementation of SITL (Simulator-In-The-Loop) for WiMAX network enables the communication with real time traffic between hosts through the simulated network. To bridge the MAC layer with a real host, we develop a real traffic communication interface which seamlessly integrates with the WiMAX MAC layer.

- Further, we present a comprehensive evaluation of our proposed system using various applications and protocols (UDP, TCP and Ping). Through the evaluation we identified and fixed few bugs to improve the application performance especially for TCP: 1) bandwidth request processing issue at BS, 2) packet size issue in mac queue, 3) bandwidth wastage issue at SS, and 4) ICMP packet dropping issue in packet classifier. All were fixed and support added to run ping applications.

This paper is organized as follows. In section II we introduce the design of the emulation system and its implementation in ns-3. In section III we present the experimental setup and result analysis. In section IV we briefly discuss the related work. Section V concludes this paper with remarks, future work and references.

II. System Design and Implementation

The aim of this section is to briefly describe a high level overview of the system in order to allow the emulation of WiMAX network. The proposed system is shown in Fig.1 and briefly explained below. This system is mainly composed of two key components: a) the real traffic communication interface which is an integration of TAP device with WiMAX module to pass complete frames from an underlying OS to the simulated network and vice versa, b) once traffic propagates through the simulated WiMAX network provided by ns3, frames are then passed to an ethernet through the real network communication interface. Furthermore, we review the existing implementation of ns-3 based WiMAX module and do few additional changes to seamlessly integrate these components into it.

A. Real Traffic Communication Interface (RTCI)

This component of the system leverages the TAP device emulation features [4] and integrates them into WiMAX module to provide communication between real host and the simulated network. By doing so, MAC layer has connectivity with real host and the legacy operating system applications should be enabled to send and receive from the simulation scenario. In the following we describe the basic characteristics of this component.

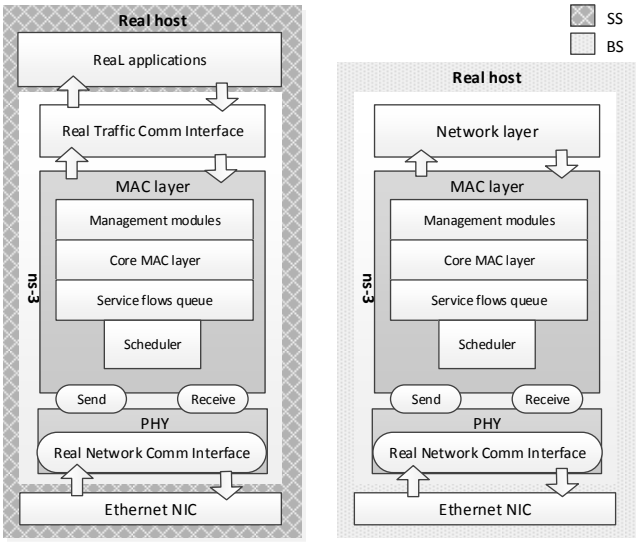

Fig.1. Overview of proposed emulation system for WiMAX network.

TAP device offers various modes of operation, but the most relevant are *LocalDevice* mode and *BridgedDevice* mode. In *LocalDevice* mode, ns-3 governs the creation and a configuration of virtual interfaces itself and local processes are attached with them. Whereas, *BridgedDevice* mode utilizes existing virtual interfaces facility and ns-3 connects to them. This mode is suitable for our requirements and recommended in [11]. As the execution starts, ns-3 process forks and generates child processes according to the number of nodes present in the emulation. The main process of ns-3 opens an IPC socket and refers the parent-side socket path to the child process. The sub process then connects to RTCI which is specified in ns-3 configuration script. The file descriptor of this device is sent via IPC socket back to the parent and child process ends [12]. As shown in Fig.1, by having the file descriptor of real traffic communication interface the MAC layer is bridged in the form of tunnel with legacy operating system for real traffic communication.

To ensure transparent communications, we create and configure a virtual interface in the underlying OS which is bind with the MAC layer, traffic generated by real applications is captured and passed through this interface and encapsulated in UDP datagrams. Moreover, UDP datagrams are sent over a socket towards the emulator host. At their destination, frames are de-encapsulated to be finally delivered to the proper structure connected to the MAC layer in the emulator. To reach the first goal, that is, real traffic interfacing with MAC-layer which is not provided in the standard module, therefore we had to extend the MAC-layer functionality.

B. Real Network Communication Interface (RNCI)

Our emulation system integrates instances of operating system executing networking resources into ns-3 instances. The most important cornerstone in our system is RNCI, to provide real-world communication with the physical layer and its connectivity with the simulated network. In the following we will explain in more details.

The major design requirement of our system is to enable the incorporation of live network traffic into an emulation scenario.

Normally, it is that assumed real-world communication interfacing into emulation system is not possible. However, interfacing real-world traffic with an emulation system is possible at different levels of the protocol stack. As Fig.1 displays how we made it possible. Our system design employs an RNCI that emulates a networking card. The concept of this design inspired by is *EmuNetDevice* and has been tuned to match our requirements. The creation and linkage of RNCI with real network is similar to RTCI. However, RNCI does the following two important operations to send and receive live traffic successfully.

1. *Packet transmission over real-network*

The application generates packets to be passed down and wrapped in appropriate layers on its way out of the system. When the burst (group of packets at PHY) is about to cross the boundary between ns-3 and underlying OS, the RNCI copies all channel parameters that are necessary for the receiver to peform further actions. Once burst is ready to be sent over the network, burst data structure with necessary parameters is written to the memory buffer reserved by RNCI.

As algorithm 1 highlights the major steps, RNCI copies whole burst data into buffer with parameters. Such as, ns-3 packet structure is equivalent to real network packet; also packet's data area is untouched and can not be modified. Thus, we create a packet out of burst from real network. In addition, layer 2 header is appended due to basic traits of *EmuNetDevice*. In the final round, structure of the created packet is written to the memory buffer. Then this structure is handed over the send function to be sent over the real network.

***Algorithm** 1: Packets transmission on the real network*

```
//Sending packets burst on real network with channel parameters
INPUT: PacketBurst, SrcAddr, Params
00: BEGIN
01:   INIT Protocol number
02:   INIT BufferSize DETERMINE  size of the whole Burst and Params
03:   GET Memory allocation as per BufferSize THEN
04:         CALL memcpy(copy Params in allocated memory)
05:   PUT Whole data of Burst
06:         CALL CopyData(Allocated memory buffer, size of data to be
copied)
07:   Create a Packet out of memory buffer
08:   FREE (Memory buffer)
09:   Build Ethernet header with source and destination address
10:   CASE m_enacpMode  OF
11:           LLC : Add (header, ProtoNbr, Size) BREAK;
12:           DIX  : Add ( ProtoNbr) BREAK;
13:    DEFAULT: ERROR
14:   ENDCASE
15:   PUT Whole data of Packet that just created out of memory buffer
16:         CALL CopyData ( PacketBuffer, size of data)
17:   CALL  sendto (socket, PacketBuffer, size of data, socket
parameters)
18: END
```

2. *Packet reception from real-network*

At the receiving end, data is received and queued on the ns-3 side; multithread process of ns-3 utilizes its read thread functionality of RNCI to asynchronously receive data. Once data is received, an event is created in the emulator thread. At a specific time interval, the emulator thread triggers a previously scheduled callback function of RNCI.

Algorithm 2 presents the overview of packet reception from the real network in RNCI. It creates a packet from memory, then free the memory space since the data was received, it also takes care of the pending thread to avoid memory overflow at OS level. Secondly, RNCI removes and records additional header of the newly created packet with protocol number and channel parameters. In the final stage, packet data is copied into the buffer. Finally, a burst is created out of this buffer and supplied to the physical layer with the necessary channel parameters to perform further actions.

***Algorithm** 2: Packets reception from real network*

```
// Receiving  packets burst from real network with channel parameters
INPUT: Buffer, size of Buffer
00: BEGIN
01:   DECREMENT Read threads
02:   Create a Packet out of received Buffer
03:   FREE (Buffer)
03:   CALL To remove header of newly created packet
04:   CASE Capsulation mode  OF
05:           LLC :
06                    IF (Packet size less than 1500) THEN
07                        Remove packet header and get Protocol number
08                    ELSE
09                        Get directly Protocol number
10                    END IF  BREAK;
11:           DIX  : Get protocol type BREAK;
12:    DEFAULT: ERROR
13:   ENDCASE
14:   IF (Protocol number is not equal to emulation protocol number) THEN
15:      Irrelevant Packet and RETRUN
16:   ELSE
17:      PUT  (Parameters out of received Buffer into allocated memory)
18:      CALL Create a burst  out of received buffer
19:      FREE (Buffer)
20:      CALL Receiving function of Phy layer and give to burst with
parameters
21:   END IF
22: END
```

C. *Additional changes in ns-3*

While designing the emulation system, we had to review the existing ns-3 WiMAX module thoroughly to implement the emulation features. To integrate these features in WiMAX module seamlessly, we improved in the existing code by adding new features and fixing few bugs. The most important features we added are emulation packet forwarding support at BS and data structure support to carry necessary channel parameters. And the bugs that were fixed in the existing module are: 1) ICMP protocol support in packet classifier; 2) in BS-uplink scheduler allocation of available symbols; 3) fixing available symbols issue in SS scheduler; 4) fixing of fragment size issue in WiMAX mac queue. Herein, we describe in detail the new features and bug fixes.

1. *Added features*

a) Packet forwarding to upper layer: In [13] [14], WiMAX module for ns-3 is proposed and improved. The first one is very simple with basic implementation and the latter is an improved version. However, the existing version lacks emulation capability. To overcome this limitation, we have designed RTCI which is also called SITL for real data communication only for SSs. On the other hand, BS has a role of administrator in the network; it communicates with upper layers and helps in packet forwarding to the network. Thus, it is meaningless to provide this functionality at BS end. Furthermore, by default RTCI receives packet in promiscuous mode and BS does not know about it.

To resolve this issue and avoid packet choking we add the additional functionality in WiMAX MAC layer to recognize the destined object. According to the design principles, when a packet arrives in this function with parameters, the function checks whether this packet is for BS or SS. If the arrived packet is for BS then non promiscuous function is called else destination address for that packet is discover and promiscuous function is called. In the end, packet is delivered to RTCI for further processing.

b) Data structure support for parameters: Existing module simulates OFDM channel in efficient manner with few parameters, those are block size, first or last block, receiver power, frequency and modulation. These parameters are calculated and sent by sender along with the burst. On the receiver side, these parameters are received and feed to PHY layer to perform the action based on them, such as SNR and error rate calculation based on received power and packet dropping based on frequency and the modulation mismatching.

To emulate same channel behavior in the emulation system these parameters are necessary at the receiver side. Therefore, we have developed a data structure to carry these mandatory parameters. In burst, which is already being delivered to RNCI these mandatory parameters are copied in developed data structure and also handed over to RNCI for other processing.

2. *Bug fixing in existing modules*

Since the main goal of this is to serve the emulation system, but another important concern regarding the real-time evaluation is perfect working of existing ns-3 module. During the development, we have found few bugs in the existing WiMAX code and we had to fix them briefly as follows:

a) ICMP support: The existing packet classifier maps incoming packets into appropriate service flow on a set of principles. A service flow ID (SFID) is associated with each flow. When packet is received by a classifier it checks if there is a record in packet matching rules list corresponding to the processed packet. If such record is found, then based on the corresponding SFID the packet is appended to the queue of the service flow. However, the existing classifier classifies only UDP and TCP based packets. It classifies a packet by matching protocol number. Hence, currently classifier implementation does not support ICMP based packets. In order to evaluate system using ping (real-world and simulated) applications, we fixed this bug by implementing ICMP support in the classifier to match ICMP packet in to appropriate service flow. Moreover, the service flow is also created based on the protocol number. Therefore, we also resolve this issue by providing the protocol number to the service flow manager of ICMP service flow.

b) Symbols allocation in BS-uplink scheduler: Existing WiMAX module supports three kinds of schedulers (simple, mbqos and rtps) all have different characteristics. Mostly researchers prefer simple scheduler due to straightfoward functionality. However, current implemenation of simple scheduler wastes a number of symbols and returns without bandwith allocation if available symbols are less than the requsted symbols. Often, it happens with heavy load. Thus, we have fixed it by allocating all availabe symbols instead of returning false. Now, simple scheduler allocates all available symbols, in case a request can not be fullfilled and works fine even in heavy load.

c) Symbols calculation in SS scheduler: Similar to BS-uplink scheduler, we have found a bug in SS scheduler. In current code, due to numerical errors number of requied symbols could be larger than the available symbols. Consequently, the allocated bandwidth resources are wasted by SS. To tackle down this bug, to make sure the symbols substraction does not lead to negative available symbols. If such case, we ignore this condition and assume all available symbols filled correctly and execution continues. Hence, SS scheduler works perfectly even with heavy traffic load.

d) Fragment size issue in MAC queue: When packet dequeued from MAC queue, a fragment size check is performed. If fragment size exceeds the packet size then segmentation fault error occurs, and halts emulation execution. This becomes common in real-traffic flows. To cope with this bug, we make sure fragment size does not exceed the packet size by performing a check. If so, we set fragment size equal to packet size. By doing that, the emulation system execution smoothly.

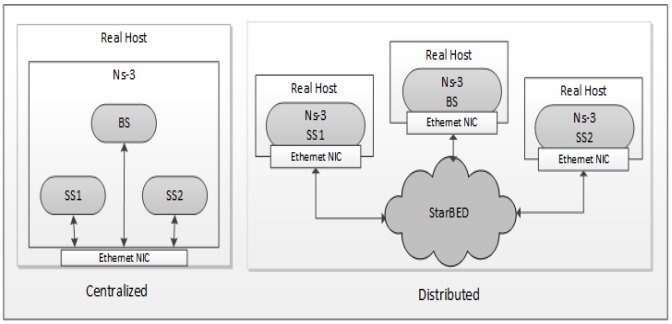

Fig. 2. Overview of centralized and distributed approaches.

III. EXPERIMENTAL RESULTS

This section mainly focuses on the evaluation of the emulation system that we designed and implemented. We have performed several tests in order to verify that the results of the experiments done on our emulation system are in accordance with those of the ns-3 simulation experiments. We also investigate the influence of the BS scheduler on performance.

A. *Experimental setup*

The emulation experiments that we present involve two kinds of approaches and their system level parameters are as follows:

- The two different approaches for simulation and emulation experiments are shown in Fig.2. First we begin with single machine. This is a MacBook Pro with 2.53GHz Intel Core 2 Duo processor and 4 GB of 1066MHz DDR3 memory. The operating system is Ubuntu LTS 12.04 (64bit). We have prepared five experimental scenarios for centralized and distributed approaches. Their details are as follow; 1) pure simulation represents the performance of existing WiMAX ns-3 module, 2) centralized emulation evaluates the capability of RNCI with simulated traffic, 3) SITL (Simulator-In-The-Loop) scenario evaluates the RTCI capability only in simulated network. We run all these three scenarios on a single machine using the centralized approach as depicted in Fig.2, 4) distributed emulation represents the performance of RNCI in a distributed environment, 5) distributed SITL shows the capability of RTCI and RNCI in a distributed environment.

The last two scenarios are run using the distributed approach as in Fig.2.

We use the distributed approach to run distributed emulation and distributed + SITL emulation. This approach spreads BS and SSs on different machines over a testbed that is called StarBED. StarBED is a large scale wired-network testbed at the Hokuriku StarBED Technology Center of the National Institute of Information and Communications Technology, located in Ishikawa, Japan [15]. The ns-3.18 release is used in both (centralized and distributed) approaches. Each scenario contains 3 nodes, 1 BS and 2 SSs.

- *WiMAX simulation/emulation parameters*: Are presented in Fig.2, we design an emulation system to model a single BS of two SSs (sender and receiver) connection through IEEE 802.16 air interface implemented in ns-3. Our major focus is to evaluate the real traffic in real-time WiMAX environment. In order to this, we use the following ns-3 default parameters to configure the MAC and physical layer in WiMAX.

Table 1. System level simulation/emulation parameters.

Parameter	Value
Spectrum	5GHz
Bandwidth	10MHz
Carriers N_{FFT}	256
Data carriers	192
Guard carries	1/4
Frame Duration	10ms
Tx power	30 dBm
Frame ration	1:1

B. *Experimental results & discussion*

This section is dedicated to evaluate the implementation of emulation system in the WiMAX network. The first set of experiments is devoted to validate the correctness of the designed system, and study whether emulation system may introduce performance degradation or not. The second set of experiments illustrates the impact of the emulation system and simulation on *round trip time* (RTT). The last set of experiments makes comparison between simple and RTPS scheduler with different modulation and coding schemes.

We first investigate the average throughput between two hosts. In this experiment the sender produces either UDP or TCP traffic. For the UDP stream, we run *UdpClientServer* application in centralized and distributed approaches to measure the throughput. However, the UDP stream for SITL and distributed SITL experiments is generated by *Iperf*. We set the datagram size to 1300 bytes in the application layer, since by default Ethernet card maximum MTU size is 1500 bytes. Accordingly, we set less than 1500 bytes to avoid any packet dropping. For TCP traffic, we use *OnOffApplication* in centralized and distributed approaches. *Iperf* is used for SITL and distributed SITL, the TCP payload is also set to 1300 bytes, and TCP window size is kept default. The measurements are done over a period of 300 seconds and average values are reported over that interval as computed from the ns-3 ASCII tracing files and for real-traffic by *Iperf*. For all kind of application traffic, the scheduler configures rtPS service flow and QAM16-1/2 modulation and coding schemes are used.

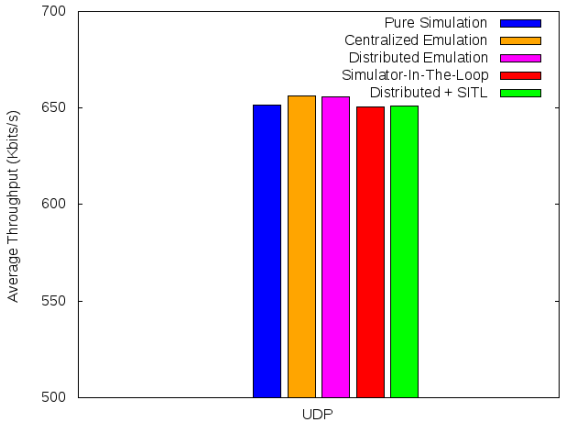

Fig.3. UDP performance comparison between simulation and emulation system.

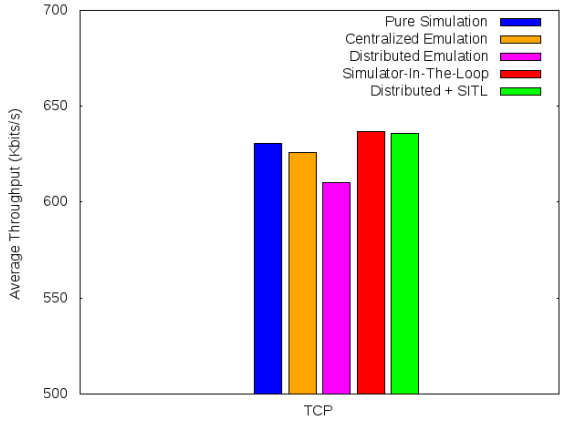

Fig.4. TCP performance comparison between simulation and emulation system.

Let us look at our first experiment, here we only discuss the results obtained with QAM16-1/2 modulation and coding rate, however those for other modulation schemes are similar. Fig.3 shows that generally there is a good agreement between simulation and emulation results in all experiment scenarios. For both simulated and real-time traffic our emulation system produces realistic throughput in the right magnitude. Most notably the average throughput obtained using centralized emulation is 1% higher than the one measured in simulation.

Next we present the results obtained for TCP in all experiments scenarios. Fig.4 confirms that there is also reasonable agreement between simulation and emulation world regarding TCP traffic. One thing to note is that the throughput obtained using simulation is 1% higher than the achieved by centralized emulation and it is 13% higher than distributed emulation. It is not a normal behavior of emulation system in distributed environment. We believe the underlying TCP implementation has influenced the distributed environment, since the results obtained for TCP using *Iperf* application in distributed emulation are 1% higher than simulation. We also noticed TCP throughput is lower than that for UDP in all scenarios, since the inelastic nature of UDP forces it to push more data on the channel.

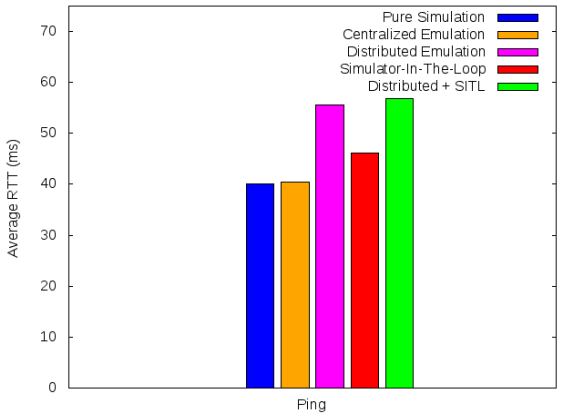

Fig.5. Average RTT results for simulation and emulation system.

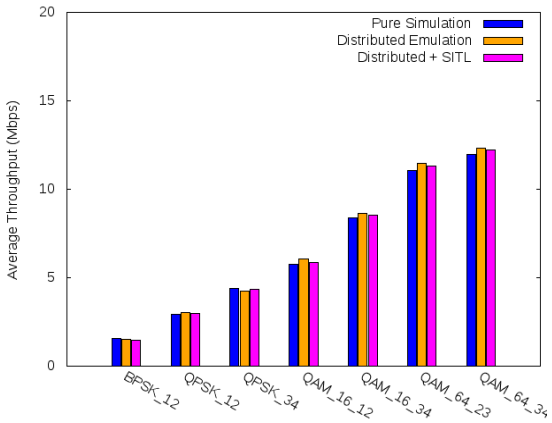

Fig.6. Average throughput results with different modulation coding schemes for simulation and emulation system.

Analogous to throughput measurements we now present second set of experiments for *round trip time* (RTT) between two hosts for simulation and emulation scenarios. For simulation and emulation experiments we use a ping application which is provided by ns-3 and works like the real one. For this purpose we fixed the bug and provided ICMP support in the WiMAX ns-3 packet classifier. In distributed SITL and SITL emulation we use Linux's default ping application to measure the delay.

As expected in Fig.5, the round trip time is same for simulation and centralized emulation, since they are on the same machine. However, RTT taken in SITL experiment on single machine is 13% more than corresponding experiment. We regard this difference as natural disparity caused by multithreading processes, since ping application is running on the underlying operating system. In this case, ns-3 delivers ICMP packets to RTCI then the packet is received by the underlying OS. Thus, we assume, it encompasses all latencies caused by packet processing inside the communication interface.

Most notably the round trip time taken using distributed emulation are almost 29% more than centralized scenarios. By far it is the largest fraction of the RTT. Expect the above mentioned multithreading cause, we believe there exist other sources of delay, may be this is imposed by running experiments asynchronously over testbed, since we are using simple shell script to run these experiments. Altogether, the latencies caused by this and the other communication actions produced by our emulation system amounts to more than a third of the above mentioned delay. This fraction is definitely not negligible for latency sensitive applications. We are still investigating the cause of this specific discrepancy. However, we have shown above that our emulation system constantly achieves lower RTT than those comparable to it. We conclude that the timing behavior of our emulation system is well suitable for the analysis of real-time applications. Especially, the measured RTTs are constantly equal to the measurements taken in simulation world.

In the next set of experiments we evaluate the impact of BS scheduler on throughput with respect to different MCS (modulation and Coding Schemes). The results explained above validate the operation of simple scheduler with rtPS service flow which is used by default in WiMAX ns-3 module. Due to its naive scheduling operations, the attained throughput is very lower compared to theory. Thus, RTPS scheduler is configured for rtPS service flow to quantify the performance of BS operations accurately, because according to the scheduler mechanism higher priority is given to rtPS service flows. For these experiments we use only UDP traffic with the same mentioned parameters.

As depicted in Fig.6, the observed throughput for all modulation schemes is almost the same in simulation and the distributed emulation. One thing to note is that, RTPS scheduler out-performs compared to simple scheduler. The simple scheduler even with higher modulation and coding schemes such as QAM64-3/4 barely achieves 1.5Mbps. Thus, RTPS scheduler with rtPS service flow has excellent performance. On average, the obtained throughput with RTPS scheduler for all modulation coding schemes has the same performance trends.

IV. RELATED WORK

One approach for WiMAX experiments is to make use of real wireless testbeds. This approach is employed, for instance, by WiMAX Extension to Isolated Research Data networks (WEIRD) [16]. WEIRD is the first testbed deployed over four countries in Europe used to demonstrate the new applications such that environmental monitoring, telemedicine and generic that can be used in WiMAX. GENI provides mobile WiMAX virtualized experimental testbed [17], to evaluate applications and protocols in realistic settings. This testbed gives local and remote researchers a robust set of scripting, experiment control, management, and measurement tools to run their experiments. These types of approaches suffer from complexities of experimental design and require deep knowledge to understand the shortcomings.

There are a few systems that use emulation in a similar manner with our emulation system to reproduce wireless network conditions on a wired network testbed. In this category NCTUns [18], and WEBS [19], offer the possibility to do so-called HITL (Hardware-In-The-Loo) simulations. These kinds of approaches simulate experiment environment in real-time with real network devices via interfaces that have the capability to inject packets from the simulation into the physical network nodes and vice versa. However, since the real devices are not integrated with the simulated environment, only their traffic is taken into account,

and not their overall influence. Additionally, these emulation-based testbeds fail to offer the possibility of creating realistic mobile scenarios in the way our system does. Our emulation approach does not have this drawback since it seamlessly integrates emulated system with the physical one.

V. CONCLUSION AND FUTURE WORK

In this paper a new WiMAX emulation system is presented that combines some of the advantages of real-world testbeds such as the use of real network applications and protocols, with those of simulation environments, such as good condition control, repeatability and propagation environment. In contrast to other approaches this enables seamlessly integrating emulation realm with physical world.

We have presented an in-depth discussion of our ns-3 based WiMAX emulation system and a series of experiments aimed at evaluating the proposed system in centralized and distributed settings as well as in scenarios involving SITL (for real traffic). The experimental results we have shown prove a reasonable agreement between the results obtained through emulation and the results obtained through simulation. Throughput differences ranged between 1 and 13% in our experiments, depending on experimental setup. We noticed some throughput and RTTs discrepancies. We shall investigate these discrepancies as our future work.

We also discussed the results of ns-3 WiMAX BS schedulers in context of different modulation coding rates. These experimental results also demonstrate the feasibility and the most important features of our proposed emulation system. From our experimental results we conclude that our system is well suited for testing protocols, and opens up the possibility to evaluate the performance of applications in real-time.

While in this paper we only focused on discussing the ns-3 based proposed emulation system and its implementation, one of the directions of our future work and we are currently working on it, is to integrate this emulation system with QOMET [20], to evaluate WiMAX MAC layer more thoroughly. The degree of achievable scalability of any emulation framework is very important. Therefore, another future direction refers to make it scalable to support large number of nodes in emulation space.

REFERENCES

[1] "IEEE standard for Local and Metropolitan Area Networks part 16: Air interface for fixed and mobile broadband wireless access systems amendment 2: Physical and medium access control layers for combined fixed and mobile operation in licensed bands and corrigendum 1," IEEE Std 802.16e-2005 and IEEE Std 802.16-2004, Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004), pp. 0–1–822, 2006.

[2] WEINGAERTNER, E., VOM LEHN, H., and WEHRLE ,K. "Device-driver enabled wireless network emulation". In Proc. SIMUTools 2011 (Barcelona, Spain, March 2011).

[3] University of Southern California, Information Sciences Institute. NS-2 network simulator. http://isi.edu/nsnam/ns/.

[4] Ns-3 website. http://www.nsnam.org/.

[5] Scalable Network Technologies, Inc. QualNet Developer. http://www.scalablenetworks.com/.

[6] OPNET website. http://www.opnet.com/.

[7] Chen, F., and Dressler, F. "A Simulation model of IEEE 802.15.4 in OMNeT++." In *6. GI/ITG KuVS* Fachgespräch Drahtlose Sensornetze, Poster Session (Aachen, Germany, July 2007), pp. 35--38.

[8] Elias Weingärtner , Florian Schmidt , Hendrik Vom Lehn , Tobias Heer , Klaus Wehrle. "SliceTime: a platform for scalable and accurate network emulation." In proceedings of the 8th USENIX conference on Networked systems design and implementation, March 30-April 01, 2011, Boston, MA.

[9] K. Fall. "Network emulation in the vint/NS simulator." In Proc. of the 4th IEEE Symp. on Computers and Communications (ISCC), pages 244–250, Sharm El Sheik, Red Sea, Egypt, July 1999.

[10] Razvan Beuran "Introduction to Network Emulation" Pan Stanford publishing, 2012.

[11] Alberto Alvarez , Rafael Orea , Sergio Cabrero , Xabiel G. Pañeda , Roberto García , David Melendi, "Limitations of network emulation with single-machine and distributed ns-3." Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, March 15-19, 2010, Torremolinos, Malaga, Spain.

[12] https://www.hostedredmine.com/projects/contiki-ns3 integration/wiki/Preliminary_Design_Notes

[13] Jahanzeb Farooq , Thierry Turletti, "An IEEE 802.16 WiMAX module for the NS-3 simulator" Proceedings of the 2nd International Conference on Simulation Tools and Techniques, March 02-06, 2009, Rome, Italy.

[14] M. A. Ismail , G. Piro , L. A. Grieco , T. Turletti, "An improved IEEE 802.16 WiMAX module for the ns-3 simulator", Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, March 15-19, 2010, Torremolinos, Malaga, Spain.

[15] Toshiyuki Miyachi , Ken-ichi Chinen , Yoichi Shinoda, "StarBED and SpringOS: large-scale general purpose network testbed and supporting software", Proceedings of the 1st international conference on Performance evaluation methodolgies and tools, October 11-13, 2006, Pisa, Italy.

[16] S. Mignanti, G. Tamea, I. Marchetti, M. Castellano, A. Cimmino, F. Andreotti, M. Spada, P. M. Neves, G. Landi, P. Simoes, and K. Pentikousis, "WEIRD testbeds with fixed and mobile WiMAX technology for user applications, telemedicine and monitoring of impervious areas," in Proc. Fourth International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM), Innsbruck, Austria, March 2008.

[17] Global Environment for Network Innovation. http://www.geni.net/.

[18] Shie-Yuan Wang; Lin, C.C.; Koo, P.H.; Haung, Y.M.; Jen-Shun Yang;, "NCTUns emulation testbed for evaluating real-life applications over WiMAX networks," Proc. IEEE 21st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), pp. 2030-2034, Sept. 2010.

[19] R.K.K, M. Raj, K. Balakrishnan and D. Das, "WEBS: WiMAX Emulation Testbed to Benchmark Streaming Multimedia QoS," IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA), 2009, pp. 1-6.

[20] R. Beuran, J. Nakata, T. Okada, L. T. Nguyen, Y. Tan, and Y. Shinoda. "A multi-purpose wireless network emulator: QOMET", In Proc. of IEEE International Conference on Advanced Information Networking and Applications (AINA 2008) Workshops, FINA 2008 symposium, pages 223–228, Okinawa, Japan, March 2008.